Aribtrary Precision for Semi-analytic Bending of Thin Fibers

Benjamin Schreyer
benontheplanet@gmail.com

Abstract

In building flexures with possible application to devices such as torsion balances, optical suspensions, or pendulums the fibers become thin and semi-analytic calculations of their bending become unstable with fixed size floating point. To enable more computationally efficient numerical explorations of flexure designs we provide simple analytical results which show that instability of semi-analytic bending simulation is due to small angle exponential growth of the bending angle. The analytic solutions are used to provide efficient guesses for applying the shooting method to bending with an arbitrary precision implementation of RK45 which resolves the instability.

1 Introduction

1.1 The bending equation

Consider a sheet-like flexure (one transverse dimension of the flexure is much smaller than the other) suspending a weight. The weight experiences gravitational force $F_{w,0} = mg$ and a side force G_0. The bending angle θ and moment M a distance s along the flexure depend on boundary conditions and material parameter E. The geometry of bending has the following governing equations.

$$\frac{dM}{ds} = F_{w,0}\sin(\theta(s)) + G_0\cos(\theta(s)) \tag{1}$$

$$\frac{d\theta}{ds} = \frac{M(s)}{EI(s)} \tag{2}$$

The geometry of the unbent material is encoded in $I(s)$ the second moment area of the cross section. For the case we study:

$$I(s) = \frac{h(s)^3 b}{12}, \tag{3}$$

where $h(s)$ is the shape of the bending flexure and b is the thickness of the thin sheet flexure.

Bending can be solved using many ODE solvers and one sided boundary conditions. Then to impose a boundary condition on the other end of the flexure, $s = L$, the shooting method can be used. This approach is effective but fails in extremely flexible cases where the sin term dominates. Our implementation will shoot for a desired final bending angle θ_0.

1.2 Floating point representation

There are standards for floating point representation, for our purposes is it only important to know that a floating point number is represented roughly by two integers as $N = m \cdot 2^l$. The base of two is unimportant but is chosen for convenience with digital devices. To model a fixed-size floating point number, assign m and l a certain number of bits in representation. This makes hardware much more efficient due to standardization and the containment of m and l to say a single 64 bit word as is the case with float64 types. This inherently imposes a smallest and largest number possibly represented which is ok for most uses. Here for extreme bending we need to use a library mpmath for python which implements floating point calculations with m and l represented with as much memory as needed. This means that while the smallest number one can represent with float64 is something like $2^{-2^{10}} = 10^{-308}$. If we take seriously the arbitrary precision of mpmath and are using a computer with 1MB of memory for our number the smallest number representable is $2^{-(2^{1000000})}$. For representing physical quantities this large exponent is not needed, one can introduce a larger or smaller unit. For the calculation of bending however, the limitations of float64's exponent prevent calculations that appear in practical cases.

2 Analysis

2.1 Extreme exponents in bending

The problem arising with exponent limitations in bending calculation are confusing because they involve scales like 10^{-308}. If something is so small it goes unmeasured, but because a symmetry breaking numerical trick of this scale is needed when $G_0 = 0$. We must represent such small values in our solver which is not possible with a float64. A small initial condition on M, like $M = 10^{-400}$ is needed for weighted bending with no side force G_0. Without a small nonzero M the problem is symmetric and would not favor a leftward or rightward bend. If $G_0 \neq 0$ then this numerical trick is not needed as the side force breaks the left-right symmetry of the solver.

