Large Exponents in Shooting Method for Thin Flexures

authors go here – Ben is first

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

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

in: The International Journal for Numerical Methods in Engineering (short form),Applied Numerical Mathematics. Or metrology journal?

Abstract.

In designing single-side clamped flexures as part of torsion balances or pendulums for scientific use structures become thin and semi-analytic calculations of their bending become infeasible with standard double precision. Semi-analytic calculations can be more efficient than finite element methods allowing faster design optimization. We exhibit simple analytical results which show that failure of double precision semi-analytic bending simulation is due to small angle exponential growth of the bending angle. We test an arbitrary precision implementation of Runge-Kutta 45 integration resolving cases where standard double precision bending model implementations fail.

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

1. Introduction

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 [1–4]. We study the numerical solution of a single flexural element. Consider a flexure suspending a weight, $F_w = mg$, and additionally a deflecting force F_s that acts at the end of the flexure. The flexure angle $\theta(s)$ and moment $M(s)$ a distance s along the neutral axis depend on boundary conditions and uniform elastic modulus E. We assume the flexure is clamped at $s = 0$ with an initial tangent angle and bending moment. The flexure has a length along its neutral axis L. The geometry of deformation can be determined by the relations

$$\frac{dM}{ds} = F_w(s)\sin(\theta(s)) + F_s(s)\cos(\theta(s)) \tag{1}$$

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

on the moment and angle [4]. For our application we assume $E(s), F_w(s)$ and $F_s(s)$ constant. An example geometry, converted to cartesian coordinates, is shown in Figure 1. The geometry of the unbent flexure's cross

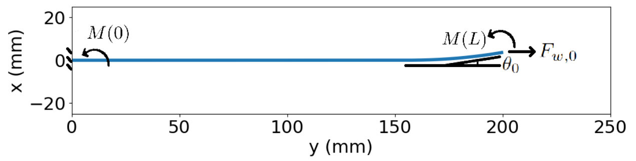


Figure 1: End torque bending to $\theta_0 = 10°$ of a flexure with a rectangular cross section with varying $h(s)$. The parameters are those of Figure 2.

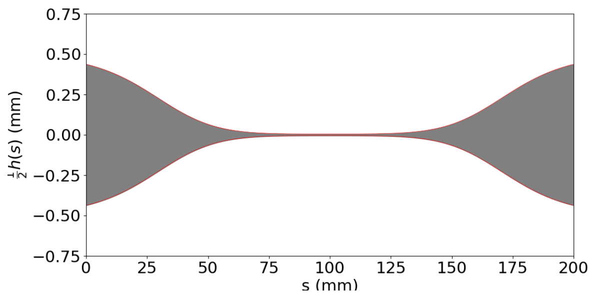


Figure 2: A thin planar flexure profile where for reasonable parameters ($E = 131 \times 10^9$Pa, $F_w = 1$N, $b = 0.1$mm) float64 is not sufficient when applying clamp-sided shooting. The flexure becomes 100 times smaller at its waist. Such a flexure should not fail to support the load F_w if made of a high yield strength alloy such as precipitation hardened copper beryllium (> 1000 MPa).

section is captured by $I(s)$ the second moment of area of the plane perpendicular to the neutral axis. For the cases of a circular cross section and rectangular cross section there are formulae for second moments of area,

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

$$I_\square(s) = \frac{bh(s)^3}{12}. \tag{4}$$

We show a neutral axis perpendicular view of a varying rectangular cross section flexure we studied that is characterized by Equation 4 in Figure 2.

A numerical solution can be obtained by using standard double precision (64 bits, also referred to as float64) ordinary differential equation (ODE) solvers that consider a single sided boundary condition. The boundary condition on the other end of the flexure, $\theta(L) = \theta_0$, is obtained by the shooting method. For example, a final bending angle of $\theta_0 = 10°$ of a flexure is apparent at the curled end in Figure 1. The angle the flexure's tangent makes with the y axis at its end is θ_0 and thus is useful as a physically measurable boundary condition. Precision beyond float64 was used to solve for the deformation in Figure 1 because when numerically integrating the geometry $M(0) \sim 10^{-596}$ which is outside the capability of float64. Without relying on correctness of numerical methods, for constant $I(s)$, we have small angle analytical solutions [Eq. 1, Eq. 2, 5] with $M(L) = (M(0) - \frac{F_s}{\sqrt{\frac{F_w}{EI}}})\frac{e^{\sqrt{\frac{F_w}{EI}}L}}{2}$. For large enough $\sqrt{\frac{F_w}{EI}}L$ many bits are needed in the floating-point exponent to represent $M(0)$.

