# Deflection Computation in Hexrod

Frank Stetzer

Last update: April 16, 2020

## 1 Backgroud

I have been playing around with deflection analysis of split cane rods for a couple of years. My goal was to include the method in Hexrod, so the its users have another option to evaluate rod tapers and design new tapers based on previous tapers, similar to the way new tapers can be created by manipulating stress curves. But I have no engineering background, and reading the textbooks and literature usually resulted in more questions than answers. The consensus (by no means unanimous) is that a cane rod can be modeled as a tapered cantilever beam, but beyond that what you should assume is open to debate.

In November 2019 Henk Verhaar shared on the Classic Fly Rod Forum a link to an undergraduate honors thesis by Bennett Scully, containing an interative algorithm to solve the deflection formulas for a cane rod under a static load, i.e. a weight suspended from the tip. I jumped on this right away, rewrote the algorithm in Perl (my language for quick and dirty calculations), and began to experiment with it. I reported some of my results back to the Forum. This paper describes how I extended and implemented Scully's algorithm in Hexrod. I do not include any examples here; you can run your own in Hexrod.

## 2 Static Deflection of a Cantilever Beam

Static deflection of a cantilever beam is a major topic is mechanical engineering textbooks. In the simplest case, the beam is horizontal, has no deflection due to its own weight, and is deflected only by weight at a single point, such as the end. If this weight is small, the deflection of the free end of the beam is only a small amount, perhaps one percent of its length. Under the assumption that the beam is of uniform cross section throughout its length, the curvature of the beam is at its maximum at the fixed end and zero at its free end. Under all these restrictions, and given the stiffness (Modulus of Elasticity) of the material, the deflection and curvature at any point along the beam is readily calculated. This is called *Small Deflection Theory*.

When the beam is like a fly rod, however, the solution becomes more complicated. The deflection of the rod is no longer small, and it is tapered with a specific taper from butt to tip. Under these circumstances, called *Large Deflection Theory*, there is no longer a simple solution. Formula for beams subject to large deflections are quite complex, involving systems of nonlinear differential equations, and the taper adds to the complexity: we can no longer be certain the maximum curvature to be at the fixed end.

## 3 Scully's Algorithm for Static Deflection

The derivation of Scully's algorithm is described on pages 11-13 of his thesis.

Scully actually derives two algorithms, one based on *large deflection theory* for a cantilever beam, and the other based on *small deflection theory*. On the face of it, you would expect the first to be more appropriate for a beam like a fly rod, since small deflection theory assumes that the deflection is only a tiny percent of the beam length. After comparing computer solutions to actual rod deflection however, Scully determines that his small deflection algorithm gives results closer to the actual deflected rod than the alternative. (More on this in section 10.2) This algorithm depends on treating the rod as a set of small intervals, computing the small theory deflection of each, and accumulating the results.

