Biophysical Journal, Volume 97

Supporting Material

Simulation of cyclic dynein-driven sliding, splitting and reassociation in an outer doublet pair

Charles J. Brokaw

SUPPORTING MATERIAL: METHODS

Details of the computational methods for the case where doublet B is straight

Each doublet is modelled as a sequence of equal-length straight segments, connected by joints at which bending can occur. Joint 1 is at the distal end of segment 1. Computations were routinely performed with a segment length of 120 nm. Each segment has a local z,y coordinate system based at the basal end of the segment, with the segment on the z axis. There is a global Z,Y coordinate system at the base of the B doublet. Segment 1 of doublet B remains on the Z axis, at Y = 0. Segment 1 of doublet A remains parallel to the Z axis, with x = X and Y= doubletseparation, usually 60 nm. (Note that the presentations in Figs. 1,5,6 are inverted relative to the scheme shown here.) The shape of each doublet is represented by an array of curvatures at each joint. For each time iteration, the first step is to use the array of curvatures at each joint to obtain the angular orientation of each segment and the positions of each joint, relative to the global coordinate system. For both doublets, the angle of segment 1 is 0 rad. For 10 equally spaced points on each segment of doublet A, this information is used to calculate the distance, perpendicular to the segment, to the B doublet. These distances are placed in an array ENY and are obtained by calling the following subroutine for each segment:

```
//************************************************************
bool CModel::getBjointForSeg(double* Bangles, double angle, int* Bsegs, ZYPoint* Apoints,
        const int Aseg,ZYPoint* Bpoints, zypoint* bpointarray,
                double& shear, double* ENY,const int lastseg, const double ds,
                const double doubletseparation){
        int Bseg=2;
        double bestdistance = 10000;
        double trydistance;
        zypoint bpoint;
        bool goodContact = true;

        for (int tryseg = 2; tryseg<=lastseg; tryseg++){
                trydistance = distance(Apoints[Aseg],Bpoints[tryseg]);
                if (trydistance<bestdistance){
                        bestdistance = trydistance;
                        Bseg = tryseg;}
                }
        // Bseg is now the segment with distal end closest to distal end of Aseg
        Bsegs[Aseg] = Bseg;
        //transpose Bpoints[Bseg] to local coordinates of Aseg:
                //translation to 0,0 at Apoints[Aseg-1]:
        bpoint.z = Bpoints[Bseg].Z - Apoints[Aseg-1].Z;
        bpoint.y = Bpoints[Bseg].Y - Apoints[Aseg-1].Y;
                // rotation by the angle of Aseg:
        bpointarray[Aseg].z = cos(angle)*bpoint.z + sin(angle)*bpoint.y;
        bpointarray[Aseg].y = -sin(angle)*bpoint.z + cos(angle)*bpoint.y;
```

```
        if(bestdistance>separationLimit) // limit usually 300 nm
                return false;

        if(fabs(angle)>angleLimit) // limit usually 0.3 rad
                return false;

        double Ay = -(bpointarray[Aseg].y +doubletseparation); // normally close to 0
        double Az = bpointarray[Aseg].z;
                if(goodContact){//calculate separation
                        double tanangle = tan(angle);
                        double a = Ay - Az*tanangle;//separation at base of A seg
                        for(int i=0; i<10;i++)
                                ENY[i] = a + (0.05 + 0.1*i)*ds*tanangle;
                        }
        return goodContact;
        }
**************************
```

The next step is to use each distance ENY[i] to calculate 10 values for the forces on each segment, resulting from dynein motor force (parallel to the segment) and adhesive force (normal to the segment). These 10 values are then averaged to obtain the mean forces on each segment. Because some of the forces change nonlinearly with distance, this averaging processs gives better results in some marginal cases.

These calculations are shown by the following code fragment which uses the constants C1,C2,C3,C4 as described in METHODS to obtain four variables: The dynein motor force density on a segment is activeF. The dynein adhesive force density on a segment is varEN. A viscosity-like term that reduces the dynein motor force in proportion to longitudinal velocity is activeCL. The stiffness of the adhesive elasticity is varstiffness, which is used to stabilize the computations, as described below. For simplicity, only the steadystate dynein force model is shown here, although a non-steadystate method, described previously (7, 8), was also used for most of the computations. These two methods produced similar results under most conditions.

```
// active force control:
                double factor = 0;
                if(goodContact&&(C3>C2)){
                        factor = 1.0;// default case for ENY near target
                        if(ENY<C1)
                                factor = 0;// no active force
                          else if(ENY>C3)
                                factor = 0;// no active force
                          else if (ENY>C2)// linear decrease to 0 at C3
                                factor = 1 -(ENY-C2)/(C3-C2);
                        }
                double activeF=factor*fo;
                double activeCL=factor*fo*ESCB/recoveryrate;
//adhesive force density and stiffness in y direction:
                double varEN = 0;
                double varstiffness = 0.0;
```

```
double stf = 0.0;
if(goodContact &&(C3>C2)){
        if((ENY<C3) ){
                if(ENY<0)
                        stf = C4;
                else{
                        if(ENY<=C2)
                                stf = 1.0;
                        else
                                stf = 1 -(ENY-C2)/(C3-C2);
                        }
                }
        }
varEN=EN*ENY*stf;
varstiffness=EN*stf;
```

