Protein Science (1994), *3*:198–210. Cambridge University Press. Printed in the USA.

Sculpting proteins interactively: Continual energy minimization embedded in a graphical modeling system

MARK C. SURLES,[1] JANE S. RICHARDSON,[2] DAVID C. RICHARDSON,[2] AND FREDERICK P. BROOKS, JR.[3]

[1] San Diego Supercomputer Center, San Diego, California 92186-9784
[2] Department of Biochemistry, Duke University, Durham, North Carolina 27710
[3] Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina 27599-3175

(RECEIVED October 13, 1993; ACCEPTED December 1, 1993)

Abstract

We describe a new paradigm for modeling proteins in interactive computer graphics systems—continual maintenance of a physically valid representation, combined with direct user control and visualization. This is achieved by a fast algorithm for energy minimization, capable of real-time performance on all atoms of a small protein, plus graphically specified user tugs. The modeling system, called *Sculpt*, rigidly constrains bond lengths, bond angles, and planar groups (similar to existing interactive modeling programs), while it applies elastic restraints to minimize the potential energy due to torsions, hydrogen bonds, and van der Waals and electrostatic[4] interactions (similar to existing batch minimization programs), and user-specified springs. The graphical interface can show bad and/or favorable contacts, and individual energy terms can be turned on or off to determine their effects and interactions.

Sculpt finds a local minimum of the total energy that satisfies all the constraints using an augmented Lagrange-multiplier method; calculation time increases only linearly with the number of atoms because the matrix of constraint gradients is sparse and banded. On a 100-MHz MIPS R4000 processor (Silicon Graphics Indigo), *Sculpt* achieves 11 updates per second on a 20-residue fragment and 2 updates per second on an 80-residue protein, using all atoms except non-H-bonding hydrogens, and without electrostatic interactions. Applications of *Sculpt* are described: to reverse the direction of bundle packing in a designed 4-helix bundle protein, to fold up a 2-stranded β-ribbon into an approximate β-barrel, and to design the sequence and conformation of a 30-residue peptide that mimics one partner of a protein subunit interaction.

Computer models that are both interactive and physically realistic (within the limitations of a given force field) have 2 significant advantages: (1) they make feasible the modeling of very large changes (such as needed for de novo design), and (2) they help the user understand how different energy terms interact to stabilize a given conformation. The *Sculpt* paradigm combines many of the best features of interactive graphical modeling, energy minimization, and actual physical models, and we propose it as an especially productive way to use current and future increases in computer speed.

Keywords: energy minimization; interactive computer graphics; molecular modeling; protein structure.

We describe a new paradigm for modeling proteins in interactive computer graphics systems—continual maintenance of a physically valid representation, combined with direct user control and visualization. A modeling system, called *Sculpt*, maintains valid bond lengths, bond angles, and van der Waals separations in a model as a user changes its structure by interactively moving atoms, peptides, and side chains. Like a physical model, *Sculpt* prevents certain movements due to bond rigidity, propagates changes throughout a model due to coupling in secondary structure or packing, and changes local conformation when degrees of freedom allow. Moreover, *Sculpt* keeps the model in a conformation that locally minimizes the potential energy due to torsion angles and hydrogen bond, van der Waals, and electrostatic[4] interactions for all atoms of the model. Together, *Sculpt* both maintains interactive performance and also

Reprint requests to: Mark C. Surles, San Diego Supercomputer Center, P.O. Box 85608, San Diego, California 92186-9784; e-mail: surles@sdsc.edu.

[4] A model of electrostatic interactions is currently under development.

incorporates physical properties that previously required batch computation.

Figure 1 shows a 20-residue α-helix modeled in *Sculpt*. The user moves the phenylalanine by picking it with a mouse (red arrow) and dragging it toward the cyan backbone. As the ring turns, it collides with the backbone carbonyl carbon (denoted by the red wireframe shell) and has minor contact with the nearby leucine (denoted by red dot surface). These conflicts in turn change the backbone conformation and side chain position. All this happens interactively (11 updates per second), which gives the user the impression of changing a real, physical model. Through a series of atom tugs over 5 min, a user wound a 20-residue extended chain into this helical conformation. The user sees bad contacts immediately, and likewise, *Sculpt* moves atoms to minimize the high potential energy caused by such contacts.

Motivation

Two goals motivate this work: (1) to enhance understanding of protein structure and function by placing more realistic computer models at the biochemist's fingertips, and (2) to reduce the manual and computational repairs needed after modeling sessions with interactive graphics. Such repairs are required in homology modeling when a residue is inserted or changed, in drug design when a protein is created with particular shape and properties, and routinely in de novo design to reconcile local conformation with overall design criteria.

Consider the simple change of flipping a peptide in the middle of the backbone with interactive modeling systems such as *Sybyl* (Tripos Associates), *Quanta* (MSI), and *InsightII* (Biosym Technologies). A change in φ flips the peptide but also moves the rest of the chain. A change in ψ and dihedral angles further along brings the chain close to its original position. When everything seems close enough to the desired new conformation, one runs a batch energy minimizer that shifts atoms into a minimum energy conformation. Other atoms may move in unintended ways if the user did not exactly reposition the chain. The direct grab-and-flip of the peptide in *Sculpt* is more intuitive, quick, and exact. *Sculpt*'s savings in labor and in potential errors are even greater in more complicated tasks such as moving a helix, changing the twist of a sheet, or repacking side chains.

Modeling paradigm

To maintain useful interactive performance, we find that 1 Hz is a minimum update rate. This limits problem size, properties modeled, and realism of the properties. On a 100-MHz MIPS R4000 processor (Silicon Graphics Indigo), *Sculpt* maintains 11 Hz on the 20-residue model in Figure 1, 2 Hz on an 80-residue model, and 1 Hz on a 180-residue model. The performance decreases *linearly* with increased model size.

We make an approximation that significantly improves performance without appreciably decreasing accuracy. Properties whose deformation requires very large potential energy changes relative to others (i.e., stiff properties) are replaced with rigid constraints. Covalent bond lengths and angles and planar dihedral angles are constrained to ideal values; hydrogen bonds, multivalue dihedral angles, and van der Waals and electrostatic interactions are modeled with potential energy functions. Min-

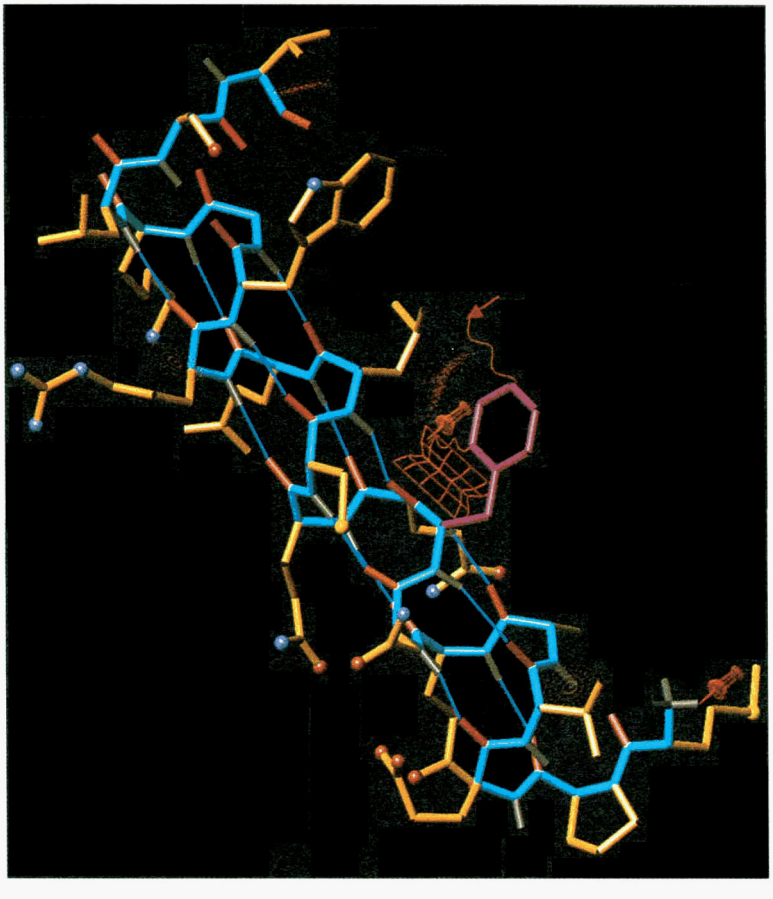

Fig. 1. Photo from *Sculpt* of a user tugging a Phe with the mouse, denoted by the orange spring between the atom and red cursor. A previous spring inserted by the user pulls it toward the orange thumbtack; the red thumbtack holds an atom in place. As the ring turns at 11 updates per second, it collides (red wireframe shell) with the carbonyl carbon and has minor contact (red dot surface) with the nearby leucine. This changes the backbone and side chain conformation. The 20-residue model of this α-helix has main chain (cyan), C-O (red), N-H (brown), and side chain (tan) bonds represented with tubes, and hydrogen bonds (blue) represented with vectors.

imizing functions with similar potential energies, subject to constraints, requires much less computation, in this application, than minimizing all the energies without any constraints.

