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. Flexures in torsion balances and precision mechanisms often exhibit extreme aspect ratios, leading to exponential scaling in Euler-Bernoulli bending models. Standard double-precision arithmetic cannot resolve the small initial conditions required for accurate solutions. We present a semi-analytic method—combining an efficient 1D bending model with adaptive Runge-Kutta-Fehlberg integration in arbitrary precision—that overcomes this limitation. We define a numerical criterion for when extended precision is necessary and provide an open-source Python implementation that remains stable across a wide range of flexure geometries.

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

1. Introduction

Compliant mechanisms have applications in medicine [1], transportation [2], robotics, precision positioning [3], and measurement systems [4–6]. Unlike traditional mechanisms with joints, they rely on elastic deformation and often incorporate loaded flexures as critical elements. Their basic behavior can be modeled using shear-free beam theory. To reduce computational demands, several authors have developed semi-analytic models of these systems [6–9].

We describe numerical methods that occupy the middle ground between fully analytical solutions and finite element analysis (FEA), which discretizes the three-dimensional geometry using a mesh. To distinguish these methods, we refer to them as *semi-analytic*, as they reduce the problem to solving a one-dimensional ordinary differential equation (ODE). These approaches offer significant computational advantages for modeling flexures.

Standard ODE solvers rely on fixed-precision floating-point arithmetic, and, as a consequence, these solvers can fail when modeling shear-free beam bending with an *extreme aspect ratio*. We refer to a flexure as having an 'extreme aspect ratio' when its length-to-thickness ratio leads to numerical underflow or instability during standard bending computations. We derive a formal criterion for this in Sec. 2.3. In such cases, the internal bending moment required to model the deflection can span many orders of magnitude, exceeding the dynamic range of the IEEE 754 double-precision format. To overcome this limitation, we implement a Runge-Kutta-Fehlberg integrator with arbitrary-precision floating-point arithmetic. This enables accurate modeling of flexures that lie beyond the capability of ODE solvers provided by common scientific programming languages. We also provide guidance on the level of numerical precision required for a given geometry.

1.1. The single flexure

The bending of a loaded single flexural element under torque and transverse force is one of the simplest systems to analyze. Yet even this simple case can challenge numerical methods, particularly because of precision loss and instability at extreme aspect ratios. Consider a flexure, illustrated in Figure 1, that supports a weight $F_\mathrm{w} = mg$ and is subjected to a transverse force F_d at its free end ($s = L$). The variable s measures position along the neutral axis from $s = 0$ at the clamped end to $s = L$ at the free end. The angle relative to the vertical is denoted $\theta(s)$, while the internal bending moment is $M(s)$. Without loss of generality, we set $\theta(0) = 0$ as illustrated in Figure 1. The deformation of the flexure is governed by coupled differential equations:

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

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

where $E(s)$ denotes the elastic modulus and $I(s)$ the second moment of area about the neutral axis. In the simplest case, $E(s), F_\mathrm{w}(s)$, and $F_\mathrm{d}(s)$ are constant, independent of s.

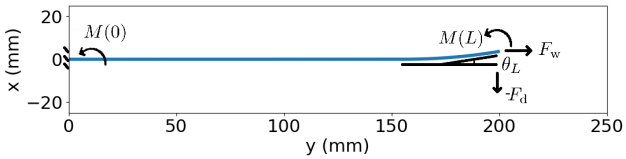


Figure 1: A loaded 200 mm long flexure is clamped at the top. The figure is shown rotated such that the flexure is lying on its side and gravity is acting on the $+y$ direction. A torque is applied such that the tangent at the end of the flexure is $\theta_L = 10°$. The coordinate s is measured along the neutral axis of the flexure and is distinct from the Cartesian axes.

The cross-sectional geometry of the unbent flexure is characterized by $I(s)$, the second moment of area about the neutral axis. For a circular cross section with radius $r(s)$ and a rectangular cross section with width b and thickness $h(s)$, $I(s)$ is given by

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

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

respectively. In general, as illustrated in Fig. 2, the cross-sectional parameters vary with s, requiring a numerical solution for the resulting bending curve.

