Stable numerical technique to calculate the bending of flexures with extreme aspect ratios

Benjamin Schreyer[1], Lorenz Keck[2], Jon R. Pratt[1], Stephan Schlamminger[1]

[1]National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD, USA

[2] SLAC National Accelerator Lab, 2575 Sand Hill Rd, Menlo Park, CA 94025, USA

E-mail: `stephan.schlamminger@nist.gov`

Abstract. In designing flexures as part of torsion balances or pendulums for scientific use, structures become thin and long making semi-analytic calculations of their bending infeasible with standard double-precision variables. Semi-analytic calculations can be more efficient than finite element methods, allowing faster design optimization. We present straightforward analytical results demonstrating the origin of failure in semi-analytic bending simulations performed using double precision. Given the exponential scaling of the initial condition, a minute starting moment is required for the shooting method. To address this issue, we developed a Runge-Kutta 45 integration scheme combined with a semi-analytic bending model, that successfully yields accurate results where standard double-precision bending model implementations fail.

Keywords: arbitrary precision, compliant mechanism, double precision, Euler-Bernoulli beam, Runge-Kutta

1. Introduction

Loaded flexure elements are common in compliant mechanism design. A mechanism is compliant if it relies on elastic deformation of engineered structures to achieve static or dynamic mobility requirements. Compliant mechanisms have many applications such as in medicine [1], transportation [2], robotics [2], precision positioning [3], and measurement systems [2, 4–6]. The basic function of compliant mechanisms can be modeled by using shear-free beams. Several authors have sought semi-analytic models of these structures to decrease the need for computational resources [6–9].

We describe numerical methods that lie between a fully analytic solution and FEA tools, which define a mesh. We call these methods semi-analytic because they reduce the problem to solving an ordinary differential equation (ODE). In moving to these more computationally efficient models for flexures used in scientific measurement, we find that common ODE solvers use floating-point number formats that fail to model extreme aspect ratio shear-free beam bending. The initial internal bending moment or side forces, required to model the deflection of extreme aspect ratio beams, scale by such an order of magnitude that more floating-point exponent bits are required than double precision offers. We have implemented a Runge-Kutta integrator in arbitrary precision floating-point to demonstrate that we may successfully model flexures outside the capability of ODE solvers provided by common scientific programming languages.

1.1. *The single flexure*

We study the numerical solution of a single flexural element. Even in such a simple case typical double precision variables will fail to correctly model extreme aspect ratio shear-free beam bending. Consider a flexure suspending a weight, $F_\mathrm{w} = mg$, and additionally a deflecting force F_d. In our simplified study the forces act at the end of the flexure. The flexure angle $\theta(s)$ and moment $M(s)$ given at a distance s from the origin along the neutral axis depend on boundary conditions, and elastic modulus E. The moment $M(s)$ is a torque internal to the flexure material and is not directly related to rotational inertia of any physical body. We assume the flexure is clamped at $s = 0$ with an initial tangent angle of zero, shown in Figure 1. We can take the initial tangent angle to be zero without loss of generality. If in some problematic coordinates the initial tangent angle is nonzero we rotate the plane to zero the initial tangent angle. In zeroing the initial tangent angle F_d and F_w will change in accordance with the rotated coordinate system. The flexure has a length along its neutral axis L. The geometry of deformation can be determined by the coupled differential equations below.

$$\frac{\mathrm{d}M}{\mathrm{d}s} = F_\mathrm{w}(s)\sin\left(\theta\left(s\right)\right) + F_\mathrm{d}(s)\cos\left(\theta\left(s\right)\right) \tag{1}$$

$$\frac{\mathrm{d}\theta}{\mathrm{d}s} = \frac{M(s)}{E(s)I(s)} \tag{2}$$

for the moment and angle, respectively [9]. In the simplest case, $E(s), F_\mathrm{w}(s)$, and $F_\mathrm{d}(s)$ are constant, i.e., independent of s. An example geometry, $\theta(s)$, is shown in Figure 1.

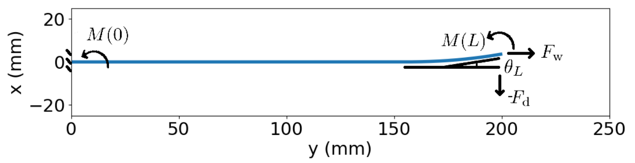


Figure 1: End torque bending to $\theta_L = 10°$ of a loaded flexure. The flexure is drawn lying on its side – gravity is acting on the $+y$ direction. Coordinate s runs along the deformed flexure and so is not cartesian.

