Description of extensions `ddebiftool_extra_psol` and `ddebiftool_extra_rotsym`

Jan Sieber

January 13, 2015

This addendum describes two extensions to DDE-BifTool [1, 2, 3, 4, 5], a bifurcation analysis toolbox running in `Matlab`[1] or `octave`[2]. The extension `ddebiftool_extra_psol` enables continuation of periodic orbit bifurcations for delay-differential equations with constant or state-dependent delay. The extension `ddebiftool_extra_rotsym` enables continuation of relative equilibria, relative periodic orbits and their local bifurcations in systems with rotational symmetry. The extensions are implemented by creating *extended problems* and re-using the core DDE-BifTool algorithms for periodic orbits.

Contents

1 Overview **1**

2 Functionality **2**
2.1 Functions of interest in user scripts 2
2.1.1 `SetupPOfold` for folds of periodic orbits 2
2.1.2 `SetupTorusBifurcation` for torus bifurcations 5
2.1.3 `SetupPeriodDoubling` for period doublings 5

3 List of demos **6**

4 Relative equilibria and periodic orbits in systems with rotational symmetry **6**

1 Overview

To get up and running quickly, users familiar with DDE-BifTool should look at the demo in folder `minimal_demo` (script `rundemo.m`). The results are pre-computed and stored in `mat` files. Otherwise, please see the main manual and the tutorial demos `neuron`, `sd_demo` and `hom_demo` for an introduction to using basic DDE-BifTool.
This extension introduces three functions that can be called by the user:

[1] `http://www.mathworks.com`
[2] `http://www.gnu.org/software/octave`

- `SetupPOfold` to initialize continuation of folds of periodic orbits,
- `SetupTorusBifurcation` to initialize continuation of torus bifurcations, and
- `SetupPeriodDoubling` to initialize continuation of period doublings (this is identical to torus bifurcations).

2 Functionality

The folder `ddebiftool_extra_psol` contains functions that implement tracking of
- saddle-nodes (folds),
- period doublings, and
- torus bifurcations

of periodic orbits in two parameters. This is done by creating an *extended* DDE from the user-provided right-hand side. The periodic orbits of the extended DDEs are then continued using DDE-BifTool's original routines for periodic orbits. To access the additional routines, the folder `ddebiftool_extra_psol` has to be added to the path, for example:

```
addpath('../../ddebiftool');            % path to original DDE-BifTool
addpath('../../ddebiftool_extra_psol'); % path to additional routines
```

(adapt to your own folder structure).

2.1 Functions of interest in user scripts

2.1.1 `SetupPOfold` for folds of periodic orbits

Header

```
function [pfuncs,pbranch,suc]=SetupPOfold(funcs,branch,ind,varargin)
```

Brief description Sets up functions defining right-hand side (in `pfuncs`) and the initial part of the branch with the first two points (in `pbranch`) of a curve of folds of periodic orbits.

Inputs
- `funcs` Structure, containing the user-defined right-hand side, conditions, delays, etc., as created with `set_funcs`.
- `branch` Branch of periodic orbits along which the fold that the user intends to track was detected. This has to be a structure of the format expected and created by DDE-BifTool routines such as `df_brnch` or `br_contn`.
- `ind` index of point along the branch that is close to the fold. The periodic orbit `branch.point(ind)` will be used to create an initial guess.
- Optional inputs (key-value pairs in arbitrary order, default in brackets):
  - `'contpar'` (`[]`, integer) *index of continuation parameters*. This argument can be used in two ways:
    - empty (not given): the indices given in `branch.parameters.free` will be used.
    - array `k` with more than a single integer: `k` will *replace* the parameters in the field `branch.parameters.free`.