We solve the flexure bending problem as a two-point boundary-value problem using a shooting method [10,11]: an initial guess for $M(0)$ is iteratively updated so that the terminal condition $\theta(s = L) = \theta_L$ is satisfied within a user-specified tolerance. The value $M(L)$ follows directly from the solution, and F_w is treated as a constant specified by the user. Standard IEEE-754 double precision can only represent normal numbers down to $\approx 2.2 \times 10^{-308}$. The example shown in Figure 1 requires resolving a bending moment ratio $M(0)/M(L) \sim 10^{-596}$, which lies well below this range and therefore underflows in binary64. Consequently, we employ higher-precision arithmetic beyond float64 to obtain a stable solution.

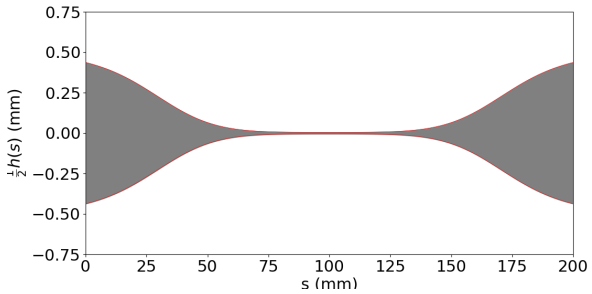


Figure 2: A thin planar flexure profile with parameters $E = 1.31 \times 10^5\,\mathrm{MPa}, F_\mathrm{w} = 1\,\mathrm{N}, b = 0.1\,\mathrm{mm}$. Standard double precision (float64) fails to maintain accuracy when applying clamp-sided shooting. The flexure thickness decreases by a factor of 100 at its waist.

A flexure with a constant cross section and no F_d provides a simple example of this numerical requirement, since its analytical solution is given by Speake (see Eq. (1) and Eq. (2) in [12]). It is

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

The ratio $M(0)/M(L)$ becomes exponentially smaller as I decreases. This scaling soon exceeds the dynamic range of IEEE-754 double precision (float64), causing the shooting method to fail.

Although analytical solutions exist for flexures with constant cross sections, practical designs often involve non-uniform geometries, necessitating numerical approaches. Finite element analysis (FEA) is a general method for such problems, but solving the underlying differential equations directly offers substantial computational advantages: the problem reduces to one dimension, avoiding full 3D meshing and enabling faster iteration during design. Among numerical strategies, ODE-relaxation methods [11, 13] and shooting techniques have both been applied successfully. The shooting method, in particular, has proven effective for flexures with moderate aspect ratios [6, 9], and we extend this approach to the extreme-aspect-ratio regime below.

1.2. The shooting method

To compute the neutral axis of a deflected flexure, we numerically integrate the coupled bending equations (Eqs.1–2). We use the shooting method to enforce boundary conditions, such as clamping and desired end angle $\hat{\theta}$. The shooting method transforms a boundary-value problem into a series of initial-value problems [10, 11], adjusting the initial condition, e.g., $p = M(0)$, until the boundary conditions are met. This iteration is formulated as a root-finding problem for the auxiliary function

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

where $\theta_L(p)$ and $\hat{\theta}$ are the calculated and desired end angles, respectively. The shooting method, when implemented with float64, fails for flexures with extreme aspect ratios. In such cases, the required shooting parameter p underflows, preventing the root from being found. When this occurs, $\theta(L)$ diverges to nonphysical values (larger than π), and the end-angle condition cannot be satisfied. Figure 2 shows a geometry where this failure occurs. For simplicity, we set $F_\mathrm{d} = 0$, making $M(0)$ the sole shooting parameter that must match the desired boundary condition, $\theta_L(p) = \hat{\theta}$.

1.3. Floating-point Representation

Limitations of numerical methods that rely on fixed-size floating-point arithmetic are well known. Such issues arise in many contexts: for example, the restricted range of floating-point exponents has been documented in calculations involving Legendre polynomials [14, 15], and insufficient precision can cause numerical orbits to lose periodicity [16]. Double- or single-precision arithmetic may also fail to correctly resolve the behavior of chaotic systems, such as selecting among solutions of the Lorenz attractor [17]. More broadly, floating-point limitations can produce spurious solutions that are purely numerical artifacts [18]. In some cases, such errors can be detected and diagnosed automatically using dynamic program analysis techniques [19, 20].

