Problem Set 11 – Due April 24 at Midnight.

Submit to the course website. Make sure your name is in the file header. Note any collaborations in the header as well. Each student must turn in their own set of solutions.

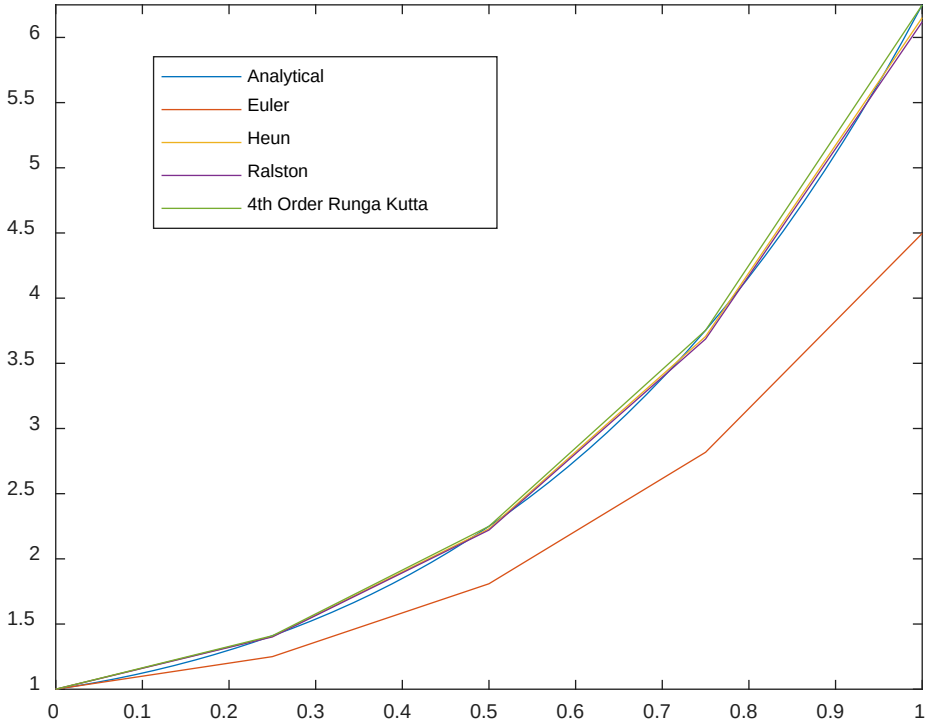
Problem 1 – Question 22.2

```
% PSET11_1
% Coded by Nigel F. Reuel on 11.18.2016
% This code solves problem 22.2
%
% Part A)
% Separation of variables yields (y)^(-1/2)dy = (1+4x)dx.  Integrating both
% sides will give you 2*y^(1/2) = x+2*x^2+C.  We know that y(0) = 1, so C
% must = 2.  Let's make the answer an anonymous function
Y1 = @(x) ((x+2*x^2+2)/2)^2;
fplot(Y1,[0 1])
hold on
%
% I am going to do parts B, C, and D all in the same loop
h = 0.25;
t = (0:h:1)'; n = length(t);
y0 = 1;
dydt = @(x,y) (1+4.*x).*(y).^(1/2);
% Preallocate memory
yb = y0*ones(n,1); %y for part b euler method
yc = y0*ones(n,1); %y for part c Heun method
yd = y0*ones(n,1); %y for part d ralston method
for i = 1:n-1 %implement Euler's method
    yb(i+1) = yb(i) + dydt(t(i),yb(i))*(t(i+1)-t(i)); % <-- Euler
    % ----
    ycnext = yc(i) + dydt(t(i),yc(i))*(t(i+1)-t(i)); % <-- Next step for Heun
    SlopeAvg = (dydt(t(i),yc(i))+dydt(t(i+1),ycnext))/2; % Avg slope for Heun
    yc(i+1) = yc(i) + SlopeAvg*(t(i+1)-t(i)); % Calculate next step for Heun
    %-----
    k1 = dydt(t(i),yd(i));
    k2 = dydt(t(i)+3/4*h,yd(i)+3/4*k1*h);
    yd(i+1) = yd(i)+ (1/3*k1+2/3*k2)*h;
end
plot(t,yb,t,yc,t,yd)
% Part (e) 4th order Runga Kutta Model
[tp,yp] = rk4sys(dydt,[0 1],y0,h);
plot(tp,yp)
legend('Analytical','Euler','Heun','Ralston','4th Order Runga Kutta')



function [tp,yp] = rk4sys(dydt,tspan,y0,h,varargin)
% rk4sys: fourth-order Runge-Kutta for a system of ODEs
% [t,y] = rk4sys(dydt,tspan,y0,h,p1,p2,...): integrates
% a system of ODEs with fourth-order RK method
% input:
% dydt = name of the M-file that evaluates the ODEs
% tspan = [ti, tf]; initial and final times with output
% generated at interval of h, or
% = [t0 t1 ... tf]; specific times where solution output
```

```
% y0 = initial values of dependent variables
% h = step size
% p1,p2,... = additional parameters used by dydt
% output:
% tp = vector of independent variable
% yp = vector of solution for dependent variables
if nargin<4,error('at least 4 input arguments required'), end
if any(diff(tspan)<=0),error('tspan not ascending order'), end
n = length(tspan);
ti = tspan(1);tf = tspan(n);
if n == 2
t = (ti:h:tf)'; n = length(t);
if t(n)<tf
t(n+1) = tf;
n = n+1;
end
else
t = tspan;
end
tt = ti; y(1,:) = y0;
np = 1; tp(np) = tt; yp(np,:) = y(1,:);
i=1;
while(1)
tend = t(np+1);
hh = t(np+1) - t(np);
if hh>h,hh = h;end
while(1)
if tt+hh>tend,hh = tend-tt;end
k1 = dydt(tt,y(i,:),varargin{:})';
ymid = y(i,:) + k1.*hh./2;
k2 = dydt(tt+hh/2,ymid,varargin{:})';
ymid = y(i,:) + k2*hh/2;
k3 = dydt(tt+hh/2,ymid,varargin{:})';
yend = y(i,:) + k3*hh;
k4 = dydt(tt+hh,yend,varargin{:})';
phi = (k1+2*(k2+k3)+k4)/6;
y(i+1,:) = y(i,:) + phi*hh;
tt = tt+hh;
i=i+1;
if tt>=tend,break,end
end
np = np+1; tp(np) = tt; yp(np,:) = y(i,:);
if tt>=tf,break,end
end
end
```