The mathematical model contains the sum of the potential energies and a list of nonlinear equality constraints (e.g., distance(N,H) = 1 Å). When a user pulls an atom, *Sculpt* adds the potential energy in a spring between the atom and cursor to the model's potential energy. *Sculpt* then finds a new conformation that locally minimizes the total potential energy while satisfying the constraints.

Factoring out the stiff properties allows much larger atom movement than pure energy-based minimization, but in turn, it requires solving a system of nonlinear equations on each iteration. Constrained minimization has previously been avoided in interactive applications because, in general, the computation in solving the system of equations increases with the cube of the number of atoms. We show in this paper that for protein models the computation can be arranged to increase only linearly with the number of atoms. The long backbone (relative to short side chains) in proteins yields a system of equations that is sparse and banded. Thus, the constraints allow larger atom movement by removing stiff properties and only introduce a banded system of equations that can be solved efficiently.

This basic concept is not new, only its use in this type of interactive applications. Both molecular dynamics and crystallographic refinement use constraints to improve either speed or observation-to-parameter ratio. Molecular dynamics simulations often constrain the bond length between hydrogens and heavy atoms in an algorithm called SHAKE. Some crystallographic refinement systems employ constraints to idealize bond geometry while restraining (minimizing energy) weaker properties (Hendrickson & Konnert, 1980). Our techniques build on this concept of constraining stiff properties to decrease computation; we employ different mathematical solutions because we constrain many more properties.

Sculpt molecular model

We constrain a stiff property to its ideal value and only let a weak or multivalue property vary. This decreases model realism slightly, but our intended applications do not require accurately modeling minute changes in these properties. Figure 2 lists the constrained and restrained (energetic) properties modeled in *Sculpt*. A *constraint* requires that the value of a function equal a fixed value, and an *energy* (or *re*straint) imposes a penalty as a function value varies from its ideal value. Modeling a bond distance d with ideal value $\bar{d}$ as a constraint requires that $d = \bar{d}$ or equivalently, $d - \bar{d} = 0$. A spring restraint obeying Hooke's Law models the variance of the distance from the ideal value with $Energy = k(d - \bar{d})^2$, where k describes the spring stiffness.

We require that the constraints remain defined throughout a modeling session; *Sculpt* does not model the breaking and forming of covalent bonds. This restriction does not apply to hydrogen bonds and nonbonded interactions, because they use an energy model.

Sculpt's potential energy model differs in a few aspects, mentioned below, from models such as CHARMM (Brooks et al., 1983), Amber (Weiner et al., 1984), and Cedar (Hermans). These differences allowed quicker system development but are not essential to the method. *Sculpt* reads the ideal values and spring constants from a user-defined dictionary. The dictionary has parameters for models that have implicit hydrogens, hydrogen-donors only, or all atoms.

Protein property	Mathematical model
Bond length Bond angle Single dihedral angle	Constrained to ideal
Multi-value dihedral angle	Spring to nearest ideal angle
Hydrogen bond	Spring to ideal length and angle
Van der Waals interaction	6-Å, 4-8 Lennard-Jones potential
User tug	Zero-length spring
Electrostatic charge	10-Å, Coulomb potential (under development)
Solvent interaction	Not treated

Fig. 2. The mathematical model for each protein property treated in *Sculpt*. Covalent bond lengths, angles, and single-value dihedral angles are constrained to their ideal value; other properties use a potential energy model.

Dihedral angles with multiple ideal values (such as the 3 staggered conformers of a side chain χ angle)

We use a spring centered at the ideal dihedral angle closest to the current angle. This still allows transitions among the ideal angles, although it yields a higher energy barrier between two ideal angles than CHARMM's cosine model. Backbone φ and ψ angles are not treated this way, since they interact and are not well approximated by simple functions; however, they are handled by van der Waals interactions of the surrounding atoms.

Hydrogen bonds

A hydrogen bond is modeled with 2 springs—one for the bond length and the other for the angle between the donor bond and the hydrogen bond. Constraints are not used because hydrogen bonds are much weaker than covalent bonds, and they can change during a session. Currently all hydrogen bonds are specified at program initialization in order to preserve desired secondary structures. We will investigate methods for dynamic modeling of hydrogen bonds, either using neighbor lists or dipole interactions.

4-8 van der Waals approximation

We model the van der Waals interaction energy with a modified Lennard-Jones function; that function is evaluated on atoms within a 6-Å neighborhood, as is common for macromolecular calculations. We change the 6 and 12 exponents in the Lennard-Jones model to 4 and 8. Figure 3 illustrates the differences between our model and the 6-12 Lennard-Jones model for a given energy minimum, E_m, and separation at the minimum energy, R_m. This widens the energy well and yields a slower ascent, but also allows slightly closer atoms. The slower ascent allows faster solution of the equations. Our model also clamps the maximum rate of change of the energy after which the rate of energy increase is constant. The positive energy still repels atoms but allows much faster minimization since extremely large energies never occur. Since the computer model remains physically valid, we encounter extremely close atoms only if the user turns off the van der Waals model and at program initialization.

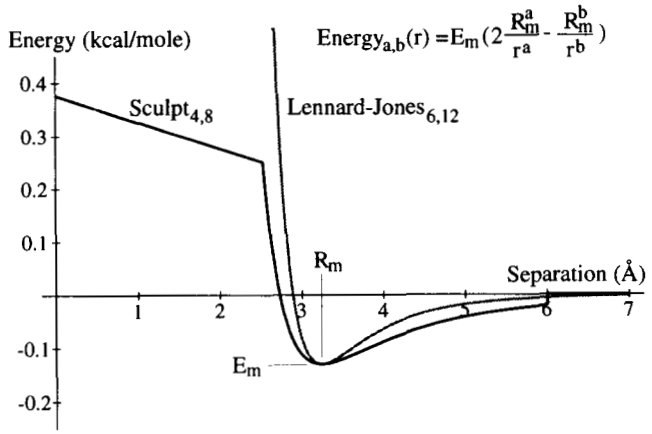

Fig. 3. Comparison of van der Waals potential using Lennard–Jones $(a,b = 6,12)$ and *Sculpt* $(a,b = 4,8)$ model shown for a given separation, R_m, with minimum energy, E_m. *Sculpt*'s exponents yield a wider and less steep energy well. *Sculpt* clamps the maximum rate of energy increase and only evaluates the potential on atoms within a 6-Å radius.

Electrostatic interactions

We are currently implementing a model of electrostatic interactions within a 10-Å radius. The performance results in this paper do not model electrostatic interactions except where explicitly noted. In those cases we estimated the extra time by modeling the van der Waals interaction among all atoms within a 10-Å radius in addition to the normal van der Waals interaction in a 6-Å radius. This gives an upper bound on the time required to model electrostatic interactions since, for example, it includes atoms with no dipoles. Although the present system behaves very well, one sees some effect of omitting electrostatics, such as a stickiness between oxygen atoms.

Mathematical formulation

We use the following mathematical notation for the remainder of the paper: an n-element vector $\boldsymbol{x}$ (boldface denotes vectors) holds the variables, a 3-dimensional position for each atom; the real-valued function $e(\boldsymbol{x})$ denotes the sum of all the energy functions; and the m-element vector $\boldsymbol{c}(\boldsymbol{x})$ is the vector of the m constraint functions. Specifically, row i in $\boldsymbol{c}(\boldsymbol{x})$ contains a constraint function, $c_i(\boldsymbol{x})$, minus its ideal value, $\bar{c}_i$, as follows:

$$\boldsymbol{c}(\boldsymbol{x}) = \begin{bmatrix} c_1(\boldsymbol{x}) - \bar{c}_1 \\ c_2(\boldsymbol{x}) - \bar{c}_2 \\ \vdots \\ c_m(\boldsymbol{x}) - \bar{c}_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

On each update *Sculpt* finds a local minimum of the total energy, $e(\boldsymbol{x})$, that satisfies the set of constraint functions, $\boldsymbol{c}(\boldsymbol{x}) = 0$. We find a constrained minimum by evaluating the gradient of the energy and constraints and solving a system of equations. We detail the method, called the *augmented Lagrange-multiplier method*, in the Mathematical solution section. Before we explain the mathematics, let us consider the contents of the gradients. The energy gradient, denoted $\boldsymbol{E}$, is a vector with n elements; each element is the first partial derivative with respect to one of the variables. The gradient of the constraints, however, is an $n \times m$ matrix; each column represents the gradient of a single constraint. We factor this matrix when solving the linear equations. For general problems (i.e., nonprotein) this factorization can take too much time for interactive performance except on small models, because the computation increases with the cube of the number of variables. However, the next section shows that the matrix in the protein application has properties that allow efficient factorization of the matrix; in fact, the computation increases only linearly with the number of atoms.

