Using distance geomtry to generate structures

David A. Case

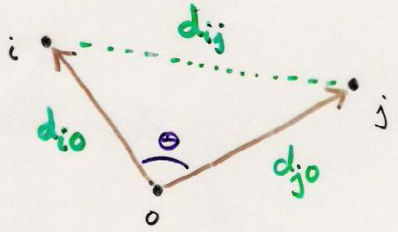
Genomic systems and structures, Spring, 2009

Converting distances to structures

Fundamentals of distance geometry

Define metric matrix $g_{ij} \equiv \underset{\sim}{X}_i \cdot \underset{\sim}{X}_j$

$= d_{io}\, d_{jo} \cos\Theta$



$= \frac{1}{2}\left(d_{io}^2 + d_{jo}^2 - d_{ij}^2\right)$

Theorem: Distances correspond to 3-D object iff. $\underset{\approx}{g}$ is a rank-3 matrix

$$\left[\ \underset{\approx}{g}\ \right] = \left[\ \underset{\approx}{W}^T\ \right]\left[\begin{matrix}\lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \lambda_3 & \\ 0 & & & 0\end{matrix}\right]\left[\ \underset{\approx}{W}\ \right]$$

$$g_{ij} = \sum_{k=1}^{3} w_{ik}\, w_{jk}\, \lambda_k = \sum_{k=1}^{3} X_{ik}\, X_{jk}$$

$$\therefore \quad X_{ik} = \lambda_k^{1/2}\, w_{ik}$$

Metric Matrix Distance Geometry

To describe a molecule in terms of the distances between atoms,there are many constraints on the distances, since for N atoms there are $N(N-1)/2$distances but only $3N$ coordinates. General considerations for the conditions required to "embed" a set of interatomic distances into a realizable three-dimensional object forms the subject of distance geometry. The basic approach starts from the *metric matrix* that contains the scalar products of the vectors $\mathbf{x}_i$ that give the positions of the atoms:

$$g_{ij} \equiv \mathbf{x}_i \cdot \mathbf{x}_j \tag{1}$$

These matrix elements can be expressed in terms of the distances d_{ij}:

$$g_{ij} = 2(d_{i0}^2 + d_{j0}^2 - d_{ij}^2) \tag{2}$$

If the origin ("0") is chosen at the centroid of the atoms, then it can be shown that distances from this point can be computed from the interatomic distances alone. A fundamental theorem of distance geometry states that a set of distances can correspond to a three-dimensional object only if the metric matrix $\mathbf{g}$ is rank three, i.e. if it has three positive and N-3 zero eigenvalues. This may be made plausible by thinking of the eigenanalysis as a principal component analysis: all of the distance properties of the molecule should be describable in terms of three "components," which would be the x, y and z coordinates.

Metric matrix distance geometry (part 2)

If we denote the eigenvector matrix as $\mathbf{w}$ and the eigenvalues λ, the metric matrix can be written in two ways:

$$g_{ij} = \sum_{k=1}^{3} x_{ik} x_{jk} = \sum_{k=1}^{3} w_{ik} w_{jk} \lambda_k \tag{3}$$

The first equality follows from the definition of the metric tensor, Eq. (1); the upper limit of three in the second summation reflects the fact that a rank three matrix has only three non-zero eigenvalues. Eq. (3) then provides an expression for the coordinates $\mathbf{x}_i$ in terms of the eigenvalues and eigenvectors of the metric matrix:

$$x_{ik} = \lambda_k^{1/2} w_{ik} \tag{4}$$

Using imprecise distances

If the input distances are not exact, then in general the metric matrix will have more than three non-zero eigenvalues, but an approximate scheme can be made by using Eq. (4) with the three largest eigenvalues. Since information is lost by discarding the remaining eigenvectors, the resulting distances will not agree with the input distances, but will approximate them in a certain optimal fashion. If one only knows a distance range, then some choice of distance to be used must be made.

