Biophysical Journal, Volume *114*

Supplemental Information

Microtubule Polymerization and Cross-Link Dynamics Explain Axonal Stiffness and Damage

Rijk de Rooij and Ellen Kuhl

Axonal force, stiffness, and damage as emergent properties of microtubule polymerization, crosslink dynamics, and physical forces – Supplemental material –

R. de Rooij, E. Kuhl

MICROTUBULE DYNAMICS

Here we briefly summarize the computational implementation of microtubule dynamics in our axon model. In our simulations, we allow polymerization and depolymerization of microtubules at their plus ends through a `Mechanism` object that we assign to each `MicroTubule` in our model [1].

The first step is to include microtubule dynamics into our `MicroTubule` object [1], see Figure 1. The `MicroTubule` object consists of *active* and *inactive* extended bar elements `EBarX`. Only the *active* elements make up the microtubule. The *inactive* elements are always present in the background, but do not contribute to the axon. Active elements may become inactive when the microtubule is depolymerizing and, vice-versa, inactive elements become active during polymerization. To ensure that the inactive elements do not contribute to the axon, no crosslink can ever be attached to an inactive element. Consequently, if an element becomes inactive due to depolymerization, all crosslinks that were attached to this element will automatically detach.

To keep track of active and inactive elements, our `MicroTubule` object points to its first node and element, (`n0`, `e0`), to its last *active* node and element, (`n1`, `e1`), and to its last *inactive* node and element, (`n2`, `e2`), see Figure 1. While (`n0`, `e0`) and (`n2`, `e2`) remain constant throughout the simulation, (`n1`, `e1`) is continuously updated throughout the simulation as a result of the continuous polymerization and depolymerization.

Four parameters characterize microtubule dynamics: the rates

Algorithm 1 Pseudo code of `mechanism.Apply()` function for MT polymerization.

```
                                  ▷ Apply mechanism to MicroTubule mt
mt.timeToChange←mt.timeToChange−Δt
if mt.timeToChange< 0 then
    Change MT state: mt.ChangeMTState().
else if mt.state==Polymerizing then
    for Element mt.e1 to mt.e2 do
        if el.timeToNextEvent< 0 then
            el.state←Microtubule
            el.timeToNextEvent←∞
            mt.n1←mt.n1+1
            mt.e1←mt.e1+1
            Update pointers in Fig. 1.
        end if
    end for
else if mt.state==Depolymerizing then
    for Element mt.e0 to mt.e1 do
        if el.timeToNextEvent< 0 then
            el.state←MicrotubuleInactive
            el.timeToNextEvent←∞
            mt.n1←mt.n1−1
            mt.e1←mt.e1−1
            Update pointers in Fig. 1.
        end if
    end for
end if
```

Algorithm 2 Pseudo code of `mechanism.ChangeMTState()` function for MT polymerization.

```
                                  ▷ Change state of MicroTubule mt
if mt.state==Polymerizing then
    mt.state←Depolymerizing
    mt.timeToChange←random()*tDepoly
else if mt.state==Depolymerizing then
    mt.state←Polymerizing
    mt.timeToChange←random()*tPoly
end if
        ▷ Update timeToNextEvent for all elements in this microtubule
for Element el from mt.e0 to mt.e2 do
    Let x_1 be the x-coordinate of Element mt.e1.
    Let x_c be the x-coordinate of Element el.
    if mt.state==Polymerizing then
        if el.state==MicrotubuleInactive then
            el.timeToNextEvent←(x_c − x_1)/polyRate
        else
            el.timeToNextEvent←∞
        end if
    else if mt.state==Depolymerizing then
        if el.state==MicrotubuleInactive then
            el.timeToNextEvent←∞
        else
            el.timeToNextEvent←(x_1 − x_c)/depolyRate
        end if
    end if
end for
```

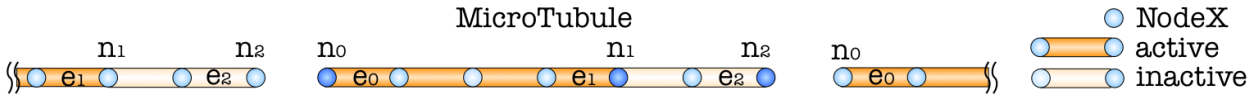

Figure 1: Microtubule object with its elements and nodes. Each `MicroTubule` object consists of *active* and *inactive* extended bar objects `EBarX`; each bar object has two extended nodes `NodeX`. Microtubule polymerization and depolymerization take place only at the distal end. Upon polymerization, the microtubule pointers to its last *active* node and element (`n1`,`e1`) move to the right, towards the distal end; upon depolymerization, they move to the left, towards the proximal end. The first node and element (`n0`,`e0`) remain fixed to limit microtubule polymerization exclusively to the distal end. The node and element (`n2`,`e2`) also remain fixed and mark the maximum length of the microtubule.

and times of polymerization and depolymerization. We apply the `Mechanism` object ithroughout the entire simulation by calling `mechanism.Apply()` at the beginning of each time step. This function consists of two main parts, see Algorithm 1. First, it checks whether the microtubule has to change its current state from polymerizing to depolymerizing or vice versa. Second, it applies the necessary updates for the next time step to polymerize or depolymerize the microtubules. Algorithm 2 summarizes the pseudo-code for changing the current microtubule state.

