To print photographs or files of computational results (figures and/or data) obtained using Molecular Simulations software, acknowledge the source in the format:

Computational results obtained using software programs from Molecular Simulations Inc.—dynamics calculations were done with the Discover® program, using the CFF91 forcefield, ab initio calculations were done with the DMol program, and graphical displays were printed out from the Cerius² molecular modeling system.

To reference a Molecular Simulations publication in another publication, no author should be specified and Molecular Simulations Inc. should be considered the publisher. For example:

Contents

1. Introduction
   What is Homology? ........................................ 11
   Hardware and Installation .............................. 12
   How to Invoke Homology ................................. 12
   Program Environment ................................. 13
   Homology and Insight II .............................. 13
   Saving Homology Information with Insight II ....... 13
   Command Logging and Restarting .................... 13
   Homology and Discover .............................. 14
   Operations ............................................ 14

2. Theory
   Background ............................................. 15
   Homology Model Building ............................ 16
   Searching Sequence Databases With the FASTA Program
     ktup Value ........................................ 17
     Scoring Local Regions ............................ 17
     Joining Regions .................................. 18
     Optimizing the Sequence Matching ............... 18
     Explicit Statistical Estimation .................. 18
   Determining Structurally Conserved Regions (SCRs) .19
     Manual Determination of Structurally Conserved
       Regions ........................................ 20
     Automatic Determination of Structurally Conserved
       Regions ........................................ 20
     Automatic Sequence Alignment Methods .......... 25
     Needleman and Wunsch Algorithm for Pairwise
       Alignment ........................................ 25
     MSI’s Pairwise Alignment Procedure ............. 33
     MSI’s Automatic Multiple Sequence Alignment ... 38
     align123 ......................................... 49
     Scoring Matrices .................................. 57
   Superimposing structures based on sequence alignment .59
     Multiple Structure Alignment .................... 61
     Simultaneous Superposition of Structures ........ 62

Homology
5. Methodology

Step 1: Determine Which Proteins Are Related to the Model Protein

Step 2: Determining Structurally Conserved Regions (SCRs)

Step 3: Sequence Alignment
Multiple Sequence Alignment
Specifying the Initial Search Zone
Specifying a Mandatory Sequence
Automatic Calculation of Pairwise Threshold
Statistical Significance and Alternate Sequence Coloring
Characteristics of m-boxes
Subsets
Automatic Superimposing of Structures
Adjusting the Sensitivity and Selectivity of the Search
Handling of Existing Boxes
Interrupting the Search
Excessive Calculation Time
Single Search Mode
Manual Mode

Step 4: Assigning Coordinates Within the SCRs
Step 5: Building Loop or Variable Regions (VRs)
Searching for and Displaying Loops
Generating and Displaying Loops
Building Coordinates for the VRs

Step 6: Conformational Search for Side Chains Using Rotamers
Running Discover/CHARMm with Homology-Built Model Structures
End Repair
Splice Repair
Energy Minimization
Molecular Dynamics

Step 7: Refining the Structure with Discover/CHARMm
Structure Checking
Residue Dihedral Angles
Secondary Structure Classification
Algorithmic Implementation
United Atom Models
Setting Atomic Radii

Definition of Computed Surface Areas and their Significance
Total Surface Area
Relative Surface Area and the Tripeptide Model
Polar and Apolar Surface Area
Limitations in Implementation
Conclusion

6. Tutorial

Introduction
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardcopy and Pilot online tutorials</td>
<td>188</td>
</tr>
<tr>
<td>Hardcopy lessons</td>
<td>189</td>
</tr>
<tr>
<td>Lesson 4a: Finding structurally conserved regions</td>
<td>189</td>
</tr>
<tr>
<td>Lesson 4b: Building SCR and loops</td>
<td>210</td>
</tr>
<tr>
<td>Lesson 10: Finding alternative multiple sequence alignments</td>
<td>233</td>
</tr>
<tr>
<td>A. References</td>
<td>247</td>
</tr>
<tr>
<td>B. File Formats</td>
<td>255</td>
</tr>
<tr>
<td>Introduction</td>
<td>255</td>
</tr>
<tr>
<td>Amino Acid Scoring Matrices</td>
<td>255</td>
</tr>
<tr>
<td>User Scoring Matrix Files</td>
<td>256</td>
</tr>
<tr>
<td>Sequence Alignment Command</td>
<td>256</td>
</tr>
<tr>
<td>Input Databases Command</td>
<td>256</td>
</tr>
<tr>
<td>Get Sequence, Alignment and Databases commands</td>
<td>258</td>
</tr>
<tr>
<td>Secondary structure file</td>
<td>260</td>
</tr>
<tr>
<td>C. Glossary</td>
<td>263</td>
</tr>
<tr>
<td>D. seq_extract Utility</td>
<td>269</td>
</tr>
<tr>
<td>Output</td>
<td>269</td>
</tr>
<tr>
<td>Results Displayed on the Screen</td>
<td>269</td>
</tr>
<tr>
<td>Output Files</td>
<td>270</td>
</tr>
<tr>
<td>Execution Options</td>
<td>271</td>
</tr>
<tr>
<td>E. Matrices</td>
<td>275</td>
</tr>
<tr>
<td>Sequence Alignment Matrices</td>
<td>275</td>
</tr>
<tr>
<td>Identity Matrix</td>
<td>275</td>
</tr>
<tr>
<td>Codon Substitution Matrix</td>
<td>276</td>
</tr>
<tr>
<td>Dayhoff Evolutionary Mutation Matrix</td>
<td>276</td>
</tr>
<tr>
<td>Hydrophobicity Matrix</td>
<td>277</td>
</tr>
<tr>
<td>Input Databases Command Matrices</td>
<td>277</td>
</tr>
<tr>
<td>Identity Matrix</td>
<td>277</td>
</tr>
<tr>
<td>Codon Substitution Matrix</td>
<td>278</td>
</tr>
<tr>
<td>Dayhoff Evolutionary Mutation Matrix</td>
<td>279</td>
</tr>
<tr>
<td>Hydrophobicity Matrix</td>
<td>279</td>
</tr>
<tr>
<td>F. Hydrophobicity Scale Values</td>
<td>281</td>
</tr>
<tr>
<td>Amino Acid Values</td>
<td>281</td>
</tr>
<tr>
<td>Threshold Values</td>
<td>282</td>
</tr>
</tbody>
</table>
G. Sequence Databases

Protein Sequence Databases ........................................ 283
   NBRF ........................................ 283
   Swiss-Prot ........................................ 284

DNA Sequence Databases ........................................ 284
   GenBank ........................................ 284
   EMBL Data Library ........................................ 285

H. align123 Standalone

HELP 1: General help for align123 ......................... 287
HELP 2: Help for multiple alignments .................... 288
HELP 3: Help for pairwise alignment parameters ....... 289
HELP 4: Help for multiple alignment parameters ...... 291
HELP A: Help for protein gap parameters. ............. 292
HELP 5: Help for output format options ................ 293
HELP 6: Help for profile and structure alignments .... 294
HELP B: Help for secondary structure / gap penalty masks . 295
HELP C: Help for secondary structure / gap penalty mask output options ............................. 297
HELP 7: Help for phylogenetic trees .................... 297
HELP 8: Help for choosing a weight matrix ............ 298
HELP 9: Help for command line parameters DATA (sequences) ........................................ 300
HELP 10: Help for tree output format options ........ 303
What is Homology?

Homology is a computer program package that helps you build the structure of a protein when you know only its amino acid sequence and the complete atomic structure of at least one other reference protein.

The reference (or known) protein can be imported in any format by MSI’s interactive molecular graphics program, Insight II. The reference protein must be structurally homologous to the model protein you are trying to build (also called the unknown or sequence protein). Structural segments are taken directly from the reference protein in regions which are thought to be conserved within the family of homologous proteins.

Homology is especially suited to situations where more than one reference structure is available. In these cases, sequence similarities alone need not be used to determine what portions of the proteins are conserved. Instead, the 3D structures themselves can be used to give a reliable prediction of the areas expected to have similar conformations. Wherever the reference proteins are found to have conserved conformations, the unknown protein is assumed to have the same conformation as well.

Homology is a collection of tools that leads you through the process of building a protein by homology. Homology is not designed to be a “black box” program for assembling purely hypothetical structures. Instead, it provides you with the ability to build a model that is chemically and physically reasonable.

With Homology you can:

1. Search protein databases for proteins similar to the model protein you are building.
1. Introduction

2. Display and align amino acid sequences.

3. Find \textit{structurally conserved regions} (SCRs) in reference proteins.

4. Propose structures for the loops or \textit{variable regions} (VRs) between the SCRs.

5. Copy coordinates from the reference proteins to the model.

6. Refine the newly built structure using the molecular mechanics and dynamics facilities available in MSI’s molecular simulation program, Discover.

\section*{Hardware and Installation}

For information regarding hardware, operating system level, and installation, please refer to the MSI Installation Guide. Note that refinement of Homology-built structures requires that \textit{Discover} also be installed at your site. The Discover refinement (via the Homology \textbf{Refine} pulldown) is active with either a Discover or CHARMm license.

\section*{How to Invoke Homology}

Homology is a module within \textit{Insight II}. You invoke it by selecting \textbf{Homology} from the \textbf{Module} pulldown on the top menu bar. When the module is activated, a series of Homology-specific pull-downs appear on the lower menu bar. Commands in these pull-downs can be used whenever the pull-downs are displayed; core commands in the pull-downs on the top menu bar can be used with all modules. Of course, you can use other modules without losing any information or disrupting the Homology session.
Program Environment

Homology and Insight II

As mentioned before, Homology is an application within Insight II. Most Insight II commands can be used on molecules created with Homology and vice versa. All rules and conventions for Insight II apply when you are working with Homology. For example, commands in Homology can be typed in or selected from the pulldowns and parameter blocks, just like commands in Insight II.

Note: For more information on the basic operations, procedures, and functionality of Insight II, please refer to the Insight II User Guide. For detailed information on how to use commands in Insight II, see the Insight II Reference Guide.

Saving Homology Information with Insight II

You can save a Homology modeling session at any time using the Insight II Save_Folder command in the File pulldown in the top menu bar. You can retrieve this information for subsequent use by selecting the Restore_Folder command.

The information in the sequence window is highly interdependent; therefore, objects with associated sequences can only be saved and restored on an all-or-none basis. Folders containing Homology information cannot be edited with the Remove_Folder command.

Command Logging and Restarting

During each Homology session, a record is kept of the commands you execute. This log is written into the file named WBLOGFILE in your current working directory. Upon exiting Insight II, this file is renamed to insight.log. All commands, whether typed or selected from the pulldowns, are logged. If a session needs to be reconstructed because of a system or program failure, the Source_File command in the File pulldown can be used to reissue the commands in the same order they were originally entered.
1. Introduction

All mouse movement operations that alter the relative positions of the amino acid sequences in the sequence window are also logged. Therefore, using the `Source_File` command will also re-establish a sequence alignment.

**Homology and Discover**

The models built with Homology can be refined using Discover/CHARMm in two ways.

One way is to open a Discover/CHARMm session by selecting `Discover/CHARMm` from the `Module` pulldown while you are in Insight II. The Discover/CHARMm module enables you to set up interactive and background jobs with or without constraints and external forces.

The second way to communicate with Discover is by using the commands `SpliceRepair`, `Relax`, and `Explore`, which are found in the `Refine` pulldown of the Homology module. By using these commands, strategies specific to refining the structure of a protein model built by Homology are conveniently predefined. The CHARMm simulation engine is not available for this command, however, if you have the CHARMm but not the Discover license, Discover will still be activated by this command.

**Note:** For more information on the basic operations, procedures, and functionality of Discover/CHARMm, please refer to the Discover/CHARMm manual, as well as information on the Discover/CHARMm module in the Insight II User Guide.

**Operations**

In addition to the basic operations that you use with Insight II, the use of Homology requires an understanding of the operations of the sequence window.
Determining protein and nucleic acid sequences is now routine in molecular biology laboratories. As a result, the rate at which sequence information is published has increased dramatically. Several organizations have compiled databases to centralize the published information and to facilitate its use in research (EMBL database, PIR/NBRF database, GenBank database). Structural information from x-ray crystallographic or NMR results, on the other hand, is obtained much more slowly. For instance, if the protein is not a simple mutation of one whose structure is already known, it can take from one to five years to perform a complete structural determination. Because of the disparate rates of sequence and structure determination, there are many proteins for which sequence information is known but the three-dimensional structure is not.

It is advantageous to develop a method whereby the conformation of a newly characterized protein can be predicted from its amino acid sequence. Since a priori folding rules for proteins have not yet been developed, any structural prediction must be based on the conformations of reference proteins. This assumes that the structures of the unknown and reference proteins are homologous and that a model of the unknown protein can be built from the reference proteins.

Early work dealing with building a protein by homology used only one known structure (Browne et al. 1969, Shotton et al. 1970). Amino acid similarities between the known and unknown proteins were used to determine where one protein would resemble the other. The sequences were aligned and then the coordinates of the reference protein were used to predict those of the unknown protein.
2. Theory

More structural approaches have been suggested by Greer and by Blundell (Greer 1980, 1981, 1985; Blundell 1987, 1988). In their methods, more than one reference protein is used, and a greater emphasis is placed on the conformational similarities between the proteins. Less emphasis is placed on sequence alignment alone as a basis for the model. By determining which portions of the molecules do not vary from one member of a protein family to another, there is greater confidence that extrapolation to a new member will be accurate.

**Homology Model Building**

The process of building a model for a protein using Homology is divided into the following steps:

7. Determine which proteins are related to the model protein.
8. Determine structurally conserved regions (SCRs).
9. Align the amino acid sequence of the unknown protein with those of the reference protein(s) within the SCRs.
10. Assign coordinates in the conserved regions.
11. Predict conformations for the rest of the peptide chain, including loops between the SCRs and possibly the N- and C-termini.
12. Search for the optimum side chain conformations for residues that differ from those in the reference proteins.
13. Use energy minimization and molecular dynamics to refine the molecular structure so that steric strain introduced during the model-building process can be relieved.

**Searching Sequence Databases With the FASTA Program**

In many homology modeling projects, one knows ahead of time the type of model to be built. That is, the sequence used as the basis of the model is frequently one from a well-known protein family for which related proteins are known. Under these circumstances, the choice of reference proteins is trivial, and occasionally, the overall sequence alignment for the family is also obvious.
Homology Model Building

It may be the case, however, that the set of related proteins is not known. Then one must find examples of similar sequences in a database. These can be any of the sequence databases available (see Appendix G, Sequence Databases) or one made from the proteins in the Brookhaven Protein Databank. The FASTA program (Lipman and Pearson 1985; Pearson and Lipman 1988, Pearson 1990) can search any of these files to find a number of related sequences and display a pairwise alignment between the given new sequence and each found sequence. In this way, the protein family can be elucidated, and a set of potential reference proteins can be identified.

**ktup Value**

The FASTA program compares the given sequence to each of the sequences in the database in turn. The search efficiency is enhanced by avoiding the need to compare all combinations of residues explicitly. Instead, a *ktup* value is defined as a specific residue string. For proteins, for instance, the allowed *ktup* values are 1 (a single residue) or 2. A *ktup* of 2 means that each pair of residues along the sequence can be one of $20 \times 20$, or 400 pair types, and that there need be only $1/2 \times 1/2 = 1/4$, the number of comparisons made. For nucleic acids, *ktup* can range from 1 to 6.

**Scoring Local Regions**

The first step in the search process is to find segments of matching residues that have no substitutions and no gaps. For each pairwise comparison, each sequence is parsed into *ktups*. Then, the position of each *ktup* along the chain is found. When the sequences are compared, the relative positions of matching *ktups* (the offsets) are calculated. Finally, for each possible offset, the number of matches is counted. For example, if $ktup = 1$ and Sequence #1 is WSQTRK and Sequence #2 is SQATRK, then the matching residues are S, Q, T, R, and K. S is found at position 2 in Sequence #1 and position 1 in Sequence #2. The offset is $(1 – 2) = –1$. The same is true for Q, so that offset ($–1$) has a count of 2 instances. T is at position 4 in both sequences, giving an offset of 0. R and K also have offsets of 0, so that offset (0) has a count of 3. TRK and SQ are now two high-scoring local regions.

The best-scoring local regions are then rescored allowing substitutions, but not insertions or deletions. This is done by using a scor-
2. Theory

An average score for the region is calculated by averaging the pairwise residue values. After this step, the highest scoring stretches are called initial regions, each with an init1 score.

Joining Regions

Initial regions can be joined together to make longer stretches of matched residues. A penalty is imposed to join two segments into one, similar to the gap penalty used for the automatic sequence alignment method (see MSI's Pairwise Alignment Procedure on page 2-33). Only regions with scores above a chosen threshold are joined. Composite scores (initn) are calculated and used to rank the matches.

Optimizing the Sequence Matching

Finally, you can request that the sequence matching be optimized using a version of the Needleman and Wunsch algorithm (see page 2-25). Ranking by this optimized score can sometimes change the order found for best matches.

Explicit Statistical Estimation

Version 3.1 of FASTA provides several major improvements over previous versions, the most important of which is the incorporation of explicit statistical estimates and appropriate normalization of similarity scores.

The quality of the fit of the extreme value distribution to the actual distribution of similarity scores is summarized with the Kolmogorov-Smirnov statistic. In general, statistical values less than 0.10 (N=30) indicate excellent agreement between the actual and theoretical distributions.

With explicit expectation calculations, the program now shows all scores and alignments with expectations less than 10.0 (with optimized scores, 2.0 without optimization).

Seqfold and Profiles_3D can be used to identify templates when FASTA fails. They use secondary structure enhanced sequence searching methods or 3D structure information in homology identification and are more sensitive in identifying remote homology relationships.
Determining Structurally Conserved Regions (SCRs)

Building a model of a protein by homology is based on the fact that there are regions in all proteins of a particular family that are nearly identical in structure. These regions tend to be at the inner cores of the proteins where differences in peptide chain topology would have profound effects on the overall conformation of the molecule. Thus, the secondary structural units of the proteins in a family—the alpha helices and beta sheets—lie in the same orientations relative to one another. The loop regions, which are at the surfaces of the molecules, may differ.

While amino acid sequences of the inner core peptide segments are generally conserved, there may be some conservative discrepancies. Amino acids with similar chemical or physical properties are often substituted, for example, ILE → VAL. From time to time, amino acids with quite different properties may be substituted, for example, PHE → LYS. As long as the conformation of the backbone atoms of the peptide chain is not dramatically altered, such a non-conservative substitution is of little consequence. The overall shape of the protein will not differ much from others in the same family.

Therefore, it is more realistic to examine all known structures to see where the proteins are structurally conserved rather than conserved merely in sequence. One way to do this is to compare the proteins by calculating the RMS difference of the backbone atoms of the peptide segments thought to be structurally conserved. A low RMS value implies that the segments would serve as good templates for building regardless of the types or conformations of their side chains.

In practice, this means that two proteins are compared segment by segment. A moving window approach is used to find a self-consistent set of segments with low RMS differences in the coordinates of their backbone atoms. At the same time, the orientations of the segments in relation to the rest of the protein must be similar. Finally, no SCR can span more than one secondary structural unit in order to allow for insertions and deletions of residues, which are normally seen at turns and loops.
2. Theory

**Manual Determination of Structurally Conserved Regions**

Structurally conserved regions can be found by comparing short segments from each of two proteins. The RMS difference of the coordinates of the segments’ backbone atoms is calculated iteratively as the range of residues being compared is adjusted in both extent and position. At the same time, superposition is done, treating the protein molecules as rigid bodies. Regions that have a low local RMS value are chosen for the pair of segments. In addition, the relative orientations of the proteins as determined by the superposition must be similar. This is judged visually on the screen. Finally, in order to allow for insertions and deletions of residues that are normally seen in turns and loops, no SCR can span more than one secondary structural unit.

**Automatic Determination of Structurally Conserved Regions**

In order to automatically compare two protein structures, a representation that is independent of any coordinate frame must be found. In general, superposition of the structures is not possible because it is not known in advance which atoms in the two molecules are corresponding. This is especially true when the sizes of the two proteins are different. An interatomic distance matrix constructed from the protein’s C\textsubscript{\alpha} coordinates contains all the needed information. With the exception of chirality, it is a complete representation of the conformation of the protein; however, for molecules as large as proteins, mirror images do not exist.

For both proteins that are being compared, a C\textsubscript{\alpha} distance matrix is constructed.
Since a structurally conserved region is defined as a short contiguous peptide segment, small portions of the distance matrices of the two proteins are compared at a time. For each comparison, a \textit{probe} adjacent to the diagonal of the distance matrix is taken from one of the proteins. Since the distance matrix is symmetric about the diagonal, only the triangular region on one side is used (see Figure 1). The size of the probe is user-specified.

To compare peptide segments, the probe from one protein is overlaid on the matrix of the other, and an RMS difference of the matrix elements is calculated. Many such calculations are done as the probe is passed along the length of the diagonal of the second matrix. When all desired comparisons have been made, a new probe is taken that is one residue farther downstream, and the process is repeated (see Figure 2). For each probe, the minimum RMS value and its corresponding residue matching is saved as long as the RMS score is below a user-specified threshold. In practice, each probe is not compared to every possible segment in the second protein. It is assumed that within a margin of error, residues approximately the same distance along the chains of the two proteins should correspond if the proteins are approximately the
2. Theory

same length. The number of residues upstream and downstream from the nominal alignment that are examined are also user-specified.

![Diagram of protein alignment](image)

**Figure 2. Assessing Peptide Segment Similarities**

A probe from the distance matrix of Protein 1 is overlaid onto that of Protein 2, and the RMS difference of the corresponding matrix elements is calculated. The probe is then moved down the diagonal of the second matrix, and the calculation is repeated. A minimum value, which represents the best match of the peptide segment in Protein 1 to a segment in Protein 2, is saved if the RMS difference is below a specified threshold. The entire process is then repeated with a new probe from Protein 1.

It may not be appropriate to keep the peptide segments found in the first step as the final set of SCRs. Because of the nature of the algorithm, a single probe could have been matched to more than one segment in the second protein, or vice versa, implying an inconsistent alignment. Such a situation can occur, for example, when a portion of an $\alpha$-helix of one protein is matched to successive turns of a helix in the other protein. Improper matches can also be made between segments that have the same local conformation, but are not oriented the same way in both proteins. In this case, it would not be possible to determine which orientation was correct, so neither segment can be used for model building. Note that the number of inconsistently matched segments increases as the probe size is decreased or the RMS threshold is increased.

Therefore, the candidate segments must be checked to make sure that they are self-consistent. They are examined pairwise to see if
both segments from one protein have the same relative orientation as the two corresponding regions from the other protein. This is done by comparing the off-diagonal blocks of matrix elements from the distance matrices of the two proteins. If the RMS difference of the off-diagonal blocks (see Figure 3) is below a second user-specified threshold, then the two pairs are considered to have similar orientations.

![Figure 3. Checking the Relative Orientations of Pairs of Segments](image)

To build a reliable model, the relative orientations of all the SCRs must be similar. Information about the orientation of two segments is contained in the off-diagonal block of matrix elements. If the RMS difference of the matrix elements in the off-diagonal blocks of two proteins is small, then the two segments are oriented similarly in both proteins. Thus, if segment A corresponds to segment A', and segment B corresponds to segment B', then if the RMS difference of AB and A'B' is below a specified threshold, then A and B have the same relative orientation as A' and B'.

The final set of segments are those that are consistent with the majority of others that were found. A matrix is formed of all the putative matches. An RMS difference comparison of the off-diagonal matrix elements of every possible combination of segments is made (see Figure 4). If the RMS score between a segment and more than half of the other segments is below the threshold, then it is retained. This restriction is necessary to eliminate sets of segments that are consistent with each other, but are not consistent with most of the remainder of the protein.
2. Theory

Many of the final segments are overlapping. The final set of SCRs is found by scanning the lengths of the sequences and combining overlapping regions, stopping when one or more residues are not in any matched segment. The sequences are then aligned to reflect the correspondence of the residues in the two proteins. The positions of the SCRs are indicated by boxes that appear around the matched segments in the sequence window.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>—</td>
<td>—</td>
<td>below</td>
<td>below</td>
<td>above</td>
<td>above</td>
</tr>
<tr>
<td>2</td>
<td>below</td>
<td>—</td>
<td>below</td>
<td>below</td>
<td>above</td>
<td>above</td>
</tr>
<tr>
<td>3</td>
<td>below</td>
<td>below</td>
<td>—</td>
<td>below</td>
<td>above</td>
<td>above</td>
</tr>
<tr>
<td>4</td>
<td>below</td>
<td>below</td>
<td>below</td>
<td>—</td>
<td>below</td>
<td>above</td>
</tr>
<tr>
<td>5</td>
<td>above</td>
<td>above</td>
<td>above</td>
<td>below</td>
<td>—</td>
<td>below</td>
</tr>
<tr>
<td>6</td>
<td>above</td>
<td>above</td>
<td>above</td>
<td>above</td>
<td>below</td>
<td>—</td>
</tr>
</tbody>
</table>

**Figure 4. Finding a Self-Consistent Set of Peptide Segments**

A segment can only be used for model building when it has similar relative orientations with the majority of the other segments in the protein. This is ensured by forming a matrix of all possible pairs of segments and calculating the RMS differences of the off-diagonal matrix elements for every combination. A segment is kept as an SCR only if the RMS differences between it and the majority of other segments in the protein are below a specified threshold. In the example presented in the figure, segments 5 and 6 are consistent with each other, but not with the rest of the protein. Therefore, only segments 1-4 are retained.

It should be noted that the current algorithm has no knowledge of secondary structure. Ideally, a structurally conserved region should be terminated at the end of a secondary structural unit, so that, for example, a β-sheet would be comprised of a separate SCR for each strand.

The algorithm just described finds SCRs in a pair of protein structures and aligns their sequences to bring the corresponding structurally conserved segments into register. This method can be
extended to the simultaneous alignment of more than two sequences based on the similarity of their corresponding three-dimensional structures. The method has much in common with the divide-and-conquer multiple sequence alignment algorithm (see MSI’s Automatic Multiple Sequence Alignment on page 2-38). A detailed explanation is therefore presented following the description of multiple sequence alignment (see Multiple Structure Alignment on page 2-61).

**Automatic Sequence Alignment Methods**

Sequence alignment is a central technique in homology modeling. It is used in two distinct phases of a project. First, it is used in determining which areas of the reference proteins are conserved in sequence and thus suggests where the reference proteins may also be structurally conserved. Second, sequence alignment is used to establish a one-to-one correspondence between the amino acids of the reference proteins and those of the unknown protein in the structurally conserved regions. This correspondence is the basis of the transferring of coordinates from the reference to the model protein. Homology provides two automatic pairwise alignment procedures, one for each of these applications. In addition, Homology provides an automatic multiple alignment procedure that can simultaneously align as many as ten amino acid sequences.

**Needleman and Wunsch Algorithm for Pairwise Alignment**

When it is uncertain where structurally conserved regions (SCRs) are to be found in the reference proteins, it is often helpful to find regions of sequence conservation first. The Needleman and Wunsch algorithm (1970) is capable of aligning two sequences in such a way as to identify the regions of correspondence. It assigns scores for every pairwise comparison between corresponding amino acids in the sequences. A higher score is given to good matches. At the same time, gaps are introduced into the sequences to allow for the simultaneous alignment of several regions when they are not of the same length. Gaps are given negative scores. The overall balance between the number of good amino acid matches and the least number of required gaps leads to an optimum alignment.
2. Theory

The first step in the procedure is to set up a *comparison matrix* between the two sequences in question. This matrix has dimensions equal to the lengths of the two sequences. Each matrix element is taken directly from one of several scoring matrices, which in turn represents the likelihood of one amino acid being replaced by the other in the sequence. Figure 5 shows one of the scoring matrices, the identity matrix, which gives a score of 1 for identical matches and 0 for all nonidentical pairs.

A comparison matrix for the two sequences:

```
A C F G S T V I Q N
```

and

```
C F G H A S T V Q N
```

is shown in Figure 6. There are only 1s and 0s in the matrix, reflecting the scores taken from the identity matrix. Thus, there are 1s in the intersections of C and C, F and F, and G and G, but 0s in the intersections of A and C and A and F. Note that there are obvious patterns in the matches between residues, but gaps need to be introduced to maximize the correspondence.

The second step in the alignment procedure is to *process the comparison matrix*. For each matrix element, the adjacent partial row and column are examined (see Figure 7). The maximum value found there is added to the value of the current matrix element. In the example, the current matrix element is the intersection between the S row and the S column. Note that the processing of the comparison matrix has already been done from the bottom right corner up to this element.
### Figure 5. Amino Acid Identity Matrix

<table>
<thead>
<tr>
<th></th>
<th>ALA</th>
<th>ARG</th>
<th>ASN</th>
<th>ASP</th>
<th>CYS</th>
<th>GLN</th>
<th>GLU</th>
<th>HIS</th>
<th>ILE</th>
<th>LEU</th>
<th>LYS</th>
<th>MET</th>
<th>PHE</th>
<th>PRO</th>
<th>SER</th>
<th>THR</th>
<th>TRP</th>
<th>TYR</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALA</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ARG</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ASN</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ASP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CYS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GLU</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GLY</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HIS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ILE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LEU</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LYS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MET</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PHE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PRO</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SER</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>THR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TRP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TYR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>VAL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Alphabets (Amino Acids):**

- ALA: Alanine
- ARG: Arginine
- ASN: Asparagine
- ASP: Aspartic Acid
- CYS: Cysteine
- GLN: Glutamine
- GLU: Glutamic Acid
- HIS: Histidine
- ILE: Isoleucine
- LEU: Leucine
- LYS: Lysine
- MET: Methionine
- PHE: Phenylalanine
- PRO: Proline
- SER: Serine
- THR: Threonine
- TRP: Tryptophan
- TYR: Tyrosine
- VAL: Valine
2. Theory

Note that all the values in the comparison matrix have already been changed from those in Figure 6. The highlighted partial row and column in Figure 7 shows that the maximum value is 4 at the intersection of the T row and T column. There is already a 1 in the current matrix element, and this is added to the 4 to produce a final value of 5. Note that this is, in fact, the value shown in the processed comparison matrix of Figure 8.

The third and final step in the alignment procedure is to find the maximum pathway through the processed comparison matrix. The search, beginning at the upper left corner of the matrix and proceeding to the lower right, is in the opposite direction of the second step. The maximum score in the top row or the leftmost column is taken as the initial match. In this case, this is the 8 at the intersection of C and C. The adjacent partial row and column are examined for the next larger score (see Figure 9). Here, the result is the 7 at the intersection of F and F.

If the maximum score is not found in the diagonally adjacent square, then a gap insertion is implied. Whether the gap is in the

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>F</th>
<th>G</th>
<th>S</th>
<th>T</th>
<th>V</th>
<th>I</th>
<th>Q</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 6. Comparison Matrix Between the Two Sequences

The values for each matrix element are taken directly from the corresponding entry of the identity matrix.
first or second sequence depends on whether the maximum score is found along the partial row or partial column. An example of this is in the maximum path diagram shown in Figure 10(a). Moving from the (G,G) square (score of 6) to the (S,S) square (score of 5) requires the skipping of two rows of the matrix. This corresponds to a gap region of two residues in the first sequence in the final alignment (Figure 10(b)).

![Figure 7. Processing the Comparison Matrix](image)

Processing begins at the lower right corner of the matrix and proceeds up and to the left. The maximum value of the previous row and column is found for each matrix element. The new value of the current element is the sum of its original value and the maximum.
2. Theory

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>F</th>
<th>G</th>
<th>S</th>
<th>T</th>
<th>V</th>
<th>I</th>
<th>Q</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>7</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>V</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Q</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 8. Processed Comparison Matrix**

Each matrix element now includes information about the number of amino acid matches farther down the sequences.
Figure 9. Final Maximum Pathway and Corresponding Sequence Alignment

The search begins at the maximum score in the first row or column. Then, for each subsequent step, the maximum score in the partial row and column starting diagonally adjacent (highlighted) is found.
2. Theory

In most cases, a penalty is used to minimize the number of gap regions. When a gap penalty is used, its value is subtracted from all the matrix elements of the partial row and column, except for the diagonally adjacent one, both in the processing of the comparison matrix and as the maximum path is being traced. This has the effect of making the diagonally adjacent move through the matrix, representing no gap insertion, seem more favorable. This constant gap penalty parameter applies equally to all gaps, regardless of their lengths, and is essentially a penalty for the creation of a gap. Better results are obtained with a gap penalty function that also

![Figure 10. Final Maximum Pathway and Corresponding Sequence Alignment](image)

Each step through the matrix in (a) implies an amino acid match in (b). When two steps are diagonally adjacent, there is progression by one residue along both sequences (no gap). If one or more rows are skipped, there is a gap in the sequence represented along the top of the matrix. If one or more columns is skipped, there is a gap in the sequence represented along the side.
includes a second term proportional to the length of the gap. The proportionality constant is known as the gap length penalty. The effect of a nonzero gap length penalty is to favor shorter gaps over longer ones. Both the gap penalty and the gap length penalty are heuristic parameters, meaning that there is no sound theoretical basis for choosing their optimal values. The best values depend on the relatedness of the sequences, and on the scoring matrix used, and must be found by trial and error.

**MSI's Pairwise Alignment Procedure**

After structurally conserved regions (SCRs) are determined, the alignment procedure just described cannot be used. By definition, SCRs cannot contain insertions or deletions. Gap regions must therefore be disallowed within SCRs. The standard Needleman and Wunsch algorithm does not have provisions for treating SCRs in a special manner. It places a gap at any location to optimize amino acid matching.

The proprietary algorithm used by the Homology program is an enhancement of the Needleman and Wunsch procedure. This algorithm incorporates structural information into the sequence alignment. It sets up the same comparison matrix as before, but there is no processing step. Instead, the search for the optimum path through the matrix is done immediately. Each region corresponding to an SCR is treated independently. No gaps are allowed within any conserved region.

An example is shown in Figure 11. A third sequence is added to the alignment done previously. It is shown in lowercase letters to indicate that it represents a protein for which no coordinates are known. The SCRs of the other two sequences are shown in boxes.
2. Theory

A comparison matrix for the new sequence and the first sequence of the aligned pair is shown in Figure 12. The unknown protein’s sequence is always listed along the left side, and the reference protein’s sequence is listed across the top. Notice that a complete matrix has not been made. Instead, the only columns constructed are those that correspond to the amino acids of the reference sequence that are in SCRs. All other amino acids are ignored. Also, notice that heavy vertical lines have been drawn between the SCRs. The lines indicate the only places where gaps are permitted to be inserted, between the SCRs.

The traversal through the matrix is also much different from the standard method. Since no gaps are made within the SCRs, the only valid path through a set of columns bounded by heavy vertical lines is a diagonal path. For each SCR, the score for every diagonal path through the appropriate columns is evaluated, and the maximum is chosen. No proposed path can start higher in the matrix than the end of the path found in the previous SCR because that would imply a backtracking of the alignment. In other words, no residue of the unknown sequence (along the left side of the matrix) can be matched to more than one residue of the sequence along the top.

Figure 11. Addition of a Third, Unknown Sequence to the Previous Alignment

A third, unknown sequence (lowercase letters) is added to the previous alignment. Boxes are drawn around the conserved regions of the aligned pair of sequences.
When the path through the matrix is mapped back to the final alignment, gaps can be placed in the proper locations. Since there are residues in the reference protein’s sequence (along the top of the matrix) that do not appear in the matrix, it is not obvious at first glance whether a vertical discontinuity in the path represents a gap. It must be determined whether or not the vertical jump is greater or less than the number of intervening residues. If the jump is greater, then the gap is put in the reference sequence; if the jump is less, then the gap is put in the unknown protein’s sequence. All gaps in the reference proteins are arbitrarily placed just after the ends of the SCRs.

In the example shown in Figure 13(a), the maximum diagonal path through the first SCR begins at (c,C) and ends at (g,G) for a score of 2. In the second SCR, the maximum path begins at (s,S) and ends at
2. Theory

at (v,V) for a score of 3. In the third SCR, it begins at (q,Q) and ends at (n,N) for a score of 2.

The pathway through the matrix is mapped to the final alignment shown in Figure 13(b). Gap regions were inserted between the SCRs based on the number of rows skipped when moving from path to path in the comparison matrix. Between the first and second SCRs, one row was skipped, but there is already a two-residue gap in the alignment shown in Figure 11. Therefore, the new gap was placed in the unknown (lowercase) sequence. Between the second and third SCRs, two rows were skipped, and there is only one residue between the SCRs in the original alignment. Therefore, the new gap must be in the reference sequence. Since the alignment between the two reference sequences could not be disrupted, the gap was put in both of the sequences.
Since gaps are only permitted between SCRs, this algorithm has the advantage of including structural information in the sequence alignment. Because the proper transfer of coordinates depends heavily on the alignment, a more realistic model should be the result.

---

**Figure 13. Mapping the Pathway Through the Matrix**

(a) The final path through the comparison matrix. Each SCR is examined independently. Only diagonal paths are allowed, reflecting the fact that gaps are forbidden within them. The maximum diagonal path is found for the first SCR. For each subsequent region, the correct maximum path must begin lower in the matrix than the end of the previous segment. (b) The final sequence alignment.
2. Theory

**MSI’s Automatic Multiple Sequence Alignment**

As explained earlier, sequence alignment is an essential part of homology modeling, both as a means for finding SCRs in the reference proteins and for aligning those SCRs to the sequence of the model protein. The automatic alignment methods described in the preceding sections operate only on pairs of sequences. Most modeling projects, however, require the alignment of more than two sequences, since it is desirable to use as many reference proteins as possible. Homology provides a powerful method for the automatic simultaneous alignment of more than two sequences.

The simultaneous alignment of more than two sequences, commonly called the *multiple alignment problem*, is significantly more difficult than is the alignment of a pair of sequences. This is because the number of possible alignments increases exponentially with the number of sequences to be aligned. Although there is no ideal method for solving the multiple alignment problem, many heuristic algorithms have been developed to provide approximate solutions. Most fall into one of three categories, the advantages and disadvantages of which are summarized below. The algorithm used by the Alignment/Multiple_Sequence command in Homology belongs to the third category.

**Globally Optimal Multiple Alignment**

This method extends the type of algorithm used in the pairwise Alignment/Pairwise_Sequence command (Needleman and Wunsch 1970) to a multidimensional space (one dimension for each sequence to be aligned). The advantages of this method are that it is fully automatic, and the result is guaranteed to be the optimal alignment with respect to a given similarity scoring matrix and gap penalty. Its most serious disadvantage is that memory usage and calculation time increase exponentially with the number of sequences. Other drawbacks are that the results are sensitive to heuristic gap penalties, and it may be difficult to determine which portions of the aligned sequences are more highly conserved than others.

In practice, no more than three protein sequences can be aligned using the straightforward multidimensional extension of the Needleman-Wunsch algorithm, mainly because of the limitations of memory and calculation time (Murata et al. 1985).

Lipman et al. (1989) presented a new dynamic programming algorithm that can find the globally optimal alignment without search-
Homology Model Building

iterating the entire multidimensional comparison matrix, but even this method is limited in practice to only four or five sequences.

**Iterative Construction of a Multiple Alignment from Pairwise Global Sub-alignments**

This method constructs a multiple alignment from repeated global alignments of pairs of sequences. The algorithm begins with the alignment of two sequences using a Needleman-Wunsch or similar method. A third sequence is then aligned to one of the first two, or to the alignment of the first two (i.e., to a consensus or average sequence derived from the alignment). This step is repeated for each of the remaining sequences to be aligned.

This method is automatic and has the advantages of high speed and low memory requirements. Its main disadvantage is that the final result depends on the order in which the sequences are added into the composite alignment. Significant regions of similarity among the sequences can sometimes be missed because of this.

There are many variations of this approach in the literature. A popular strategy is to begin by performing all possible global pairwise alignments. The sequences are then added into the composite alignment in the order determined by their similarity scores, from most to least closely related (Barton 1990). Another variation is that of Berger and Munson (1991), in which the groupings for the construction of the composite alignment are chosen randomly. The alignment is repeated with different randomly-chosen subgroupings until the alignment score shows no improvement with further iterations.

**Multiple Alignment Based on Local Similarity Measures**

This category comprises a wide variety of algorithms, including the one used here. Most of these algorithms do not use dynamic programming methods, but instead build the alignment by searching for local regions of similarity among the sequences. Examples are the methods of Johnson and Doolittle (1986), Bacon and Anderson (1986), Vingron and Argos (1989), Depiereux and Feytmans (1991), and Schuler et al. (1991). Most compare relatively short, fixed-length segments, typically 10 – 20 residues, at different relative positions in the sequences. In some algorithms the range of the relative positions may be restricted to a fixed-length window, also about 10 – 20 residues long, to reduce memory requirements and calculation time. Most of these algorithms are fast and efficient in
2. Theory

their use of memory, although in some implementations, calculation time increases exponentially with the number of sequences (e.g., Johnson and Doolittle 1986).

One of the more recent methods in this category has special advantages for homology modeling. This is the segment pair overlap algorithm developed by Schuler et al. (1991) for their program MACAW. The main advantage of this method is that it not only aligns the sequences, but it also identifies blocks of related sequence segments and estimates the probabilities that these high-scoring blocks could be found by chance. Those blocks showing high statistical significance are likely to contain structurally conserved regions. The method is also fast and, except when used with highly repetitive sequences, efficient in its use of memory. Another advantage of this method is that it imposes no restrictions on the length of a block or on the relative positions of the segments that constitute the block. The Alignment/Multiple_Sequence command uses a modified version of the segment pair overlap algorithm.

A significant innovation in MSI’s implementation of the method is that it automatically controls the repetitive application of the segment pair overlap search. This automated search strategy is described in detail following a description of the basic segment pair overlap algorithm.

**Overview of the Segment Pair Overlap Algorithm**

Strictly speaking, the segment pair overlap algorithm (Schuler et al. 1991) is not an alignment algorithm. Rather, it is a search algorithm, the purpose of which is to find related sequence segments. The insertion of gaps to align these segments is not part of the method itself, but must instead be done as a separate operation after all related sequence segments have been found.

The core of the segment pair overlap algorithm is the search for blocks of mutually related sequence segments. The procedure imposes on this search the constraint that all pairs of segments within the block must show some minimal degree of relatedness. This simple and reasonable constraint eliminates vast regions of the space to be searched, thus greatly reducing calculation time. Once the blocks of related segments are found, the alignment is generated simply by inserting gaps as necessary to bring the related segments into alignment. No gap penalty is required.
Schuler et al. (1991) introduced two concepts essential to the understanding of their search algorithm: the \textit{m-diagonal} and the \textit{m-block}. A diagonal in their terminology is simply an alignment, with no gaps, between two or more sequences. A 2-diagonal specifies an alignment of two sequences, and an m-diagonal specifies an alignment of m sequences. For example, consider the two short sequences MAIRLY and MAILY. Figure 14 shows a complete list of the ten possible 2-diagonals that can be constructed from these sequences.

<table>
<thead>
<tr>
<th>MAIRLY</th>
<th>MAIRLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIRLY</td>
<td>MAIRLY</td>
</tr>
<tr>
<td>MAILY</td>
<td>MAIRLY</td>
</tr>
<tr>
<td>MAILY</td>
<td>MAIRLY</td>
</tr>
<tr>
<td>MAILY</td>
<td>MAIRLY</td>
</tr>
<tr>
<td>MAILY</td>
<td>MAIRLY</td>
</tr>
</tbody>
</table>

\textbf{Figure 14. All Possible 2-diagonals for Two Sequences}

The term diagonal comes from the fact that the aligned residues can be represented as diagonal lines in a residue comparison matrix. For example, two of the 2-diagonals from the above list are shown in Figure 15.
Similarly, a 3-diagonal could be represented as a diagonal line passing through a three-dimensional residue comparison matrix. A 3-diagonal is said to “contain” three 2-diagonals. For example, the 3-diagonal:

```
MAIRLY
MAILY
MALIL
```

contains these three 2-diagonals

```
MAIRLY
MAILY
MAILY
```

```
CWIIRLY
MAILY
```

In the Schuler et al. (1991) terminology, an m-block is a subregion of an n-diagonal (n ≥ m) comprising aligned segments of equal length (with no gaps), one from each of the m sequences in the m-block. For example, the sequence segments enclosed by the rectangular box below constitute one of many possible 3-blocks contained within this 3-diagonal
The goal of the segment pair overlap algorithm is to find blocks of similar sequence segments. The above example illustrates such a block, since the three segments it contains are identical. Normally the similarities of the aligned residues are calculated from a residue scoring matrix such as the PAM 250 matrix (Dayhoff et al. 1978), but other scoring matrices can be used.

For purposes of illustration, consider measuring the similarity of segments in an m-block using the identity matrix. Two aligned residues receive a score of 1 if they are identical and a score of 0 otherwise. When more than two sequence segments are aligned, such as in the above 3-block, the score is calculated as the sum of all possible pairwise scores. In the terminology of Schuler et al. (1991), this is the \( \text{SP score} \) (Sum of Pairs). For example, three different pairs can be made from the three aligned leucine residues in the above 3-block. Each pair receives a score of 1, so the SP score of the column of leucine residues is \( 1 + 1 + 1 = 3 \). Likewise the SP score of the column of tyrosine residues is 3. The SP score for the entire 3-block is defined as the sum of the SP scores of all of the columns it contains, or \( 3 + 3 = 6 \) in the above example. The SP score is the only measure of sequence similarity actually used in the process of searching for blocks.

The segment pair overlap algorithm comprises these three steps:

1. Find all 2-diagonals containing 2-blocks with scores that exceed a user specified value. This value is called the pairwise score threshold. It is during this step that huge regions of the alignment search space are eliminated from further consideration.

2. From the list of 2-diagonals found in step 1, construct all possible m-diagonals such that all of the 2-diagonals contained in each of them are in the list from step 1. In other words, find all m-diagonals for which all contained 2-diagonals pass the test in step 1. Schuler et al. refer to this step as the expanding or merging of 2-diagonals into m-diagonals. Those m-diagonals for which \( m \) is greater than a user-specified value are kept for use in step...
2. Theory

3; all others are discarded. This user-specified value is the minimum number of sequences that each m-block must contain.

3. Schuler et al. (1991) refer to the final step of the algorithm as the parsing of m-diagonals into blocks. For each of the m-diagonals found in step 2, find all 2-blocks within the diagonal that have SP scores greater than the pairwise score threshold. Those regions over which a sufficient number of these 2-blocks overlap are identified and reported as m-blocks likely to contain homologous sequence segments. It is from this step that the segment pair overlap algorithm gets its name. What constitutes a “sufficient number” of overlapping 2-blocks is one of the heuristics of the algorithm. In the implementation of Schuler et al. (1991), this number is $1 + \frac{(m - 1)(m - 2)}{2}$. This criterion guarantees that each sequence segment in the m-block is contained in at least one of the overlapping 2-blocks. An alternative (and more discriminating) overlap criterion requires that $\frac{m(m - 1)}{2}$ 2-blocks must overlap to form an m-block. This strict overlap criterion guarantees that each sequence segment in the m-block is contained in all of the overlapping 2-blocks. Either of these two overlap criteria can be used in the Alignment/Multiple_Sequence command. Their advantages and disadvantages are discussed in Chapter 5, Methodology.

Estimating the Statistical Significance of an m-block Not all of the m-blocks found by a segment pair overlap search are equally significant. Large m-blocks with high SP scores may be highly significant and probably contain structurally conserved regions. Smaller m-blocks, however, might have positive but relatively low SP scores that might occur by chance, even in a set of random sequences.

Recent studies in statistical theory (Karlin et al. 1990, Karlin and Altschul 1990) provide a simple method for estimating the statistical significance of blocks of high scoring sequence segments. The essential question is: Given a high-scoring block found by searching a set of sequences, what is the probability that a block with equal or greater score could be found by chance in a search of random, unrelated sequences of the same lengths?

The theory assumes a model of chance in which each of the 20 amino acids has a specific probability of occurrence. In practice, these probabilities are estimated from the measured frequencies of
occurrence of the amino acids in real proteins. The theory also assumes that these probabilities are position-independent and have no Markov dependence (that is, the occurrence of a particular amino acid at position $i$ in the sequence has no effect on the probabilities of occurrence of the amino acids at position $i + 1$). Although this is a simplified model for real proteins, it allows the derivation of a simple equation for estimating statistical significance.

$$p = 1 - \exp \left[ -KN \exp \left( -\lambda S \right) \right]$$  \hspace{1cm} \textbf{Eq. 1}

where $p$ is the probability that a block having score greater than or equal to $S$ could be found by searching a sequence space $N$ (the product of the lengths of the individual sequences). The constants $K$ and $\lambda$ depend on the number of sequence segments in the block and on the values in the scoring matrix (for details see Karlin and Altschul 1990).

Statistical significance only has meaning with respect to a null hypothesis. When significance is estimated using Eq. 1, a critical element of the null hypothesis is $N$, the size of the search space. In some cases, calculating $N$ as the product of the sequence lengths may not be appropriate. For example, consider an alignment of three sequences A, B and C. Suppose sequences B and C were chosen for alignment with A only because a search of a large sequence database showed them to be similar to A. In this case the effective length of sequences A and B used in calculating $N$ should be the length of the entire database, since this is a more meaningful measure of the space actually searched. For these reasons, a $p$ value calculated from Eq. 1 should never be published without a description of the null hypothesis and the method by which the search space $N$ was calculated. For further discussion see Karlin and Altschul (1990), Karlin and Brendel (1992), and Schuler et al. (1991).

The results of a segment pair overlap search do not by themselves constitute an alignment of the sequences. Gaps must still be inserted to bring the segments of each m-block into alignment. For example, consider the three sequences given below:
2. Theory

Assume that a segment pair overlap search of these sequences finds these m-blocks:

1: \text{D M A I R L Y W S}
2: \text{M A I L Y W S}
3: \text{C W I Q L Y W S T}

To bring these segments into alignment, gaps must be inserted thus:

1: \text{L Y W S}
2: \text{L Y W S}
3: \text{L Y W S}

1: \text{M A I}
2: \text{M A I}

Important Characteristics of the Segment Pair Overlap Algorithm

The user-specified pairwise score threshold (see step 1 on page 2-43) has a great effect on the performance of the algorithm. If this threshold is low, a single search consisting of the three steps listed on page 2-43 will find many m-blocks, but the search will be slow and memory-intensive. If the threshold is high, the search will be fast, but it will find fewer m-blocks (possibly none at all). If the sequences contain many internally repeated segments, then the number of diagonals to search, and the consequent demands on memory, may grow exponentially with the number of sequences.
These characteristics are well suited to a “divide-and-conquer” strategy for multiple alignment. The essential idea is to begin with a search that uses a high pairwise score threshold. This quickly finds the most statistically significant m-blocks. These m-blocks divide the sequences into shorter subregions, each of which can then be searched separately using lower pairwise score thresholds. Since these searches involve only portions of the sequences, rather than their entire lengths, they require much less time and memory than would a search at the same low threshold over the full lengths of the sequences. Dividing the sequences in this way also reduces the problems caused by highly repetitive sequences.

**MSI’s Automatic Divide-and-Conquer Algorithm** A significant innovation in the implementation of the Alignment/Multiple_Sequence command is that the divide-and-conquer strategy just described is fully automated.

The search begins with a pairwise score threshold specified by the user. The initial search is further constrained by the requirement that \( m \), the number of sequence segments in the m-blocks found, must be equal to the total number of sequences being searched. If no m-blocks are found using the user-specified threshold, then the threshold is lowered by a fixed percentage and the search is repeated. This cycle of searching and threshold reduction is repeated until one of the following two conditions is met:

1. One or more statistically significant m-blocks are found, or

2. The current threshold is so low that there is little chance that further searches at lower thresholds would find significant m-blocks. (The criteria for recognizing this condition are discussed in detail in Chapter 5, *Methodology*.)