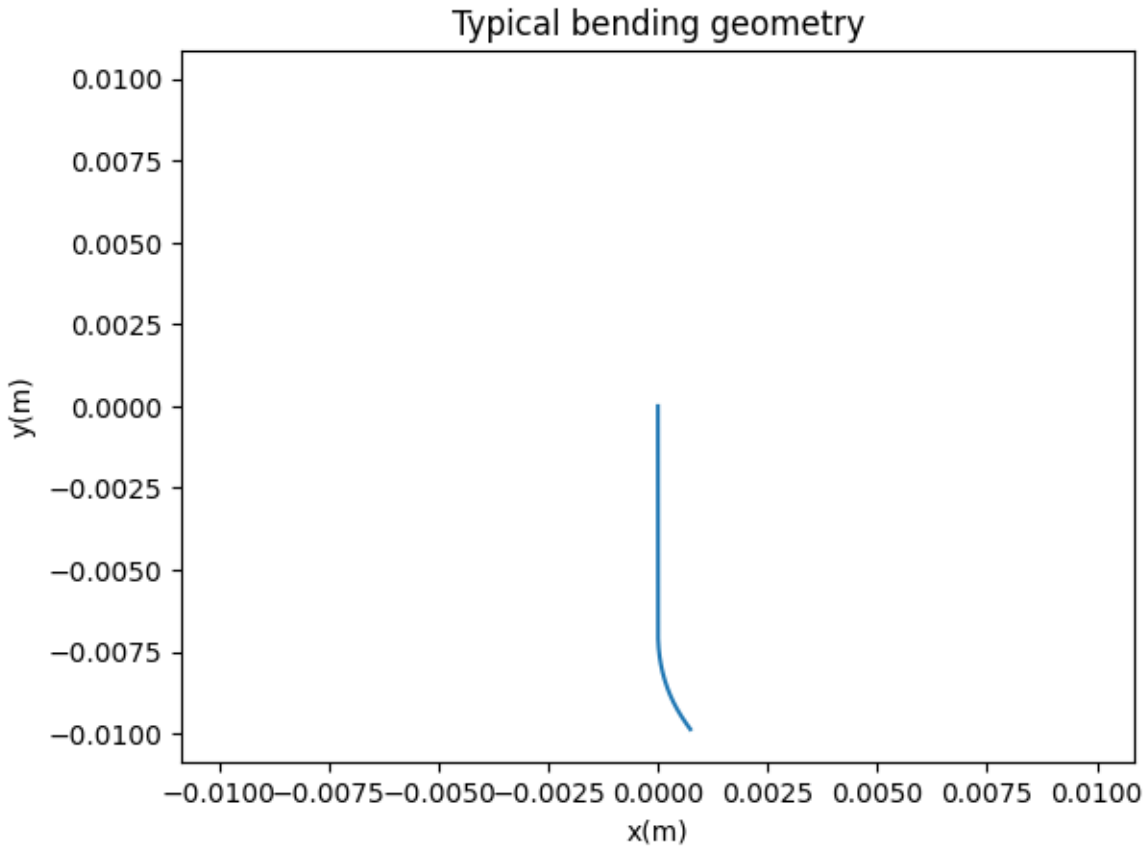

Figure 1: A solution to the bending problem taking the path $\theta(s)$ into Cartesian coordinates. The flexure is mounted at the origin.

When one tries to solve for the bending of a material in our application, the boundary conditions $M(0)$ and $\theta(0)$ are set. Then the shooting method is used on the final angle to reach a solution with $\theta(L) = \theta_0$. When applied this algorithm works, except for when the geometry of the flexure arm becomes thin as in Figure 2.1.