The shape of the unbent flexure's cross section is captured by $I(s)$, the second moment of area of the plane perpendicular to the neutral axis. The equation for the second moment of area is

$$I_\circ(s) = \frac{\pi}{4} r^4\left(s\right) \text{ , and} \tag{3}$$

$$I_\square(s) = \frac{1}{12} b\, h^3\left(s\right), \tag{4}$$

for a circular cross section with radius $r(s)$ and a rectangular cross section with width b and thickness $h(s)$, respectively. In general, the cross-sectional parameters vary as a function of s, necessitating a numerical solution.

A numerical solution of the bending of such a flexure can be obtained by ODE solvers that consider a single-sided boundary condition. We study the case where θ_L and F_w are given, $M(0)$ and $M(L)$ are solved for. Typical implementation of these solvers, for example in Python, use standard double precision which uses 64 bits and is also referred to as float64 [10]. In our algorithm the boundary condition at the free end of the flexure, $\theta(s = L) = \theta_L$, is met within a user specified tolerance. The shooting method [11], which evaluates multiple initial value problems to solve a boundary value problem, is used to satisfy the end boundary condition. For example, a final bending angle of $\theta_L = 10°$ of a flexure is apparent at the curled end in Figure 1. The angle the flexure's tangent

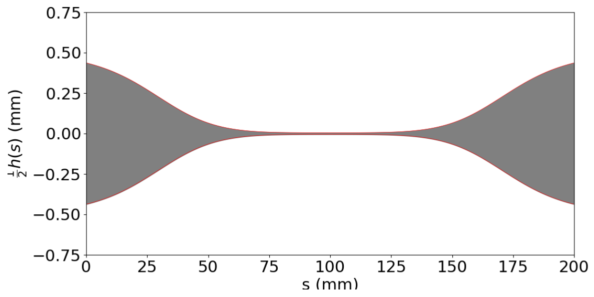


Figure 2: A thin planar flexure profile where for reasonable parameters ($E = 1.31 \times 10^5$ MPa, $F_\mathrm{w} = 1$ N, $b = 0.1$ mm) float64 is not sufficient when applying clamp-sided shooting. The flexure becomes 100 times smaller at its waist.

makes with the y axis at its end is θ_L, a quantity that is useful as a physically measurable boundary condition. Precision beyond float64 was used to solve for the deformation, $x(s)$ and $y(s)$ or equivalently $\theta(s)$, in Figure 1. We require more precision than float64 because when numerically integrating the geometry $M(0)/M(L) \sim 10^{-596}$. Such scaling is outside the capability of float64, which can only represent nonzero numbers that are larger in magnitude than 2×10^{-308}.

This numerical requirement can be easily demonstrated for a flexure with a constant cross section in the absence of F_d, which can be analytically solved. According to Eq. (1) and Eq. (2) in [12] which describes the analytical solution for bending of a flexure with constant cross section, we find

$$M(0) = 2M(L)e^{-\alpha L} \text{ with } \alpha^2 = \frac{F_\mathrm{w}}{EI}. \tag{5}$$

the ratio $M(0)/M(L)$ becomes exponentially smaller for decreasing I (thinner and thinner flexure cross section), eventually demanding many more exponent bits than float64 can provide.

The purpose of this article is to numerically solve the bending of thin flexures. As mentioned above, for constant cross sections, the system of differential equations can be solved analytically. In practice, however, loaded flexures are engineered or constrained by specifications to have a non-constant cross section, and a numerical calculation is the only feasible technique to solve the bending of the flexure. Nonlinear bending can be solved with finite element analysis or semi-analytically, as the system of differential equations shown above. The latter is faster because the meshing occurs only in one direction and is, hence, more efficient and can speed up the development cycle for compliant mechanisms [4,5,13]. ODE-relaxation methods [11,14] offer one way to solve the system of differential equations. Alternatively, the shooting method can be employed and has been previously used with flexures that have modest aspect ratios [6,9]. A detailed description follows below.

