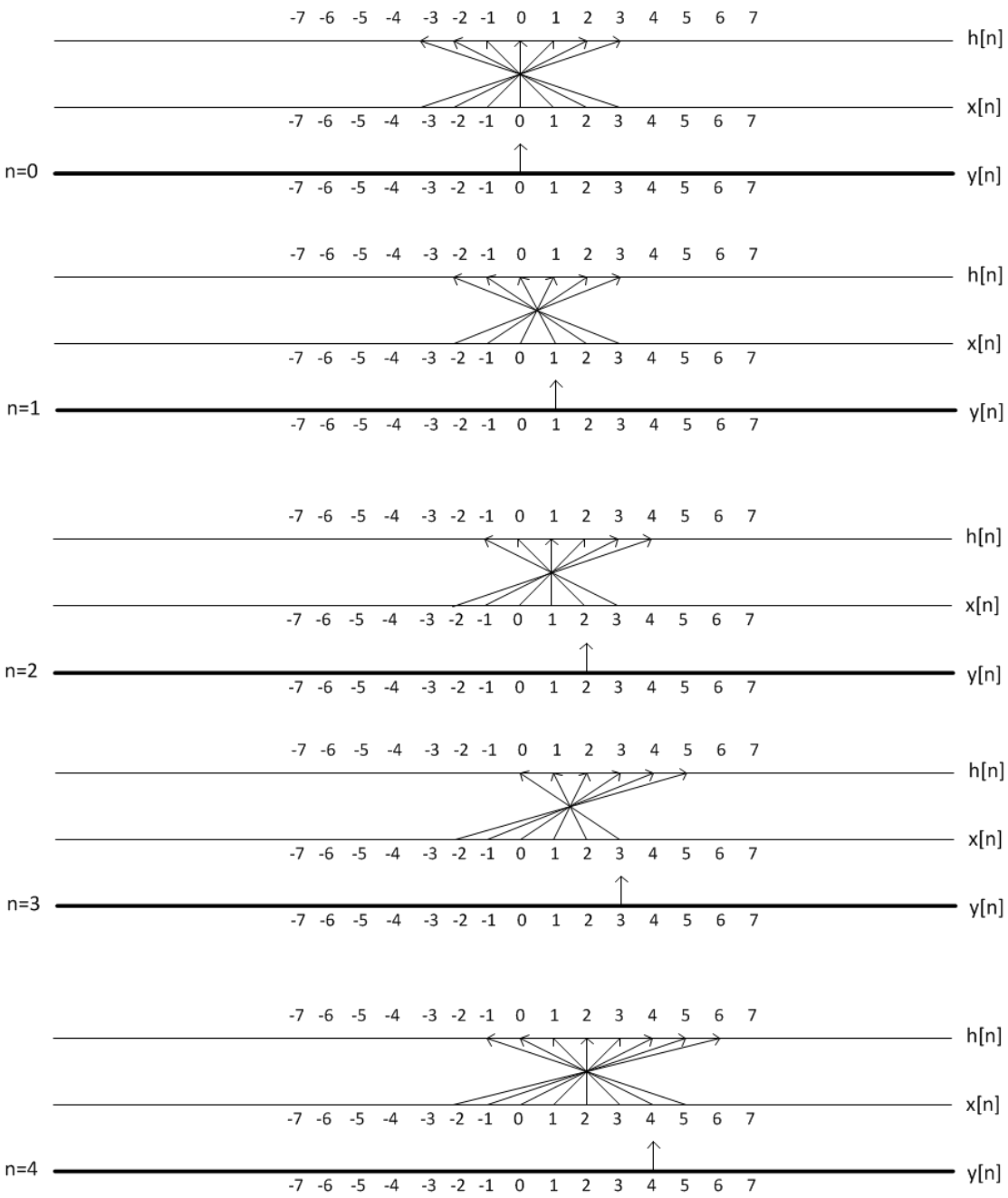
up

Study notes for Math 127.
Mostly collected from the net by Nasser M. Abbasi

These are misc. class notes wrote during taking math 127 at UC Berkeley.

Convolution diagram I did (need to check this)



Graphical representation of the convolution sum. 'y' is the response of the system. 'x' is the input, and 'h' is the response of the system to a unit impulse

$$y[n] = \sum_{k=-\infty}^{k=+\infty} x[k]h[n-k]$$

Nasser Abbasi
oct 4, 2002.
convolution.vsd

From http://mathworld.wolfram.com/LinkingNumber.html

Formally, a link is one or more disjointly embedded circles in three-space. More informally, a link is an assembly of knots with mutual entanglements.

A link invariant defined for a two-component oriented link as the sum of $+1$ crossings and -1 crossing over all crossings between the two links divided by 2. For components α and β,

$$Lk(\alpha, \beta) \equiv \tfrac{1}{2} \sum_{p \in \alpha \sqcap \beta} \epsilon(p),$$

where $\alpha \sqcap \beta$ is the set of crossings of α with β, and $\epsilon(p)$ is the sign of the crossing. The linking number of a splittable two-component link is always 0.

SEE ALSO:

Calugareanu Theorem, Gauss Integral, Jones Polynomial, Link, Twist, Writhe

The twist of a ribbon measures how much it twists around its axis and is defined as the integral of the incremental twist around the ribbon. A formula for the twist is given by

$$\mathrm{Tw}(K) = \frac{1}{2\pi} \int_K ds\, \varepsilon_{\mu\nu\alpha} \frac{dx^\mu}{ds} n^\nu \frac{dn^\alpha}{ds}, \tag{1}$$

where K is parameterized by $x^\mu(s)$ for $0 \le s \le L$ along the length of the knot by parameter s, and the frame K_f associated with K is

$$y^\mu = x^\mu(s) + \varepsilon n^\mu(s), \tag{2}$$

where ε is a small parameter and $n^\mu(s)$ is a unit vector field normal to the curve at s (Kaul 1999).

Letting Lk be the linking number of the two components of a ribbon, Tw be the twist, and Wr be the writhe, then the calugareanu theorem states that

$$\mathrm{Lk}(R) = \mathrm{Tw}(R) + \mathrm{Wr}(R) \tag{3}$$

(Adams 1994, p. 187).

Calugareanu Theorem

SEE ALSO: Letting Lk be the linking number of the two components of a ribbon, Tw be the twist, and Wr be the writhe, then

$$\mathrm{Lk}(K) = \mathrm{Tw}(K) + \mathrm{Wr}(K).$$

(Adams 1994, p. 187).

Gauss Integral, Linking Number, Twist, Writhe

Gauss Integral

Consider two closed oriented space curves $f_1 : C_1 \to \mathbb{R}^3$ and $f_2 : C_2 \to \mathbb{R}^3$, where C_1 and C_2 are distinct circles, f_1 and f_2 are differentiable C^1 functions, and $f_1(C_1)$ and $f_2(C_2)$ are disjoint loci. Let $\mathrm{Lk}(f_1, f_2)$ be the linking number of the two curves, then the Gauss integral is

$$\mathrm{Lk}(f_1, f_2) = \frac{1}{4\pi} \int_{C_1 \times C_2} dS.$$

From the net

Here are some references having to do with DNA and Differential Geometry that mey be of interest here.

DNA and Differential Geometry, William Pohl in Mathematical Intelligencer (ca. 1991)

Pohl, W. F. "Some Integral Formulas for Space Curves and their Generalization",
Amer. J. of Math., 90, 1321-1345 (1968)

Pohl, W. F. "The Self Linking Number of a Closed Space Curve",
J. of Math. and Mech. 17, 975-986 (1968)

White, J. H. "Self Linking and the Gauss Integral in Higher Dimensions",
Amer. J. of Math. 91, 693-728 (1969)

Fuller, F. B. "The Writhing Number of a Space Curve",
Proc. Natl. Acad. Sci USA, 68, 815-819 (1971)

Fuller, F. B. "Decomposition of the Linking Number of a Closed Ribbon:
A Problem from Molecular biology", Proc. Natl. Acad. Sci USA,
75, 3557 (1978)

Some of these are pretty heavy reading, but some are fairly accessible.

-- Jeff Horn

Calculation of link number
The number of times the two strands of DNA double helix are interwound,
i.e., the link number *Lk*, is a topologic invariant quantity for closed DNA

When is a DNA strand considered supercoiled?

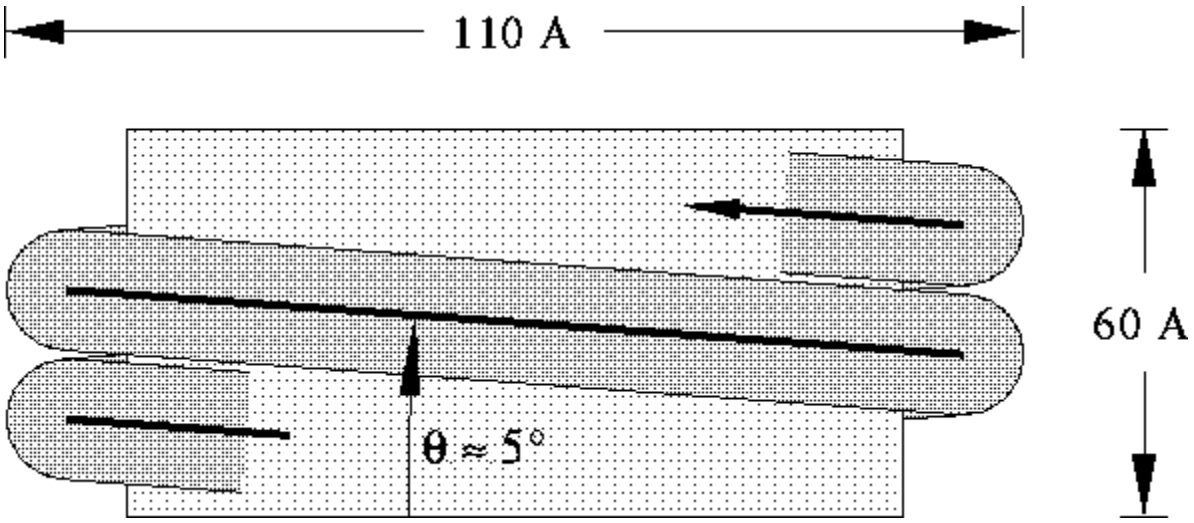
from http://www.scripps.edu/case/nab5/NAB-sh-7.5.html
hints for HW1 problem 4:

Nucleosome Model

While the DNA duplex is locally rather stiff, many DNA molecules are sufficiently long that they can be bent into a wide variety of both open and closed curves. Some examples would be simple closed circles, supercoiled closed circles that have relaxed into circles with twists, and the nucleosome core fragment, where the duplex itself is wound into a short helix.

The overall strategy for wrapping DNA around a curve is to create the curve, find the points on the curve that contain the base pair origins, place the base pairs at these points, oriented so that their helical axes are tangent to the curve, and finally rotate the base pairs so that they have the correct helical twist. In the example below, the simplifying assumption is made that the rise is constant at 3.38 ÅÅ.

The nucleosome core fragment is composed of duplex DNA wound in a left handed helix around a cental protein core. A typical core fragment has about 145 base pairs of duplex DNA forming about 1.75 superhelical turns. Measurements of the overall dimensions of the core fragment indicate that there is very little space between adjacent wraps of the duplex. A side view of a schematic of core particle is shown below.



Computing the points at which to place the base pairs on a helix requires us to spiral an inelastic wire (representing the helical axis of the bent duplex) around a cylinder (representing the protein core). The system is described by four numbers of which only three are independent. They are the number of base pairs *n*, the number of turns its makes

around the protein core *t*, the "winding" angle θ (which controls how quickly the the helix advances along the axis of the core) and the helix radius *r*. Both the the number of base pairs and the number of turns around the core can be measured. The leaves two choices for the third parameter. Since the relationship of the winding angle to the overall particle geometry seems more clear than that of the radius, this code lets the user specify the number of turns, the number of base pairs and the winding angle, then computes the helical radius and the displacement along the helix axis for each base pair:

$$d = 3.38\sin(\theta); \quad \phi = 360t/(n-1)$$
$$r = \frac{3.38(n-1)\cos(\theta)}{2\pi t}$$

where *d* and ϕ are the displacement along and rotation about the protein core axis for each base pair.

These relationships are easily derived. Let the nucleosome core particle be oriented so that its helical axis is along the global Y-axis and the lower cap of the protein core is in the XZ plane. Consider the circle that is the projection of the helical axis of the DNA duplex onto the XZ plane. As the duplex spirals along the core particle it will go around the circle t times, for a total rotation of $360t°$. The duplex contains $n-1$ steps, resulting $360t/(n-1)°$ of rotation between successive base pairs.

```
1   // Program 10.  Create simple nucleosome model.
2   #define PI  3.141593
3   #define RISE    3.38
4   #define TWIST   36.0
5   int         b, nbp; int getbase();
6   float       nt, theta, phi, rad, dy, ttw, len, plen, side;
7   molecule    m, m1;
8   matrix      matdx, matrx, maty, matry, mattw;
9   string      sbase, abase;
10
11  nt = atof( argv[ 2 ] );     // number of turns
12  nbp = atoi( argv[ 3 ] );    // number of base pairs
13  theta = atof( argv[ 4 ] ); // winding angle
14
15  dy = RISE * sin( theta );
16  phi = 360.0 * nt / ( nbp-1 );
17  rad = (( nbp-1 )*RISE*cos( theta ))/( 2*PI*nt );
18
19  matdx = newtransform( rad, 0.0, 0.0, 0.0, 0.0, 0.0 );
20  matrx = newtransform( 0.0, 0.0, 0.0, -theta, 0.0, 0.0 );
21
22  m = newmolecule();
23  addstrand( m, "A" ); addstrand( m, "B" );
24  ttw = 0.0;
25  for( b = 1; b <= nbp; b = b + 1 ){
26      getbase( b, sbase, abase );
27      m1 = wc_helix( sbase, "", "dna", abase, "", "dna",
28          2.25, -4.96, 0.0, 0.0 );
29      mattw = newtransform( 0., 0., 0., 0., 0., ttw );
30      transformmol( mattw, m1, NULL );
31      transformmol( matrx, m1, NULL );
32      transformmol( matdx, m1, NULL );
33      maty = newtransform( 0.,dy*(b-1),0., 0.,-phi*(b-1),0.);
34      transformmol( maty, m1, NULL );
35
36      mergestr( m, "A", "last", m1, "sense", "first" );
37      mergestr( m, "B", "first", m1, "anti", "last" );
38      if( b > 1 ){
39          connectres( m, "A", b - 1, "O3'", b, "P" );
40          connectres( m, "B", 1, "O3'", 2, "P" );
41      }
42      ttw += TWIST; if( ttw >= 360.0 ) ttw -= 360.0;
43  }
44  putpdb( "nuc.pdb", m );
```

Finding the radius of the superhelix is a little tricky. In general a single turn of the helix will not contain an integral number of base pairs. For example, using typical numbers of 1.75 turns and 145 base pairs requires ≈ 82.9 base pairs to make one turn. An approximate solution can be found by considering the ideal superhelix that the DNA duplex is wrapped around. Let L be the arc length of this helix. Then $L\cos(\theta)$ is the arc length of its

projection into the XZ plane. Since this projection is an overwound circle, L is also equal to $2\pi rt$, where t is the number of turns and r is the unknown radius. Now L is not known but is approximately $3.38(n-1)$. Substituting and solving for r gives Eq. (_&rad).

