CBE 450 Chemical Reactor Fundamentals

Fall, 2011

Sample Codes

David Keffer
Department of Chemical and Biomolecular Engineering
University of Tennessee
dkeffer@utk.edu

This document provides the sample input for a Newton-Raphson code in the function funkeval and the sample input for a Runge-Kutta code in the function sysodeinput to iteratively solve for the inlet temperature in a cooling jacket running counter current to the flow in Plug Flow Reactor.

In essence, each iteration of the Newton-Raphson routine requires that we solve the system of ODEs three times. The first time provides the value of the inlet jacket temperature at the current guess of the outlet jacket temperature. The second two calls are used to calculate the numerical derivative of the function with respect to outlet jacket temperature.

We don't need any global statements in this example, because Tjout is never used in sysodeinput.m

The code would be invoked at the command line by starting the Newton-Raphson code,

```
[x0,err] = newraph_nd(450.0)
```

```
function f = funkeval(x)
Tjout = x;
Tjin_spec = 298.15; %K
CAin = 10000.0; % mol/m^3
CBin = 6000.0; % mol/m^3
CCin = 0.0; % mol/m^3
CDin = 5000.0; % mol/m^3
CEin = 0.0; % mol/m^3
CSin = 30000.0; % mol/m^3
Tin = 500.0; % K
m = 2;
n = 1000;
xo = 0;
xf = 100;
yo = [CAin, CBin, CCin, CDin, CEin, CSin, Tin, Tjout];
[y,x] = sysode(m,n,xo,xf,yo);
% Tjin_calc = y(n+1,8);  % Use this line if unnormalized
Tjin_calc = y(n+1,8)*Tjout;  % Use this line if normalized
f = Tjin_spec - Tjin_calc;
fprintf(' Tjout %e Tjin %e Tjin_spec %e \n', Tjout, Tjin_calc, Tjin_spec');
```

This is the sysodeinput.m code, which could be run independently in sysode.m or called by the Newton Raphson routine above. (Note that the jacket flowrate is negative, which makes this example counter-current flow.)

```
function dydt = sysodeinput(x,y,nvec);
%
% two simultaneous reactions in solvent S
% A + B --> C
% A + D --> E
%
% example usage:
% [y,x] = sysode(2,1000,0,10,[10000,6000,0,5000,0,30000,500,450]);
%
CA = y(1); % mol/m^3
CB = y(2); % mol/m^3
CC = y(3); % mol/m^3
CD = y(4); % mol/m^3
CE = y(5); % mol/m^3
CS = y(6); % mol/m^3
T = y(7); % K
Tj = y(8); % K
%
% stoichiometry
%
nuA1 = -1.0;
nuB1 = -1.0;
nuC1 = 1.0;
nuD1 = 0.0;
nuE1 = 0.0;
nuS1 = 0.0;
%
nuA2 = -1.0;
nuB2 = 0.0;
nuC2 = 0.0;
nuD2 = -1.0;
nuE2 = 1.0;
nuS2 = 0.0;
%
% rate law for reaction 1
%
R = 8.314; % J/mol/K
ko1 = 1.0e-2; % liter/mol/sec
ko1 = ko1/1000; % m^3/mol/sec
Ea1 = 2500; % J/mol
k1 = ko1*exp(-Ea1/(R*T));
r1 = k1*CA*CB;
%
% pure component heat capacities
%
CpA = 4.0; %J/mol/K
CpB = 5.0; %J/mol/K
CpC = 8.0; %J/mol/K
CpD = 3.0; %J/mol/K
CpE = 6.0; %J/mol/K
CpS = 3.0; %J/mol/K
%
% enthalpies of formation
```

```
%
Tref = 298.15; % K
pref = 101325; % Pa
HfrefA = -1000; %J/mol
HfrefB = -10000; % J/mol
HfrefC = -15000; % J/mol
HfrefD = -8000; % J/mol
HfrefE = -100000; % J/mol
HA = CpA*(T-Tref) + HfrefA;
HB = CpB*(T-Tref) + HfrefB;
HC = CpC*(T-Tref) + HfrefC;
HD = CpD*(T-Tref) + HfrefD;
HE = CpE*(T-Tref) + HfrefE;
DHR1 = nuA1*HA + nuB1*HB + nuC1*HC + nuD1*HD + nuE1*HE;
DHR2 = nuA2*HA + nuB2*HB + nuC2*HC + nuD2*HD + nuE2*HE;
%
% rate law for reaction 2
%
ko2 = 1.0e-1; % liter/mol/sec
ko2 = ko2/1000; % m^3/mol/sec
Ea2 = 3500; % J/mol
k2 = ko2*exp(-Ea2/(R*T));
r2 = k2*CA*CD;
%
% mole fractions
%
CT = CA + CB + CC + CD + CE + CS;
xA = CA/CT;
xB = CB/CT;
xC = CC/CT;
xD = CD/CT;
xE = CE/CT;
xS = CS/CT;
%
% mixture heat capacity
%
Cpmix = xA*CpA + xB*CpB + xC*CpC + xD*CpD + xE*CpE + xS*CpS;
%
% volumetric flowrate
%
F = 1; % liter/sec
F = F/1000; % cubic meters/sec
%
% circular pipe
%
Dp = 0.10; % m
Across = 0.25*pi*Dp*Dp; % m^2
l = 5; % m
V = Across*l; % m^3
%
% axial velocity
%
vz = F/Across; % m/s
%
% residence time
%
```

