Automated site-directed drug design: a method for the generation of general three-dimensional molecular graphs

Richard A. Lewis

Biomolecular Modelling Laboratory, Imperial Cancer Research Fund, London, UK and Department of Pharmaceutical Chemistry, University of California, San Francisco, USA

A new algorithm for creating diverse, irregular and physically reasonable three-dimensional linear atomic chains is described. The linear chains of atoms, or molecular graphs, are generated by solving a series of trigonometric equations within geometric constraints for a given set of atom types. The nature and number of the chains that are produced can be controlled by changing the palette of atom types, so that a chemist user could generate template suggestions that are synthetically relevant to a drug design project. Testing has shown that the method is sufficiently robust to be used in a general context. The molecular graphs could serve as useful structural templates for joining up regions in an active site where a ligand might interact strongly with the receptor. This paper is concerned with the description and proof of the methodology. The approach will form part of a larger structural tool kit for helping chemists to design novel ligands for a specified site.

Keywords: drug design, structure generation, molecular graphs, torsion angle fitting

INTRODUCTION

The goal of drug design is to understand the principles of molecular recognition that control the formation of drug–receptor complexes, and to use this knowledge in the design of novel ligands. Methods for site-directed drug design are predicated upon the existence of an atomic model of the receptor site. It is not necessary for the model to be accurate; it should be treated as a hypothesis to be verified by experiment. Atomic models may be derived from X-ray crystallography, nuclear magnetic resonance or protein homology studies. Ligands that are complementary to the site could be designed and synthesized; the results of binding affinity experiments should then be used to refine the model of the site. The *de novo* design of ligands is still a very difficult task. This paper describes a new tool for generating three-dimensional (3D) atomic substructures that will aid medicinal chemists in suggesting new and interesting target molecules. The method is intended to complement existing approaches and will form part of a molecular graphics-based structural tool kit for drug design.[1]

The specificity of interaction in many drug–receptor complexes is controlled by a few key receptor groups. The receptor binding site model can be surveyed to locate these sites of interaction.[2] Probe-based methods, such as GRID,[3] will explore the active site and contour it according to a measure of *in vacuo* binding energy. A feature of this approach is that it can be used to place small chemical groups inside the active site in positions of optimum enthalpy. This procedure is an important first step towards the design of novel compounds to bind to the site; there remains the difficult task of joining the regions of good interaction together to generate sensible, synthetically accessible, target molecules (Figure 1).

There are three principal methods for trying to construct molecular skeletons that bridge regions of favorable interaction between external (ligand) groups and receptor. The technique of database searching[4,5] can be used to find compounds that fit a site or a pharmacophore. Chains also can be built manually, but this is a complex time-consuming exercise for a chemist to perform. Structure generation programs[6–8] automatically produce molecular graphs that fit the site and link the interaction regimes, through the use of structural templates. These approaches have been recently combined within an interactive molecular graphics environment in the BUILDER program.[1]

Structure generation algorithms are combinatorial and would take an infeasibly long time to run, unless many simplifying heuristics are used. The diversity of molecular structures that can be produced is therefore limited. A method for chain generation based on torsion-angle constraints is a com-

Address reprint requests to Dr. Lewis at his present address: Dagenham Research Centre, Rhône-Poulenc Rorer Ltd., Rainham Road South, Dagenham, Essex RM10 7XS, UK.
Received 19 August 1991; accepted 7 October 1991

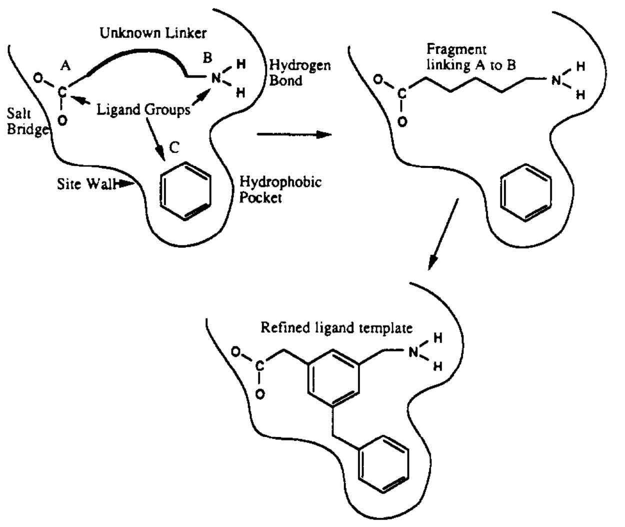


Figure 1. A chemist can begin the design of a new lead compound by graphically selecting and positioning complementary chemical groups A, B, *and* C *to interact with receptor site points. The torsion program can then be used to suggest a sensible chemical fragment to link groups* A *and* B. *The process can be repeated to incorporate group* C *into a refined ligand template.*

plementary technique that will produce a wider diversity of structures, without the need for excessive manual intervention or fitting. This would give the drug designer an extended choice of methods for creating structures; the greater diversity of suggested structures will aid in the design of several different target molecules for the site.

An unbranched chain of atoms has several degrees of rotational freedom, so that it could bridge a spatial gap between two chemical objects. The chemical objects would be small groups, like carboxylates or amines, positioned according to some design criterion. The constricted site would be formed by the surface of the receptor active site. The positions of the atoms in the chain depend only on the torsion angles, as all other quantities are known. If the positions of the chemical objects to be joined are also known, it is possible to determine the torsion angles necessary to bridge the gap and hence to generate explicitly a chemical chain that could connect the objects in an energetically reasonable manner.

METHODS

The program described here creates 3D chains by combining a structure generator with a set of routines for solving trigonometric equations in torsion-angle space. The torsion routines are a new implementation of the chain closure procedure described by Go and Scheraga:[9] this method has also been implemented by Bruccoleri[10,11] in the CONGEN package for constructing protein loops. An extensively modified version of the CLSCHN routine from CONGEN is used to solve the torsion equations for a system containing 6 degrees of freedom.

The mathematical analysis of chain closure given below was first described by Go and Scheraga[9] for a system containing 6 degrees of freedom. A local coordinate system can be defined by three nonlinear points. For a linked chain of three atoms, the origin may be defined by the first atom, the positive x-axis by the second atom and the xy-plane by the third atom. The z-axis is given by the cross product of the vectors in the xy-plane. In future, we will refer to a local coordinate system by the term *coordinate triad,* to refer to the way in which the local coordinate system is defined. The geometric relationship between two atoms of specified orientation (given by the directions of the vectors to bonded neighbors) can be defined in terms of the transformations required to superpose an orthogonal coordinate system based around the second atom onto a coordinate system based around the first atom (Figure 2).

