Comparing continuation methods

Shooting vs ManLab vs AUTO vs IPOPT

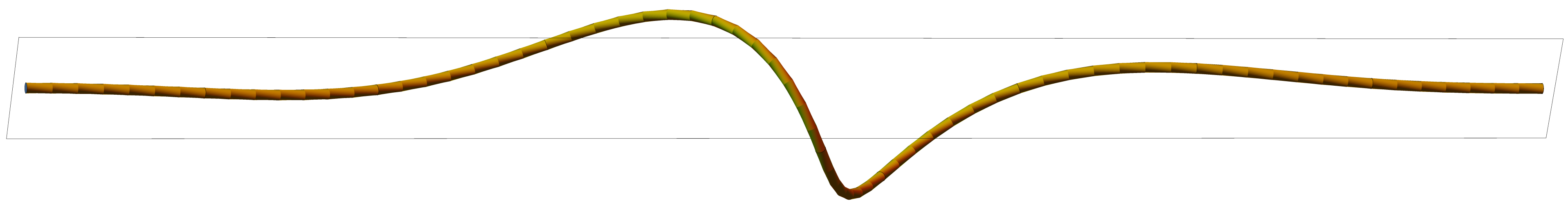
S. Neukirch
A. Lazarus **CNRS & Sorbonne Univ. - Paris, France**

O. Thomas **Arts et Métiers ParisTech, Lille, France**
E. Cottanceau **ANSYS France**

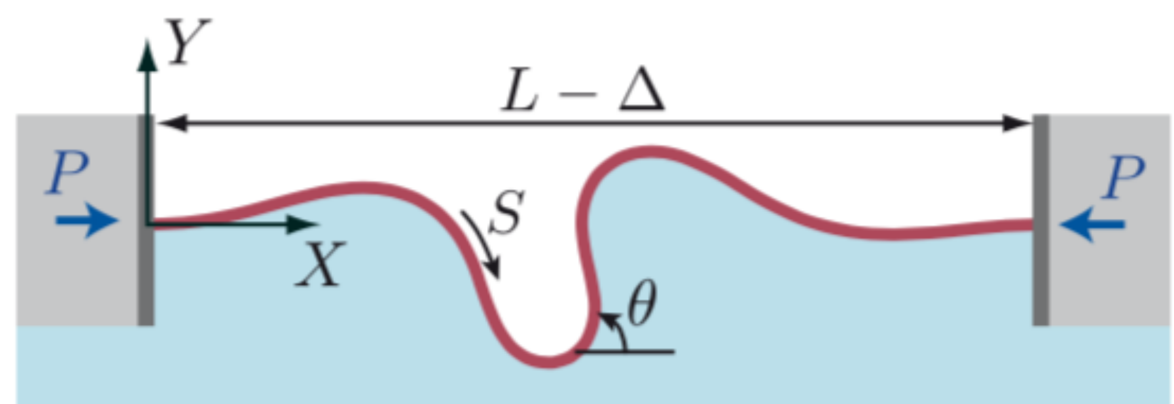
R. Charrondière
F. Bertails-Descoubes **INRIA, Univ. Grenoble Alpes, France**

Boundary value problems

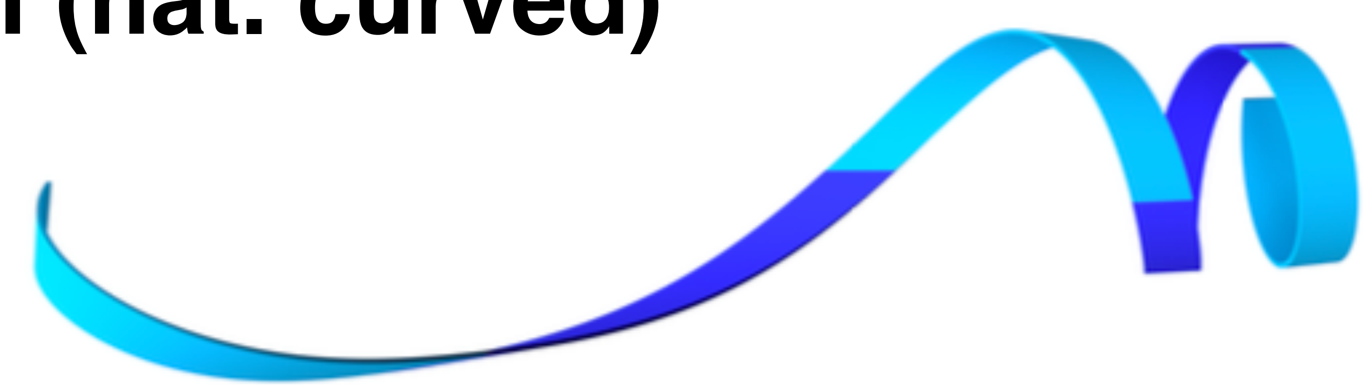
1 - Twisted rod (Kirchhoff model)



2 - Planar Elastica with fluid foundation



3 - Elastic ribbon (nat. curved)



Numerical Methods

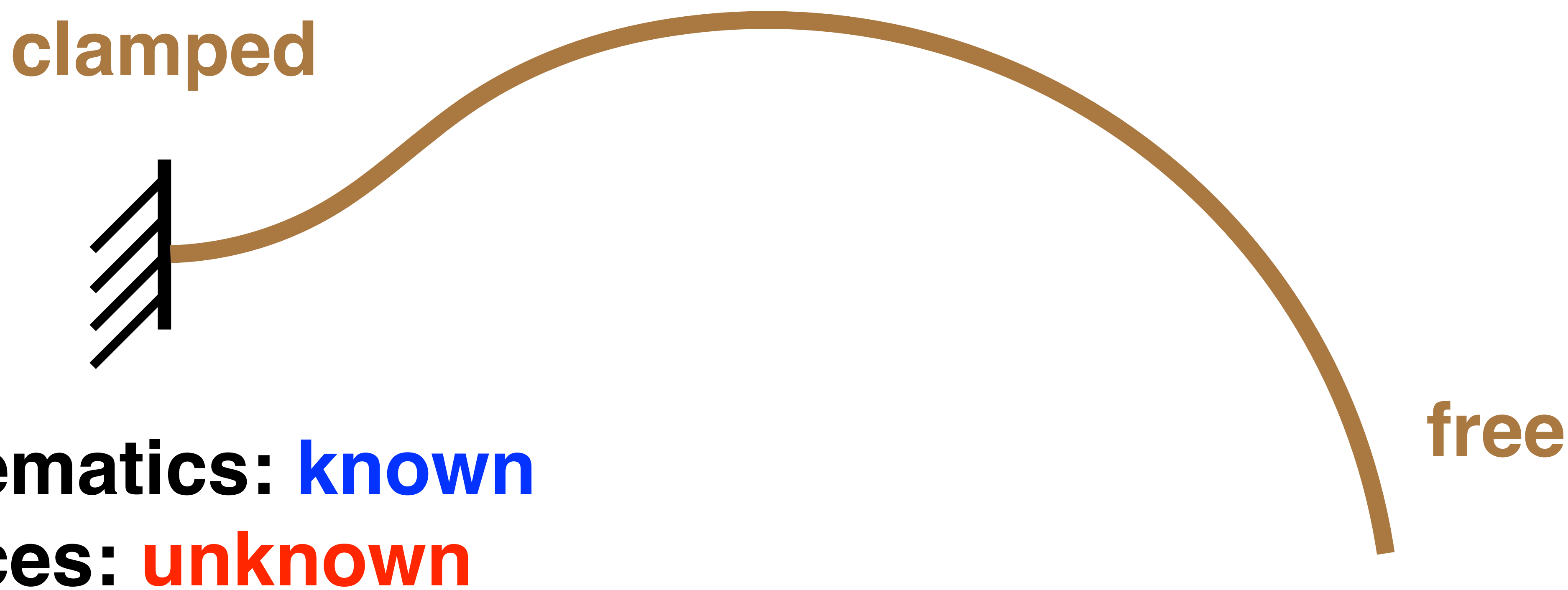
1 - Shooting (Mathematica) - quick to set up

2 - AUTO (Fortran or C) - fastest

3 - ManLab (Matlab) - interactive

4 - IPOPT (C++) - always converges

Shooting (1D structure)

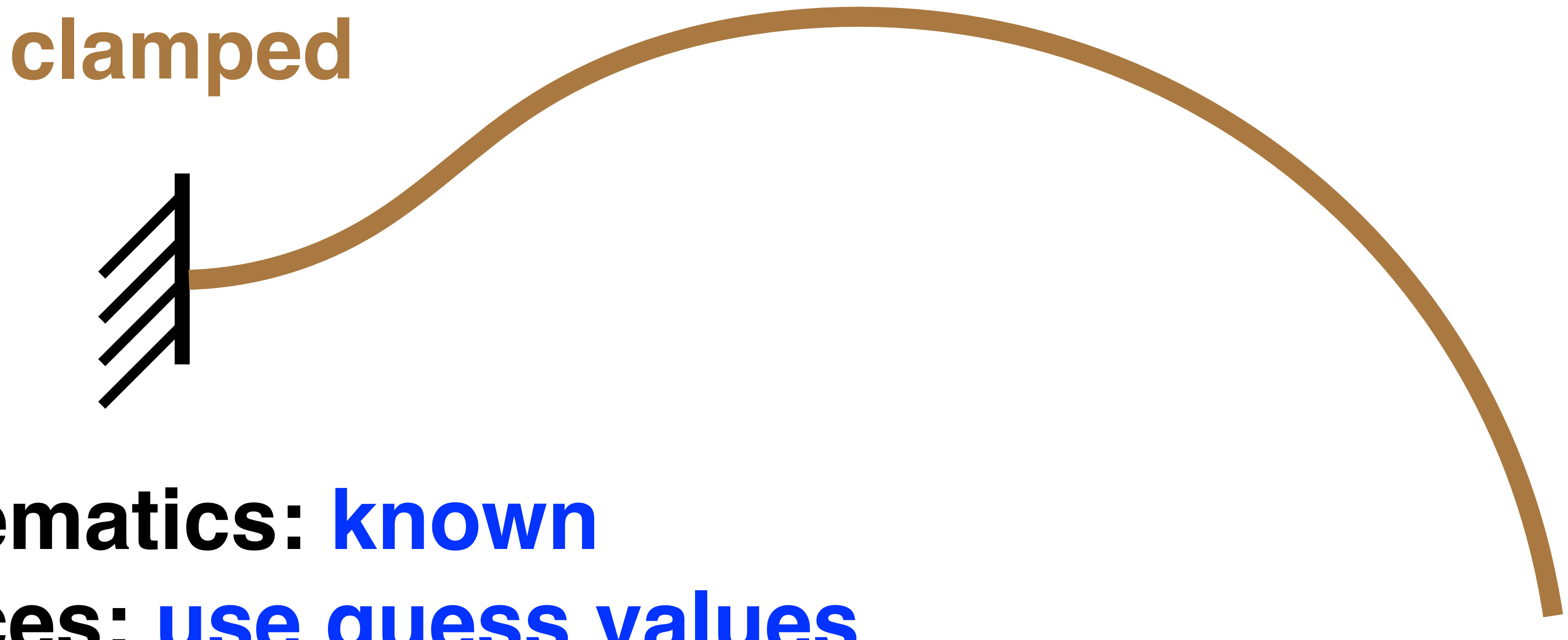


Kinematics: **known**
Forces: **unknown**

Kinematics: **unknown**
Forces: **known**

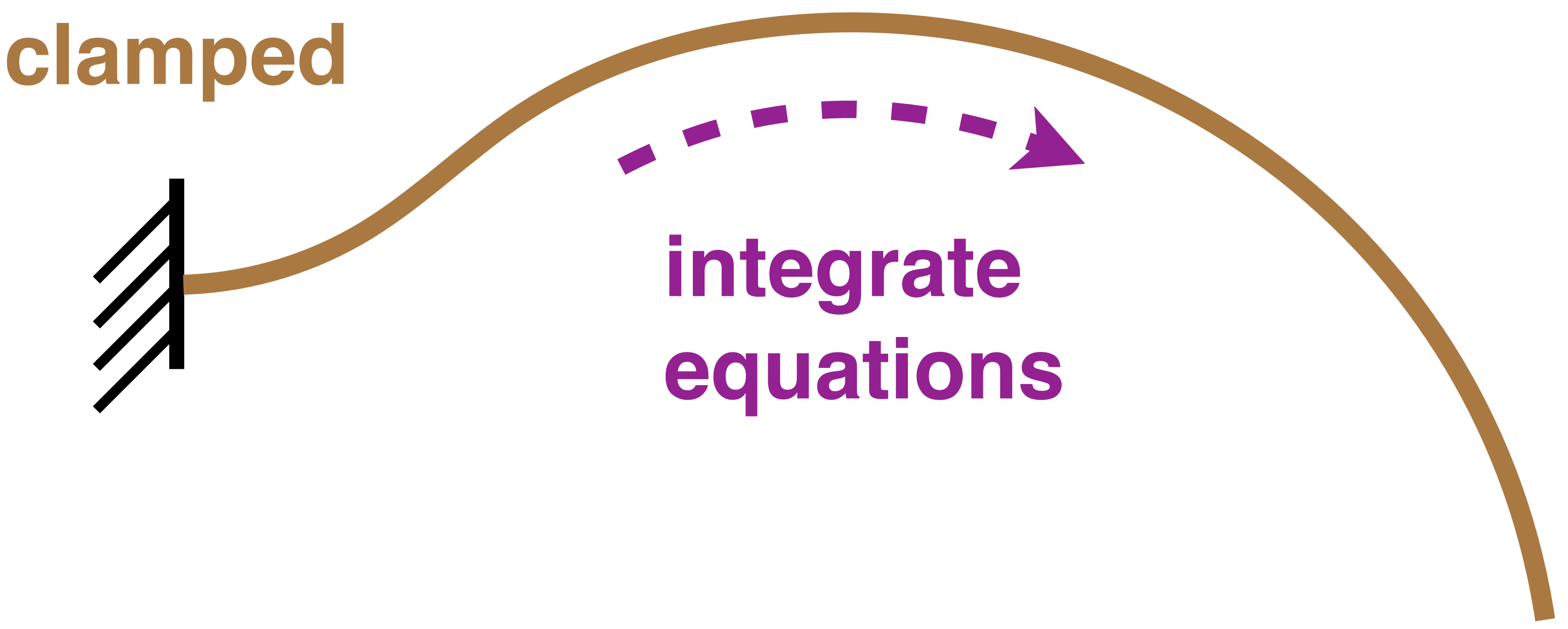
***idea*: transform the BVP into an IVP**

Shooting (1D structure)

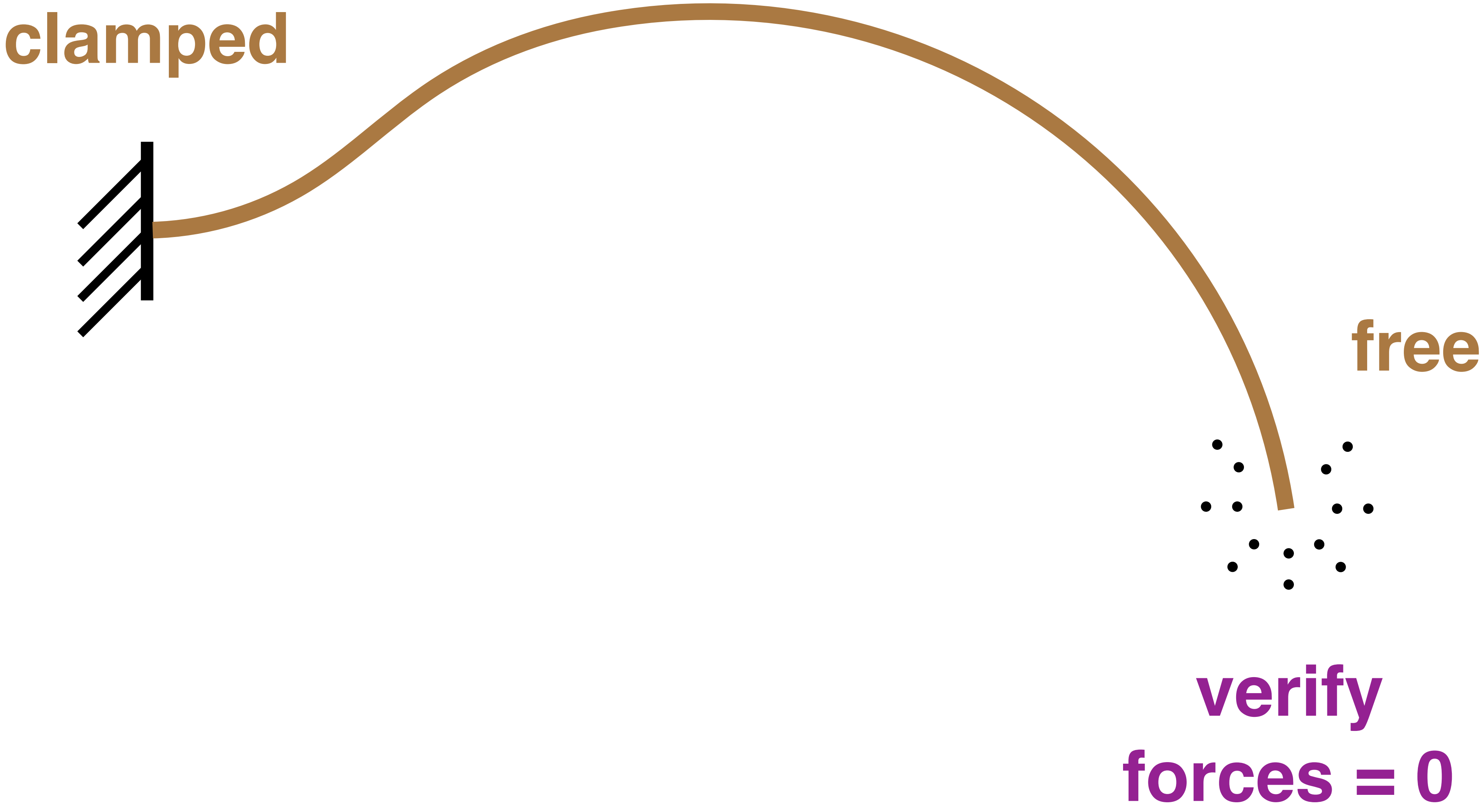


Kinematics: **known**
Forces: **use guess values**

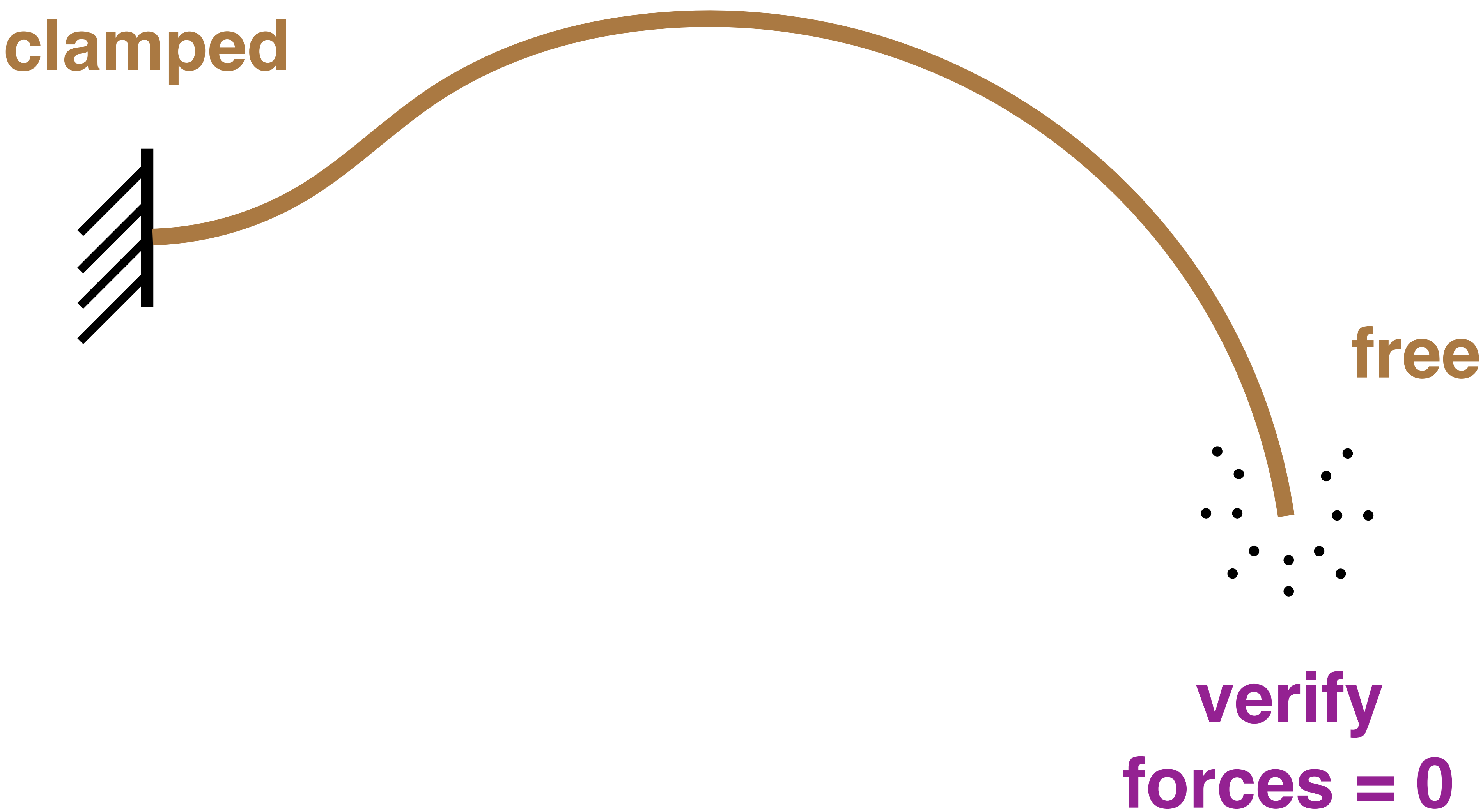
Shooting (1D structure)



Shooting (1D structure)



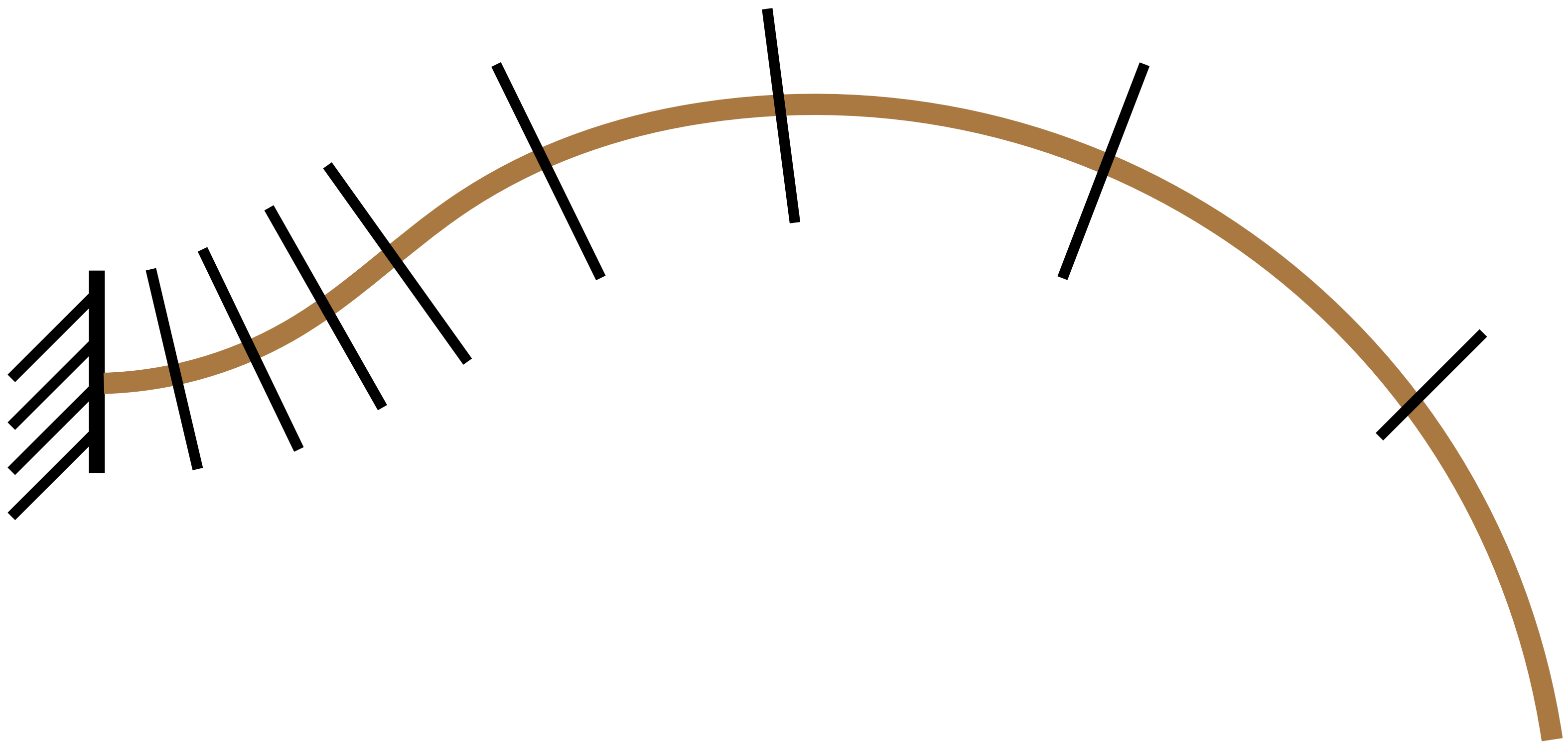
Shooting (1D structure)



If not: start again with new (updated) guessed values

collocation: AUTO

v1 : 1976



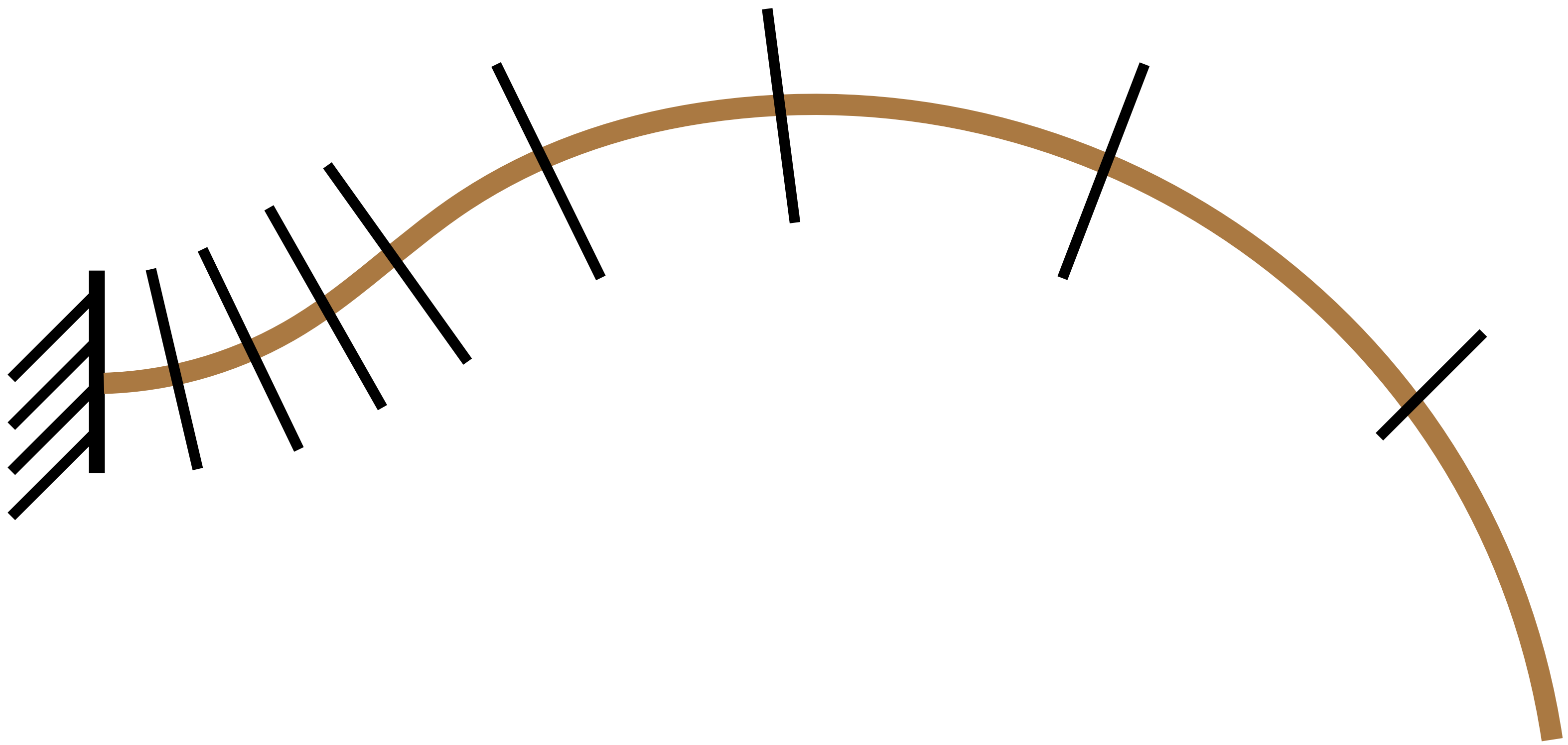
NTST = 10 or 40 or … segments (adaptative mesh)
Lagrange polynomials of degree NCOL = 3 to 7
collocation: equations in strong form, satisfied at gauss points

R(U)=0 : newton-chord method

$$R'(U)\,(U_{new} - U) = -R(U)$$

Linear system (block diag.) size = NDIM x NTST x NCOL

collocation: AUTO



Linear system (block diag.) size = NDIM x NTST x NCOL

R'(U) (Unew - U) = -R(U)

Condensation (gauss elimination) — ? AUTO secret ? —

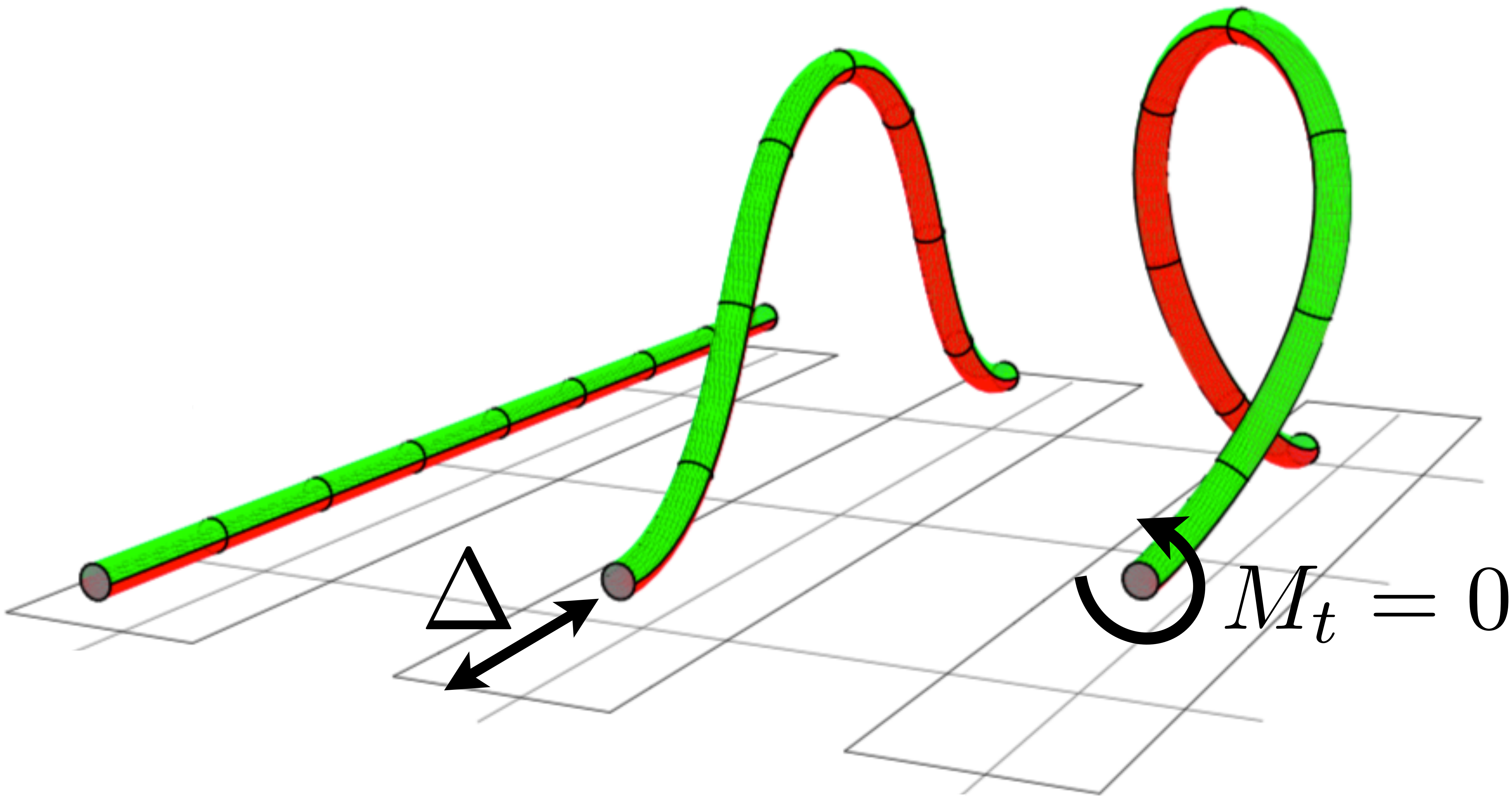
==> full matrix, but of size NDIM only

***advantage*: simply enter equilibrium equations (ODE system)**

```
!==========================================================================
   SUBROUTINE FUNC(NDIM,U,ICP,PAR,IJAC,F,DFDU,DFDP)
!==========================================================================

   IMPLICIT NONE
   INTEGER, INTENT(IN) :: NDIM, IJAC, ICP(*)
   DOUBLE PRECISION, INTENT(IN) :: U(NDIM), PAR(*)
   DOUBLE PRECISION, INTENT(OUT) :: F(NDIM)
   DOUBLE PRECISION, INTENT(INOUT) :: DFDU(NDIM,*), DFDP(NDIM,*)

   DOUBLE PRECISION x,y,th,m,nx,ny
   ! getpar()
   double precision m0,nx0,ny0,eta,y12,th12,delta,P ! a recopier
   integer npar,typeconti    ! a recopier
   ! fin getpar()
!--------------------------------------------------------------------------

! --- On lit les PAR
   CALL GETPAR(m0,nx0,ny0,eta,y12,th12,delta,P,typeconti, &
                    PAR,npar)
! FIN On lit les PAR

   x=U(1)
   y=U(2)
   th=U(3)
   m=U(4)
   nx=U(5)
   ny=U(6)

   F(1) = dcos(th)              ! x'=cos th
   F(2) = dsin(th)              ! y'=sin th
   F(3) = m                     ! th'=m
   F(4) = nx*dsin(th)-ny*dcos(th) ! m' = nx sin th - ny cos th
   F(5) = -eta**4*y*dsin(th)   ! nx' = -eta^4 y sin th
   F(6) =  eta**4*y*dcos(th)   ! ny' =  eta^4 y cos th
```

```fortran
!==========================================================================
      SUBROUTINE BCND(NDIM,PAR,ICP,NBC,U0,U1,FB,IJAC,DBC)
!==========================================================================

      IMPLICIT NONE
      INTEGER, INTENT(IN) :: NDIM, ICP(*), NBC, IJAC
      DOUBLE PRECISION, INTENT(IN) :: PAR(*), U0(NDIM), U1(NDIM)
      DOUBLE PRECISION, INTENT(OUT) :: FB(NBC)
      DOUBLE PRECISION, INTENT(INOUT) :: DBC(NBC,*)

      ! getpar()
      double precision m0,nx0,ny0,eta,y12,th12,delta,P ! a recopier
      integer npar,typeconti     ! a re copier
      ! fin getpar()
      double precision temp
!--------------------------------------------------------------------------


! --- On lit les PAR
      CALL GETPAR(m0,nx0,ny0,eta,y12,th12,delta,P,typeconti, &
                  PAR,npar)
! FIN On lit les PAR



! --- Initial conditions (at s=0)
      FB(01)=U0(1)-0.d0   ! x(0)=0
      FB(02)=U0(2)-0.d0   ! y(0)=0
      FB(03)=U0(3)-0.d0   ! th(0=0)
      FB(04)=U0(4)-m0
      FB(05)=U0(5)-nx0
      FB(06)=U0(6)-ny0
! FIN Initial conditions (at s=0)

! --- Final conditions (at s=1)
      FB(07)=U1(2)-0.d0              ! y(1)=0
      FB(08)=U1(3)-0.d0              ! th(1)=0
! FIN Final conditions (at s=1)
```

Example: 3D Kirchhoff rod



imposed: axial displacement Δ **and zero torque** $M_t = 0$

Kirchhoff equations for elastic rods

kinematics

$$x' = d_{3x}$$
$$y' = d_{3y}$$
$$z' = d_{3z}$$
$$d'_{3x} = u_2\, d_{1x} - u_1\, d_{2x}$$
$$d'_{3y} = u_2\, d_{1y} - u_1\, d_{2y}$$
$$d'_{3z} = u_2\, d_{1z} - u_1\, d_{2z}$$
$$d'_{1x} = u_3\, d_{2x} - u_2\, d_{3x}$$
$$d'_{1y} = u_3\, d_{2y} - u_2\, d_{3y}$$
$$d'_{1z} = u_3\, d_{2z} - u_2\, d_{3z}$$
$$d'_{2x} = u_1\, d_{3x} - u_3\, d_{1x}$$
$$d'_{2y} = u_1\, d_{3y} - u_3\, d_{1y}$$
$$d'_{2z} = u_1\, d_{3z} - u_3\, d_{1z}.$$

dynamics

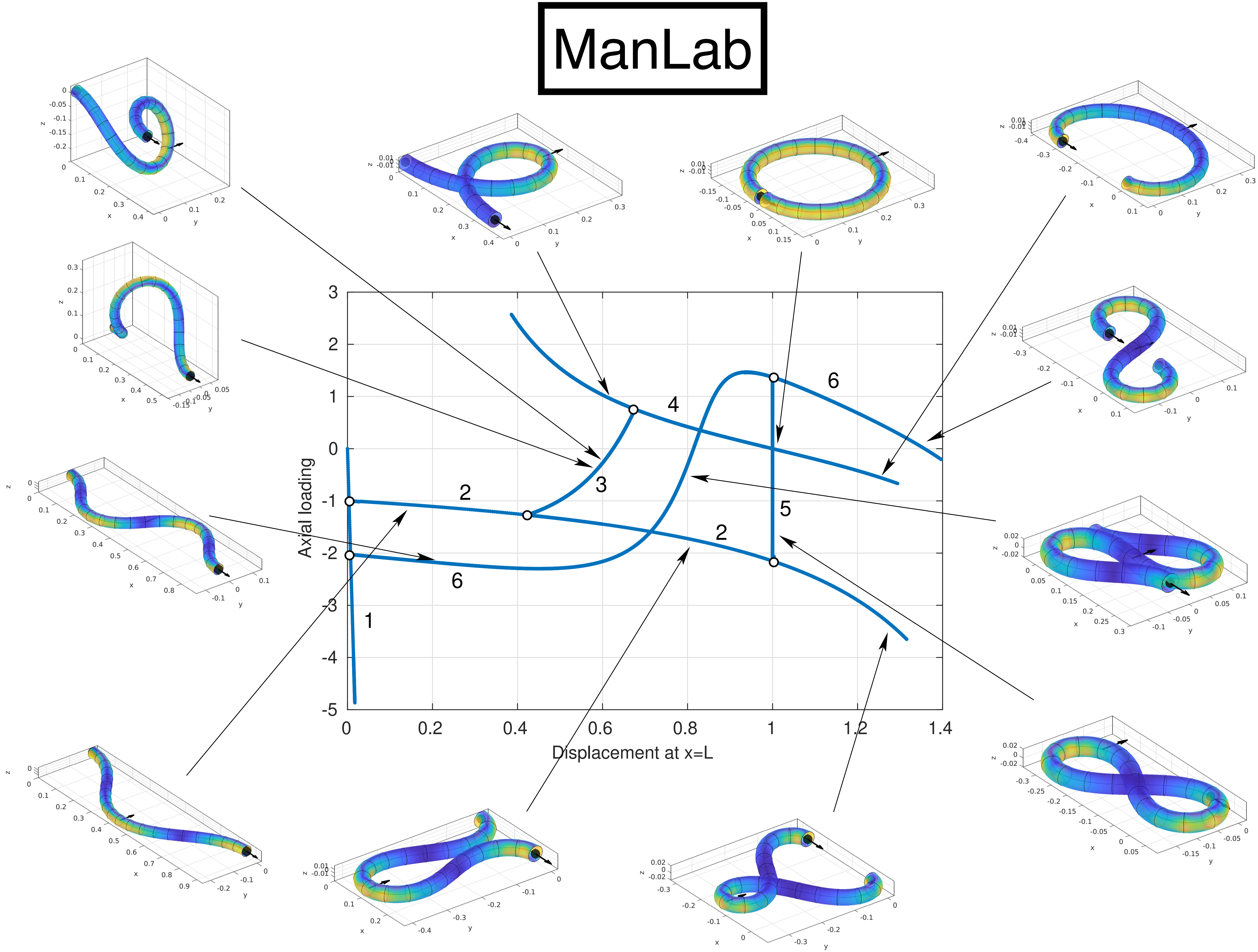
$$n'_1 = n_2\, u_3 - n_3 u_2 - f_1 + \rho A\,(\ddot{x}\, d_{1x} + \ddot{y}\, d_{1y} + \ddot{z}\, d_{1z})$$
$$n'_2 = n_3\, u_1 - n_1 u_3 - f_2 + \rho A\,(\ddot{x}\, d_{2x} + \ddot{y}\, d_{2y} + \ddot{z}\, d_{2z})$$
$$n'_3 = n_1\, u_2 - n_2 u_1 - f_3 + \rho A\,(\ddot{x}\, d_{3x} + \ddot{y}\, d_{3y} + \ddot{z}\, d_{3z})$$
$$m'_1 = m_2\, u_3 - m_3 u_2 + n_2$$
$$m'_2 = m_3\, u_1 - m_1 u_3 - n_1$$
$$m'_3 = m_1\, u_2 - m_2 u_1$$

$$m_1 = K_1\, u_1\,, \quad m_2 = K_2\, u_2\,, \quad m_3 = K_3\, u_3$$

constitutive relations

(blue line) (brown line) 18 ODE system

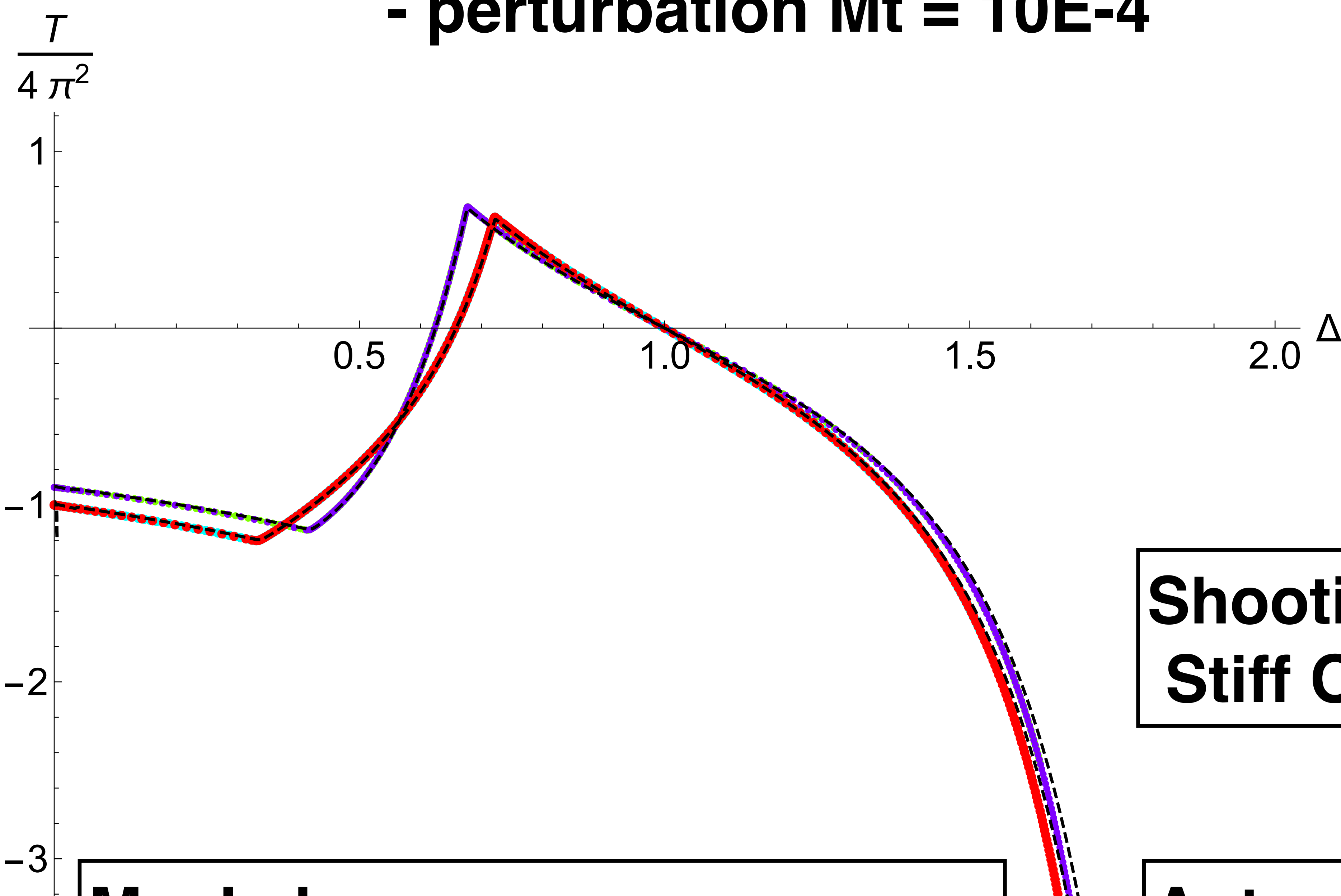
(red line) 3 algebraic eq.



ManLab
Axial loading
Displacement at x=L
0
0.2
0.4
0.6
0.8
1
1.2
1.4
3
2
1
0
-1
-2
-3
-4
-5
1
2
3
4
5
6
2
6

Comparison

Model: - anisotropic cross-section (K1=1, K2=0.9)
- perturbation Mt = 10E-4

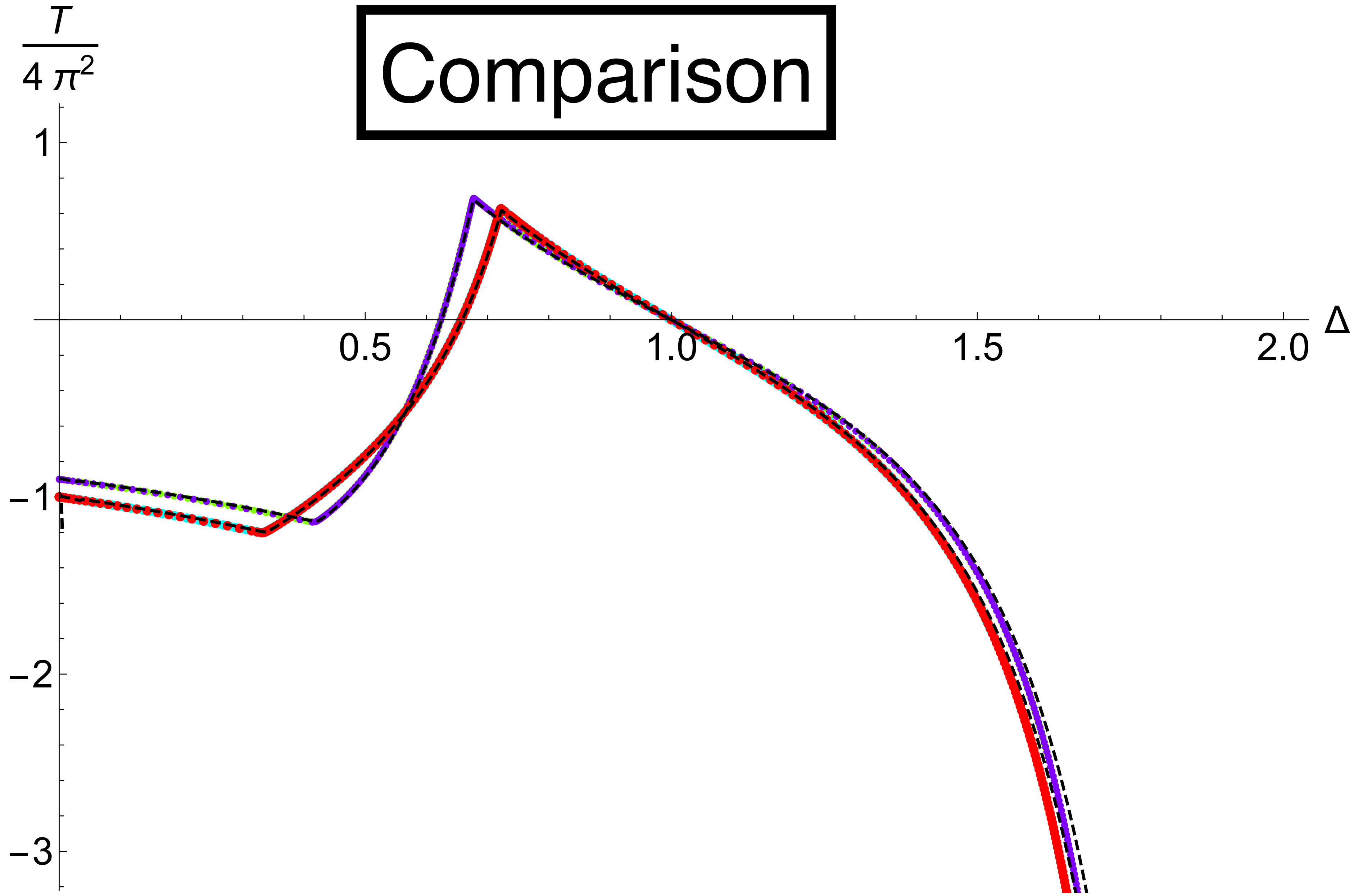


Shooting:
Stiff ODE solver

ManLab:
perturbation Q = 10E-18
finite elements, 20 segments

Auto:
NTST=12 segments
NCOL=3 degree poly.