If condition 1 is met, then the newly found m-blocks are saved for alignment, and the process of searching and threshold reduction is repeated for the remaining regions, or *zones*, between the significant m-blocks.

If condition 2 occurs, then any further searching at lower thresholds would probably be unproductive. There may, however, be undiscovered significant m-blocks for which \( m \) is less than the number of sequences being aligned. At this point, therefore, the constraint on \( m \) is decremented. In other words, the number of segments in the m-block is allowed to be one less than the number of
2. Theory

sequences being aligned. At the same time, the pairwise score threshold is reset to the original specified value. Again, the remaining zones are searched repeatedly, the threshold being decremented after each search, until either condition 1 or 2 is met.

This entire process is repeated until the constraint on $m$ reaches a user-specified minimum level, or until the remaining zones between m-blocks are so short that they are not worth searching. The latter condition is detected by comparing the zone lengths to a user-specified minimum searchable zone length.

The segment pair overlap algorithm may occasionally find m-blocks that are incompatible with each other. For example, the sequences listed below:

$$
\text{DNAIRLYWS} \\
\text{MAILYWS} \\
\text{CWILYWSTMAL}
$$

contain two incompatible m-blocks:

$$
\text{MAIL} \quad \text{LYWS} \\
\text{MAIL} \quad \text{LYWS} \\
\text{MAIL} \quad \text{LYWS}
$$

The problem in this case is that M A I is on the left of L Y W S in the first two sequences but on the right in the third sequence. It is impossible to align all three segments of both m-blocks simultaneously. The divide-and-conquer algorithm used in Homology resolves such conflicts by discarding the least statistically significant of the conflicting m-blocks. In the above example, if the 3-block of L Y W S segments were more significant, it would be kept and the 3-block of M A I segments would be discarded. The sequences would then be divided into two zones for further searching, one on the left and the other on the right of the L Y W S 3-block. If the 2-block of M A I segments in the left zone were significant, it would be found in a subsequent search of this zone and added to the alignment.
Homology Model Building

Finally, after all the significant, nonconflicting m-blocks have been found, the algorithm automatically inserts gaps to bring all segments of the m-blocks into proper alignment.

align123

align123 is a multiple sequence alignment method based on the CLUSTAL W program (Thompson et al, 1994a). CLUSTAL W aligns multiple sequences using a progressive pairwise alignment algorithm. In align123, the term for scoring a match of secondary structure is added to the original CLUSTAL W multiple sequence alignment score.

align123 first generates all possible pairwise alignments for a list of sequences and then builds a guide tree based on their pairwise sequence identity, aligning the sequences following the order of the guide tree.

Several unique features in align123 improve the sensitivity of the alignment of divergent protein sequences (Thompson et al, 1994a).

1. Individual weights are assigned to each sequence in a partial alignment in order to downweight near-duplicate sequences and upweight the most divergent ones.

2. Amino acid substitution matrices are varied at different alignment stages according to the divergence of the sequences to be aligned.

3. Residue-specific gap penalties and locally reduced gap penalties in hydrophilic regions encourage new gaps in potential loop regions rather than regular secondary structure.

4. Positions in early alignments where gaps have been opened receive locally reduced gap penalties to encourage the opening of new gaps at these positions.

Multiple sequence alignment

The basic multiple alignment algorithm consists of three main stages:

1. All pairs of sequences are aligned separately in order to calculate a distance matrix giving the divergence of each pair of sequences.

2. A guide tree is calculated from the distance matrix.
2. Theory

3. The sequences are progressively aligned according to the branching order in the guide tree.

1. The distance matrix/pairwise alignments

In the first stage, the pairwise distances are calculated based on all pairwise sequence alignments. Two methods are offered to perform the pairwise alignments, the FASTA type approach and the full dynamic programming approach. Based on the pairwise alignment, the evolutionary distance between two sequences is defined as the mean number of differences per site in an alignment (ignoring sites with a gap):

\[ \Delta (\text{distance}) = 1.0 - \frac{\text{percent identity}}{100} \]  
\[ \text{Eq. 2} \]

2. The guide tree

The trees used to guide the final multiple alignment process are calculated from the distance matrix of step 1 using the Neighbor-Joining method (Saitou et al, 1987). This produces unrooted trees with branch lengths proportional to the estimated divergence along each branch. The root is placed by a mid-point method (Thompson et al, 1994b) at a position where the means of the branch lengths on either side of the root are equal (Figure 16). These trees are also used to derive a weight for each sequence (Thompson et al, 1994b). The weights are dependent upon the distance from the root of the tree, and sequences which have a common branch with other sequences share the weight derived from the shared branch. As depicted in Figure 16(b), the weight for any sequence \( S \) which has a path consistency of \( n \) branches to the root can be calculated as:

\[ W = \sum_{i=1}^{n} \left( \frac{\Delta_i}{O_i} \right) \]  
\[ \text{Eq. 3} \]

Where \( O_i \) is the branch order defined as the number of sequences that own the branch.

The weights are then normalized with the maximum weight equal to 1.0.
Figure 16. (A) An an unrooted tree such as can be built by the neighbor-joining method. The numbers along the branches are the percentage divergence of the sequences. (B) The same tree rooted by PROFILEWEIGHT. The weights assigned to each sequence are shown on the right-hand side. *(3.0/1 + 2.0/2 + 0.7/3) = 4.2
2. Theory

3. Progressive alignment

The basic procedure at this stage is to use a series of pairwise alignments to align larger and larger groups of sequences, following the branching order in the guide tree. The alignment proceeds from the tips of the rooted tree towards the root. At each stage, a full dynamic programming algorithm is used to align two existing alignments or sequences. The score at each alignment position is the average of all the pairwise substitution matrix scores from the amino acids in the two sets of sequences. If either set of sequences contains one or more gaps in one of the positions being considered, each gap versus a residue is scored as zero. The default amino acid substitution matrices in align123 are rescored to have only positive values. Therefore, this treatment of gaps treats the score of a residue versus a gap as having the worst possible score. When sequences are weighted, each substitution matrix value is multiplied by the weights from the two sequences.

Sequence weighting

Sequence weights calculated from the guide tree are used as simple multiplication factors for scoring positions from different sequences or prealigned groups of sequences. Groups of closely related sequences receive lowered weights because they contain much duplicated information. Highly divergent sequences without any close relatives receive high weights.

Initial gap penalties

You may set initial values of the gap opening penalty (GOP) and gap extension penalty (GEP) from the menu. align123 then automatically attempts to adjust the gap penalties for each sequence alignment, depending on the following factors.

1. Dependence on the substitution matrix

Varying the gap penalties used with different substitution matrices can improve the accuracy of sequence alignments. align123 uses the average score for two mismatched residues (i.e., off-diagonal values in the matrix) as a scaling factor for the GOP.

2. Dependence on the similarity of the sequences

The percent identity of the two (groups of) sequences to be aligned is used to increase the GOP for closely related sequences and decrease it for more divergent sequences on a linear scale.
3. Dependence on the lengths of the sequences

The scores for both true and false sequence alignments grow with the length of the sequences. GOP is scaled by the logarithm of the length of the shorter sequence.

Using these three modifications, the GOP calculated by the program is:

\[ \text{GOP} \rightarrow (\text{GOP} + \log(\text{MIN}(N, M))) \cdot (\text{average residue mismatch score}) \cdot (\text{percent identity scaling factor}) \]  

Eq. 4

Where \( N \) and \( M \) are the lengths of the two sequences.

4. Dependence on the difference in the lengths of the sequences

The GEP is modified depending on the difference between the lengths of the two sequences to be aligned. If one sequence is much shorter than the other, the GEP is increased to inhibit too many long gaps in the shorter sequence. The initial GEP calculated by the program is:

\[ \text{GEP} \rightarrow \text{GEP} \left( 1.0 + \log\left( \frac{N}{M} \right) \right) \]  

Eq. 5

Where \( N \) and \( M \) are the lengths of the two sequences.

Position-specific gap penalties

In most dynamic programming applications, the initial gap opening and extension penalties are applied equally at every position in the sequence, regardless of the location of a gap, except for terminal gaps which are usually allowed at no cost. In align123, before any pair of sequences or prealigned groups of sequences are aligned, a table of gap opening penalties for every position in the two (sets of) sequences are generated. align123 treats the initial gap opening penalty in a position-specific manner, in order to make gaps more or less likely at different positions.

The local gap penalty modification rules are applied in a hierarchical manner. The exact details of each rule are given below. First, if there is a gap at a position, the gap opening and gap extension penalties are lowered; the other rules do not apply. This makes gaps more likely at positions where there are already gaps. If there is no gap at a position, then the gap opening penalty is increased.
2. Theory

if the position is within eight residues of an existing gap. This discourages gaps that are too close together. Finally, at any position within a run of hydrophilic residues, the penalty is decreased. These runs usually indicate loop regions in protein structures. If there is no run of hydrophilic residues, the penalty is modified using a table of residue specific gap propensities (S. Pascarella and P. Argos, 1992). These propensities were derived by counting the frequency of each residue at either end of gaps in alignments of proteins of known structure.

1. Lowered gap penalties at existing gaps

If there are already gaps at a position, then the GOP is reduced in proportion to the number of sequences with a gap at this position and the GEP is lowered by a half. The new gap opening penalty is calculated as:

\[ \text{GOP} \rightarrow \text{GOP} \cdot 0.3 \cdot \left( \frac{\text{no. of sequences without a gap}}{\text{no. of sequences}} \right) \]

**Eq. 6**

2. Increased gap penalties near existing gaps

If a position does not have any gaps but is within eight residues of an existing gap, the GOP is increased by:

\[ \text{GOP} \rightarrow \text{GOP} \left( 2 + \left( \frac{8 - \text{distance from gap}}{2} \right) \right) \]

**Eq. 7**

3. Reduced gap penalties in hydrophilic stretches

Any run of five hydrophilic residues is considered to be a hydrophilic stretch. You may specify residues that are to be considered hydrophilic, but they are conservatively set to D, E, G, K, N, Q, P, R or S by default. If, at any position, there are no gaps and any of the sequences has such a stretch, the GOP is reduced by one third.

4. Residue specific penalties

If there is no hydrophilic stretch and the position does not contain any gaps, then the GOP is multiplied by one of the 20 numbers in Table 1, depending on the residue. If there is a mixture of residues
at a position, the multiplication factor is the average of all the contributions from each sequence.

Table 1. Pascarella and Argos residue specific gap modification factors.

<table>
<thead>
<tr>
<th>Residue</th>
<th>Modification Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.13</td>
</tr>
<tr>
<td>C</td>
<td>1.13</td>
</tr>
<tr>
<td>D</td>
<td>0.96</td>
</tr>
<tr>
<td>E</td>
<td>1.31</td>
</tr>
<tr>
<td>F</td>
<td>1.20</td>
</tr>
<tr>
<td>G</td>
<td>0.61</td>
</tr>
<tr>
<td>H</td>
<td>1.00</td>
</tr>
<tr>
<td>I</td>
<td>1.32</td>
</tr>
<tr>
<td>K</td>
<td>0.96</td>
</tr>
<tr>
<td>L</td>
<td>1.21</td>
</tr>
<tr>
<td>M</td>
<td>1.29</td>
</tr>
<tr>
<td>N</td>
<td>0.63</td>
</tr>
<tr>
<td>P</td>
<td>0.74</td>
</tr>
<tr>
<td>Q</td>
<td>1.07</td>
</tr>
<tr>
<td>R</td>
<td>0.72</td>
</tr>
<tr>
<td>S</td>
<td>0.76</td>
</tr>
<tr>
<td>T</td>
<td>0.89</td>
</tr>
<tr>
<td>V</td>
<td>1.25</td>
</tr>
<tr>
<td>Y</td>
<td>1.00</td>
</tr>
<tr>
<td>W</td>
<td>1.23</td>
</tr>
</tbody>
</table>

The values are normalized around a mean value of 1.0 for H. The lower the value, the greater the chance of having an adjacent gap. These are derived from the original table of relative frequencies of gaps adjacent to each residue by subtraction from 2.0.

Secondary_Structure

The term for scoring a secondary structure match is added to the multiple sequence alignment score but not to the initial pairwise alignment used to build the guide tree.

The secondary structure profile score at a specific alignment position for a set of sequences is calculated as:

\[
ss_{prf}(\text{Helix}) = \frac{\sum SS_{score}(\text{Helix}) \cdot W_i}{\sum W_i}
\]

\[
ss_{prf}(\text{Strand}) = \frac{\sum SS_{score}(\text{Strand}) \cdot W_i}{\sum W_i}
\]

Eq. 8
2. Theory

where $SS_{score}$ is the confidence score for the predicted secondary structure at a given position in the sequence. It is set to 0 if the secondary structure of that sequence does not exist. Otherwise, it is set to a scale of increasing confidence from 1 to 9. The confidence score is always set to 9 for Kabsch_Sander classification. $seqWeight$ is computed from the guide tree. At any given alignment position, then, the secondary structure alignment score between two sets of sequences can be calculated as:

$$SS_{score} = SS_{Wt} \cdot ss_{pfr1} (Helix) \cdot ss_{pfr2} (Helix)$$

$$+ SS_{Wt} \cdot ss_{pfr1} (Strand) \cdot ss_{pfr2} (Strand)$$

Eq. 9

where $SS_{Wt}$ is the score for matching a secondary structure element.

Substitution matrices

Three main series of substitution matrices are available to you:

- Dayhoff PAM series (Dayhoff et al., 1978)
- BLOSUM series (Henikoff et al., 1992)
- Gonnet series (Gonnet et al., 1992)

The default is the BLOSUM series. In each case, there is a choice of matrix ranging from strict ones, useful for comparing very closely related sequences to very “soft” ones that are useful for comparing very distantly related sequences. Depending on the distance measured from the guide tree between the two sequences or groups of sequences to be compared, different matrices from the selected series are used.

### Table 2. Relationship of Sequence Distance to Substitution Matrix

<table>
<thead>
<tr>
<th>Distance</th>
<th>BLOSUM series</th>
<th>Distance</th>
<th>PAM series</th>
<th>Distance</th>
<th>Gonnet series</th>
</tr>
</thead>
<tbody>
<tr>
<td>80-100</td>
<td>BLOSUM80</td>
<td>80-100</td>
<td>PAM20</td>
<td>65-100</td>
<td>Gonnet40</td>
</tr>
<tr>
<td>60-80</td>
<td>BLOSUM62</td>
<td>60-80</td>
<td>PAM60</td>
<td>45-65</td>
<td>Gonnet80</td>
</tr>
<tr>
<td>30-60</td>
<td>BLOSUM45</td>
<td>40-60</td>
<td>PAM120</td>
<td>35-45</td>
<td>Gonnet120</td>
</tr>
<tr>
<td>0-30</td>
<td>BLOSUM30</td>
<td>0-40</td>
<td>PAM350</td>
<td>25-35</td>
<td>Gonnet160</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15-25</td>
<td>Gonnet250</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0-15</td>
<td>Gonnet300</td>
</tr>
</tbody>
</table>

---

56  Homology
Divergent sequences

The most divergent sequences (most different, on average from all of the other sequences) are usually the most difficult to align correctly. It is sometimes better to delay the incorporation of these sequences until all of the more easily aligned sequences are merged first. This may give a better chance of correctly placing the gaps and matching weakly conserved positions against the rest of the sequences. A choice is offered to set a cutoff (default is 40% identity or less with any other sequence) that will delay the alignment of the divergent sequences until all of the rest have been aligned.

Alignment to an alignment

Profile alignment is used to align two existing alignments (either of which may consist of just one sequence) or to add a series of new sequences to an existing alignment. This is useful because one may wish to build up a multiple alignment gradually, choosing different parameters manually, or correcting intermediate errors as the alignment proceeds. Often, just a few sequences cause misalignments in the progressive algorithm and these can be removed from the process and then added at the end by profile alignment. A second use is where one has a high quality reference alignment and wishes to keep it fixed while adding new sequences automatically.

Scoring Matrices

When the Alignment/Pairwise_Sequence command sets up a residue comparison matrix, it can consult any one of four standard scoring matrices or a customizable matrix that you supply as a text file.

These matrices represent the most likely amino acid substitutions based on physical, chemical, or statistical criteria. High values in the matrix imply that a substitution can occur, and low values imply that a substitution is unlikely. The four standard scoring matrices available to the Alignment/Pairwise_Sequence command are as follows:

- The identity matrix. This is a simple matrix that gives a score of one for amino acids that match and a zero score for any amino acids that do not. The utility of this matrix is limited, but it is most effective when you are using the automatic alignment method with SCRs (see MSI’s Automatic Multiple Sequence Alignment on page 2-38).
2. Theory

♦ The codon substitution matrix. The values for this matrix are derived from the DNA base triplets coding for the amino acid pairs. For each pair, all of the possible nucleotide triplets are examined. A score is assigned based on the minimum number of point mutations required to transform one amino acid into the other. Identical amino acids get a score of 9. One required point mutation yields a score of 3, and two mutations yield a score of 1. If all bases are different, a score of 0 is assigned.

♦ The mutation matrix. Also known as the Dayhoff or PAM250 matrix (Dayhoff et al. 1983), this matrix is obtained by examining the number of substitutions from one amino acid to another in related proteins across different species. It emphasizes conservation of special residue types, such as cysteine and tryptophan. It also gives acceptable scores for residues that do not match. This is acceptable in most cases since it allows the substitution of amino acids with similar properties. Large scores are given to identities and substitutions that frequently occur, and low scores are given to substitutions that are not observed. This is the most widely used scoring matrix and is used to find an initial alignment when two sequences have just been read in.

♦ The hydrophobicity matrix. This matrix is derived from the hydrophobicity scale of Engelman et al. (1986). This scale (see Appendix F, Hydrophobicity Scale Values) lists the relative hydrophobicities of the 20 amino acids in a range from –3.7 to 12.3. For any pair of amino acids, the score, $S_{12}$, is calculated as:

$$S_{12} = R - |H_1 - H_2|$$  \hspace{1cm} \text{Eq. 10}

where $R$ is the range of possible values (16), and $H_1$ and $H_2$ are the individual hydrophobicity values.

You can also define a custom matrix using your own scoring criteria:

♦ The user matrix. This matrix is defined to measure another physical or genetic property of your choice. It is stored in a file consisting of 20 lines with 20 values in each line corresponding to all combinations of the amino acids. This matrix is loaded by choosing User as the Scoring Matrix in the command in the Alignment pulldown.
Superimposing structures based on sequence alignment

Unlike the `Alignment/Pairwise_Sequence` command, the `Alignment/Multiple_Sequence` command can use only two scoring matrices. One of these is the Dayhoff PAM 250 mutation matrix described above. The other is the Dayhoff PAM 120 mutation scoring matrix. The two matrices were derived from the same evolutionary data, but differ in the way they were normalized. The PAM 250 matrix is normalized for use with more distantly related sequences. The relative advantages and disadvantages of these two matrices are discussed in Chapter 5, *Methodology*.

Superimposing structures based on sequence alignment

For sequences that have associated three-dimensional structures, automatic structural superimposition can be used to

- Superimpose structures according to their sequence alignments
- Guide you in manually adjusting the alignment

The *quaternion algorithm* (Sutcliffe et al.) is used to minimize the RMS deviation over all corresponding pairs of alpha carbon atoms of all residues which are aligned in every structure. These residues may satisfy certain criteria: either spatial proximity, or physical and chemical similarity. Alternatively, all the aligned residues can be superimposed regardless of their coordinates or chemical nature. Residues selected this way are called *matches*.

Spatial proximity of two residues is determined as the distance between the C-alpha atoms of those residues after superimposing the whole structures.

Physical and chemical similarity is defined according to any of the following criteria:

- **Secondary Structure** - residues belong to the same type of secondary structure
- **Strong Similarity** - residues are identical or strongly conserved
- **Weak Similarity** - residues are identical, strongly or weakly conserved
- **Identity** - residues are identical
2. Theory

*Strong similarity* and *weak similarity* are defined as follows.

Two residues are considered to be strongly similar if they both belong to one of the following groups:

- STA
- NEQK
- NHQK
- NDEQ
- QHRK
- MILV
- MILF
- HY
- FYW

Two residues are considered to be weekly similar if they both belong to one of the following groups:

- CSA
- ATV
- SAG
- STNK
- STPA
- SGND
- SND
- EQK
- NDEQHK
- NEQHRK
- FVLIM
- HFY

The *quaternion algorithm* is the core of our superposition schema which has two levels of iterations:

**Inner level of iterations** Only a fixed set of residues satisfying the criteria mentioned above are involved in calculating the necessary spatial transformations of the coordinates of the selected structures. In the first iteration, all structures are superimposed on the first using the quaternion algorithm, and the frame set of coordinates is constructed by averaging the resulting superimposed coordinates. In each of the further iterations (the total number is internally set to 10), the structures are superimposed on the current frame with subsequent frame recalculation.

**Outer level of iterations** If matches have to be selected using spatial proximity criteria, all the aligned residues are used for the ini-
Superimposing structures based on sequence alignment

tial superimposition and the creation of the initial set of matches. During each of the subsequent (user-specified) number of iterations, structures are superimposed using the current set of matches. If you so specify, the superimposition is done automatically with the calculation of matches. This is especially useful when you want the changes in the alignment to be automatically reflected in the structural superimposition (see Superimposing structures in Methodology for more information).

Multiple Structure Alignment

The divide-and-conquer multiple alignment algorithm is not limited to alignment based on sequence similarity. It requires only some means of identifying regions of pairwise similarity between two sequences. In multiple sequence alignment, these regions of local pairwise similarity are the 2-blocks found in step 1 of the segment pair overlap search (page 43). The 2-blocks found by the automatic structure alignment algorithm (see Automatic Determination of Structurally Conserved Regions on page 2-20) can be used in an analogous way for a divide-and-conquer multiple structure alignment.

The multiple structure alignment algorithm begins by finding all such structurally-conserved 2-blocks in all possible pairs of the sequences selected for alignment. It then merges them into diagonals of higher dimension and parses these into m-blocks in much the same way as is done in multiple sequence alignment. The 2-blocks are merged into m-blocks of more than two sequences wherever a sufficient number of the overlapping 2-blocks are mutually consistent. What constitutes a “sufficient number” of overlapping 2-blocks depends on whether or not the user-selectable strict overlap criterion is used. When the strict overlap criterion is used, the number of overlapping 2-blocks required to form an m-block is \( m (m - 1)/2 \); that is, all possible pairs in the block of \( m \) sequence segments must be structurally similar. When the strict overlap criterion is not used, there must still be at least one column of the m-block in which all possible pairs lie in mutually consistent overlapping 2-blocks, but the m-block may extend into columns where less than the maximum possible number of 2-blocks overlap. In these extended regions, the number of overlapping 2-blocks must be at least \( 1 + (m - 1) (m - 2)/2 \).
2. Theory

Blocks of the highest dimension are found first, and the unaligned zones between these are then recursively searched for blocks of lower dimension. The result is an alignment of multiple sequences based not on their sequence similarity, but on the similarity of their corresponding three-dimensional structures. The algorithm makes no attempt to optimize the alignment of residues between the m-blocks.

**Simultaneous Superposition of Structures**

The m-blocks found in a multiple structure alignment are known to be structurally conserved regions. The m-blocks found in multiple sequence alignment show a high degree of sequence conservation, but should only be considered "putative structurally conserved regions", since no measure of structural similarity was used to find them. Nevertheless, highly significant m-blocks found in multiple sequence alignment often correspond fairly well to structurally conserved regions.

For sequences that have associated three-dimensional structures, the m-blocks found in a multiple alignment (based on either sequence or structural similarity) can be used for the simultaneous superposition of the corresponding structures. The Levenburg-Marquardt algorithm (Press et al. 1988) is used to minimize the RMS deviation over all corresponding pairs of alpha carbon atoms of all the residues aligned within m-blocks. The superposition is done simultaneously for all the structures involved, not iteratively for successive pairs of structures. The result is therefore a globally optimal superposition with respect to the residues in the m-blocks.

**Assignment of Coordinates Within a Conserved Region**

Once the correspondence between amino acids in the reference and model sequences has been made, the coordinates for an SCR can be assigned. The reference proteins’ coordinates are used as a basis for this assignment. Before the coordinates of the reference protein are copied, they are transformed into the same coordinate frame as the model.

Where the side chains of the reference and model proteins are the same at corresponding locations along the sequence of the SCR, all the coordinates for the amino acid are transferred. Where they dif-
Superimposing structures based on sequence alignment

fer, the backbone coordinates are transferred, but the side chain atoms are automatically replaced to preserve the model protein’s residue types.

When a residue is replaced, the replacement residue is first aligned to the backbone of the original residue. After the backbone is aligned, the dihedral angles in common with the residue being replaced are also aligned. In this way, the conformation of the reference side chain is preserved as much as possible. If the model protein’s side chain has more atoms than the reference protein’s, the more distal atoms are given an extended conformation. New charges and potential functions types are taken from the residue library.

Occasionally, the predicted side chain conformation is not adequate. If the local environment of the residue is markedly different in the model as compared to the reference protein, then it is possible that the side chain may bump another part of the molecule. In these cases, it is necessary to perform a conformational search for the affected side chains. Mutually compatible conformations must be found. Since it is not possible to generate and evaluate every possible conformation, a rougher approximation must be used. For this purpose, combinations of commonly found side chain rotamers are used. The details of this procedure are described later in this section.

Assignment of Coordinates for Loop or Variable Regions

During the course of building a model with Homology, it is necessary to find appropriate coordinates for variable regions of the peptide chain. This can be done by finding peptide segments in other proteins that fit properly into the model’s spatial environment or by generating a segment de novo. The former method is implemented in the Search Loops command discussed in the next section. Search Loops has the advantage that all loops found are guaranteed to have internal geometries that are reasonable and that resemble known protein conformations. They may not, however, fit properly into the given model protein framework. Indeed, they may have severe steric overlaps with the already built conserved regions. If this is found to be the case, then it is advisable to use the Generate Loops command instead, which is discussed in Generate Loops Command on page 2-67.
2. Theory

**Search Loops Command**

The **Search Loops** command is used to search the Brookhaven protein database for regions of proteins that meet a defined geometric criterion. This command uses an existing alpha-carbon distance matrix to search for regions of proteins whose alpha-carbon distances best fit those of the selected region of the protein being studied, while meeting the additional constraint of having the specified number of residues present between the regions of interest.

The best fit is calculated in the following way:

1. Distance matrices for all the perflex (pe) and postflex (po) (Figure 17) residues were calculated for the given loop region:

   \[ d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}, \quad i, j \in pe,po \quad \textbf{Eq. 11} \]
Superimposing structures based on sequence alignment

2. This distance matrix is compared to the distance matrices of all possible loop candidates (c) using a distance difference measure calculated over all N perflex and postflex positions.

\[ D_c^2 = \frac{2}{N(N+1)} \sum_{i \in pe,po} \sum_{j \in pe,po, j > 1} \left( d_{ij} - d_{c(i)r_c(j)}^2 \right), i,j \in pe,po \]

**Eq. 12**

**Figure 17. Geometry Definition for the Search Loops Command**
2. Theory

where \( r_c(i) \) is the residue in the candidate loop \( c \) corresponding to the perflex or postflex residue \( i \).

Candidate loop coordinates, in addition to being used as models for flex residues, are supposed to replace the postflex, postflex residues’ coordinates of the target in accordance with the correspondence \( r_c(i) \) for the perflex and postflex residues.

Candidate loops are sorted according to measure \( D_c ^2 \).

The ten best loops from the search are retained for further examination. A flex region is defined as that portion of the molecule which is not included in the search; its geometry is allowed to vary, and is not a criterion in the search. The residues leading up to the flex region are defined to be the preflex residues, and the residues going away from the flex region are defined as the postflex residues.

The number of distances compared in the search can be calculated from \( (N ^2 – N)/2 \), where \( N \) is the number of preflex residues plus postflex residues.

The \textbf{Search Loops} command can be used to find suitable geometries for variable regions. The results of the search are a direct result of the files used to build the distance matrix. The inclusion of all of the Brookhaven files when building the distance matrix may lead to certain searches where the result is a selection between very similar proteins since the Brookhaven database has varying numbers of solved structures for different types of proteins.

The distance matrix is built with an external utility provided with \textit{Insight II}. Please refer to documentation on the \texttt{pdb_find_distance} utility in the \textit{Insight II} User Guide, Chapter 11.

The results of the \textbf{Search Loops} command are displayed with the \textbf{Display Loops} command. The results of the search are ranked according to “goodness” of fit to the desired structure. \textbf{Loop 1} has the best fit while \textbf{Loop 10} has the poorest. In addition to the actual loop structure, additional pieces of information are presented. The RMS of the alpha-carbon distances is presented along with the name of the file in which the match was found, the starting sequence number for the match, and the actual loop sequence. When the matches are displayed they are superimposed on the original protein. You can control whether the superimposition is done by aligning only the two residues at the base of the flex.
Superimposing structures based on sequence alignment

region, or the alpha-carbons along the entire preflex and postflex residue regions. The RMS value of the superimposition is also presented, and varies depending on the type of superposition selected. An appropriate loop is incorporated automatically using the AssignCoords command in the Loops pulldown.

Generate Loops Command

With the Generate Loops command, a peptide backbone chain is built between two conserved peptide segments using randomly generated values for all the loops’ φ’s and ψ’s. The chain is defined starting from the N-terminal end of the loop being built. A set of distances is defined about the base of the loop that must meet certain criteria in order for the loop to be considered closed. Two atoms in each of the Start and Stop Residues flanking the loop (see Figure 17) define six distances that describe the geometry about the base. Only distances d1 through d4 need to be considered to define the loop closure. The other two (d5 and d6) are bonds along the peptide chain and can be ignored. In the process of closing the loop, the values for the generated φ’s and ψ’s are adjusted until the four distance criteria are met.

Specifically, a function is defined for the distances in terms of the dihedral angles. The differences between the desired distances and their current values are minimized using a linearized Lagrange multiplier method. An exact equation is not used. Instead, an equation valid only for small changes in the dihedral angles is employed, and an iterative approach is taken to achieve the amount of movement required. The function is in the form

\[
\Delta d_i = \sum_{j=1}^{n} D_j^i \Delta \theta_j , \tag{Eq. 13}
\]

where \(i\) ranges from 1 to 4 (the number of distances to fit) and \(n\) is the number of torsion angles. Note that \(n\) must be at least 4 (or 2 residues) in order for this algorithm to work. The quantity \(D_j^i\) is shorthand for...
2. Theory

\[ D_j^i = \frac{\partial D_j}{\partial \theta_i} , \]  \hspace{1cm} \textbf{Eq. 14}

Rearranging and setting all the $\Delta d$’s to 0 (minimizing them to achieve closure) yields a series of equations of the form

\[ 0 = \sum_{i=1}^{4} \lambda_i D_j^i + \Delta \theta_j , \]  \hspace{1cm} \textbf{Eq. 15}

where the $\lambda_i$ are Lagrange multipliers and $j$ ranges from 1 to $n$ (the number of dihedral angles to vary).

The series of equations in Eq. 13 and Eq. 15 can be solved simultaneously to obtain solutions for the values of all the $\Delta \theta$’s. These are kept to a maximum of 10˚ to ensure the linearity of the function and therefore the validity of the approximation in the derivation. (For a complete description of the method, see Shenkin et al. 1987.)

After a series of iterations, the loop is closed, except in the case where the distance between the ends of the loop is too great for an extended chain of the specified number of residues to span. If closure was successful to the degree of accuracy specified by the Convergence criterion within the maximum number of Closure Iterations specified, then the loop is rotated as a rigid body to superimpose it onto the residues on the rest of the protein at either end of the loop. The geometry at the base of the loop is then checked for proper chirality. Loops that have opposite chirality are discarded.
Finally, the loops must be screened on the basis of steric overlap violations. All loops that are found to have unacceptable contacts are rejected. Bumps are not treated as simple hard sphere contacts, but instead, you can specify the degree of overlap that can be tolerated. The minimum allowed distance for any pair of atoms is taken as

\[ R_{\min} = (r_i + r_j) (1 - \text{overlap}) \]  

where \( r_i \) and \( r_j \) are the van der Waals radii of the two atoms and \( \text{overlap} \) is a number from 0 to 1 specifying the fraction of hard sphere interpenetration that is permitted. When any distance is smaller than that allowed, the entire loop is rejected. First, the atoms within the loop are checked against each other, then the loop atoms are checked with the rest of the atoms of the protein. A separate overlap parameter is provided for both internal and

---

Figure 18. Geometry Definitions for the Generate Loops Command
2. Theory

external checks. Note that since energy minimization can correct some bad contacts, a fairly large overlap factor is suggested. Otherwise many loops will be rejected, and the generation of better loops may take considerable time.

The conformations generated for the side chains of the residues in generated loops are arbitrarily set to be fully extended. Extended side chain conformations are never seen in naturally occurring proteins, and so a conformational search must be done. As with the conserved regions, rotamers can be used to reduce the number of conformational states that need to be evaluated.

Side Chain Conformational Searches Using Rotamer Libraries

The prediction of the conformation of a protein is a difficult task. Even assuming that all bond lengths and bond angles remain fixed, there are still two rotatable backbone bonds per residue, Φ and Ψ. Thus, the unambiguous determination of the overall topology of the peptide chain is, by itself, an intractable problem. In addition, the side chains of many of the residues have one or more degrees of freedom. Therefore, even the de novo determination of the best conformations for several moving side chains is not possible.

Fortunately, it has been found through statistical studies of protein structures that side chains adopt only a small number of the many possible conformations available to them. In fact, an amino acid residue with a side chain that can be specified with 2 χ angles has on the order of four to six commonly seen conformations or rotamers (Ponder and Richards 1987). All of these common conformations are combinations of the familiar gauche and anti moieties. For longer side chains, more combinations of the allowed χ angles makes for a greater number of possibilities. But these amino acids are few, and so the number of additional rotamers is not too great.

The correct rotamer to use for a particular residue is largely determined by the local environment. Side chains generally adopt conformations where they are closely packed. Inside the protein core, hydrophobic interactions predominate, and exact complementarity of chemical group topographies is important. Solvent molecules are excluded. Matching of hydrogen bond donors and
acceptors and charged groups is also sometimes important. At the surface, there is much more freedom of movement, but packing is also important there.

Libraries of rotamer frequencies have been compiled (Ponder and Richards 1987, Benedetti et al. 1983). It has been found that among homologous proteins, corresponding residues retain virtually the same rotameric state. Within a range of χ values, 80% of the identical residues and 75% of the mutated residues (Summers et al. 1987) have the same conformations. Also, it has been found that certain rotamers are almost always associated with certain secondary structural motifs (McGregor et al. 1987).

In most situations where such constraints do not apply, the best combination of rotamers for a given set of side chains is difficult to determine. Even with a relatively small number of rotamers to choose from, when several side chains that can interact with each other are considered at once, the combinatorics can be very large. In these cases, not every combination can be tried, and a systematic search procedure must be defined.

If the moving side chains can be parsed into interacting clusters, then it may be possible to test every possible combination of rotamers in each so-called “island” (P. S. Shenkin, private communication). Combinations between islands would not need to be tested because non-interacting side chains would have no influence on each other. This method works for algorithms where the screening criterion is steric overlap. It is much easier to define non-interacting side chains as those that can never touch, regardless of the conformation. When the evaluation criteria include more longer range terms, the islands tend to be bigger, until there is a single cluster for the whole molecule. If nonbond energy terms are included, then there is never a clean cutoff where one cluster ends and another begins.

Another method was suggested by J. Novotny (see Mas et al., 1992). Here a list of residues specified as moving is defined. The best rotamer is selected for the first residue in the list based on energy criteria, that is, the lowest energy. Then, the best rotamer is selected for the next moving side chain, and so on. A cycle is defined as one complete pass through the list. The search stops when the energy does not change appreciably from one cycle to the next. This method has the advantage that it is simple, and in practice, it is fast. One can never be certain, however, that the best
combination has been found since the search is not exhaustive. But reasonable conformations are usually found. This is the algorithm implemented in Homology.

When calculating the nonbond energy for evaluation purposes, not every possible interatomic interaction is evaluated. Only energy terms involving residues included in the list of moving side chains are calculated, and then only when they are within a specified cutoff distance. Thus moving/moving and moving/fixed interactions are evaluated, but fixed/fixed are not. As a result, the reported energy values may be different for a molecule, depending on the number of side chains that have been specified as moving, even if the conformation is the same. This is done in order to speed the calculations as much as possible. In addition, the determination of residue neighbors is redone only when a side chain moves, and then only for that particular side chain.

Another time-saving feature is the internal storage of nonbond energy terms. For each rotamer of each residue that has a moving side chain, the sum of all nonbond energies of interaction with neighboring fixed residues is stored in a look-up table. These energy terms are called “static energies” because they involve interactions with fixed side chains. They are calculated only once, at the beginning of the search, for all rotamers of all active residues (i.e., residues that have moving side chains). Similarly, the nonbond interaction energies between pairs of moving side chains are stored as they are calculated during the conformational search, so that they need only be calculated once.

### Refinement of the Model Using Molecular Mechanics

Many structural artifacts can be introduced into the model protein as it is being built. These include the substitution of large side chains for small ones, strained peptide bonds between segments taken from different reference proteins, and nonoptimum conformations for loops. All of these artifacts can be addressed with the use of molecular mechanics calculations. These include both energy minimization and molecular dynamics.

### The Potential Energy Equation

The potential energy of a molecular system can be represented by:
Superimposing structures based on sequence alignment

\[ E_{pot} = \sum_b D_b \left[ 1 - \exp\left( -\alpha (b - b_0) \right) \right]^2 + \frac{1}{2} \sum_b H_b (\theta - \theta_0)^2 \]

\[ + \frac{1}{2} \sum_q H_q \left[ 1 + s \cos (n\phi) \right] + \frac{1}{2} \sum_{\chi} H_{\chi\chi} X^2 \]

\[ + \sum_b \sum_{b'} F_{bb'} (b - b_0) (b' - b'_0) + \sum_b \sum_{\theta} F_{\theta\theta} (\theta - \theta_0) (\theta - \theta_0) \]

\[ + \sum_b \sum_{\theta} F_{b\theta} (b - b_0) (\theta - \theta_0) + \sum_{\phi} F_{\phi\phi} \cos \phi (\theta - \theta_0) (\theta - \theta_0) \]

\[ + \sum_{\chi} \sum_{\chi'} F_{\chi\chi'} \chi' + \sum_{i} \sum_{j > i} \left[ \frac{A_{ij}}{r_{ij}^2} - \frac{\mathbf{E}_{ij}}{r_{ij}} + \mathbf{q}_i \mathbf{q}_j \right] \]

Eq. 17 describes the potential energy of the molecular system in terms of:

- The energy necessary to stretch a bond.
- The energy necessary to distort the angle between three atoms.
- The energy necessary to rotate atoms about their bond axis.
- The energy necessary to move an atom out of the plane defined by the three atoms to which it is bonded.
- The energy that represents the coupling effects of one of the above energies with another (also known as cross terms).
- The energy associated with the attractive, repulsive, and electrostatic forces between atoms that are not bonded to one another.

These nonbonded energies do not need to be calculated for atoms that are beyond a specified distance from each other because the energy contribution would be small. Furthermore, if the confor-
2. Theory

Information of the model protein is not expected to change dramatically, then the list of atoms that are within the threshold distance does not need to be updated frequently. Both of these approximations can substantially reduce the computation time for a simulation.

Energy Minimization

The Discover program (licensed separately; can be invoked by either Discover or CHARMM license) provides functionality to minimize the energy of a molecular system to produce a model protein that is chemically and conformationally reasonable. The Refine pulldown in the Homology module provides interfaces to the Discover program.

There are many parameters that are associated with the minimization of the energy expression. The interfaces allow for the specification of which optimization algorithm to use. Choices include steepest descents and conjugate gradients. The interfaces also allow for the specification of other parameters, such as the number of iterations to perform, the convergence criterion, whether or not to include cross terms when evaluating the energy, and whether or not to include charges when evaluating the energy. The three commands in the Refine pulldown that interface with Discover — SpliceRepair, Relax, and Explore — each allow you to specify these parameters.

Energy Constraints

A technique commonly used when performing energy minimizations is to include an additional term in the energy expression to bias the expression toward a specific value for a degree of freedom. Energy restraints can be used to perform such tasks as holding or forcing a dihedral angle to a specific value (torsion forcing), or forcing a distance to a specific value (distance restraints), and biasing a portion of a molecule toward its starting conformation (tethering). In particular, the atoms of interest, as well as the force (or penalty) associated with the deviations from the appropriate target, can be defined. Another simple but useful technique is to hold one or more atoms fixed in space while performing the optimization (fixing).
Superimposing structures based on sequence alignment

Constraints can be used for specific regions of the model protein in the SpliceRepair, Relax, and Explore commands in the Refine pulldown. One exception is that of distance restraints, which are done separately using the GenericDis command in the Refine pulldown.

Molecular Dynamics

A known limitation to any optimization algorithm is that only one minimum will be found for a given starting point on the energy surface. Furthermore, this minimum may not be a minimum of interest because it is far away (energetically) from the true or global minimum. Molecular dynamics offers a solution to the minimization problem by taking advantage of the fact that the negative of the energy gradient is equal to the force exerted on the molecule as a consequence of its current conformation.

Given the force and mass for each atom, Newton’s equation of motion ($F=ma$) can be integrated to predict where atoms will move over a short time interval. Molecular dynamics can be used to explore the conformational space a molecule could visit.

The Explore command, in the Refine pulldown, provides a mechanism for specifying the parameters for a dynamics simulation of a single loop region. This command allows you to specify:

♦ the temperature for the simulation.
♦ the number of equilibration steps to be taken to reach the simulation temperature (initialize dynamics).
♦ the total number of steps used for data collection (resume dynamics).
♦ whether or not to include cross terms.
♦ whether or not to include charges during the simulation.

Secondary Structure Prediction

The terms primary, secondary, and tertiary structure are widely used to describe protein architecture. The primary structure refers to the amino acid sequence of the protein. The secondary structure is the local spatial conformation of the protein backbone due to repetitive phi and psi angles. Examples of secondary structure include
2. Theory

the α-helix, β-sheet, and β-turn. Finally, the tertiary structure is the complete arrangement in three-dimensional space of all atoms in the protein including side chains, disulfide bridges, and H-bonds.

Predicting the secondary structure from the amino acid sequence has been widely practiced. One popular means to such predictions is the use of empirical statistical methods that apply parameters obtained from the analysis of known sequences and structures. Two of the most popular empirical methods, the Chou-Fasman method and the GOR II method, have been implemented in Homology and are briefly described in the following two sections.

The Chou-Fasman Method

The Chou-Fasman method of secondary structure prediction uses a set of rules based on parameters determined from the examination of 29 protein X-ray structures containing 4741 amino acid residues. The frequency with which a particular amino acid type is found in a particular secondary structure classification is correlated (Prevelige and Fasman 1989). A set of rules developed from these observations is consulted to predict the likelihood that a residue in a protein will exhibit one of several secondary structures in a given sequence. As described, these rules have a great deal of interdependence. The algorithm used in Homology follows these rules as closely as possible to assign residues to α-helices, β-sheets, or reverse turns.

The parameters required for the Chou-Fasman method are:

- \( P_\alpha \), the measure of an amino acid preference for α-helical conformations.
- \( P_\beta \), the measure of an amino acid preference for β-sheet formation.
- \( P_\tau \), the measure of an amino acid preference for being in a β-turn.

These probabilities are always averaged over four residues in the sequence. \( P_\tau \) is also accompanied by four other parameters that describe the positional frequency of the amino acid within the β-turn; these four positional parameters are denoted \( f_0, f_1, f_2, f_3 \) and \( f_2, f_3, f_0, f_1 \) to represent the four positions of the β-turn. For example, some amino acids have a profound positional preference; for example, proline is found 30 percent in position two and only 4 percent in
Superimposing structures based on sequence alignment

position 3. The rules followed for the Chou-Fasman method are as follows.

**Helices** A cluster of four helix forming residues out of six residues initiates a helix. PRO, ASP, and GLU may only occupy the first three positions at the N-terminal end of the helix region. HIS, LYS, and ARG may occupy only the last three positions on the C-terminal end of the helix. The nucleation site may have no more than one-third helix breakers.

The helix is then extended in both directions (unless limited by the occurrence of a PRO, ASP, GLU, HIS, LYS, or ARG as described above) until either a tetrapeptide breaker is found or one of the aforementioned residues is encountered. A tetrapeptide breaker consists of a tetrapeptide that both satisfies the breaker pattern described by Chou-Fasman and has an average $P_\alpha$ less than 1.0. Once the tetrapeptide breaker is found, adjacent residues within the breaker that are not helix breakers themselves may be added back to the helix.

$\langle P_\alpha \rangle$ for the entire helix must be $\geq 1.03$ and greater than $\langle P_\beta \rangle$. The number of helix formers must be at least half and the number of helix breakers must be less than a third. Finally, the helix region must be at least six residues in length.

**Strands** A cluster of three β-sheet formers out of five residues will initiate a β-sheet. The β-sheet is then propagated in both directions until terminated by a tetrapeptide breaker. Again, a tetrapeptide breaker has a $\langle P_\beta \rangle < 1.00$ and satisfies a breaker pattern.

A charged residue (ASP, GLU, HIS, LYS, ARG, or PRO) will also break the β-sheet propagation unless it occurs in a tetrapeptide where $\langle P_\beta \rangle > \langle P_\alpha \rangle$ and $\langle P_\beta \rangle > 1.0$.

Once the tetrapeptide breaker is found, adjacent residues within the breaker that are not β-sheet breakers themselves may be reincorporated into the region, unless this causes a charged residue violation as described above.

$\langle P_\beta \rangle$ for the entire region must be at least 1.05 and greater than $\langle P_\alpha \rangle$. The number of β-sheet formers must be at least one-half and the number of β-sheet breakers must be less than one-third. Finally, the strand region must be at least three residues in length.
2. Theory

**Turns** The probability of a turn at residue $i$ is calculated from $P_t = f_i \times f_{i+1} \times f_{i+2} \times f_{i+3}$. Tetra-peptides with $P_t > 7.5 \times 10^{-5}$ as well as $\langle P_t \rangle > 1.00$ and $\langle P_\alpha \rangle < \langle P_\beta \rangle > P_\beta$ are predicted as $\beta$-turns. Overlapping turns are resolved by taking the turn with the largest turn frequency product.

**Overlapping Regions** Regions that may be either helical or strand may be resolved by following rules outlined by Chou and Fasman. However, these rules do not lend themselves to an objective algorithmic approach and therefore, were not implemented.

Data from the algorithm is presented in two ways. First, the average $P_\alpha, P_\beta$, and $P_t$ values (averaged over four residues) are plotted in a graph. It is possible to examine the plots on the graph to judge which secondary structure type is most likely for any particular residue. The program is also capable of following the rules described to produce subsets summarizing assignments for each residue to a particular conformation. Because of the possible ambiguity of alpha helical and $\beta$-sheet regions mentioned, there may be some overlap in the assignments.

**The GOR II Method**

The second empirical statistical method implemented in *Homology* is GOR II (Garnier and Robson 1989). The GOR II method uses statistics that were calculated from the secondary structure assignments of 75 proteins and peptides done by Kabsch and Sander (1983). The database contains 12,757 residues that were classified into four structure types: 29.7% as $\alpha$-helix, 19.7% as $\beta$-sheet, 12.2% as $\beta$-turn, and 38.3% as random coil.

For each residue in the database, an information value reflecting its secondary structure type and the types of its neighboring amino acids 8 residues on either side was collected. The information is presented in four tables, one each for the conformation types helix, strand, turn, and coil. In each table, values for the 20 amino acids are listed in positions $-8m8$. For $m = 0$, the information value is the intrinsic tendency of an amino acid to be in a certain conformation. For $m \neq 0$, the information value is how a certain amino acid a particular distance away might influence this tendency.
Superimposing structures based on sequence alignment

From the tabulated data, the algorithm calculates the information value for each amino acid at position \( j \), summing the information values for the neighboring amino acid types from \( m = -8 \) to \( 8 \) according to:

\[
\sum_{m} I(S_j = X; X; R_{j+m}) \tag{Eq. 18}
\]

where \( I(S_j = X; X; R_{j+m}) \) is the information value from the table for conformation \( X \) of the influence of amino acid type \( R \) at position \( j+m \) on residue \( S \) at position \( j \). This sum is calculated for each of the four secondary structure types. Residue \( S_j \) is predicted to be in the conformational state for which the value of the four summations is highest.

As with the Chou-Fasman method, the data calculated here is presented in two ways. Graphs are created depicting the information values for each residue in all the possible conformational states. Thus, there are four plots in each graph, one each for helix (colored green), strand (red), turn (magenta), and coil (cyan). Subsets are also created for each secondary structure type.

The third method implemented in Homologoy is DSC, which is developed by Ross D. King.

**Hydrophobicity Profiles**

It is often useful to examine the relative hydrophobicity or hydrophilicity values of the amino acids in a protein sequence. Since hydrophobic residues tend to be more buried in the interior of the molecule and hydrophilic residues are more exposed to solvent, a profile of these values can indicate the overall folding pattern. Specifically, a long stretch of hydrophobic residues can indicate a buried \( \beta \)-strand, and a short spike of hydrophilicity can indicate a turn.

A graph can be drawn to visualize the hydrophobicity of a protein along its sequence. If the data is presented without modification, a very rough plot is the result, and overall trends cannot be discerned. Since proteins do not fold based on the tendencies of single residues, but rather on the characteristics of larger groups of residues, a running average of the hydrophobicity values of the residues is more appropriate.
2. Theory

In this implementation, a moving window averaging technique is
used to smooth the data. The width of the window is user-con-
trolled. The value plotted is given by:

\[
H_i = \frac{(1 + (2 - 1))}{2} \sum_{m = i - (w - 1)/2}^{i + (w - 1)} H_m,
\]

Eq. 19

where \(H_i\) is the plotted hydrophobicity value at residue \(i\) and \(w\) is
the window width. The window width must be a positive odd
integer. Also, at the end points of a chain, the averaging is done
only for the available points on the terminal side of the \(i^{th}\) residue.

Each of the hydrophobicity scales available within Homology is
divided into three hydrophobicity levels: hydrophobic, neutral,
and hydrophilic. This division was done by examining the order
of the amino acids and their values. Where an appropriate break in
the values was found between two amino acids that are generally
thought to be in different classes, the average of the values of those
amino acids was taken to be the hydrophobic or hydrophilic
threshold for that scale. Thus, for the Engelman and Steitz scale,
the value for ALA (hydrophobic) is -1.6 and that of THR (neutral)
is -1.2. Therefore, the hydrophobic level for that scale is -1.4. Simi-
larly, since the value for TYR (neutral) is 0.70 and that of HIS
(hydrophilic) is 3.00, the hydrophilic level is 1.85. The threshold
levels provided are somewhat arbitrary and are only meant to be
suggested defaults. You may choose other values after examining
the scales themselves.

The scales used in this implementation are listed in Appendix F,
Hydrophobicity Scale Values.

These levels are used to assign a particular amino acid in a protein
to one of the three categories. A residue with a value between the
hydrophobic level and hydrophilic level is classified as neutral.
Depending on the scale in question, those residues above both
thresholds or below both thresholds are hydrophobic or hydro-
philic. Thus, with the Janin scale, as the numeric value increases,
the hydrophobicity of the amino acid increases, and the hydropho-
bic level is above the hydrophilic level. Therefore, those residues
above the hydrophobic level are hydrophobic, and those below the hydrophilic level are hydrophilic. For the Engelman-Steitz scale, the opposite is true. Here, amino acids with a higher numeric value are regarded as hydrophilic, and the hydrophilic level is above the hydrophobic level. Therefore, residues with a value above the hydrophilic level are classified as hydrophilic, and those below the hydrophobic level are hydrophobic.

