Bifurcation diagram of system of ordinary differential equations by continuation algorithm

Boris Fačkovec

29th September 2011

Summary

The program takes system of ordinary differential equations with one parameter and starting point on input and calculates one curve of stationary solutions and its stability. The algorithm was proposed by Kubicek in 1976.

Let N be number of differential equations and M be number of steps of continuation. The code is $O(N^3M)$ complex in big O notation. It stores whole trajectory in RAM, therefore it has $O(M)$ memory consumption for long trajectories (in usual cases $M > N^2$) and $O(N^2)$ for short ones.

The program works fine with MATLAB 2008b and with GNU Octave, version 3.2.4 .

Brief theory

Stationary solution

Let $\mathbf{f}$ be set of N functions of N variables and one parameter. The equations (1) form a system of ordinary differential equations with a parameter.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x_1}...\mathbf{x_N}, \alpha) \tag{1}$$

Stationary solutions of this system are the points $\mathbf{x}_0$ in the phase space, where $\mathbf{f}(\mathbf{x}_0) = \mathbf{0}$ and depend on parameter α.

A characteristic quantity (one of variables $x_1...x_N$, their average, norm of vector $\mathbf{x}_0$ etc) plotted against α is called bifurcation diagram [1]. Construction of bifurcation diagram is the fundamental task of non-linear dynamic system analysis. The simplest approach to bifurcation diagram appears to be solving set of nonlinear equations for particular α. However, convergence of Newton method is not ensured if starting too far from stationary point, especially in many-dimensional phase spaces. Such calculation would be very time-consuming, as one would have to start from many different

An approach to obtain whole curve of stationary solutions up to infinite accuracy [2] is here described and implemented.

[1] "diagram řešení" in Czech. Continuation can be used to plot curves of limit points and Hopf bifurcations in 2D bifurcation diagrams ("bifurkační diagram" in Czech) for 2-parametric systems. I have implemented the latter continuation algorithm in different MATLAB function package.

[2] Oversimplification! Round-off errors can trap the algorithm in bifurcation points, where 2 or more curves cross each other.

Searching for stationary solutions

A stationary solution can be found solving undetermined system of N non-linear algebraic equations with $N+1$ variables ($x_1 ... x_N$, α). Such system can be solved by Newton method after careful freezing one parameter. For some stationary solutions

$$\frac{\partial x_k}{\partial \bar{x}} = 0 \tag{2}$$

These stationary solutions are called limit points and they correspond to saddle-node bifurcations. In these points, x_i cannot be frozen. Problem emanating from freezing wrong variable is depicted on Figure 1.

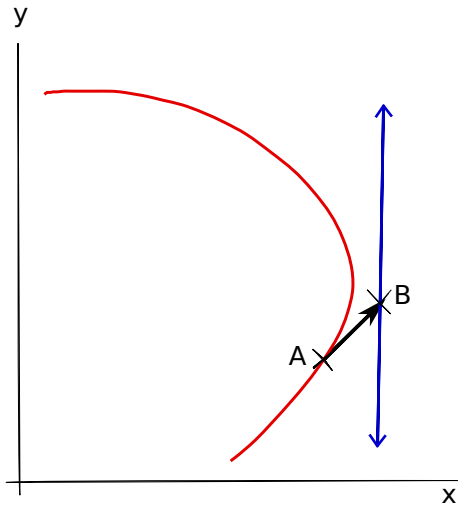

Figure 1: Incapability to correct position when wrong variable is frozen. Predictor approached from A to B. If x variable is frozen, Newton method can move only along blue lines, therefore cannot return to correct trajectory (red).

The problem can be solved by diagonalization of Jacobi matrix with Gauss-Jordan method using maximum pivot. The last remaining column represents the best variable to be frozen.

