(Note: These tutorials are meant to provide illustrative examples of how to use the AMBER software suite to carry out simulations that can be run on a simple workstation in a reasonable period of time. They do not necessarily provide the optimal choice of parameters or methods for the particular application area.)


# Section 2: Setting up duplex DNA: polyA-polyT

***Updated for AMBER 16***

The first stage of this tutorial is to create the necessary input files required by *sander* for performing minimization and molecular dynamics.

There are a number of methods for creating these input files. In this example we will use a program written specifically for this purpose: **LEaP**. This is a program that reads in force field, topology and coordinate information and produces the files necessary for production calculations (*i.e.* minimization, molecular dynamics, analysis, etc.). There are two versions of this program provided with AMBER 14 & AmberTools 15. A graphical version called *xleap* and a terminal interface called *tleap*. (There are also completely new versions being produced called *gleap* and *sleap* respectively that will ultimately replace *xleap* and *tleap* but discussion of this is beyond the scope of this tutorial.) Since we want to "see" graphical representations of our models we will use *xleap* in this tutorial.

The approximate ordering of this section is as follows:

1. Where do I get the coordinates?
2. What representation should I use and what should I simulate?
   A discussion of issues to consider before starting...
3. Building the prmtop and inpcrd files... This will be done using *xleap*.

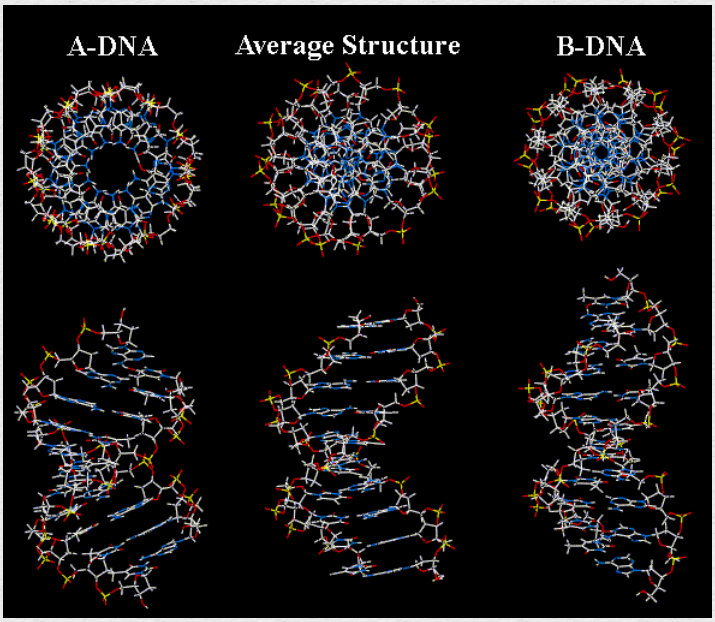
## 2.1) Generating the coordinates of the model structure

The first step in any modeling project is to develop the initial model structure. Although in principle, one could use **`xleap`** to build a model structure by hand, this is only practical for the smallest of systems. The difficulty in both manipulating and predicting the structure of large biomolecular systems means that building a structure by hand is not usually a sensible undertaking. Instead, experimentally determined structures are used. These can be found by searching through databases of crystal or NMR structures such as the *Protein Data Bank* or the *Cambridge Structural Database*. With nucleic acids, users can also search the *Nucleic Acid Database*.

When experimental structures are not available, all hope is not lost since there are a variety of programs that facilitate building model structures using homology modeling and predictive techniques; the list of possible sources is beyond the scope of this tutorial. However, it is worth mentioning that for nucleic acid structure prediction, Dave Case and Tom Macke, formerly of The Scripps Research Institute, have developed the NAB molecular manipulation language which facilitates the building of complex nucleic acid structures. NAB is included as part of the AmberTools package (see *http://www.ambermd.org/* for more info). Numerous methods also exist for predicting the structure of proteins but in general such structure prediction is still in its infancy. Thus a good experimental structure is typically preferred. If, however, a predicted protein starting structure is all that is available it should be noted that these typically require more elaborate minimization and equilibration procedures prior to production dynamics simulations than do structures found by experimental methods.

**Hurdle #1**: As pointed out in the later tutorials, sometimes dealing with Brookhaven PDB files (as provided by the Protein Data Bank) can be rather tricky due to variations in naming conventions. The naming, and often formatting issues, that "break" the programs are the first hurdle that must be overcome in order to perform detailed molecular simulation. Generally within AMBER, if you are using standard amino acid or nucleic acid residues, at most all that is necessary is some slight massaging of the PDB format and atom names. If, however, you wish to simulate complex carbohydrates, lipids or non-standard protein residues, then you may be required to develop parameters and topology information. This, however, is a more advanced issue that will be deferred until later tutorials.