The resulting nab code is shown in Program 2. This code requires three arguments--the number of turns, the number of base pairs and the winding angle. In lines 15-17, the helical rise (dy), twist (phi) and radius (rad) are computed according to the formulas developed above.

Two constant transformation matrices, matdx and matrx are created in lines 19-20. matdx is used to move the newly created base pair along the X-axis to the circle that is the helix's projection onto the XZ plane. matrx is used to rotate the new base pair about the X-axis so it will be tangent to the local helix of spirally wound duplex. The model of the nucleosome will be built in the molecule m which is created and given two strands "A" and "B" in line 23. The variable ttw will hold the total local helical twist for each base pair.

The molecule is created in the loop in lines 25-43. The user specified function getbase() takes the number of the current base pair (b) and returns two strings that specify the actual nucleotides to use at this position. These two strings are converted into a single base pair using the nab builtin wc_helix(). The new base pair is in the XY plane with its origin at the global origin and its helical axis along Z oriented so that the 5'-3' direction is positive.

Each base pair must be rotated about its Z-axis so that when it is added to the global helix it has the correct amount of helical twist with respect to the previous base. This rotation is performed in lines 29-30. Once the base pair has the correct helical twist it must rotated about the X-axis so that its local origin will be tangent to the global helical axes (line 31). The properly-oriented base is next moved into place on the global helix in two stages in lines 32-34. It is first moved along the X-axis (line 32) so it intersects the circle in the XZ plane that is projection of the duplex's helical axis. Then it is simultaneously rotated about and displaced along the global Y-axis to move it to final place in the nucleosome. Since both these movements are with respect to the same axis, they can be combined into a single transformation.

The newly positioned base pair in m1 is added to the growing molecule in m using two calls to the nab buitin mergestr(). Note that since the two strands of a DNA duplex are antiparallel, the base of the "sense" strand of molecule m1 is added *after* the last base of the "A" strand of molecule m and the base of the "anti" strand of molecule m1 is *before* the first base of the "B" strand of molecule m. For all base pairs except the first one, the new base pair must be bonded to its predecessor. Finally, the total twist (ttw) is updated and adjusted to remain in the interval [0,360) in line 42. After all base pairs have been

created, the loop exits, and the molecule is written out. The coordinates are saved in PDB format using the nab builtin putpdb().

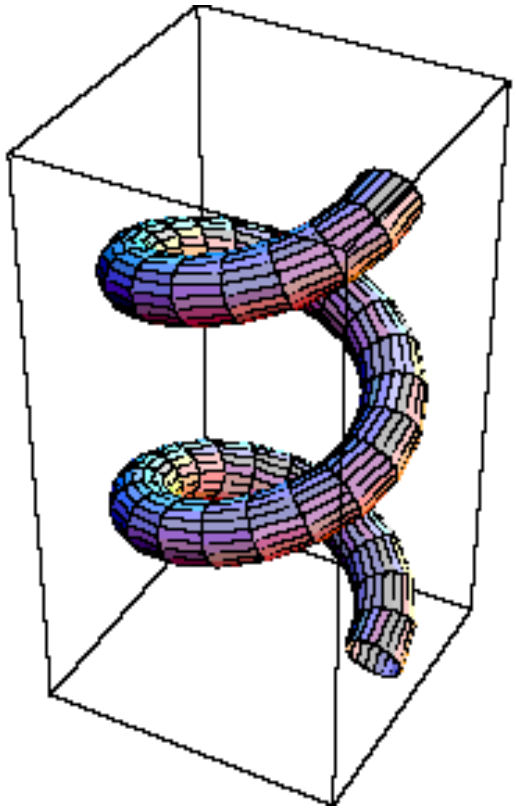
[Contents] [Previous] [Next]
Updated on November 10, 2000. Comments to case@scripps.edu

from www.ma.umist.ac.uk\kd\geomview\geometry.html

curvature---which is the rate of change of angular direction per unit arc length s.

The Fundamental Theorem of Space Curves states that space curves are classified, up to a rotation and translation, by their curvature and torsion. Torsion measures the departure of a curve from a plane.

So, given a starting point and two functions of arc length representing curvature and torsion, the curve can be determined. We know that constant curvature in the plane gives a circle; if we have constant torsion as well then we obtain a circular helix---shown here as a tube:



White's Theorem states that the topological link number Lk for a pair of closed curves is the sum of their twist number Tw and their writhe number Wr:
Lk = Tw + Wr

From http://www.rwgrayprojects.com/Lynn/HelixKnot/helixknot01.html
parmeters of the helix: (length, radius and number of cycles around the cylinder

from http://members.tripod.com/vismath8/malkovsky/Section5.htm
This below shows the equation for the helix in paramoteric form. Same as HW 1, problem 4. replace r by a, and b by h.

Example 7. *The vectors of the trihedra of a helix with a parametric representation*

$\mathbf{x}(s) = (r\cos(\omega s), r\sin(\omega s), h\,\omega s)$ *with* $s \in \mathbf{R}$ *where* $r > 0$, $h \in \mathbf{R}$ *and* $\omega = 1 / (r^2+h^2)^{1/2}$ arc constants.

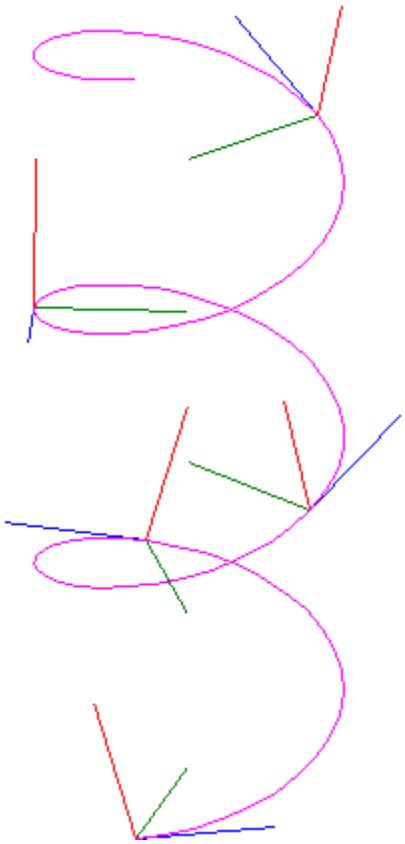


Figure 19. The vectors of trihedra of a helix

From http://www.cco.caltech.edu/~brokawc/Suppl3D/TTW.htm

The change in this local coordinate system as it moves along the curve can be described by the Frenet-Serret equations:

$$\begin{aligned} d\mathbf{t}/ds &= \kappa\,\mathbf{n}, \\ d\mathbf{n}/ds &= -\kappa\,\mathbf{t} + \tau\,\mathbf{b}, \\ d\mathbf{b}/ds &= -\tau\,\mathbf{n}. \end{aligned} \qquad (4)$$

The new scalar variable, τ , is the torsion of the curve. Equation (4) tells us that the two scalar functions $\kappa(s)$ and $\tau(s)$ are sufficient to describe the change in the local coordinate system as it moves along the curve, and therefore are sufficient to describe the shape of the curve.

Below from http://www.math.umd.edu/users/jmr/241/curves2.htm

We are now interested in the derivative of the unit normal with respect to arclength. By differentiating the equation $T \cdot N = 0$ with respect to s, we obtain $\frac{dT}{ds} \cdot N + T \cdot \frac{dN}{ds} = 0$, from which it follows that $\frac{dN}{ds} \cdot T = -\kappa$. Since N is a unit vector, we know that $\frac{dN}{ds}$ is perpendicular to N. The torsion τ is defined to be $\frac{dN}{ds} \cdot B$. Note that since the direction of B is determined independently of $\frac{dN}{ds}$, the torsion, unlike the curvature, is signed. Notice also that for a plane curve, the binormal is identically perpendicular to the plane in which the curve lies and the torsion is 0.Thus we have the Frenet-Serret formulae:

$$\frac{dT}{ds} = \kappa N$$
$$\frac{dN}{ds} = -\kappa T + \tau B$$
$$\frac{dB}{ds} = -\tau N$$

from http://planetmath.org/encyclopedia/Torsion.html

torsion (Definition)

The torsion of a space curve is a complement to the curvature of a curve; curvature measures in the osculating plane what torsion measurs in the normal plane. Thus

$$\tau(t) = \frac{1}{\sigma(t)}$$

where σ is the radius of torsion (analogous to radius of curvature). Because torsion is measured orthogonally to curvature, curves in a plane can have a non-zero curvature, but they will always have zero torsion. An example of a curve with both non-zero torsion and curvature is a helix.
A more analysis friendly way to find torsion is by the equation

$$\tau(t) = -\vec{N}(t) + \vec{B}'(t)$$

where $\vec{N}$ and $\vec{B}$ are the unit normal and binormal vectors, respectively. Since $\vec{N} = \frac{1}{\kappa}\vec{T}' = \rho\vec{T}'$, the above can also be written in terms of the radius of curvature.

From http://math.stanford.edu/courses/math51h/51htext2.pdf

Example. Consider the helix

$$r = (R\cos t, R\sin t, ct),\ t \in [0, \infty).$$

The arc-length function is given by the formula

$$s(t) = \int_0^t \sqrt{R^2 + c^2}du = t\sqrt{R^2 + c^2},$$

and hence in the natural parameterization the helix is defined by the equations

$$\mathbf{r} = (R\cos\frac{s}{\sqrt{R^2 + c^2}}, r\sin\frac{s}{\sqrt{R^2 + c^2}}, c\frac{s}{\sqrt{R^2 + c^2}}),\ s \in [0, \infty).$$

Then

$$\mathbf{r}''(s) = -\frac{R}{r^2 + c^2}((\cos\frac{s}{\sqrt{R^2 + c^2}}, \sin\frac{s}{\sqrt{R^2 + c^2}}, 0)$$

and

$$k(s) = ||\mathbf{r}''(s)|| = \frac{R}{R^2 + c^2}.$$

The vector $\mathbf{b}(s) = \mathbf{T}(s) \times \mathbf{n}(s)$ is called the *binormal*. It is a unit normal vector to the osculating plane $P(s)$. Notice that the derivative $\mathbf{b}'(s)$ is proportional to the principal normal $\mathbf{n}(s)$. Indeed, $\mathbf{b}'(s) \perp \mathbf{b}(s) = 0$ because $\mathbf{b} \cdot \mathbf{b} = 1$ (see the above argument that $\mathbf{T}' \perp \mathbf{T}$ and $\mathbf{b}' \cdot \mathbf{T} = 0$, because $\mathbf{T} \cdot \mathbf{b} = 0$ and hence

$$0 = (\mathbf{T} \cdot \mathbf{b})' = \mathbf{T}' \cdot \mathbf{b} + \mathbf{T} \cdot \mathbf{b}' = k\mathbf{n} \cdot \mathbf{b} + \mathbf{T} \cdot \mathbf{b}' = \mathbf{T} \cdot \mathbf{b}'.$$

The Reidemeister manipulations suffice to completely untie any knot
that is equivalent to the unknot,

From: Mark-Jason Dominus (mjd@op.net)
Subject: Re: Knot Question
Newsgroups: sci.math
Date: 1996/03/19 View: Complete Thread (7 articles) | Original Format

In article <4iif1b$71p@news1.radix.net>, Jim Ward <jfw@radix.net> wrote:
>the normalized bracket polynomial is invariant under the
>Reidermeister moves. Does this imply that if A and the unknot have
>the same polynomial, then A is the unknot?

Alas, no.

Here's what you can do instead:

Project the knot and list the over and undercrossings as before. Name each crossing with a different letter, and write X+ for an overcrossing at X and X- for an undercrossing at X.

You now have a character string that looks like (for example)

```
A+B-C+A-B+C-          (trefoil knot)
(empty)             (unknot)
A+A-                (unknot with a twist in it)
A+B+B-A-              (unknot with two twists in it)
   D+B-A-C+B+D-C-A+         (tangled version of trefoil)
```

These `descriptors' are circular, meaning that they have no particular beginning or end. The following descriptors are all identical:

```
B-C+D-B+A-D+C-A+
C+D-B+A-D+C-A+B-
D-B+A-D+C-A+B-C+
B+A-D+C-A+B-C+D-                    knot)
A-D+C-A+B-C+D-B+                  eight
D+C-A+B-C+D-B+A-              figure-
C-A+B-C+D-B+A-D+            a
A+B-C+D-B+A-D+C-       (it's
```

Also, we shouldn't care whether we read a descriptor forwards or backwards. (Unless for some reason we need to worry about knot chirality, which we don't for this problem, since the unknot is not chiral.)

Now:

1. Reidemeister move I is applicable iff the descriptor contains a sequence of the form `P+P-'. To perform RMI on the knot, just delete the P+P- from the descriptor.

2. Reidemeister move II is applicable iff the descriptor contains sequences of the form `P+Q+' and `P-Q-'. (`P+Q+' and `Q-P-' will also work.) To perform RMII on the knot, just delete the two sequences from the descriptor. Note that if you have P+Q- and P-Q+, you *can't* apply RMII.

3. Reidemeister move III is applicable iff the descriptor
contains sequences of the form `P+Q+', `P-R-', and `Q-R+'.
(One or more of these may be reversed, as in paragraph 2.)
To perform RMIII on the knot, reverse them, replacing the
three sequences with `Q+P+', `R-P-', and `R+Q-',
respectively.

The Reidemeister manipulations suffice to completely untie any knot that is equivalent to the unknot, and therefore, if D is the descriptor of such a knot, there is some sequence of operations 1, 2, and 3 which will reduce D to the empty string.

Here, then, is an algorithm for determining whether or not a given descriptor D represents the unknot:

You need a stack, S, and a list L. L will record the knots that you have analyzed already, so that you don't analyze anything twice. Both start out empty.

Push D onto the stack. Append D to list L.

TOP:

If the stack is empty, halt and report failure. Otherwise...

Pop the top of the stack into X.

If X is the empty string, then D was equivalent to the unknot.
Halt and report success. Otherwise...

If X appears in list L, we have already analyzed it, so goto
TOP without doing anything further. Otherwise...

Append X to list L, so that we don't analyze it again.

If X has the form described in (1) or (2) above, then
Apply the appropriate Reidemeister transformation.
The result is configuration Y.
Push Y onto the stack.
Goto TOP.
Otherwise...

If X has the form described in (3) above, then,
for each such P,Q,R,
apply the indicated transformation to X, yielding Y_{P,Q,R}.

Push Y_{P,Q,R} onto the stack
Goto TOP.
Otherwise...

Goto TOP.

This is a very simple algorithm: It's just a depth-first search of the space of knots which are simpler than the original and reachable from it by a sequence of Reidemeister moves. If you want a BFS instead of a DFS, just replace stasck S with a queue. The worst-case running time is exponential in the number of crossings, but I don't believe anyone knows anything better.

This algorithm works fine for knots with more than one component. You just need to maintain one descriptor for each component.