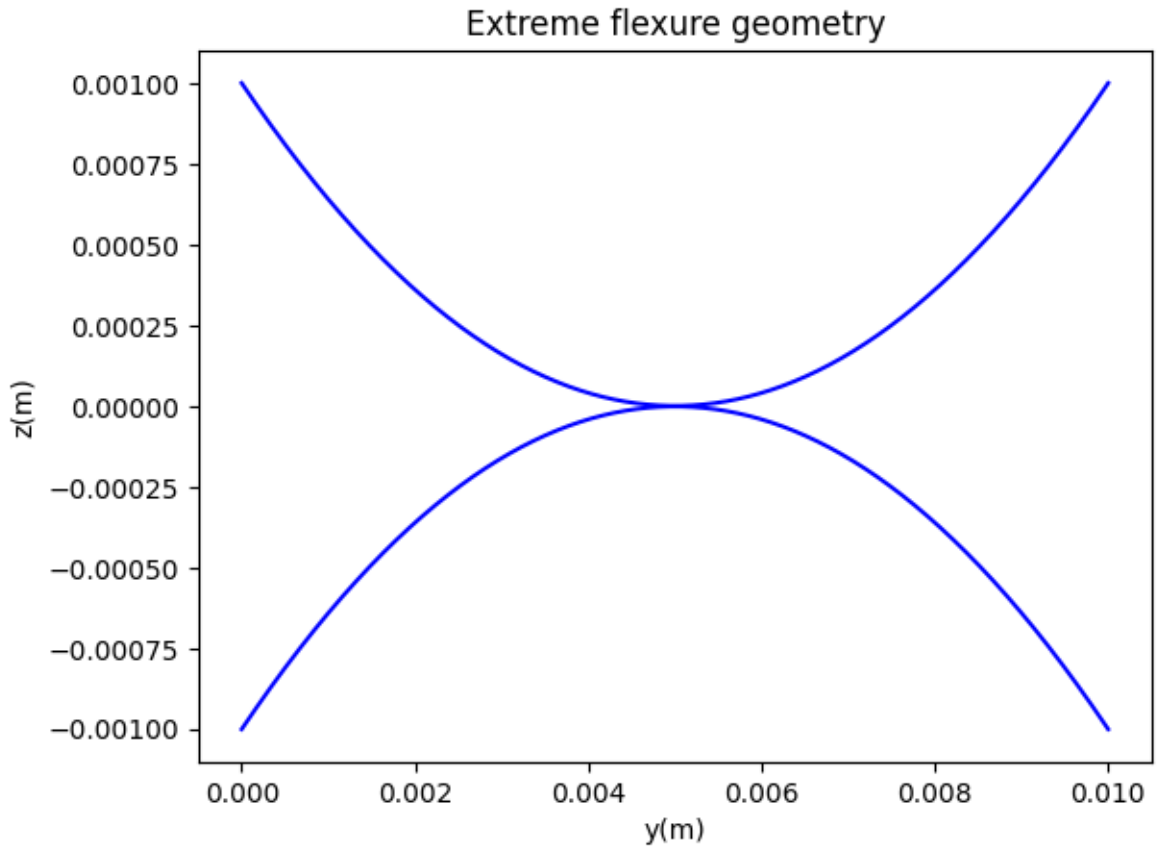

Figure 2: A geometry ($\frac{1}{2}h(s)$) for the flexure which is extremely small at its center. It shrinks from its base to be $10,000$ times thinner at its center and the sheet thickness $b = 0.1$mm

According to the equation for the second moment (3) and the variation of θ (2) when the geometry gets extremely thin the rate of change of θ will become extremely large. The result is that to get reasonable resulting $M(L)$ and $\theta(L)$ in a shooting approach the initial conditions must be extremely small. Then the final value for θ does not grow beyond $\frac{\pi}{2}$ which is where oscillations in θ and M begin. For example a calculation can be made that gives $\theta(L) = 1$ for the thin geometry pictured. To acheive this numerically, float64 or float128 will not allow a convenient solution. Convenient means without a headache of rescaling and dealing with the inherent scale of 2π from trigonometric functions. Instead the mpmath library for python can be used when the geometry is thin for implementing the shooting method applied to the bending equation.