### 2.1.1) Creating our DNA duplex using NAB




Pictured above are models of the decamer poly(A)-poly(T) shown end-on (top) and with a view into the minor (top half of the molecule) and major (bottom half of the molecule) groove (bottom). On the left is canonical A-DNA, the center is the average structure previously discussed, and on the right is canonical B-DNA. The method for creating the canonical structures is discussed below.

Since we don't have an experimentally determined structure for our 10-mer DNA duplex, we are going to have to create one from scratch. A good resource is *w3dna.rutgers.edu*, which allows one to create a wide variety of nucleic acid structures. Here, we will use the *fd_helix()* routine in NAB.

> ***A note on force fields:*** *A number of different force fields are supplied with AMBER. In AMBER v5.0 and v6.0 the default field was the Cornell et al. (1995) or* **`parm94.dat`** *force field (referred to as FF94 in AMBER v8.0 and later). In AMBER v10.0 and 11 the force field recommended for the simulation of proteins and nucleic acids in explicit solvent was either the FF99SB or the FF03 force field which contained several improvements over the FF94 force field. As of Amber 12 a new FF12SB force field was developed which was further refined in AMBER 14 as the FF14SB force field. FF14SB is an updated version of FF99SB and is the current recommended force field for use in simulating amino acids in AMBER v16. Note that as of Amber 16, amino acids, nucleic acids, and solvent/ions have been separated into different leaprc files. There are two parameters sets for DNA: BSC1 and OL15. Although we will use BSC1 parameters in this tutorial, both BSC1 and OL15 are improvements over the older BSC0 parameters. More details on the available force fields can be found in the AMBER manual and the papers referenced within.*

With the BSC1 force field, phosphates are considered to be part of the nucleotide (as is standard). Because the terminal groups of DNA are different (*i.e.* 3'- and 5'- terminal residues), the names have to be different to associate the appropriate topology (*i.e.* list of bonds, atoms, etc.) and parameters. For DNA, the residue names used are DA5, DA and DA3 for a 5' terminal adenine, a non terminal adenine, or a 3' terminal adenine respectively. [A single isolated adenine nucleotide is DAN.]

So, with this in mind we can construct the input file required by **`NAB`** to build our 10-mer polyA-polyT DNA duplex in the Arnott B-DNA canonical structure. This program is given below. Basically, this is building two strands of A-T paired DNA. For more specific information about the various options, see the *manual*.)

```
molecule m;
m = fd_helix( "abdna", "aaaaaaaaaa", "dna" );
putpdb( "nuc.pdb", m, "-wwpdb");
```

Put the above into a plain-text file called *nuc.nab*.

*Note:* Before we run any of the programs provided with AMBER, we need to make sure the Unix shell environment variable that specifies where AMBER is installed is set properly. This is necessary for **`xleap`** and lets the system know where the executables are located ($AMBERHOME/bin).

If you are running bash, csh or tcsh check the AMBERHOME environment variable is set by typing:

```
echo $AMBERHOME
```

If you see:

```
AMBERHOME: Undefined variable.
```

(Using csh or tcsh) or simply a blank line (bash) then the AMBERHOME environment variable was not set properly and you need to initialize it to point to the AMBER installation directory. If AMBER is installed in /usr/local/amber14 then you would type:

```
setenv AMBERHOME /usr/local/amber14
```

(csh or tcsh) or

```
export AMBERHOME=/usr/local/amber14
```

(bash).

While you are at it, you may want to update your **`.cshrc`** (csh), or **`.bashrc`** (bash) file to define this variable and add the AMBER binary directory to your path (or list of directories searched for executables) at login. In your **`.bash_login`** file [or the global **`/etc/bashrc`** file] (if using the bash shell), add the following (substituting the correct path to AMBER):

```
export AMBERHOME=/usr/local/amber14
export PATH=$PATH:$AMBERHOME/bin
```

For further details on installation please refer to the AMBER manual.