From http://www.freelearning.com/knots/intro.htm#num1

One invariant is the **minimal crossing number**. The minimal crossing number of a knot is the least number of crossings that appear in any projection of the knot. For example, the unknot has a minimal crossing number of 0. The trefoil knot has a minimal crossing number of 3.

From http://members.tripod.com/vismath5/bor/bor6.htm

For example, by duplicating one ring in Borromean rings, we obtain 4-Borromean link, and continuing in the same manner, *n*-Borromean links (*n*=5,6,7…). Different links of that infinite series follow from other choices of rings that will be duplicated.

We can define an evolutionary distance between two sequences as the number of point mutations that are necessary to evolve one sequence to other. (More specifically, the distance is the minimal number of mutations.)

From http://www.wi.mit.edu/nap/2002/nap_press_02_arachne.html on archne

Various assembly programs have been previously reported, including SEQAID, CAP, PHRAP, TIGR and the Celera assembler, among others, but this is the first of its kind that is publicly available whole genome sequence assembler.

On nucleosomes

From http://www.rcsb.org/pdb/molecules/pdb7_1.html

Each nucleosome is composed of eight "histone" proteins bundled tightly together at the center (shown here in blue), encircled by two loops of DNA (shown here in orange). The histone proteins, however, are not completely globular like most other proteins. They have long tails, which comprise nearly a quarter of their length.

From http://www.web-books.com/MoBio/Free/Ch3D1.htm

A chromosome contains five types of histones: H1 (or H5), H2A, H2B, H3 and H4. H1 and its homologous protein H5 are involved in higher-order structures. The other four types of histones associate with DNA to form nucleosomes. H1 (or H5) has about 220 residues. Other types of histones are smaller, each consisting of 100-150 residues.

See http://www.accessexcellence.org/AB/GG/nucleosome.html for diagram of necleosomes

In fact chromosomal DNA is packaged into a compact structure with the help of specialized proteins called **histones**. The complex DNA plus histones in eucaryotic cells is called **chromatin**.

The fundamental packing unit is known as a **nucleosome.** Each nucleosome is about 11nm in diameter. The DNA double helix wraps around a central core of eight histone protein molecules (an octamer) to form a single nucleosome. A second histone (H1 in the illustration) fastens the DNA to the nucleosome core. The total mass of this complex is about 100,000 daltons.

Nucleosomes are usually packed together, with the aid of a histone (H1,) to form a 30nm large fiber. As a 30nm fiber, the typical human chromosome would be about 0.1cm in length and would span the nucleus 100 times. This suggests higher orders of packaging, to give a chromosome the compact structure seen in a typical karyotype (metaphase) cell.

From http://opbs.okstate.edu/~melcher/MG/MGW1/MG11226.html

At the first level of structure, eight histone molecules are clustered on 146 bp of DNA. These clusters, called **nucleosomes**, are separated from one another by 20 to 100 bp (depending on cell type, organism and physiological status) of spacer DNA.

- Extensive characterization of nucleosomes by neutron diffraction, crystallization, X-ray diffraction, protein crosslinking, immuno electron microscopy and other techniques revealed further properties.
  - The DNA is wrapped left-handed around a 3.2 nm radius core of histones;
  - There are 1.8 turns of DNA per nucleosome.
  - DNase nicking of nucleosomes shows an average periodicity of 10.7 bp in center; 10.0 near ends.
  - The DNA is kinked; kinks are found at +/- 1 and +/- 4.
  - Alpha helical regions of histones have basic residues that contact the major groove of the DNA.
- The folding of DNA in nucleosomes results in a compaction factor of about 7 5. Higher levels of chromatin structure involve histone H1 and non-histone chromosomal proteins.

nucleosome

Repeating units of organization of chromatin fibres in chromosomes, consisting of around 200 base pairs, and two molecules each of the histones H2A, H2B, H3 and H4. Most of the DNA (around 140 base pairs) is believed to be wound around a core formed by the histones, the remainder joins adjacent nucleosomes, thus forming a structure reminiscent of a string of beads.

http://www.average.org/~pruss/Nucleosomes/othersites.html have links on nuclesome.

- nucleosome consists of:
  - core particle
    - histone octamer (2 copies each of H2A, H2B, H3 and H4)
    - 146 bp of DNA wrapped around octamer
  - linker DNA
    - 54 bp of DNA
  - histone H1
    - not found in all "beads" on the string
    - is found in all nucleosomes in the chromatin fibre
    - bound to linker DNA and core particle
    - responsible for higher order folding of chromatin structure

from http://www.lbl.gov/Science-Articles/Research-Review/Magazine/1997-fall/vr/DNA.html

DNA is typically wrapped around a mixture of proteins called "histone" that provide the giant molecule with structural support. Repeating segments of DNA, some 165 to 240 base pairs in length, combine with eight histone molecules to form a distinct unit called a "nucleosome."

Nucleosomes are in turn organized into even larger units called "chromatin."

From http://aesop.rutgers.edu/~molgen/nucleosomes.htm

6. Model was derived partly from the dimensions of the components (Fig 19.5), 6 nm x 11 nm, length of DNA is ~2 x the circumference of the particle 34 nm. Height of the particle is slightly larger than 2 x the diameter of DNA ~ 4 nm.

The nucleosome is ~10 nm diameter, produces the 10 nm fiber (Fig 19.17)

From http://www.mbg.cornell.edu/biobm332/exams/exams97-00/00finalkey.html

1. **L = linking number, it is the number of times one strand winds around the other**
   T = twisting number, it is the number of base pairs in a B form DNA divided by 10.5
   W = writhing number, it is the number of supercoils

1. The topology of a circular B-form DNA molecule has the following values; L = 395, T = 400 and W = -5. How many nucleosomes are there on this DNA molecule? (Assume that each nucleosome generates one negative supercoil.)

   5

check this below. Nice review.

http://www.cmb.uab.edu/courses/Lectures/harvey.pdf

http://www.mindfully.org/GE/DNA.htm

see http://www.mpip-mainz.mpg.de/~heli/chromatin.html for best diagram I saw of nucleosome

from http://intranet.siu.edu/~mbmb/sample%20test2.html

5.Nucleosomes package DNA within the cell by wrapping DNA around the exterior of the protein's cylindrical structure. About 200 base pairs of DNA is wrapped around the nucleosome in a left-handed helical fashion and makes 1.7 wraps per nucleosome. Starting with a relaxed 2 kilobase pair closed circular DNA, determine what the linking number, writhe and twist are after 10 nucleosomes are bound per DNA molecule. The relaxed plasmid is B-DNA with 10 base pairs per helical turn (10 points).

From http://web.uvic.ca/sciweb/Courses/B300/DNA%20supercoiling.html

DNA supercoiling - an outline

The strands of topologically closed DNA, for example, covalently closed circular plasmid DNA, or DNA that loops out of the scaffolding structure in eukaryotic cells (cf. Figure 24-28), cannot change the number of times they cross each other: this is called their linking number
If the twist (the basic double helical turns of the two strands about each other) of the double helix in topologically closed DNA changes, there is supercoiling ("writhing"; the coiled-coil) to keep the linking number constant

Most cellular DNA is underwound (917-918)

A sample of circular DNA isolated from a cell, e.g. plasmid DNA, if constrained to lie flat, would have fewer twists of one strand about the other than B DNA does - this is caused by intracellular conditions that favour partial unwinding (for example, parts of the DNA that are being transcribed into RNA are unwound)
This is called underwinding, and is characteristic of natural DNAs

When underwound DNA is isolated, it tends to wind into a normal B DNA helix (10.5 bases per turn); in doing so, it generates negative supercoils because the linking number cannot change (Figure 24-12)
The partially unwound state, and the fully wound-up, supercoiled state, are equivalent in linking number (Figure 24-13)

DNA underwinding is defined by topological linking number (918-921)
Topologically closed DNA is characterised by its linking number, Lk, which is an integer and which cannot change as long as no covalent bonds are broken

Lk is related to the twist (Tw) and the writhe (Wr) numbers by

$$Lk = Tw + Wr$$

Tw is about the same as the basic, B DNA winding number (10.5 bp per turn in relaxed DNA), but may differ from this due to topological stress

Writhe is the supercoiling number, the coiling of the DNA coil

Tw and Wr need not be integers

If Tw changes, then Wr must change correspondingly, to keep Lk constant

For relaxed, closed circular B DNA (no supercoils), Lk and Tw are equal to the number of base pairs divided by 10.5, and Wr is zero

The superhelical density σ is defined as the number of turns that have been added or subtracted in the supercoiled DNA, compared to the relaxed state, divided by the total number of turns in the DNA if it were relaxed (normally bp/10.5)

Typically, σ is between -.05 and -.07 (5-7% underwinding) in isolated natural DNA

Supercoils in isolated plasmid DNA are in the form shown in Figure 24-16; this is called the plectonemic form of supercoiled DNA

Right-handed, plectonemic DNA has negative supercoils

Underwound DNA facilitates the formation of cruciforms (Figure 24-18), because both have unpaired bases

Topoisomerases catalyze changes in the linking number of DNA (921-922)

Topoisomerases allow the linking number to change

Type I topoisomerases cause a transient break in one strand of duplex DNA, allow the open end to rotate around the other strand, and then re-seal the broken strand, to change the linking number by 1

Type II topoisomerases change the value of Lk by 2, by creating a double-stranded break and allowing *two* strands to pass through before re-sealing the break (Figure 24-20)

An unusual example of a Type II topoisomerase is E. coli gyrase, which can add negative supercoils to an already negatively supercoiled DNA, using the energy of ATP hydrolysis to drive the reaction; eukaryotic cells do not have gyrase activity, but they do have Type II topoisomerases which can relax supercoiled DNA

Eukaryotic topoisomerases of Type II can relax positive and negative supercoils, as can eukaryotic topos of Type I

Plasmid DNAs differing in numbers of supercoils have different degrees of compactness, and will have different mobilities on gel electrophoresis, as shown for isolated plasmid treated with Type I Topoisomerase (Figure 24-19)

<u>DNA compaction requires a special form of supercoiling (922-923)</u>

The plectonemic form of supercoiling causes only a moderate compaction of DNA

The alternative, solenoidal, form is more compact (Figure 24-22)

The solenoidal form exists in natural eukaryotic DNA, which is wound around histone cores in the nucleosomes in a left-handed (negative) supercoil (Figure 24-23)

This negative supercoiling of the DNA as the nucleosome forms introduces positive supercoils in the segments of DNA linking nucleosomes, and these must be relaxed by topoisomerases

From
http://216.239.33.100/search?q=cache:VtcfbuDe32YC:www.uovs.ac.za/faculties/nat/mkboc/biochem/Super.htm+twist+nucleosomes&hl=en&ie=UTF-8

Supercoiling

Introduction

Supercoiling simply means coiling of a coil. Supercoiling and topology, although perhaps at first glance abstract mathematical concepts, have very relevant application in molecular biology. The DNA molecule is subject to topological constraints, and these have very real effects on the function of DNA. Negative supercoiling can stabilise secondary DNA structures such as hairpin loops, cruciforms, and also facilitate the formation of a melted region in the transition of a transcriptional pre-initiation complex (PIC) to a elongating complex. Also, the DNA in both pro- and eukaryotes are naturally negatively supercoiled. In prokaryotes this is due to the action of gyrases (these are enzymes like topoisomerases, but induce supercoiling in an ATP-dependent manner). In eukaryotes, the packaging of DNA into chromatin causes the DNA template (after removal of proteins) to be supercoiled. In addition, the passage of the RNA polymerase along the DNA molecule generates a twin supercoiled domain. The region behind the polymerase is negatively supercoiled, and the region in front of the polymerase is positively supercoiled. This superhelical stess is normally relaxed by topisomerases. In yeast, there are two types: topoisomerase I (topo I) and topoisomerase II (topo II). Topo I induces a single strand nick, and will relax the DNA molecule in units of 1. Topo II, predictably, cuts both strands, and changes the superhelicity by units of 2. These enzymes appear to be required for normal DNA function, and are involved in the relaxing of superhelical stress that accumulates during transcription and replication of DNA. Thus, it is clear that an understanding of many of the processes that can influence DNA function, requires some understanding of supercoiling.

Classic Linking Theory (CLT).

The essential concept that is used in a theoretical study of supercoiling is the ribbon. The ribbon has two sides (which can represent the phosphodiester backbones of the DNA duplex), and it has an axis, equidistant from the ribbon edges, equivalent to the helix axis. There are three parameters that are important when considering supercoiling: the linking number (L_k), the **twist** (T_w) and the wrythe (W_r). The L_k and T_w is a function of the edge of the ribbon, and has no meaning for a one-dimensional line, such as an axis. The W_r, on the other hand, is a function of the ribbon (or helix) axis, and describes the shape of the axis in space. These three parameters are related by the function

$$L_k = T_w + W_r \quad 1$$

This function simply states that the L_k of a molecule is the sum of the T_w and W_r parameters, and that if the L_k of a molecule is kept constant, but the shape of the axis (the W_r) is changed, the T_w must change by an equal amount of opposite sign.
The linking number, roughly speaking, is the number of times the one DNA strand (or ribbon edge) crosses the other in space. This is a topological property, since the smooth deformation of the molecule does not change the linking number. In this sense a doughnut and a coffee cup is topologically equivalent: both has a single hole, and, were it made from soft clay, the one can be deformed into the other without breaking the clay or introducing additional holes into it.
The linking number of a closed (the ends of the ribbon or DNA molecule meet, forming a circle) can be calculated by inspection by viewing the entire ribbon from any orientation at an infinite distance (watch those taxi fares). If one concentrates on only one edge, and count the number of times this edge crosses in front of the other, one is, in effect, calculating the linking number. Note that the linking number has a sign. If the edge one is inspecting crosses the other in a right-handed (clockwise) manner, the sign is positive. Thus, in viewing the entire ribbon, if the one edge crossed in the one direction and then crosses back in the other direction, the net L_k is zero, since the one crossing contributes +1, and the reverse crossing

Figure 1. The linking number of a DNA molecule. The schematic shows a small circular DNA molecule where the one stand crosses the other a total of 6 times. The L_k of this molecule is thus 6. Since the helix is right-handed, the sign of the linking number is positive, and $L_k = +6$.
On a plane, the **twist** is defines as 1 if the ribbon rates about the ribbon axis by 360º. Thus, a flat ribbon that is bent in a plane, does not have **twist**. The T_w is not a topological property, since the deformation of the ribbon (or DNA molecule) in space can change the **twist**. When the ribbon is wound flat onto a cylinder, the **twist** of the ribbon is given by the equation:
$T_w = N\sin\alpha$ 2
Where N is the number of times (in units of radians, i.e. one rotation is 2Π) that the ribbon revolves around the cylinder, and α is the pitch angle of the helix. Note that T_w is normalised to 2Π . When the winding of the ribbon onto a cylinder is very shallow (i.e., every gyre tends to a circle) the **twist** approaches 0 [$2\Pi \times \sin(0)/2\Pi = 1\times 0 = 0$].
The wrythe is not an easy parameter to calculate. Generally, this parameter can be arrived at by knowing the L_k and T_w of a molecule

Virtual surface linking theory (VSLT)

Despite its impressive name, virtual surface linking theory provides a more rigorous and more easily understood theoretical frame within which to calculate supercoiling.
The L_k in VSLT is calculated relative to a surface, where the orientation of the surface is defined as shown in Figure 2.