When specified by the user, subsets are created for each of the three levels of hydrophobicity. Residues are classified based on the threshold specific to the scale chosen and the width of the averaging window. Stretches of residues that are contiguous in sequence that are of the same type are grouped together in a single subset numbered as $protein\_name$\$scale\_id\$type n, where the type is PHOBIC, PHILIC, or NEUTRAL, and n is the number of the segment starting with 1 at the N-terminus. Subsets of the form $protein\_name$\$scale\_id\$type are also created that are the union of all residues of that type. The subsets can be used as input to many other commands, such as Color Molecule or Label Molecule.

**Solvent Accessible Surfaces**

**Definition of Solvent Accessible Surface Area**

The solvent accessible surface (SAS) of a molecule is defined as the surface area of the molecule exposed to solvent. More specifically, our definition follows that proposed by Lee and Richards (1971), which is described by the center of a solvent sphere (called a probe) rolled over the surface of the molecule. The use of the center of the probe means that the surface area is directly proportional to the number of solvent molecules that could fit on the surface. The Lee and Richards SAS model has been studied for its value as a term in a molecular mechanics forcefield, and has been shown to predict thermodynamic properties with reasonable accuracy (Ooi et al., 1987; Still et al., 1990).

It should be noted that an observation has recently been made by Jackson and Sternberg (1993), that the molecular surface (that is, the contact surface between probe and molecule rather than the surface at the probe center) correlates better with certain thermo-
2. Theory

dynamic properties of the molecule than the classical SAS. This alternative contact molecular surface area can be calculated using the Connolly (1983) surface algorithm present in the Insight II Molecule/Surface command; this method is discussed in the Insight II User Guide (Viewer Module Chapter, Molecule/Surface section).

Significance of Solvent Accessible Surface Area

A major reason for measuring surface area in proteins is to try to use the area as part of a free energy term in a forcefield to help maintain the proper fold of the protein during molecular dynamics simulations. The reason that this application seems feasible is that surface area is proportional to the free energy of hydration of a solute molecule (Eisenberg and McLachlan, 1986; Ooi et al, 1987; Wesson and Eisenberg, 1992), and this solvation effect is a major stabilizing force in protein structure (for review see Dill, 1990). In support of the free energy being proportional to surface area, it has been shown that when the free energy of hydration is set to be proportional to surface area, accurate calculations of enthalpy and heat capacity of hydration can be made (Ooi et al, 1987; Vila et al, 1991). These properties are important components of the total free energy of the system. While in principal it is true that the entire solvent effect cannot be simulated by surface area terms alone (since electrostatic terms are not proportional to surface area (Still, et al. 1990)), surface area still promises to play a key role in the ongoing development of continuum solvation models (that is, not using explicit solvent) for molecular dynamics simulations.

The potential importance of surface area in protein structural dynamics calculations is enormous. The preferred method for simulating protein dynamics today is to use explicit water during the simulation, but this is computationally very expensive (20 ps for a protein of 300 residues and 160 water molecules takes about 2 hours on an SGI R10K machine). The goal of structural biologists is to obtain highly significant results with forcefields and simulations that can be done in a reasonable amount of time. Since it takes proteins about a second to fold in nature, a true simulation of this event would require $10^8$ years on a Cray X-MP—clearly an impossibility. Some other method—faster than explicit solvent and better than a dielectric term—is needed to break through this
Solvent Accessible Surfaces

computational barrier. Potential energy terms proportional to surface area are an attractive component of the solution to this problem.

Solvation Module versus ProStat

The Solvation module in the Insight II software relates the solvent accessible surface area to an energy term in the forcefield, and provides several standard hydrophobic models to choose from (for example, Wesson and Eisenberg (1992), Still et al. (1990), Ooi et al. (1987), and Vila et al. (1991)). In addition, Solvation uses the Poisson-Boltzmann equations to calculate charge effects. Because of the complex nature of the computations, they are not usually applied at each stage of dynamics, but are applied after the simulation on the final structure, following the direction taken by Vila et al. (1991).

The aims of ProStat are limited to comparison of the surface area itself (not as a free energy term), which is then applied as a quantitative or statistical analysis performed on the final structures. ProStat can calculate the relative surface area, and the polar/apolar areas. The ProStat analysis is tailored to compare several proteins with each other, or several conformers of the same protein, or many conformers from a dynamics trajectory. As such, ProStat assumes a direct linear relationship between surface area and solvation free energy (as in Chothia, 1974), and uses the surface area values for the comparison of solvation effects.

Solvent Accessible Surface Area for Protein Structure Validation

Given that solvent accessible surface area is important in protein structure and stability, how can we use surface area today to help validate protein structures? It seems reasonable a priori to use surface area (which is proportional to solvation free energy) for discriminating the native conformation of a protein among a collection of near-native conformations, which are minimized in the absence of solvent. This structure validation is the ultimate goal of the surface area analysis in ProStat.
2. Theory

A detailed and comprehensive study was undertaken by Vila et al. (1991) to compare the solvation free energy values of structures minimized without solvation terms, using 21 different combinations and permutations of published and new methodology. They showed that 11 of the methods they tested were able to discriminate between the native and near-native structures of bovine pancreatic trypsin inhibitor (their Table VI). This result has profound implications for structure validation (assuming it is generally applicable to other proteins). Unfortunately, however, they also concluded that when the solvation terms were added to the force-field energy terms, the resulting change was not large enough to overcome the large negative correlation of the forcefield energy with correct structures; that is, the solvation free energies were not large enough to overcome the inaccuracy of the forcefield calculation—inaccuracies which presumably arise due to the absence of explicit water in the calculation. It may be that better solvation models (perhaps including charge effects) will correct this problem in the future; for now, it seems best to compute the solvation term after molecular dynamics and minimization is finished, and to use the value as an independent criteria (independent of force-field energy) to help select the best structure from a number of similar structures.

Rules for Protein Validation

Unfortunately there are no absolute rules to apply for using surface area for protein validation; fortunately there are a number of suggestions that have been put forward.

1. The first suggestion is that the total accessible surface area in water is proportional to the hydrophobic free energy of a folded protein. The approximation given in Chothia (1974) is that the hydrophobic free energy of a protein residue is 24 calories per square angstrom of accessible surface area. (This free energy value was actually based on the residue side chain only; to derive the side chain values, the residue hydrophobicity values and the residue accessible surface area values were reduced by subtracting the equivalent values for glycine.) In support of the relationship between energy and surface area, Novotny et al. (1984) showed that incorrectly folded proteins have significantly larger surface areas than correctly folded proteins. Fur-
Solvent Accessible Surfaces

thermore, studies using the total accessible surface area as a term in dynamics simulations worked better than other parameter sets involving atomic solvation parameters (von Freyberg et al., 1993).

2. The second suggestion is to analyze the distribution of polar and apolar atoms between the interior core and the surface. In general, the surface of a protein is polar, and the internal core is apolar. The distribution of atoms between the two locations correlates with (and is responsible for) this observation (note that polar or charged residues can reside in the core if they are neutralized by a neighbor or by internal water). In support of the importance of polarity, Novotny et al. (1984) have noted that there is a greater exposure of apolar residues in mis-folded protein structures. Their observation suggests the use of a direct comparison between the polar surface area of multiple conformers.
2. Theory
In addition to the basic operations you use with Insight II, the use of Homology requires an understanding of the sequence window. It displays an alignment of amino acid sequences and allows you to examine and compare them and the corresponding three-dimensional structures to find areas of similarity.

**Sequence Window**

The sequence window first appears when you use the Sequences/Extract or Sequences/Get commands. It disappears when there are no more sequences to display, either because the sequences were removed with the Sequences/Delete command or the molecule was deleted with the Object/Delete command. You can manage the window like any other Unix window, in that you can move it anywhere on the screen, resize it, or iconify it. You may not, however, close it except by deleting the sequences displayed in it. The sequence window is composed of the sequence display and the controls.

**Sequence Display**

The sequence display consists of:

- sequence rows
- sequence gaps
- sequence blocks
- residue counters
- protein name label
- tick marks
3. Implementation

A sequence row is a single row of upper- or lowercase letters that correspond to single-letter codes of a protein’s amino acid sequence. The number of letters is adjusted dynamically to nearly span the width of the sequence window. Each time you Extract or Get a sequence, a new sequence row appears at the bottom of the screen.

A sequence gap is a dash (“.”) in a sequence row that shows where there has been a deletion of one or more amino acids in one sequence relative to another. Sequence gaps are used to vertically align residues thought to be equivalent among the displayed protein sequences.

A sequence block is a group of sequence rows that are aligned to show relationships between the corresponding residues. If one or more sequences is too long to fit on a single row, then there will be more than one sequence block and that sequence will wrap from block to block.

A residue counter appears at each end of every sequence row. As the sequence rows are shifted back and forth through the sequence window, the numbers shown in the residue counters change to reflect the identities of the residues at the left and right ends of the row, that is, the N- and C-terminal ends of the polypeptide segment. If the row does not extend all the way to the end of the window because of the nature of the alignment, the identity of the end-most residue is shown.

A protein name label appears at the left side of each sequence row, just before the residue counter. Only the first six characters of the protein name appear.

Tick marks appear at the bottom of each sequence block and allow you to more easily determine the identifying name (generally the residue number) of a particular residue. The tick marks appear in three different sizes, the smallest represents a distance of one residue along the peptide chain, the next largest five, and the largest tick mark represents a distance of ten residues. Thus, you can more easily “count” over from either edge of the row and determine the corresponding number for the residue in which you are interested.

The sequence rows, residue counters, protein name label, and tick marks comprise a display called the sequence display. The sequence display allows you to:
Sequence Window

- display sequence information,
- align sequences to determine regions of structural and sequence similarity of peptide segments, and
- identify regions for coordinate transfer between reference and model proteins.

Controls

The sequence window controls allow you to alter the display characteristics of the sequence alignment and change the actions of the mouse. The controls consist of:

- the scroll bars
- the Mode selector
- the Tick Marks toggle
- the Font Size button
- the Help button
- the Color by selector

The scroll bars control what is viewed in the window. When the entire sequence alignment is larger than can be displayed at once in the window, the vertical scroll bar can move the sequence display up and down in the viewing area. The length and position of the vertical scroll bar reflect the amount of the alignment that is currently above and below the viewing area. The horizontal scroll bar moves all the sequence rows left and right. As they move, the residue counters change to reflect the current leftmost and rightmost residues in each sequence block. As residues move off the left or right ends, they wrap to the previous or next block, respectively.

The Mode selector changes the action of the mouse. When set to Seq, the mouse moves sequences left or right and adds, deletes, and moves sequence gaps. When it is set to Box, it creates, moves, resizes, and deletes sequence boxes (discussed later).

The Tick Marks toggle turns the display of tick marks on and off. When tick marks are off, less vertical space is used for each sequence block, and more of the alignment can be viewed with a given window size.
3. Implementation

When the *Font Size* button is pressed, a parameter block appears that lets you control the size of the letters used to depict the sequence. Point sizes in the range of 6 to 48 are available. The size is most easily changed by using the slider.

The *Help* button activates the Insight II xhelp program to display a window containing a description of the sequence window operations.

Finally, the *Color by* selector controls how the residue letters will be colored. When set to **C-alpha**, each letter is the same color as the alpha carbon of the corresponding residue in the three-dimensional display. This is the usual setting. When **p-value** is selected, residues in m-blocks are colored a various shades of magenta that encode the statistical significance of their sequence similarity. When set to **Contents**, all residues in m-blocks are colored cyan. The concepts of m-blocks and the boxes that enclose them (m-boxes) are discussed later in this section and in Chapter 5.

![Figure 19. Homology Amino Acid Sequences Window](image)

90 Homology
Sequence Boxes

A sequence box is drawn around the letters in a sequence display to indicate areas where a structural or sequence alignment is proposed, or where the sequences or 3D structures of two peptide segments are being compared.

You can create a sequence box with the Initialize command in the Boxes pulldown or with the mouse when the Mode is set to Box. After a box is created, you can change its horizontal size and position using the mouse. The vertical size of the box is determined at the time it is initialized and cannot be changed.

Figure 20. Sequence Box

A sequence box created in this way defines a relationship between exactly two sequence segments of equal length from two different proteins. If there are more than two sequences in the sequence window, a box can span more than two sequence rows vertically, but it still defines a relationship between only two sequence segments: those in the top and bottom rows within the box. There are other types of boxes used in the Homology program that can define relationships among more than two sequence segments (summary boxes and m-boxes). These are created by alignment algorithms, not by direct click-and-drag operations with the mouse, and they are discussed in detail in later chapters.

Sequence boxes are sometimes used as input to commands that operate on two segments of two proteins. For example, the command that copies 3D coordinates from part of a reference protein...
3. Implementation

to part of the model protein requires as input a box that includes a segment of the reference and a segment of the model. To specify a box for this command, you pick a residue within the box using the left mouse button, and the box turns yellow to indicate that it is selected.

It is not unusual to have several boxes that partially or completely overlap one another. If you try to select a specific box that overlaps others, you may find that one of the other overlapping boxes turns yellow instead of the one you wanted. In this case you can select the desired box by clicking the mouse button repeatedly. With each click, a different one of the overlapping boxes gets selected and turns yellow. In this way you can easily cycle through all overlapping boxes to examine them individually and to select any of them for other operations.

Sequence boxes can be frozen or unfrozen. A frozen box cannot change size or position, nor can the residues inside it be changed. If sequences that are contained in the box are moved, then it moves along so as not to have its residue contents changed.

The Homology program uses several different types of boxes, each of which has a particular color associated with it.

<table>
<thead>
<tr>
<th>Table 3. Sequence Box Color Coding</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Box Type</strong></td>
<td><strong>Color</strong></td>
</tr>
<tr>
<td>Regular sequence boxes</td>
<td>cyan</td>
</tr>
<tr>
<td>Frozen sequence boxes</td>
<td>red</td>
</tr>
<tr>
<td>M-boxes (used in Multiple Alignment)</td>
<td>medium blue</td>
</tr>
<tr>
<td>Frozen m-boxes</td>
<td>magenta</td>
</tr>
<tr>
<td>Active sequence box (the box that you are currently scrolling)</td>
<td>green</td>
</tr>
<tr>
<td>Summary boxes</td>
<td>white</td>
</tr>
<tr>
<td>Selected box (the box you pick during execution of a command such as Freeze Boxes)</td>
<td>yellow</td>
</tr>
<tr>
<td>Prompt boxes and suggestion boxes (offered as visual aids during the execution of commands)</td>
<td>yellow</td>
</tr>
</tbody>
</table>
These boxes may overlap partially or completely. Therefore, a hierarchical order has been established so that the most important, i.e., most relevant, box is always visible. The order of the hierarchy is described below:

- selected box
- prompt box
- suggestion box
- active box
- summary box
- frozen m-box
- non-frozen m-box
- frozen box
- non-frozen box

Because the selected box is at the top of the hierarchy, when you cycle through a group of boxes (selecting each of the overlapping boxes), you are guaranteed to see each box in its entirety when it is selected.

The following box operations can be accomplished through either mouse movements or menu commands: initialization, freezing, unfreezing, and deletion.

---

**Sequence Gaps**

A *sequence gap* is a dash (-) you insert into a sequence row to align it with another row. This is done in regions where there has been an amino acid insertion in one of the proteins.

Sequence gaps are created in two ways: manually, by using the mouse in Seq mode; and automatically, using the Pairwise Sequence, Multiple Sequence, or Structure commands in the Alignment pulldown. Note also that a hidden command, Scroll Gaps, may be activated when Initialize Boxes is used if the selected residues are not already vertically aligned.
3. Implementation

Manipulating the Sequence Display

Each sequence row shows a representation of a protein molecule’s covalent configuration in the horizontal direction. Letters that are directly above or below one another are said to be aligned, and they are regarded as representing equivalent residues in the corresponding three-dimensional structures. The order of the letters in a row cannot be changed, nor can a row be split into two or two rows combined. But the relative alignment of the sequences can be easily controlled. This is best done using the mouse, but some operations can also be done using commands found in the menus.

For most operations, you use the mouse to pick and then drag a residue or gap symbol. Dragging means that you press and hold down a mouse button while moving the mouse. Instead of dragging, you can instead click briefly on the sequence display with the appropriate mouse button, release the button, move to the final position of the dragging operation, and click the same button while holding down the <Shift> key. This is particularly useful if you need to use the scroll bars to move to a new area of the sequence display, something you cannot do while dragging.

Scrolling Modes

Two main components of the sequence display can be manipulated. They are the sequences themselves and the sequence boxes. The latter highlight regions where the sequences or structures are thought to be equivalent or are being measured for similarity. Which is currently being manipulated is controlled by the Mode, which may be either Seq or Box. You can change the Mode with the Mode selector at the bottom of the sequence window or by
Manipulating the Sequence Display

holding down the <Ctrl> key and clicking with the middle mouse button anywhere in the black area of the window.

These two modes determine which of three possible things will be affected:

♦ The position and relative alignment of the sequence rows
♦ The position and lateral size of the sequence boxes
♦ The insertion, deletion, and position of sequence gaps

Seq Mode

When the Mode selector is in the Seq position, the mouse controls the movement of sequence rows and sequence gaps. All three mouse buttons are used, but only one button is used at a time.

Moving sequences  To move a sequence, you place the cursor over a residue, and press the middle mouse button and hold it down. Dragging the mouse left or right moves one entire sequence row in that direction. As a residue in the sequence row reaches the end of the block, it wraps to the previous or next block. You can also move a sequence an entire block at a time by dragging the mouse up or down. If unfrozen sequence boxes exist that are associated with the sequence row being moved, the sequence simply moves through the boxes while they remain stationary in the display. You will be unable to continue moving a sequence row if movement of the sequence row would result in a gap being drawn into any box or a sequence being moved so far that a box would go beyond the row’s N- or C-terminus.

Manipulating gaps  To create a sequence gap, place the cursor over a residue and pick and drag with the left or right mouse button. If the left button is used, and the movement is to the left, a gap is created to the left of the picked residue. If the right button is used to pick and drag to the right, then the gap will be to the right of the picked residue. You can easily close the gap by moving the mouse in the opposite direction, that is, dragging the mouse to the right while holding the left button closes a gap, as does dragging to the left while holding the right button.

Gaps can be moved by picking a gap character (−) with the middle button and dragging to the left or right. If the movement continues
3. Implementation

to the N- or C-terminus of the sequence, then the gap passes off the end, deleting the gap. If a gap character is picked, and the left or right button is used to drag the mouse, then the gap is split into two. That is, residue letters move into the gap where you pick, moving half the gap in the direction of the mouse movement.

Box Mode

 Initializing boxes  You can initialize, or create, boxes in either of two ways. The Initialize command, found in the Boxes pulldown, takes two residue identifiers as input parameters. These can be entered by clicking on the residue letters in the sequence window with the left mouse button, or clicking on any atom in the residues in the three-dimensional model. Either way, a yellow prompt box is drawn around both of the residues in the sequence display. If the residues are not already vertically aligned, a yellow suggestion box is drawn in the shorter of the two sequence rows between the two picked residues, indicating that a gap must be inserted somewhere there. The hidden Scroll Gaps command activates, and you must pick a residue insertion point for the gap, again with the left mouse button. The green (active) box that is created using this method is only one residue wide and usually is expanded to be the correct size, as described below.

Although boxes can be created with the click-and-drag or click-and-<Shift>-click mouse movements, the left mouse button is not available for box creation if there is an open parameter block. In this case, you must either close the parameter block or use the Boxes/Initialize command to create the box.

Sequence boxes can also be created with the mouse. Simply press and hold the left mouse button over the residue that will be one of the four corners of the box and drag towards the residue that will be in the diagonally opposite corner. As you drag, a yellow prompt box follows your movements exactly. If you go from one sequence block to another adjacent one, the prompt box wraps, always spanning the appropriate sequence rows. When you release the left button, a green (active) box appears. If, by your mouse movements, you had placed the prompt box in a place where a box could not normally go, such as enclosing a gap symbol, the program would make as large a box as possible that started at the initially picked residue and went in the requested direction.
Manipulating the Sequence Display

**Moving and resizing boxes**  Both these operations can only be done using the mouse. The two are similar in that the box to be changed is first selected and then altered. Moving uses the middle mouse button, and resizing uses the right mouse button. Select the box by pressing and holding the mouse button when the cursor is over any residue that is inside the box. Since any particular residue may be in several overlapping boxes simultaneously, be sure to select the correct box. Repeatedly pressing the middle or right mouse button makes each overlapping box turn yellow in turn, indicating which box is selected and will be altered once the mouse is dragged.

To move a box, select it with the middle mouse button and drag to the left or right. As you begin to move it, the box turns from yellow (selected) to green (active). If the box approaches the end of a sequence block, it wraps to the next, making it appear split. You can move the box to a different sequence block by moving it up or down. However, the box cannot proceed past a sequence gap. If you want to move a box from one side of a gap to the other, you need to delete the box and reinitialize it on the other side.

To resize a box, select it with the right mouse button, and drag to the left or right. When the cursor reaches a residue just inside the left or right edge of the box, the box turns green (active), and that edge moves with the mouse. Moving the left edge to the left or the right edge to the right expands the box, while moving in the opposite direction shrinks it. Moving up or down does not make the box taller or shorter, since the sequence rows involved with the box were set when it was made. Instead, moving up or down makes the box larger, wrapping it to the next sequence block.

**Freezing and unfreezing boxes**  When a section of an alignment has been decided upon, it is necessary to freeze the box in order to prevent inadvertent disruption of the alignment. If there subsequently needs to be a change in a frozen region, then the box must be temporarily unfrozen so the change can be made. Boxes can be frozen and unfrozen either with menu commands or with the mouse. When you select either of the menu commands *Freeze* or *UnFreeze* in the *Boxes* pulldown, you need to choose a box by picking any residue inside it with the mouse. As with all menu commands, the picking is done with the left mouse button. The correct box is selected by repeatedly picking the residue until the
3. Implementation

box is yellow. Selecting Execute completes the operation. Frozen boxes become red, and newly unfrozen boxes become cyan.

To use the mouse to freeze or unfreeze a box, select it by pressing and holding with the middle or right mouse buttons. Without moving the mouse at all, click the left mouse button. The box changes its state from unfrozen to frozen, or vice versa. To do the opposite action, repeat the same procedure. The box toggles back and forth between frozen (red) and unfrozen (cyan) states, but it must be reselected each time.

Deleting boxes  Boxes can be deleted either using the menu or the mouse. When you select the Boxes/Delete command, select the correct box with the left mouse button and Execute. To use the mouse to delete a box, select the box with the middle or right mouse buttons. Without moving the mouse, press and hold the <Ctrl> key and click the left mouse button. A frozen box must first be unfrozen, as described above, before it can be deleted.

Summary boxes are usually created as part of the manual determination of SCRs (see Manual Determination of SCRs). Since this process may involve significant time and effort, the Homology program includes safeguards to prevent the accidental deletion of summary boxes. Summary boxes cannot be deleted by the methods just described. Instead, you must execute the Boxes/Summarize command with the Summarize Operation parameter set to Delete.
This chapter describes the functionality of each pulldown and its associated commands.

As with Insight II, Homology commands are organized in a hierarchy. At the top of the organization are the modules, next are the pulldowns, and at the bottom of the hierarchy are the commands.

**Modules**

Homology is one of several optional modules available in addition to Insight II’s core modules. The modules typically included with Insight II are **Viewer**, **Biopolymer**, **Builder**, **Docking**, **Mopac**, **Discover**, and **Analysis**. Each module is comprised of a group of pulldowns with related commands.

**Pulldowns**

Pulldowns are located below modules in the command hierarchy. When a particular module is selected, the lower menu bar displays the pulldowns associated with that module. The **Sequences** pulldown, for example, is a set of all commands having to do with sequence creation, deletion, display, and manipulation.

The Homology module contains several pulldowns that are available on the lower menu bar:

- **Sequences pulldown**.
- **Boxes pulldown**.
- **Loops pulldown**.
4. Command summary

Residue pulldown.
Databases pulldown.
Background Job pulldown.
Alignment pulldown.
By Residue pulldown
Refine pulldown.
ProStat pulldown.
Consensus pulldown (optional).
Profiles 3D pulldown (optional).
Modeler pulldown (optional).
Seqfold pulldown (optional).

Commands

Commands are located below pulldowns in the command hierarchy. For example, the Boxes pulldown contains the commands: Initialize, Delete, Freeze, Unfreeze, Summarize.

The remainder of this section, organized by pulldown, briefly describes the functionality of each pulldown and the commands associated with it.

Sequences pulldown

The Sequences pulldown includes all commands having to do with creation, deletion, display, and manipulation of sequences.

The commands in this pulldown are Get, Put, Extract, Copy, Delete, AssignCoords, and Color.

Get/

The Get command reads files containing single-letter amino acid codes or alignment files that have the format of the Put command. If the file contains an alignment, several protein sequence rows are
Sequences pulldown

read in with their protein name labels and gaps inserted where indicated in the file by gap symbols ("-".). The Get command displays the sequences (in lowercase letters) to indicate that no coordinates are known for the protein(s).

Put

The Put command writes output to files of either single sequence rows or full alignments. Note that the default file extension for single sequence rows is .seq and for alignments the default extension is .align.

Extract

The Extract command displays rows of amino acid sequences (in uppercase letters) from a previously loaded protein molecule. This command also adds hydrogens to all proteins and caps them in a standard way, so that all proteins are represented identically. Finally, this command displays only the alpha-carbon trace of the designated proteins.

Copy

The Copy command copies the amino acid sequence row and subset definitions from one protein to another. Both proteins must have identical sequences. The protein from which the information is copied must have a sequence displayed, and the protein to which it is copied must not have a sequence displayed. This command is useful when you are doing successive Discover jobs.

Delete

The Delete command removes the sequence display for a molecule, or when used with a sequence-only row, deletes the associated data object.

AssignCoords

When an area of correspondence has been defined between a reference protein and the unknown sequence, the coordinates are transferred with the AssignCoords command. This command copies coordinates from the reference protein to the model protein when the amino acid types match at corresponding positions. Where they don’t match, automatic side-chain replacement is done. This maintains the residue types of the model protein, but also preserves the orientation of the reference protein.

Color

The Color command colors all the atoms of all the residues in a specified sequence box.

Convert_Format

The Convert_Format reads in sequence or alignment file and converts it to a format selected by user.
4. Command summary

**Boxes pulldown**

The Boxes pulldown contains commands that manipulate the sequence boxes. The sequence boxes indicate contiguous peptide segments where pairs of sequences or the corresponding 3D structures are being compared.

The six commands under this pulldown are **Initialize, Delete, Freeze, UnFreeze, and Summarize.**

- **Initialize**
  - The Initialize command creates sequence boxes around residues in any two rows of the sequence display.

- **Delete**
  - The Delete command removes one or all sequence boxes from the display.

- **Freeze**
  - The Freeze command is used to prevent further movement of a box or any sequence through it once the position and lateral size of a particular box is found to be optimum, i.e., when it accurately indicates the location of a structurally conserved region (SCR). Either one or all sequence boxes can be frozen at once.

- **UnFreeze**
  - The UnFreeze command re-enables the movement of a box, or any sequence through it, after it has been frozen. Either one or all sequence boxes can be unfrozen at once.

- **Summarize**
  - When several SCRs have been defined for two or more reference proteins, the Summarize command finds the maximum number of overlapping residues among all the proteins in each region to define a set of SCRs thought to be valid for any member of the homologous family, including the unknown protein.

  Each SCR is indicated by a summary box drawn around the residues in the sequence display. Summary boxes are white and include all sequence rows defined as having SCRs. They are automatically frozen and cannot be unfrozen. This command can also be used to delete all summary boxes as well as create them. An enumerated parameter in the command chooses between the two modes. When summary boxes are deleted, their associated subsets are not deleted. This feature is useful when a completed structural alignment must be modified extensively.
**Loops pulldown**

The **Loops** pulldown contains the commands to search the Brookhaven protein database for loops, generate loop structures, display loops, and assign coordinates to build variable regions (VRs) between SCRs. The commands included under this pulldown are **Search**, **Generate**, **Display**, and **AssignCoords**.

**Search**

The **Search** command examines a file containing a distance matrix of alpha-carbon atoms obtained from the Brookhaven Protein Databank. Pattern matching techniques are used to find similarities between the SCRs at either end of the loop, and peptide segments at either end of a loop in the Protein Data Bank.

**Generate**

The **Generate** command generates alternate loop regions using the random tweak method developed by the Levinthal group (Shenkin et al. 1987). Random generation of $\phi$ and $\psi$ followed by iterative adjustments under geometric constraints produces a group of loop structures which are near the energy minimum.

**Display**

**Display** displays any one of the ten best loops found based on a criterion of RMS deviations of the alpha-carbons in the flanking segments.

**AssignCoords**

**AssignCoords** copies the coordinates from the chosen loop to the variable region of the model protein. As with **Sequences/AssignCoords**, whenever the amino acid types do not match, automatic side-chain replacement is done. This maintains the residue types of the model protein, but preserves the orientation of the reference protein.

**Residue pulldown**

The **Residue** pulldown contains commands that operate on lists of possible conformations for specified side chains. The commands in this pulldown are: **Append**, **Repeat**, **Rename**, **Delete**, **Replace**, **Manual_Rotamer**, **Auto_Rotamer**, and **List**.
4. Command summary

Append
The **Residue/Append** command allows you to build polypeptides by sequentially adding residues to an existing chain or to create a new peptide containing a single residue. The geometry of the addition is controlled by choosing from a set of standard geometries or by specifying a new geometry using the \( \Phi \), \( \Psi \), and \( \Omega \) Angle parameters. The default set of residues is comprised of the 20 standard amino acids, their charged side chain derivatives, D Proline, and the standard capping groups.

Repeat
The **Residue/Repeat** command allows you to create a new peptide consisting of a series of identical subunits. The geometry of the final peptide can be controlled by specifying the dihedral angle between the subunits.

Rename
The **Residue/Rename** command allows you to change the names of residues (such as LYS or PHE). This is useful for modeling unnatural amino acids. The **Residues To Rename** parameter specifies the residues whose names will be changed. The residues are specified by the monomer/residue specification. The **New Residue Name** parameter specifies the names (such as LYS or PHE) to be used for the residues.

Delete
The **Residue/Delete** command allows you to remove a specified residue from its parent molecule. To delete multiple residues from the same molecule, you must reissue the command since wild-carding and the use of ranges is not acceptable. The residue specified must be present within the molecule indicated. If this is the last residue within a molecule, then the entire molecule is deleted.

Replace
The **Residue/Replace** command allows you to replace one residue of an existing protein with another from the set of defined residues known to Insight. The backbone atoms do not move, but the remaining atoms of the side chain of the new residue assume a configuration which is comparable with the original residue. The chirality of the residue being used for the replacement can also be controlled.

Manual_Rotamer
The **Residue/Manual_Rotamer** command places one of the rotamers appropriate for a particular amino acid type at the position of the specified residue. You can cycle through all the possible **Rotamer Choices** by repeating the command. If the **Evaluate_Energy** parameter is on, then a single-point nonbond energy calculation is performed, calculating the electrostatic and van der Waals interactions between the specified residue and the residues that are
within the specified distance. If the **Bump_Check** parameter is on, the newly placed side chain will be tested for steric overlap against the rest of the protein. You can reject all of the choices and return to the original side chain conformation by selecting **Original** as the **Rotamer Choice**.

**Auto_Rotamer**

The **Residue/Auto_Rotamer** command finds a locally optimal combination of rotamer choices for a given list of moving residues. A search is performed starting at the beginning of the list and proceeding down. For each residue, the rotamer choice that produces the lowest energy is retained, and the search continues with the next moving residue. The search stops when the energy has not changed, or when the maximum number of iterations has been exceeded.

**List**

The **Residue/List** command is used to list the coordinate and topology details for a specified residue. You may also list molecular mechanics information, such as partial charges and potential function atom types.

---

**Databases pulldown**

The **Databases** pulldown contains commands that allow you to control the input and output parameters for the search and the submission of the job. The commands under this pulldown are **Input**, **Output**, and **Run_FASTA**.

**Input**

The **Databases/Input** command specifies the input parameters for the database search procedure. It is divided into two separate sections. The first section specifies the probe sequence and the type and names of the databases to be searched. Parameters that are specific to the database type, protein or nucleic acid, are also set at that time. The second section has parameters for the search procedure itself. That is, how to calculate the match scores.

**Output**

The **Databases/Output** command specifies how to present the data produced by the FASTA program. The number of sequences reported can also be indicated here.

**Run_FASTA**

The **Databases/Run_FASTA** command submits the background FASTA job.
4. Command summary

**Motif_Search**

**Motif_Search** helps you to identify proteins with structural motifs that closely resemble a part or whole of a query protein.

The query protein can be read in any Insight II-readable format but is translated to the pdb format prior to the search. The database of proteins consists of the November 1999 release of the Protein Data Bank/Brookhaven database.

On output, a list of all possible hits is prepared in table format. Any hit can then be visualized on two levels. First, the matching secondary elements of the hit can be schematically shown as a collection of overlapping arrows. Secondly, the whole matching structure can be superimposed onto the query structure.

---

**Background Job pulldown**

The **Background_Job** pulldown allows you to set up background jobs to run concurrently or interactively with the Insight II program. You are given the choice of whether to send background jobs to a local or remote host. This pulldown is generic and is found in many Insight II modules that run background jobs. The **Background_Job** pulldown contains the following commands: **Setup_Bkgd_Job**, **Completion_Status**, **Kill_Bkgd_Job**, and **Control_Bkgd_Job**.

**Setup_Bkgd_Job**

The **Background_Job/Setup_Bkgd_Job** command allows you to set up the execution mode and select the host upon which to run a job. In addition, this command can be used to control the notification method for background job completion and cleanup options.

**Control_Bkgd_Job**

The **Background_Job/Control_Bkgd_Job** command allows you to coordinate running background jobs by detaching or attaching selected background jobs to the Insight II program. In addition, this command allows you to specify the interval for invoking a task specific to a particular background job for processing its output.

**Completion_Status**

The **Background_Job/Completion_Status** command allows you to monitor and evaluate the completion status of one or all of the background jobs. In addition, this command can be used to look up the meaning of a given return status code.
Alignment pulldown

The **Kill_Bkgd_Job** command is used to terminate execution of a background job that has been submitted during the current session.

Alignment pulldown

The commands in the Alignment pulldown perform calculations to compare the three-dimensional structures and sequences of two or more proteins. The purpose of this comparison is the determination of which parts of the proteins correspond. Those portions of the molecules that match are said to be conserved, and they can be used as a basis for homology model building. The Alignment pulldown includes three commands: **Pairwise_Sequence**, **Multiple_Sequence**, and **Structure**. There are manual and automatic versions of these commands.

**Load_Alignment**

The **Load_Alignment** command adjusts the alignment of sequences in the sequence window according to information read from an alignment file. It only reads in alignments for sequences already in the sequence window. Sequences in alignment files, but not in the sequence window will be ignored.

**Pairwise_Sequence**

The **Pairwise_Sequence** command compares the amino acid sequences of two proteins and either performs an automatic sequence alignment or sets conditions to do manual comparative scoring.

**Multiple_Sequence**

The **Multiple_Sequence** command simultaneously aligns up to ten amino acid sequences. It identifies regions over which the sequences are mutually related and estimates the statistical significance of those relationships. Those regions containing highly significant relationships are likely to contain structurally conserved regions.

**Structure**

The **Structure** command compares the three-dimensional structures of up to ten proteins and either automatically aligns their sequences based on structural similarity or sets conditions to do manual comparative scoring between the two segments enclosed in the currently active box. Whenever possible, conserved regions for a family of homologous proteins should be found structurally rather than with sequence similarities alone.
4. Command summary

**Color_Overlap**
The **Color_Overlap** command colors one of the sequences of the alignment (model) according to the alignment overlap with specified sequences (templates). It allows you to choose the model and templates and to specify colors for the overlap part and loop residues.

**Superimpose_Aln**
The **Superimpose_Aln** command superimposes two or more structures according to their sequence alignment.

It allows you to:
1. Select some or all of the structures in the sequence window for superimposition.
2. Select different criteria to create matched residue sets which are used to superimpose structures.
3. Draw monitors (a dashed green line) between matched residues of selected structures.
4. Force the automatic superimposition of selected structures following every manual sequence window operation.
5. Create an assembly of superimposed structures and create sequence boxes of matched residues after superimposition.

**Align123**
The **Align123** command calls the align123 program to align all sequences in the sequence window.

Align123 aligns multiple sequences using a progressive pairwise alignment algorithm based on sequence similarity and secondary structure matches. It first generates all possible pairwise alignments for a list of sequences and then builds the guide tree based on their pairwise sequence identity and aligns the sequences following the order of the guide tree.

If the secondary structure option is turned on, the secondary structure will be predicted using one of the following methods: Chou-Fasman, GOR_II, or DSC for sequences. For know structures, the Kabsch_Sander method will be used to calculated the secondary structure. Secondary structure for all proteins will be concatenated into one file (named as __jobname.ss) and used by the align123 program to score the alignment. If SS_Source is set to **Read_from_File**, then a user specified secondary structure file will be used as the secondary structure source. The file format of the secondary structure file is explained in the file format section (with “.ss” file extension; see Secondary structure file).
**By_Residue pulldown**

**LocalAlignment**

The LocalAlignment command performs a local alignment of two sequences chosen from the sequence window.

**Percent_Identity**

The Percent_Identity command calculates the percentage sequence identity between two sequences based on their alignment in the sequence window.

---

**By_Residue pulldown**

The By_Residue pulldown contains commands for data profiles and/or analyses. This pulldown includes two commands: Hydrophobicity and SecondaryPred.

**Hydrophobicity**

The Hydrophobicity command creates a hydrophobicity profile of a molecule, providing either a graphical representation or subsets that define hydrophobic, hydrophilic, and neutral regions. Several different hydrophobicity scales may be used. The data may be used “as-is” or it may be smoothed using a running-window averaging technique.

**SecondaryPred**

The SecondaryPred command creates a secondary structure profile of a molecule, providing either a graphical representation or subsets that define secondary structure regions.

**Color_Conservation**

The Color_Conservation command colors sequence window entries according to the residue conservation in the current alignment.

**Color_By_Structure**

The Color_By_Structure command allows you to color a protein object (all or one by one) according to its secondary structure as specified by SecStrSubset.

---

**Refine pulldown**

The Refine pulldown contains commands that take the Homology-built model and produce a structure that is physically and chemically reasonable. This is done largely through an interface with the Discover molecular mechanics program. This interface is made more convenient through the use of subsets that have been defined for SCRs, loops, mutated (replaced) residues, and splice
4. Command summary

points between peptide segments during the course of the model building procedure.

This pulldown contains the following five commands: EndRepair, GenericDis, SpliceRepair, Relax, and Explore.

EndRepair

The Refine/EndRepair command recognizes that coordinates are only assigned to the model protein in SCRs and the loops between them. Thus if these regions do not extend to the N- and/or C-terminal ends of the protein, no coordinates are assigned there. The Refine/EndRepair command takes coordinates from the standard amino acid library and assigns them to these remaining residues in an extended conformation.

GenericDis

The Refine/GenericDis command imposes distance restraints which may be useful when performing molecular dynamics calculations with the Refine/Explore command. If more flexibility is desired in the Discover calculations, you can go directly to the Discover module and set up jobs there using the subset definitions created by the Homology module.

SpliceRepair

The Refine/SpliceRepair command is used to repair cis peptide bonds, or long or short peptide bonds that might be introduced into the model. This can happen when coordinates are assigned to segments taken from different proteins. The repair is done by setting up a Discover job with an optional torsion forcing of the peptide bonds to 180°. Note that any number of splice points can be repaired in one Discover run.

Relax

The Refine/Relax command sets up an energy minimization calculation on any selected region(s) of the model protein, such as the N-terminal region or SCR side chains. Any combination of region types and specific segments can be relaxed at one time. Thus, segments adjacent in space can be minimized simultaneously, so that intertwined side chains can be relaxed together. Optional tethering and choice of minimization algorithms are provided.

Explore

The Refine/Explore command sets up molecular dynamics runs for variable regions. You can choose which single loop is unconstrained during the simulation, and whether or not tethering should be employed. Minimization is performed on intermediate structures along the dynamics trajectory with a user-selectable algorithm. History and archive files are both created automatically.
Consensus pulldown

This is an optional pulldown. Please see the Consensus manual for more information.

Profiles_3D pulldown

This is an optional pulldown. Please see the Profiles-3D manual for more information.

ProStat pulldown

The ProStat pulldown contains commands for the analysis of protein structure. The commands include Struct_Check, Residue_Dihedral, SecondaryClassify, and Access_Surf.

Struct_Check

The Struct_Check command allows you to check protein-specific bond lengths, angles, and torsions in a protein 3D structure against the corresponding values in a knowledge base derived from accurate small molecule crystallographic studies. These data can help you to highlight erroneous structural features of the 3D structure. These may then be listed to the textport or tabulated using a per-residue Spreadsheet table (this can be automatically created using the Prostat/Struct_Check command). The numerical property values in the table make use of the graphing capabilities built into the Spreadsheet window to create 2D and 3D graphs for data visualization. A spreadsheet of monomer properties may be used to create colored, variable width molecular ribbon diagrams with the Molecule/Ribbon command in Insight II.

Residue_Dihedral

The Residue_Dihedral command enables you to tabulate peptide/protein-specific dihedral angles. These include the backbone $\phi$, $\psi$, $\omega$ and sidechain $\chi_1$, $\chi_2$, $\chi_3$, and $\chi_4$ angles. The calculation can be performed on individual molecules or on an assembly of conformers of the same molecule. In the latter case the individual dihedrals can be tabulated along with the minimum, maximum, and circular variance (see Residue Dihedral Angles) in the selected
4. Command summary

dihedral across the assembly. The circular variance thus provides a measure of the conformational variability across the assembly of conformers.

SecondaryClassify

The SecondaryClassify command can compute protein secondary structure classification using either the method of Kabsch and Sander (1983) or the existing classification derived from the PDB format file that was the source of the protein molecule. The command will either create a new residue table with a classification column, or add a classification column to an existing per-residue spreadsheet. SecondaryClassify can also create subsets for use by other Insight II commands. A residue table with a classification column can serve as input to the Molecule/SecondaryRender command.

Access_Surf

The Access_Surf command calculates the solvent accessible surface (SAS) area for a molecule. The terminology and definitions for this procedure are taken from Lee and Richards (1971). The SAS is the area traced out by the center of a solvent molecule rolled across the surface. The algorithm is a version of the Lee and Richards (1971) method as modified by Shrake and Rupley (1973) for speed.

Modeler pulldown

This is an optional pulldown. Please see the Modeler manual for more information.

Seqfold pulldown

This is an optional pulldown. Please see the Seqfold manual for more information.
This chapter describes the proper order and combination of commands that you use for a typical project using the Homology module. To summarize, the process consists of the following steps:

1. Determine which proteins are related to the model protein.
2. Determine structurally conserved regions (SCRs).
3. Align sequences.
4. Assign coordinates within the SCRs.
5. Build loop or variable regions (VRs).
6. Explore the possible range of conformations of side chains for particular residues.
7. Refine the model structure with Discover/CHARMm.

The chapter includes commands from both Homology and Insight II. For more information on these commands, refer to Chapter 4, Command summary, and the Insight II Reference Guide.

**Step 1: Determine Which Proteins Are Related to the Model Protein**

Many of the known proteins fall into families. This happens through evolution, and proteins from more closely related species resemble one another more than those from distantly related species. The resemblance is in terms of function, amino acid sequence, and three-dimensional structure. It can be generally assumed that if two proteins have similar sequences, then their three-dimensional structures will be similar.
5. Methodology

**Sequence Database Searching**

The first step in predicting a protein structure from its amino acid sequence is to identify the family to which the sequence belongs. This is most easily done by searching a database of amino acid sequences. The sequence database files must be installed so that they can be accessed by the sequence database searching software. This procedure is explained in detail in the MSI System Guide.

The search can be restricted to include only the sequences of those proteins for which complete tertiary structures are known. This is a good way to find reference structures. To do this, you must first run the program `seq_extract` to create a database file that contains only sequences for which the structure is known (see Appendix D, `seq_extract Utility`).

MSI provides two alternative methods for finding reference proteins: Seqfold which uses the secondary structure enhanced sequence comparison method; and Profiles_3D. The `Find Structures` command in Profiles_3D directly tests the compatibility of a sequence with 3D folds, rather than with the sequences associated with those 3D folds. Both are highly sensitive methods that may succeed in cases where a conventional sequence similarity search fails. They are described in the Seqfold and Profiles_3D documentation. The remainder of this section concerns only conventional sequence similarity searches using the FASTA algorithm in the Homology program.

If there is only one known reference protein for your sequence, a search of the complete database of sequences can be especially valuable. A multiple alignment of related sequences often reveals those places in the sequences that are most highly conserved (by sequence similarity criteria). These regions are likely to be structurally conserved and can therefore reveal the best places in the reference protein from which to copy coordinates as you build the conserved core structure of your model.

The commands in the Databases pulldown provide a means to search sequence databases for proteins that are similar to a given sequence. The database(s) are not distributed by MSI but can be obtained from other sources (see Appendix G, Sequence Databases), and should be stored as one or more disk files accessible from your workstation.
Step 1: Determine Which Proteins Are Related to the Model Protein

The search is accomplished by running the FASTA program (Lipman and Pearson 1985, Pearson and Lipman 1988, Pearson 1990) as a background job (see the Background Job Pulldown). Two of the commands in the pulldown, Input Databases and Output Databases, specify the various options to the FASTA program and the Run_FASTA Databases command submits the background job. The commands in the Background Job pulldown control the execution of the process, specify the host machine, and query the job status.

All the parameter values for these commands are saved after a job is submitted. Therefore, multiple jobs can easily be run using the same parameters or with one or a few parameters modified before resubmitting the job. But once a parameter is set by selecting Execute, there is no way to go back to the original default value. Insight II must be restarted to accomplish that.

The Input Databases command specifies parameters concerning the execution of the FASTA search program. The command is divided into two parts. The set of parameters that are active is determined by the value of the End_Definition boolean parameter. When the command is first selected (with End_Definition off), parameters relating to the specification of sequences and databases are active. When End_Definition is toggled on, parameters having to do with the calculation of match scores become active.

With the End_Definition parameter off, the Probe Sequence parameter specifies which given sequence is to be matched to the database. It must be a part of the sequence display at the bottom of the screen, but it can be either a reference protein or a sequence with no defined coordinates. The Extract Sequences command can be used to display the sequence of the former type and Get Sequences for the latter type. The Activation parameter controls the creation and modification of the list of Database files to search. Thus, you can Add a Database file, Delete a specific one, Clear all the entries, or List the Database files to be searched. The choice of Database files is controlled by the Database Type parameter. There is a separate list of Protein and DNA databases, and the types cannot be mixed within a list. If you wish to change from one type to another, the current list must be cleared first. If the Database Type is Protein, you can specify the Scoring Matrix that is used to calculate the init1 score (see Chapter 2, Theory). This matrix gives high scores for residue identities and conservative substitu-
5. Methodology

tions and low scores for rare substitutions. See Scoring Matrices for a discussion of them. If the Database Type is DNA, then you can choose whether to consider matches in one direction (Downstream) or in both directions. Finally, the value of the Ktup parameter can be set. It specifies the number of residues to consider as a group during the initial scanning of the database. The higher the value, the faster the scan, but the less sensitive the search.

When the End_Definition parameter of the Input Databases command is toggled on, three other parameters become active. The Optimize parameter specifies whether initially matched regions are to be rescoring using the Needleman and Wunsch method (see Chapter 2, Theory). If so, then the Optimize Threshold value specifies the scores above which optimization is done. When high-scoring single regions are found within a single sequence, they can be joined together to form a longer matched region containing insertions and deletions. Segments scoring higher than the Joining Threshold will be joined in this way. As a general rule, if optimization is done, the two threshold values are kept the same value.

The FASTA program creates several output files. One contains information concerning the run itself; another contains the results of the database search. The format of the latter file is specified with the parameters of the Output Databases command and its name is specified by the File Name parameter. If the Input Databases command has been previously executed, and there is a current choice for Probe Sequence, a default filename is constructed by adding the extension .fasta to the end of the sequence name. A value-aid is also provided that lists files that have already been written. If you choose a File Name that already exists, the file is overwritten upon execution.

The Alignment Style parameter specifies whether only the regions of the sequences found to be overlapping are written to the output file as an alignment, or whether the whole sequences is, with the similar regions demarcated with “x”s.

The Marking Style parameter controls the way the residue similarities and differences between the Probe Sequence and the matched sequences are indicated. Values for the Marking Style parameter are shown below:
Step 1: Determine Which Proteins Are Related to the Model Protein

**Ident_Conservative**  **Conservtiv_and_Non**  **Replacements_Only**

<table>
<thead>
<tr>
<th>Ident_Conservative</th>
<th>Conservtiv_and_Non</th>
<th>Replacements_Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>MWKTCGPPYT</td>
<td>MWKTCGPPYT</td>
<td>MWKTCGPPYT</td>
</tr>
<tr>
<td>MWKSCGYPYT</td>
<td>MWKSCGYPYT</td>
<td></td>
</tr>
</tbody>
</table>

If **Ident_Conservative** is chosen for the Marking Style, matches between identical residues are shown with a colon (" : ") and conservative substitutions are shown with a period (" . "). If **Conservtiv_and_Non** is chosen, conservative substitutions are denoted by an "x" and nonconservative substitutions by an "X". If **Replacements_Only** is chosen, replacements are shown with the appropriate amino acid code and identities are indicated with a period (" . "). If **LibSequences_Only** is chosen, only the aligned library sequences will be shown without the query sequence. This can be used to build a primitive multiple alignment.

The number of **Scores to Show** can also be controlled as well as the number of **Alignments to Show**. The latter parameter also specifies the number of sequence files that are written upon completion. For each sequence found and alignment shown, a sequence file is written with a format compatible with the **Get Sequences** command. The name is derived from the database entry code (accession code) of the sequence with the extension .fasta.seq. Finally, if optimization was done in the search, you may sort the sequences found based on the optimized score rather than the **initn score**.

Most of the parameters for the **Input Databases** and **Output Databases** commands have valid default values that can be used. The exceptions are the **Probe Sequence**, **Database Type**, and **Database** parameters for the **Input Databases** command and the **File Name** parameter for the **Output Databases** command.

After setting up both the input and output parameters, the **Run_FASTA** command is used to submit the background job. Any arbitrary string can be entered for the **FASTA Run Name**. This identifies the job for the system. The system then supplies a job number that is used when monitoring the job. The background job creates a file to log any job execution information or errors. It is named bkgd_job_<jobname><n>.csh_log, where <n> is the job number.
5. Methodology

When the background job completes, you can examine the results in the output file named by the Output Databases command. Any desired sequence can be read into Homology with the Get Sequences command. The alignments seen in the output file can be reproduced using the scrolling buttons.

Motif searching

The structural similarity of different proteins may be due to a shared ancestor. On the other hand, it may simply be the result of the fact that the number of possible protein motifs is much smaller than the number of proteins. Even though new structures are reported almost every day, most of those structures are somewhat similar to structures already present in the Protein Data Bank/Brookhaven (see Setting Up the Brookhaven PDB for information about configuring the Brookhaven PDB for use in Insight II). It is estimated that the number of all possible, distinct folds is in the order of 1000. Therefore, it is essential to have a fast method for scanning the existing structural databases to identify possible structural matches.

A Motif_Search scan is performed in the following way. The query protein is first annotated with a secondary structure calculated using an algorithm similar to the Kabsch and Sander algorithm. Turn predictions are ignored and alpha-helices and beta-strands are smoothed to avoid sudden breaks in otherwise continuous secondary structure elements. Terminals of secondary structure elements are then determined and the best line is fitted to the element using its inertia tensor. The Motif_Search structural database consists of secondary structure element vectors calculated in the same way for all proteins in the current release of PDB.