***Running NAB***
To run **`NAB`**, you just type this:

```
nab nuc.nab
./a.out (Note: type "./a.exe" for  Cygwin/Windows)
```

This should produce a *nuc.pdb* file, which is the model structure of our DNA duplex. It contains the Cartesian coordinates of all of the atoms in the duplex in locations determined from fibre-diffraction data. The file should consist of 640 lines with a single line for each atom, 638 atoms in total as well as two lines containing the word TER, one after the first 10mer chain and one at the end:

| *LINE DEFINITION* | *ATOM NO.* | *ATOM NAME* | *RESIDUE NAME* | *RESIDUE NO.* | *X* | *Y* | *Z* |
|---|---|---|---|---|---|---|---|
| ATOM | 1 | HO5' | DA | 1 | -0.423 | -8.150 | -2.094 |
| ATOM | 2 | O5' | DA | 1 | 0.427 | -7.826 | -1.788 |
| ... | ... | ... | ... | ... | ... | ... | ... |

## 2.1.2) Loading the structure into Leap

The next step is to take a look at the model structure. It is always a good idea to look at the models before trying to use them. In this way problems can often be identified before running expensive calculations. **`xleap`** works fine for displaying models assuming that the appropriate residue definition files are loaded into **`xleap`** and the residue names in the pdb file are consistent with what **`xleap`** expects. Alternatively a range of freely

available and commercial packages exist for viewing pdb files. A very good program, although not the only choice, is VMD (*http://www.ks.uiuc.edu/Research/vmd/*) which is freely available for academic research. For the moment we will stick to using ***xleap*** since this is what we will be using to create the input files for our simulations. Other methods of visualization will be covered in later sections of this tutorial.

In addition to residue names, ***xleap*** also requires information about the chain connectivity. Residues can be connected to other residues from both sides, only one side or not at all. In ***xleap*** lingo, the residue can have a "head" or a "tail" (or both).

- **head and tail**: the residue has connectivity from both sides such as an internal residue in a protein or nucleic acid. Examples are alanine "ALA" and thymine "DT".
- **tail only**: the residue is a terminal residue at the start of a chain. Examples are the N-terminal residues of proteins "NALA" or the 5'-terminal residue of nucleic acids "DT5".
- **head only**: the residue is a terminal residue at the end of a chain. Examples are the C-terminal residues of proteins "CALA" or the 3'-terminal residue of nucleic acids "DT3".
- **no head or tail**: the residue doesn't connect to other residues. Examples are an isolated nucleotide "TN" or water "WAT" or an ion "Na+".

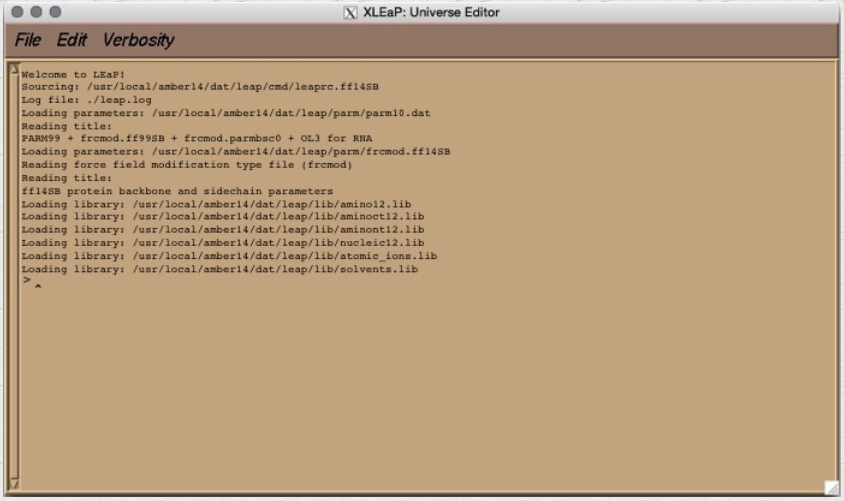
The various different residues and their names are defined in library files that ***xleap*** loads upon starting. More on this later. For the moment it is sufficient to say that the names used for the residues in the pdb files must match those defined in the default ***xleap*** library files or in user defined library files.

***xleap*** is fairly simple in that it expects that all residues in the pdb file are connected in the order in which they are listed unless they are separated by a "TER" card (*i.e.* a single line that says "TER") and that the first residue read in is "tail only" and the last residue is "head only". The "TER" separator ends the chain and begins a new one. It is important to remember this when initially creating the input files for your simulation. In our case NAB automatically added the TER cards for us.

Let's take a look at our DNA model. The first step is to start up the graphical version of LEaP:

```
$AMBERHOME/bin/xleap -s -f $AMBERHOME/dat/leap/cmd/leaprc.DNA.bsc1
```

This should load ***xleap*** and you should see something like this appear:




A quick note on the command line used to start ***xleap***: The command line shown above contains a couple of options which are worthy of comment at this point. When ***xleap*** loads, it initially has to open a series of library and parameter files that define the force field parameters to be used and the residue maps etc. Since AmberTools ships with a range of different force fields, each suited to different types of simulation, it is important to tell ***xleap*** which force field we wish to use. This is what the command line options used above are for. The ***"-s"*** switch tells ***xleap*** to ignore any user defined defaults which might otherwise override our selection, while the **"-f $AMBERHOME/dat/leap/cmd/leaprc.DNA.bsc1"** switch tells ***xleap*** to execute the start-up script for the BSC1 DNA force field parameters. This script contains commands that cause ***xleap*** to load all of the configuration files required for the BSC1 force field. If you look in the **$AMBERHOME/dat/leap/cmd/** directory you will find a number of different **leaprc** files such as **leaprc.protein.ff14SB** (Amber FF14SB protein force field), **leaprc.water.tip3p** (parameters for TIP3P water/monovalent ions) etc.