The relationship between vectors $\mathbf{r}_0$ and $\mathbf{r}_1$ in adjacent coordinate triads can be described in terms of a translation and two rotations:

$$\mathbf{r}_0 = \mathbf{p}_0 + \mathbf{T}_0\mathbf{R}_1\mathbf{r}_1 \tag{1}$$

These rotations are analogous to a bond angle (θ_i) and bond dihedral (ω_i) rotation and the translation to a displacement along the x-axis by a bond length (d_i).

$$\mathbf{p}_i = \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{T}_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\omega_i) & -\sin(\omega_i) \\ 0 & \sin(\omega_i) & \cos(\omega_i) \end{bmatrix}$$

The relationship between vectors $\mathbf{r}_0$ and $\mathbf{r}_n$, described in two coordinate triads separated by a chain of n intermediate coordinate triads, can be obtained by multiplying out the individual relations to give Equation (2):

$$\begin{aligned}\mathbf{r}_0 = {} & \mathbf{p}_0 + \mathbf{T}_0\mathbf{R}_1\mathbf{p}_1 + \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{p}_2 \\ & + \cdots + \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2 \cdots \mathbf{T}_{n-2}\mathbf{R}_{n-1}\mathbf{p}_{n-1} \\ & + \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2 \cdots \mathbf{T}_{n-2}\mathbf{R}_{n-1}\mathbf{T}_{n-1}\mathbf{R}_n\mathbf{r}_n \end{aligned} \tag{2}$$

If $\mathbf{r}_n$ is the zero vector, the position vector for the origin of

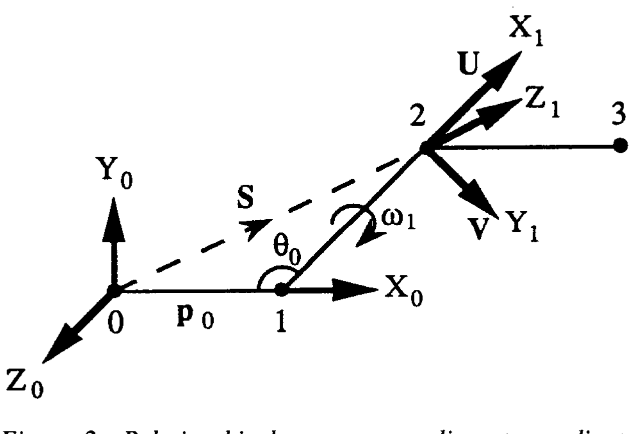


Figure 2. Relationship between two adjacent coordinate systems, $X_0/Y_0/Z_0$ *and* $X_1/Y_1/Z_1$ *can be described in terms of the angles* θ_0 *and* ω_1, *and the vector* $\mathbf{p}_0$.

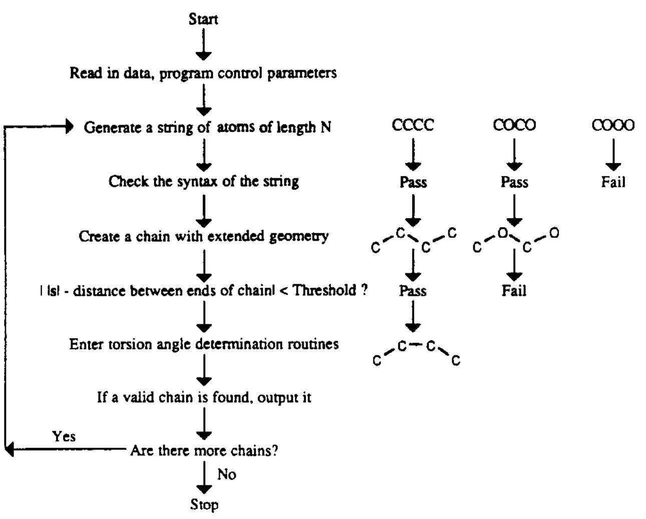
the nth coordinate triad $\mathbf{s}$ can be expressed as:

$$\mathbf{s} = \mathbf{p}_0 + \mathbf{T}_0\mathbf{R}_1\mathbf{p}_1 + \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{p}_2 + \cdots + \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2 \cdots \mathbf{T}_{n-2}\mathbf{R}_{n-1}\mathbf{p}_{n-1} \quad (3)$$

If it is assumed that bond lengths and angles are known quantities; that is, their values can be read from a table of standard bond parameters, Equation (3) now contains unknown terms that involve only torsion angles (in $\mathbf{R}_i$). It can be shown that there are 6 independent equations that describe the relationship between coordinate triad vectors. This implies that systems containing up to 6 unknown torsion angles can be solved. A system that contains only 5 unknown torsion angles is therefore overdetermined. The geometric constraints are applied by fixing the position in space of the first and last coordinate triads in the chain: this gives the 6 necessary pieces of information needed to determine the values of the dependent variables, the torsion angles. The vector $\mathbf{p}_i$ does not have to have zero y- and z-components for this argument to be valid. Go and Scheraga[9] limited their discussion to the special case of the peptide unit, where only alternate $\mathbf{p}$ vectors could have nonzero y-components. However in any other case, the author has found that the analysis of the vector equations (given in Appendix A) becomes very complex, especially if the z-component is nonzero. The simple form of $\mathbf{p}_i$ is assumed in the analysis, as there are very few bonding arrangements where the more general form of $\mathbf{p}_i$ is needed. The basic equations for describing a chain of coordinate triads in space are given in Appendix A, together with a new analysis of the case, not considered by Go and Scheraga, where there are 6 constraints and only 5 torsion angles. Similarly, the CLSCHN routine from CONGEN is restricted to the case of finding 6 torsion angles only. This restriction has been lifted in our program; an implementation of the novel analytic method for finding the values of the 5 torsion angles is given in FORTRAN 77 code in Appendix B. The analyses of the cases where there are 1, 2, 3 or 4 torsion angles are similar but more straightforward, and have been omitted for brevity. In addition, the CLSCHN routine has been greatly rewritten to convert it from the FLEX programming language into portable, modular FORTRAN 77 code.

Bruccoleri and others have used the chain closure procedure to find acceptable geometries of short protein loops.[11] In this work, the chain closure technique is extended beyond the geometry of the peptide unit and is applied to the generation of 3D molecular graphs. The geometric constraints are derived from the coordinate triads defined by two sets of three linked atoms. These coordinate triads could be defined by chemical groups that have been positioned to interact strongly with key atoms in the binding site. This information may be obtained from the results of a program like GRID,[3] by using atoms contained in an existing molecular fragment, or simply by modeling in small chemical groups.

The flowchart in Figure 3 describes, in pseudo code, the structure of the program. There are two main components to the program: a structure generator and a set of routines to solve the torsion equations. A modified version of the CLSCHN routine[10] is used to solve the torsion equations for systems with 6 degrees of freedom. The program accepts Brookhaven database format[12] files but can be simply changed



Start
Read in data, program control parameters
Generate a string of atoms of length N
Check the syntax of the string
Create a chain with extended geometry
I IsI - distance between ends of chainI < Threshold ?
Enter torsion angle determination routines
If a valid chain is found, output it
Yes
Are there more chains?
No
Stop
CCCC
COCO
COOO
Pass
Pass
Fail
Pass
Fail

Figure 3. Flow chart of the program logic, with three examples. The string COOO fails through having forbidden syntax and the string COCO fails by being too short. The string CCCC produces a viable chain that satisfies the constraints.

to accept other formats. The other input to the program consists of standard data files containing van der Waals radii, bond length and bond angle information and program parameters. A string of atom symbols is generated, parsed and then expanded with the correct values of bond lengths and angles to produce a prototype 3D chain. This information is passed into the routines that solve the torsion equations, to see whether the chain can satisfy the constraints and produce a set of real values for the torsion angles. The program iterates to generate another chain. The output of the program consists of a file (in Brookhaven format) containing a series of different chains that meet all the geometric requirements imposed by the positions of the coordinate triads. These chains should be displayed and modified to suggest target compounds for synthesis and testing.