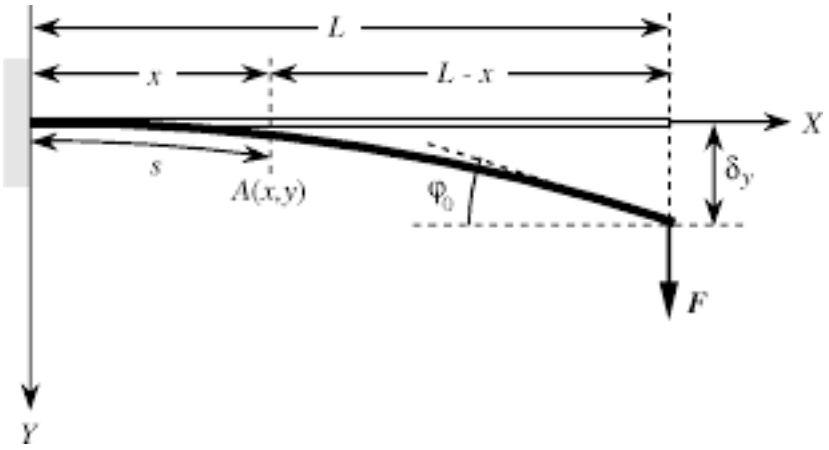


Figure 1: Small deflection diagram

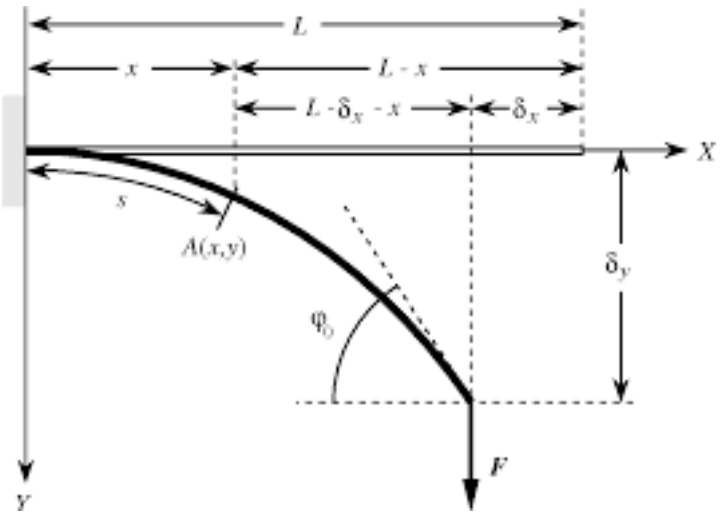


Figure 2: Large deflection diagram

# 4 Scully's Algorithm for Static Deflection

## 4.1 Notation

I am assuming customary U.S. measurements here: inches, pounds, etc.

A minor bookkeeping headache in implementing the deflection calculations in Hexrod was the need to move smoothly between normal rodmaker accounting and Scully's model. In rod-

maker laccounting, taper dimensions, ferrule locations etc. are located by distance from the rod tip. If we define a vector of dimensions, e.g. **DIM[I]**, the index **I** begins at 0 inches (the tip) and ends at the top of the grip, what Hexrod calls the action length **AL** inches, for a total of **AL+1** measurements. Scully's notation does not follow this convention, so care must be taken in going between the two.

1. His algorithm defines the action length as 1 and divides it into **N-1** segments of length **DS=1/(N-1)**. He uses the value **N=100** in his calculations, which typically makes the segment less than an inch in length; these are numbered from 1 (the segment just above the grip), to **N-1** at the tip. (See section 10.1)

2. The **N** end points of each segment he calls *nodes* (no relation to our nodes).

3. The X and Y deflection, the angle and curvature will be computed at each node; call these **XDEF[I]**, **YDEF[I]**, **THETA[I]**, and **CURV[I]**. **(XDEF[I],YDEF[I])** is the coordinate of deflected rod at node **I**; the coordinate of the node at the grip is **(0,0)** and in a perfectly straight undeflected rod, the coordinate of the tip is **(1,0)**. As the rod deflects **XDEF** becomes smaller and **YDEF** becomes negative. The angle at that node **THETA[I]** is in radians and **CURV[I]** measures the sharpness of the deflection curve at that point. For units of curvature refer to an engineering text; the tighter the curve the higher the curvature.

4. In addition we will need the Modulus of Elasticity of bamboo, which Scully sets to **MoE=5,988,000** psi, the Moment of Inertia (stiffness) of a solid hexagonal beam segment, **MoI[I]=0.0601407*DIM[I]**4** at node **I** , and the force (weight) **P** in pounds suspended from the tip. **DIM[I]** is the rod dimension at node **I** in inches.