Considerable attention has been paid recently to improving the performance of distance geometry by examining the ways in which the bounds are "smoothed" and by which distances are selected between the bounds. Triangle bound inequalities can improve consistency among the bounds, and NAB implements the "random pairwise metrization" algorithm developed by Jay Ponder. Methods like these are important especially for underconstrained problems, where a goal is to generate a reasonably random distribution of acceptable structures, and the difference between individual members of the ensemble may be quite large.

An alternative procedure, which we call "random embedding", implements the procedure of deGroot *et al.* for satisfying distance constraints. This does not use the embedding idea discussed above, but rather randomly corrects individual distances, ignoring all couplings between distances.

Creating and manipulating bounds, embedding structures

```
bounds newbounds(  );
int andbounds( );
int orbounds(  );
int setbounds(  );
int useboundsfrom(  );
int setboundsfromdb( );
int tsmooth( );
int embed( );
```

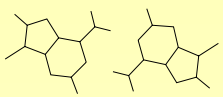
Distance geometry templates

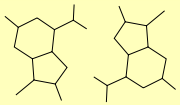
The useboundsfrom() function can be used with structures supplied by the user, or by canonical structures supplied with the nab distribution called "templates". These templates include stacking schemes for all standard residues in a A-DNA, B-DNA, C-DNA, D-DNA, T-DNA, Z-DNA, A-RNA, or A'-RNA stack. Also included are the 28 possible basepairing schemes as described in Saenger.
A typical use of these templates would be to set the bounds between two residues to some percentage of the idealized distance described by the template. In this case, the template would be the reference molecule (the second molecule passed to the function). A typical call might be:

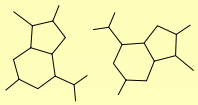
```
useboundsfrom(b, m, "1:2,3:??,H?^'T]", get-
pdb( PATH + "gc.bdna.pdb" ), "::??,H?[^'T]", 0.1 );
```

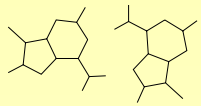
where PATH is $NABHOME/dgdb/stacking/. This call sets the bounds of all the base atoms in residues 2 (GUA) and 3 (CYT) of strand 1 to be within 10% of the distances found in the template.

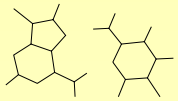
Typical base pair templates

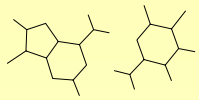

aa.I.pdb

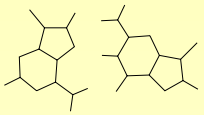

aa.II.pdb

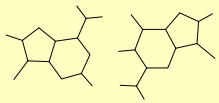

aa.V.pdb

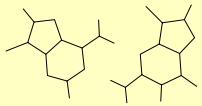

aa.Va.pdb

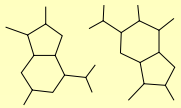

ac.XXV.pdb

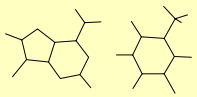

ac.XXVI.pdb

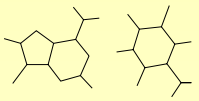

ag.IX.pdb

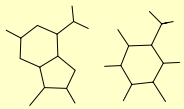

ag.VIII.pdb

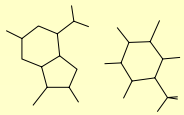

ag.X.pdb

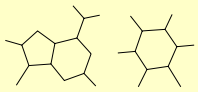

ag.IX.pdb

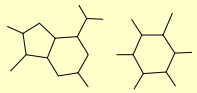

at.XX.pdb
(Watson-Crick)

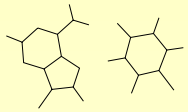

at.XXI.pdb
(Reversed Watson-Crick)

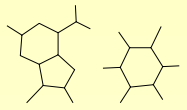

at.XXIII.pdb
(Hoogsteen)