Structure of Jacobian matrix

The matrix containing the constraint gradient is called the Jacobian matrix. Excluding disulfide bridges, all covalent bonds in proteins, and therefore all the constraints, are along the backbone or within individual side chains. This property yields a very sparse, banded Jacobian matrix (i.e., the nonzeros lie within a small, fixed distance from the diagonal). The computation required to factor such a matrix is linearly proportional to the number of atoms. We first describe the matrix structure for proteins without disulfide bridges and then generalize this for ones with disulfide bridges.

Case 1: No disulfide bridge

Assume the variables (an x, y, z Cartesian coordinate) are numbered sequentially in the order of the Protein Data Bank format: for each residue, first come the heavy backbone atoms, then the heavy side chain atoms, and finally the hydrogen atoms. For example, atoms in poly-alanine are numbered N, C_α, C, O, C_β, H, N, etc. Define the index distance as the separation of variable indices. With this numbering scheme, all covalent bonds (excluding SS bridges) are defined on variables whose index distance is less than some fixed, maximum distance. This is also true for all angles and torsion angles defined by covalent bonds. Nonbonded interactions and hydrogen bonds can be defined on variables with a large index distance, but they are not covalently bonded and thus are not constrained.

Figure 4 illustrates the Jacobian matrix for a 2-residue segment. Each column represents the gradient of 1 distance constraint. Shading patterns in the matrix correspond to patterned atoms in the segment. Elements in a column are zero except at variables where the constraint is defined. The figure omits angle and dihedral angle constraints, as well as the hydrogens, for simplicity.

The maximum index distance is a constant, independent of the number of residues. The largest index distance between any 2 atoms occurs between the carbonyl carbon of tryptophan and the backbone nitrogen of the following residue because tryptophan has the most atoms. The maximum index distance in any constraint occurs in the dihedral angle that models this peptide (O-C-N-C_α) because the dihedral angle references the most variables and tryptophan contains the mosts atoms.

Case 2: Disulfide bridges

Disulfide bridges do not change the sparsity pattern of the Jacobian very much. A bridge introduces a few columns that denote the gradient of constraints defining the bond length and angles of the bridge. These columns can have arbitrarily large index distance because the constraints are defined on atoms in

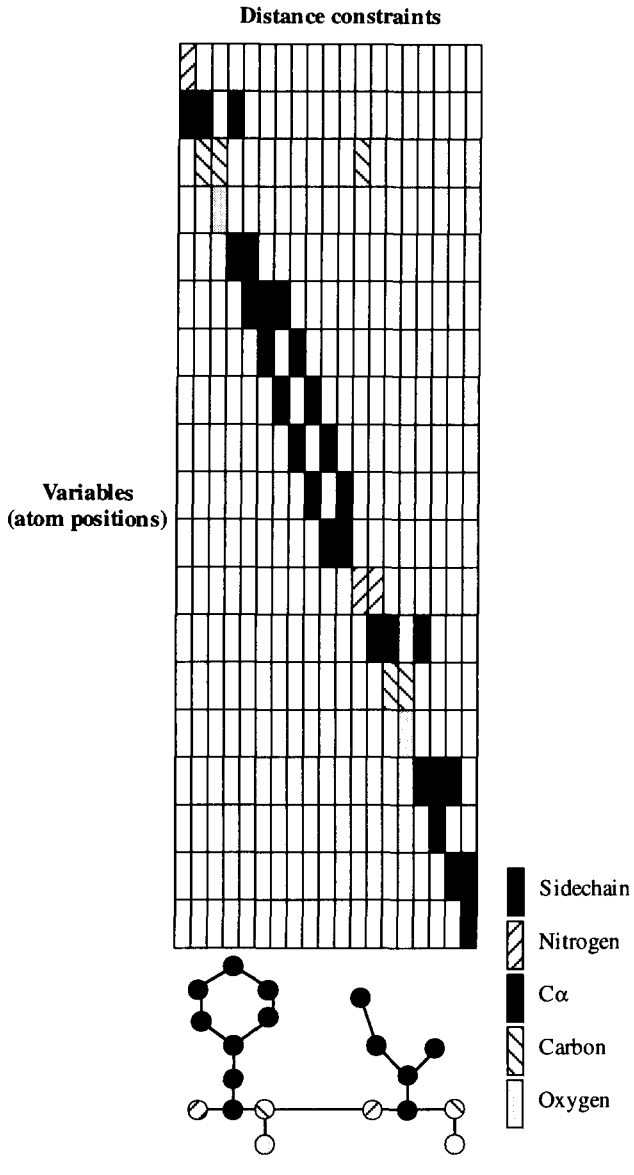

Fig. 4. Illustration of Jacobian matrix (matrix of constraint gradients) for the 2-residue segment at the bottom. The rows denote atom positions (a 3-dimensional Cartesian coordinate per row), and the columns denote the gradient of the distance constraints that model covalent bonds. Patterns in rows refer to atoms in the segment; blank cells are zero. Angles and dihedral angles, as well as the hydrogens, are omitted for simplicity.

residues arbitrarily down the sequence. The mathematics discussed in the next section handle this case efficiently. The computation required to factor the Jacobian does not increase as long as there are only a few such columns relative to the total number of columns. Fortunately, proteins only have a limited number of bridges.

Figure 5 shows the sparsity pattern of the Jacobian for *Felix* (Hecht et al., 1990), an 80-residue, 692-atom protein with 1 disulfide; only the backbone amide hydrogens were modeled. Black dots indicate nonzeros. Notice that the disulfide adds a few nonzeros far below the diagonal. This matrix has 1,772 columns (constraints) and 2,076 rows (variables).

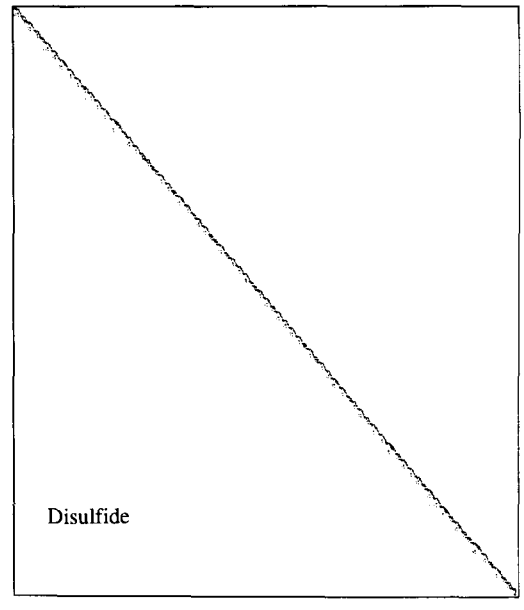

Fig. 5. Sparsity pattern of the Jacobian for *Felix*, an 80-residue, 692-atom protein with 1 disulfide. Black dots indicate nonzeros. Notice that the disulfide adds a few nonzeros far below the diagonal. This matrix has 1,772 columns (constraints) and 2,076 rows (variables).

Mathematical solution

Sculpt finds a local minimum of the energy that satisfies the constraints using an augmented Lagrange-multiplier method. This section details the algorithm, bounds its computational requirements, and compares it to other algorithms.

The first-order necessary conditions for a local constrained minimum are illustrated in Figure 6 for a 2-dimensional energy function with only 1 constraint. The solid line shows values that satisfy the constraint; dashed lines show isovalue contours of the energy. The solution must lie on the solid line, thus satisfying the constraint. At the nonoptimal point, $\boldsymbol{x}$, a step in the direction $\boldsymbol{d}$ reduces the energy and maintains the constraint. At the solution, $\boldsymbol{x}^*$, the energy gradient, $\boldsymbol{E}$, and the constraint gradient, $\boldsymbol{J}$, also align; i.e., the energy gradient is a scalar multiple of the constraint gradient, where λ is the scalar multiple.

Fletcher (1987) showed that the first-order necessary conditions for a local constrained minimum require (1) that the constraints be satisfied, and (2) that the energy gradient be a linear combination of the constraint gradients. For a problem with n variables and m constraints, these conditions require (1) $\boldsymbol{c} = \boldsymbol{0}$ and (2) $[\boldsymbol{J}]\boldsymbol{\lambda} = \boldsymbol{E}$, where $\boldsymbol{E}$ is the gradient of the energy, $\boldsymbol{J}$ is the $n \times m$ Jacobian matrix of constraint gradients, and $\boldsymbol{\lambda}$ is a vector of m scalar multiples called Lagrange multipliers.