A floating-point number N is represented in a computer using three components: a sign bit ($\pm$), an unsigned integer δ (bit width d) encoding the mantissa (or significand), and a signed integer λ (bit width l) encoding the exponent. For *normalized* numbers, the significand is interpreted as $1+\delta/2^d$ so the value of the floating-point number is

$$N = \pm(1 + \frac{\delta}{2^d}) \cdot 2^\lambda. \tag{7}$$

In the IEEE standard [21], double precision binary (float64) is encoded with 64 bits: 1 sign bit, $l = 11$ exponent bits, and $d = 52$ mantissa bits. The exponent is stored using a bias of $1023 = 2^{(l-1)} - 1$, resulting in a range $-2^{(l-1)} + 2 = -1022 \leq \lambda \leq 2^{(l-1)} - 1 = 1023$. spanning the representable values from $\approx 2.2 \times 10^{-308}$ to $\approx 1.8 \times 10^{308}$.

Subnormal numbers extend the range down to 5×10^{-324}. These occur when the exponent field is zero, which corresponds to $\lambda = -1022$. In that case, leading 1 is no longer implicit in the significant, and

the value is computed as

$$N = \pm\frac{\delta}{2^d} \cdot 2^{-1022}. \tag{8}$$

Subnormal numbers are a mechanism to extend the representable range below the smallest normalized value, but they come at the cost of reduced precision and are generally unsuitable for accurate numerical computation. Therefore, in this article, we limit ourselves to the dynamic range provided by normalized numbers.

We define the *dynamic range* of a floating point number with l exponent bits as the ratio of one to the smallest positive normal number. It is $2^{2^{l-1}}$. This definition of the dynamic range ignores the floating point numbers with positive exponents, which is appropriate for the bending problem, since the end angle $\theta(L)$ is of order unity and hence $\theta(s)$ for $s < L$ utilizes the negative exponent range. With this definition, the problem can be solved without rescaling, see Section 1.4.

Quadruple precision (float128) uses $l = 15$ and, hence, provides a much larger dynamic range of $\approx 10^{4932}$, sufficient for the thin flexures considered here.

This larger exponent range of float128 enables the shooting method to resolve initial conditions that produce the desired end angles. We therefore employ the Python library mpmath [22], which supports arbitrary-precision floating-point arithmetic, ensuring that the dynamic range is big enough for the physical problem, see Section 2.3.

1.4. Why Rescaling Cannot Replace Extended Precision

A common strategy to address numerical errors arising from extreme magnitudes is to rescale variables to a dimensionless form. For example, rather than computing the moment $M(s)$ in N m, one might work with the dimensionless ratio $M(s)/M_n$, where M_n is a reference moment, e.g., 1×10^{-20} N m. As shown in Eq. 5, the ratio $M(L)/M(0)$ between the free and fixed ends of the flexure can span hundreds of orders of magnitude. A single global scaling factor does not compress this exponential range, as it cancels out in relative calculations.

A possible solution would be to introduce a piecewise scaling function $\gamma(s)$ that adapts locally to the flexure geometry. However, this complicates the treatment of boundary conditions and may disrupt continuity. While such an approach could be developed, it is beyond the scope of this article.

Scaling the angle is even more complicated because of the non-linearity of the trigonometric function. In Taylor approximation, a $\theta(s)$ rescaled as $\theta(s) = \gamma\theta_n(s)$, yields

$$\cos\theta(s) = \cos\Big(\gamma\theta_n(s)\Big) \approx 1 - \frac{1}{2}\gamma^2\theta_n(s)^2. \tag{9}$$

There is no obvious rescale factor that works for the entire flexure. The correct implementation of piecewise scaling is as difficult as coding the trigonometric functions from scratch.

In summary, rescaling alone cannot address the exponential dynamic range challenge, whereas increasing the floating-point exponent width offers a more robust and straightforward solution.

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}	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}	m^4
α	$\sqrt{F_\mathrm{w}/(EI)}$	1.27×10^{3}	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}	N/m^2
$\sqrt{E/\sigma_\mathrm{W}}$		7.90	
L/r		3.00×10^{3}	

2. Analytical Solutions

Analytical solutions help explain why bending thin flexures becomes numerically difficult. They show how moments and angles can grow exponentially, making standard floating-point methods unreliable.

2.1. Large Exponents in Bending

The moment at the clamped end of the flexure is vanishingly small, yet a finite value is required to produce the correct shape from the coupled differential equations. If $F_\mathrm{d} = 0$ and $M(0) = 0$, the differential equations yield a straight flexure. To achieve bending of the flexure to either side, a small symmetry-breaking moment must be present. The solution in Figure 1 was determined with a very small initial value $M(0)$.