2.2 Approximate analytical solutions

Sine term only To illustrate beyond a qualitative understanding that there is an exponential increase present in bending, consider the region where θ is small and $F_{w,0}\sin(\theta)$

dominates. The small angle holds true for part of all bending geometries with $\theta(0) = 0$. Additionally consider a constant width geometry $I(s) = C$. Then the equations for bending simplify to

$$\frac{d^2\theta}{ds^2} = \frac{F_{w,0}}{EC}\theta. \tag{4}$$

The solution is trivially exponential. The scaling of θ from the initial $\theta(0)$ to $\theta(l)$ is easily quantified.

$$\theta(l) = \exp(\sqrt{\frac{F_{w,0}}{EC}}l)\theta(0) \tag{5}$$

Now to estimate the total scaling over a varying geometry, for example Figure 2.1, approximate the geometry as piecewise-constant. The end angle will be

$$\theta(L) = \theta(0)\exp(\sqrt{\frac{F_{w,0}}{E}}\int_0^L \frac{1}{\sqrt{I(s)}}ds). \tag{6}$$

Calculating this scale factor for the geometry above and $E = 7 \times 10^{10}$Pa and $F_{w,0} = 120$N the scale factor is 10^{19000}. In simulation the correct initial condition chosen to produce the bending plots given was order 10^{-1000} (to cancel the exponential growth). This means the actual scale exponent was 1000. The inaccuracy of the guess is due to the piecewise-constant approximation. This easy to calculate estimate provides a good starting guess when shooting for initial conditions.

Cosine term only The same assumptions are made as in the sine case, only differing by considering a dominant $G_0\cos(\theta)$ term instead, and removing the piecewise-constant approximation. The approximate solution for small θ and $\theta(0) = 0$ is

$$\theta(L) = \int_0^L \frac{G_0 s}{EI(s)}ds. \tag{7}$$

General case We do not provide analysis for the general case as it is empirically determined that applying the cosine approximation, and then scaling by the scale factor of the sine approximation provides a sufficient initial guess at G_0 for shooting the bending ODE when both terms contribute.

3 Algorithm for general semi-analytic bending

We assume the bending angle is less than $\frac{\pi}{2}$ to avoid an oscillating regime on θ and M which could make shooting for solutions more difficult.

3.1 Estimation

The shooting method alone is theoretically sufficient to find bending solutions, but it is much faster to provide an initial guess based off of approximate solutions. If such an initial guess is not provided naive shooting would search an interval that appears to be exponentially large in the integral of $\frac{1}{\sqrt{I(s)}}$.

Sine term only estimate The scale factor $\exp(\sqrt{\frac{F_{w,0}}{E}}\int_0^L \frac{1}{\sqrt{I(s)}}ds)$ is calculated for the bending geometry. Then because there is some error in this estimate, the arbitrary precision RK45 (APRK45) solver is used to determine an ancillary bending angle $\theta^\star$ for an initial $M^\star(0) = \exp(-\sqrt{\frac{F_{w,0}}{E}}\int_0^L \frac{1}{\sqrt{I(s)}}ds)$. Then the guess for the shooting stage is determined as $M(0) = \frac{\theta_0}{\theta^\star}M^\star(0)$. This is simply leveraging the linear nature of the approximate ODE to rescale the purely analytic estimate to better acheive θ_0 under the small angle approximation.

Cosine term only estimate When a side force is present it determines the final bending angle. Instead of shooting the initial condition $M(0)$, we have $M(0) = 0$ and are shooting G_0. In this case we can use the cosine only approximate solution to figure that a good guess for G_0 to produce a bending through angle θ_0 is $G_0 = (\int_0^L \frac{s}{EI(s)})^{-1}\theta_0$.

Both terms significant In this case, G_0 breaks symmetry but the cosine only solution does not provide a good guess. The math is hard so we don't do it and instead take a guess which works well empirically. Instead find the scale factor for the sine only solution, and multiply it on the initial guess for G_0 produced for a cosine only estimate. That is $G_0 = (\int_0^L \frac{s}{EI(s)})^{-1}\exp(-\sqrt{\frac{F_{w,0}}{E}}\int_0^L \frac{1}{\sqrt{I(s)}}ds)\theta_0$. Then run an iteration of APRK45 on this guess, and refine it by approximate linearity as in the sine term only estimate.

