Elastic instabilities using `auto-07p`

S Ganga Prasath

1 Introduction

Elastic materials are characterised by their ability to undergo local dilation, shear and twist under externally applied load and relax to a neutral configuration when the load is removed. Often when the applied load exceeds a critical value these materials undergo instabilities characterized by a smooth or dramatic change in their morphology. These instabilities are captured by identifying the forces at play and deriving governing equations for the deformation of the material. The deformation field in the material is then related to the internal stress through a constituent relation, Hooke's law being a famous one. These elastic instability problems, as we shall see, are boundary value problems which are notorious to solve even numerically (analytical solutions are available only in isolated instances).

In this set of 4 tutorials we will see how to solve for the deformation field of the material under various active and passive forces and capture the instabilities using `auto-07p`, a numerical continuation package. The purpose of this tutorial really is to introduce you to the power of `auto-07p` to approach these problems. Though the documentation is a very good place to start, I realized during my learning process that there are a few important technical gaps that are often not discussed if one were to go from the documentation to implementation.

In this 4 part series we will be discussing problems each with an unique aspect in terms of implementation in `auto-07p`. It will become clear what I mean by this as we go over the examples.The problems we will be looking at are

- Bending of an *elastica*
- Buckling of a sheet
- Oscillating Beam
- Undulatory propulsion on land

2 `auto-07p` framework

The framework that we use to solve elastic instability problems has 3 major steps.

(*i*) To identify the model equations that describe the deformation field in elastic structure, derived either phenomenologically or from first principles. This is the part that captures the physics of the problem.

(*ii*) Using `auto-07p` to solve the system with the appropriate boundary conditions and continue the solution along a physical parameter in the system to see if the morphology changes with this parameter. `auto-07p` accepts first-order differential equations of the form

$$u'(t) = f(u(t), p), \quad f(.,.), u(.) \in \mathrm{R}^n,$$

where p are the parameters in the problem.If we have a higher order system, we need to convert and represent it in this form. `auto-07p` finds solution to this equations i.e. $f(u(t), p) = 0$ and will continue it along the different parameters p in the model. We will then look at the process of segregating the solution from the files that `auto-07p` spits out, post-processing them to understand the results.

(*iii*) Of course, the last step is to interpret the results and compare with experiments if any. And then go over to step (*i*) if there is mismatch between experiments and theoretical results.

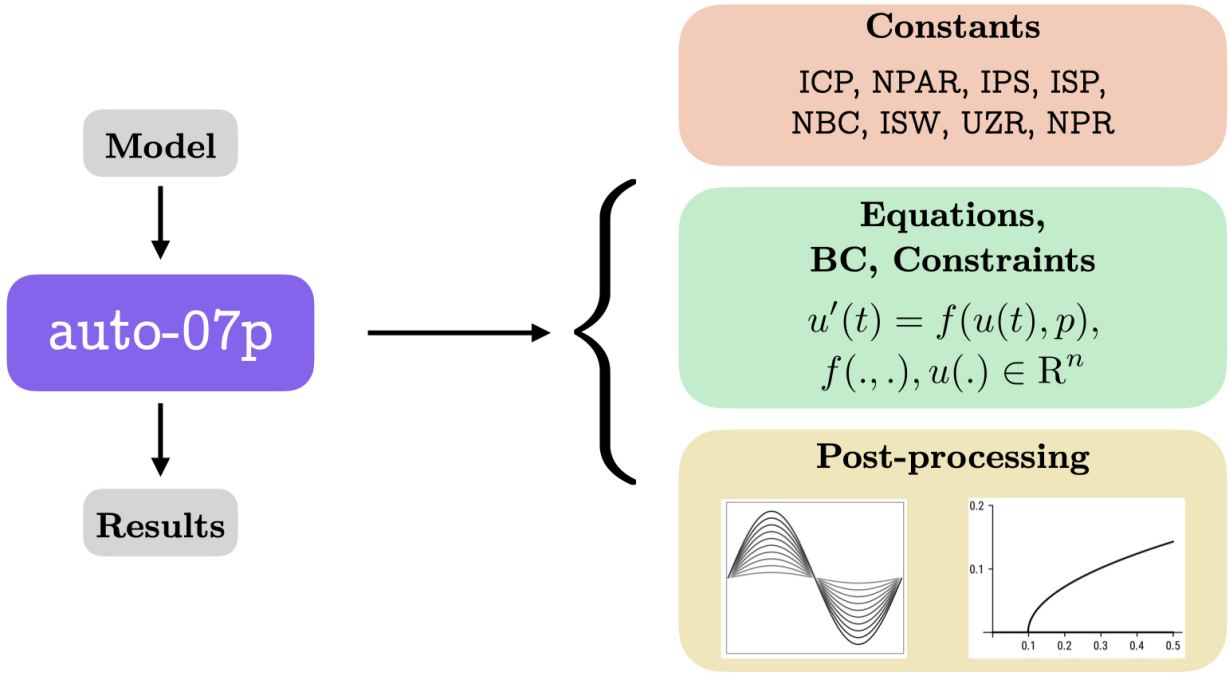
The step I will be focusing in these tutorials is (*ii*) where we assume that the equations required to describe the elastic structure are already available. We will then extract the bifurcation diagram as well as plot the solution files.