For the flexure defined by the parameters in Table 1, a solution for $\theta(s)$ can be found by initiating the calculation with a very small moment $M(0)$. For example, achieving $\theta(L) = 1\,\text{rad}$ requires $M(0) \approx 1\times10^{-330}\,\text{N m}$, a value beyond the representable range of float64. The angles of the neutral fiber and bending moments exhibit exponential growth, which amplifies the limitations of floating-point precision.

2.2. Small-angle Solutions

Solutions for beam deformation involving hyperbolic trigonometric functions are well-documented in textbooks and are widely used in practical research [12,24]. Here, we revisit these solutions in a form that highlights their exponential nature. Any solution with $\theta(0) = 0$ will match the small-angle behavior over part of the flexure length. Therefore, the numerical limitations revealed by small-angle solutions also apply to large-angle bending problems.

In this example, we consider a constant-width geometry with $I(s) = I_0$ and $\theta(0) = 0$. We focus on the case where $\alpha L \gg 1$. We adapt the solutions of Speake [12], replacing his coordinate x with s, which is justified under the small-angle approximation. For $\alpha L \gg 1$, $\sinh(\alpha L) \approx \cosh(\alpha L) \approx \frac{1}{2}e^{\alpha L}$. Applying these to Eqs. (1) and (2) of [12] yields

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

Using the compliance matrix in [12] gives

$$\theta(L) \approx \frac{F_\text{d}}{F_\text{w}}(1 - 2e^{-\alpha L}) + \frac{\alpha M(L)}{F_\text{w}}. \tag{11}$$

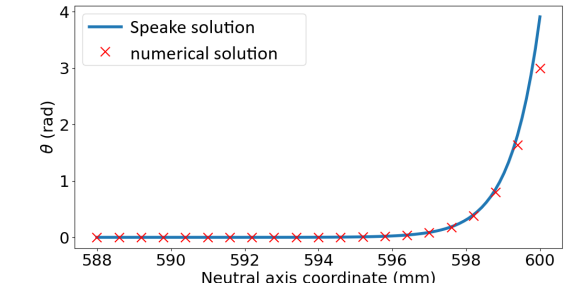
Substituting Eq. (10) into Eq. (11) and ignoring terms that are not exponentially large in αL yields

$$\theta(L) \approx \frac{1}{2F_\text{w}}\left(\alpha M(0) - F_\text{d}\right)e^{\alpha L}. \tag{12}$$

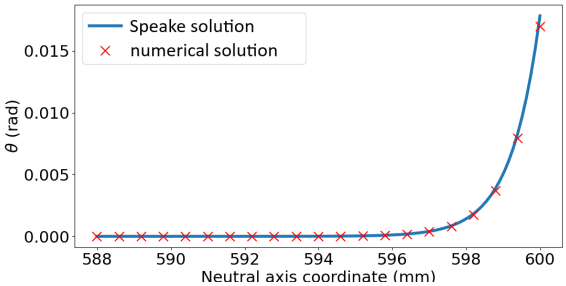
This explicitly shows the exponential sensitivity of $\theta(L)$ to the tiny initial moment $M(0)$ and the applied transverse load F_d, which drives the floating-point exponent limitations discussed above. Figure 3 compares the exponential small-angle analytical solution for $\theta(s)$ with numerical calculations.

2.3. When Is a Flexure Considered Extreme Aspect Ratio?

The analytical solution above allows us to formally define the conditions under which a flexure is considered to have an extreme aspect ratio, relative to the available floating-point exponent range.



(a) large-angle bend



(b) small-angle bend

Figure 3: Bending of a loaded flexure under torque, using the parameters in Table 1, shown over the final 12 mm of the flexure. Panel (a) shows a large deflection, while panel (b) shows a small deflection. In panel (a), the breakdown of the small-angle approximation is evident as the small-angle solution deviates from the numerical result.

Consider a flexure with constant circular cross section loaded by a force F_w, resulting in a stress

$$\sigma_\text{w} = \frac{F_\text{w}}{\pi r^2} \tag{13}$$

According to Eq. 12, the dynamic range required for the computation scales as $\exp(\alpha L)$. We define a flexure as having an extreme aspect ratio if this required dynamic range exceeds the representable range of a binary floating-point number with exponent bit width l, i.e.,

$$e^{\alpha L} > 2^{2^{l-1}} \quad \longrightarrow \quad \alpha L > 2^{l-1}\ln 2. \tag{14}$$

Here, the $2^{2^{l-1}}$ reflects the maximum dynamic range permitted by the floating-point exponent.