Comparison

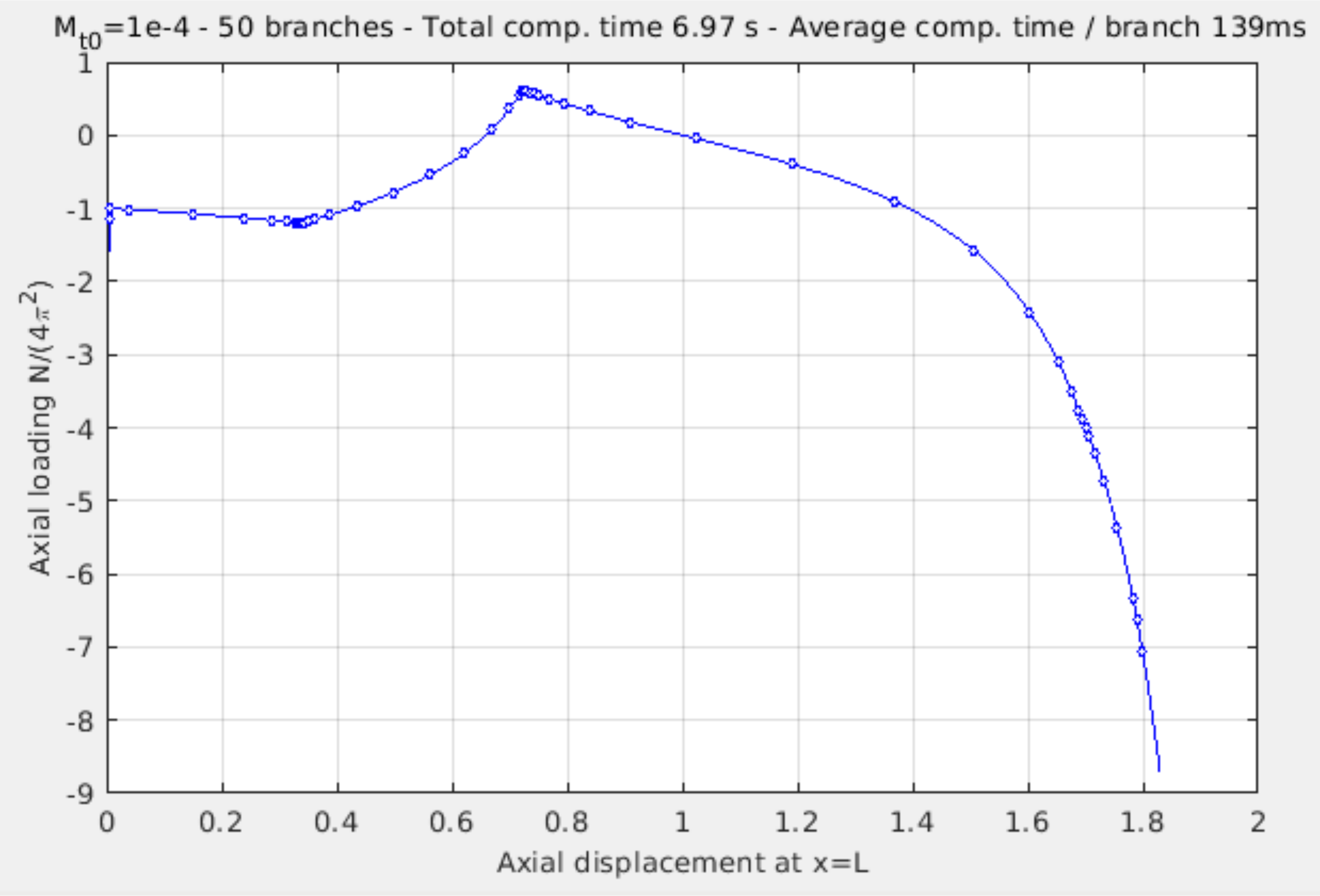


Shooting: 615 pts (142 sec) and 494 pts (95 sec)
ManLab: 41 branches (5.6 sec) and 50 branches (7 sec)
AUTO: 560 pts (0.9 sec) and 370 pts (0.65 sec)

rem: shooting does not go down enough

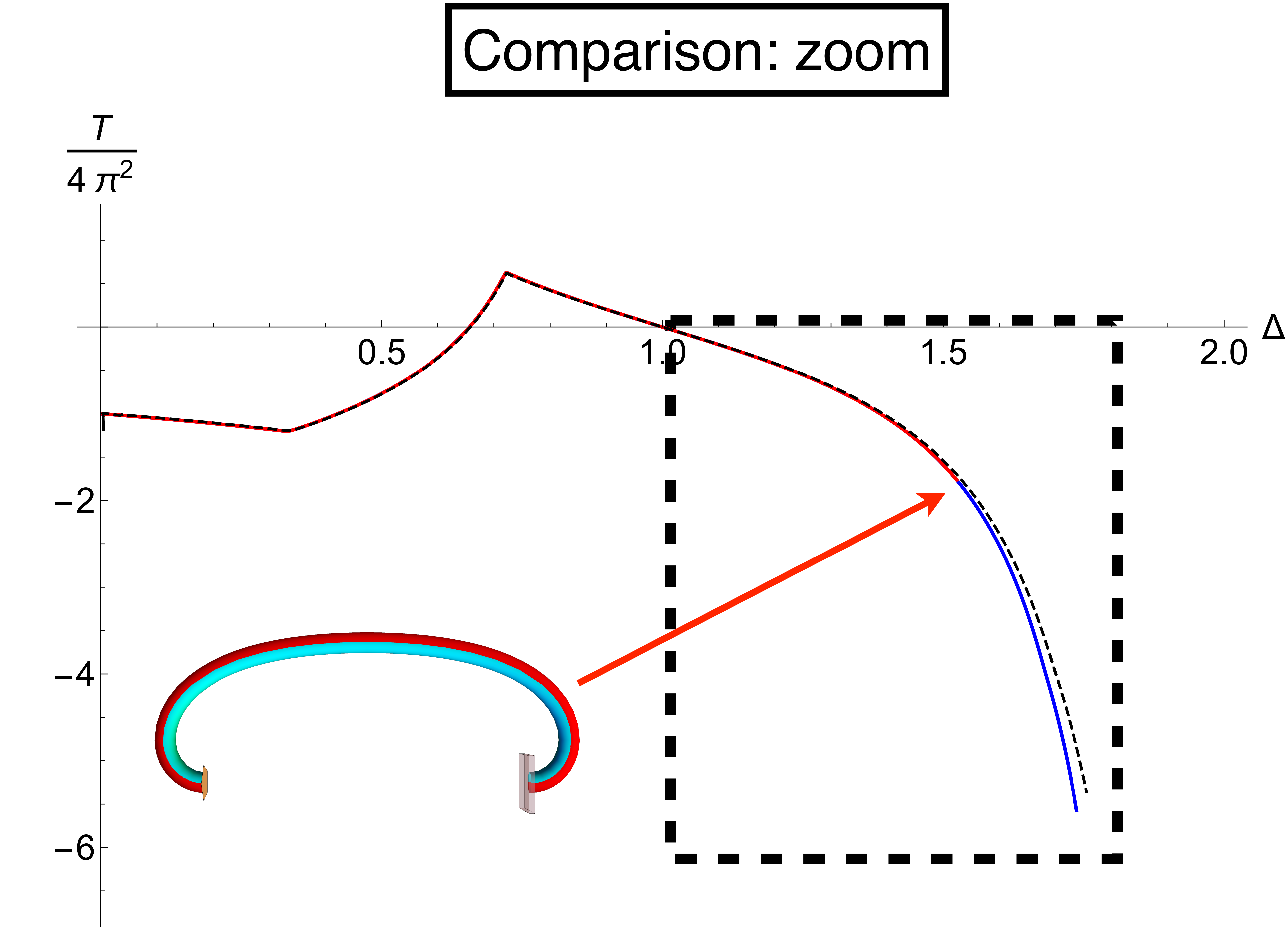
rem: ManLab starts from trivial state
AUTO and shooting from a 1st nonlinear solution

Comparison



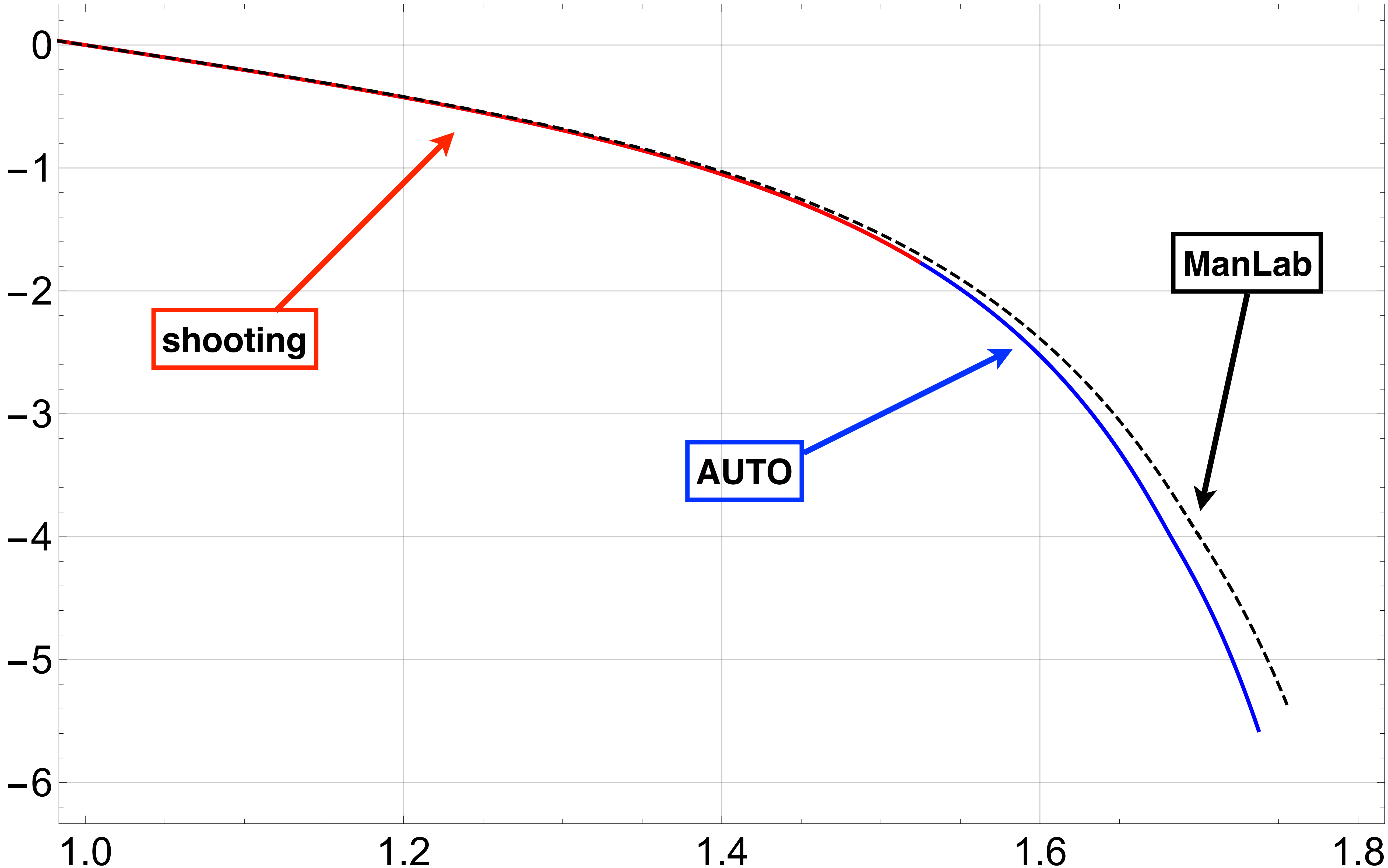
ManLab: ~ 140 ms / branch

rem :
Taylor 20
no corrector step



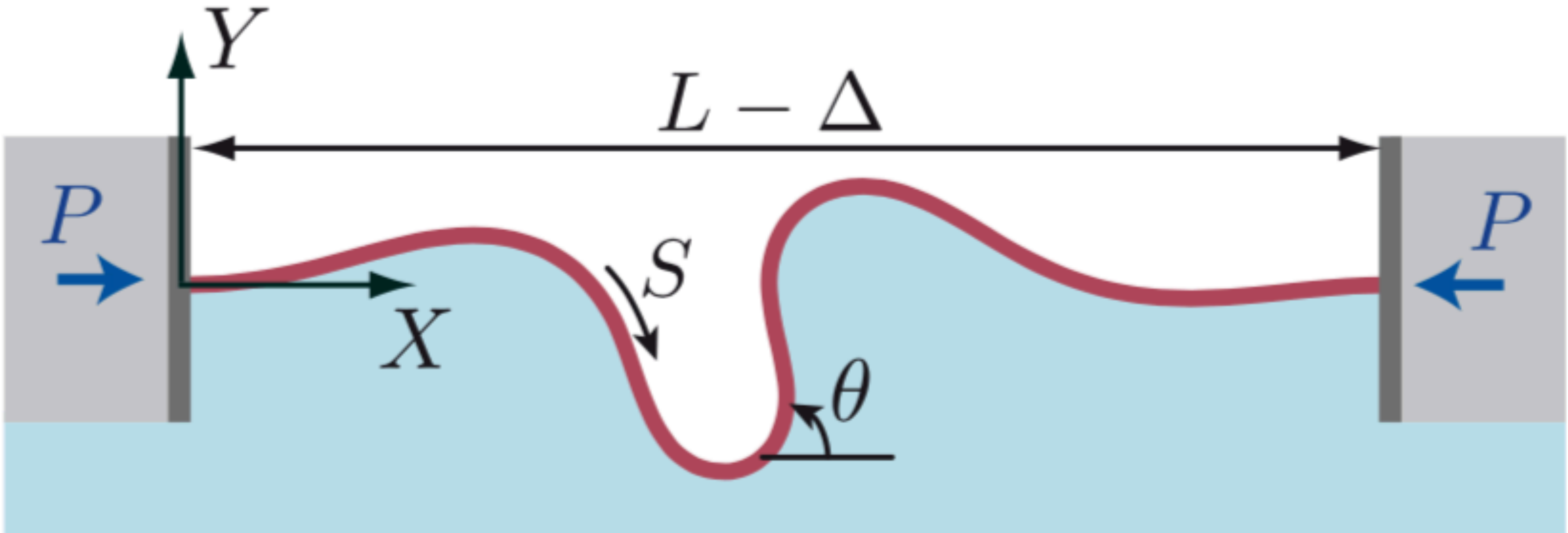
Comparison: zoom
T/(4π²)
Δ
0.5
1.0
1.5
2.0
−2
−4
−6

Comparison



rem: perturbative shear force Q=10E-18 in ManLab

2nd Example: beam on foundation

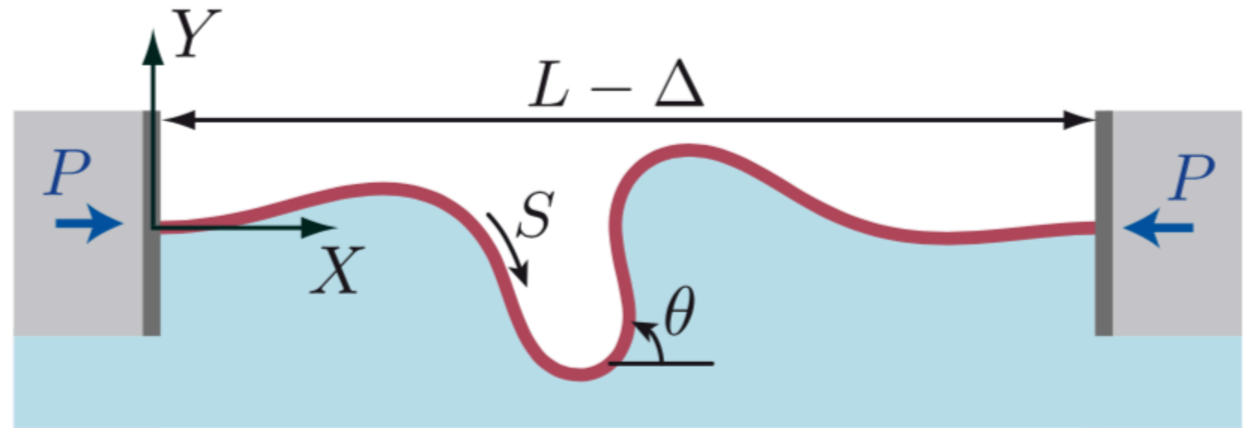


Planar elastica on hydrostatic (nonlinear) foundation
Mode switching: secondary bifurcations

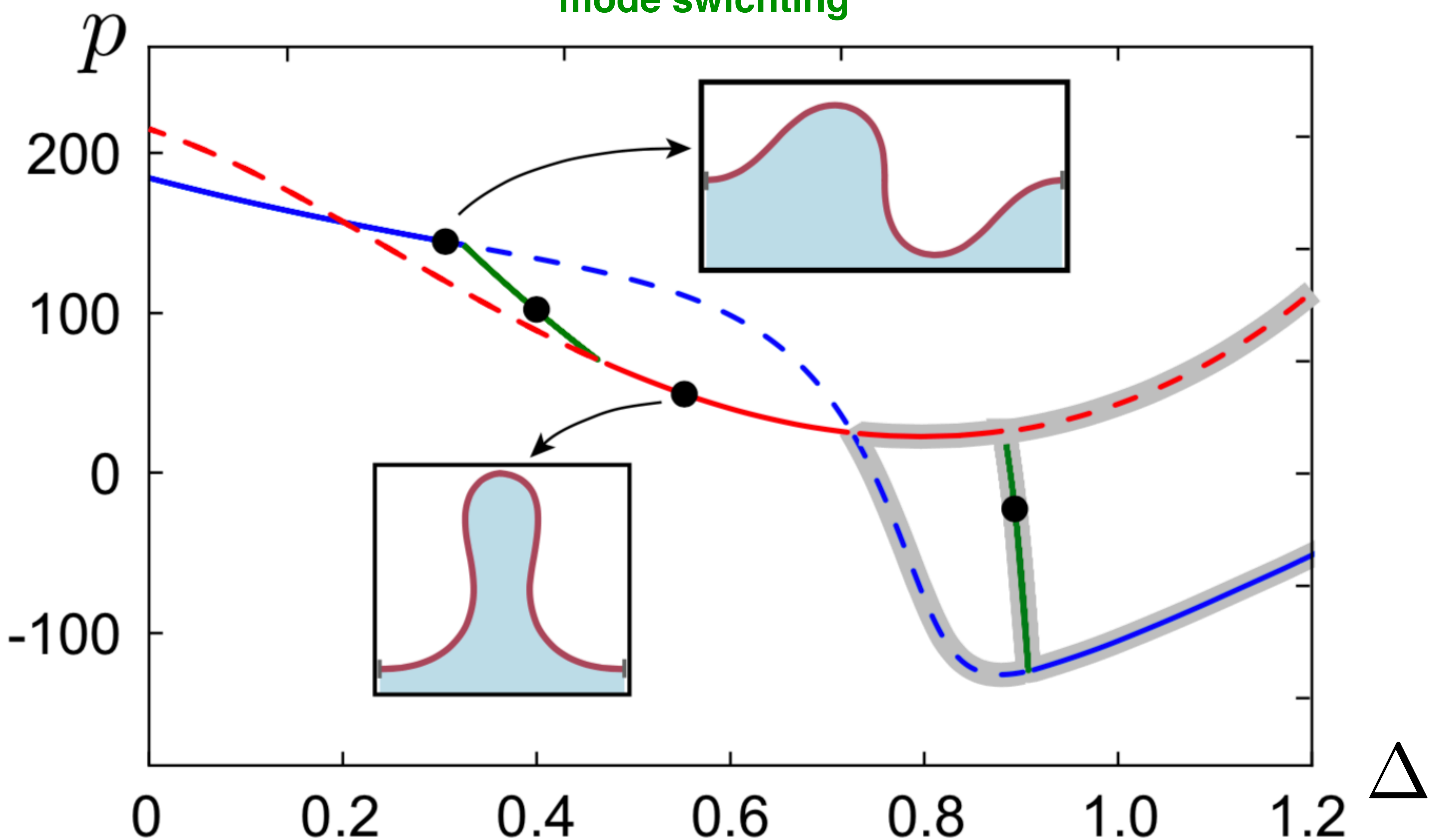
imposed: axial displacement Δ

Beam on foundation

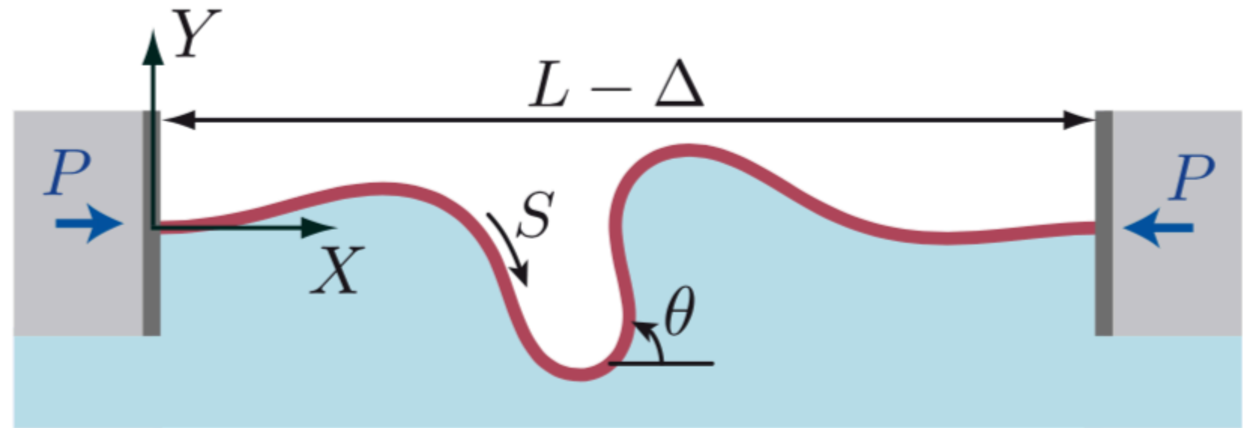
$\eta = 7$ **foundation stiffness**



mode swichting



Beam on foundation



$$\eta = 7 \ \ (\eta^4 = 2401)$$

singular perturbation

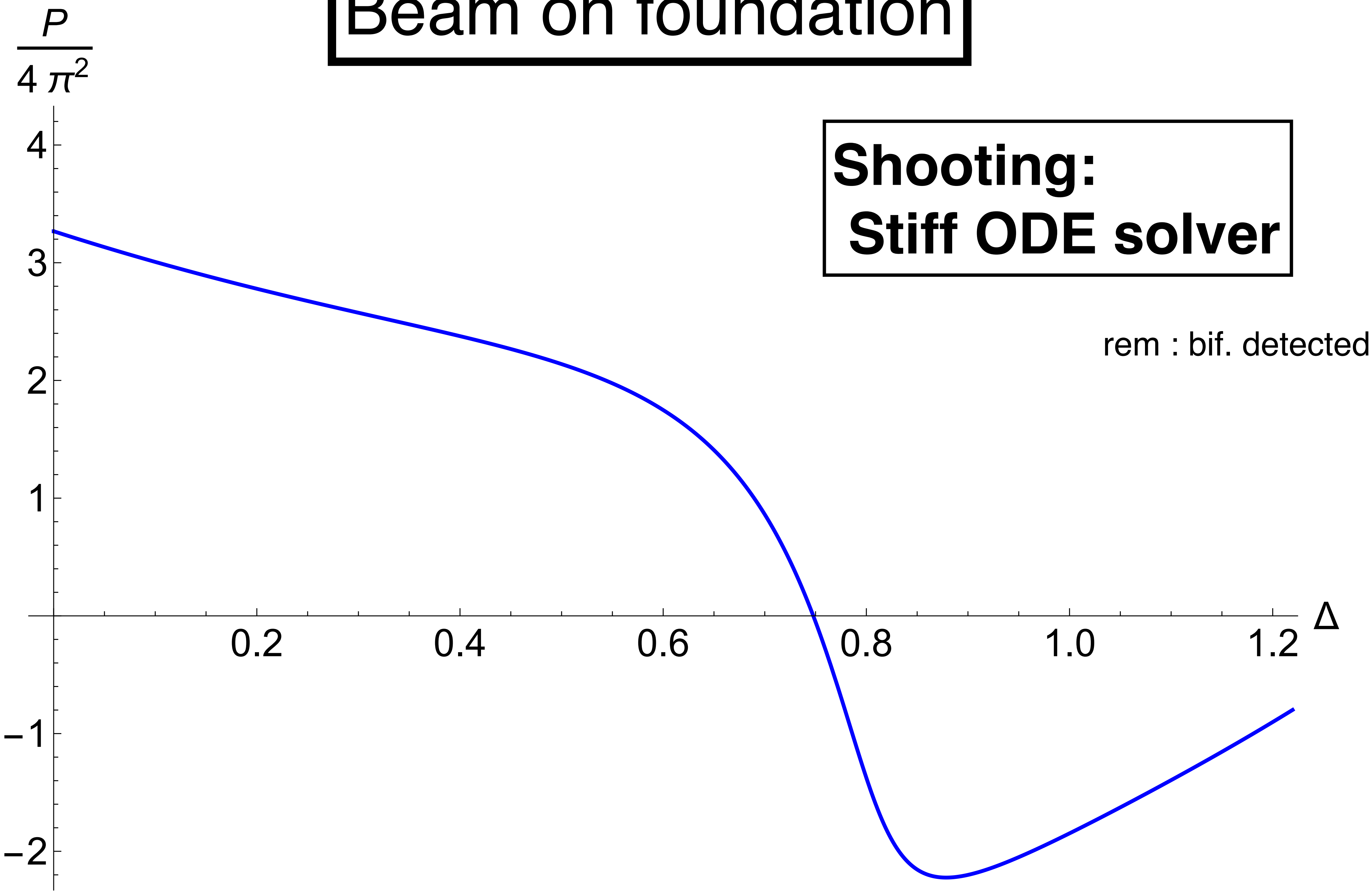
$$x'(s) = \cos\theta(s)$$
$$y'(s) = \sin\theta(s)$$
$$\theta'(s) = m(s)$$
$$m'(s) = n_x(s)\sin\theta(s) - n_y(s)\cos\theta(s)$$
$$n_x'(s) = -\eta^4 y(s)\sin\theta(s)$$
$$n_y'(s) = \eta^4 y(s)\cos\theta(s)$$

$$x(0) = 0$$
$$x(1) = 1 - \Delta$$
$$y(0) = 0$$
$$y(1) = 0$$
$$\theta(0) = 0$$
$$\theta(1) = 0$$

ODE system

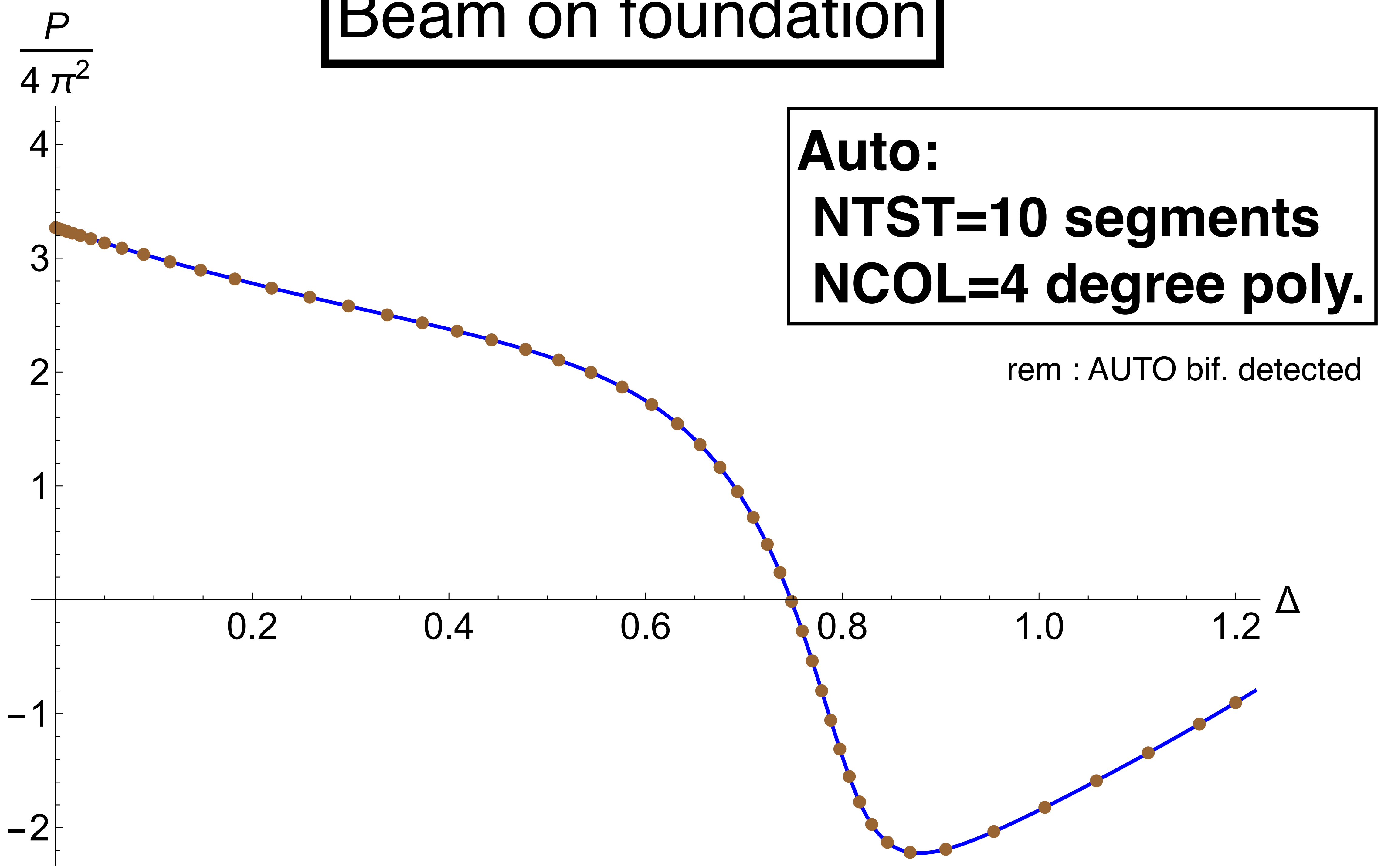
bound. cond.

Beam on foundation



Shooting: 720 pts (160 sec)

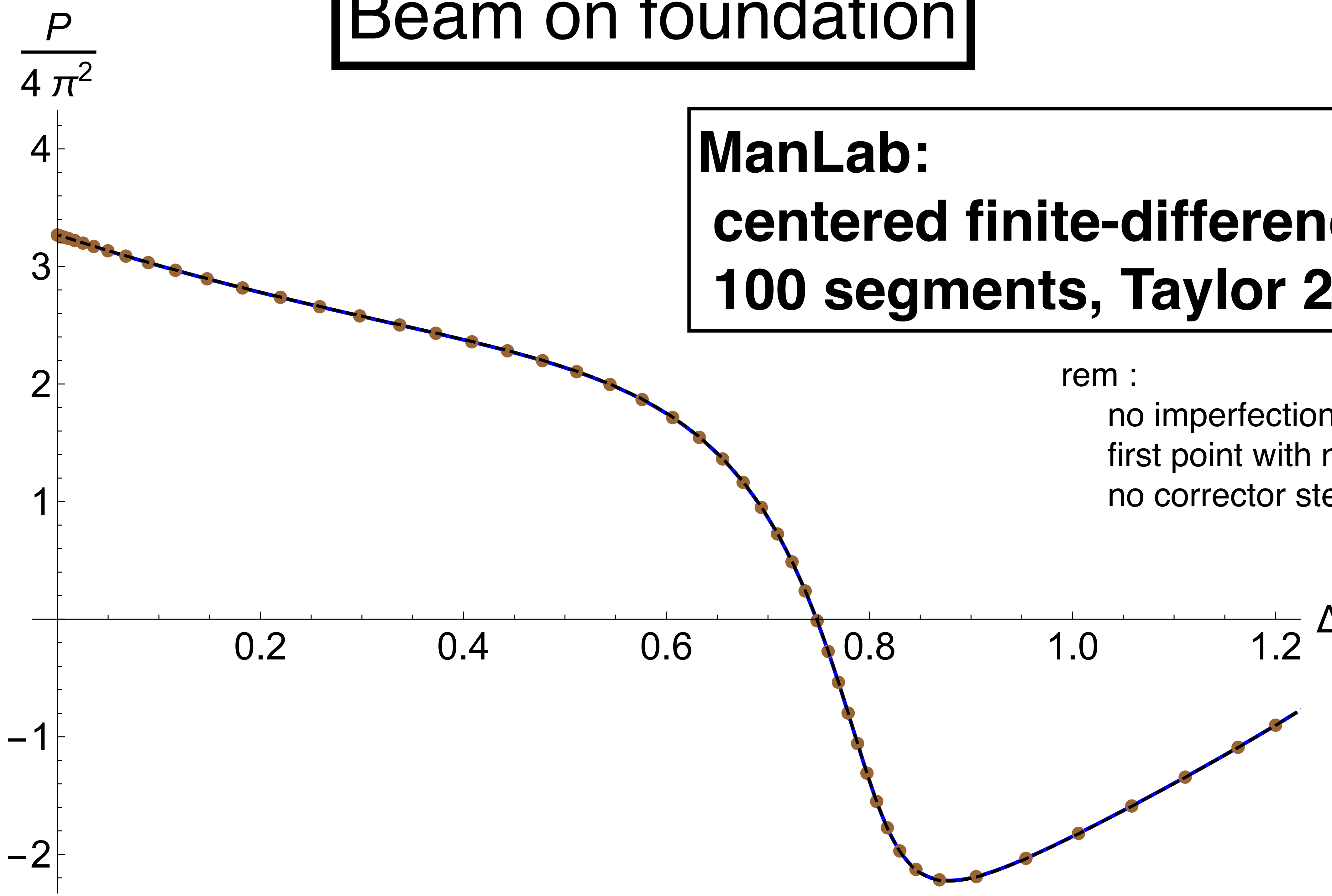
Beam on foundation



Shooting: 720 pts (160 sec)

AUTO: 54 pts (0.1sec)

Beam on foundation

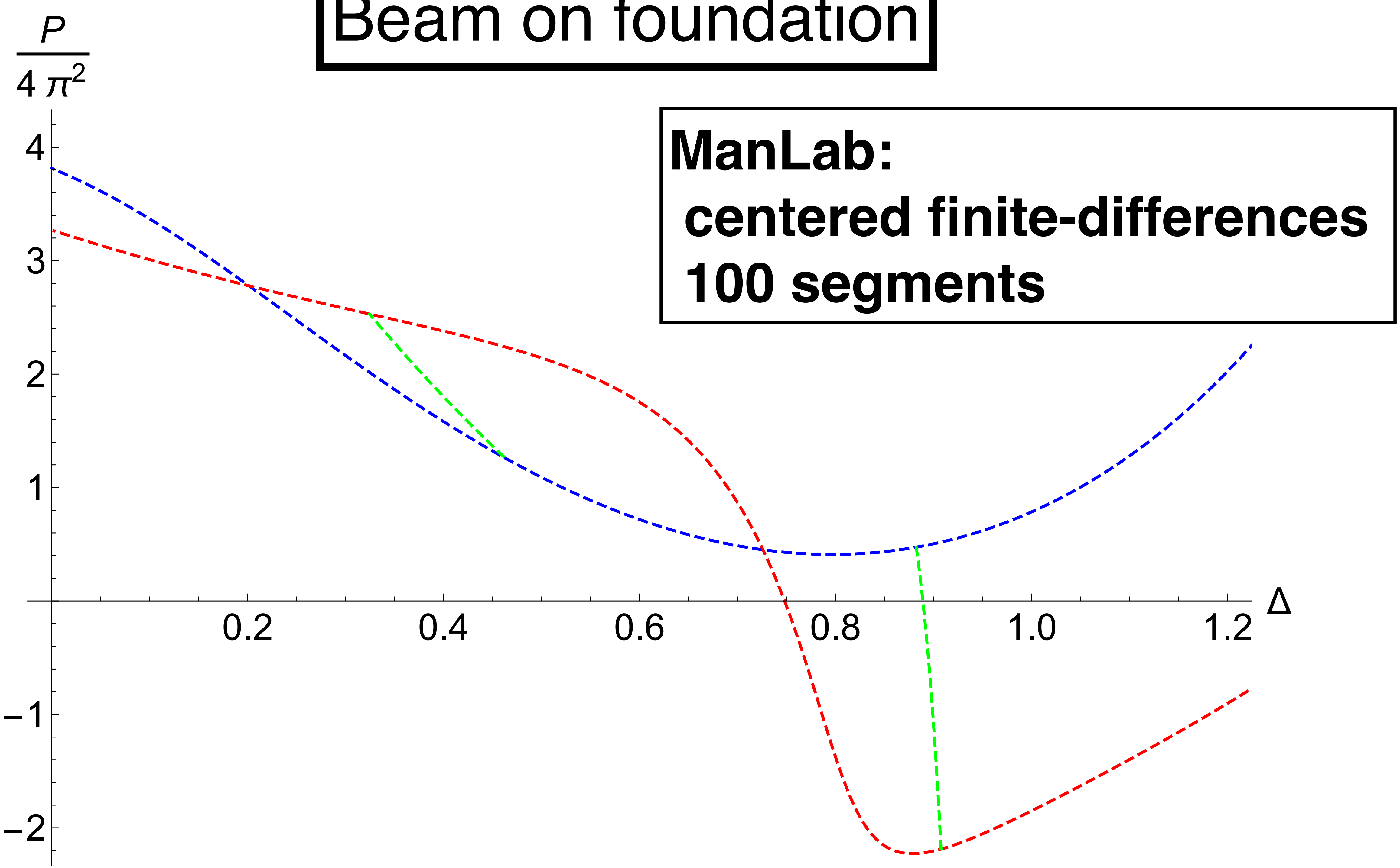


ManLab:
centered finite-differences
100 segments, Taylor 20

rem :
no imperfection
first point with newton
no corrector step

Shooting: 720 pts (160 sec)
AUTO: 54 pts (0.1sec)
ManLab: 21 branches (1.6sec)

Beam on foundation

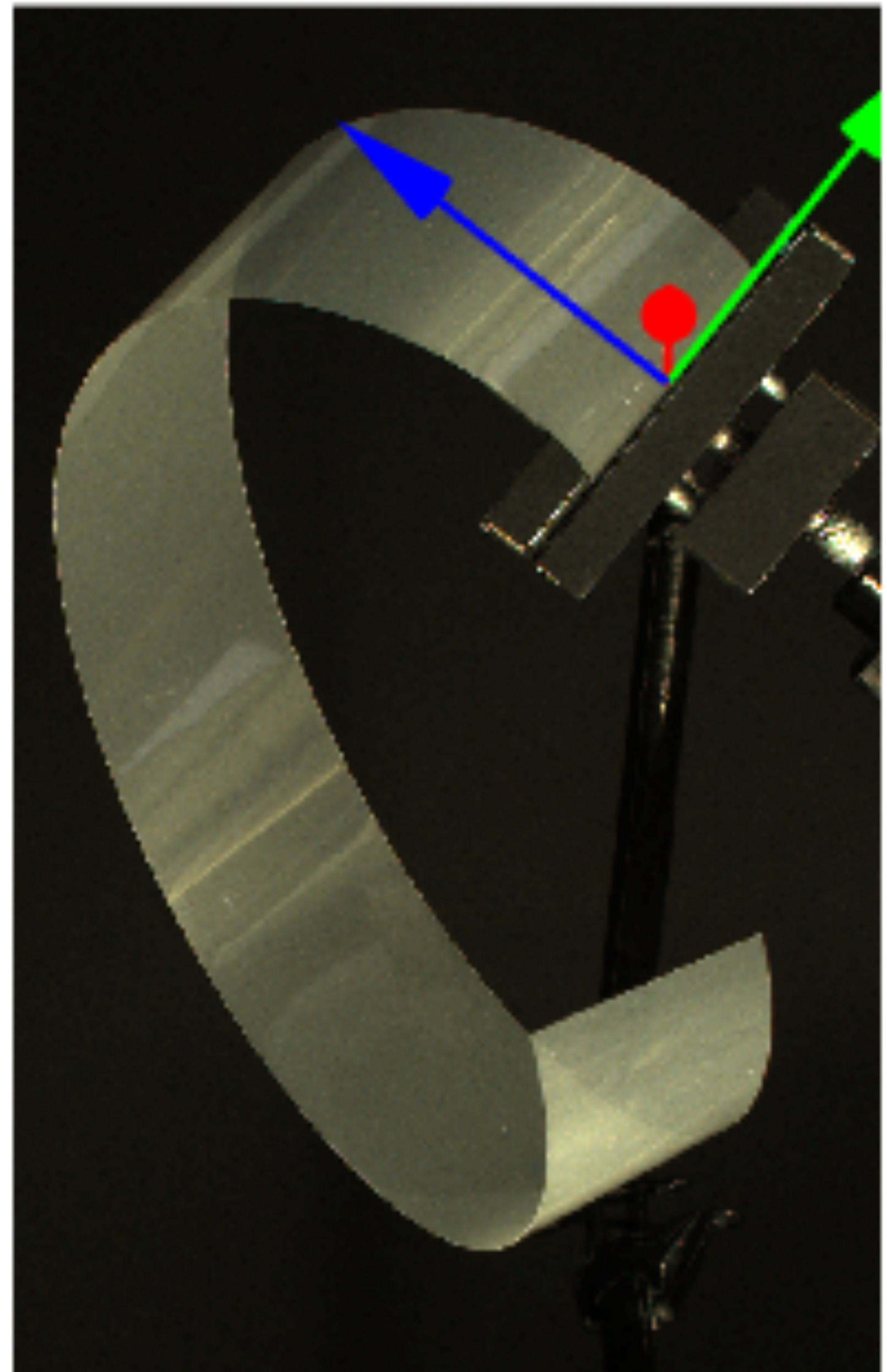


ManLab: 8 sec CPU (real ~ 2 min) point+click GUI

```matlab
function [Rf,dRf] = equations(sys,Uf,dUf)
% Equations of the system of the form Rf(Uf) = C + L(Uf) + Q(Uf,Uf).

R = zeros(sys.neq,1);
Ra = zeros(sys.neq_aux,1);

N = sys.parameters.N;
eta = sys.parameters.eta;
K = sys.parameters.K;
eps = sys.parameters.eps;

x = Uf(1:N-1);
y = Uf(N:2*(N-1));
th = Uf(2*(N-1)+1:3*(N-1));
nind = 3*(N-1);

m = Uf(nind+1:nind+(N+1));
nx = Uf(nind+(N+1)+1:nind+2*(N+1));
ny = Uf(nind+2*(N+1)+1:nind+3*(N+1));
lambda = Uf(sys.neq+1);
thm = Uf(sys.neq+1+1:sys.neq+1+N);      dthm = dUf(sys.neq+1+1:sys.neq+1+N);
c = Uf(sys.neq+1+N+1:sys.neq+1+2*N);    dc = dUf(sys.neq+1+N+1:sys.neq+1+2*N);
s = Uf(sys.neq+1+2*N+1:sys.neq+1+3*N);    ds = dUf(sys.neq+1+2*N+1:sys.neq+1+3*N);

%%% Residues
R(1:N,1) = K * [-x(1) ; x ; (2*(1-lambda)-x(end))] - c;       dR    = zeros(size(R));
R(N+1:2*N,1) = K * [-y(1) ; y ; -y(end)] - s;
R(2*N+1:3*N,1) = K * [-th(1) ; th ; -th(end)] - 0.5*m(1:end-1) - 0.5*m(2:end);

R(3*N+1:4*N,1) = K * m - 0.5*nx(1:end-1).*s - 0.5*nx(2:end).*s + 0.5*ny(1:end-1).*c + 0.5*ny(2:end).*c;
R(4*N+1:5*N,1) = K * nx + 0.5*eta^4*[-y(1) ; y].*s + 0.5*eta^4*[y ; -y(end)].*s;
R(5*N+1:6*N,1) = K * ny - 0.5*eta^4*[-y(1) ; y].*c - 0.5*eta^4*[y ; -y(end)].*c - eps*ones(N,1);

dRa = zeros(size(Ra));

Ra(1:N,1) = thm - 0.5*[-th(1) ; th] - 0.5*[th ; -th(end)];
Ra(N+1:2*N,1) = c - cos(thm);      dRa(N+1:2*N,1) = dc + s.*dthm;
Ra(2*N+1:3*N,1) = s - sin(thm);      dRa(2*N+1:3*N,1) = ds - c.*dthm;

%%% Concatenation
Rf= [R ; Ra];
dRf=[dR;dRa];
```

3rd example: Elastic ribbon

Clamped-Free
naturally curved ribbon
sagging under its own weight



L = 29 cm
w = 3 cm
h = 0.1 mm
Rcurv = 3.75 cm

PET : PolyEthylene Terephthalate

E = 3.4 Gpa
nu = 0.4
rho = 1250 kg/m3

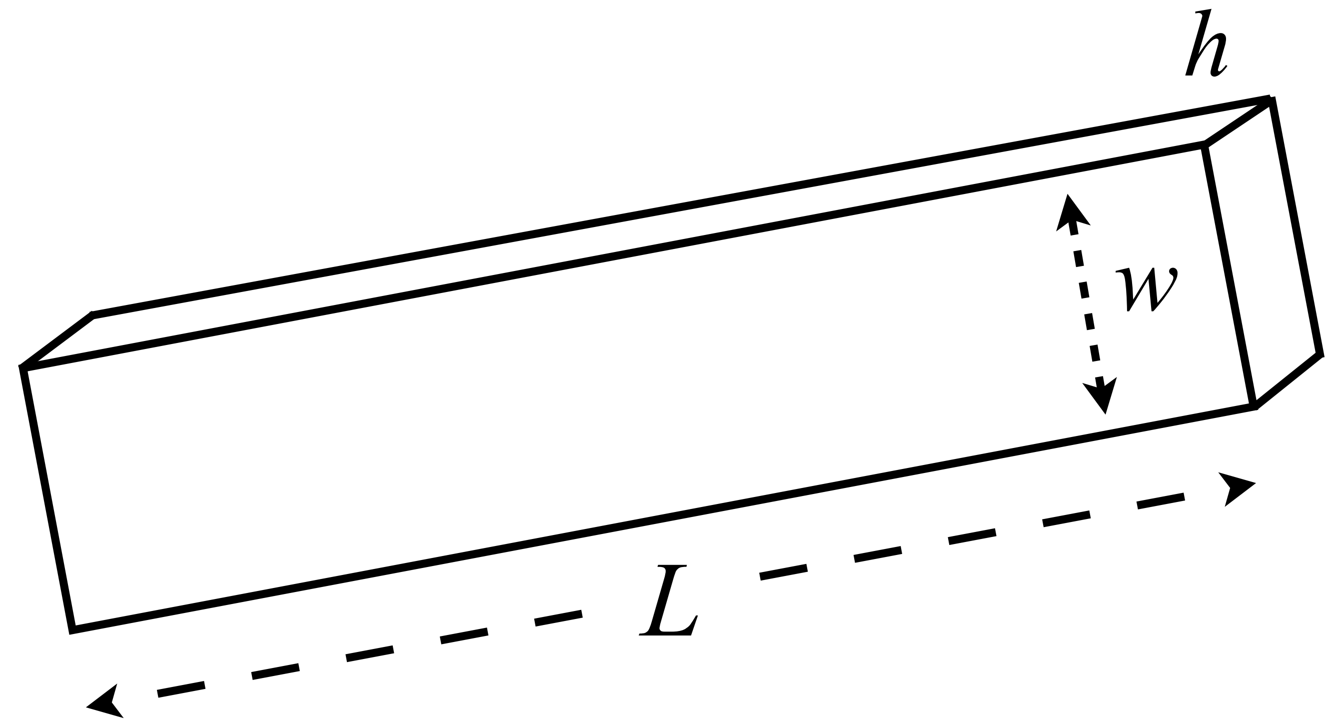
Victor Romero (d'Alembert / INRIA)