***XLEAP MENU BUG***: Note if you find that the menu's in ***xleap*** are not working please check that your numlock light is turned off. For some reason having numlock on prevents the ***xleap*** menus from operating correctly.

Older force fields (FF99SB and older) used values for ion parameters that have been found to be inaccurate, in part because they were not water type specific. Amber now uses parameters for ions that are specific to the water type used for solvation. More detail can be found in Section 3 of the Amber manual. While parameters were previously sourced by default, we must load the ion parameters for TIP3P water for newer force fields, using the command:

```
source leaprc.water.tip3p
```

We can now load a pdb into ***xleap*** using the **loadpdb** command. This will create a new *unit* in ***xleap*** and load the specified pdb into that unit. We can then subsequently view and edit the new unit using the **edit** command.

So, to load the pdb file we just created into a new unit called "dna1" type the following in the ***xleap*** window (make sure the ***xleap*** window is highlighted and your mouse cursor is within the window):

```
dna1= loadpdb "nuc.pdb"
```

This should result in output, similar to the following, appearing in the ***xleap*** window:

```
> dna1=loadpdb "nuc.pdb"
Loading PDB file: ./nuc.pdb
  total atoms in file: 638
>
```

To look at the structure, in ***xleap***, use the **edit** command and specify the unit you wish to edit, in this case the one we just created "dna1":

```
edit dna1
```

This should open the ***xleap*** editor window and display the molecule.

Within this window the LEFT mouse button allows atom selection (drag and drop), the center mouse button rotates the molecule and the RIGHT mouse button translates it. If the center button does not work you can simulate it by holding down the *Ctrl* key and the LEFT mouse button. To make the image larger or smaller, simultaneously hold down the center and RIGHT buttons (or *Ctrl* and RIGHT) and move the mouse up to zoom in or down to zoom out.

If you played with selecting atoms using the left mouse button you can deselect a region by holding down the *Shift* key while drawing the selection rectangle. To select everything, double click the LEFT button (and to deselect, do the same while holding down the *Shift* key).

Take a look at the structure. You should be able to see the perfect symmetry of canonical B-form geometry DNA. The perfect symmetry of canonical duplexes is based on analysis of long fibers of DNA. Real nucleic acids don't necessarily adopt this perfect symmetry as will become apparent once we start to carry out molecular dynamics on this 10-mer.

## 2.2) What level of simulation am I going to attempt?

Once you have got a suitable model structure the next step is to decide what level of simulation realism is to be used. The complexity of the calculation centers on the evaluation of the pairwise non-bonded and Coulombic interactions. Extra complexity is introduced by using periodic boundaries and Ewald methods to treat long ranged electrostatics and/or by evaluating non-additive effects such as induced polarization.

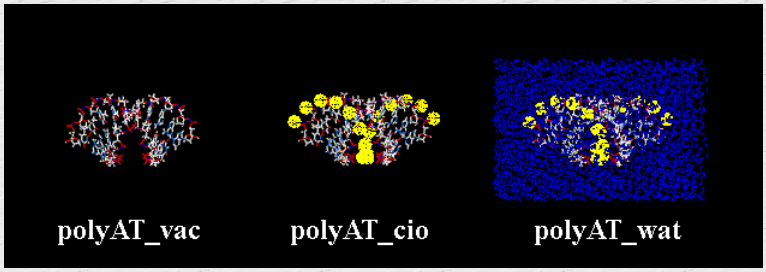
Water is an integral part of nucleic acid structure and thus some representation of solvent effects is fairly critical. Simulations *in vacuo* have been performed where the screening of solvent is modeled by distance dependent or sigmoidal dielectric functions (the latter of which is not implemented in AMBER 10.0). Additionally tricks have been applied to keep the base pairs from fraying, through the addition of Watson-Crick base pair restraints and the reduction of the charges on the phosphate groups. Newer versions of AMBER (6.0 and above) contain the generalized Born model for implicit solvation which, though more expensive, provides a much better implicit solvent representation than simply using a distance dependent dielectric constant.

However, even with advances in computer power and methodological improvements, such as the application of Ewald methods, which allow routine simulations of nucleic acids with explicit solvent and counterions in the nanosecond time range, there is still dependence of the results on the molecular mechanical force field. It is therefore important to understand the inadequacies of the force field being used.

