CANION USER GUIDE
(V3.5, updated October 2016)

CANION is program for analyzing the distribution of ions, solute atoms, or water molecules surrounding (or belonging to) a nucleic acid. The data for CANION can be provided in three formats: (1) a molecular dynamics trajectory that has already been analyzed with Curves+ (*file*.cdi); (2) a density matrix in Gaussian cube format (*file*.cub); (3) a simplified snapshot format with a specified number of ion/atom Cartesian coordinates per snapshot (*file*.pts). We will start by describing how to generate data for Canion using Curves+ (see also the Curves+ user guide) and the options for analyzing this data in Canion. We will then describe the differences when using the other input options. In the following text we will refer only to analyzing "ions", but the same analysis can equally be applied to atoms other than ions, including the oxygen or hydrogen atoms of solvent water molecules, or atoms belonging to the nucleic acid solute molecule or to bound ligands or proteins. The latest version of CANION includes the possibility of using data on local helical axis curvature to look specifically at the inside or outside of curved DNA segments and thus investigate the effect of curvature on the environment. It also adds a sliding window analysis in order to reveal the time-dependent fluctuations of ion populations.

References:

1) Analyzing ion distributions around DNA. R. Lavery, J.H. Maddocks, M. Pasi, K. Zakrzewska Nucleic Acids Res. 42 (2014) 8130
2) Unraveling the sequence-dependent polymorphic behavior of d(CpG) steps in B-DNA. P.D. Dans, I. Faustino, F. Battistini, K. Zakrzewska, R. Lavery, M. Orozco Nucleic Acids Res. 42 (2014) 11304
3) Analyzing ion distributions around DNA: sequence-dependence of potassium ion distributions from microsecond molecular dynamics. M. Pasi, J.H. Maddocks, R. Lavery Nucleic Acids Res.(2015) doi: 10.1093/nar/gkv080

First step: CURVES+ analysis options

The ions to be analyzed in CURVES+ are specified in the file standard_i.lib. Each line contains a name (in single quotes) and a formal charge. Lines starting with # are treated as comments. Here is an example:

```
# Ion library
# Each line gives the ion/atom name (in single quotes) and its
formal charge
# Up to 40 ion/atom types are allowed in this library file
'K+' 1
'Cl-' -1
'P' -1
'C1*' 0
'O' 0
```

Note sugar ring atoms names in Amber input containing quotes are changed to stars by Curves+ (e.g. C1' → C1*). Therefore, only starred sugar atom names are used in standard_i.lib. When analyzing a single structure or a molecular dynamics trajectory, Curves+ will always read the nucleic acid, but the only other atoms read are those specified in the standard_i.lib file. These can include any atoms/ions in the input data, including atoms of the nucleic acid or of any molecule interacting with the nucleic acid.

To analyze ion positions, use the Curves+ namelist option ions=.t. (see Curves+ user guide). For each snapshot composing a trajectory, Curves+ calculates a helical axis and then determines the position of the ions with respect to this axis. This involves finding the point on the axis that corresponds to the shortest perpendicular distance to the ion. The ion position is then defined by the distance of this point along the axis in units of base pair steps (**D**), the length of the perpendicular vector (**R**) and the angle of this vector with respect to the vector of the local axis frame pointing towards the strand whose 5'-3' direction is aligned with the helical axis (**A**). This choice places the center of the minor groove in B-DNA at roughly at **a** = 90° and the center of the major groove roughly at **a** = 270°. Taken together, **DRA** constitute curvilinear helicoidal coordinates (CHC).

The resulting data is output by Curves+ in a .cdi file containing the CHC of each ion in each snapshot. Note that ions will only be analyzed within a radius of 30 Å from the helical axis of the nucleic acid (beyond which the nucleic acid has little influence on their distribution). Ions lying beyond the ends of the helical axis are similarly ignored (a necessary choice given the CHC used to analyze ion distributions)

Analyzing the CHC data in Canion also requires a set of reference frames describing an average helical axis. This can be obtained by generating an average nucleic acid structure for the trajectory (for example, using ptraj in Amber Tools) and analyzing this structure in Curves+ using the option axfrm=.t. (see Curves+ user guide). This will generate an .afr file containing helical axis frames at each base level of the nucleic acid (and also the data on local curvature). Alternatively, Canion can generate a uniform linear helical axis internally, using helical rise and twist values set by the user (see below). Note that in this case filtering on the basis of curvature will naturally have no effect.