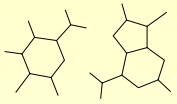

at.XXIV.pdb
(Reversed Hoogsteen)

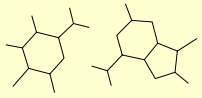

au.XX.pdb
(Watson-Crick)

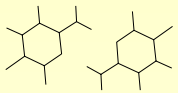

au.XXI.pdb
(Reversed Watson-Crick)

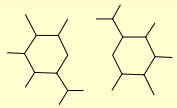

au.XXIII.pdb
(Hoogsteen)

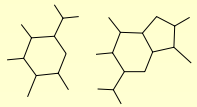

au.XXIV.pdb
(Reversed Hoogsteen)

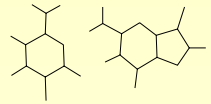

ca.XXV.pdb

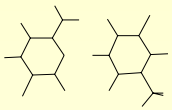

ca.XXVI.pdb

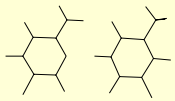

cc.XIV.pdb

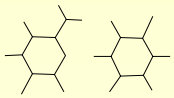

cc.XV.pdb

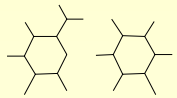

cg.XIX.pdb
(Watson-Crick)

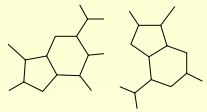

cg.XXII.pdb
(Reversed Watson-Crick)

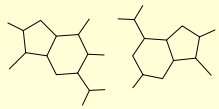

ct.XVII.pdb

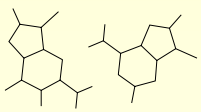

ct.XVIII.pdb

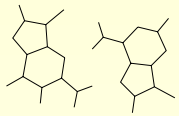

cu.XVII.pdb

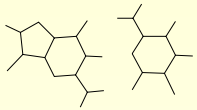

cu.XVIII.pdb

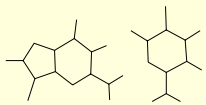

ga.IX.pdb

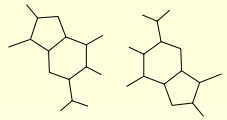

ga.VIII.pdb

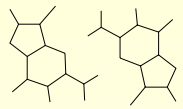

ga.X.pdb


ga.XI.pdb


gc.XIX.pdb
(Watson-Crick)

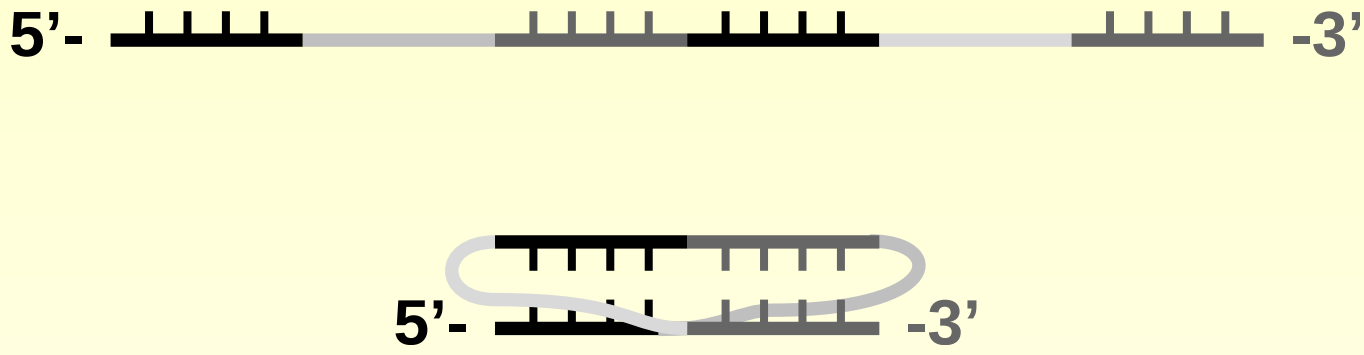

gc.XXII.pdb
(Reversed Watson-Crick)


