Table of Contents

Intro to user-defined functions

```
% ReactorLab.net
```

Why write user-defined functions?

```
% As an example of NOT using a user-defined function,
% let's say you want to do the same calculation several
% places in a program

% first time used somewhere in a big program
a = 1;
b = 2;
c = a .^ 2 + b

% next time used somewhere else in a big program, etc., etc.
a = 1.5;
b = 2.5;
c = a .^ 2 + b

% IF YOU NEED TO CHANGE the calculation, you have to find it many
% places in a program and make the same change in every place,
% lot's of work and prone to error.

% INSTEAD OF DOING THIS, write a user-defined function
% in a separate m-file, then use it ("call" it) with the same
% line of code everywhere in the main program.
% NOW if you need to change the calculation, you only have to
% make the change in ONE PLACE, the function m-file.

% first time used somewhere in a big program
a = 1;
b = 2;
cc = myFunFunc(a,b)

% next time used somewhere else in a big program, etc., etc.
a = 1.5;
b = 2.5;
cc = myFunFunc(a,b)

% type command "lists" contents of file so we see it when Publish
type myFunFunc.m
```

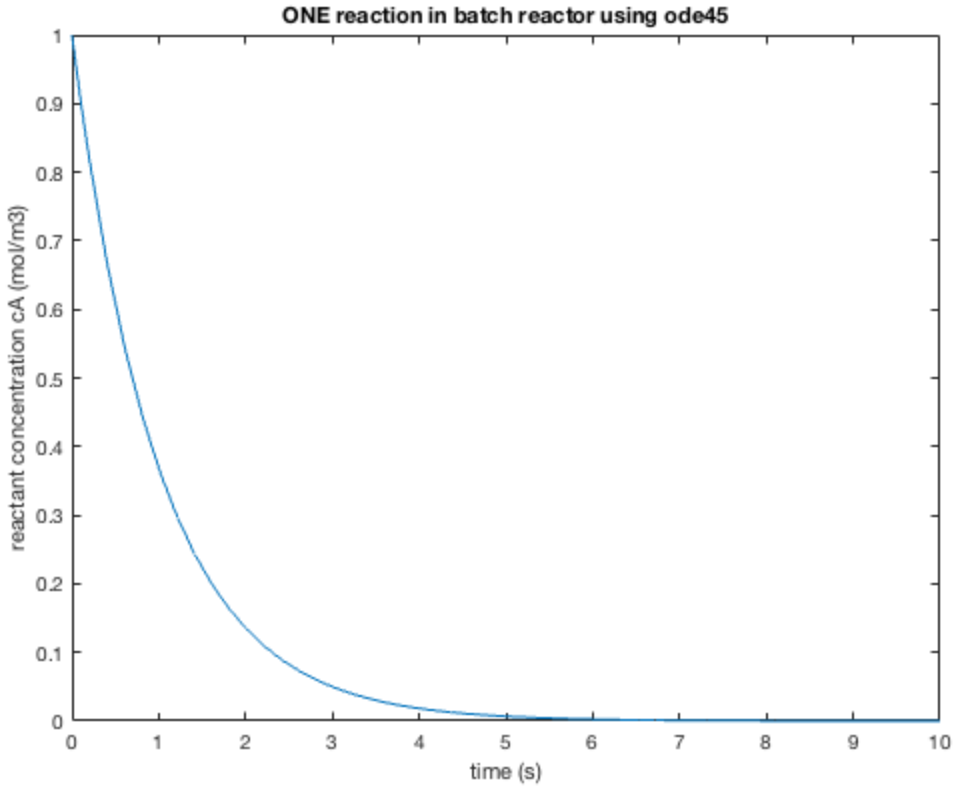
```
c =

     3


c =

    4.7500


cc =

     3


cc =

    4.7500
```

```
function result = myFunFunc(x,y)
% The first block of comments starting line 2 in a function file
% displays with the Matlab "help" command
%
% Function myFunFunc(x,y) first input argument is a scalar or array
% second argument is a scalar or is an array of same size as
% first argument; returns x .^ 2 + y;

% Line 1 format: function returnValues = functionName(input arguments)

% function name must be same as m-file name (+ m-file has .m
 extension)

% the names used here are "local" to this file
% the values in the input argument list are passed to function
% by position in the order in which they are listed, not by name

result = x .^ 2 + y; % use dot operator to handle array inputs
```

User-defined function used with standard Mat-lab function

```
% one chemical reaction in batch reactor
% integrated numerically using standard function ode45
% reaction is A > P (equilibrium almost all P)
% rate of production of A is rA = -k*cA
% component balance on A in constant volume batch reactor
% dcA/dt = rA = -k*cA

clear all
```

