Solving for solutions in Auto-07p

This file explains how to solve the boundary value problem (BVP) described in Section 3.4 of the paper using Auto-07p [1], a powerful differential equation solver that uses the arc length continuation.

There are two cases discussed in Section 5 of the paper: the preferred long wavelength case and the short wavelength case. We have two folders corresponding to these cases in this directory:

1. **Preferred Long Wavelength**
2. **Short Wavelength**

In each folder, there are two sub-folders corresponding to respective solution scenarios:

- **OnePerversionSolutions**
- **TwelvePerversionSolutions**

These sub-folders contain the files required for solving the BVP. The computed solutions are shown in the videos in the parent Videos-ResultsFromNumericalSimulations folder.

STEPS FOR OBTAINING SOLUTIONS

The process of obtaining solutions involves two main steps:

- **Solving the BVP in Auto-07p**
- **Post-processing the computed solutions**

We will explain these steps using the preferred long wavelength one-perversion case as an example. The procedure is similar for the other cases. Each subfolder contains Auto-07p files and solutions are stored in the respective Output.zip. You can use the long wavelength one-perversion case as a guide for post-processing solutions for other cases.

1. SOLVING THE BVP IN AUTO-07P

To solve the BVP, follow these instructions:

- Go to the folder:
  `/PreferredLongWavelength/OnePerversionSolutions`
- Open Auto-07p and run the solve.py script by executing the command:
  `execfile('solve.py')`

The solve.py script depends on two .f90 files, which set up the equilibrium equations, boundary conditions, and initial values for parameters. Specifically, Auto-07p requires us to define the differential equations as a first-order system in the subroutine FUNC in the .f90 files.

The table below explains the variables used in the .f90 files and their corresponding quantities in the paper:

variable used in the .f90 files	quantity in the paper
theta	θ
tau	τ
dTau	$\frac{\mathrm{d}\tau}{\mathrm{d}S}$
d2Tau	$\frac{\mathrm{d}^2\tau}{\mathrm{d}S^2}$
d3TauByl	$\frac{1}{\ell}\frac{\mathrm{d}^3\tau}{\mathrm{d}S^3}$
M	M

Also, we use variable Su $\in [0,1]$ as the independent variable for setting up the differential equations, where we define Su=S/l. The seventh differential equation in the subroutine FUNC along with boundary seventh boundary condition in the subroutine BCND specifies Su.

There are three parameters in the FUNC subroutine:

1. epsilon (ε in the paper) controls the separation between ribbon ends.
2. mExt is the amplitude of the external sinusoidal moment used for the guess step explained below.
3. The third parameter, defined in subroutine PVLS, measures $\|\tau\|_{\mathrm{rms}}$ in the computed solutions. We refer to it as l2Tau in the constants file c.guessStep and the script file solve.py.

The first-order differential equations are expressed in the form `F(i) = ...` in the `.f90` files, where:

		corresponding quantity in the paper
F(1) =	$\frac{\mathrm{d}\,\mathtt{theta}}{\mathrm{d}\,\mathtt{Su}}$	$= \ell\tau$
F(2) =	$\frac{\mathrm{d}\,\mathtt{tau}}{\mathrm{d}\,\mathtt{Su}}$	$= \ell\frac{\mathrm{d}\tau}{\mathrm{d}S}$
F(3) =	$\frac{\mathrm{d}\,\mathtt{dTau}}{\mathrm{d}\,\mathtt{Su}}$	$= \ell\frac{\mathrm{d}^2\tau}{\mathrm{d}S^2}$
F(4) =	$\frac{\mathrm{d}\,\mathtt{d2Tau}}{\mathrm{d}\,\mathtt{Su}}$	$= \ell\frac{\mathrm{d}^3\tau}{\mathrm{d}S^3}$
F(5) =	$\frac{\mathrm{d}\,\mathtt{d3TauByl}}{\mathrm{d}\,\mathtt{Su}}$	$= \frac{\mathrm{d}^4\tau}{\mathrm{d}S^4}$ (expression solved from eq. 3.10 in the paper)
F(6) =	$\frac{\mathrm{d}\,\mathtt{M}}{\mathrm{d}\,\mathtt{Su}}$	$= 0$ (the equilibrium equation)

Loading procedure. Using analytical calculations based on the dispersion relation in equation 4.5 of the paper, we anticipate that buckling points may be observed close to each other as we decrease the separation between ribbon ends. For example, in the preferred long wavelength case, the first twelve buckling points may occur between $\varepsilon = -29.04$ and $\varepsilon = -31.02$ as per Figure 5.3(a).

