Homework No.3

Osamu Katagiri-Tanaka : A01212611

August 30, 2020

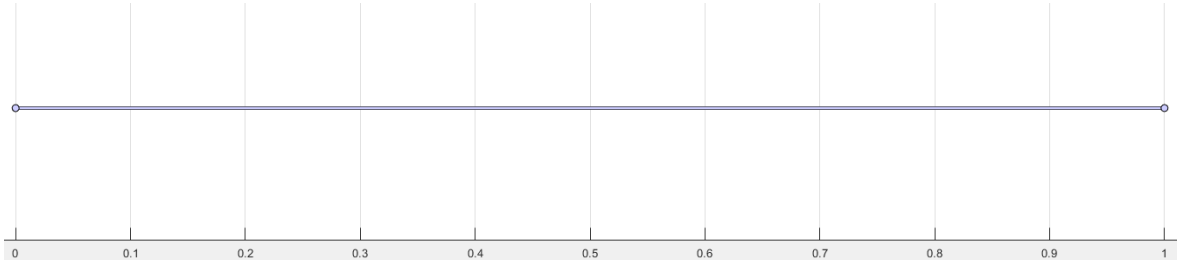
1 Problem Statement

The equation of conservation of chemical species under a chemical reaction of decomposition can be represented with the PDE given below.

$$\frac{\partial C}{\partial t} = \vec{\nabla} \cdot (D\vec{\nabla}C) - \vec{v} \cdot \vec{\nabla}C - kC^n$$

If a tubular catalytic chemical reactor initially filled with an inert solvent ($C = 0$) is fed by a stream of component "A" with a concentration of $1kmol/m^3$ ($C = 1$) and speed of $1m/s$ ($v = 1$), calculate the distribution of "A" across the reactor and as a function of time $C(x, t)$. The dispersion coefficient of the component "A" is $0.02m^2/s$ ($D = 0.01$), the kinetic decomposition coefficient $0.05s^{-1}$ ($k = 0.05$). The chemical decomposition kinetics is first order ($n = 1$).

2 Sketch



3 Assumptions and Approximations

- The reactor is 1-m long

4 Physical constants

Not applicable

5 Physical Transport or Thermodynamic Properties

- Conservation of chemical species under a chemical reaction
- Molar balance in a 1D flow
- 1D convection and diffraction

6 Calculations

The molar balance in axial direction for a 1D flow can be written as:

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} - v\frac{\partial C}{\partial x} - kC^n$$

The initial condition IC is:

$$C|_{t=0} = 0,\ 0 \leq x \leq 1$$

The boundary conditions BCs are:

$$C|_{x=0} = 1,\ t > 0$$

$$\left.\frac{\partial C}{\partial t}\right|_{x=L} = 0,\ t \geq 0$$

Where,
D is the diffusion coefficient
C is the injection concentration
v is the velocity of fluid injection
k is the first order kinetic coefficient
L is the length of domain
t is the simulation time
x is the distance mesh

6.1 PDEPE solver

The built in function PDEPE, solves a general problem of a 1-D (parabolic or elliptic) partial differential equation, for a Cartesian, cylindrical or spherical coordinates of the from:

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t} = \frac{1}{x^m}\frac{\partial}{\partial x}\left(x^m f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right) + s\left(x,t,u,\frac{\partial u}{\partial x}\right)$$

Where,
$m = 0$ represents the symmetry of the problem (0 for slab, 1 for cylindrical, or 2 for spherical)
$c = 1$ is a diagonal matrix
$f = D\frac{\partial u}{\partial x}$ is the flux term
$s = -v\frac{\partial u}{\partial x} - ku^n$ is the source term
c, f, and s correspond to coefficients in the standard PDE equation form expected by pdepe. These coefficients are coded in terms of the input variables x, t, u, and $dudx$. Listing 1 implements a function that calculates the values of the coefficients c, f, and s.

```
function [c, f, s] = DiffusionPDEfun(x, t, u, dudx, P)
    % Parameters
    D  = P(1);
    k  = P(3);
    vo = P(4);

    % PDE
    c = 1;
    f = D .* dudx;
    s = - k * u - vo * dudx;
```

```
end
```