For every protein from the database, Motif_Search looks for all possible matches for subsets of three secondary structure elements. Pairs of matching subsets are called triplets. Triplets have to be consistent -- i.e., the types of secondary structure elements and the direction (unless the Mix_Order parameter is specified) of the reference protein must match those of the query protein. Additionally, the spatial compactness of the triplet is constrained by the Compactness parameter, and the triplet RMS (the root mean square deviation of the hit from the query structure) must be lower than the value specified in the RMS_limit parameter.
Step 1: Determine Which Proteins Are Related to the Model Protein

Having identified all possible triplets that fulfill the above criteria, Motif_Search tries to merge triplets that have common secondary structure elements, provided that the merged substructure still satisfies the RMS deviation specified in the RMS_limit parameter. If the number of secondary structure elements in merged substructures is greater than the value of the Element_limit parameter, the substructure is marked as a hit and included in a hit list file.

When the Motif_Action parameter is set to Scan_Motif_DB, then the Motif_Search command spawns a background job that produces a human readable file (described in the file formats appendix). This file is read and analyzed by both the Load_Scan_Results and View_Scan_Results states. Additionally, loading structures of hits requires setting the INSIGHT_PDB environment variable to point to the top of the directory tree that contains pdb files.

A Motif_Search readable database of structures can be generated from the list of pdb files. This feature is not supported by the graphical interface, but an appropriate BCL macro does exist:

```
Motif_Generate Databases -Use_PDB_Definition pdb.flist motif_search.db
```

where Use_PDB_Definition has the same meaning as in the Motif_Search command. pdb.flist is a file containing a list of pdb files — one per line with full path. motif_search.db is the name of the created database. Alternatively, the command line version motif_search can also be used, with a command line of the form:

```
motif_search -gen -list file_list -database file.db
```

where file_list is a list of PDB file names (full path, one per line) and file.db is the name of the database created.

The Motif_Search command allows you to perform fast browsing of the provided comprehensive structural database in the search for structural motifs similar to whole or part of the query protein. A typical Motif_Search session consist of following steps:

- Scan structural database with the query structure.
- Load and browse different hits.

Additionally, you can prepare your own version of the structural database and scan against this database. (For a full discussion of the parameters in Motif_Search, see the xhelp in the Insight II interface).
5. Methodology

**Scanning the database**

In the Scan_Motif_DB state of the Motif_Action parameter, the Motif_Search parameter block allows you to prepare the proper query for the subsequent background job. Main parameters that control the behavior of the search are RMS_limit, Element_limit and Compactness.

RMS_limit specifies the maximum root mean square deviation of the matching substructures. For identification of closely related structures, it can be lowered to about 1 Angstrom, whereas, when distant structural relatives are sought, values in the range of 3 or 4 Angstroms are suggested. The Element_limit parameter specifies the minimum number of secondary structure elements for a substructure to be considered a match. For close matches, this number can approach the total number of secondary structure elements in the query structure. On the other hand, if distantly related structures significantly differ in the number of secondary structure elements, Element_limit should to be specified as a lower bound on the expected number of common subsets of secondary structure elements. Also, the search for a relatively small common motif may be performed by setting Element_limit to the element size of the motif. The Compactness parameter limits the size of the matching substructure and is used to restrict the existence of non-local substructures in large proteins. However, it also prevents matches from merging and therefore should be used cautiously.

The Default structure library provided by MSI consists of all structures from the January 1997 PDB release. It is possible, however, to use other databases by specifying a name in the DB_Name field (visible only when Use_Local_DB is toggled on. Instructions on how to prepare the database are located at the end of the methodology and implementation section.

By default, Motif_Search uses automatic secondary structure assignments. Additionally, secondary structure annotations from the PDB file can be used instead by specifying Use_PDB_Definition. This is not recommended, however, since the database has been prepared consistently with automatic annotation.

**Loading and Viewing matching structures**

Loading results of the database scan can be accomplished by choosing the Load_Scan_Result value of the Motif_Action parameter. Note that you can load results from as many files as you wish. For each output file, a table containing matching structures will be produced. The table name consists of the protein name, capitalized background job name and string “MOTIFS”
Step 1: Determine Which Proteins Are Related to the Model Protein

separated with an underscore (_), i.e. table GOF_MYFIRST_ MOTIFS correspond to the “myfirst” background job of the Motif_search for galactose oxidase (GOF). The job name used as a prefix may be abbreviated if it is too long (>11 characters). The PDB code of the structure that contains a matching fragment is reported together with the number of matching elements, the root mean square deviation of the match from the query protein and the deviation, defined as a radius of gyration of the matching substructure.

There are two levels of browsing hits. In the first level, only the matching substructure can be shown, represented as a set of yellow arrows, superimposed onto a set of white arrows that represent all the secondary structure elements of the query protein. This allows you to explore the extent of the overlap and assess the quality of the match. Query and matching arrows are available as MS_TAR and MS_REF objects, respectively. Note that loading a new set of vectors will overwrite the MS_REF object.

The second level involves loading a whole structure that contains matching substructures and overlapping it with the query protein. The loaded structure is available as a protein called MOTIF_code, where code is the four character PDB code of the hit. In order to successfully perform this step, the environment variable INSIGHT_PDB has to be defined and must point to a directory tree which contains the pdb files. Often, the matching substructure is a small part of the whole structure. It is advantageous, therefore, to use Load_Vectors and Load_Structures simultaneously, which indicates which parts of the structures are considered a true hit and helps to assess how significant such a match is. Note that Load_Vectors and Load_Structures are not coupled and, in principle, can depict different matches.

The currently provided database of structures is complete but redundant. Many structures are duplicated a few and in some cases hundreds of times, which can complicate the analysis. New databases, tailored to specific needs, can easily be prepared by using the Motif_Search database generation options from the UNIX command line as previously described above.

To become acquainted with Motif_Search capabilities, run the Motif Database Searching tutorial from the Homology tutorials collection. In addition to basic operations, use of Motif_Search requires an understanding of the Insight II subset definition and
5. Methodology

subset operations. For more information, read the relevant chapter in the Insight II User Guide.

Step 2: Determining Structurally Conserved Regions (SCRs)

When two or more reference protein structures are available, it is advantageous to use the information they provide to establish structural guidelines for the family of proteins under consideration. Therefore, the first step in building a model protein by homology is determining what regions are structurally conserved or constant among all the reference proteins. It can then be assumed that the unknown protein has the same conformation in these conserved regions.

The proteins are read in with the Get command in the Molecule pulldown (top menu bar). Sequences for the reference proteins are then displayed by selecting the Extract command from the Sequences pulldown in the Homology module. This can take up to a minute for each protein. The sequences are shown in the sequence display at the bottom of the screen.

The Homology module provides two methods for finding SCRs, one that is fast and automatic, and another that requires an interactive, manual search. If the reference proteins are sufficiently similar in structure, then the automatic method is preferable. If, however, the structures are less well-conserved, the automatic method may not be able to find the regions of structural similarity. In such cases the manual method is preferable.

Automatic Determination of SCRs

The automatic method constructs a Cα distance matrix for each molecule and compares small portions of the distance matrices at a time. RMS differences of the distance matrix elements are calculated to assess whether peptide segments are similar. Both local similarities and global orientations of the segments are examined. All overlapping segments are combined to form larger nonoverlapping SCRs. Segments of the sequences that all correspond to a
Step 2: Determining Structurally Conserved Regions (SCRs)

given SCR form an m-block. An m-block is a group of \( m \) mutually related sequence segments, all the same length, that are related by their sequence similarity and/or structural similarity. (In this section the terms SCR and m-block are used interchangeably, but you should remember that the m-block is a more general concept). The segments in the structurally conserved m-blocks are aligned in the sequence window and enclosed by boxes. Also, the three-dimensional structures are superimposed so as to minimize the RMS deviations among corresponding atoms in the SCRs. For a complete description of the method used, see *Automatic Determination of Structurally Conserved Regions*.

To use this method, select the Structure command from the Alignment pulldown and choose Automatic as the Struct Align Mode.

**Specifying the Initial Search Zone**

The automatic Alignment/Structure command searches for SCRs in a “zone” of sequence segments. Normally this zone includes the full lengths of all the reference proteins. The Seq to Align 1 through Seq to Align 10 parameters specify the sequences to be searched for SCRs and aligned. By default they contain the names of all sequences present in the sequence display area for which there are corresponding three-dimensional structures. If there are fewer than ten sequences present, the unused Seq to Align parameters are set to None.

Each sequence name can be followed by an optional colon and residue number to specify the leftmost (closest to N-terminal) residue of the zone to be searched during the alignment. Any portions of the sequences outside this zone will not be aligned. If the residue number is omitted, then the zone begins at the N-terminus.

The length of the zone is specified by the Zone Length parameter. By default, it is set to the length of the longest sequence in the sequence display. Zone Length may be longer than all of the sequences, in which case the search zone extends to the C-termini of all sequences selected for alignment.

The Seq to Align parameters can be set by picking the sequences in the sequence display area, selecting the molecule names in the value-aid, or picking the molecules on the screen. If the Mol/Res Pick Level parameter is set to Residue, then individual residues can be picked to indicate the leftmost residue to be searched in the
selected sequence. A sequence name can be deleted by selecting the parameter field, pressing the <Delete> key, and then pressing <Return>.

If you try to enter the same protein name in more than one Seq to Align parameter, then the redundant entries will just overwrite the first instance of that name. For example, if you enter FBJ:13 in Seq to Align 1 and then try to enter FBJ:22 in Seq to Align 2, FBJ:22 will appear in Seq to Align 1 and Seq to Align 2 will be left unchanged. This feature prevents duplication of sequence names and facilitates adjustment of the starting points of the search zone.

The Minimum Seq Per Blk parameter specifies the minimum number of sequence segments that an m-block must contain to be included in the alignment. It must be greater than or equal to 2, and less than or equal to the number of sequences selected for alignment. The default value is 2.

The search for SCRs initially covers the entire zone selected for alignment. As m-blocks are found, they subdivide the sequences into shorter zones that are in turn recursively searched for smaller m-blocks. The Minimum Zone Length parameter specifies the length, in residues, of the shortest zone that is considered worth searching. If any of the sequence segments between the m-blocks are shorter than Minimum Zone Length, then these will be excluded from the new search zones. Large values of Minimum Zone Length result in faster but less complete alignments. Minimum Zone Length must be greater than or equal to 1. The default value is 3.

The parameter Minimum Blk Length specifies the minimum length, in number of residues, that an m-block must have to be incorporated into the alignment. Any m-blocks shorter than Minimum Blk Length are automatically rejected.

**Optimizing the Automatic Search for SCRs**

Four numeric parameters control the structural similarity search: Probe Size, Contiguous Thresh, Scan Limit, and Orientation Thresh.

The Probe Size is the length of the portions of the distance matrices that are compared. If this parameter is small (three to five residues), then more matches will be found locally, but many of them
Step 2: Determining Structurally Conserved Regions (SCRs)

will be screened out as spurious later in the algorithm when global orientations are considered. Values in the range 6 to 10 are recommended.

The **Contiguous Thresh** parameter indicates how closely matched two segments must be before they are accepted as similar. The lower the value, the more stringent the criterion, and the fewer the number of segments will be found, leading to shorter SCRs in the final analysis.

The **Scan Limit** indicates how far apart in sequence two segments can be before an attempt is made to match them. The higher the value, the farther apart they can be. If you expect that there is a large insertion in one of the proteins, then you should increase the value above the default of 20 residues.

The **Orientation Thresh** is a measure of how similar the relative orientations of two segments must be before they can be considered to be in the final set of SCRs. All pairs of matched segments are checked for self-consistency. The higher this value, the more segments are accepted, and the longer the final SCRs are as overlapping segments are combined.

The default values of these four parameters work well for proteins that are highly similar in their structurally conserved regions, such as the trypsin family of serine proteases or the C-type lysozymes used in the tutorials (Chapter 6, Tutorial). For less well-conserved proteins, however, some adjustment of the **Contiguous Thresh** and **Orientation Thresh** parameters may be necessary. In many cases, only **Orientation Thresh** need be increased. If no improvement is seen when **Orientation Thresh** is increased to about 3 or 4, then it may help to increase **Contiguous Thresh** as well. Normally **Contiguous Thresh** should be kept smaller than **Orientation Thresh**. As the thresholds are increased, the algorithm becomes less selective in its criteria for determining SCRs. Beyond a certain point, the thresholds may be so large that the SCRs found may be spurious. There is no perfect guideline for choosing the best values of these thresholds for all protein families, but, in general, if **Contiguous Thresh** and **Orientation Thresh** must be made larger than about 2 and 6, respectively, to find SCRs, then the SCRs are likely to be inaccurate. For structures this poorly conserved you should use either the manual method for finding SCRs or the **Alignment/Multiple_Sequence** command.
5. Methodology

The four parameters just described control the search for structural similarity between two proteins. If more than two proteins are selected for alignment by the automatic Alignment/Structure command, then all possible pairs in the selected proteins are searched for structural similarity. As explained in Chapter 2, Theory, these pairwise SCRs (2-blocks) are merged into m-blocks of more than two sequences wherever a sufficient number of the overlapping 2-blocks are mutually consistent. When the parameter Strict_Overlap is on, this “sufficient number” is \( m (m-1) / 2 \), i.e., all possible pairs must be structurally similar to from a block of \( m \) sequence segments. When Strict_Overlap is off, a smaller number of overlapping 2-blocks is sufficient (see Theory for a more detailed explanation). When Strict_Overlap is on, m-blocks tend to be shorter, but their constituent segments are structurally more similar to one another. This often gives the best results for highly similar, closely related reference structures, such as the lysozymes in tutorial lesson 2 in Chapter 6, Tutorial. If the structures are less closely related, then the algorithm may be too conservative in setting the boundaries of the SCRs unless Strict_Overlap is turned off.

Subsets

For each sequence that is contained within one or more m-blocks, the Alignment/Structure command creates subsets that specify those regions of the sequence contained within the m-blocks. The subsets are named \(<\text{protein_name}$AUTO$SCR<n>\) for each individual region, and \(<\text{protein_name}$AUTO$SCR for the union of the regions for the given protein. These are ordinary subsets and can be used as input in other commands, such as the Molecule/Color and Molecule/Label commands.

The creation of subsets can be suppressed by setting the Create_Subsets parameter to off. This is reduces calculation time and is therefore useful when you are running the command many times with different combinations of parameters to optimize an alignment.

Automatic Superimposing of Structures

If two or more sequences of the reference proteins are contained within m-blocks, and if the parameter Superimpose SCRs is on, then the Alignment/Structure command automatically superim-
Step 2: Determining Structurally Conserved Regions (SCRs)

poses their structures. It does this so as to minimize the RMS deviation over all corresponding alpha carbon atoms of residues within the m-blocks. This RMS deviation appears in the information area. If more than two structures are superimposed, and if the parameter Create_RMS_Table is on, then a table is also created that shows the RMS deviations between the corresponding alpha carbon atoms of each pair of superimposed proteins.

If the parameter Create_Assembly is on, then all superimposed structures are grouped into an assembly. An assembly is simply a grouping of molecules that allows them to be operated on as a whole. In particular, it allows them to be connected to the rotation and translation dials independently of other molecules not in the assembly. This is useful when only some of the structures on the screen are related by m-blocks and superimposed. The superimposed structures can be moved away from the unrelated structures and rotated independently of them. The assembly is named ASY_FAMILY, since it normally will be a grouping of proteins that belong to one family. Similarly, the RMS table described above is named TBL_RMS_FAMILY.

If the proteins selected for alignment are from two or more unrelated families, the Alignment/Structure command can align each family independently of the others. The grouping of the proteins into families will be evident from the contents of the m-blocks. In such cases the structures in each family will be independently superimposed, and a separate RMS table and assembly will be created for each family. The assembly and table names will be distinguished by 2-digit numbers appended to the names (e.g., ASY_FAMILY01, ASY_FAMILY02, etc.).

Characteristics of m-boxes

Each m-block found by the Alignment/Structure command is enclosed in a special type of box known as an m-box. The main difference between an m-box and a normal SCR box is that an m-box can contain more than two sequence segments, whereas a normal SCR box can only contain two sequence segments. Like normal SCR boxes, m-boxes can be frozen (not movable), and unfrozen (movable). They are magenta when frozen and blue when unfrozen. The Alignment/Structure command creates them in the frozen state if the parameter Freeze_New_Boxes is on; otherwise it creates them in the unfrozen state.
5. Methodology

When the Alignment/Structure command finishes execution, it automatically switches the sequence window into the Contents coloring mode. In this mode the sequence segments that belong to m-blocks are colored, and all other residues are white. An m-block also has associated with it a p-value that represents the statistical significance of the sequence similarity of its constituent segments. The p-value can be displayed via color-coding by switching the sequence window coloring mode to p-value. This feature is explained in detail in the description of multiple sequence alignment (see Statistical Significance and Alternate Sequence Coloring).

Handling of Existing Boxes

The Delete_Boxes parameter determines how the Alignment/Structure command deals with existing boxes. The alignment cannot be done if any boxes exist within the zone selected for alignment, or if any boxes other than m-boxes exist outside the zone. If any such boxes exist and Delete_Boxes is off, then the program issues an error message and aborts the alignment. If Delete_Boxes is on, then the command automatically deletes any boxes that would otherwise prevent the alignment. Summary boxes are an exception: if present when Delete_Boxes is on, the program issues an error message and aborts the alignment without deleting the summary boxes (or any other boxes). The default value for Delete_Boxes is off.

Interrupting the Search

The Alignment/Structure command can be interrupted at any time by pressing the <Esc> key. A window appears that asks you to confirm or cancel your interrupt request. If you cancel the interrupt request, the calculation resumes. If you confirm the interrupt, any m-blocks that have been found up to that point will be incorporated into the alignment, and a message appears in the information area warning you that the alignment may be incomplete.

Manual Determination of SCRs

Manual determination of SCRs is done by interactively measuring the structural similarity between pairs of protein segments. Each such pair of segments is specified by a box that you create and
Step 2: Determining Structurally Conserved Regions (SCRs)

manipulate. You move the box, change its length and alter the alignment of the sequences it contains while monitoring an RMS score that reflects the structural similarity of the two structural segments that correspond to the two sequence segments in the box. When you have found all the pairwise SCRs in this way, you then execute a command that merges them, where they overlap, into SCRs that contain more than two segments. These are enclosed in white boxes called summary boxes. They are similar to m-boxes, in that they contain more than two sequences, but, unlike m-boxes, summary boxes are permanently frozen. The final step in the procedure is to superimpose the three-dimensional structures over the SCRs specified by the summary boxes.

Finding Pairwise SCRs

To determine the best possible structurally conserved region in a particular part of two reference proteins, you need to perform RMS calculations. The position and size of the active (green) sequence box determines which residues are included in the calculations.

The Structure command in the Alignment pulldown is used to perform RMS calculations. When Manual is selected as the Struct Align Mode, internal flags are set so that a new RMS calculation is performed every time the active box is altered or a sequence is scrolled through it.

The RMS calculation is performed over the four backbone atoms of all residues enclosed by the active box, if these atoms are present. If one or both of the proteins contains only Cα atoms, then the comparison is done only over Cα atoms. The two proteins are optimally superimposed, via rigid rotation, over the corresponding regions enclosed by the active box, and labels appear on the residues that are being compared. The superposition, RMS value, and labels are all updated whenever the contents of the box change (for example, as a consequence of moving the box, changing its size, or moving a sequence through it).

Criteria for Evaluating Manually-Determined SCRs

Four criteria are used to determine whether a putative SCR is a good choice.
5. Methodology

♦ Reasonably low RMS value - Because you are comparing backbone coordinates over such a small stretch of the protein, usually 10-15 residues, the expected RMS is much lower than for comparisons between whole proteins. Values less than 0.75 Å are appropriate. Values more than 1.0 Å usually indicate misalignment.

♦ Overall structural alignment - When the RMS deviation is very high, the two proteins being compared do not line up well. Superimposing the peptide segment causes the rest of the molecule to be rotated as well. If the two segments do not have the same orientation in their respective molecules, then the molecules cannot be aligned. Even if the RMS value is low in this situation, the peptide segments are poor choices for an SCR because it is impossible to choose which of the two orientations is correct for the model you are building.

♦ Increase in RMS - At the ends of an SCR, the RMS value rises steeply as the peptide chains diverge. When the sequence box is expanded beyond the limits of an SCR, the RMS increases abruptly.

♦ No manually determined SCR can span more than one secondary structural unit of a protein. In this definition, an antiparallel beta sheet would not be called a single conserved region. Instead, each of its beta strands would be classified as SCRs. This restriction is imposed to allow for insertions and deletions in the amino acid sequences, which are primarily seen at turns and loops.

It is best to find the SCRs in an orderly fashion from N- to C-terminus. If more than two reference proteins are available, SCRs can be determined between all combinations of proteins. They need not be of the same length at this stage. As they are decided upon, the positions of the boxes defining the SCRs should be preserved with the Freeze Boxes command. This prevents any inadvertent alteration of the boxes when scrolling sequences or introducing gaps elsewhere.

Summarizing the Manually-Determined SCRs

After all areas of structural conservation have been found, the Summarize command in the Boxes pulldown should be used to create subsets (abbreviations) for all the consensus SCRs in the ref-
Step 2: Determining Structurally Conserved Regions (SCRs)

If conserved regions have been defined in any reference proteins, this command examines all the proposed SCRs indicated by all the sequence boxes. In each region, it defines the final SCR as the maximum number of residues that are found in all overlapping boxes. An equal number of sequence boxes must exist for each pair of proteins, and every conserved region must have boxes defined for every possible pair. White summary boxes are drawn around the regions, and all the boxes used in the definitions for the summary boxes are frozen if they were not already.

The subsets created by the **Summarize Boxes** command are named as

<protein_name>$SCR<n> for each individual region, and <protein_name>$SCR for the union of the regions for a given protein. These are ordinary subsets and can be used as input in other commands, such as **Color** and **Label** in the **Molecule** pulldown.

Superimposing Reference Proteins Using Manually-Determined SCRs

As a final step in the manual determination of SCRs, the subsets created by the **Summarize Boxes** command should be used to superimpose the three dimensional structures of the reference proteins. The purpose of this is to define a consistent global coordinate frame for all the reference proteins. If this is not done, there may be a structural misalignment between any segments in the model protein built from different reference proteins.

The reference proteins are superimposed over their corresponding manually-determined SCRs by using the **Superimpose** command in the **Transform** pulldown. In the **Superimpose** parameter block, the **Molecule Pick Level** should be set to **Subset** and the **Superposition Mode** should be **Backbone** or **Trace**. Decide which protein should be the target molecule and enter protein1$SCR and protein2$SCR as the **Source** and **Target Specs**. After each of the reference proteins has been superimposed onto the same target protein, assignment of coordinates to the model protein can be done from any reference protein.
5. Methodology

Superimposing structures

For sequences that have associated three-dimensional structures, automatic structural superimposition can be used to:

- Superimpose structures according to their sequence alignments
- Guide you in manually adjusting the alignment

Superimpose_Aln can be used to assist a broad spectrum of alignment methods. This diversity of methods includes:

- Manual alignment changes you make in the sequence window with the mouse
- Any automatic structural or sequence engines in Insight II
- Loading an alignment obtained using a method which is external to Insight II

You must perform the following steps:

- Select the structures using the operations under Select_Structure in the Superimpose_Op Select mode of the Superimpose_Aln command and press Execute.

  (For information about the Superimpose_Aln command, see the Superimpose_Aln in the Command summary chapter).

- In the same command, display the parameters defining the superimposition mode by selecting GenerateMatches.

- Click on the AutoSuperimpose checkbox in GenerateMatches.

If the AutoSuperimpose checkbox is on, the superimposition is done automatically with the calculation of matches. This is especially useful when you want the changes in the alignment to be automatically reflected in the structural superimposition.

The Superimpose_Aln command assists sequence alignment tools by automatically superimposing structures and monitoring the statistics of the superimposition in response to any change made to the current alignment.

If the AutoSuperimpose checkbox is off, only matches will be calculated over the existing alignment and spatial arrangement of the structures.
To perform a superimposition, with **AutoSuperimpose** on, the **Superimpose_Aln** command should be executed twice: the first time in **GenerateMatches** mode, the second time in **Superimpose** mode.

In **GenerateMatches** mode, when the **AutoSuperimpose** checkbox is on, each change in the alignment will automatically recalculate the matches and the structural superimposition. Simultaneously, the RMS distances from all the proteins to the first structure and the number of matches will be displayed as a short, one line summary in the textport.

Here is an example one line summary:

\[ \text{EST}(0.00/0.00)\text{TGN}(1.00/0.90)\#\text{res}(208/100) \]

EST and TGN are the proteins which have been superimposed. All the proteins are superimposed on the first one; that is why both numbers for the RMSD (in parentheses) are zero for EST.

The first number in each set of parentheses refers to the parameter value for all aligned residues, while the second refers to the parameter value calculated only over the matches.

The last set of parameters starting with the keyword “#res” indicates the number of aligned residues (208) and the number of matches (100).

To get out of the automatic superimposition mode, you must **Execute** the command in the **Off** mode after selecting the superimposition results you desire (**Generate Boxes**, **Assemble**, **Tabulate_RMS**).

When setting parameters in the **Generate Matches** mode, you can choose to use a color coding mechanism (**Use_Colors**) both to mark elements of the secondary structure and to discriminate visually between matches and non-matches. The same colors will be used as foreground colors for corresponding residues in the sequence alignment window.

In addition, you can choose to display spatial relationships between matched residues in the 3D window using a simplified distance-monitor (the **Draw Monitors** section of the parameter window). Green dashed lines are drawn for all residue pairs comprising the set of the matching residues at the current position. When the number of monitors for one residue exceeds 10, the
5. Methodology

monitors are drawn only between consecutive sequences in order of their alignment or, if selected one by one, in the order of their addition to the set of superimposed sequences.

You can choose to report the results of the alignment in any combination of the following ways:

♦ **GenerateBoxes** Boxes may be created in the matched regions of the alignment joining all pairs in the selected structures.

♦ **Assemble** Superimposed structures may be grouped as an assembly so that their spatial orientation will remain fixed during further superimposition (**KeepAssemIntact** parameter in **Superimpose** mode).

♦ **Tabulate_RMS** A table of total pairwise RMS differences may be created as well as detailed tables of residue C-alpha atom deviations from the corresponding frame positions and the corresponding atoms in the first structure.

---

**Multiple Sequence Alignment as an Alternative to the Manual Method**

There are some homology modeling problems in which the structures are not sufficiently similar to allow automatic determination of SCRs, and yet the sequence similarity is still high enough that good results can be obtained with the **Alignment/Multiple_Sequence** command. In these situations the **Alignment/Multiple_Sequence** command may provide a fast and simple alternative to the manual method of determining SCRs just described. The **Alignment/Multiple_Sequence** command is described in detail under **Multiple Sequence Alignment**.

---

**Step 3: Sequence Alignment**

The alignment phase is most critical to the final structure of the model. It determines the correspondence between residues in the reference proteins and the model protein. The positions of conserved regions and lengths of variable regions are directly affected by the alignment of sequences.
Step 3: Sequence Alignment

If sequence alignment information from a previous project has been saved in a file with the `Sequences/Put` command, it can be read in and reused at this stage. The `Sequences/Put` command stores complete alignments with protein names, relative positions, and sequence gap information. The names of these files have the extension .align. Sequence boxes are not saved in these files, so the positions of any SCRs cannot be stored and retrieved by this method. A .align file can be read in by executing the `Sequences/Get` command with the `Get Mode` parameter set to `Alignment`. Any files with the extension .align are displayed in the value-aid of the `File Name` parameter. Picking one of these with the mouse and selecting `Execute` reads in the alignment contained in the file.

If no prior alignment information is available, the sequences can be automatically aligned using either the `Alignment/Pairwise_Sequence`, `Alignment/Multiple_Sequence`, or `Alignment/Align123` command. The `Alignment/Pairwise_Sequence` command aligns two sequences at a time. The `Alignment/Multiple_Sequence` command can align as many as twenty sequences simultaneously. `Alignment/Multiple_Sequence` can be slow for large sets of sequences. For large sets of sequences, use `Align123`, which can align the maximum number of sequences which can be loaded into the sequence window (50).

The `Alignment/Pairwise_Sequence` command uses one of three methods for aligning a pair of sequences depending on whether or not the reference proteins have been structurally aligned. The appropriate method is chosen automatically.

The first method is that of Needleman and Wunsch (1970; see Needleman and Wunsch Algorithm for Pairwise Alignment). It is used to find an initial alignment when two sequences have just been read in and contain no sequence boxes, m-boxes, or summary boxes. This type of alignment maximizes the matching of amino acid types while minimizing the number of gaps that are inserted. The process can be done repeatedly with the parameters adjusted each time as necessary to optimize the alignment.

The second method is used when several reference proteins have been structurally aligned using the manual method and their SCRs have been summarized. It is then possible to automatically align a sequence protein to one of the reference proteins. This enhanced algorithm (see Needleman and Wunsch Algorithm for Pairwise Alignment) incorporates structural information into the sequence align-
5. Methodology

The program scores the sequence similarity only in the conserved regions and allows gaps to be inserted only between SCRs (i.e., between summary boxes). When a gap is required between two summary boxes in the reference protein, a corresponding gap is placed in each of the other reference proteins included in those summary boxes. In this way the existing structural alignment of the reference proteins with each other is not disrupted by the new alignment of the unknown to the reference protein.

The third method is used when several reference proteins have been structurally aligned using the automatic method. In this case the reference proteins contain m-boxes, not summary boxes. A sequence (typically the model sequence that has no three-dimensional structure) is aligned to one of the reference proteins using the standard Needleman-Wunsch algorithm. Unlike the alignment of a sequence to summary boxes, the alignment of a sequence to m-boxes can place gaps in the sequence opposite an m-box, or in the reference sequences within an m-box. The m-boxes are split automatically as necessary to accommodate the insertion of these gaps.

Choosing a Scoring Matrix

Whenever sequence comparisons are made, either manually or automatically, a scoring matrix is used to measure the degree of similarity between the amino acids.

There are four of these 20 × 20 matrices available for use with the Alignment/Pairwise_Sequence command, and two matrices available for use with the Alignment/Multiple_Sequence command (see Scoring Matrices for a description of each matrix). You must select one of these matrices before executing the Alignment/Pairwise_Sequence or Alignment/Multiple_Sequence command. The selection of a matrix for one of these commands has no effect on the matrix selection of the other command. For each of these two commands the selection remains in effect until a new matrix selection is made.
Automatic Sequence Alignment without SCRs

If it is not obvious where to begin the search for structurally conserved regions, or if there is only one reference protein, automatic sequence alignment should be done. Either the Alignment/Pairwise_Sequence command or the Alignment/Multiple_Sequence command can be used in this situation. The use of the Alignment/Multiple_Sequence command is explained in a separate section below (Multiple Sequence Alignment). This section concerns only the Alignment/Pairwise_Sequence command.

When no summary boxes have been defined, the Alignment/Pairwise_Sequence command uses the method of Needleman and Wunsch described in Needleman and Wunsch Algorithm for Pairwise Alignment. This standard method inserts sequence gaps into both sequences with equal weight to match amino acid similarity as much as possible. A Gap Penalty (a parameter of the Alignment/Pairwise_Sequence command) is associated with starting a new gap region; the higher the value, the lower the number of gaps. The cost of inserting a gap can also be made to increase in proportion to the length of the gap. The Gap Length Penalty parameter is the proportionality constant. The total cost of inserting a gap is:

\[ \text{cost} = \text{Gap Penalty} + \text{Gap Length Penalty} \times (L - 1) \]

where L is the length of the gap in number of residues. The cost is subtracted from the alignment score. A higher Gap Length Penalty tends to produce shorter gaps. If Gap Length Penalty is set to zero, all gaps are penalized equally, regardless of length. You can try using different values for the Gap Penalty and Gap Length Penalty parameters to determine which are best suited for a particular pair of sequences. The computation is invoked by using the Alignment/Pairwise_Sequence command with the Seq Align Mode set to Automatic. It is recommended that Mutation be used for the Scoring Matrix parameter. The calculation can be performed on any two sequences, either reference or unknown. If gaps were already in either of the sequences, they are deleted prior to alignment. It is therefore possible to perform the calculation several times, using different values for the Gap Penalty and Gap Length Penalty parameters, until a satisfactory alignment is
5. Methodology

obtained. The calculation takes from one to several minutes depending on the lengths of the sequences.

**Automatic Sequence Alignment with Automatically-Determined SCRs**

After all the structurally conserved regions in the reference proteins have been found using the automatic method, the sequence of the unknown protein can be aligned with the reference proteins using the Alignment/Pairwise_Sequence command. This alignment must be done before coordinates can be copied from the SCRs of the reference protein to the model protein.

In this case the reference sequences contain m-boxes and the model sequences contain no boxes. The Alignment/Pairwise_Sequence command works exactly as just described for the case in which both sequences contain no boxes. You select one of the reference sequences for alignment with the model sequence. The best choice is usually the reference sequence most similar to the model sequence. The algorithm of Needleman and Wunsch is used to align the sequences, and gaps are inserted as necessary into either sequence to optimize the alignment score. The gap penalty parameters work exactly as when no boxes are present.

Although only one reference sequence is being aligned to the model sequence, any gaps inserted into the reference sequence are automatically duplicated as necessary in the other reference sequences to maintain the correct multiple alignment established by the m-blocks. If a gap must be inserted within an m-block, the corresponding m-box is split at that point to accommodate the gap. It may seem inappropriate to allow the insertion of a gap within an m-block, since an m-block found by the automatic Alignment/Structure command represents a structurally conserved region. Strictly speaking, though, such an m-block may actually contain several structurally conserved regions that are separated by surface loops that just happen to be structurally similar in the chosen reference proteins. A more distantly related protein from the same family may have an insertion in one of those loops, thus requiring a gap within the m-block.

The alignment may be repeated as often as necessary to experiment with different parameter values. If a prior execution of the
Step 3: Sequence Alignment

command inserted a gap that split an m-box, and the current execution does not create that gap, the m-box will remain split. Repeated executions with different gap penalty parameters may therefore fragment the m-boxes unnecessarily. This does not cause any serious problems, since the residues that are aligned within m-boxes remain the same. It can, however, affect the coloring of the m-boxes in p-value mode, since the fragmented m-boxes are shorter and therefore less statistically significant. If you wish to eliminate this fragmentation, the best strategy is to execute the Alignment/Pairwise sequence command as many times as necessary to find the best parameters, then rerun the Alignment/Structure command to restore the original (unfragmented) m-boxes, and finally rerun the Alignment/Pairwise_Sequence command with the optimal parameters.

Automatic Sequence Alignment with Manually-Determined SCRs

After all the structurally conserved regions in the reference proteins have been found using the manual method, and the Boxes/Summarize command has been executed (see Manual Determination of SCRs), the sequence of the unknown protein can be aligned with the conserved regions using the Alignment/Pairwise_Sequence command. This alignment must be done before coordinates can be copied from the SCRs of the reference protein to the model protein.

The presence of summary boxes tells the program that the standard algorithm of Needleman and Wunsch should not be used. Instead a specialized proprietary method (described in MSI's Automatic Divide-and-Conquer Algorithm) is used that does not permit the insertion of gaps within any SCR. The best possible amino acid match for each SCR is made with gaps inserted between them, if necessary. This algorithm has the advantage of incorporating structural information into the alignment.

Like the Needleman and Wunsch automatic sequence alignment method, computation for the proprietary method is invoked by using the Alignment/Pairwise_Sequence command and setting the Seq Align Mode to Automatic. In this case, however, it is recommended that Identity be used as the type of Scoring Matrix because it has more stringent matching criteria. Because gaps are
5. Methodology

only allowed between SCRs, there are fewer chances to correct a
spurious match farther down the sequence.

This nonstandard algorithm ignores any residue in the reference
protein that is not within a summary box. It matches residues only
within SCRs. Gaps are disallowed within SCRs, but can be put in
either sequence between SCRs. If a gap is put in a reference protein
that is tied to other rows via frozen or summary boxes, then gaps
are put in all the reference protein rows at the same column posi-
tion. The calculation takes from one to several minutes to com-
plete. The Gap Penalty and Gap Length Penalty parameters are
not used when performing automatic alignment with summary
boxes.

Since not all the gaps are deleted prior to the calculation, the com-
mand is not repeatable. If you wish to try several alternative
sequence alignments by this method, first save all objects using the
File/Save_Folder command. This stores the original structural
alignment of the reference proteins. You can then do the first align-
ment of the unknown and reference protein. When you are ready
to try the next such alignment, delete all objects using the Object/
Delete command, restore the original structural alignment using
the File/Restore_Folder command, and then execute the next
sequence alignment.

**Pairwise Manual Sequence Alignment**

The results of the automatic alignment procedures can be manu-
ally refined if necessary. Pairwise manual alignment is activated
by executing the Alignment/Pairwise_Sequence command with
the Seq Align Mode parameter set to Manual.

As with structural alignment, internal flags are set up to calculate
an average match score for all residues contained in the active
(green) sequence box whenever it is altered or a sequence is
scrolled through it. The active box should be adjusted and the
alignment done to maximize the score over a particular region that
has the same width as the corresponding SCR found in the refer-
ence proteins.

As explained in the following section, alignments produced by the
Alignment/
Multiple_Sequence command can also be manually refined.
Multiple Sequence Alignment

The Alignment/Multiple_Sequence command is typically used when you have only one reference protein, but you also have one or more sequences from the same family in addition to the model sequence. In this situation you cannot determine the SCRs by structural comparisons, as you could with multiple reference structures. The next best thing is to determine where the proteins are conserved with respect to sequence similarity. By reading in additional sequences from the same family and running the Alignment/Multiple_Sequence command, you can get a more accurate estimate of the locations of the conserved regions than you could from the reference and model sequences alone.

The Alignment/Multiple_Sequence command can also be used to align the model sequence and multiple reference proteins, but it will do so using only sequence similarity information, not structural similarity information. Normally the automatic Alignment/Structure command is a better choice in this situation, but, if the reference proteins are not sufficiently similar in structure, then Alignment/Multiple_Sequence may give better results and is faster than manual determination of SCRs.

The Alignment/Multiple_Sequence command simultaneously aligns up to ten amino acid sequences. Typically the best results are obtained with three to eight sequences. Using the segment pair overlap algorithm (see Automatic Sequence Alignment Methods), the command finds regions over which the sequences are mutually related and estimates the statistical significance of those relationships. Each such block of mutually related sequence segments is called an m-block, where \( m \) is the number of sequence segments in the block. Those m-blocks containing highly significant relationships are likely to contain structurally conserved regions. The algorithm automatically inserts gaps into the sequence display as necessary to bring the contents of the m-blocks into alignment. It makes no attempt to optimize the alignment of residues that are not contained within the m-blocks. It surrounds each m-block with a special type of box known as an m-box. Unlike the normal boxes that you can create interactively when the sequence window is in Box mode, m-boxes can contain more than two sequence segments. The Alignment/Multiple_Sequence command cannot be used if summary boxes are present.
5. Methodology

Search Modes

The **Alignment/Multiple_Sequence** command can be used in either of three modes that are selected via the **Mult Align Mode** parameter.

When **Manual** is chosen as the **Mult Align Mode**, the command simply creates a single m-box having the location and length that you specify. No automatic searching for high-scoring m-blocks is done in this mode. This mode can be useful when you know that certain segments of the sequences should be aligned with one another (e.g., if they contain homologous active-site residues).

When **Single_Search** is chosen as the **Mult Align Mode**, the command searches the sequences only once each time **Execute** is selected. This normally finds only a few m-blocks (possibly none) and therefore produces an incomplete alignment. The alignment can be completed by executing additional single searches. This interactive searching strategy allows manual adjustment of the parameters between searches.

When **Automatic** is chosen as the **Mult Align Mode**, the command searches the sequences repeatedly, each time adjusting the search parameters according to heuristic rules. You can modify the behavior of these rules, and thereby adjust the thoroughness of the repetitive automatic searching, by changing certain parameters of the **Alignment/Multiple_Sequence** command. The repetitive searching continues until no more statistically significant m-blocks can be found. **Automatic** mode is the preferred method for most alignment problems.

**Parameters that Control the Automatic Repetitive Search**

When **Mult Align Mode** is set to **Automatic**, the behavior of the automatic divide-and-conquer algorithm is controlled by the **Pairwise Threshold**, **Minimum Zone Length**, **Num Related Seqs**, **Minimum Seq Per Blk**, **Not Signif p**, and **Dimension Bias** parameters.

As explained in the Theory section, the sensitivity of each segment pair overlap search is determined by the pairwise score threshold, $t$. In addition, each search in **Automatic** mode is further constrained by the requirement that $m$, the number of sequence segments in the m-blocks found, must be greater than or equal to an internal variable $m_{min}$. The divide-and-conquer algorithm adjusts the variables $t$ and $m_{min}$ after each segment pair overlap search.
The first search begins with \( t \) equal to the Pairwise Threshold parameter and with \( m_{\text{min}} \) equal to the Num Related Seqs parameter. The default value of Num Related Seqs is the total number of sequences selected for alignment. If no m-blocks are found, then \( t \) is lowered by a fixed percentage and the search is repeated. This cycle of searching and threshold reduction is repeated until one of the following two conditions is met:

1. One or more statistically significant m-blocks are found, or
2. The current threshold is so low that there is little chance that further searches at lower thresholds would find significant m-blocks.

If the first condition occurs, then the newly found m-blocks are saved for alignment, and the process of searching and threshold reductions is repeated for the remaining zones between the significant m-blocks. An m-block is considered significant if its \( p \) value (from Eq. 2-1 in the Homology 2.0 User Guide) is less than the Not Signif \( p \) parameter. As m-blocks are found, they are inserted into the alignment in the order of their statistical significance (most significant first). An m-block is rejected if it overlaps or conflicts with another m-block that is already in the alignment.

The second condition occurs when either of the following criteria is met:

a. The search found one or more m-blocks, but all of them were statistically insignificant, or
b. The search found no m-blocks, and \( t \) is less than or equal to \((1 - \text{Dimension Bias}) \ (t_{\text{min}})\).

The internal parameter \( t_{\text{min}} \) is the lowest pairwise threshold that could be expected to find a significant m-block, assuming that all possible segment pairs in the m-block have identical pairwise scores equal to \( t_{\text{min}} \). It is calculated using the inverse of Eq. 2-1 in the Homology User Guide, with \( p \) set equal to Not Signif \( p \). Because the pairwise segment scores within a real m-block are rarely identical, \( t_{\text{min}} \) is only an estimate of the lowest pairwise score threshold that can find significant m-blocks. In practice it is usually best to continue searching for m-blocks until \( t \) is somewhat lower than \( t_{\text{min}} \). The parameter Dimension Bias determines how...
5. Methodology

much lower than $t_{\text{min}}$ the threshold $t$ is allowed to go. **Dimension Bias** can have any value between 0 and 1, inclusive.

If the second condition occurs, then any further searching at lower thresholds would probably be unproductive. There may, however, be undiscovered significant m-blocks for which $m$ is less than the current value of $m_{\text{min}}$. At this point, therefore, $m_{\text{min}}$ is decremented. At the same time, the threshold $t$ is reset to the value of **Pairwise Threshold**. Again the remaining zones are searched repeatedly, the threshold being decremented after each search, until either the first or second condition is met.

This entire process is repeated until $m_{\text{min}}$ becomes less than the parameter **Minimum Seq Per Blk**, or until the remaining zones between m-blocks are so short that they are not worth searching. A zone is considered too short to search if it does not contain two or more sequence segments of length greater than or equal to **Minimum Zone Length**.

With this algorithm in mind, the intuitive meanings of the parameters become clear. The algorithm has an intrinsic bias towards selecting m-blocks with the highest possible dimension, $m$. This bias arises from the fact that the searching begins with $m_{\text{min}}$ equal to **Num Related Seqs**, which is normally the total number of sequences selected for alignment (the highest possible $m$). Any blocks of lower dimension are ignored until no more significant blocks of dimension $m_{\text{min}}$ can be found. Blocks of lower dimension are allowed into the alignment only after the second condition is met. If parameters are adjusted such that the second condition is more easily satisfied, then the bias towards m-blocks of high dimension is reduced. Reducing the **Dimension Bias** parameter has exactly this effect. Notice however, that the second condition is also satisfied if the last search finds only insignificant m-blocks. In many alignment problems this occurs before $t$ becomes less than $t_{\text{min}}$, and in these cases adjusting **Dimension Bias** has no effect on the results.

The **Num Related Seqs** parameter specifies the number of sequences known to be related to one another. Normally it is the total number of sequences selected for alignment. If, however, you are not certain that all of the sequences are related to one another, you can speed the alignment by reducing **Num Related Seqs**. This allows the inclusion into the alignment of m-blocks with $m$ equal to this lower number, even in the earliest searches. Reducing **Num
Step 3: Sequence Alignment

**Related Seqs** thereby reduces the bias towards blocks of high dimension. Note that if **Num Related Seqs** is less than the number of sequences selected, blocks containing more than Num Related Seqs sequence segments can still be found since blocks of dimension $m_{\text{min}}$ or higher are acceptable.

The **Not Signif p** parameter is the criterion for statistical significance. Any m-block with $p$ greater than or equal to **Not Signif p** is considered insignificant and is rejected from the alignment. **Not Signif p** can also affect the bias towards blocks of high dimension, since $t_{\text{min}}$ is a function of **Not Signif p**. Larger values of **Not Signif p** tend to produce a more complete alignment, with more blocks and possibly greater bias towards blocks of high dimension. Smaller values of **Not Signif p** produce alignments with fewer but more significant m-blocks.

The **Minimum Seq Per Blk** parameter specifies the minimum number of sequence segments that an m-block must contain to be included in the alignment. It must be greater than or equal to 2, and less than or equal to the number of sequences selected for alignment. The default value is 2.

The **Minimum Zone Length** parameter specifies the length, in residues, of the shortest zone that is considered worth searching. If any of the sequence segments between the m-blocks are shorter than **Minimum Zone Length**, then these will be excluded from the new search zones. Large values of **Minimum Zone Length** result in faster but less complete alignments. **Minimum Zone Length** must be greater than or equal to 1. The default value is 3.

The **Pairwise Threshold** parameter specifies the first and highest pairwise score threshold used in the segment pair overlap searches. For each of the later searches, the threshold is automatically decremented as necessary to increase the sensitivity of the search. The default value depends on the number and lengths of the sequences selected for alignment (see *Automatic Calculation of Pairwise Threshold*).

**Specifying the Initial Search Zone**

The **Seq to Align 1** through **Seq to Align 10** parameters specify the sequences to be aligned. By default they contain the names of all sequences present in the sequence display area. If there are fewer
5. Methodology

than ten sequences present, then the unused **Seq to Align** parameters are set to **None**.

Each sequence name can be followed by an optional colon and residue number to specify the leftmost (closest to N-terminal) residue of the zone to be searched during the alignment. Any portions of the sequences outside this zone will not be aligned. If the residue number is omitted, then the zone begins at the N-terminus.

The length of the zone is specified by the **Zone Length** parameter. By default, it is set to the length of the longest sequence in the sequence display. **Zone Length** may be longer than all of the sequences, in which case the search zone extends to the C-termini of all sequences selected for alignment.

The **Seq to Align** parameters can be set by picking the sequences in the sequence display area, selecting the molecule names in the value-aid, or picking the molecules on the screen. If the **Mol/Res Pick Level** parameter is set to **Residue**, then individual residues can be picked to indicate the leftmost residue to be searched in the selected sequence. A sequence name can be deleted by selecting the parameter field, pressing the <Delete> key, and then pressing <Return>.

If you try to enter the same protein name in more than one **Seq to Align** parameter, then the redundant entries will just overwrite the first instance of that name. For example, if you enter FBJ:13 in **Seq to Align 1** and then try to enter FBJ:22 in **Seq to Align 2**, FBJ:22 will appear in **Seq to Align 1** and **Seq to Align 2** will be left unchanged. This feature prevents duplication of sequence names and facilitates adjustment of the starting points of the search zone.

**Specifying a Mandatory Sequence**

The alignment can be constrained to reject all m-blocks that do not contain the sequence specified in the **Mandatory Seq** parameter. It is sometimes useful to set **Mandatory Seq** to the name of the model (unknown) protein. If the algorithm finds overlapping m-blocks, only one of which contains the model protein, and if the block containing the model protein is not the most significant, then this block would normally be rejected from the alignment. If, however, **Mandatory Seq** specifies that the model protein is a mandatory sequence, then the more significant blocks that lack this sequence will be rejected instead.
Automatic Calculation of Pairwise Threshold

The optimal value for Pairwise Threshold depends in a complex way on the scoring matrix used and on the number, lengths, and relatedness of the sequences selected for alignment. The default value of Pairwise Threshold approximates this optimum and is calculated from the number and lengths of the sequences specified in the initial search zone. It is automatically recalculated whenever there is a change in the Mult Scoring Matrix parameter, the Zone Length parameter, or in the number of sequences listed in the Seq to Align parameters.

This automatic recalculation of Pairwise Threshold is suppressed if you manually enter a new Pairwise Threshold value. If you do this and later wish to reactivate the automatic calculation of Pairwise Threshold, you can do so simply by deleting the current Pairwise Threshold value with the <Delete> key. When you then press <Return>, the automatically calculated value appears.

Statistical Significance and Alternate Sequence Coloring

The $p$ value of an m-block is defined as the probability that a block of equal or greater score could be found by chance in a set of random sequences of the same lengths as those in the sequence display. It is calculated using Eq. 10 in the Theory section. The search space $N$ in that equation is calculated as the product of the lengths of the $n$ sequences in the display if the block contains $n$ sequence segments. If the block contains only $m$ segments, where $m<n$, then the search space is calculated as the sum, over all possible choices of $m$ sequences, of the product of the sequence lengths.

Note the distinction between the search space, used in calculating the statistical significance, and the search zone that is selected for alignment. The former is always calculated from the full lengths of all sequences in the sequence display. The latter may be smaller than this, encompassing only a subset of the sequences or a restricted portion of their lengths. If you align some sequences and then add another sequence to the display, the statistical significance of the m-blocks will become lower because the search space has become larger.

The Alignment/Multiple_Sequence command surrounds each m-block with a magenta box (called an m-box) and colors the contents of the m-block magenta as well. The intensity of this magenta
5. Methodology

coloring indicates the statistical significance of the m-block, with paler shades indicating lower statistical significance (larger p). Residues that are white are not significantly related to any others.

The range of p values over which this color change occurs is determined by the parameters High Signif p and Not Signif p in the Alignment/Multiple_Sequence command. All m-blocks for which p is less than High Signif p are considered statistically significant and are colored the darkest magenta. Those for which p lies between High Signif p and Not Signif p are of questionable significance and are colored in successively paler shades as p approaches Not Signif p. Those for which p is greater than Not Signif p are insignificant and are colored white.

The High Signif p and Not Signif p parameters can be changed any time the Alignment/Multiple_Sequence command is active. They are stored internally, even if the Alignment/Multiple_Sequence command is not executed. The coloring of the m-blocks is immediately updated when either of these parameters is changed. When you enter a new value for one of these parameters, it is stored internally when you press the <Return> key, or when you move the focus by selecting another parameter with the mouse. If, however, you enter a new value and immediately select Cancel, then the newly entered value will be discarded.

The magenta coloring of the sequences does not replace the original coloring, but instead constitutes an alternate coloring mode. The button labeled p-value in the lower left corner of the sequence window activates this alternate coloring mode. The Alignment/Multiple_Sequence command always activates the p-value color mode to display the statistical significance of its results. Under some circumstances the contents of an m-block can become statistically insignificant (for example, when you manually shorten the box or change the scoring matrix). If this happens when the coloring mode is set to p-value, it is impossible to know which sequence segments are members of the block, because they are all white. You can always see the which segments belong to a block, regardless of its significance, by switching the coloring mode to Contents. In this mode all segments belonging to a block are colored and all other residues are white.