Continuation of stationary solutions

introducing artificial parameter z,

$$\sum_{i=1}^{N} \left(\frac{\mathrm{d}x_i}{\mathrm{d}z} \right)^2 + \left(\frac{\mathrm{d}\alpha}{\mathrm{d}z} \right)^2 = 1 \tag{3}$$

meaning length of the curve in Eucleides metric. Parameter α will be treated the same as the variables of Equation (1), let therefore $x_{N+1} = \alpha$. Differentiation of (1) yields

$$\frac{\mathrm{d}f_i}{\mathrm{d}z} = \sum_{j=1}^{N+1} \frac{\partial f_i}{\partial x_j} \frac{\mathrm{d}x_j}{\mathrm{d}z} = 0 \tag{4}$$

must be zero in stationary point. System of N linear equations (1) has unique non-zero solution iff the matrix

$$\mathbf{J_k} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_{k-1}} & \frac{\partial f_1}{\partial x_{k+1}} & \dots & \frac{\partial f_1}{\partial x_{N+1}} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{\partial f_N}{\partial x_1} & \dots & \frac{\partial f_N}{\partial x_{k-1}} & \frac{\partial f_N}{\partial x_{k+1}} & \dots & \frac{\partial f_N}{\partial x_{N+1}} \end{bmatrix} \tag{5}$$

is regular. This condition can be again ensured by diagonalization of full $N \times N+1$ Jacobi matrix with Gauss-Jordan method using maximum pivot. Let the elements of the last remaining (k^{th}) column denote $\beta_1 ... \beta_{k-1}, \beta_{k+1} ... \beta_{N+1}$. Then,

$$\frac{\mathrm{d}x_k}{\mathrm{d}z} = \pm \left(1 + \sum_{i=1, i\neq k}^{N+1} \beta_i^2 \right)^{-\frac{1}{2}} \tag{6}$$

$$\frac{\mathrm{d}x_i}{\mathrm{d}z} = \beta_i \frac{\mathrm{d}x_k}{\mathrm{d}z}, \qquad i \in \{1, 2, ..., k-1, k+1, ..., n+1\} \tag{7}$$

The sign of derivative $\frac{\mathrm{d}x_k}{\mathrm{d}z}$ is kept the same as in the previous step. Remembering sings of previous derivatives keeps us in the right direction and is used in later trajectory analyses to find limit points.

Starting from known stationary solution, the curve can be prolonged using Euler method

$$x_i^{new} = x_i^{old} + \frac{\mathrm{d}x_i^{old}}{\mathrm{d}z} \Delta z, \qquad i \in \{1, 2, ..., N+1\} \tag{8}$$

Structure of the code

Summary

On Figure 2, you can see hierarchy of the program. Functions at the ends of arrows call functions from which particular arrows are pointing.

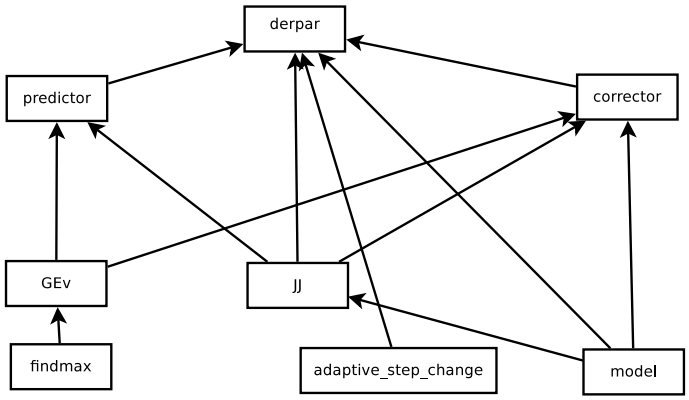

Figure 2: Hierarchy of functions

The following functions are used (all included in standard Octave and Matlab):

abs	find	real
ceil	imag	save
clc	inv	sign
eig	max	size
error	num2cell	sqrt
eye	ones	zeros

Derivatives are obtained numerically. Step size is adaptively changed depending on number of Newton iterations in previous step.

Overview of functions

in alphabetical order. N denotes number of variables (equations).

- `adaptive_step_change(step, iter)` - returns new stepsize, takes previous stepsize `step` and number of iterations of corrector as input. Default optimum stepsize change factor as a function of Newton interations was determined based on author's experiences with few particular models and it is probably not the best choice for other models. Alternative scheme can be introduced best by altering the `nasobic` vector in `adaptive_step_change.m`. Turning off adaptive step change can be done by deleting corresponding lines in derpar.m, but will probably cause crash in limit points and is not recommended.

- `corrector(xvec, hder, emtol)` - applies Newton method to correct the predicted stationary solution `xvec`. First, it calculates Jacobi matrix, then it finds the variable to freeze, finally corrects by Newton, allowing at most 11 iterations. `xvec` is vector $N \times 1$, `hder` denotes step in numerical differentiation (default 1E-8), `emtol` denotes maximum error tolerance in Newton method (default 1E-4).