Elastic ribbon

rod $L \gg h, w$

plate $L, w \gg h$

ribbon $L \gg w \gg h$



Elastic energy for a plate

$$E_{\text{bend}} = \frac{D}{2} \int\int \left\{ (1-\nu)\,\mathrm{Tr}\,K^2 + \nu (TrK)^2 \right\} dS$$

$$E_{\text{ext}} = \frac{A}{2} \int\int \left\{ (1-\nu)\,\mathrm{Tr}\,\epsilon^2 + \nu (Tr\epsilon)^2 \right\} dS$$

Elastic ribbon

$$K = \begin{pmatrix} K_x & K_{xy} \\ K_{xy} & K_y \end{pmatrix} \qquad \epsilon = \begin{pmatrix} \epsilon_{xx} & \epsilon_{xy} \\ \epsilon_{xy} & \epsilon_{yy} \end{pmatrix}$$

$$D = \frac{Yh^3}{12(1-\nu^2)} \qquad A = \frac{Yh}{(1-\nu^2)}$$

Elastic energy for a plate

$$E_{\mathrm{bend}} = \frac{D}{2} \int\int \left\{ (1-\nu)\,\mathrm{Tr}\,K^2 + \nu (TrK)^2 \right\} dS$$

$$E_{\mathrm{ext}} = \frac{A}{2} \int\int \left\{ (1-\nu)\,\mathrm{Tr}\,\epsilon^2 + \nu (Tr\epsilon)^2 \right\} dS$$

Elastic ribbon

Assume inextensibility:
=> developable surface
=> generatrices

Sadowsky 1930
Wunderlich 1962
Starostin 2008
Dias 2014

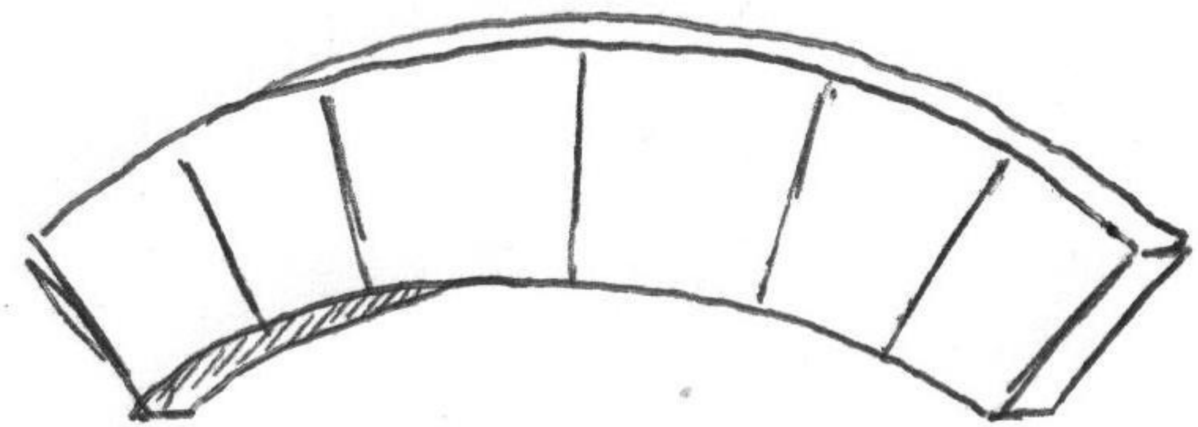
Elastic energy for a plate

$$E_{\mathrm{bend}} = \frac{D}{2} \int\int \left\{ (1-\nu)\,\mathrm{Tr}\, K^2 + \nu (TrK)^2 \right\} dS$$

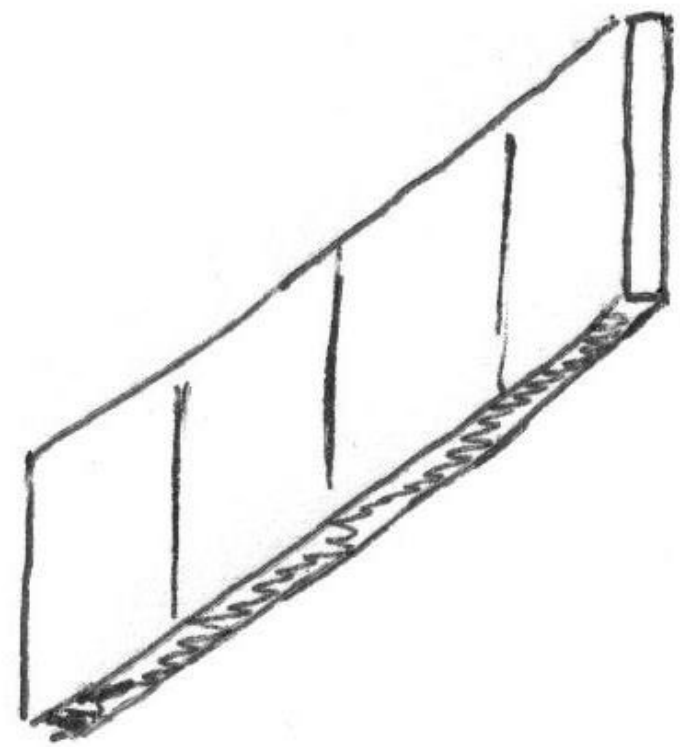
$$E_{\mathrm{ext}} = \frac{A}{2} \int\int \left\{ (1-\nu)\,\mathrm{Tr}\, \epsilon^2 + \nu (Tr\epsilon)^2 \right\} dS$$

Elastic ribbon

no geodesic curvature



no shear



Equations for elastic ribbons

kinematics

$$x' = d_{3x}$$
$$y' = d_{3y}$$
$$z' = d_{3z}$$
$$d'_{3x} = u_2\, d_{1x} - u_1\, d_{2x}$$
$$d'_{3y} = u_2\, d_{1y} - u_1\, d_{2y}$$
$$d'_{3z} = u_2\, d_{1z} - u_1\, d_{2z}$$
$$d'_{1x} = u_3\, d_{2x} - u_2\, d_{3x}$$
$$d'_{1y} = u_3\, d_{2y} - u_2\, d_{3y}$$
$$d'_{1z} = u_3\, d_{2z} - u_2\, d_{3z}$$
$$d'_{2x} = u_1\, d_{3x} - u_3\, d_{1x}$$
$$d'_{2y} = u_1\, d_{3y} - u_3\, d_{1y}$$
$$d'_{2z} = u_1\, d_{3z} - u_3\, d_{1z}.$$

dynamics

$$n'_1 = n_2\, u_3 - n_3 u_2 - f_1 + \rho A\,(\ddot{x}\, d_{1x} + \ddot{y}\, d_{1y} + \ddot{z}\, d_{1z})$$
$$n'_2 = n_3\, u_1 - n_1 u_3 - f_2 + \rho A\,(\ddot{x}\, d_{2x} + \ddot{y}\, d_{2y} + \ddot{z}\, d_{2z})$$
$$n'_3 = n_1\, u_2 - n_2 u_1 - f_3 + \rho A\,(\ddot{x}\, d_{3x} + \ddot{y}\, d_{3y} + \ddot{z}\, d_{3z})$$
$$m'_1 = m_2\, u_3 - m_3 u_2 + n_2$$
$$m'_2 = m_3\, u_1 - m_1 u_3 - n_1$$
$$m'_3 = m_1\, u_2 - m_2 u_1$$

nonlinear constitutive relations

$$m_1 = K\left(1 - \frac{u_3^4}{u_1^4}\right) u_1$$
$$u_2 = 0$$
$$m_3 = 2K\left(1 + \frac{u_3^2}{u_1^2}\right) u_3$$

Dias & Audoly (JMPS) 2014

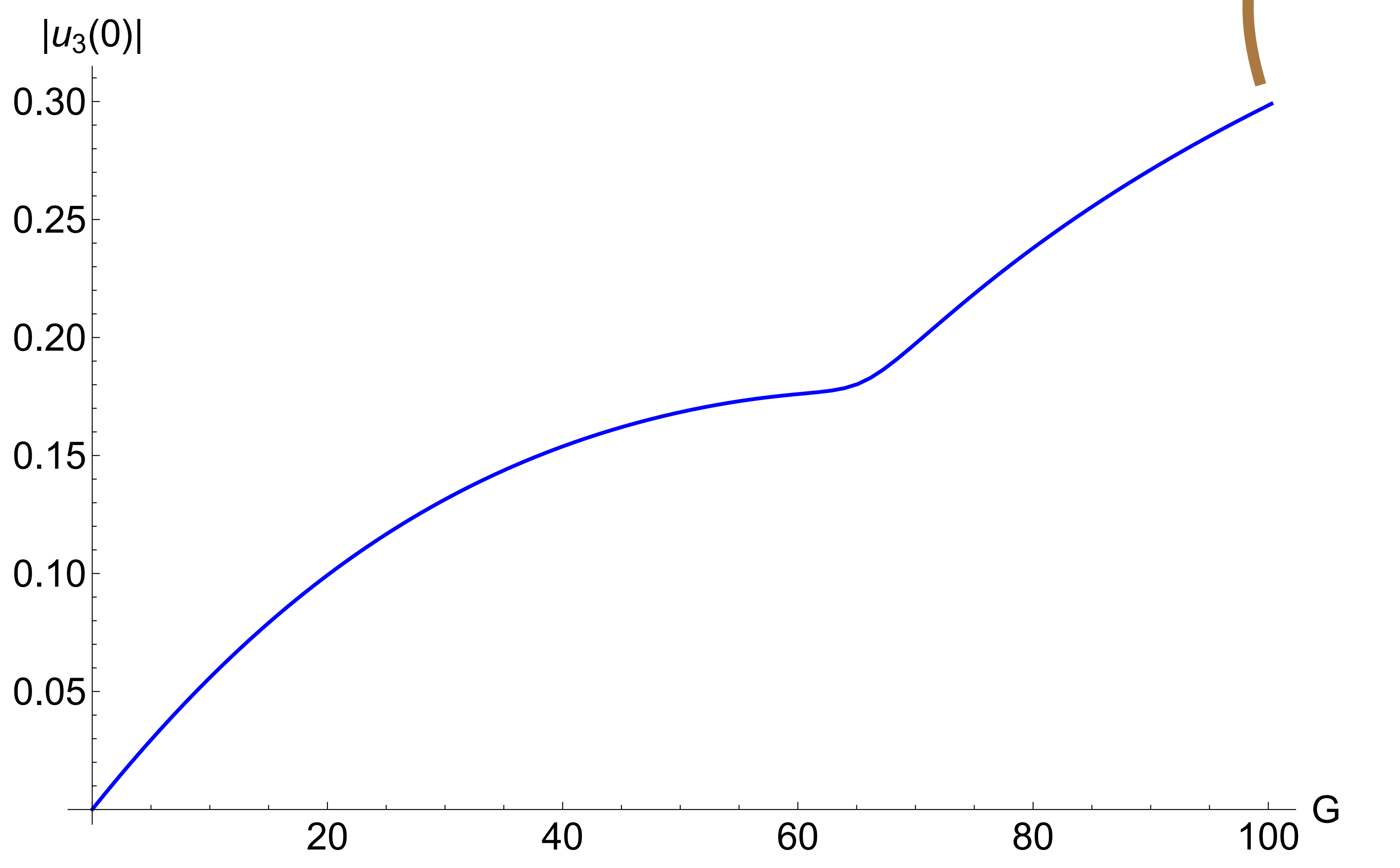
Elastic ribbon

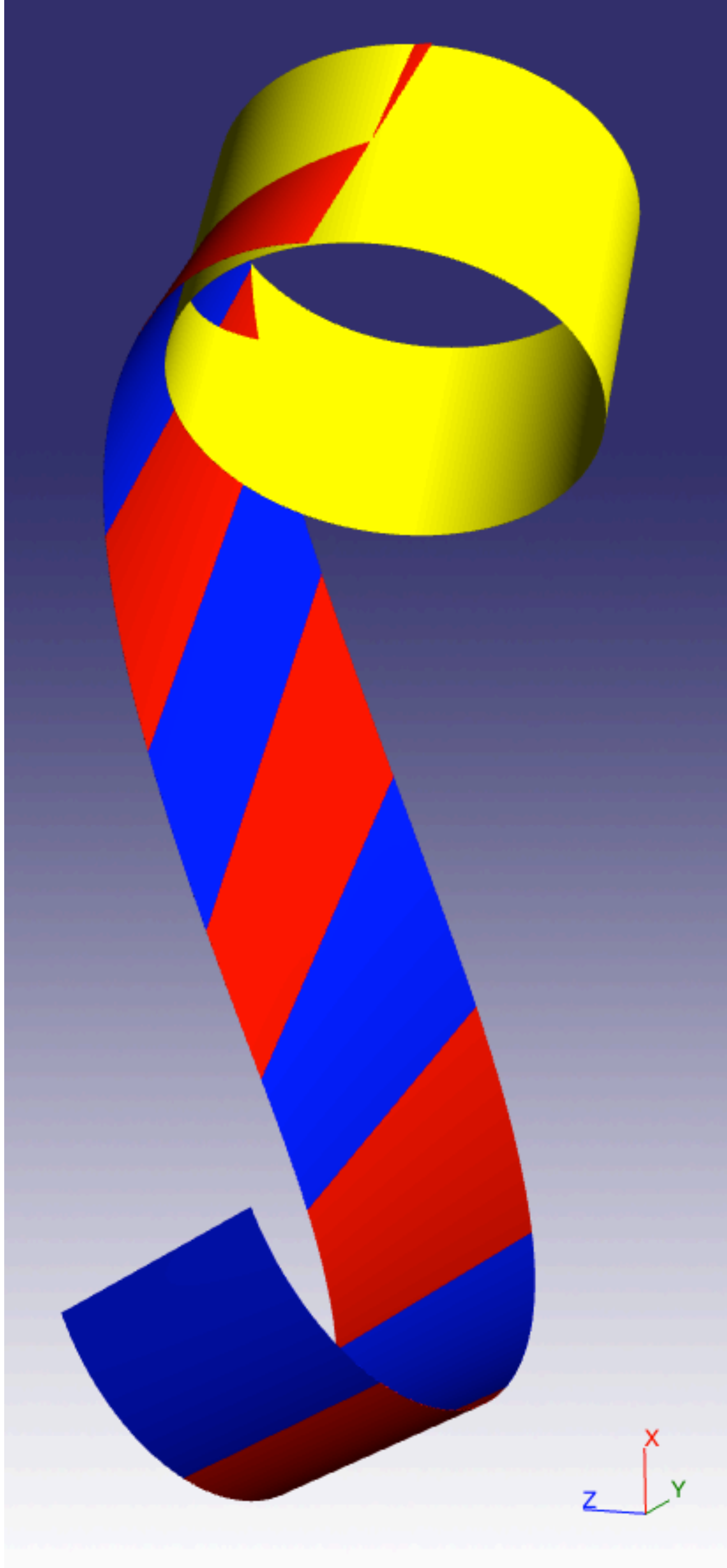
Goal: obtain K=10, G=100

adim natural curvature

adim weight

Shooting: 42 pts (8sec)





Elastic ribbon

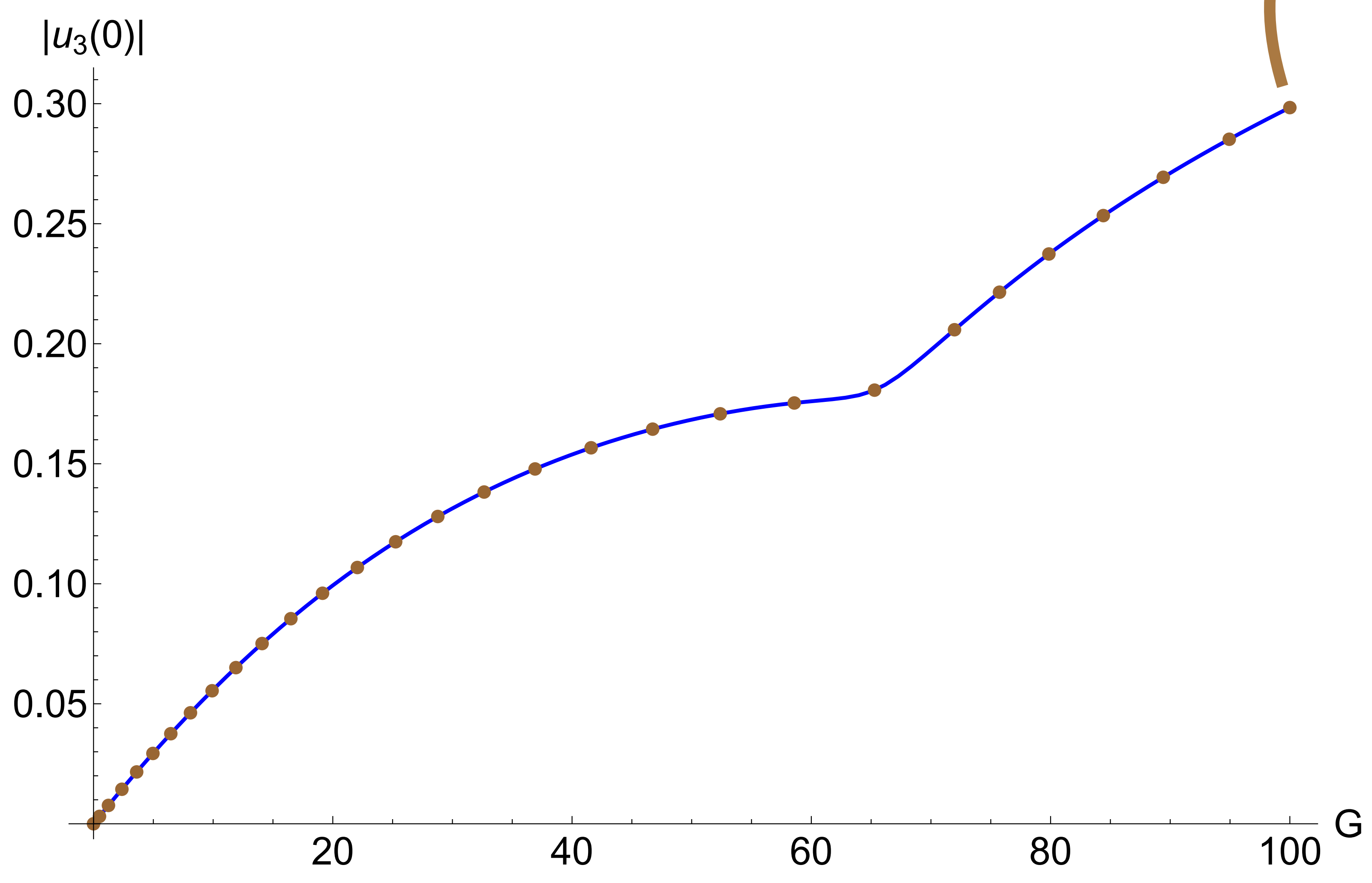
Goal: obtain K=10, G=100

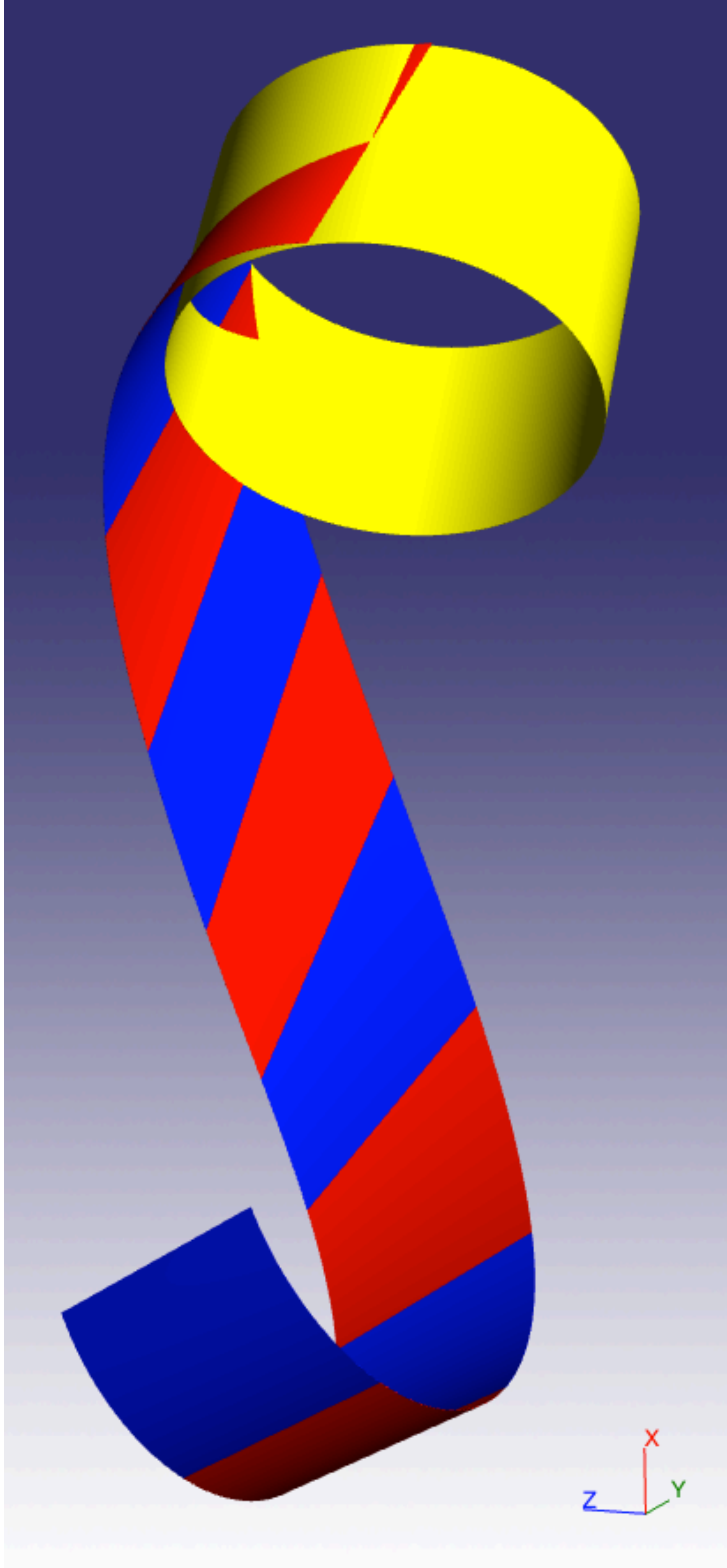
adim natural curvature

adim weight

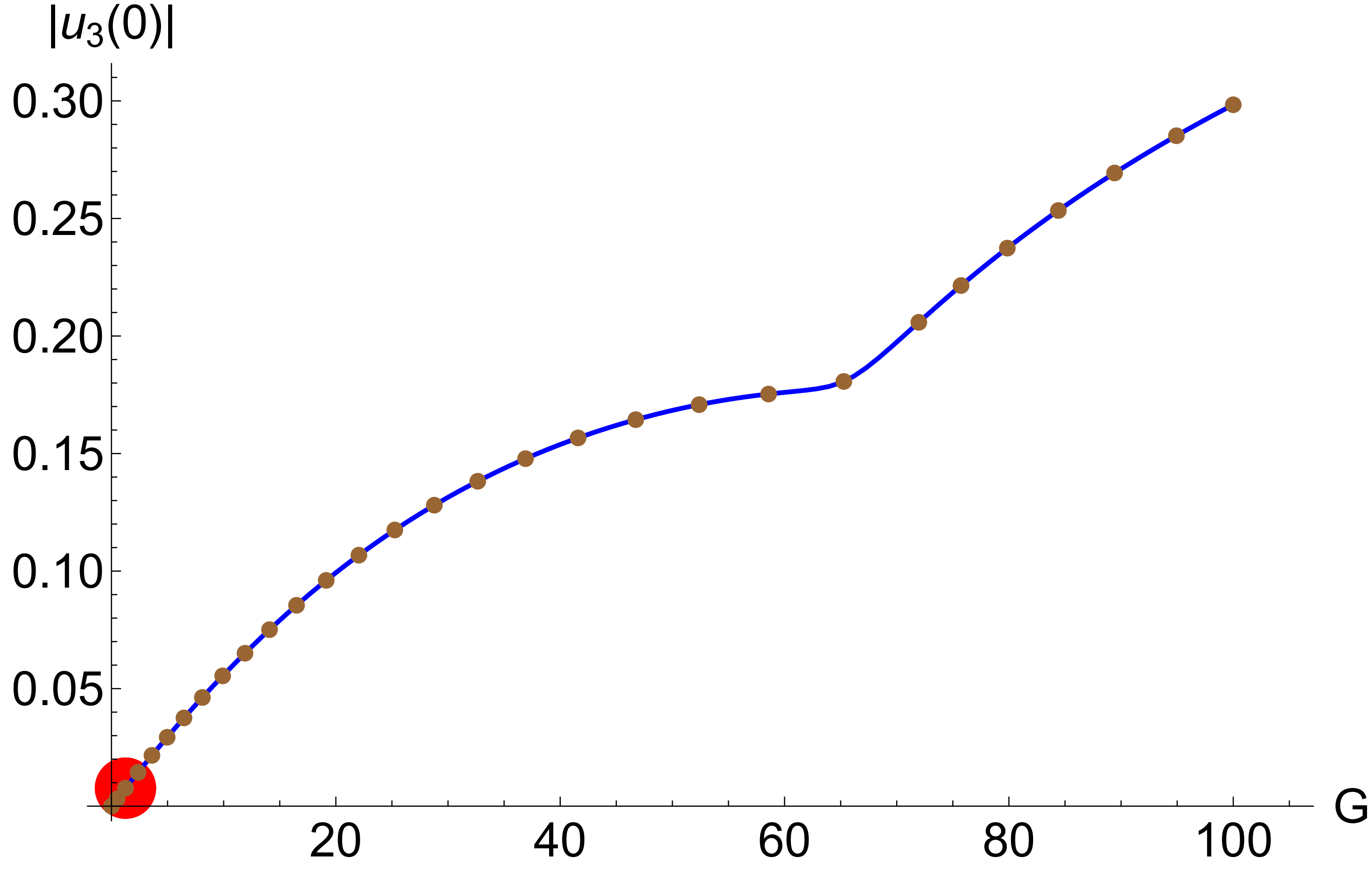
Shooting: 42 pts (8sec)

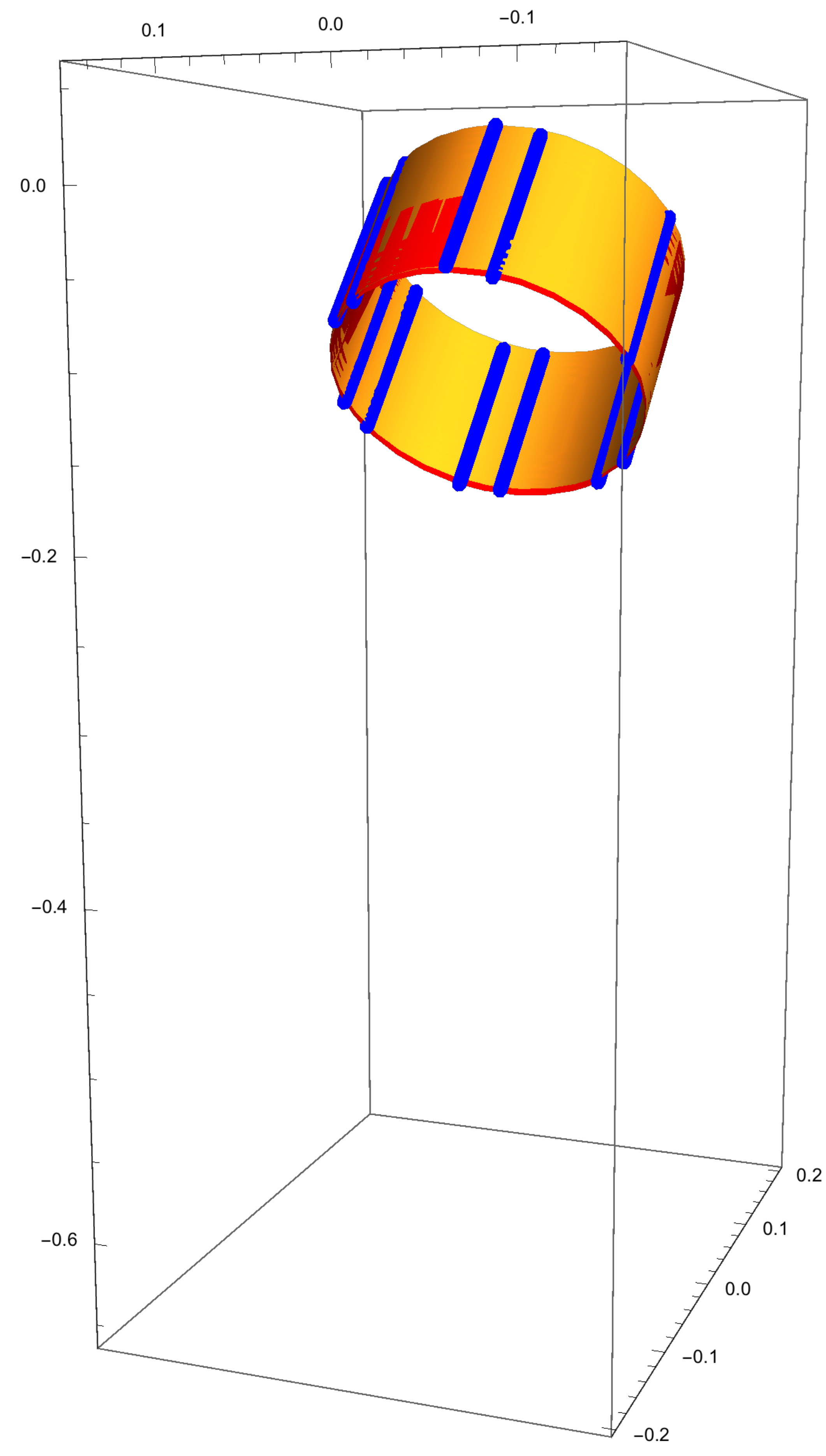
AUTO: 30 pts (0.11sec) (NTST=10, NCOL=4)



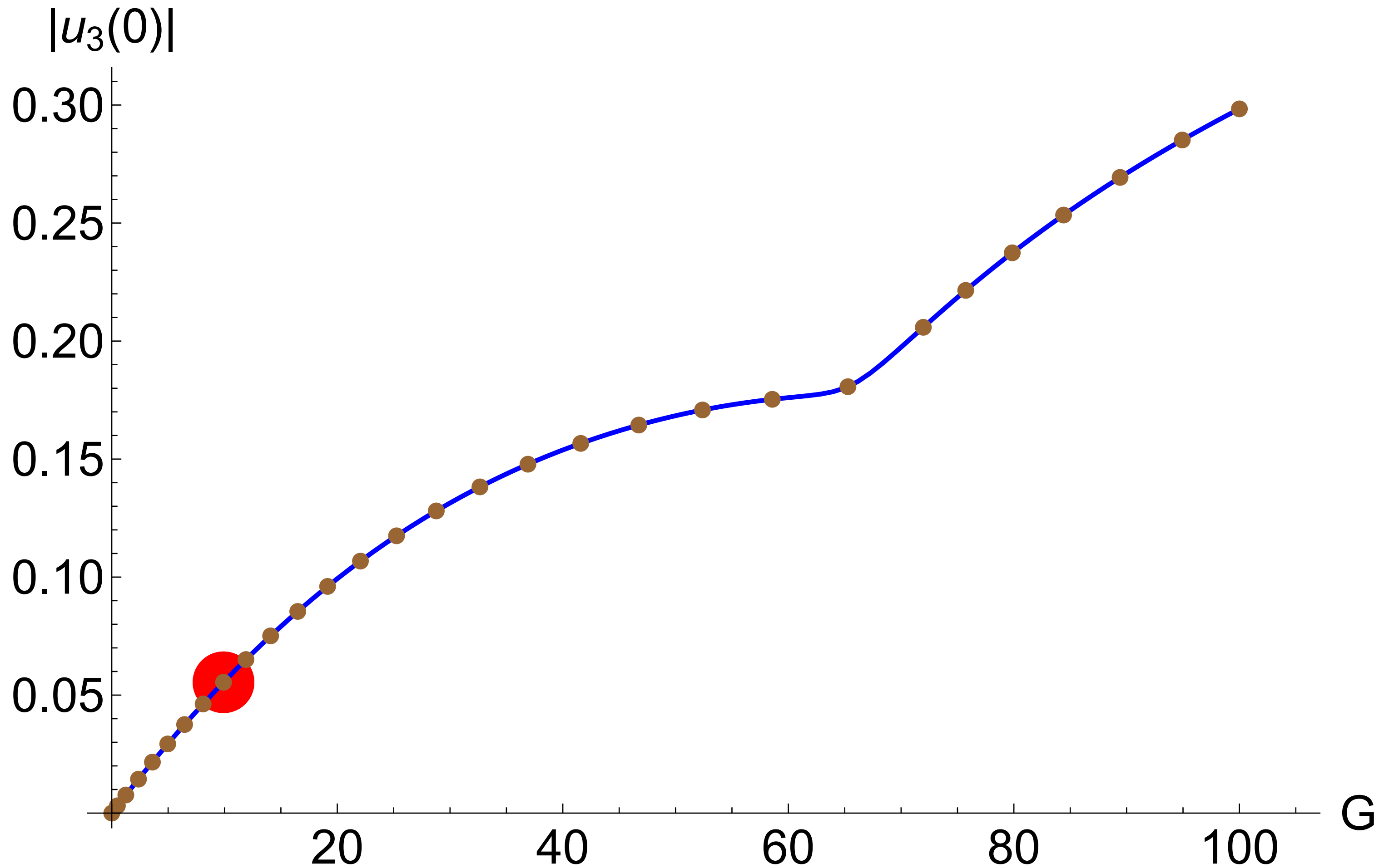


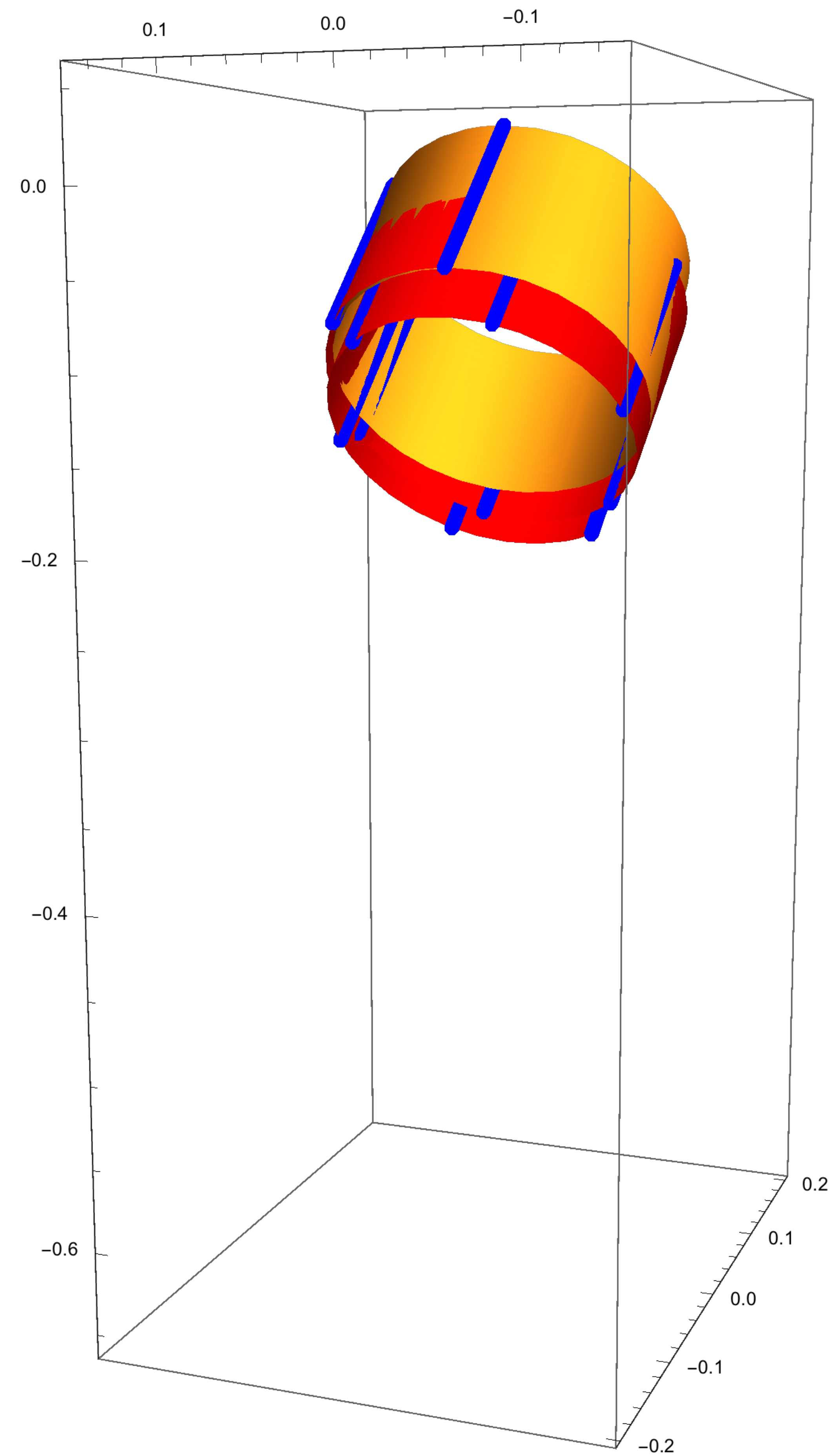
Shooting & AUTO:
sequence of equilibrium



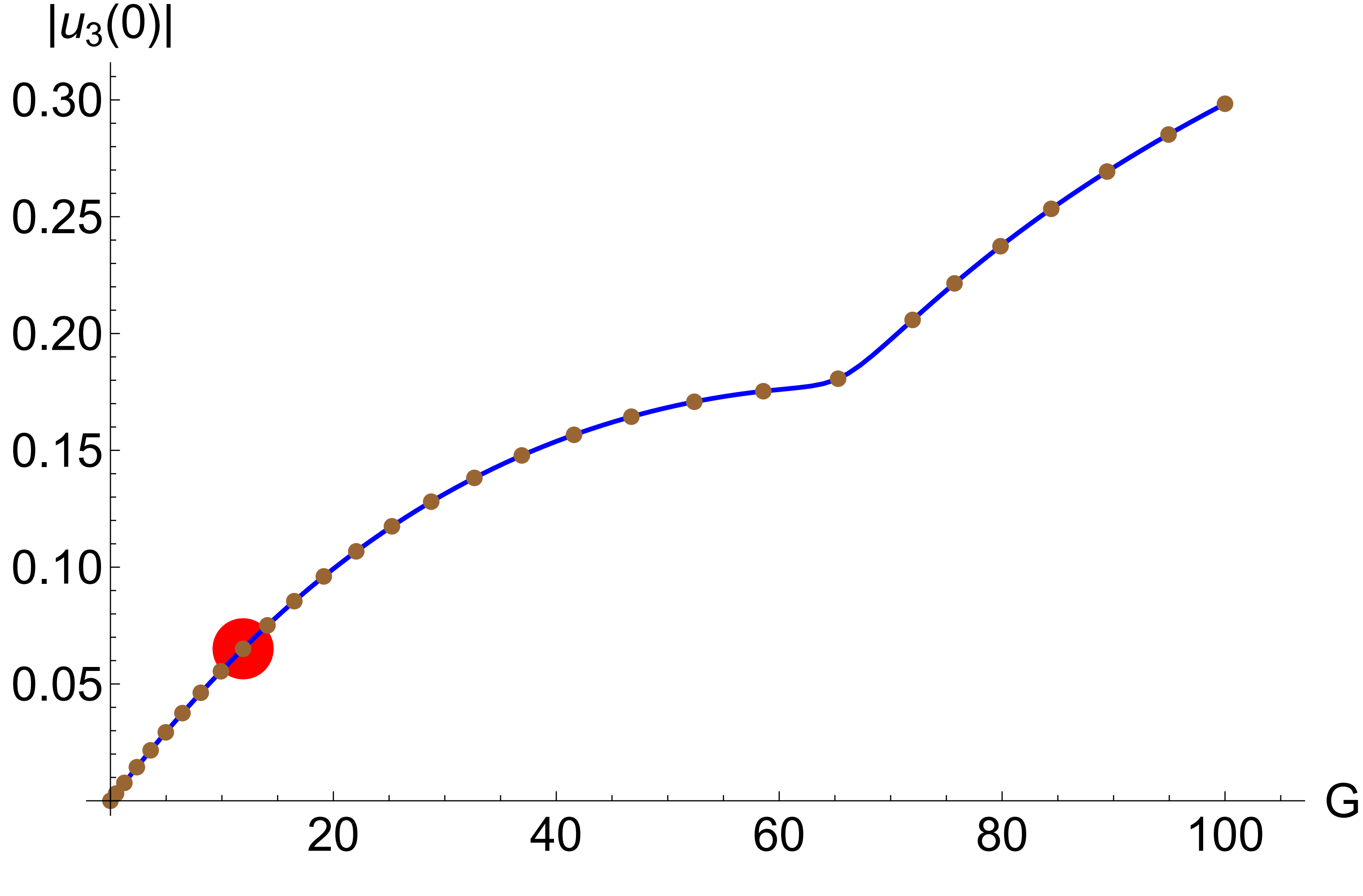


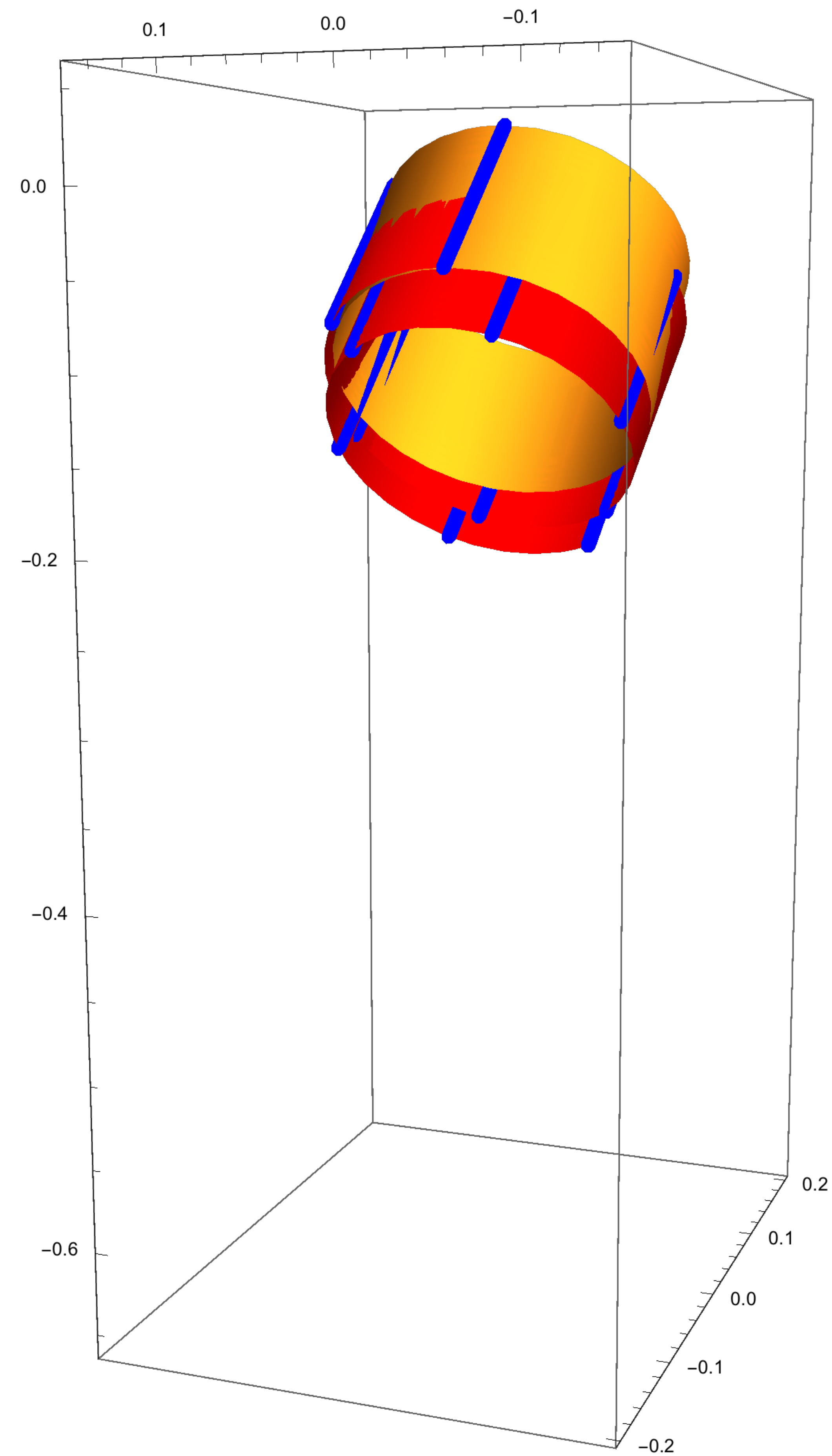
Shooting & AUTO:
sequence of equilibrium



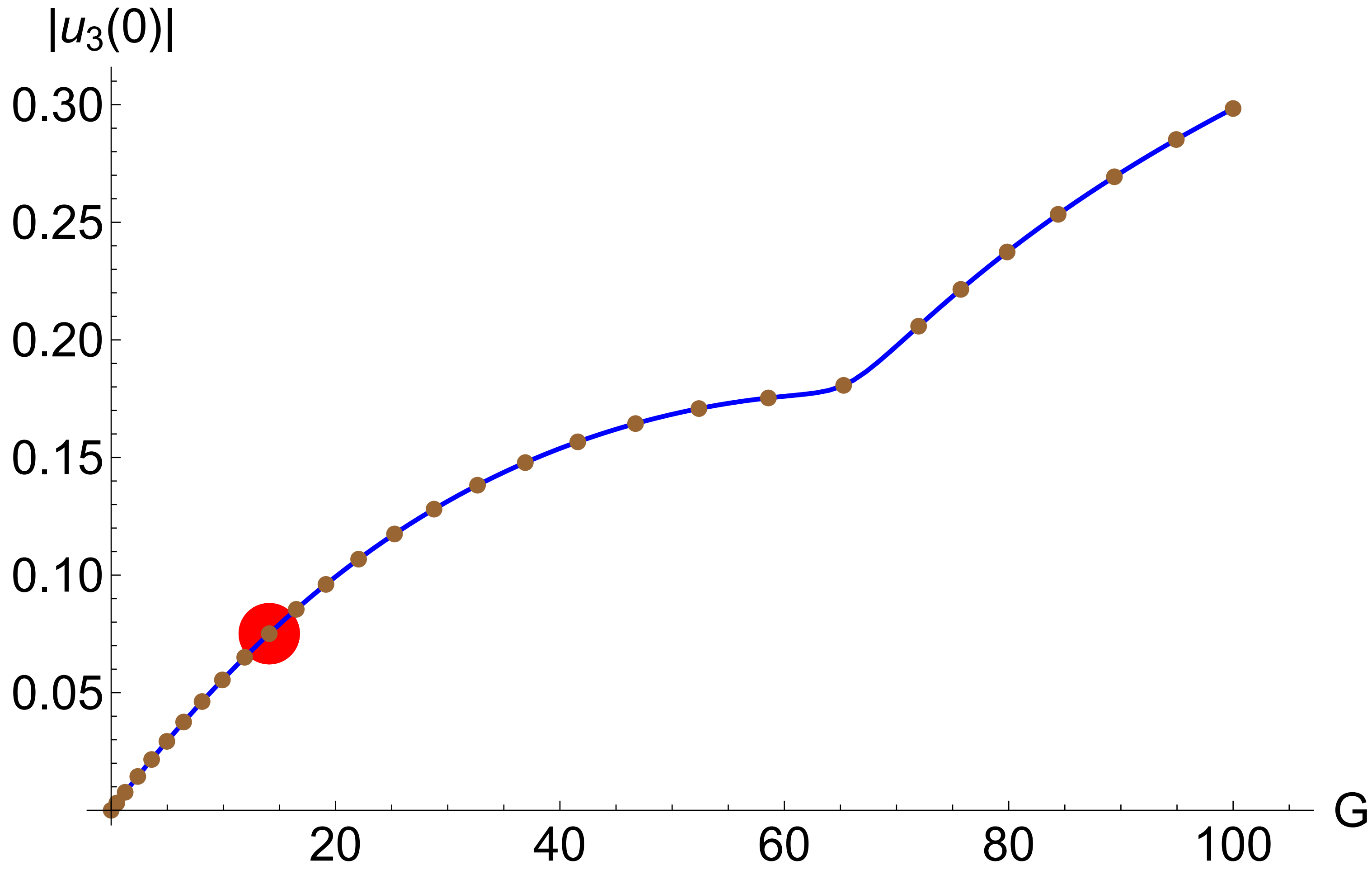


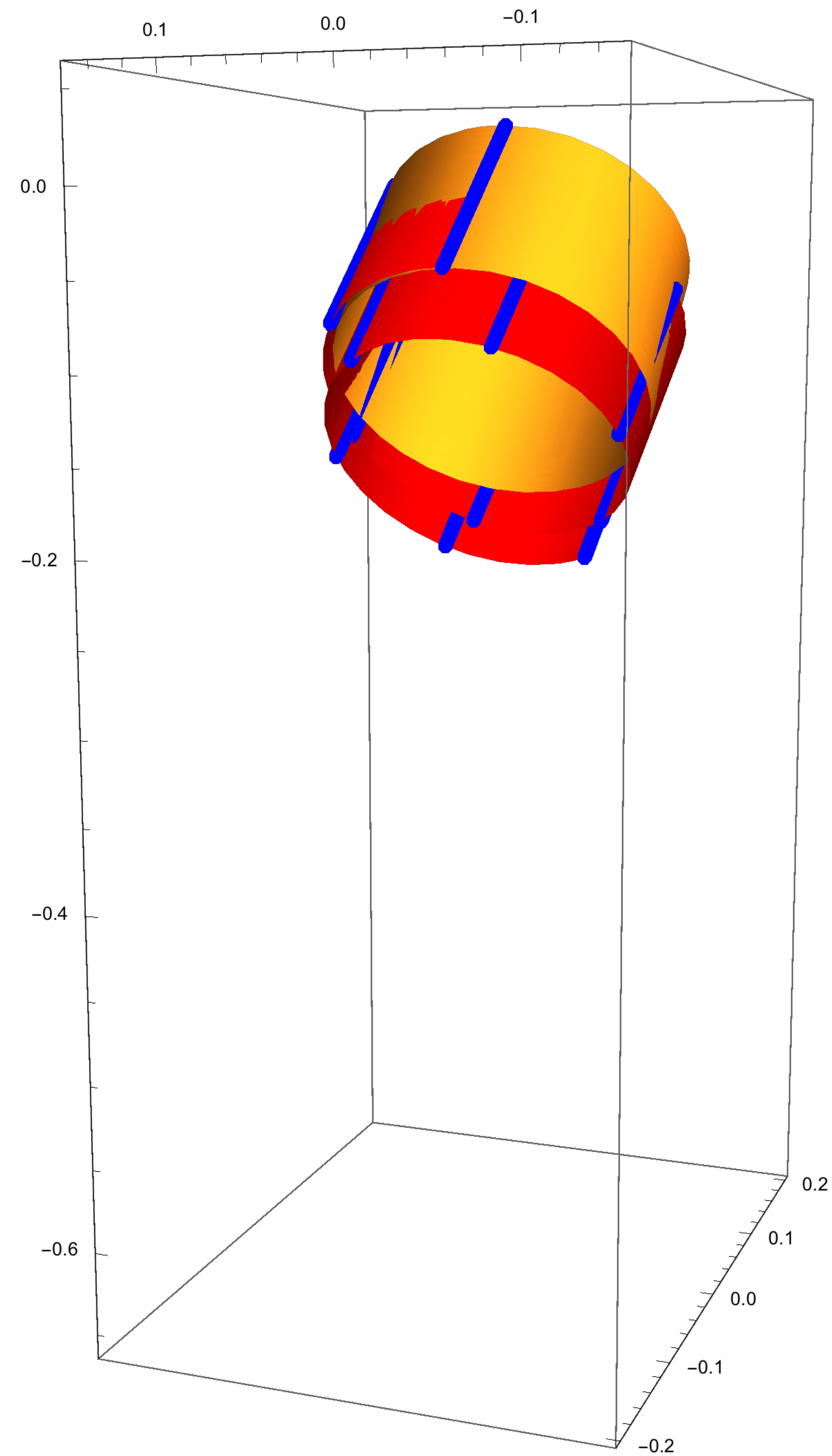
Shooting & AUTO:
sequence of equilibrium