The Alignment/Multiple_Sequence command also displays a statistical summary of the quality of the alignment in the information area. This summary includes:
Step 3: Sequence Alignment

♦ the number of m-blocks found
♦ the number of residues contained within the m-blocks (as a percentage of the total number of residues)
♦ the number of residue pairs contained within the m-blocks (for each column of an m-block there are \((m - 1)/2\) residue pairs)
♦ the percentage of these residue pairs in which the two amino acids are identical
♦ the mean similarity score of all the residue pairs within the m-blocks
♦ the worst (i.e., minimum) similarity score among all the residue pairs in the m-blocks

The statistical significance calculated by the Alignment/Multiple_Sequence command is only a rough estimate, based on a simplified model of the statistical nature of protein sequences. More importantly, its meaning is based on the null hypothesis that the m-block in question was found by searching random sequences having the same lengths as those in the sequence display. In practice the sequences used in homology modeling come from the much larger search space of a sequence or structural database. For this reason, the \(p\) values calculated by the Alignment/Multiple_Sequence command are underestimates. They are best used as guides to the relative significance of the m-blocks found, rather than as indicators of absolute significance.

Characteristics of m-boxes

Like normal SCR boxes, m-boxes can be frozen (not movable), and unfrozen (movable). They are magenta when frozen and blue when unfrozen. The Alignment/Multiple_Sequence command creates them in the frozen state if the parameter Freeze_New_Boxes is on; otherwise it creates them in the unfrozen state.

An m-box turns green when it is the active box in Scroll Boxes mode. When an unfrozen m-box becomes active, a statistical summary of its contents appears in the information area. This includes the \(p\) value, the number of residue pairs in the block, the percentage of these that are identical, the mean score of all the pairs in the block, and the worst (lowest) score among all the pairs. If \(p\) is less than 0.0001, then its logarithm (base 10) is displayed instead.
5. Methodology

When you change the scoring matrix parameter, the statistical summary is recalculated and updated in the information area. The coloring of the m-blocks is also updated if the coloring mode is set to p-value. This is necessary because the score is a function of the scoring matrix. The statistical summary and m-block coloring are also updated whenever the contents of the active m-box change, either because the box changes in size or position, or because a sequence is scrolled through it.

Symbols other than amino acid symbols, such as gap symbols, are never allowed within an m-block. The Alignment/Multiple_Sequence algorithm also requires that m-blocks never overlap. The enclosing m-boxes may in some cases appear to overlap, but their constituent sequence segments never do.

An unfrozen m-box can be used as a window for exploring the local statistical significance of the alignment in greater detail. As the m-box is moved, the coloring of its contents changes to indicate the significance of the alignment within the box.

Subsets

For each sequence that is contained within one or more m-blocks, the Alignment/Multiple_Sequence command creates subsets that specify those regions of the sequence contained within the m-blocks. The subsets are named <protein_name>$MULT$BLK<n> for each individual region, and <protein_name>$MULT$BLK for the union of the regions for the given protein. These are ordinary subsets and can be used as input in other commands, such as the Molecule/Color and Molecule/Label commands.

The creation of subsets can be suppressed by setting the Create_Subsets parameter to off. This reduces calculation time and is therefore useful when you are running the command many times with different combinations of parameters to optimize an alignment.

Automatic Superimposing of Structures

If two or more sequences that are contained within m-blocks have three-dimensional structures (i.e., are reference proteins), and if the parameter Superimpose_SCRs is on, then the Alignment/Multiple_Sequence command automatically superimposes their
Step 3: Sequence Alignment

structures. It does this so as to minimize the RMS deviation over all corresponding alpha carbon atoms of residues within the m-blocks. This RMS deviation appears in the information area. If more than two structures are superimposed, and if the parameter \texttt{Create_RMS_Table} is on, then a table is also created that shows the RMS deviations between the corresponding alpha carbon atoms of each pair of superimposed proteins. From these RMS scores, and by visually inspecting the quality of the superposition of the structures, you can judge to what extent the m-blocks correspond to structurally conserved regions.

If the parameter \texttt{Create_Assembly} is on, then all superimposed structures are grouped into an assembly. An assembly is simply a grouping of molecules that allows them to be operated on as a whole. In particular, it allows them to be connected to the rotation and translation dials independently of other molecules not in the assembly. This is useful when only some of the structures on the screen are related by m-blocks and superimposed. The superimposed structures can be moved away from the unrelated structures and rotated independently of them. The assembly is named \texttt{ASY_FAMILY}, since it normally will be a grouping of proteins that belong to one family. Similarly, the RMS table described above is named \texttt{TBL_RMS_FAMILY}.

If the proteins selected for alignment are from two or more unrelated families, the \texttt{Alignment/Multiple_Sequence} command can align each family independently of the others. The grouping of the proteins into families will be evident from the contents of the m-blocks. In such cases the structures in each family will be independently superimposed, and a separate RMS table and assembly will be created for each family. The assembly and table names will be distinguished by 2-digit numbers appended to the names (e.g., \texttt{ASY_FAMILY01}, \texttt{ASY_FAMILY02}, etc.).

Adjusting the Sensitivity and Selectivity of the Search

In the search for m-blocks there is a trade-off between sensitivity and selectivity. If a search is too sensitive, then the alignment it produces contains many m-blocks, but some of these may contain insignificant or incorrect local alignments. If a search is too selective, then the m-blocks it finds are likely to be significant and correct, but large portions of the sequences may contain no m-blocks, and important relationships may be overlooked. The most impor-
5. Methodology

tant parameters for adjusting sensitivity and selectivity are Mult Scoring Matrix, Strict Overlap, Not Signif p, and Dimension Bias.

The Mult Scoring Matrix parameter determines which amino acid scoring matrix is used during the search for m-blocks and in the calculation of the statistical significance of each m-block.

The two choices for the Mult Scoring Matrix, PAM_120 and PAM_250, are both evolutionary mutation matrices. They were derived from the same evolutionary data but differ in the way they were normalized. In general, the PAM_250 matrix produces alignments containing larger m-blocks than does the PAM_120 matrix. The PAM_250 matrix is often more sensitive to weak similarities in distantly related sequences than is the PAM_120 matrix. The PAM_120 matrix is more selective and tends to find m-blocks that are more likely to be structurally conserved. The default value for Mult Scoring Matrix is PAM_120.

As described in the Theory section, step 3 of the segment pair overlap algorithm can use either of two criteria for parsing an m-diagonal into m-blocks. The Strict Overlap parameter specifies which of these criteria is used. It determines the degree of mutual relatedness that must exist among the sequences in a local region to form an m-block in that region. When Strict Overlap is on, each sequence segment within an m-block is guaranteed to be related to every other segment in the block. This means that each possible pair of segments within the block lies within a region, over which the cumulative sequence similarity score for that pair exceeds the current pairwise score threshold, $t$. When Strict Overlap is off, each sequence segment within the m-block is guaranteed to be related to at least one other segment in the m-block, but not necessarily to all of them. If only two sequences are being aligned, these two criteria are identical and Strict Overlap has no effect on the results.

If the sequences are closely related (i.e., the percent identity is greater than about 60% within m-blocks), then changing the Strict Overlap parameter may have little effect. For less closely related sequences, however, the m-blocks found with Strict Overlap on tend to be shorter and tend to contain fewer sequences than when Strict Overlap is off. The alignment may therefore be less complete with Strict Overlap on, but the m-blocks found are more likely to be structurally conserved over their entire lengths.
Step 3: Sequence Alignment

If the sequences are distantly related (i.e., the percent identity is less than about 25% within m-blocks), then significant relationships among the sequences may be missed if the alignment is performed with StrictOverlap on.

The selections of the Mult Scoring Matrix and StrictOverlap parameters can be adjusted in tandem to cover the spectrum from high sensitivity to high selectivity. For most sequence families that are sufficiently conserved to be useful for homology modeling, the spectrum is spanned as shown in the following table. There is, however, no simple way to predict which combination of these parameter values will give the best results for a given alignment problem.

<table>
<thead>
<tr>
<th>Scoring Matrix</th>
<th>StrictOverlap</th>
<th>Sensitivity</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAM_250</td>
<td>off</td>
<td>highest</td>
<td>lowest</td>
</tr>
<tr>
<td>PAM_250</td>
<td>on</td>
<td>↓</td>
<td>fl</td>
</tr>
<tr>
<td>PAM_120</td>
<td>off</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAM_120</td>
<td>on</td>
<td>lowest</td>
<td>highest</td>
</tr>
</tbody>
</table>

The NotSignif p and Dimension Bias parameters also affect the sensitivity of the search for m-blocks. Increasing either of these parameters tends to increase the sensitivity of the search, although changing NotSignif p usually has a much greater effect than does changing Dimension Bias.

Handling of Existing Boxes

The Delete_Boxes parameter determines how the Alignment/Multiple_Sequence command deals with existing boxes. Multiple alignment cannot be done if any boxes exist within the zone selected for alignment, or if any boxes other than m-boxes exist outside the zone. If any such boxes exist and Delete_Boxes is off, then the program issues an error message and aborts the alignment. If Delete_Boxes is on, then the Alignment/Multiple_Sequence command automatically deletes any boxes that would...
5. Methodology

otherwise prevent the alignment. Summary boxes are an exception: if present when Delete_Boxes is on, the program issues an error message and aborts the alignment without deleting the summary boxes (or any other boxes). The default value for Delete_Boxes is off.

Interrupting the Search

The Alignment/Multiple_Sequence command can be interrupted at any time by pressing the <Esc> key. A window will appear that asks you to confirm or cancel your interrupt request. If you cancel the interrupt request, the calculation resumes. If you confirm the interrupt, any m-blocks that have been found up to that point will be incorporated into the alignment, and a message will appear in the information area warning you that the alignment may be incomplete. The interrupt capability is useful for difficult alignment problems that result in excessive calculation time (see below). The best strategy in such cases is to interrupt the command and then rerun it using a better choice of parameters.

Excessive Calculation Time

As explained in the Theory section, the time required to find the globally optimal solution to the multiple alignment problem increases exponentially with the number of sequences. The Alignment/Multiple_Sequence command embodies a heuristic algorithm that, in most cases, avoids this exponential complexity. It does so by taking advantage of the fact that related biological sequences are not random, but instead typically contain highly conserved regions separated by less well-conserved regions. The calculation time therefore depends not only on the number and lengths of the sequences, but also on their relatedness. For most sequence families that are sufficiently conserved to be useful for homology modeling, the Alignment/Multiple_Sequence command requires on the order of tens of seconds or less to align the sequences.

There are, however, special circumstances that can result in unacceptably long calculation times. These include:

♦ Distantly related or unrelated sequences.
♦ Highly repetitive sequences.
Step 3: Sequence Alignment

- Greatly differing sequence lengths.
- **Pairwise Threshold** too low.

Even in these situations it is usually possible to obtain a good multiple alignment in a reasonably short time by appropriate adjustment of the parameters. The best strategies, listed roughly in order from most effective to least effective, are as follows:

- Subdivide the problem.
  
  This involves searching for m-blocks over several restricted search zones, each of which covers only part of the full lengths of the sequences. This is especially helpful when the sequences differ greatly in length. In most cases only two or three separate search zones are necessary.

- Reduce the value of the **Num Related Seqs** parameter.
  
  This is especially helpful when not all of the sequences are known to be related. Note that m-blocks of higher dimension will not necessarily be excluded.

- Increase the sensitivity of the search by changing **Mult Scoring Matrix** and/or **Strict_Overlap**.
  
  This reduces the time necessary to find the first m-blocks, so that the problem is internally subdivided sooner.

- Reduce **Dimension Bias** and/or **Not Signif p**.
  
  Although these changes reduce the sensitivity of the search, they generally have no effect on the time required to find the first m-block. Instead they tend to reduce the number of searches done before \( m_{\text{min}} \) is decremented.

- Increase **Minimum Zone Length**.
  
  This speeds the search by making it less thorough. It may result in the omission of some of the shorter and less significant m-blocks.

- Increase **Pairwise Threshold**.
  
  It is rarely helpful to increase **Pairwise Threshold** above the automatically calculated default value. If, however, you enter a **Pairwise Threshold** lower than the default and find that the calculation time is unacceptably long, then using a higher **Pairwise Threshold** will help.
5. Methodology

♦ Use **Single_Search** mode.

**Single_Search** mode can be used, several times and over different search zones if necessary, to find m-blocks quickly. To build a complete alignment this way is tedious, so this is usually the method of last resort.

**Single_Search Mode**

When **Mult Align Mode** is set to **Single_Search**, the **Alignment/Multiple_Sequence** command executes a single segment pair overlap search each time **Execute** is selected. The pairwise score threshold used in this search is the value of the **Pairwise Threshold** parameter. The search rejects any m-block that contains fewer than **Minimum Seq Per Blk** sequence segments. If **Pairwise Threshold** is too high, the search will find nothing. If it is too low, the search will produce many m-blocks, but the calculation time may be excessive. The automatically calculated default **Pairwise Threshold** is usually a good choice for the first search.

In **Single_Search** mode the parameters **Minimum Zone Length**, **Num Related Seqs**, and **Dimension Bias** are inactive because their only purpose is to control repetitive searching in Automatic mode. All other parameters have the same meanings in both **Single_Search** and Automatic mode. In **Single_Search** mode the command must be executed repeatedly, each time with manual adjustments of **Pairwise Threshold**, and perhaps other parameters, to build a complete multiple alignment.

**Single_Search** mode is most useful for difficult alignment problems that involve distantly related or highly repetitive sequences. It is also useful if you only want to find the most significant blocks of a certain dimension (specified by **Minimum Seq Per Blk**), without generating a complete alignment. This is a good way to find similar sequence motifs in otherwise unrelated sequences.

**Manual Mode**

When **Mult Align Mode** is set to **Manual**, the **Alignment/Multiple_Sequence** command creates a single m-block each time **Execute** is selected. You specify the location and contents of the m-block via the **Seq to Align** and **Block Length** parameters. When any of the **Seq to Align** parameters is in focus, the **Mol/Res Pick Level** parameter also appears and is set to **Residue** by default. You
Step 4: Assigning Coordinates Within the SCRs

specify the contents of the m-block in a manner exactly analogous to the way you specify an initial search zone in Automatic mode. You specify the leftmost residue of each sequence segment simply by picking that residue with the mouse in the sequence window while any one of the Seq to Align parameters is in focus. To exclude a sequence from the m-block, select its Seq to Align parameter, press the <Delete> key and then press <Return>. You specify the length of the block via the integer parameter Block Length.

In Manual mode the parameters Zone Length, Mandatory Seq, Minimum Seq Per Blk, Pairwise Threshold, Strict_Overlap, Minimum Zone Length, Num Related Seqs, and Dimension Bias are all inactive because these parameters are only used when searching for m-blocks. In Manual mode there is no searching involved, because you specify the exact location and contents of the m-block. All other parameters have the same meanings as in Single_Search and Automatic mode.

Manual mode is useful for forcing the alignment of residues that may have low sequence similarity, and therefore are not properly aligned in Single_Search or Automatic mode, but that you know should be aligned because of their common biological function in the proteins. In such a situation it is usually best to create as much of the alignment as possible in Automatic mode, then correct any misaligned regions using Manual mode. The Manual mode of the Alignment/Multiple_Sequence command can also be used in this way to correct an alignment produced by the Alignment/Structure command.

Step 4: Assigning Coordinates Within the SCRs

Once conserved regions have been defined, and the sequence of the unknown protein has been aligned in these same regions, it is possible to begin building the model. Choose a particular SCR, and make sure there is a sequence box of the same length that contains both the unknown protein’s sequence and an appropriate reference protein’s sequence. The choice of which reference protein to use for each SCR can be made by comparing the sequence alignment scores found with Manual Sequence Alignment (see page 5-140).
5. Methodology

The **AssignCoords** command in the **Sequences** pulldown is used to transfer coordinates from a reference protein to a model protein in a region defined by a sequence box that you select. The box must contain exactly one reference and one unknown sequence, and it must be frozen. An m-box can be used only if it meets these criteria; since m-boxes often contain more than two sequences, they often cannot be used. You can assign coordinates in a region enclosed by such an m-box by creating a normal SCR box that overlaps the m-box.

You select a box as input to the **AssignCoords** command by picking a residue within it. The selected box turns yellow. If there are several boxes that overlap the picked residue, you can repeatedly press and release the mouse button to select each of them in turn, until the desired box turns yellow.

When **Execute** is selected, the coordinates are copied from the reference protein to the model. If the two proteins have the same amino acids at corresponding positions in the sequence, then the coordinates are simply copied. Wherever they don’t match, the side chains are automatically replaced. When this is done, the side chain atoms that are in common are copied, and for the rest, library amino acids are used and placed along the same direction as the reference protein. When the command has finished, the newly built segment of the model is displayed on the screen, and the corresponding amino acid code letters are promoted to uppercase in the sequence window. A list of steric clashes in the model is reported in the information area and in the textport if the parameter **Bump_Check** is on.

If two or more reference proteins will be used as sources of coordinates in different SCRs, then all the reference proteins must be superimposed beforehand. This superimposition of the reference proteins is extremely important: without it, the resulting model structure will be grossly incorrect. The reference structures can be superimposed either automatically by the **Alignment/Structure** or **Alignment/Multiple_Sequence** command, or from the results of manual structure alignment as described in *Superimposing Reference Proteins Using Manually-Determined SCRs*. 
Step 5: Building Loop or Variable Regions (VRs)

Variable regions (VRs) are sections of the proteins that come between the SCRs in the sequence. They tend to be found at the surfaces of the proteins where differences in amino acid properties and conformation are much better tolerated. They can sometimes be taken directly from one of the reference proteins, but even here they cannot be classified as conserved because the segment is not common to all members of the protein family.

If one of the reference proteins can be used to define coordinates for a variable region, the AssignCoords Sequences command should be used with the Segment Definition parameter set to Designated Loop. But if none of the reference proteins can be used, appropriate coordinates must be found elsewhere.

Searching for and Displaying Loops

The Search Loops command in the Loops pulldown scans the Brookhaven Protein Databank for protein structures of a predefined length that would fit properly into the model protein between two SCRs (see Setting Up the Brookhaven PDB for information about configuring the Brookhaven PDB for use in Insight II). The search is done by comparing the alpha-carbon distance matrix of the flanking SCR peptides with a precalculated matrix for all known proteins that have the same number of flanking residues and an intervening peptide segment of the given length. Ten loops are reported that have the best overall values for RMS differences between the model and database proteins in the flanking, or preflex and postflex, peptide segments.

The command is easily invoked by picking the two sequence boxes on either side of the loop or variable region (VR). The boxes must contain residues from the model protein for which coordinates have been defined (uppercase letters) and must be picked from left to right. Picking the boxes automatically fills all the required parameters for the command. The Start and Stop Residues indicate the residues of the model protein adjacent to other end of the loop that contains the valid coordinates. Flex Residues represent the number of residues within the loop. Preflex and Postflex Res-
5. Methodology

**idues** are the total number of residues of the model protein on either side of the loop. Any or all parameters can be entered by hand, but must be done with caution because an error could be made concerning the identity and number of residues with valid coordinates. If the sum of the preflex, flex, and postflex regions is greater than 40, then the preflex and postflex regions should be shortened as needed. They may not, however, be lengthened beyond the automatically determined values.

When **Execute** is selected, the database is scanned for the best ten loops, and then the **Display Loops** command is automatically activated. This command allows you to examine any of the ten choices at a time. Choose a loop and select **Execute** to view it. This can be done repeatedly. The **Tails** option determines whether the preflex and postflex residues are included when superimposing the loop with the model protein. Toggle it **on**. **Remove** deletes all the loops from the system. The best choice for insertion into the model is the one that has a low RMS and extends away from the body of the protein.

**Generating and Displaying Loops**

The **Generate Loops** command in the **Loops** pulldown should be used when no appropriate loop choices can be found with the **Search Loops** command. This can happen when there are no high resolution proteins similar to the model being built. The side chains of the ten loops found may overlap with the conserved regions of the model or the topologies of their backbones may trace obviously implausible paths. In such cases, the **Generate Loops** command creates a series of ten choices **de novo**. They are not necessarily the “best” loops that this algorithm could create, but they are the first ten loops to have passed all the geometric and algorithmic criteria established by the parameters of the **Generate Loops** command.

As with the **Search Loops** command, the **Start and Stop Residues** are defined as the SCR residues of the model protein at either end of the loop itself. They are easily specified by picking, with the mouse, the sequence boxes to the left and right of the loop region. Note that the boxes must contain residues from the model protein for which coordinates have been defined (represented by uppercase letters) and must be picked from left to right. After specifying
Step 5: Building Loop or Variable Regions (VRs)

the Start and Stop Residues, the Flex Residues parameter is automatically filled in with the number of residues in the loop.

The main chain dihedral angles of the loop, $\phi$ and $\psi$, are initially set to random values. A constrained minimization algorithm then adjust these angles as necessary to make the ends of the loop fit the adjacent SCRs. The Seed parameter is used as a seed to the random number generator that initializes these angles. If the command is run twice with identical parameters, including the same Seed, then the loops generated will be identical in the two cases.

All $\Omega$ angles are set to 180° (trans), except for the peptide bonds on the N-terminal side of proline residues, which can be either cis or trans. When the Pro_Torsions parameter is set to Cis, all prolines in the loop will be cis. Similarly, when it is Trans, all prolines in the loop will be trans. When it is set to Random, the prolines in the loop are randomly made either cis or trans, with a 50% probability of each.

Convergence, Closure Iterations, and Scale Torsions parameters control the loop closure process. As the base distance differences are minimized, the distances approach their ideal values. When the changes from one iteration to the next become small, the RMS difference is compared to the preselected value for Convergence. The smaller the value, the more exact the closure and hence, the larger the amount of computer time required. Closure Iterations specifies the number of cycles permitted before the loop closure process is aborted for any single generated loop. Too small a value here causes all proposed loops to be rejected, but the generation continues indefinitely. Therefore a high value is recommended. Finally, since the algorithm for calculating the changes in $\phi$ and $\psi$ very often yields small proposed values, the Scale Torsions parameter is a multiplicative factor forcing the changes in $\phi$ and $\psi$ to be larger, thus speeding the closure process.

In some situations the Generate Loops command may run indefinitely without converging. This can happen if the loop is too short to span the distance in three-dimensional space between the adjacent SCRs, even when the loop is fully extended. If this happens, interrupt the command by pressing the <Esc> key. A dialog box will appear, asking you to confirm the interrupt request, which you should do. You must then shorten the adjoining SCRs, thereby making the loop between them longer and more flexible. Often only one or two additional residues in the loop will suffice. To
5. Methodology

shorten the SCRs, unfreeze the two boxes that were used as input to the AssignCoords command and shorten one or both of them by one residue so as to make the loop between them longer. Then freeze the boxes and AssignCoords from them again. You should then be able to run **Generate Loops** again.

After the loop has been closed, checked for proper chirality, and superimposed onto the rest of the model protein, it is checked for steric overlaps. This is done in two stages. First, the atoms in the loop are checked among themselves to see whether there are internal steric clashes. Then the atoms of the loop are checked to see if they overlap with any part of the rest of the protein. Parameters are provided to assess each part independently. The **Internal Overlap** and **External Overlap** parameters specify the degree of atomic interpenetration that will be permitted before a violation is reported, and the proposed loop is rejected. They range from 0 to 1, with a higher number indicating a greater tolerance. Since the loops created here are meant to be approximate, and the structures will probably be refined with energy minimization via Discover, large values for **Internal Overlap** and **External Overlap** are suggested. The default values of both are 0.8.

After the **Generate Loops** command completes, the **Display Loops** command is automatically activated. You may choose one or more loops to be viewed simultaneously. After choosing one best loop, select the **AssignCoords Loops** command to transfer the coordinates from the proposed loop choice to the model protein.

The resulting loops will have extended conformations for all the side chains in the loop. Manual rotation, energy minimization, molecular dynamics, or a combination of these methods must be used to obtain a final structure for any loop created with **Generate Loops**.

**Building Coordinates for the VRs**

The **AssignCoords Loops** command is used to copy the coordinates from the chosen loop to the model protein. The command lists the same loop choices as **Display Loops**, with the last loop chosen still indicated and displayed. Selecting **Execute** triggers the command to copy the coordinates from the reference loop to the model protein just as **AssignCoords Sequences** did. As with
Step 6: Conformational Search for Side Chains Using Rotamers

AssignCoords Sequences, side chain replacements are done appropriately.

Step 6: Conformational Search for Side Chains Using Rotamers

On many occasions, it is desirable to explore the possible range of conformations for a certain residue’s side chain or those of a set of interacting residues. There are two circumstances when conformational exploration is commonly done. The first is when the coordinates of a set of residues in the model protein have been taken from a reference protein with different amino acid types. The side chain may not fit properly in the new environment because it is larger or branches differently than the old one. The second is after the Loops/Generate command has been executed to create a loop de novo. That algorithm can only provide extended side chain conformations for the residues in the loop. Extended side chains are never seen in naturally occurring proteins and are only intended to be arbitrary starting points for a more careful conformational analysis.

Depending on how many side chains will be moved, either the Residue/Manual_Rotamer or Residue/Auto_Rotamer command can be used. The former is intended to find the best positions for a small number of residues. The residues are adjusted one at a time, and no attempt is made to deal with side chains that may be intertwined. Each new conformation can be evaluated by checking for improper steric overlap or in terms of its relative nonbond energy. The Residue/Auto_Rotamer command finds the optimum set of conformations for a user-defined list of moving residues. The optimum conformation is defined as the one with the lowest nonbond energy. An iterative approach is used, where residues in the list of moving residues are examined one at a time, according to the order in which they were added to the list.

To use the Residue/Manual_Rotamer command, first set the Manual Operation parameter to Review. Then, choose the residue for which the best conformation will be found by picking it from the screen or sequence display, or by typing its name in the parameter block. To evaluate the relative stability of each proposed confor-
5. Methodology

In the software, toggle the Evaluate_Energy parameter on. The Nonbond Cutoff parameter specifies how close two residues must be before their interaction energy will be calculated. The higher the value, the faster, but less accurate, the calculation will be because fewer interactions will be included. You may also test whether the side chain bumps any part of the rest of the protein by toggling Bump_Check to on. The Overlap parameter specifies the percentage of van der Waals overlap that is permitted before a bump is flagged. The proper range is from 0 to 1, and the higher the value, the fewer the number of bumps that will be found. Proceed by selecting Next or Previous for the Rotamer Choice. A new side chain rotamer is placed on the backbone, either forward or backward in the list in the library for that amino acid type. You may also check the relative stability of the initial conformation by setting the Rotamer Choice to Original at any time.

The output of each execution of the Residue/Manual_Rotamer command can be seen in the information area at the bottom of the screen. The information displayed includes the residue name and number, and a rotamer number that identifies which rotamer entry is currently displayed on the screen. If selected, the calculated single-point nonbond energy, and/or the bumps between the moving residue and the rest of the protein are also given. The best rotamer choice is the one producing the lowest energy or the fewest bumps. You may switch the active residue at any time; the last conformation of the previously selected residue remains in place. If you wish to keep the conformation you have found, select Done as the Manual Operation, fill in the name of the Done Molecule from the value-aid, select Keep as the Update Mode, and select Execute. Selecting Undo for the Update Mode instead returns all altered side chains to their original conformations.

Whenever a new residue is altered by this or the Residue/Auto_Rotamer command, it is added to a list of active rotamer residues. The list contains the identities of the residues and their original conformations. It is maintained until either command is executed with Update Mode set to Done. Thus, you can cancel out of the commands and do other work without completing the conformational search. You can also transfer the list from one command to the other. The usefulness of the latter is described below.

The Residue/Auto_Rotamer command is organized in a similar fashion to the Residue/Manual_Rotamer command in that it is
Step 6: Conformational Search for Side Chains Using Rotamers

divided into separate modes. To begin, the Automatic Operation parameter must be set to Setup. Then, a list of moving residues needs to be specified. Note that this list is the same as that used by the Rotamers/Manual command. New entries to the list are appended when the Activation parameter is set to Add and a residue is picked from the screen or sequence display, or when its name is typed in the Residue Spec parameter. If the Residue Pick Level is set to Subset, then a range of residues can be specified by picking from the list of subsets provided in the value-aid. If a residue or subset is picked, then the command executes automatically. However, you need to select Execute if the residue’s name is typed, or if a residue range is typed. Residues can be deleted from the list when Activation is set to Delete and the command is executed. List pops the textport forward and displays the current list of moving residues. Finally, Clear removes all entries from the list that are currently in their original conformations, but not those for which the conformation has been altered. Thus if a mistake is made adding residues to the list, previously altered conformations will not be destroyed by selecting Clear.

The search algorithm does not test all possible combinations of rotamers for a given list of residues. Instead, an iterative approach is taken where each residue in the list is examined in turn. For each residue, the rotamer choice producing the lowest nonbond energy is found. All library entries as well as the original conformation are tested. A cycle is defined as a single pass through the list of residues in the order in which they were added. After each cycle, the energy is compared to the last one. If it has not dropped appreciably, the search is stopped. Note that the energy can never rise from one cycle to the next because the lowest-energy rotamer is always chosen.

The ability to transfer the active rotamer residue list between the Residue/Manual_Rotamer and Residue/Auto_Rotamer commands is designed to permit a semi-automatic conformational search to be performed. Because not every combination is tried, it is possible that a favorable conformation may be missed. If it is known in advance that a certain combination of rotamers is likely to be the best, as found by the Residue/Manual_Rotamer command, then that can be set up in advance. The residues involved will already be part of the active rotamer residue list, and the specific rotamer combination will be evaluated.
5. Methodology

After the list of moving residues has been properly specified, select Search as the Automatic Operation. Now the Convergence and the Maximum Cycles parameters, that control the termination of the iterative search process, can be specified. The Convergence criterion is satisfied when two successive search cycles have energies that are the same within the specified tolerance. If the Convergence criterion has not been satisfied before Maximum Cycles have been performed, the search is aborted. As with the Residue/Manual_Rotamer command, if you wish to keep the conformation for all the residues that were found, select Done as the Automatic Operation, the appropriate molecule as the Rotamer Molecule, and Keep as the Update Mode, and then select Execute. Selecting Undo instead of Keep returns all the side chains to their original conformations.

On some occasions, it may be found that many of the residues involved in an automatic conformational search remain in their original conformations, but that these conformations are suspected not to be correct. This condition can arise when one of the side chains is locked into an incorrect rotamer by other surrounding side chains. It may be barred from the correct conformation because of very tight packing.

There are two approaches that may alleviate this situation. The first has been mentioned before, namely manually placing rotamers thought to be correct. If they are not known in advance, then it is possible to eliminate the steric problems by temporarily removing the side chains from the protein. Use the Residue/Replace command in the Biopolymer module to replace all the residues to be moved by alanines (except glycines and prolines). Then change the most tightly packed residue back to its original type. Use the Residue/Auto_Rotamer or the Residue/Manual_Rotamer command to find the lowest energy conformation. Continue to replace one or more residues at a time with their original types using the Rotamers commands at each step. The best overall conformation should be the result.

As explained in the Theory section, the Residue/Auto_Rotamer command stores the nonbond energies it calculates in two separate lists. The list of nonbond energies of interaction between moving and fixed side chains (“static” energies) is calculated all at once before the conformational search begins. The list of nonbond energies of interaction between pairs of moving side chains is created
Step 7: Refining the Structure with Discover/CHARMm

and enlarged as the conformational search progresses. Because the stored energies are reused in the later cycles of the search, the early cycles are more time-consuming, and the initial calculation of the static energies is typically the most time-consuming step of all. The algorithm is sufficiently fast, however, that it is feasible to include in the search all side chains of a protein of average size. The calculation time is roughly proportional to the square of the number of active residues. A search involving 100 side chains of elastase takes about six minutes on an Iris 4D-35; a search involving all the movable side chains (191 total) takes about 40 minutes. If necessary, the command can be interrupted at any time by pressing the <Esc> key.

The conformation found by the Residue/Auto_Rotamer command should be energy minimized. This is because the local environment of the side chain is not exactly the same as that for which the statistics were derived. Also, the rotamer χ values used are averages over many different conformations. Therefore, it is recommended that you use the Refine/Relax command or the commands in the Discover module before proposing a final structure.

Step 7: Refining the Structure with Discover/CHARMm

Once steps 3, 4, and 5 are completed, most of the coordinates for the model protein have been defined. However, coordinates for the N- and C-terminal regions may not have been defined. In other regions, the coordinates that were initially assigned may be less than optimal. This is because larger side chains may have been built where small ones existed in the reference proteins, and because the peptide segments that are copied from the various reference proteins may not meet properly at the splice points. Therefore, the final step in the model building process is completing and refining the model protein structure.
5. Methodology

Running Discover/CHARMm with Homology-Built Model Structures

There are two ways to use Discover/CHARMm with protein models built from sequence data. The most convenient way is to use the commands in the Refine pulldown in the Homology module. These commands are specific to the Homology program. When you use this method, all potentials, charges, and hydrogens of the protein are automatically examined and, if necessary, corrected.

The other way is to use the Discover/CHARMm interface by selecting Discover/CHARMm from the Module pulldown. However, in this case, it is up to you to check and correct potentials, partial charges, and hydrogens before submitting a Discover/CHARMm job. You are warned upon leaving the Homology module if there are problems with the model protein. To correct these problems, invoke the Builder module by selecting Builder from the Module pulldown. For hydrogens, use the Hydrogen command in the Modify pulldown. To check and correct potential atom types and partial charges, use the commands in the Force-field pulldown.

Note that a model structure built with Homology may contain unacceptable steric overlaps. If the structure is submitted in this form, the Discover/CHARMm program notices the overlap violations and exits without performing any refinement. It may be necessary to manually refine the structure’s coordinates prior to submission to Discover/CHARMm. Use the Bump command from the Measure pulldown to check for any overlaps. Use a value of 0.8 for the Overlap parameter. If any bumps are found, it is necessary to relieve them by moving atoms with the Torsion command in the Transform pulldown.

The Bump command does not find unacceptable steric overlaps caused by one to three interactions or short bond lengths. If either of these are present, the Discover/CHARMm job cannot run, and an error specifying the overlapping atoms is printed in the .out file. It is necessary to relieve the strain before resubmitting the job. This is done using the Geometry command in the Modify pulldown in the Builder module.

Several of the commands in the Refine pulldown set up and optionally submit Discover/CHARMm runs, either interactively
Step 7: Refining the Structure with Discover/CHARMm

or in the background. Since these runs can be time consuming, it is not practical to do all the required work during one interactive session. Most of the interface between Homology and Discover/CHARMm depends heavily on the subset definitions and internal flags for various parts of the model protein that were created during the course of coordinate assignment. Therefore it is necessary to save these definitions from one session to the next.

The Copy Sequences command is provided to propagate the required information from one version of the model protein to the next as it is refined in a series of successive Discover/CHARMm calculations. If you have used Discover/CHARMm in the batch mode, the refined structure contained in the output file (.cor file for minimizations, .arc file for molecular dynamics) must be read in with the Get Molecule command. Then select Copy Sequences. Enter the name of the original (unrefined) molecule into the Copy FROM Sequence parameter and the name of the newly loaded (refined) molecule into the Copy TO Molecule parameter. The sequence row, all internal residue status flags, and subsets will be copied from one protein to the other. All the commands in the Refine pulldown that rely on this information will now be able to operate properly to set up the next Discover/CHARMm job.

To speed up the calculation, cutoffs are used for the nonbond energy evaluation. Specifically, the value of the Discover/CHARMm constant, cutdis, is set to 7.0 Å, swtdis is set to 1.5 Å, and cutoff is set to 8.0 Å. (Refer to the Discover/CHARMm manual for a complete description of these variables.) You can change the values of any of these parameters by choosing Command File for the Computation Mode parameter and editing the input file before the job is submitted.

End Repair

Coordinates are only assigned to structurally conserved regions and the variable regions between them. If the SCRs do not extend to the N- and C-termini of the protein, then the coordinates will not have been assigned there. This is generally the case because the ends of a protein are often flexible. Since the reference proteins have been deemed unsuitable here, there is no model from which to obtain coordinates, and so the conformations for the end regions must be assigned arbitrarily.
5. Methodology

The EndRepair command allows you to add coordinates by taking amino acid templates from the internal library within Insight II. The conformation given is simply an extended chain. Although this is not optimal, energy minimization can be performed on these regions at a later time.

To use the EndRepair command, select it from the Refine pull-down. When the parameter block appears, enter the object name by picking the object on the screen, typing its name, or selecting it from the list provided. Selecting Execute triggers the command.

Splice Repair

Since the various peptide segments used to build up the model protein may have been obtained from a large number of sources, the junction or splice points between the segments may not be smooth. Even though all the reference proteins have presumably been superimposed using all the SCRs as a guide before coordinate assignment (see Summarizing the Manually-Determined SCRs), and all the loops have been superimposed in the tail regions prior to coordinate transfer, there is no guarantee that the segments meet properly. There is a danger that the peptide bonds do not have the proper length, or that they are not trans. Therefore, there is a provision to repair the splice point peptide bonds.

Whenever coordinates are set with the AssignCoords Sequences or AssignCoords Loops commands, subset definitions are created to keep track of the locations of all the splice points. The SpliceRepair command sets up a molecular mechanics simulation with Discover, placing a torsion force on the omega angle(s) of one or all of these peptide bonds. Most of the atoms of the protein are held fixed during the calculation with the exception of the atoms in close proximity (along the sequence) to the splice. If a particular splice is between an SCR and a loop, then only two residues of the loop adjacent to the SCR are left free to move. If the splice is between two adjacent SCRs with no intervening loop, then only the few atoms surrounding the peptide bond are free to move. These are the alpha-carbons and side chains of the two residues flanking the peptide bond, and the carbonyl and imino groups of the bond itself.
Step 7: Refining the Structure with Discover/CHARMm

The **SpliceRepair** command is used by indicating the splice points to be repaired. One, several, or all splice points can be repaired at one time. These parameters can be filled in by choosing from the lists provided. (Note that when the object is selected, the values of all the splice points’ dihedral angles and bond lengths are calculated and displayed in the textport.) Torsion forcing of the peptide omega bond(s) to 180° is optional as is the force constant (suggested value is 50). Tethering of all moving atoms is also optional (suggested value is 100). You also have control of the minimization method, the maximum number of iterations allowed, and the derivative cutoff. The calculation can be performed with Morse functions, cross terms, or charges, or any combination. As with all Discover jobs run from Insight II, it can be run interactively, in the background, or with only a command file written out to be submitted at a later time.

---

**Energy Minimization**

It is possible that areas of the model protein may be involved in steric overlaps. These may have arisen because a large side chain has been placed in a position where a small side chain existed in the reference protein source, or a peptide segment from one reference protein may be interacting badly with a segment from another reference protein, or perhaps a loop segment may not fit perfectly with the core of the newly built model.

The **Relax** command provides for the selective minimization of sections of the model protein. The classes of regions that can be refined include the N- and C-termini, the variable regions or loops (side chains or all atoms), the side chains in the SCRs that were replaced (mutated) using the **AssignCoords Sequences** command, and all the side chains of the SCRs. Note that the backbone atoms of the SCRs are not one of the classes that can be chosen. This is because experience has shown that altering the conformation of the SCRs only serves to decrease the agreement between the final model and the homologous family of proteins.

The **Relax** command is organized similarly to the **SpliceRepair** command. You can specify one or several classes of subsets to be minimized, and any number and combination of qualifying regions. If **Single** is chosen for **Subset Groups**, a list of the appropriate subsets is displayed. As before, tethering, the type of mini-
5. Methodology

mization, the minimization parameters, and the type of job submission can be specified.

**Molecular Dynamics**

The final step in the structural refinement of the model protein is the exploration of conformational space for the variable regions (loops) using molecular dynamics. The Explore command in the Refine pulldown can accomplish this task. It is designed to perform minimization periodically during the dynamics run and save all intermediate minimized structures in an archive file for later review and analysis.

The Explore command is quite flexible and encompasses many parameters. As before, the object is entered by choosing from the list or picking the object. Only the names of the appropriate variable regions are displayed in the list of subsets in the Subset Name value-aid. Select the parameters for minimization, as well as parameters for molecular dynamics, such as **Initialization Step**, **Temperature**, and **History**. The **Number of Cycles** parameter is the number of combined dynamics and minimization loops that is performed and archive file frames that are saved.

**Step 8: Validating Results**

**ProStat** is a set of tools dedicated to the analysis of peptide and protein three-dimensional structures derived from experimental data obtained from X-ray, NMR, or from modelling procedures (such as Homology methods).

The analysis methods provided include checking of geometrical criteria (such as bond lengths, bond angles, and torsions) against a knowledge base (derived from small molecule crystallographic studies, calculation and tabulation of backbone and sidechain dihedral for single/multiple conformations, calculation of solvent accessible surface area, and calculation of the secondary structure classification).

Many of the properties computed by **ProStat** are per-residue properties, and many of the criteria used for determining the structural
Step 8: Validating Results

integrity are from per-residue spreadsheets. ProStat can either use an existing table or create one automatically as required.

Structure Checking

The ProStat/Struct_Check command provides access to a means of checking the bond lengths, bond angles, torsions, and other geometric properties of a 3D peptide or protein structure which is very easy to use but comprehensive. The aim of this command is to provide an assessment of the geometric correctness of the structure and also to focus the attention of the experimentalist or model builder on problem areas. These problem areas may be regions of an experimentally determined structure where there was a lack of data, or where noise or other experimental difficulties hinder the interpretation of the data.

The bond length and bond angle reference data used in ProStat were derived from a statistical survey (Engh and Huber 1991) of the common amino acids and appropriate chemical fragments extracted from the Cambridge Structural Databank (CSD) (Allen and Kennard 1993). The X-ray structures of compounds found in the CSD are sufficiently small that their molecular parameters can be determined fully from the diffraction data without using a forcefield or other means or referencing the bond lengths and bond angles. The structures drawn from the CSD were used to derive mean and standard deviation values for specific bond lengths and angles. In ProStat, the bond length reference data are stored in a file called pro_bond.dat. The format of this file is illustrated below.

! Information derived from
! (1991),
! File rules ! 1) Later lines take precedence over
! earlier ones with
! atoms of same names.
! 2) Atom name in first column is associated with
! residue name
! in third column.
! 3) Only wildcarding allowed is a single * character
! this implies a match with any residue name
CA C * 1.525 0.021
CA C GLY 1.516 0.018
5. Methodology

<table>
<thead>
<tr>
<th>Bond</th>
<th>Reference Length</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-O</td>
<td>1.231</td>
<td>0.020</td>
</tr>
<tr>
<td>C-B-C</td>
<td>1.530</td>
<td>0.020</td>
</tr>
<tr>
<td>C-B-C-ALA</td>
<td>1.521</td>
<td>0.033</td>
</tr>
<tr>
<td>C-B-C-ILE</td>
<td>1.540</td>
<td>0.027</td>
</tr>
<tr>
<td>C-B-C-THR</td>
<td>1.540</td>
<td>0.027</td>
</tr>
<tr>
<td>C-B-C-VAL</td>
<td>1.540</td>
<td>0.027</td>
</tr>
<tr>
<td>N-C</td>
<td>1.458</td>
<td>0.019</td>
</tr>
<tr>
<td>N-C-PRO</td>
<td>1.466</td>
<td>0.015</td>
</tr>
<tr>
<td>N-C-GLY</td>
<td>1.451</td>
<td>0.016</td>
</tr>
<tr>
<td>N-C-PRO</td>
<td>1.341</td>
<td>0.016</td>
</tr>
<tr>
<td>S-S-CYS</td>
<td>2.000</td>
<td>0.100</td>
</tr>
</tbody>
</table>

Any line beginning with the “!” character is a comment line.

When the **ProStat/Struct_Check** command is run, Insight II first looks for the `pro_bond.dat` file in the current directory. If the file is found, it is used to define the bond length reference data, and the bond length checking proceeds. If no such file exists in the current directory, then Insight II looks for this file in the directory given by `$ROOT/data/macromol` if it finds the file here, it goes ahead with the program, otherwise the bond length checking is not performed. This method allows an expert user to modify the reference values and standard deviations should the need arise, using a local copy of the `pro_bond.dat` file. The `pro_bond.dat` file is extensible, in that any new bonds added to the file are checked by the program. Since **ProStat** is a protein-specific tool, a given monomer in Insight II is only checked if the monomer name corresponds to one of the 20 common amino acid types. The program does not give an error message, but rather simply does not perform the checking. This allows, for example, the protein portion of a protein-DNA complex to be checked for protein geometry distortions without a profusion of error messages relating to non-protein monomer types found in the DNA.

For bond angle checking, **ProStat** seeks a file called `pro_angle.dat`, using the same directory searching sequence just discussed. The format of this file is:

```plaintext
! MSI Protein Bond Angle Table 1
! Created Sept 7 1994.
! Residue Specific bond Angles and standard deviations
```
Step 8: Validating Results

Information derived from

File rules
1) Later lines take precedence over earlier ones with
atoms of same names.
2) Atom name in column 1 is associated with residue name
in column 4, 2 with 5 etc.
3) Only wildcarding allowed is a single * character
this implies a match with any residue name
4) A zero entry implies this specific bond not present

<table>
<thead>
<tr>
<th>Bond</th>
<th>Atom 1</th>
<th>Atom 2</th>
<th>Bond Angle (deg)</th>
<th>Distance (Å)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C N CA * *</td>
<td>121.7 1.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C N CA * GLY GLY</td>
<td>120.6 1.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C N CA * PRO PRO</td>
<td>122.6 5.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA C N * *</td>
<td>116.2 2.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA C N GLY GLY *</td>
<td>116.4 2.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA C N * PRO</td>
<td>116.9 1.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA C N GLY GLY PRO</td>
<td>118.2 2.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA C O * *</td>
<td>120.8 1.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA C O GLY GLY GLY</td>
<td>120.8 2.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CB CA C * *</td>
<td>110.1 1.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CB CA C ALA ALA ALA</td>
<td>110.5 1.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CB CA C ILE ILE ILE</td>
<td>109.1 2.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CB CA C THR THR THR</td>
<td>109.1 2.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CB CA C VAL VAL VAL</td>
<td>109.1 2.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N CA C * *</td>
<td>111.2 2.8 N CA C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLY GLY</td>
<td>112.5 2.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N CA C PRO *</td>
<td>111.8 2.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N CA C PRO GLY GLY</td>
<td>0.0 0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N CA CB * *</td>
<td>110.5 1.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N CA CB ILE ILE ILE</td>
<td>111.5 1.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N CA CB THR THR THR</td>
<td>111.5 1.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N CA CB VAL VAL VAL</td>
<td>111.5 1.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N CA CB ALA ALA ALA</td>
<td>110.4 1.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N CA CB PRO PRO PRO</td>
<td>103.0 1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O C N * *</td>
<td>123.0 1.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O C N * PRO</td>
<td>122.0 1.4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Once again, an expert user can modify or extend the file and the program will work directly with the modified or extended pro_angle.dat file. In order to use modified files successfully, you should note the rules of precedence and format found in the comment lines of these files.
5. Methodology

**ProStat** checks the torsion angles and other properties of a peptide/protein structure using stereochemical parameters derived from high resolution protein structures (Morris et al. 1992). The properties for the model structure are computed on a per residue basis and are compared to the values in Table 5 (Laskowski et al. 1993), where all angles are measured in degrees.

Table 5. Parameters for Torsion Angles

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>phi-psi angles in Ramachandran plot favored regions</td>
<td>&gt;90%</td>
<td></td>
</tr>
<tr>
<td>chi1 dihedral angle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gauche minus</td>
<td>64.1</td>
<td>15.7</td>
</tr>
<tr>
<td>trans</td>
<td>183.6</td>
<td>16.8</td>
</tr>
<tr>
<td>gauche plus</td>
<td>-66.7</td>
<td>15.0</td>
</tr>
<tr>
<td>chi2 dihedral angle</td>
<td>177.4</td>
<td>18.5</td>
</tr>
<tr>
<td>proline phi</td>
<td>-65.4</td>
<td>11.2</td>
</tr>
<tr>
<td>helix phi</td>
<td>65.3</td>
<td>11.9</td>
</tr>
<tr>
<td>helix psi</td>
<td>-39.4</td>
<td>11.3</td>
</tr>
<tr>
<td>chi3 S-S bridge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>right handed</td>
<td>96.8</td>
<td>14.8</td>
</tr>
<tr>
<td>left handed</td>
<td>-85.8</td>
<td>10.7</td>
</tr>
<tr>
<td>Omega dihedral angle</td>
<td>180.0</td>
<td>5.8</td>
</tr>
<tr>
<td>CA virtual torsion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA-N-C-CB</td>
<td>33.9</td>
<td>3.5</td>
</tr>
<tr>
<td>Kabsch and Sander main chain H-bond energy (KCal/Mol)</td>
<td>-2.02</td>
<td>0.75</td>
</tr>
</tbody>
</table>

The **ProStat** data file holding these values is called Pro_misc.dat and is searched for in the same manner as the other data files. This data file contains keywords that specify the property name, along with the mean and standard deviation. This implies that the values in the data file can be modified, but unlike the bond length and bond angle data files, new properties cannot be added to the file so that they are automatically checked by **ProStat**.
Step 8: Validating Results

The CA virtual torsion provides a simple check of the handedness of a chiral alpha carbon center that is not dependent upon the R/S Cahn-Ingold-Prelog notation. Note that for L forms of the 20 common amino acids, all 20 should have a positive virtual torsion around the value of 34 degrees; while for D amino acids the value will be \(-34\) degrees. In the standard R/S nomenclature, the alpha carbons of the common L-amino acids are \(S\), while \(L\)-Cys alone has an \(R\) configuration alpha carbon.

The **ProStat/Struct_Check** command checks for erroneous bond lengths, bond angles, and torsions. The command has three methods for outputting the results:

1. highlighting erroneous bonds
2. listing erroneous bonds to the textport
3. listing the bonds in a per-residue spreadsheet

The per-residue spreadsheet will contain the value of the property as measured in the model structure, the reference value, the difference between the property and the reference value, and the difference expressed as a number of standard deviations. Only those bonds, angles, and torsions whose number of standard deviations exceeds the threshold you specified in the user interface are output. When you specify a table name in the user interface, **ProStat** uses a spreadsheet of that name to output the data. If a table of the name you specify already exists, **ProStat** uses a spreadsheet of that name; otherwise **ProStat** appends the molecule name to the given name and automatically creates a per-residue spreadsheet with the concatenated name. This allows automatic creation of multiple spreadsheets when a wildcard is given for the molecule name and more than one molecule is present in Insight II.

When checking whether a given residue’s phi-psi values fall within the favored regions of the Ramachandran plot, **ProStat** uses discrete 10 degree by 10 degree digitized regions of the phi-psi space (Macarthur and Thornton 1993).

When **ProStat/Color_Molecule** is on, **ProStat** highlights the two atoms defining a bond when it detects an erroneous bond. Similarly, when it detects an erroneous angle it highlights only the central atom of the three atoms defining the bond, while an erroneous torsion causes highlighting of the central two atoms defining that torsion. In the case of the CA virtual torsion check, only the CA
5. Methodology

atom is highlighted if this property is deemed outside the threshold.

**Residue Dihedral Angles**

The **ProStat/Residue_Dihedral** command provides an easy-to-use method of calculating and tabulating peptide and protein-specific backbone and sidechain dihedral angles. The built-in graphing capabilities of spreadsheets are then used to quickly generate 2D and 3D graphs of these dihedrals, and to search for correlations between these and other per residue properties. The calculation and tabulation of each of the individual dihedral angles is controlled by the parameter buttons (Phi, Psi, omega). Once a **Residue_Table_Name** has been given, then the calculated per-residue dihedral angles are stored there, if a spreadsheet of the chosen name exists. If a spreadsheet of this name does not exist then a new one is created, whose name is found by adding the molecule name to the end of the original name. This allows several spreadsheets to be created for individual molecules with a single command.