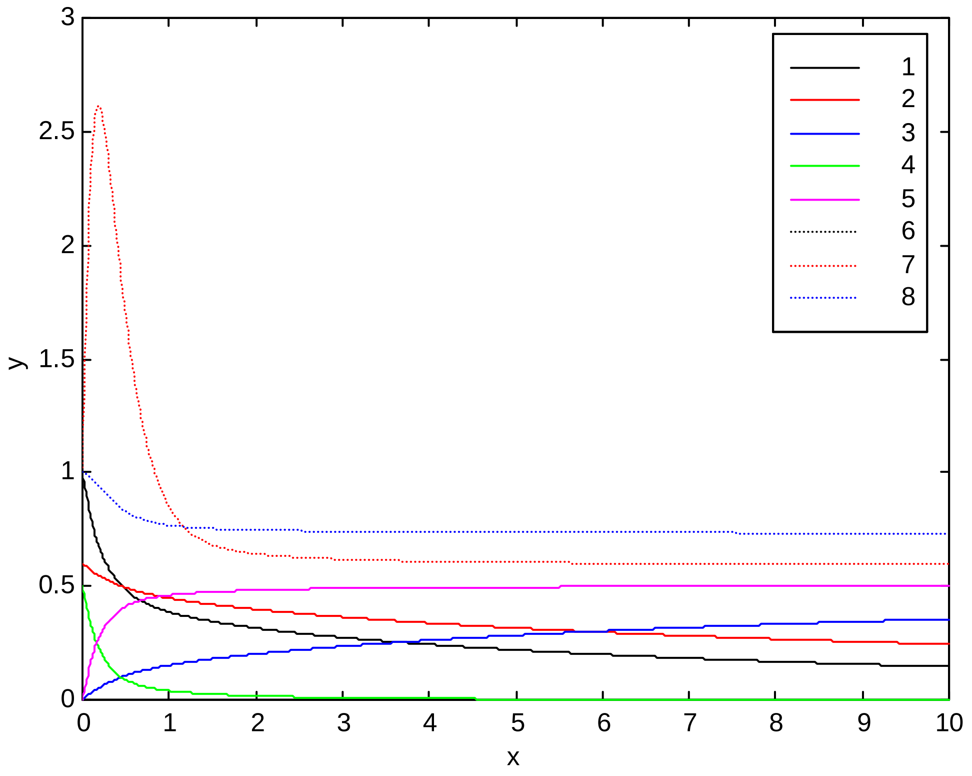
```
tr = l/vz; % sec
%
% heat loss information
%
As = 3.0; % m^2
U = 1500.0; % J/s/m^2/K
Qdot = As*U*(Tj-T);
Cpj = 4.184; % J/mol/K
Cj = 55.6; % mol/liter
Cj = Cj*1000; % mol/cubic meter
Fj = -20.0; % liters/sec
Fj = Fj/1000; % cubic meters/sec
Dpj = 0.30; % m
Acrossj = 0.25*pi*Dpj*Dpj - 0.25*pi*Dp*Dp; % m^2
Vj = Acrossj*l; % m^3
vzj = Fj/Acrossj; % m/s
%
% molar balances
%
% dCAdz = nuA*r/v
dydt(1) = (nuA1*r1 + nuA2*r2)/vz;
dydt(2) = (nuB1*r1 + nuB2*r2)/vz;
dydt(3) = (nuC1*r1 + nuC2*r2)/vz;
dydt(4) = (nuD1*r1 + nuD2*r2)/vz;
dydt(5) = (nuE1*r1 + nuE2*r2)/vz;
dydt(6) = (nuS1*r1 + nuS2*r2)/vz;
dydt(7) = (-DHR1*r1 - DHR2*r2 + Qdot/V)/(vz*CT*Cpmix);
dydt(8) = -Qdot/(vzj*Vj*Cj*Cpj);
```

The screen output looked like this:

```
» [x0,err] = newraph_nd(450.0)
 Tjout 4.500000e+002 Tjin 3.431576e+002 Tjin_spec 2.981500e+002
 Tjout 4.500100e+002 Tjin 3.431680e+002 Tjin_spec 2.981500e+002
 Tjout 4.499900e+002 Tjin 3.431473e+002 Tjin_spec 2.981500e+002
icount = 1 xold = 4.500000e+002 f = -4.500764e+001 df = -1.033993e+000 xnew = 4.064720e+002 err = 1.000000e+002
 Tjout 4.064720e+002 Tjin 2.981821e+002 Tjin_spec 2.981500e+002
 Tjout 4.064820e+002 Tjin 2.981924e+002 Tjin_spec 2.981500e+002
 Tjout 4.064620e+002 Tjin 2.981718e+002 Tjin_spec 2.981500e+002
icount = 2 xold = 4.064720e+002 f = -3.209244e-002 df = -1.032328e+000 xnew = 4.064409e+002 err = 7.648697e-005
 Tjout 4.064409e+002 Tjin 2.981500e+002 Tjin_spec 2.981500e+002
 Tjout 4.064509e+002 Tjin 2.981603e+002 Tjin_spec 2.981500e+002
 Tjout 4.064309e+002 Tjin 2.981397e+002 Tjin_spec 2.981500e+002
icount = 3 xold = 4.064409e+002 f = -2.583488e-008 df = -1.032327e+000 xnew = 4.064409e+002 err = 6.157324e-011

x0 =  406.4409
err =  6.1573e-011
```

The final plot looked like this:



3
2.5
2
1.5
1
0.5
0
y
0
1
2
3
4
5
6
7
8
9
10
x
1
2
3
4
5
6
7
8