## 4.2 Pseudocode

If we consider a rod being bent by a weight at the tip, and the weight is suddenly reduced, we can imagine the rod oscillating back and forth between too much deflection and too little until it comes to a new equilibrium. This is a crude analogy to Scully's algorithm. It has an equation for **CURV** in term of **XDEF**, and two for **XDEF** and **YDEF** in terms of **CURV**, which alternate until the oscillation stops, as determined by the vertical tip deflection **YDEF[N]**. Here is the algorithm:

```
** Number of nodes used by Scully in his examples
N = 100

** set Modulus of elasticity for Tonkin cane
MoE = 5988000

** compute Moment of Inertia for hexagonal beam
for I = 1 to N-1
  MoI[I] = 0.0601407 * DIM[I]**4
end

** Angle at the very first node (at the grip) is fixed at zero
THETA[1] = 0


** The weight in pounds suspended from the rod tip
P = 0.05

** Length of a segment
DS = 1 / (N-1)
```

```
** TOL is the desired tolerance for ending the iterations;
** here 0.1 vertical inches at the tip
TOL = 0.1
CHANGE_TOL = 2 * TOL

** Initial tip deflection to start iteration;
** any number greater than CHANGE_TOL
OLD_TIP = 1

** Initial cantilever beam shape (no deflection)
for I = 1 to N
  XDEF[I] = I * DS
  YDEF[I] = 0
end

** This is the loop which iterates until the beam reaches equilibrium

while CHANGE_TIP > TOL

  for I = 1 to N-1
    CURV[I] = P * (XDEF[N] - XDEF[I]) / (MoE * MoI[I])
  end

  for I = 2 to N
    THETA[I] = CURV[I] * DS + THETA[I-1]
    XDEF[I] = XDEF[I-1] + DS * cos(THETA[I-1])
    YDEF[I] = YDEF[I-1] - DS * sin(THETA[I-1])
  end

  NEW_TIP = YDEF[N]
  CHANGE_TIP = 100 * abs((NEW_TIP - OLD_TIP) / OLD_TIP)
  OLD_TIP = NEW_TIP

** end of the while loop
end
```

### 4.3 Implementation & Testing

I coded the algorithm in Perl, with these changes:

1. Reasoning that you would not want the number of nodes **N** to be a constant regardless of rod (action) length, I made it a function of length: **N=(AL+1) * INCREMENT_DIVISOR**. I ended up using **INCREMENT_DIVISOR=5** in Hexrod. This makes the segments 0.2 inches in length, which is probably much smaller than necessary. The size of the segment does make a slight difference in the resulting deflection and curvature. More on this below in section 10.1.

2. Another change was to define **DS=(AL+1)/N** to make inches the units of deflection and curvature.

3. I set the default value **MoE=5,300,000** psi which is more reasonable in my judgment. This can be changed within Hexrod if desired.

I tested the program with a variety of different rod tapers and weights **P**. The tapers needed to be interpolated from one-inch stations to 0.2 inch stations, and the vector reversed to go from **I=1** at the rod grip to **I=N** at the tip. What I found was, for many tapers, the algorithm would not converge if the weight **P** was too large. If you look at the graphs of deflection curves Scully includes in his thesis, all the deflections quite modest. I suspect he had difficulty getting the algorithm to converge as well. Henk Verhaar reported similar problems.

The convergence problem manifested itself by the deflection oscillating between very high and very low values. If you keep count of the iterations (executions of the **while** block), even numbered iterations have too little deflection and odd numbered too much, and they do not want to converge to the middle.

# 5 Improved Algorithm

## 5.1 Solving the convergence problem

I wondered if this convergence problem might be overcome if the deflection (actually the curvature) was given a little nudge away from the extreme values. This led me to modifying the algorithm as follows (after much experimentation):

1. Define a constant **ACCEL** for the degree of convergence acceleration (nudge). Start with **ACCEL=0.3**.
2. Define a vector **CURV_OLD** of length **N** with initial values 0
3. Add an iteration counter **ITER**, starting at 1 for the first time through the **while** loop
4. After the first **for** loop, which calculated a new **CURV** vector
   (a) If **ITER** is odd, caclulate a **CURV_NEW** vector which is the just computed **CURV** reduced by a factor of **ACCEL**. So initially, this is a 30% reduction.
   (b) if **ITER** is even, **CURV_NEW** is increased by a factor of **ACCEL**
5. **CURV_NEW** replaces vector **CURV** and **CURV_OLD**
6. The second **for** loop which calculates **XDEF**, **YDEF**, and **THETA** is executed, and the **while** loop repeats with a test for convergence
7. If the iteration count reaches 50 without convergence, **ACCEL** is increased to 0.4 and the entire procedure repeated.
8. If iteration count again reaches 50, **ACCEL** is set to 0.5 and the procedure is repeated
9. If convergence is still not achieved, the procedure is abandoned without deflection being computed

The values of **ACCEL** and the maximum iterations of 50 were determined by experimenting with different tapers and different weights **P**. In general, the heavier the weight, the higher the value if **ACCEL** and the more iterations necessary for convergence.