1.2. *The shooting method*

To find curves which represent the neutral axis of a deflected flexure we must numerically integrate the coupled bending ODEs. To ensure that boundary conditions are met, such as clamping and a desired bending angle $\hat{\theta}$ we use the shooting method. The shooting method reduces a boundary value problem to an initial value problem, where the two-sided boundary conditions can be iteratively satisfied by varying the initial condition [11]. The iteration step can be formulated as a root-finding problem for an auxiliary function that enforces the boundary condition. For bending, the goal is to find the root of the difference between the calculated and desired end angles, $\theta_L(p)$ and $\hat{\theta}$ respectively,

$$G(p) = \theta_L(p) - \hat{\theta}. \tag{6}$$

The root is found by varying the shooting parameter p. Evaluating the auxiliary function requires a numerical integration for a given p of Eq. 1 and Eq. 2 from $s = 0$ to $s = L$ for each iteration, to obtain $\theta_L(p)$. For bending calculations, the shooting method is effective, but when implemented with float64, it fails in thin cases. Approximate solutions let us further define a condition for a flexure to be *thin* in terms of geometric, loading, and material properties of the flexure. Figure 2 shows a flexure construction geometry where a float64 solver would fail. The result of such a failure is that no root will be found. The p required to achieve this root for bending simulation is too small, but nonzero. Hence, it can not be appropriately approximated with float64. In our specific implementation when the initial condition cannot shrink enough to represent the desired solution, the fiber angle $\theta(s)$ will take on values far greater in order of magnitude than π. Such a failure results in a nonphysical solution which does not satisfy the end angle boundary condition at $s = L$. Here, for simplicity, we always fix $F_\mathrm{d} = 0$. Without a side force $M(0)$ is the shooting parameter and determines the final bending angle. With F_d fixed the number of undetermined initial conditions (one) matches the number of boundary conditions to be satisfied (one, θ_L).

1.3. *Floating-point representation*

Limitations of numerical methods for differential equations based on fixed-size floating-point are well known. The specific problem of floating-point exponent limitations has been addressed for calculations regarding

Legendre polynomials [15, 16]. Abad et al. [17] show the failure of periodicity for numerical orbits with too few decimals of precision. Double and single precision can be insufficient for numerically determining a choice of Lorenz attractor [18]. Limitations of floating-point representation can also lead to the appearance of anomalous solutions that are only exhibited by numerically approximated systems [19]. In some cases finding such errors can be automated using dynamic program analysis [20, 21].

A floating-point number N is represented on a computer using three binary encoded values, a sign bit ($\pm$, and two integers d and l as $N = \pm\frac{d}{D} \cdot 2^l$. The multi-bit wide value d and l are referred to as the mantissa and the exponent, respectively. The mantissa is normalized to be less than one with D. As the name implies, float64 uses a total of 64 bits to encode the binary N. According to the IEEE standard [10], the most significant bit, the sign bit, is followed by an 11-bit wide (signed) exponent and a 52-bit wide (unsigned) mantissa. In double precision, the exponent lies between $-1022 \leq l \leq 1023$, limiting the calculations to those which have an order of magnitude between 10^{-308} and 10^{308}. For the calculation of bending the limitations of float64's exponent prevent calculations that appear in practical cases. As we shall discuss below, float64 is not sufficient for the calculations of thin flexures. Quadruple precision or float128 extends this range enough for the thin flexures we attempted to model. Float128 uses a 15-bit wide exponent and a 112-bit wide mantissa [10]. Now, $-16382 \leq l \leq 16383$, which limits the exponents to 10^{-4932} and 10^{4932}. One can represent smaller numbers than 10^{-4932} down to 6.6×10^{-4966} by using very small numbers for the mantissa. However, by doing so, one trades precision for magnitude, and full loss of mantissa precision disallows achieving arbitrary end angles with any precision. Without increasing the number of bits representing the exponent, initial conditions that lead to any desired order one radian final bending angle are not discoverable by the shooting method. We use the library mpmath [22] for Python to perform our calculations. It implements floating-point calculations with an exponent range that exceeds the practical application of the bending model.

A common strategy to mitigate numerical errors associated with extremely large or small absolute values is to rescale the variables. For instance, rather than computing the moment $M(s)$ in N m, one might calculate it as a dimensionless ratio $M(s)/M_n$, where M_n is a very small moment, for example, 1×10^{-20} N m. However, two key issues limit the effectiveness of this approach. First, as shown in Eq. 5, the ratio $M(L)/M(0)$, comparing the moment at the free and fixed ends of the flexure, can span hundreds of orders of magnitude. A uniform scaling factor does not address this, as it cancels out in the ratio. Second, even if the angle, which may also grow exponentially by Eq. 2, is rescaled as $\theta(s) = r(s)\theta_n(s)$, numerical instabilities persist. This is because of trigonometric functions which are sensitive to the absolute magnitude of their arguments. For example, for small angles $\theta(s)$,

$$\cos\theta(s) = \cos\left(r(s)\theta_n(s)\right) \approx 1 - \frac{1}{2}r(s)^2\theta_n(s)^2. \quad (7)$$

Hence, one still needs the full numerical precision to obtain the correct result. So, rescaling the variables with common factors does not lead to a satisfactory solution, but increasing the floating-point exponent size does without adding too much complexity.