Now, back to the point of this discussion, if you can afford it, include the solvent and the explicit net-neutralizing counterions. Also pay attention to the force field applied and be aware of its limitations. Use methods which properly treat long range electrostatic interactions, such as Ewald methods. However, remember that adding explicit water is expensive. While a nanosecond or so of *in vacuo* DNA simulation can take only minutes on a 3GHz P4, adding a periodic box of water that surrounds the DNA by roughly 10 angstroms, extends the simulation to several days. Given that it is normally necessary to run the simulation a couple of times (due to errors, sampling issues, etc.), these simulations can get very costly.

### 2.2.1) The types of simulations to be run in this tutorial

For the purpose of this tutorial we will build 3 different DNA models. The first will be an *in vacuo* model of the poly(A)-poly(T) structure (named polyAT_vac), an *in vacuo* model of the poly(A)-poly(T) structure with explicit counterions (named polyAT_cio), and a TIP3P (water) solvated model of the poly(A)-poly(T) structure in a periodic box (named polyAT_wat). The *in vacuo* model will be applied in simulations to get a feel for MD and then the solvated model will be used for periodic boundary simulations using a particle mesh Ewald treatment. The *in vacuo* model with the explicit ions will not be used for simulation but it is a good idea to build it in case it is needed for later analysis.




In order to simplify post simulation analysis of the trajectories it is useful to have all three sets of **prmtop** files. This is because often in the analysis of the trajectory displaying the solvent is not normally necessary and the visualization packages will run much faster if the solvent is removed from the trajectory file before loading. Obviously the water is necessary for calculating radial distribution functions, analyzing water structure, and other properties, however it isn't necessary for calculating helicoidal parameters, determining average structures, etc. Therefore, to minimize disk space usage, and quicken the analysis, we often strip the water and/or counterions. The three separate **prmtop** files are useful to have around since you need to use a **prmtop** that matches the structure of your (possibly stripped) trajectory for programs such as *cpptraj*, *VMD* etc.

## 2.3) Building the `prmtop` and `inpcrd` files

Now that we have a starting pdb (***nuc.pdb***) and an understanding of some of the issues surrounding different types of classical MD simulations, we are ready to start building the input files necessary for the MD engine in AMBER 15, ***sander***.

The first step is the building of residues. Many proteins contain coenzymes as well as standard amino acids. These coenzymes are not normally pre-defined in the AMBER database and so are considered to be non-standard residues. It is necessary to provide structural information and force field parameters for all of the non-standard residues that will be present in your simulation before you can create the ***sander*** input files. Fortunately, if you are using standard nucleic acid or amino acid residues, as we will be in this tutorial, this step is not necessary since all of the residues are pre-built in the AMBER database. Later tutorials will cover what to do if you have non-standard residues.

### 2.3.1) LEaP

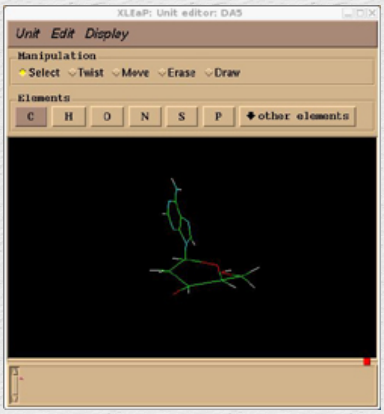
If ***xleap*** is not already running, start it up and load ion parameters again.

```
$AMBERHOME/bin/xleap -s -f $AMBERHOME/dat/leap/cmd/leaprc.DNA.bsc1
source leaprc.water.tip3p
```

In so doing, you should have seen that the appropriate libraries were loaded, as specified by the -f flag.

To see if you have the residues we need for this example, you can check to see if the residues have been properly loaded by trying to edit the unit DA5, i.e. in ***xleap***:

```
edit DA5
```




If the edit command opens a window showing something like this, then the residues were loaded correctly.