## 5.2 Pseudocode

This algorithm allows the successful computation of deflection for much larger forces **P** than Scully's original algorithm. Here is the pseudocode. I admit I got lazy and used a **goto**. So sue me :-).

```
** set Modulus of elasticity for cane
MoE = 5300000

** compute Moment of Inertia for hexagonal beam
for I = 1 to N-1
  MoI[I] = 0.0601407 * DIM[I]**4
end

** Angle of the very first segment (at the grip) is fixed at zero
THETA[1] = 0

** Number of nodes
INCREMENT_DIVISOR = 5
N = (AL + 1) * INCREMENT_DIVISOR

** The weight in pounds suspended from the rod tip
P = 0.05

** Length of a segment
DS = 1 / (N-1)

** TOL is the desired tolerace for ending the iterations;
** here 0.1 vertical inches at the tip
TOL = 0.1
CHANGE_TOL = 2 * TOL

** ACCEL is the proportion reduction or increase in curvature to aid
** or speed convergence
ACCEL = 0.30

** this is the place we return to with a larger ACCEL
** if convergence not achieved
label NEW_ACCEL

** Initial tip deflection to start iteration;
** any number greater than CHANGE_TOL
OLD_TIP = 1

** Initial cantilever beam shape (no deflection)
for I = 1 to N
  XDEF[I] = I * DS
  YDEF[I] = 0
  CURV_OLD[I] = 0
end

** This is the loop which iterates until the beam reaches equilibrium
ITER = 0
```

```
while CHANGE_TIP > TOL

  ITER = ITER + 1
  for I = 1 to N-1
    CURV[I] = P * (XDEF[N] - XDEF[I]) / (MoE * MoI[I])
  end

** these loops adjust the curvature values back toward the middle,
** depending on whether the iteration is odd or even
  if (odd(ITER)) then
    for I = 1 to N-1
      CURV[I] = CURV[I] - ACCEL * (CURV[I] - CURV_OLD[I])
      CURV_OLD[I] = CURV[I]
    end
  else if (even(ITER)) then
    for I = 1 to N-1
      CURV[I] = CURV[I] + ACCEL * (CURV[I] - CURV_OLD[I])
      CURV_OLD[I] = CURV[I]
    end
  end if

  for I = 2 to N
    THETA[I] = CURV[I] * DS + THETA[I-1]
    XDEF[I] = XDEF[I-1] + DS * cos(THETA[I-1])
    YDEF[I] = YDEF[I-1] - DS * sin(THETA[I-1])
  end

** now test whether the iterative loop has converged after 50 iterations
** if not, increase ACCEL by 0.1 and restart the procedure
  if (ITER = 50) then
    if (ACCEL < 0.5) then
      ACCEL = ACCEL + 0.1
      goto NEW_ACCEL
    end if
    else then
** convergence has failed; do whatever is appropriate and
      die
    end if

  NEW_TIP = YDEF[N]
  CHANGE_TIP = 100 * abs((NEW_TIP - OLD_TIP) / OLD_TIP)
  OLD_TIP = NEW_TIP

 ** end of the while loop
end
```

## 5.3 Implementation and testing

The new algorithm solved the convergence problem for all reasonable tapers and deflection weights that I tried. For problems with small weights where the original algorithm worked, the convergence was quicker and the answers were the same.

There are a couple of small items which you will have to address if you want to implement this algorithm.

1. The algorithm computes the curvature for nodes **I=1..N-1**. For a cantilever beam, the curvature at **I=N** (the rod tip) is by definition 0 so you can fill in this value.

2. The algorithm computes the **XDEF** and **YDEF** and angle **THETA** vectors for **I=2..N**. I added the the values at **I=1** (the rod grip or start of the action) by linear projection from the values at points **I=2,3**.