CANION analysis using data from Curves+

CANION reads the ion CHC data from the .cdi file generated by Curves+. It optionally reads an average helical axis (generated as a set of frames by Curves+ as described above). The main options of Canion are controlled with a namelist input. All namelist parameters have default values (see below) that will be used unless new values are given in the namelist section of the input (i.e. between "&inp" and "&end").

CANION namelist variables

Name	Type	Default	Meaning
axfrm	character		I/P .afr file name defining the helical axis
dat	character		I/P file with extension .cdi, .cub or .pts
lis	character		O/P .lis list file
prop	character		I/P **prop**.ser file for filtering data (see below)
seq	character	* (i.e. all)	base sequence to be analyzed (composed of upper case letters)
seqin	character		For cube/points I/P, give 1st strand sequence of corresponding DNA
solute	character		I/P pdb file containing reference DNA conformation for calculating solute volume
type	character	* (i.e. all)	ion type to be analyzed, using names as given in standard_i.lib
clim	real	0.4	minimal value of curvature to distinguish curved DNA from straight DNA (equivalent to a radius of curvature of 100 Å)
grid	real	1.0 Å	Grid spacing for 3D distribution
sris	real	3.38 Å	standard rise for helical axis if no I/P .afr file
stwi	real	34.5°	standard twist for helical axis if no I/P .afr file
alow	real	0.0°	Lower **A** angle sampling limit
ahig	real	360.0°	Upper **A** angle sampling limit
dlow	real	1.0 bp	Lower **D** bp sampling limit (1 → nlev)
dhigh	real	500.0 bp	Upper **D** bp sampling limit (1 → nlev)
rlow	real	0.0 Å	Lower **R** radius sampling limit
rhig	real	30.0 Å	Upper **R** radius sampling limit
pmin	real	-500	minimum filtering limit using I/P .ser data
pmax	real	500	maximum filtering limit using I/P .ser data

Name	Type	Default	Meaning
itst	integer	0	First snapshot to analyze
itnd	integer	0	Last snapshot to analyze
itdel	integer	1	Spacing of snapshots to analyze
istep	integer	0	If >0 O/P total number of ions every isteps to **lis**.stp file
iwin	integer	0	if iwin > 0 a sliding window average of the population is generated in a .win file. The maximum allowed value of iwin is 100,000
acvol	logical	.f.	Set **true** to generate an _S.cub file showing the volume used for ion analysis (Note that this option cancels any other ion analysis)
circ	logical	.f.	Set **true** for minicircle analysis
rmsf	logical	.f.	Set **true** for rmsf calculation per "ion"
series	logical	.f.	Set **true** for series output
dbrac	logical	.f.	read dlow, dhigh "brackets" after namelist I/P
ins	logical	.t.	Set **false** to ignore the inside face of curved DNA (and all straight DNA).
outs	logical	.t.	Set **false** to ignore the outside face of curved DNA (and all straight DNA).

Important notes:
(1) namelist input starts with &inp and ends with &end
(2) do not put quotes around character variables
(3) file names can contain up to 128 characters

Further explanations

1) Selecting the ions to analyze:
type selects the ions (or atoms) to be analyzed. It can be a name (corresponding to one of the entries in the standard_i.lib file) used during the Curves+ analysis, or a number (1, -1 or 0 zero - meaning analyze all cations, anions or neutral species), or "*" meaning analyze all available data.

2) Selecting the sequence elements to analyze:
seq selects base sequences to analyze, it contains a character string made up of A, G, C, T, R, Y, S, K, W, M or *. (Note: R=A/G, Y=C/T, S=G/C, K=G/T, W=A/T, M=A/C). The I/P sequence will be searched for any occurrences of this string. If the string has an odd number of characters, the analysis is limited to the central base pair of the string (extending ½ a base pair step in each direction). If the string has an even number of characters, the central base pair step is analyzed. **seq**=* will analyze the whole I/P oligomer. Note that the **seq** string is searched for in both strands. If it occurs in the second strand the 2D results are dyad inverted so they are compatible with first strand results. Which axis segments of the nucleic acid finally are analyzed are indicated in the

Canion .lis file. The example below shows a fragment of the Canion output for an sample oligomer using seq=TT:

```
Preselect = C      T      T      C      T      A      T      A      A
            |------|^^^^^^|------|------|------|------|------|vvvvvv|vvv

            A      A      G      G      C      T      G
          vvv|vvvvvv|------|------|------|------|------|
```

- for this oligomer "TT" selects four base pair steps, one in the 1st strand ('^') and three in the 2nd strand 'v'). Each step is divided into six subdivisions.