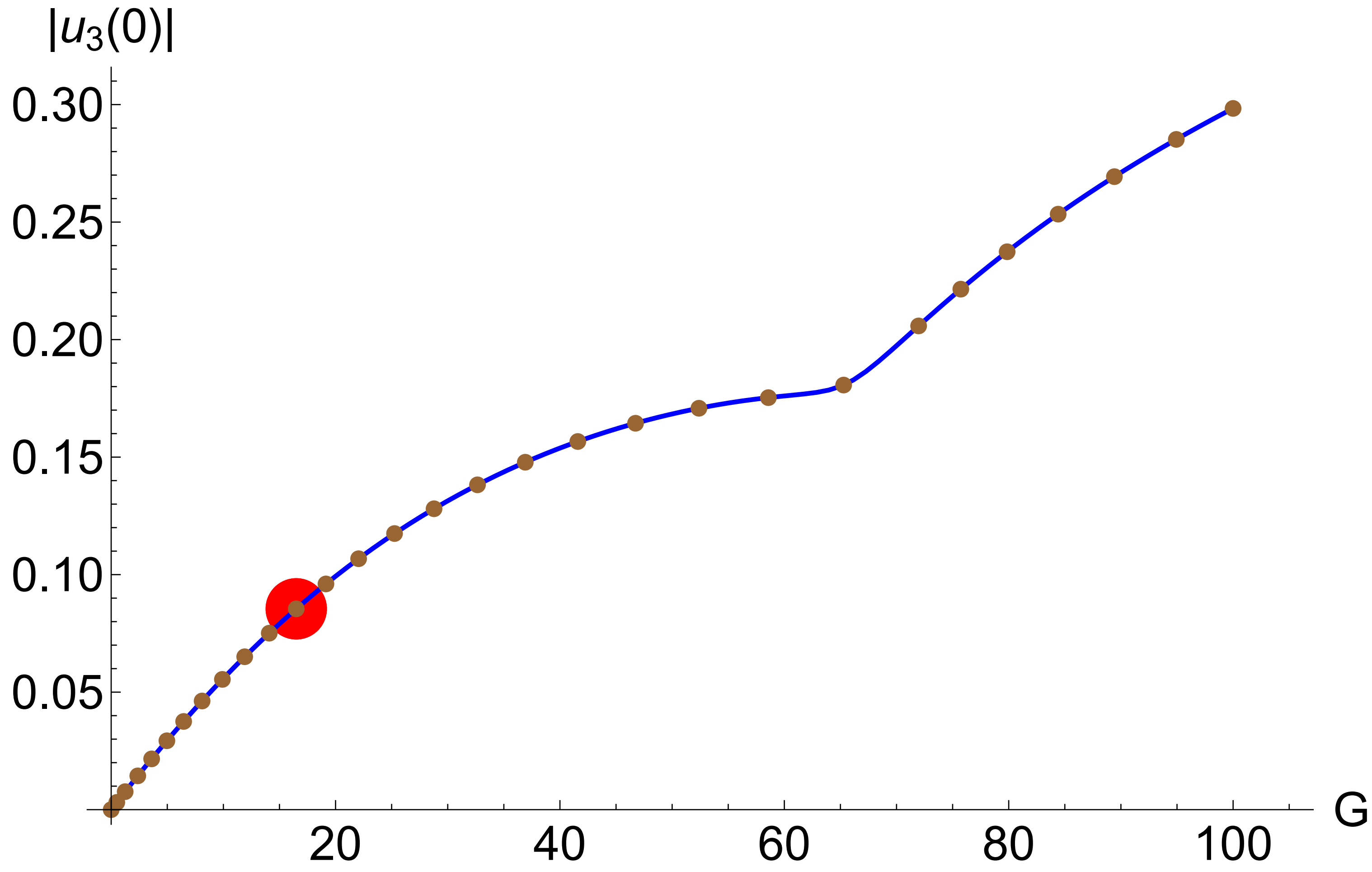


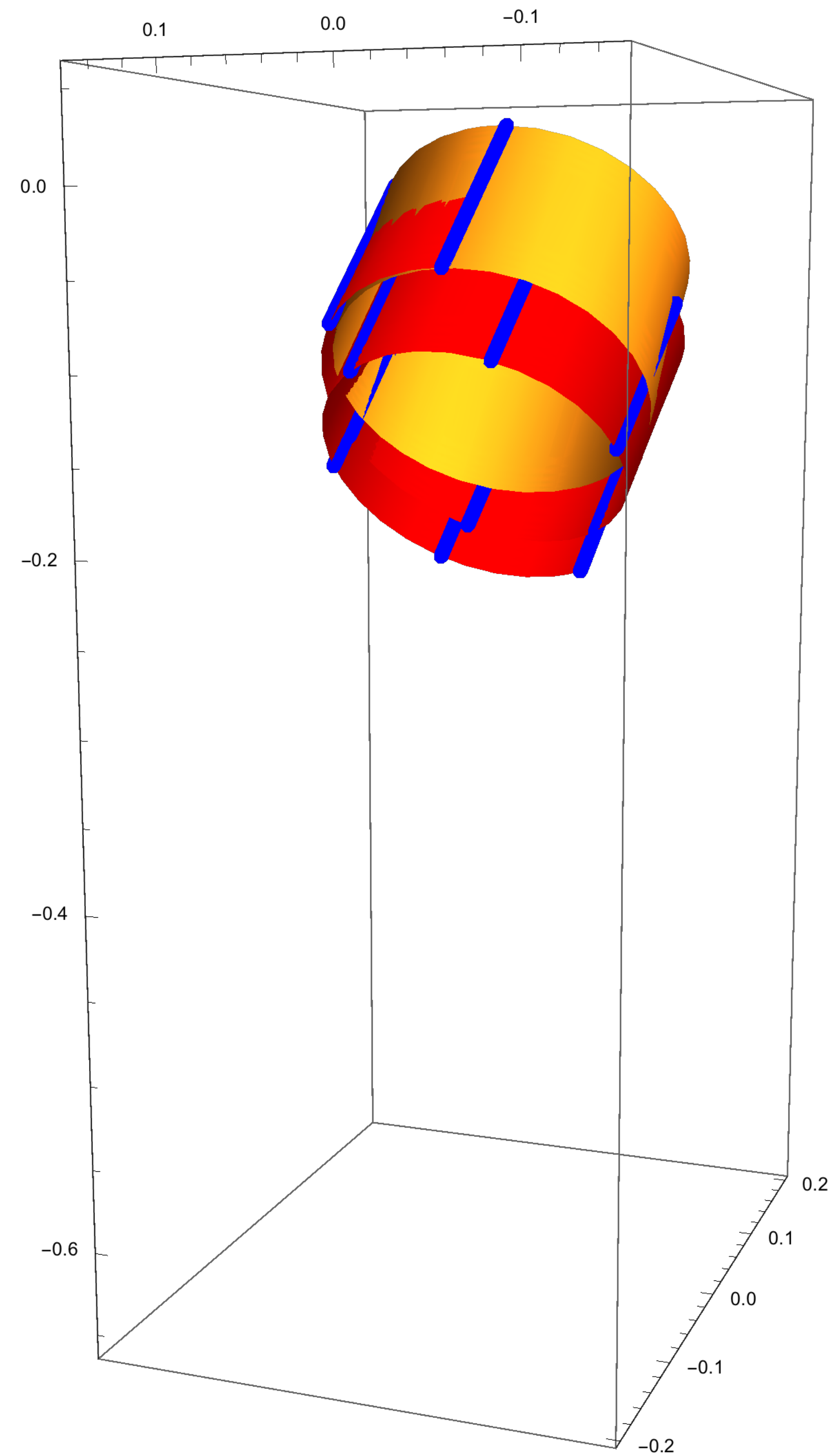
Shooting & AUTO:
sequence of equilibrium



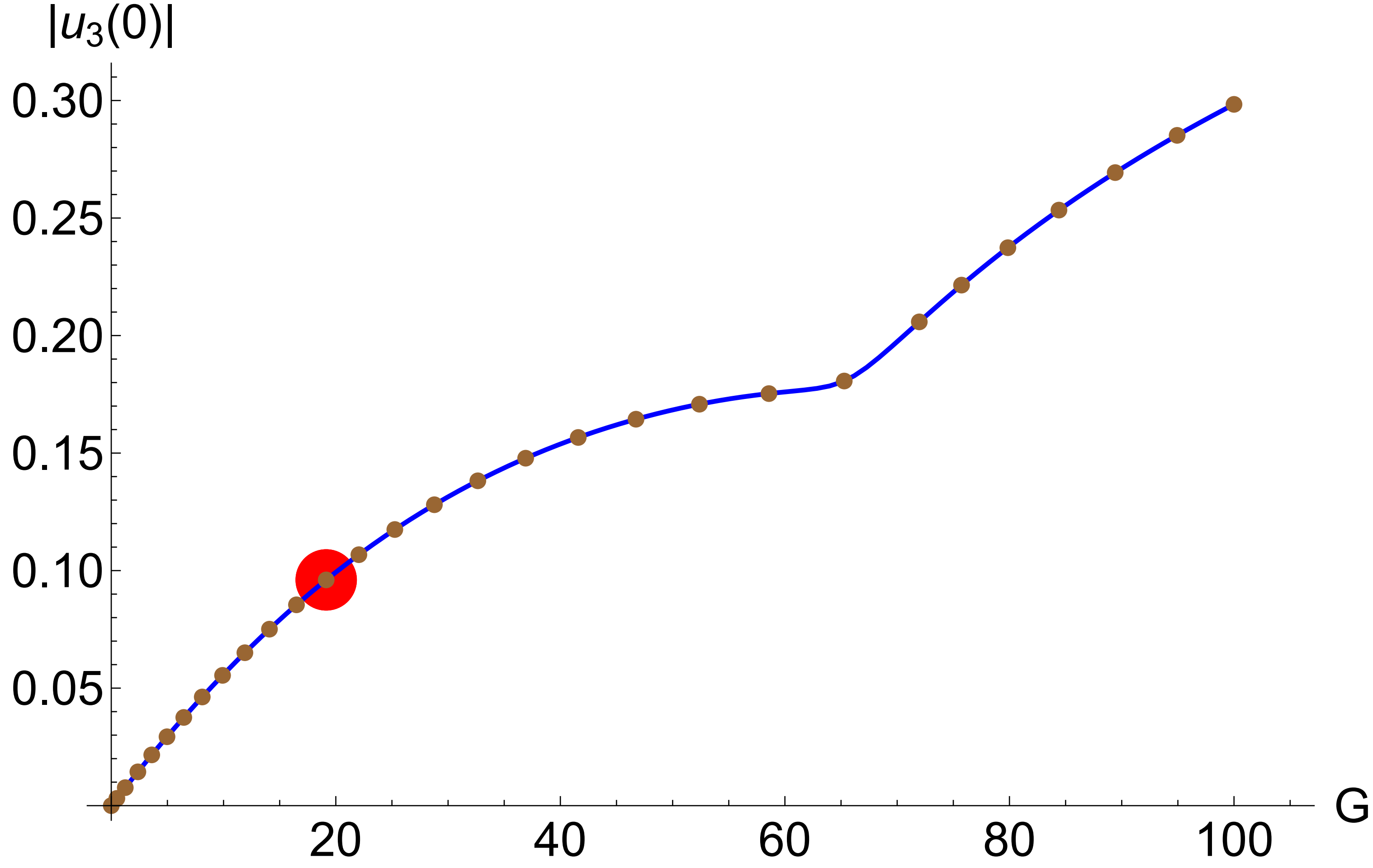


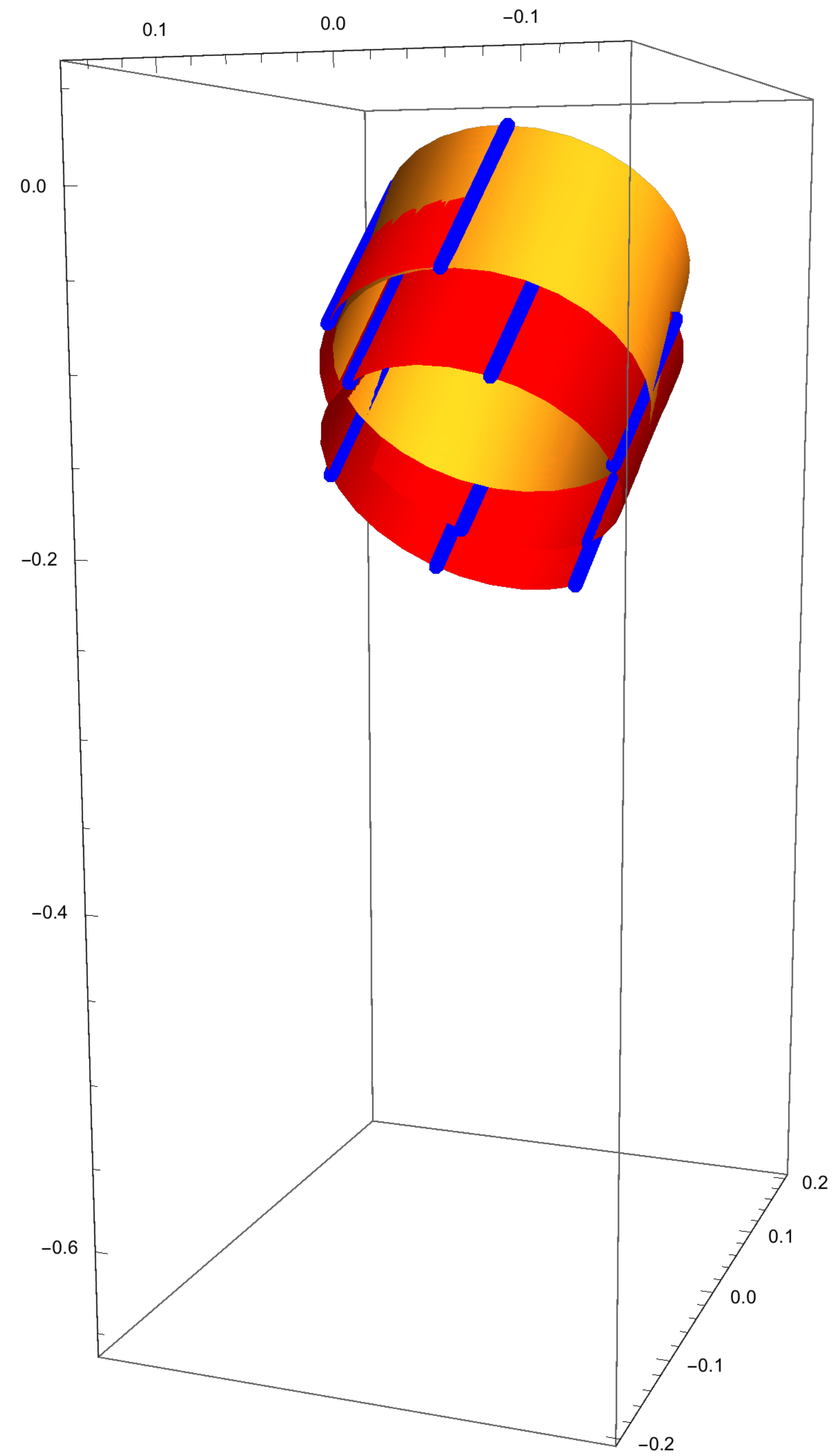
Shooting & AUTO: sequence of equilibrium



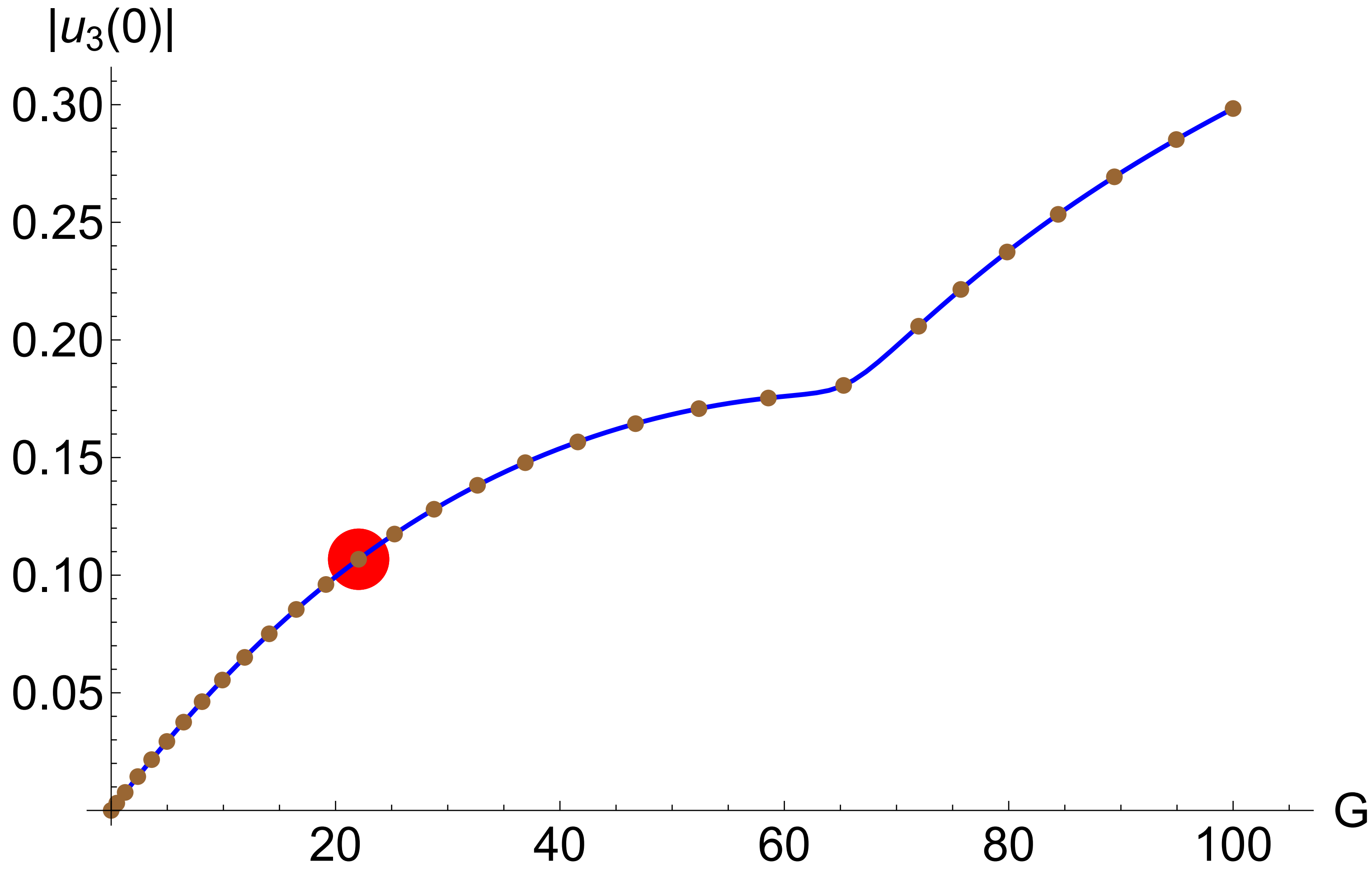


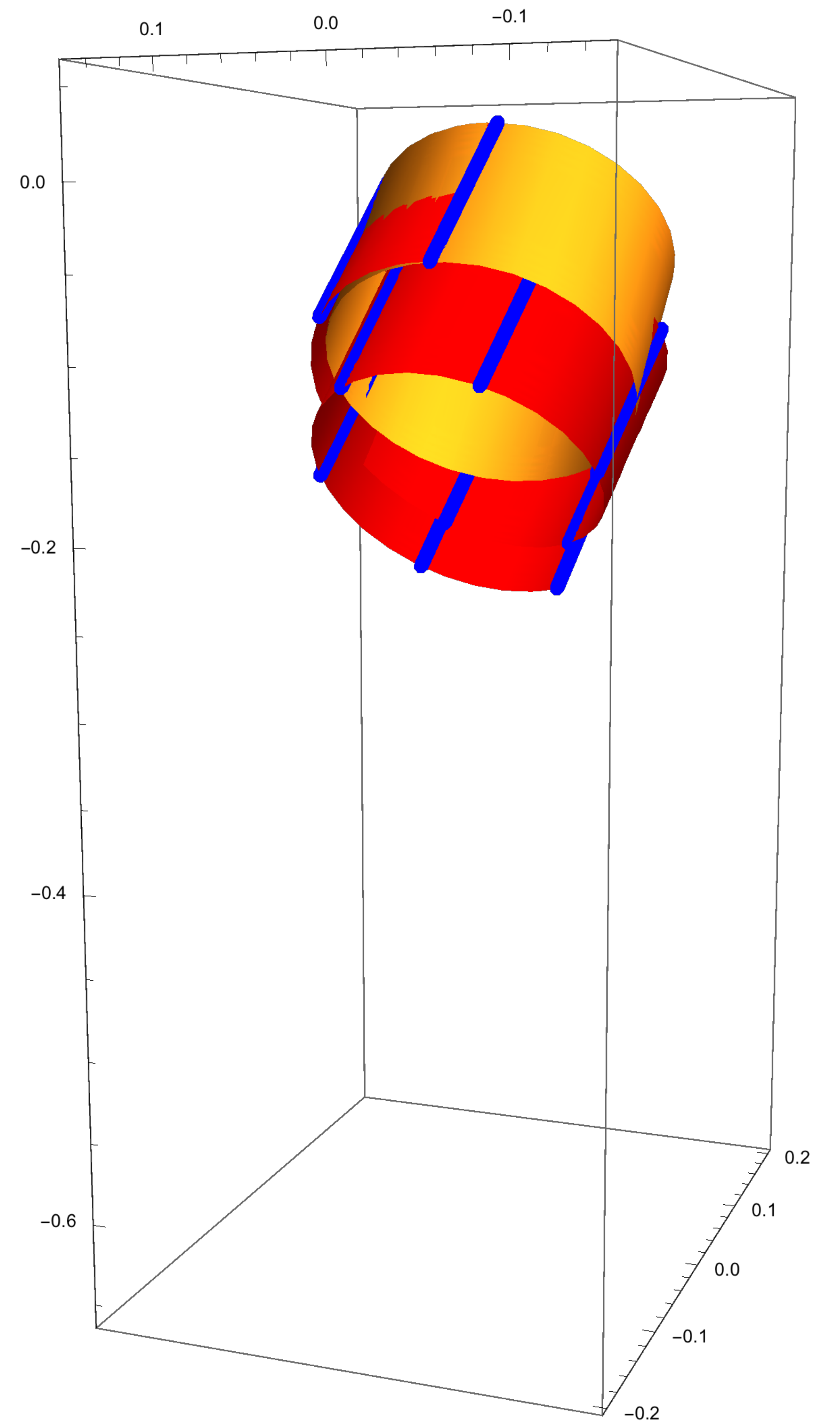
Shooting & AUTO: sequence of equilibrium



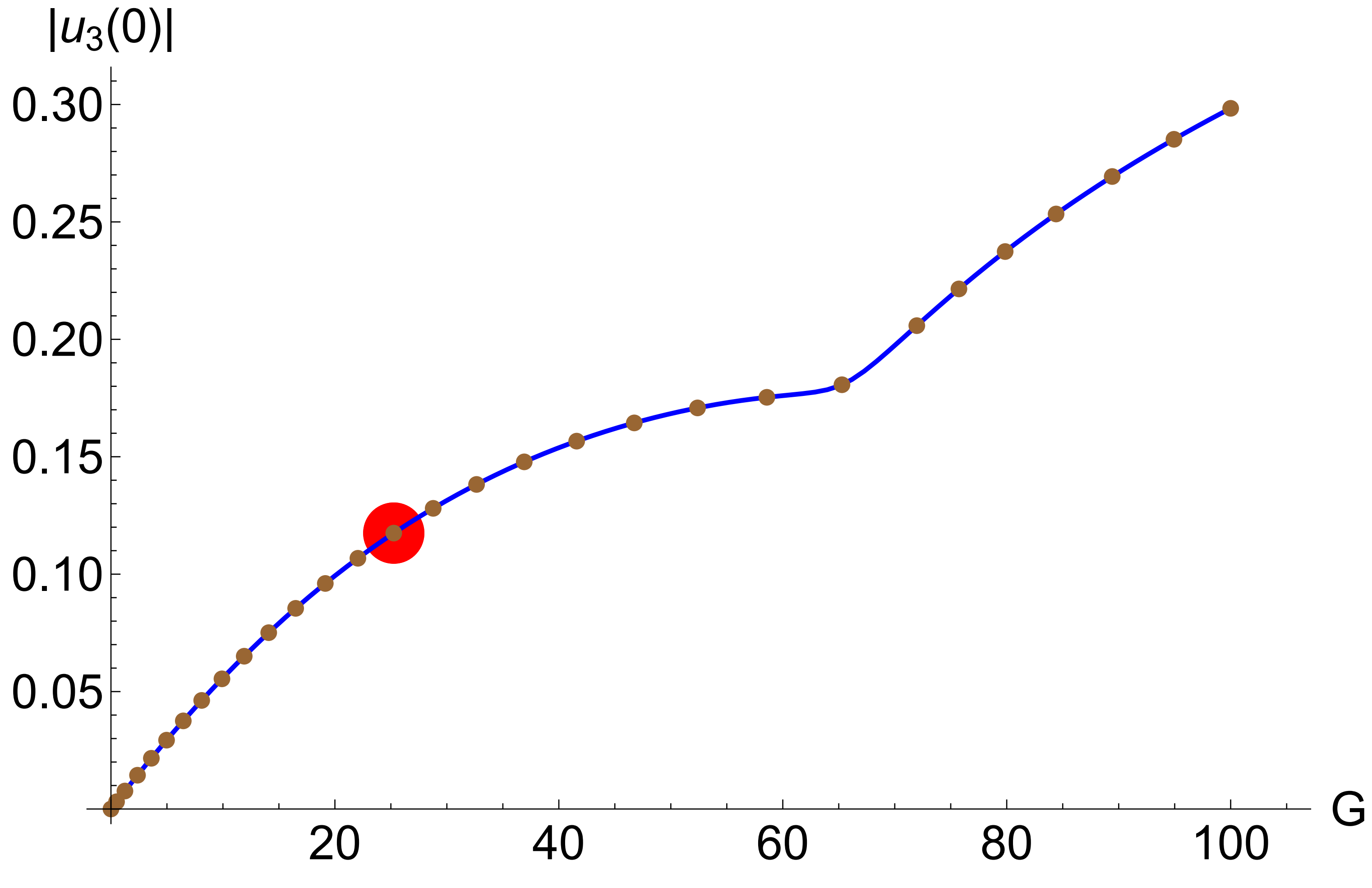


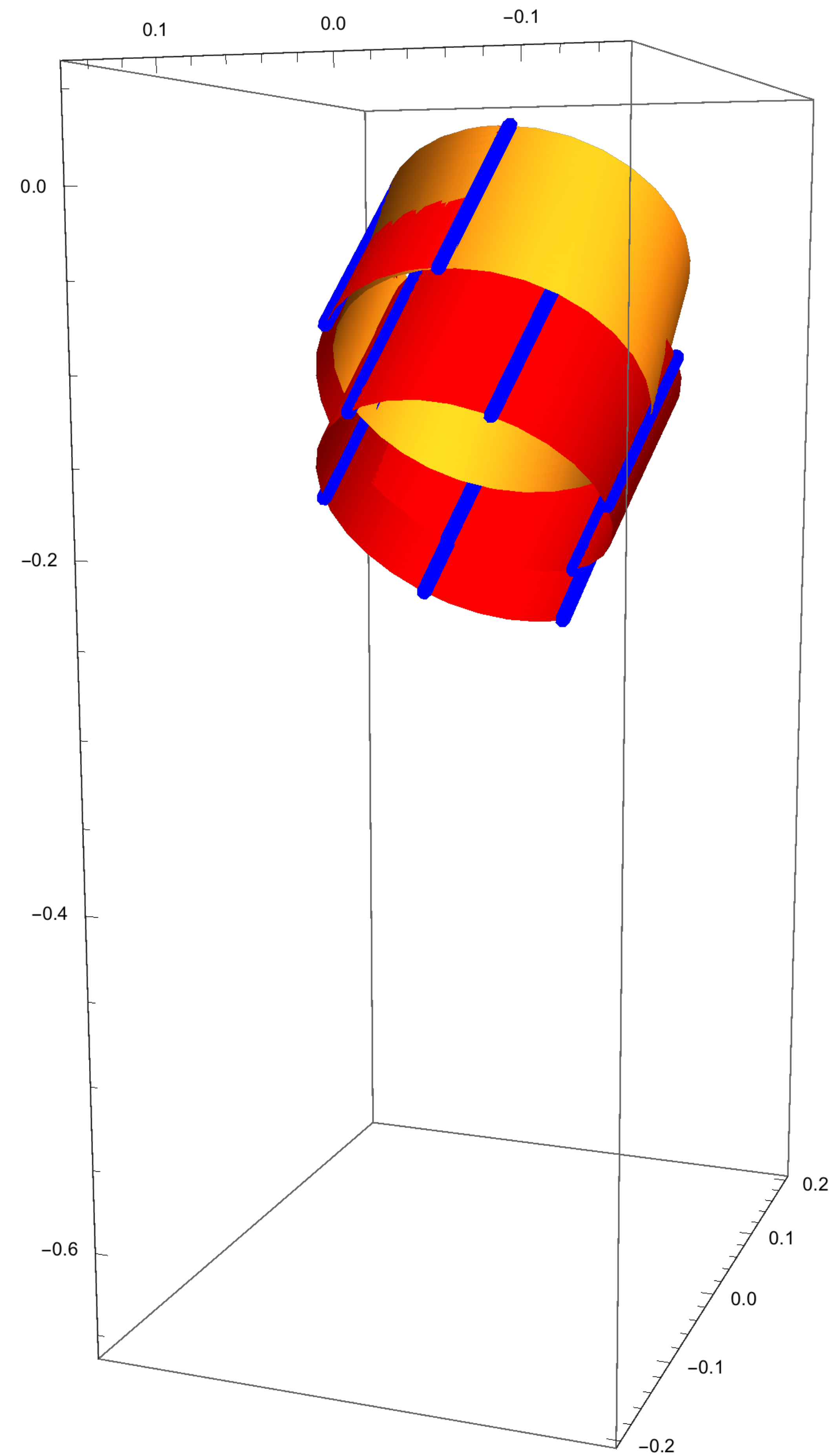
Shooting & AUTO:
sequence of equilibrium



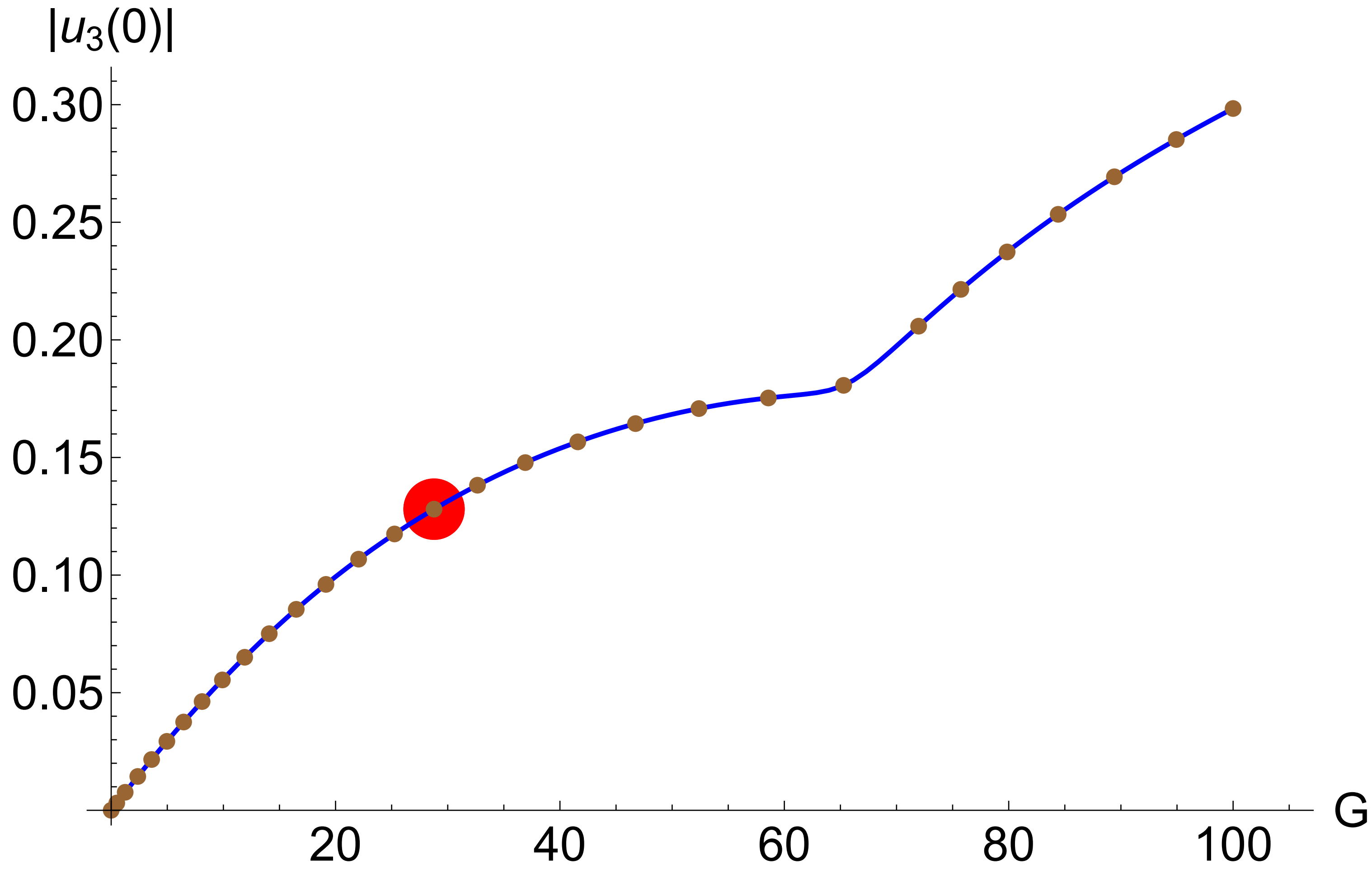


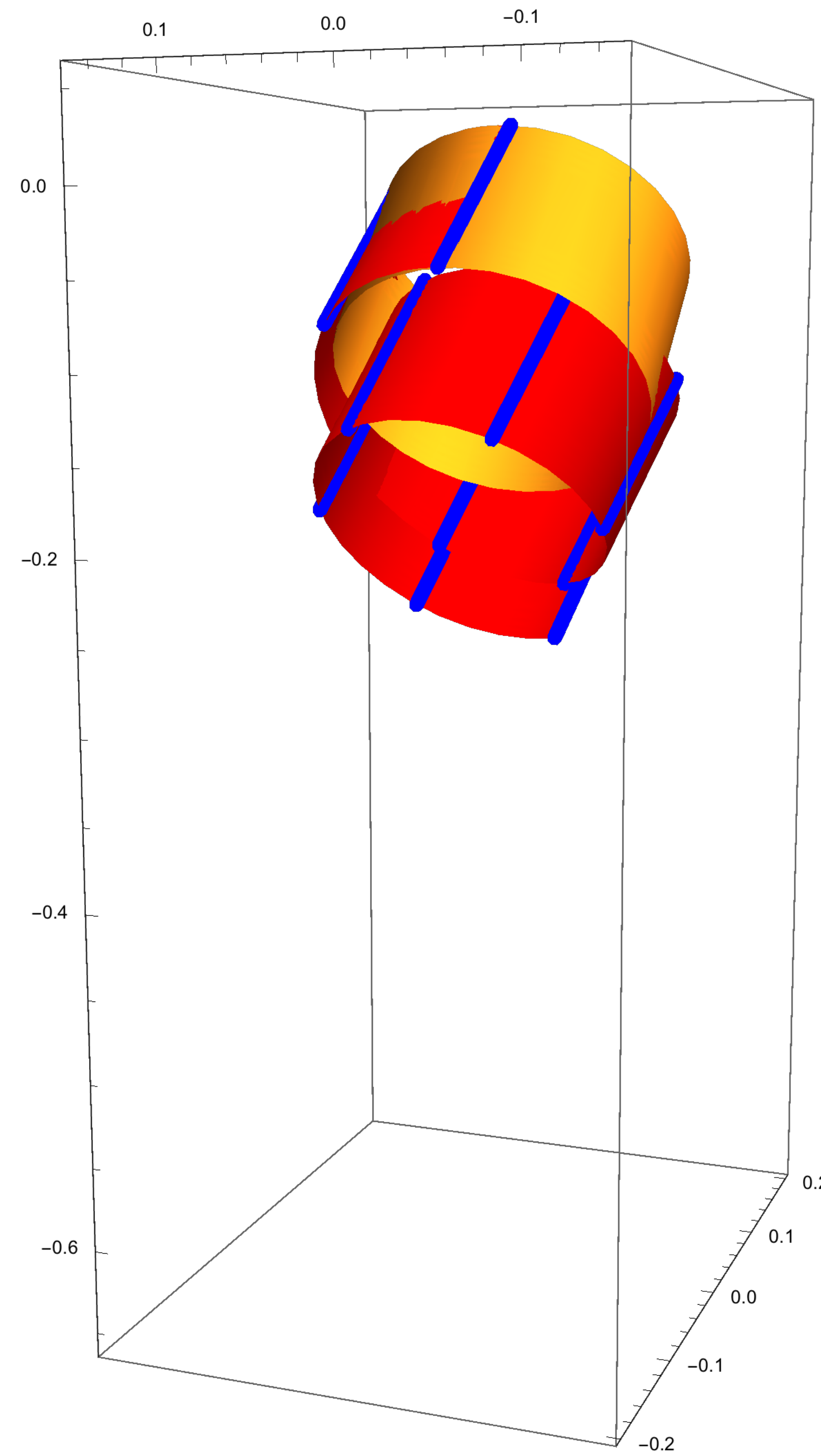
Shooting & AUTO: sequence of equilibrium



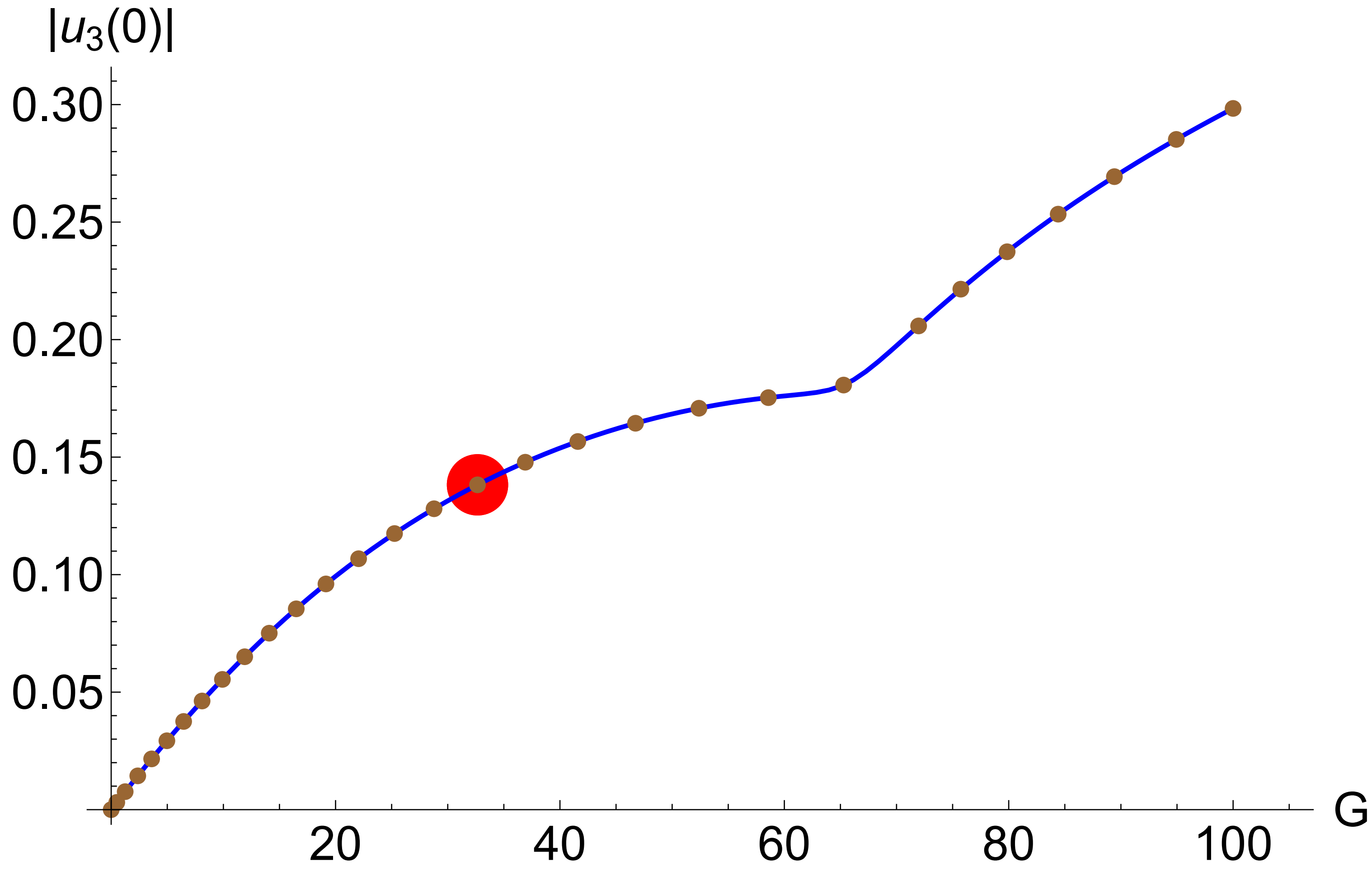


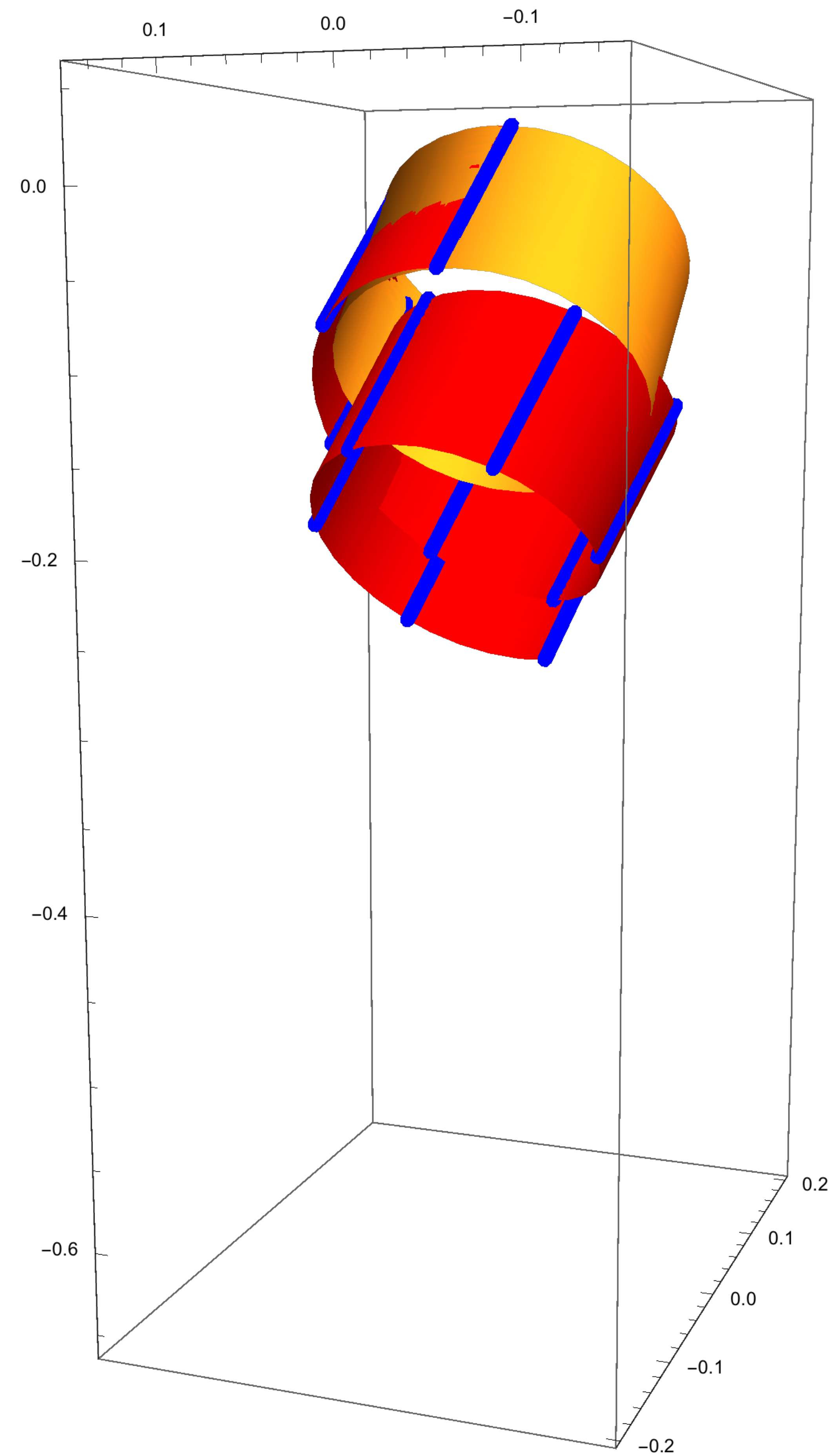
Shooting & AUTO:
sequence of equilibrium