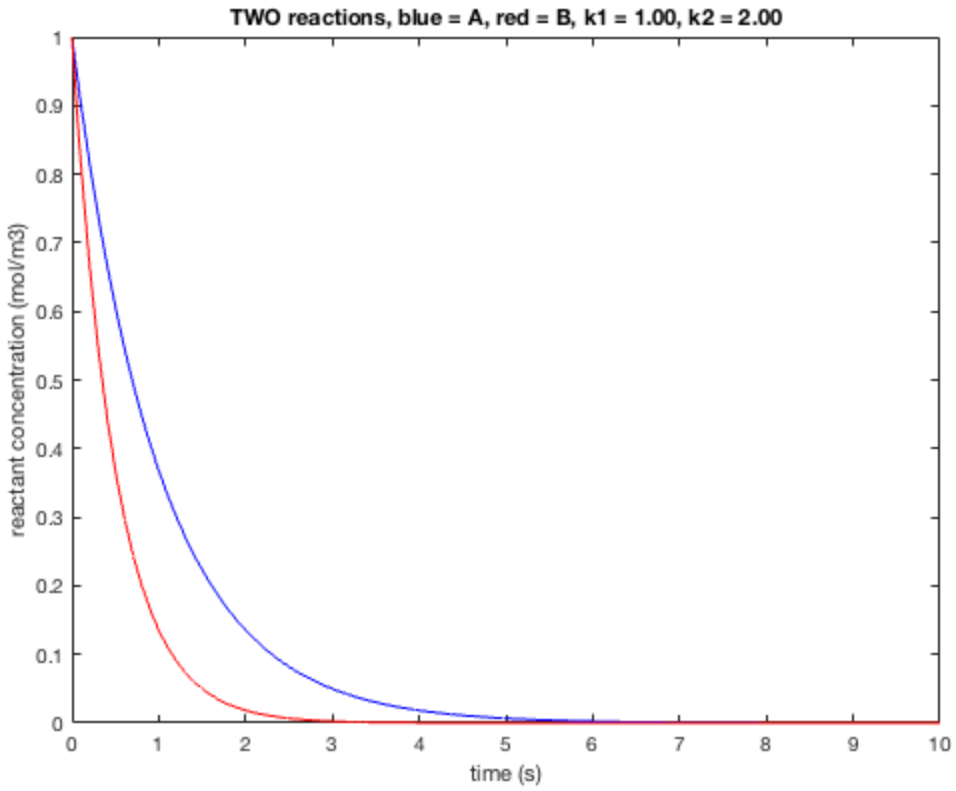
```
cA0 = 1; % mol/m3, initial conc (initial value of dependent variable)
tspan = [0 10]; % s, time span (span of independent variable)

% ode45 is a standard Matlab function
% that needs our function 'batchFunc' as an input

% in this example, the rate coefficient k value is specified in
% batchFunc.m so don't need to pass in this call to ode45

[t,cA] = ode45('batchFunc',tspan,cA0);

plot(t,cA)
title('ONE reaction in batch reactor using ode45');
ylabel('reactant concentration cA (mol/m3)')
xlabel('time (s)')

type batchFunc


function dcdt = batchFunc(t,c)
% for chemical reaction A > P (equilibrium almost all P)
% returns dcdt = dc/dt (can't use / in variable name)
% where t is time, c is conc of A, k is rate coefficient

% specify parameter value
k = 1; % 1/s, rate coefficient

dcdt = -k*c; % mol/m3/s
```



User-defined function - integrate TWO simultaneous ODEs

```
% TWO chemical reactions in batch reactor
% this example is called an independent multiple reaction system
% reaction 1 is A > P1 (equilibrium almost all P1)
% reaction 2 is B > P2 (equilibrium almost all P2)
% rate of production of A is rA = -k1*cA
% rate of production of B is rB = -k2*cB
% component balances in constant volume batch reactor
% dcA/dt = rA = -k1*cA
% dcB/dt = rB = -k2*cB

clear all

% in this example, we specify rate coefficient values in the
% main program and then pass them to our function batch2Func in
% the call to standard function ode45

k = [1 2]; % 1/s, rate coefficients

c0 = [1 1];     % mol/m3, initial concentations of A and B
tspan = [0 10]; % s, time span over which to integrate
```

```
% use this method to pass parameter values to user-defined function
[t,c] = ode45(@(t,c) batch2Func(t,c,k),tspan,c0,k);

% pull out results into separate column vectors for plotting
cA = c(:,1);
cB = c(:,2);

plot(t,cA,'b',t,cB,'r')

% use sprintf to display parameter values in plot title
tt = sprintf('TWO reactions, blue = A, red = B, k1 = %3.2f, k2 =
 %3.2f',k);
title(tt)

ylabel('reactant concentration (mol/m3)')
xlabel('time (s)')

type batch2Func


function dcdt = batch2Func(t,c,k)
% example of function file with more than one line of calculations

% ode45 function requires result returned as a column vector

dcdt(1,1) = -k(1)*c(1); % mol/m3/s, current dcA/dt value
dcdt(2,1) = -k(2)*c(2); % mol/m3/s, current dcB/dt value
```