No direct, closed-form solution exists for finding a constrained local minimum because the necessary conditions give $n + m$ nonlinear equations with $n + m$ unknowns. We use a 2-step, iterative algorithm: first, estimate the Lagrange multipliers at the solution; second, minimize an unconstrained func-

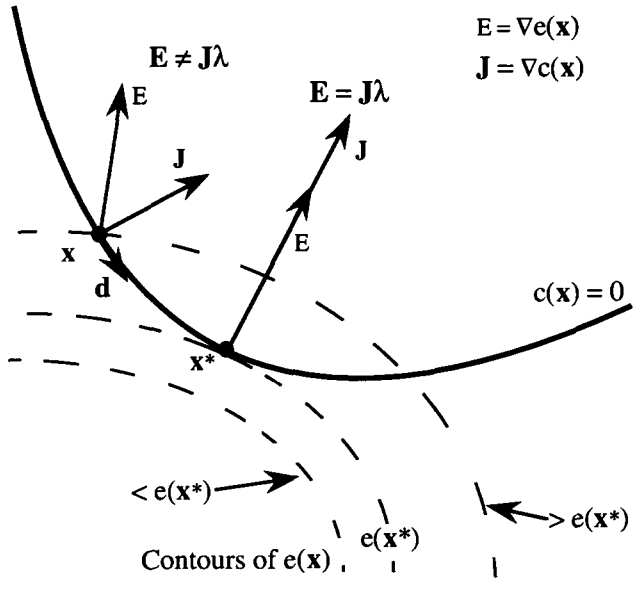

Fig. 6. Two first-order necessary conditions for a local constrained minimum of a 2-dimensional energy function (isovalue contours indicated with dashed lines) and only 1 constraint (solid curve). First, the solution must satisfy the constraint $c(\boldsymbol{x}) = 0$. Away from the solution, at point $\boldsymbol{x}$, a step in the direction $\boldsymbol{d}$ reduces the energy and maintains the constraint. At the solution, $\boldsymbol{x}^*$, the energy gradient, $\boldsymbol{E}$, and the constraint gradient, $\boldsymbol{J}$, align. This second condition requires that the energy gradient is a scalar multiple of the constraint gradient, where λ is the scalar multiple.

tion that combines the energy and constraint gradients with the estimated multipliers.

Lagrange-multiplier estimate

We estimate the Lagrange multipliers with the first-order multiplier method (Gill et al., 1981, p. 248). Given the gradients at the current position, this finds a least-squares approximation to the necessary conditions by solving for λ in $[\boldsymbol{J}]\boldsymbol{\lambda} = \boldsymbol{E}$. The system is overconstrained as there are more rows (model variables) than columns (constraints). Note: λ is overconstrained, not the model variables, $\boldsymbol{x}$.

We multiply each side by the matrix transpose (thus producing a square matrix) before solving for λ: $[\boldsymbol{J}^T\boldsymbol{J}]\boldsymbol{\lambda} = \boldsymbol{J}^T\boldsymbol{E}$. This approach has known drawbacks such as squaring the condition number (i.e., squaring its sensitivity to roundoff error), requiring a matrix–matrix multiplication, and factoring a matrix with more nonzeros (Luenberger, 1973). However, we find it works well for this application due to the sparsity pattern of the matrix. The product of a band matrix times its transpose yields a matrix with wider bandwidth and a few more nonzeros. The resulting matrix is symmetric and positive definite, 2 properties exploited by most sparse linear algebra packages (Duff et al., 1986).

Unconstrained minimization

We use the estimate of the multipliers in an unconstrained minimization of a function that combines the energy and constraint gradients and penalizes constraint violations. The function, called the augmented Lagrangian, is $L(\mathbf{x},\boldsymbol{\lambda},p) = e - \boldsymbol{\lambda}^T\mathbf{c} + p\mathbf{c}^T\mathbf{c}$. The third term penalizes the function when constraints are violated ($\mathbf{c} \neq \mathbf{0}$), which can occur from a poor initial configuration or from using a first-order approximation of the nonlinear constraints. We set the penalty, p, to the error in the least-squares approximation (i.e., $p \leftarrow \|\boldsymbol{E} - \boldsymbol{J}\boldsymbol{\lambda}\|_2$). We find the minimum with respect to $\boldsymbol{x}$ of the augmented Lagrangian using the steepest descent method. This method moves in the direction of the negative gradient of the function: $-\boldsymbol{E} + \boldsymbol{J}\boldsymbol{\lambda} - 2p\boldsymbol{J}\mathbf{c}$.

Figure 7 summarizes the steps in our iterative constrained minimization algorithm. This method is presented in Gill et al. (1981, p. 227), and its relative merits are described in Surles (1992a). The multipliers and variables converge to a constrained local minimum (Hestenes, 1975). We observe that in practice this algorithm finds the solution in 1 or 2 iterations, since the algorithm begins near a local solution. At the end of each iteration, *Sculpt* displays the new atom positions.

Computational requirements

The Lagrange-multiplier estimate requires the most computation, since it multiplies 2 matrices and factors the result. The computation required to estimate the multipliers in our case is proportional to the number of atoms, due to the band structure of the Jacobian; in general, the computation would increase with the cube of the problem size.

A multiplication of a banded matrix times its transpose yields another banded matrix. The nonzeros from a disulfide bridge that lie outside the band of the Jacobian yield nonzeros outside the band in the product. Figure 8 shows the sparsity pattern arising from multiplying the matrix in Figure 5 by its transpose.

If the product has a bandwidth b, as illustrated in Figure 9, there are at most $2bm$ nonzero entries.[5] Calculating each of these elements requires a row-column multiplication (e.g., element $(i,j) = \text{row}_i(\mathbf{J}^T) * \text{column}_j(\mathbf{J})$). The computation required is thus bounded, since the number of nonzeros in each row and column is bounded.

Factoring the matrix using Gaussian elimination requires approximately b^2m operations. For each of the m diagonals, the b nonzeros below it must be eliminated. Eliminating each of these requires multiplying the b elements in each row. When disulfide bridges are present, the nonzeros outside the bandwidth

[5] Protein models have many fewer nonzeros in practice, since many elements within the band are zero.

1.	Given x, compute the energy, constraints, and their derivatives	Evaluate e, c, E, J
2.	Estimate multipliers	Solve for λ: $[J^TJ]\lambda = J^TE$
3.	Set penalty to error in least-squares approximation	$p \leftarrow \|E - J\lambda\|_2$
4.	Minimize the augmented Lagrangian using the steepest descent	$x \leftarrow -E + J\lambda - 2pJc$

Fig. 7. Iterative algorithm used to find a constrained minimum. When the user tugs or releases an atom, energy is added or removed. *Sculpt* then executes this algorithm, which changes the atom positions ($\boldsymbol{x}$), and displays the results.

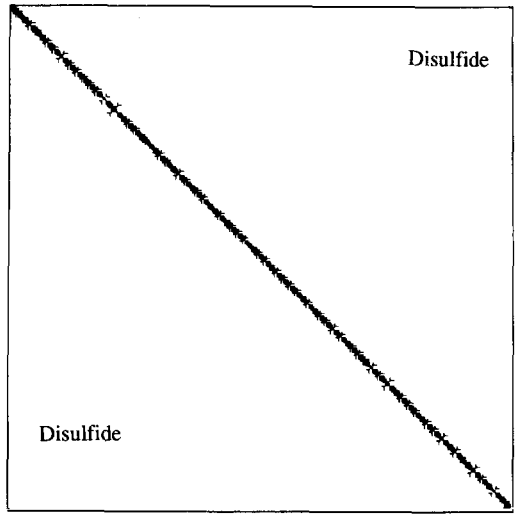

Fig. 8. Sparsity pattern of the matrix product (J^TJ), where the Jacobian, J, is shown in Figure 5. Black dots indicate nonzeros. The matrix is square (1,772 × 1,772) and symmetric. The disulfide introduces a few nonzeros far above and below the diagonal.

only fill elements in their same column as the rows are reduced. Thus, a very small, predictable fill-in results. More detailed analysis of the required computation appears in Surles (1992b).

We solve the system of equations with the banded matrix solver, *F01MCF*, from the Numerical Algorithms Group (NAG, 1981). The method is specialized for symmetric matrices with variable, or "skyline," bandwidth. Row pivoting is not necessary because the matrix is positive definite.

Comparison with other methods

Other constrained minimization algorithms avoid the matrix-matrix multiplication required in our least-squares approximation. We tried several iterative least-squares solvers, but none performed as well as the algorithm described here. We observed that the solvers typically required around m iterations, each requiring a matrix-vector multiplication. Also, the rectangular Jacobian matrix is not symmetric or positive definite—properties on which iterative solvers perform best.