When the **Assembly** parameter is on, an assembly of conformers of the same molecule, as might be obtained from a distance geometry or simulated annealing calculation, are used in the calculation of the dihedral angles. If the **Dihedral_Value** parameter is on, individual tables are created for each molecule in the assembly, and the selected phi, psi, and other dihedral angles stored in them. If the **Dihedral_Value** parameter is off, then only one spreadsheet is created, and statistics relating to the dihedral angle distribution are stored there. The minimum value, maximum value, and circular variance of the selected dihedrals across the assembly of conformers are computed and stored in the spreadsheet if the parameters **Min_Value**, **Max_Value**, and **Circular_Variance** are turned on.

Since a dihedral angle is a circular rather than a linear measure, it can be inappropriate to average these in a linear fashion—the mean and standard deviation thus derived may be meaningless. **ProStat** therefore uses the circular variance (Mardia 1972) as a measure of the variability amongst a set of dihedral angle values. If a given dihedral angle has the value $\theta_i$ in conformer $i$, then the circular variance is defined as:
Step 8: Validating Results

\[ Var = 1 - \left( \frac{R}{n} \right) \] \hspace{1cm} Eq. 20

where

\[ R^2 = \left( \sum_{i=1}^{n} \cos \theta_i \right)^2 + \left( \sum_{i=1}^{n} \sin \theta_i \right)^2 \] \hspace{1cm} Eq. 21

The circular variance can only vary between values of 0 and 1; a small value indicates a tight clustering of the dihedral angles.

The per-residue spreadsheet tables used in ProStat may be used in creating variable width and blended color ribbons using the Molecule/Ribbon and Molecule/Color_Ribbon commands within Insight II.

**Secondary Structure Classification**

The ProStat/SecondaryClassify command provides a simple means of computing and storing protein secondary structure classifications in a per-residue spreadsheet. When the Classify_Method parameter is set to Kabsch_Sander, the 3D coordinates are used to calculate the secondary structure classification for each residue, with the algorithm of Kabsch and Sander (Kabsch and Sander 1983). In this case, a set of Insight II subsets are also created if the Create_Subsets parameter is on. These may subsequently be used in place of an explicit atom specification for many of the commands within the Insight II program. When the Classify_Method parameter is set to PDB_Classification, then the Insight II program stores the secondary structure information derived from the PDB file (which is currently stored in Insight II) in the spreadsheet. Saving the secondary structure classification in a spreadsheet allows you to easily edit the classification. Finally, the secondary structure classification in the spreadsheet may be utilized in the construction of Richardson-type protein diagrams using the Molecule/SecondaryRender command in the Insight II software.
5. Methodology

Algorithmic Implementation

The solvent accessible surface (SAS) method is available in the ProStat/Access_Surf command. The ProStat pulldown is located in several modules, including NMR_Refine, Homology, and Xsight. The ProStat/Access_Surf command runs a program called hydrosopy in the background that implements the Lee and Richards (1971) algorithm as further refined by Shlake and Rupley (1973). The same hydrosopy program is used in the Solvation module within the Insight II interface, thereby facilitating the comparison of the results between Solvation and ProStat.

United Atom Models

Although Lee and Richards (1971) and Shlake and Rupley (1973) did not explicitly include hydrogen atoms in their model system (they used a united atom model instead), our commercial implementation is more flexible. You can use an all-atom model or you can designate a united atom model. There are two main choices for radii: a VDW radii option with radii from Insight II, or a user-defined radii option that must be used for united atoms. We have also provided for the inclusion of hydrogens (Heavy_Atoms_Only = Off) or their exclusion (Heavy_Atoms_Only = On). For critical applications without hydrogens, you should use a united atom model with the Heavy_Atoms_Only option set to On (alternatively, hydrogen atoms could be given a radius = 0.0, but this method requires unnecessarily-longer compute times).

Setting Atomic Radii

The united atom models for radii can be incorporated into the ProStat/Access_Surf command by specifying a radius file in the Radii_File parameter. The radii file format consists of three fields on each line, separated by white space:

<table>
<thead>
<tr>
<th>atom name</th>
<th>radius (angstroms)</th>
<th>element (optional)</th>
</tr>
</thead>
</table>

The element designation is given within brackets, like [C] for carbon or [!C] for not carbon. For example, the line:

* 2.0 [C]
indicates that all atoms with the atom_name = * (wild card for any
ame) should be given radius = 2.0 if their element type is also car-on. The line:

C 1.5 [C]

indicates all atoms with atom_name = “C” should be given radius
= 1.5 if their element type is [C]. Note that there is a hierarchy to
the file descriptors—in the example above, all “C” atoms will first
have a radius = 2.0, until the second line is read and the radius is
re-set to 1.5 for “C” atoms. It is important to consider this hierar-
chy when designing the file. The more general lines should pre-
cede the more specific lines in the file.

Table 6 shows the united atom model from Shlake and Rupley
(1973) along with the descriptor line for the radii file that is appro-
appropriate:

<table>
<thead>
<tr>
<th>Atom Type</th>
<th>Radius (Å)</th>
<th>User Radii File</th>
</tr>
</thead>
<tbody>
<tr>
<td>all N</td>
<td>1.5</td>
<td>* 1.5 [N]</td>
</tr>
<tr>
<td>all O</td>
<td>1.4</td>
<td>* 1.4 [O]</td>
</tr>
<tr>
<td>all S</td>
<td>1.85</td>
<td>* 1.85 [S]</td>
</tr>
<tr>
<td>all C</td>
<td>2.0</td>
<td>* 2.0 [C]</td>
</tr>
<tr>
<td>carbonyl C</td>
<td>1.5</td>
<td>C 1.5 [C]</td>
</tr>
<tr>
<td>aromatic C</td>
<td>1.85</td>
<td>PHECG 1.85 [C]</td>
</tr>
<tr>
<td>Zn²⁺⁺</td>
<td>0.74</td>
<td>Z 0.74 [Zn]</td>
</tr>
<tr>
<td>water</td>
<td>1.4</td>
<td>not applicable</td>
</tr>
</tbody>
</table>

The radii file format can also be used to specify specific atoms. For
example, if residue number 23 is Phe, and a specific radius = 1.80
is desired for the CG atom, the line would read:

PHE_23:CG 1.80

It is important to realize that the atom_name field must match
exactly. For example, PHE can sometimes be called PHEN, PHEC, or
PHEn depending upon charge and location in the protein. All res-
idue type names need to be present with their own lines in the
radii file (the wildcard ‘?’ is not supported).
5. Methodology

The file format supports the comment character ‘#’ anywhere on the line, and the ‘!’ comment character at the beginning of the line.

A final test of the correctness of the radii file is to run a ProStat/Access_Surf calculation and look at the output file with the name <job_name>.atm. This file contains the atomic radii used by the hydroscopy program. The atom radii in that file should contain the values that you desire for the corresponding atoms. If you detect errors, you can modify (or change the hierarchy of) the lines in the radii file to correct the atom’s radius.

Another example of a united atom model is provided in Table 7:

<table>
<thead>
<tr>
<th>Atom Type</th>
<th>Radius (Å)</th>
<th>User Radii File</th>
</tr>
</thead>
<tbody>
<tr>
<td>all side-chain atoms</td>
<td>1.80</td>
<td>* 1.80†</td>
</tr>
<tr>
<td>main-chain carbonyl O</td>
<td>1.52</td>
<td>O 1.52 [O]</td>
</tr>
<tr>
<td>main-chain amide NH</td>
<td>1.55</td>
<td>N 1.55 [N]</td>
</tr>
<tr>
<td>Cα</td>
<td>1.70</td>
<td>CA 1.70 [C]</td>
</tr>
<tr>
<td>main-chain carbonyl C</td>
<td>1.80</td>
<td>C 1.80 [C]</td>
</tr>
<tr>
<td>Iron heme</td>
<td>0.64</td>
<td>Fe 0.64 [Fe]</td>
</tr>
</tbody>
</table>

†the number 1.80 is for all atoms!

Definition of Computed Surface Areas and their Significance

ProStat supplies three measurements of surface area for your analysis, sampled at the level of individual residues: total surface area, fractional surface area, and polar/nonpolar surface area. Each of these areas will be discussed next.

Total Surface Area

The total surface area of a residue is determined by one run of hydroscopy on the protein. The surface area is always calculated at the atomic level internally, and summed to yield the per residue value. The total surface area of a particular residue is obviously due to two factors: the size of the residue itself (that is, Gly vs. Arg)
and the extent to which the residue is buried in the molecule. It is sometimes possible to compare the total surface area of residues between molecules and get a meaningful comparison, for example between conformers of the same protein. However, the relative surface area is usually a more useful parameter since it involves a normalization that eliminates the effects due to different residue types.

**Relative Surface Area and the Tripeptide Model**

The relative surface area is defined as the fraction (or percentage) of the maximum surface area of a residue that is exposed to solvent. This value goes from 0.0 to 1.0 (or from 0% to 100%) irrespective of the residue type. The value for a residue that is fully exposed is 100%, and the value for a residue that is fully buried is 0%. The formula is simply $A_i / A_{oi}$ where $A_i$ is the area calculated for residue ‘i’ in the current molecule, and $A_{oi}$ is the maximum (or total) surface area for residue ‘i’. Obviously this analysis requires a method for the determination of the maximum surface area $A_{oi}$ for each residue.

It might seem reasonable to use canonical values from the literature for the maximum surface area $A_{oi}$ of the residues (for example, from Shrake and Rupley, 1973), but this idea is impractical in our implementation. The fact that the atomic radii can be defined by the user clearly invalidates the use of a single canonical reference value. Therefore, we have adopted the rigorous treatment of determining the total solvent accessible surface area for each residue in its location in the current protein by directly calculating the surface area using an *in situ* tripeptide model.

The tripeptide model requires a separate hydroscopy run for each residue in the protein; the runs are launched sequentially after the regular run so that your machine is not swamped with processes. The tripeptide model that we have adopted is the Gly-X-Gly model, which has been used previously by Eisenberg and McLachlan (1986) and Shrake and Rupley (1973) (although they used a canonical rather than an in situ model). The tripeptide model for residue X is implemented by mutating the amino acids on each side of X to glycine (thus removing their sidechains but not otherwise changing their coordinates), and then determining the surface area of the central X residue by running hydroscopy on the peptide. Residues at either end of the protein have only one
5. Methodology

neighbor, so that a dipeptide model is used for them (X-Gly or Gly-X). The surface area value from each run is used for the $A_o$ for that particular residue.

**Polar and Apolar Surface Area**

**ProStat** provides the option of analyzing the size of polar to apolar surface of a residue. As noted by Novotny et al. (1984), the apolar surface area is larger in incorrectly folded proteins.

There are two options for determining the polarity of atoms in **ProStat**. One option is to choose to use the Polarity_Source = Default option. This option uses the simple criterion that nitrogen, oxygen, and sulphur (and their hydrogens, if present) are polar, and carbon (and its hydrogens, if present) are apolar (Wesson and Eisenberg, 1992); any other atoms are undetermined (that is, they are given the value “None” in the <run name>.atm file). The second option is to choose Polarity_Source = File, and to specify a file name that contains polarity information in a format similar to the radii file discussed above, and detailed below.

The file format consists of three fields:

<table>
<thead>
<tr>
<th>atom_name</th>
<th>polarity</th>
<th>element (optional)</th>
</tr>
</thead>
</table>

Comment characters ‘#’ can be used anywhere on the line, and ‘!’ at the beginning of the line only. Here are some sample lines with their meaning:

```
!=========================================
* nonpolar [H] # all hydrogens
HN* polar [H] # all hydrogens connected to N
HO* polar [H] # all hydrogens connected to O
HIS:* polar [!C] # all atoms NOT carbon in all HIS residues
HIS_1:OXT none [O] # “none” means exclude from calculation
!=========================================
```

Note that the rules (lines) are applied from top to bottom in the file (like the radii file), so consider to the hierarchy when designing the file. The polarity of the atoms can be checked in the <run name>.atm file that is written when a job is started.

**Limitations in Implementation**

The calculation of **polar** surface area values requires that the molecule be continuously present in the Insight II interface for the job.
Step 8: Validating Results

to successfully finish. If the molecule is deleted, atomic values (rather than residue values) are written to the textport. Because the Insight II software performs the summation over all atoms, and because some polarity information is stored with each atom in the molecule, the same molecule must be available at the end of the job. For this same reason, no other surface area calculation is allowed on that molecule while it is waiting, since the new job could change the attributes stored with the atoms.

Conclusion

If you complete all of the steps described in the order suggested here, the model protein retains much of the structural information contained in the reference proteins.

The refinement procedure ensures that the protein has proper physical and chemical properties. The overall character of the homologous family is maintained through the use of structural constraints.

Libraries of structurally conserved regions for many families of proteins can be built with repeated use of the Homology module. These libraries can be saved and used again to speed the early stages of future projects.
5. Methodology
The tutorial lessons, both online and hardcopy, provide you with a series of 13 lessons designed to teach you how to use the Homology program.

♦ The first lessons familiarize you with the concepts of homology modeling and the Homology program’s interface.

♦ You learn how to save and retrieve sequences and sequence alignments.

♦ Next, you build the structure of a protein (the *unknown* or *model* protein) based on structural and sequence homology to other related proteins (the *reference* proteins).

♦ Another lesson demonstrates how to find proteins related to the *model* protein in a database of protein sequences.

♦ You find the structurally conserved regions (SCRs) of two proteins automatically.

♦ You learn the use of the **Alignment/Multiple_Sequence** command and investigate various alignment strategies.

♦ You learn how to use the sequence searching database tools to find other sequences similar in sequence to the *model* protein.

♦ The next lesson shows an example of systematic conformational searching using the **Residue/Auto_Rotamer** and **Residue/Manual_Rotamer** commands.

♦ Finally, you analyze a protein’s hydrophobicity profile and predict its secondary structure.
6. Tutorial

**Hardcopy and Pilot online tutorials**

To access the online tutorials for Homology, click the mortarboard icon in the Insight II interface.

When the Pilot interface appears, click the open file button (the leftmost button at the bottom of the window).

Then, from the Open Tutorial window, select *Homology tutorials* and choose from the list of available lessons (note that some of the lessons are not available as Pilot online tutorials, but only as hardcopy in this manual):

**Lesson 1** - Demonstrating the Homology Sequence Alignment Window
**Lesson 2** - Saving and Retrieving Sequences and Sequence Alignments
**Lesson 3** - Building a Protein with Homology: An Overview
**Lesson 4** - Finding SCRs/Building SCRs and Loops (User documentation only)
**Lesson 5** - Structural Refinement
**Lesson 6** - Sequence Database Searching
**Lesson 7** - Multiple Structure Alignment
**Lesson 8** - Pairwise Sequence Alignment
**Lesson 9** - Multiple Alignment of Serine Protease Sequences
**Lesson 10** - Finding Alternative Multiple Sequence Alignments (User documentation only)
**Lesson 11** - Multiple Alignment of Distantly Related Sequences
**Lesson 12** - Manipulating Side Chain Conformations With Rotamers
**Lesson 13** - Hydrophobicity and Secondary Structure Profiles
**Lesson 14** - Generating Protein 3D Structures using Modeler
**Lesson 15** - Analysis of Modeler Results
**Lesson 16** - Motif Database Searching

For a more complete description of Pilot and its use, click the online help button in the Pilot interface or see the introduction chapter in the Insight II manual.
Lesson 4a: Finding structurally conserved regions

In this lesson you locate areas of three reference proteins that are structurally conserved. This is done by comparing the backbone coordinates of the reference proteins with an iterative RMS procedure. The topics covered in this lesson are:

♦ Reading in reference proteins from the Brookhaven database.
♦ Displaying the amino acid sequences for the reference proteins.
♦ Displaying only the relevant residues.
♦ Using the mouse in the sequence window to move the amino acid sequences back and forth, insert and delete gaps, move residues from one end of a gap to another, and color residues on the screen.
♦ Creating sequence boxes to compare structural and sequence similarities of the reference proteins.

1. Invoke Insight II

At the system prompt enter the following command:

> biosym_tutorial -i

The status of your tutorial directory appears and you are asked for the
6. Tutorial

*product names you wish to install.*

If the listed status of Homology is “Not Installed”, type:

```
>homology
```

and press the <Enter> key to install Homology. Otherwise, proceed to the next step.

The tutorial script creates a new tutorial directory under your home directory. Necessary files are copied into the tutorial directory.

Press <Enter> again.

A list of options appears on the screen.

Enter 3 and press <Enter> to change to the tutorial directory.

The list of options appears again.

Enter 1 and press <Enter> to start Insight II. After Insight II is loaded, go to the MSI icon in the upper left of your window and select the **Homology** module.

2. Load the Reference Proteins

First you must find areas of the reference proteins that are nearly identical in structure. This is more reliable than using sequence similarity because two proteins with similar sequences do not necessarily have similar conformations, and two proteins with different sequences may have similar conformations. Therefore, it is best to use two or more reference proteins whenever possible in order to find areas of structural
conservation.

Select the Molecule/Get command. To read in files from the Brookhaven database, choose PDB as the Get File Type parameter. The File Name value-aid appears to the right of the parameter block.

Scroll down the list of files in the value-aid and select pdb1f19.ent. The file name appears in the File Name parameter box and F19 appears in the Get Molecule parameter box. Select Execute.

The mouse Fab fragment R19.9 appears on the screen.

Now, while still in the Molecule/Get command, choose pdb1fbj.ent from the list of files in the value-aid and select Execute.

The mouse Fab fragment J539 appears on the screen.

Repeat the Molecule/Get command once again, but this time select pdb1mcp.ent from the list of files to read in. Select Execute.

The mouse Fab fragment MC/PC603 appears on the screen.

All three of the reference proteins should be displayed on the screen.

3. Color the Reference Proteins

Now color the reference proteins so that they can be referred to more
Select the Molecule/Color command. Select F19 from the Molecule Specs value-aid (F19 automatically appears in the Molecule Spec parameter). When the Color Palette appears, pick magenta and then Execute.

Now, while still in the Molecule/Color command, select the Color parameter and pick red from the Color Palette. Select FB) from the Molecule Spec value-aid. The command triggers at this point.

Remain in the Molecule/Color command. Select the Color parameter again and this time pick yellow from the Color Palette. Select MCP from the Molecule Spec value-aid. The command triggers at this point.

The F19 molecule is colored magenta, FB) red, and MCP yellow.

4. Display the Amino Acid Sequences for the Reference Proteins

Next, you need to display the amino acid sequences for the reference proteins.

Select the Sequences/Extract command. Choose F19 from the Objects list. The name F19 appears in the Extract Molec Name parameter, and the command is triggered.

Alternatively, you could have picked one of the protein molecules on the screen, or you could have extracted sequences for all of the proteins by typing * in the Extract Molec Name parameter box.
Note that the **Sequences/Extract** command is a trigger command, which means the command is automatically executed and you do not need to select **Execute**.

A new window appears that is labeled **Amino Acid Sequences**. Contained within the window is the sequence for F19, wrapped between several rows. The window also has controls to alter the appearance of the sequence, such as **Color by** and **Font**, scroll bars, and mode buttons. Note also that the 3D display has been altered, in that hydrogens have been added to the protein to permit subsequent coordinate transfer to the amino acids of the unknown protein's sequence, but they have not been displayed. In fact, the display of protein F19 changes to an alpha-carbon trace to aid in the visual structural alignment.

Now display the amino acid sequence for FBJ.

---

While still in the **Sequences/Extract** command, select **FBJ** from the **Objects** list. The name **FBJ** appears in the **Extract Molec Name** parameter, and the command is triggered.

---

Remain in the **Sequences/Extract** command, this time select **MCP** from the **Objects** list to display the sequence for MCP.

---

You should now have three sequence rows in the sequence window. The rows are arranged in blocks of aligned segments. The rows are drawn from the top down in the order that they were extracted, and the color of each row matches the color of the corresponding protein on the screen: F19 is at the top in magenta, FBJ is in the middle in red, and MCP is at the bottom in yellow.

5. **Display the Relevant Residues**

Because the human Bence-Jones protein REI has only 107 residues, you want to display only the relevant residues (the first 107 or so) of
6. Tutorial

the reference immunoglobulins.


It may be necessary to move the proteins or increase the slab thickness to get all of the objects on the screen without clipping. The dials (located in the lower corner of the screen) should already be connected to the world, so use the mouse on the dial icons to translate the molecules in X, Y, and Z until they are centered. Then, select the slab thickness function key <F12> and use the mouse to increase the thickness until there is no clipping.

6. Scroll the Sequence Display

The sequence display can be manipulated with the mouse to achieve the optimum alignment. Although manual alignment methods will be demonstrated here, automatic procedures also exist to alter the alignment.

Place the cursor anywhere in the middle row of sequences and press and hold down the middle mouse button. The identity of the touched residue appears in the Residue ID area of the sequence window. Move the mouse to the left.

The middle sequence row (FBJ) moves left to follow the cursor

While continuing to hold down the cursor, move the sequence back to the right.

The sequence display should be back to its original alignment; all three left residue counters should once again display residue L1 and the right residue counters should display L66.
For replay purposes, all scrolling operations are recorded in your .log file.

Sequence scrolling cannot be used when you are in the middle of setting up a command. You might want to do this to locate a residue that is currently not displayed because it is above or below the top or bottom of the sequence window. However, the following example illustrates how the same thing can be done while the Molecule/Color command is being set up. The task is to color FBJ:L132 yellow. That residue is currently not visible in the sequence window.

Select the Molecule/Color command. Select Monomer/Residue as the Molecule Pick Level. Select the Color parameter and choose yellow.

Place the cursor at the topmost edge of the sequence window, and press and hold the left mouse button. While holding the button down, enlarge the window upwards by moving the mouse up. Release the mouse button.

The window is resized, and more blocks of the sequence alignment are visible. Residue FBJ:L132 is the rightmost residue in the middle row of the second aligned block.

Now, pick the residue and Execute to complete the operation.

The residue turns yellow.

Select Cancel to exit the Molecule/Color command.

7. Inserting Gaps
6. Tutorial

In order to insert or delete sequence gaps on either side of a residue, the sequence window must be in **Seq Mode**, as it still should be. And instead of using the middle mouse button, you use the left and right buttons.

Use the left mouse button to pick the middle S in the sequence **ASSSV** in the middle of the red sequence row (FBJ) in the topmost sequence block. Hold the button down, and drag the mouse to the left three residues.

FBJ:SER L27 appears in the information area at the bottom of the window, and three cyan gap symbols (- - -) appear in the sequence row to the right of the residue.

8. Deleting Gaps

Pick the same S (FBJ:L27), which is now in the string **ASSSV**, with the left mouse button. Hold the button down, and drag the mouse back to the right three residues. Any further movement causes a beep.

All three gaps symbols are deleted, and the sequence string **ASSSV** is restored.

9. Moving Gaps

Just as sequence rows were moved with the middle mouse button, so too are sequence gaps. First, re-establish the gap.

Pick the rightmost S in the sequence **ASSSV** with the left mouse button, and drag the mouse to the left two residues.

The alignment should look like this:
As you drag the mouse one residue, the alignment will look like this:

\[
\begin{array}{cccccccc}
F19 & S & Q & I & I & S & N & Y & L \\
PB1 & S & S & S & V & & & & \\
MCP & S & Q & S & L & L & N & S & G
\end{array}
\]

As you continue to drag it to the right another residue, the gap continues moving to the right, and the alignment becomes:

\[
\begin{array}{cccccccc}
F19 & S & Q & I & I & S & N & Y & L \\
PB1 & S & S & S & V & & & & \\
MCP & S & Q & S & L & L & N & S & G
\end{array}
\]

To move the gap back to the right, you simply drag the mouse in the opposite direction.

The display should be the same as it was before you began shifting the gaps:
Finally, delete the gap region.

The residue counters for all three rows should be back at the original display of L1 and L66 (if you have not changed the width of the sequence window).

10. Color a Residue or Molecule Using the Sequence Display

Select the Molecule/Color command. Select Monomer/Residue as the Molecule Pick Level parameter. Pick the central S in the sequence ASSSV of FB, and FBJ:L27 appears in the Molecule Spec parameter. In the Color Palette value-aid, select the cyan box, then select Execute.

Both the sequence letter and the residue are now cyan. The color of the sequence letters is always the same as the alpha-carbon of the residue as long as the Color by mode is set to C-alpha.

While still in the Molecule/Color command, color the whole FB molecule cyan. Choose Molecule as the Molecule Pick Level. Pick any letter in the middle row (FB).

11. Create Sequence Boxes Manually
To find areas of structural homology, you need to compare limited regions of the reference proteins. By creating sequence boxes around letters in the sequence display you can conveniently focus on a single pair of peptide segments at a time.

Besides the manual method discussed here, an automatic method is also available. It is discussed in great detail in lessons 3 and 7 (available as Pilot online tutorial lessons).

Make sure that the sequence window is in **Box Mode**. This is indicated in the bottom of the window. Click with the left mouse button if it is not. Place the cursor over the second letter (I) in the top row (F19), press and hold the left mouse button: **F19:ILE L2** appears in the **Residue ID** area of the window.

At the same time, a yellow **prompt box** appears around the sequence letters for the residue that was picked.

While still holding down the left mouse button, drag the mouse down to cover the second letter (I) in the middle row (FB1), and **FB1:ILE L2** appears in the **Residue ID** area. Release the mouse.

Notice that as you drag, the prompt box stretches to follow the cursor, and when you release the button, the prompt box is replaced with a sequence box (green) placed around both of the included residues. You could have made the box any size when you were dragging. Only pre-aligned residues can be enclosed using this method.
6. Tutorial

Now begin moving the newly created box.

Pick Box from the Mode selector with the left mouse button. Place the cursor anywhere within the box (on either I) and press and hold the right mouse button. Drag the mouse to the right. As the right edge of the box is reached, the edge moves with the cursor. Continue to move the edge two residues, expanding the box by that amount. Release the mouse button.

Place the cursor inside the box again and press and hold the middle mouse button. Drag the mouse to the right, and the whole box moves. Move it only one residue. Release the mouse button.

Place the cursor anywhere in the center of the box and press and hold the right mouse button. Drag the mouse to the left. As the left edge of the box is reached, the edge begins to move. Continue to move the edge left to the N-terminus of the protein. (If you try moving the mouse any further, you will hear a beep.) Release the mouse button.

Finally, maneuver the box to enclose only the second and third residues of the (FBJ) and top (F19) rows by moving the left and right edges of the box. Remember to release the button when switching between left and right edges.

The sequence display should look like this:

```
F19 D I V M T Q P S S L S V S A G E R V I M S C K S Q S L L N
F B J E L V L T Q P A I T A A S L G Q K V T I T C S A S S S V S S
M C P D I V M T Q P S S L S V S A G E R V I M S C K S Q S L L N
```

12. Comparing Structural Similarity

When you have an active sequence box (that is, when a sequence box is green), you can continuously monitor either the structural or sequence similarity of the two peptide segments indicated by the box.
Hardcopy lessons

For structural similarity, the RMS deviation of the backbone atoms in the peptide segments is calculated. At the same time, the two peptide segments are superimposed.

Structural similarity is recalculated whenever the contents of the box are changed. This happens every time you move a single sequence row through a box in Seq Mode, or when you expand, contract, or move a box. The structural or sequence similarity (or both) is determined, depending on whether or not the monitoring mode has been selected. The calculation is repeated every time any of these actions happen.

Make the molecular display less cluttered for the next operation by selecting the Object/Blank command. When the parameter block appears, choose F19 from the list, the command automatically triggers, and the magenta protein disappears.

The Blank command turns off the display of the entire molecule without losing the information about which atoms were on or off.

Now, make a new sequence box.

With the left mouse button, click and drag between two of the cysteine residues in the sequences that are vertically aligned in the middle (FBJ) and bottom (MCP) rows of the sequence display. In the bottom sequence row, the C is in the string MSCKS. FBJ:CYS L23 and MCP:CYS L23 should appear in the parameter boxes. Release the mouse button, and a new box is created.

Notice that the original sequence box has turned cyan. This means it...
6. Tutorial

is no longer active. Only one box at a time can be active.

Pick any residue inside the original box with the middle mouse button, and notice that the box turns yellow. Drag the mouse to the left, and the box moves to the left. It also turns green, indicating that it is now the active box. Drag the mouse to the right to move the box back to the right.

Turn structural monitoring on by selecting the Alignment/Structure command. When the parameter block appears, select Manual as the Struct Align Mode parameter, and then select Execute.

Now expand the active sequence box (green) to the right by using the right mouse button to drag the right edge of the box. Continue dragging to the right until the box is three residues wide.

No RMS calculation is performed until there is a minimum of three pairs of residues in the box. Then, an RMS calculation is performed and residue labels appear on the two proteins along the peptide segments indicated by the sequence box (L23, L24, and L25 for both proteins). The RMS value reported in the information area should be 0.280.

Drag the right edge of the box to the right one more residue, for a total of four pairs in the box.

The labels change, and a new RMS value of 0.293 is reported.

Note that an RMS calculation on sequence comparison is only done for residues within the active sequence box. At this point, the sequence
display should look like this:

<table>
<thead>
<tr>
<th>F19</th>
<th>DIOMTQTSSLASLGDRTISCRASQDISN</th>
</tr>
</thead>
<tbody>
<tr>
<td>FBJ</td>
<td>EVLTPAIITAALSGLKVTITCSASSVSS</td>
</tr>
<tr>
<td>MCP</td>
<td>DIVMTQPSSLSSAGERVMSCKSSQSLLN</td>
</tr>
</tbody>
</table>

13. Perform a Sequence Comparison

For sequence similarity, a user-selectable scoring matrix is used to calculate how closely matched the two sequences are over the region.

Select the Alignment/Pairwise_Sequence command. In the parameter block, select Manual as the Seq Align Mode parameter. A new sequence similarity score is calculated each time the contents of the active box change. Use the default parameters of Mutation for the Scoring Matrix and 10.00 for Matrix Scaling and select Execute.

Expand the box to the right three more residues (seven total).

The sequence display should look like this:

<table>
<thead>
<tr>
<th>F19</th>
<th>DIOMTQTSSLASLGDRTISCRASQDISN</th>
</tr>
</thead>
<tbody>
<tr>
<td>FBJ</td>
<td>EVLTPAIITAALSGLKVTITCSASSVSS</td>
</tr>
<tr>
<td>MCP</td>
<td>DIVMTQPSSLSSAGERVMSCKSSQSLLN</td>
</tr>
</tbody>
</table>

Both RMS values and sequence similarity scores are reported in the information area. The sequence similarity score reported in the information area is 28.000, 26.667, and 25.714. The larger the score the better.

Now see what happens when this alignment is disrupted by scrolling.
6. Tutorial

one sequence relative to the other.

Change to **Seq Mode** by picking the selector at the bottom of the sequence window.

Click and drag on any residue in the bottom sequence row (MCP) with the middle mouse button. Drag the bottom sequence to the left by one residue. The sequence similarity score shown in the information area decreases to only \(-2.857\).

Realign the sequences by dragging the bottom row (MCP) to the right one residue.

**Turn off sequence monitoring, so that only RMS scores are reported.**

Select the **Alignment/Pairwise_Sequence** command and set the **Seq Align Mode** to **off** and select **Execute**.

14. Finding a Structurally Conserved Region (SCR)

Pick the **Box Mode** selector button to turn it **on**. Enlarge the box by two residues: drag the right edge of the box to the right two residues so it encloses nine pairs of residues.

Notice that the RMS becomes quite large (more than 2.66), and the overall orientation of the two proteins is quite poor. This indicates that
Hardcopy lessons

the box is beyond the boundaries of the structurally conserved region.

Drag the right edge of the box to the left until there are only four pairs of residues in the box.

Note that although you may start the mouse dragging at any point in the box, the right edge will only begin to move when the cursor is adjacent to the edge.

The orientation has improved, and the RMS score has dropped to 0.293 Å again.

Now expand the box to the left until there are nine pairs of residues in the box.

The sequence display should look like this:

```
P19 D1QMTQSLAASLGDRVTISCRAQDISN
FBJ EIVLTTPAISSLGQKVTLCTSAASSS
MCP DIVMTQPSLVSAGERVIXSCKSSQSL
```

The final RMS value is 0.480 Å. The SCR terminates here; this is primarily because the ends of the beta strand were reached. Expanding the box further results in a higher RMS value as well.

15. Freeze the Sequence Boxes

Once you have decided that a certain sequence box encloses the proper residues to define an SCR, it is always a good idea to freeze the box so it cannot be moved, and so the contents of the box cannot be inadver-
Select the **Boxes/Freeze** command. Pick any letter inside the green box. The box turns yellow to indicate it has been selected. Select **Execute**.

*The box is frozen and turns red.*

Select **Cancel** to exit the command.

*You could also have used the mouse to freeze the box. To demonstrate, unfreeze it and freeze it again.*

Select the box with either the middle or right mouse buttons. It turns yellow. While holding down that button, tap the left mouse button, and the box turns cyan. Now select it again and tap the left button. It turns red.

*Now that the box is frozen, it is impossible to move the box relative to the sequences inside.*

Switch to sequence scrolling mode by picking the **Seq Mode** button at the bottom left corner of the screen.
Pick and drag the bottom sequence row (MCP) to the left one residue with the middle mouse button.

*You would expect only the bottom row to move, but instead, the top and middle rows both move left. The program did this in order to move the bottom sequence as you requested without disrupting the contents of the frozen box.*

*If there are many frozen boxes between several combinations of the*
proteins, the program resolves all of the interdependencies of all the rows. As many rows move as are required to keep all the frozen boxes intact.

16. Manipulating Gaps

Pick the first L in the sequence QSLLN just to the right of the red box. The specification MCP:LEU L29 appears in the information. Make sure the window is in Seq Mode.

Open a gap to the right of the L by pressing and holding down the right mouse button and dragging the mouse to the right.

Open a gap to the left of the L by pressing and holding down the left mouse button and dragging the mouse to the left.

Notice that all of the middle row moves to the left.

Drag the mouse more.

Notice that the program does not allow any disruption inside the frozen box.

Return the display back to its original state by picking the L with the left mouse button and dragging it to the right and then picking it with the right button and dragging it to the left.

17. Finding a Second SCR

Often it is easier to locate SCRs by looking at the protein structures on the screen. Because the protein residues are linked to the letters in the sequence display, a sequence box can be initialized just as easily by picking the residues on the protein itself.
6. Tutorial

To make it easier for you to see what is happening, color two of the residues in the proteins where the next SCR will be found.

Select the Molecule/Color command. When the parameter block appears, set the Molecule Pick Level to Monomer/Residue. Pick the L in the sequence NFLAW in the right side of the bottom row (MCP) of the sequence display. MCP:L39 appears in the Molecule Spec parameter. Pick the magenta box from the Color Palette value-aid and then Execute.

The letter and residue are colored magenta.

Pick the L in the sequence SSLHW in the middle row (FBJ); the Molecule/Color command is triggered again and colors FBJ:L32 magenta.

Note that both residues are already physically close to one another. You may have to rotate the molecules slightly to see them better.

Now select the Boxes/Initialize command. Instead of picking the residues from the sequence display, this time pick both magenta residues on the screen.

Yellow prompt boxes appear around letters in the sequence display to indicate the residues you picked. Notice that the two residues are not already vertically aligned. It is obvious that you need to insert a gap in one of the sequences or scroll one to align the two letters.

Select Execute in the Boxes/Initialize parameter block.

The program notices that the two letters are not aligned and that it is not possible to automatically scroll one sequence in relation to the
other without disrupting the contents of a sequence box. (In this situation, it does not matter whether a box is frozen or not. No boxes will be disrupted.) A warning is issued that a gap must be inserted.

Select **Done** in the error window.

At this point, an invisible command called **Gaps/Scroll** automatically becomes active with all parameters set in the parameter block except the gap position. The number of gap symbols to insert is set to 7, which is the difference in length of the two proteins in this region.

Also, a possible region for the gap insertion is indicated by the suggestion box (also in yellow) across a portion of the middle row.

While in the **Gaps/Scroll** command, pick the third S in the sequence ASSVS5SS in FBj, and **FBj:L28** appears in the Insert Delete Point parameter. Select **Execute**.

The prompt boxes and suggestion box disappear, the gap is made to the right of FBj:L28, and a new sequence box is made.

The structural monitoring mode is still in effect. It is now possible to find the limits of the SCR. This is left for you as an exercise. (Refer to steps 12, 13, and 14.)

**18. Quit Insight II and Homology**

To exit Homology, type **quit** at the command prompt and press <Enter>. To confirm, press <Enter> again.

The preceding section demonstrated the technique for finding SCRs manually. They can also be found automatically. Refer to Lesson 10: Finding alternative multiple sequence alignments (see page 6-233).
6. Tutorial

Lesson 4b: Building SCRs and loops

In this lesson, the sequence of the unknown protein is aligned to the conserved regions of the reference proteins and coordinates are assigned based on that alignment. Loop insertions are made between the conserved regions. The concepts covered in this lesson are:

- Finding consensus SCRs.
- Defining subsets.
- Placing the reference proteins in the same coordinate frame.
- Reading the unknown protein’s sequence and aligning it with the sequences of the reference proteins.
- Assigning coordinates to SCRs.
- Assigning coordinates to a loop region.

1.Invoke Insight II

At the system prompt enter the following command:

> biosym_tutorial -i

The status of your tutorial directory appears and you are asked for the product names you wish to install.

If the listed status of Homology is “Not Installed”, type:

> homology

and press the <Enter> key to install Homology. Otherwise, proceed to the next step.

The tutorial script creates a new tutorial directory under your home
directory. Necessary files are copied into the tutorial directory.

Press <Enter> again.

A list of options appears on the screen.

Enter 3 and press <Enter> to change to the tutorial directory.

The list of options appears again.

Enter 1 and press <Enter> to start Insight II. After Insight II is loaded, go to the MSI icon in the upper left of your window and select the Homology module.

2. Read In the File Containing SCRs From the Reference Proteins

Select the File/Restore_Folder command. Select the Folder Name parameter to advance you to the list of files in the Folder Name value-aid. Pick homology_part2.psv. This file contains all the SCRs for the three immunoglobulins used in Lesson 4a. Select Execute.

The three proteins appear on the screen, and the amino acid sequences and all the SCRs for this family of proteins appear in the sequence window.

Press function key <F11> (connect world).
6. Tutorial

You can now move the objects on the screen as one unit using the mouse or dials.

3. Finding Consensus SCRs

Look at the sequence display. For every SCR along the protein sequences, there are three overlapping red sequence boxes. Each box includes a pair of proteins, that is, protein 1 and protein 2; protein 2 and protein 3; and protein 1 and protein 3. Since the boxes are all the same color and share some boundaries with each other, it is difficult to visually differentiate them.

To help see each box, select the Boxes/UnFreeze command. Pick the I in the middle sequence (FBJ) KVTITCSASSS repeatedly. A different box turns yellow after each pick. You may also step through the boxes by picking a residue in the box with the middle or right mouse button repeatedly. (The sequence window must be in Box Mode.)

Notice that the box containing proteins 1 (F19) and 3 (MCP) appears to include protein 2 (FBJ) as well. This is not the case; sequence boxes can only include two sequences. This has an effect when you are scrolling through a frozen box. For instance, if there is a frozen box containing only the top row and the third row, it is still possible to scroll the row in between because it is not really a member of the box.

Notice that the sequence boxes are not the same width in any one region. This is because the conformations of the proteins diverge at slightly different places along the peptide chain. Since the model you are building must be reliable, it must only be comprised of components that are the same in all known members of the protein family. Therefore, you must be able to find residues that are involved in SCRs found between all combinations of the reference proteins.

To do this, select the Boxes/Summarize command. When the parameter block appears, accept the Action parameter default value of Create, and select Execute.
Summary boxes appear for each consensus SCR. These boxes indicate the aligned regions where all proteins in the family are expected to be structurally conserved. Although these boxes are white and not red (like the boxes which are frozen using the Boxes/Freeze command, or by clicking the right and left mouse buttons simultaneously), summary boxes are automatically frozen and cannot be unfrozen.

Note: If you ever need to change the summary boxes, unfreeze the boxes that were used in their making, move those boxes to better positions, and then use the Boxes/Summarize command again. The old summary boxes will be deleted and replaced by new ones. To delete the summary boxes without making new ones, execute the Boxes/Summarize command with the Summarize Operation set to Delete.

4. Subset Definitions

To help keep track of where the SCRs can be found, the program creates subset definitions when summary boxes are made. These subsets are named protein_name$SCRn for each region and protein_name$SCR for the union of all the individual regions for each protein.

To list these subsets, select the Subset/List command. When the parameter block appears, set the Collection Level parameter to Monomer/Residue and choose F19$SCR1 from the Subset List.

The textport automatically appears, and the definition of the subset, F19:L3-L6, is given.

Now choose F19$SCR, and the definition for all the SCRs in the protein is given.

Notice that there are 10 ranges of residues, corresponding to the 10
6. Tutorial

SCRs found by the program.

Toggle function key <F9> to hide the textport window.

5. Color a Subset

You can use these subsets as input to other commands, such as the Molecule/Color or Molecule/Label commands.

Select the Molecule/Color command. Choose Subset from the Molecule Pick Level parameter, and all the newly created subsets appear in the Molecule Specs value-aid list. Select F19$SCR. Now select the yellow box from the Color Palette, and then select Execute.

The amino acid sequence letters in the top row that are inside the summary boxes turn yellow along with the corresponding residues on the screen.

Scroll down the subset list and select FBJS$SCR, and those regions turn yellow. Finally choose MCP$SCR from the list. Now select Cancel.

6. Place the Reference Proteins in the Same Coordinate Frame

Before you can assign atomic coordinates to a new sequence, you must align the structures of all the reference proteins in space. If coordinates are assigned to adjacent peptide segments from different reference proteins without first aligning the proteins to a common reference coord-
In the coordinate system, there may be misalignment at the junctions.

Select the **Transform/Superimpose** command. When the parameter block appears, select **Subset** as the **Molecule Pick Level** parameter, and **Backbone** as the **Superposition Mode** parameter. From the list of subsets in the **Molecule Specs** value-aid, choose **FBJ$SCR** as the **Source Spec** and **F19$SCR** as the **Target Spec**.

This command automatically executes because it is a trigger command, so you do not have to select **Execute**.

Toggle **End Definition** to **on** and select **Execute** again.

The program calculates the RMS deviation of the backbone atoms over all the SCRs and superimposes FB1 onto F19 in those areas. The RMS value, shown in the information area at the bottom of the screen, should be 1.013 Å.

Now, select **MCP$SCR** as the **Source Spec** and **F19$SCR** as the **Target Spec**. After the command automatically triggers, toggle **End Definition** to **on**, and select **Execute**.

The SCRs of MCP are superimposed onto F19 with an RMS deviation of 0.971. Notice how well the proteins are aligned. It shows how structurally homologous they are.

Note that if the automatic structural alignment method had been used, the superimposition would have been done automatically.
6. Tutorial

7. Read in a Sequence File

Select the **Sequences/Get** command. When the parameter block appears, choose `rei.seq` from the list in the **Files** value-aid. REI appears as the **Get Sequence Name** parameter. Select **Execute**.

The sequence is loaded and a new sequence row appears at the top of the sequence display. Notice that the letters in this row are lowercase to indicate that there are no atomic coordinates associated with the residues.

8. Automatic Sequence Alignment With SCRs

In this exercise, you have already created summary boxes. Therefore, the program determines that the enhanced automatic alignment procedure will be used.

Select the **Alignment/Pairwise_Sequence** command. When the parameter block appears, choose **Automatic** as the **Seq Align Mode**, and **Identity** as the type of **Scoring Matrix**. From the **Objects List** in the **Sequence 1** value-aid, choose MCP. Then select REI in the **Objects List** for the **Sequence 2** parameter.

The parameters **Gap Penalty** and **Gap Length Penalty** are not used when you are doing automatic alignment with summary boxes.

Select **Execute**.

In the REI sequence, gaps are inserted wherever they are needed. However, gaps placed in MCP are inserted between the SCRs, and gaps are also inserted in all other proteins included in the summary boxes.
9. Determine Which Reference Protein Segment is Best

Since there are three reference proteins available as models, you can choose any of the three segments for each SCR. Unless you know in advance a reason to choose one protein over another, sequence similarity is the only criterion available.

Select the Alignment/Pairwise_Sequence command. When the parameter block appears, choose Manual as the Seq Align Mode parameter, and Mutation as the Scoring Matrix. Select Execute to turn on sequence score monitoring.

Put the sequence window into Box Mode. Pick the first q in REI (the bottom row) with the left mouse button. A yellow prompt box appears around the q. Drag the mouse up and to the right until it is over the first Q in MCP. Release the button to create a new sequence box that is four residues wide.

Sequence similarity scores appear in the information area of the main Insight II window. They are an identity of 55.14% for the whole protein or a similarity score of 27.5 for the local region.

The sequence display now looks like this:

F19  < L1> DIOMTQP-TSSLSASLQDRTKTSCRASQD-------INYNLMYQ0KPQDT < L42>
FBJ  < L1> EDVLTOSPLATAEAL-QKTVITCSASSS-------VSSLHWYQ0KPSGTS < L41>
MCP  < L1> DIYMTQSPSSLSVSA-GERMTCCKSSQSLNSGQKFLAVYQ0KPQBP < L48>
REI  < 1> diatcospsslssas-vgdrvtitcqas-------qdiikynwyyqtpgka < 42>

Make another new box. This time, pick the first q in the bottom row (REI) and drag to the first Q in the second row (FBJ).
6. Tutorial

You have created another box in the same region that is also four residues wide.

F19 < L1> D1QMT0P-TSLSASLGDRVT1GCRAS1D------ISNTLNWY1QPDGT < L42>
FBJ < L1> E1VLTO-PAITAASL-0QKUT10CSASS------VSSLHY1QSK6TS < L41>
MCP < L1> D1VMT0SPSILVSA-0ERVTM5CKSSQLNNGONQKNPLAWY1QPGP < L48>
REI < 1> digmtospsios-s-vgdrvtitcqas-------qdiikylwnyqtpgka < 42>

The similarity score for this pair of segments is 22.50, which is worse than the first score.

Finally, make a third box between the top (F19) and bottom (REI) rows over the same four-residue segment.

The score for this pair of proteins is much higher than the others—42.50, which is what you would expect, because the sequences of the unknown protein and the top sequence row (protein F19) are identical in this segment. Thus, the F19 sequence is the best reference protein for this segment.

10. Delete the Unwanted Sequence Boxes

Since the third box you made is the best one in this region, it is a good idea to delete the other two.

Select the Boxes/Delete command. You could also use the mouse to delete the other two boxes, but since they are all overlapping, this method may be easier.

Because there are so many boxes here, it may be difficult for you to dif-
differentiate the box you want to save from the boxes you want to delete.

Since you know that the bottom row (REI) is included in all three boxes, pick any residue in the bottom row (REI) that is included in these overlapping boxes.

Because selected boxes are drawn with the highest precedence, a yellow box can be seen around the bottom two sequence rows (REI and MCP). In the Box Num parameter, 40 appears.

Pick the residue again.

A yellow box appears enclosing the bottom (REI) and second rows (FBJ), indicating that this box includes the bottom and second sequence rows. In the Box Num parameter, 41 appears.

Pick the residue a third time.

A box that includes all the rows is highlighted in yellow. 42 appears in the Box Num parameter. This is the box you want to save because it includes the bottom and top rows.

Pick the residue a fourth time.

The first box turns yellow again.

Select Execute.

The box is deleted, making the red box beneath it visible. Even though
6. Tutorial

box 40 was just deleted, 40 appears in the **Box Num** parameter because the boxes have been renumbered.

Pick the residue again.

The box between the bottom and second rows turns yellow.

Select **Execute** to delete it.

The sequence display should now look like this:

```
F19  < L1> DIOMTOP-TSSLASLGDRTSICRASGD-----ISNYLNWYQGDPGT < L42>
FBJ  < L1> EVLTQSPITLSL-GQKVVTCSASSS------VSSLHYQGKSQTS < L41>
MCP  < L1> DIOMTOPSPSSLSVSA-GERVIMSCCKSSQLNSONQNFALWYQGQPQP < L48>
REI  < 1> DIGMTQPSSLSAS-VGDRVTITCQAS------QDIKYNWYQQTQGK < 42>
```

11. Assigning Coordinates to SCRs

In addition to their role of indicating regions of comparison between peptide segments, sequence boxes also can indicate how and where coordinates are to be copied from one protein to another. In order to do this, the sequence box must enclose a real protein (one with coordinates) and a sequence protein.

First, the box must be frozen. Select the **Boxes/Freeze** command. Pick any residue in REI that is inside the box. It turns yellow and the specification 40 appears in the **Box Num** parameter. Now select **Execute**, and the box turns red, indicating it is frozen.
You are now ready to assign coordinates to this peptide segment.

Select the **Sequences/AssignCoords** command. After the parameter block appears, pick any residue in the box (again, REI (the bottom row) is easiest).

It turns yellow, and the specification 40 appears in the **Box Num** parameter.

Select **Execute**.

The coordinates are copied directly from protein F19 in the top row to REI (the bottom row). When the copying is complete, the atoms of REI appear on the screen superimposed onto F19, and the letters in the bottom row of the sequence box are changed to uppercase.

Now assign coordinates to the next SCR.

Make sure the sequence window is in **Box Mode**. Pick the last s in the spss sequence in the bottom row (REI) with the left mouse button. Drag the mouse up and right to the last S in the sequence SLSVS in the MCP molecule in the third row. Release the mouse button to create a box five residues wide.

<table>
<thead>
<tr>
<th>F19</th>
<th>DIQNTQ-TSSLASLGDRVTITCSRASQD------IINLYNHWQKPDEG</th>
<th>L42</th>
</tr>
</thead>
<tbody>
<tr>
<td>FBJ</td>
<td>ELVTQSPAIHAAS-LGKVITCASSS------VSSLHWQKSGTS</td>
<td>L41</td>
</tr>
<tr>
<td>MCP</td>
<td>DIQNTQSPSLSVSA-GERVTHSOKSSQLWGNQKFKLWYQKPGQP</td>
<td>L48</td>
</tr>
<tr>
<td>REI</td>
<td>diQNTQspelsas-vgdrvtitcqas--------qdiiklynwyqtpgka</td>
<td>42</td>
</tr>
</tbody>
</table>

The similarity score, which appears in the information area, is 24.00,
6. Tutorial

which is the highest in this region.

Now freeze the box. Select the **Boxes/Freeze** command. When the parameter block appears, pick a residue of REI in the box, and the specification **41** appears in the **Box Num** parameter. Select **Execute**.

Then select the **Sequences/AssignCoords** command, pick a residue in the box, and **Execute**.

This time, not all the amino acids match. In the fourth position in the box, MCP has a valine while REI has an alanine. When the program encounters this discrepancy, it loads in the residue library and substitutes the side chain of an alanine for a valine as the coordinates are copied. The replacement is done in such a way as to minimize the RMS deviation between the positions of the original residue’s side chain and the new one. This causes the new side chain to point in the same direction as the one on the reference protein. In the information area and the textport window, the program tells you that the side chain is being substituted. In addition, you are warned if there are any nonbond contacts that are too close.

Finally, assign coordinates to a third region.

In **Box Mode**, pick the first **t** in the bottom row below the third SCR (**REI:THR 20** appears in the **Residue ID** area). Drag the mouse up and right to the **S** in the sequence CRAS of F19 in the top row (**F19:SER L26**). Release the mouse button.
The similarity score is 37.14. In this region, both F19 and FBJ have the same score. F19 was chosen because further investigation showed that F19 was a better model than FBJ in other regions. It is desirable to minimize the number of reference proteins used in model building because then there are fewer transitions from one coordinate frame to another. Also, note that with F19 there are two conservative substitutions, but with FBJ there is a single non-conservative substitution.

Freeze the box with the Boxes/Freeze command as before, and use AssignCoords Sequences to transfer the coordinates from F19 to REI. When you execute these commands, 42 appears in the Box Num parameter.

12. Subset definitions

When you use the Sequences/AssignCoords command, the program creates subset definitions, just as it did when you used the Boxes/Summarize command. The subsets are named protein_name$SCRn and protein_name$SCR. In this case, the subsets are named REI$SCR1, REI$SCR2, REI$SCR3, and REI$SCR. Other subsets are defined as well.