3) Selecting the oligomer fragment to analyze:
dlow, **dhig** can also be used to select a segment of the oligomer, between base pair levels 1 and nlev (where nlev is the number of base pairs in the oligomer). With the default values (**dlow**=**dhig**=0) the entire oligomer is treated. **dlow** and **dhig** (where dlow < dhig) can bet set to any real number between 1 and nlev. Note that **seq** and **dlow/dihig** can be used together, but be careful to check the Canion output fille to see what has actually been selected. If you need to analyze several segments of an oligomer in one Canion run, set **dbrac**=.t. and read sets of **dlow**, **dhigh** values (one pair per line) after the namelist input.

4) Selecting the radial and angular ranges to analyze:
rlow and **rhig** fix the minimal and maximal distances from the helical axis taken into account in the analysis. It may be useful to note that in canonical B-DNA the phosphorus atoms are roughly 10.25 Å from the axis and rhig=10.25 is generally used to limit the analysis to the grooves of the double helix.
alow and **ahig** fix the minimal and maximal angle range to analyze. Note that angle around the helical axis are measured from 0° to 360°, where 0° corresponds to the strand whose 5'-3' direction is aligned with the helical axis in canonical B-DNA and 90° to the center of the minor groove. In canonical B-DNA, the phosphorus atoms lie at 33° and 147°. If **alow** ≤ **ahig**, angles between these values are accepted, i.e. **alow**=33° and **ahig**=147° defines the minor groove. If **alow** > **ahig**, angles outside the **ahig** to **alow** range are accepted, i.e. **alow**=147° and **ahig**=33° defines the major groove (and accepts angles 0°-33° and 147°-360°).

5) Selecting the snapshots to analyze:
itst, itnd, itdel enable the I/P data to be filtered in order to study part of the trajectory or to increase the spacing between the snapshots analyzed. **itst** sets the first snapshot analyzed, **itnd** sets the last snapshot and **itdel** sets the spacing between analyzed snapshots. Setting **itst**=n (n> 0) and leaving **itnd**=0, will analyze only snapshot number n. If itst=0 and itnd=0, itst is set to 1 and itnd is not limited.

6) Filtering the analysis on the basis of curvature:
The variable **clim** divides your DNA into two regions: curved if the curvature of the corresponding base pair steps is > **clim** and straight if it is ≤ **clim**. Note curvature is defined so that 1.0 corresponds to a radius of curvature of 40 Å, which is roughly the curvature of DNA within a nucleosome. The default value of **clim** is 0.4, corresponding

to a 100 Å radius of curvature. The variables **ins** and **outs** then let you filter certain zones of DNA: (1) **ins** = .t. / **outs** = .t. is the default and leads to no filtering based on curvature; (2) **ins** = .t. / **outs** = .f. will only analyze the inside face of curved DNA segments and will ignore straight DNA; (3) **ins** = .f. / **outs** = .t. will only analyze the outside face of curved DNA segments and will ignore straight DNA; (4) **ins** = .f. / **outs** = .f. will only analyze straight DNA segments. Within curved DNA the inside face is defined as the angle range ±90° with respect to the vector indicating the local direction of curvature, the remaining angular space corresponds to the outside face. Curvature filtering is summarized in the table below:

Curved DNA (curv ≥ **clim**)	**ins**	**outs**	Straight DNA (curv < **clim**)
All	**.T.**	**.T.**	All
Inside face only	**.T.**	**.F.**	None (only curved)
Outside face only	**.F.**	**.T.**	None (only curved)
None	**.F.**	**.F.**	All (only straight)

7) Filtering the analysis on the basis of existing data:
The namelist options **prop, pmin, pmax** trigger reading a **prop**.ser file from a Canal analysis (which should contain 1 line of data per snapshot, with one data value per base pair level in the oligomer being analyzed). **iprop** determines the base pair level to be used for filtering. If, for a given snapshot, the corresponding data value falls outside the pmin/pmax limits the ion data is not taken into account. This option makes it possible to correlate ion distributions with given helical or backbone parameter states.