We also examined a reduced gradient method (Rosen, 1961) that projects the energy gradient onto the constraint surface. This method factors an $m \times m$ matrix formed from m linearly independent rows of the Jacobian. It avoids the matrix-matrix multiplication and factors a matrix with fewer nonzeros. The chief disadvantage is the computation required for picking the independent rows. Once the rows are picked, the algorithm can use them for many iterations (typically 20–100) before a dependency arises and a new set is needed. While using a set of rows, the algorithm runs approximately 20% faster than ours. However, picking a new set of rows requires 20 times more computation than one iteration. The user experiences a long delay whenever the system picks a new set of rows.

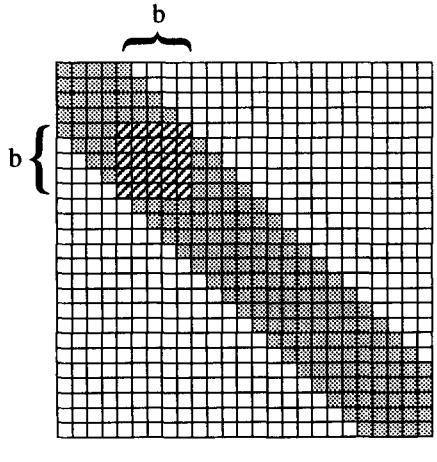

Fig. 9. A matrix with bandwidth b. The bandwidth is the maximum column (row) separation of nonzero entries in any row (column). White squares represent zeros.

Applications

To demonstrate how *Sculpt* operates, consider the task of flipping 1 peptide in a protein by 180° without significantly changing the rest of the model. This is simple to do with brass or plastic models, but is cumbersome in most computer modeling systems. In *Sculpt* the user clicks on the carbonyl oxygen and drags the mouse (and the spring) around the peptide bond. *Sculpt* responds by bringing the carbonyl around, changing the local conformation and side chain positions to satisfy constraints and minimize energies. Near 90° there are unavoidable overlaps between nearby atoms (e.g., CO to C_β), which can be displayed as red shells and are also evident in a slower atom movement (the atom moves less per cycle because other forces oppose it). Toward the end, the structure rapidly settles into the new favorable conformation. Kinemage 1 illustrates this process on an 8-residue segment with a tight turn, and Figure 10 presents the initial and final steps with an 80-residue protein. The orange coils show springs attached by the user, and the red shell indicates a bad contact. The time for this operation depends on the system performance, which depends on the number of atoms. For a small peptide, as in Kinemage 1, it is limited mainly by the user tug rate; when embedded in an 80-residue protein as in Figure 10, a complete flip takes 15 s.

Redesigning a dimer interface

As a real application, we used *Sculpt* to design the sequence and conformation of a peptide to mimic one half of the dimer interaction of the HIV protease (in collaboration with Lilia Babé at the University of California-San Francisco, who will test its inhibition of viral replication in vivo, in comparison with simpler β-strand peptides [Babé et al., 1992]). We kept 1 subunit of the protease fixed, with the atoms near the dimer contact (residues 1–10, 23–29, 86–99) included in the van der Waals calculations and the rest ignored. Parts of the second subunit provided a starting model for the peptide (modified from Brookhaven Data Bank file 5HVP; Fitzgerald et al., 1990). We fixed both the conformation and sequence of the N-terminal and C-terminal β-strands (residues 1–5 and 95–99), which form an interdigitated β-sheet between the 2 subunits. We also fixed the main chain of residues 24–26, which form a short, antiparallel β interaction between subunits next to the active-site aspartates. Connections between these 3 pieces were modeled using modifications of the original residues 6–11, 27–29, and 87–94.

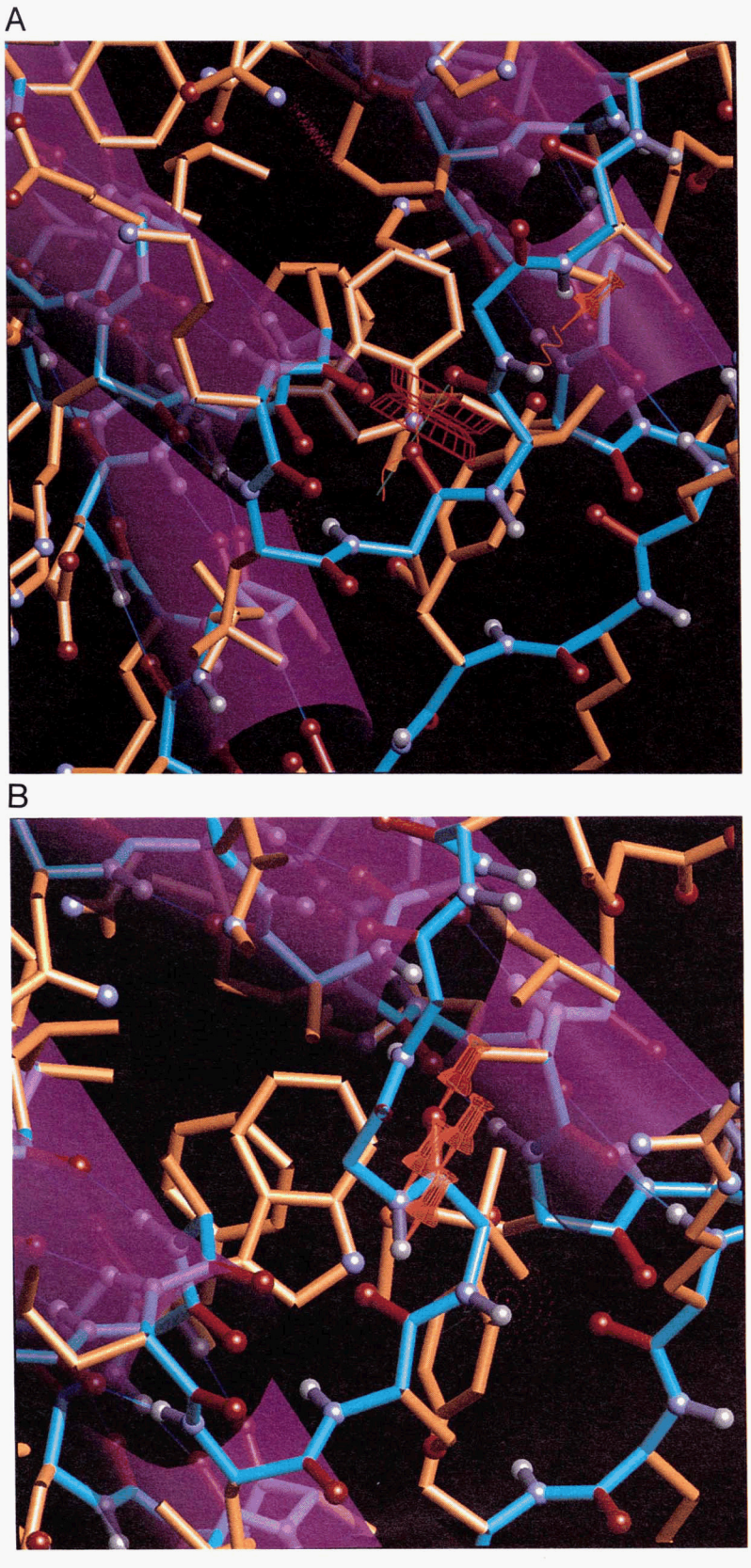

Fig. 10. The top photo shows a bad contact (red wireframe) between adjacent carbonyl groups in an intermediate model of Felix. Springs show the tugs applied by a user to flip the peptide and remove the bad contact. The bottom photo shows the resulting conformation, with thumbtacks highlighting atoms of the flipped peptide. Kinemage 1 animates the actual process of flipping a peptide with *Sculpt* on a simpler model.

We used *Sculpt* to move the 3 loose chain segments into favorable conformations that: (1) substitute for essentially all dimer contacts made in the native protease (except for the *flaps* at the bottom of Fig. 11), (2) join up properly, and (3) place hydrophobic side chains in contact positions and hydrophilic ones at exposed sites in the peptide–protease complex. We tried several conformational strategies over 2 h, and the *sculpting* process prompted a number of changes in sequence and 1 change in length from our initial guess. The final peptide is 30 residues long, 11 of which (underlined) differ from the native sequence:

PEITLWQRLSDLHTGSGSPELTQKGCTLNF.

Figure 11 shows the backbone of the HIV model with dimer A in yellow and dimer B in magenta. Dim vectors were not modeled; bright yellow vectors were fixed but used in the van der Waals interactions. The tubes show the movable parts; magenta denotes the original model and cyan shows the result. The two red tubes highlight the connections among the 3 segments. Kinemage 2 illustrates the contacts in the modeled peptide–protease complex.