Close the window by selecting Unit -> Close from the top menu [Do **not** use the X button as it will quit all of XLEaP]. (If the menu's don't work turn off NumLock). You can also list all the defined residues in ***LEaP*** by typing "**list**". You should see something like the following:

```
> list
AG          AL          Ag          BA          BR          Be          CA          CD
CE          CHCL3BOX    CL          CO          CR          CS          CU          CU1
Ce          Cl-         Cr          DA          DA3         DA5         DAN         DC
DC3         DC4         DC5         DCN         DG          DG3         DG5         DGN
DT          DT3         DT5         DTN         Dy          EU          EU3         Er
F           FB3         FB3BOX      FB4         FB4BOX      FE          FE2         GD3
H3O+        HE+         HG          HOH         HZ+         Hf          IN          IOD
K           K+          LA          LI          LU          MEOHBOX     MG          MN
NA          NH4         NI          NMABOX      Na+         Nd          OPC         OPCBOX
PB          PD          PL3         POL3BOX     PR          PT          Pu          QSPCFWBOX
RB          Ra          SM          SPC         SPCBOX      SPCFWBOX    SPF         SPG
SR          Sm          Sn          T4E         TB          TIP3PBOX    TIP3PFBOX   TIP4PBOX
TIP4PEWBOXTIP5PBOX      TL          TP3         TP4         TP5         TPF         Th
Tl          Tm          U4+         V2+         WAT         Y           YB2         ZN
Zr          parm10
>
```

**Note**: for a list of available ***xleap*** commands type "help" in the main ***xleap*** window. For help on a specific command type "**help *command***". E.g. for help with loadpdb you should get the following on typing "**help loadpdb**":

```
> help loadpdb
variable = loadPdb filename
STRING _filename_
```

Load a Protein Databank format file with the file name _filename_.

The sequence numbers of the RESIDUEs will be determined from the order of residues within the PDB file ATOM records. For each residue in the PDB file, LEaP searches the variables currently defined for variable names that match the residue name. If a match is found, then the contents of the variable are copied into the UNIT created for the PDB structure. If no PDB `TER' card separates the current residue from the previous one, a bond is created between the connect1 ATOM of the previous residue and the connect0 atom of the new one. As atoms are read from the ATOM records, their coordinates are written into the correspondingly named ATOMs within the residue being built. If the entire residue is read and it is found that ATOM coordinates are missing, then external coordinates are built from the internal coordinates that were defined in the matching UNIT (residue) variable. This allows LEaP to build coordinates for hydrogens and lone pairs which are not specified in PDB files.

Now, let's go back to the beginning and assume everything was set up properly; the distraction above was simply to give you a little insight to what goes on behind the scenes... Remember, using software like it is a black box is *dangerous*, especially in research.

Following this rather long aside, we can now return to where we were before. Remember that we had loaded our model:

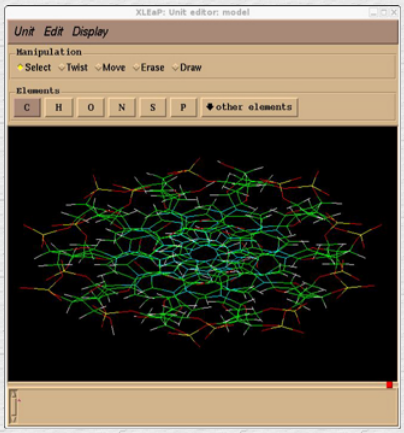
```
dna1 = loadpdb "nuc.pdb"
```

To look at the newly created "unit" named "dna1" type:

```
edit dna1
```

---

This should cause the unit editor window to pop up as shown below:




---

To create our first **prmtop** and **inpcrd** files, we simply issue the following command in the main ***xleap*** window:

```
saveamberparm dna1 polyAT_vac.prmtop polyAT_vac.inpcrd
```

This should give you the following output (the warning concerns the fact that we did not neutralize our system - more on this later):

```
> saveamberparm dna1 polyAT_vac.prmtop polyAT_vac.inpcrd
Checking Unit.
WARNING: The unperturbed charge of the unit: -18.000000 is not zero.

 -- ignoring the warning.

Building topology.
Building atom parameters.
Building bond parameters.
Building angle parameters.
Building proper torsion parameters.
Building improper torsion parameters.
 total 110 improper torsions applied
Building H-Bond parameters.
Not Marking per-residue atom chain types.
Marking per-residue atom chain types.
 (no restraints)
```

and create the following two files:

- *polyAT_vac.prmtop*
- *polyAT_vac.inpcrd*

To remind you about these files:

- **prmtop**: The parameter/topology file. This defines the connectivity and parameters for our current model. This information is static, or in other words, it doesn't change during the simulation. The prmtop we created above is called **polyAT_vac.prmtop**.
- **inpcrd**: The coordinates (and optionally box coordinates and velocities). This is data is not static and changes during the simulations (although the file is unaltered). Above we created an initial set of coordinates called **polyAT_vac.inpcrd**.