3. The user is probably not interested in the deflection and curvature at all 400+ points used in the algorithm, so these must be interpolated back to the one-inch stations used by Hexrod in its graphs, tables, etc. Since this turned out to be a common operation, I wrote two subroutines. **VECTOR_INTERP** interpolates from a shorter (coarser) vector (such as dimensions at 1 inch stations) to a longer (finer) vector used in the deflection algorithm. **VECTOR_UNINTERP** reverses the process, going from the long vector back to the shorter. These subroutines (and one more) are used again in the next algorithm. Remember that the vectors used in the algorithm are indexed opposite of the usual rod maker customs, and begin at **N=1** not **0**.

## 6 Utility of Static Deflection

Static deflection only involves the rod taper and the weight at the tip. It is the theoretical equivalent of the traditional deflection board, where the actual rod is mounted horizontally by the grip, a known weight is suspended from the tip and the resulting deflection is traced onto paper. It is also similar to the *Common Cents* system used to describe rod actions. It does not include the obvious deflection inducing factor of the weight of the rod with its bamboo, ferrules, guides and varnish. It certainly does not include the forces of casting a line.

It seems to me that the primary utility of static deflection would be in characterizing a rod, by developing some rules which translate the deflection numbers, curvature and angles into things like rod speed and action. Similarly, it might prove useful in comparing tapers to each other. Another use might be in guide location.

The algorithm is easily modified to accommodate different rod geometries and hollowed rods simply by changing the formula for **MoI[I]**. These are implemented in Hexrod.

## 7 Moving Beyond Static Deflection

```
for I = 1 to N-1
  CURV[I] = P * (XDEF[N] - XDEF[I]) / (MoE * MoI[I])
end
```

If we look at the above snippet from the static deflection algorithm (and we've skimmed an engineering textbook) we might recognize the equation for curvature at node **I** as the *moment of force* divided by the flexural rigidity. The moment of force term includes only the weight at the tip **P** and the horizontal distance of station **I** from the deflected tip **N**. The traditional formula for rod stress, due to Garisson, includes several more weights: the weight of the line beyond the tip is similar to **P** in that it is "hung from the tip," the weights of the bamboo, varnish and guides, and line in the guides are all distributed along the rod in some way, and the weight of the ferrules is incorporated at particular stations. In addition, there is the "impact factor," which Garrison

included as the weight multiplier due to the motion of casting and estimated to be 4.0; it is equivalent to the more modern concept of *g-force*.

It is a small modification to the algorithm to include these weights and produce what I call in Hexrod the *casting deflection*. I hesitated calling it dynamic deflection since it only includes one dynamic component, the g-force, and ignores many others: the complex input of forces by the caster throughout the cast, including things like different casting strokes and line haul, the changing angle of the line to the rod tip, air resistance, and so forth.

### 7.1 Implementation

Define the vector **`WEIGHT[I]`** to contain the sum of all the weights on the segment from node **`I`** to **`I+1`**. **`WEIGHT[N]`** contains the weight of the line beyond the rod tip only. Other segments contain the weights of the bamboo, varnish and guides, line in the guides, and any ferrule. Let **`G`** be the desired g-force, which can be any reasonable value from 1 (just the weight of the components with no casting force) on up, with 4 being a reasonable default. The curvature is now computed as

```
for I = 1 to N-1
  MOMENT = 0
  for J = I+1 to N
    MOMENT = MOMENT + G * WEIGHT[J]  * (XDEF[J] - XDEF[I])
  end
  CURV[I] = MOMENT / (MoE * MoI[I])
end
```

The rest of the algorithm stays the same.

A couple small points on the calculation of the **`WEIGHT`** vector. The units are in pounds, not ounces as in usual stress calculations, to cancel units with the denominator value of **`MoE`** in psi. And the weights cannot be just *interpolated* from inches to the algorithm units, but *spread* so the total remains the same. I wrote another subroutine **`SPREAD_VECTOR`** to aid in this.

## 8 Utility of Casting Deflection

IMHO casting deflection provides a much more useful deflection solution than static deflection. These are some of its benefits:

1. The curvature vector can be transformed into stress values by multiplying each element by **`MoE`** in ounces and 0.5 times the rod dimension **`DIM[I]`**. Unlike Garrison's stress numbers, these values account for the deflection of the rod. They tend to show lower stress overall (since the deflected rod is shorter) and stress shifted from the tip area toward the butt, where the angle **`THETA`** is lower. By increasing the g-force impact factor, you can approach the sort of stress curve that Milward found by working backward from high speed photography (R.E.Milward, 2010, pp 124-8)

2. Just as with stress calculations, it is possible to start with a deflection (the vectors **`XDEF`** and **`CURV`**) and derive the taper. This opens up the possibility of modifying some characteristic of the rod (length, line weight etc.) and finding a new taper with the same deflection. The implementation of this is in the next section.