Modeling β-barrel folding

To illustrate and study our hypothesis for the folding of Greek-key β-barrels, we first used *Sculpt* to unfold the 8-stranded, "jellyroll" Greek key of residues 19–98 of catabolite gene activator protein (Brookhaven Data Bank file 3GAP; Weber & Steitz, 1987) into a long, twisted, 2-stranded β-ribbon. There were 24 β-sheet hydrogen bonds in the starting structure; 9 more were formed and 2 broken during the unfolding; side chains were truncated at C_β. The smooth performance on unfolding encouraged us to try the more significant folding direction.

Next, we generated a 32-residue (poly-Ala), idealized strand of twisted, somewhat-curled β structure with alternating φ, ψ values of (−130°, 160°) and (−90°, 120°), docked 2 copies on a twisted β-ribbon from LDH (residues 265–293 of Brookhaven file 1LDM; Abad-Zapatero et al., 1987), closed a hairpin turn on one end with a Gly-Gly type I′ turn, formed the β-sheet hydrogen bonds, and uncurled the ribbon somewhat. We then folded that idealized ribbon, by tugging near its ends, into a 6-stranded, Greek-key β-barrel. Figure 12 shows 7 snapshots from this latter, 2-h *Sculpt* session with the backbone colored cyan in one direction and orange in the other. Interactions at each end of the final barrel were especially interesting, and in a later session (shown in Kinemage 3) we folded up a similar hairpin with Gly, Pro, and Tyr at some critical locations in the barrel-end connections. This second ribbon was made compact by a more realistic motion of pushing both ends together and letting a loop curl in the middle.

Reversing the directionality of a 4-helix bundle

To test the strengths and weaknesses of *Sculpt* for de novo model building, we made a major change in the tertiary structure of

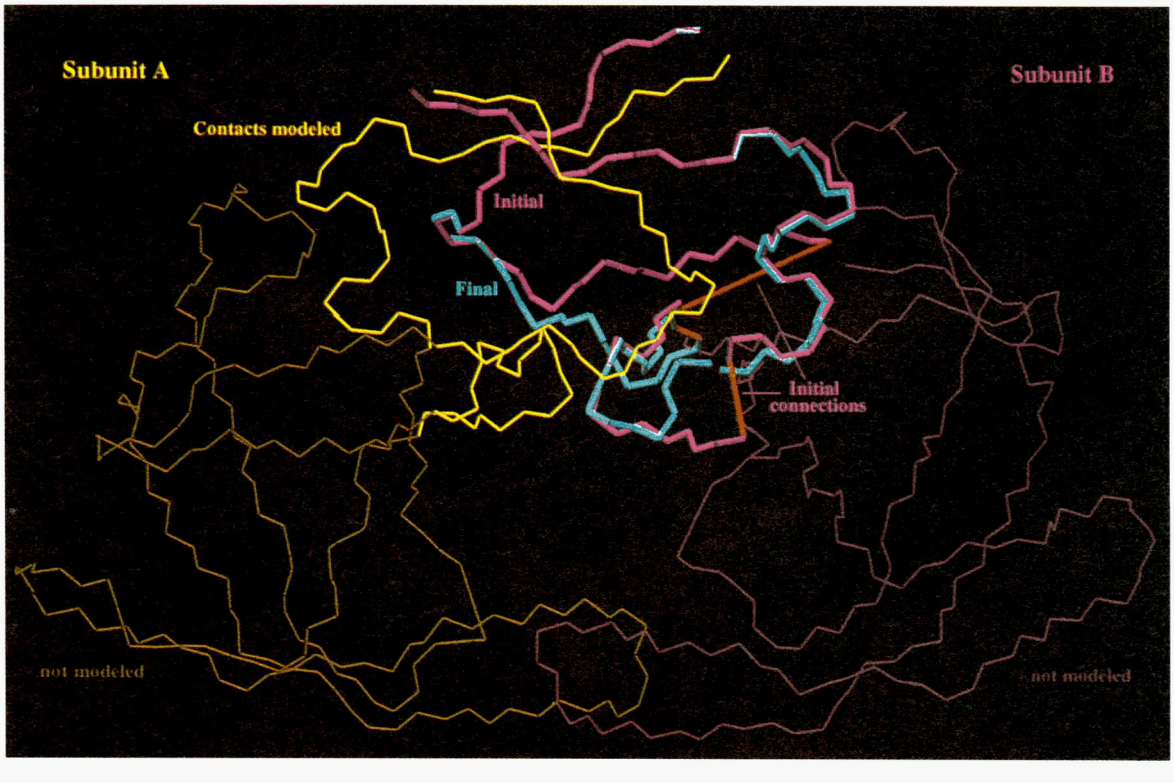

Fig. 11. Results of using *Sculpt* for designing the sequence and conformation of a peptide to mimic one half of the dimer interaction of the HIV protease (PDB file 5HVP). The main chain is yellow for dimer A and magenta for dimer B. Dim vectors were not modeled; bright yellow vectors were not movable, but those residues were used in the van der Waals interactions. Tubes show the original (magenta) and final (cyan) segments modeled in *Sculpt*; they overlap at the N- and C-terminal β-strands. Red tubes indicate the breaks between the original 3 segments, where the dim magenta parts of that subunit were omitted.

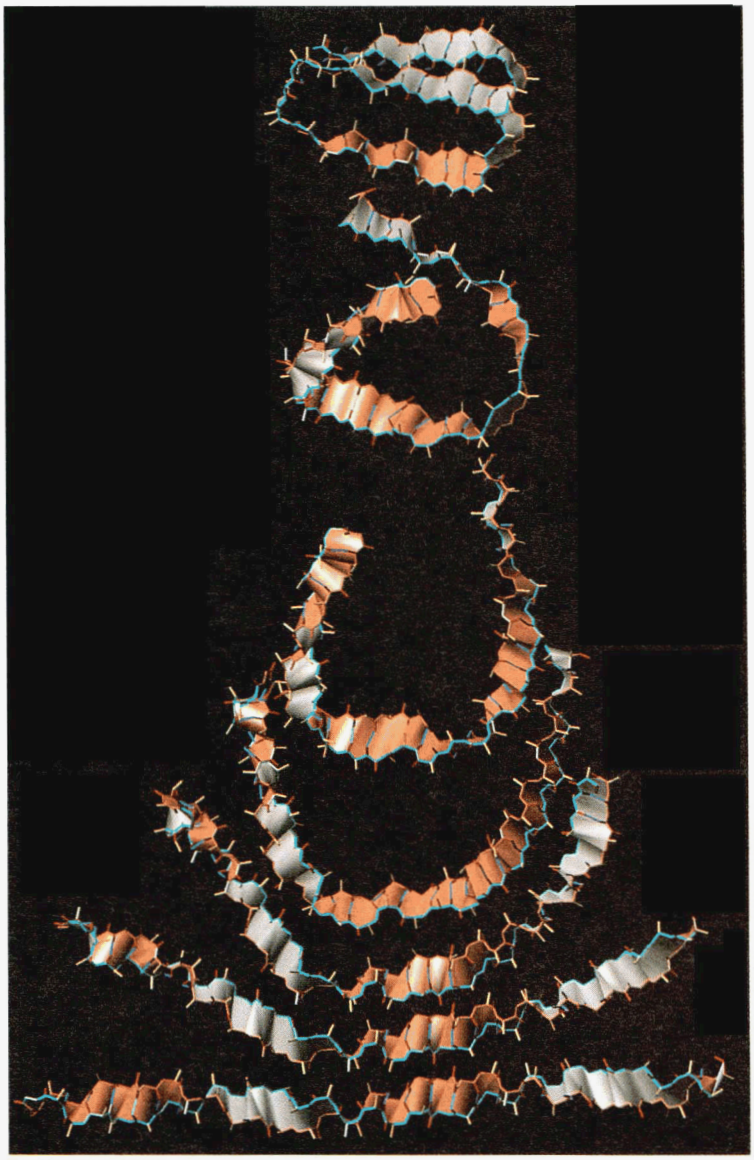

Fig. 12. Seven snapshots from a *Sculpt* session show the results of tugging a 2-stranded, antiparallel β-ribbon (bottom) into a 6-stranded, Greek-key β-barrel (top). The backbone is cyan in one direction and orange in the other. The sides of the β-sheet surface are colored differently to indicate twists. The starting model is a pair of 32-residue (poly-Ala), idealized, twisted β-strands connected by a Gly-Gly type I′ hairpin turn.

an 80-residue protein. We began with the designed structure of Felix, a 4-helix bundle protein (Brookhaven Data Bank file 1FLX; Hecht et al., 1990). The top of Figure 13 shows the backbone and hydrophobic side chains of the original structure, with helix A coming out of the page. This is the common "up & down" helix-bundle connectivity, folded in a clockwise order around the bundle. There is a designed disulfide bridge connecting helices A and D, whose sulfurs are shown as yellow spheres. In doing de novo design of proteins it is just as important to do negative design that avoids major alternative structures as it is to do positive design for the desired arrangement. Therefore, using *Sculpt* we created another 4-helix model that folds in a counterclockwise order; the cysteines now cannot form a disulfide bridge but instead are at the outer edge of helix contacts on opposite sides of the molecule. The result is illustrated in the bottom half of Figure 13. The original interhelical connections are colored yellow, showing that helices A and C were unwound by 1 residue and helices B and D were wound by 1 residue. In the original model the hydrophobic side chains in the A–D and B–C helix contacts are red, and the ones in the A–B and C–D contacts are purple; after the modeling session, the faces have turned so that the red side chains are in the A–B and C–D contacts, and the purple ones in A–D and B–C.