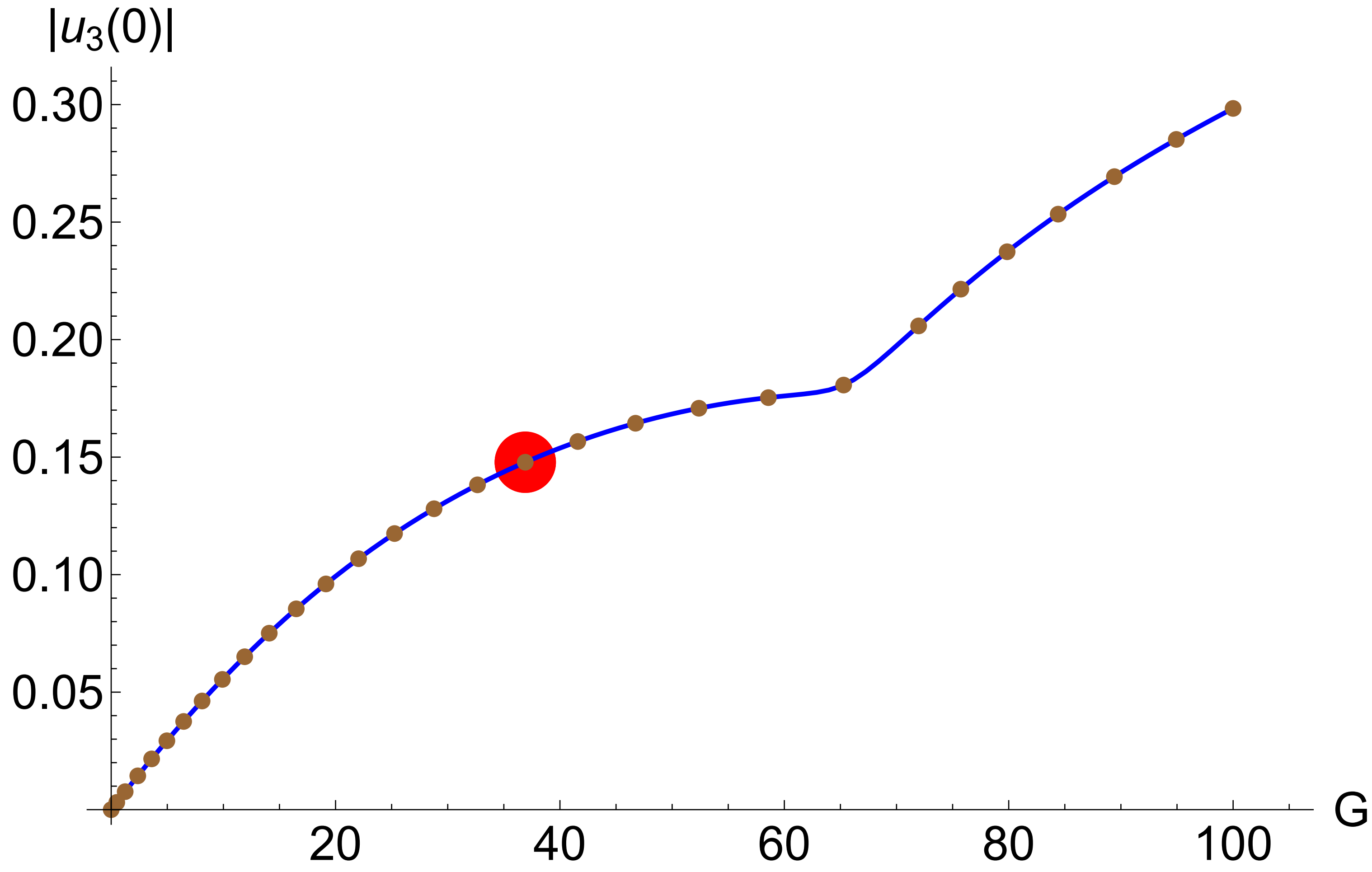


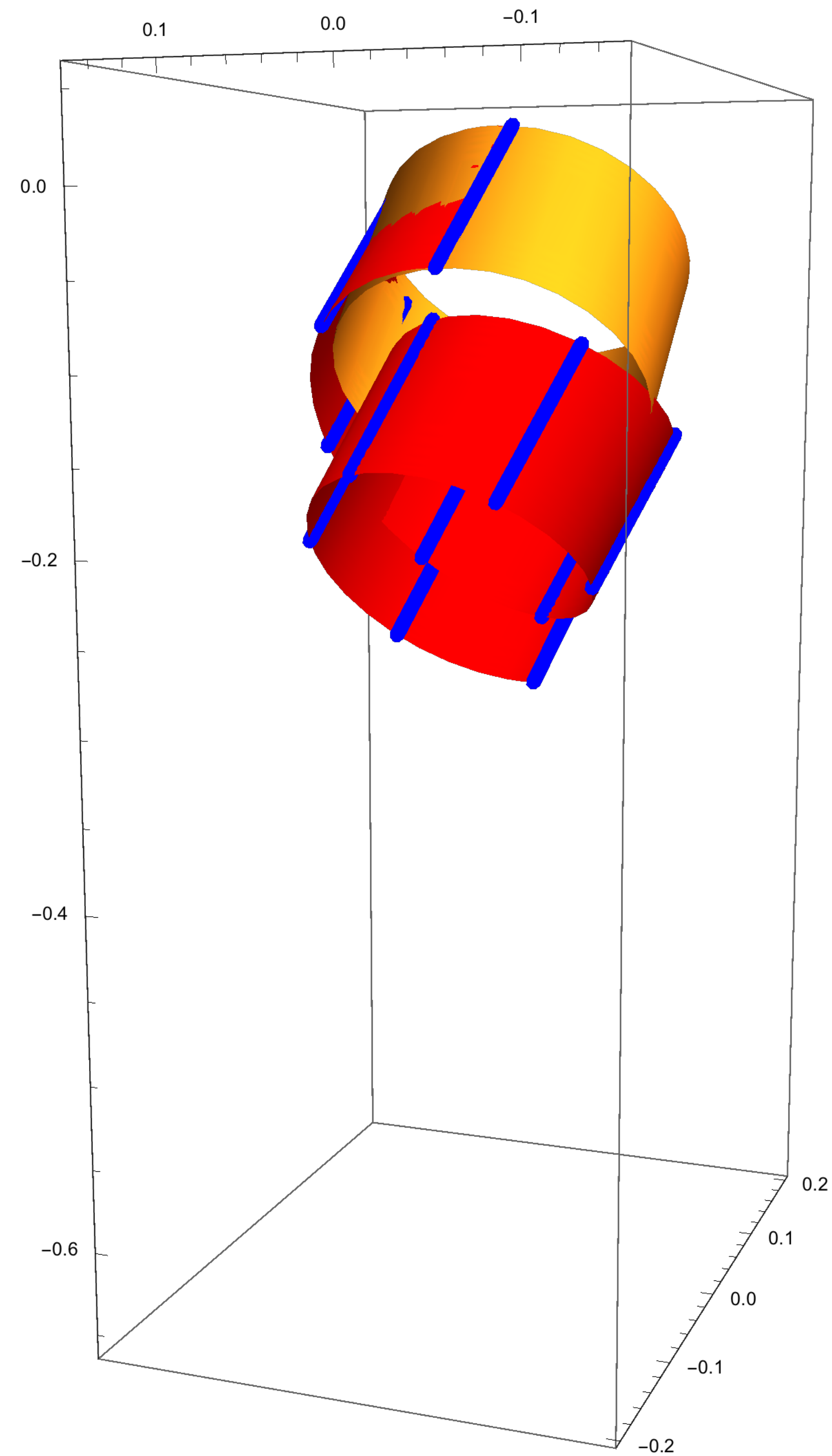
Shooting & AUTO:
sequence of equilibrium



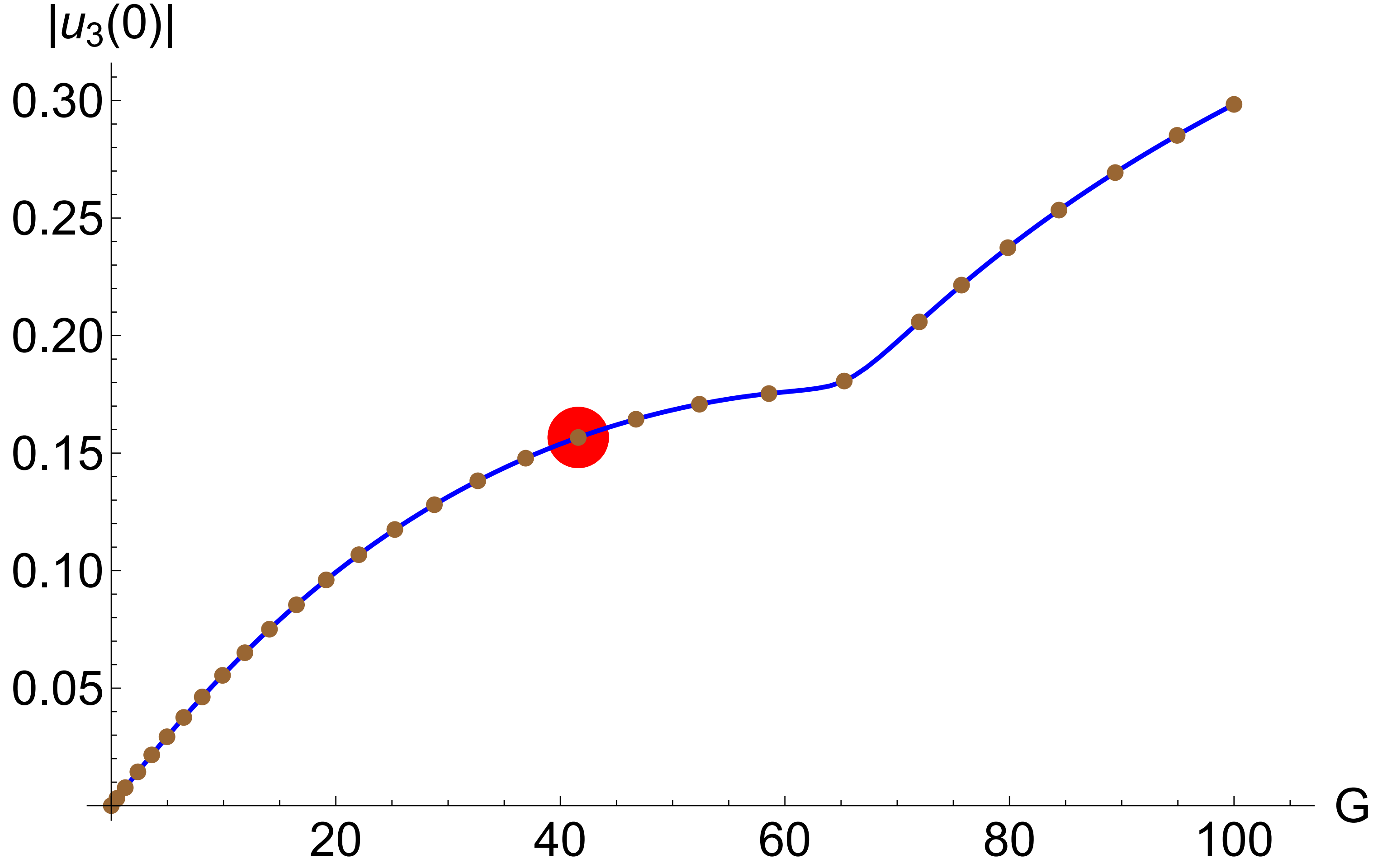


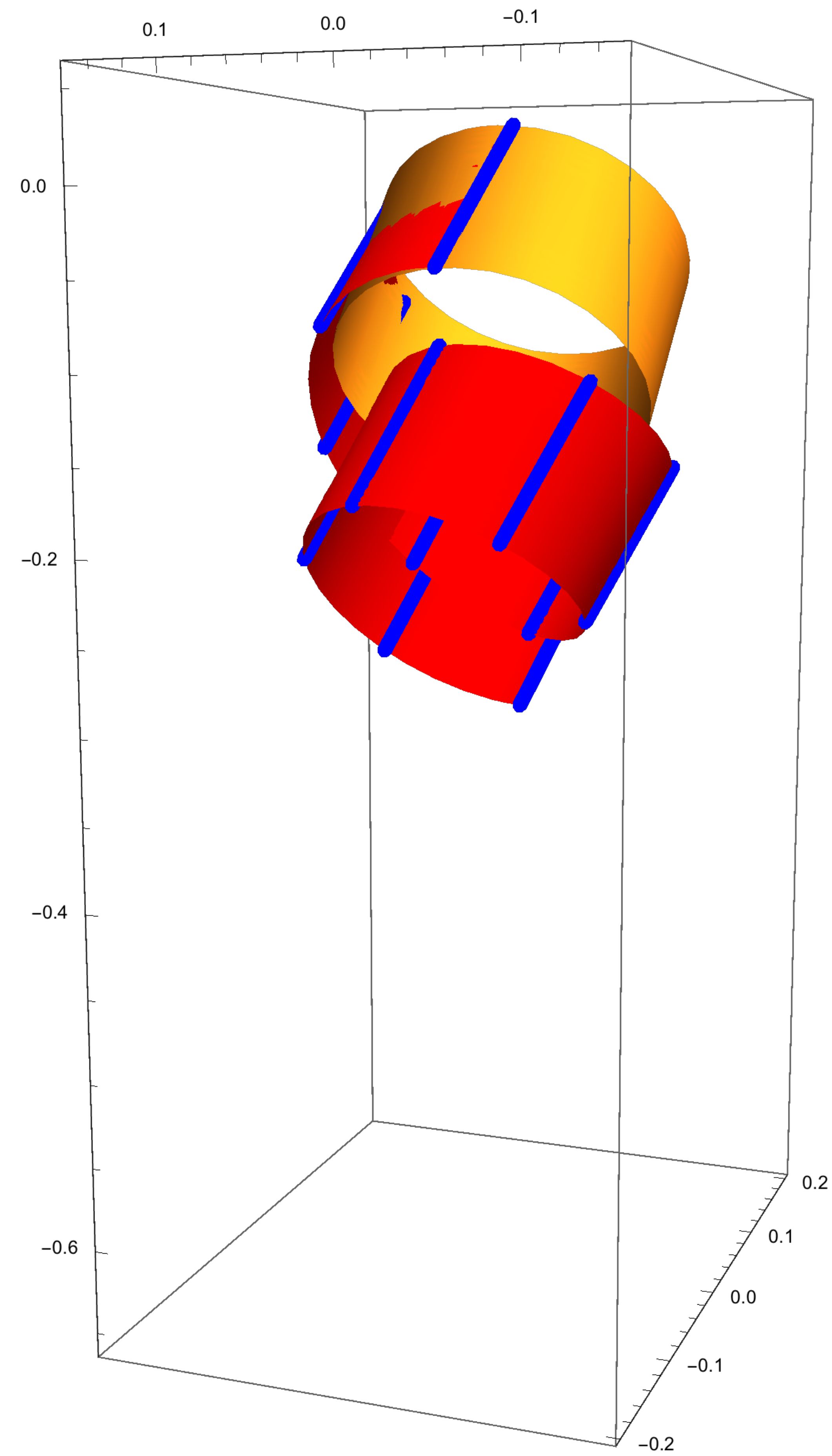
Shooting & AUTO:
sequence of equilibrium



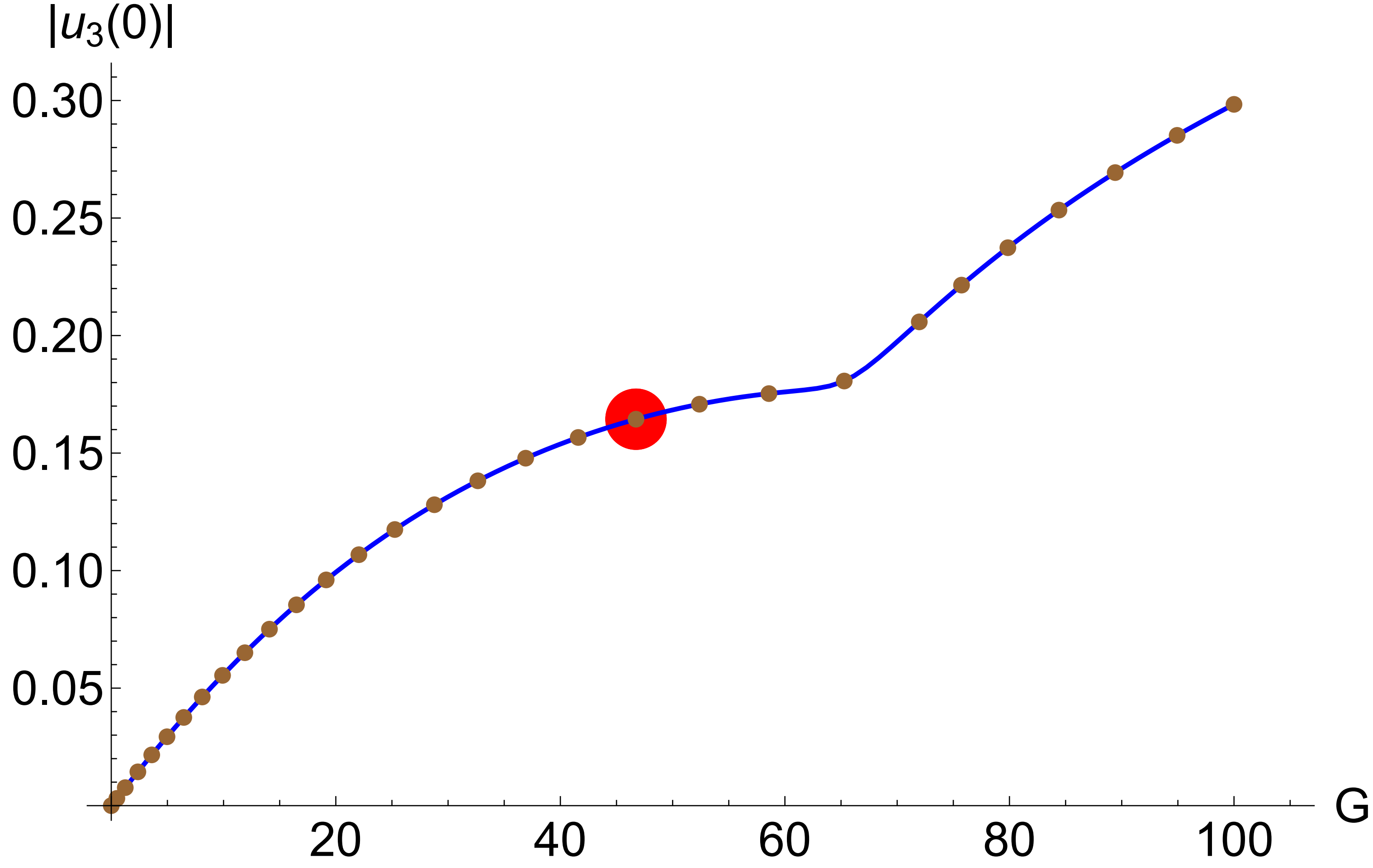


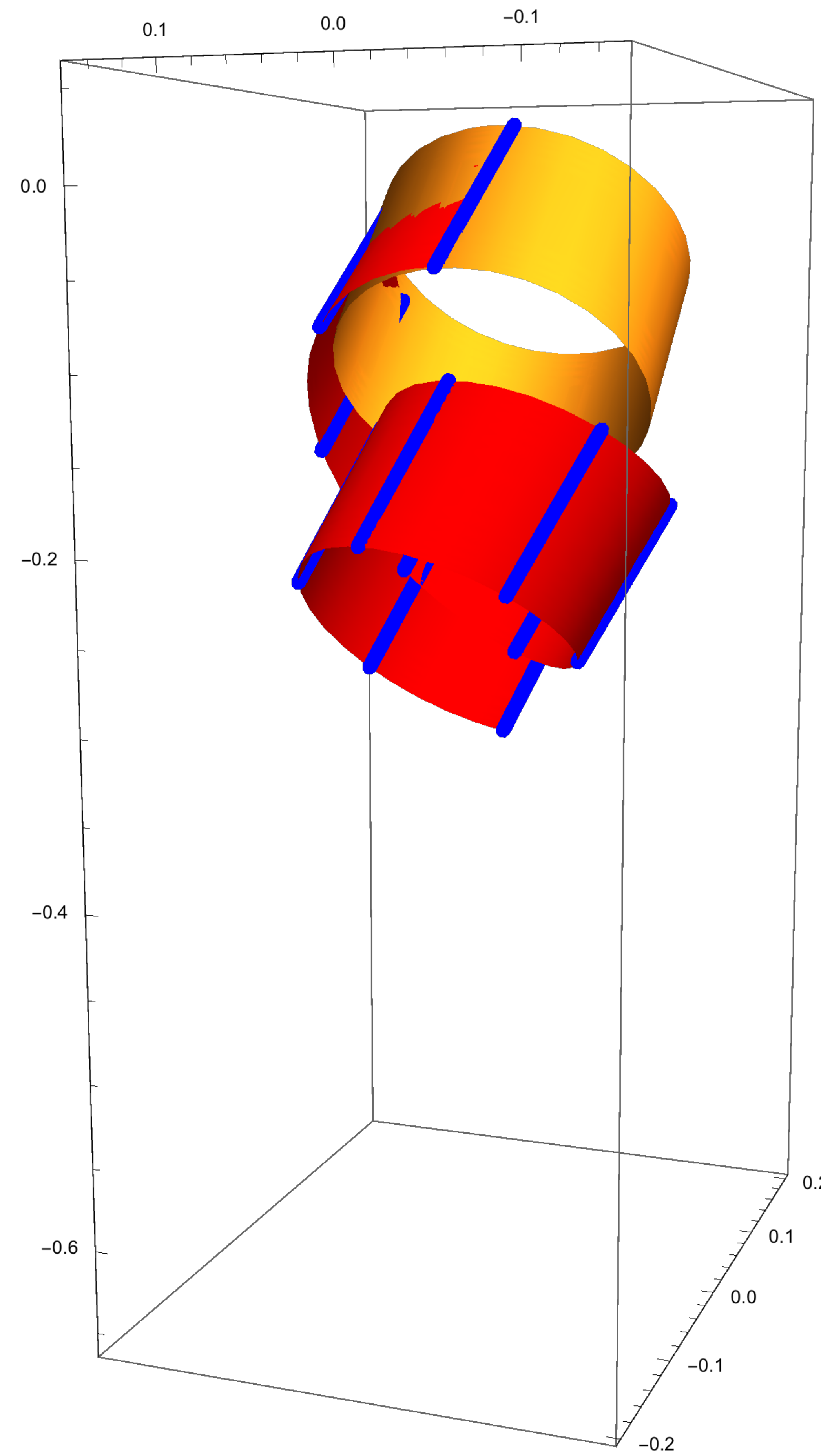
Shooting & AUTO:
sequence of equilibrium



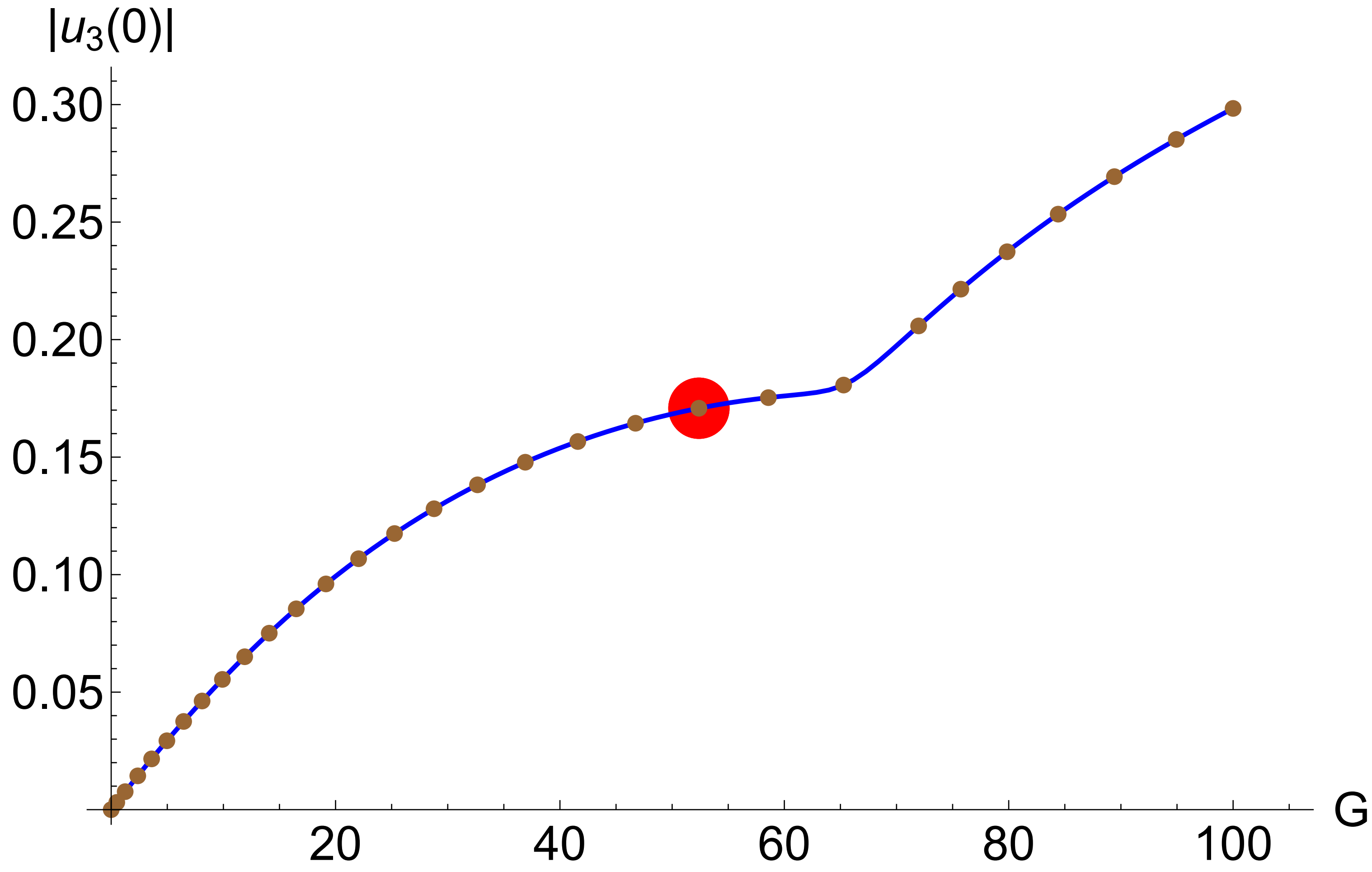


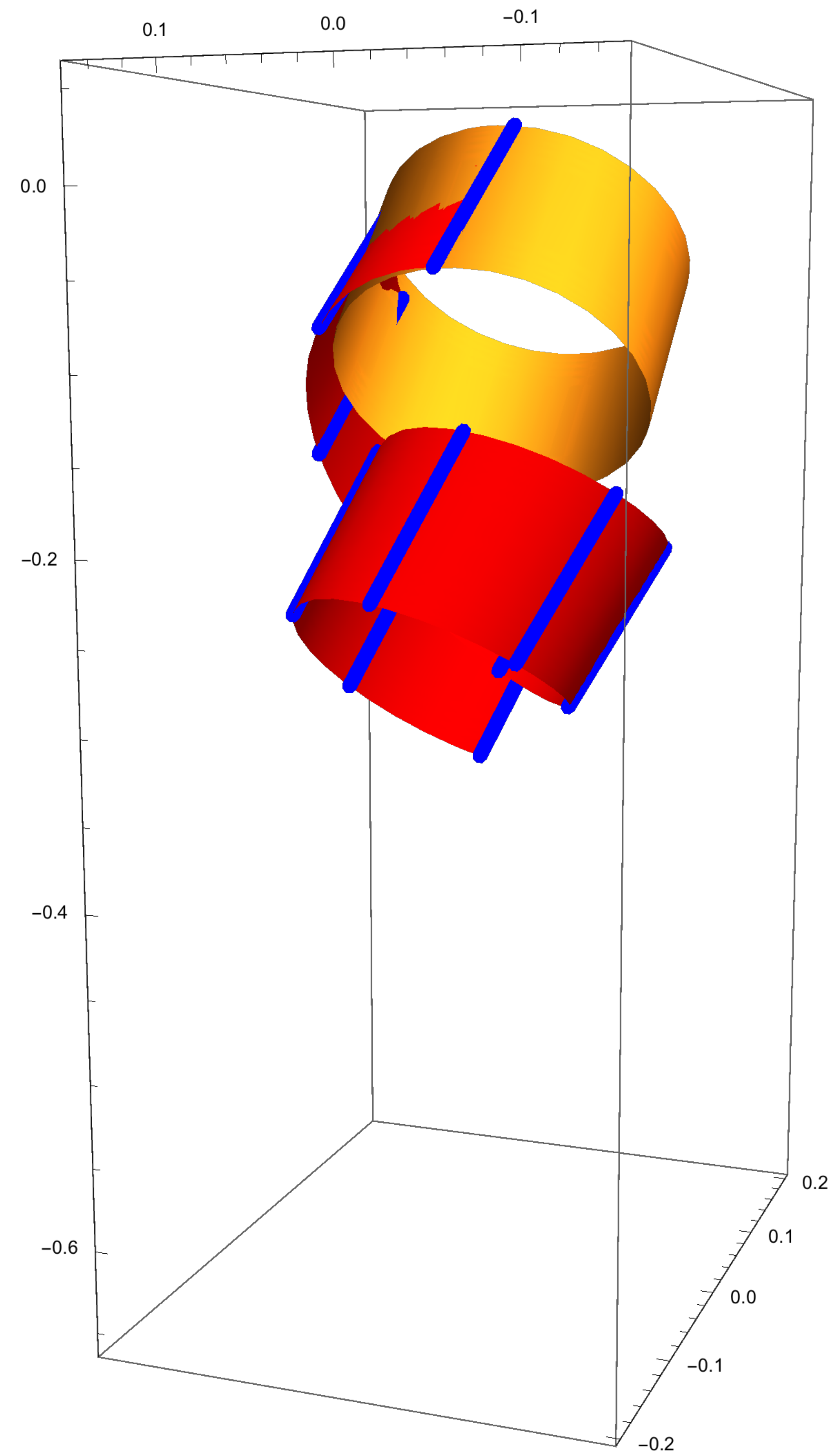
Shooting & AUTO: sequence of equilibrium



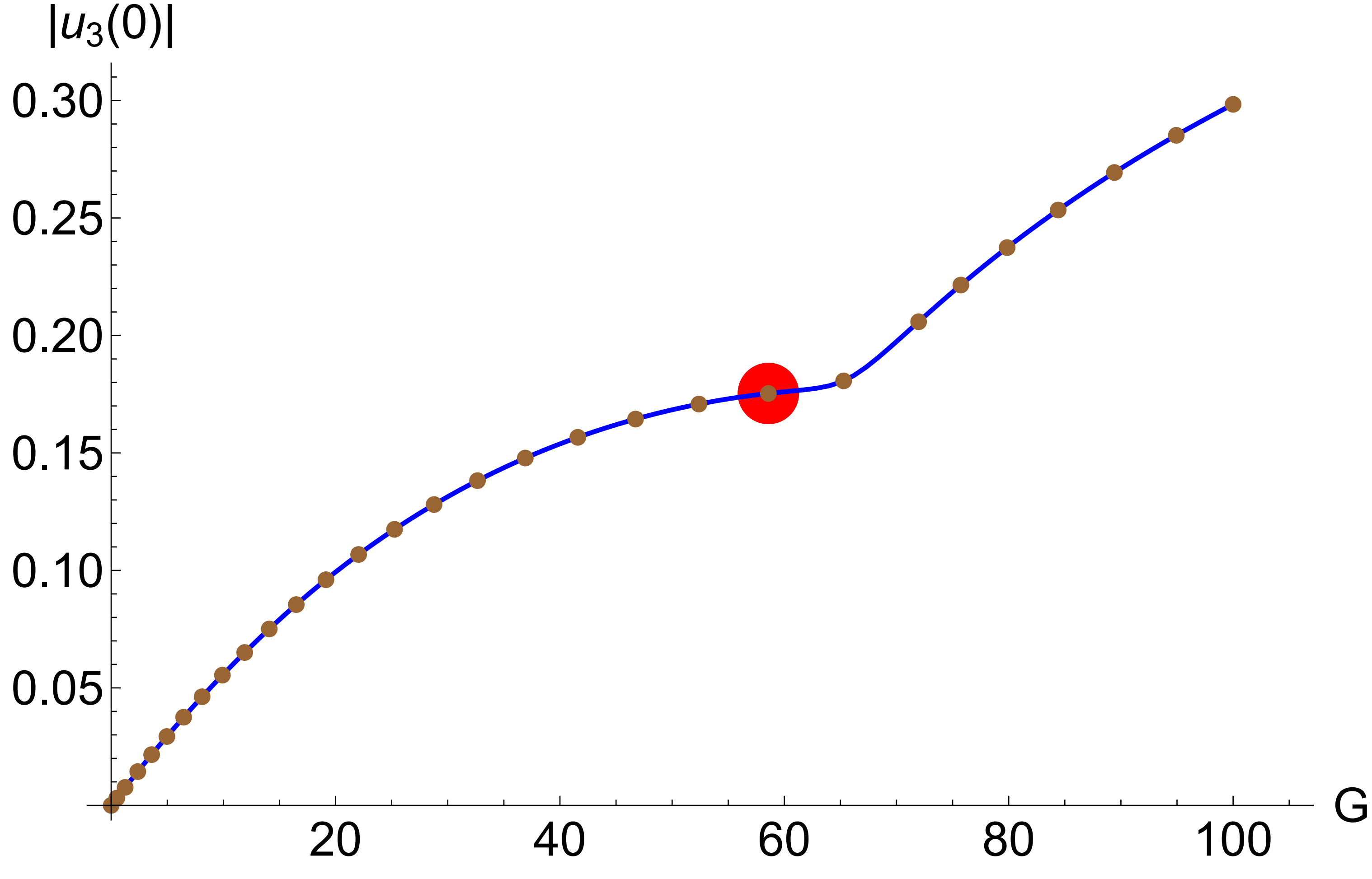


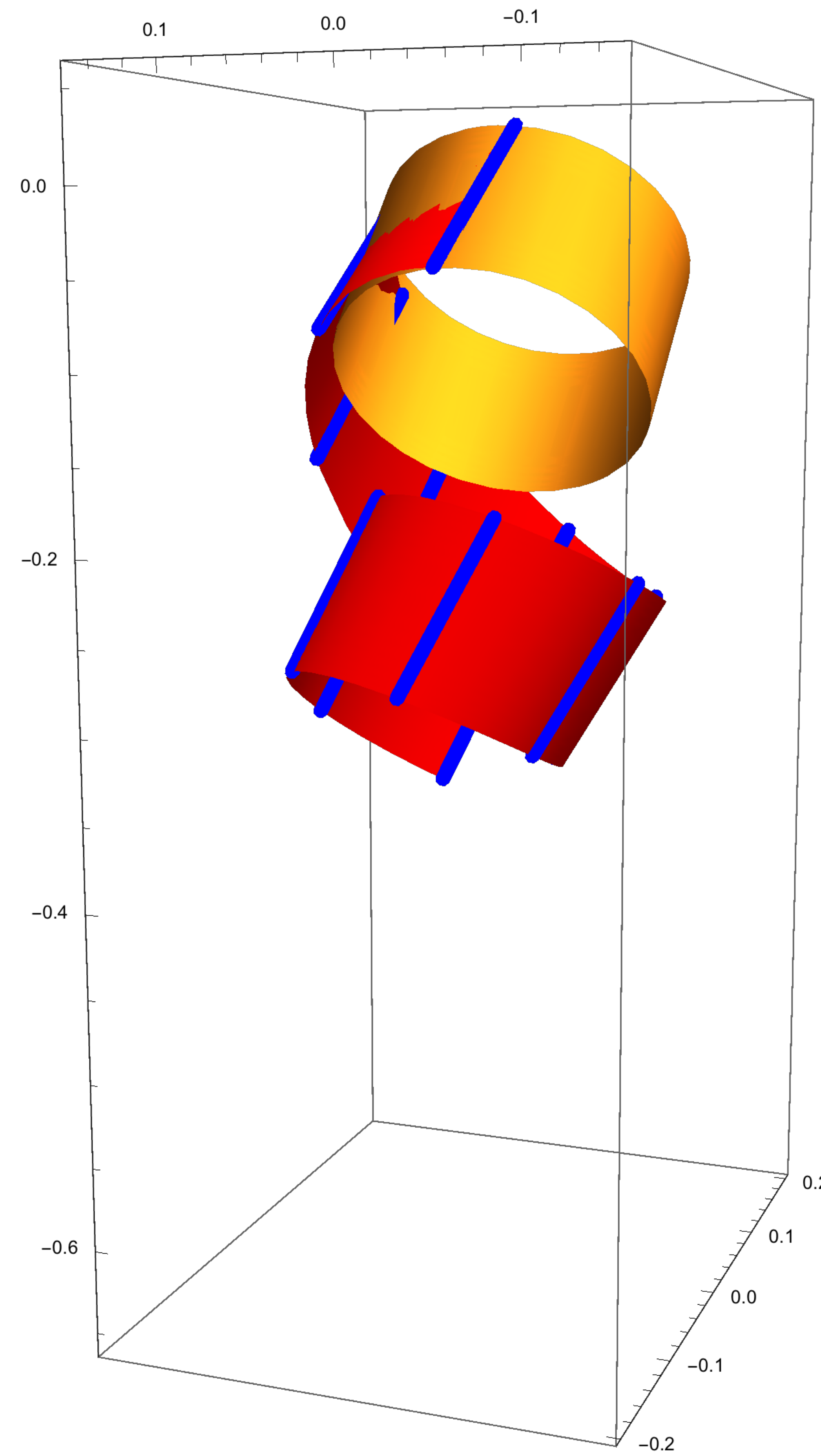
Shooting & AUTO: sequence of equilibrium



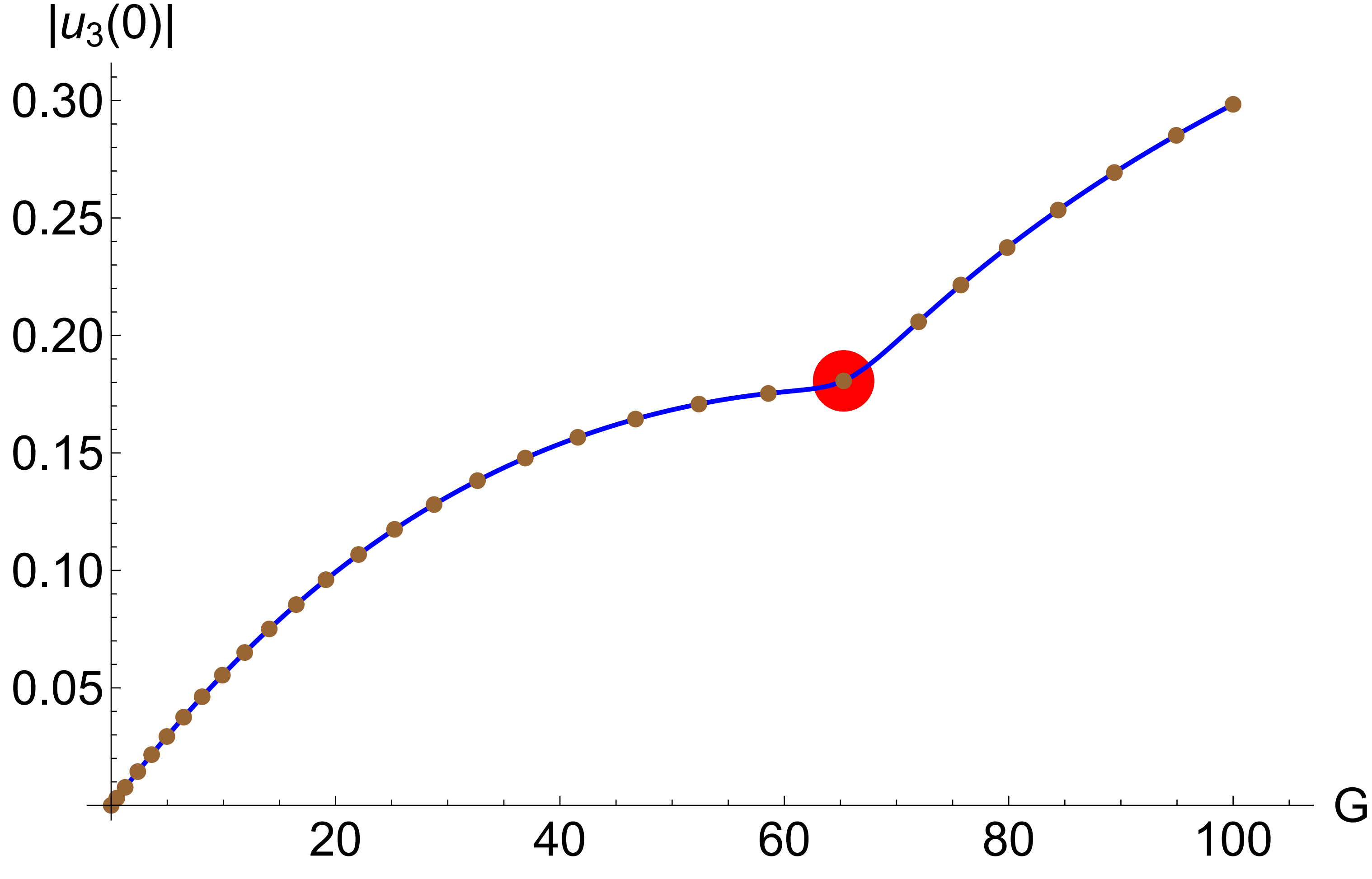


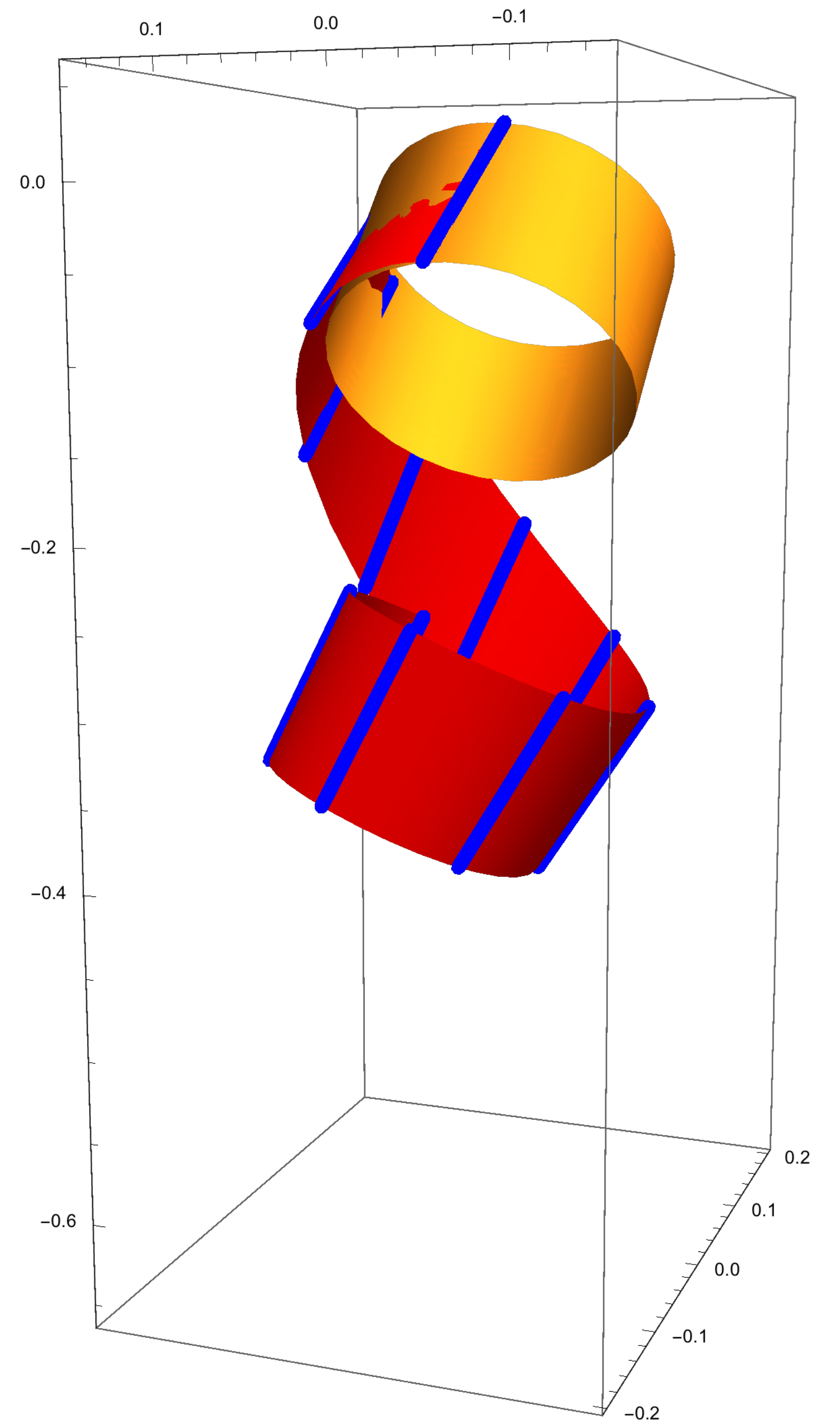
Shooting & AUTO:
sequence of equilibrium



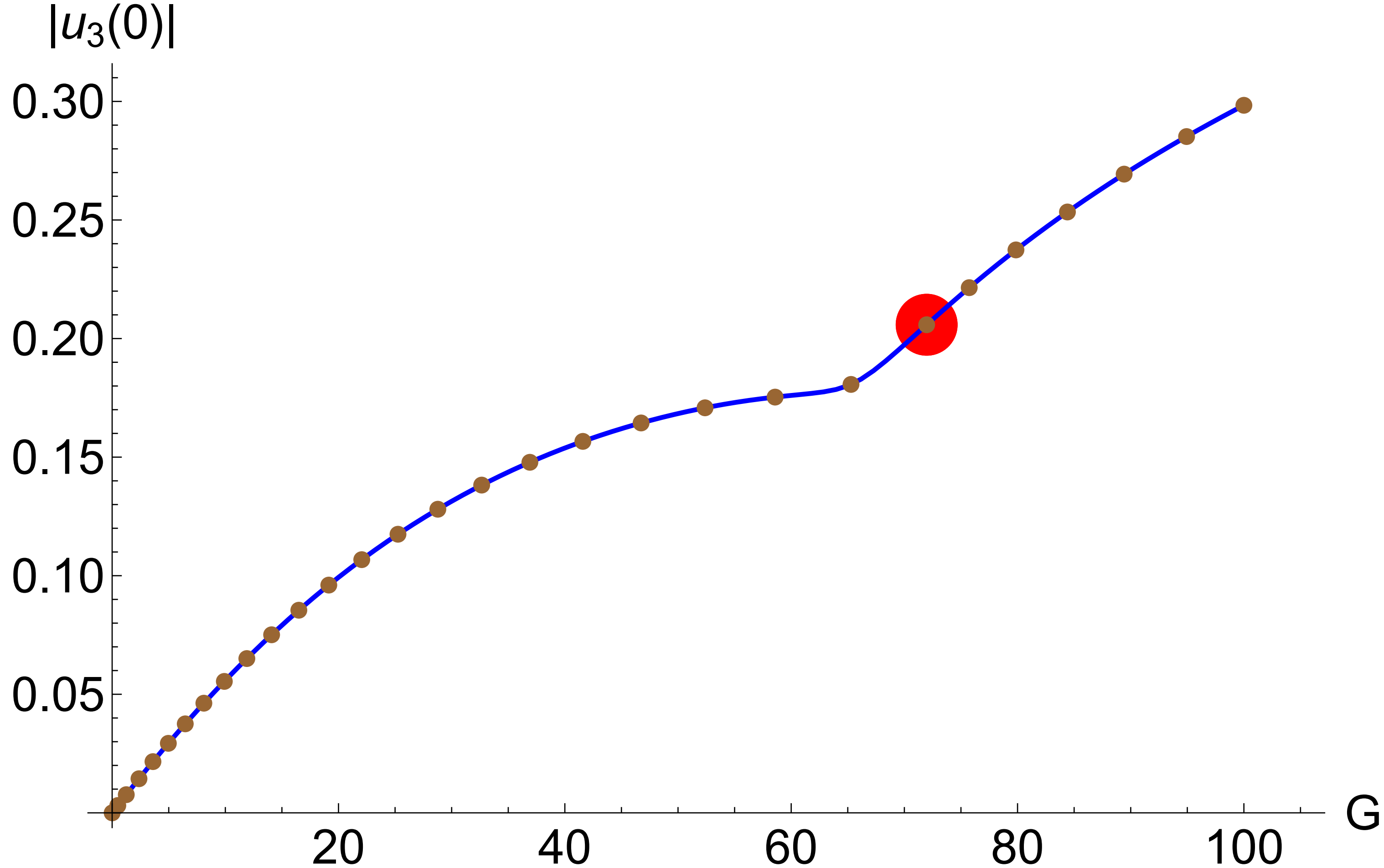


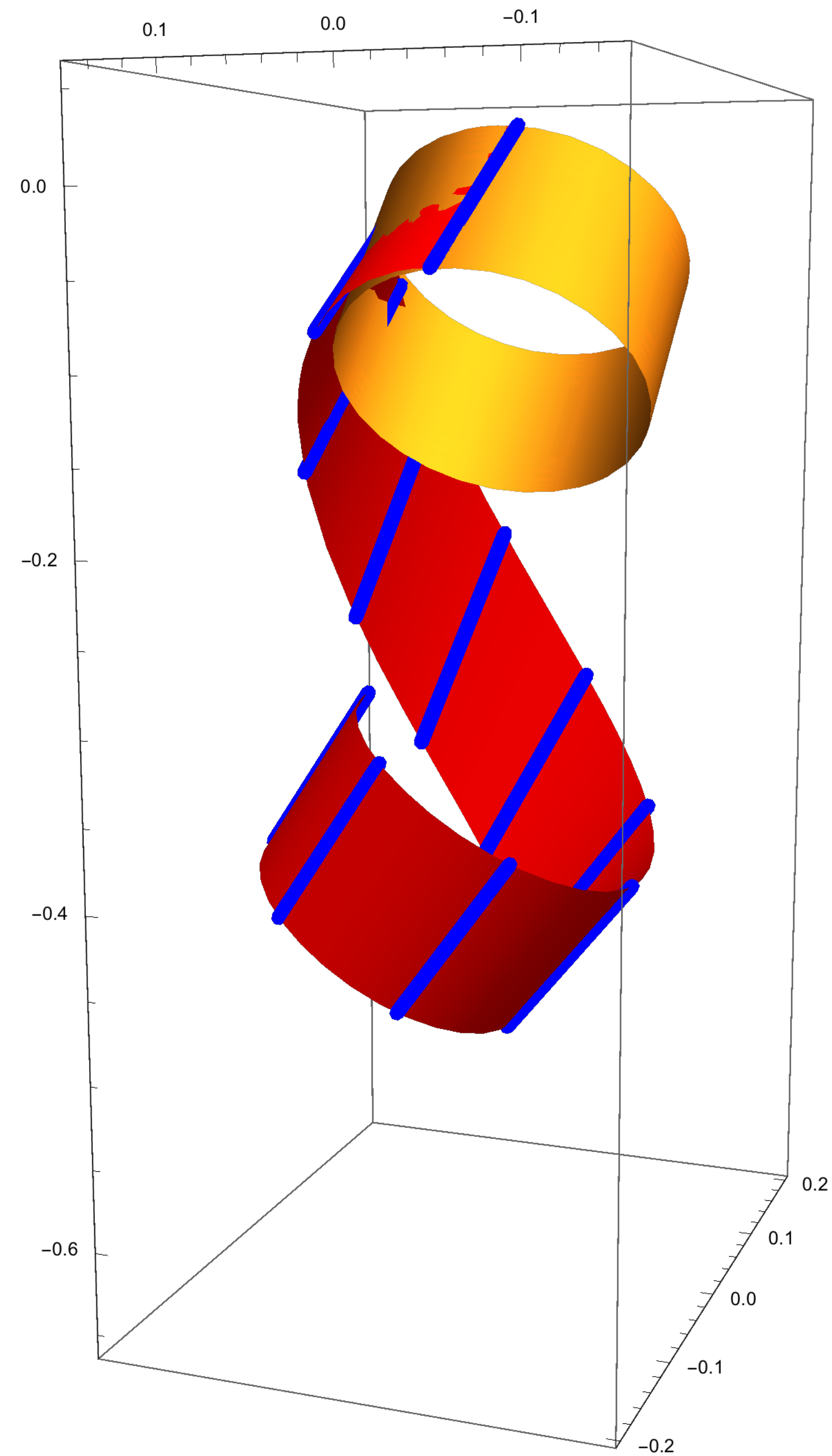
Shooting & AUTO:
sequence of equilibrium