2. Analytical solutions

2.1. Large exponents in bending

The problem that arises with exponent limitations in the outlined bending calculations is unintuitive because it involves scales smaller than the float64 minimum 10^{-308}. If a moment, force, or angle is so small it goes unmeasured, but is present according to the bending model. Numerically, this scale is of practical use because the growth of the flexure angle and moment can be exponential in s when the angle is small. With $F_\mathrm{d} = 0$ and $M(0) = 0$ the numerical approximation problem is symmetric and would not favor a leftward or rightward bend geometry. The solution shown in Figure 1 was determined with a very small initial value $M(0)$. With unbroken symmetry ($\theta(0) = 0\,\mathrm{rad}, M(0) = 0\,\mathrm{N\,m}$), Eq. 1 and Eq. 2 have null right-hand side. The result is zero bending for all s, which carries over to numerical solutions.

To achieve a bending solution, we may introduce a small but non-negligible moment at the clamp. For the flexure defined by the parameters in Table 1, achieving a final bending angle on the order of one radian requires an initial moment $M(0)$ of order 1×10^{-330} N m. We must represent such small values in our ODE solver, which is not possible with float64. Further, we show that in general the system exhibits exponential growth and corresponding limitations due to floating-point precision.

2.2. Small angle solutions

Hyperbolic trigonometric functions as solutions to beam deformation have been documented in textbooks and are often used in practical research [12, 24]. Here, we investigate these approximate solutions in a form that is amenable to exposing exponent limitations. Any solution with $\theta(0) = 0$ will closely match the behavior of small angle solutions in some portion of

Table 1: Parameters of a circular cross section flexure used in [23], assuming $F_\mathrm{d} = 0$. The parameters above the single line are the geometric and mechanical properties from which the parameters below the line are calculated.

Par.	Eq.	Value	
E		7.3×10^{10}	$\mathrm{N/m^2}$
L		6.00×10^{-1}	m
r		2.00×10^{-4}	m
F_w	mg	1.47×10^{2}	N
F_d		0	N
I	$\pi r^4/4$	1.26×10^{-15}	$\mathrm{m^4}$
α	$\sqrt{F_\mathrm{w}/(EI)}$	1.27×10^{3}	$\mathrm{m^{-1}}$
αL		7.60×10^{2}	
$\lfloor 2^{l-1} \ln 2 \rfloor_{50}$	for $l = 11$	7×10^{2}	
$e^{\alpha L}$		1.16×10^{330}	
σ_w	$F_\mathrm{w}/(r^2\pi)$	1.176×10^{9}	$\mathrm{N/m^2}$
$\sqrt{E/\sigma_\mathrm{W}}$		7.90	
L/r		3.00×10^{3}	

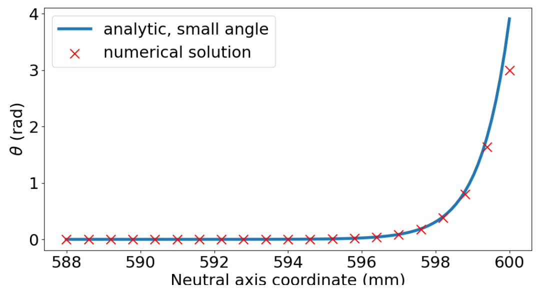
the flexure length. Therefore, numerical limitations exposed via small angle solutions will also affect large angle bending problems.

To show exponential behavior in bending, consider the solutions where θ is small. For $F_\mathrm{d} = 0$, the second order equation obtained by combining Eq 1 and Eq 2 yields an exponential solution for both $M(s)$ and $\theta(s)$.