A normal vector to the surface is defined, with the direction of the vector deduced by the "right-hand rule". Thus, in A, the direction of the vector is up. The direction of this vector becomes important when calculating the sign of the L_k. The L_k itself is defined as the number of times the helix axis crosses the spanning surface of the ribbon. The sign of L_k is positive if the axis direction (arbitrarily defined) crosses the spanning surface in the same direction as the normal vector. The L_k is negative in the inverse situation, as shown in Figure 3.

Figure 3. Calculation of the L_k and the association sign.

One can also calculate L_k in VSLT by looking at the whole structure from infinite distance, or looking at the shadow that is produces by illuminating with a light from infinite distance. In this case, if the strand on top is rotated in a clockwise direction by less than 180º to point in the same direction as the helix axis, the crossing (node) is assigned a negative value. If the rotation is anti-clockwise, the node is positive. Each such node contributes ½ L_k unit. Note the similarity of this calculation with that given for CLT, above. The determination of the sign of the node is illustrated in Figure 4.

Figure 4. Calculation of the sign of a crossing node

In VSLT the W_r is defined as the number of times that the helix axis crosses itself in a plane projection, with the sign of the node calculated as shown in Fig. 3. However, careful consideration will reveal that the number of times that the axis crosses itself is dependent on the specific orientation of the projection. Thus, the W_r is the average of the sum of the individual W_rp over all possible projections. Thus, W_r is not necessarily an integer. It also follows directly that $W_r = 0$ for any molecule constrained to a plane, irrespective of the complexity of the helix axis in the plane. This is so because there is no possible orientation in which the axis will cross itself in any projection.
It is obvious that W_r is not a quantity that can be easily calculated for complex shapes. However, again, W_r can be arrived at by measuring other more tenable parameter such as L_k which is a function of W_r.

Twist

The definition of **twist** in VSLT is mathematically complex to calculate, but relatively simple to understand. **Twist** is not simply the number of times the one DNA strand crosses the helix axis, since the helix axis itself can contribute to T_w. The definition is as follows: A plane, perpendicular to the helix axis, tansects the helix at a given point **A**. From this point **a**, a vector runs to a point **c** of the one DNA strand, represented by curve **C**. As the plane is moved forward by an infinitesimal amount, the vector **ac** will rotate since the DNA strand revolves around **A**. If the helix axis **A** is planar, the T_w is simply the number of times that **ac** rotates about **A**. However, when **A** is not planar, the orientation of the coordinate system xyz, defined at a point $\mathbf{a_0}$, must be taken into consideration. This Cartesian coordinate system xyz is defined so that the z axis coincides with the infinitesimal length of **A**. As the plane is now moved along the axis **A**, the component (or projection) of **ac** is the important quantity. The **twist** is the sum of the xy component angles **ac** describes in the coordinate system of $\mathbf{a_0}$ as it is moved along the entire length of **A**.

Figure 5. Definition of T_w. A is the helix axis, **C** is the phosphodiester backbone, and **ac** is a vector connecting **A** to **C** at points **a** and **c**. T_w is the sum of the xy component of the rotation of **ac** as the transecting plane, which stays perpendicular to **A**, is moved through the DNA circle.

Figure 6. Definition of the virtual surface. The surface coincides with **A**, and has a vector **v** perpendicular to the surface at point **a**.

This definition of T_w can be further refined by using a [virtual] surface as reference. Although this may seem unnecessarily complex, it actually provides a very physical framework within which to analyze real DNA molecules. Take the framework described above, and place the axis **A** on a surface (that may be curved). Another vector **v** is now defined that is perpendicular to this plane, and in the plane of the original **ac** vector. As this plane is moved along **A**, the winding number (Φ) is defined as the number of times that **ac** rotates past **v**. Since we are considering a closed circular molecule (L_k has no physical meaning in linear molecules), Φ must necessarily be an integer. The helical period h of the DNA is simply the length of the DNA divided by Φ ($h = N/\Phi$).

The helical repeat of DNA is often measured by adsorbing the molecule to a surface, and examining the number of nuclease cleavage sites over the length of the molecule, where it is assumed that maximal cleavage will occur where the DNA molecule is farthest from the surface. The relation between the helical period so measured and Φ is now immediately obvious.

What is the relation between Φ and T_w?

Although Φ is a component of T_w, Φ certainly does not equal T_w. An additional parameter, surface **twist** (ST_w), which measures the way in which the reference vector **v** changes, and therefore the virtual surface on which the axis **A** lies, is required to fully define T**w**. This is most clearly seen when considering an observer that walks along **A** on the surface. This observer can count how many times the vector **ac** crosses his or her field of vision. This is simply Φ . However, the observer will not be able to tell whether he or she has rotated around **A** without reference to something else. This something else is a displacement curve. This displacement curve is simply the trace of the observer's head as he or she walks along the surface. This displacement curves measures the number of times (or fraction thereof) that the observer rotated about **A**. Forward and backward movements to not contribute to this curve. For instance, if the axis **A** lies on the equator of a sphere, the observer can walk around the sphere on the equator, yet not rotate around the axis **A**. Thus, in this case, ST_w is 0. In physical terms, ST_w measures the shape of the DNA molecule in space.

This immediately takes us to the surface linking number SL_k which measures the linking of **A** with the displacement curve.

$SL_k = ST_w + W_r$. 3

SL_k for a helix formed on a sphere (interwound or plectonemic supercoiling) is 0. This is because the displacement curve is not linked to **A** at all: it can be the central axis of the spheroid. Thus, in this case $ST_w = -Wr$. For toroidal supercoiling (the type found in chromatin), SL_k is simply the number of toroidal turns, defined in the same way as for the Φ .

By combining equations 1 and 3, we arrive at

$L_k = ST_w + \Phi + W_r$ 4

$= SL_k + \Phi$ 5

Thus, the L_k of a molecule can be determined from Φ which can be measures as the helical period relative to a surface, and SL_k, which can be calculated from the shape of the molecule.
For any closed DNA on a spehroid, $SL_k = 0$ and therefore $Lk = \Phi$. For toroidal DNA, SL_k is simply the number of times that the helix winds about the super helix axis (n). Usually, a DNA that is totally relaxed (for instance nicked and then religated) is assigned a linking number N/Φ , since the relaxed molecule will be nearly planar and have little contributions from ST_w and W_r. The linking number of such a molecule is often written as L_{k0}. It is possible to measure the difference in linking number between this molecule and another in an agarose gel, since the degree of supercoiling compacts the DNA, and its migration through the gel matrix is different. Individual topoisomers can be so resolved, and the ΔL_k of a specific topoisomer calculated by counting the number of topoisomers between the L_{k0} species, and the band (topoisomer) of interest.
A length independent number, the specific linking difference (σ), is calculated as
$\sigma = \Delta L_k/L_{k0}$ 6
For bacterial plasmids, this is usually –0.06.

Nucleosomes and the linking number paradox

When viewing the structure of the nucleosome, it is seen that the helix winds almost two times around the histone octamer over a length of 168bp. Yet, when the linking difference of a nucleosome is measured, it is found to be ~-1.1. This became known as the *linking number paradox*. Armed with our understanding of supercoiling theory, we can now investigate why this may be so.
In **nucleosomes**, we have ~81bp per superhelical turn. Say we have a 4600bp circular molecule containing nucleosome cores, we will have 57 left-handed superhelical turns. The change in the winding number Φ for one supercoil in going from free DNA (which has a helical period, h, of ~10.5) to nucleosomal DNA (where h = 10.0) is

$\Delta \Phi = N/h_{nuc} – (N/h_0)$
$=(81/10.0) – (81/10.5)$
$= 0.39$

Thus, for 57 supercoils $\Delta \Phi = 22$.
Now,

$\Delta L_k = SL_k + \Delta \Phi$
$= -57 + 22$
$= -35$

Thus
$\Delta Lk = -0.61n$
or, the linking number of the nucleosome is $0.61 \times -2 = -1.2$

Applications

One dimensional resolution of topoisomers

This is good below

http://wine1.sb.fsu.edu/bch4053/Lecture21/Lecture21.htm

from http://searchlauncher.bcm.tmc.edu/help/AlignmentScore.html

The **alignment score** is the sum of the scores specified for each of the aligned pairs of letters, and letters with nulls, in the alignment. The higher the alignment score, the better the alignment.

---

http://www.cbil.upenn.edu/genlang/papers/fft.html

Cheever, E.A., Overton, G.C., and Searls, D.B. (1991) "Fast Fourier Transform-Based Correlation of DNA Sequences using Complex Plane Encoding" Computer Applications in the Biosciences 7(2):143-159.

The detection of similarities between DNA sequences can be accomplished using the signal processing technique of cross correlation. An early method used the Fast Fourier Transform (FFT) to perform correlations on DNA sequences in O(n log n) time for any length sequence [Felsenstein et al, 1982]. However, this method requires many FFTs (nine), runs no faster if one sequence is much shorter than the other, and measures only global similarity, so that significant short local matches may be missed. We report that, through the use of alternative encodings of the DNA sequence in the complex plane, the number of FFTs performed can be traded off against (1) signal-to-noise ratio, and (2) a certain degree of filtering for local similarity via k-tuple correlation. Also, when comparing probe sequences against much longer targets, the algorithm can be sped up by decomposing the target and performing multiple small FFTs in an overlap-save arrangement. Finally, by decomposing the probe sequence as well, the detection of local similarities can be further enhanced. With current advances in extremely fast hardware implementations of signal processing operations, this approach may prove more practical than heretofore.

From
http://hades.ph.tn.tudelft.nl/Internal/PHServices/Documentation/MathWorld/math/math/e/e350.htm

Euler Graph

A Graph containing an Eulerian Circuit. An undirected Graph is Eulerian Iff every Vertex has Even Degree. A Directed Graph is Eulerian Iff every Vertex has equal Indegree and

Outdegree. A planar Bipartite Graph is Dual to a planar Euler graph and vice versa. The number of Euler graphs with n nodes are 1, 1, 2, 3, 7, 16, 54, 243, ... (Sloane's A002854).

Eulerian Circuit

An Eulerian Trail which starts and ends at the same Vertex. In other words, it is a Cycle which uses each Edge exactly once. The term Eulerian Cycle is also used synonymously with Eulerian circuit. For technical reasons, Eulerian circuits are easier to study mathematically than are Hamiltonian Circuits. As a generalization of the Königsberg Bridge Problem, Euler showed (without proof) that a Connected Graph has an Eulerian circuit Iff it has no Vertices of Odd Degree.

Eulerian Trail

A Walk on the Edges of a Graph which uses each Edge exactly once. A Connected Graph has an Eulerian trail Iff it has at most two Vertices of Odd Degree.

From http://www.utc.edu/~cpmawata/petersen/lesson12.htm

An *Euler circuit* on a graph *G* is a circuit that visits each vertex of *G* and uses every edge of *G*.

Vertex Degree

The degree of a Vertex of a Graph is the number of Edges which touch the Vertex, also called the Local Degree. The Vertex degree of a point A in a Graph, denoted $\rho(A)$, satisfies

$$\sum_{i=1}^{n} \rho(A_i) = 2E,$$

where E is the total number of Edges. Directed Graphs have two types of degrees, known as the Indegree and the Outdegree.

Indegree

The number of inward directed Edges from a given Vertex in a Directed Graph.

Outdegree

The number of outward directed Edges from a given Vertex in a Directed Graph.

NP-Complete Problem

A problem which is both NP (solvable in nondeterministic Polynomial time) and NP-Hard (can be translated into any other NP-Problem). Examples of NP-hard problems include the Hamiltonian Cycle and Traveling Salesman Problems.

In a landmark paper, Karp (1972) showed that 21 intractable combinatorial computational problems are all NP-complete.
See also Hamiltonian Cycle, NP-Hard Problem, NP-Problem, P-Problem, Traveling Salesman Problem

NP-Problem

A problem is assigned to the NP (nondeterministic Polynomial time) class if it is solvable in polynomial time by a nondeterministic Turing Machine. (A nondeterministic Turing Machine is a ``parallel'' Turing Machine which can take many computational paths simultaneously, with the restriction that the parallel Turing machines cannot communicate.) A P-Problem (whose solution time is bounded by a polynomial) is always also NP. If a solution to an NP problem is known, it can be reduced to a single P (Polynomial time) verification.

Linear Programming, long known to be NP and thought *not* to be P, was shown to be P by L. Khachian in 1979. It is not known if all apparently NP problems are actually P.

A problem is said to be NP-Hard if an Algorithm for solving it can be translated into one for solving any other NP-problem problem. It is much easier to show that a problem is NP than to show that it is NP-Hard. A problem which is both NP and NP-Hard is called an NP-Complete Problem.
See also Complexity Theory, NP-Complete Problem, NP-Hard Problem, P-Problem, Turing Machine

P-Problem

A problem is assigned to the P (Polynomial time) class if the number of steps is bounded by a Polynomial.
See also Complexity Theory, NP-Complete Problem, NP-Hard Problem, NP-Problem

Complexity Theory

The theory of classifying problems based on how difficult they are to solve. A problem is assigned to the P-Problem (Polynomial time) class if the number of steps needed to solve it is bounded by some Power of the problem's size. A problem is assigned to the NP-Problem (nondeterministic Polynomial time) class if it permits a nondeterministic solution and the number of steps of the solution is bounded by some power of the problem's size. The class of P-Problems is a subset of the class of NP-Problems, but there also exist problems which are not NP.

However, if a solution is known to an NP-Problem, it can be reduced to a single period verification. A problem is NP-Complete if an Algorithm for solving it can be translated into one for solving any other NP-Problem. Examples of NP-Complete Problems include the Hamiltonian Cycle and Traveling Salesman Problems. Linear Programming, thought to be an NP-Problem, was shown to actually be a P-Problem by L. Khachian in 1979. It is not known if all apparently NP-Problems are actually P-Problems.
See also Bit Complexity, NP-Complete Problem, NP-Problem, P-Problem

Math 455.1 • 1 April 2002

Algorithm to Find Euler trail in Eulerian graph
Input
A connected graph G in which each vertex has even degree.
Output
An Eulerian trail in G.
Begin
(0) Remove all loops from G.
(1) (1.1) Select an arbitrary vertex v_0 of G;
(1.2) form some cycle C in G from v_0 to v_0 ¶use Cycle Lemma method◊;

and
(1.3) remove all edges in C, leaving a subgraph H of G.
(2) While H has edges . . .
(2.1) select some vertex v of H that is an end of some edge of C;
(2.2) select some cycle S in H from v to v;
(2.3) let C be the new walk obtained by inserting S into C at vertex v ¶so that C is a trail in G◊;and

(2.4) let H be the subgraph of G obtained by removing from H all edges of S and all the isolated (that is, degree 0) vertices of H.
(3) At each vertex w of C, insert all loops of G at w.
(4) Return C.
End

Theorem 2. (Fleury's algorithm)
Let *G* be an Eulerian graph. Then the following construction is always possible, and produces an Eulerian trail of *G*.
Start at any vertex *u* and traverse the edges arbitrarily, except subject to 2 rules:
(a) erase the edges as they are traversed, and the isolated vertices resulted (if any);
(b) use a bridge only if there is no alternative.