3.2 Shooting

Then a shooting step is done in either G_0 or $M(0)$ with the APRK45 solver for side force significant and side force insignificant cases respectively. With the parameters we have tried and the initial estimates above the shooting method achieves satisfactory accuracy order percent with order 10 shooting steps. We have found that secant method shooting was about twice as fast as binary search shooting for our parameters. Regula falsi method shooting is somewhat slower than secant method but has more consistent convergence so is preferred.

3.3 Speed and error

The fixed step-size arbitrary precision solver leverages the RK45 coefficients to provide calculable and sufficiently small error. The geometry $I(s)$ is sampled at a fixed interval and cubic

spline interpolated to feed the solver. Errors for our parameters were much smaller than our tolerance with RK45. Time to make a single bending calculation for 100 sample points along the fiber was about 5 seconds on a personal computer with python3 implementation. Implementing the arbitrary precision solver in a compiled language would greatly boost this performance to order ten calculations a second.

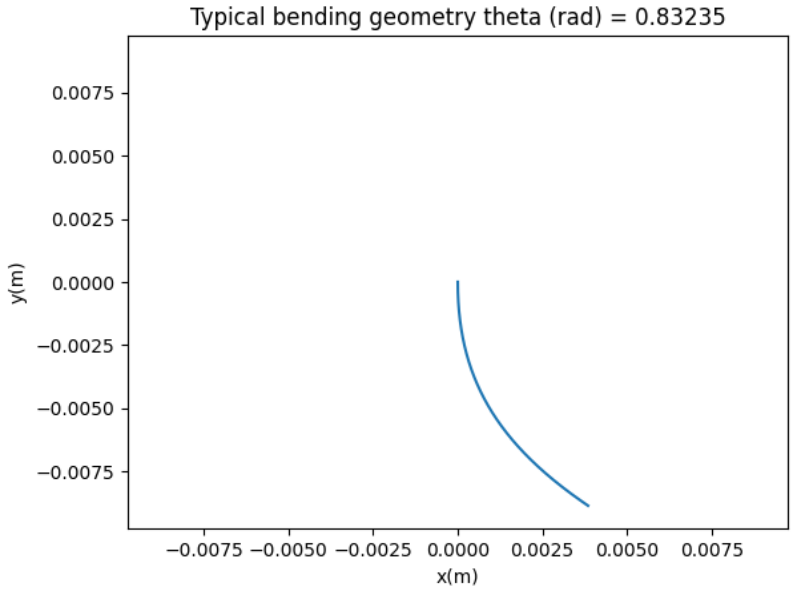

Figure 3: Bending of a nearly rectangular flexure.

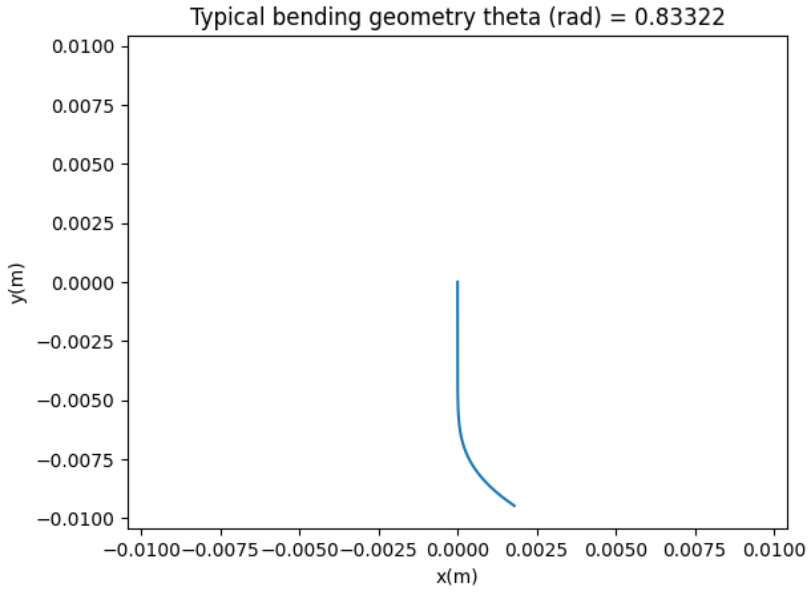

Figure 4: Bending of flexure that reaches a minimum half of its initial thickness.