- `'dir'` (`[]`, integer) *initial direction of branch*; index of parameter to be changed for second point along branch of folds. If empty, then only a single fold point will be computed.
- `'step'` (`1e-3`, real) how much the parameter is changed. For the second point on the branch the relation holds

  ```
  pbranch.point(2).parameter(dir)=...
        pbranch.point(1).parameter(dir)+step;
  ```

- `'correc'` (`true`, logical) Apply `p_correc` to the initial points.
- `'sys_deri'` (`1e-4`), `'sys_dtau'` (`1e-4`), `'hjac'` (`1e-4`) deviations used in finite-difference formulas. If the user has provided the functions `sys_deri` and `sys_dtau` then these options will not be used for constant delays. For state-dependent delays the analytic Jacobian of the extended system is not yet implemented such that `'hjac'` will be used in the finite-difference approximation of the Jacobian.

Outputs If the output `suc` is non-zero, then `pfuncs` and `pbranch` can be fed into `br_contn`. For example,

```
pbrlong=br_contn(pfuncs,pbranch,100);
```

- `pfuncs` Structure defining right-hand side of the extended DDE for folds of periodic orbits. The structure has an additional field `'get_comp'`, which can be used to extract the original solution components from the solution of the extended system (removing additional components and prameters, see example).
- `pbranch` A DDE-BifTool branch structure, containing the first point or the first two points (if desired by setting the optional argument `'dir'`).
- `suc` (logical) indicates success of approximation (and correction if desrired) of initial points.

Example See `minimal_demo`, the Duffing oscillator with delayed feedback of the form

$$\ddot{x}(t) + d\dot{x}(t) + ax(t) + x(t)^3 + b[x(t) - x(t-\tau)] = 0, \tag{1}$$

with parameters $p = (\tau, a, b, d)$ (initially $p = (0, 0.5, 0.6, 0.05)$). After continuation of the family of periodic orbits in τ (p_1) one has a branch

```
>> per_orb
per_orb =
       method: [1x1 struct]
    parameter: [1x1 struct]
        point: [1x173 struct]
```

with 173 points, of which point 52 is close to a fold

```
>> ind_fold
ind_fold =    52
>> per_orb.point(ind_fold).stability.mu(1:3)
ans =
   1.0000
   0.9985
  -0.0263 + 0.0219i
```

such that one can call

```
[pfuncs,pbranch]=SetupPOfold(funcs,per_orb,ind_fold,...
    'contpar',[3,1],'dir',3,'step',-1e-3);
pbranch=br_contn(pfuncs,pbranch,60);
```

The free parameters and the intial step in p_3 (as requested by the arguments `'dir'` and `'step'`) are

```
>> pbranch.parameter.free
ans =     3     1     5     6     7
>> pbranch.point(2).parameter(3)
ans =    0.5990
>> pbranch.point(1).parameter(3)
ans =    0.6000
```

Note that the extended system introduces the additional parameters $(\beta, T_{\text{copy}}, \tau_{\text{ext}})$ (see [6]), which are stored at indices 5 to 7. The extensions are also visible when looking at the points:

```
>> pbranch.point(1)
ans =    kind: 'psol'
    parameter: [1x7 double]
         mesh: [1x81 double]
       degree: 4
      profile: [4x81 double]
       period: 0.5386
```

Use `pfuncs.get_comp` to remove all additional components and parameters:

```
>> pfuncs.get_comp(pbranch.point(1),'solution')
ans =    kind: 'psol'
    parameter: [0.9343 0.5000 0.6000 0.0500]
         mesh: [1x81 double]
       degree: 4
      profile: [2x81 double]
       period: 0.5386
```

The function `pfuncs.get_comp` supports the strings

- `'kind'` returning `'POfold'`,
- `'solution'` removing all extensions, returning a point of the same format as the original periodic orbits,
- `'nullvector'` returning the extended components of `point.profile` (in the example, rows 3 and 4), and the extension parameter β (in the example in position 5) as a `'psol'` structure,
- `'delays'` returning the additional parameter(s) τ_{ext} (see [6] for their meaning).

Warning For problems with constant delays the extended problem `pfuncs` introduces $n_\tau(n_\tau+1)/2$ additional delays as parameter τ_{ext}. For problems with state-dependent delays the additional delays do not show up as additional parameters. However, the number of additional delays is $n_\tau(n_\tau+1)$. Since the user-defined functions have to be called at all delay-shifted collocation points the computations may become slow for problems with many delays (particularly for state-dependent delay problems).