From http://www2.toki.or.id/book/AlgDesignManual/BOOK/BOOK4/NODE165.HTM

There are well-known conditions for determining whether a graph contains an Eulerian cycle, or path:

- An *undirected* graph contains an Eulerian *cycle* iff (1) it is connected and (2) each vertex is of even degree.
- An *undirected* graph contains an Eulerian *path* iff (1) it is connected and (2) all but two vertices are of even degree. These two vertices will be the start and end points of the path.
- A *directed* graph contains an Eulerian *cycle* iff (1) it is connected and (2) each vertex has the same in-degree as out-degree.
- Finally, a *directed* graph contains an Eulerian *path* iff (1) it is connected and (2) all but two vertices have the same in-degree as out-degree, and these two vertices have their in-degree and out-degree differ by one.

From http://www.ctl.ua.edu/math103/euler/howcanwe.htm

How can we tell if a graph has an Euler path or circuit?

In solving the Königsberg bridge problem, Euler proved three theorems which answer this question.

EULER'S THEOREMS

Euler's Theorem 1
If a graph has any vertices of odd degree, then it CANNOT have an EULER CIRCUIT.
AND
If a graph is connected and every vertex has even degree, then it has

AT LEAST ONE EULER CIRCUIT (usually more).

Euler's Theorem 2
If a graph has more than 2 vertices of odd degree, then it CANNOT have an EULER PATH.
AND
If a graph is connected and has exactly 2 vertices of odd degree, then it has AT LEAST ONE EULER PATH (usually more). Any such path must start at one of the odd-degree vertices and end at the other.

Euler's Theorem 3
The sum of the degrees of all the vertices of a graph is an even number (exactly twice the number of edges).
In every graph, the number of vertices of odd degree must be even.

The implications of Euler's theorems are summarized in the table below:

# of ODD Vertices	Implication (for a connected graph)
0	There is at least one Euler Circuit.
1	THIS IS IMPOSSIBLE! (Try it yourself.)
2	There is no Euler Circuit but at least 1 Euler Path.
more than 2	There are no Euler Circuits or Euler Paths.

Keep in mind:

- An Euler Circuit IS a type of Euler Path but an Euler Path is not necessarily an Euler Circuit.
- The sum of the degrees of all the vertices of a graph is twice the number of edges.
- The number of vertices of odd degree must be even.

Back

From http://www.ctl.ua.edu/math103/euler/ifagraph.htm

(Remember, only connected graphs with no vertices of odd deGMATe have Euler circuits.)

oct 10, 2002 notes

How many Euler circuits are there starting from one random vertex? Are there are the same number of Euler Circuits starting from any vertex?

To help me understand about Alu, these are quotes from the internet related to Alu.

From http://www.accessexcellence.org/AE/AEPC/DNA/detection.html

> The Alu family of short interspersed repeated DNA elements are distributed throughout primate genomes. Over the past 65 million years, the Alu sequence has amplified via an RNA-mediated transposition process to a copy number of about 500,000--comprising an estimated 5% of the human genome

> An estimated 500-2,000 Alu elements are mostly restricted to the human genome

From http://www.sequenceanalysis.com/glossary.html#alu

> Alu
> A family of approx. 300 bp repetitive sequences, found dispersed throughout the *human* genome. Almost any 100 kb human nucleotide sequence will have Alu sequences within it.

From the internet, found the definition of Alu sequence:

> **Alu sequences** A family of 300-bp sequences occurring nearly a million times in the human genome.

http://users.aber.ac.uk/obb0/pcr.htm

The alu sequence has 900,000 copies throughout the human genome.

From http://www.iephb.nw.ru/labs/lab38/spirov/hox_pro/rare-alu.html

Over the past 30 to 60 million years these insertions have occurred repeatedly, leaving roughly a million copies of *Alu* scattered through the human genome and making up almost 10 per cent of all the DNA in each cell. During this time, the sequences of the various *Alus* have begun to diverge, so that four distinct subfamilies of *Alu* can now be recognised.

The results also provide the first clear evidence that most *Alus* could have the potential to regulate human genes. Apparently most *Alus* have little effect on nearby genes, perhaps because they are bundled deep within folds of DNA. But with a million *Alus* strewn

randomly through the genome during the course of primate evolution, at least a few are likely to have landed where they could regulate a nearby gene. When this occurred, the effect would be equivalent to randomly twisting a knob on an instrument panel. Usually the effect would be harmful, but once in a while it might produce an interesting and beneficial genetic novelty. It seems that over the last 30 to 50 million years, it would provide good evolutionary fodder.

This is reference paper for HW3, problem 2.

Roy-Engel AM, Salem AH, Oyeniran OO, Deininger L, Hedges DJ, Kilroy GE, Batzer MA, Deininger PL.

Tulane Cancer Center, SL-66, Department of Environmental Health Sciences, Tulane University-Health Sciences Center, New Orleans, Louisiana 70112, USA.

Long and short interspersed elements (LINEs and SINEs) are retroelements that make up almost half of the human genome. L1 and Alu represent the most prolific human LINE and SINE families, respectively. Only a few Alu elements are able to retropose, and the factors determining their retroposition capacity are poorly understood. The data presented in this paper indicate that the length of Alu "A-tails" is one of the principal factors in determining the retropositional capability of an Alu element**.**

The A stretches of the Alu subfamilies analyzed, both old (Alu S and J) and young (Ya5), had a Poisson distribution of A-tail lengths with a mean size of 21 and 26,

respectively. In contrast, the **A-tails of very recent Alu insertions (disease causing) were all between 40 and 97 bp in length.**

The L1 elements analyzed displayed a similar tendency, in which the "disease"-associated elements have much longer A-tails (mean of 77) than do the elements even from the young Ta subfamily (mean of 41). Analysis of the draft sequence of the human genome showed that only about 1000 of the over one million Alu elements have tails of 40 or more adenosine residues in length.

The presence of these long A stretches shows a strong bias toward the actively amplifying subfamilies, consistent with their playing a major role in the amplification process. Evaluation of the 19 Alu elements retrieved from the draft sequence of the human genome that are identical to the Alu Ya5a2 insert in the NF1 gene showed that only five have tails with 40 or more adenosine residues. Sequence analysis of the loci with the Alu elements containing the longest A-tails (7 of the 19) from the genomes of the NF1 patient and the father revealed that there are at least two loci with A-tails long enough to serve as source elements within our model. Analysis of the A-tail lengths of 12 Ya5a2 elements in diverse human population groups showed substantial variability in both the Alu A-tail length and sequence homogeneity. On the basis of these observations, **a model is presented for the role of A-tail length in determining which Alu elements**

are active. [The sequence data from this study have been submitted to GenBank under accession nos. AF504933-AF505511.]

retroposition

Simple backward displacement of a structure or organ, as the uterus, without inclination, bending, retroversion, or retroflexion.

From http://www.dhushara.com/book/upd2/jan01/hgwww/hgp.htm

In the human, coding sequences comprise less than 5% of the genome (see below), whereas repeat sequences account for at least 50% and probably much more. Broadly, the repeats fall into five classes:

(1)transposon-derived repeats, often referred to as interspersed repeats;

(2) inactive (partially) retroposed copies of cellular genes (including protein-coding genes and small structural RNAs), usually referred to as processed pseudogenes;

(3) simple sequence repeats, consisting of direct repetitions of relatively short k-mers such as (A)n, (CA)n or (CGG)n;

(4) segmental duplications, consisting of blocks of around 10±300 kb that have been copied from one region of the genome into another region; and

(5) blocks of tandemly repeated sequences, such as at centromeres, telomeres, the short arms of acrocentric chromosomes and ribosomal gene clusters. (These regions are intentionally under-represented in the draft genome sequence and are not discussed here.)

Classes of transposable elements. In mammals, almost all transposable elements fall into one of four types (Fig. 17), of which three transpose through RNA intermediates and one transposes directly as DNA. These are long interspersed elements (LINEs), short interspersed elements (SINEs), LTR retrotransposons and DNA transposons.

Three distantly related LINE families are found in the human genome: LINE1, LINE2 and LINE3 Only LINE1 is still active.

SINEs are wildly successful freeloaders on the backs of LINE elements. They are short (about 100±400 bp), harbour an internal polymerase III promoter and encode no proteins.

The promoter regions of all known SINEs are derived from tRNA sequences, with the exception of a single monophyletic family of SINEs derived from the signal recognition particle component 7SL. This family, which also does not share its 39 end with a LINE, includes the only active SINE in the human genome: the Alu element.

If the sequences above represent DNA sequences, then taking the max of f*g corresponds to finding the maximum overlap between two DNA sequences

From http://ccrma-www.stanford.edu/~jos/mdft/Convolution_Theorem.html

Convolution Theorem

Theorem: For any $x, y \in \mathbf{C}^N$,

$$\boxed{x * y \leftrightarrow X \cdot Y}$$

Proof:

$$\begin{aligned}
\mathrm{DFT}_k(x*y) &\triangleq \sum_{n=0}^{N-1}(x*y)_n e^{-j2\pi nk/N} \\
&\triangleq \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(m)y(n-m)e^{-j2\pi nk/N} \\
&= \sum_{m=0}^{N-1} x(m) \sum_{n=0}^{N-1} \underbrace{y(n-m)e^{-j2\pi nk/N}}_{e^{-j2\pi mk/N}Y(k)} \\
&= \left(\sum_{m=0}^{N-1} x(m)e^{-j2\pi mk/N}\right) Y(k) \quad \text{(by the Shift Theorem)} \\
&\triangleq X(k)Y(k)
\end{aligned}$$

This is perhaps the most important single Fourier theorem of all. It is the basis of a large number of applications of the FFT. Since the FFT provides a fast Fourier transform, it also provides *fast convolution*, thanks to the convolution theorem. It turns out that using the FFT to perform convolution is really more efficient in practice only for reasonably long convolutions, such as $N > 100$. For much

longer convolutions, the savings become enormous compared with ``direct'' convolution. This happens because direct convolution requires on the order of N^2 operations (multiplications and additions), while FFT-based convolution requires on the order of $N \lg(N)$ operations.

The following simple Matlab example illustrates how much faster convolution can be performed using the FFT:

```
>> N = 1024;        % FFT much faster at this length
>> t = 0:N-1;       % [0,1,2,...,N-1]
>> h = exp(-t);     % filter impulse reponse = sampled exponential
>> H = fft(h);      % filter frequency response
>> x = ones(1,N);   % input = dc (any example will do)
>> t0 = clock; y = conv(x,h); t1 = etime(clock,t0);         % Direct
>> t0 = clock; y = ifft(fft(x) .* H); t2 = etime(clock,t0); % FFT
>> sprintf(['Elapsed time for direct convolution = %f sec\n',...
        'Elapsed time for FFT convolution = %f sec\n',...
        'Ratio = %f (Direct/FFT)'],t1,t2,t1/t2)

ans =

Elapsed time for direct convolution = 8.542129 sec
Elapsed time for FFT convolution = 0.075038 sec
Ratio = 113.837376 (Direct/FFT)
```

From: Martin (martin.jonsson@cetus.se)
Subject: Re: ? Use FFT to do the **convolution** in finite interval
View: Complete Thread (17 articles) | Original Format
Newsgroups: comp.soft-sys.matlab
Date: 2001-12-13 00:38:35 PST

Convolution in finite interval can easyly be done by correlation(xcorr) if you got Signal Processing Tollbox
and the vectors are of equal length. conv(a,b)=xcorr(a,fliplr(b))

Example conv(a,b)=xcorr(a,fliplr(b))
a=[1 2 3 4 5 6 5 4 3 2 1 1 2 3 4 5];
b=[2 3 4 5 6 5 4 3 2 1 1 2 3 4 5 1];
k=xcorr(a,b);

```
figure,plot(k)
bb=fliplr(b);
c=conv(a,bb);%the same as xcorr(a,b)
hold on
plot(c,'r--')
%vector c and k will be the same
```

For finit conv interval use xcorr(a,bb,lag)
Example
```
lag=2;
convfinite=xcorr(a,bb,lag);%the same as conv(a,b) but with finit interval.
figure,plot(convfinite)
```
With lag you look at the center + 2(lag) sampels backwords and 2(lag) forward.

If you don't have the signal processing toolbox or the vectors of different length you could just fix the vectors
or write a little code snippet witch does the **convolution**.

Exampel of finit **convolution**.
```
a=[1 1 1 2 2 2 3 3 3 4]; % length 10
b=[1 1 2 2 3 3 4]; % length 7
ct=conv(a,b);% length 10+7-1=16
%for finite conv.
start=3; %start of finite conv
stop=7; %end of finite conv
%bb=fliplr(b);%bb=[4 3 3 2 2 1 1];
for r=start:stop
   if (r > length(a))
      c(r-start+1)=sum( fliplr(b(r-length(a)+1:r)).*a );
   elseif (r > length(b))
      c(r-start+1)=sum( a(r-length(b)+1:r).*fliplr(b) );
   else
      c(r-start+1)=sum( a(1:r).*fliplr(b(1:r)) );
   end
end
```
%plots the whole **convolution** ct and the finit c in tha same figure to compere.
```
figure,plot(ct)
hold on
plot([3 4 5 6 7],c,'r--')
```

---

from http://www.nist.gov/dads/HTML/viterbiAlgorithm.html

Viterbi algorithm

(algorithm)

Definition: An algorithm to compute the optimal (most likely) state sequence in a *hidden Markov model* given a sequence of observed outputs.

See also *Baum Welch algorithm*.

Note: Also used to decode, i.e. remove noise from, linear error-correcting codes.

hidden Markov model

(data structure)

Definition: A variant of a *finite state machine* having a set of *states*, Q, an output *alphabet*, O, transition probabilities, A, output probabilities, B, and initial state probabilities, Π. The current state is not observable. Instead, each state produces an output with a certain probability, B. Usually the states, Q, and outputs, O, are understood, so an HMM is said to be a triple, (A, B, Π).

Formal Definition: After Michael Cohen.

- $A = \{a_{ij} = P(q_j \text{ at } t+1 \mid q_i \text{ at } t)\}$, where $P(a \mid b)$ is the conditional probability of a given b, $t \geq 1$ is time, and $q_i \in Q$.
  Informally, A is the probability that the next state is q_j given that the current state is q_i.
- $B = \{b_{ik} = P(o_k \mid q_i)\}$, where $o_k \in O$.
  Informally, B is the probability that the output is o_k given that the current state is q_i.
- $\Pi = \{p_i = P(q_i \text{ at } t=1)\}$.

Also known as HMM.

See also *Markov chain*, *Baum Welch algorithm*, *Viterbi algorithm*.

Markov chain

(data structure)

Definition: A *finite state machine* with probabilities for each transition, that is, a probability that the next state is s_j given that the current state is s_i.

See also *hidden Markov model*.

Note: Equivalently, a weighted, directed graph in which the weights correspond to the probability of that transition. In other words, the weights are nonnegative and the total weight of outgoing edges is positive. If the weights are normalized, the total weight, including self-loops, is 1. After [Leda98].