For a circular cross section, this implies

$$\alpha L = L\sqrt{\frac{4F_\text{w}}{E\pi r^4}} > 2^{l-1}\ln 2. \tag{15}$$

Combining Eq. 13 and Eq. 15 gives a compact criterion for when a flexure exceeds the dynamic range

of floating-point formats:

$$\frac{L}{r} > \underbrace{\frac{\ln 2}{4}}_{0.17} \sqrt{\frac{E}{\sigma_\mathrm{w}}} 2^l. \tag{16}$$

A calculation remains stable and accurate if

$$\alpha L < 2^{l-1} \ln 2. \tag{17}$$

To provide extra margin for numerical stability, we round the right side of the inequality down to the nearest multiple of N_margin denoted by $\lfloor \cdots \rfloor_{N_\mathrm{margin}}$. Hence, we recommend

$$\alpha L < \underbrace{\lfloor 2^{l-1} \ln 2 \rfloor_{N_\mathrm{margin}}}_{\beta}, \tag{18}$$

with $N_\mathrm{margin} = 50$. For example, for $l = 11$, we obtain $2^{l-1} \ln 2 = 709.8$ and, hence, $\beta = \lfloor 2^{l-1} \ln 2 \rfloor_{50} = 700$.

For a flexure with variable cross section, α becomes a function of arc length, and the inequality in Eq. (18) must hold for its maximum value along the flexure:

$$\max \left(\sqrt{\frac{F_\mathrm{w}}{EI(s)}} \right) L < \beta. \tag{19}$$

Table 2 summarizes the floating-point precision required to satisfy this condition for various β (i.e., exponent bit–widths l). For the bending parameters in Table 1, we find $\alpha L = 760 > \beta_\mathrm{double} \approx 700$, so IEEE-754 double precision is inadequate for this case.

3. Practical Tips for the Shooting Method

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 goal is to determine the value of p such that the numerical solution of the coupled ODEs satisfies $\theta(L) = \hat{\theta}$. We follow a two-stage approach, based on standard techniques from numerical analysis [10, 11].

In the first stage, 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 the ODE system remains linear in this regime, the initial moment required to reach the target angle $\hat{\theta}$ can be estimated by scaling [10]:

If the desired angle is small, $\hat{\theta} < \theta_\mathrm{small}$ with $\theta_\mathrm{small} = 0.1\,\mathrm{rad}$, we set

$$p^\dagger = \frac{\hat{\theta}}{\theta^\star} p^\star. \tag{20}$$

Otherwise, to ensure convergence despite nonlinearity or sign change of trigonometric functions at large angles, we target θ_small instead and set

$$p^\dagger = \frac{\theta_\mathrm{small}}{\theta^\star} p^\star. \tag{21}$$

This yields an initial condition that produces an end angle of 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 $G(p) = \theta_L(p) - \hat{\theta} = 0$, with $p \in [p^\dagger/64, 64p^\dagger]$ around the linearized guess as the search domain.

We successfully tested this two-step process for robustness with a variety of flexure geometries. The first step results in a small, but correct search range for the second step, that then provides the final, possibly tiny, $M(0)$. The algorithm works for arbitrary floating point precision implemented by mpmath. The pseudo-code for the algorithm is given below.

Algorithm 1 Two step algorithm to find an initial condition p, e.g. $M(0)$, that gives a final bending angle $\hat{\theta}$. Subroutine APRKF45(p) numerically integrates the bending differential equation for initial condition p in arbitrary precision and returns the end angle $\theta_L(p)$.

procedure BEND TO $\hat{\theta}$, VARY p
  $\theta_\mathrm{small} \leftarrow 0.1$ rad
  $p^\star \leftarrow$ smallest magnitude floating-point value
  $\theta^\star \leftarrow$ APRKF45($p^\star$)
  if $\hat{\theta} < \theta_\mathrm{small}$ **then**
    $p^\dagger \leftarrow \frac{\hat{\theta}}{\theta^\star} p^\star$
  else
    $p^\dagger \leftarrow \frac{\theta_\mathrm{small}}{\theta^\star} p\star$
  $S \leftarrow [\frac{1}{64} p^\dagger, 64p^\dagger]$
  return root(APRKF45(p) $- \hat{\theta}$) for $p \in S$