GEOMETRIC NONLINEARITY

Our results display a nonlinear relation between axonal stress and strain, even though all individual constituents of the axon model are linear elastic. Here we show that these nonlinearities are of geometric nature and result from crosslink rotation. We derive a simple analytical model for a representative volume element consisting of two microtubules connected by two crosslinks, see Figure 2. In the unloaded axon, the two crosslinks are inclined against the axonal cross section by $+\theta_0$ and $-\theta_0$, where θ_0 is the crosslink angle between two microtubules that are separated by the distance w. We assume the microtubules are rigid and the elastic rigidity of the crosslinks is EA_0. We apply a displacement u and calculate the angles θ_1 and θ_2 of the deformed crosslinks in the loaded axon as

$$\tan\theta_1 = -\tan\theta_0 + \frac{u}{w} \quad \text{and} \quad \tan\theta_2 = +\tan\theta_0 + \frac{u}{w}\,.$$

The force-displacement relation then becomes

$$F = EA_0\,[\,\varepsilon_1 \sin\theta_1 + \varepsilon_2 \sin\theta_2\,]\,,$$

where $\varepsilon_i = \cos\theta_0/\cos\theta_i - 1$ is the strain in the two crosslinks. This equation explains the nonlinearity in the force-displacement relation, which is entirely caused by the change in crosslink angle θ_i and depends strongly on the crosslink angle θ_0 in the initial unloaded axon.

Figure 2 shows characteristic force-displacement curves for different crosslink angles θ_0. For small initial crosslink angles θ_0, the analytical model displays a geometric stiffening with increased loading; for higher initial crosslink angles θ_0, the model displays an initial softening, followed by stiffening as the displacement increases. The analytically predicted nonlinearity in Figure 2 is qualitatively similar to the computationally simulated nonlinearity of our axonal model. Yet, the axon model differs from our simplified analytical model in several aspects:

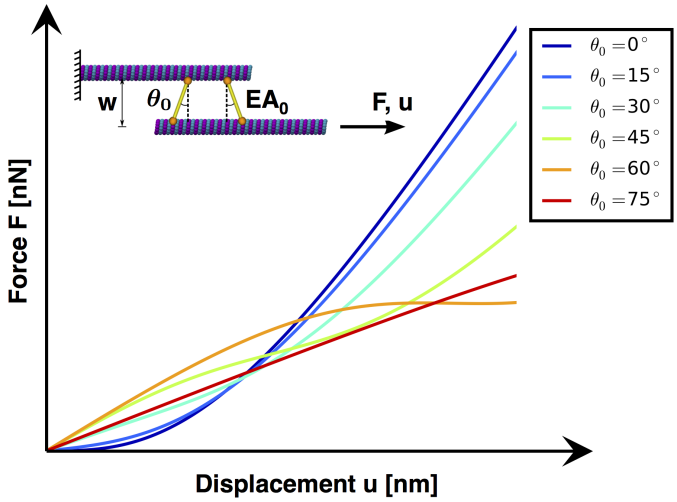

Figure 2: Our analytical model of two rigid microtubules connected by two elastic crosslinks predicts that the overall force $F = EA_0\,[\,\varepsilon_1 \sin\theta_1 + \varepsilon_2 \sin\theta_2\,]$ increases nonlinearly with the applied displacement u and that this nonlinearity critically depends on the initial crosslink angle θ_0.

First, the axon models consists of many microtubules with different overlaps and a complex interplay of crosslinks acting either in parallel or in series. Second, the analytical model assumes that exactly half of the crosslinks are inclined with $+\theta_0$ the other half with $-\theta_0$. On average, this ratio is the same for our axon model, but it may slightly vary for individual overlaps of two microtubules in a simulation. Third, the crosslinks in our analytical model remain attached to the microtubules, whereas the crosslinks in our computational axon model detach and reattach continuously. This difference becomes especially noticeable when the loading rate is lower than the attachment and detachment rate of the crosslinks.

REFERENCES

[1] de Rooij, R., K. E. Miller, and E. Kuhl, 2017. Modeling molecular mechanisms in the axon. *Computational Mechanics* 59:523–537.