Listing 1: PDE function for equations

PDEPE requires an 'initial condition function', which is defined as a function that defines the initial condition. For $t = t_o = 0$ and all x, the solution satisfies the initial condition of the form:

$$u(x, t_o) = u_o(x)$$

PDEPE calls the initial condition function with an argument x, which evaluates the initial values for the solution at x in vector u_o. The number of elements in u_o is equal to the number of equations. Listing 2 implements the constant initial condition.

```
function u0 = DiffusionICfun(x, P)
    % u0 = u0(x)
    u0 = 0;
end
```

Listing 2: Initial condition function

The third function required by the PDEPE solver is the 'boundary condition function'. The boundary condition function spefifies the boundary conditions for all t, the solution satisfy the boundary condition of the form:

$$p(x, t, u) + q(x, t)f\left(x, t, u, \frac{\partial u}{\partial x}\right) = 0$$

Listing 3 implements a function that defines the terms p and q of the boundary conditions. $u1$ is the approximate solution of the left boundary, ur is the approximate solution of the right boundary, pl and ql are vectors corresponding to p and q evaluated at xl, and pr and qr are vectors corresponding to p and q evaluated at xr.

```
function [pl, ql, pr, qr] = DiffusionBCfun(xl, u1, xr, ur, t, P)
    % BCs: No flux boundary at the right boundary and constant
    % concentration on the left boundary
    c0 = P(2);
    pl = u1 - c0;
    ql = 0;
    pr = 0;
    qr = 1;
end
```

Listing 3: Boundary condition function

Listing 4 calls the PDEPE solver with the parameters P specified in the 'Problem Statement' section. The PDE function of equations, initial condition function, and boundary condition function are passed to the *pdepe()* function. Matlab's *pdepe()* solves a system of PDEs with one spatial variable and time. The equations being solved are specified in Listing 1, the initial value is coded in Listing 2, and the boundary conditions are defined in Listing 3. The ODEs resulting from the discretization in space are integrated at the time values t. *pdepe()* returns values of the solution on a mesh provided in x.

```
% Parameters
P(1) = 0.01;                    % Diffusion coefficient D
P(2) = 1.0;                     % Injection concentration c0
P(3) = 0.5;                     % First order kinetic coefficient k
P(4) = 1.0;                     % Velocity of fluid injection vo
L    = 1;                       % Length of domain
maxt = 1;                       % Max. simulation time
m    = 0;                       % Parameter corresponding to the symmetry
%                                 of the problem (0 for slab, 1 for
%                                 cylinder, or 2 for sphere)
step = 32;
t    = linspace(0, maxt, step); % Tspan
x    = linspace(0, L, step);    % xmesh

sol = pdepe(          ...
    m,                ...
    @DiffusionPDEfun, ... % Function containing the PDEs
    @DiffusionICfun,  ... % Function containing the ICs for t=0 at all x
    @DiffusionBCfun,  ... % Function containing the BCs for x=0 and x=L
    x,                ... % Spatial mesh
    t,                ... % Time span of integration
    [],               ... % Options
    P                 ... % Parameters
);

% ui(t,x) = (tspan(j),xmesh(k)).
u = sol;
```

Listing 4: PDEPE solver

Listings 5 and 6 visualize *pdepe()* return values. Figures 1 and 2 are the generated plots with the PDEPE results.

```
% plot limits
dlim = 0.02;
x_lim = [0 - dlim, L + dlim];
t_lim = [0 - dlim, maxt + dlim];
u_lim = [0 - dlim, P(2) + dlim];

figure
for n = linspace(1, step, step)
    hold all
    plot(x, u(n, :))
end
xlabel('Distance x')
ylabel('Species u')
axis([x_lim(1) x_lim(2) u_lim(1) u_lim(2)])
```