The generation of a string of atoms is performed by a milometer-type algorithm. The palette of atoms used in the string generation is under user control. A larger set of atoms will give rise to a greater diversity of chains but the program will take longer to run. The number of different strings of length n that can be constructed from k objects is k^n. The task is therefore combinatorial in nature and must be pruned to achieve reasonable running times. At present, the program is parameterized only for primary atom types like C sp^3, N sp^3 and O sp^3. Other atom types and composite atom groups are being added to this basis set.[13]

The next two stages of the program apply filters to quickly remove strings that are obviously useless. The syntax of the string of atoms is checked and parsed against a set of user-controllable rules. The BADLIST from DENDRAL[14] has been used as a basis for the syntax rules. Strings that have disallowed syntax, corresponding to unstable arrangements of atoms, such as O sp^3–O sp^3–O sp^3, are removed. Other rules can be written to control which atoms are in certain positions in the chain, and so on.

Each string is expanded into an extended chain with ideal bond angles and lengths and torsion angles set to 180° by repeatedly applying the transformation in Equation (1). The bonding parameters are obtained from a look-up table that contains standard equilibrium bond statistics. The distance between any two atoms in the chain is maximal for that chain composition, as all the torsions are staggered. If the magnitude of the vector s—Equation (3); Figure 2—is greater than the distance between the atoms at the end of the chain, the chain cannot possibly bridge the gap and is discarded; otherwise the chain is taken forward for torsion fitting. A slight error is allowed in the comparison of the chain length and s: this is set externally as the matching threshold. Values of 0.5 Å and 1.0 Å have been used for the threshold in this work.

The new routines for solving the torsion equations are analytic and fast. Some care has been taken to ensure that the routines are robust, so that small errors in the roots of the trigonometric equations are removed where necessary. Errors due to rounding and the imaginary components of equation roots are approximated where possible, and all trigonometric results are subjected to renormalization using the standard identity $\cos^2(\theta) + \sin^2(\theta) = 1$. This protocol is intended to prevent errors propagating during back-substitution of results. The torsion-solving routines return the dihedral angles that are used to set up the correct coordinates of the chain by applying the transformations detailed in Equation (1). The atoms in the adjusted chain can then be checked for van der Waals (steric) clashes with the site atoms. Some slight bumps are allowed, because a receptor site is plastic and might be able to accommodate a steric clash at only a slight energetic cost. If there are no solutions to the equations, it is because the chain, represented by that set of bond angles and bond lengths, cannot be adjusted to fit the constraints. Some chains do not match the constraints exactly, due to the approximations that have been applied. The matching threshold is used to filter these chains out, by specifying the minimum acceptable root-mean-square (rms) match between the chain atoms and the atoms used to define the geometric constraints.

The chains that give rise to a viable set of torsion angles, and that do not fail the bump check, are output for further examination by the user. The cycle of chain generation and testing is repeated for all strings containing 1 to 6 torsion angles and every permutation of the available atom types.

The program was tested by three different procedures. In each experiment, the geometric constraints were set up from coordinate triads derived from an arbitrary alkane (undecane) conformer built using the Nemesis molecular modeling program.[15] Atom palettes of C sp^3 only or C sp^3, N sp^3 and O sp^3 were used in the test runs and the bump-check option was turned off. The runs were performed on a VAX 8700 operating under VMS 5.3. The routines were stringently tested with constraints of significantly non-ideal geometry. These tests imitate the real drug design situation, where there may be uncertainty about the exact positioning of a ligand group. A method that requires the atoms defining the constraints to be ideally positioned would be useless.

Experiment 1: perturbation of bond parameters

A staggered conformer of undecane was built using the Nemesis molecular modeling program,[15] using the Nemesis default values for bond lengths and angles. The geometric constraints were defined by consecutive triplets of atoms in the undecane molecule, corresponding to alkyl chains with 3, 4, 5 or 6 intervening bonds (torsion angles). For this experiment, the atom palette used by the generation program was set to contain only the C sp^3 atom type. The default values for bond angles and lengths used in the atom palette look-up table are not the same as those used by Nemesis:[15] the average size of the differences between the two sets of values is about 1%. The chain length was set to be the same as the number of bonds connecting the two coordinate triads in the reference molecule.

Experiment 2: rotational perturbation of the geometric constraints

The following two experiments involve the random perturbation of the geometric constraints. The atom palette was set at C sp^3, N sp^3 and O sp^3, and the matching threshold at 1.0 Å. The coordinates of one consecutive triplet of atoms of a reference undecane molecule were used to define a reference coordinate triad. A second consecutive triplet of atoms, separated by 3, 4, 5 or 6 intervening bonds (torsion angles) from the first set, was used to define the target coordinate triad. The target coordinate triad was moved towards the reference coordinate triad along a vector of magnitude 0.5 Å. This was to introduce some latitude into the trial; the chains could then contract slightly to fit the perturbed coordinate triad. The translated target coordinate triad was subjected to a compound Euler rotation (Figure 4). Each Euler angle was set randomly to have a value between 0° and the *Tilt* parameter, which was set at either 5° or 10°. One hundred random trials were made.

Experiment 3: translational perturbation of the geometric constraints

In this test, the target coordinate triad was subjected to a translational jolt of size 0.5 or 1.0 Å and of random direction, while the reference coordinate triad was held steady (Figure 4). The atom palette was set at C sp^3 only or C sp^3, N sp^3 and O sp^3, and the matching threshold was set at 0.5 Å. One hundred random trials were made.

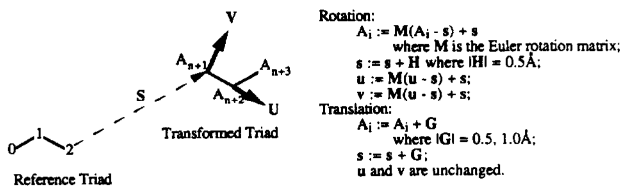


Figure 4. Random tests of the algorithm were performed by perturbing coordinate triads obtained from reference sets of ideal geometry. The perturbations could translate or rotate the target coordinate triad of atoms A_1, A_2, A_3 *to give a transformed coordinate triad, according to the equations given in the figure.*

RESULTS

The experiments were devised with the aim of checking whether the routines were robust enough to overcome the perturbations in the geometric constraints without crashing, and whether the program could generate chains of similar geometry to the reference molecule.

Experiment 1: perturbation of bond parameters

It was observed that chains of similar geometry to the reference undecane molecule were always generated: this result was reproduced for a variety of other conformations of the undecane molecule. The rms difference between the reference molecule and the generated chains was always less than 0.1 Å. This result demonstrates that the method can accommodate minor deviations from the ideal.

