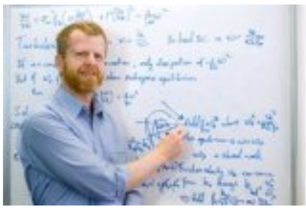

Construct centre manifolds of ordinary or delay differential equations (autonomous)

by Prof A. J. Roberts

anthony.roberts@adelaide.edu.au

Part of the research underlying this web service was funded by the Australian Research Council

Contact me
Questions marked by * are required.
Name: *
Email: *
Message:

Your input data to be sent for processing

```
% 101.166.166.211 Friday 28th June 2019 08:07:14 PM
ff_:=tp mat((u3*u2,u3*u1-u4*u1/small
,0,u5,-u4));
freqm_:=mat((0,0,0,1,-1));
ee_:=tp mat((1,0,0,0,0),
(0,1,0,0,0),
(0,0,1,0,0),
(0,0,0,1,i),
(0,0,0,1,-i));
zz_:=tp mat((1,0,0,0,0),
(0,1,0,0,0),
(0,0,1,0,0),
(0,0,0,1,i),
(0,0,0,1,-i));
toosmall:=5;
factor small,s(3);
end;
```

Use your browser's back button to return to the form to rerun your data with any modifications.

Writing data to file

Success, wrote your data to file; attempting execution.

Processing data

Computer algebra result

Download a pdf version of the following report
(and/or download the pdfLaTeX source for your editing).

```
total no. of modes  n := 5

no. of centre modes m := 5
```

Submit Reset

```
Input dimensions are OK

Normalising the left-eigenvectors:

       [1  0  0    0       0    ]
       [                        ]
       [0  1  0    0       0    ]
       [                        ]
       [0  0  1    0       0    ]
       [                        ]
zz_ := [           1       1    ]
       [0  0  0   ---     ---   ]
       [           2       2    ]
       [                        ]
       [          1         1   ]
       [0  0  0  ---*i   - ---*i]
       [          2         2   ]


***** sub

Find the linear operator is

       [        0           0  0  0  0]
       [                              ]
       [        -1                    ]
       [ - small  *u4 + u3  0  0  0  0]
       [                              ]
ll_ := [        0           0  0  0  0]
       [                              ]
       [        0           0  0  0  0]
       [                              ]
       [        0           0  0  0  0]


Check centre subspace linearisation

WARNING: I NEED TO ADJUST LINEAR OPERATOR

         [        0           0  0  0  0 ]
         [                               ]
         [        -1                     ]
         [ - small  *u4 + u3  0  0  0  0 ]
         [                               ]
lladj := [        0           0  0  0  0 ]
         [                               ]
         [        0           0  0  0  -1]
         [                               ]
         [        0           0  0  1  0 ]


       [0  0  0  0   0]
       [              ]
       [0  0  0  0   0]
       [              ]
ll_ := [0  0  0  0   0]
       [              ]
       [0  0  0  0   1]
       [              ]
       [0  0  0  -1  0]


Analyse ODE/DDE system du/dt =

[     small*u2*u3      ]
[                      ]
[ - u1*u4 + small*u1*u3]
[                      ]
[          0           ]
[                      ]
```

```
[          u5          ]
[                      ]
[         - u4         ]


*** s already defined as operator

Start iterative construction of centre manifold

lengthres := {4,1,1}

Time: 10 ms

lengthres := {4,1,1}

Time: 0 ms

lengthres := {7,1,1}

Time: 20 ms

lengthres := {13,1,1}

Time: 10 ms

lengthres := {25,1,1}

Time: 29 ms

lengthres := {26,1,1}

Time: 51 ms

lengthres := {22,1,1}

Time: 80 ms

lengthres := {13,1,1}

Time: 70 ms

lengthres := {7,1,1}

Time: 80 ms

lengthres := {1,1,1}

Time: 70 ms  plus GC time: 10 ms

iter := 10010

SUCCESS: converged to an expansion

The centre manifold is (to one order lower)

u1 = s(1)

      + s(3)*small*(cis(t)*s(4)*s(1) + cis( - t)*s(5)*s(1))

          2      2
     + s(3) *small *(2*cis(t)*s(4)*s(2)*i

         1              2
       + ---*cis(2*t)*s(4) *s(1) - 2*cis( - t)*s(5)*s(2)*i
         4

         1                  2             3      3
       + ---*cis( - 2*t)*s(5) *s(1)) + s(3) *small *(
         4

                                   25                  2
```

```
         - 4*cis(t)*s(4)*s(1) + ----*cis(t)*s(5)*s(4) *s(1)
                                  4

            3              2
         + ---*cis(2*t)*s(4) *s(2)*i
            4

            1               3
         + ----*cis(3*t)*s(4) *s(1) - 4*cis( - t)*s(5)*s(1)
            36

            25                2
         + ----*cis( - t)*s(5) *s(4)*s(1)
            4

            3                  2
         - ---*cis( - 2*t)*s(5) *s(2)*i
            4

            1                   3
         + ----*cis( - 3*t)*s(5) *s(1))
            36

u2 = s(2) + cis(t)*s(4)*s(1)*i - cis( - t)*s(5)*s(1)*i +

     s(3)*small*( - cis(t)*s(4)*s(2)

                     1               2
                   + ---*cis(2*t)*s(4) *s(1)*i
                     2

                   - cis( - t)*s(5)*s(2)

                     1                   2              2
                   - ---*cis( - 2*t)*s(5) *s(1)*i) + s(3)
                     2

          2
     *small *( - 2*cis(t)*s(4)*s(1)*i

                  9                  2
                + ---*cis(t)*s(5)*s(4) *s(1)*i
                  4

                  5               2
                - ---*cis(2*t)*s(4) *s(2)
                  4

                  1                3
                + ----*cis(3*t)*s(4) *s(1)*i
                  12

                + 2*cis( - t)*s(5)*s(1)*i

                  9                2
                - ---*cis( - t)*s(5) *s(4)*s(1)*i
                  4

                  5                  2
                - ---*cis( - 2*t)*s(5) *s(2)
                  4

                  1                   3             3
                - ----*cis( - 3*t)*s(5) *s(1)*i) + s(3)
                  12

          3                                         2
     *small *(4*cis(t)*s(4)*s(2) - 3*cis(t)*s(5)*s(4) *s(2)

                  11               2
                - ----*cis(2*t)*s(4) *s(1)*i
```

```
                 4

                 79                   3
              + ----*cis(2*t)*s(5)*s(4) *s(1)*i
                 18

                 5                3
              - ----*cis(3*t)*s(4) *s(2)
                 18

                  1                4
              + -----*cis(4*t)*s(4) *s(1)*i
                 144

              + 4*cis( - t)*s(5)*s(2)

                                 2
              - 3*cis( - t)*s(5) *s(4)*s(2)

                 11                   2
              + ----*cis( - 2*t)*s(5) *s(1)*i
                 4

                 79                   3
              - ----*cis( - 2*t)*s(5) *s(4)*s(1)*i
                 18

                 5                    3
              - ----*cis( - 3*t)*s(5) *s(2)
                 18

                  1                   4
              - -----*cis( - 4*t)*s(5) *s(1)*i)
                 144

u3 = s(3)

u4 = cis(t)*s(4) + cis( - t)*s(5)

u5 = cis(t)*s(4)*i - cis( - t)*s(5)*i

The evolution of the real/complex amplitudes

ds(1)/dt = s(3)*small*s(2)

                                                          3
ds(2)/dt = s(3)*small*(s(1) - 2*s(5)*s(4)*s(1)) + s(3)

                3
           *small

                                       25      2     2
           *(8*s(5)*s(4)*s(1) - ----*s(5) *s(4) *s(1))
                                       2

ds(3)/dt = 0

ds(4)/dt = 0

ds(5)/dt = 0
```