8) Root mean square fluctuations:
Setting **rmsf**=.t. uses the combination of the ion CHC and the average helical axis to map the ions into Cartesian space and then calculates their average position (*lis file*_cen.pdb) and their root mean square fluctuation values (*lis file*.rmsf). A single pass rmsf algorithm makes this calculation possible with a single read of the trajectory file. This option is generally used for solute atoms and not for solvent molecules or ions.

9) **circ**:
Should be set to .t. when minicircles are analyzed. In this case, seq can be used to search sequences that overlap the minicircle junction (base pairs N and 1).

10) **istep**:
Outputs the average ion population (in a **lis**.stp file) every **istep** snapshots. This is useful for calculating the convergence of an ion population in a volume specified by a combination of dlow/dhig/rlow/rhig/alow/ahig values. **istep** is not compatible with .cub input data.

11) **iwin**: Outputs the ion population calculated as a sliding window average (using a width of **iwin** snapshots) into a **lis**.win file. While **istep** is a good guide to convergence, **iwin** provides information on the time fluctuation of populations.

12) **seqin**:
Inputting the first strand base sequence of the oligomer analyzed via **seqin** is necessary when reading .cub or .pts data.

13) **Series output:**
If **series** is true, the snapshot number and the number of ions counted for the snapshot is output in a file (**lis**.cser). This option is generally useful when a limited zone of space is analyzed.

14) Taking **solute** volume into account:
By reading a pdb file containing the nucleic acid fragment (and nothing else), it is possible to take into account the space occupied by the van der Waals volume of the nucleic acid and to eliminate this volume when calculating molarities. This gives a better view of the balance of ion densities between the major and minor grooves, where the space occupied by the nucleic acid is different. Note that the solute conformation should be consistent with the **axfrm** data describing its helical axis.
We find that for a trajectory analysis it is generally best to use the average structure obtained after superposing the snapshots along the trajectory (rather than using the snapshot with the smallest rmsd to this average structure).

15) **acvol** is a new option that creates a _S.cub file (Gaussian cube format) that can be used to visualize the volume of space around a nucleic acid that is being analyzed for ion distributions. If **acvol** is .true. the file is created, but no further ion analysis is performed. You can use this option to test your choice of alow/ahig/dlow/dhig/rlow/rhig and seq to check if you are targeting the correct regions. If so, set acvol=.f. and re-run the script to perform the desired analysis.

Other input data options

There are two alternatives to reading .cdi data from Curves+

1) .pts input
This is a simplified input where a single file contains an unlimited number of snapshots. Each snapshot starts with the number of ions to be read (format i10). This is followed by the Cartesian coordinates of the ions as a list of x,y,z values (format 3f8.3). Note that this input does not allow more the one type of ion to be read and so **type** should be left at its default value "*".

2) .cub input
This allows a cubic density matrix to be read in Gaussian cube format. The format begins with two title lines (any text), then a line specifying the x, y, z coordinates of the lower corner of the cubic space (format 4x, 3f10.3) and three lines specifying the number of points along each axis (respectively, ixd, iyd, izd) and a unit vector defining the axis (format i4, 3f10.3). The density matrix *his* is then read in free format:

```
do i=1,ixd
do j=1,iyd
read(2,*) (his(I,j,k), k=1,izd)
enddo
enddo
```

Output files:

CANION analyzes ion/atom distributions in 1D, 2D and 3D

1D) histograms of distributions with respect to **d**, **r**, and **a** (in files .d, .r and .a). In the **r** histogram, each bar is divided by **r** to correct for the increasing radial volume being analyzed. Expressed in molarity.

2D) histograms of the distributions with respect to **dr**, **da** and **ra** (in files .dr, .da and .ra). The **dr** distribution is corrected for increasing radial volume as described above. The **ra** distribution is plotted in polar coordinates. Expressed in molarity.

3D) an ion/atom density matrix is generated with respect to the average (or standard) helical axis using **grid** to set the spacing. Note that you can use any axis for this analysis (as long as it has the correct length). For example, you could analyze a minicircle using a straight axis. The output format is a Gaussian density matrix, expressed in units of molarity.