Other chains of alternative geometry were generated from the same sets of geometric constraints. The analysis of the torsion equations involves the solution of several quadratic equations, so that there can be multiple solutions for each set of constraints. An upper limit on the number of solutions is twice the number of quadratic equations involved. The limit for 3 or fewer torsion angles is one; for 4 angles there are two solutions; for 5 angles, four; and for 6 angles, eight. This feature will be useful for generating alternate chains to fit a constricted site.

Experiment 2: rotational perturbation of the geometric constraints

The results of the random *Tilt* test are described in Table 1. The percentage of successful runs (where at least one solution was produced for the set of random constraints) is given. The average number of chains (to 1 significant figure) generated in each successful run is given: this figure should be compared to the theoretical upper limit for the run. There is a substantial variance to ⟨*chains*⟩: for the trial with the number of torsion angles = 4, atom types = C sp^3, N sp^3 and O sp^3, and *Tilt* = 5°, the number of chains found for a given set of constraints varied between 90 and 4. The figure given for ⟨*chains*⟩ is intended to give an indication of the potential diversity of the chains that could be found by this approach. The difference between the figures for ⟨*chains*⟩ and the upper limit reflects not only that some chains failed to give a viable set of torsion angles for the constraints, but also that some chains will have been filtered out by the syntax rules and the matching threshold.

As well as the chains that are filtered out as described above, some failures arose because the rotation of a coordinate triad will sometimes produce an impossible kink; an extreme case would occur when the target coordinate triad is effectively inverted and the chain would have to loop back to join the coordinate triad from the correct direction. The low rate of failure seen in the *Tilt* test arises from the way in which the torsion equations are solved. The vector **s** is held constant, and **u** and **v** are being varied. The perturbations can be accommodated easily by changes in the last few dihedral angles, to produce chains that match the constraints well (fit is measured by an rms statistic). The values of the first dihedrals are used to compute the values of the later dihedrals; only a low level of error can be tolerated, as the error will be propagated and increased during back-substitution. This restriction can be relaxed in the later stages of the calculation. The algorithm is more robust towards changes in orientation than changes in separation of the constraining coordinate triads.

Experiment 3: translational perturbation of the geometric constraints

The results of experiment 3 are given in Table 2. The column headings are the same as in Table 1. The failure rate here is higher than in experiment 2. Runs performed with the larger palette of atoms generated more solutions, as there were more strings to test, but one would not expect to see the full increase implied by the combinatorial expansion in the absolute number of strings.

The failure rate was variable, but generally there were more successful runs for chains of length 4 and 6. The reasons for this are subtle: one would expect that the greater conformational freedom of the longer chains would allow them to fold up and thereby accommodate the deviations from ideal geometry, giving a trend of increasing solutions with length. The reason that chains of length 5 do less well arises directly from the way in which the trigonometric equations are solved. The code for initialization and for determining the value of ω_1 (Appendix B) involves only the constraint term **s**. The value of **s** is slightly perturbed in this experiment, but this effect does not feed strongly into the computation of ω_1. As a result, the values of the dihedrals ω_1 and ω_2 are similar for most of the chains, but the later equations for finding ω_4 and ω_5 (in routine FindLastOmegas; Appendix B) cannot be satisfied, even within the approximate limits. Chains of length 4 are determined in a more sensitive way to changes in **s**, and the algorithm for chains of length 6 uses Bruccoleri's protocol for perturbing bond

Table 1. Results of experiments involving the random rotation of the target coordinate triad away from ideal geometry

Number of torsion angles	Atoms	*Tilt*/degrees	CPU Time/s	% Successes	⟨*chains*⟩ found	Upper limit
4	C, N, O	5	37	100	50	162
4	C, N, O	10	19	10	25	
5	C, N, O	5	65	100	70	972
5	C, N, O	10	62	89	50	
6	C, N, O	5	2768	100	600	5832
6	C, N, O	10	2152	97	300	

Table 2. Results of experiments involving the random translation of the target coordinate triad away from ideal geometry

Number of torsion angles	Atoms	*Jolt*/Å	CPU Time/s	% Successes	⟨*chains*⟩ found	Upper limit
4	C, N, O	0.5	33	100	50	162
4	C, N, O	1.0	19	10	40	
4	C	0.5	5	55	2	2
4	C	1.0	5	6	2	
5	C, N, O	0.5	46	12	50	972
5	C, N, O	1.0	45	10	50	
5	C	0.5	5	6	2	4
5	C	1.0	5	6	2	
6	C, N, O	0.5	465	32	900	5832
6	C, N, O	1.0	821	30	2800	
6	C	0.5	6	30	4	8
6	C	1.0	6	13	6	

lengths and angles to overcome early errors. All methods will fail when the jolt moves the target coordinate triad further away from the reference coordinate triad. If the magnitude of **s** exceeds the sum of the maximum chain length and the matching threshold, no solution could possibly be found, and the strings will be filtered out before fitting. The drop-off in performance with an increase in the jolt size from 0.5 to 1.0 Å for chains of length 4 is understandable. A perturbation of 1.0 Å could change **s** by up to 20%, as in this case the distance between the target and reference coordinate triads is only 5.0 Å. It is encouraging that any solutions at all were found.

These random tests indicate that, overall, the method is fast and robust and could be of general use in the generation of 3D chains for site-directed drug design.

DISCUSSION

Numerical stability

A new method for generating and testing general 3D molecular chains has been described in this paper. The approach involves the analytic solution of trigonometric equations in torsion-angle space. An important criticism that could be leveled against this technique is that the analytical algorithms may not be numerically robust. A small and unpredictable variation in the geometric constraints, corresponding to a small displacement of one of the coordinate triads could make the difference between success and failure in finding a chain. There are a number of ways in which the possibility of failure can be reduced. There are two cases to be considered: for chains containing 5 or less torsion angles, the new analytic routines are used and a series of quadratic equations must be satisfied. For chains containing 6 torsion angles, Bruccoleri's CLSCHN routine is used: the roots are found numerically, as the torsion equations have no closed solution.

The approach adopted by Bruccoleri[10] perturbs the bond length and angle parameters before restarting the process of finding the torsion angles. This procedure is only carried out if the program has detected a near-miss when trying to find allowed values for the dihedrals: this means that the polynomial torsion equation for ω_1 does not have real roots and does not cross the ω_1-axis. The values of the independent angles and lengths are changed by a minimizer. The minimum of the torsion equation is lowered until the ω_1-axis is crossed and real roots to the torsion equation are found. This is satisfactory for building protein backbones, where there is only one standard unit, the peptide linkage. There is, of course, a penalty in the form of increased run time, depending on the fineness of the search over angle and length space. This method will not be so useful when there may be several different atomic units. The perturbations are introduced at this level, in the sense that bonds to C sp^3, N sp^3 or O sp^3 do not have greatly different dimensions. These small differences are analogous to an iterative perturbation step under Bruccoleri's protocol.