- `derpar(xinit, ders, mstep, nosteps, emtol, hder)` - main function of the algorithm, returns trajectory and saves it to file `traj.dat`. Starting point `xinit` (vector $N \times 1$) should be wisely chosen - it should not be a bifurcation point (see Limitations section). $N \times 1$ vector `ders` determines to which ortant a curve tangent points, i.e. it determines direction of the curve (see Equation 6). Its elements might be only 1's or -1's. Maximum step size `mstep` should be better set (reasonably) small in order to obtain smooth curve. The algorithm will optimize stepsize during continuation. Number of steps `nosteps` determines computational time required for integration rather than length of curve on output. Hence, should user wish longer curve, he/she should continue from the last point of previous continuation using the same `ders` (present in output, see section "Structure of output"). `emtol` and `hder` and are optional. denotes tolerance of Newton method and should be set reasonably small depending on steepness and complexity of the vector field of your model. Default `hder` (shift in numerical differentiation) of 1E-8 is sufficiently small to represent correct derivatives and sufficiently big not to introduce significant round-off error. The same `hder` and `emtol` are used for all called functions.

- `findmax(A)` - finds maximum value element in a matrix, returns its position as a 1×2 vector.

- `GEv(JJ)` - Gauss-Jordan elimination using maximum pivot of a matrix `JJ`. `JJ` must be $N \times N+1$ matrix. Returns the elements of the last remaining column and its index.

- `JJ(vec,hder)` - returns Jacobi matrix of the model using numerical derivatives.

- `model(xvec)` - **the only file that must be altered by user**. System of ordinary differential exuations must be written in the form briefly described in `model.m` file. The residual form is familiar to all users of `odeXY` or `fzero` functions used in MATLAB.

- `predictor(xvec,ders,step,hder)` - uses Euler method to calculate prediction of a new point of the curve of stationary solutions (see Equation 8).

Usage

After download, unpack and change to directory with the functions.

1. Write your model in `model.m` file in the form described in the previous section (see also solved examples).

2. In Octave or MATLAB command line use function `derpar(xinit, N, mstep, nosteps, emtol, hder)` (see previous section and worked examples).

Seek your trajectory in output file.

Structure of output

On Figure 3, you can see a typical output.

```
# Created by Octave 3.2.4, Wed Sep 28 20:23:18 2011 CEST <boris@nunu>
# name: traj
# type: matrix
# rows: 101
# columns: 14
 0 0.1041514826995712 -1.9 0.9488757506437782 3.563539207208429e-15 1 1 1 0.9038001874925956 -0.8967418800576166 -0.8967418800576166 0.31568051578031 69 -0.31568051578031 69 4
 0.01 0.1049705514420629 -1.889266214501826 0.9447699034158407 5.793014867379207e-05 1 1 -1 0.8927080319371006 -0.8909511899080069 -0.8909511899080069 0.3145059763162015 -0.3145059763162015 2
 0.02 0.1058039309099885 -1.878549513884405 0.940776999682173 1.387778780781446e-17 1 1 -1 0.8816441945449272 -0.8851643286400535 -0.8851643286400535 0.3132543788171694 -0.3132543788171694 2
 0.032 0.1068024916136135 -1.865709599642909 0.935992748277233 8.237479414879853e-05 1 1 -1 0.8683717305835791 -0.8782249392991213 -0.8782249392991213 0.3115969938504438 -0.3115969938504438 2
 0.044 0.1078222105321931 -1.852893366527636 0.931368337086029 1 4.443059973708341e-16 1 1 -1 0.8551351068379178 -0.8712883067807377 -0.8712883067807377 0.3098254239165553 -0.3098254239165553 2
 0.0584 0.1090522637330128 -1.837541684123432 0.9258823449952607 4.440892098500626e-16 1 1 -1 0.8392638612018973 -0.8629678013849507 -0.8629678013849507 0.3074905445289687 -0.3074905445289687 2
 0.07568 0.1105478277466034 -1.819158753198526 0.9194539443316947 0 1 1 -1 0.82024568218974 -0.8529845024751738 -0.8529845024751738 0.3044061772154446 -0.3044061772154446 2
```

Figure 3: Sample output for system of 2 differential equations (CSTR1EXO).

