Shiming Duan, 8/15/2010

LambertW_DDE Toolbox

V1.0 (for testing)

1. Introduction

This documentation provides a short instruction on how to apply the LambertW_DDE toolbox to solve the problem of delay differential equations (DDEs). The functionality of the toolbox is to calculate the solution for a given time-delay system, analyze the stability, observability and controlliablity and design a closed-loop control system with desired performance using the Lambert W function approaches introduced in the book and the supplementary webpages. These approaches are embedded in the functions of the toolbox, which hopefully facilitates the use of this book for users.

2. System requirement and installation

- The toolbox is developed and tested using Matlab 2009b
- Must have the "Symbolic Math Toolbox" and "Optimization Toolbox" for Matlab installed
- To use the toolbox, download the zip file and extract all the files inside to a folder

3. Main functions

The main functions of the toolbox are listed in Table 1:

Table 1 List of the main functions of the toolbox

Name	Description
lambertw_matrix	Calculate matrix Lambert W functions
find_Sk.m	Find $\mathbf{S_k}$ and $\mathbf{Q_k}$ for a given branch
find_CI.m	Calculate $\mathbf{C^I}$ under specific initial conditions for a given branch
find_CN.m	Calculate $\mathbf{C^N}$ for a given branch
pwcont_test.m	Controllability test for DDEs
pwobs_test.m	Observability test for DDEs
cont_gramian_dde.m	Calculate controllability gramian for DDEs
obser_gramian_dde.m	Calculate observability gramian for DDEs
place_dde.m	Rightmost eigenvalue assignment for DDEs
stabilityradius_dde.m	Calculate stability radius for DDEs

examples.m	List various examples for using this toolbox; each cell is a short example and can be evaluated separately (Ctr+Enter)

4. Examples

a. Calculate matrix Lambert W functions (`lambertw_matrix.m`)

- The function can handle repeated eigenvalues and the hybrid branch case
- To calculate the matrix Lambert W function with arguments W = `[1 1 1;0 1 2;0 0 3]` for branch -1 and W = `[1 1 1;0 1 2;0 0 0]` for branch 3:

```
>> lambertw_matrix(-1,[1 1 1;0 1 2;0 0 3])
>> lambertw_matrix(3,[1 1 1;0 1 2;0 0 0])
```

b. Find $\mathbf{S_k}$ (`find_Sk.m`)

- The function '`find_Sk`' finds the solution of S_k for certain branches
- When $\mathbf{A_d}$ is singular, one can fix the redundant elements in Q matrix and improve the speed of search
- To calculate the $\mathbf{S_k}$ for branch -1, 0, 1, 2 for the system with $\mathbf{A}$ = [0 1; -4.6985 0], $\mathbf{A_d}$= [0 0; -2 -3] and h =1:

```
>> h= 1; A = [0 1;-9.397/2 0]; Ad = [0 0 ;-2 -3];
>> Q_ini = [1 1;1 1]; N = -1:2
>> DDE_Sol = find_Sk(A,Ad,h,N,Q_ini);
```

If you note that the first row of Q is redundant, you can specify `Q_ini = [inf inf;1 1]`. In the optimization, the elements in Q with initial condition `inf` will be fixed to be 0.

c. Find $\mathbf{C^N}$ and $\mathbf{C^I}$ (`find_CN.m and find_CI.m`)

- Need to solve for $\mathbf{S_k}$ before using these two functions
- To find $\mathbf{C^I}$, the initial condition $\mathbf{x}_0$ and $\mathbf{g}(t)$ must be predetermined
- To calculate the $\mathbf{C^N}$ and $\mathbf{C^I}$ for branch -1 for the system with $\mathbf{A}$ = [-1 -3; 2 -5], $\mathbf{A_d}$= [1.66 -0.697;0.93 -0.330] and h=1 given $\mathbf{x}_0$ = [1;0] and $\mathbf{g}(t)$ = [sin(t);cos(t)]:

```
>> h= 1; A = [-1 -3;2 -5]; Ad = [1.66 -0.697;0.93 -0.330];
>> Q_ini = [inf inf;1 1]; N = -1;
>> DDE_Sol = find_Sk(A,Ad,h,N,Q_ini);
>> [CN, uniqueness] = find_CN(A,Ad,h,Sk);
>> syms t
>> g = [sin(t);0];x0=[1;0];
>> [CI, uniqueness] = find_CI(A,Ad,h,Sk,g,x0)
```

- `uniqueness`=1 means the coefficient has been obtained correctly
- For the hybrid branch case, the input S_k should be a scalar instead of a matrix.

d. Piecewise controllability and observability tests (`pwobs_test.m and pwcontr_test.m`)
- Return 1 or 0 for being piecewise controllable/observable or not
- To perform the test for a time-delay system, simply enter the coefficients and run the function:

```
>> A = [0 1;-9.397 0]; Ad = [0 0;-2 -3]; h=1; C = [0 1];
>> pwobs_test(A,Ad,C,h)
```

e. Calculate gramian (`contr_gramian_dde.m and obs_gramian_dde.m`)
- Assumes that $\mathbf{S_k}$ and $\mathbf{C^N}$ have been obtained
- To calculate the gramian for a specific time instant t_1:

```
>> load gramian_test.mat % load solutions to the DDE;
>> t1 = 4; % observability at t1 = 4 sec.
>> C = [0 1];
>> B = [0;1];
>> ob_gramm_lambert = obs_gramian_dde(Result(1:4),C,t1) %
approximate the observability gramain using the first 4 branches
>> ct_gramm_lambert = contr_gramian_dde(Result(1:4),B,t1) %
approximate the controllability gramain using the first 4
      branches
```

f. Stability radius (`stabilityradius_dde.m`)
- The coefficients $\mathbf{E}$, $\mathbf{F_1}$, $\mathbf{F_2}$ for the structured uncertainty must be determined first
- To calculate the stability radius, enter the coefficients of the system:

```
>> I=eye(2);B=[0;1];h=0.1;
>> E = I; F1 = I; F2 = I;
>> A=[0 0;0 1]; Ad=[-1 -1;0 -0.9];
>> sr = stabilityradius_dde(A,Ad,E,F1,F2,h)
```

g. Eigenvalue assignment (`place_dde.m`)
- There are 3 controller modes (**u= K*x, u = $\mathbf{K_d}$*$\mathbf{x_d}$ or u = K*x+$\mathbf{K_d}$*$\mathbf{x_d}$**)
- There are 1 observer mode ($\dot{\mathbf{e}} = (\mathbf{A} - \mathbf{LC})\mathbf{e} + \mathbf{A_d}\mathbf{e_d}$)
- To place the rightmost eigenvalue of a time delay system with $\mathbf{A}$ = [0 1; -4.6985 0], $\mathbf{A_d}$= [0 0; 0 0], **B**=[0;1] and h =1 with feedback **u = $\mathbf{K_d}$*$\mathbf{x_d}$**:

```
>> A = [0 1;-9.397/2 0]; Ad = [0 0;0 0];B = [0;1]; h =0.2;% open
loop
>> pole_desired=[-1+2i]; ;% desired rightmost eigenvalue
>> Q_ini = [inf inf;1 1] ;% initial condition for Q; first row is
redundant
>> Kd_ini = [0 0];% initial condition for Kd
>> contr_mode = 2; % u = Kd*x(t-h)
>> Kd = place_dde(A,Ad,B,h,pole_desired,contr_mode,Kd_ini,Q_ini);
```

For more examples, please check out `example.m`

For more information about a certain function, please refer to the comments on the top of the function or enter

```
>> help func_name
```