List the subsets by selecting the Subset/List command. When the parameter block appears, select Monomer/Residue for the Collection Level parameters, scroll down the value-aid list, and select REI$MUTATED.

The textport window appears and the residues listed are REI:13, 22, and 24, which are the substitutions Val 13→Ala, Ser 22→Thr, and
6. Tutorial

Arg 24→Gln.

Now scroll down the list and select REI$SPLICE.

The residues listed are REI:6, 14, and 26, which are the ends of the SCRs.

You will need to use this information later because these are sites of potentially bad peptide bond geometries.

Hide the textport window by pressing the <F9> function key.

13. Perform a Loop Search and Assign Coordinates

In order to obtain the same results as presented here, you must have a pdb_ca_distance.dat file in your BIOSYM/data/biopolymer directory created with the pdb_entry.dat file provided with this release media. Please refer to documentation on the pdb_find_distance utility in the Insight II Guide, Appendix C. In addition, INSIGHT_PDB must refer to the directory where the Brookhaven Protein Databank is installed.

In areas other than the SCRs, the conformation of the unknown protein is largely undetermined. When none of the reference proteins can serve as a good source of atomic coordinates, either because their segments are the wrong length or the sequences are so different, a search through the Brookhaven Protein Data Bank for an appropriate peptide loop may be best.

Select the Loops/Search command. Pick any residue inside REI's second box.

The box turns yellow and REI:14 appears as the Start Residue parameter. Default values of 5 are provided for the values of both Pre-
**flex Residues and Postflex Residues.**

Now pick a residue inside REI’s third box.

The box turns yellow and REI:20 is set as the Stop Residue parameter. The value of 5 is automatically set as the default number of Flex Residues.

Select **Execute**.

The program searches the alpha-carbon distance matrix file for peptide segments that are \((5 + 5 + 5)\) residues long that have approximately the same conformations of pre- and postflex regions. The loops will be displayed in the information area as they are found and are listed in order of increasing RMS deviation to REI.

**14. Examine and Display the Loop Choices**

When the search is completed, the **Loops/Display** command is automatically activated. This command allows you to examine all the loops that were found in order to see which make a good fit with the rest of the model protein.

Toggle the **Loop 1** and **Tails** options to **on**, and select **Execute**.

The first loop is from the file pdb1mfa.ent and has a start residue of 9L. Also notice that the RMS deviation of the tail residues between your model and the loop in the file is 0.27 Å. The match is very good.

Toggle the **Loop 2** and **Loop 3** options to **on** and select **Execute**.
Loops 1, 2, and 3 are displayed. When more than one loop is displayed, the information displayed at the top of the graphics window is abbreviated.

For a complete listing of each loop that is displayed, press function key <F9> to view the textport window.

Loop 2 is from the file pdb2fbj.ent. Loop 3 is from the file pdb1hnv.ent. In fact, most of the loops found are immunoglobulins. The exact identities of the loops found are dependent on the composition of the loops database that has been installed at your site.

Toggle Loop 2 and Loop 3 options to off, leaving only Loop 1 on, and select Execute.

15. Assign the Coordinates From the Best Loop

Select Cancel to exit the Loops/Display command. Then select Cancel to exit the Loops/Search command.

Select the Loops/AssignCoords command.

Loop 1 is already selected from before. At this point, you could choose another loop, but select Execute to use Loop 1.

The coordinates are transferred just as in the Sequences/AssignCoords command, the loop is displayed, and the corresponding letters in REI change to uppercase.
When you use the **Loops/AssignCoords** command, subsets are defined just like when you used **Sequences/AssignCoords**. The subsets specific to the loops are `protein_name$LOOPn` and `protein_name$LOOP`; in this case, they are `REI$LOOP` and `REI$LOOP1`.

### 16. Generate Loops Using Random Tweak Algorithm

You can also generate loop segments using the **Loops/Generate** command. The **Loops/Generate** command generates loops using the random tweak method developed by the Levinthal group.

Select the **Loops/Generate** command. Pick any residue inside REI’s second box.

The box turns yellow and **REI:14** appears as the **Start Residue** parameter.

Now pick a residue inside REI’s third box.

The box turns yellow and **REI:20** is set as the **Stop Residue** parameter and 5 is filled in as the number of **Flex Residues**. Note that this is the same loop region you examined with the **Loops/Search** command.

The remaining parameters allow you to input geometric and algorithmic criteria.
6. Tutorial

For the Convergence parameter, which specifies the precision for which closure must be achieved, accept the default value of 0.0500.

For the Closure Iterations parameter, which specifies the maximum number of iterations performed to attempt a particular loop closure, accept the default value of 1000.

For the Internal Overlap parameter, which designates the percent overlap permitted between atoms within the loop, enter 0.5.

For the External Overlap parameter, which designates the percent overlap permitted between atoms within the loop and atoms external to the loop, enter 0.5.

We are allowing a 0.5 overlap in order to speed up the loop generation process. Normally, you would not want to permit such a large overlap.

For the Scale Torsions parameter, which is a scale factor used to speed up the algorithm, accept the default of 60.

Accept the default for the Pro_Torsion parameter which sets the Ω angles for all the prolines in the loop to trans.

Select Execute.

The program generates 10 loop choices, each sharing the same sequence and length as the reference loop.

17. Examine and Display the Loop Choices

As with the Loops/Search command, the loops created through the Loops/Generate command can be displayed using the Loops/Display command.
The Loop/Display command is automatically activated upon completion of the Loops/Generate command.

Toggle the Loop 1 option to on.

Note that with loops created through Loops/Generate, all loops are displayed as if the Tail option were on.

Select Execute.

Loops are displayed with accompanying text information on the rms value.

18. Assign the Coordinates From the Loop

As with the Loops/Search command, the coordinates from the loops created through the Loops/Generate command can be transferred to the model protein using the Loops/AssignCoords command.

Select Cancel to exit the Loops/Display command.
Then, select Cancel to exit the Loops/Generate command.

Select the Loops/AssignCoords command. Loop 1 is already selected from before, although you could choose another loop even though you did not display it. Select Execute to use Loop 1.

The coordinates are transferred from loop choice 1 to the model protein and the new loop region with the model protein is displayed.

Again, when you use Loops/AssignCoords, subsets are defined for the loop region using the form protein_name$LOOPn, where n is the...
6. Tutorial

number of the loop region, in order from left to right, in the model protein.

19. Assigning Coordinates to Loops Directly

Occasionally, a segment from one of the reference proteins has the same sequence and length as the model sequence protein. Even though the segment may not be in every reference protein and, therefore, could not be classified as an SCR, it is probably a better choice for a loop than any loop found in a search through the PDB database or generated de novo. A direct assignment of coordinates is the best thing to do in this situation.

A new box is created around three pairs of residues.

F19  < L1> DIOMTQ- TSSL SASL GAVT ISTCRAS- D----- ISNTLVWYQQPDGT < L42>
FBJ  < L1> DIQLTSPA ISAASL GAVT ISTCRAS- S----- VSSLHWYQQSGTS < L41>
MCP  < L1> DIOMTUESLVS-ERVTM SCKSSS QLLNSGQNRFLAWYQQPQP < L48>
REI  < 1> diontep-LSAS-vgdvTITCQAS-------qiikylnwqtpgka < 42>

Freeze the box by selecting it with the right mouse button, holding, and clicking the left mouse button.
The box turns red.

Now select the **Sequences/AssignCoords** command, and pick any residue inside the box. Specify **Designated Loop** as the **Segment Definition** parameter.

Normally this command is used to assign coordinates to SCRs, but in this case it forces the subset definitions to be consistent with the segment that is being classified as a loop.

Select **Execute**.

Select the **Subset/List** command. When the parameter block appears, select **Monomer/Residue** for the **Collection Level** parameter. Then, scroll down the list and select **REI$LOOP1** and then **REI$LOOP2**.

Notice that the subsets are redefined each time a new coordinate assignment is made; **REI$LOOP1** is defined as residues **REI:7–9** and **REI$LOOP2** is **REI:15–19** even though the second loop was assigned first. This means segments can be reassigned whenever appropriate. New coordinates are given to the model and all subsets are resolved.

Be careful when you reassign coordinates to a region. You should always reassign all the residues in the region. If you don’t, although the coordinates may not be too unrealistic, the subset definition would be inconsistent.

**20. Freeze, UnFreeze, or Delete All Boxes with Star Feature**

For the **Boxes/Freeze**, **Boxes/UnFreeze**, and **Boxes/Delete** commands, there is an option available that allows you to perform these operations on all sequence boxes. Simply typing * (asterisk) for the **Box Num** parameter denotes that the command is to be performed on all sequence boxes.
6. Tutorial

all sequence boxes. Although the Box Num parameter is used for other commands, the * feature is available only the Boxes/Freeze, Boxes/UnFreeze, and Boxes/Delete commands. The usual rules for these commands still apply. That is, frozen sequence boxes may not be deleted until they are unfrozen using the Boxes/UnFreeze command.

As an example, delete all the sequence boxes you have created in this portion of the tutorial.

Select the Boxes/Delete command. Enter * for the Box_Num parameter and select Execute.

An error is displayed indicating that frozen sequence boxes exist.

Select the Boxes/UnFreeze command. Enter * for the Box_Num parameter and select Execute.

All the sequence boxes will be unfrozen (except for the summary boxes, which although they are required to stay frozen, can nonetheless still be deleted).

Now you are ready to delete the boxes.

Again select the Boxes/Delete command. Enter * for the Box_Num parameter and select Execute.

All the boxes are deleted. Alternatively, if you wanted to delete only the summary boxes, you could use the Boxes/Summarize command, setting the Summarize Operation parameter to Delete.
21. Quit Insight II and Homology

To exit Homology type quit at the command prompt and press <Enter>. To confirm, press <Enter> again.

Lesson 10: Finding alternative multiple sequence alignments

In this lesson you will explore alternative alignments that can be obtained by varying the parameters of the Alignment/Multiple_Sequence command. You will also simultaneously align two immunoglobulins and the six serine proteases, and you will examine a region of surprising sequence similarity shared by these unrelated proteins.

1. Invoke Insight II

At the system prompt enter the following command:

> biosym_tutorial -i

The status of your tutorial directory appears and you are asked for the product names you wish to install.

If the listed status of Homology is “Not Installed”, type:

> homology

and press the <Enter> key to install Homology. Otherwise, proceed to the next step.

The tutorial script creates a new tutorial directory under your home
6. Tutorial

directory. Necessary files are copied into the tutorial directory.

Press <Enter> again.

A list of options appears on the screen.

Enter 3 and press <Enter> to change to the tutorial directory.

The list of options appears again.

Enter 1 and press <Enter> to start Insight II. After Insight II is loaded, go to the MSI icon in the upper left of your window and select the Homology module.

2. Read in the File Containing the Sequences to be Aligned

Select the File/Restore_Folder command. Choose homology_serprot.psv from the value-aid and select Execute.
After the file has been read, select Cancel.

The proteins are six mammalian serine proteases.

3. Align the Sequences

Select the Alignment/Multiple_Sequence command, and select Execute.
When the alignment is finished, select **Cancel**.

4. **Scroll to the Sixth, Seventh, and Eighth m-blocks**

Make sure the sequence window is in **Seq** mode.

Scroll the sequence to the left, using the middle mouse button until the leftmost residue counter for sequence EST is 156.

Three m-blocks are now visible: a 5-block eighteen residues long, a 6-block four residues long, and a 6-block fifteen residues long. Note the pale color of the middle block, indicating its relatively low significance.

The Alignment/Multiple_Sequence command has an intrinsic bias in favor of blocks of high dimension, e.g., m-blocks containing the largest possible number of sequences. For example, alternative m-blocks containing fewer but longer sequence segments can be constructed in the region of the pale 6-block. Some of these alternative blocks have higher statistical significance than the 6-block, but the algorithm chooses the 6-block because it contains more sequences.

You can alter this bias towards blocks of high dimension by changing the parameters **Dimension Bias**, **Not Signif p**, or **Num Related Seqs**. The parameters **Mandatory Seq** and **Strict_Overlap** can also be used to find alternative alignments.

The **Dimension Bias** parameter is a real number between 0 and 1, inclusive. Higher values give a greater bias towards blocks of high dimension. The algorithm still retains some of this bias even when **Dimension Bias** is zero. The **Dimension Bias** parameter provides fine tuning of this bias, and for some sequences it has little or no effect.
6. Tutorial

on the results. This is the case for these six serine proteases. You must vary other parameters to find alternative alignments.

5. Change the Not Signif p Parameter to Find an Alternative Alignment

As already discussed, the parameters High Signif p and Not Signif p affect the alternate coloring of the m-blocks. The Not Signif p parameter serves an additional purpose. During its search for m-blocks, the Alignment/Multiple_Sequence command rejects as insignificant all m-blocks for which p is greater than or equal to Not Signif p.

Select the Alignment/Multiple_Sequence command. Enter 0.01 for the Not Signif p parameter and press <Enter>.

Notice that the middle 6-block is now white. This means its p value is greater than the newly entered Not Signif p. This m-block is therefore not statistically significant by this criterion. Toggle the Delete_Boxes parameter to on. Select Execute.

Because the 6-block in the middle is no longer considered significant, it is omitted from the new alignment. In its place appears a longer 5-block that has a higher statistical significance.

6. Restore the Original Alignment

To restore the original alignment, enter the original value of 0.1 for the Not Signif p parameter, toggle on the Delete_Boxes parameter and select Execute.

7. Specify a Mandatory Sequence to Find an Alternative Alignment

The Mandatory Seq parameter specifies the name of a sequence that must be included in all m-blocks. It constrains the alignment by rejecting all m-blocks that do not contain the mandatory sequence. In
the portion of the alignment that is now on the screen, none of the EST residues in the left half of the display are aligned with other sequences (i.e., they are not contained within an m-block).

Most of the sequence TGN within the 5-block is yellow, indicating that this is one of the SCRs found by the automatic Alignment/Structure command. That command found a region with similar structure in the EST sequence (the yellow residues below the 5-block), but the Alignment/Multiple_Sequence command did not include those residues in the block. There are at least two possible explanations for this.

One possibility is that this is a case of high structural similarity without a correspondingly high sequence similarity. Another possibility is that there exists a less significant m-block that contains some of these EST residues, but that overlaps the higher scoring 5-block. Such a block would be rejected from the alignment in favor of the 5-block.

The less significant block can be revealed, however, by making EST the mandatory sequence. This will force the rejection of the 5-block and allow any block of lower significance that contains EST to appear in that region.

Enter the name EST for the Mandatory Seq parameter. Make sure that Delete_Boxes is on, and select Execute.

The 5-block is gone and a new 2-block appears that contains some of the previously unaligned EST residues.

Although more of EST is now aligned in this region, the yellow SCR on the left is still not aligned. This appears to be a region of high struc-
6. Tutorial

tural but low sequence similarity.

8. Restore the Original Alignment

To clear the Mandatory Seq parameter, select it with the mouse, press the <Delete> key, and then press <Enter>. When the sequence name is deleted, it is replaced by the default value None, indicating that no mandatory sequence has been specified.

To restore the original alignment, make sure that Delete Boxes is on and select Execute.

9. Use the Strict_Overlap Criterion to Find an Alternative Alignment

The Alignment/Multiple_Sequenc command can use either of two criteria for determining which sequence segments belong in a given m-block. The default criterion guarantees that each sequence segment in the block is similar to at least one other sequence segment in the block. Similarity in this case means that the two sequence segments come from a region over which their cumulative score exceeds a pairwise score threshold. The default criterion was used in the present alignment. The alternative criterion is more discriminating, in that it requires that each sequence segment in the m-block must be similar to all others in the m-block. This criterion is activated by toggling Strict_Overlap to on in the Alignment/Multiple_Sequenc command. The strict overlap criterion tends to produce m-blocks containing fewer sequences, but the sequence segments within them are more closely related to one another.

Select the Alignment/Multiple_Sequenc command. Toggle the Strict_Overlap parameter to on. For the Zone Length enter 250. Make sure that Delete Boxes is on and select Execute.

The Zone Length parameter specifies the number of residues of each
sequence that will be included in the alignment. Limiting the **Zone Length** to 250 speeds the alignment by excluding the duplicate copies of RP2 and PKA.

Using the strict overlap criterion, the command finds four m-blocks where before it had found three. The region that before contained the 5-block now contains a longer 3-block. The four-residue-long 6-block has become a longer 5-block of higher significance. The rightmost 6-block has become two shorter blocks, one of which is now a 3-block. The other is still a 6-block, however, indicating that each of its sequence segments is related to all others in the block. This suggests that this is a highly conserved region.

The 6-block contains the aligned catalytic serine residues, confirming the suspicion that this is a highly conserved region.

Select the Std button to display the standard colors.

To restore the original alignment, toggle the Strict_Overlap parameter to off. Make sure that Zone Length is 250 and that Delete_Boxes is on. Select Execute.

11. Read in Two Sequences that are Not Serine Proteases
In many alignment problems there may be one or more sequences that are only distantly related to the others, or perhaps not related at all. With the proper choice of parameters, the Alignment/Multiple_Sequence command copes well with these situations.

Select the Molecule/Get command. Choose PDB for the Get File Type parameter. Enter pdbrei.ent for the File Name parameter, and select Execute.

After the molecule has been read in, enter pdb2rhe.ent for the File Name parameter, and select Execute.

Select the Sequences/Extract command. For the Extract Molec Name parameter type an asterisk (*) and press <Enter>. After the new sequences appear, select Cancel.

These are two immunoglobulin fragments, one of which was modeled in Lesson 4. They are completely unrelated to the serine proteases.

Notice that the color of the middle m-block has changed from pale magenta to white, indicating that it is no longer significant. This is because the p value is defined as the probability that a block of equal or greater score could be found by chance in a set of random sequences of the same lengths as those in the sequence display. Because you have just added two sequences to the sequence display, the size of the search space used in the calculation of p is now larger. The probability of finding a block with this score by chance is therefore larger.

12. Use the Num Related Seqs Parameter to Reduce the Bias that Favors Blocks of High Dimension

If you try to align all eight of these sequences using the default parameters, the Alignment/Multiple_Sequence command will run for an unacceptably long time. The reason for this is that it will waste con-
siderable time trying in vain to find blocks of eight mutually related sequence segments. You can avoid this by telling the Alignment/Multiple_Sequence command that only six of the sequences are known to be related.

Select the Alignment/Multiple_Sequence command and enter 250 for the Zone Length parameter. Enter the value 6 for the Num Related Seqs parameter.

This tells the Alignment/Multiple_Sequence command to accept m-blocks with as few as six sequence segments in the first search pass. It does not specify which of the six sequences are related, nor does it exclude the possibility of finding highly significant 7-block or 8-blocks, if these exist. Some 7- or 8-blocks may be overlooked, however, if they overlap 6-blocks of much greater significance.

Make sure that Delete_Boxes is on and select Execute.

13. Scroll through the Sequences to Examine the Alignment

Use the horizontal scroll bar until the N-termini of the sequences are visible.

As you scroll through the alignment, notice that the two immunoglobulins have been correctly aligned with each other, as have the six serine proteases, and that no m-block includes members of both of these unrelated protein families.

14. Use Single_Search Mode to Find a 7-block

Although this result suggests otherwise, there is in fact a statistically significant 7-block that spans these two protein families. It was not found in this alignment because a more significant 6-block of serine proteases overlapped it. You could find it by repeating the last command.
with Num Related Seqs set to 7, but this would take a long time. The 7-block can be found quickly, however, by using the Alignment/Multiple_Sequence command in single-search mode.

Choose **Single_Search** for the **Mult Align Mode** parameter.

The parameters **Minimum Zone Length**, **Num Related Seqs** and **Dimension Bias** become inactive because they are only used for automatic alignment. In single-search mode, the Alignment/Multiple_Sequence command performs one segment pair overlap search, which may or may not find m-blocks depending on the value of the **Pairwise Threshold** parameter.

Enter 7 for the **Minimum Seq Per Blk**.

This causes all m-blocks containing fewer than seven sequences to be rejected from the alignment.

Enter **250** for the **Zone Length** parameter. Make sure the **Delete_Boxes** parameter is on. Select **Execute**.

The alignment summary in the information area indicates that there are four m-blocks in the alignment. These, however, are only the four 2-blocks in the duplicated RP2 and PKA sequences. Although the **Delete_Boxes** parameter was on, they were not deleted from the alignment because they lie outside the zone selected for alignment (i.e., beyond the specified **Zone Length** of 250). This search therefore found nothing new.

Set the **Pairwise Threshold** parameter to **25** and press <Enter>. Select **Execute**.
Now a fifth m-block has been found. If the REI sequence is not currently visible, enlarge the sequence window until you can see REI. This indicates that the REI sequence has been shifted off the screen in that direction.

Find the 7-block by using the scroll bars or enlarging the sequence window.

15. Test the 7-block for Biological Significance

Look carefully at this 7-block. It includes the segment IIKYLNW-YQQT from the REI sequence and six similar segments from the serine proteases. Its dark color indicates that it is highly significant (in fact, \( p < 10^{-12} \) for this block). For all but one of these eleven residues, an identical residue can be found in the same position in one or more of the corresponding serine protease segments. Some, like the tryptophan and the first tyrosine, appear to be highly conserved. For the last six of these residues, the identical six residues occur in the corresponding position of the CHA sequence. The important issue, however, is whether or not these sequences share a common 3D structure in this region.

You can test this using manual structural alignment. To make the results easier to interpret, first suppress the display of irrelevant structures.

Select the Object/Blank command. Make sure that Blank Operation is on. Enter RHE for the Object Name parameter and select Execute.

With the Object/Blank command still active, pick any residue of EST in the sequence display. The command triggers immediately, blanking the EST molecule.
Now only the CHA molecule, and the portion of REI that lies within the 7-block, are displayed.

Select the **Molecule/Display** command. Choose **Only** for the **Display Operation** parameter. For the **Molecule Pick Level** parameter select **Molecule**. For the **Molecule Spec** parameter enter **REI:A29-A39:CA**. Select **Execute**.

As you enlarge the box, the two alpha carbon traces are superimposed, and the RMS score is displayed in the information area. It is obvious that the serine protease and the immunoglobulin have no structural similarity in this region: the former is a helix and the latter is a beta strand.

This result illustrates that sequence similarity does not necessarily imply structural similarity. It also raises a subtle but important statistical point. The statistical significance calculated by the **Align-**
ment/Multiple_Sequence command is only a rough estimate, based on a simplified model of the statistical nature of protein sequences. More importantly, its meaning is based on the null hypothesis that the m-block in question was found by searching random sequences having the same lengths as those in the sequence display. In practice the sequences used in homology modeling come from the much larger search space of a sequence or structural database. For this reason, the p values calculated by the Alignment/Multiple_Sequence command are underestimates. They are best used as guides to the relative significance of the m-blocks found, rather than as indicators of absolute significance.

16. Quit Insight II and Homology

To exit Homology, type quit at the command prompt and press <Enter>. To confirm, press <Enter> again.
6. Tutorial


A. References


EMBL Data Library; European Molecular Biology Laboratory, Postfach 102209, 6900, Heidelberg, Germany


GenBank database; IntelliGenetics, Inc., 700 El Camino Real East, Mountain View, CA 94040


Karlin, S.; Altschul, S.F. “Methods for assessing the statistical significance of molecular sequence features by using general
A. References


PIR/NBRF database, National Biomedical Research Foundation, Georgetown University Medical Center, 3900 Reservoir Rd., NW, Washington, D.C. 20007


A. References


A. References
File Formats

Introduction

Many files used by Homology are described within the MSI Products Common File Formats book, including:

- .seq (amino acid sequence files)
- .align (sequence alignment files)

The amino acid scoring matrices used by Homology are explained in this appendix.

Amino Acid Scoring Matrices

When sequence alignment calculations are performed with the Sequence Alignment command, or when a sequence database search is done with the commands in the Databases pulldown, a scoring matrix is consulted that lists the similarities of the amino acids. This is equivalent to assessing the likelihood of one amino acid type being substituted for another. Four such matrices are provided with the program, that base the comparisons on four different criteria.

1. Identity Matrix
2. Codon Substitution Matrix
3. Dayhoff Evolutionary Mutation Matrix
4. Hydrophobicity Matrix

Refer to Appendix E, Matrices, to examine the matrices themselves.
B. File Formats

User Scoring Matrix Files

You can also enter your own matrix that uses a scoring method of your choice. The format of this file depends on whether the Sequence Alignment command or the Input Databases command is used. Both formats are described below.

Sequence Alignment Command

Each matrix file consists of 20 lines of 20 real numbers each (free format), which correspond to a standard matrix of all 20 amino acids. The numbers are entered in the alphabetical order of the amino acids' three-letter codes.

ALA ARG ASN ASP CYC GLN GLU GLY HIS ILE LEU LYS MET PHE SER THR TRP TYR VAL

Example:

```
  ALA ARG ASN ASP CYC GLN GLU GLY HIS ILE LEU LYS MET PHE SER THR TRP TYR VAL
  ALA  2  6  0  2  0  0  1  1  1  1  1  1  0  0  0  0  0  0  0  0
  ARG -2  6  0 -1 -1  2  2  2  0  0  0  0  0  0  0  0  0  0  0  0  0
  ASN  0  0  2  2 -4  1  1  0  2  2 -3  1  2 -4 -1  1  0  0  0  0  0
  ASP  0 -1  2  4 -5  2  3  1  1 -2 -4  0  3  6  1  0  0  0  0  0  0
  CYC -2 -4 -5 12 -5 -5 -5 -3 -2 -6 -5 -5 -4 -2 0 0 0 0 0 0 0 0
  GLN  0  1  1  2 -5  4  2 -1  2 -2  2 -1 -1 -5 0 0 -1 -5 -4 -5
  GLU  0 -1  1  3 -5  2  4  0  1 -2 -3  0 -2 -5 -1 0 0 0 0 0 0 0
  GLY  1 -2  0  1 -2  0  5  2 -3 -4  2 -3 -5 -1 1 0 0 0 0 0 0
  HIS  -1  2  1 -1  2  2  6 -3 -2  0  2 -2  2 0 -1 1 -2 0 0 0 0
  ILE -1 -2 -2  2 -2 -2  2  3 -2  2  2  2  2  1 -1 -1 5 1 4 4 4 4
  LEU -2 -2  3 -4 -3  0 -2 -3  4  2  0  3 -4 2 -3 -3 0 2 -2 1 2
  LYS -1  3  1  0 -5  1  0  2  0  2 -3  5  0 -5 -1 0 0 0 3 4 2
  MET -1  0 -2  3 -5 -1  2 -2  3  2  4  0  6  0  0  2 -2 -1 -4 2 2
  PHE -4 -4 -4 -4 -4 -4 -4 -5 -5 -5 -5  2  1 -2 -5 0 0 -5 -3 0 7 -1
  PRO  0  1  0  3  0  0  1  0  0  1  0  3  0  0  0 0 0 0 0 0 0
  SER  1  0  1  0  0 -1  0  1 -1 -1 -1  0  3  2  3 1 2 1 -2 -3 1
  THR  -1  1  0  0  0  2  1  0  0  1  0  1  0  -3 0 1 3 5 5 0
  TRP -6 -2 -4 -7 -8 -9 -5 -7 -7 -7 -7 -2 -2 -3 -4 0 0 -6 -2 -5 17 0 8
  TYR -3 -4 -2 -4 0 -4 -6 -5 0 -1 0 -1 -1 2 -2 7 -5 -3 -3 0 10 1
  VAL  0 -2 -1 -2 -2 -2 -2 -1 2 4 2 -2 2 -2 -1 -1 0 0 2 -2 4
```

Input Databases Command

The scoring matrix user file format for the FASTA sequence database searching program differs from that used by Homology in its Sequence Alignment command. The format used by FASTA is described below.
When constructing your own scoring matrix files, place them in your current working directory and be sure to append the extension .fasta.mat, e.g., filename.fasta.mat. Files with this extension appear in the value-aid of the Input Databases command and are easily distinguished from scoring matrices files in the Sequence Alignment format.

Table 8:

<table>
<thead>
<tr>
<th>Line #</th>
<th>Description and Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td>;P Title</td>
</tr>
<tr>
<td></td>
<td>Title must be less than 77 characters; the P stands for protein.</td>
</tr>
<tr>
<td>Line 2</td>
<td>Parameters used by the program for calculating match scores. These values are all used internally by the FASTA program. For best results, the values should not be changed from the default settings listed below:</td>
</tr>
<tr>
<td></td>
<td>4 50 100 5 2 70 3</td>
</tr>
<tr>
<td>Line 3</td>
<td>Penalty values for adding gaps into a sequence alignment. The first value is the penalty score to initialize a gap; the second is the penalty to extend a gap. The penalty values are applied by subtracting their value from the total score. Default values are:</td>
</tr>
<tr>
<td></td>
<td>-12 -4</td>
</tr>
<tr>
<td>Line 4</td>
<td>Character(s) denoting end-of-sequence. The default values are:</td>
</tr>
<tr>
<td></td>
<td>@ and *</td>
</tr>
<tr>
<td>Line 5</td>
<td>The 23 possible amino acid types. Besides the usual 20, FASTA recognizes the ambiguous characters: B (D or N), Z (E or Q), and X (unknown). The order must be:</td>
</tr>
<tr>
<td></td>
<td>A R N D C Q E G H I L K M F P S T W Y V B Z X</td>
</tr>
<tr>
<td>Line 6</td>
<td>The amino acid types for the 23 characters above. Note that B is considered to be D, Z is E, and X is A:</td>
</tr>
<tr>
<td></td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 3 6 0</td>
</tr>
</tbody>
</table>
B. File Formats

Table 8:

<table>
<thead>
<tr>
<th>Line #</th>
<th>Description and Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines 7-29</td>
<td>The lower triangle of the scoring matrix. Each entry represents the value of a substitution of one amino acid for another. The position in the file indicates which substitution is being scored. If written as a triangle, consider the axes to be the amino acids listed in the order given.</td>
</tr>
</tbody>
</table>

Example:

```
;P genetic code matrix
4 50 106 5 2 70 3
-12 -4
A R N D C Q E G H L K M F P S T Y V B E Z X
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 3 6 0
```

Get Sequence, Alignment and Databases commands

The following is a description of sequence file formats used by the Get Sequence command and amino acid scoring matrices used by Alignment and Databases commands.

Sequence File Formats:

1. Biosym format (.seq):
Biosym sequence file format is a text file containing amino acid sequence (in 20 single letter codes) only. No annotations are allowed and the maximum length of each line is 80 characters.

2. Pearson (FASTA) format (.aa):
   Homology implementation of this format utilizes a subset of FASTA format definition, namely: the first line should start with the ‘>’ character and is skipped. Following lines should contain amino acid sequences in standard 20 single letter code. The reading continues until the EOF or ‘*’ characters are encountered.

   For a full description of the FASTA format see:
   http://www.cs.virginia.edu/brochure/profs/pearson.html

3. GCG format (.gcg):
   GCG format can contain arbitrary comments between the beginning of the file and a line that contains a “..” string. The following lines each contain an opening integer token (number of the first residue — not used insight II numbering) followed by the sequence in 20 single letter code. Reading proceeds until an EOF is reached.

   For a full description of GCG format see:
   http://www.gcg.com/products/documentation.html

4. NBRF-PIR format (.pir):
   The PIR format implementation utilizes a subset of the PIR format definition, namely: the first line should start with “>P1;” string and is skipped. A second line can contain an arbitrary comment. The following lines should contain amino acid sequences in standard 20 single letter code. The reading continues until the EOF or ‘*’ characters are encountered.

   For full description of PIR format see:
   http://www-nbrf.georgetown.edu/pirwww/

5. Swiss-Prot format (.sws):
   A Swiss-Prot file is expected to have a line that begins with an “SQ” string. The following lines should contain a sequence in
B. File Formats

20 single letter standard code. The last line should start with the “//” string.

For full description of Swiss-Prot format see:

http://www.expasy.ch/txt/userman.txt

---

**Secondary structure file**

The secondary structure file contains the sequence and (predicted) secondary structure. Here is an example showing the file’s format:

```
#SEQID PDI
# PTI, residues 1 - 58.
# GOR assignment
#num id predsec predrel pH pE pL
1 R - 0 0 0 0
2 P - 0 0 0 0
3 D - 0 0 0 0
4 F - 0 0 0 0
5 C - 0 0 0 0
6 L - 0 0 0 0
7 E - 0 0 0 0
8 P - 0 0 0 0
9 P - 0 0 0 0
10 Y - 0 0 0 0
11 T - 0 0 0 0
12 G - 0 0 0 0
13 P - 0 0 0 0
14 C - 0 0 0 0
15 K - 0 0 0 0
16 A - 0 0 0 0
17 R H 6 0 0 0
18 I H 5 0 0 0
19 I H 5 0 0 0
20 R E 4 0 0 0
21 Y E 5 0 0 0
22 F E 5 0 0 0
23 Y E 1 0 0 0
```

---

260  Homology
Secondary structure file

The comment line starts with a “#” sign with the following exceptions:

#SEQID line: specifies the name of the sequence only object

#PDBID line: specifies the name of the sequence with known structure

#num line - defines the order and composition of information carried in the file.

The Following keywords are acceptable:

num: residue number (e.g., from the PDB file).

id: residue identity.

predsec: the predicted secondary structure.

“H”: helix

“E”: strand

“_”: coil

predrel: the relative significance of the predicted secondary structure.

sec: the actual Kabsch-Sander secondary structure.

secrel: the relative significance of the secondary structure. It should always be set to 9.

A, R, ..., Y: the residue type propensities.
B. File Formats

262 Homology
**active box**—A sequence box that currently can be moved in relation to the residues in the sequence window. It is colored green.

**active vs. inactive parameters**—An active parameter is one that may have its value changed. An inactive parameter is not changeable. Active parameters appear in bold. Inactive parameters appear in grey.

**codon matrix**—An amino acid similarity scoring matrix. Amino acids are considered to be similar (and, thus, receive a higher score) when a small number of DNA bases must be changed to cause a mutation from one amino acid being compared to the other.

**choose**—Refers only to the selection of options (within parameter blocks or value-aids) using the mouse.

**focus**—The yellow area of the parameter block. The program is indicating the location in the parameter block where the next information should be added. You may move the focus by selecting another part of the parameter block.

**frozen box**—A sequence box that cannot be moved and through which no sequence can be passed. Most frozen boxes are colored red, but summary boxes, which are white, are also frozen.

**gap penalty**—A numerical value subtracted from the total similarity score to insert a sequence gap when performing automatic sequence alignments. Depending on the algorithm, there can be a single penalty, or separate values to initiate or extend a gap region.

**hydrophobicity matrix**—An amino acid similarity scoring matrix. Amino acids are considered to be similar (and, thus, receive a higher score) when the difference between their hydrophobicity scores is small.
C. Glossary

identity matrix—An amino acid similarity scoring matrix. Amino acids are considered to be similar (and, thus, receive a higher score) only when identical amino acids are compared.

initial region—A single segment of contiguous amino acids found to match between the probe sequence and sequences in database files.

ktup—The number of amino acids to be considered as an indivisible unit when assessing similarity scores during sequence database searching. The number of possible ktups is equal to the number of amino acids (20) raised to the power of ktup. A lookup table is constructed to speed amino acid similarity scoring.

loop—See variable region.

lower menu bar—The lower of two white menu bars displayed at the top of the Insight II screen, showing the pulldowns available. The pulldowns shown depend on what module is active, although some are common to more than one module.

model protein—The protein whose molecular structure is not known. Only the amino acid sequence is known. It is the protein being built.

module—The functional division of the program. A family of related commands further grouped into pulldowns. For instance, Homology is a module. A module is selected by choosing from the Module pulldown in the top menu bar.

overlap—The fraction of interpenetration permitted between any pair of atoms when screening for high energy nonbond contacts.

parameter block—The block of information that appears when a command is selected. You are required to fill in enough information to make a command meaningful, at which point the Execute box becomes active. However, you may wish to fill in more than the minimum amount of information.

pick—Refers to selecting an atom, molecule, or similar object from the screen (e.g., you pick an atom by placing the cursor over its screen image and pressing the right mouse button).

probe length—For the Sequence Alignment command, the number of contiguous residues for which $C_\alpha$ distances are compared to assess local structural similarity.
probe sequence—For the Input Databases command, the sequence to which entries in the databases are compared. The FASTA searching background job finds sequences similar to the probe sequence, based upon various matching criteria.

primary structure—The sequence of amino acids of a protein.

prompt box—A sequence box one row high and one residue wide used to give visual feedback for a chosen residue in the Initialize Boxes command.

real protein—See reference protein.

reference protein—A protein whose molecular structure is already known. In a homology project, the coordinates of the reference proteins are used to build the model of the unknown or sequence protein.

rotamer—One of several conformational states available to an amino acid side chain that is frequently seen in globular proteins. The conformations are generally a subset of the possible combinations of the familiar gauche and anti torsion angles.

residue counter—The indicators at the ends of each sequence row that specify the identities of the residues at the left and right ends of the sequence display.

scroll—To move a component of the sequence display. Sequence rows, sequence gaps, and sequence boxes can be scrolled.

scoring matrix—A symmetric matrix indicating the degree of similarity between any two amino acids. The number of matrix elements is equal to the number of amino acid types squared. The values of the elements depends on the scoring criterion used.

secondary structure—The collection of contiguous peptide segments with repetitive values for $\phi$ and $\psi$. Each segment may be $\alpha$-helix or $\beta$-sheet. Proteins are classified according to the types and arrangements of their peptide segments. Secondary structure tends to be conserved among related proteins.

select—General term, refers to pointing to an object and pressing the right mouse button to select particular modules, pulldowns, commands, and options (in parameter blocks and value-aids).

selected box—A sequence box that has just been picked with the mouse, to be used as input to a command. It is colored yellow.
C. Glossary

**sequence box**—Any box drawn around residues in the sequence display. Most sequence boxes are used to compare the structural or sequence similarities of the short peptide segments they enclose. (See also prompt box, suggestion box, and summary box.)

**sequence gap**—A dash symbol (-) that is used in a sequence row to indicate areas of residue insertion or deletion.

**sequence protein**—See model protein.

**sequence row**—A single string of single-letter amino acid codes in the sequence display corresponding to a particular protein.

**splice point**—The junction between two peptide segments, especially when they are taken from different reference proteins.

**structurally conserved region (SCR)**—A peptide segment that has been shown by RMS calculations to have a similar conformation to a segment of equal length in another protein.

**suggestion box**—A type of sequence box that indicates an appropriate linear peptide region for a sequence gap. It appears automatically when the Initialize Boxes command notices that a gap must be inserted to align the selected residues. It is colored yellow.

**summary box**—A type of sequence box that encloses the residues defining structurally conserved regions from several reference proteins. A summary box is drawn around the residues found to be within SCRs common to all reference proteins. Summary boxes are colored white.

**tertiary structure**—The complete 3D structure of a protein.

**toggle**—The act of selecting and changing a parameter that has an on/off state.

**top menu bar**—The upper of two white menu bars displayed at the top of the Insight II screen, showing the pulldowns available. The top menu bar pulldowns are constant, regardless of which module is currently active.

**trigger function**—Pressing <Enter> when the focus is in a trigger parameter automatically “triggers” (executes) the command. It is equivalent to selecting Execute. Trigger parameter boxes are denoted by a double border.

**unknown protein**—See sequence protein.
user matrix—An amino acid *scoring matrix* defined in a file created by the user. Each matrix element indicates the degree of similarity between the two amino acid types implied by the intersection of a row and column of the matrix. The format of the file is different, depending on whether the Sequence Alignment command (extension .mat) or the Input Databases command (.fasta.mat) is used to read the file.

value-aid—A window that pops up to help you complete a parameter block.

variable region (VR)—A peptide segment of the *model protein* that is not found in every *reference protein*. Also called a loop, this type of region is located between two SCRs in the amino acid sequence.
C. Glossary
The purpose of the `seq_extract` utility is to extract amino acid sequences from Brookhaven Protein Databank files.

The first step in homology modeling is to find suitable reference proteins. The utility program `seq_extract` greatly facilitates this step by extracting all protein sequences from the Brookhaven Protein Databank (PDB) and collecting them into a single sequence database file called `pdb.seq`.

The `pdb.seq` file uses the file format used by the National Biomedical Research Foundation (NBRF), and it can be searched using the database searching software accessible via the Databases pull-down. See Sequence Database Searching for an explanation of how to search a sequence database file.

If you type

```
> seq_extract
```

with nothing else on the command line, the program extracts as much amino acid sequence information as possible. It does this by searching all files in the directory specified by the environment variable `INSIGHT_PDB`, if this is defined; otherwise it searches all files in the current directory. The output files are placed in the current directory. Alternatively, you can restrict the search to a subset of the PDB files using various command line options (see below).

### Output

#### Results Displayed on the Screen

As the program runs, it displays an underscore for each file from which a protein sequence could be extracted and a dot for each file
D. seq_extract Utility

that lacked protein sequence information. Below is a sample of the screen output:

```
seq_extract Version 1.0
Searching /4d2/usr2/brookhaven/ for PDB files.
Reading input files

- done
659 of 789 files contained protein sequences.
Writing output files

Output Files

In addition to the pdb.seq file, the program also creates a log file named seq_extract.log. The pdb.seq file contains the extracted sequences. The seq_extract.log file contains two lists of file names. The first list contains the filenames from which sequences were extracted, and the second, filenames of those for which the extraction failed, as well as explanations for each extraction failure.

In creating pdb.seq, the program translates the three-letter amino acid symbols used in the PDB files to their equivalent one-letter symbols as defined by the IUPAC-IUB standard:

<table>
<thead>
<tr>
<th>A</th>
<th>ALA</th>
<th>alanine</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>ASX</td>
<td>ASP or ASN, unspecified</td>
</tr>
</tbody>
</table>
```
Note that the present version of Homology cannot read sequences containing B, Z, or X (refer to the \texttt{-R} option described below).

The sequence identifier name used in each sequence entry in the pdb.seq file is the four-character identifier of the PDB file from which the sequence was extracted (e.g., 1ACX for actinoxanthin).

For multiple-chain proteins seq_extract creates a separate sequence entry for each chain. The identifier of such an entry is the PDB four-character identifier with a one-letter chain identifier appended (e.g., 0DCHA and 0DCHB for hemoglobin A and B chains, respectively).

### Execution Options

The program provides several command line options. Type \texttt{seq\_extract -help} to display the usage of the seq\_extract utility:
D. seq_extract Utility

usage:

% seq_extract [options] [pdb_files]

valid options:

- help    display this message
- c       combine chains
- r       reject files w/ unknown residues (X)
- R       reject files w/ unknowns or ambiguities
- s       use strict PDB syntax test
- v       verbose screen output
- d pdb_dirdirectory to search for PDB data files
- f file_of_filesfile containing a list of PDB data file names
- l filenamelog file name
- o filenameoutput (sequence) file name
- m max_line_lenmaximum length of sequence lines in output

Each of the options is described more fully below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c</td>
<td>Concatenates the sequences of the separate chains in each multiple-chain protein into a single sequence entry in file pdb.seq.</td>
</tr>
<tr>
<td>-r</td>
<td>Rejects all PDB files containing unknown or atypical residues (those that would otherwise have been translated to X).</td>
</tr>
<tr>
<td>-R</td>
<td>Rejects all PDB files containing either unknown or atypical residues (identical to the -r option) OR the ambiguous amino acid residues (those that would otherwise have been translated to B or Z).</td>
</tr>
<tr>
<td>-s</td>
<td>Each PDB sequence line contains an integer that is supposed to be the total number of residues in the chain. As of this writing there are six files in the PDB for which this number is incorrect. Most of these are harmless typos, but in one (file 1L14) an entire line of 13 residues is missing from the sequence. By default, seq_extract ignores errors of this kind, but files containing them can be rejected via the -s option.</td>
</tr>
<tr>
<td>-v</td>
<td>Causes the program to display on the screen, as the extractions proceed, the same information that goes into the .log file.</td>
</tr>
</tbody>
</table>
### Execution Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-d</code></td>
<td>If you specify a directory to be searched via the <code>-d</code> option, <code>seq_extract</code> searches only that directory for PDB files. If you omit the <code>-d</code> option, <code>seq_extract</code> searches first in the <code>$INSIGHT_PDB</code> directory, and then, if that fails, in the current working directory.</td>
</tr>
<tr>
<td><code>-f</code></td>
<td>You may specify a subset of PDB files to search by including their names on the command line. Alternatively or in addition, you may provide a list of PDB filenames in a separate text file via the <code>-f</code> option.</td>
</tr>
<tr>
<td><code>-l</code></td>
<td>Renames the <code>.log</code> file. By specifying a full pathname you can use this option to place the file in a directory other than the one in which <code>seq_extract</code> is executed.</td>
</tr>
<tr>
<td><code>-o</code></td>
<td>Renames the sequence database file. By specifying a full pathname you can use this option to place the file in a directory other than the one in which <code>seq_extract</code> is executed.</td>
</tr>
<tr>
<td><code>-m</code></td>
<td>Changes the maximum allowed length of the sequence lines in <code>pdb.seq</code>. The default line length is 70 characters.</td>
</tr>
</tbody>
</table>
D. seq_extract Utility
## Sequence Alignment Matrices

### Identity Matrix

<table>
<thead>
<tr>
<th></th>
<th>ALA</th>
<th>ARG</th>
<th>ASN</th>
<th>ASP</th>
<th>CYS</th>
<th>GLN</th>
<th>GLU</th>
<th>HYS</th>
<th>ILE</th>
<th>LEU</th>
<th>LYS</th>
<th>MET</th>
<th>PHE</th>
<th>PRO</th>
<th>SER</th>
<th>THR</th>
<th>TRP</th>
<th>TYR</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALA</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>ARG</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>ASN</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>ASP</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>CYS</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>GLN</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>GLU</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>HIS</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>ILE</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>LEU</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>LYS</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>MET</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>PHE</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>PRO</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>SER</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>THR</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>TRP</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>TYR</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>VAL</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
E. Matrices

### Codon Substitution Matrix

<table>
<thead>
<tr>
<th></th>
<th>ALA</th>
<th>ARG</th>
<th>ASN</th>
<th>ASP</th>
<th>CYM</th>
<th>GLN</th>
<th>GLU</th>
<th>HIS</th>
<th>ILE</th>
<th>LEU</th>
<th>lys</th>
<th>PHE</th>
<th>PRO</th>
<th>SER</th>
<th>THR</th>
<th>TRP</th>
<th>tyr</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALA</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ARG</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ASN</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ASP</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CYM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>GLN</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GLU</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HIS</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ILE</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>LEU</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>lys</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PHE</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PRO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SER</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>THR</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TRP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>tyr</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VAL</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Dayhoff Evolutionary Mutation Matrix

<table>
<thead>
<tr>
<th></th>
<th>ALA</th>
<th>ARG</th>
<th>ASN</th>
<th>ASP</th>
<th>CYM</th>
<th>GLN</th>
<th>GLU</th>
<th>HIS</th>
<th>ILE</th>
<th>LEU</th>
<th>lys</th>
<th>PHE</th>
<th>PRO</th>
<th>SER</th>
<th>THR</th>
<th>TRP</th>
<th>tyr</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALA</td>
<td>2</td>
<td>-1</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>ARG</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ASN</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ASP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CYM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GLU</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HIS</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>ILE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LEU</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lys</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PHE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PRO</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SER</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>THR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TRP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>tyr</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VAL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

276 Homology
### Hydrophobicity Matrix

<table>
<thead>
<tr>
<th></th>
<th>ALA</th>
<th>ARG</th>
<th>ASN</th>
<th>ASP</th>
<th>CYS</th>
<th>GLN</th>
<th>GLU</th>
<th>GLY</th>
<th>HIS</th>
<th>ILE</th>
<th>LEU</th>
<th>LYS</th>
<th>MET</th>
<th>PHE</th>
<th>PRO</th>
<th>SER</th>
<th>THR</th>
<th>TRP</th>
<th>TYR</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALA</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>15</td>
<td>11</td>
<td>15</td>
<td>6</td>
<td>14</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>15</td>
<td>6</td>
<td>14</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>ARG</td>
<td>2</td>
<td>6</td>
<td>9</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ASN</td>
<td>10</td>
<td>3</td>
<td>16</td>
<td>12</td>
<td>3</td>
<td>15</td>
<td>13</td>
<td>10</td>
<td>14</td>
<td>6</td>
<td>5</td>
<td>12</td>
<td>6</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>ASP</td>
<td>5</td>
<td>13</td>
<td>6</td>
<td>11</td>
<td>15</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>CYS</td>
<td>2</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>14</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>13</td>
<td>15</td>
<td>6</td>
<td>15</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>GLN</td>
<td>10</td>
<td>8</td>
<td>15</td>
<td>11</td>
<td>10</td>
<td>16</td>
<td>12</td>
<td>11</td>
<td>15</td>
<td>9</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>GLU</td>
<td>6</td>
<td>12</td>
<td>15</td>
<td>6</td>
<td>12</td>
<td>16</td>
<td>7</td>
<td>11</td>
<td>5</td>
<td>5</td>
<td>15</td>
<td>4</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>9</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>GLY</td>
<td>15</td>
<td>3</td>
<td>10</td>
<td>15</td>
<td>11</td>
<td>7</td>
<td>16</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>6</td>
<td>14</td>
<td>13</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>HIS</td>
<td>11</td>
<td>7</td>
<td>14</td>
<td>11</td>
<td>15</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>14</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>ILE</td>
<td>15</td>
<td>1</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>4</td>
<td>16</td>
<td>15</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>LEU</td>
<td>15</td>
<td>5</td>
<td>2</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>4</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>14</td>
<td>15</td>
<td>13</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>LYS</td>
<td>6</td>
<td>10</td>
<td>12</td>
<td>10</td>
<td>11</td>
<td>15</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>MET</td>
<td>14</td>
<td>0</td>
<td>8</td>
<td>15</td>
<td>5</td>
<td>4</td>
<td>14</td>
<td>10</td>
<td>16</td>
<td>15</td>
<td>4</td>
<td>16</td>
<td>16</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>12</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>PHE</td>
<td>14</td>
<td>0</td>
<td>9</td>
<td>14</td>
<td>8</td>
<td>12</td>
<td>9</td>
<td>15</td>
<td>15</td>
<td>4</td>
<td>16</td>
<td>16</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>12</td>
<td>15</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>PRO</td>
<td>14</td>
<td>4</td>
<td>11</td>
<td>14</td>
<td>12</td>
<td>9</td>
<td>15</td>
<td>12</td>
<td>12</td>
<td>7</td>
<td>12</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>15</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>SER</td>
<td>10</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>11</td>
<td>7</td>
<td>16</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>7</td>
<td>13</td>
<td>15</td>
<td>16</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>15</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>THR</td>
<td>10</td>
<td>2</td>
<td>11</td>
<td>15</td>
<td>11</td>
<td>7</td>
<td>16</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>7</td>
<td>13</td>
<td>15</td>
<td>16</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>15</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TYR</td>
<td>14</td>
<td>0</td>
<td>2</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>5</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>15</td>
<td>13</td>
<td>15</td>
<td>13</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>VAL</td>
<td>10</td>
<td>3</td>
<td>16</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>5</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>15</td>
<td>13</td>
<td>15</td>
<td>13</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

### Input Databases Command Matrices

#### Identity Matrix
### Codon Substitution Matrix

|       | A  | R  | N  | D  | C  | Q  | E  | G  | H  | I  | L  | K  | M  | F  | P  | S  | T  | W  | Y  | V  | B  | Z  | X |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| A     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| R     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| N     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| D     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| C     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| Q     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| E     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| G     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| H     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| I     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| L     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| K     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| M     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| F     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| P     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| S     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| T     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| W     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| Y     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| V     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| B     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| Z     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
| X     | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 | .5 |
### Dayhoff Evolutionary Mutation Matrix

|   | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | V | B | Z | X |
| A | -2 | -2 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | -2 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| H | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| I | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| K | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Q | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| T | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Hydrophobicity Matrix

|   | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | V | B | Z | X |
| A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| I | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| K | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Q | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| T | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Homology 279
## E. Matrices