gg.III.pdb


gg.IV.pdb

Building an RNA pseudoknot

In addition to the standard helix generating functions, nab provides extensive support for generating initial structures from low structural information. As an example, we will describe the construction of a model of an RNA pseudoknot based on a small number of secondary and tertiary structure descriptions. Shen and Tinoco (*J. Mol. Biol.* **247,** 963-978, 1995) used the molecular mechanics program X-PLOR to determine the three dimensional structure of a 34 nucleotide RNA sequence that folds into a pseudoknot. This pseudoknot promotes frame shifting in Mouse Mammary Tumor Virus. A pseudoknot is a single stranded nucleic acid molecule that contains two improperly nested hairpin loops as shown below. NMR distance and angle constraints were converted into a three dimensional structure using a two stage restrained molecular dynamics protocol. Here we show how a three-dimensional model can be constructed using just a few key features derived from the NMR investigation.



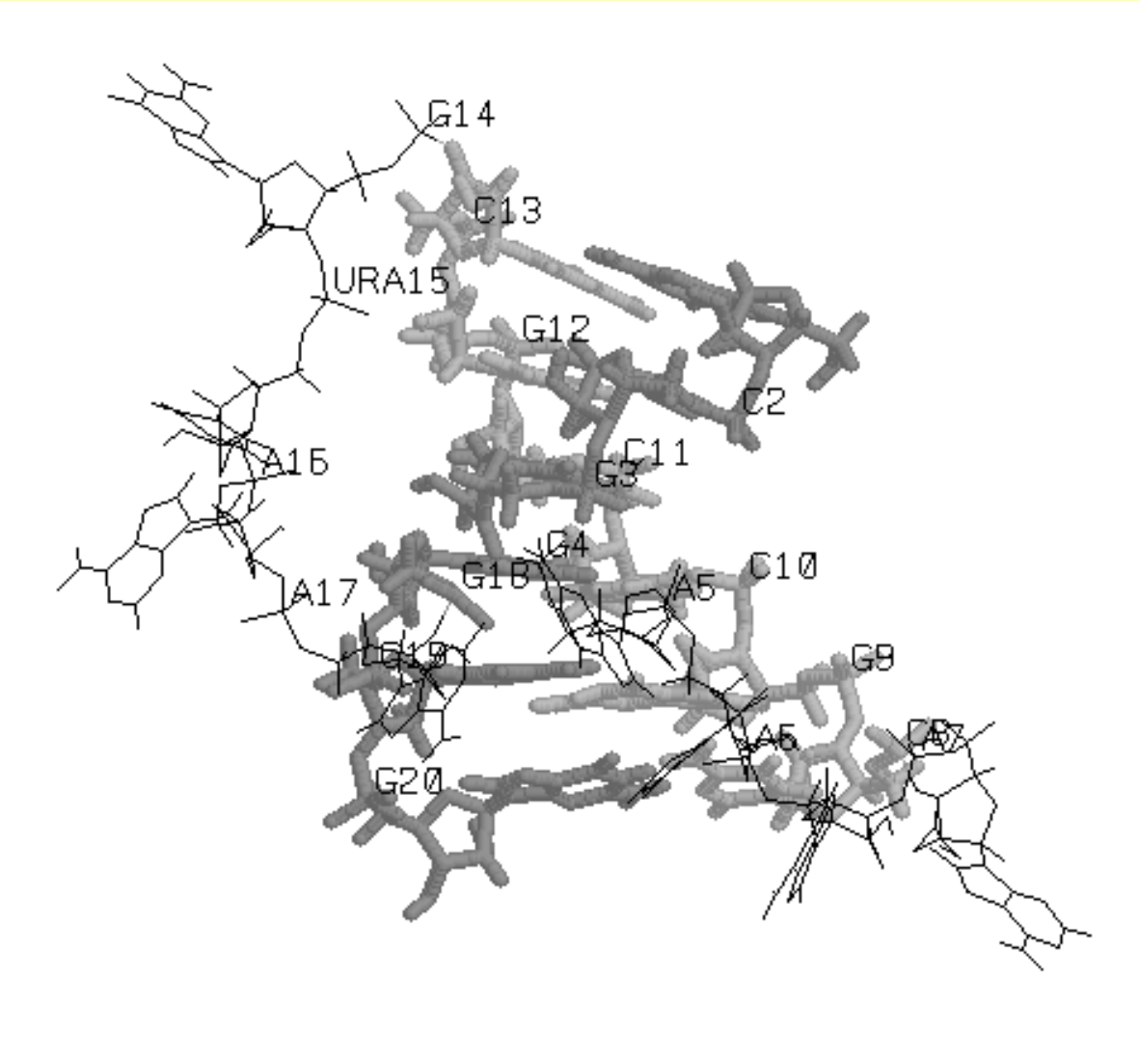
Sample program to create the pseudoknot

```
molecule m;
float xyz[ dynamic ],f[ dynamic ],v[ dynamic ];
bounds b;
int i, seqlen;
float fret;

string seq, opt;
seq = "gcggaaacgccgcguaagcg";
seqlen = length(seq);
m = link_na("1", seq, "rna.amber94.rlb", "rna", "35");
allocate xyz[ 4*m.natoms ];
allocate f[ 4*m.natoms ];
allocate v[ 4*m.natoms ];
b = newbounds(m, "");

for ( i = 1; i <= seqlen; i = i + 1) {
      useboundsfrom(b, m, sprintf("1:%d:??,H?[^'T]", i), m,
            sprintf("1:%d:??,H?[^'T]", i), 0.0 );
}
setboundsfromdb(b, m, "1:1:", "1:2:", "arna.stack.db", 1.0);
setboundsfromdb(b, m, "1:2:", "1:3:", "arna.stack.db", 1.0);
```