Now we want to create a topology that has explicit net neutralizing counterions. There are a number of different ways to add ions to a structure. In this example we shall use the addions command implemented in ***xleap***. This method works by constructing a Coulombic potential on a 1.0 angstrom grid and then placing counterions one at a time at the points of lowest/highest electrostatic potential. The command to do this is as follows (the '0' means 'neutralize'):

```
addions dna1 Na+ 0
```

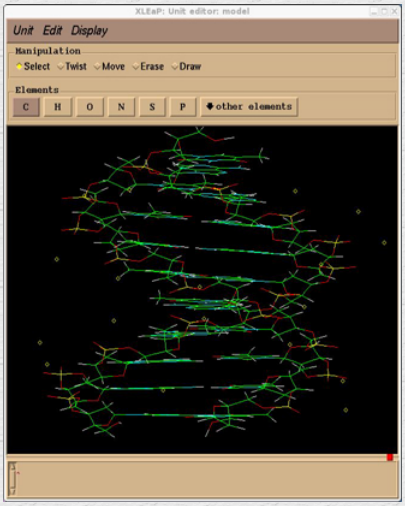
This should add a total of 18 sodium anions to counteract the -18 charge of the DNA chain. (*Note: The command works on integers and so if your system charge is -17.999 then it will only add 17 counterions. In this case the simple solution is to explicitly specify the number of counter ions you need in place of the zero.*)

The output from this command should be similar to the following:

```
> addions dna1 Na+ 0
18 Na+ ions required to neutralize.
Adding 18 counter ions to "dna1" using 1A grid
Grid extends from solute vdw + 3.65  to  9.75
Resolution:      1.00 Angstrom.
grid build: 0 sec
 (no solvent present)
Calculating grid charges
charges: 1 sec
Placed Na+ in dna1 at (6.44, 3.95, 17.79).
Placed Na+ in dna1 at (5.44, -5.05, 10.79).
Placed Na+ in dna1 at (-10.56, 5.95, 13.79).
Placed Na+ in dna1 at (-10.56, -6.05, 19.79).
Placed Na+ in dna1 at (-1.56, 11.95, 9.79).
Placed Na+ in dna1 at (-10.56, -4.05, 6.79).
Placed Na+ in dna1 at (-6.56, 4.95, 27.79).
Placed Na+ in dna1 at (11.44, -8.05, 22.79).
Placed Na+ in dna1 at (0.44, -12.05, 13.79).
Placed Na+ in dna1 at (11.44, 7.95, 10.79).
Placed Na+ in dna1 at (1.44, 11.95, 19.79).
Placed Na+ in dna1 at (10.44, -9.05, 4.79).
Placed Na+ in dna1 at (-7.56, 7.95, -0.21).
Placed Na+ in dna1 at (-11.56, -8.05, 27.79).
Placed Na+ in dna1 at (13.44, 1.95, 24.79).
Placed Na+ in dna1 at (-2.56, -12.05, 23.79).
Placed Na+ in dna1 at (-10.56, 8.95, 21.79).
Placed Na+ in dna1 at (13.44, 0.95, 3.79).

Done adding ions.
```

Note: you should always check that the number of ions you were expecting have actually been added. It is also a good idea to view the new structure to ensure that the charges have been placed as intended. By editing the "dna1" we can see where the ions have been added:

```
edit dna1
```




The DNA1 unit with ions shown as little diamonds.

Now we are once again ready to write the **prmtop** and **inpcrd** files, this time for our neutralized system:

```
saveamberparm dna1 polyAT_cio.prmtop polyAT_cio.inpcrd
```

Output files: *polyAT_cio.prmtop*, *polyAT_cio.inpcrd*

The final input files to create are for solvated DNA with explicit counterions. We have our "dna1" unit already built with counterions so the next step is to solvate it with explicit water. This is done with the command "solvatebox". For our DNA, we will put an 8 angstrom buffer of TIP3P water around the DNA in each direction. In this way all atoms in the DNA starting structure will be no less than 8 angstroms from the edge of the water box. Before we do this, however, for reasons that will become clear later we should create a copy of our dna1 and call it dna2:

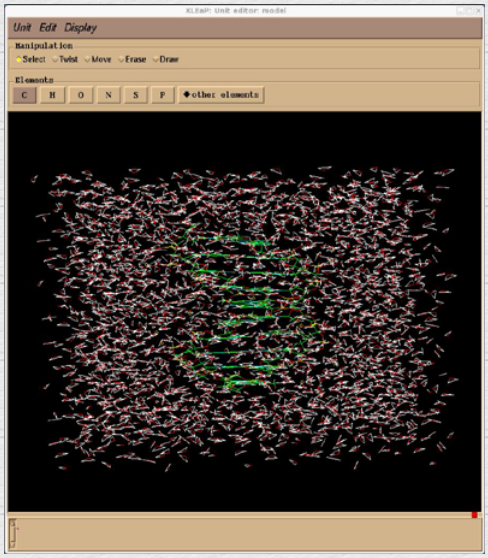
```
dna2 = copy dna1
```

The following creates a rectangular box of water around the DNA:

```
solvatebox dna1 TIP3PBOX 8.0
```