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | B | Z | X |
| A | 17 | | | | | | | | | | | | | | | | | | | | | | | | |
| R | -2 | 17 | | | | | | | | | | | | | | | | | | | | | | | |
| N | 0 | 7 | 17 | | | | | | | | | | | | | | | | | | | | | | |
| D | 1 | 13 | 11 | 17 | | | | | | | | | | | | | | | | | | | | | | |
| C | 17 | -2 | 7 | 1 | 17 | | | | | | | | | | | | | | | | | | | | | | |
| Q | 0 | 6 | 10 | 10 | 0 | 17 | | | | | | | | | | | | | | | | | | | | | | |
| E | 3 | 11 | 13 | 16 | 3 | 11 | 17 | | | | | | | | | | | | | | | | | | | | | | |
| G | 16 | -2 | 0 | 2 | 16 | 10 | 4 | 17 | | | | | | | | | | | | | | | | | | | | | | |
| H | 10 | 4 | 14 | 8 | 10 | 10 | 10 | 11 | 17 | | | | | | | | | | | | | | | | | | | | | | |
| I | 18 | -5 | 6 | 0 | 18 | 7 | 1 | 14 | 8 | 17 | | | | | | | | | | | | | | | | | | | | | | |
| L | 16 | -5 | 6 | 0 | 16 | 7 | 1 | 14 | 0 | 17 | 17 | | | | | | | | | | | | | | | | | | | | |
| K | 3 | 13 | 11 | 17 | 1 | 10 | 15 | 3 | 5 | 0 | 0 | 17 | | | | | | | | | | | | | | | | | | | | |
| M | 14 | -6 | 6 | -2 | 16 | 7 | 0 | 14 | 8 | 17 | 16 | 0 | 17 | | | | | | | | | | | | | | | | | | | | |
| P | 14 | -6 | 6 | -2 | 14 | 6 | 0 | 13 | 7 | 16 | 12 | 0 | 17 | 17 | | | | | | | | | | | | | | | | | | | | |
| S | 14 | 0 | 10 | 4 | 14 | 11 | 6 | 16 | 13 | 13 | 15 | 4 | 11 | 11 | 17 | | | | | | | | | | | | | | | | | | | | |
| T | 17 | -2 | 0 | 3 | 18 | 10 | 4 | 17 | 11 | 14 | 14 | 0 | 14 | 14 | 10 | 10 | 17 | | | | | | | | | | | | | | | | | | | | |
| W | 17 | -3 | 7 | 1 | 17 | 8 | 3 | 18 | 10 | 18 | 18 | 1 | 16 | 14 | 14 | 16 | 16 | 16 | 17 | | | | | | | | | | | | | | | | | | | | |
| Y | 16 | 0 | 11 | 5 | 12 | 12 | 7 | 14 | 14 | 11 | 12 | 6 | 11 | 11 | 17 | 16 | 13 | 12 | 17 | | | | | | | | | | | | | | | | | | | | |
| V | 16 | 5 | 7 | 0 | 15 | 7 | 1 | 14 | 8 | 17 | 17 | 1 | 18 | 18 | 13 | 14 | 10 | 16 | 13 | 17 | | | | | | | | | | | | | | | | | | | | |
| B | 3 | 9 | 14 | 4 | 13 | 15 | 8 | 14 | 2 | 3 | 14 | 2 | 2 | 7 | 6 | 5 | 4 | 8 | 3 | 17 | | | | | | | | | | | | | | | | | | | | |
| Z | 7 | 7 | 16 | 11 | 7 | 15 | 12 | 9 | 14 | 5 | 9 | 12 | 5 | 4 | 10 | 9 | 9 | 7 | 11 | 6 | 14 | 17 | | | | | | | | | | | | | | | | | | | | |
| X | 17 | -3 | 0 | 1 | 10 | 0 | 3 | 10 | 10 | 15 | 15 | 2 | 14 | 14 | 14 | 10 | 16 | 17 | 14 | 16 | 16 | 5 | 7 | 17 | | | | | | | | | | | | | | | | | | | | |

---

280  Homology
## Hydrophobicity Scale Values

### Amino Acid Values

<table>
<thead>
<tr>
<th></th>
<th>ENGLEMAN-STEITZ</th>
<th>HOPP-WOODS</th>
<th>KYTE-DOOLITTLE</th>
<th>JANIN</th>
<th>CHOTHIA</th>
<th>EISENBERG-WEISS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHE</td>
<td>-3.7</td>
<td>-2.5</td>
<td>2.8</td>
<td>0.5</td>
<td>0.0</td>
<td>0.61</td>
</tr>
<tr>
<td>MET</td>
<td>-3.4</td>
<td>-1.3</td>
<td>1.9</td>
<td>0.4</td>
<td>-0.24</td>
<td>0.26</td>
</tr>
<tr>
<td>ILE</td>
<td>-3.1</td>
<td>-1.8</td>
<td>4.5</td>
<td>0.7</td>
<td>0.24</td>
<td>0.73</td>
</tr>
<tr>
<td>LEU</td>
<td>-2.8</td>
<td>-1.8</td>
<td>3.8</td>
<td>0.5</td>
<td>-0.12</td>
<td>0.53</td>
</tr>
<tr>
<td>VAL</td>
<td>-2.6</td>
<td>-1.5</td>
<td>4.2</td>
<td>0.6</td>
<td>0.09</td>
<td>0.54</td>
</tr>
<tr>
<td>CYS</td>
<td>-2.0</td>
<td>-1.0</td>
<td>2.5</td>
<td>0.9</td>
<td>0.0</td>
<td>0.04</td>
</tr>
<tr>
<td>TRP</td>
<td>-1.9</td>
<td>-3.4</td>
<td>-0.9</td>
<td>0.3</td>
<td>-0.59</td>
<td>0.37</td>
</tr>
<tr>
<td>ALA</td>
<td>-1.6</td>
<td>-0.5</td>
<td>1.8</td>
<td>0.3</td>
<td>-0.29</td>
<td>0.25</td>
</tr>
<tr>
<td>THR</td>
<td>-1.2</td>
<td>-0.4</td>
<td>-0.7</td>
<td>-0.2</td>
<td>-0.71</td>
<td>-0.18</td>
</tr>
<tr>
<td>GLY</td>
<td>-1.0</td>
<td>0.0</td>
<td>-0.4</td>
<td>0.3</td>
<td>-0.34</td>
<td>0.16</td>
</tr>
<tr>
<td>SER</td>
<td>-0.6</td>
<td>0.3</td>
<td>-0.8</td>
<td>-0.1</td>
<td>-0.75</td>
<td>-0.26</td>
</tr>
<tr>
<td>PRO</td>
<td>0.2</td>
<td>0.0</td>
<td>-1.6</td>
<td>-0.3</td>
<td>-0.90</td>
<td>-0.07</td>
</tr>
<tr>
<td>TYR</td>
<td>0.7</td>
<td>-2.3</td>
<td>-1.3</td>
<td>-0.4</td>
<td>-1.02</td>
<td>0.02</td>
</tr>
<tr>
<td>HIS</td>
<td>3.0</td>
<td>-0.5</td>
<td>-3.2</td>
<td>-0.1</td>
<td>-0.94</td>
<td>-0.40</td>
</tr>
<tr>
<td>GLN</td>
<td>4.1</td>
<td>0.2</td>
<td>-3.5</td>
<td>-0.7</td>
<td>-1.53</td>
<td>-0.69</td>
</tr>
<tr>
<td>ASN</td>
<td>4.8</td>
<td>0.2</td>
<td>-3.5</td>
<td>-0.5</td>
<td>-1.18</td>
<td>-0.64</td>
</tr>
<tr>
<td>GLU</td>
<td>8.2</td>
<td>3.0</td>
<td>-3.5</td>
<td>-0.7</td>
<td>-0.90</td>
<td>-0.62</td>
</tr>
<tr>
<td>LYS</td>
<td>8.8</td>
<td>3.0</td>
<td>-3.9</td>
<td>-1.8</td>
<td>-2.05</td>
<td>-1.1</td>
</tr>
<tr>
<td>ASP</td>
<td>9.2</td>
<td>3.0</td>
<td>-3.5</td>
<td>-0.6</td>
<td>-1.02</td>
<td>-0.72</td>
</tr>
<tr>
<td>ARG</td>
<td>12.3</td>
<td>3.0</td>
<td>-4.5</td>
<td>-1.4</td>
<td>-2.71</td>
<td>-1.8</td>
</tr>
</tbody>
</table>
### F. Hydrophobicity Scale Values

**Threshold Values**

<table>
<thead>
<tr>
<th></th>
<th>Hydrophobic</th>
<th>Hydrophilic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1.4</td>
<td>1.85</td>
</tr>
<tr>
<td></td>
<td>-0.75</td>
<td>1.65</td>
</tr>
<tr>
<td></td>
<td>0.70</td>
<td>-2.4</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>-0.45</td>
</tr>
<tr>
<td></td>
<td>-0.47</td>
<td>-0.98</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>-0.51</td>
</tr>
</tbody>
</table>
The Homology module is equipped to take advantage of publicly available sequence databases. Since MSI does not distribute these databases, the list below provides the necessary information.

**Protein Sequence Databases**

**NBRF**

The NBRF effort is supported by the Protein Identification Resource (PIR) project (Barker et al. 1990).

**Direct all U.S. inquires to:**

National Center for Biotechnology Information National Library of Medicine Bldg. 38A Room 8N - 803 8600 Rockville Pike Bethesda, MD 20894

Telephone: 301-496-2475 Fax: 301-480-9241 E-Mail: info@ncbi.nlm.nih.gov

The databases can be downloaded by anonymous ftp from ncbi.nlm.nih.gov (130. 14. 25. 1) in:
directory: repository/PIR/vms/

pir1. seq. Z
pir2. seq. Z
pir3. seq. Z
G. Sequence Databases

**European inquiries:**

MIPS: Martinsrieder Institut für Proteinsequenzen
Max-Planck-Institut für Biochemie
D-8033 Martinsried
MEWES@VAX1.MIPS.MPG.DBP.DE
Germany

**Asian or Australian inquiries:**

JIPID: International Protein Information Database in Japan
Science University of Tokyo
2669 Yamazaki
Noda 278
Japan

---

**Swiss-Prot**

Refer to the information given for the EMBL Data Library (Kahn and Cameron 1990).

---

**DNA Sequence Databases**

**GenBank**

Direct all U.S. inquires to:

National Center for Biotechnology Information
Bldg. 38A Room 8N - 803
8600 Rockville Pike
Bethesda, MD 20894

---

284 Homology
DNA Sequence Databases

Telephone: 301-496-2475
Fax: 301-480-9241
E-Mail: info@ncbi.nlm.nih.gov

The databases can be downloaded by anonymous ftp from
ncbi.nlm.nih.gov (130. 14. 25. 1) in directory:

Genbank: ‘genbank’ directory compressed in *.z files.

EMBL Data Library

All inquiries (Khan and Cameron 1990):

Data Submissions Telephone: +49-6221-387-258
EMBL Data Library Telefax: +49-6221-387-519
Postfach 10.2209 Telex: 461613 (embl d)
6900 Heidelberg
Germany

datasubs@EMBL-Heidelberg.DE (data submission)
datalib@EMBL-Heidelberg.DE (questions requiring a personal
response)
DATALIB@EMBL.BITNET (general inquires)
G. Sequence Databases
align123 Standalone

To run align123 from the command line, enter align123 and return. This documentation is the same as the on-line help text for the align123 standalone program written by Toby Gibson, Des Higgins, and Julie Thompson (EMBL, Heidelberg, Germany. May 1994).

HELP 1: General help for align123

align123 is a general purpose multiple alignment program for DNA or proteins.

SEQUENCE INPUT: all sequences must be in 1 file, one after another. Seven formats are automatically recognized: NBRF/PIR, EMBL/SWISSPROT, Pearson (FASTA), Clustal (*.aln), GCG/MSF (Pileup), GCG9/RSF and GDE flat file. All non-alphabetic characters (spaces, digits, punctuation marks) are ignored except “-” which is used to indicate a GAP (“.” in GCG/MSF).

To do a MULTIPLE ALIGNMENT on a set of sequences, use item 1 from this menu to INPUT them; go to menu item 2 to do the multiple alignment.

PROFILE ALIGNMENTS (menu item 3) are used to align 2 alignments. Use this to add a new sequence to an old alignment, or to use secondary structure to guide the alignment process. GAPS in the old alignments are indicated using the “-” character. PROFILES can be input in ANY of the allowed formats; just use “-” (or “.” for MSF/RSF) for each gap position.

PHYLOGENETIC TREES (menu item 4) can be calculated from old alignments (read in with “-” characters to indicate gaps) OR after a multiple alignment while the alignment is still in memory.
The program tries to automatically recognize the different file formats used and to guess whether the sequences are amino acid or nucleotide. This is not always foolproof.

FASTA and NBRF/PIR formats are recognized by having a “>” as the first character in the file.

EMBL/Swiss Prot formats are recognized by the letters ID at the start of the file (the token for the entry name field).

CLUSTAL format is recognized by the word CLUSTAL at the beginning of the file.

GCG/MSF format is recognized by one of the following:
- The word PileUp at the start of the file.
- The word !!AA_MULTIPLE_ALIGNMENT or !!NA_MULTIPLE_ALIGNMENT at the start of the file.
- The word MSF on the first line of the line, and the characters .. at the end of this line.

GCG/RSF format is recognized by the word !!RICH_SEQUENCE at the beginning of the file.

If 85% or more of the characters in the sequence are from A,C,G,T,U or N, the sequence will be assumed to be nucleotide. This works in 97.3% of cases but watch out!

**HELP 2: Help for multiple alignments**

If you have already loaded sequences, use menu item 1 to do the complete multiple alignment. You will be prompted for 2 output files: 1 for the alignment itself; another to store a dendrogram that describes the similarity of the sequences to each other.

Multiple alignments are carried out in 3 stages (automatically done from menu item 1 ...Do complete multiple alignments now):

1. All sequences are compared to each other (pairwise alignments);

2. A dendrogram (like a phylogenetic tree) is constructed, describing the approximate groupings of the sequences by similarity (stored in a file).
3. The final multiple alignment is carried out, using the dendrogram as a guide.

PAIRWISE ALIGNMENT parameters control the speed/sensitivity of the initial alignments.

MULTIPLE ALIGNMENT parameters control the gaps in the final multiple alignments.

RESET GAPS (menu item 7) will remove any new gaps introduced into the sequences during multiple alignment if you wish to change the parameters and try again. This only takes effect just before you do a second multiple alignment. You can make phylogenetic trees after alignment whether or not this is ON. If you turn this OFF, the new gaps are kept even if you do a second multiple alignment. This allows you to iterate the alignment gradually. Sometimes, the alignment is improved by a second or third pass.

SCREEN DISPLAY (menu item 8) can be used to send the output alignments to the screen as well as to the output file.

You can skip the first stages (pairwise alignments; dendrogram) by using an old dendrogram file (menu item 3); or you can just produce the dendrogram with no final multiple alignment (menu item 2).

OUTPUT FORMAT: Menu item 9 (format options) allows you to choose from 5 different alignment formats (CLUSTAL, GCG, NBRF/PIR, PHYLIP and GDE).

 HELP 3: Help for pairwise alignment parameters

A distance is calculated between every pair of sequences and these are used to construct the dendrogram which guides the final multiple alignment. The scores are calculated from separate pairwise alignments. These can be calculated using 2 methods: dynamic programming (slow but accurate) or by the method of Wilbur and Lipman (extremely fast but approximate).

You can choose between the 2 alignment methods using menu option 8. The slow/accurate method is fine for short sequences but will be VERY SLOW for many (e.g. >20) long (e.g. >1000 residue) sequences.
SLOW/ACCURATE alignment parameters:
These parameters do not have any affect on the speed of the align-
ments. They are used to give initial alignments which are then res-
cored to give percent identity scores. These % scores are the ones
which are displayed on the screen. The scores are converted to dis-
tances for the trees.

1. **Gap Open Penalty**: the penalty for opening a gap in the align-
   ment.

2. **Gap extension penalty**: the penalty for extending a gap by 1
   residue.

3. **Protein weight matrix**: the scoring table which describes the
   similarity of each amino acid to each other.

4. **DNA weight matrix**: the scores assigned to matches and mis-
   matches (including IUB ambiguity codes).

FAST/APPROXIMATE alignment parameters:
These similarity scores are calculated from fast, approximate, glo-
bal alignments, which are controlled by 4 parameters. 2 techniques
are used to make these alignments very fast:

1. only exactly matching fragments (k-tuples) are considered;
2. only the “best” diagonals (the ones with most k-tuple matches)
   are used.

**K-TUPLE SIZE**: This is the size of exactly matching fragment that
is used. INCREASE for speed (max= 2 for proteins; 4 for DNA),
DECREASE for sensitivity. For longer sequences (e.g. >1000 resi-
dues) you may need to increase the default.

**GAP PENALTY**: This is a penalty for each gap in the fast align-
ments. It has little affect on the speed or sensitivity except for
extreme values.

**TOP DIAGONALS**: The number of k-tuple matches on each diag-
onal (in an imaginary dot-matrix plot) is calculated. Only the best
ones (with most matches) are used in the alignment. This parame-
ter specifies how many. Decrease for speed; increase for sensitivity.

**WINDOW SIZE**: This is the number of diagonals around each of
the “best” diagonals that will be used. Decrease for speed; increase
for sensitivity.
HELP 4: Help for multiple alignment parameters

These parameters control the final multiple alignment. This is the core of the program and the details are complicated. To fully understand the use of the parameters and the scoring system, you will have to refer to the documentation.

Each step in the final multiple alignment consists of aligning two alignments or sequences. This is done progressively, following the branching order in the GUIDE TREE. The basic parameters to control this are two gap penalties and the scores for various identical/non-identical residues.

1. and 2. The GAP PENALTIES are set by menu items 1 and 2. These control the cost of opening up every new gap and the cost of every item in a gap. Increasing the gap opening penalty will make gaps less frequent. Increasing the gap extension penalty will make gaps shorter. Terminal gaps are not penalized.

3. The DELAY DIVERGENT SEQUENCES switch, delays the alignment of the most distantly related sequences until after the most closely related sequences have been aligned. The setting shows the percent identity level required to delay the addition of a sequence; sequences that are less identical than this level to any other sequences will be aligned later.

4. The TRANSITION WEIGHT gives transitions (A <-> G or C <-> T i.e. purine-purine or pyrimidine-pyrimidine substitutions) a weight between 0 and 1; a weight of zero means that the transitions are scored as mismatches, while a weight of 1 gives the transitions the match score. For distantly related DNA sequences, the weight should be near to zero; for closely related sequences it can be useful to assign a higher score.

5. PROTEIN WEIGHT MATRIX leads to a new menu where you are offered a choice of weight matrices. The default for proteins is the BLOSUM series of matrices by Jorja and Steven Henikoff. Note, a series is used! The actual matrix that is used depends on how similar the sequences to be aligned at this alignment step are. Different matrices work differently at each evolutionary distance.
6. DNA WEIGHT MATRIX leads to a new menu where a single matrix (not a series) can be selected. The default is the matrix used by BESTFIT for comparison of nucleic acid sequences. Further help is offered in the weight matrix menu.

7. In the weight matrices, you can use negative as well as positive values if you wish, although the matrix will be automatically adjusted to all positive scores, unless the NEGATIVE MATRIX option is selected.

8. PROTEIN GAP PARAMETERS displays a menu allowing you to set some Gap Penalty options which are only used in protein alignments.

**HELP A: Help for protein gap parameters.**

1. RESIDUE SPECIFIC PENALTIES are amino acid specific gap penalties that reduce or increase the gap opening penalties at each position in the alignment or sequence. See the documentation for details. As an example, positions that are rich in glycine are more likely to have an adjacent gap than positions that are rich in valine.

2. and 3. HYDROPHILIC GAP PENALTIES are used to increase the chances of a gap within a run (5 or more residues) of hydrophilic amino acids; these are likely to be loop or random coil regions where gaps are more common. The residues that are “considered” to be hydrophilic are set by menu item 3.

4. GAP SEPARATION DISTANCE tries to decrease the chances of gaps being too close to each other. Gaps that are less than this distance apart are penalized more than other gaps. This does not prevent close gaps; it makes them less frequent, promoting a block-like appearance of the alignment.

5. END GAP SEPARATION treats end gaps just like internal gaps for the purposes of avoiding gaps that are too close (set by GAP SEPARATION DISTANCE above). If you turn this off, end gaps will be ignored for this purpose. This is useful when you wish to align fragments where the end gaps are not biologically meaningful.
HELP 5: Help for output format options.

Five output formats are offered. You can choose more than one (or all 5 if you wish).

CLUSTAL format output is a self explanatory alignment format. It shows the sequences aligned in blocks. It can be read in again at a later date to (for example) calculate a phylogenetic tree or add a new sequence with a profile alignment.

GCG output can be used by any of the GCG programs that can work on multiple alignments (e.g. PRETTY, PROFILEMAKE, PLOTALIGN). It is the same as the GCG .msf format files (multiple sequence file); new in version 7 of GCG.

PHYLIP format output can be used for input to the PHYLIP package of Joe Felsenstein. This is an extremely widely used package for doing every imaginable form of phylogenetic analysis (MUCH more than the modest introduction offered by this program).

NBRF/PIR: this is the same as the standard PIR format with ONE ADDITION. Gap characters “-” are used to indicate the positions of gaps in the multiple alignment. These files can be re-used as input in any part of clustal that allows sequences (or alignments or profiles) to be read in.

GDE: this format is used by the GDE package of Steven Smith.

GDE OUTPUT CASE: sequences in GDE format may be written in either upper or lower case.

ALIGN123 SEQUENCE NUMBERS: residue numbers may be added to the end of the alignment lines in align123 format.

OUTPUT ORDER is used to control the order of the sequences in the output alignments. By default, the order corresponds to the order in which the sequences were aligned (from the guide tree/dendrogram), thus automatically grouping closely related sequences. This switch can be used to set the order to the same as the input file.

PARAMETER OUTPUT: This option allows you to save all your parameter settings in a parameter file. This file can be used subsequently to rerun align123 using the same parameters.
HELP 6: Help for profile and structure alignments

By PROFILE ALIGNMENT, we mean alignment using existing alignments. Profile alignments allow you to store alignments of your favorite sequences and add new sequences to them in small bunches at a time. A profile is simply an alignment of one or more sequences (e.g. an alignment output file from align123). Each input can be a single sequence. One or both sets of input sequences may include secondary structure assignments or gap penalty masks to guide the alignment.

The profiles can be in any of the allowed input formats with “-” characters used to specify gaps (except for MSF/RSF where “.” is used).

You have to specify the 2 profiles by choosing menu items 1 and 2 and giving 2 file names. Then Menu item 3 will align the 2 profiles to each other. Secondary structure masks in either profile can be used to guide the alignment.

Menu item 4 will take the sequences in the second profile and align them to the first profile, 1 at a time. This is useful to add some new sequences to an existing alignment, or to align a set of sequences to a known structure. In this case, the second profile need not be pre-aligned.

The alignment parameters can be set using menu items 5, 6 and 7. These are EXACTLY the same parameters as used by the general, automatic multiple alignment procedure. The general multiple alignment procedure is simply a series of profile alignments. Carrying out a series of profile alignments on larger and larger groups of sequences, allows you to manually build up a complete alignment, if necessary editing intermediate alignments.

SECONDARY STRUCTURE OPTIONS. Menu Option 0 allows you to set secondary structure parameters. If a solved structure is available, it can be used to guide the alignment by raising gap penalties within secondary structure elements, so that gaps will preferentially be inserted into unstructured surface loop regions. Alternatively, a user-specified gap penalty mask can be supplied for a similar purpose.
HELP B: Help for secondary structure / gap penalty masks

A gap penalty mask is a series of numbers between 1 and 9, one per position in the alignment. Each number specifies how much the gap opening penalty is to be raised at that position (raised by multiplying the basic gap opening penalty by the number) i.e. a mask figure of 1 at a position means no change in gap opening penalty; a figure of 4 means that the gap opening penalty is four times greater at that position, making gaps 4 times harder to open.

The format for gap penalty masks and secondary structure masks is explained in the help under option 0 (secondary structure options).

HELP B: Help for secondary structure / gap penalty masks

The use of secondary structure-based penalties has been shown to improve the accuracy of multiple alignment. Therefore align123 now allows gap penalty masks to be supplied with the input sequences. The masks work by raising gap penalties in specified regions (typically secondary structure elements) so that gaps are preferentially opened in the less well conserved regions (typically surface loops).

Options 1 and 2 control whether the input secondary structure information or gap penalty masks will be used.

Option 3 controls whether the secondary structure and gap penalty masks should be included in the output alignment.

Options 4 and 5 provide the value for raising the gap penalty at core Alpha Helical (A) and Beta Strand (B) residues. In CLUSTAL format, capital residues denote the A and B core structure notation. Basic gap penalties are multiplied by the amount specified.

Option 6 provides the value for the gap penalty in Loops. By default this penalty is not raised. In CLUSTAL format, loops are specified by “.” in the secondary structure notation.

Option 7 provides the value for setting the gap penalty at the ends of secondary structures. Ends of secondary structures are observed to grow and/or shrink in related structures. Therefore by default these are given intermediate values, lower than the core
penalties. All secondary structure read in as lower case in CLUSTAL format gets the reduced terminal penalty.

Options 8 and 9 specify the range of structure termini for the intermediate penalties. In the alignment output, these are indicated as lower case. For Alpha Helices, by default, the range spans the end helical turn. For Beta Strands, the default range spans the end residue and the adjacent loop residue, since sequence conservation often extends beyond the actual H-bonded Beta Strand.

align123 can read the masks from SWISS-PROT, CLUSTAL or GDE format input files. For many 3-D protein structures, secondary structure information is recorded in the feature tables of SWISS-PROT database entries. You should always check that the assignments are correct - some are quite inaccurate. align123 looks for SWISS-PROT HELIX and STRAND assignments e.g.

```
FT HELIX 100 115
FT STRAND 118 119
```

The structure and penalty masks can also be read from CLUSTAL alignment format as comment lines beginning "!SS_" or "!GM_" e.g.

```
!SS_HBA_HUMA ..aaaAAAAAAAAaa.aaaaaaaaaaaaaaaaaaaaaaaa............aaaaaAAAA
!GM_HBA_HUMA 1122244444422122242244444442222222221111111111222444444
HBA_HUMA VLSFADTKTVKANGIKVHAGEYGAELERMFLSFTKTYFPHFDLSHGSQVKGHGK
```

Note that the mask itself is a set of numbers between 1 and 9 each of which is assigned to the residue(s) in the same column below.

In GDE flat file format, the masks are specified as text and the names must begin with SS_ or GM_.

Either a structure or penalty mask or both may be used. If both are included in an alignment, the user will be asked which is to be used.
HELP C: Help for secondary structure / gap penalty mask output options

The options in this menu let you choose whether or not to include the masks in the align123 output alignments. Showing both is useful for understanding how the masks work. The secondary structure information is itself useful in judging the alignment quality and in seeing how residue conservation patterns vary with secondary structure.

HELP 7: Help for phylogenetic trees

1. Before calculating a tree, you must have an ALIGNMENT in memory. This can be input in any format or you should have just carried out a full multiple alignment and the alignment is still in memory. Remember YOU MUST ALIGN THE SEQUENCES FIRST!!!!

   The method used is the NJ (Neighbor Joining) method of Saitou and Nei. First you calculate distances (percent divergence) between all pairs of sequence from a multiple alignment; second you apply the NJ method to the distance matrix.

2. EXCLUDE POSITIONS WITH GAPS? With this option, any alignment positions where ANY of the sequences have a gap will be ignored. This means that “like” will be compared to “like” in all distances. It also, automatically throws away the most ambiguous parts of the alignment, which are concentrated around gaps (usually). The disadvantage is that you may throw away much of the data if there are many gaps.

3. CORRECT FOR MULTIPLE SUBSTITUTIONS? For small divergence (say <10%) this option makes no difference. For greater divergence, this option corrects for the fact that observed distances underestimate actual evolutionary distances. This is because, as sequences diverge, more than one substitution will happen at many sites. However, you only see one difference when you look at the present day sequences. Therefore, this option has the effect of stretching branch lengths
in trees (especially long branches). The corrections used here (for DNA or proteins) are both due to Motoo Kimura. See the documentation for details.

For very divergent sequences, the distances cannot be reliably corrected. You will be warned if this happens. Even if none of the distances in a data set exceed the reliable threshold, if you bootstrap the data, some of the bootstrap distances may randomly exceed the safe limit.

4. To calculate a tree, use option 4 (DRAW TREE NOW). This gives an UNROOTED tree and all branch lengths. The root of the tree can only be inferred by using an outgroup (a sequence that you are certain branches at the outside of the tree .... certain on biological grounds) OR if you assume a degree of constancy in the “molecular clock”, you can place the root in the “middle” of the tree (roughly equidistant from all tips).

5. BOOTSTRAPPING is a method for deriving confidence values for the groupings in a tree (first adapted for trees by Joe Felsenstein). It involves making N random samples of sites from the alignment (N should be LARGE, e.g. 500 - 1000); drawing N trees (1 from each sample) and counting how many times each grouping from the original tree occurs in the sample trees. You must supply a seed number for the random number generator. Different runs with the same seed will give the same answer. See the documentation for details.

6. OUTPUT FORMATS: three different formats are allowed. None of these displays the tree visually. You must make the tree yourself (on paper) using the results OR get the PHYLIP package and use the tree drawing facilities there. (Get the PHYLIP package anyway if you are interested in trees).

**HELP 8: Help for choosing a weight matrix**

For protein alignments, you use a weight matrix to determine the similarity of non-identical amino acids. For example, Tyr aligned with Phe is usually judged to be “better” than Tyr aligned with Pro.
There are three “in-built” series of weight matrices offered. Each consists of several matrices which work differently at different evolutionary distances. To see the exact details, read the documentation. Crudely, we store several matrices in memory, spanning the full range of amino acid distance (from almost identical sequences to highly divergent ones). For very similar sequences, it is best to use a strict weight matrix which only gives a high score to identities and the most favored conservative substitutions. For more divergent sequences, it is appropriate to use “softer” matrices which give a high score to many other frequent substitutions.

1. BLOSUM (Henikoff). These matrices appear to be the best available for carrying out data base similarity (homology searches). The matrices used are: Blosum80, 62, 45 and 30.

2. PAM (Dayhoff). These have been extremely widely used since the late ‘70s. We use the PAM 120, 160, 250 and 350 matrices.

3. GONNET. These matrices were derived using almost the same procedure as the Dayhoff one (above) but are much more up to date and are based on a far larger data set. They appear to be more sensitive than the Dayhoff series. We use the GONNET 40, 80, 120, 160, 250 and 350 matrices.

We also supply an identity matrix which gives a score of 1.0 to two identical amino acids and a score of zero otherwise. This matrix is not very useful. Alternatively, you can read in your own (just one matrix, not a series).

A new matrix can be read from a file on disk, if the filename consists only of lower case characters. The values in the new weight matrix must be integers and the scores should be similarities. You can use negative as well as positive values if you wish, although the matrix will be automatically adjusted to all positive scores.

For DNA, a single matrix (not a series) is used. Two hard-coded matrices are available:

1. IUB. This is the default scoring matrix used by BESTFIT for the comparison of nucleic acid sequences. X’s and N’s are treated as matches to any IUB ambiguity symbol. All matches score 1.9; all mismatches for IUB symbols score 0.

2. ALIGN123(1.6). The previous system used by align123, in which matches score 1.0 and mismatches score 0. All matches for IUB symbols also score 0.
INPUT FORMAT The format used for a new matrix is the same as the BLAST program. Any lines beginning with a # character are assumed to be comments. The first non-comment line should contain a list of amino acids in any order, using the 1 letter code, followed by a * character. This should be followed by a square matrix of integer scores, with one row and one column for each amino acid. The last row and column of the matrix (corresponding to the * character) contain the minimum score over the whole matrix.

HELP 9: Help for command line parameters DATA (sequences)

/INFILE=file.ext :input sequences.
/PROFILE1=file.ext and /PROFILE2=file.ext :profiles (old alignment).

VERBS (do things)

/OPTIONS :list the command line parameters
/HELP or /CHECK :outline the command line params.
/ALIGN :do full multiple alignment
/TREE :calculate NJ tree.
/BOOTSTRAP(=n) :bootstrap a NJ tree (n= number of bootstraps; def. = 1000).
/CONVERT :output the input sequences in a different file format.

PARAMETERS (set things)

***General settings:****

/INTERACTIVE :read command line, then enter normal interactive menus
/QUICKTREE :use FAST algorithm for the alignment guide tree
/NEGATIVE :protein alignment with negative values in matrix
/OUTFILE= :sequence alignment file name
/OUTPUT= :GCG, GDE, PHYLIP or PIR
/OUTORDER= :INPUT or ALIGNED
HELP 9: Help for command line parameters DATA (sequences)

/CASE :LOWER or UPPER (for GDE output only)
/SEQNOS= :OFF or ON (for Clustal output only)

***Fast Pairwise Alignments:***
/KTUPLE=n :word size
/TOPDIAGS=n :number of best diags.
/WINDOW=n :window around best diags.
/PAIRGAP=n :gap penalty
/SCORE :PERCENT or ABSOLUTE

***Slow Pairwise Alignments:***
/PWMATRIX= :Protein weight matrix=BLOSUM, PAM, GONNET, ID or filename
/PWDNAMATRIX= :DNA weight matrix=IUB, ALIGN123 or filename
/PWGAPOPEN=f :gap opening penalty
/PWGAPEXT=f :gap opening penalty

***Multiple Alignments:***
/NEWTREE= :file for new guide tree
/USETREE= :file for old guide tree
/MATRIX= :Protein weight matrix=blosum, pam, gonnet, id or filename
/DNAMATRIX= :DNA weight matrix=IUB, ALIGN123 or filename
/GAPOPEN=f :gap opening penalty
/GAPDIST=n :gap separation pen. range
/NOPGAP :residue-specific gaps off
/NOHGAP :hydrophilic gaps off
/HGAPRESIDUES= :list hydrophilic res.
/MAXDIV=n :% ident. for delay
/TWPEIGHT=f :transitions weighting

***Profile Alignments:***
/PROFILE =Merge two alignments by profile alignment
/NEWTREE1= :file for new guide tree for profile1
H. align123 Standalone

/NEWTREE2= :file for new guide tree for profile2
/USETREE1= :file for old guide tree for profile1
/USETREE2= :file for old guide tree for profile2

***Sequence to Profile Alignments:***
/SEQUENCES :Sequentially add profile2 sequences to profile1 alignment
/NEWTREE= :file for new guide tree
/USETREE= :file for old guide tree

***Structure Alignments:***
/NOSECSTR1 :do not use secondary structure/gap penalty mask for profile 1
/NOSECSTR2 :do not use secondary structure/gap penalty mask for profile 2
/SECSTROUT= :STRUCTURE or MASK or BOTH or NONE output in alignment file
/HELIXGAP=n :gap penalty for helix core residues
/STRANDGAP=n :gap penalty for strand core residues
/LOOPGAP=n :gap penalty for loop regions
/TERMINALGAP=n :gap penalty for structure termini
/HELIXENDIN=n :number of residues inside helix to be treated as terminal
/HELIXENDOUT=n :number of residues outside helix to be treated as terminal
/STRANDENDIN=n :number of residues inside strand to be treated as terminal
/STRANDENDOUT=n :number of residues outside strand to be treated as terminal

***Trees:***
/OPUTTREE=nj OR phylip OR dist
/SEED=n :seed number for bootstraps.
/KIMURA :use Kimura’s correction. /TOSSGAPS :ignore positions with gaps.
HELP 10: Help for tree output format options

Three output formats are offered:
1. Clustal,
2. Phylip,
3. Just the distances.

None of these formats displays the results graphically. To see a graphic representation, get the PHYLIP package and use format 2) below. It can be imported into the PHYLIP programs RETREE, DRAWTREE and DRAWGRAM and displayed graphically.

1. Clustal format output. This format is verbose and lists all of the distances between the sequences and the number of alignment positions used for each. The tree is described at the end of the file. It lists the sequences that are joined at each alignment step and the branch lengths. After two sequences are joined, it is referred to later as a NODE. The number of a NODE is the number of the lowest sequence in that NODE.

2. Phylip format output. This format is the New Hampshire format, used by many phylogenetic analysis packages. It consists of a series of nested parentheses, describing the branching order, with the sequence names and branch lengths. It can be used by the RETREE, DRAWTREE and DRAWGRAM programs of the PHYLIP package to see the trees graphically. This is the same format used during multiple alignment for the guide trees. Some other packages that can read and display New Hampshire format are TreeTool, TreeView, Phylowin and NJPlot.

3. The distances only. This format just outputs a matrix of all the pairwise distances in a format that can be used by the Phylip package. It used to be useful when one could not produce distances from protein sequences in the Phylip package but is now redundant (Protdist of Phylip 3.5 now does this).

4. TOGGLE PHYLIP BOOTSTRAP POSITIONS By default, the bootstrap values are placed on the nodes of the phylip format output tree. This is inaccurate as the bootstrap values should be associated with the tree branches and not the nodes. However,
H. align123 Standalone

this format can be read and displayed by TreeTool, TreeView and Phylowin. An option is available to correctly place the bootstrap values on the branches with which they are associated.
Index

A
Access_Surf command, ProStat pulldown, 112
Alignment pulldown, 93
  Multiple_Sequence, 38, 39, 40, 44, 59
divide-and-conquer algorithm, 47
  Pairwise_Sequence, 38
scoring matrices, 57
Alignment pulldown
  commands in, 107
  Sequence, 136
  Structure, 123
alignment, See sequence alignment and structural alignment
align123
  standalone, 287
theory, 49
Align123 command, Alignment pulldown, 108
amino acid, 255
Append command, Residue pulldown, 104
AssignCoords command, Loops pulldown, 67, 103
AssignCoords command, Sequences pulldown, 99, 101
AssignCoords Loops command, Loops pulldown, 162
AssignCoords Sequences command, Sequences pulldown, 158, 162
Auto_Rotamer command, Residue pulldown, 105
  Boxes pulldown, 91, 102
  Boxes pulldown
    Freeze, 130, 205, 212
    Scroll, 204
    Summarize, 212
  Brookhaven Protein Databank, 159
  Builder module, 168
  See also Insight II User Guide
  Bump command, Measure pulldown, 168
  See also Insight II User Guide
bumps, 69
By_Residue pulldown
  commands in, 109

B
Background_Job
  commands in, 106
basics, 94
best fit
calculated, 64
Blank Object command, Object pulldown, 201
  See also Insight II User Guide
  Boxes pulldown, 91, 102
  Boxes pulldown
    Freeze, 130, 205, 212
    Scroll, 204
    Summarize, 212
  Brookhaven Protein Databank, 159
  Builder module, 168
  See also Insight II User Guide
  Bump command, Measure pulldown, 168
  See also Insight II User Guide
bumps, 69
By_Residue pulldown
  commands in, 109

C
Chou-Fasman method, 77, 78
CLUSTAL W
  see align123
  ClustalW
distance matrix/pairwise alignments, 50
guide tree, 50
multiple alignment algorithm, 49
Neighbor-Joining method, 50
Codon substitution matrix, 255
Color command, 81
Color command, Sequences pulldown, 101
Color Molecule command, Molecule pulldown, 191, 198
  See also Insight II User Guide
Color_By_Structure command, By_Residue pulldown, 109
Color_Conservation command, By_Residue pulldown, 109
Color_Overlap command, Alignment pulldown, 108
command logging, 13
commands, 94
  Access_Surf, 112
  Align123, 108
  Append, 104
AssignCoords, 101, 103
Auto_Rotamer, 105
Color, 101
Color_By_Structure, 109
Color_Conservation, 109
Color_Overlap, 108
Completion_Status, 106
Control_Bkgd_Job, 106
Convert_Format, 101
Copy, 101
Delete, 101, 102, 104
Display, 103
EndRepair, 110
Explore, 110
Extract, 101
Freeze, 102
Generate, 103
GenericDis, 110
Get command, 100
Hydrophobicity, 109
Initialize, 102
Input, 105
Kill_Bkgd_Job, 107
List, 105
Load_Alignment, 107
LocalAlignment, 109
Manual_Rotamer, 104
Motif_Search, 106, 118
use of, 119
Multiple_Sequence, 107
Output, 105
Pairwise_Sequence, 107
Percent_Identity, 109
Put, 101
Relax, 110
Rename, 104
Repeat, 104
Replace, 104
Residue_Dihedral, 111
Run_FASTA, 105
Search, 103
SecondaryClassify, 112
SecondaryPred, 109
Setup_Bkgd_Job, 106
SpliceRepair, 110
Structure, 107
Struct_Check, 111
Summarize, 102
Superimpose_Aln, 108
UnFreeze, 102
commands in, 102
comparison matrix
  in Needleman and Wunsch algorithm, 26
  maximum pathway, 28, 34
  processing, 26
Completion_Status command, Background_Job pulldown, 106
conformation
  using rotamer libraries, 70
Control_Bkgd_Job command, Background_Job pulldown, 106
Convert_Format command, Sequences pulldown, 101
Copy command, Sequences pulldown, 101
Copy Sequences command, Sequences pulldown, 169
creating, 93

D

Databases pulldown, 114, 269
commands in, 105
Dayhoff evolutionary mutation matrix, 255
Delete command, Boxes pulldown, 102
Delete command, Residue pulldown, 104
Delete command, Sequences pulldown, 101
Discover
  and Homology, 14, 167
  energy constraints, 74
  energy minimization, 74, 170, 171
  molecular dynamics, 75
  molecular dynamics, 172
  potential energy equation, 72
Display command, Loops pulldown, 66, 103
Display Loops command, Loops pulldown, 160, 162
Display Molecule command, Molecule pulldown, 194
  See also Insight II User Guide
distance, 64
distance criteria, 67
distance matrix, 20
divide-and-conquer algorithm, 47
divide-and-conquer algorithm, automatic, 47
end repair, 169
EndRepair command, Refine pulldown, 110, 169
energy
  constraints, 74
  energy minimization, 74, 170, 171
Explore command, Refine pulldown, 74, 75, 110, 172
Extract, 87
Extract command, Sequences pulldown, 101
Extract Sequences command, Sequences pulldown, 122, 123, 192

FASTA program, 16, 17, 18
files
  user scoring matrix, 256
Folder pulldown
  Restore
  Save, 13
Folder pulldown
  Restore, 211
Forcefield pulldown, 168
  See also Insight II: User Guide
Freeze Boxes command, Boxes pulldown, 130, 205
Freeze command, Boxes pulldown, 102

gap length penalty, 33
gap penalty, 32
Gaps pulldown
  Scroll, 93
Generate command, Loops pulldown, 67, 103
Generate Loops command, Loops pulldown, 160
GenericDis command, Refine pulldown, 75, 110
Geometry command, Modify pulldown, 168
  See also Insight II User Guide
Get command, Sequences pulldown, 100
Get Molecule command, Molecule pulldown, 191
  See also Insight II User Guide
Get Sequences command, Sequences pulldown, 87, 122, 216

Homology program
  invoking, 12
  methodology, 113
homology, structural approaches, 16
homology, structural approaches, 122
Hydrogen command, Modify pulldown, 168
  See also Insight II User Guide
hydrophobicity, 255
  profiles, 79
  scales, 80, 281
Hydrophobicity command, By_Residue pulldown, 109

identity matrix, 26, 255
initial region, scoring, 18
Initialize, 91
Initialize command, Boxes pulldown, 102
Input command, Databases pulldown, 105
Input Databases command, Databases pulldown, 256
Insight II and Homology, 13
installation
  See Installation Guide
issuing with scrolling buttons, 94

Kill_Bkgd_Job command, Background_Job pulldown, 107
ktup value (FASTA program), 17

Label command, 81
Lagrange multiplier method, linearized, 67
List command, Residue pulldown, 105
List Subset command, Subset pulldown, 213
See also *Insight II* User Guide
Load_Alignment command, Alignment pulldown, 107
local regions
scoring, 17
LocalAlignment command, Alignment pulldown, 109
loops
coordinate assignment, 64, 159, 230
displaying, 66, 159, 225
generating, 67
searching for, 64, 159, 224
Loops pulldown
AssignCoords, 67
Display, 66
Generate, 67
Search, 66
Loops pulldown
AssignCoords, 162
commands in, 103
Display, 160, 162
Generate, 160
Search, 159

**M**
Manual_Rotamer command, Residue pulldown, 104
matrices
identity, Codon, Dayhoff, hydrophobicity, 255
matrix, 64, 255
m-block, 41
estimating statistical significance, 44
relative significance, 44
m-diagonal, 41
Measure pulldown
Bump, 168
minimization, 74
model building procedure, 16
model protein, 11
determining related proteins, 113
modes, 94
Modify pulldown
Geometry, 168
Hydrogen, 168
molecular dynamics, 75
free energy term, 82
molecular mechanics, 72
Molecule pulldown, 81
Molecule pulldown
Color, 191, 198
Display, 194
Get, 191
Molecule/Surface command see *Insight II* User Guide, 82
motif search scan, 118
motifs, 118
Motif_Search command, Databases pulldown, 106, 118
Multiple, 92, 93, 128, 135, 136, 140, 142, 147, 149, 153, 154, 156, 157, 233, 234, 236, 237, 238, 240, 242, 244, 245
multiple alignment, 25, 38, 47
automatic, 38
pairwise global sub-alignments, 39
superposition, 62
Multiple command, Alignment pulldown, 39
Multiple_Sequence command, Alignment pulldown, 40, 44, 59, 107

**N**
Needleman and Wunsch algorithm, 25, 33, 38
NOE restraints, 74

**O**
Object pulldown
Blank, 201
Output command, Databases pulldown, 105
overlap, steric, 69

**P**
p value (significance), 45
pairwise score threshold, 43, 46
Pairwise_Sequence command, Alignment pulldown, 38, 107
pdb_find_distance utility, 66
See also *Insight II* Reference Guide Utilities & Appendices
Percent_Identity command, Alignment pulldown, 109
potential energy equation, 72
probe, 21
   solvent accessible surface, 81
profiles
   hydrophobicity, 79
   secondary structure, 75
ProStat pulldown, 83
   commands in, 111
protein motifs, 118
protein name label, 87
pulldowns
   Alignment, 107
   Background_Job, 106
   Databases, 105
   Refine, 109
Put command, Sequences pulldown, 101

Q
quaternion algorithm, 59, 60

R
reference protein, 11
   coloring, 191
   comparing, 198, 200
   finding, 269
   loading, 190
Refine pulldown
   Explore, 75
   Explore, 74, 168
   commands in, 109
   EndRepair, 169
   Explore, 172
   Relax, 171
   SpliceRepair, 170
refinement, using molecular mechanics, 167
Relax command, Refine pulldown, 74, 110, 171
Rename command, Residue pulldown, 104
Repeat command, Residue pulldown, 104
Replace command, Residue pulldown, 104
residue counters, 87
residues
   similarity, 60
Residue_Dihedral command
ProStat pulldown, 111
restarting a session, 13
Restore Folder command, Folder pulldown, 13
   See also Insight II User Guide
Restore Folder command, Folder pulldown, 211
retrieving a session, 13
RMS
   in Display Loops, 66
   RMS calculations, 19, 200
      distance matrix, C_\alpha, 122
      iterative, 128, 134
   RMS threshold
      effect of increasing, 22
   Rotamer, 163, 164, 187
   rotamer, 70, 71, 165
   rotamer libraries
      side chain conformational searches, 70
Run_FASTA command, Databases pulldown, 105

S
Save Folder command, Folder pulldown, 13
   See also Insight II User Guide, 13
saving a session, 13
scales
   hydrophobicity, 80
   scoring local regions, 17
   scoring matrices, 26, 57, 255
      choosing one, 136
      codon substitution matrix, 58
      hydrophobicity matrix, 58
      identity matrix, 57
      mutation matrix, 58
      user matrix, 58
   scoring matrix, 17
SCR see structurally conserved regions, 19
Scroll Boxes command, Boxes pulldown, 204
Scroll Gaps command, Gaps pulldown, 93
Scroll Sequences command, Sequences pulldown, 194
scrolling buttons, 94
scrolling buttons, modes
   Scroll Boxes mode, 204
   Scroll Gaps mode, 194
Search command, Loops pulldown, 66, 103
Search Loops command, Loops pulldown, 159
secondary structure prediction, 75
secondary structure, 24
secondary structure prediction
  Chou-Fasman method, 76
  GOR II method, 78
SecondaryClassify command, ProStat pulldown, 112
SecondaryPred command, By_Residue pulldown, 109
segment pair overlap algorithm, 40, 46
Sequence Alignment, 93, 137, 138, 139, 140
sequence alignment, 33, 91, 135, 157, 187, 216
  automatic, 25, 33
  automatic (with SCRs), 33
  automatic (with SCRs), 139, 216
  automatic (without SCRs), 137, 138
  manual, 140
Sequence Alignment command, Alignment pulldown, 136, 256
sequence alignment See multiple alignment, 38
sequence boxes
  active, 202, 203
  coloring, 101
  comparing reference protein structures, 122
  creating with Initialize command, 91, 102
deleting, 102
freezing, 102
unfreezing, 102
sequence database
  searching, 114
sequence databases, searching, 16
sequence display, 87
  deleting, 101
  residue counters, 198
scrolling, 194, 203
sequence gaps, 93
deleting, 196
sequence protein, 11
  See also model protein
sequence rows, 87
Sequences pulldown, 87
  Get, 87
Sequences pulldown
AssignCoords, 158, 162
commands in, 100
Copy, 169
Extract, 122, 123, 192
Get, 122, 216
Scroll, 194
Setup_Bkgd_Job command, Background_Job pulldown, 106
side chain conformations, 163
significance of m-blocks (p value), 44
Solvation module see Insight II User Guide, 83
Solvent Accessible Surface (SAS), 81
SP score, 43
splice repair, 170
SpliceRepair command, Refine pulldown, 74, 110, 170
statistical significance of m-blocks, 44
steric overlap, 69
strict overlap criterion in segment pair overlap, 44, 61
strong similarity, 60
structural alignment, 200
  automatic, 20
  manual, 20
  manual, 202
structural misalignment
  correcting with Superimpose command, 131
structural superimposition, 59, 132
structurally conserved regions (SCRs), 12, 19
  automatic determination of, 20, 122
determine coordinate assignment, 62, 220
determining, 122, 204, 207
manual determination, 20
structure
  validation with ProStat, 83
Structure Alignment command, Alignment pulldown, 93, 123
Structure command, Alignment pulldown, 107
Struct_Check command, ProStat pulldown, 111
subset
  based on hydrophobicity, 81
subset definitions, 130, 131, 169, 170, 171, 213, 223, 226
coloring and labeling, 214
Subset pulldown
  List, 213
Sum of Pairs (SP), 43
Summarize Boxes command, Boxes pull-
down, 212
Summarize command, Boxes pulldown, 102
Superimpose command, Transform pulldown, 131, 215
  See also Insight II User Guide
superimposition of structures, 59, 132
superposition
  simultaneous, in multiple alignment, 62

T

tethering, 74
tick marks, 87
Torsion command, Transform pulldown, 168
  See also Insight II User Guide
torsion forcing, 74
Transform pulldown
  Superimpose, 131, 215
  Torsion
tutorials
  building SCRs and loops, 210
  finding SCRs, 189

U

UnFreeze Boxes command, Boxes pulldown, 212
UnFreeze command, Boxes pulldown, 102
unknown protein, 11
  See also model protein
user scoring matrix file, 256

V

variable region, 12
variable regions, 159
  building coordinates for, 162
  coordinate assignment, 63, 159, 230

W

weak similarity, 60