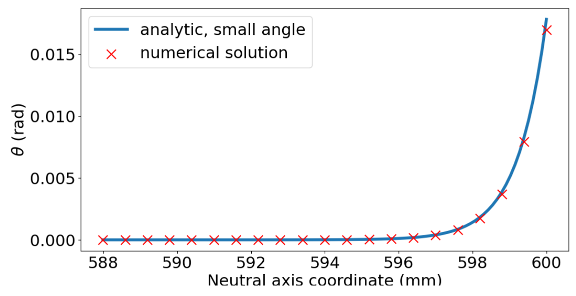
For more detail, consider a constant width geometry $I(s) = I_0$ and $\theta(0) = 0$. In this case, we can draw upon the solutions provided by Speake [12]. A minor but notable difference lies in the coordinate system: Speake formulates the problem using x and y, whereas we employ θ and arc length s. Within the framework of small angle bending, the approximation $x \approx s$ holds, simplifying the change of coordinates. Analytical approximate bending solutions are sufficiently accurate to capture and quantify the numerical instabilities under discussion. Speake describes the linear dependence between the torques and forces, and the deformation in single flexure systems. Linearity allows the deformation and external torques and forces to be related by a compliance matrix. The compliance matrix given in [12] can be used to find the end bending angle, $\theta(L)$. To expose exponent limitations, we consider large growth in $M(s)$ and $\theta(s)$. For the analytical solutions [12], this means $\alpha L \gg 1$, with $\alpha^2 = F_\mathrm{w}/(EI_0)$. In this case, $\sinh \alpha L \approx \cosh \alpha L \approx \frac{1}{2} e^{\alpha L}$ and, consequently, $\tanh \alpha L \approx 1$. Using Eq. (1) and Eq. (2) in [12] together with the above approximations, we find that

$$M(L) \approx \frac{1}{2}\left(M(0) - \frac{F_\mathrm{d}}{\alpha}\right) e^{\alpha L}. \tag{8}$$

Since the term in parentheses is constant along the flexure, we have exponential dependence. This is unsurprising, since the static bending modes are hyperbolic trigonometric functions. Furthermore, $\theta(s)$ can be obtained by integrating $M(s)$ according to Eq. 2. Hence, the scaling of $M(s)$ leads to the same scaling in $\theta(s)$, see Figure 3.



(a) Large angle bend



(b) Small angle bend

Figure 3: Bending angle near the end of a loaded torque bending flexure with parameters as in Table 1. Insets (a) and (b) show small and large deflections, respectively. For (a), the small-angle solution deviates from the numerical solver since the bend angle is sufficiently large.

Further, the final bending angle is given using Eq. (3) in [12] by

$$\theta(L) \approx \frac{F_\mathrm{d}}{F_\mathrm{w}}(1 - 2e^{-\alpha L}) + \frac{\alpha M(L)}{F_\mathrm{w}}. \tag{9}$$

Finally, we can substitute $M(L)$ and ignore terms that are not exponentially large in αL. We find a form that lets us study the floating-point exponent range needed to result in an order one end angle,

$$\theta(L) \approx \frac{1}{2F_\mathrm{w}} \left(\alpha M(0) - F_\mathrm{d}\right) e^{\alpha L} \quad \text{for } \alpha L \gg 1. \tag{10}$$

The difference $(\alpha M(0) - F_\mathrm{d})$ is not small relative to $\alpha M(0)$ or F_d in general. Therefore, in cases where αL is large we may require very small parameters F_d and $M(0)$.

2.2.1. Definition of a long thin flexure

The approximate analytical solution given above allows us to formally define what we mean by a thin flexure for a calculation with a certain floating-point exponent size. Assume a circular cross section flexure is loaded with force F_w, which provides a stress σ_w. Further, we have a larger exponent than the l bit floating-point exponent in the scaling of the initial condition $M(0)$ to reach $\theta(L)$, assuming $F_\mathrm{d} = 0$ for simplicity. We have two conditions up to small constants,

$$\sigma_\mathrm{w} = \frac{F_\mathrm{w}}{\pi r^2} \tag{11}$$

$$2L\sqrt{\frac{F_\mathrm{w}}{E\pi r^4}} > 2^{l-1}\ln 2. \tag{12}$$

Combining Eq. 11 and Eq. 12 produces a succinct approximate criterion of an extreme aspect ratio,

$$\frac{L}{r} > \underbrace{\frac{\ln 2}{4}}_{0.17}\sqrt{\frac{E}{\sigma_\mathrm{w}}}2^l. \tag{13}$$

