# Changes of functionality or code in core DDE-BifTool routines from v. 2.03 to v. 3.0

Jan Sieber

April 5, 2017

Substantial changes to the following core DDE-BifTool [1, 2, 3, 4, 5] routines were made.

**psol_jac** Enabled vectorization, re-use for Floquet multipliers and vectors, and nested state-dependent delays [6]. Additional optional inputs (as name-value pairs):

- `'wrapJ'` (default `true`): if `false` mesh gets extended backward and forward in time to cover all delayed time points. For `wrapJ==false` no augmentation is done (that is, derivatives w.r.t. period or parameters are not calculated) and no phase or boundary conditions are appended to output matrix `J`.
- `'bc'` (default `true`): controls whether to append boundary conditions.
- `'c_is_tvals'` (default `false`): entries of argument `c` are interpreted as the collocation points in the full interval (usually `c` is empty or giving the collocation points relative to a subinterval). The residual is only calculated in these points (incompatible with `'bc'==true`).
- `'Dtmat'` (default `eye(size(psol_prof,1))`): pre-factor in front of time derivative. This permits evaluation of algebraic constraints.

Additional outputs:

- additional output `tT` (array of delays, scaled by period)
- additional output `extmesh` (mesh of time points extended back to `-max(tT(:))` if `wrapJ` is `false`, otherwise equal to argument `mesh`.

The loops are re-arranged to enable a single vectorized call of the user functions in the new function `psol_sysvals`. Moreover, the evaluation of `xx` at points in $[0,1]$ now uses `psol_eva` to avoid code repetition.

**p_correc** Changed to avoid code duplication and permit finding of zero-crossings of state-dependent delays with arbitrary levels of nesting. The original contained a large chunk of code repeating code from `psol_jac` to create a constraint of the form $\tau_j(t_z) = 0$, $\tau_j'(t_z) = 0$ for a fixed $t_z \in [0,1]$. This code has been replaced by an addiitonal function `delay_zero_cond`, which in turn calls the now more flexible `psol_jac` to perform the calculations.

**mult_app** Changed to avoid code duplication.

- Has second (optional) output argument `eigenfuncs`, returning the eigenfunctions on the extended mesh `extmesh` as output by `psol_jac`. The extended mesh (`extmesh` is the third output in this case.

- The original version repeated the code for the Jacobian from `psol_jac` and appended code for calculating the monodromy matrix. These two parts have been replaced by calls to the new more flexible `psol_jac` and the manual calculation of the monodromy matrix has been replaced by a call to the backslash operator.
- If the delays are negative another (more general, but possibly more expensive) algorithm is used. The Jacobian $J$ on `extmesh`, as output by `psol_jac` has dimensions $N_r \times N_c =$ `n*(length(mesh)-1)` $\times$ `n*length(extmesh)`. We solve an augmented and a generalized eigenvalue problem. Let $N_{\text{ext}} = N_c - N_r$ be the difference between the column and row dimensions of the unwrapped Jacobian $J$. Then the generalized eigenvalue problem for the eigenpair $(\mu, v)$ is

$$\begin{bmatrix} J & \\ 0_{N_{\text{ext}} \times (N_c - N_{\text{ext}})} & \mathrm{I}_{N_{\text{ext}}} \end{bmatrix} v = \mu \begin{bmatrix} & 0_{N_r \times N_c} \\ \mathrm{I}_{N_{\text{ext}}} & 0_{N_{\text{ext}} \times (N_c - N_{\text{ext}})} \end{bmatrix} v.$$

  This generalized eigenvalue problem can in principle also be used to detect bifurcations of periodic orbits with delays of mixed signs. It gives up to numerical round-off errors results that are identical to the results from the monodromy matrix used in the original code. However, it operates on a pair of large full matrices, becoming expensive for large delays or fine discretizations.

**`p_topsol`** Uses the general routine `mult_crit` instead of `mult_dbl` and `mult_one` to compute eigenfunction for critical Floquet multipliers. This avoids code duplication. Both, `mult_dbl` and `mult_one`, originally duplicated code from the original `psol_jac` and `mult_app`. The routine `mult_crit` calls `mult_app` instead.

**`psol_eva`, `p_tau`, `p_tsgn`, `poly_elg`, `poly_del`, `poly_lgr`, `poly_dla`** Changed to support and speed up call with many evaluation points.

**`df_deriv` and `df_derit`** Both functions have been amended to enable (pass on) vectorization such that they can now perform the requested operation for arguments `xx` of size $n \times (n_\tau + 1) \times n_{\text{vec}}$ and return Jacobians of the corresponding shape. They also apply central difference formulas, making them slower, but potentially more accurate. Note that this may result in errors in scripts that previously worked. For example, if the right-hand side becomes invalid for certain negative arguments and this argument is close to 0.

**`stst_stabil` and `get_pts_h_new`** In the computation of eigenvalues of equilibria an a-priori heuristics estimates where eigenvalues can lie in the complex plane and adjusts the discretization stepsize accordingly (see [5] for technical details). The implementation in v. 2.03 resulted in error messages in various common situations, for example, if the system had more than 3 delays, if all delays were zero, or if the estimates returned only real parts less than `minimal_real_part`.

#### Additional auxiliary functions

- `p_dot` computes the dot product between two points of kind `'psol'`. The function uses the Gauss weights for the evaluation of the integral. The options `'free_par_ind'` (default

empty) and 'period' (default false) set which free parameters (non-zero list of indices are included into the dot product. The option 'derivatives' (default [0,0]) sets hiw often each of the two profiles is differentiated before taking the dot product (useful for computing products of type $\int_0^1 (\dot{p}(t))^T q(t)\,\mathrm{d}t$).

- delay_zero_cond is a function that has a format suitable for use with method.extra_condition. It returns the residual and the Jacobian for $\tau_j(t_z) = 0$ for point types 'stst','hopf' and 'fold', and $\tau_j(t_z) = 0$, $\tau_j'(t_z) = 0$ for point type 'psol'.

- VAopX and sparse_blkdiag functions enabling vectorized matrix multiplication. For a $n \times m \times p$ array A and a $m \times q \times p$ array B

  ```
  C=VAopX(A,B,'*')
  ```

  gives an $n \times q \times p$ array C consisting of the matrix products for each of the $p$ stacked matrices.

- psol_sysvals checks if funcs.x_vectorized is set and performs user function calls (either vectorized or not). This part has been factored out of psol_jac to make psol_jac less complex.

- dde_set_options is an auxiliary routine used for treatment of optional arguments.

## 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.