Palmer has recently introduced a similar approach[16] that focuses on maintaining the standard geometry of the peptide unit in different amino acid residues. The search for torsion angles is extended by lengthening the chain so that there are more than six variable dihedrals. The excess dihedrals are independent variables that are set up by a minimizing routine before the remaining six dependent angles are determined. The advantage of working solely with dihedral angles is that the standard equilibrium geometry of each unit in the chain is maintained. Grid searches of the independent angles may also be performed to increase the chances of finding several chains that satisfy the geometric constraints.

The chances of missing a quasisolution also can be lessened by increasing the numerical robustness of the algorithms. The existence of real noncomplex roots to a quadratic equation is determined by the absence of an imaginary term in the discriminant. A small imaginary component of a discriminant can be discarded to give two identical roots. These roots are approximate: as other variables are obtained by back substitution, it is important to recognize the danger of propagating errors. Error checks and renormalizations are performed at several stages of the computation. A direct result of accepting small errors is that the final geometry of the generated chain will not be the same as the reference chain. The maximum allowed deviation is set through the threshold matching parameter, typically at a value of 0.5 Å. This is acceptable because of the nature of the molecular graph being created. The model of the site cannot be assumed to be completely accurate, and the coordinate triads will not be necessarily oriented in a position of optimal free energy within the site. It is therefore more important to create molecular templates that suggest synthetically accessible target structures to the drug designer. It would be possible to use the testing protocols to perturb a coordinate triad until a viable chain is found. The coordinate triad will obviously be forced out of position by a jolting or reorienting transformation. The magnitude of the perturbations can be controlled by the user; their size could be set according to the sensitivity to changes in geometry of the ligand–receptor interactions represented by the coordinate triads. A salt bridge in a tight pocket should not be disturbed, but a hydrogen bond might be more tolerant of changes in the orientation of the interacting groups.

The final strategy chosen is a combination of atom type diversity, error flattening for chains with 1 to 5 torsion angles (where the equations are solved analytically) and bond length and angle iteration for chains with 6 torsion angles (where the equations are solved numerically).

Filtering and scoring the chains

The introduction of a greater palette of atom types is the subject of current research. A direct result of introducing more atom types is that several hundred different chains could be generated for a given set of constraints. For instance, there are 729 different possible strings of length 6 that can be made up from 3 atom types. The combinatorial explosion in chain composition and geometries usefully increases the constraint space that can be satisfied by a chain of a given length. Some chains will be syntactically disallowed: it is very hard to predict the fraction that will be filtered out, as this is strongly dependent on the atom types and the rules used. Geometric filtering, on the basis of extended chain length, can also be very effective in pruning many potential 3D molecular graphs at an early stage. As with all heuristics, its effectiveness is context-dependent.

By the end of the generation and torsion-fitting stages, a tenfold reduction in the number of chains has been observed. Due to the number of molecular graphs that can be produced, it is still necessary to score the suggestions so that the most promising are ranked highest. It would be possible to evaluate the internal energy of the chains, but as the bond lengths and angles are set at ideal values, only the torsion-energy term would be providing any discrimination. It would not be sensible to exclude a molecular template on this ground alone, unless it was in a high energy conformation. A force field calculation of the receptor–ligand system is also not an ideal scoring metric, as energy functions only compute the enthalpic stability of a structure, not the free energy. When a ligand binds to a receptor, it is removed from solvent and fixed within a binding pocket. When the compound contains several freely rotatable bonds, there will be a substantial entropic penalty to binding. A designer should seek to brace up the generated chain with rings to reduce the size of the entropic cost of binding.

A more sensible ranking scheme might be based on synthetic accessibility. This is equivalent to parsing the string of atom types and matching it up to a synthetic template whose fitness is known.[17] This again is not entirely satisfactory, as these templates are only suggestions; they should not be seen as ends within themselves, but as sensible templates that can, and should be, modified to match more closely the demands of complementarity and synthetic accessibility.

At present, the ranking function is based on rms fit of the chain atoms to the coordinate triads that define the geometric constraints. It is envisaged that, ultimately, the program will be interfaced into the BUILDER package.[1] Here, one would be trying to link together larger fragments such as ring systems, as well as perhaps small fragments, for instance, a carboxylate group. It is important to maintain a more precise geometry, so that the generated chains that have the best fit to the existing fragments should be considered first. The overproduction of suggestions is a benign difficulty, given the objective of the program is to provide a diverse selection of molecular graphs to link interaction regimes.

CONCLUSION

A procedure for generating general 3D chains within geometric constraints is described. It has been shown to be a fast and reliable method for creating molecular graphs, without the need for excessive manual intervention or fitting. The program was tested for robustness and was found to be stable towards perturbations in the geometric constraints. Each constraint is made up of a coordinate triad of three atoms: in drug design, the constraints could be derived from the coordinates of chemical groups positioned to interact strongly with receptor atoms. These molecular graphs could then serve as useful structural templates for joining up regions in a specified active site where the ligand might interact strongly with the receptor.

The program has been designed for ease of interactive use: the diversity and number of molecular graphs can be controlled by changing the palette of atom types, and the threshold of acceptable fit of the chains to the constraints is also under user control. The method is intended to complement existing approaches for site-directed drug design, and to form a useful component of any molecular graphics-based structural tool kit for drug design.

ACKNOWLEDGEMENTS

RAL would like to thank the Royal Commission of 1851 for a Research Fellowship and the Fulbright Commission for a Senior Scholarship. This work was partially supported by grant GM-31497 (Prof. I.D. Kuntz, Principal Investigator). Dr. R.E. Bruccoleri is gratefully acknowledged for providing a copy of the CONGEN program, and Prof. I.D. Kuntz, Dr. M.J.E. Sternberg and Dr. P.M. Dean are thanked for helpful discussions.