3 Bending of an *elastica*

Before we start the tutorial, it might be useful to note that `auto-07p` codes for all the examples are available in this link. The first problem we will solve in this tutorial is the deformation of an inextensible slender filament known as the *elastica*. We solve this in 2-dimensions where the structure can be represented only using curvature information along the filament (captured up to global translations and rotations). The concept of an elastica is very old and dates back to the times of Bernoulli and Euler (read more about it here). We consider an elastica hinged at one end while the other end experiences a point force $\vec{p} = (p_x, p_y)$. This problem is described in detail in Prof. Audoly's lecture notes. The elastic energy under an externally applied point force is

$$\mathcal{E} = \underbrace{\frac{1}{2}\int_0^L B\kappa^2(s)\ \mathrm{d}s}_{\text{Bending energy}} - \underbrace{\int_0^L (p_x \cos\psi + p_y \sin\psi)\ \mathrm{d}s}_{\text{Potential energy}}. \tag{1}$$

Here s is the arc-length and the first term in the above equation is the bending energy term while the next is the work done/stored potential energy due to the applied force. The work done is nothing but $-\vec{p}.\vec{r}(L)$ where $\vec{r}(L)$ is the end point. The equilibrium equation by



extremising this energy or equivalently the Euler-Lagrange equation is

$$B\psi''(s) - p_x \sin\psi + p_y \cos\psi = 0. \tag{2}$$

Since we have fixed end on one side, this translates to $\psi(0) = 0$ and a torque free boundary at the other side implies $\kappa(L) = \psi'(L) = 0$. This equation can be non-dimensionalized using L as the length-scale and (B/L) as the energy scale. After non-dimensionalization we get $\psi''(s) - p_x \sin\psi + p_y \cos\psi = 0$. The rescaled forces are $\tilde{p}_i = p_i/(BL^2)$, $i = x, y$ (we have dropped the tildes for simplicity).

In order to implement this equation in `auto-07p` we need to first convert it into a set of first order differential equations. We define $u_1 = \psi(s), u_2 = \psi'(s)$ and this results in:

$$u_1' = u_2, \tag{3}$$
$$u_2' = p_x \sin u_1 + p_y \cos u_1. \tag{4}$$

We know from geometry that $x'(s) = \cos\psi(s), y'(s) = \sin\psi(s)$, which can also be combined with the above equations to obtain a set of 4 equations. We supplement it with boundary conditions: $u_1(0) = u_2(1) = x(0) = y(0) = 0$.