We now find a computational success criterion applicable for arbitrary flexure geometry. We compare the scale factor $e^{\alpha L}$ with the maximum scaling representable with a floating-point number

$$e^{\alpha L} < 2^{2^{l-1}}. \tag{14}$$

Taking a logarithm and rounding down to the nearest lower number that is divisible by N, written as $\lfloor \cdots \rfloor_N$, we preserve the inequality to get

$$\alpha L < \beta = \lfloor 2^{l-1}\ln 2 \rfloor_N. \tag{15}$$

We chose $N = 50$ somewhat arbitrarily, but it is reasonable to do heavy rounding to ensure numerical stability. Now we make the bound applicable to simulating flexures with variable cross section geometry, $I(s)$ not constant. To overestimate the left hand side assume the characteristic exponential growth length scale $\alpha^{-1} = (\frac{F_\mathrm{w}}{EI(s)})^{-\frac{1}{2}}$ is at a minimum along the flexure's entire length L. To achieve this maximize the inverse length scale as $\max_s(\sqrt{\frac{F_\mathrm{w}}{EI(s)}})$. We now have a condition that determines the floating-point type needed for bending

$$\max_s(\sqrt{\frac{F_\mathrm{w}}{EI(s)}})L < \beta. \tag{16}$$

We provide Table 2, which informs one of the required floating-point precision for solving bending. For bending parameters from Table 1 we have $\alpha L = 760 > \beta_\mathrm{double} = 700$ so double precision is inadequate to calculate bending.

3. Numerical solution

To compute a bending solution that satisfies a desired end angle $\hat{\theta}$, we use the shooting method, treating the initial bending moment $M(0)$ as the shooting parameter p. The aim is to determine the value of p such that the numerical solution of the coupled ODEs yields $\theta(s = L) = \hat{\theta}$. We approach this in two stages, following the standard techniques outlined in *Numerical Recipes* [11].

In the first stage, we generate an initial guess for p based on the approximately linear response of the flexure in the small-angle regime. We begin by choosing $p^\star$, the smallest representable nonzero value in the floating-point format being used. Integrating the ODEs with this $p^\star$ yields a corresponding trial bending angle $\theta^\star = \theta_L(p^\star)$. Assuming small-angle linearity, the initial moment required to reach the target angle $\hat{\theta}$ can be estimated by scaling:

If $\hat{\theta} < \theta_\mathrm{small} = 0.1$ rad, we set

$$p = (\hat{\theta}/\theta^\star) \cdot p^\star; \tag{17}$$

Otherwise, to avoid complications from nonlinearities or trigonometric sign flips at large angles, we target θ_small instead and set

$$p = (\theta_\mathrm{small}/\theta^\star) \cdot p^\star. \tag{18}$$

This yields an initial condition that produces an end angle on the correct order of magnitude.

In the second stage, we refine this estimate using a nonlinear root-finding algorithm. Specifically, we apply the Anderson-Björck method [25] to solve $G(p) = \theta_L(p) - \hat{\theta} = 0$, using the interval $(p/32, 32p)$ around the linearized guess as the search domain. This two-step process with (1) estimating the exponent and (2) refining the mantissa enables robust and accurate determination of the required initial moment $M(0)$, even when p lies far below the limits of standard double precision. We give pseudocode for the algorithm we have described in this section as Algorithm 1.

3.1. Speed and error

The fixed step-size arbitrary precision Runge-Kutta 45 (APRK45) solver we implemented leverages fourth- and fifth-order 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. We generate stepwise error estimates. Errors for our parameters were much smaller than our tolerance with APRK45 as shown in Figure 4

Table 2: Lookup table for bending solver IEEE754 precision. $\max_s(\sqrt{\frac{F_\mathrm{w}}{EI(s)}})L < \beta$ means the precision is sufficient. The column ϵ gives the smallest full-precision positive value in the corresponding number format.

name	total bits	exp. bits, l	ϵ	β $\lfloor 2^{l-1}\ln 2\rfloor_{50}$
single	32	8	1×10^{-38}	50
double	64	11	2×10^{-308}	700
quadruple	128	15	3.3×10^{-4932}	11 350
octuple	256	19	$1.5 \times 10^{-78\,913}$	181 700

Algorithm 1 Algorithm to decide an initial condition p that gives a final bending angle $\hat{\theta}$. Subroutine APRK45(x) numerically integrates the bending ODE for initial condition x in arbitrary precision and returns the end angle $\theta_L(x)$.

```
1: procedure Bend to $\hat{\theta}$, vary $p$
2:     $\theta_\mathrm{small} \leftarrow 0.1$ rad
3:     $p \leftarrow$ smallest magnitude floating-point value
4:     $\theta^\star \leftarrow$ APRK45($p$)
5:     if $\hat{\theta} < \theta_\mathrm{small}$ then
           $p \leftarrow \frac{\hat{\theta}}{\theta^\star} p$
6:     else
           $p \leftarrow \frac{\theta_\mathrm{small}}{\theta^\star} p$
7:     $S \leftarrow (\frac{p}{32}, 32p)$
8:     return root(APRK45($x$) $- \hat{\theta}, S$)
```

