

17-30 January 1995 CHR, continuing from "Oseen", 2 November 1994

Use Zimm (1980) formulation to solve hydrodynamics of a given subunit array. Compare various aspects of the Kirkwood/Bloomfield et al. (which I will sometimes call "K/B") approximation.

■ General functions to define interaction tensor, velocity perturbations, etc.

```
Off[General::spell];
Off[General::spell1];

avg[x_] := Apply[Plus, x] / Length[x];

l[r_] := Sqrt[r.r];
rr[r_] := Outer[Times, r, r];
Id[n_] := IdentityMatrix[n];
zero[n_] := Table[0, {n}, {n}];

T[r_] := 1 / (8 Pi nu l[r]) (Id[Length[r]] + rr[r] / l[r]^2);

MatrixForm[T[{x, y, z}]]
```

Velocity and velocity perturbation functions.

```
V[i_, {p_, radius_}] := Id[3] / (6 Pi nu radius[[i]])
dV[i_, j_, {p_, radius_}] := T[p[[j]]] - p[[i]];
u[i_, {p_, radius_}] := {-p[[i, 2]], p[[i, 1]], 0};
z[i_, {p_, radius_}] := {p[[i, 2]], -p[[i, 1]], 0};
```

Demonstrate some velocity profiles

The perturbation of the velocity of the fluid at a given point $r\{x,y,z\}$ with respect to a frictional sphere is given by the following expression for dv for a force $f\{x,y,z\}$ acting on the center.

The viscosity ν has typically the units of poise, or $\text{g}/(\text{cm sec})$. If distances are in cm and force is $(\text{g cm}/\text{sec}^2)$, the velocity given here is cm/sec .

The relative velocity $dv0$ is the relative solvent velocity perturbation about a particle travelling with whatever velocity results from a unit force acting on it, expressed in terms of the distance relative to the particle radius. The factor of the particle radius cancels. A positive value says that the sphere is dragging fluid with it.

```
ClearAll[dv, dv0];
dv[r_, f_] := T[r].f;
dv0[r0_, f0_] := dv[r0, f0] / (6 Pi nu);

dv[{1, 0, 0} cm, {1, 0, 0} g cm/s^2] /. nu -> g / (cm s)

{
  
$$\frac{\text{cm}^2}{4 \sqrt{\text{cm}^2} \text{Pi s}}, 0, 0$$

}

dv0[{1, 0, 0}, {1, 0, 0}]

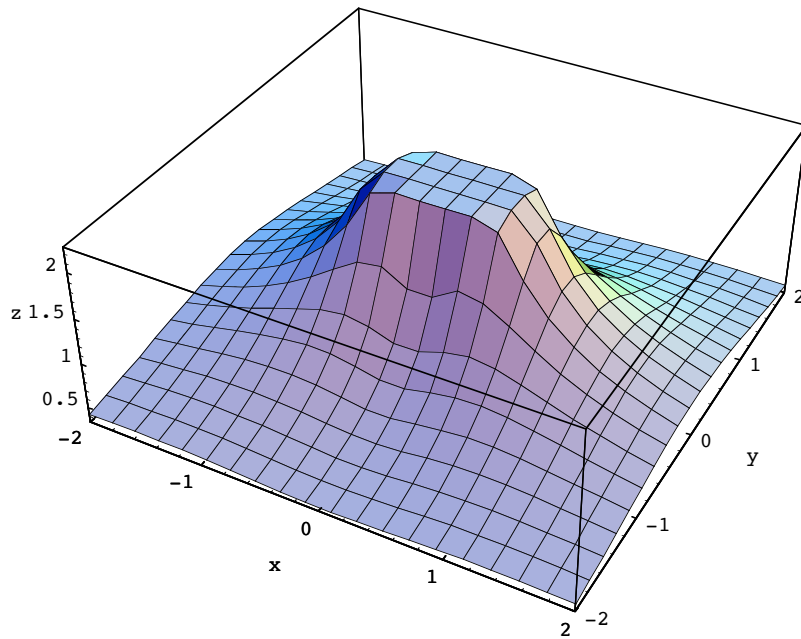
{
  
$$\frac{3}{2}, 0, 0$$

}

dv0[{1.1, 0, 0}, {fx, fy, fz}]

{1.36364 fx, 0.681818 fy, 0.681818 fz}

Plot3D[r = {x, y, 0}; l[N[dv0[r, {1, 0, 0}]]],
  {x, -2, 2}, {y, -2, 2}, PlotPoints -> 20,
  AxesLabel -> {"x", "y", "z"}];
```