3.1 Constants, `*.f90/*.c`, Scripts

For each problem we are interested in solving in `auto-07p` , there are at least 3 files required: (i) The constant file (which has a filename as c.*) that inform `auto-07p` of the details

Constants	Description	Values for elastica
ICP	Continuation parameters	p_x/p_y
NPAR	Number of parameters	4
IPS	Type of problem	4 for boundary value problems
ISP	Switch for bifurcation detection	2 will identify all kinds
NBC	Number of boundary conditions	4
ISW	Mode for branch switching	1 for normal, -1 for switching to different
UZR	User defined points for output	p_x/p_y
UZSTOP	User defined range for parameters	$p_x \in [-80, 20]$
NPR	Print & save every NPR steps	20

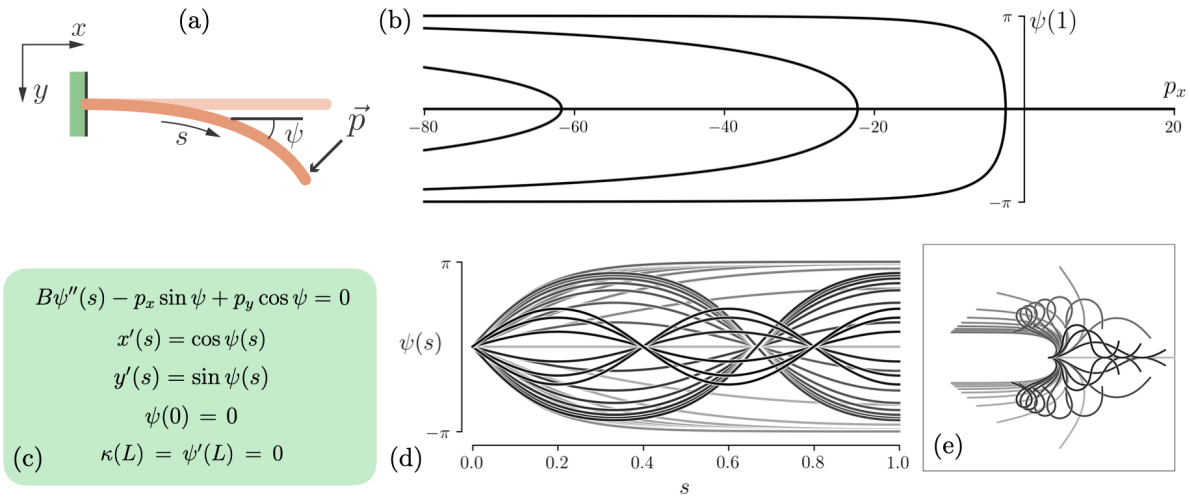
Table 1: List of constants in the `auto-07p` with their description and values based on problem of interest.

of the problem and the numerical details required for the package to prepare the solver. (*ii*) `FORTRAN/C` file with the governing equations, boundary conditions, integral constraints and the variables that need to be probed from the solution in order to plot them later to understand the solution. (*iii*) The most important file in this set is the `python` script that runs `auto-07p` with the specified constants. The script will help us span the phase space and continue along different solution branches.

In Table. **??** we note some of the most important constants that we will use in this tutorial set. A very useful cheat-sheet is in the manual under the title "Quick reference" of chapter 10. The `fortran` code `etica.f90` has the equations as a set of first order couple differential equations with (p_x, p_y) as the parameters to vary (we vary only one in this example). This file also has the initial solution for the specified parameters from which we need to continue in order to identify the bifurcations in the system. This is a subtle detail which needs to be kept in mind which is that you always need to know a solution to start continuation. In most cases, we are always able to find the trivial solution, like here $\psi(s) = 0, x(s) = s, y(s) = 0$ for $p_x = p_y = 0$. An important thing to note is that the function `PVLS` has 3 parameters that probe the value of $\psi(1), x(1), y(1)$.

```
etica = load('etica')
mu = run(etica)
mu = mu + run(mu,DS='-')
mu = mu + run(mu('BP1'),ISW=-1)
mu = mu + run(mu('BP1'),DS='-',ISW=-1)
mu = mu + run(mu('BP2'),ISW=-1)
mu = mu + run(mu('BP2'),DS='-',ISW=-1)
mu = mu + run(mu('BP3'),ISW=-1)
mu = mu + run(mu('BP3'),DS='-',ISW=-1)
# Relabel solutions
mu = relabel(mu)
# Save to b.mu, s.mu, and d.mu
save(mu,'mu')
# Plot bifurcation diagram
p = plot(mu)
```

Bending of an *elastica*




```
p.config(bifurcation_y=['psi(1)'])
#clean the directory
clean()
wait()
```

Listing 1: Python script to run `auto-07p` and continue solutions along different branches for the elastica problem.