.ria) Accumulated ion population as a function of radius **r**.

.stp) Average ion population at chosen time steps (see namelist option **istep**).

.cser) number of ions in chosen volume at each snapshot (see namelist option **series**).

Sample I/P file for CANION:

```
rm Ant_KG.*
/Users/rl/Code/util/canion <<!
 &inp dat=Antion4.cdi,axfrm=caver,solute=aver,
 lis=Ant_KG,seq=G,type=K+, &end
!
```

- this run analyzes potassium ions around GC base pairs (using the ion position data in Antion4.cdi and the average DNA conformation in aver.pdb, with corresponding helical axis frames in caver.afr)

Current size limits and other fixed parameters

N1) Base pair levels	200
N2) Histogram bins	1200
N3) Number of ion/atom types	50
N4) Ions read per snapshot	15000
N5) points in axis frames	45000
N6) 3D histogram bins	501
N7) Number of ion/atom types for rmsf	500
N8) Number of atoms in solute PDB	8000

These limits can be changed by modifying the parameter statements in the main program or in the appropriate subroutines. Any changes must be made consistently throughout the program:

Main program	n1, n2, n3, n4, n5, n6, n7
dotdelta	n1, n5
screw	n5
intaxe	n1, n2, n5
solvol	n1, n2, n5, n8

Other parameters fixed in the program:

shell	30 Å, radial limit of the CHC analysis
spa	0.2, grid spacing per degree along **A**
kpd	6, grid spacing per bp step along **D**
kpr	2, grid spacing per Å along **R**
nspl	200, spline interpolation points along axis per bp step

Appendix: MatLab scripts for analysis

MatLab offers a convenient way to display output from CANION. Here are some functions for plotting 1D and 2D data. For 3D data, use molecular graphic programs such as CHIMERA (http://www.cgl.ucsf.edu/chimera/) to read the .cub files output by Canion.

1D plot: using Matlab command csing('file.r',[slen])
- where the file extension can be .r, .d or .a. The length of the nucleic acid fragment is assumed to be 18, but it can be reset with the optional variable slen

```
function x=csing(fir,slen)
a1=33;
a2=147;
pr=10.25;
siz=18;
   if nargin==2
   siz=slen;
   end
lw=2.0;
head=['CANION 1D:  ',fir];
scr=get(0,'ScreenSize');
figure('Position',[scr(3)/2 scr(4)/2 scr(3)/2 scr(4)/2],'Name',head)
fr=load(fir);
flim=size(fr,1);
plot(fr(1:flim,1),fr(1:flim,2),'k','LineWidth',2)
set(gca,'FontSize',18);
ylabel('Molarity','FontSize',18);
limy=ylim;
    hold on;
    if size(strfind(fir,'.r'),1)==1
    ylim(limy);
    plot([pr;pr],limy,'k','LineWidth',lw);
    xlabel('R (\AA)','Interpreter','Latex','FontSize',18);
    elseif size(strfind(fir,'.a'),1)==1
        ylim(limy);
        xlim([1,360]);
        plot([a1;a1],limy,'-k','LineWidth',lw);
        plot([a2;a2],limy,'-k','LineWidth',lw);
        xlabel('A (degrees)','FontSize',18);
    elseif size(strfind(fir,'.d'),1)==1
        ylim(limy);
        xlim([1,siz]);
        xlabel('D (bp)','FontSize',18);
    end
```

Comparing two 1D plots: using Matlab command csing('file1.r','file2.r', [slen]). Note that the y-axis limit is set by the maximum y-value in file1.r.

```
function x=csing2(fir,sec,slen)
a1=33;
a2=147;
pr=10.25;
siz=18;
   if nargin==3
   siz=slen;
   end
lw=2.0;
head=['CANION 1D:  ',fir];
scr=get(0,'ScreenSize');
figure('Position',[scr(3)/2 scr(4)/2 scr(3)/2 scr(4)/2],'Name',head)
fr=load(fir);
sc=load(sec);
plot(fr(:,1),fr(:,2),'k','LineWidth',2)
limy=ylim;
ylabel('Molarity','FontSize',18);
hold on;
plot(sc(:,1),sc(:,2)-min(sc(:,2)),'--k','LineWidth',2)
    if size(strfind(fir,'.r'),1)==1
    ylim(limy);
    plot([pr;pr],limy,'--k','LineWidth',lw);
    xlabel('R (\AA)','Interpreter','Latex','FontSize',18);
    elseif size(strfind(fir,'.a'),1)==1
        ylim(limy);
        xlim([1,360]);
        plot([a1;a1],limy,'k','LineWidth',lw);
        plot([a2;a2],limy,'k','LineWidth',lw);
        xlabel('A (degrees)','FontSize',18);
    elseif size(strfind(fir,'.d'),1)==1
        ylim(limy);
        xlim([1,siz])
        xlabel('D (bp)','FontSize',18);
    end
```