Analytical
Euler
Heun
Ralston
4th Order Runga Kutta
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

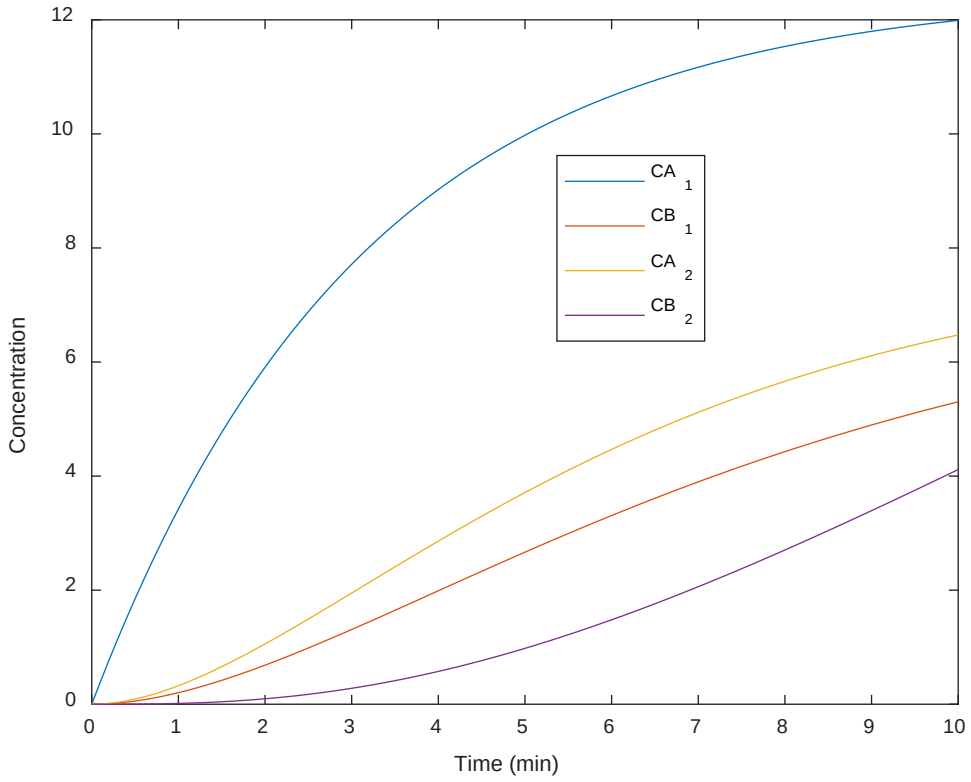
Problem 2 – Question 22.18. Plot all four dependent variables versus time. At what time does CB2 = 5?

```
% PSET11_2
% Coded by NFR on 11.18.2016
% This code solves problem 22.18 from the textbook
%
% Setup all of the rate equations as a sub function.  Then use the rk4sys
[tp,yp] = rk4sys(@sys,[0 10],[0 0 0 0],0.1);
plot(tp,yp)
xlabel('Time (min)')
ylabel('Concentration')
legend('CA_1','CB_1','CA_2','CB_2')


function dy = sys(t,y)
% y(1) = A1
% y(2) = B1
% y(3) = A2
% y(4) = B2
%
k = 0.12;
tao = 5;
Cao = 20;
dy = [1/tao*(Cao-y(1))-k*y(1);
    -1/tao*y(2)+k*y(1);
    1/tao*(y(1)-y(3))-k*y(3);
    1/tao*(y(2)-y(4))+k*y(4)];
end

function [tp,yp] = rk4sys(dydt,tspan,y0,h,varargin)
% rk4sys: fourth-order Runge-Kutta for a system of ODEs
% [t,y] = rk4sys(dydt,tspan,y0,h,p1,p2,...): integrates
% a system of ODEs with fourth-order RK method
% input:
% dydt = name of the M-file that evaluates the ODEs
% tspan = [ti, tf]; initial and final times with output
% generated at interval of h, or
% = [t0 t1 ... tf]; specific times where solution output
% y0 = initial values of dependent variables
% h = step size
% p1,p2,... = additional parameters used by dydt
% output:
% tp = vector of independent variable
% yp = vector of solution for dependent variables
if nargin<4,error('at least 4 input arguments required'), end
if any(diff(tspan)<=0),error('tspan not ascending order'), end
n = length(tspan);
ti = tspan(1);tf = tspan(n);
if n == 2
t = (ti:h:tf)'; n = length(t);
if t(n)<tf
t(n+1) = tf;
n = n+1;
```

```
end
else
t = tspan;
end
tt = ti; y(1,:) = y0;
np = 1; tp(np) = tt; yp(np,:) = y(1,:);
i=1;
while(1)
tend = t(np+1);
hh = t(np+1) - t(np);
if hh>h,hh = h;end
while(1)
if tt+hh>tend,hh = tend-tt;end
k1 = dydt(tt,y(i,:),varargin{:})';
ymid = y(i,:) + k1.*hh./2;
k2 = dydt(tt+hh/2,ymid,varargin{:})';
ymid = y(i,:) + k2*hh/2;
k3 = dydt(tt+hh/2,ymid,varargin{:})';
yend = y(i,:) + k3*hh;
k4 = dydt(tt+hh,yend,varargin{:})';
phi = (k1+2*(k2+k3)+k4)/6;
y(i+1,:) = y(i,:) + phi*hh;
tt = tt+hh;
i=i+1;
if tt>=tend,break,end
end
np = np+1; tp(np) = tt; yp(np,:) = y(i,:);
if tt>=tf,break,end
end
end
```