2.1.2 `SetupTorusBifurcation` for torus bifurcations

Header

```
function [trfuncs,trbranch,suc]=SetupTorusBifurcation(...
                                        funcs,branch,ind,varargin)
```

Brief description Sets up the right-hand side (in `trfuncs`) and the first two points (in `trbranch`) of a curve of torus bifurcations or period doublings of periodic orbits (which one depends on the Floquet multipliers of `branch.point(ind)`). The meaning of inputs and outputs are the same as for `SetupPOfold`, except that the points on the resulting branch lie on a torus or period doubling bifurcation curve. Also, the right-hand side of the extended DDE is different from the corresponding output of `SetupPOfold` (see [6] for the extended system).

Example (See again `minimal_demo`.) The point with index `ind_tr1` has two complex Floquet multipliers near the unit circle:

```
>> per_orb.point(ind_tr1).stability.mu(1:5)
ans =
   1.1643
  -0.6292 + 0.8380i
  -0.6292 - 0.8380i
   1.0000
   0.5968
```

So, one can call

```
[trfuncs,trbranch1]=SetupTorusBifurcation(funcs,per_orb,ind_tr1,...
    'contpar',[3,1],'dir',3,'step',-1e-3);
trbranch1=br_contn(trfuncs,trbranch1,50);
```

to continue the torus bifurcation in the two parameters (τ, b). An interesting parameter to watch during torus continuation is ω, the first extended parameter, which is the angle of the complex Floquet multipliers on the unit circle in multiples of π (so, if the Floquet multiplier is at -1, ω equals 1 for period doublings).

Warning For problems with state-dependent delays the extended problem `pfuncs` introduces $n_\tau(n_\tau+1)$ additional delays. The additional delays do not show up as additional parameters. Since the user-defined functions have to be called at all delay-shifted collocation points the computations may become slow for problems with many delays.

2.1.3 `SetupPeriodDoubling` for period doublings

Header

```
function [trfuncs,trbranch,suc]=SetupPeriodDoubling(...
                                        funcs,branch,ind,varargin)
```

This function is a wrapper, just calling `SetupTorusBifurcation`, to avoid the misleading name. The free parameter ω equals unity for period doublings (corresponding to a rotation by π).

3 List of demos

- **`minimal_demo`**: Duffing oscillator with delayed feedback. The delay is a parameter [7]. This demonstrates usage of `set_funcs` and continuation of folds and torus bifurcations of periodic orbits in two parameters.
- **`Mackey-Glass`**: Mackey-Glass equation

$$\dot{x}(t) = \beta \frac{x(t-\tau)}{1+x(t-\tau)^n} - \gamma x(t).$$

  (`http://www.scholarpedia.org/article/Mackey-Glass_equation`). The delay is a parameter. This demonstrates the continuation of period doublings in two parameters. This demo has a switch `x_vectorize` at the top. Change it to `false` to see the effect of vectorization on speed.
- **`nested`**: Demonstrates continuation of periodic orbits, computation of their stability and continuation of fold of periodic orbits in two parameters for a system with state-dependent delays and arbitrary levels of nesting. The default system is

$$\dot{x}(t) = x(t - p_1 - x(t - p_1 - x(t - p_1 - x(t)))) + p_2 x^5.$$

- **`humphriesetal`**: example studied in [8],

$$\dot{x}(t) = -\gamma x(t) - \kappa_1 x(t - \alpha_1 - c x(t)) - \kappa_2 x(t - \alpha_2 - c x(t)).$$

  Demonstrates continuation of folds of periodic orbits and torus bifurcations for an equation with state-dependent delay.