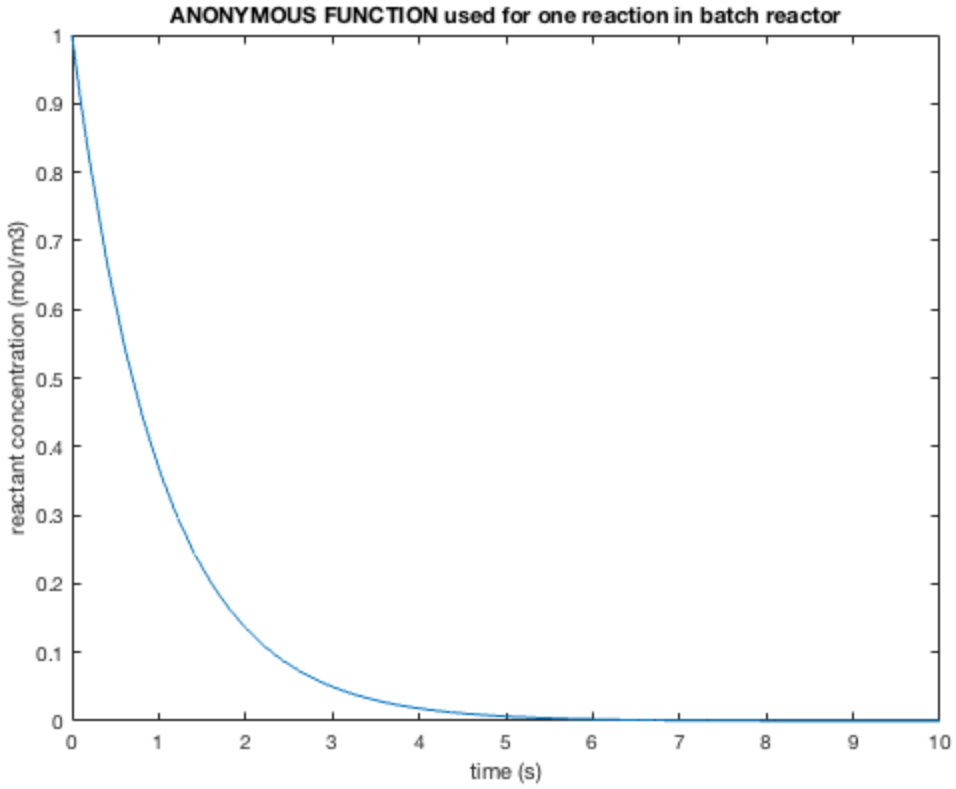


One-line "anonymous" user-defined function

```
% when a function can be defined in one line of code, you
% can define it as an "anonymous" function in the main program
% and do not need a separate m-file

% chemical reaction in batch reactor
% integrated numerically an "anonymous" function
% reaction is A > P (equilibrium almost all P)
% rate of production of A is rA = -k*cA
% component balance on A in a constant volume batch reactor
% dcA/dt = rA = -k*cA

clear all

k = 1;          % 1/s, rate coefficient
cA0 = 1;        % mol/m3, initial concentration of A
tspan = [0 10]; % s, time span

% since only one derivative, we can use an anonymous function (one
 line)
% FORMAT: variableName = @(input arguments) math expression
% note variable t not used here in expression on right side
%    but it could be used
```

```
batchFA = @(t,x) -k*x;

% well, at least the right-hand side is "anonymous"

% test this - it works
c = batchFA(0,1)

% now use anonymous function as argument of standard function ode45

% this works but deactivate by % to test the version below
% [t,cA] = ode45(batchFA,tspan,cA0); % note no '' around named anon
 func

% put anonymous function directly into argument list
% now it really is "anonymous"
[t,cA] = ode45(@(t,x) -k*x,tspan,cA0); % note no '' around anon func

plot(t,cA)
title('ANONYMOUS FUNCTION used for one reaction in batch reactor')
ylabel('reactant concentration (mol/m3)')
xlabel('time (s)')
```

```
c =

    -1
```



Published with MATLAB® R2016a