The `python` script `etica.auto` runs the code and finds the bifurcation diagram for the specified set of parameters which we describe in detail now. We can run this code after installing `auto-07p` by simply typing `auto etica.auto` in the terminal. The first line in the script loads the files (`c.etica, etica.f90`) and the `run` command runs it for the specified constants. Since in the initial solution we start is for $p_x = 0$ and from the constants file we see that $p_x \in [-80, 20]$, running the code continues along the solution branch for $p_x \in (0, 20]$. We see that there are no bifurcations and we can now take a detour and change the direction of continuation by setting `DS='-'` which will continue now from $p_x = 20 \to -80$. We find there are 3 bifurcation points that are displayed as `BP`i, which stands for Branch Point (other solution types are described in chapter 6 in the manual) and $i = 1, 2, 3$ is the index of the branch point. We can now change the solution branch from the original branch, which is the straight configuration of the elastica, to the bent configuration using `ISW=-1` and continue till $p_x = -80$. We then continue along the negative direction to identify the solution when elastica is bent the opposite direction. We then move to the next branch starting at `BP2` and perform the same set of actions, similarly for `BP3`. All the results are then saved and can also be visualized using the plot interface. However we use the results saved in the `s.mu` to plot the solution. We briefly describe how this works and the implementation is provided in

the plotElastica.ipynb.

When we save the results, auto-07p generates 3 files b.mu, s.mu, d.mu corresponding to bifurcation diagram, solution and diagnostics. The bifurcation diagram has information very similar to what is displayed in the terminal when we run the program but with finer resolution depending on the specified parameters. The solution file on the other hand get saved every textttNPR steps as specified in the constant file. Further the solution file is a single file with all the solutions and thus has a specific format in which it is saved. The initial lines in the file have the parameter values at which the solution is saved followed by the solution itself. We process the bifurcation diagram and the solution using python package pandas which makes it easy to segregate and plot the data.

I am sharing here the format of the output file taken from this link, which has valuable information on how to process the files and will help make sense of the jupyter-notebook I have shared.

The files fort.8 or s.NAME have the following format:

```
first identifying line:
   ibr   :   the index of the branch
   ntot  :   the index of the point
   itp   :   the type of point
   lab   :   the label of the point
   nfpr  :   the number of free parameters used in the computation
   isw   :   the value of isw used in the computation
   ntpl  :   the number of points in the time interval [0,1]
       for which solution data are written
   nar   :   the number of values written per point
       (nar=ndim+1, since t and u(i), i=1,..,ndim are written)
   nrowpr:   the number of lines printed following the identifying line
       and before the next data set or the end of the file
       (used for quickly skipping a data set when searching)
   ntst  :   the number of time intervals used in the discretization
   ncol  :   the number of collocation points used
   nparx :   the dimension of the array par
following this are ntpl lines containing
   t u_1(t) u_2(t) ... u_ndim(t)
following this is a line containing the indices of the free parameters
   icp(i) for i=1,...,nfpr
followed by a line containing the derivative of parameters wrt arclength
   rl_dot(i) for i=1,...,nfpr
following this are ntpl lines containing the derivative of the solution wrt arclength
   u_dot_1(t) u_dot_2(t) ... u_dot_ndim(t)
followed by the parameter values
   par(i) for i=1,...,nparx
```