REFERENCES

1 Lewis, R.A., Roe, D.C., Huang, C., Ferrin, T.E., Langridge, R. and Kuntz, I.D. Automated site-directed drug

design using molecular lattices. *J. Mol. Graph.* 1992, **10**, 66–78
2 Danziger, D.J. and Dean, P.M. Automated site-directed drug design: the prediction and observation of ligand point positions at hydrogen bonding regions on protein surfaces. *Proc. Roy. Soc.* 1989, **B236**, 115–124
3 Boobbyer, D.N., Goodford, P.J., McWhinnie, P.M. and Wade, R.C. New hydrogen bond potentials for use in determining energetically favorable binding sites on molecules of known structure. *J. Med. Chem.* 1989, **32**, 1083–1094
4 DesJarlais, R.L., Sheridan, R.P., Seibel, G.L., Dixon, J.S., Kuntz, I.D. and Venkataraghavan, R. Using shape complementarity as an initial screen in designing ligands for a receptor binding site of known three-dimensional structure. *J. Med. Chem.* 1989, **31**, 722–729
5 Van Drie, J., Weininger, D. and Martin, Y.C. ALADDIN: an integrated tool for computer-assisted molecular design and pharmacophore recognition from geometric, steric, and substructure searching of three-dimensional molecular structures. *J. Comp.-Aided Mol. Design* 1989, **3**, 225–251
6 Lewis, R.A. and Dean, P.M. Automated site-directed drug design: the concept of spacer skeletons for primary structure generation. *Proc. Roy. Soc.* 1989, **B236**, 125–140
7 Lewis, R.A. and Dean, P.M. Automated site-directed drug design: the formation of molecular templates in primary structure generation. *Proc. Roy. Soc.* 1989, **B236**, 141–162
8 Lewis, R.A. Automated site-directed drug design: approaches to the formation of three-dimensional molecular graphs. *J. Comp.-Aided Mol. Design* 1990, **4**, 205–210
9 Go, N. and Scheraga, H.A. Ring closure and local conformational deformations of chain molecules. *Macromolecules* 1970, **3**, 178–187
10 Bruccoleri, R.E. and Karplus, M. Chain closure with bond angles variations. *Macromolecules* 1985, **18**, 2767–2773
11 Bruccoleri, R.E. and Karplus, M. Prediction of the folding of short polypeptide segments by uniform conformational sampling. *Biopolymers* 1987, **26**, 137–168
12 Bernstein, F.C., Koetzle, T.F., Williams, G.J.B., Meyer, E.F., Brice, M.P., Rogers, J.R., Kennard, O., Shimanouchi, T. and Tasumi, M. The Protein Data Bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.* 1977, **112**, 535–542
13 Lewis, R.A. 1991, unpublished results
14 Lindsay, R.K., Buchanan, B.G., Feigenbaum, E.A. and Lederberg, J. *Applications of Artificial Intelligence for Organic Chemistry: the DENDRAL project.* McGraw-Hill, New York (1980)
15 Nemesis V1.0. 1990, available from Oxford Molecular Ltd
16 Palmer, K.A. and Scheraga, H.A. Standard-geometry chains fitted to X-ray derived structures: validation of the rigid geometry approximation. I. Chain closure through a limited search of "loop" conformations. *J. Comp. Chem.* 1991, **12**, 505–526
17 Hendrickson, J.B., Grier, D.L. and Toczko, A.G. A logic-based program for synthesis design. *J. Am. Chem. Soc.* 1985, **107**, 5228–5238

APPENDIX A

The geometric relationship between linked atoms along a chain can be described in terms of two rotation matrices and a vector. The vector $\mathbf{p}_i$ describes the displacement along the bond vector; the matrix $\mathbf{T}_i$ describes the rotation through the bond angle θ_i and $\mathbf{R}_i$ the rotation through the dihedral ω_i.

$$\mathbf{p}_i = \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{T}_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\omega_i) & -\sin(\omega_i) \\ 0 & \sin(\omega_i) & \cos(\omega_i) \end{bmatrix}$$

The three basic equations describing the torsion relations are

$$\mathbf{s}_n = \mathbf{p}_0 + \mathbf{T}_0\mathbf{R}_1\mathbf{p}_1 + \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{p}_2 + \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{T}_2\mathbf{R}_3\mathbf{p}_3 + \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2 \cdots \mathbf{T}_{n-2}\mathbf{R}_{n-1}\mathbf{p}_{n-1} \quad \text{(A.1)}$$

$$\mathbf{u}_n = \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2 \cdots \mathbf{T}_{n-1}\mathbf{R}_n\mathbf{i} \quad \text{(A.2)}$$

$$\mathbf{v}_n = \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2 \cdots \mathbf{T}_{n-1}\mathbf{R}_n\mathbf{j} \quad \text{(A.3)}$$

The vectors $\mathbf{s}_n$, $\mathbf{u}_n$ and $\mathbf{v}_n$ are derived from the constraints and have known numerical values. The vectors $\mathbf{p}_i$ and the matrices $\mathbf{T}_i$ are derived from the bond length and angle information associated with the string of atoms. $\mathbf{i}$ and $\mathbf{j}$ are unit vectors along the x- and y-axes, respectively. The unknown variables are all contained within the matrices $\mathbf{R}_i$.

We now give an analysis of the equations to deduce explicit analytic equations for each of the independent variables. Much of the algebraic manipulation is straightforward but lengthy and has been omitted for the sake of brevity. Readers wanting to see each step in detail are recommended to follow the analysis with the aid of a symbolic manipulation package such as Mathematica (Wolfram, S. *Mathematica.* (1988) Addison-Wesley, New York). The final expressions are given in the accompanying FORTRAN 77 code. Let

$$\mathbf{q}_i = \mathbf{p}_{2i} + \mathbf{T}_{2i}\mathbf{R}_{2i+1}\mathbf{p}_{2i+1} = \begin{pmatrix} d_{2i} + d_{2i+1}\cos(\theta_{2i}) \\ d_{2i+1}\sin(\theta_{2i}) \\ 0 \end{pmatrix} \quad \text{(A.4)}$$

$$\mathbf{t} = \mathbf{s} - \mathbf{q}_0 \quad \text{(A.5)}$$

For the case where there are 5 unknown torsion angles, substitution of Equations (A.4) and (A.5) into Equation (A.1) gives:

$$\mathbf{t} = \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{q}_1 + \mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{T}_2\mathbf{R}_3\mathbf{T}_3\mathbf{R}_4\mathbf{q}_2 \qquad (A.6)$$

or

$$\mathbf{r} = (\mathbf{T}_0\mathbf{R}_1\mathbf{T}_1)^{-1}\mathbf{t} = \mathbf{R}_2(\mathbf{q}_1 + \mathbf{q}_3) \qquad (A.7)$$

where $\mathbf{q}_3 = \mathbf{T}_2\mathbf{R}_3\mathbf{T}_3\mathbf{R}_4\mathbf{q}_2$.

Note that $\mathbf{r}$ only contains terms in ω_1 and that, because length is invariant under rotation, $|\mathbf{r}| = |\mathbf{t}|$ and $|\mathbf{q}_3| = |\mathbf{q}_2|$. Using this last fact, we can rearrange Equation (A.7) to give

$$|\mathbf{q}_2|^2 = |\mathbf{q}_3|^2 = |\mathbf{R}_2^{-1}\mathbf{r} - \mathbf{q}_1|^2 \qquad (A.8)$$

Equation (A.8) can be expanded to

$$\mathbf{r}_y \cos(\omega_2) + \mathbf{r}_z \sin(\omega_2) = (\mathbf{r}^2 - \mathbf{q}_1^2 - \mathbf{q}_2^2 - 2\mathbf{r}_x\mathbf{q}_{1,x})/2\mathbf{q}_{1,y} \qquad (A.9)$$

where the subscripts x, y, and z indicate the x-, y- or z-component of a particular vector. Equation (A.9) is of the form $A + B\cos(\phi) + C\sin(\phi) = 0$, which can be cast into a quadratic equation using the identity $\cos^2(\phi) + \sin^2(\phi) = 1$. The resulting quadratic has a discriminant term given by

$$4B^2(B^2 + C^2 - A^2) \geq 0 \qquad (A.10)$$

If there are to be real solutions to Equation (A.9), then the inequality in Equation (A.10) must be true. Substituting the coefficients from Equation (A.9) into Equation (A.10) and using the Pythagorean relation $\mathbf{r}^2 = \mathbf{r}_x^2 + \mathbf{r}_y^2 + \mathbf{r}_z^2$ to eliminate $\mathbf{r}_y$ and $\mathbf{r}_z$, we obtain another quadratic inequality in $\mathbf{r}_x$, all other quantities being known.