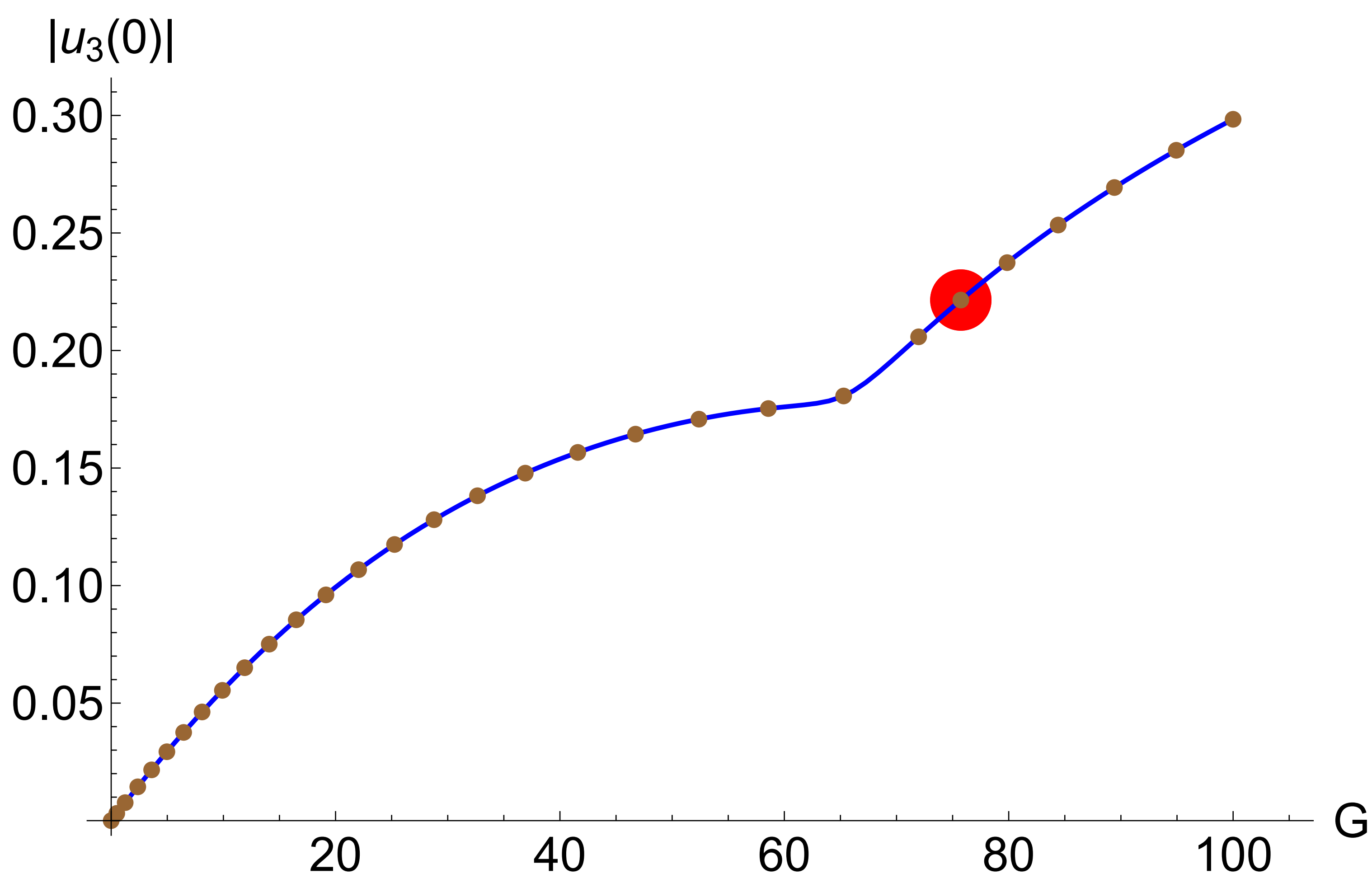


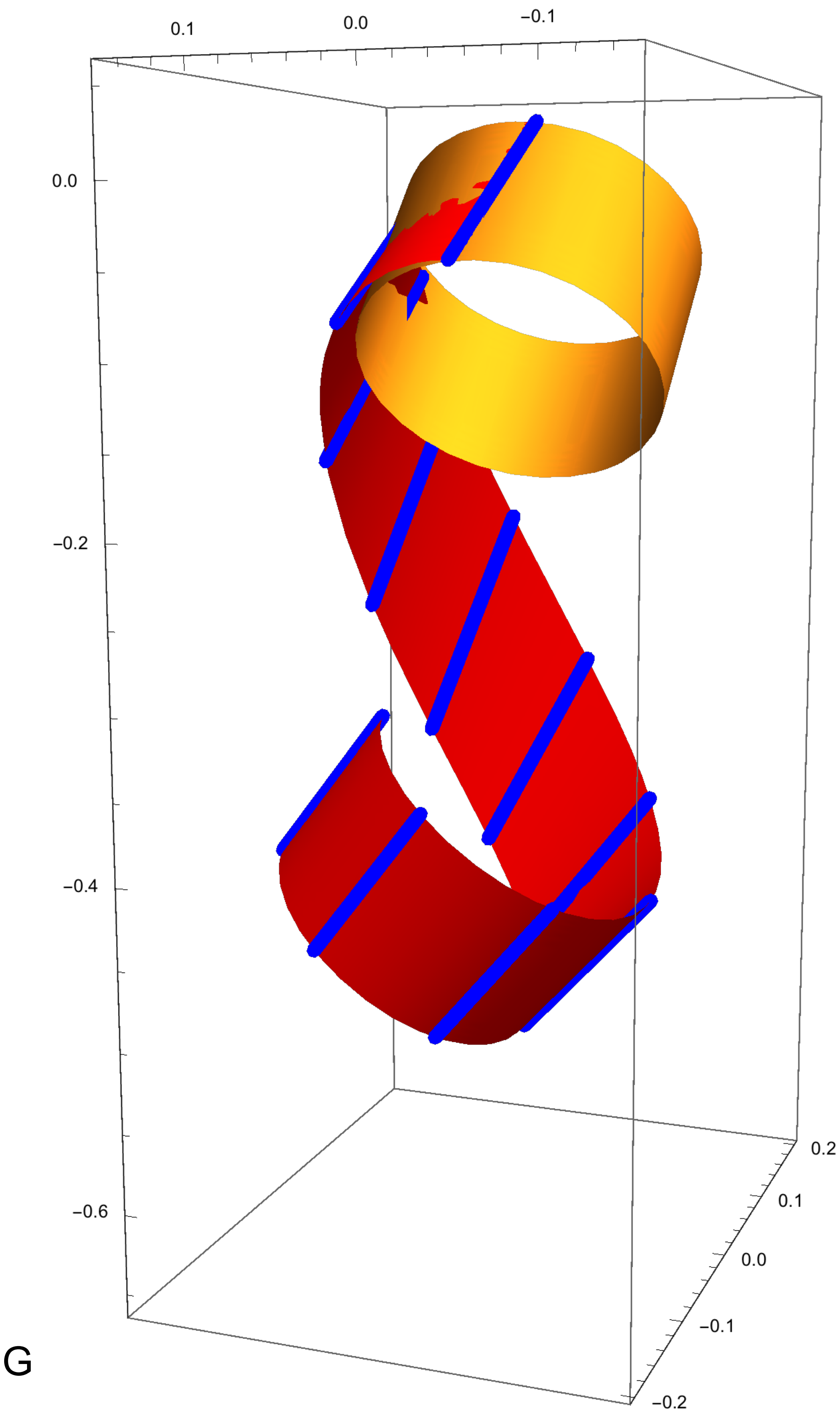
Shooting & AUTO:
sequence of equilibrium



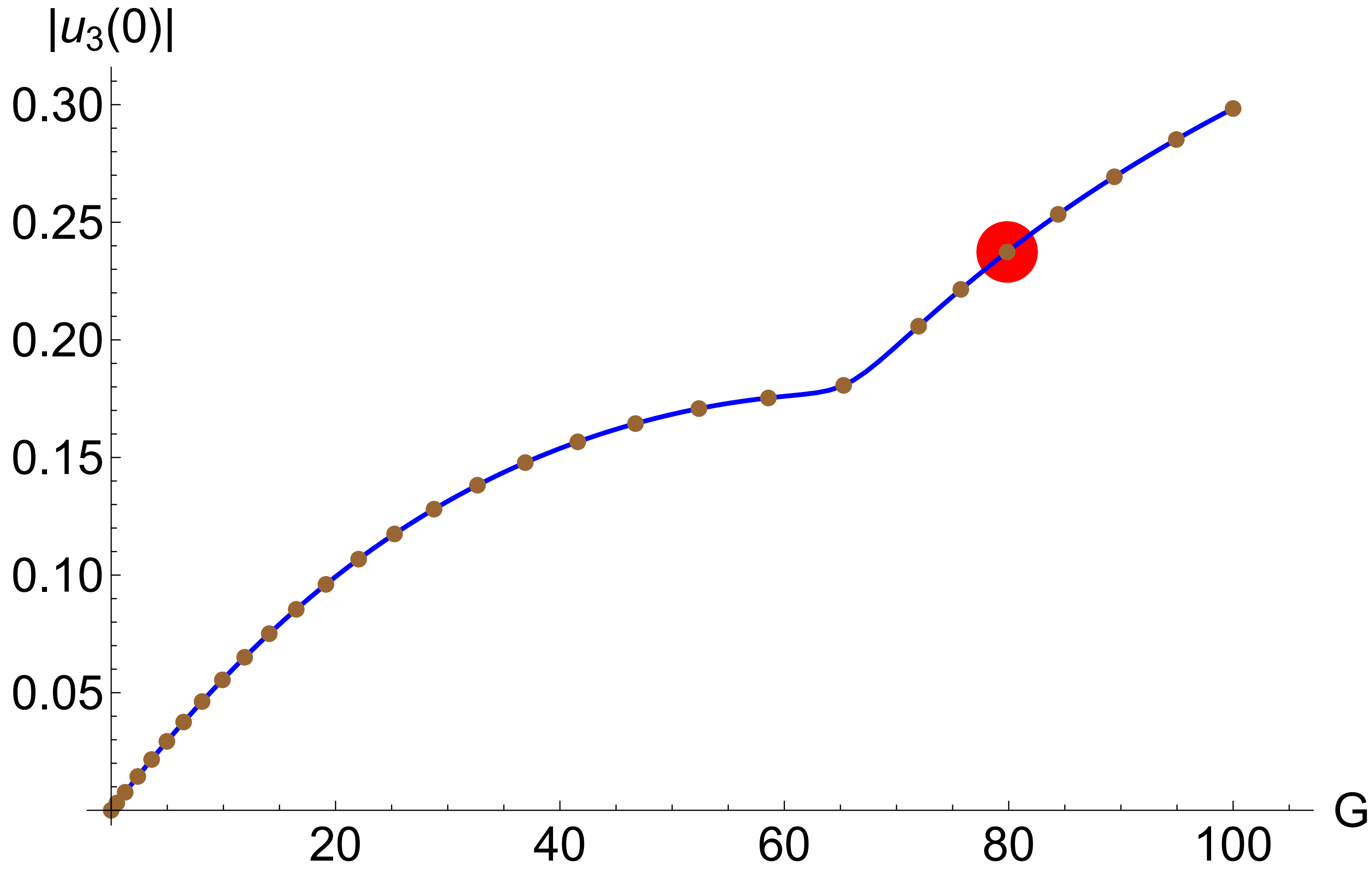


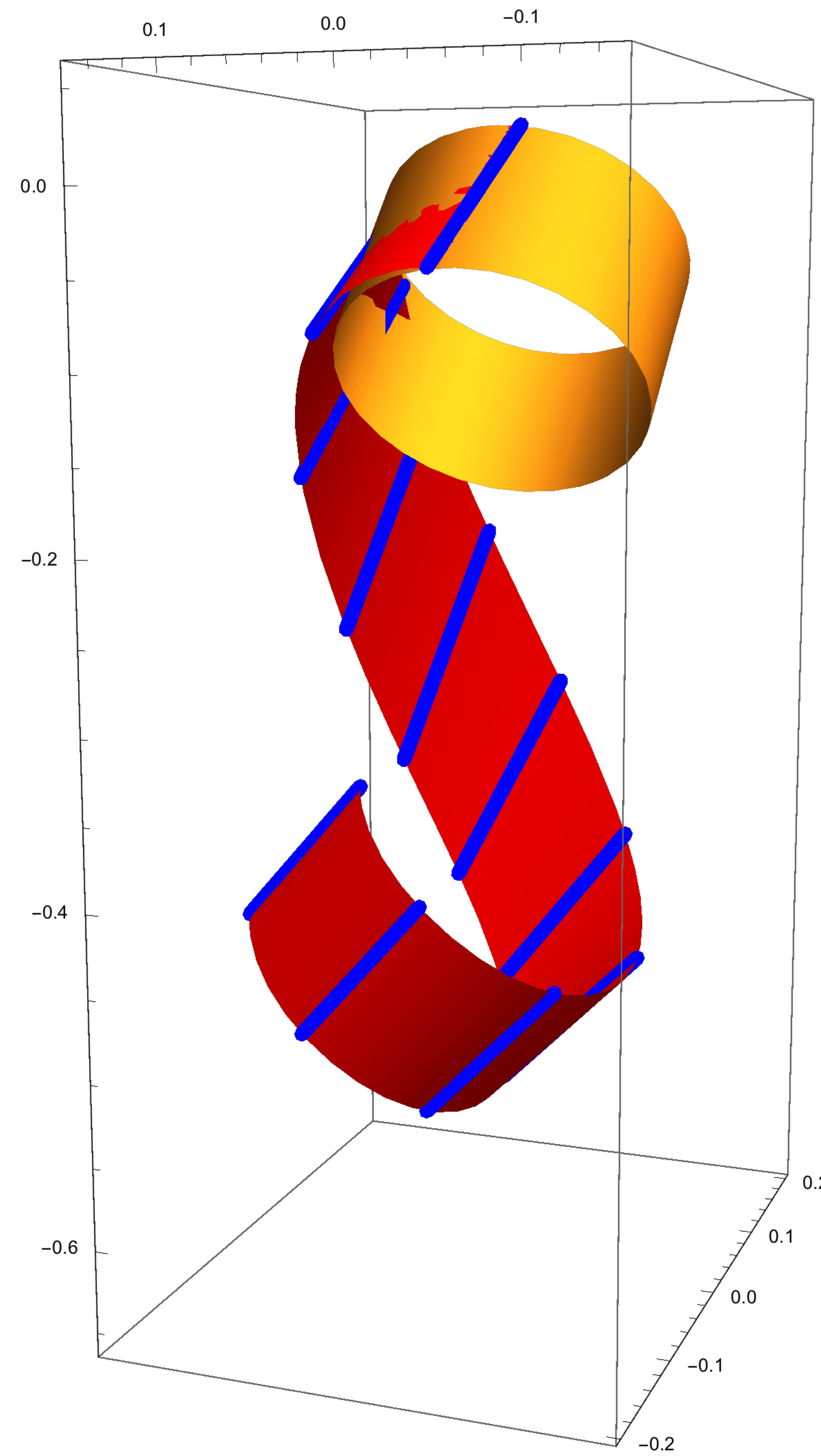
Shooting & AUTO:
sequence of equilibrium



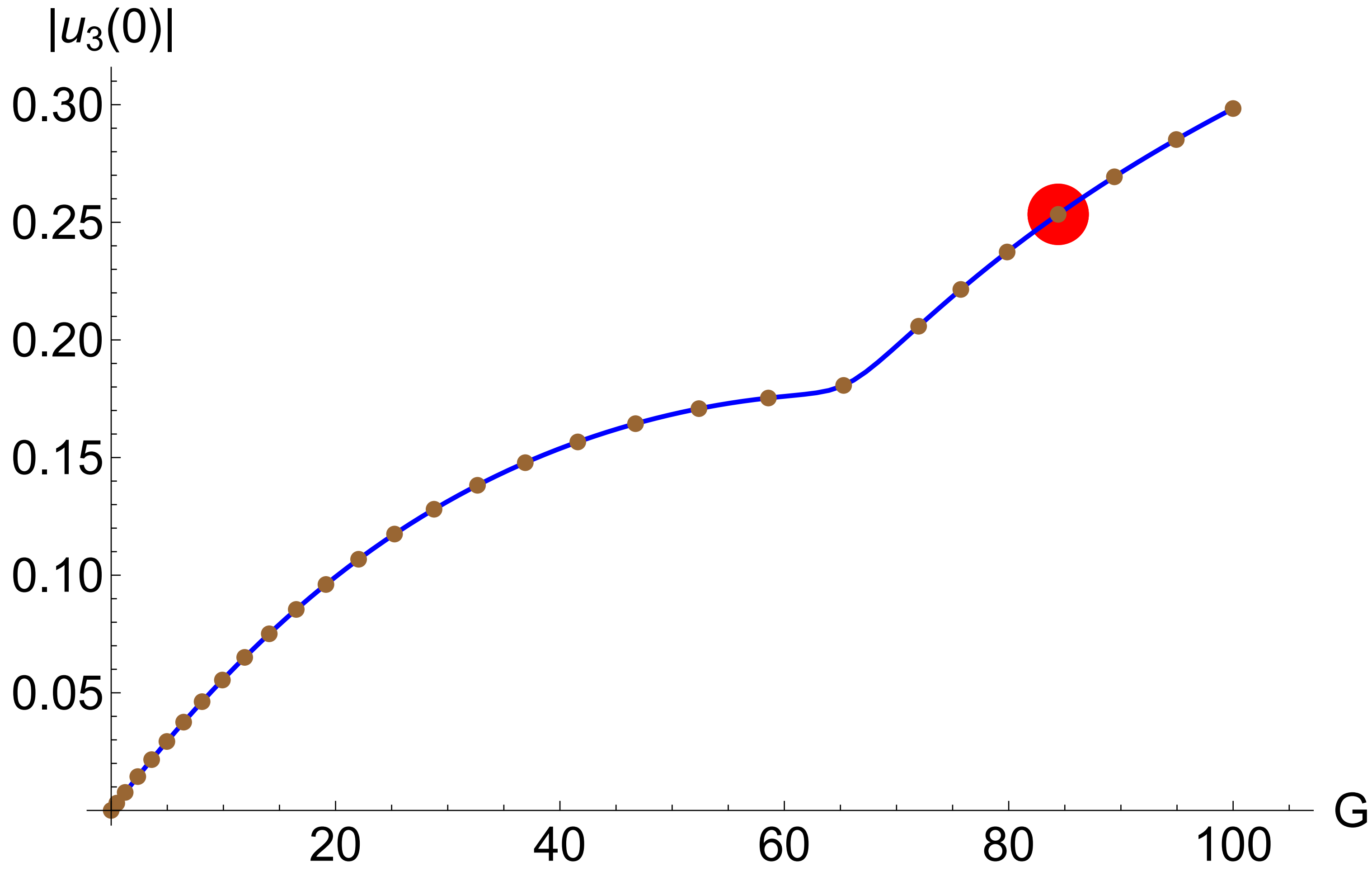


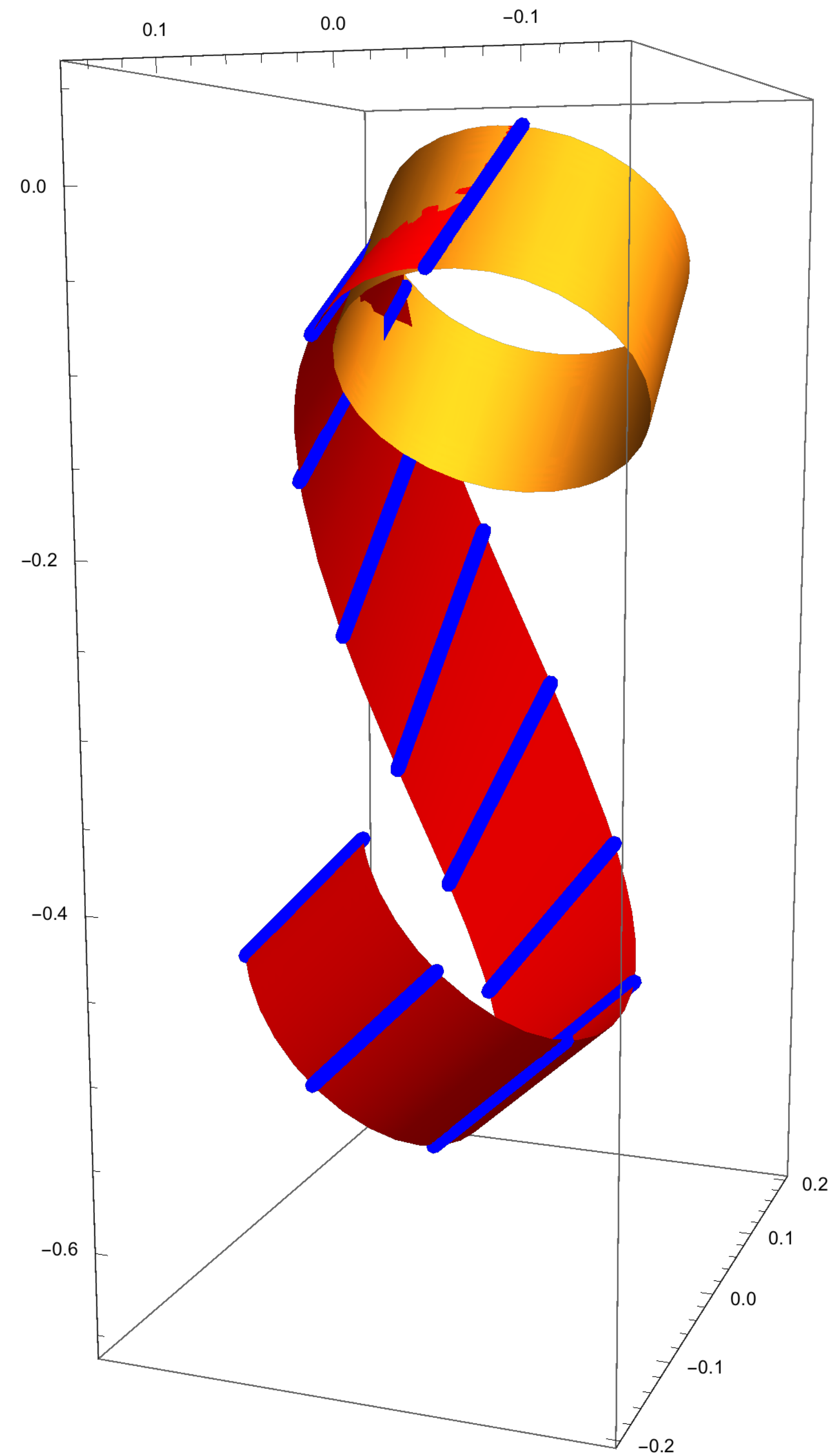
Shooting & AUTO: sequence of equilibrium



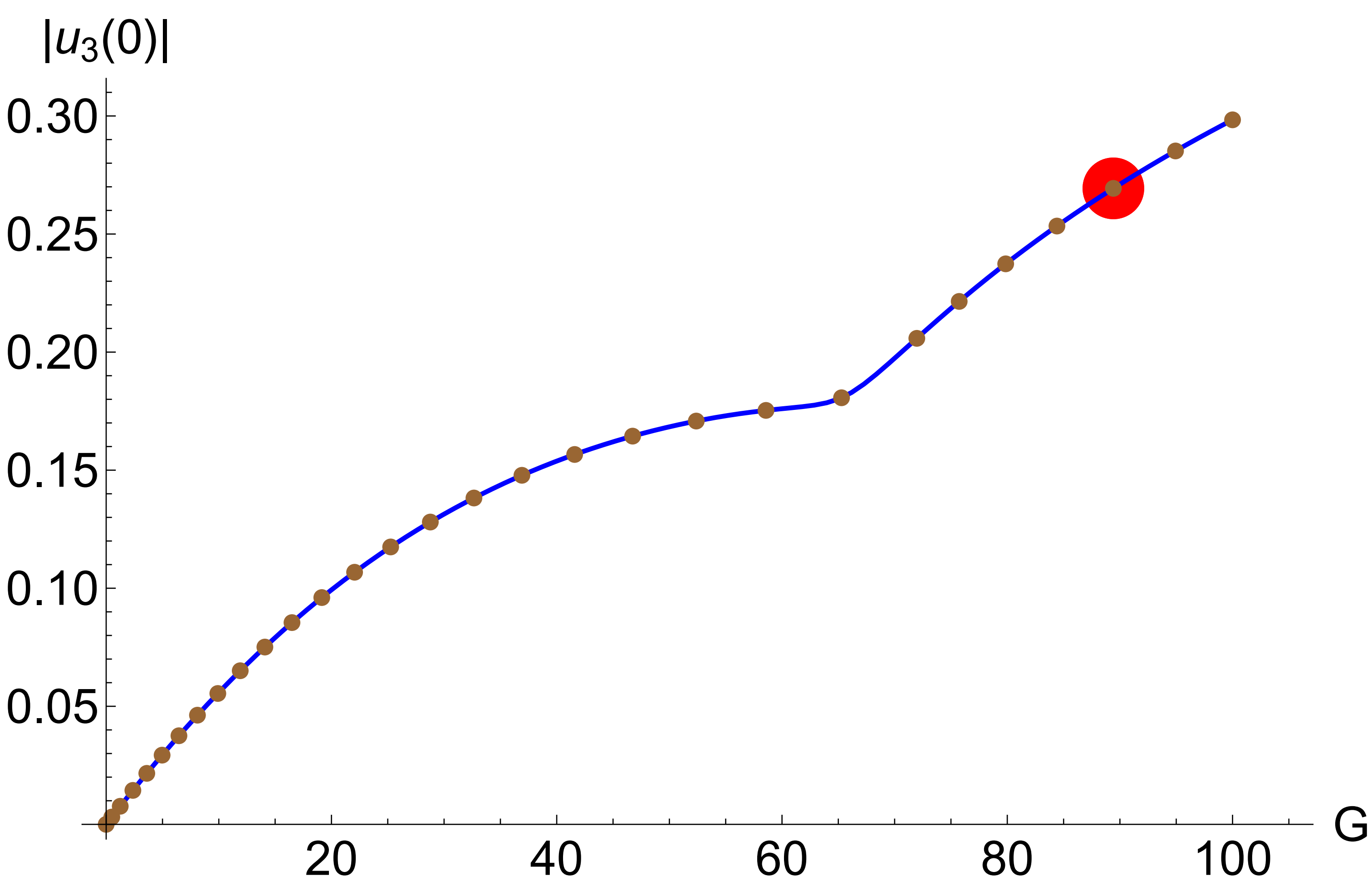


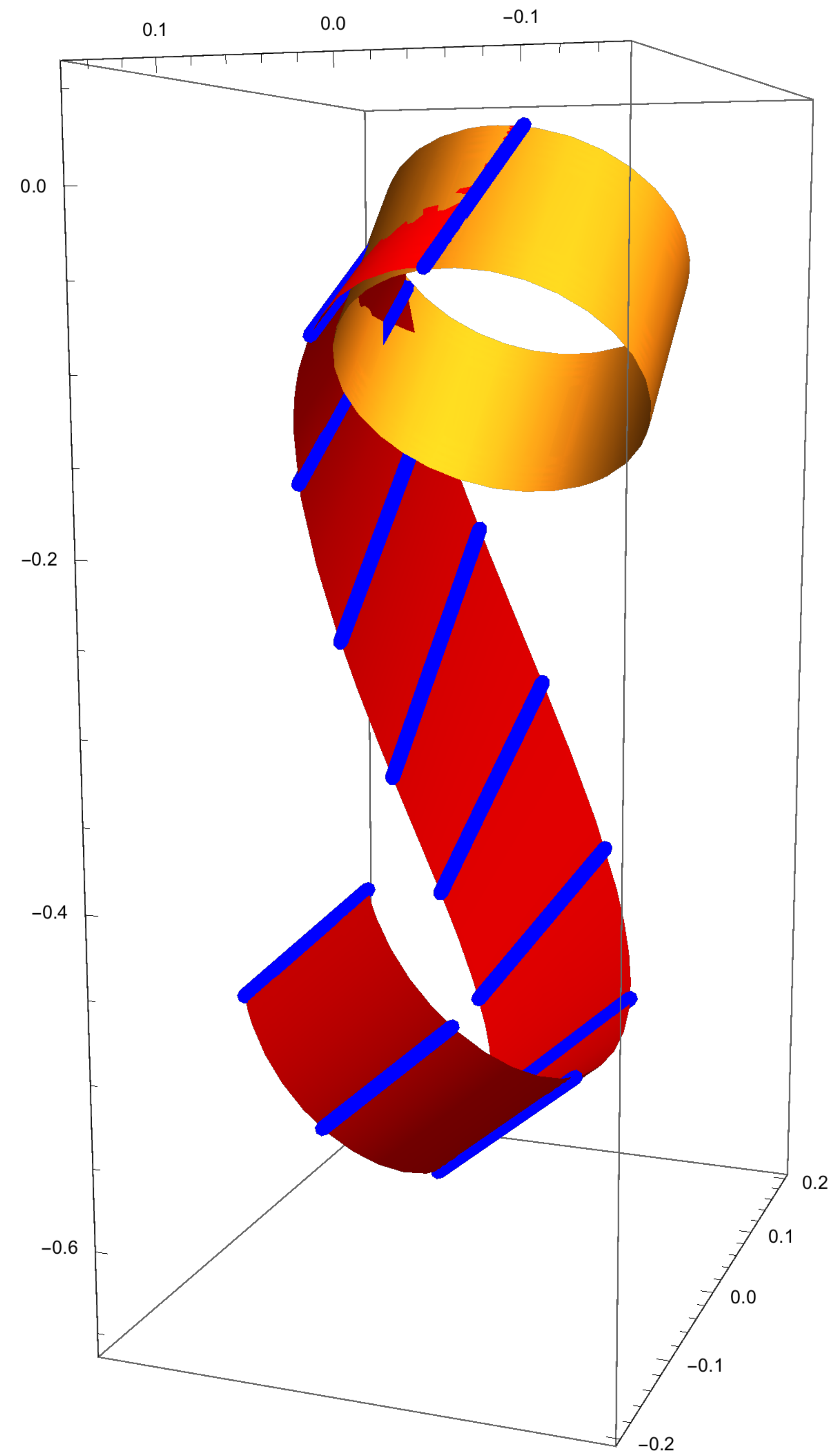
Shooting & AUTO:
sequence of equilibrium



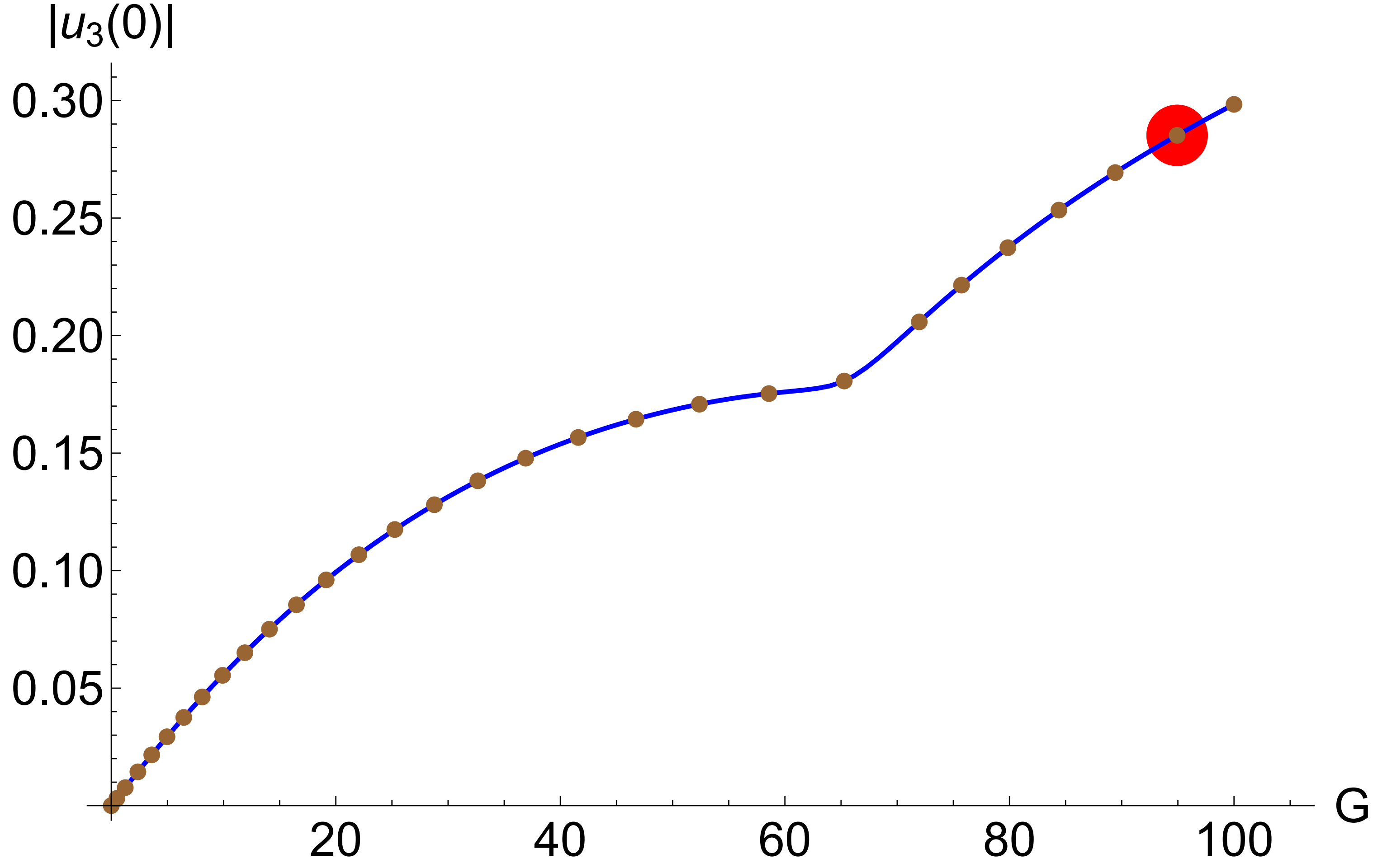


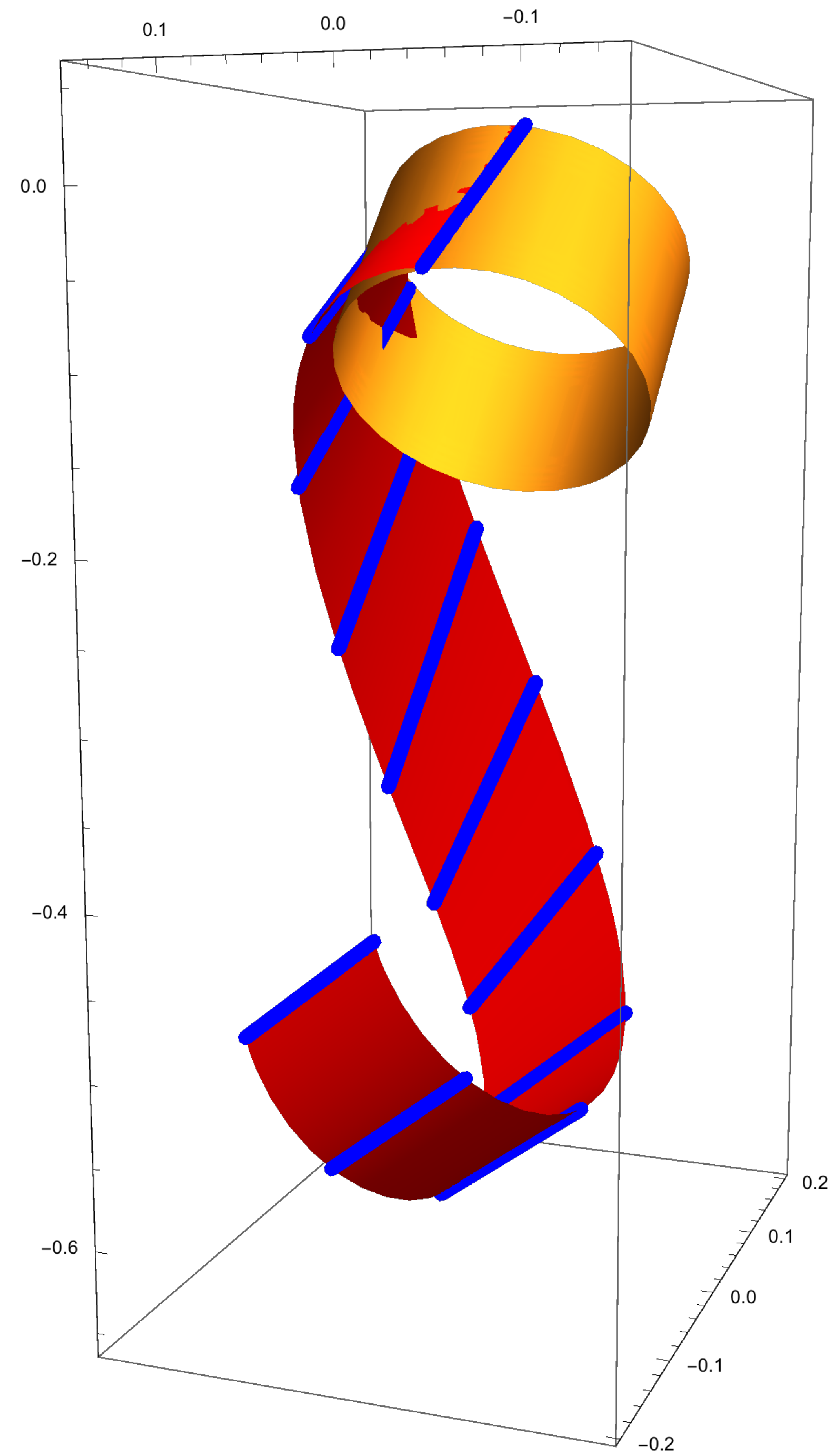
Shooting & AUTO:
sequence of equilibrium



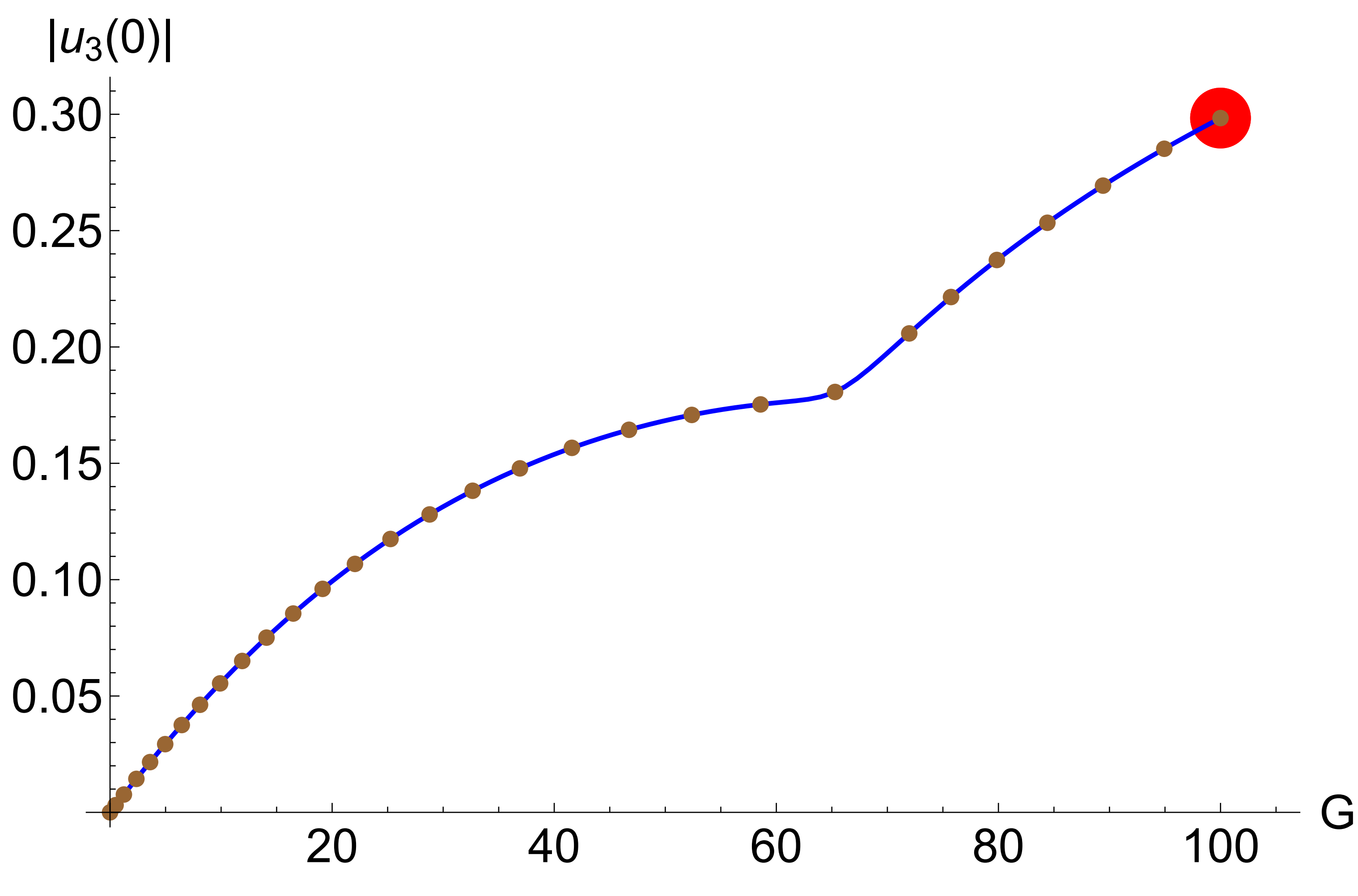


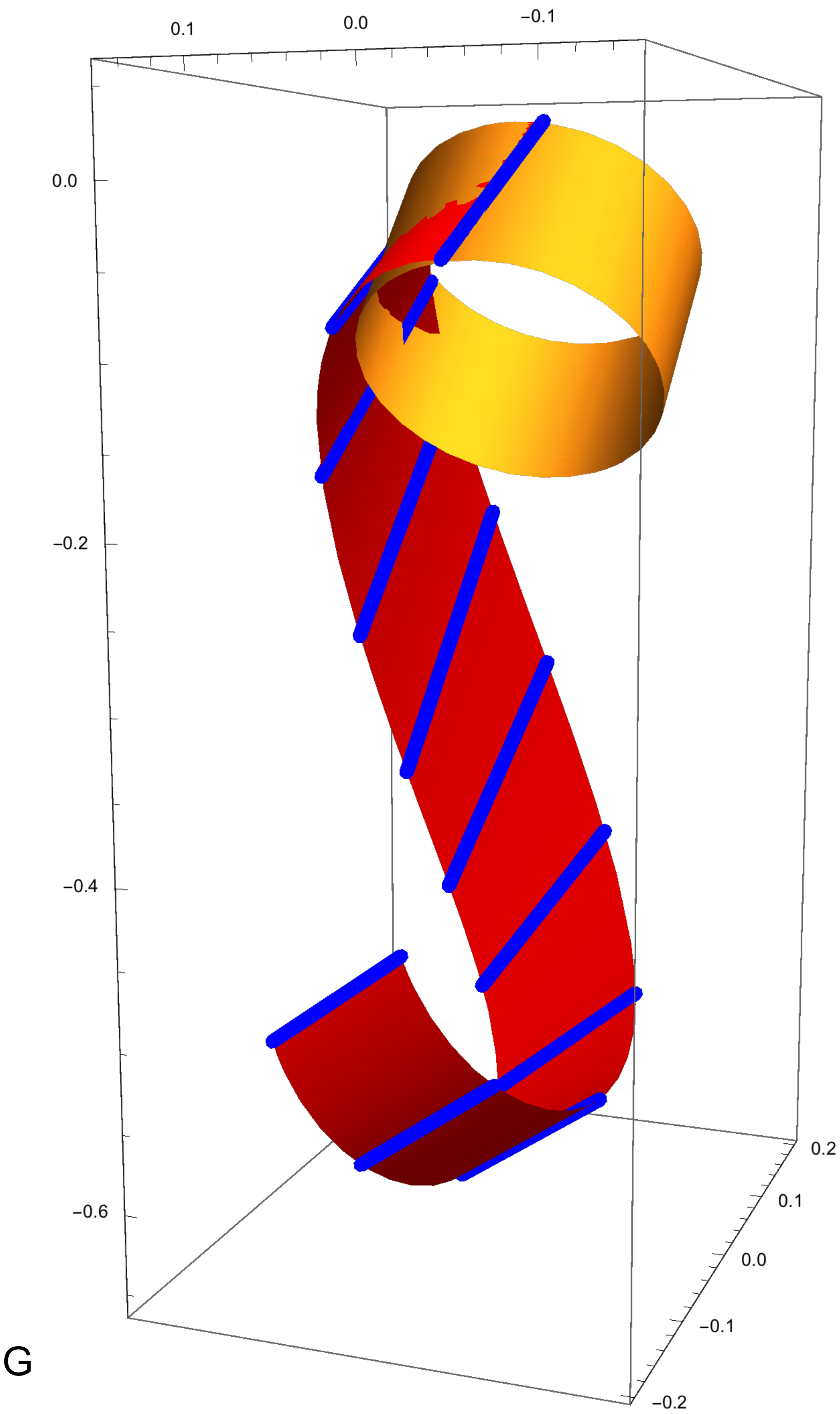
Shooting & AUTO: sequence of equilibrium