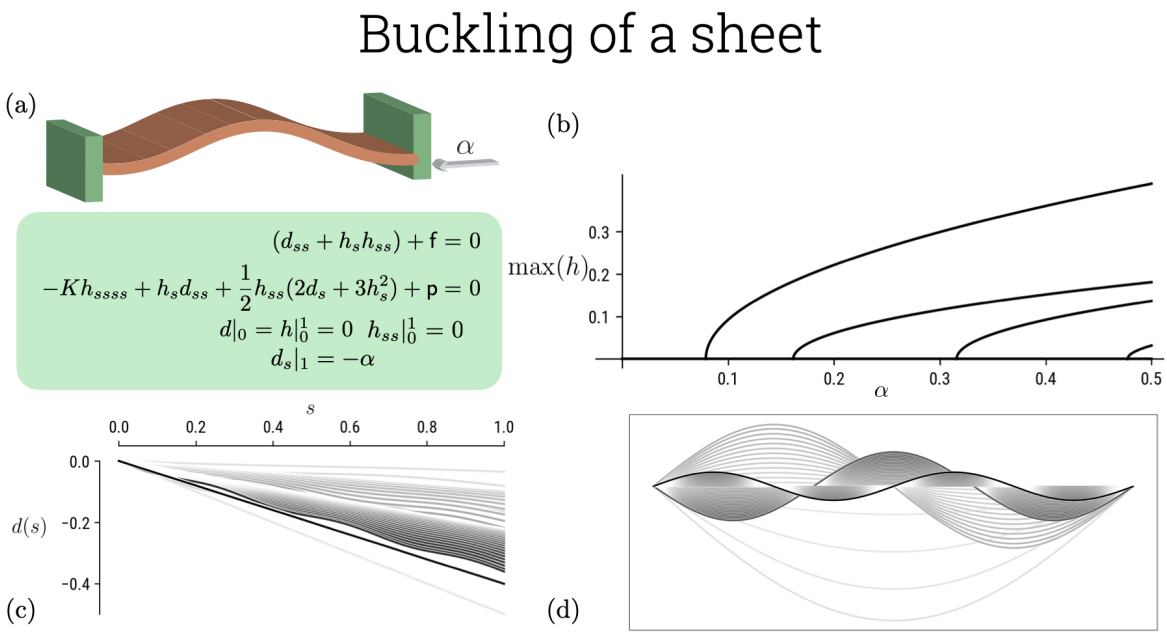
4 Buckling of a sheet

Föppl-von Karman equations in 2D captures the buckling of a sheet/strip/beam. The morphology of the sheet is captured using the deformation field along the in-plane direction is $u(x)$ and the displacement along the out-of-plane direction is $w(x)$. The governing equations for this deformation field is

$$S(u_{xx} + w_x w_{xx}) + f = 0, \tag{5}$$

$$-Bw_{xxxx} + S[w_x u_{xx} + \frac{1}{2} w_{xx}(2u_x + 3w_x^2)] + p = 0. \tag{6}$$

S is the stretching modulus, given by $S = E\xi$ and B the bending modulus is EI where I is the second moment of area of the sheet, where ξ is the thickness of the sheet, E the Young's modulus, f and p are the external forces per unit length.



We can non-dimensionalize the force balance equations using the body-length of the organism L as the length-scale and E the Young's modulus of the material as the stress-scale. Further we know that $S \sim E\xi, B \sim E\xi^3$ and using this we can then write the non-dimensional equations as

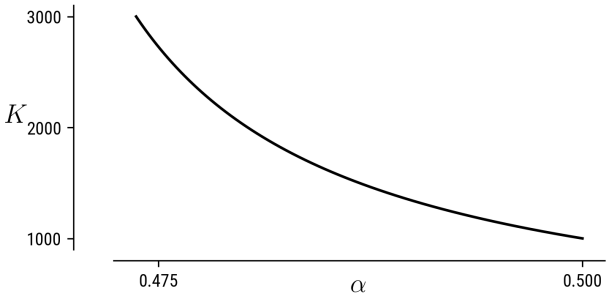
$$(d_{ss} + h_s h_{ss}) + \mathsf{f} = 0, \tag{7}$$

$$-Kh_{ssss} + h_s d_{ss} + \frac{1}{2} h_{ss}(2d_s + 3h_s^2) + \mathsf{p} = 0, \tag{8}$$

where $x = sL, u = dL, w = hL, K = (\xi/L)^2$ is the geometric quantity highlighting slenderness, also known as the von Kármán number, $\mathsf{f} = (fL/E\xi), \mathsf{p} = (pL/E\xi)$ are the non-dimensional forces. We can solve this system with the following boundary conditions: fixed

ends, $d|_0 = h|_0^1 = 0$; zero applied moments, $h_{ss}|_0^1 = 0$; applied tangential strain, $d_s|_1 = -\alpha$. We can again perform a similar analysis to that of the elastica for a fixed value of K and changing the applied strain and is shown in the figure below. The code corresponding to these plots is in the `1parameter` folder inside Tutorial2_FvK.