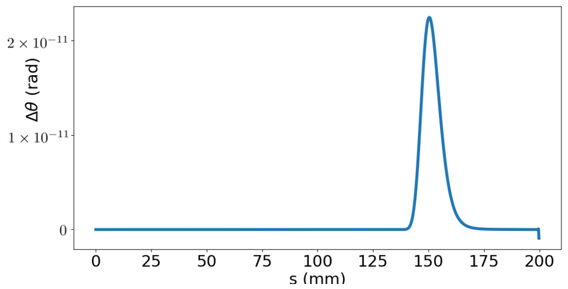


Figure 4: Local truncation error estimates, $\Delta\theta$, for $\theta(s)$ generated by the embedded Runge-Kutta 45 algorithm [26] when calculating bending for a flexure with nonuniform profile, see Figure 2. If local truncation errors of a numerical method are similar in magnitude to the relevant variables, the solver has surely failed. This error estimate is different from a direct comparison with analytical solutions because we have no such analytical solution for the non-constant flexure cross-section case.

for an analytically intractable bending problem. In Figure 4 the estimated error is associated with solving $F_\mathrm{d} = 0$ bending with 1000 uniform Runge-Kutta steps for a ribbon as in Figure 2 with bending geometry shown in Figure 1. These parameters can only be solved by shooting from the clamped end if we can represent $M(0) \sim 1 \times 10^{-596}$ N m. The time to do shooting for a single bend for the provided plots was about 10 seconds on a personal computer with Python implementation. By using a more carefully chosen or adaptive Runge-Kutta step size and implementing the integrator in a compiled language this runtime could be reduced to some milliseconds. The Python implementation used in this work is available at: https://github.com/usnistgov/BeamBending.

To validate our numerical solver, we compare it against the analytical solution for a constant cross section with $F_\mathrm{d} = 0$, where the solution involves only the hyperbolic sine function. We choose the test case of a constant circular cross section with the parameters shown in Table 1. Figure 3 shows this test case. This calculation requires a larger floating-point exponent than double. We calculate bending to final angles of 3 rad and 0.017 rad. Practically 3 rad is an extreme bend but serves to well exhibit the small angle approximation. When the angle of bending is small the numerical result and the approximate analytic constant cross sectionsolution should coincide.

4. Conclusions

This work enables more computationally efficient modeling and optimization of compliant mechanisms that appear in precision measurement devices. For extreme aspect ratio flexures, we conclude that it is feasible to calculate their deformation using semi-analytic methods. Euler-Bernoulli bending exhibits exponential growth, made clear by approximate solutions. Based on this behavior we forward the need to allocate more floating-point exponent bits to model extreme aspect ratio flexures. In order to properly model flexure designs, double precision values can be exchanged for higher precision values according to Table 2. To give a practical demonstration of floating-point limitations we expose, we use arbitrary precision floating-point to make science-relevant bending calculations beyond the limits of the double precision exponent.

References

[1] T. L. Thomas, V. Kalpathy Venkiteswaran, G. K. Ananthasuresh, and S. Misra. Surgical applications of compliant mechanisms: A review. *Journal of Mechanisms and Robotics*, 13(2):020801, 01 2021.

[2] Z. Wang and H. Hu. Analysis and optimization of a compliant mechanism-based digital force/weight sensor. *IEEE Sensors Journal*, 5(6):1243–1250, 2005.

[3] Z. Wu and Q. Xu. Survey on recent designs of compliant micro-/nano-positioning stages. *Actuators*, 7(1), 2018.

[4] L. Keck. *Flexure-based mechanism for a Kibble Balance*. PhD thesis, Ilmenau, Feb 2025. Dissertation, Technische Universität Ilmenau, 2024.

[5] L. Keck, G. Shaw, R. Theska, and S. Schlamminger. Design of an electrostatic balance mechanism to measure optical power of 100 kW. *IEEE Transactions on Instrumentation and Measurement*, 70:1–9, 2021.

[6] L. Keck, S. Schlamminger, R. Theska, F. Seifert, and D. Haddad. Flexures for kibble balances: minimizing the effects of anelastic relaxation. *Metrologia*, 61(4):045006, Jul 2024.