■ *Define matrix flatteners to set up the linear hydrodynamics equations:*

```
flatmat[M_]:=Module[{Mt,Mf},
  Mt=Transpose[M,{1,3,2,4}];
  Mf=Table[Flatten[Mt[[i,j]]],{i,Length[Mt]},{j,Length[Mt[[1]]]}];
  N[Flatten[Mf,1]]
];

flatvec[x_]:=Flatten[x,1];

complete[M_,geo_]:=Module[{base,n,npoints,Mc},
  (* Ugly but necessary, since the additional equations required
    to solve the tableau destroy the "tensorness" of the matrix
    of matrices. *)
  dim=Length[M];
  n=dim/3-1;
  Mc=Table[0,{dim+1},{dim+1}];
  Do[ Mc[[i,j]]=M[[i,j]],{i,dim},{j,dim}];
  Do[ base=(i-1)*3;
    Mc[[base+1,dim+1]]=Evaluate[u[i,geo]][[1]];
    Mc[[base+2,dim+1]]=Evaluate[u[i,geo]][[2]];
    Mc[[base+3,dim+1]]=Evaluate[u[i,geo]][[3]];
    {i,n}];
  bottom=Flatten[Table[z[i,geo],{i,n}]];
  Do[ Mc[[dim+1,j]]=bottom[[j]],{j,3*n}];
  Return[Mc]
];
```

Show tensor->matrix flattening in action:

```
M={ {{a1111,a1112},{a1121,a1122}},{a1211,a1212},{a1221,a1222}},
  {{a2111,a2112},{a2121,a2122}},{a2211,a2212},{a2221,a2222}} };

MatrixForm[M]

a1111 a1112 a1211 a1212
a1121 a1122 a1221 a1222

a2111 a2112 a2211 a2212
a2121 a2122 a2221 a2222
```

```
MatrixForm[Transpose[M,{1,3,2,4}]]
```

```
a1111 a1112 a1121 a1122
a1211 a1212 a1221 a1222
```

```
a2111 a2112 a2121 a2122
a2211 a2212 a2221 a2222
```

```
MatrixForm[flatmat[M]]
```

```
a1111 a1112 a1211 a1212
a1121 a1122 a1221 a1222
a2111 a2112 a2211 a2212
a2121 a2122 a2221 a2222
```

■ *Random-orientation matrix definition*

These are from my Nov'94 hydrodynamics program for the random orientation of a chain

```
Ax[w_]:={{1,0,0},
           {0,Cos[w],Sin[w]},
           {0,-Sin[w],Cos[w]}};
Ay[w_]:={{Cos[w],0,-Sin[w]},
           {0,1,0},
           {Sin[w],0,Cos[w]}};
Az[w_]:={{Cos[w],Sin[w],0},
           {-Sin[w],Cos[w],0},
           {0,0,1}};

AO:=Az[Random[Real,N[{0,2 Pi}]]].Ax[Random[Real,N[{0,2 Pi}]]].
    Az[Random[Real,N[{0,2 Pi}]]];
(* Random overall orientation matrix *)

rotate[A_,o_]:=Module[{i,j,depth},
  (* Routine to rotate vectors even when they are assembled
    into groups and lists of those groups. Rotation matrix
    is A, vector object is o *)
  depth=Depth[N[o]];
  Which[ depth==1,Print["Not done: ",depth];Return[o],
         depth==2,Return[N[A].o],
         depth==3,Return[Transpose[N[A].Transpose[o]]],
         depth==4,Return[Map[
           Transpose[N[A].Transpose[#]] &,o ]
         ],
         depth>=5,Print["Not done: ",depth];Return[o]
  ];
];
```

■ *Higher level (automating) routines*

```
element[i_,j_,geo_]:=If[And[i<n+1,j<n+1],
  If[i==j,V[i,geo],dV[i,j,geo]],
  If[i<j,-Id[3],If[i==j,zero[3],Id[3]]]
];
```

```

ClearAll[hydro];
hydro[geo_]:=Module[{},
  (* Set up and solve tableau AC.xc = yc for xc for a given
    rigid set of particles at positions p with radii s and
    for a given viscosity nu (here nu->1). 'geo' is a
    structurecontaining a list of coordinates and an
    associated list of hydrodynamic radii *)
  n=Length[geo[[1]]];
  If[ test,
    Print["Hydrodynamics of an assembly of ",n," spheres"];
    Print[" {x,y,z}    radius"];
    Print[MatrixForm[Transpose[geo]]]
  ];
  A=Table[element[i,j,geo]/.nu->1,{i,n+1},{j,n+1}];
  AA=flatmat[A];
  AC=complete[AA,geo];
  xx=Join[Table[Map[#<>ToString[i] &,{ "fx", "fy", "fz"}],{i,n}],
    {"ux", "uy", "uz", "omega z"}];
  xc=flatvec[xx];
  yy=Join[Table[{0,0,0},{n}],{0,0,1,0}];
  yc=flatvec[yy];
  If[test,Print["Matrix is size: ",Dimensions[AC]]];
  G=Inverse[AC];
  data=Append[data,G.yc];
  If[test,Return[MatrixForm[Transpose[{xc,G.yc}]]]]
];