Note that in these specifications, the stiffness is not equal to the spatial derivative of the force. Instead, the stiffness is based on the idea that the variations in force and stiffness in the region between C2 and C 3 result from a change in the number of force-producing dyneins, and that this number does not change significantly within one time step of the computations. This is a crude approximation, which should be replaced in future work by more realistic mocdelling of dynein attachment kinetics.

The methods for setting up the moment balance equation, including velocity-dependent moments resulting from viscous resistances, were originally described in (6, S1). To add the dynein adhesive force normal to the segment, the bending moment at each joint resulting from dynein forces that do not depend on velocity can be obtained by replacing Eqs. 4-6 in reference (6) with the following equations (in code form):

```
M[seg] = M[seg-1] +ds*cos(angle[seg])*(FY[seg-1])+0.5*ds*ds*varEN
                -ds*sin(angle[seg])*(FZ[seg-1] );
// transform local dF to global and sum:
FZ[seg]=FZ[seg-1] + ds*cos(angle[seg])*activeF-ds*sin(angle[seg])*varEN;
FY[seg]=FY[seg-1] + ds*sin(angle[seg])*activeF+ds*cos(angle[seg])*varEN;
```

In these as in earlier computations (beginning with (7)) it has been found necessary to stabilize the computations by using implicit forms for the moments resulting from the relatively stiff elastic resistances. For instance, for the elastic bending moment

$$M_E(t+\Delta t) = E_B(\kappa + \Delta t\, \partial\kappa/\partial t) .$$

This is is accomplished by placing $-\Delta t*E_B$ into the M[j][j] terms of the coefficient matrix, where they will be multiplied by the unknown rate of bending of each joint. The elastic resistance responsible for the adhesive force, E_N, is handled in a similar manner by adding Δt*varstiffness to the normal viscous drag coefficient, C_N. Stabilization for the elasticity of the dynein force production is provided by adding activeCL to the longitudinal viscous drag coefficient, C_L. In the

steadystate force model, activeCL is a larger effective viscosity, and has the additional effect of reducing the active force to match the steadystate velocity. In the non-steadystate model, activeCL becomes equivalent to $\Delta t*ESCB*f_0$. These devices are not fully satisfactory, because they effectively introduce artificial viscous resistances, which significantly affect the results, and result in sensitivity of the reults to the length of the computational time step, Δt. Computations were routinely performed with 100 segments 120 nm in length, and $\Delta t = 0.025$ ms. With Example A, no differences could be seen with 50nm, 100 nm or 150 nm segments, but a 4% increase in frequency was obtained with 200 nm segments. Increasing the time step size to 50 ms decreasedthe cycle frequency from 6.7 s^{-1} to6.5 s^{-1}. Decreasing the time step size to 10 ms increased the cycle frquency to 7.0 s^{-1}.

Details of the computational methods for the case where doublet B is curved.

Curvature is introduced by initializing the curvature of doublet B with a constant value, C5. It is then assumed that the neutral position of the doublet pair corresponds to two concentric circular arcs that maintain a constant separation between the doublets (usually 60 nm). Only cases where doublet B is the inner arc are considered. The elastic neutral curvature of doublet A is then –1/(doubletseparation–1/C5). This value is subtracted from values of the curvature of the A doublet, before multiplying by E_B to obtain values of elastic bending moment in doublet A.

The additional modification is the replacement of tan(angle) by tan(angle–Bangles[Bseg]) in the getBjointForSeg method. This method is an approximation, justified by recognition that the method is only used when thedoublets are close together, and therefore nearly parallel, so that when an an A segment overlaps two B segments, the change in angle between the two B segments will be small. The test for goodContact also uses angle–Bangles[Bseg] in this case.

This method for introducing curvature was also used to break the symmetry of the system, so that buckling can occur when the force is sufficient. A value of curvature of 10^{-8} rad μm^{-1} was routinely used. For testing conditions for buckling with curved doublets, the computation was started with $f_0 = 0$. After doublet A moved to its equilibrium position with a constant separation from doublet B, the value of f_0 was increased to the test value.

The author will gladly supply more complete programming and/or a working program (Mac PPC) on request.

Supplementary reference:

S1 Brokaw CJ. 1972. Computer simulation of flagellar movement. I. Demonstration of stable bend propagation and bend initiation by the sliding filament model. Biophys J 12: 564-586.