Shooting & AUTO:
sequence of equilibrium





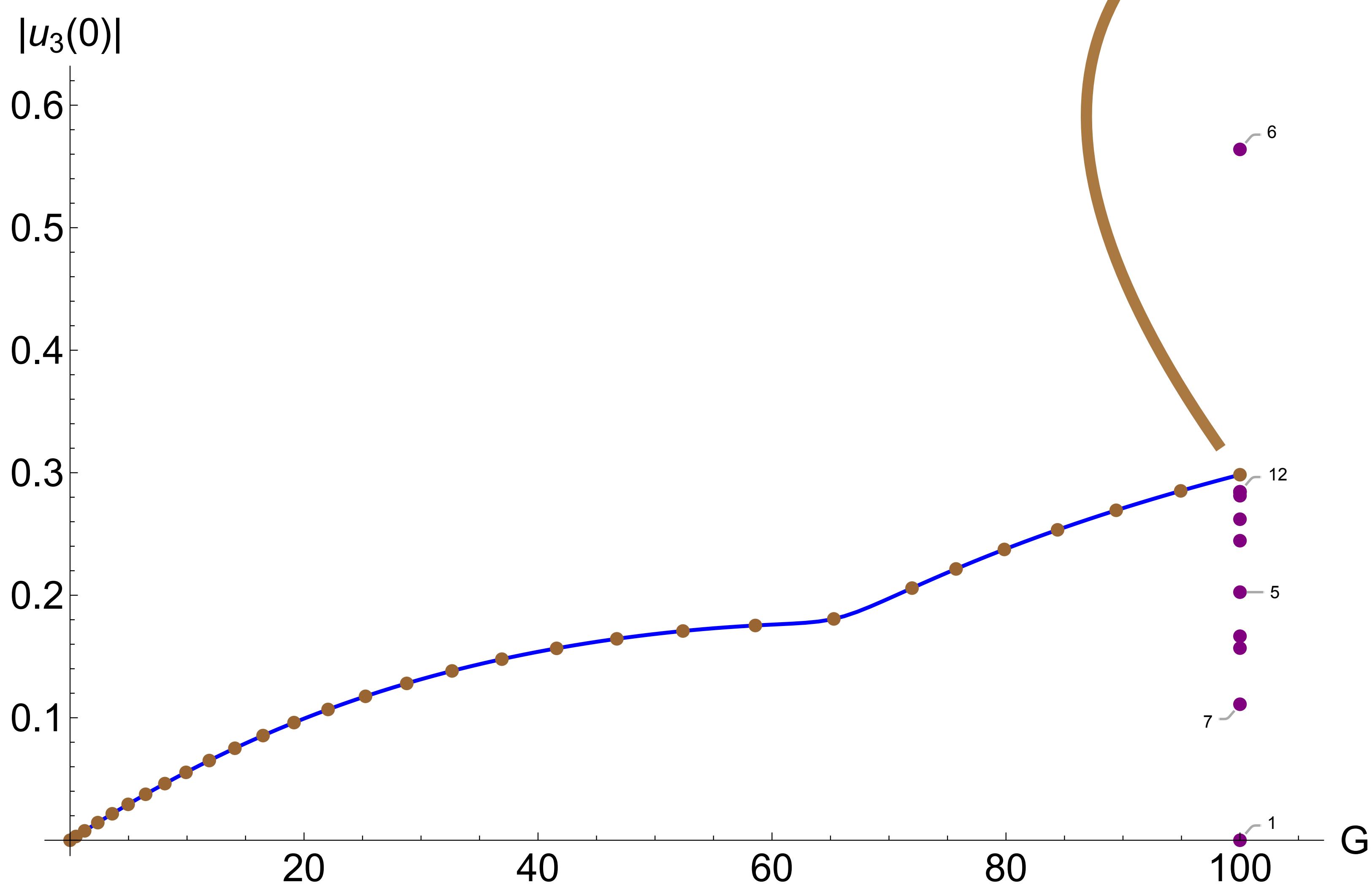
Elastic ribbon

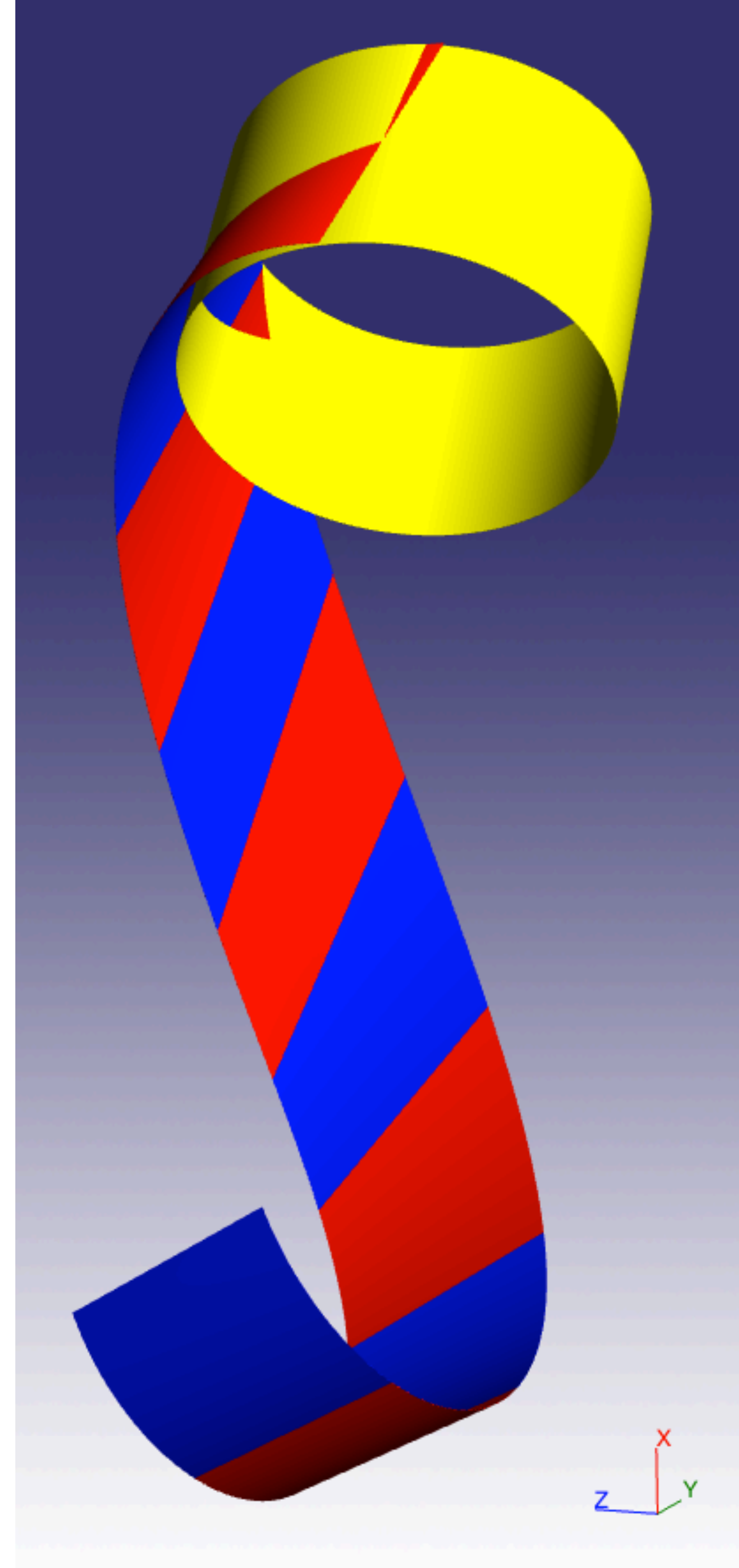
Goal: obtain K=10, G=100

adim natural curvature (K)

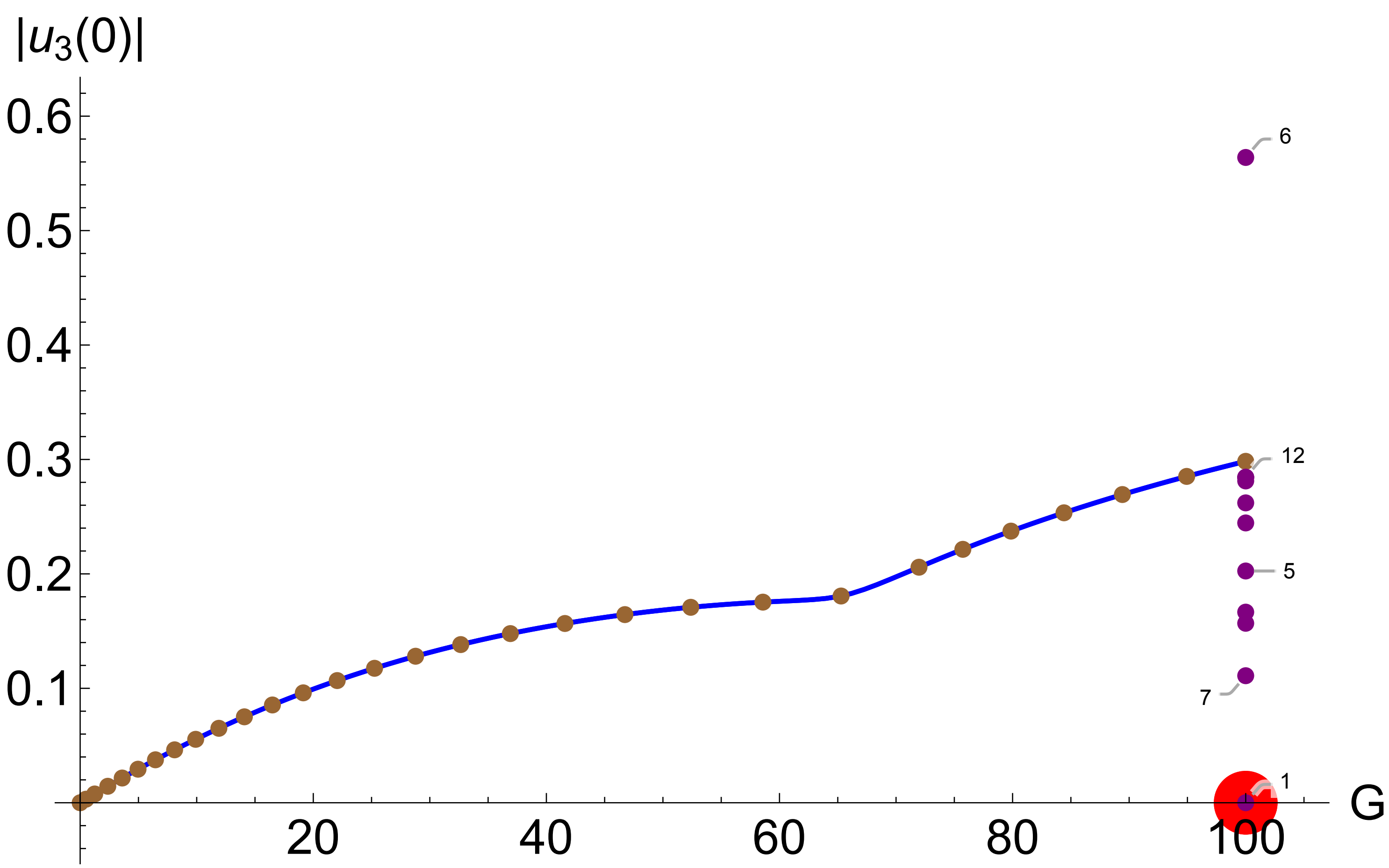
adim weight (G)

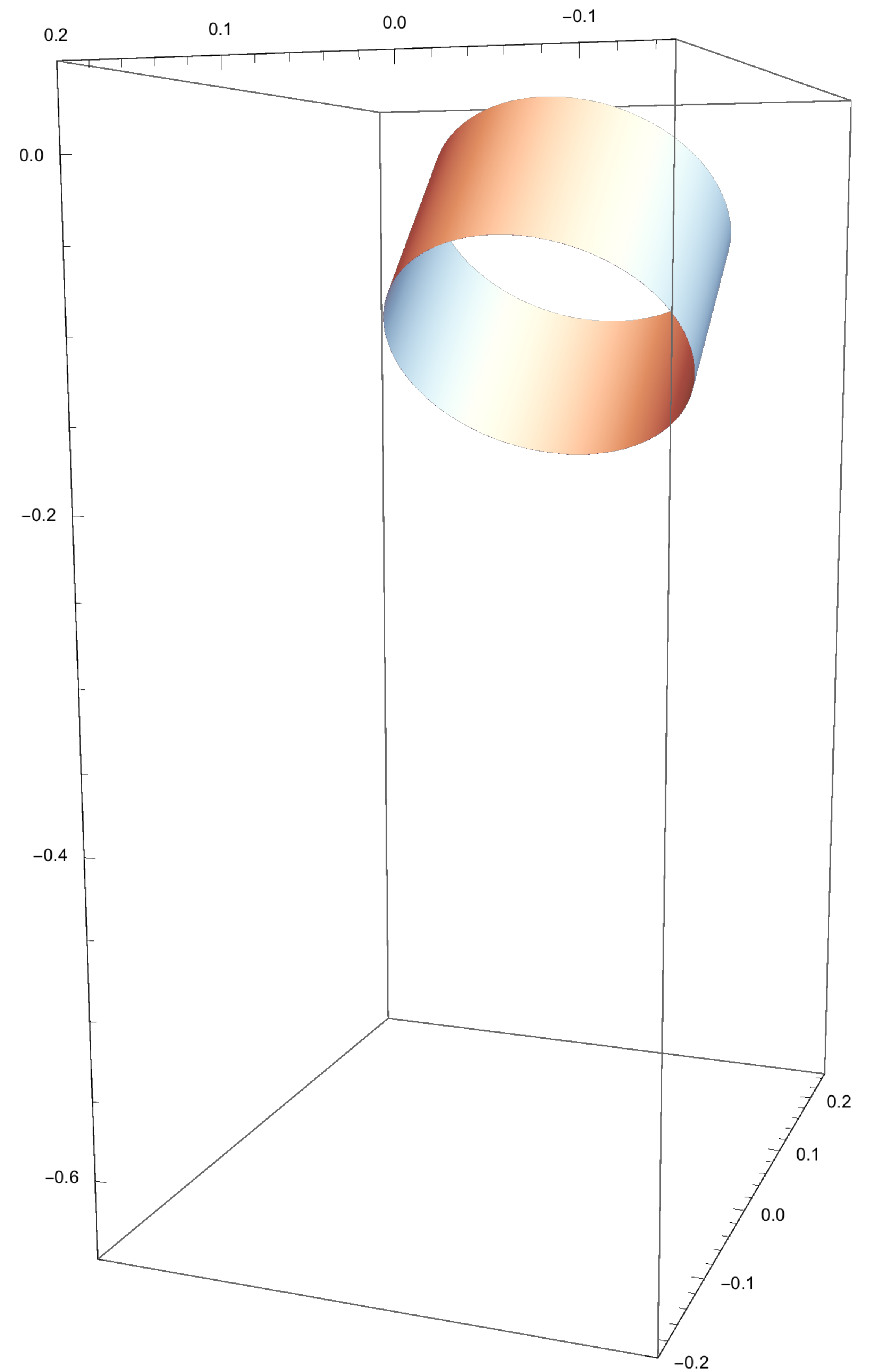
Shooting: 42 pts (8sec)
AUTO: 30 pts (0.11sec) (NTST=10, NCOL=4)
IPOPT: 11 pts (0.09sec) (high order elem. 10 seg.)



