A user manual for the Matlab package computing all the characteristic roots of delay differential equations in a given right half plane using a spectral method

Zhen Wu and Wim Michiels
Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A, 3001 Heverlee, Belgium
{Zhen.Wu,Wim.Michiels}@cs.kuleuven.be

May, 2011

In order to describe how to use this matlab package, we are going to consider a system of DDE with three delays:

$$\dot{x}(t) = A_0 x(t-\tau_0) + A_1 x(t-\tau_1) + A_2 x(t-\tau_2) + A_3 x(t-\tau_3), \tag{1}$$

where $x(t) \in \mathbb{R}^3$, delays $\tau_0, \tau_1, \tau_2, \tau_3 = 0,\ 0.1,\ 0.15,\ 0.25$ respectively and coefficient matrices are

$$A_0 = \begin{bmatrix} -9.6713 & -9.7546 & -9.4913 \\ 1.8381 & 1.7961 & 9.5716 \\ 1.3647 & -2.7957 & -7.3561 \end{bmatrix},\ A_1 = \begin{bmatrix} 1.0115 & -9.3006 & 5.3222 \\ 7.2688 & -1.1960 & 9.9968 \\ 3.6508 & -1.2035 & -4.8507 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 7.7163 & 4.5911 & -5.5072 \\ -9.0056 & -0.0260 & -7.5404 \\ -3.3669 & 0.9332 & -0.2958 \end{bmatrix},\ A_3 = \begin{bmatrix} 7.4808 & -7.2571 & 9.4377 \\ 2.8285 & -7.1768 & -1.4221 \\ -1.0353 & 9.6519 & 5.1208 \end{bmatrix}.$$

1 Create Time-delay system from input data

To create a Time-delay object from input data, first we define the matrices and delays:

```
>> A0=[-9.6713 -9.7546 -9.4913; 1.8381, 1.7961, 9.5716; 1.3647, -2.7957, -7.3561];
>> A1=[1.0115, -9.3006, 5.3222; 7.2688, -1.1960, 9.9968; 3.6508, -1.2035, -4.8507];
>> A2=[7.7163, 4.5911, -5.5072; -9.0056, -0.0260, -7.5404; -3.3669, 0.9332, -0.2958];
>> A3=[7.4808, -7.2571, 9.4377; 2.8285, -7.1768, -1.4221; -1.0353, 9.6519, 5.1208];
>> hA=[0, 0.1, 0.15, 0.25];
```

then we create the system **tds**:

```
>> tds = tds_create({A0 A1 A2 A3},hA)
tds =
       E: {[3x3 double]}
      hE: 0
       A: {[3x3 double]  [3x3 double]  [3x3 double]  [3x3 double]}
      hA: [0 0.100000000000000 0.150000000000000 0.250000000000000]
      B1: {}
     hB1: []
      C1: {}
     hC1: []
     D11: {}
    hD11: []
      B2: {}
```

```
    hB2: []
     C2: {}
    hC2: []
    D12: {}
   hD12: []
    D21: {}
   hD21: []
    D22: {}
   hD22: []
```

Remark 1. *If users choose to define the Time-delay systems by themselves, they can use the following matlab function to check if their systems are valid:*

```
>> tds_check_valid(tds)
```

2 Computing all characteristic roots of DDEs in a given right half plane $\Re(\lambda) \geq r$

tdsrootsoptions.m

```
>> options=tdsrootsoptions
options =
              minimal_real_part: []
    max_size_eigenvalue_problem: 1000
          newton_max_iterations: 20
                  root_accuracy: 1.000000000000000e-08
       commensurate_basic_delay: []
           number_grid_points_p: 20
              new_basic_delay_q: 100
```

- **options.minimal_real_part** is the minimal real part of the characteristic roots. If you leave it empty, it will automatically choose $-1/\tau_m$, where τ_m is the maximum of delays.
- **options.max_size_eigenvalue_problem** is the maximal size of the corresponding eigenvalue problem.
- **options.newton_max_iterations** is the maximal number of iterations by using Newton correction.
- **options.root_accuracy** is the root accracy of corrected characteristic roots by Newton correction.
- If users set **options.commensurate_basic_delay** $= \tau$, then our code will check if delays are commensurate based on τ, i.e., check if τ_k/τ is an integer, for all $k = 1, \ldots, m$. If delays are commensurate, we use an adapted heuristic to determine the number of discretization points.
- **options.number_grid_points_p** is the number of grid points in the interval $[0, 2\pi)$ for the discretization of Ψ.
- **options.new_basic_delay_q** is used for approximating delays by commensurate delays if $m > 3$ or τ_m/**options.commensurate_basic_delay** > **options.new_basic_delay_q**, then delays are approximated by mutiples of τ_m/**options.new_basic_delay_q** in the estimation of the roots' location.

tds_characteristic_roots.m

```
>> [eigenvalues, N, size_eigenvalue_problem]=tds_charateristic_roots(tds, options)
```

- Inputs: **tds** is a DDE structure created by **tds_create**; **options** is from **tdsrootsoptions.m**
- Outputs: approximations of characteristic roots (**eigenvalues.l0**), corrected roots (**eigenvalues.l1**), the number of discretization points (**N**) and the size of the corresponding eigenvalue problem (**size_eigenvalue_problem**).

eigenplot.m

```
>> eigenplot(eigenvalues)
```

- This function is to plot approximations (**eigenvalues.l0**) and the corrected roots (**eigenvalues.l1**)

As an example, we compute all characteristic roots of system (1) with their real part $\Re(\lambda) \geq -7$. In Section 2.1, delays are treated as independent delays, while in Section 2.2, thet are treated as commensurate delays.