Under few lines of text starting with #s added by MATLAB, you can find trajectory, one point in each line. Let N be number of differential equations. Following table outlines meaning of output located in n^{th} column of output trajectory.

from column	to column	output means
1		length of the curve from beginning
2	$N+1$	variables $x_1 \ldots x_N$
$N+2$		parameter α
$N+3$		error of predictor in this step
$N+4$	$2N+3$	derivatives $\frac{\mathrm{d}x_i}{\mathrm{d}z}$
$2N+4$		derivative $\frac{\mathrm{d}\alpha}{\mathrm{d}z}$
$2N+5$		determinant of Jacobi matrix
$2N+6$	$3N+5$	real parts of eigenvalues of Jacobi matrix important for stability determination
$3N+6$	$4N+5$	imaginary parts of eigenvalues of Jacobi matrix important for identification of Hopf bifurcations and focus-node transitions
$4N+6$		index of pivot in Gauss-Jordan eliminations

Solved example

Model CSTR1EXOe

Model of continuous ideally stirred tank reactor with exothermal reaction was evolved [Holodniok et al., 1986]. The model contains two variables (dimensionless temperature Θ, conversion x) and one parameter (dimensionless residence time τ).

1. model.m file looks like as follows:

```
function resi = model(xvec)
% resi = model(xvec)
% dynamical model in the form dx_i/dt = resi_i(xvec)
% model(xvec) is one-parametric system of vector fields with variables xvec(1:N) and a parameter xvec(N+1)
% xvec     - point in phase space, N+1 x 1
% resi     - (derived from residual) right side of the equation, vector field


pars = [ 20, 10, 1, -5, 1, 1, 0.6];
v=num2cell(pars);
[gamma, B, Lambda, Thetac, k0, a, beta]=deal(v{:});

u=num2cell(xvec);
[x, Theta, tau]=deal(u{:});

% useful expressions
m = (1+Theta/gamma);
e = exp(Theta/m);
f = tau*k0*(1-x)*e;

% number of equations always in this form
resi = zeros(size(xvec,1)-1,1);
% expressions for dx_i/dt should be written in this form
resi(1) = -Lambda*x + f;
resi(2) = -Lambda*Theta + B*f - a*tau*(Theta-Thetac);
```

Figure 4: Content of model.m file in computing CSTR1EXO model. Other parameters can be found in enclosed file `cstr1exo.m`.

2. `derpar([0 0 0]', [ 1 1 1 ]', 0.05, 1000, 1e-4, 1e-8)`

3. plot of traj(:,2) (conversion) against traj(:,4) (residence time) using Gnuplot yelds:

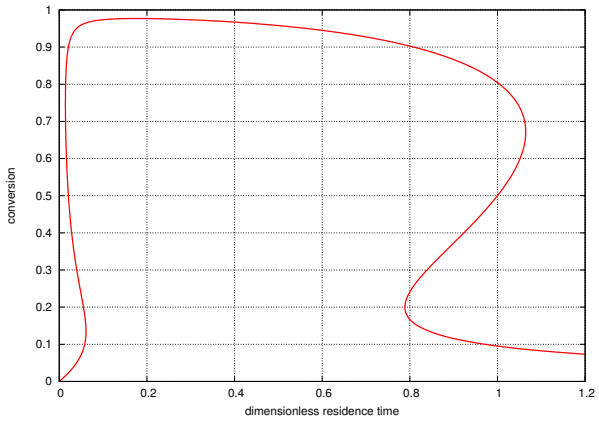

Figure 5: Curve of stationary solutions for model of continuous stirred tank reactor. Smoothness is ensured using small `mstep` and higher `nosteps`.

Stationary solution is stable iff real parts of all eigenvalues are negative.

Limitations

- The algorithm cannot start directly from a bifurcation point. In case of bifurcation point $\mathbf{J}_k$ is singular and program fails. Fix to this problem will be in future version.
- The algorithm cannot discover isolated curves of stationary solutions.
- Scaling of quantities of a model to similar order of magnitude is recommended.

References

M Holodniok, A Klíč, M Kubíček, and M Marek. *Metody analýzy nelineárních dynamický modelu*. 1986.

M Kubíček. Algorithm 502: Dependence of Solution of Nonlinear Systems on a Parameter. *ACM Trans. Math Software*, (2):98, 1976.