Sample program, continued

```
tsmooth(b, 0.0005);
opt = "seed=571, gdist=0, ntpr=50, k4d=2.0, randpair=5.";
dg_options( b, opt );
embed(b, xyz );

for ( i = 3000; i > 2800; i = i - 100 ){
      conjgrad( xyz, 4*m.natoms, fret, db_viol, 0.1, 10., 500 );

      dg_options( b, "ntpr=1000, k4d=0.2" );
      mm_options( "ntpr_md=50, zerov=1, temp0=" +sprintf("%d.",i));
      md( 4*m.natoms, 1000, xyz, f, v, db_viol );

      dg_options( b, "ntpr=1000, k4d=4.0" );
      mm_options( "zerov=0, temp0=0., tautp=0.3" );
      md( 4*m.natoms, 8000, xyz, f, v, db_viol );
}

setmol_from_xyzw( m, NULL, xyz );
putpdb( "pseudoknot.pdb", m );
```

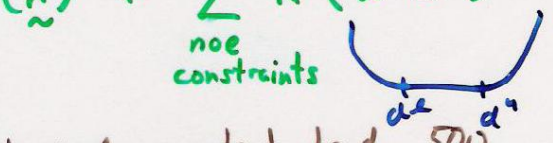
Resulting structure



Molecular dynamics-based structure refinement

Fundamentals of MD refinement

$$E(\underset{\sim}{x}) = E^{MM}(\underset{\sim}{x}) + \sum_{\substack{\text{noe} \\ \text{constraints}}} K\,(d - d^u)^2$$



K for typical covalent bond 500 kcal/mol-Å^2

K_{noe} = ?? values from 1-40 are used.

$$-\frac{\partial E}{\partial \underset{\sim}{x}} \equiv \underset{\sim}{F} = m\ddot{\underset{\approx}{x}}$$

Integrate numerically, $\frac{3}{2}NkT = K.E.$