Owing to the difficulty in finding and selecting a solution branch corresponding to a specific number of perversions, we use the following procedure as echo in `solving.py`:

1. **Guess Step**: We first apply a sinusoidal external moment to create the desired number of perversions, in this case one perversion. This loading step, called `guessStep` in `solving.py`, increases the moment amplitude from 0 to 100. Then, decrease the separation at a fixed moment until the ribbon ends is well below corresponding buckling strain predicted for the desired mode. For example, decrease separation until $\varepsilon = -45.9$. And then set the external moment to zero.
2. **Changing Separation**: With one perversion ribbon state at hand, we next perform a continuation in ε to obtain the solution curve. We adjust constants and step sizes to ensure convergence. These values are specified in `solve.py`. For explanations of various constants in `solve.py` and `c.guessStep`, refer to section A.2 of [2].

2. POST-PROCESSING THE COMPUTED SOLUTIONS

After executing the script solve.py, we should see the following `l2Tau` vs `epsilon` plot on the screen.

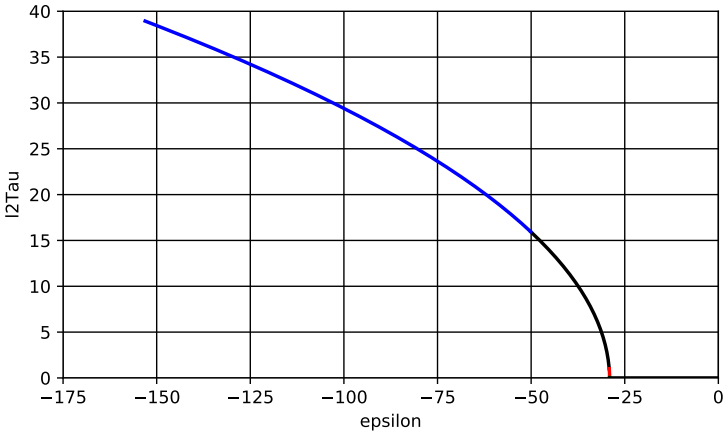

Figure 2.1. `l2Tau` vs `epsilon` plot that appears after the execution of script solve.py.

This plot corresponds to Figure 5.1 (with a negative sign for ε in the paper). Towards the end of the end of the script solve.py, we save `b.changeSeparation` file that stores the details of epsilon's increments used to compute the figure, and `s.changeSeparation` that stores the solved ribbon states.

The output from our `Auto-07p` implementation is located inside the `Output.zip`. You can unzip the folder and use the output if you want to circumvent solving the equations in `Auto-07p` and want to directly use the solutions we got. For more details on how to interpret the columns in `s.changeSeparation`, refer to section A.3 of [2].

For further post-processing, we use the Mathematica files in the `Postprocessing` folder. We will use `auto07pfilesParser.m` to import data from s.changeSeparation. We plot twist for a ribbon sate at $\varepsilon = -50$ in the Mathematica file.

BIBLIOGRAPHY

[1] E. Doedel, A. R. Champneys, F. Dercole, T. F. Fairgrieve, Y. A. Kuznetsov, B. Oldeman, R. C. Paffenroth, B. Sandstede, X. J. Wang, and C. H. Zhang. Auto-07p: continuation and bifurcation software for ordinary differential equations. 2007.

[2] B. Sandstede and D. Lloyd. Using auto for stability problmes. https://github.com/sandstede-lab/Auto07p/blob/master/auto07p_tutorial_spatial_pattern_formation/auto07p_tutorial_spatial_pattern_formation.pdf. Accessed: 2024-08-15.