Listing 5: PDEPE Distance vs. Species plot

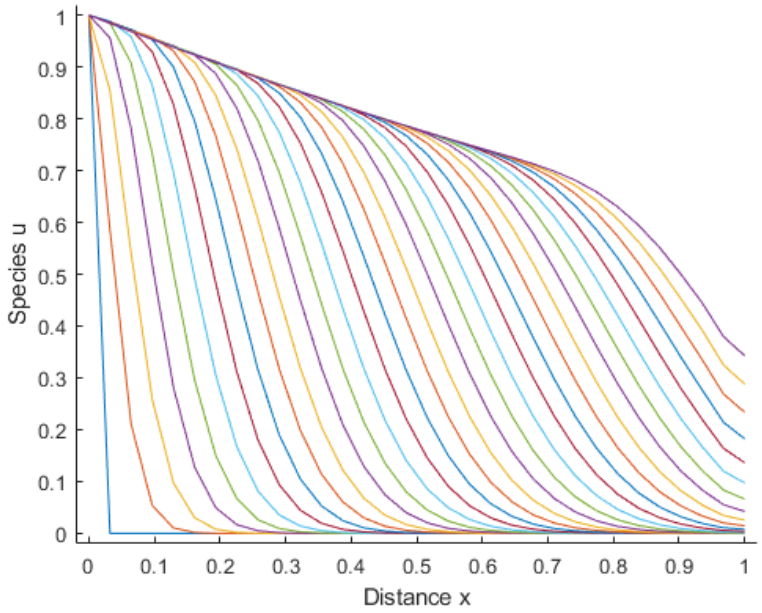


Figure 1: PDEPE - Species vs. Distance

```
figure
for n = linspace(1, step, step)
    hold all
    plot(t, u(:, n))
end
xlabel('Time t')
ylabel('Species u')
axis([t_lim(1) t_lim(2) u_lim(1) u_lim(2)])
```