The leverage we have using the `.auto` script is that we can now steer the solution branch by varying different parameters in the equation. Once you have shifted from the unbuckled configuration to a bucked one using `ISW=-1` (just as in the elastica case), we can change the continuation parameter to a different one, say K just by manipulating the `.auto` script. We can go further using `auto-07p` and continue the solution by varying both (K, α) by using the variable `ICP` and setting limits using the `UZSTOP` command. For example we continue along two variables here by adding an integral constraint. This is because we need to satisfy the constraint `NCONT = NBC+NINT-NDIM+1` where `NCONT`-number of continuation parameters, `NINT`-number of integral constraints, `NDIM`-number of dimensions. In the current problem we have `NBC=6, NDIM=6` and thus we need an integral constraint to be able to perform continuation along the two-parameter phase space. After performing this, we get a isocontours in (K, α)-plane representing morphologies of the sheet (a sample is shown in the figure below).



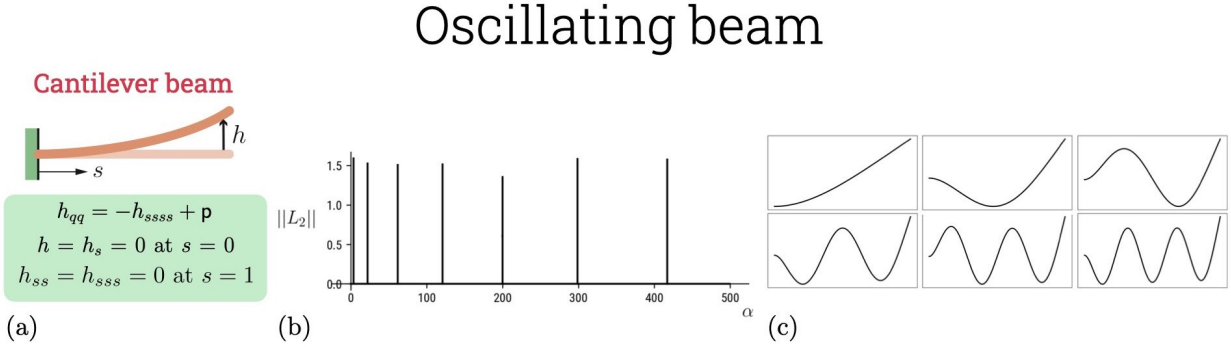
5 Oscillating beam

The small deflection theory capturing the morphology of an elastic beam is given by the Euler-Bernoulli equation which reads as

$$\alpha w_{tt} = -B w_{xxxx} + p. \tag{9}$$

Here $w(x)$ is the vertical displacement along the beam's length, α is the mass per unit length, $B = EI \sim E\xi^4$ is the bending modulus, E the Young's modulus, I the second moment of area, ξ is the beam thickness and p is the body force per unit length. We can non-dimensionalize the equation with beam length L as the length scale, $T \sim (\alpha L^4/B)^{1/2}$ as the time-scale and E as the stress-scale. Rewriting $x = sL, t = qT, w = hL$, we can write the non-dimensional equation as

$$h_{qq} = -h_{ssss} + \mathsf{p}. \tag{10}$$



Here the non-dimensional body force $\mathsf{p} = (pL^3/B)$. We are interested in solving this system for the boundary conditions: (i) cantilever beam, $h = h_s = 0$ at $s = 0$, $h_{ss} = h_{sss} = 0$ at $s = 1$, with $\mathsf{p} = 0$, (ii) unsupported beam, $h_{ss} = h_{sss} = 0$ at $s = 0, 1$ with $\mathsf{p} = \beta \sin(2\pi s)$.