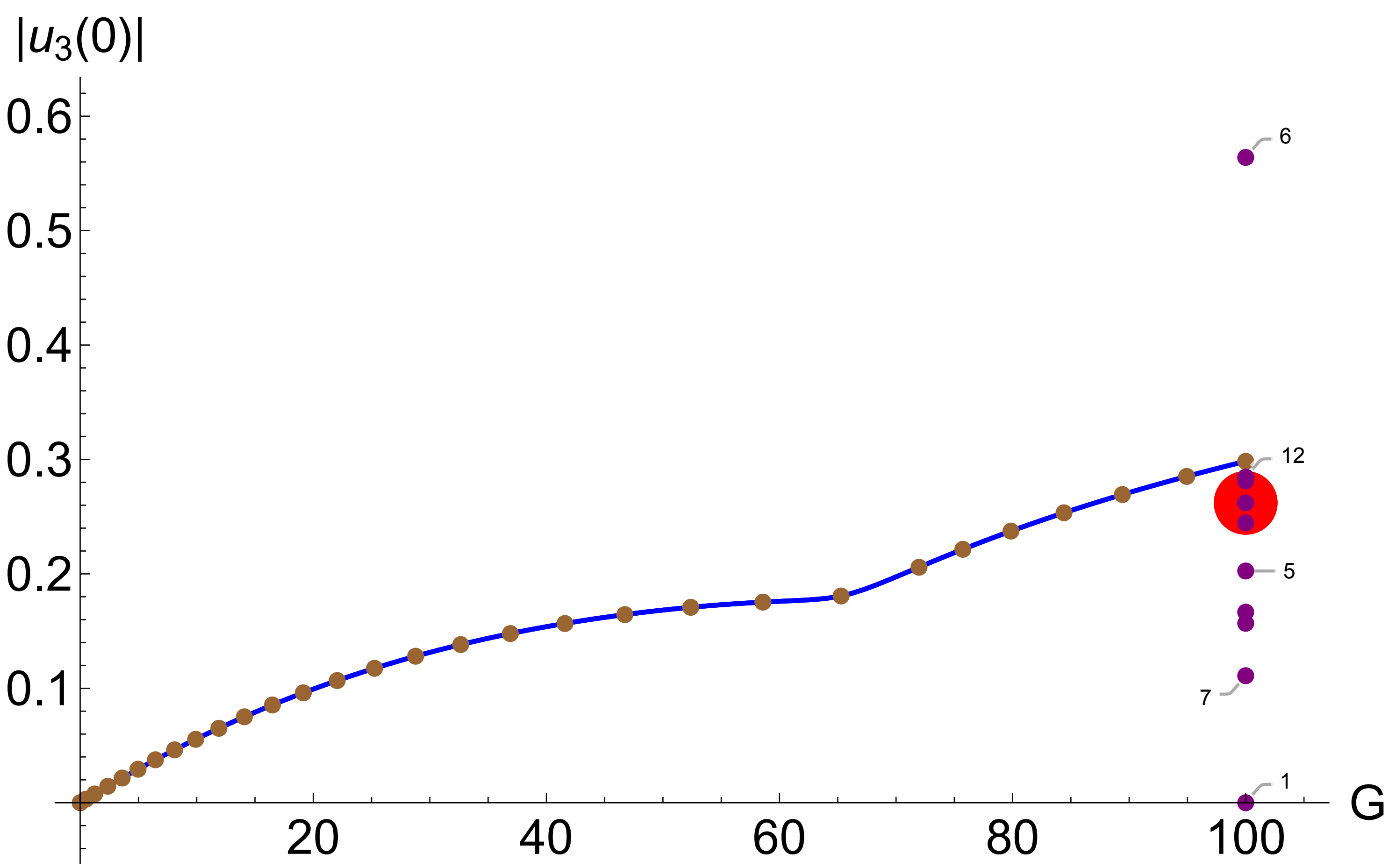


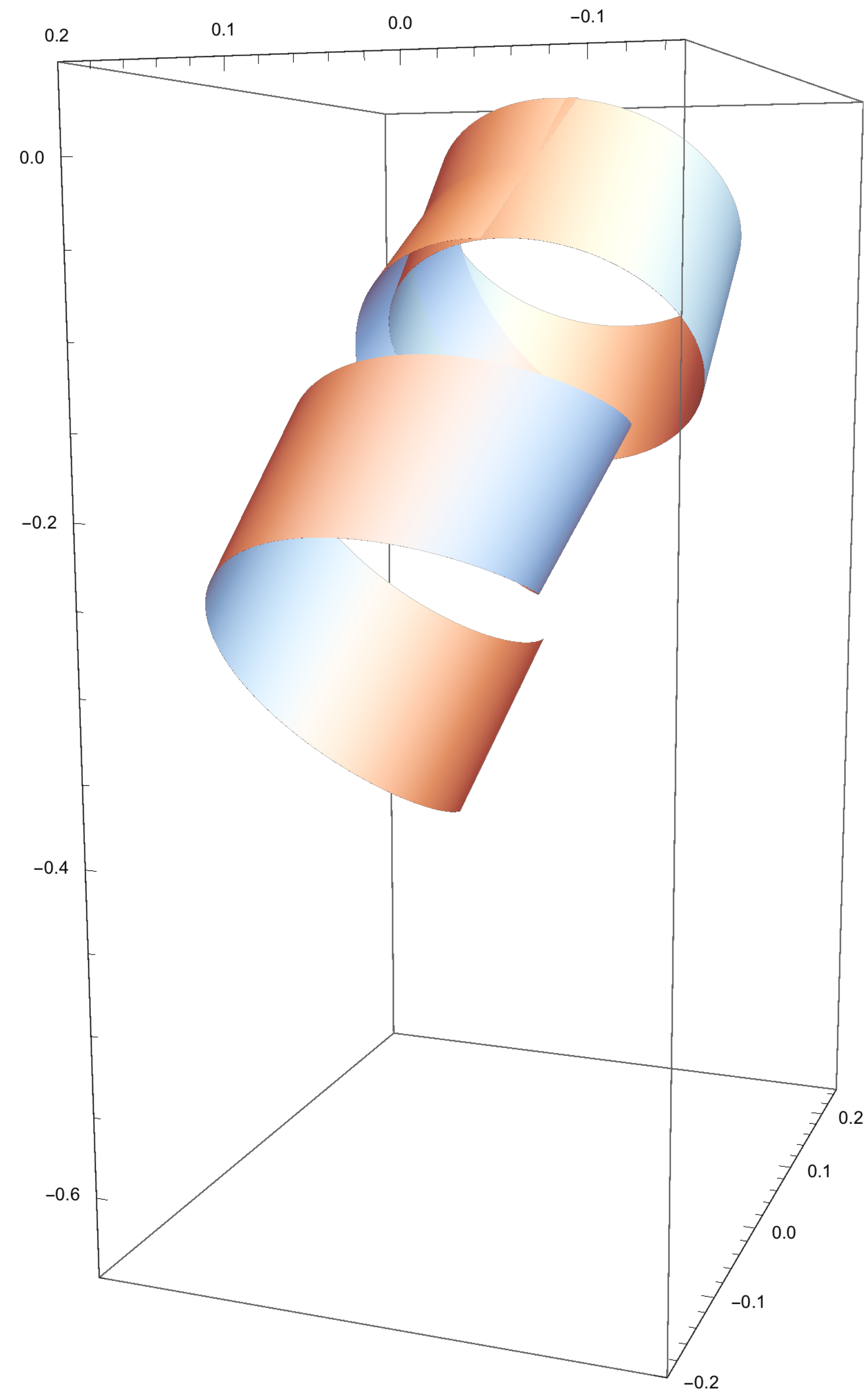
IPOPT: non equilibrium states



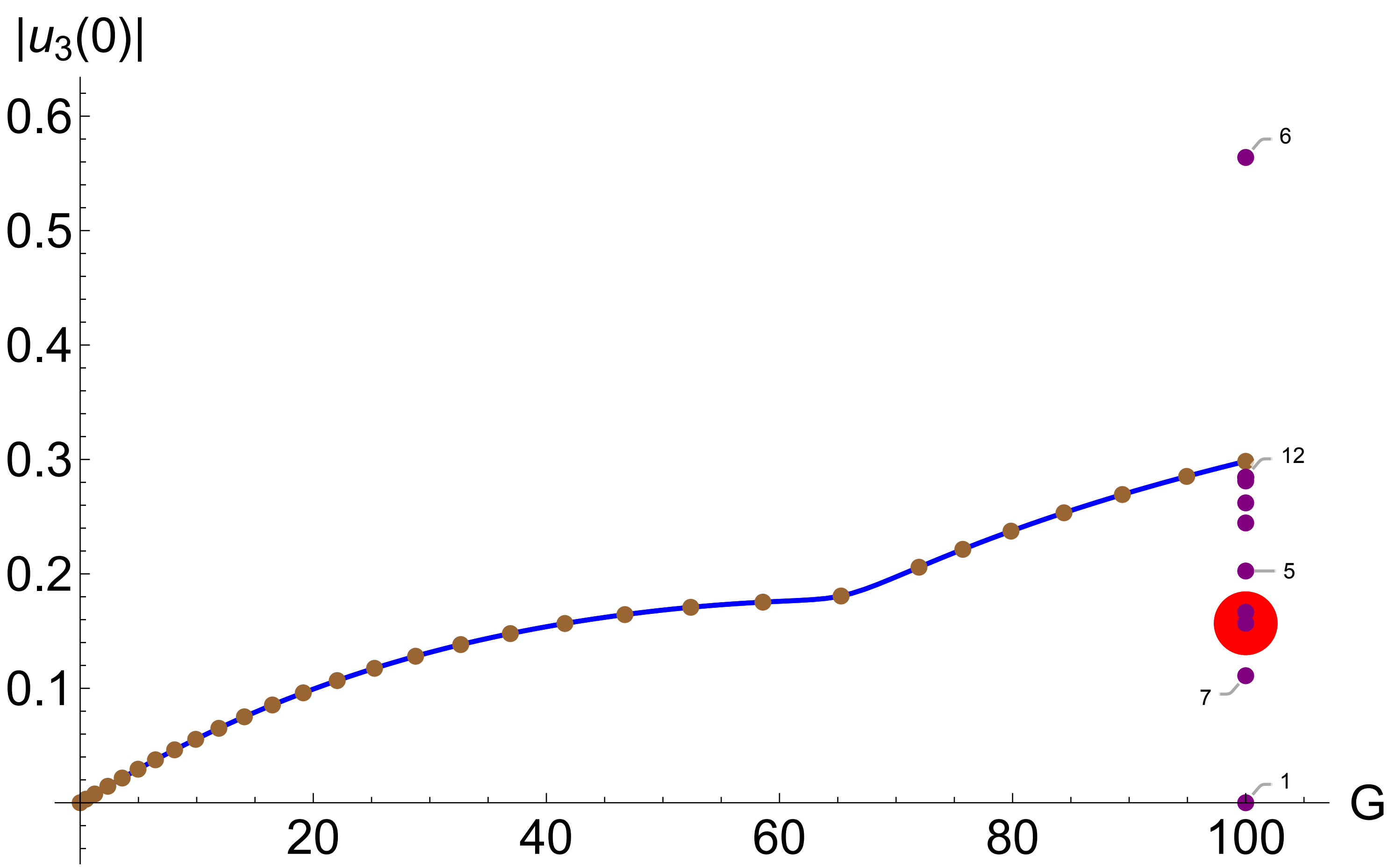


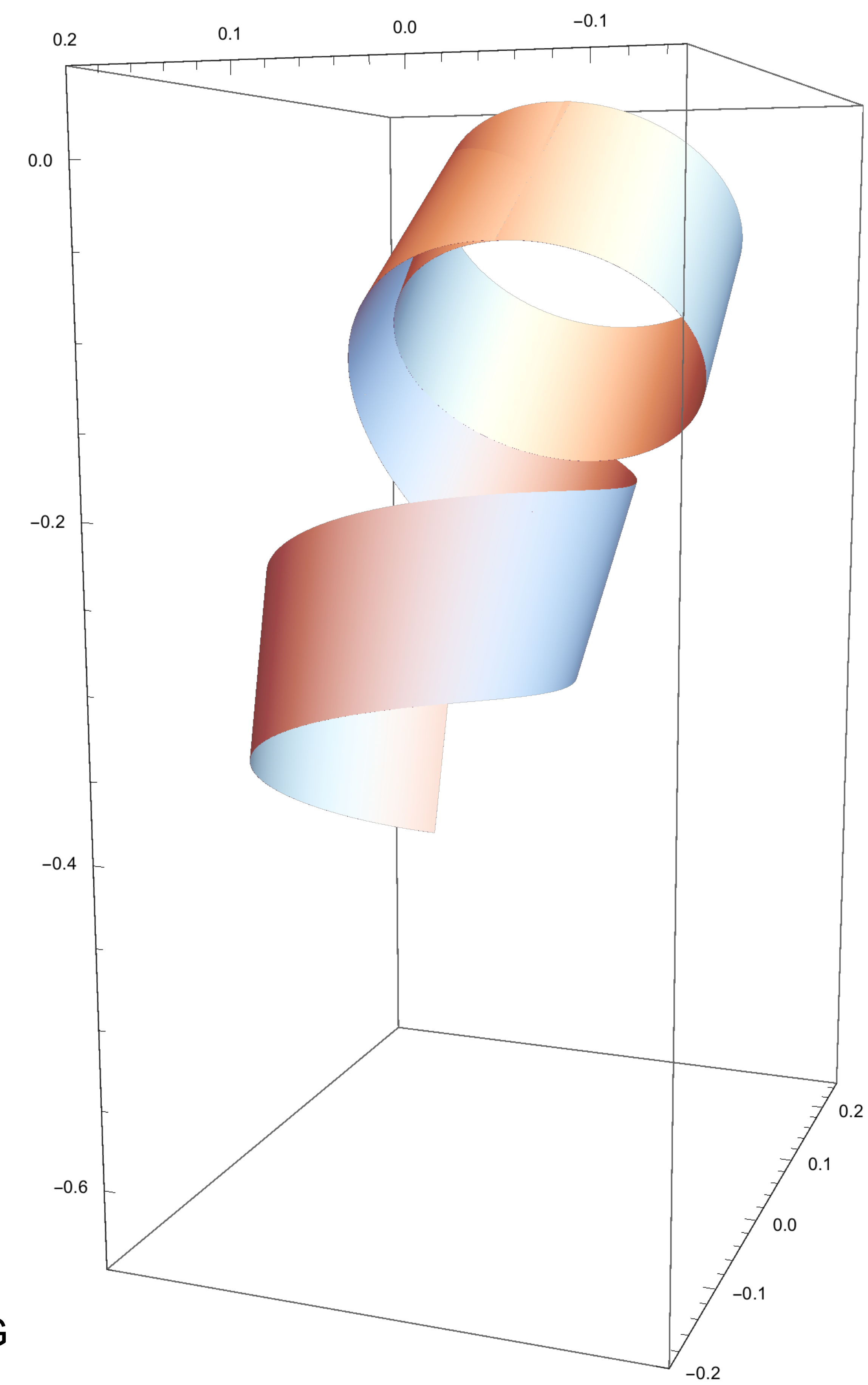
IPOPT: non equilibrium states



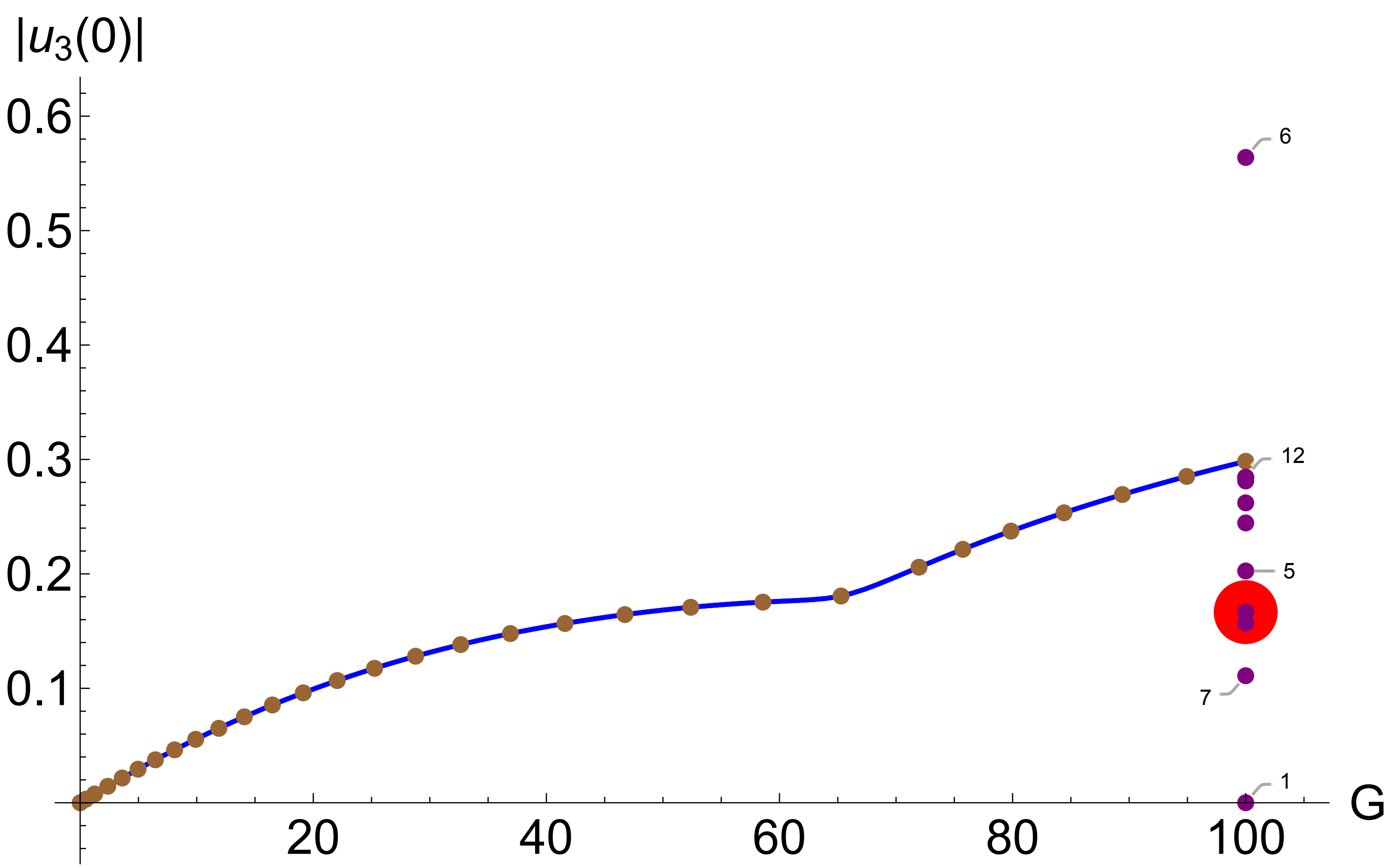


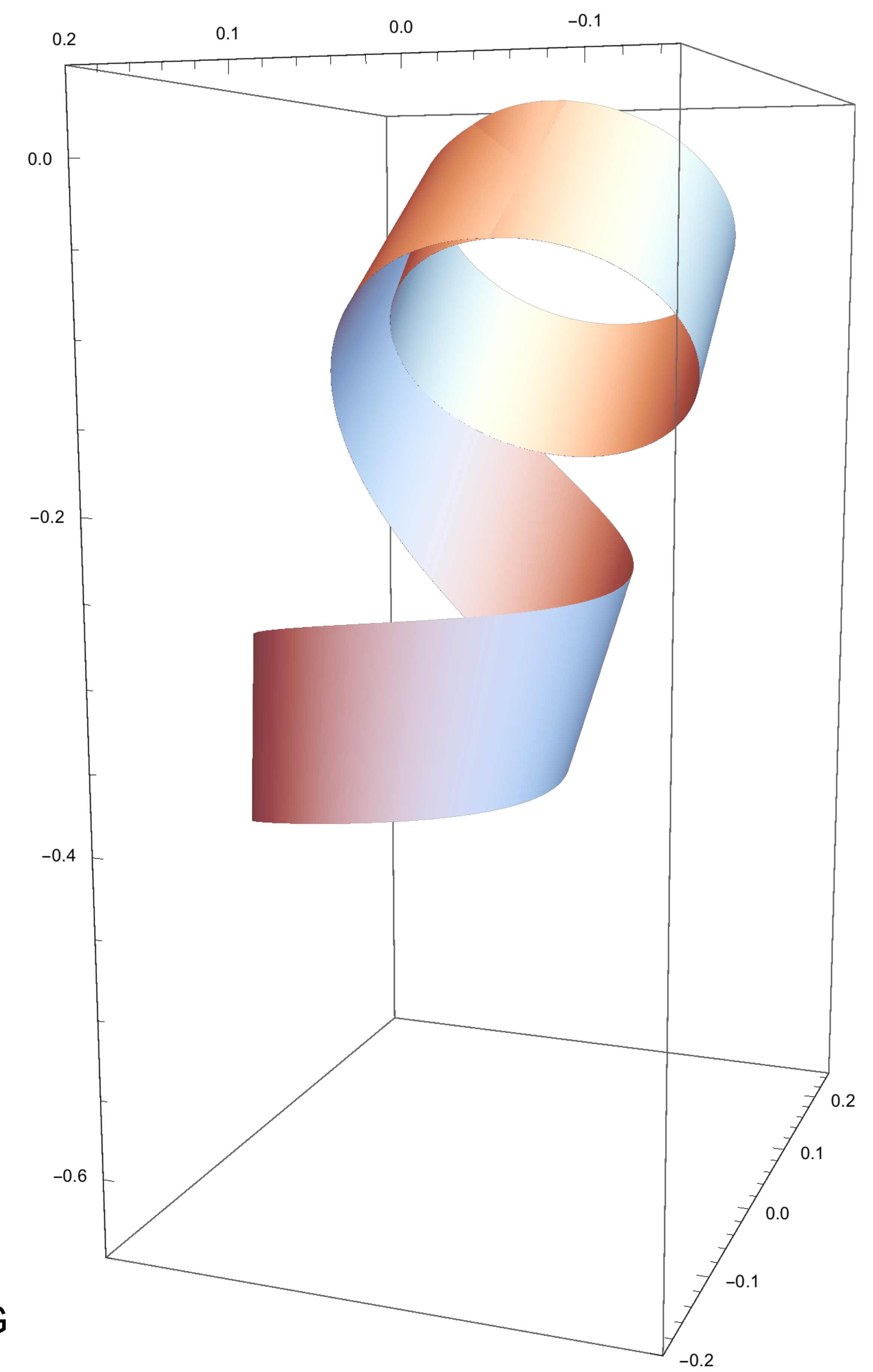
IPOPT: non equilibrium states



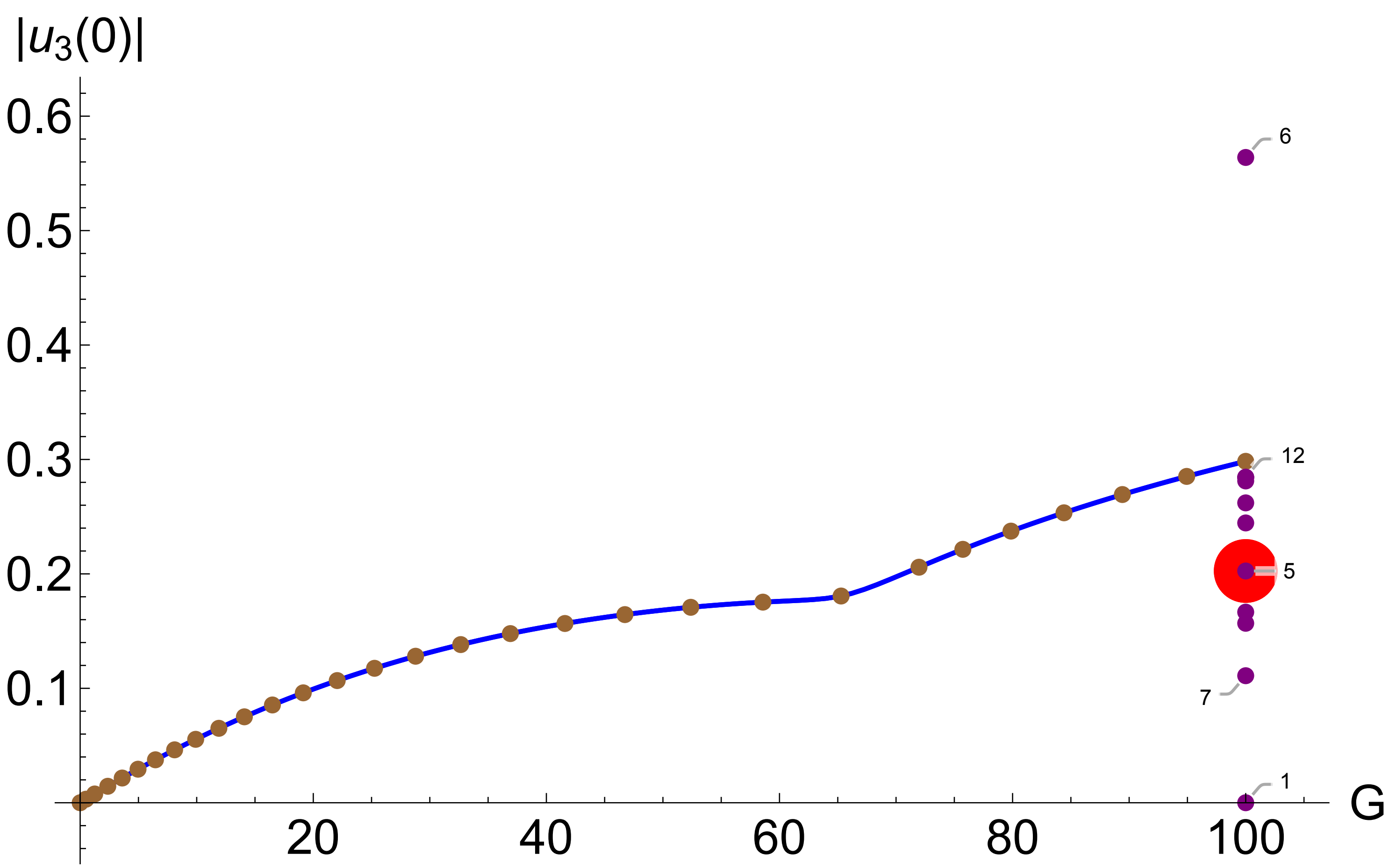


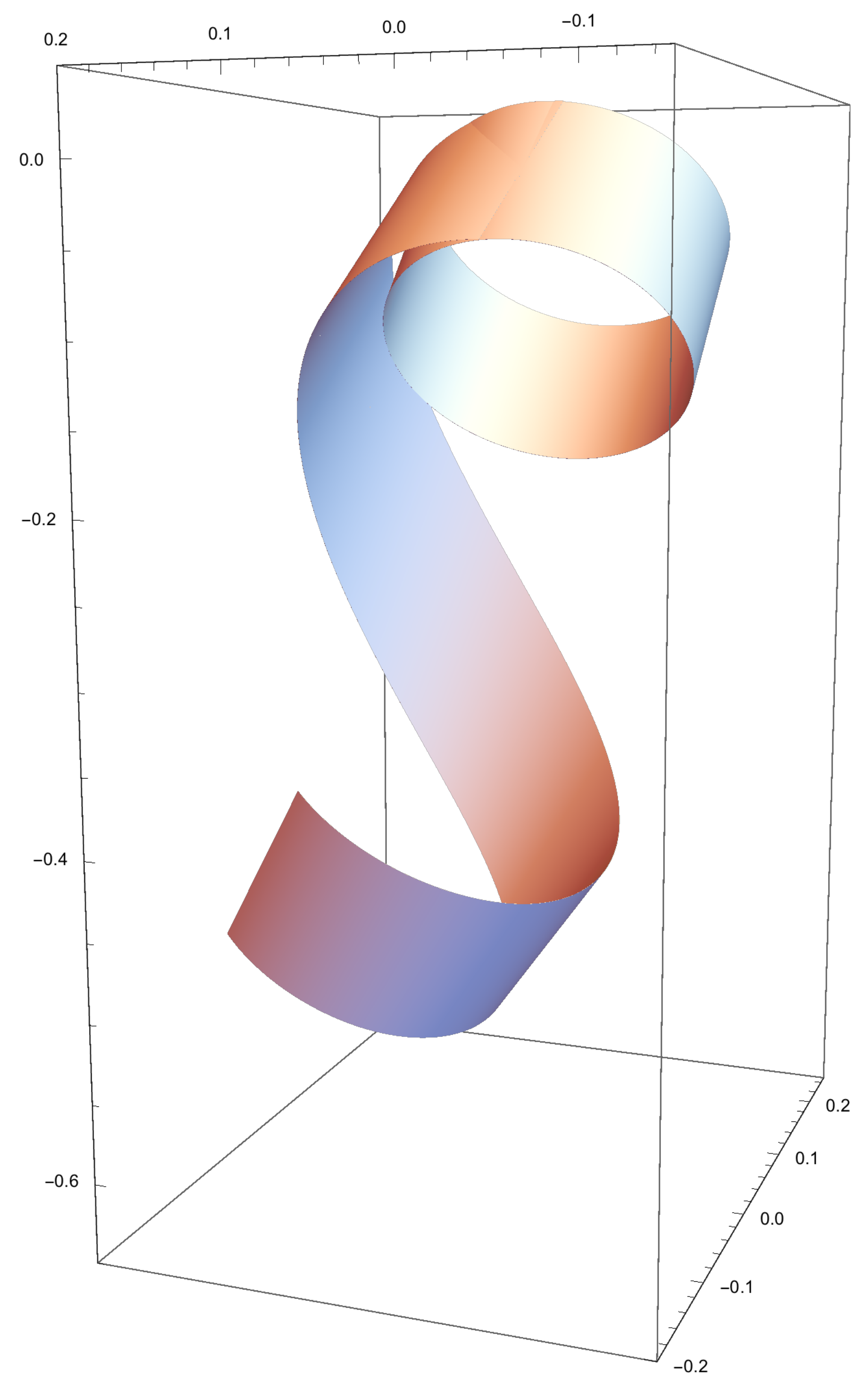
IPOPT: non equilibrium states



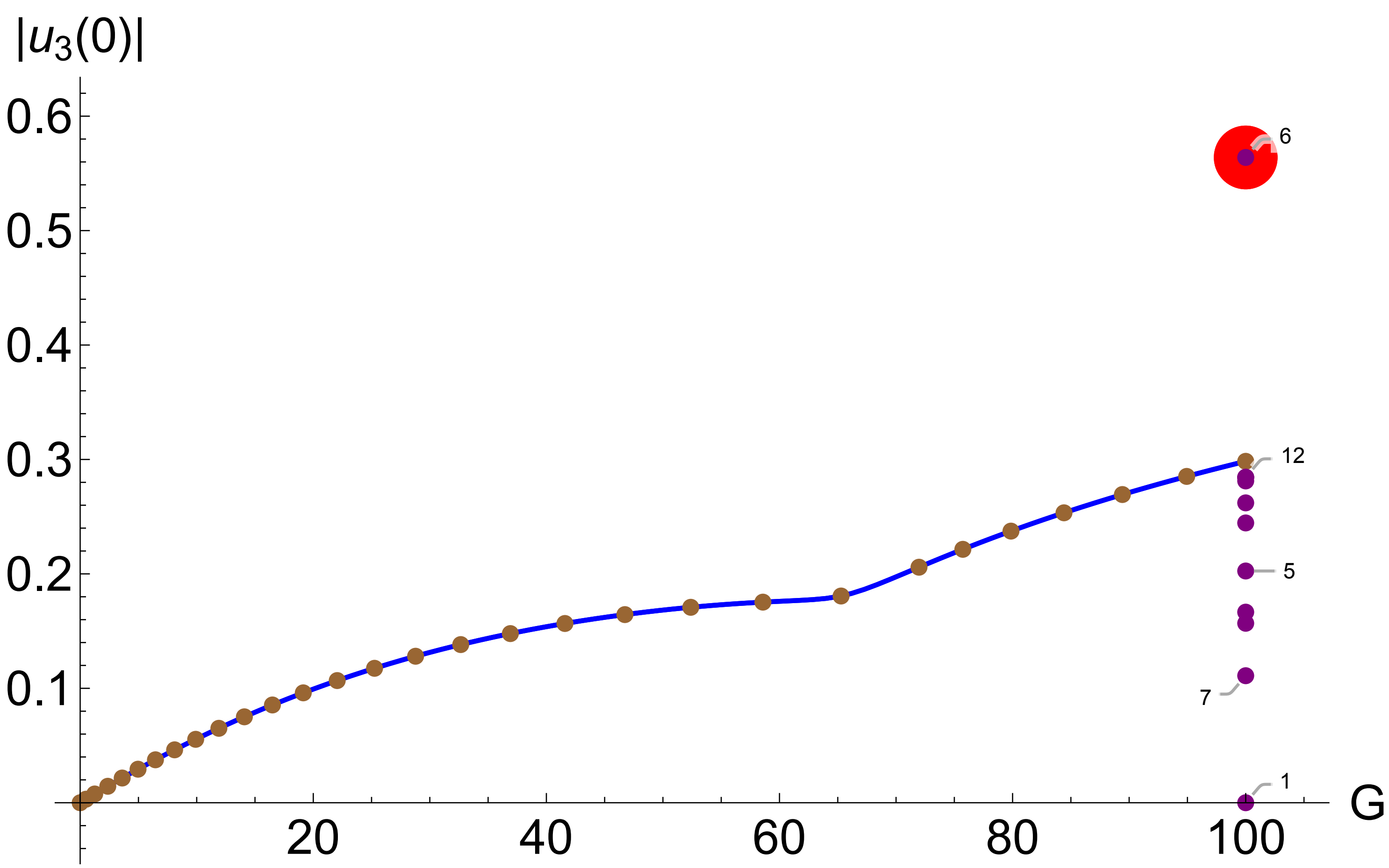


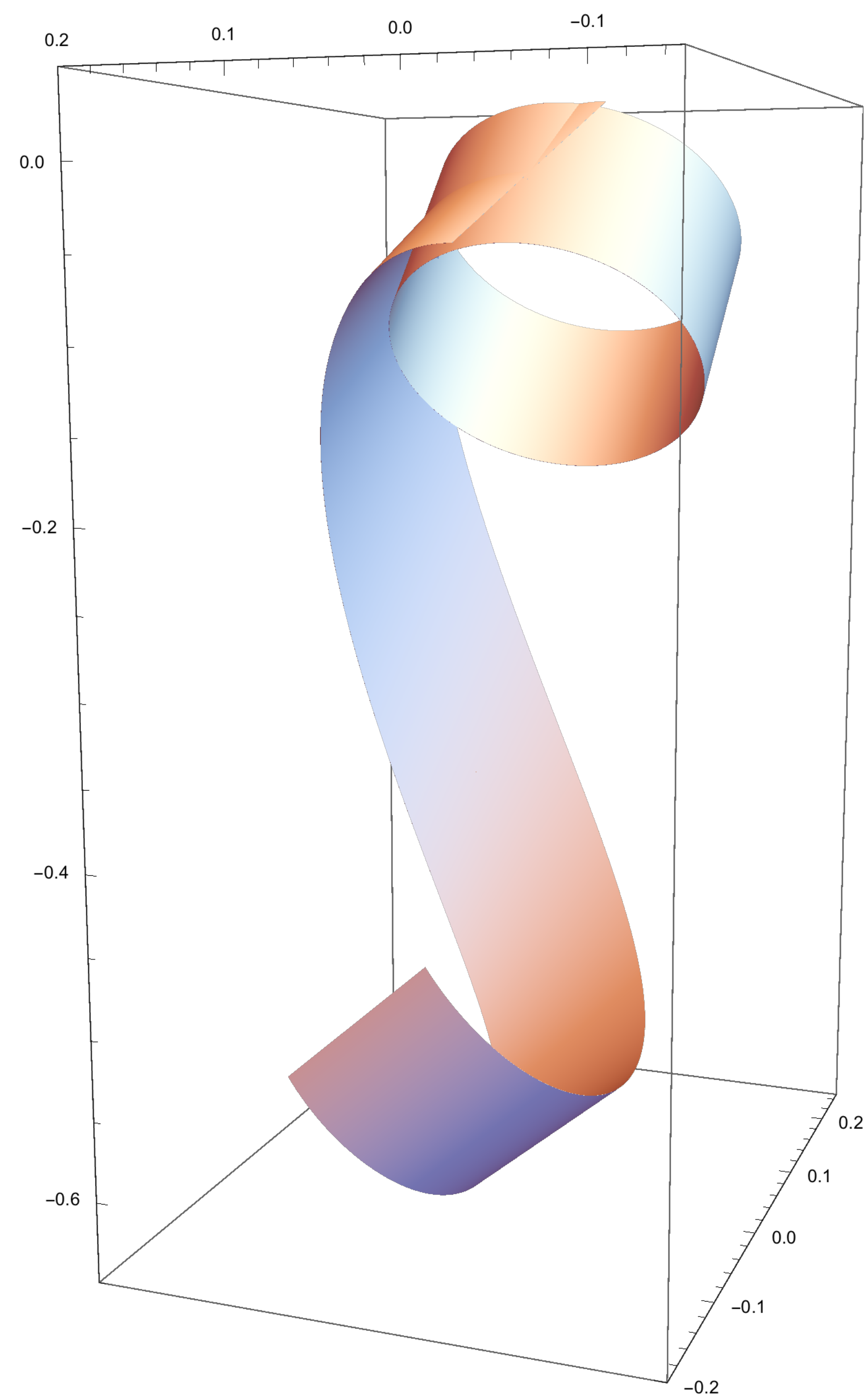
IPOPT: non equilibrium states



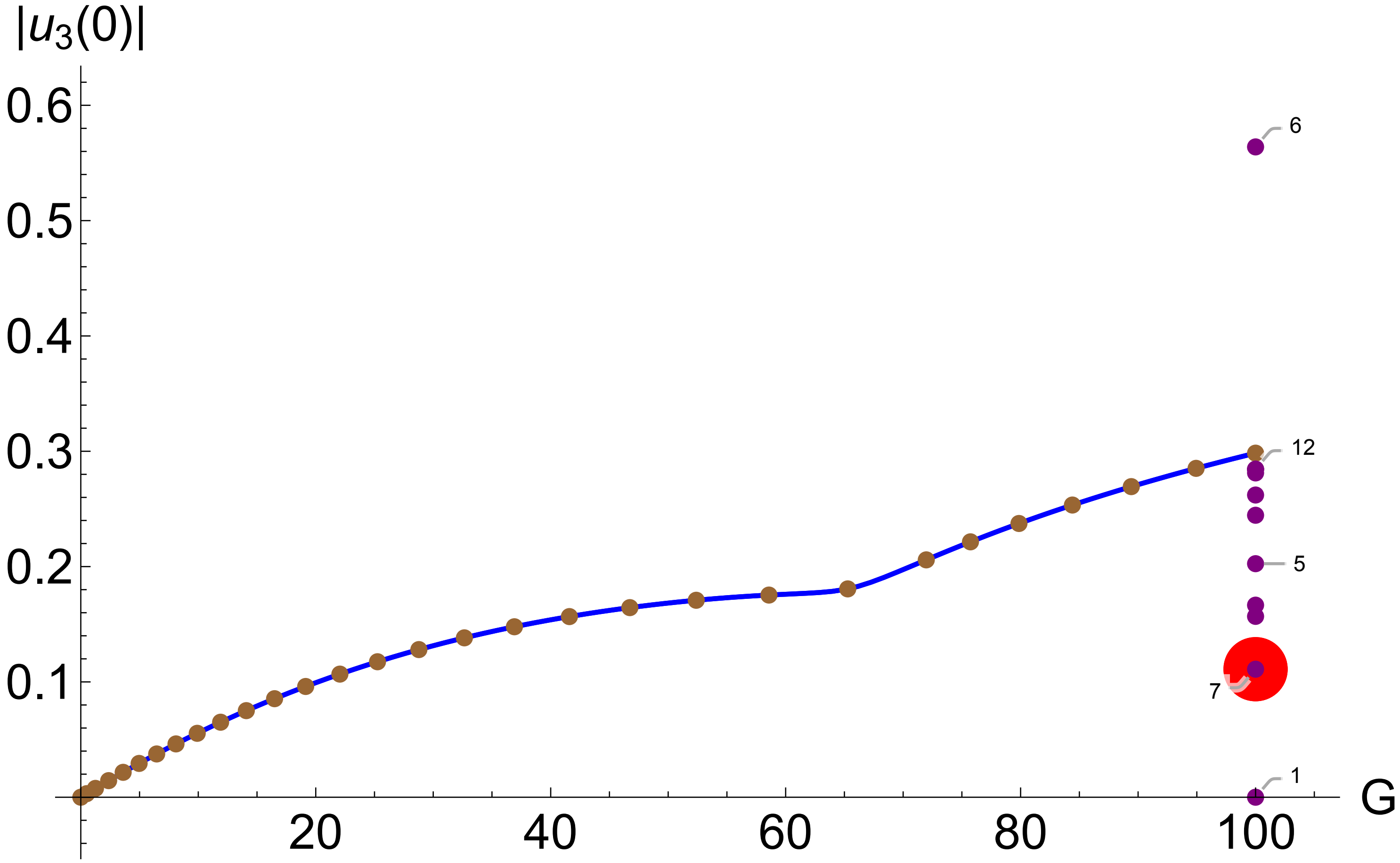


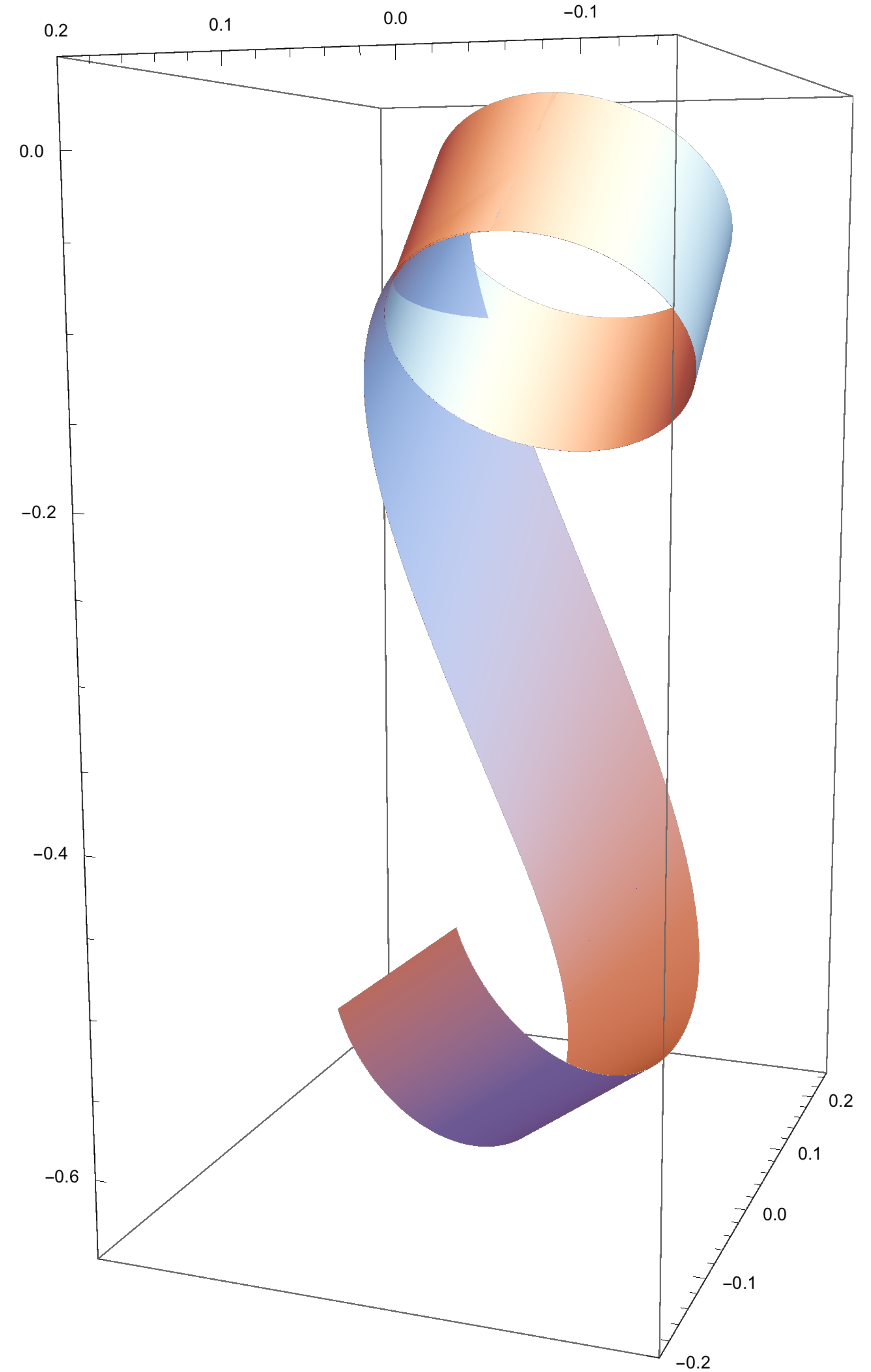
IPOPT: non equilibrium states



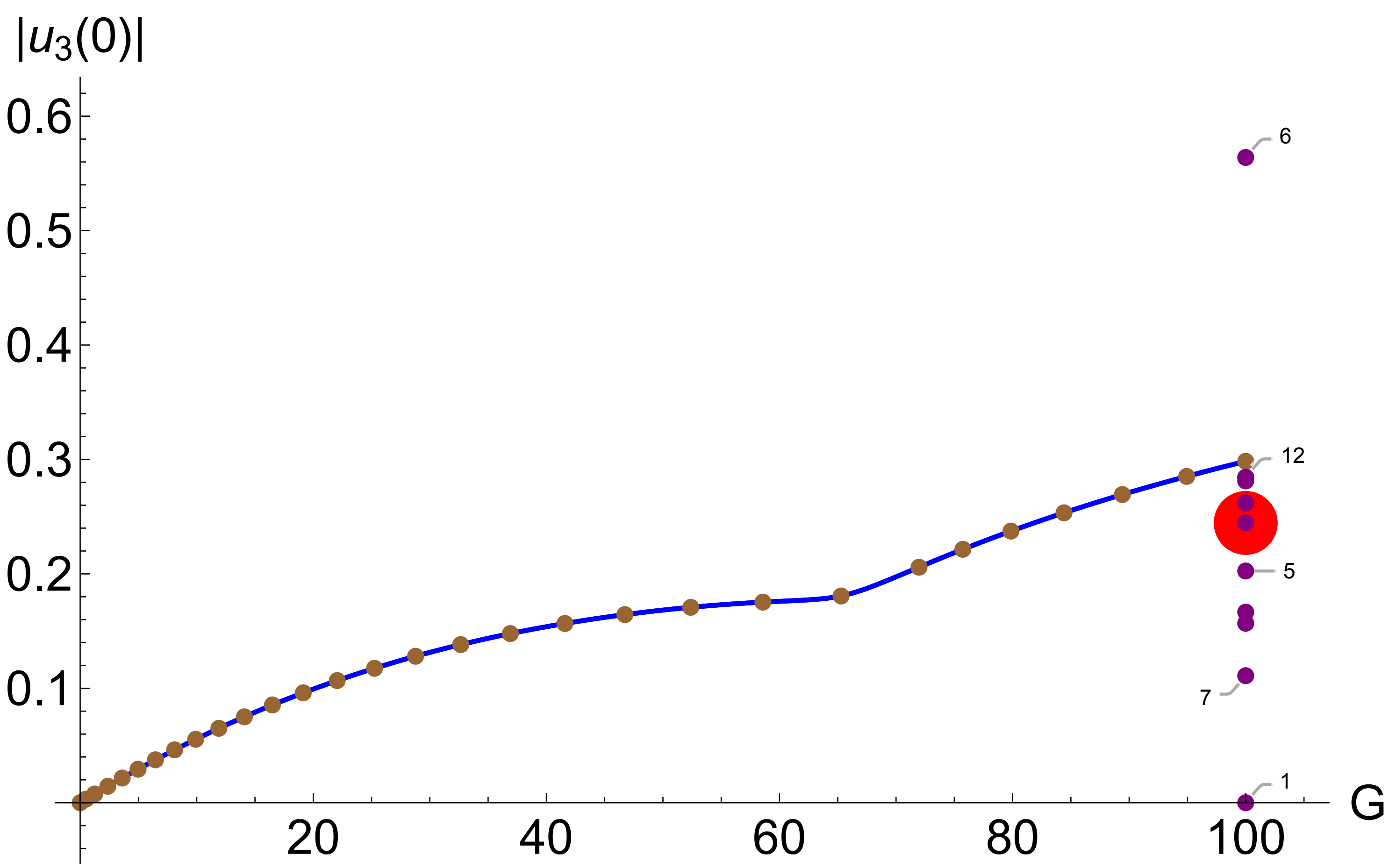


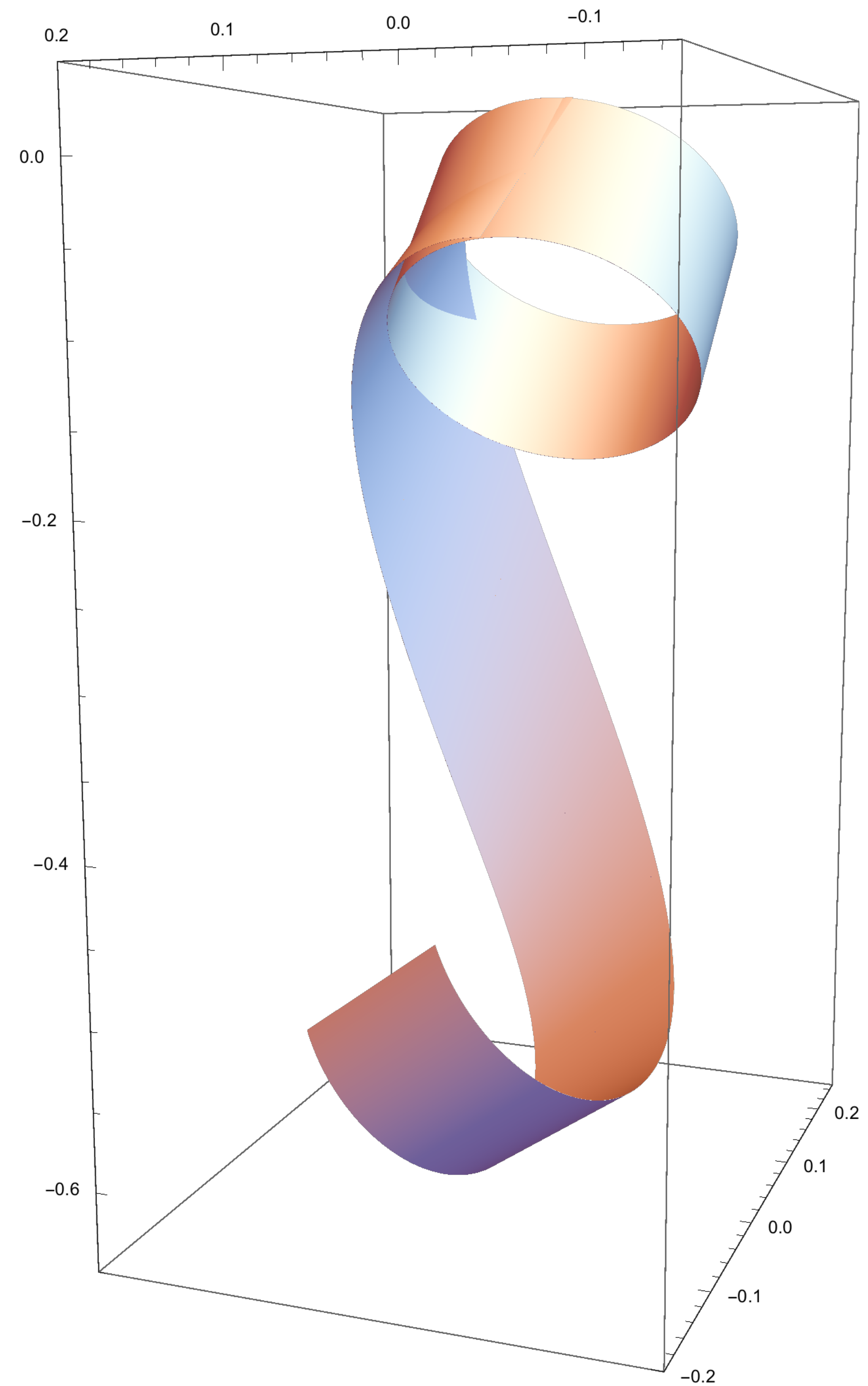
IPOPT: non equilibrium states



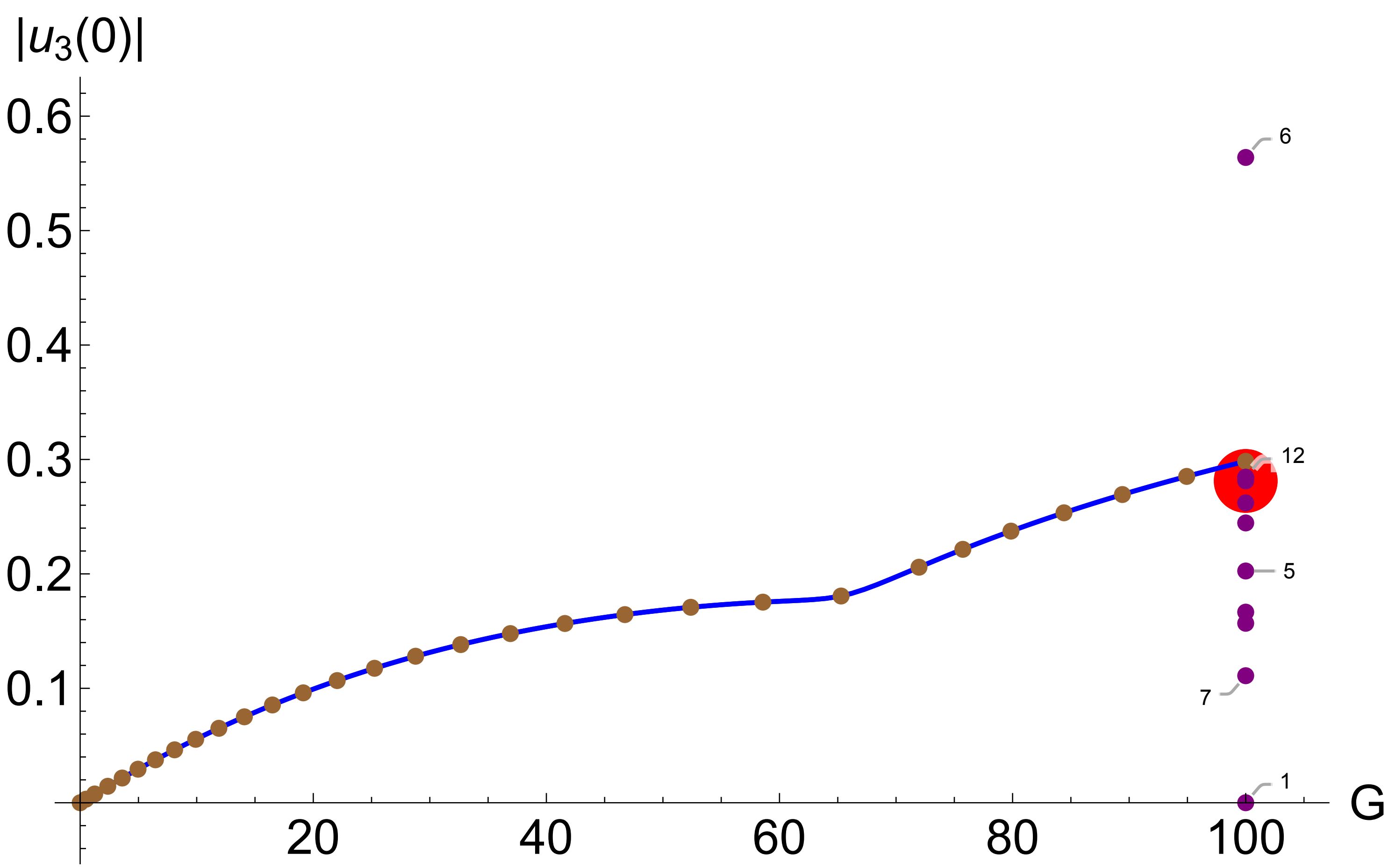


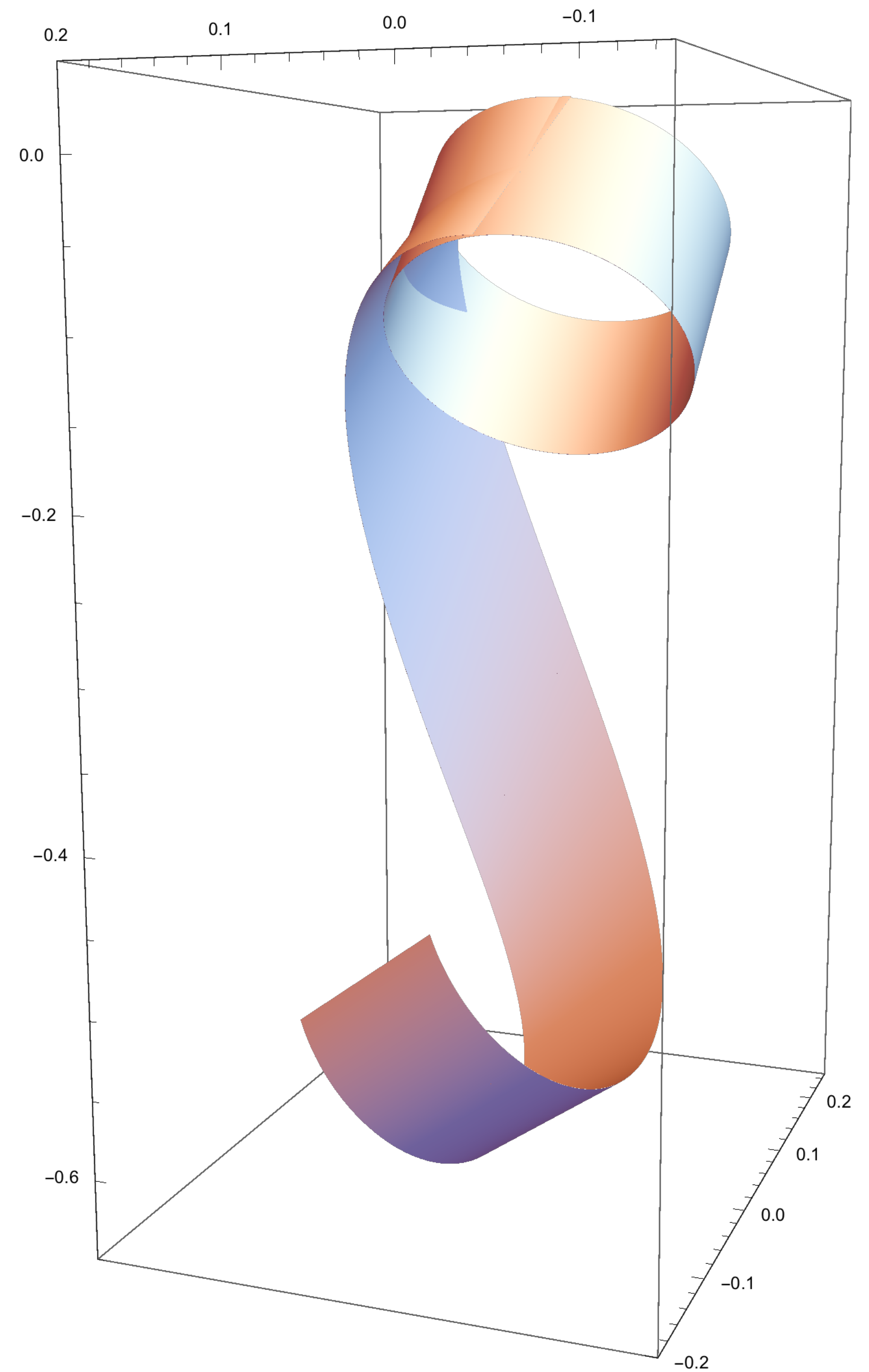
IPOPT: non equilibrium states



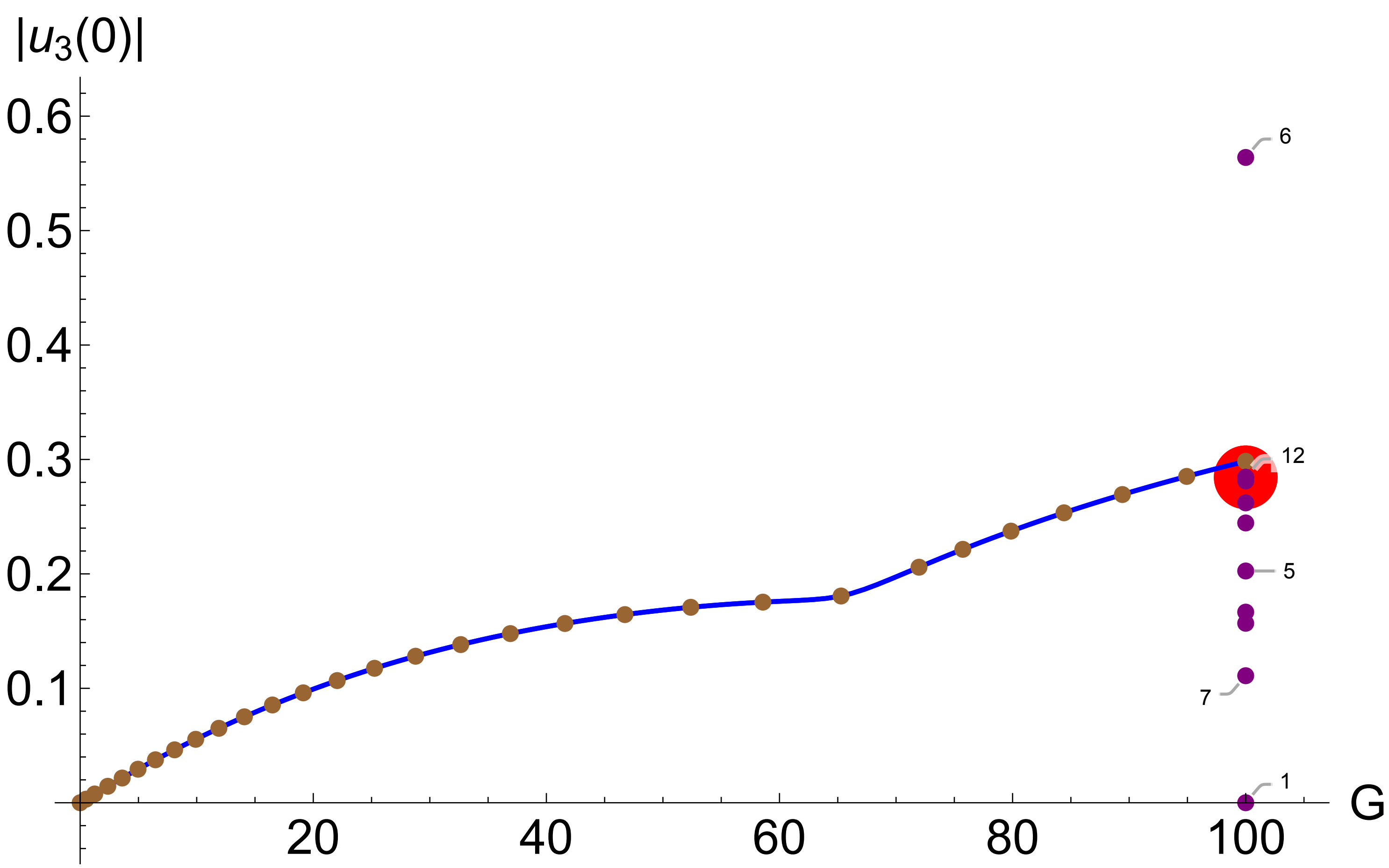


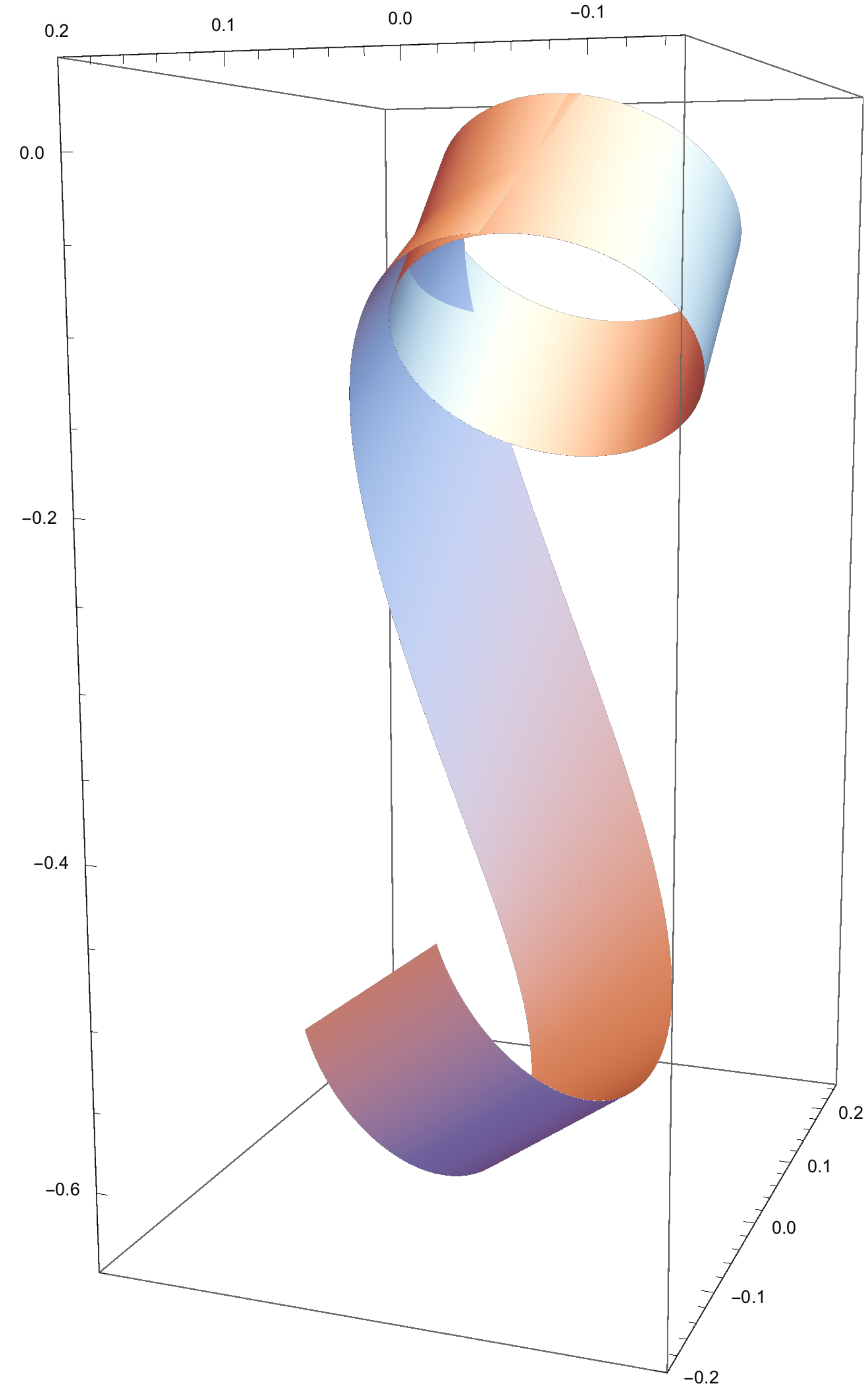
IPOPT: non equilibrium states



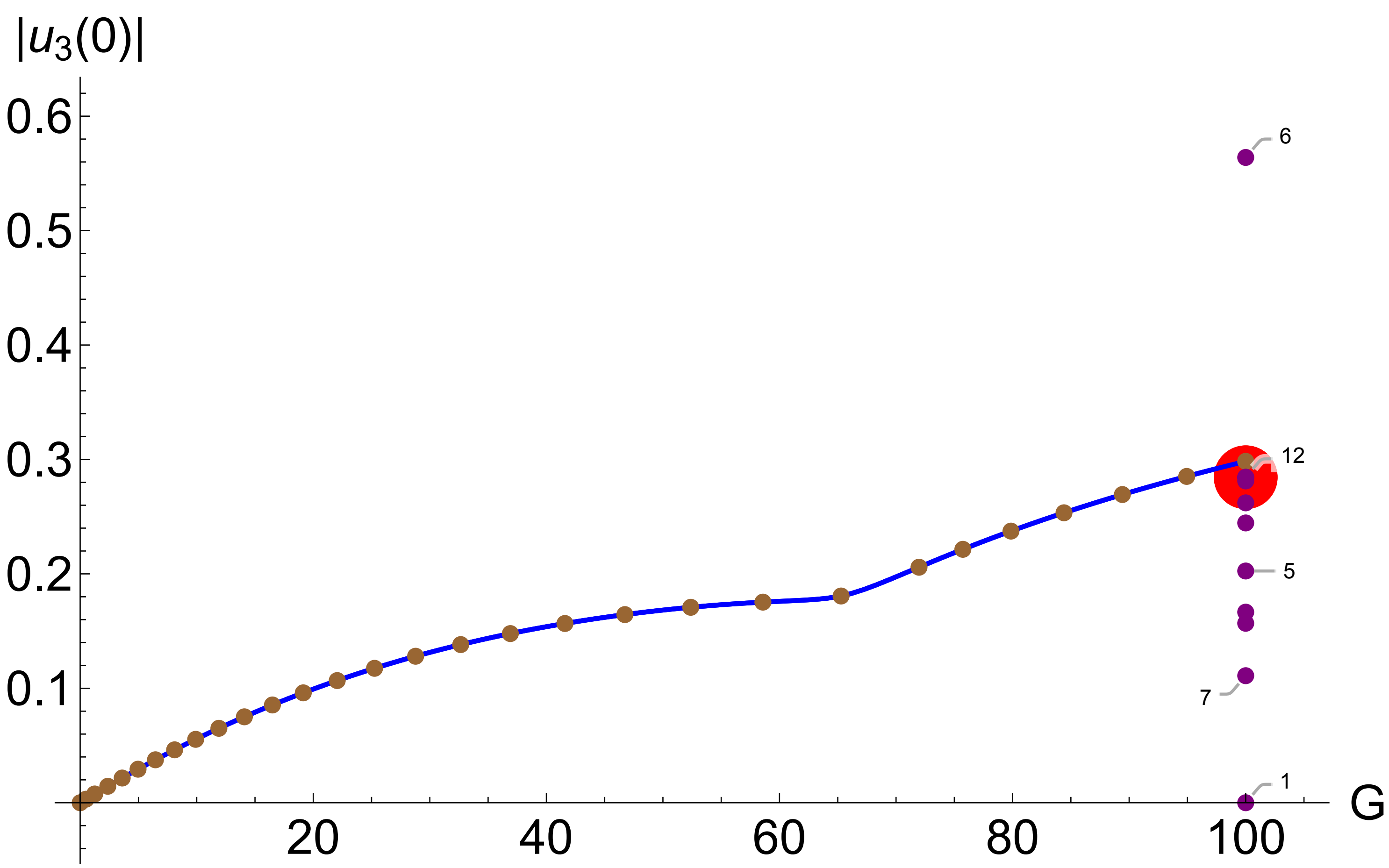


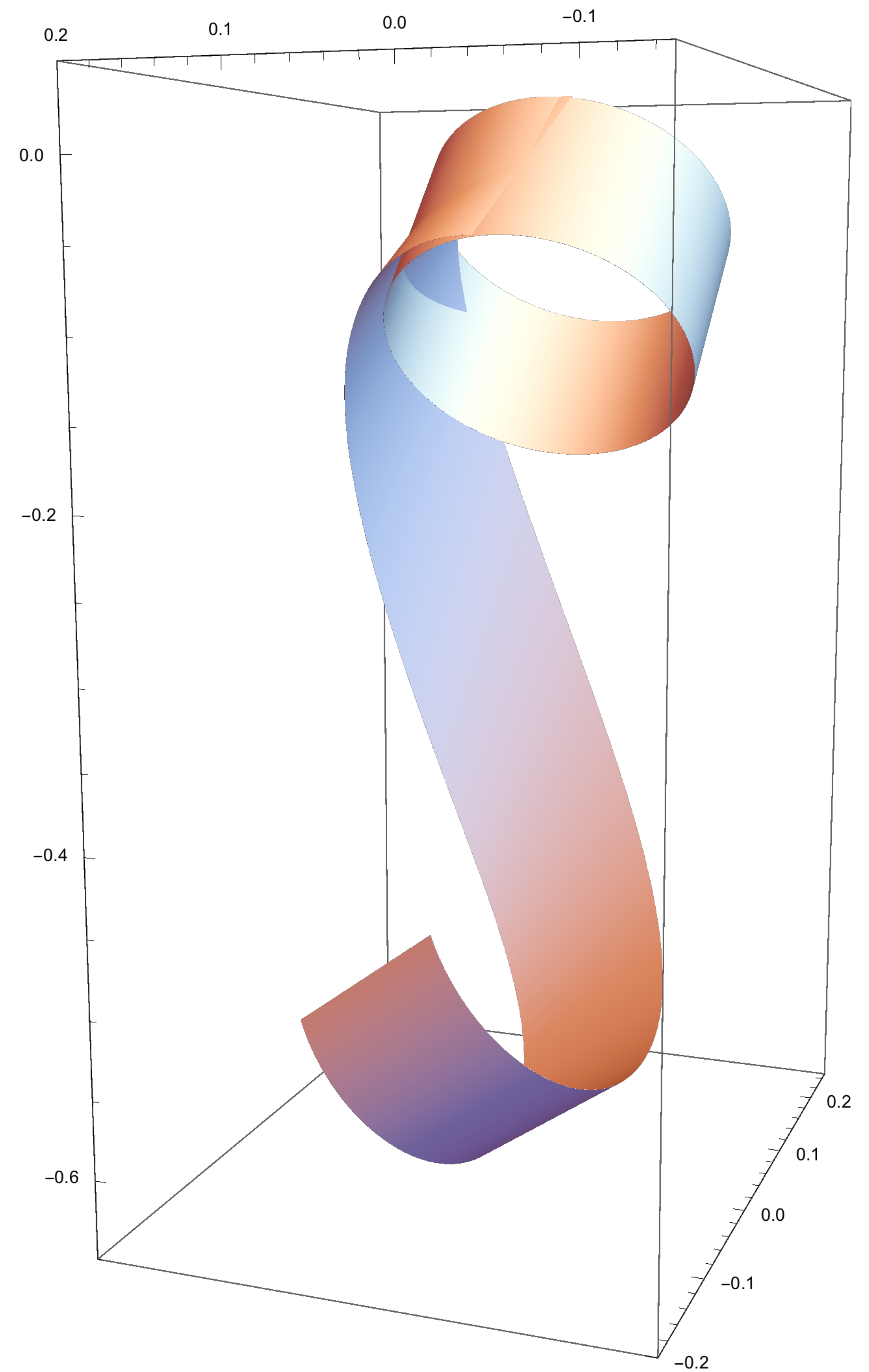
IPOPT: non equilibrium states



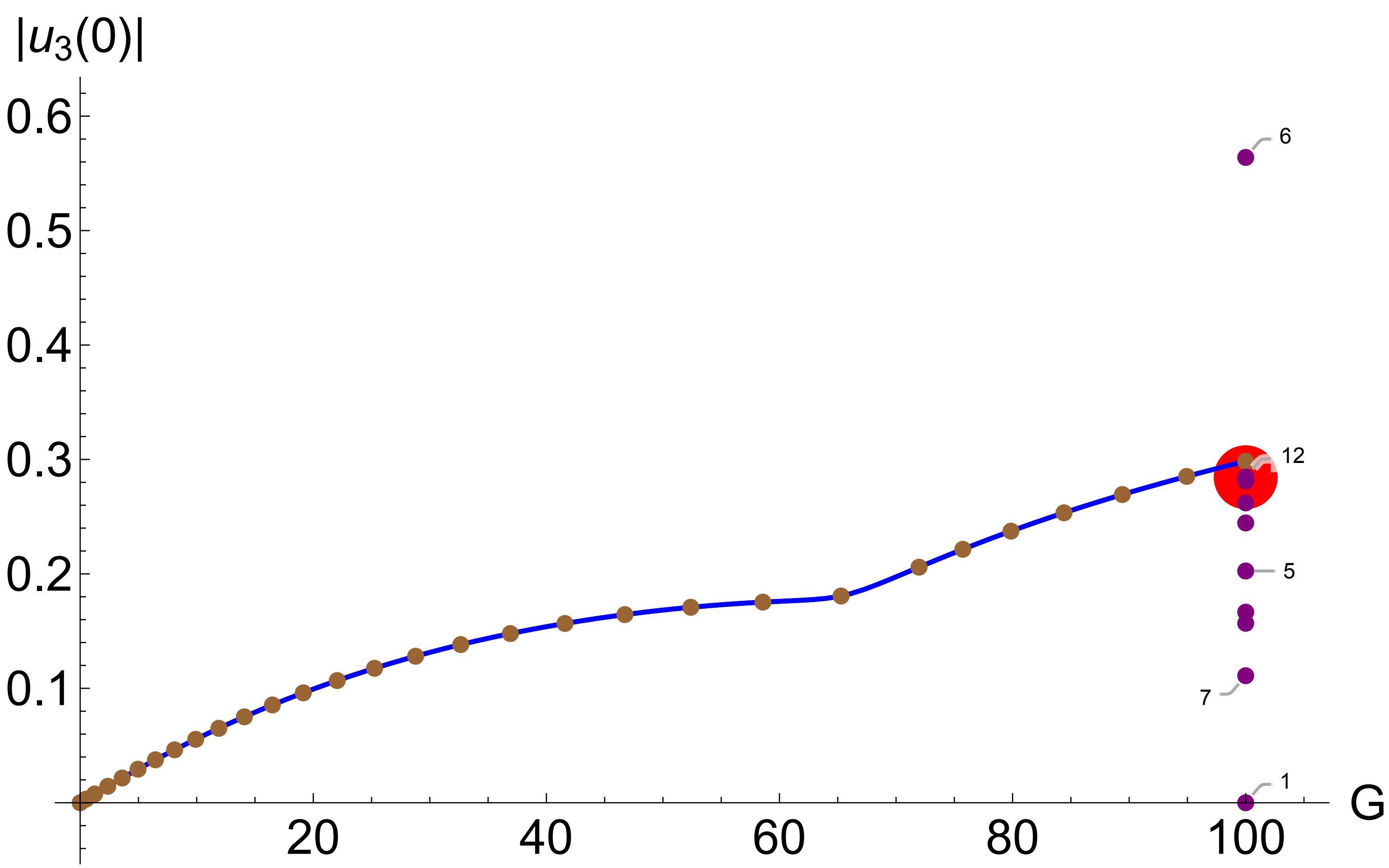


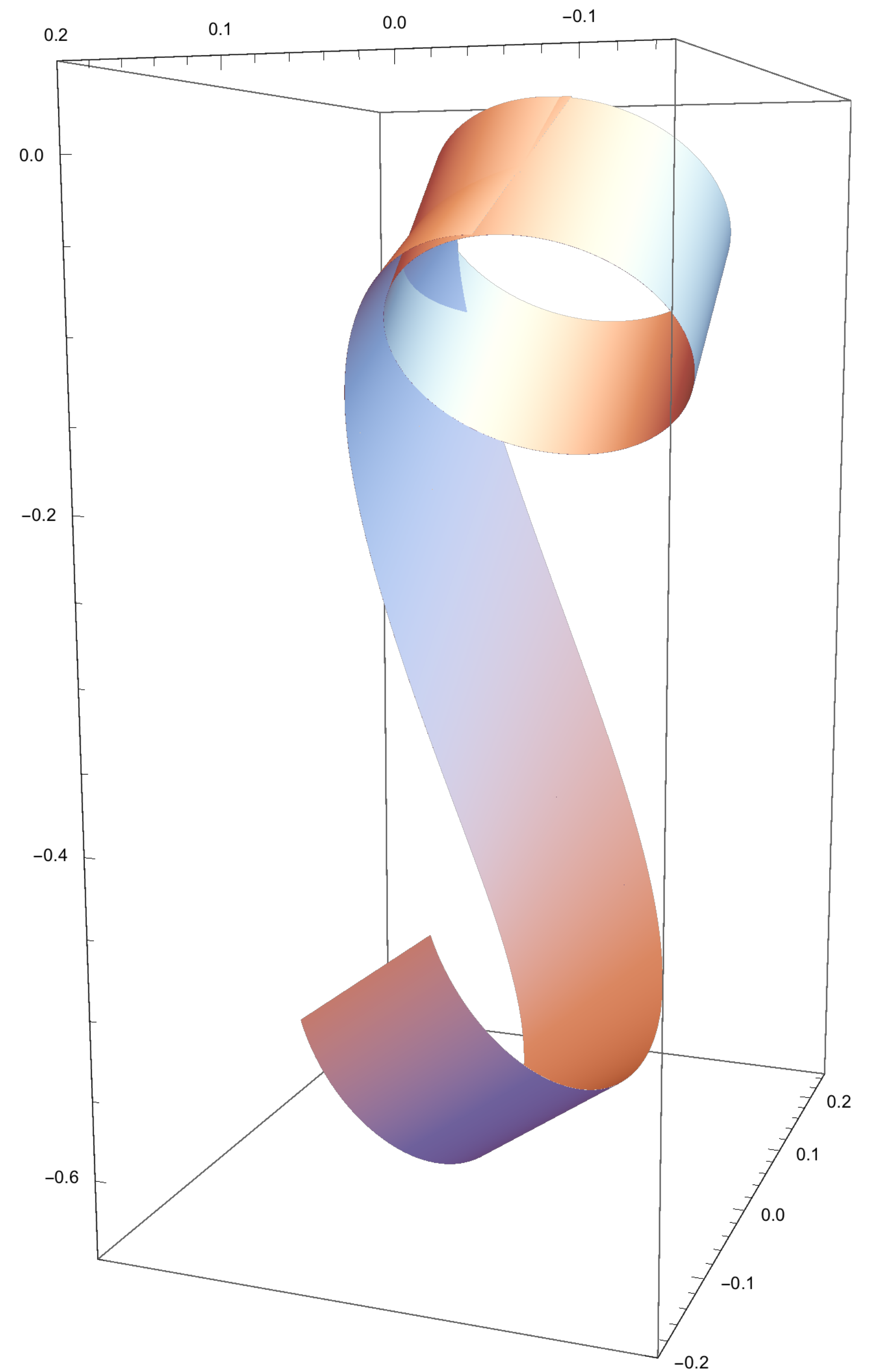
IPOPT: non equilibrium states





IPOPT: non equilibrium states





Conclusions:

Shooting: easy to set up, not robust
AUTO: fastest, command line
ManLab: interactive, no automatic discretization

Other algo: fenics, chebfun, bvpSolve (R)

ManLab in Python (jupyter lab) ?
ManLab compiled would be faster than AUTO

Fin

Beam on foundation

discretization is important: here non-centered finite-diff act like an imperfection