The purpose of this article is to numerically solve the bending of very thin flexures. For constant cross-sections, the system of differential equations can be solved analytically. In practice, however, the 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 as an ODE, the latter much faster. Intuitively this is because the Runge-Kutta method requires a linear number of steps in the size of the input, samples of $I(s)$. Hence, an ODE-based solver can speed up the development cycle for compliant mechanisms. An alternative approach to solving this problem as an ODE would be to consider relaxation methods [6, 7]. For flexures that are not super thin, Runge-Kutta solvers have been used previously [1, 4]. Typically [1, 4] the ODE integration step in the shooting method starts from the clamped end ($s = 0$) where boundary conditions are obvious.

1.1. The shooting method

The shooting method reduces a boundary value problem to an initial value problem, where the end conditions can be iteratively satisfied by varying initial conditions [6]. Iteration can be cast as root finding on an auxilliary function which encodes the desired boundary condition. In our case the root finding is on an auxilliary function G,

$$G(M(0), F_s) = \theta(L, M(0), F_s) - \theta_0. \tag{5}$$

Evaluating the auxiliary function once requires numerically integrating the ODE. Here the parameter that is varied is either $M(0)$ or F_s, never both. This approach is effective but when implemented with float64 it fails in very thin cases. We define thin further in terms of geometric, loading, and material properties of the flexure. Figure 2 shows a flexure geometry where a float64 solver would fail.

1.2. 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 [8, 9]. Other examples such as Abad et al. show a unacceptably greater deviation from exact periodicity of numerical orbits with too few decimals of precision [10]. Double and single precision can be insufficient for numerically determining a choice of Lorenz attractor [11]. Limitations of floating-point representation can also lead to the appearance of anomalous solutions which are only exhibited by the numerically approximated system [12]. In some cases finding such errors can be automated using dynamic program analysis [13, 14].

A floating-point number N is represented on a computer by a natural number, an integer and a sign ($\pm$) as $N = \pm\frac{d}{D} \cdot 2^l$. Multi-bit values d and l are called the mantissa and the exponent. The mantissa is normalized to be less than one with D. As the name implies, float64 uses 64 bits to encode the binary N, the most significant bit is the sign bit. The sign bit is followed by an 11-bit wide exponent and a 52-bit wide mantissa [15]. 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 very thin beams. Quadruple precision or float128 extends this range enough for the application discussed here. Float128 uses a 15-bit wide exponent and a 112-bit wide mantissa [15]. Together with the sign bit, the bit widths add up to 128 bits. Now, $-16382 \leq l \leq 16383$, which limits the exponents to 10^{-4932} and 10^{4932}. One can represent slightly smaller values than listed by setting the mantissa to be small relative to D, trading precision for magnitude. However full loss of mantissa precision spoils the algorithm we use to solve bending. We use the library mpmath for Python to perform our calculations. It implements floating-point calculations with an exponent range that exceeds the practical application of the bending model.

One may wonder why we cannot simply rescale the variables. First there is an inherent unitless scale in the coupled ODE variables. Calculating $\sin\theta(s)$ introduces the range $[0, 2\pi)$. Adapting between the rescaling and $[0, 2\pi)$ would require less straightforward implementation. Secondly, increasing the size of the floating-point representation is effectively the same as allowing extra rescaling of the variables. Since rescaling needs to be done dynamically, we would need to carry around extra data representing the order of magnitude of rescaling as we integrated the ODE. This is the entire point of the floating-point exponent bits.

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 something is so small it goes unmeasured. Numerically, this scale is of practical use because growth of the flexure angle is exponential when the angle is small. Without this small nonzero initial $M(0)$ or a side force F_s 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 small initial value $M(0)$. With unbroken symmetry ($\theta(0) = 0, M(0) = 0$), Equations 1 and 2 have null right-hand side. The result is zero bending for all s which carries over to numerical solutions.

To acheive a bent solution in our numerical solution we may introduce a negligible moment at the clamp. We must represent such small values in our ODE solver which is not possible with float64. For example, $M(0) \sim 10^{-331}$ to achieve a final bending angle of order unity radians with physical parameters given in Table 1. If $F_s \neq 0$ then a similar exponential growth and floating-point limitation may be observed.

To build intuition note Equation 2 entails that when the geometry becomes extremely thin ($I(s)$ is

Table 1: Parameters of a circular cross section flexure used in [16].

Parameter	Value	Unit
E	7.3×10^{10}	Pa
L	6×10^{-1}	m
r	2×10^{-4}	m
F_w	1.47×10^{2}	N

then small) the rate of change of $\theta(s)$ will become extremely large. We will explicitly show the initial condition must be exponentially small in terms of the physical parameters of the flexure to achieve a measurable end angle. This, in turn, explains why the initial values in semi analytic bending are extremely small numbers outside of float64's capability.