```

■ **Show relationships between equivalent sphere radius and distance in "lollipop" geometry**

Copy over the constants used for the equivalent sphere calculations.

```

el0=3.4; (* bp length, A *)
persistence=150 el0; (* persistence length, A *)
repeat0=10.4; (* helical repeat of free chain bp/turn *)

NAvo=6.02 10^23;
ro=1.003; (* density, g/cm^3 *)
visc=0.01016; (* viscosity, g/(cm s) or "Poise" *)

mbpa=660; (* basepair mol. weight, g/mole *)
rbpa=1.8294; (* basepair Stokes radius, A *)
vDNA=0.55; (* specific volume of DNA, cm^3/g *)
dDNA=27; (* Yamagawa-Fujii (YF) diam, A *)
switchbp=50; (* point at which we switch from YF
to ellipsoid model, bp *)

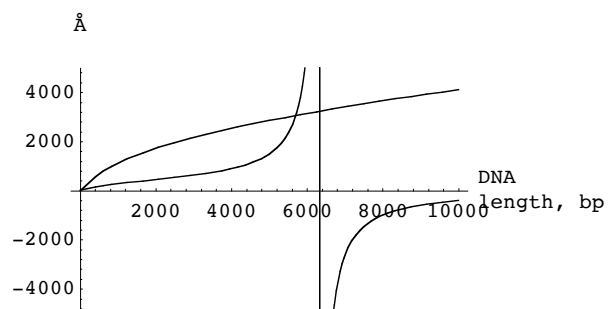
mbpa=660; (* basepair mol. weight, g/mole *)
rbpa=1.8294; (* basepair Stokes radius, A *)
vDNA=0.55; (* specific volume of DNA, cm^3/g *)
dDNA=27; (* Yamagawa-Fujii (YF) diam, A *)
switchbp=50; (* point at which we switch from YF
to ellipsoid model, bp *)

```

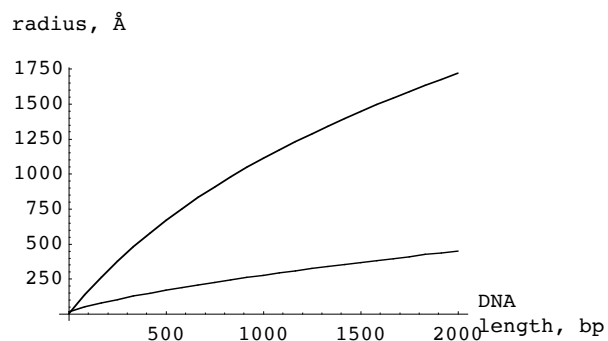
findoffset

Stokes offset 3.00764 A at 50 bp

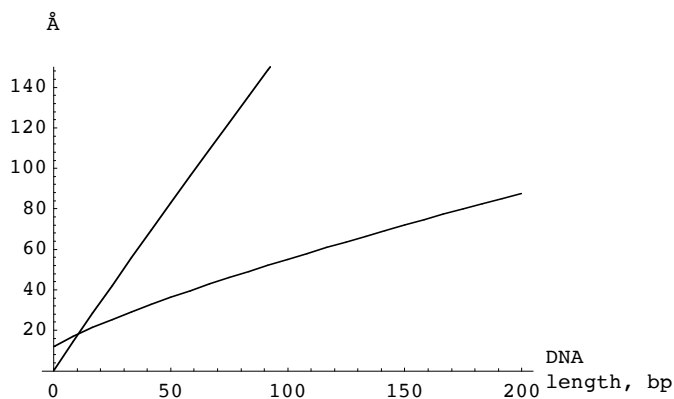
```
Plot[{rootr2[i/2],rbpb[i]},{i,0.1,10000},
PlotRange->{-5000,5000},
AxesLabel->{"DNA\nlength, bp", "Å"}];
```