$$4\mathbf{q}_1^2\mathbf{r}_x^2 - 4\mathbf{q}_{1,x}(\mathbf{r}^2 + \mathbf{q}_1^2 - \mathbf{q}_2^2)\mathbf{r}_x + (\mathbf{q}_1^4 - 2\mathbf{q}_1^2\mathbf{q}_2^2 + \mathbf{q}_2^4 + 2\mathbf{q}_1^2\mathbf{r}^2 - 2\mathbf{q}_2^2\mathbf{r}^2 + \mathbf{r}^4 - 4\mathbf{q}_{1,y}^2\mathbf{r}^2) \leq 0 \qquad (A.11)$$

Hence, if $\mathbf{r}_x$ is to obey Inequality (A.11), the discriminant of the quadratic in $\mathbf{r}_x$ must be real, and $\mathbf{r}_x$ must lie between the roots of the quadratic. The first condition gives, by expanding and factoring the discriminant term (see term D in Equation (A.13)), the bounds

$$||\mathbf{q}_1| - |\mathbf{q}_2|| \leq \mathbf{r} \leq ||\mathbf{q}_1| + |\mathbf{q}_2|| \qquad (A.12)$$

The second condition gives

$$\{4\mathbf{q}_{1,x}(\mathbf{r}^2 + \mathbf{q}_1^2 - \mathbf{q}_2) - \mathbf{q}_{1,y}D\}/2\mathbf{q}_1^2 \leq \mathbf{r}_x \leq \{4\mathbf{q}_{1,x}(\mathbf{r}^2 + \mathbf{q}_1^2 - \mathbf{q}_2^2) + \mathbf{q}_{1,y}D\}/2\mathbf{q}_1^2$$
$$D^2 = ((|\mathbf{q}_1| + |\mathbf{q}_2|)^2 - \mathbf{r}^2)(\mathbf{r}^2 - (|\mathbf{q}_1| - |\mathbf{q}_2|)^2) \qquad (A.13)$$

Conditions (A.12) and (A.13) can be checked at the outset with the value of $\mathbf{r}_x$, to see if there will be any valid solutions. We can solve for $\mathbf{r}_x$ by rearranging Equation (A.7) and equating x-components:

$$(\mathbf{R}_3\mathbf{T}_3\mathbf{q}_2)_x = (\mathbf{T}_2^{-1}(\mathbf{R}_2^{-1}\mathbf{r} - \mathbf{q}_1))_x \qquad (A.14)$$

This gives

$$\mathbf{r}_y \cos(\omega_2) + \mathbf{r}_z \sin(\omega)_2 = (\cos(\theta_3)\mathbf{q}_{2,x} + \cos(\theta)_2\mathbf{q}_{1,x} + \sin(\theta_2)\mathbf{q}_{1,y} - \mathbf{r}_x \cos(\theta_2)/\sin(\theta_2) \qquad (A.15)$$

and $\mathbf{r}_x$ follows by simultaneous solution of Equations (A.9) and (A.15); $\mathbf{r}_y$ can be found by substituting the value of $\mathbf{r}_x$ into the expansion of:

$$(\mathbf{R}_1\mathbf{T}_1\mathbf{r})_x = (\mathbf{T}_0^{-1}\mathbf{t})_x \qquad (A.16)$$

The value of ω_1 can now be determined from the corresponding y-component of Equation (A.16) and the value of $\mathbf{r}_z$ from the z-component. All the unknowns in Equation (A.9) have been found, so that ω_2 can be computed. Substituting for ω_2 in Equation (A.7) gives ω_3. To find ω_4 and ω_5, we return to Equations (A.2) and (A.3) to note that:

$$(\mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{T}_2\mathbf{R}_3\mathbf{T}_3)^{-1}\mathbf{u} = \mathbf{R}_4\mathbf{T}_4\mathbf{R}_5\mathbf{i} = \begin{pmatrix} \cos(\theta_4) \\ \sin(\theta_4)\cos(\omega_4) \\ -\sin(\omega_4)\sin(\omega_4) \end{pmatrix} \qquad (A.17)$$

$$(\mathbf{T}_0\mathbf{R}_1\mathbf{T}_1\mathbf{R}_2\mathbf{T}_2\mathbf{R}_3\mathbf{T}_3\mathbf{R}_4\mathbf{T}_4)^{-1}\mathbf{v} = \mathbf{R}_5\mathbf{j} = \begin{pmatrix} 0 \\ \cos(\omega_5) \\ \sin(\omega_5) \end{pmatrix} \qquad (A.18)$$

In both cases, the lefthand side of the vector equation can be computed by back-substitution of the other determined variables. The values of ω_4 and ω_5 are found by equating components as before.

APPENDIX B

COMMONS.H

```
c
c Include file ''commons.h''
c
        real stheta(0:5), ctheta(0:5), somega(6), comega(6)
        real p(3, 0:5), q(3, 0:2), r(3), s(3), u(3), v(3)
        real err_root, trigaccy
        common/chain_angles/stheta, ctheta, somega, comega
        common/global/p, q, r, s, u, v
        common/chain_accy/err_root, trigaccy
```