We preserved secondary structure by first rotating and translating 4 rigid segments: helix A plus the segment from A to B; helix B; helix C plus the segments from B to C and C to D; and helix D. This left the model disjoint (e.g., rotating the first segment 90° left the chain far below B). We spent approximately 3 h rejoining the segments by tugging the 3 peptides. During this part of the session, *Sculpt* modeled constraints and energies in the residues between the helices and kept the helices fixed. Next, we modeled all the atoms to adjust side chain contacts. We could not eliminate all bad contacts at the beginning and ending turns in the helices. We now use this new model as a basis both for negative design (i.e., to document the ways in which the Felix sequence suits the original model better than it suits this one) and also for designing minimal changes in the sequence that should make it prefer to fold into this alternative structure.

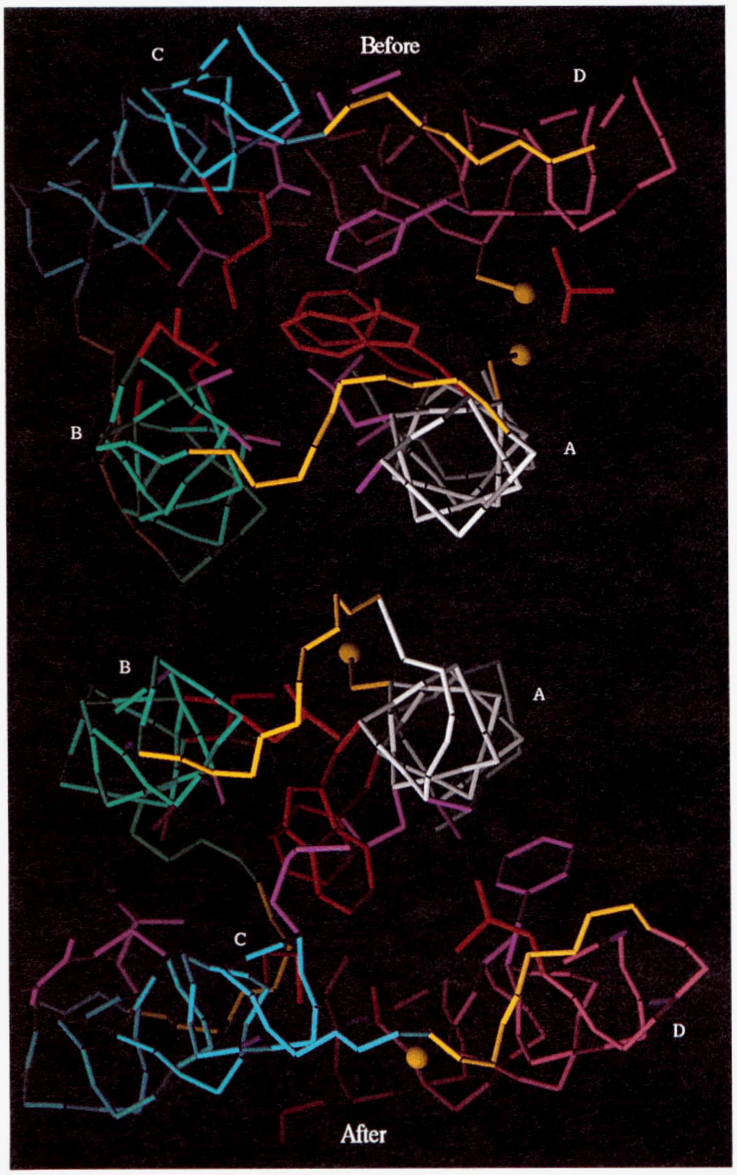

Fig. 13. The pair of models shows the results of inverting a 4-helix bundle using *Sculpt*. The top model shows the backbone and hydrophobic side chains of Felix (PDB file 1FLX), a designed 4-helix bundle protein with helix A coming out of the page and B, C, and D folding in a clockwise order. Each helix, the connecting segments, and the cysteines of the starting disulfide are colored the same in both figures. The bottom model shows the results of changing the folding to a counterclockwise order. Each helix and each helical side chain is turned approximately 90°; the yellow segments show helices A and C are unwound by 1 residue and helices B and D are wound by 1 residue. In the original model, hydrophobic side chains between helices A–D and between B–C are red, and those between A–B and C–D are purple; after the modeling session, the faces have turned so that the red side chains are between helices A–B and C–D, and the purple ones are between A–D and B–C.

Implementation and performance

Sculpt contains 2 programs: the displayer displays the graphics and user interface on a Silicon Graphics workstation, and the minimizer finds a constrained minimum. The minimizer can run either on a Silicon Graphics workstation (possibly a different one than the displayer) or on a Cray. The 2 programs communicate over Ethernet via Unix sockets; the displayer sends pick and tug information to the minimizer, which, after minimization, sends new coordinates to the displayer.

Table 1 shows the composition of 10 protein models used in a performance evaluation. The number in the model name is the number of residues. Model F80 is the Felix protein with 1 disulfide and hydrogen bonds; F20 and F40 are the first 20 and 40 residues of Felix. K356 is the cAMP-dependent protein kinase (Zheng et al., 1993) with bound ATP; others beginning with a "K" are pieces of K356 and also include ATP. Bonded energies represent hydrogen bonds and multivalue dihedral angles. The numbers of interactions are averages.

Table 2 and Figure 14 show the number of seconds per iteration of the constrained minimizer on the 10 models; communication time between the minimizer and displayer is negligible. The performance tests were run on 1, 4, and 8 processors of an 8-processor, 100-MHz MIPS R4400 workstation (Silicon Graphics Challenge) using 64-bit double precision. The algorithm has 2 parallel components detailed in Surles (1994): first, the constraint gradient and vector operations are divided over p processors, yielding approximately a p-times speedup; second, matrix factorization (step 2 in Fig. 7) runs on 1 processor while the energy gradient runs on $p - 1$ processors. When the matrix factorization takes longer than the energy gradient, as in the 8-processor case, the time for models with and without electrostatic interactions is the same. The minimizer can also run on 1 processor of a 167-MHz Cray Y-MP using 64-bit single precision. Unfortunately, the rate-limiting steps in this algorithm are not readily vectorizable, so the Cray performance is roughly 50% faster than the 1-processor SGI implementation.

The performance is listed with and without an estimate of

Table 1. *Composition of 10 models used in performance evaluation*[a]

Model	Atoms	Variables	Constraints	Bonded energies	Van der Waals interactions	Electrostatic interactions
F20	186	558	473	80	1,831	5,379
F40	341	1,023	870	196	3,347	12,293
F80	693	2,079	1,772	415	7,714	31,299
K120	1,054	3,162	2,734	418	11,197	44,542
K160	1,442	4,326	3,737	579	17,155	73,209
K200	1,810	5,430	4,683	738	21,675	94,944
K240	2,172	6,516	5,603	904	26,253	116,466
K280	2,541	7,623	6,556	1,058	31,856	141,512
K320	2,880	8,640	7,439	1,194	35,760	159,519
K356	3,191	9,573	8,244	1,324	40,613	181,893

[a] The number in the model name is the number of residues. Model F80 is the Felix protein with 1 disulfide and hydrogen bonds; F20 and F40 are the first 20 and 40 residues of Felix. Model K356 is the cAPK protein with bound ATP; others beginning with a "K" are pieces of K356 and also include ATP. Bonded energies represent hydrogen bonds and multivalue dihedral angles; electrostatics are estimated by running the van der Waals computation with a 10-Å cutoff. The numbers of interactions are averages.

electrostatic interactions. Neither includes the time to compute a new neighbor list, which is done only every 50 iterations. *Sculpt* computes the list of neighbors within a 12-Å radius of each atom for the van der Waals model (20 Å for electrostatic), using a space-subdivision algorithm (Bentley & Friedman, 1979). Then, at each cycle, it evaluates the van der Waals potential on atoms in the list that are presently within 6 Å of the reference atom. This maintains valid neighbor lists even though atoms move. The time to determine the list on 1 processor is also listed in Table 2. We believe the time to compute a new list can be reduced, and this can also run in parallel with the minimization.