Author: PEB

Baum Welch algorithm

(algorithm)

Definition: An algorithm to find *hidden Markov model* parameters A, B, and Πwith the maximum likelihood of generating the given symbol sequence in the observation vector.

See also *Viterbi algorithm*.

```
> interface(verboseproc=3);
> interface(warnlevel=4);
```

from http://uirvli.ai.uiuc.edu/dugad/

The Three Problems for HMMs

Most applications of HMMs are finally reduced to solving three main problems. These are :

Problem 1 : Given the model how do we compute P(*O*), the probability of occurrence of the observation sequence *O* = , ,..., .

Problem 2 Given the model how do we choose a state sequence *I* = , ,..., so that P(*O*,*I*),the joint probability of the observation sequence *O* = , ,..., and the state sequence given the model is maximized.

Problem 3 How do we adjust the HMM model parameters so that P(*O*) (or P(*O,I*)) is maximized.

Problems 1 and 2 can be viewed as analysis problems while Problem 3 is a typical synthesis (or model identification or training) problem.

```
> restart;
Digits:= trunc(evalhf(Digits)):
n:= 100:
rnd01:= ()-> rand()/(10.^12-11):
X:= [seq(rnd01(), i= 1..n)]:
Y:= [seq(rnd01(), i= 1..n)]:
f:= (x,y)-> sqrt(-2*ln(x))*cos(2*Pi*y):
Z:= evalf(zip(f, X, Y)):
PLOT3D(POINTS([seq([X[i], Y[i], Z[i]], i= 1..n)]),
SYMBOL(CIRCLE));
```

From http://groups.google.com/groups?q=markov+entropy&hl=en&lr=&ie=UTF-8&oe=UTF-8&selm=3v3hrn%24kd8%40nntp5.u.washington.edu&rnum=4

for an expository paper called ``An **Entropy** Primer'', or see the textbook ``Elements of Information Theory'' by Cover and Thomas, Wiley, 1991.

A related theorem (see ``Primer''), the Equipartition Theorem, says that for ergodic text sources, the log of the probability of the particular message you observed is, for sufficiently large n, very close to the source **entropy**. (The source **entropy** is defined as you wrote it for a Bernoulli source, and in a slightly different way for **Markov** shifts; see ``Primer'' and other preprints linked to my home page, or see the textbook ``An Introduction to Ergodic Theory'' by Peter Walters, Springer, 1981.)

We should maximize H(X) = -Sum{pi*log(pi)} with the constraint that Sum{pi} = 1.
(We also have another constraint that pi >= 0 for all i, but ignore it for the time.)

Lagrangian multiplier, say m, is introduced to solve the constrained maximization problem.
Now, H1(X) = -Sum{pi*log(pi)} + m(Sum{pi}-1) should be maximized.

The fact that partial derivative of H1 respective to pi (i = 1, 2, ..., n)
and m should be 0 yields the desired solution.
(And, the solution also meets the constraint pi >= 0.)

For order-K Markov chain:

H = - sum(i1=1 to m) ... sum(iK+1=1 to m) (
p(ai1, ... , aiK+1)*log(p(aiK+1|ai1, ... aiK))),

where a1, ..., am are alphabet symbols, p(x,y,z, ...) is a probability
of occurrence "x,y,z, ..." in source output and p(z| ...,x,y) is a
conditional probability of occurrence z after " ... ,x,y" in source output.

HIGH ORDER ==➔ LOW Entropy

So if a sequence is highly ordered, or in other words, highly predictable, then the entropy will be LOW.
So, random data will have high entropy.

From the net, example to exec program from java and get its output

> You might try something like this...

```
import java.io.*;

public class Test
{ public static void main (String args[])

  { try {


     Runtime myRunTime = Runtime.getRuntime();
     Process myProcess;
     InputStream theInputStream;
     byte[] theResultBytes;
     int numberResultBytes;
     int resultCode;
     String cmdString = "C:/test.bat";
     String theResultString;

     myProcess = myRunTime.exec(cmdString);
     theInputStream = myProcess.getInputStream();
```

```
    myProcess.waitFor();
    numberResultBytes = theInputStream.available();
    System.out.println("numberResultBytes: " + numberResultBytes);
    theResultBytes = new byte[numberResultBytes];
    resultCode = theInputStream.read(theResultBytes);
    theResultString = new String("");
    while(resultCode != '\0' && resultCode != -1)
    { theResultString =
        new String(theResultString + new String(theResultBytes));
      resultCode = theInputStream.read(theResultBytes);
    }
    System.err.println(theResultString);
    theInputStream.close();
   }
   catch(java.lang.InterruptedException e)
   { System.err.println("Something went wrong");
   }

   catch(java.io.IOException e)
   { System.err.println("Something went wrong");
   }
  }
}
```

---

from http://www.asp.ucar.edu/colloquium/1992/notes/part1/node9.html

The *mean* of a set of measurements $\{x_1, x_2, \ldots x_N\}$ is the average:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{2.1}$$

The *expectation value* of a quantity is the value expected if averaged over the entire parent population, and will be denoted by angle brackets: $\langle\rangle$. For example, the mean in the parent population that corresponds to the sample mean x is

$$\mu = \langle x \rangle = \lim_{N\to\infty}(\overline{x}).$$

So, the expected values is the AVERAGE, when done over the whole population.

When the sample size is smaller, then we called it the average. So, expectation is a stronger sense of the average.

- L. Allison. *Using Hirschberg's algorithm to generate random alignments.* Inf. Proc. Lett. **51** 251-255 1994.

- L. Allison and C. S. Wallace. *The posterior probability distribution of alignments and its application to parameter estimation of evolutionary trees and to optimization of multiple alignments.* Jrnl. Molec. Evol. **39** 418-430 1994.
- D. S. Hirschberg. *A linear space algorithm for computing maximal common subsequences.* Inf. Proc. Lett. **18** 341-343 1975.

D. S. Hirschberg.
Algorithms for the longest common subsequence problem.
J.ACM, 24:664-675, 1977.

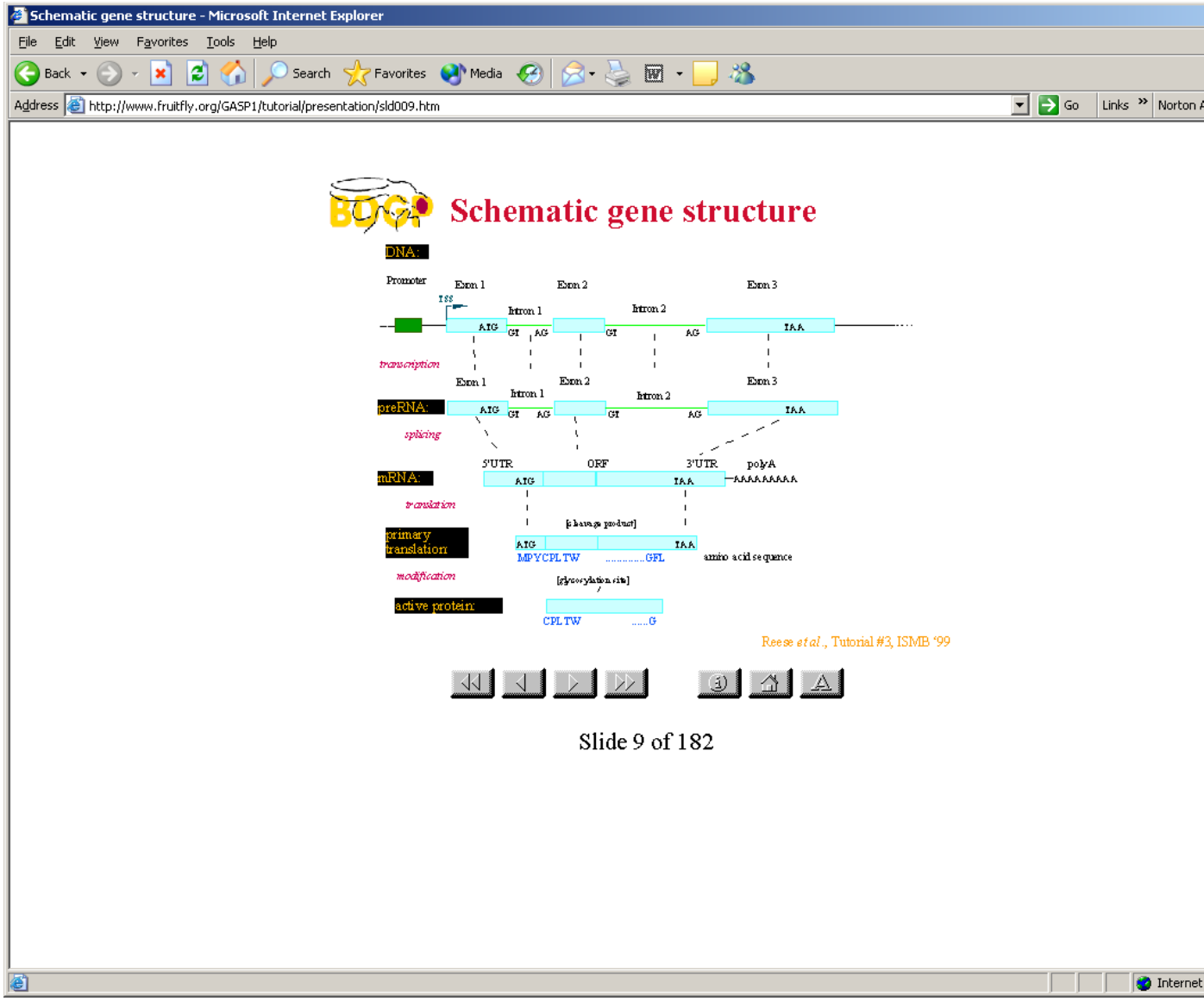
http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Dynamic/Hirsch/

this below finds time and space complexity for hirsh. Algo

http://www.math.tau.ac.il/~rshamir/algmb/98/scribe/html/lec02/node10.html

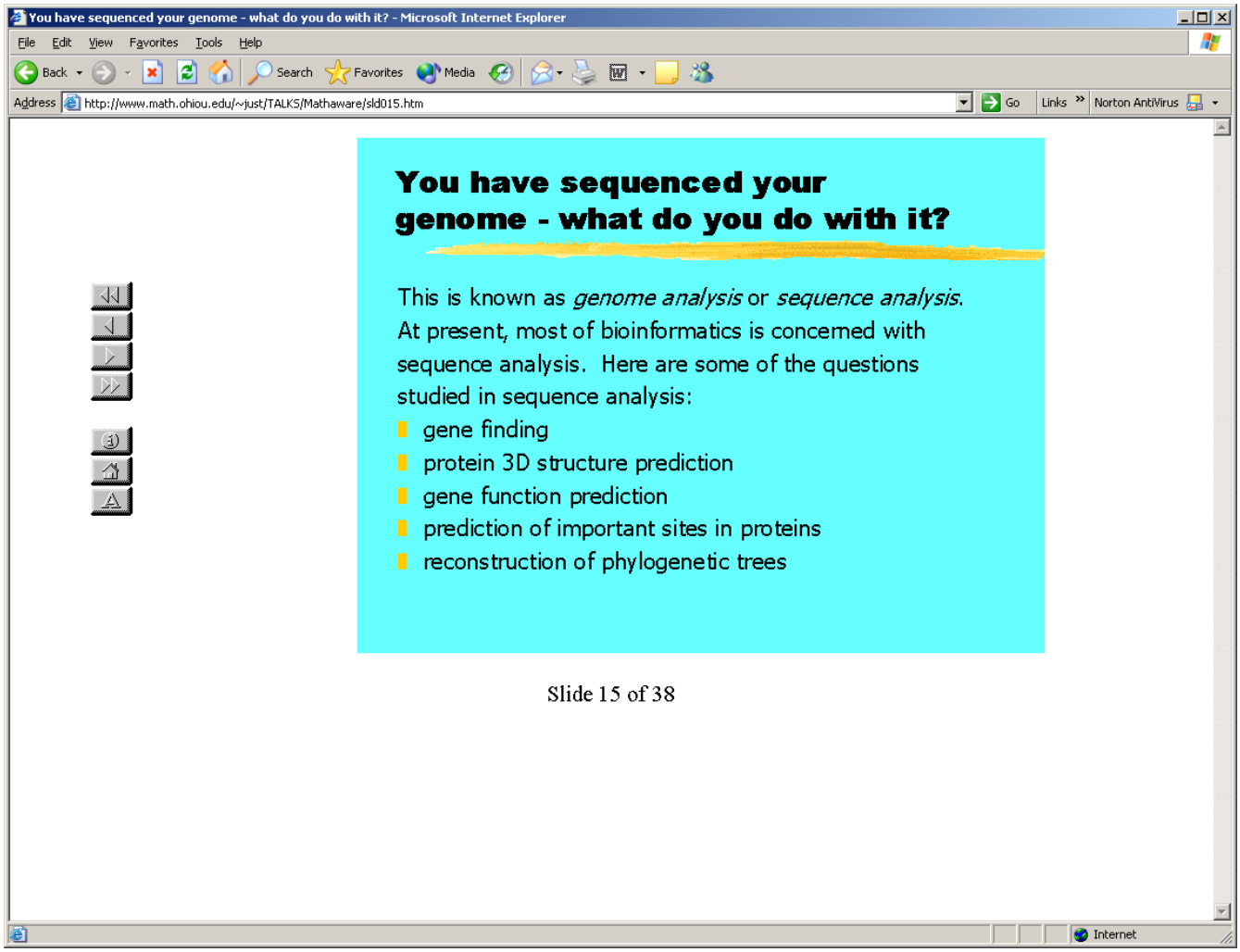
gene finding and markov, time/space estimates:
http://www.csse.monash.edu.au/~lloyd/tildeStrings/Notes/20001205HMMgene.html

contains good gene struct
http://www.fruitfly.org/GASP1/tutorial/presentation/sld009.htm