SOLVE5.F

```
      subroutine Solve5(omega, n, iter, devmax, infoprint)
      include 'commons.h'
c
c 5 torsion angles; 4 roots
c
      real omega(n), sign1, sign2, a, b, c, w, t(3)
      save sign1, sign2, a, b, c, w
      maxrt = 4
c all roots considered?
      if(abs(iter).ge.maxrt)then
        iter = 0
        return
      endif
c start of initialization
      if (iter.eq.0) then
        tsum = 0.0
        q1s = 0.0
        q1d = 0.0
        q1m = 0.0
        q2m = 0.0
        do 10, i = 1, 3
          t(i) = s(i) - q(i,0)
          tsum = tsum + t(i) * t(i)
          q1m = q1m + q(i,1)**2
          q2m = q2m + q(i,2)**2
 10     continue
        q1mr = sqrt(q1m)
        q2mr = sqrt(q2m)
        q1s = (q1mr + q2mr)**2
        q1d = (q1mr - q2mr)**2
        r2 = tsum
        disc1 = q1s - r2
        disc2 = r2 - q1d
        sign1 = 3.0
        sign2 = -1.0
c test the discriminants
        if((disc1.lt.(-err_root))
     1        .or.(disc2.lt.-(err_root)))then
          iter = -maxrt
          return
        else
          if(disc1.lt.0.0)disc1 = err_root
          if(disc2.lt.0.0)disc2 = err_root
        endif
        a = q(1,1)*(r2 + q1m - q2m)
        b = q(2,1)*sqrt(disc1*disc2)
        rxupper = (a + b)/(2.0*q1m)
        rxlower = (a - b)/(2.0*q1m)
        a1 = q(1,2)*ctheta(3) + q(1,1)*ctheta(2) + q(2,1)*stheta(2)
     1  -((r2 + q1m - q2m)*stheta(2))/(2.0*q(2,1))
        b1 = ctheta(2) - (q(1,1)*stheta(2)/q(2,1))
        r(1) = a1/b1
c test that rx can exist
        if((r(1) - rxupper).gt.err_root.or.
     1        (rxlower - r(1)).gt.err_root)then
          iter = -maxrt
          return
        else
          w = (r2 + q1m - q2m - 2.0*r(1)*q(1,1))/(2.0*q(2,1))
          r(2) = (r(1)*ctheta(1) - t(1)*ctheta(0)
    1            - t(2)*stheta(0))/stheta(1)
            a = -(r(1)*stheta(1) + r(2)*ctheta(1))
            b = t(2)*ctheta(0) - t(1)*stheta(0)
            c = t(3)
c
c initialize TrigSolve
c
            ifail = 1
            call TrigSolve(a, b, c, somega(1), comega(1),
    1          omega(1), sign1, err_root, ifail, trigaccy)
            if(ifail.eq.1)then
              iter = -maxrt
              return
            endif
          endif
        endif
c
c end of initialization, start of solution
c
c now get omega(1)
c
        ifail = 0
        call TrigSolve(a, b, c, somega(1), comega(1), omega(1),
    1       sign1, err_root, ifail, trigaccy)
        sign1 = sign1 - 2.0
        r(3) = stheta(0)*somega(1)*t(1)
    1        - ctheta(0)*somega(1)*t(2) comega(1)*t(3)
c now compute omega(2)
        ifail2 = 0
        call TrigSolve(-w, r(2), r(3), somega(2), comega(2), omega(2),
    1     sign2, err_root, ifail2, trigaccy)
        if(ifail2.eq.1)then
          iter = 0
          return
        else
          sign2 = -sign2
          iter = abs(iter) + 1
          comega(3) = (q(1,2)*ctheta(3)*ctheta(2) + q(1,1) - r(1))/
    1         (q(1,2)*stheta(3)*stheta(2))
          somega(3) = (-r(2)*somega(2) + r(3)*comega(2))/
    1         (q1,2)*stheta(3))
          call TrigNorm(somega(3), comega(3), omega(3),
    1       trigaccy, ifail)
          call FindLastOmegas(5, omega, iter, infoprint)
        endif
        return
        end
```

FINDLASTOMEGAS.F

```
        subroutine FindLastOmegas(n, omega, iter, infoprint)
        include 'commons.h'
        real*4 omega(n), ta(3), temp(3)
c
c completes the solutions; if the equations do not give the correct
c solutions within the limits of the data (2sf), -iter is returned
c get omega(n - 1)
        ta(1) = ctheta(0)*u(1) + stheta(0)*u(2)
        ta(2) = -stheta(0)*u(1) + ctheta(0)*u(2)
        ta(3) = u(3)
        do 10, i = 1, n - 2
```

```
c do inverse dihedral and bond angle rotations
          temp(1) = ta(1)
          temp(2) = comega(i)*ta(2) + somega(i)*ta(3)
          temp(3) = -somega(i)*ta(2) + comega(i)*ta(3)
          ta(1) = ctheta(i)*temp(1) + stheta(i)*temp(2)
          ta(2) = -stheta(i)*temp(1) + ctheta(i)*temp(2)
          ta(3) = temp(3)
   10   continue
          if(abs(ta(1) - ctheta(n - 1)).gt.0.2)then
            if(infoprint.gt.9) write(6,20)ta(1), n - 1, ctheta(n - 1)
          iter = -abs(iter)
          return
   20   format(1x, f14.7,' should be equal to cos(th(',i2,')) = ', f14.7)
        endif
        comega(n - 1) = ta(2)/stheta(n - 1)
        somega(n - 1) = ta(3)/stheta(n - 1)
        call TrigNorm(somega(n - 1), comega(n - 1), omega(n - 1), trigaccy, ifail)
c get omega(n)
        ta(1) = ctheta(0)*v(1) + stheta(0)*v(2)
        ta(2) = -stheta(0)*v(1) + ctheta(0)*v(2)
        ta(3) = v(3)
        do 30, i = 1, n - 1
c do inverse dihedral and bond angle rotations
          temp(1) = ta(1)
          temp(2) = comega(i)*ta(2) + somega(i)*ta(3)
          temp(3) = -somega(i)*ta(2) + comega(i)*ta(3)
          ta(1) = ctheta(i)*temp(1) + stheta(i)*temp(2)
          ta(2) = -stheta(i)*temp(1) + ctheta(i)*temp(2)
          ta(3) = temp(3)
   30   continue
        if(abs(ta(1)).gt.0.2) then
          if(infoprint.gt.9) write (6, 40)ta(1)
   40   format('the following number should be zero', f14.7)
          iter = -abs(iter)
          return
        endif
        comega(n) = ta(2)
        somega(n) = ta(3)
        call TrigNorm(somega(n), comega(n), omega(n), trigaccy, ifail)
        return
        end
```

TRIGSOLVE.F

```
        subroutine TrigSolve(a, b, c, som, com, om, sgn,
    1        rooterr, ifail, normaccy)
c
c to find stable solutions for equations of the type
c a + b.cos(w) + c.sin(w) = 0
c
        save a1, b1, c1, a0, b0, c0, t0, q
        accy = 1.0e-6
        if(ifail.ge.0)then
c
c initialize
c
          ifail = 0
          a0 = a
          b0 = sign(max(abs(b), accy),b)
          c0 = sign(max(abs(c), accy),c)
          t0 = abs(b/c0)
          a1 = b*b + c*c
          b1 = 2.0*a*c
          c1 = a*a - b*b
          disc = b1**2 - 4.0*a1*c1
          if(disc.lt.0.0)then
            if(disc.gt.-rooterr)then
              disc = 0.0
            else
              ifail = 1
            endif
          else
            disc = sqrt(disc)
          endif
          q = -0.5*(b1 + sign(disc, b1))
        endif
c end of init
        if(ifail.gt.0)then
          return
        endif
        if(t0.gt.1.0/accy)then
          ifail = 0
          som = 0.0
          com = sign(1.0, - a/b0)
          om = atan2(som, com)
          return
        endif
        if(t0.lt.accy)then
          ifail = 0
          som = sign(1.0, - a/c0)
          com = 0.0
          om = atan2(som, com)
          return
        endif
        ifail = 0
        if(sgn.lt.0.0)then
          som = q/a1
        else
          som = c1/q
        endif
        com = -(a0 + c0*som)/b0
        call TrigNorm(som, com, om, rooterr, normaccy, ifail)
        return
        end
        subroutine TrigNorm(som, com, om, accy, ifail)
c
c renormalize trig values if necessary and compute angle
c
        ac = som**2 + com**2
        dc = abs(1.0 - ac)
        if(dc.gt.accy)then
c
c normalize
c
          ac = sqrt(ac)
          som = som/ac
          com = com/ac
          ifail = 2
        endif
        om = atan2(som, com)
        return
        end
```