3. It also opens the possibility of modifying the deflection and deriving a new taper, or hollowing and deriving a taper. These are not yet implemented in Hexrod but are anticipated.

4. The basic approach can be expanded to include more sophisticated elements, for example the stiffness of ferrules compared to cane and the angle of the line to the rod (see Visner (2007)). These seem to bring the deflected stress curve close to what Milward derived using high speed photography.

Casting deflection is may be less practical in comparing tapers than static deflection, just because of the number of weight components which must be comparable. But we do not yet have the body of collective experience with deflection that we have with Garrison stress curves. We do not know its long term utility.

# 9 Derive a New Taper

Deriving a taper from deflection involves reversing the formulas of the algorithm. Again it is an iterative process, this time recalculating the weight vector. As a taper is calculated, the weight of the bamboo will change and possibly the size of the ferrules. This affects the weights along the rod so that the taper must be calculated again. This iteration continues until the taper ceases to change. It seems to take only 2-4 passes.

## 9.1 Pseudocode

This pseudocode gives a rough idea of how Hexrod derives a new taper. It leaves out the steps of (re)calculating the weight vector. Bear in mind that the **DIM** vector in the algorithm has **N** nodes, interpolated from the rod makers one-inch stations.

```
** start with a dummy DIM_NEW vector, say all zeros
** iterate until the dimensions change by less than 0.0002 inches at any station
while (max_difference(DIM, DIM_NEW) >= 0.0002)

** these are vectors, equated element by element
  DIM = DIM_NEW

** calculate new dimension vector
  for I = 1 to N-1
    MOMENT = 0
    for J = I+1 to N
      MOMENT = MOMENT + G * WEIGHT[J] * (XDEF[J] - XDEF[I])
   end
   FR = 0.0601407 * CURV[I] * MoE
   DIM_NEW[I] = (MOMENT / FR)**(1/4)
  end

** calculate new weight vector from new dimension vector
  WEIGHT = weight_calc(DIM_NEW)

 ** end of while loop
end while

DIM = DIM_NEW
```

### 9.2 Modifying the rod

Of course deriving the same taper you started with is not that exciting. Here is how Hexrod implements changing the original rod and deriving a new taper with the same deflection.

1. **Changing weights.** To change line size, length of line cast, or the number, location or type of ferrules, you need simply to change the weight vector to reflect the new rod. The derived taper will have the same deflection as the original. Be sure to check for changing ferrule sizes, which can change their contribution to the **WEIGHT** vector.

2. **Changing geometry.** To change from a Hexagonal rod, you must change the constant 0.0601407 to the appropriate value, for example 0.083333 for a Quad and 0.042720 for a Penta. And change the weight vector to incorporate the new weight of the bamboo in each segment.

3. **Changing length.** To change the action length of the rod is more complex. This is what seems reasonable to me:

   (a) Let **AL** be the original action length in inches, and **AL_NEW** the desired action length, which might be shorter or longer.

   (b) Calculate **N_NEW = (AL_NEW + 1) * INCREMENT_DIVIDER**

   (c) Scale up the values in the vector **XDEF** and scale down the values in **CURV**. Here is the pseudocode

   ```
   for I = 2 to N
     XDEF_NEW[I] = XDEF[I] * (AL_NEW / AL)
   end
   for I = 1 to N-1
     CURV_NEW[I] = CURV[I] * (AL / AL_NEW)
   end
   ```

   (d) The **XDEF_NEW** and **CURV_NEW** vectors now must be interpolated up or down from **N** points to **N_NEW**.

   (e) The original taper must be lengthened or shortened to provide the starting dimension vector.

   (f) The **WEIGHT** vector must be recalculated allowing for the bamboo change and spread to the **N_NEW-1** segments

At present, this is marked as being an experimental procedure. The results seem reasonable when compared to making the same modifications holding constant the Garrison (undeflected) stress curve. More testing and thinking is required.