from http://www.cs.mcgill.ca/~kaleigh/work/hmm/hmm_paper.html

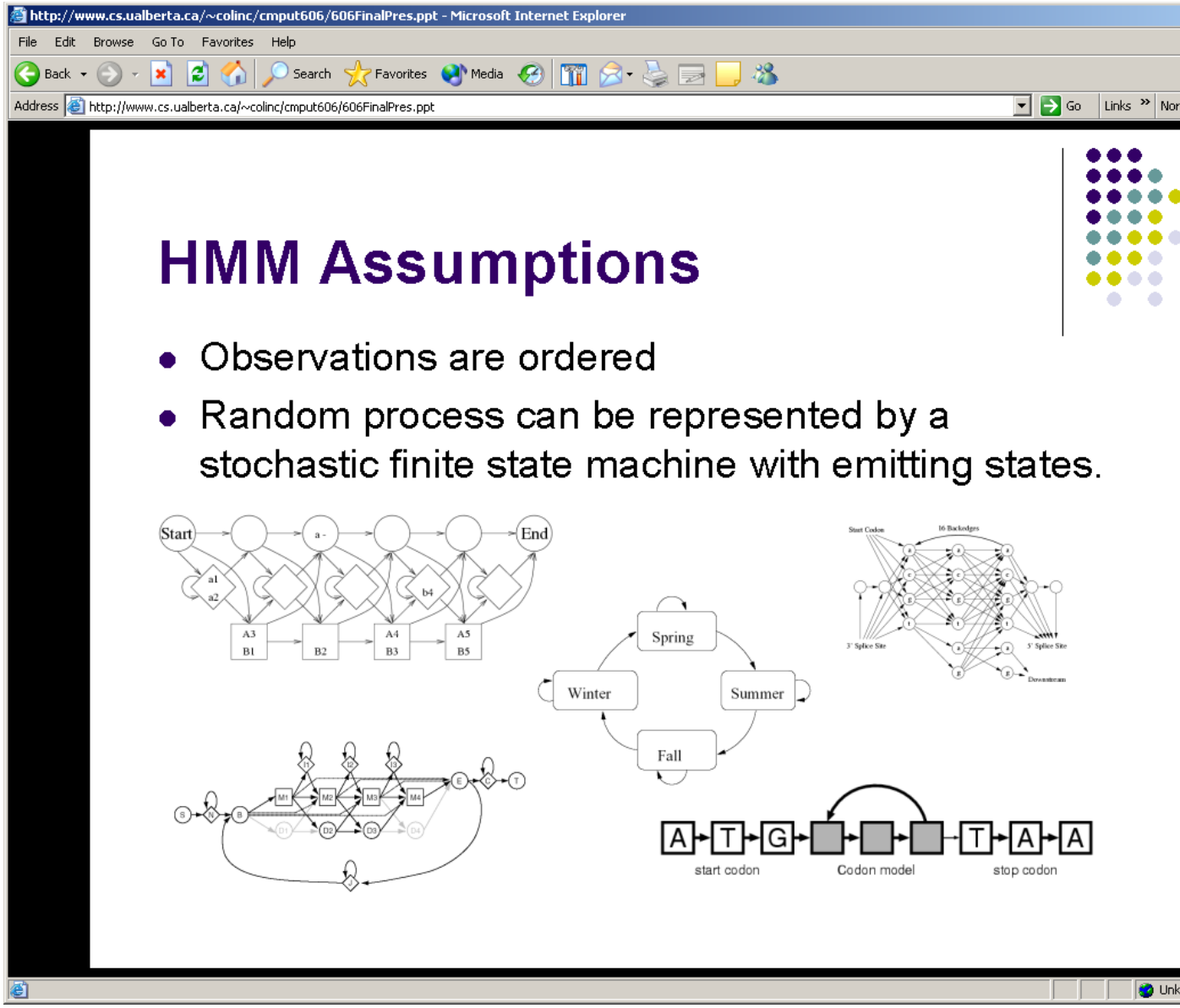
A eukaryotic gene contains coding regions called exons which may be interupted by non-coding regions called introns. The exons and introns are separated by splice sites. Regions outside genes are called intergenic. The goal of gene finding is then to annotate the sets of genomic data with the location of genes and within these genes, specific areas such as promoter regions, introns and exons.



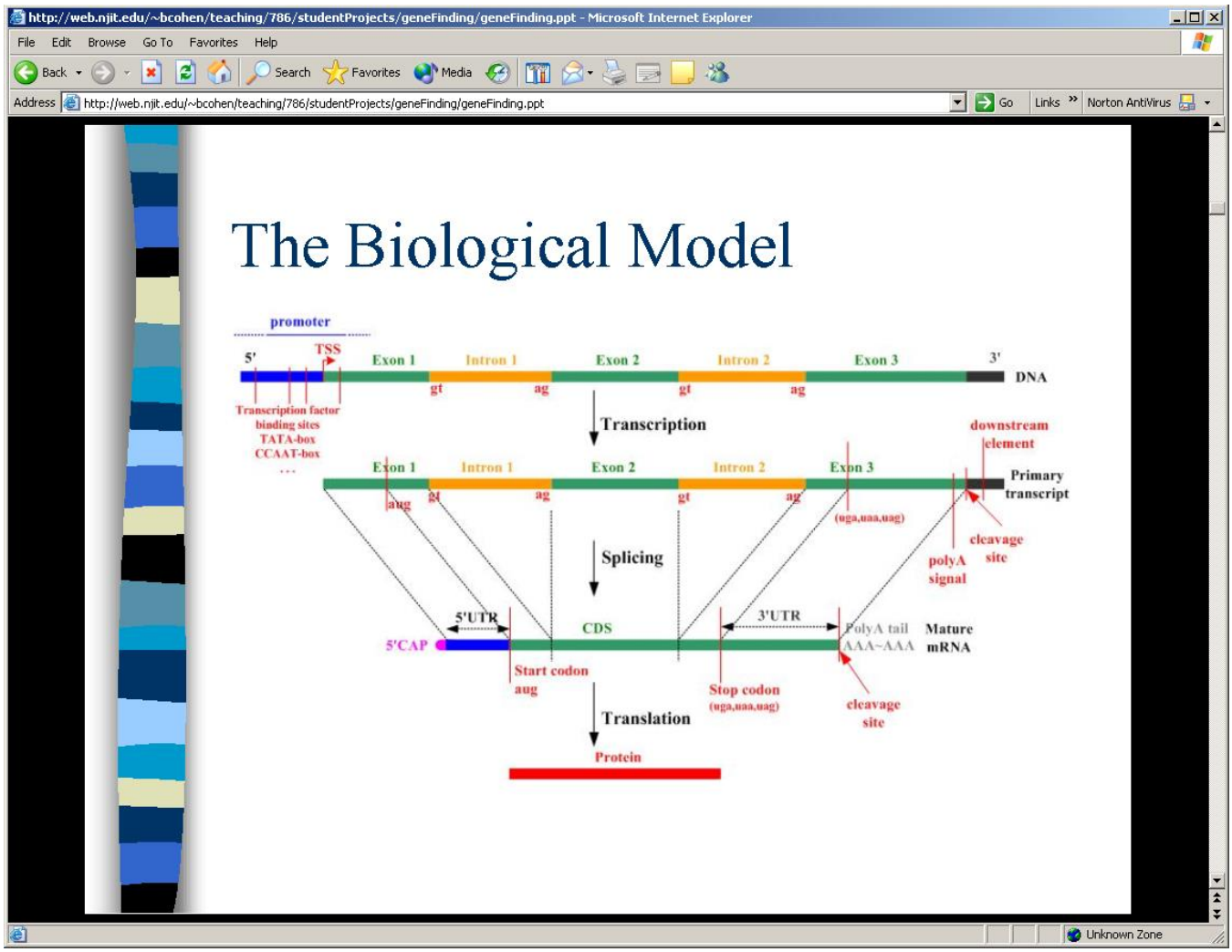
Check this:

[10]

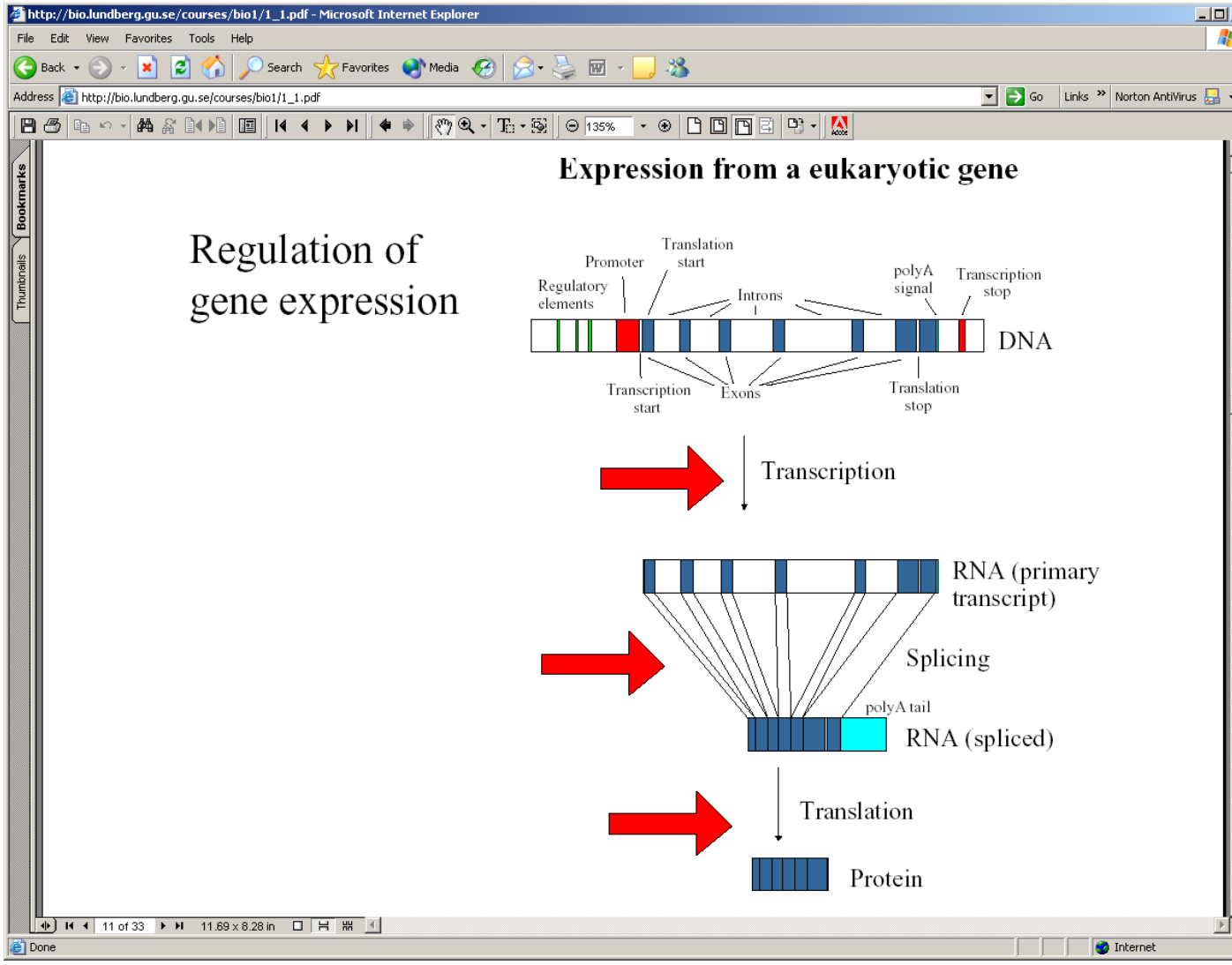
Kanungo, Tapas. HMM software learning toolkit. University of Maryland Institute for Advanced Computer Studies, Center for Automation Research, Language and Media Processing Lab - http://www.cfar.umd.edu/ kanungo/software/software.html.



from
http://web.njit.edu/~bcohen/teaching/786/studentProjects/geneFinding/geneFinding.ppt



this below from http://bio.lundberg.gu.se/courses/bio1/1_1.pdf



from http://www.genomicglossaries.com/content/proteins_glossary.asp

Poly A: A group of adenine ribonucleotides in which the phosphate residues of each adenine ribonucleotide act as bridges in forming diester linkages between the ribose moieties. [MeSH, 1976]

polyadenylation: The addition of a tail of polyadenylic acid (POLY A) to the 3' end of mRNA (RNA, MESSENGER). Polyadenylation involves recognizing the processing site signal, (AAUAAA), and cleaving of the mRNA to create a 3' OH terminal end to which poly A polymerase (POLYNUCLEOTIDE ADENYLTRANSFERASE) adds 60- 200 adenylate residues. The 3' end processing of some messenger RNAs, such as histone mRNA, is carried out by a different process that does not include the addition of poly A as described here. [MeSH, 2002]

During the maturation of messenger RNA, about 200 adenosine nucleotides are added in a polyadenylation reaction at the 3' end. These are not coded by the corresponding gene. In certain cases there are multiple alternative polyadenylation sites in the primary transcript. This was first observed in adenoviruses [127 - 131]. In cellular genes many alternative polyadenylation sites have also been found [see 132 for review]. Alternative polyadenylation sites usually involve the untranslated trailer sequence in the messenger RNA, but they can also involve translated sequences, and in this case they can affect the structure of the encoded protein. Thus multiple polyadenylation sites are one mechanism whereby a single gene can control the synthesis of more than one polypeptide. [Petter Portin in "The Origin, Development and Present Status of the Concept of the Gene: A Short Historical Account of the Discoveries" Univ. of Turku, Finland, 2000] http://www.bentham.org/cg1-1/portin/P.Protin.htm

translation: The unidirectional process that takes place on the **ribosomes** whereby the genetic information present in an **mRNA** is converted into a corresponding sequence of amino acids in a **protein**. [IUPAC Bioinorganic

transcription, genetic: The transfer of genetic information from DNA to messenger RNA by DNA- directed RNA polymerase. It includes reverse transcription and transcription of early and late genes expressed early in an organism's life cycle or during later development. [MeSH, 1973]

MAPLE:

evelyn wrote:
Just plot f,g,h with their ranges like
fplot := plot(f(x),x=a..b);
and then do
with(plots):
display({fplot,gplot,hplot});

Cheers,
Raphael

> Hello,
>
> I want to plot 3 different funtions f(x), g(x), h(x) in one plot and
> different colors. But every function has a different range. I will
> give an example:
>

```
> funcion 1: f in funcion of x in a range from x=0 to x=0.1
> funcion 2: g in funcion of x in a range from z=0.1 to x=2.5
> function 3: h in function of x in a range from x=2.5 to x=4
>
> Is that possible?
>
> Evelyn
```

CTG has been described as **start codon** for

start codon (initiation codon)

The triplet of nucleotides on a messenger RNA molecule (see codon) at which the process of translation is initiated. In eukaryotes the start codon is AUG (see genetic code), which codes for the amino acid methionine; in bacteria the start codon can be either AUG, coding for *N*-formyl methionine, or GUG, coding for valine. Compare stop codon.

This below takes a little on splice sites
http://www.fruitfly.org/GASP1/doc/format.html

maple

```
>plot3d('findstable(1,1,1,1,x,y,1)', x=0..1, y=0..1);
```

```
use `=`= Equals in
  x= 9; y= 9; z= 9;
  x= 9*6; x= x*3
  # Any number of statements can go in the use block
end use;
```

EXAM prep

Knots.
Q1. application to DNA supercolining.
Q2. explain topoisomerases and how it works.
A2.
See http://www.mun.ca/biochem/courses/3107/Lectures/Topics/supercoiling.html

The degree of supercoiling in the cell must be and is carefully controlled by the action of topoisomerases.

These enzymes catalyse the transient breaking and rejoining of DNA strands.