2D plots: using Matlab command csurf('file',{rlim}) - produces three plots for the planes *DA, DR* and *RA* from the corresponding data files. The optional variable rlim can be used to zoom in on a smaller range of R in the DR and DA plots. rlim must be ≤ rhig, where rhig was the upper limit of the radius used in the corresponding Canion calculations.

```
function x=csurf(fir,rlim)
head=['CANION 2D:  ',fir];
scr=get(0,'ScreenSize');
fda=load([fir,'.da']);
fdr=load([fir,'.dr']);
fra=load([fir,'.ra']);
kpd=6;
kpr=2;
kpa=0.2;
lw=2.0;
pr=10.25;
a1=33;  ar1=a1*pi/180;
a2=147; ar2=a2*pi/180;
dx=size(fda,1);
ax=size(fda,2);
rx=size(fra,1);
dlow=1+0.5/kpd;
dhig=dlow+(dx-1)/kpd;
rlow=0.5/kpr;
rhig=rlow+(rx-1)/kpr;
   if nargin==2
   rhig=rlim;
   rx=round(rlow+kpr*rhig+1);
   fdr=fdr(:,1:rx);
   fra=fra(1:rx,:);
   end
alow=0.5/kpa;
ahig=alow+(ax-1)/kpa;
d=linspace(dlow,dhig,dx);
r=linspace(rlow,rhig,rx);
a=linspace(alow,ahig,ax);

figure('Position',[scr(3)/9, scr(4)/2 scr(3)/2.5
scr(4)/2],'Name',head);
contourf(a,d,fda);
set(gca,'FontSize',18);
colorbar('FontSize',18);
axis square;
title('DA','FontSize',18');
ylabel('D (base pair steps)','FontSize',18);
xlabel('A (degrees)','Interpreter','Latex','FontSize',18);
limy=ylim;
    hold on;
    ylim(limy);
    xlim([1,360]);
    plot([a1;a1],limy,'-w','LineWidth',lw);
    plot([a2;a2],limy,'-w','LineWidth',lw);
figure('Position',[2*scr(3)/9, scr(4)/2 scr(3)/2.5
scr(4)/2],'Name',head);
contourf(r,d,fdr);
set(gca,'FontSize',18);
colorbar('FontSize',18);
axis square;
title('DR','FontSize',18);
ylabel('D (bp)','FontSize',18);
```

```
xlabel('R (\AA)','Interpreter','Latex','FontSize',18);
limy=ylim;
    hold on
    ylim(limy);
    plot([pr;pr],limy,'-w','LineWidth',lw);
figure('Position',[3*scr(3)/9, scr(4)/2 scr(3)/2.5
scr(4)/2],'Name',head);
[theta,r] = meshgrid(linspace(-pi,pi,ax),linspace(rlow,rhig,rx));
contourf(r.*cos(theta),r.*sin(theta),fra);
set(gca,'FontSize',18);
ylabel('Y (\AA)','Interpreter','Latex','FontSize',18);
xlabel('X (\AA)','Interpreter','Latex','FontSize',18);
    hold on
    plot([-cos(ar1)*pr;0;-cos(ar2)*pr],[-sin(ar1)*pr;0;-sin(ar2)*pr],
    '-w','LineWidth',lw);
    plot([0;0],[0;pr],'-w','LineWidth',lw);
    ang=0:0.01:2*pi;
    xp=pr*cos(ang);
    yp=pr*sin(ang);
    plot(xp,yp,'-w','LineWidth',lw);
    axis square;
    axis tight;
    colorbar('FontSize',18);
    title('RA','FontSize',18);
```