2.2. Small angle solutions

Such hyperbolic trigonometric functions as solutions to beam deformation have been well documented in textbooks and are often used in practical research [5, 17]. Here we need to look at these approximate solutions in a form that is amenable to exposing exponent limitations.

To illustrate beyond a qualitative understanding that there is an exponential increase present in bending, consider the solutions where θ is small. It is immediately clear via Equations 1 and 2 that linear approximation yields exponential solutions when $F_s = 0$.

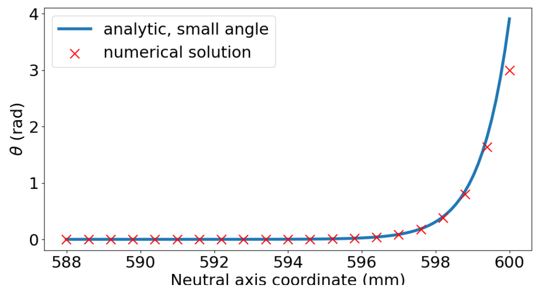
Now more formally the angle is small for part of all bending geometries with $\theta(0) = 0$. Additionally, consider a constant width geometry $I(s) = I_0$. Then we can draw on the small-angle and constant cross section solutions well organized by Speake [5]. Note Speake uses x and y coordinates, we use θ, s. Happily x and s are equivalent in linear bending which is enough to quantify numerical failures.

We evaluate the compliance matrix form given by Speake to find the final bending angle [5]. To expose exponent limitations we consider large growth in $M(s)$ and $\theta(s)$ so that $\alpha L = \sqrt{\frac{F_w}{EI_0}}L >> 1$ which means $\sinh \alpha L \approx \cosh \alpha L \approx \frac{1}{2}e^{\alpha L}$ and $\tanh \alpha L \approx 1$.

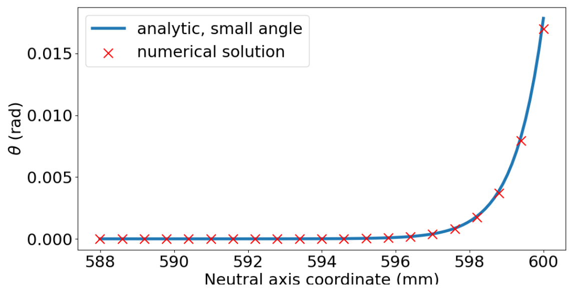
From [Eq. 1, Eq. 2, 5] we find a secondary equation under the above approximations,

$$M(L) = (M(0) - \frac{F_s}{\alpha})\frac{e^{\alpha L}}{2}. \qquad (6)$$

We can therefore focus on the exponential term as the term in parenthesis is assumed constant along the flexure. We have shown a large scaling for $M(s)$. Further $\theta(s)$ and $M(s)$ are related by a single derivative. If the scaling in $M(s)$ is large then the same is true for $\theta(s)$.W



(a) Large angle bend



(b) Small angle bend

Figure 3: Bending angle near the end of a torque bending flexure with parameters as in Table 1. The small angle solution deviates from the numerical solver as the bend angle becomes sufficiently large.

Now we observe the final bending angle [Eq. 3, 5]

$$\theta_0 = \frac{F_s}{F_w}(1 - 2e^{-\alpha L}) + \frac{\alpha M(L)}{F_w}. \qquad (7)$$

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 needeed to acheive a small θ_0,

$$\theta_0 = (\frac{M(0)}{\sqrt{F_w EI_0}} - \frac{F_s}{F_w})\frac{e^{\sqrt{\frac{F_w}{EI_0}}L}}{2}. \qquad (8)$$

Definition of thin From the exponential analytical solutions we can define what we mean by a thin flexure. Assume a circular cross section flexure is loaded with force F_w which provides a stress σ_y. Further we have a larger exponent than the b bit floating-point exponent in the scaling of the initial condition $M(0)$. We have two conditions up to respectively small proportionality

Table 2: Lookup table for bending solver IEEE754 precision. $max_s(\sqrt{\frac{F_w}{EI}})L - \ln(\theta_0) < \beta$ means the precision is sufficient. Maximizing over the flexure, $max_s(\dots)$, provides a worst case estimate for deciding failure of the numerical precision.

IEE754	β
Single	50
Double	700
Quadruple	11 350
Octuple	181 700

constants, and small logarithms, such as $\ln(\theta_0)$,

$$\sigma_y = \frac{F_w}{r^2} \tag{9}$$

$$L\sqrt{\frac{F_w}{Er^4}} >> 2^b. \tag{10}$$

We then find a condition that may define a thin flexure as relevant here:

$$\frac{L}{r} >> 2^b\sqrt{\frac{E}{\sigma_y}}. \tag{11}$$

Taking these calculations through without dropping small factors and rounding down by units of 50 we provide Table 2 which informs one of the required floating-point precision for solving bending.