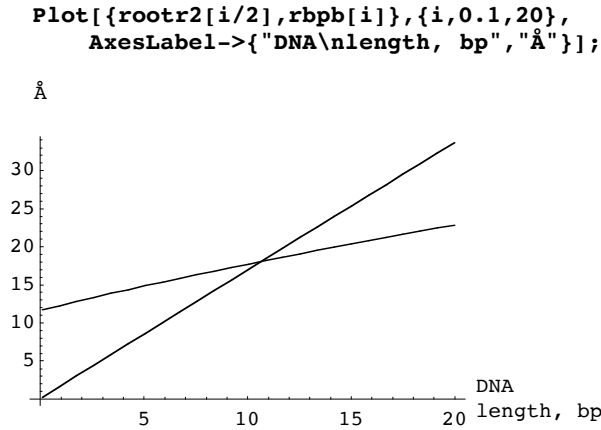


```
Plot[{rootr2[i/2],rbpb[i]},{i,0.1,2000},
AxesLabel->{"DNA\nlength, bp", "radius, Å"}];
```



```
Plot[{rootr2[i/2],rbpb[i]},{i,0.1,200},
PlotRange->{0,150},
AxesLabel->{"DNA\nlength, bp", "Å"}];
```





■ Simulate lollipops to see applicability of Kirkwood/BDvH approximation

I make a two-particle system here that is again like one in my paper. The separation is taken as the center of the array of linker spheres (smaller spheres) from the previous example. First orient perpendicular to sedimentation direction, then nearly parallel (I used a 5 degree angle with the z axis. When the linear assembly is exactly parallel, the set of equations becomes singular).

```
test=True;
data={};
nall={};
radiiall={};
uall={};
fzall={};
fBzall={};
AFzall={};
AFBzall={};

Do[ nbp=N[Exp[i/2.2]];
  points={{0,0,0},{57+rootr2[nbp/2],0,0}};
  radii={57,rbpb[nbp]};

  hydro[{points,radii}];
  ftrue=G.yc;

  ftruez=ftrue[[Range[3,3 n,3]]];
  utruez=ftrue[[3 n+3]];
  AAz=AA[[Range[3,3 n,3],Range[3,3 n,3]]];
  AFz=N[-6 Pi radii AAz.DiagonalMatrix[ftruez]];
  Do[AFz[[i,i]]=N[6 Pi radii[[i]] utruez],{i,Length[AFz]}];

  fbloomz=N[Map[#/Apply[Plus,radii] &,radii]];
  AAz=AA[[Range[3,3 n,3],Range[3,3 n,3]]];
  AFBz=N[-6 Pi radii AAz.DiagonalMatrix[fbloomz]];
  Do[AFBz[[i,i]]=N[6 Pi radii[[i]] utruez],{i,Length[AFBz]}];

  AppendTo[nall,nbp];
  AppendTo[radiiall,radii];
  AppendTo[uall,utruez];
  AppendTo[fzall,N[Apply[Plus,Apply[Plus,Transpose[AFz]]]]];
  AppendTo[fBzall,N[Apply[Plus,Apply[Plus,Transpose[AFBz]]]]];
  AppendTo[AFzall,AFz];
  AppendTo[AFBzall,AFBz],
  {i,-2,9}];
```

Results for orientation 1 (aligned in x direction)

```

MatrixForm[N[Transpose[{nall, radiiall, uall, fzall, fBzall}], 4]]

0.4029      {57., 11.88}    0.0009163      1.          0.934
0.6347      {57., 12.03}    0.0009155      1.          0.9347
1.          {57., 12.28}    0.0009143      1.          0.9358
1.575       {57., 12.66}    0.0009123      1.          0.9376
2.482       {57., 13.24}    0.0009091      1.          0.9403
3.91        {57., 14.14}    0.0009036      1.          0.9446
6.161       {57., 15.48}    0.0008942      1.          0.9509
9.706       {57., 17.49}    0.0008782      1.          0.9597
15.29       {57., 20.46}    0.0008517      1.          0.9707
24.09       {57., 24.8}     0.0008097      1.          0.9823
37.95       {57., 31.09}    0.0007486      1.          0.992
59.79       {57., 40.04}    0.0006697      1.          0.9979

r1plot1=nall;
fBplot1=fBzall;

ListPlot[Transpose[{r1plot1, fBplot1}],
  PlotJoined->True,
  AxesLabel->{"length, bp", "F(eq 4)/F(true)"},
  PlotRange->{0.9, 1}];

```