2.1 Treating delays as independent delays

```
>> options=tdsrootsoptions;
>> options.minimal_real_part=-7;
>> [eigenvalues, N, size_eigenvalue_problem]=tds_charateristic_roots(tds, options)
eigenvalues =
    l0: [12x1 double]
    l1: [12x1 double]
N =
    15
size_eigenvalue_problem =
    48    48
>>eigenplot(eigenvalues);
```

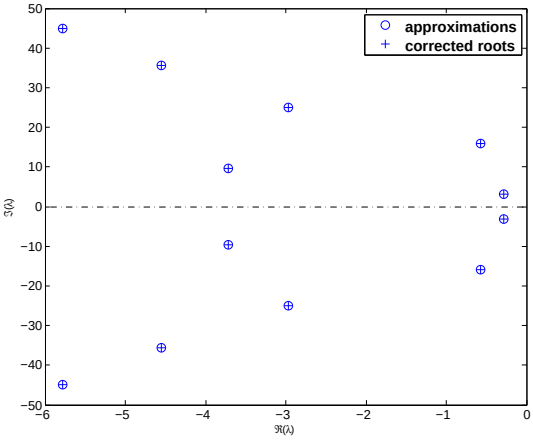

Figure 1: Characteristic roots with $\Re(\lambda) \geq -7$

2.2 Treating delays as commensurate delays

The delays of system (1) can be considered as commensurate delays, i.e.,

$$\tau_{1,2,3} = 0.05 n_{1,2,3}, \text{ where } n_{1,2,3} = 2, 3, 5.$$

Next we will compute its characteristic roots in the given right half plane $\Re(\lambda) \geq -7$

```
>> options=tdsrootsoptions;
>> options.minimal_real_part=-7;
>> options.commensurate_basic_delay=0.05;
>> [eigenvalues, N, size_eigenvalue_problem]=tds_charateristic_roots(tds, options)
eigenvalues =
    l0: [12x1 double]
    l1: [12x1 double]
N =
    12
size_eigenvalue_problem =
    39    39
```

From the results, we can see in the case of commensurate delays, by using the adapted heuristic, the size of the corresponding eigenvalue problem is reduced. This is also discussed in [1] theoretically.

3 Computing all characteristic roots of DDE in a given rectangular region

tdsrootsoptions1.m

```
>> options=tdsrootsoptions1
options =
    max_size_eigenvalue_problem: 1000
          newton_max_iterations: 20
                  root_accuracy: 1.000000000000000e-08
```

- **options.max_size_eigenvalue_problem** is the maximal size of the corresponding eigenvalue problem.
- **options.newton_max_iterations** is the maximal number of iterations by using Newton correction.
- **options.root_accuracy** is the root accracy of corrected characteristic roots by Newton correction.

tds_region_roots.m

```
>> [eigenvalues, N, size_eigenvalue_problem]=tds_region_roots(tds,region, options)
```

- Inputs: **tds** is DDE structure created by **tds_create**; **options** is from **tdsrootoptions1.m**; **region** is a row vector to define the rectangle: [left bound, right bound, lower bound, upper bound];
- Outputs: approximations of characteristic roots (**eigenvalues.l0**), corrected roots (**eigenvalues.l1**), the number of discretization points (**N**) and the size of the corresponding eigenvalue problem (**size_eigenvalue_problem**).

When computing characteristic roots in a given rectangular region, the location of the desired roots are already given. Therefore, we don't need to consider if we consider delays as independent delays or commensurate delays in estimation the region of characteristic roots.

As an example, we compute all characteristic roots of system (1) in a given rectangular, i.e., $[-10,\ 0,\ -20,\ 100]$:

```
>> options=tdsrootsoptions1;
>> region=[-10 0 -20 100];
>> [eigenvalues, N, size_eigenvalue_problem]=tds_region_roots(tds, region, options)
eigenvalues =
    l0: [14x1 double]
    l1: [14x1 double]
number_of_discretization_points =
    11
size_eigenvalue_problem =
    36    36
>> eigenplot(eigenvalues);
```

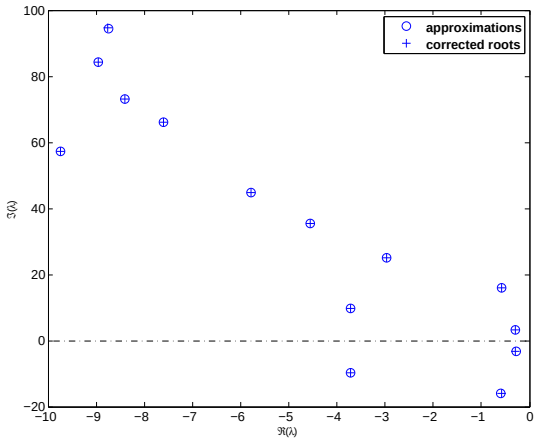

Figure 2: Characteristic roots with $-10 \leq \Re(\lambda) \leq 0$ and $-20 \leq \Im(\lambda) \leq 100$

4 Remarks

- The approach behind this Matlab package is described in [1];
- The benchmark problems on which we tested the code can be accessed by opening the file **tdsplants.mat**. In particular, the data corresponding to Example 7 of [1] are contained in the structure **tds9**.

```
>> load tdsplants
>> tds9
```

References

[1] Z. Wu and W. Michiels, Reliably computing all characteristic roots of delay differential equations in a given right half plane using a spectral method, Report TW 596, Department of Computer Science, K.U. Leuven, Leuven, Belgium, May 2011.