We use the sine only ($F_s = 0$) constant cross section analytical solution to validate our numerical solver implementation. These solutions exhibit exponential growth and in practical scenarios can not be numerically approximated with double precision arithmetic. We choose the test case of a constant circular cross section. Its parameters are given in Table 1. We calculate bending to final angles of 3 radians and 1°. Practically 3 radians is an extreme bend but serves to well exhibit the small angle approximation. When the angle of bending is still small the numerical result and the approximate analytic solution should coincide. Figure 3 shows this test case.

Calculating the scale factor $e^{\alpha L}$ for parameters of the test case (provided in Table 1) we find order 10^{330}. Therefore we could not have numerically approximated this solution with double precision.

3. Shooting when the exponent is large

We assume the final bending angle is less than $\frac{\pi}{2}$ (extended to π for $F_s = 0$) to avoid an oscillating regime on $\theta(s)$ and $M(s)$ which could make shooting for solutions more difficult. The shooting method alone is sufficient to find bending solutions. We conceptually separate shooting into two stages, first we provide an initial guess based on linearity under small angle bending. Then we perform nonlinear shooting. Roughly speaking for bending the first stage will yield the exponent of the initial condition, the second yields the mantissa.

3.1. Small angle step

We set the initial condition $M(0)$ or F_s to be as small as possible without sacrificing precision with the arbitrary precision Runge-Kutta 45(APRK45) solver. We call this the trial solution. We determine an ancillary end bending angle $\theta^\star$ for the trial solution. 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 trial solution to better achieve θ_0. This can be seen as a trivial case of shooting where the ODE is linear [6]. In studying nonlinear bending we must proceed further.

3.2. Shooting

A shooting step is done starting with the trial solution derived guess for either F_s or $M(0)$ with the APRK45 solver for side force significant and side force insignificant cases respectively. We have found that the Anderson-Björck root finding method [18] was best applied to the shooting step, rapidly converging on the correct initial condition for a wide range of cases. We give pseudocode for the algorithm we have described in this section as Algorithm 1. Shooting for F_s instead of $M(0)$ is identical except in which parameter is modified.

Algorithm 1 Semi-analytic bending

```
1: procedure SHOOT $M(0)$
2:     $M(0) \leftarrow$ smallest magnitude floating-point value
3:     $\theta^\star \leftarrow$ APRK45($M(0)$)
4:     if $\theta_0$ is small then
           $M(0) \leftarrow \frac{\theta_0}{\theta^\star}M(0)$
5:     else
           $M(0) \leftarrow \frac{\theta_{small}}{\theta^\star}M(0)$
6:     $S \leftarrow (\frac{M(0)}{32}, 32M(0))$
7:     return root($\theta_0$ − APRK45($x$), $S$)
```

3.3. Speed and error

The fixed step-size 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. Errors for our parameters were much smaller than our tolerance with

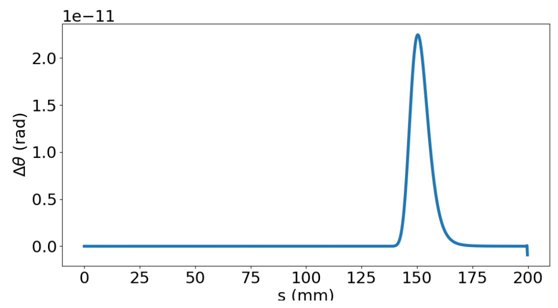


Figure 4: Point-wise error estimates for $\theta(s)$ solving $F_s = 0$ bending with 1000 uniform Runge-Kutta steps for a ribbon as in Figure 2 with bending geometry shown in Figure 1.

APRK45 as shown in Figure 4 when run on a flexure with parameters as in Figure 2. These parameters can only be solved by shooting from the clamped end if we can represent $M(0) \sim 10^{-596}$. The time to do shooting for a single bend for the provided plots was about $5 - 50$ seconds on a personal computer with Python implementation. Our implementation has been shared on github: usnistgov/BeamBending.

4. Conclusions

For very thin flexures, we have determined that it is feasible to calculate their deformation using semi-analytic methods. This was shown using approximate analytic solutions and arbitrary precision floating-point to exhibit limits of the double precision exponent. To integrate with existing tools in order to accommodate thin flexures double precision values can be exchanged for higher precision values according to Table 2. On the compiled language level this should not be difficult using ODE libraries which operate on generic types.

References

[1] 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.

[2] 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.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

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

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

[15] IEEE Computer Society. IEEE STANDARD FOR FLOATING-POINT ARITHMETIC. IEEE Std 754-2019, July 2019. DOI: 10.1109/IEEESTD.2019.8766229.

[16] 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.

[17] 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.

[18] 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.

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