Listing 6: PDEPE Time vs. Species plot

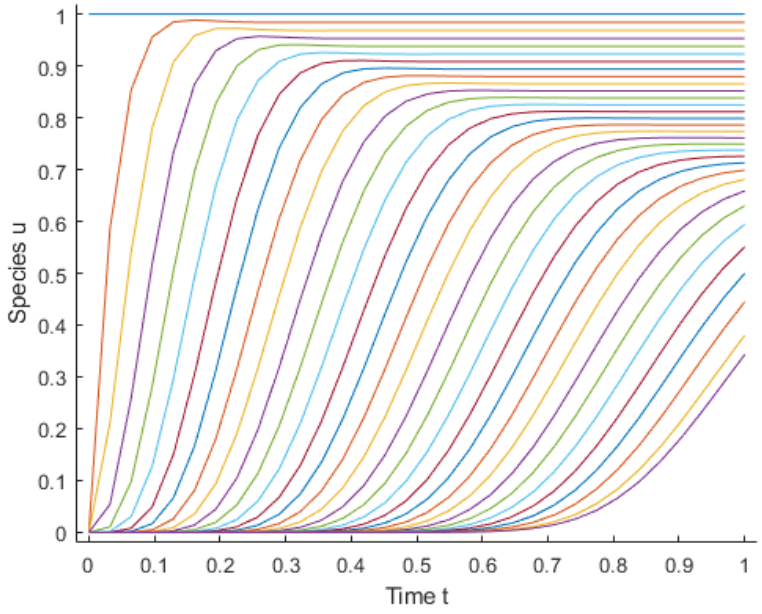


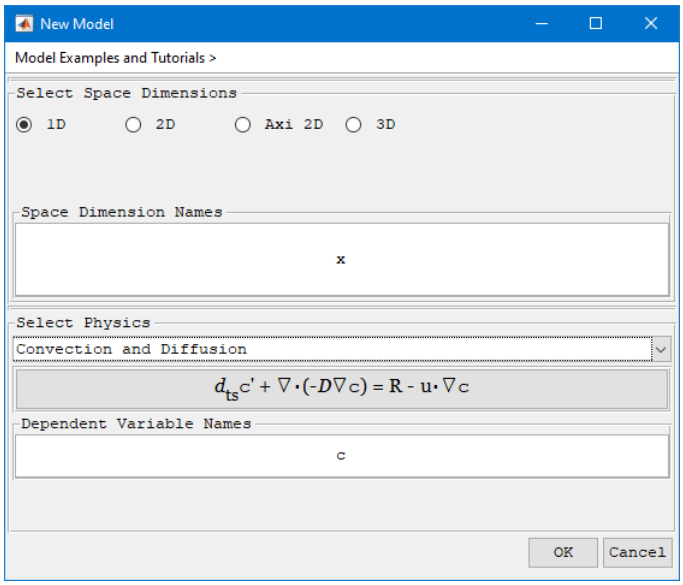
Figure 2: PDEPE - Species vs. Time

6.2 FEATool solver

Problem statement can be also tacked using *FEATool*. The following steps were taken to solve the "tubular water treatment uv-light reactor

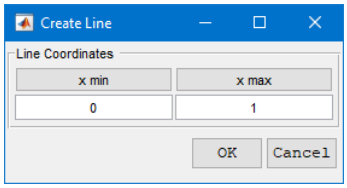
Initialization and model selection:

1. To start a new model click the New Model toolbar button, or select New Model... from the File menu.
2. Select the 1D radio button.
3. Select the Convection and Diffusion physics mode from the Select Physics drop-down menu.
4. Press OK to finish the physics mode selection.



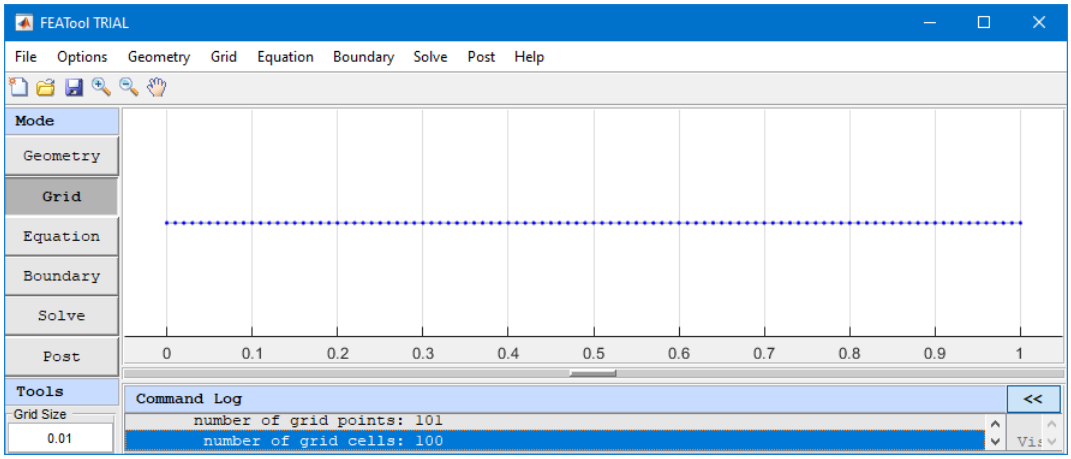
Geometry definition:

5. Press the Create line Toolbar button.
6. Press OK to finish and close the dialog box.



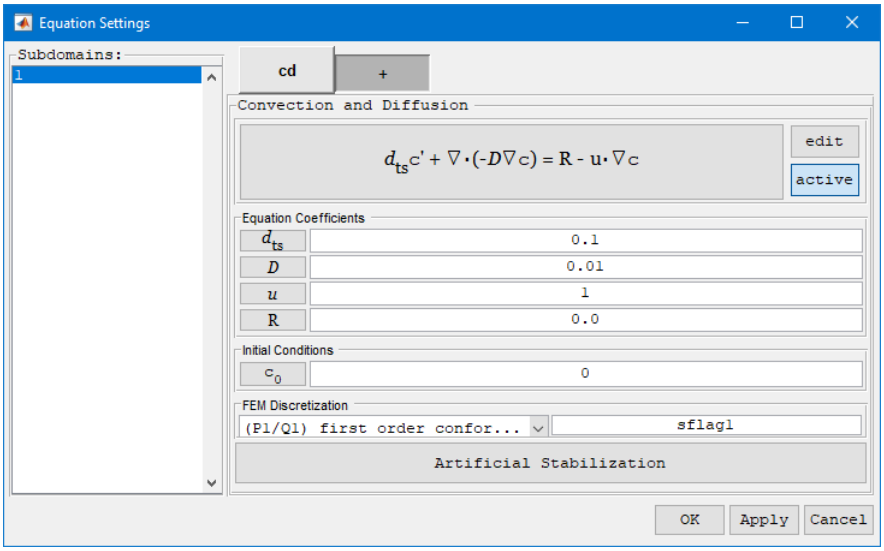
Grid specification:

7. Switch to Grid mode by clicking on the corresponding Mode Toolbar button.
8. Enter 0.01 into the Grid Size edit field.



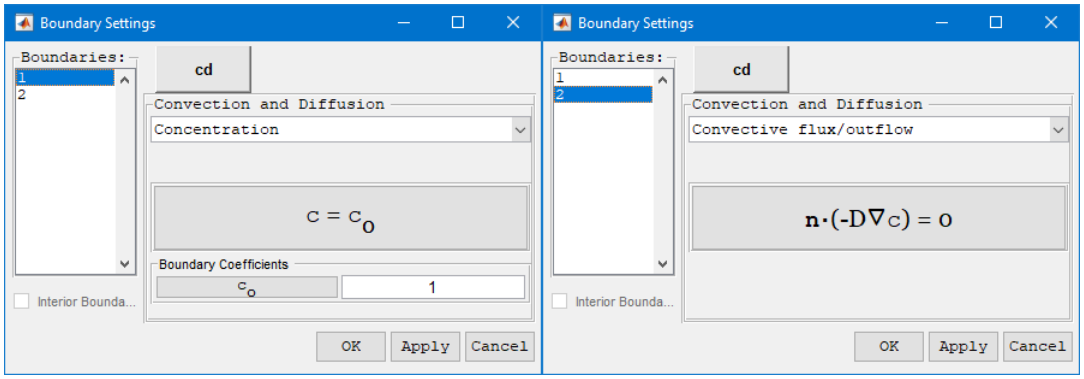
PDE parameters and initial condition:

9. Switch to Equation mode by clicking on the corresponding Mode Toolbar button.
10. Enter 0.1 into the Time scaling coefficient edit field.
11. Enter 0.01 into the Diffusion coefficient edit field.
12. Enter 1 into the Convection velocity in x-direction edit field.
13. Enter 0.0 into the Reaction rate edit field.
14. Press the Apply button.
15. Press OK to finish the equation and subdomain settings specification.



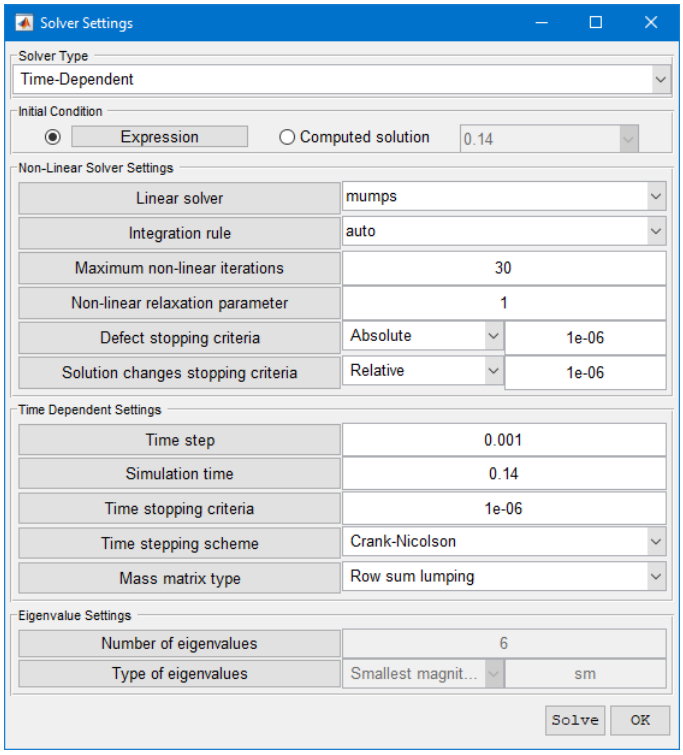
Boundary conditions:

16. Switch to Boundary mode by clicking on the corresponding Mode Toolbar button.
17. Select Concentration from the Convection and Diffusion drop-down menu.
18. Enter 1 into the Concentration edit field.
19. Select 2 in the Boundaries list box.
20. Press the Apply button.
21. Press OK to finish the boundary condition specification.



Solver settings:

22. Switch to Solve mode by clicking on the corresponding Mode Toolbar button.
23. Press the Settings Toolbar button.
24. Select Time-Dependent from the Solution and solver type drop-down menu.
25. Enter 30 into the Maximum number of non-linear iterations edit field.
26. Enter 0.001 into the Time step size edit field.
27. Enter 0.001 into the Duration of time-dependent simulation (maximum time) edit field.
28. Press the Solve button.



Solve for different time values:

29. Switch to Solve mode by clicking on the corresponding Mode Toolbar button.
30. Press the Settings Toolbar button.
31. Enter 0.010 into the Duration of time-dependent simulation (maximum time) edit field.
32. Press the Solve button.

Repeat steps 29 through 32 for different values of "Duration of time-dependent simulation (maximum time)" from 0.020 to 0.140 with 0.010 increments

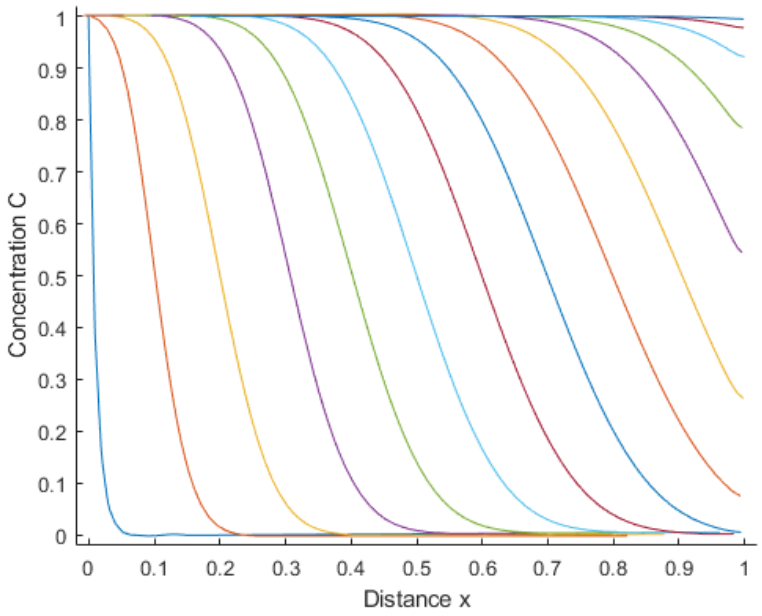


Figure 3: FEATool - Concentration vs. Distance for time t = [0.001, 0.140]

7 Discussion

The transient 1-D tubular reactor problem has been solved by two methods: using Matlab built-in solver PDEPE (Figure 1) and using the FEATool software (Figure 3). As depicted in the generated plots, the FEATool solution states that eventually the whole geometry will be at concentration $C = 1$. On the other hand, PDEPE solution suggest that steady state, the further the distance from boundary 1 (with constant concentration $C = 1$) the lower the concentration is, being concentration $C = 0.4$ the highest at distance 1.

Procedure suggests that differences are due to the boundaries being programmed in different ways within the two methods. On one hand, FEATool implements a "convective flux/outflow" boundary of the form $n \cdot (-D\nabla c) = 0$. On the other hand, PDEPE solver requires a m scalar to indicate the symmetry of the problem. When $m = 1$ parameters pl and ql are ignored by the solver, hence the difference.