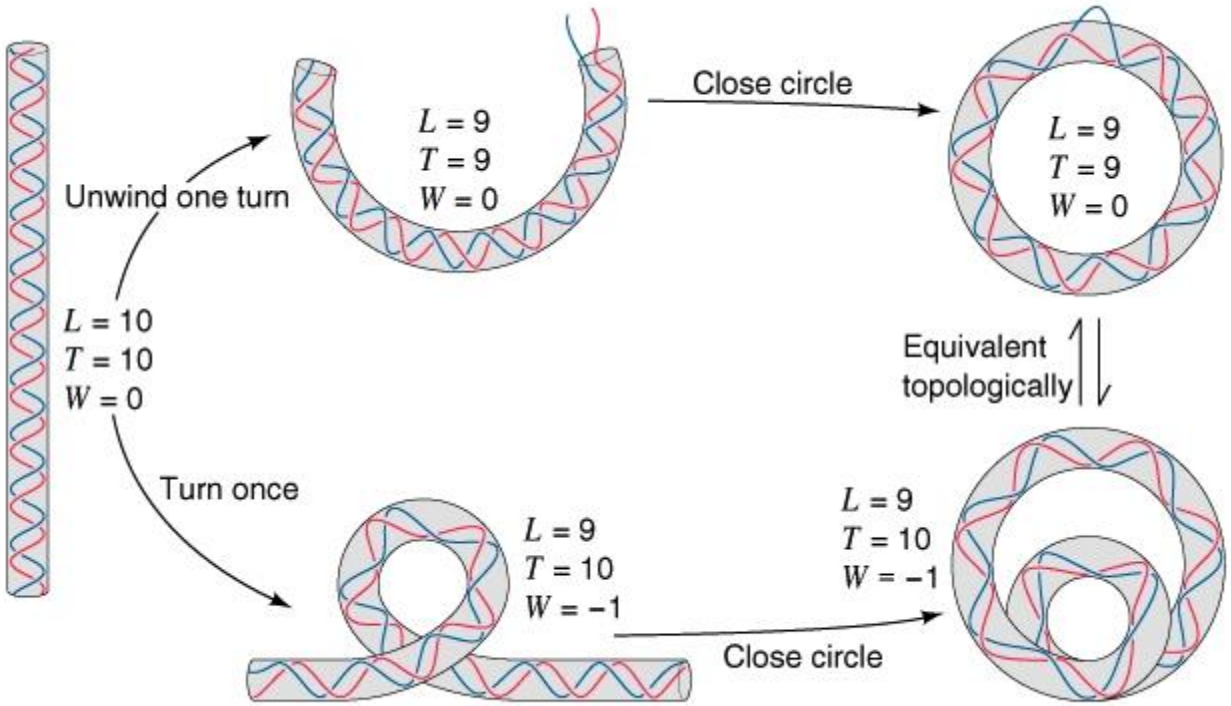
Enzymes that change the degree of supercoiling in DNA by cutting one or both strands.
Type I topoisomerases cut only one strand of DNA; type I topoisomerase of E. coli > *E. coli* (omega protein) relaxes negatively supercoiled DNA and does not act on positively supercoiled DNA.
Type II topoisomerases cut both strands of DNA; type II topoisomerase of *E. coli* (DNA gyrase) increases the degree of negative supercoiling in DNA and requires ATP. It is inhibited by several antibiotics, including nalidixic acid and ovobiocin.

The major role of topoisomerases is to prevent DNA tangling.

As a result, the type I enzyme removes supercoils from DNA one at a time, whereas the type II enzyme removes supercoils two at a time. Although the type II topoisomerase is more efficient in removing supercoils, this enzyme requires the energy from ATP hydrolysis, but the type I topoisomerase does not.

Without topoisomerases, the DNA cannot replicate normally.

Q3 what is a link number
A3 Number of times two DNA strands interwinds each others.
Another: This is only defined for two strands, not a single strand. Supercoiling in circular DNA molecules is described mathematically by the number of times the two phosphodiester backbones wrap around one another in a given distance. This quantity is the **Linking number** (Lk). If either one of the two DNA strands is nicked, this value has no meaning.
Because the entire molecule will be a closed circle, the linking number will always be an integer.



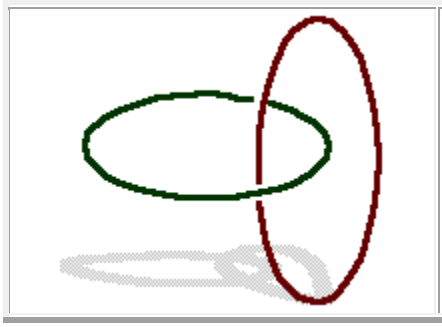
Q4. what is the twist
A4. One definition: number of times basepairs twist around the central axis.
Another: The twist of a ribbon measures how much it twists around its axis and is defined as the integral of the incremental twist around the ribbon

Q4.1 What is the Writhe number
Is a simple function of only the molecule axis vector r.

$$Wr = \frac{1}{4\pi} \int \int \mathrm{ds}\, \mathrm{ds}' \, \frac{\partial_s \mathbf{r}(s) \times \partial_{s'} \mathbf{r}(s') \cdot [\mathbf{r}(s) - \mathbf{r}(s')]}{|\mathbf{r}(s) - \mathbf{r}(s')|^3} . \tag{11}$$

Q4.2 How to find the link number

This is defined ONLY for 2 components. A KNOT does not have a link number, it has a crossing number. Linking number measures how many times one components go around the other.



The linking number is 1.

either do the –1, +1 sum, and then divide by 2 the final result as shown below:

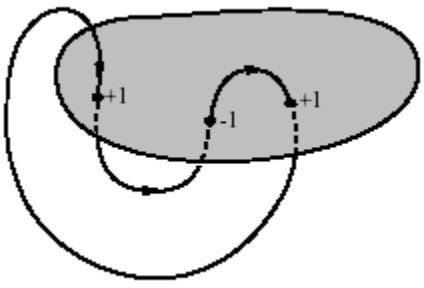
A link invariant defined for a two-component oriented link as the sum of $+1$ crossings and -1 crossing over all crossings between the two links divided by 2.

For components α and β,

$$Lk(\alpha, \beta) \equiv \tfrac{1}{2} \sum_{p \in \alpha \sqcap \beta} \epsilon(p),$$

OR use this method below. I use this method to find the linking number between two components, (it is easier)

From paper: http://www.biophysics.org/btol/img/Vologodskii.A.pdf
A nice way to find the link number. Take one strand, and make is a closed surface. Then find how many times the other strand cuts the countor. If it cuts the contour twice from the same side one after the other, then they both cancel each others. Now simply add the number the other strand cuts the closed surface.



Just make one component shaded as above, and then add arrows to make sure the direction of one strand is kept consistant, then count how many times the strands crosses the shaded plan, as above.

Q5. what is crossing number
The least number of crossings that occur in any projection of a link. In general, it is difficult to find the crossing number of a given link. Knots and links are generally tabulated based on their crossing numbers.

Q6. What is a knot
A knot is defined as a closed, non-self-intersecting curve embedded in three dimensions.
A knot is a single component link
i.e. a knot is a CLOSED curve in 3D space that does not interest itself.
Teacher definition: a knot is a function which maps a unit circle into R^3, and a link is a function which maps a collection of disjoint unit circles into R^3.

This is another definition: A **knot** is a polygon embedded in threespace

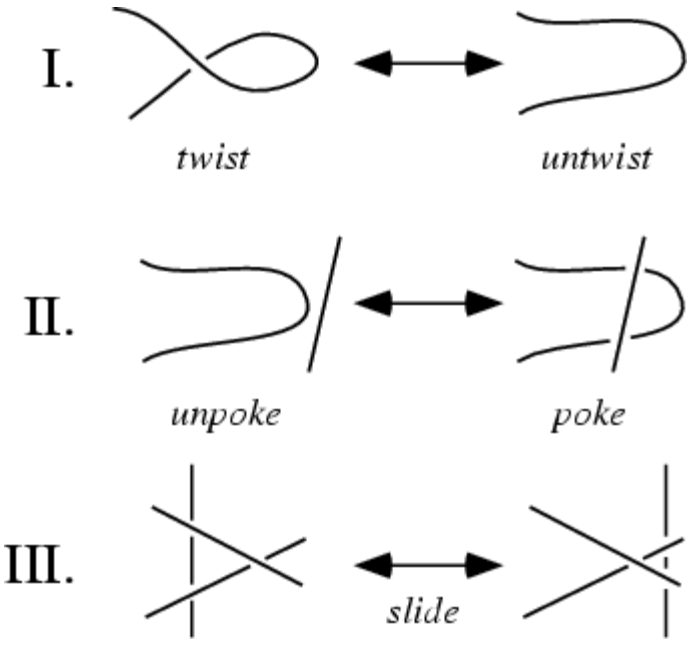
Q7. what is a component
More informally, a link or component is an assembly of knots with mutual entanglements.

Q8. what does it mean that 2 knots are equivelant?
A8. This means there exist a number of reidmester moves that transforms one knot to the other.

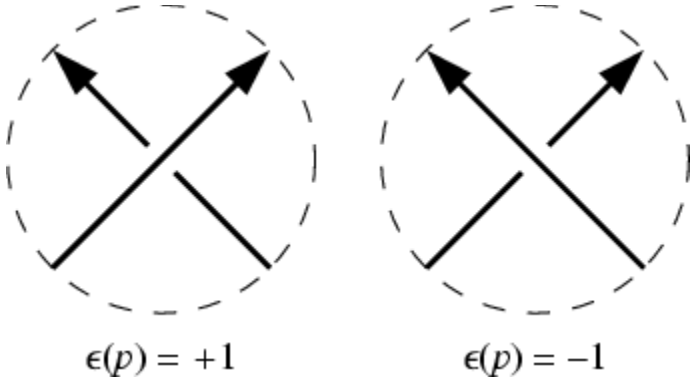
Q9 what is Jones polynomial
A9. It is a knot invariant in the form of a polynomial

Q10. what are the reidemeister moves
A10. twist, poke, slide.
Reidemeister first rigorously proved that knots exist which are distinct from the unknot. He did this by showing that all knot deformations can be reduced to a sequence of three types of "moves,"

NOTE: do both sense for this. When up/down.



Q11. What is the directed writhe ?
A11. This is when we give an orientation to the knot projection, at each crossing we have a +1 or a –1. The sum of these is the writhe.



Also called the twist number. The sum of crossings *p* of a link *L*,

$$w(L) = \sum_{p \in C(L)} \epsilon(p),$$

Q12. What is the calugareanu formula.
A12. Link number = Twist number + writhe number

Writhe number can be found as above, by counting +1, -1, or by an integral

A formula for the writhe is given by

$$\mathrm{Wr}(K) = \frac{1}{4\pi} \int_K ds \int_K dt\, e^{\mu} \frac{de^{\mu}}{ds} \frac{de^{\alpha}}{dt}$$

Twist number: The twist of a ribbon measures how much it twists around its axis and is defined as the integral of the incremental twist around the ribbon. A formula for the twist is given by

$$\mathrm{Tw}(K) = \frac{1}{2\pi} \int_K ds\, \varepsilon_{\mu\nu\alpha} \frac{dx^{\mu}}{ds} n^{\nu} \frac{dn^{\alpha}}{ds},$$

Link number is A link invariant defined for a two-component oriented link as the sum of $+1$ crossings and -1 crossing over all crossings between the two links divided by 2. For components α and β,

$$Lk(\alpha, \beta) \equiv \tfrac{1}{2} \sum_{p \in \alpha \sqcap \beta} \epsilon(p),$$

Q13 When are 2 knots equivalent?
Definition 4 Let L, L' be knots (links). Then, $L \sim L'$ if the knot diagram of L' can be obtained from the knot diagram of L using a finite sequence of *Reidemeister moves*.

Q14. How does assembly work?
A14. Assembly means finding the shortest common superstring for all the reads. This is an NP complete problem. So use the greedy algorithm. This is done in two steps. First find best overlap between each 2 reads, next put the reads togother. For first step, use convolution to find max overlap.

Q15. What is coverage in assembly?
A15. This is a measure of how many times a position is found in different reads. For example, 6x coverage means each position (base) is covered on 6 different reads or fragments.

Q16. what is a gene
A segment of a DNA molecule that contains all the information required for synthesis of a product (polypeptide chain or RNA molecule), including both coding and non-coding sequences. It is the biological unit of heredity, self-reproducing, and transmitted from parent to progeny. Each gene has a specific position (locus) on the chromosome map. From the standpoint of function, genes are conceived of as structural, operator, and regulatory genes.

OR

A gene is a segment of a chromosome that encodes instructions that allow a cell to produce a specific product, typically a protein, that initiates one specific action.

Q17. what is an exon?
A segment of a eukaryotic gene that is transcribed as part of the primary transcript and is retained, after processing, with other exons to form a functional mRNA molecule. *See* DNA; intron; split gene; splicing.

Q18. what is an intron?
intron; **intervening sequence** A segment of DNA sequence of a eukaryotic gene, not represented in the mature (final) mRNA transcript, because it is spliced out of the primary transcript before it can be translated; a process known as intron splicing. Some genes of higher eukaryotes contain a large number of introns, which make up the bulk of the DNA sequence of the gene. Introns are also found in genes whose RNA transcripts are not translated, namely eukaryotic rRNA and tRNA genes. In these cases the intron sequence does not appear in the functional RNA molecule. *cf* exon.

DNA ---(transcript)---> preRNA (intron+exons) ---(splice)--→ RNA --- (translate)-→ protein

Q19. what is HMM
A statistical method that describes a series of observations (in our case nucleotides) by a hidden stochastic process.

Another nice definition. It is like a profile. From some observations, we deduce hidden actions.

Q20. what is a suffix tree.
A given suffix tree can be used to search for a substring, pat[1..m] in O(m) time.

Q21: What repates are there?
A. Simple and interspread repeats. Simple like ACACACAC, while interspread are LINE (Long
INTerspread repeats) and SINS (Short INTerspread repeats).

Q22. What is electrophresis?
A. Sorting DNA pieces by length. DNA travel across a potential in a rate of travel ~ 1/LogL.

From help on clustALW http://www-igbmc.u-strasbg.fr/BioInfo/ClustalW/help2.html

Multiple alignments are carried out in 3 stages (automatically done from menu item 1 ...Do complete multiple alignments now):

1. all sequences are compared to each other (pairwise alignments);
2. a dendrogram (like a phylogenetic tree) is constructed, describing the approximate groupings of the sequences by similarity (stored in a file).
3. the final multiple alignment is carried out, using the dendrogram as a guide

this below also does multiple protein seq. alignment.

http://npsa-pbil.ibcp.fr/cgi-bin/npsa_automat.pl?page=npsa_clustalw.html

on divide-and-conqure, from
http://www.cse.ucsc.edu/research/compbio/papers/samspace/node2.html

The solution to this is to use a sequence alignment method that requires less space. In the case of finding the single best path, there is an elegant divide-and-conquer algorithm that requires only $O(n+m)$ space, where n is the sequence length and m is the model length [Hirschberg, 1975]. The approach of this algorithm is to find a midpoint of the best path without saving all $O(nm)$ dynamic programming entries, and then to solve two smaller problems, each of approximate size $nm/4$ using the same algorithm. This algorithm is well known in the computational biology community [Myers & Miller, 1988], and has, for example, been implemented in the HMMer package for sequence alignment to a trained HMM [Eddy *et al.*, 1995].

More interesting and more complicated is the number of unlabeled trees on n vertices ;^)

Here you can get an asymptotic in R. Otter: The number of trees in the Annals of Mathematics, Vol 49, No 3 July 1948.