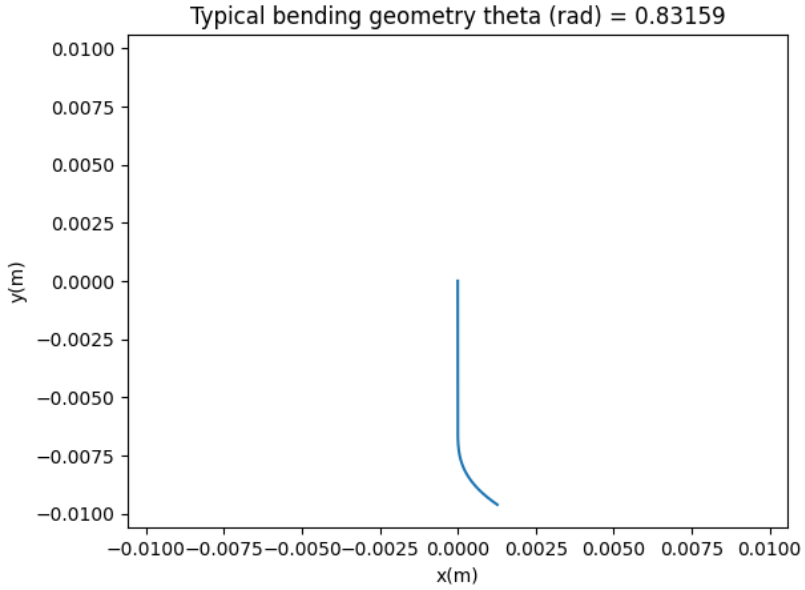

Figure 5: Bending of flexure that reaches a tenth of its initial thickness.

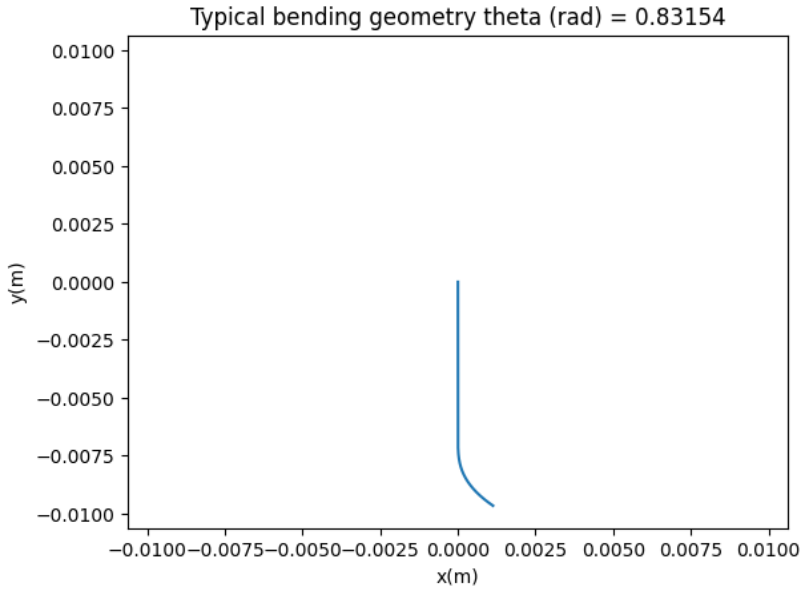

Figure 6: Bending of flexure that reaches a percent of its initial thickness.