[7] S. Henning and L. Zentner. Analytical characterization of spatial compliant mechanisms using beam theory. In *Microactuators, Microsensors and Micromechanisms*, pages 61–76. Springer International Publishing, 2023.

[8] V. Platl and L. Zentner. An analytical method for calculating the natural frequencies of spatial compliant mechanisms. *Mech. Mach. Theory*, 175:104939, 2022.

[9] S. Henning and L. Zentner. Analysis of planar compliant mechanisms based on non-linear analytical modeling including shear and lateral contraction. *Mech. Mach. Theory*, 164:104397, 2021.

[10] IEEE standard for floating-point arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pages 1–84, 2019.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

[12] C. C. Speake. Anelasticity in flexure strips revisited. *Metrologia*, 55(1):114, Jan 2018.

[13] L. Keck, K. Arumugam, L. Chao, Z. Comden, F. Seifert, D.B. Newell, D. Haddad, and S. Schlamminger. Thoughts on the Kibble-Robinson theory. *Metrologia*, 62(2):025012, mar 2025.

[14] N. Perrone and R. Kao. A general nonlinear relaxation iteration technique for solving nonlinear problems in mechanics. *J. Appl. Math. Mech.*, 38(2):371–376, Jun 1971.

[15] J. M. Smith, Olver F. W. J., and D. W. Lozier. Extended-range arithmetic and normalized legendre polynomials. *ACM Trans. Math. Softw.*, 7:93–105, 1981.

[16] T. Fukushima. Numerical computation of spherical harmonics of arbitrary degree and order by extending exponent of floating point numbers. *J. Geod.*, 86(4):271–285, Apr 2012.

[17] A. Abad, R. Barrio, and Á. Dena. Computing periodic orbits with arbitrary precision. *Phys. Rev. E*, 84:016701, Jul 2011.

[18] P. Wang, G. Huang, and Z. Wang. Analysis and application of multiple-precision computation and round-off error for nonlinear dynamical systems. *Adv. Atmos. Sci.*, 23:758–766, 2006.

[19] E. Allen, J. Burns, D. Gilliam, J. Hill, and V. Shubov. The impact of finite precision arithmetic and sensitivity on the numerical solution of partial differential equations. *Math. Comput. Model.*, 35(11):1165–1195, 2002.

[20] F. Benz, A. Hildebrandt, and S. Hack. A dynamic program analysis to find floating-point accuracy problems. *SIGPLAN Not.*, 47(6):453–462, 2012.

[21] E. T. Barr, T. Vo, V. Le, and Z. Su. Automatic detection of floating-point exceptions. *SIGPLAN Not.*, 48(1), 2013.

[22] F. Johansson et al. mpmath: a Python library for arbitrary-precision floating-point arithmetic, 2023. Accessed: 2025-05-28.

[23] S. M. Aston, M. A. Barton, A. S. Bell, N. Beveridge, B. Bland, A. J. Brummitt, G. Cagnoli, C. A. Cantley, L. Carbone, A. V. Cumming, L. Cunningham, R. M. Cutler, R. J. S. Greenhalgh, G. D. Hammond, K. Haughian, T. M. Hayler, A. Heptonstall, J. Heefner, D. Hoyland, J. Hough, R. Jones, J. S. Kissel, R. Kumar, N. A. Lockerbie, D. Lodhia, I. W. Martin, P. G. Murray, J. O'Dell, M. V. Plissi, S. Reid, J. Romie, N. A. Robertson, S. Rowan, B. Shapiro, C. C. Speake, K. A. Strain, K. V. Tokmakov, C. Torrie, A. A. van Veggel, A. Vecchio, and I. Wilmut. Update on quadruple suspension design for Advanced LIGO. *Class. Quantum Gravity*, 29(23):235004, December 2012.

[24] T. J. Quinn, C. C. Speake, and R. S. Davis. A 1 kg mass comparator using flexure-strip suspensions: Preliminary results. *Metrologia*, 23(2):87, Jan 1986.

[25] N. Anderson and Å. Björck. A new high order method of regula falsi type for computing a root of an equation. *BIT Numer. Math.*, 13:253–264, 1973.

[26] E. Fehlberg. New high-order runge-kutta formulas with step size control for systems of first-and second-order differential equations. *Zamm-zeitschrift Fur Angewandte Mathematik Und Mechanik*, 44, 1964.