4 Relative equilibria and periodic orbits in systems with rotational symmetry

As a showcase demonstrating how one can extend DDE-BifTool's functionality, the subfolder `ddebiftool_extra_rotsym` and the demo `rotsym_demo` are included. The routines in `ddebiftool_extra_rotsym` enable users to track relative equilibria and periodic orbits of DDE systems with rotational symmetry and constant delays. Specifically, if the right-hand side of the DDE

$$\dot{x} = f(x(t), x(t-\tau_1), \ldots, x(t-\tau_{n_\tau}), p) \tag{2}$$

satisfies

$$\exp(A\varphi) f(x_0, x_1, \ldots, x_{n_\tau}, p) = f(\exp(A\varphi)x_0, \exp(A\varphi)x_1, \ldots, \exp(A\varphi)x_{n_\tau}, p)$$

for some anti-symmetric matrix A ($A^T = -A$) and all $\varphi \in \mathbb{R}$, $x_0, \ldots, x_{n_\tau} \in \mathbb{R}^n$ and parameters p, then (2) possesses two special types of solutions:

- *Relative equilibria* (or *rotating waves*). These are periodic orbits of the the form

$$x(t) = \exp(A\rho t)x_0. \tag{3}$$

- *Relative periodic orbits* (or *modulated waves*). These are invariant two-tori of the form

$$x(t) = \exp(A(\rho t + \varphi))x_0(t) \tag{4}$$

  for all $t \in \mathbb{R}$ and some $\varphi \in \mathbb{R}$ and periodic function $x_0(\cdot)$ (that is, $x_0(t) = x_0(t+T)$ for all $t \in \mathbb{R}$ and some $T > 0$).

For DDE-BifTool to be useful for DDEs with rotational symmetry, it has to be able to continue relative equilibria and their bifurcations similar to classical equilibria, and relative periodic orbits and their bifurcations similar to classical periodic orbits. The function `set_rotfuncs` in `ddebiftool_extra_rotsym` creates a right-hand side `rot_rhs` out of the user-given right-hand side f and the matrix A (and $\exp(A)$ if provided) in rotating coordinates. The rotation frequency ρ is an additional unknown that gets fixed with an additional phase condition (in `rot_cond`). The other functions in `ddebiftool_extra_rotsym` create wrappers around their classical counterparts found in `ddebiftool_extra_psol`. The demo `rotsym_demo` shows how to use the extensions for rotational symmetry for the Lang-Kobayashi equations [9].

References

[1] K Engelborghs, T Luzyanina, and G Samaey. DDE-BIFTOOL v.2.00: a Matlab package for bifurcation analysis of delay differential equations. Report TW 330, Katholieke Universiteit Leuven, 2001.

[2] K. Engelborghs, T. Luzyanina, and D. Roose. Numerical bifurcation analysis of delay differential equations using DDE-BIFTOOL. *ACM Transactions on Mathematical Software*, 28(1):1–21, 2002.

[3] G. Samaey, K. Engelborghs, and D. Roose. Numerical computation of connecting orbits in delay differential equations. Report TW 329, Department of Computer Science, K.U.Leuven, Leuven, Belgium, October 2001.

[4] D. Roose and R. Szalai. Continuation and bifurcation analysis of delay differential equations. In B Krauskopf, H M Osinga, and J Galán-Vioque, editors, *Numerical Continuation Methods for Dynamical Systems*: *Path following and boundary value problems*, pages 51–75. Springer-Verlag, Dordrecht, 2007.

[5] K. Verheyden, T. Luzyanina, and D. Roose. Efficient computation of characteristic roots of delay differential equations using lms methods,. *Journal of Computational and Applied Mathematics*, 214:209–226, 2008.

[6] J. Sieber. Extended systems for delay-differential equations as implemented in the extensions to DDE-BifTool. *Figshare*, `http://dx.doi.org/10.6084/m9.figshare.757725`, 2013.

[7] S Yanchuk and P Perlikowski. Delay and periodicity. *Physical Review E*, 79(4):46221, 2009.

[8] A. R. Humphries, O. DeMasi, F. M. Magpantay, and F. Upham. Dynamics of a delay differential equation with multiple state dependent delays. *Discrete and Continuous Dynamical Systems A*, 32(8):2701–2727, 2012.

[9] R Lang and K Kobayashi. External optical feedback effects on semiconductor injection properties. *IEEE J. of Quant. El.*, 16:347–355, 1980.