This results in the following output (exact numbers may be slightly different due to round off differences between different computer architectures), editing the "dna1" (**edit dna1**) should show you the DNA in a water box:

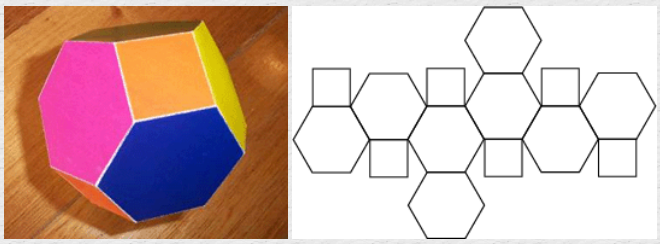
```
> solvatebox dna1 TIP3PBOX 8.0
  Solute vdw bounding box:              27.738 26.738 40.099
  Total bounding box for atom centers:  43.738 42.738 56.099
  Solvent unit box:                     18.774 18.774 18.774
  Total vdw box size:                   46.743 45.963 58.910 angstroms.
  Volume: 126564.801 A^3
  Total mass 56296.124 amu,  Density 0.739 g/cc
  Added 2767 residues.

edit dna1
```




The above output and display show us that ***xleap*** added a total of 2767 water molecules to form a rectangular box of 46.7 46 58.9 angstroms (126565.1 angstroms$^3$).

---

The box built by ***xleap*** is not cubic since DNA is a cylindrical molecule. An issue here is that the long axis of DNA could rotate (via self diffusion) such that the long axis was along the short box dimension which will, since this box will be infinitely repeated in space by the periodic boundary method, bring the ends of the DNA near their periodic images. One way to get around this would be to make the box cubic, or 58.9 x 58.9 x 58.9 angstroms, by specifying a list of numbers to the solvateBox command to force this to be cubic. However, this will add significantly more water to the calculation and slow it down tremendously. Alternatively we can use a different shape box of water. While a rectangular box is the obvious choice for tessellating in 3 dimensional space it is not the only shape that can be replicated in 3 dimensions. A more efficient shape to use, in terms of reducing the problem of solute rotation, and the one we will be using for this tutorial, is a truncated octahedron:



A truncated octahedron is a space-filling shape which is "more spherical" than a cube, thus wasting less computation on solvent molecules distant from the solute.

---

To add a truncated octahedral box of water around our DNA we use the **solvateoct** command. Since in the course of this demonstration we have already solvated our "dna1" with a rectangular box of water we shall use the copy we made "dna2". Enter the following in **xleap** to create the water box:

```
solvateoct dna2 TIP3PBOX 8.0
```

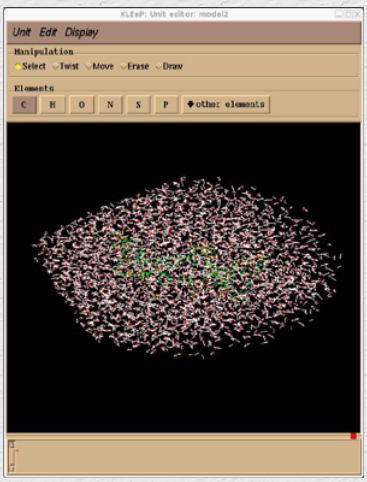
This should give the following output:

```
> solvateoct dna2 TIP3PBOX 8.0
  Scaling up box by a factor of 1.368620 to meet diagonal cut criterion
  Solute vdw bounding box:              27.987 26.927 38.921
```

```
Total bounding box for atom centers:  60.819 60.819 60.819
    (box expansion for 'iso' is  51.9%)
Solvent unit box:                   18.774 18.774 18.774
Volume: 118123.162 A^3 (oct)
Total mass 61286.556 amu,  Density 0.862 g/cc
Added 3044 residues.
```

Editing "dna2" allows you to view the truncated octahedron water box:

```
edit dna2
```




There you have it: an ice cube shaped like a truncated-octahedron.

Once again we save our AMBER **prmtop** and **inpcrd** files:

```
saveamberparm dna2 polyAT_wat.prmtop polyAT_wat.inpcrd
```

Output files: *polyAT_wat.prmtop*, *polyAT_wat.inpcrd*

Now we have our input files we can progress to the next section which introduces running minimization and molecular dynamics..

*CLICK HERE TO GO TO SECTION 3*

(Note: These tutorials are meant to provide illustrative examples of how to use the AMBER software suite to carry out simulations that can be run on a simple workstation in a reasonable period of time. They do not necessarily provide the optimal choice of parameters or methods for the particular application area.)