Appendix: FEATool script


```
{"meta":{"app":"FEATool Multiphysics","author":"","build":"20.08.221","date":
    "30-Aug-2020","descr":"HW03_FEATool","dim":1,"image":"","keyw":["conv","
    diff"],"mlver":"9.5.0.944444 (R2018b)","name":"HW03_FEATool","phys":["
    Convection and Diffusion"],"system":"PCWIN64","time":738033.60121716431,"
    title":"HW03_FEATool","type":"Convection and Diffusion","user":"","ver"
    :[1,12,4]},
"fields":["type","id","ui_arg","fcn_type","fcn_oarg"],
"data":[
["uipushtool","Standard.NewFigure",[],"ClickedCallback",[]],
["uicontrol","radio_1d",1,"Callback",[]],
```

```
["uicontrol","popup_physsel",["Convection and Diffusion"],"Callback",[]],
["uicontrol","button_dlgnew_ok",[],"Callback",[]],
["uicontrol","button_line",[],"Callback",[]],
["uicontrol","button_linegeom_ok",[],"Callback",[]],
["uicontrol","button_grid_mode",1,"Callback",[]],
["uicontrol","grid_hmax","0.01","Callback",[]],
["uicontrol","button_equation_mode",1,"Callback",[]],
["uicontrol","dts_cd","0.1","Callback",[]],
["uicontrol","d_cd","0.01","Callback",[]],
["uicontrol","u_cd","1","Callback",[]],
["uicontrol","r_cd","0.0","Callback",[]],
["uicontrol","button_dlgeqn_apply",[],"Callback",[]],
["uicontrol","button_dlgeqn_ok",[],"Callback",[]],
["uicontrol","button_boundary_mode",1,"Callback",[]],
["uicontrol","popup_selbc_cd",["Concentration"],"Callback",[]],
["uicontrol","edit_bccoef1_cd","1","Callback",[]],
["uicontrol","list_seldom",["2"],"Callback",[]],
["uicontrol","button_dlgbdr_apply",[],"Callback",[]],
["uicontrol","button_dlgbdr_ok",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","solver",[" Time-Dependent"],"Callback",[]],
["uicontrol","maxnit","30","Callback",[]],
["uicontrol","tstep","0.001","Callback",[]],
["uicontrol","tmax","0.001","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.010","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.020","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.030","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.040","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.050","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.060","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.070","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
```

```
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.080","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.090","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.100","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.110","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.120","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.130","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]],
["uicontrol","button_solve_mode",1,"Callback",[]],
["uicontrol","button_solver_settings",[],"Callback",[]],
["uicontrol","tmax","0.140","Callback",[]],
["uicontrol","button_dlgsolversettings_solve",[],"Callback",[]]
]}
```

Appendix: FEATool data extraction

Since the FEATool trial version was used in this activity, the solution data could not be extracted directly from FEATool. Data was extracted with WebPlotDigitizer. Data is available in the CSV file (FEATool.csv) attached to this activity. Listing 7 was used to create Figure 3.

```
% Import data
FEATool_exports = csvread('./FEATool.csv');

% Plotting
N = size(FEATool_exports, 2)/2; % number of curves

% plot limits
dlim = 0.02;
x_lim = [0 - dlim, 1 + dlim];
t_lim = [0 - dlim, 1 + dlim];
u_lim = [0 - dlim, 1 + dlim];

% 2D line plot
% for time t = [0.001, 0.140]
figure
for n = linspace(1, N, N)
    data = FEATool_exports(:, n*2 - 1:n*2);
```

```
    data = sortrows(data,1);
    x = data(:, 1); % distance
    u = data(:, 2); % species/concentration
    hold all
    plot(x, u)
end
xlabel('Distance x')
ylabel('Concentration C')
axis([x_lim(1) x_lim(2) u_lim(1) u_lim(2)])
```

Listing 7: PDEPE Time vs. Species plot code