AUTO-07P projects

The following three sample projects could be pursued by participants:

- Demo pp2: if you would like to use prepared AUTO files, try this demo to continue equilibria of the planar ODE, locate and continue saddle-node and Hopf bifurcations, and continue periodic orbits.
- Compute a circle using continuation: this problem allows you to implement an algebraic problem in AUTO.
- Compute and continue a pulse solution: this project is about the computation of travelling waves in AUTO via a boundary-value-problem formulation. Starting data come from an explicit solution.

Other sample project are listed on the following page.

Predator-prey model: Consider the predator-prey model

$$\dot{u} = bu(1-u) - uv - a(1-\mathrm{e}^{-cu}), \qquad \dot{v} = -v + duv$$

where $u, v \geq 0$ and $a, b, c, d > 0$. This system is used for the AUTO demo `pp2` for which details are given in the AUTO manual. The code is set up in the directory `auto-pp2-demo`: go to this directory and follow the commands outlined in the `README` file.

Compute a circle with AUTO: Trace out the circle $x^2 + y^2 = 1$ numerically with AUTO using continuation. The solution can be found in the directory `auto-circle`.

Pulses in a bistable partial differential equation: Consider the bistable PDE

$$u_t = u_{xx} - u + au^3, \qquad x \in \mathbb{R},$$

which admits the standing pulse solution $u(x,t) = \sqrt{2}\,\mathrm{sech}(x)$ when $a = 1$. Compute and continue these pulses numerically in AUTO starting from the exact solution. Experiment with using different truncation intervals and different values of NTST. The solution can be found in the directory `auto-bistable`.

Other AUTO-07P projects

Computation of real eigenvalues: Take your favorite $n \times n$ matrix A and determine its real eigenvalues numerically with AUTO via continuation:

- AUTO: solve the algebraic system $(A - \lambda)v = 0$ for v by continuation in $\lambda \in \mathbb{R}$ with starting data $(v, \lambda) = 0$. Enable detection of branch points. What do you find?
- Theory: investigate why this algorithm works.

Hopf bifurcations and branch switching: Consider the Brusselator

$$\dot{u} = a - (b+1)u + u^2 v, \qquad \dot{v} = bu - u^2 v$$

where $u, v \geq 0$ and $a, b > 0$. This system has the unique equilibrium $(u, v) = (a, b/a)$.

- AUTO: continue the equilibrium numerically and investigate its Hopf bifurcations. Trace out the curve of Hopf bifurcations in (a, b)-space, and compute the periodic orbits that bifurcate from the equilibrium. Check `fort.9` to see whether the periodic orbits are stable or unstable.
- Theory: compare the location of the Hopf bifurcation curve in (a, b)-space with the theoretical prediction.

Pitchfork bifurcations and branch switching: Consider the second-order equation

$$\ddot{u} = \sin(u)\left[\cos(u) - \frac{1}{\gamma}\right]$$

that describes a bead that slides on a wire hoop. This system has the equilibrium $u = 0$ for all values of γ. Investigate its bifurcations numerically using AUTO: branch switch onto any bifurcating solutions by (i) using the branch switching algorithm built into AUTO and (ii) adding a symmetry-breaking term.

Fronts in the Nagumo equation: Consider the Nagumo PDE

$$u_t = u_{xx} + u(u-a)(1-u), \qquad x \in \mathbb{R},$$

which admits the travelling fronts $u(x,t) = 1 + \tanh((x+ct)/2)$ with $c = (1-2a)/\sqrt{2}$ for $0 < a < 1$. Compute and continue these fronts numerically in AUTO starting from the exact solution. Experiment with using different truncation intervals and different values of NTST.

Finite-difference approximations of PDEs: Consider the Nagumo PDE

$$u_t = u_{xx} - cu_x + u(u-a)(1-u)$$

in a moving coordinate frame on a large interval $(-L, L)$ with Dirichlet boundary conditions at $x = \pm L$. Discretize this PDE in space using centered finite differences and implement the resulting large algebraic system in AUTO. Using appropriate values of L and the step size (you need to experiment), find the travelling waves $u(x) = 1 + \tanh(x/2)$ with $c = (1-2a)/\sqrt{2}$ in AUTO and determine the stability of the front by checking `fort.9`. Plot the solution profiles in MATLAB and compare them with the exact solution.