3.1. Computational Performance and Numerical Error

The fixed step-size arbitrary precision Runge-Kutta-Fehlberg 45 (APRKF45) solver we implemented leverages fourth- and fifth-order coefficients to provide calculable and sufficiently small errors. The cross-sectional moment of inertia $I(s)$ is sampled at fixed intervals and interpolated using cubic splines for use by the solver. We also implemented an arbitrary precision Runge-Kutta-Fehlberg 89 solver, with eighth- and ninth-order coefficients [26, 27], to enable low compute time bending calculations over a wider range of flexure cross sections. We generate stepwise error estimates. Errors for our parameters were much smaller than our tolerance with APRKF45 as shown in Figure 4 for an analytically intractable bending

Table 2: IEEE 754 precision requirements for the bending solver. The condition $\max_s(\sqrt{\frac{F_w}{EI(s)}})L < \beta$ that the selected floating-point format provides sufficient dynamic range for accurate computation. The column labeled ϵ lists the smallest positive normalized number representable in each 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

problem. In Figure 4 the estimated error is associated with solving $F_d = 0$ bending with 1000 uniformly spaced integrator 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 $M(0) \sim 10^{-596}$N m. To reiterate for simplicity of implementation we always use half of the floating-point dynamic range. Such a choice makes implementing trigonometric functions which are scale sensitive trivial, see Section 1.4. We verified numerical stability by decreasing the step size and confirming convergence of the bending geometry. This runtime could be reduced to milliseconds by adopting an adaptive step-size integrator and reimplementing the solver in a compiled language such as C++ or Fortran. The Python implementation used in this work is available at: https://github.com/usnistgov/BeamBending.

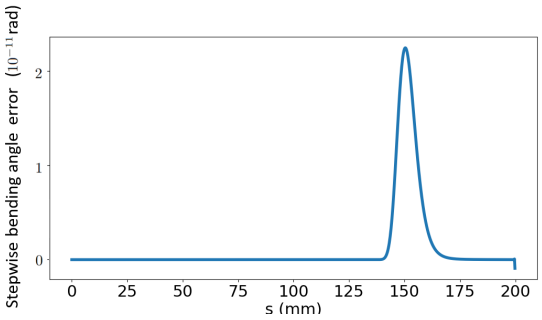


Figure 4: Local truncation error estimates at 1000 Runge-Kutta steps, $\Delta\theta$, for $\theta(s)$ generated by the embedded Runge-Kutta-Fehlberg 45 algorithm [26] when calculating bending for a flexure with nonuniform profile, see Figure 2. If local truncation errors of a numerical method are similar 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.

To validate our numerical solver, we compare it against the analytical solution for a constant cross section with $F_d = 0$, where the solution involves only the hyperbolic sine function. As a test case, we use a flexure with a constant circular cross section and the parameters listed in Table 1. This calculation requires a floating-point exponent range beyond double precision. Figure 3 shows results for two target bending angles: 3 rad and 0.017 rad. The former represents an extreme deflection and clearly illustrates the breakdown of the small-angle approximation. In the latter case, the numerical and analytical solutions agree closely, as expected.

4. Conclusions

This work presents a robust and efficient framework for simulating compliant mechanisms with extreme aspect ratio flexures. These are cases where standard double-precision methods fail because of the exponential sensitivity in Euler-Bernoulli bending. We provide a clear numerical criterion to determine when an extended exponent range is necessary.

Using arbitrary-precision Runge-Kutta-Fehlberg integrators, we demonstrate stable and accurate modeling in regimes where conventional solvers break down. Our open-source Python implementation supports fast and reliable flexure design across a wide range of geometries, making it a valuable tool for precision measurement and compliant mechanism applications.

Acknowledgments

We thank our NIST colleagues Jack Manley and John Lawall for their valuable feedback on the manuscript.

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, January 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, February 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, July 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] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. 3rd edition, 2002.

[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, January 2018.

[13] 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, June 1971.

[14] 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.

[15] 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, April 2012.

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

[17] 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.

[18] 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.

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

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

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

[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, January 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.

[27] E. Fehlberg. Classical fifth-, sixth-, seventh-, and eight-order runge-kutta formulas with stepsize control. Technical Report NASA TR R-287, National Aeronautics and Space Administration, Washington, D.C., 1968.

[28] 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, March 2025.