Though `auto-07p` is capable of solving discretized PDEs, we are interested in solving an eigenvalue problem which is coherent with our requirement to find oscillating solutions to the beam displacement. This can be done by substituting $h(s, q) = \Re[\hat{h}(s)\mathrm{e}^{i\omega q}]$ into the above equation, we then get

$$\omega^2 \hat{h} = \hat{h}_{ssss} - \hat{\mathsf{p}}. \tag{11}$$

For the cantilever beam we have the frequency of oscillation as the continuation parameter and this eigenvalue problem can be solved with the specified boundary condition in `auto-07p`. This is shown in the figure attached. The interesting aspect of the second problem of an unsupported beam is that the boundary conditions are all derivatives in h and thus do not provide a unique solution. One often needs to introduce pinning conditions to select form infinite translationally invariant solution. In `auto-07p` we do this by introducing a body forcing in the governing equation to get

$$\omega^2 \hat{h} = \hat{h}_{ssss} - \hat{\mathsf{p}} + \lambda \hat{h}. \tag{12}$$

and start the continuation with $\lambda = 1$ from which point we continue towards $\lambda = 0$. After we do this, we are free to continue along any direction as the initial solution is pinned and we have found one of the many solutions for $\lambda = 0$. In the code attached `Tutorial3_EB`, we can have explored the solution for $\omega = 1$ and continued the solution for different β. This technique of continuing along one branch and then shifting to a different branch is called *homotopy* continuation. The boundary condition is not really physical here and the solution is not interesting either but I chose this condition to illustrate this technique which is super useful in a variety of scenarios.

6 Undulatory propulsion on land

In this last tutorial, an attempt at coup de grace, we will be combining various techniques we have learnt in this tutorial and apply to the problem of locomotion on land by an active

filament aka snake. Solving the problem in `auto-07p` is challenging as there are several parameters in the system. Further we need to impose periodic boundary conditions, which we have not yet seen in this series. The problem is solved by Guo and Maha, PNAS 2007 where they describe the shape of a snake based on the body centerline. Please refer to the article for a detailed derivation, the equations read

$$x_s = \cos\theta, \tag{13}$$
$$y_s = \sin\theta, \tag{14}$$
$$\theta_s = \kappa, \tag{15}$$
$$T_s = \mu_w + \mu_p \mathrm{Pr}|\sin\theta| - \mathrm{Mo}\frac{\cos(2\pi s)}{2\pi} - \mathrm{Be}\kappa_s\kappa - \mathrm{Vi}\kappa_{ss}\kappa, \tag{16}$$
$$0 = -\mathrm{Pr}\sin\theta + \mathrm{Mo}\sin(2\pi s) + \kappa T - \mathrm{Be}\kappa_{ss} - \mathrm{Vi}\kappa_{sss}. \tag{17}$$

Here x, y are the position of the centerline, s is the arc-length, θ is the orientation of the tangent along the body, κ is the curvature and T is the tension. There are 6 parameters in the system and they are Pr, Mo, Be, Vi, μ_w, μ_p. The boundary conditions that supplement these equations are of two types, dirichlet and periodic. The dirichlet ones are: $x(0) = y(0) = y(1) = \theta(0) = \theta(1) = 0$, while the periodic ones include: $T(0) = T(1), \kappa(0) = \kappa(1), \kappa_s(0) = \kappa_s(1)$. As we have seen in our earlier tutorial, `NCONT = NBC+NINT-NDIM+1`. We have `NDIM = 7 NBC=8`, so we have two continuation parameters. We would like to find Mo vs Vi, just as in the paper.

In order to do that we split the task into two parts (problem is too hard to solve in one go). The first part is to enforce periodic boundary condition for most but not all the variables (this give confidence in the solution itself), continue from a numerically easily accessible solution to a point close to the actual region of interest. In the second part enforce the last periodic boundary condition that was not supplied earlier. In effect the first part ensure we are continuing along one branch by varying only one parameter and in the second we use homotopy continuation to asymptotically satisfy the last boundary condition. The itemize the steps taken and `Snake.auto` has the implementation of the method described here.

- Initialize: $x(s) = s, y(s) = \theta(s) = \kappa(s) = \kappa_s(s) = \kappa_{ss} = T(s) = 0$.
- Choose starting parameters: Pr $= 0.18$, Mo $= 0.0$, Be $= 0.4$, Vi $= 1.0$, $\mu_w = 0.0$, $\mu_p = 0.2$. The starting parameters are close to the one in the Guo and Maha's paper except for Mo, μ_w and Vi. These parameter we shall continue to the exact parameters in the paper. I chose these parameters upon some experimentation to see to what extent `auto-07p` is stable when we start with a trivial guess for snake shape we have chosen.
- Enforce boundary conditions: $x(0) = y(0) = y(1) = \theta(0) = \theta(1) = 0$ and $T(0) = T(1), \kappa(0) = \kappa(1)$. We have skipped $\kappa_s(0) = \kappa_s(1)$, which we shall satisfy using homotopy continuation. As have `NBC=7`, we can continue only along one parameter. We shall add the additional boundary condition at the end and identify Mo vs Vi.