In the results

- The centre manifold is parametrised by m slow variables `s(j)`.
- Each such slow variable `s(j)` is either a 'real' slow variable or a complex amplitude of an oscillatory mode $e^{i\omega t}$ for some frequency ω.
- In a real system of ODE/DDEs, the complex amplitudes occur in

complex conjugate pairs.

- I use the variable `small` (also appearing as ε) to control and order the asymptotic expansion: introducing `small` into your definition of the `nonlinear' function empowers finer control of the asymptotics. For example, in the delay ODE example the parameter $\delta$ is made 'small'. Analogously, cubic terms may best be made `small' when added to quadratic terms.
- The code does cater for degenerate cases involving generalised eigenvectors. But the code does this by modifying L, the matrix of the linearisation at the origin. The code attempts to make the smallest modification it can to remove the degeneracy, and flags the change with variable 'small' so you recover the original with small $= 1$. *Be careful* that the results are relevant to what you want.
- Similarly, the code tries to make all, except the nominated frequencies, of $\vec z^* L\vec e$ to be 'small', and also makes $\vec f(\vec 0)$ `small'. *Be very careful* that the results are relevant to what you want.
- For explanations and relevant theory, see my book Modelling emergent dynamics in complex systems, or Low- dimensional modelling of dynamics via computer algebra, or the classic Simple examples of the derivation of amplitude equations for systems of equations possessing bifurcations.
- In the case of a slow manifold, the code also computes a basis of normal vectors to the isochrons (actually normal to their tangent space at the slow manifold). Use these basis vectors:
    - to project initial conditions onto the slow manifold;
    - to project non-autonomous forcing onto the slow evolution;
    - to predict the consequences of modifying the original system; and
    - in uncertainty quantification to quantify effects on the model of uncertainties in the original system.

  For explanations and relevant theory, see Computer algebra derives correct initial conditions for low-dimensional dynamical models, or the classic Appropriate initial conditions for asymptotic descriptions of the long term evolution of dynamical systems.

If you like this web page, please link to it so others can find it more easily.