Discussion

We believe that solving these modeling tasks with *Sculpt* is significantly easier and faster than with purely geometrical, interactive modeling systems or with batch molecular dynamics methods, and that *Sculpt* provides new benefits as well. Guiding an interactive simulation while immediately viewing the results lets a user remain continually engaged in the modeling process. This seems to provide greater situational awareness and to improve perception of subtle relationships within proteins. On a number of occasions we noticed unexpected reactions in the model that, upon closer examination, resulted from nonbonded interactions competing against other properties such as bond rotations. The graphical visualization of nonbonded interactions helps identify close contacts among atoms and evaluate improvements made by moving atoms.

Sculpt can improve both productivity and understanding over previous molecular modeling systems. Present update rates allow productive new research in biochemistry, and we believe a mature *Sculpt* system will not just assist in a succession of in-

Table 2. *Seconds per iteration of the constrained minimization on 1, 4, and 8 processors of an 8-processor, 100-MHz, MIPS R4400, Silicon Graphics Challenge*[a]

	Seconds per update						Seconds for list reset	
	van der Waals			van der Waals and electrostatics				
Model	1 CPU	4 CPUs	8 CPUs	1 CPU	4 CPUs	8 CPUs	6 Å	10 Å
F20	0.09	0.04	0.04	0.14	0.04	0.04	0.03	0.05
F40	0.17	0.07	0.06	0.30	0.09	0.06	0.06	0.11
F80	0.42	0.19	0.16	0.72	0.23	0.16	0.13	0.29
K120	0.57	0.24	0.20	1.03	0.32	0.20	0.18	0.43
K160	0.81	0.33	0.28	1.56	0.49	0.28	0.28	0.70
K200	1.04	0.41	0.36	2.10	0.62	0.36	0.33	0.92
K240	1.23	0.49	0.42	2.44	0.75	0.42	0.41	1.15
K280	1.54	0.59	0.50	2.95	0.89	0.49	0.48	1.37
K320	1.65	0.65	0.56	3.36	1.02	0.56	0.73	1.75
K356	1.86	0.72	0.62	3.74	1.15	0.62	0.84	2.12

[a] Time to reset the neighbor list (done every 50 iterations) is given on the right. All computations except matrix factorization run in parallel. Factorization dominates computation after 4 processors for the van der Waals models and after 8 processors for the electrostatic models.

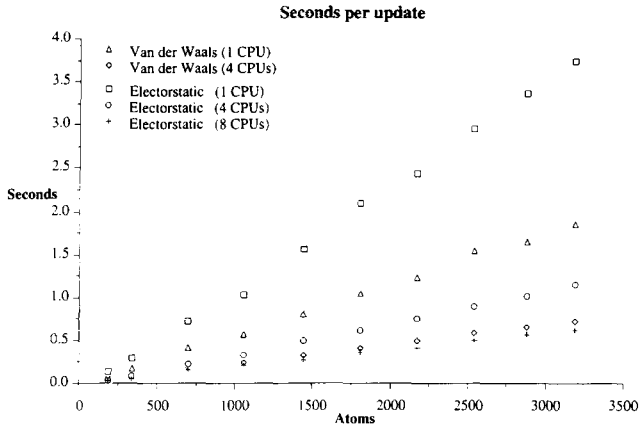

Fig. 14. Plot shows linear increase in computation for models with and without electrostatic interactions on 1, 4, and 8 100-MHz MIPS R4400 processors.

dividual modeling tasks, but also help researchers gain an intuitive understanding of how molecules behave.

Future directions

Actual implementation is in progress for the electrostatics calculation. Along with that, we are examining methods for increasing performance, so that larger proteins with more realistic energy models can run interactively. One method divides the computation over heterogeneous supercomputers; for instance, simultaneously compute the constraint gradient and matrix factorization on a Cray C90 and the nonbonded interactions on a 400-processor Intel Paragon. Another method involves reducing the number of variables and constraints by modeling secondary structures or other pieces with rigid but movable bodies. A rigid object with few variables could replace large segments of a model that a user does not want to change. For example, a user could twist a backbone into a helix and then freeze the helix by replacing its main chain atoms and bonds with a cylinder of rigid shape but movable position.

Another area for future work involves improving the method for sequence input and modification. Currently, several preprocessing steps transform a PDB file into *Sculpt* input, including the bond connectivity, ideal values, and energy constants. *Sculpt* does not allow residue insertions, deletions, or changes; one must save the coordinates, use another package to generate the new atom positions, and then restart. Internalizing those steps will greatly improve the usability of the system.

Sculpt is available to academic users who understand that it is a research system still under development. Several user-friendly features such as documentation and data exchange are incomplete or nonexistent. Please contact the first author (surles@sdsc.edu) for more information.

Acknowledgments

We thank Jan Hermans for informative discussions during the conception of *Sculpt*; Jim Begley, Dave Chen, and Rob Katz for implementing parts of the *Sculpt* interface; Silicon Graphics for running benchmarks on an 8-processor machine; Lynn Ten Eyck for support and advice; and the San Diego Supercomputer Center for access to the visualization and supercomputing facilities.

This work began in the Computer Science Department at the University of North Carolina at Chapel Hill with support from National Institutes of Health grant RR-02170. The research continues at the San Diego Supercomputer Center with support from National Science Foundation grant ASC-9211908. The Duke University authors are supported by NIH grant GM-15000.

References

Abad-Zapatero C, Griffith JP, Sussman JL, Rossmann MG. 1987. Refined crystal structure of dogfish M4 APO-lactate dehydrogenase. *J Mol Biol 198*:445–467.

Babé LM, Rose J, Craik CS. 1992. Synthetic interface peptides alter assembly of the HIV 1 and 2 proteases. *Protein Sci 1*:1244–1253.

Bentley JL, Friedman JH. 1979. Data structures for range searching. *Computing Surv 11*:397–409.

Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M. 1983. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J Computation Chem 4*:187–217.

Duff IS, Erisman AM, Reid JK. 1986. *Direct methods for sparse matrices*. Oxford, UK: Clarendon Press.

Fitzgerald PMD, McKeever BM, van Middlesworth JF, Springer JP, Heimbach JC, Leu T, Herber WK, Dixon RAF, Darke PL. 1990. Crystallographic analysis of a complex between human immunodeficiency virus type I protease and acetyl-pepstatin at 2.0-Å resolution. *J Biol Chem 265*:14209–14219.

Fletcher R. 1987. *Practical methods of optimization*. New York: John Wiley & Sons.

Gill P, Murray W, Wright M. 1981. *Practical optimization*. San Diego: Academic Press.

Hecht MH, Richardson JS, Richardson DC, Ogden RC. 1990. De novo design, expression, and characterization of Felix: A four-helix bundle protein of native-like sequence. *Science 249*:884–891.

Hendrickson WA, Konnert JH. 1980. Incorporation of stereochemical information into crystallographic refinement. In: *Computing in crystallography*. Bangalore, India: The Indian Academy of Sciences. pp 13.01–13.25.

Hestenes MR. 1975. *Optimization theory, the finite dimensional case*. New York: John Wiley and Sons.

Luenberger DG. 1973. *Introduction to linear and nonlinear programming*. Menlo Park, California: Addison-Wesley.

NAG. 1981. *NAG Fortran library manual*. Oxford, UK: The Numerical Algorithms Group Ltd.

Rosen JB. 1961. The gradient projection method for nonlinear programming, part II: Nonlinear constraints. *J SIAM 9*:514–532.

Surles MC. 1992a. An algorithm with linear complexity for interactive, physically-based modeling of large proteins. *Computer Graphics 26*:221–230.

Surles MC. 1992b. Techniques for interactive manipulation of graphical protein models [dissertation]. Chapel Hill: University of North Carolina.

Surles MC. 1994. Parallel constrained minimization for interactive protein modeling. *27th Hawaii International Conference System Science, vol. V*. New York: IEEE. pp 183–192.

Weber IT, Steitz TA. 1987. Structure of a complex of catabolite gene activator protein and cyclic AMP refined at 2.5 Å resolution. *J Mol Biol 198*:311–326.

Weiner SJ, Kollman PA, Case DA, Singh C, Ghio C, Alagona G, Profeta S, Weiner P. 1984. A new force field for molecular mechanical simulation of nucleic acids and proteins. *J Am Chem Soc 106*:765–784.

Zheng J, Knighton DR, Ten Eyck LF, Karlsson R, Xuong NH, Taylor SS, Sowadski JM. 1993. Crystal structure on the catalytic subunit of cAMP-dependent protein kinase complexed with MgATP and peptide inhibitor. *Biochemistry 32*:2154–2161.