- We pin the solution by the same technique we discussed earlier by defining a new parameter λ and set it to 1 initially which we will continue to 0 later. The equations them become:

$$T_s = \mu_w + \mu_p \mathrm{Pr}|\sin\theta| - \mathrm{Mo}\frac{\cos(2\pi s)}{2\pi} - \mathrm{Be}\kappa_s\kappa - \mathrm{Vi}\kappa_{ss}\kappa - \lambda T, \tag{18}$$

$$0 = -\mathrm{Pr}\sin\theta + \mathrm{Mo}\sin(2\pi s) + \kappa T - \mathrm{Be}\kappa_{ss} - \mathrm{Vi}\kappa_{sss} - \lambda\kappa. \tag{19}$$

  This is done because the periodic boundary condition provides only translationally invariant solutions.

- We start the continuation by varying Mo which has the body force and see that the shape of the snake change as we increase the amplitude.

- We then continue from Mo = 100 by varying $\lambda \to 0$.

- We then vary μ_w, followed by varying Vi i.e. $\mu_w = 0.1$, Vi = 0.03 to the actual parameters in the paper.

- The most important step and really the crux of this tutorial is now to enforce the new boundary condition $\kappa_s(0) = \kappa_s(1)$. This is done by having a new equations file (we call `SnakeFull.f90`) with the additional boundary condition added but with an additional parameter: $\kappa_s(0) - \kappa_s(1) + $ `PAR(8)` $= 0$. When `PAR(8)` $\to 0$, we satisfy the boundary condition exactly. We need to also supply a new constants file with `NPAR, NBC` updated.

- We can continue along Vi with Mo as the free parameter once we have satisfied all the boundary conditions and obtain the figure in the article which we show below.

Guo and Maha PNAS 2007

Undulatory propulsion on land

(a)

$$x_s = \cos\theta, \quad y_s = \sin\theta, \quad \theta_s = \kappa,$$

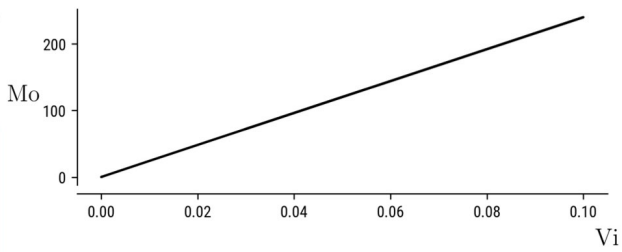
$$x(0) = y(0) = y(1) = \theta(0) = \theta(1) = 0$$

$$T(0) = T(1), \kappa(0) = \kappa(1), \kappa_s(0) = \kappa_s(1)$$

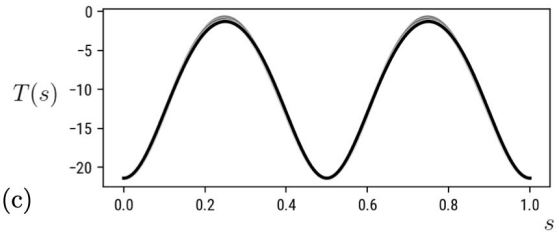
$$T_s = \mu_w + \mu_p \mathrm{Pr}|\sin\theta| - \mathrm{Mo}\frac{\cos(2\pi s)}{2\pi} - \mathrm{Be}\kappa_s\kappa - \mathrm{Vi}\kappa_{ss}\kappa,$$

$$0 = -\mathrm{Pr}\sin\theta + \mathrm{Mo}\sin(2\pi s) + \kappa T - \mathrm{Be}\kappa_{ss} - \mathrm{Vi}\kappa_{sss}.$$

(b)



(c)



(d)