## 10 Additional Topics

### 10.1 Number of Increments

In his examples, Scully without explanation divides the rod action length into 100 segments for deflection calculation. In Hexrod I chose to make the divisions smaller: 5 per inch (**INCREMENT_DIVISOR=5**.) I expect that this may be unnecessary, but computation time is minimal. However, experiments show that the resulting deflection values are very close but not identical for different size divisions. The pattern seems to be one of increasing deflection with increasing number of segments. Here is the static deflection values for a Payne 200 taper with a tip weight of 2.00 ounces, under **INCREMENT_DIVISOR** values of 5, 2, and 1. Values are given for the 0 (tip) and the 10, 30, 50, and 70 inch stations

| Table 1. Static deflection under alternative division sizes | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | INCREMENT_DIVISOR=5 | | | INCREMENT_DIVISOR=2 | | | INCREMENT_DIVISOR=1 | | |
| Station | XDEF | YDEF | CURV | XDEF | YDEF | CURV | XDEF | YDEF | CURV |
| 0 (Tip) | 84.307 | -32.989 | 0 | 84.414 | -32.807 | 0 | 84.598 | -32.499 | 0 |
| 10 | 80.114 | -24.041 | 0.030381 | 80.180 | -23.843 | 0.030548 | 80.291 | -23.509 | 0.03827 |
| 30 | 65.616 | -10.798 | 0.020461 | 65.555 | -10.639 | 0.020549 | 65.450 | -10.375 | 0.020693 |
| 50 | 47.167 | -3.735 | 0.009652 | 47.007 | -3.638 | 0.009655 | 46.737 | -3.478 | 0.006923 |
| 70 | 27.631 | -0.557 | 0.007051 | 27.392 | -0.520 | 0.00703 | 26.990 | 0.0461 | 0.06923 |

I ruled out convergence variablilty by decreasing the **`TOL`**. I may investigate this further at some point. Hard thinking may be necessary.

### 10.2 Small vs. Large Deflection Calculations

Scully provides alternative algorithms employing both small and large deflection theory. He determined that the small deflection algorithm gave a closer approximation to actual rod deflection. A Master's thesis by John Visner, *Analytical and Experimental Analysis of the Large Deflection of a Cantilever Beam Subjected to a Constant, Concentrated Force, With a Constant Angle, Applied at the Free End* (University of Akron, 2007) provides a computer program in Fortran for the solution of the problem. The program is similar to Scully's in that it subdivides the beam into many short segments and reaches its solution by an iterative procedure. Unfortunately the assumptions Visner builds into his logic, that the beam is of constant dimension and therefore that the curvature is decreasing monotonically from fixed to free end, make it unsuitable to solve cane rod deflection problems.

It is possible however the compare Visner's solutions to Scully's by considering a hexagonal rod without any taper. What I found was that Visner's results are very similar to Scully's small deflection algorithm, but deviated from his large deflection solution. What this implies about the latter I'm not sure, but it gave me more confidence that the algorithm, despite being derived from small deflection theory, is not totally inappropriate for cane rods.

## 11 References and Sources


1. This the thread on the Classic Flt Rod Forum which discussed Scully's thesis and algorithm: http://classicflyrodforum.com/forum/viewtopic.php?f=66&t=125933&hilit=deflection&start=0

2. Bennett R Scully (2018) *The Effect of Material Variability on the Deflection of Bamboo Fly Rods*. Honors Thesis, University of Maine https://digitalcommons.library.umaine.edu/cgi/viewcontent.cgi?article=1462&context=honors

3. R.E. Milward (2010) *Bamboo: Fact, Fiction and Flyrods - II*. Self-published.

4. William Hanneman *The Common Cents System*. https://www.common-cents.info

5. John C Visner (2007) *Analytical and Experimental Analysis of the Large Deflection of a Cantilever Beam Subjected to a Constant, Concentrated Force, With a Constant Angle, Applied at the Free End* Master of Science Thesis, University of Akron. https://etd.ohiolink.edu/!etd.send_file?accession=akron1196090494&disposition=inline