CB2 = 5 at roughly 11.2 minutes

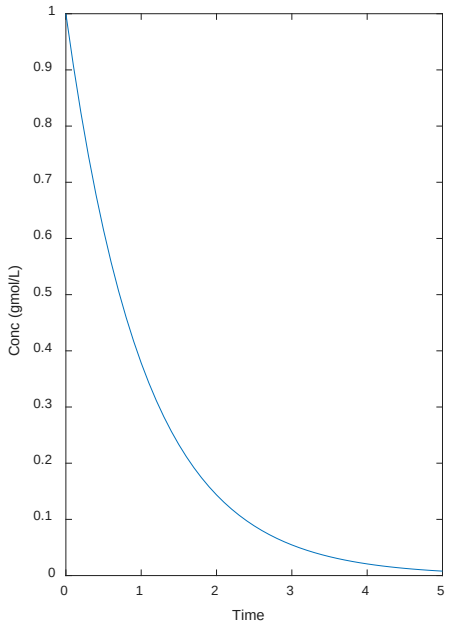
Problem 3 – Question 22.19. Let's assume the time unit is hours. Please plot temp and concentration from 0 to 5 hours. Use subplots. What is the max reactor temperature? When is 95% conversion achieved?

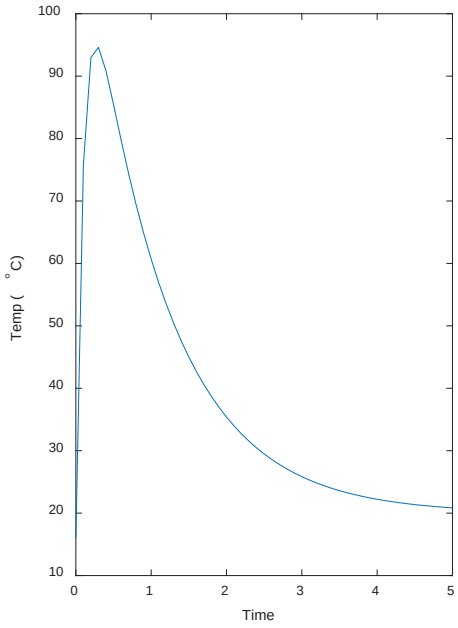
```
% PSET11_3
% Coded by Nigel F. Reuel on 11.18.2016
% This code solves problem 22.19 from the book
%
[tp,yp] = rk4sys(@sys,[0 5],[1 16],0.1);
subplot(1,2,1)
plot(tp,yp(:,1))
xlabel('Time')
ylabel('Conc (gmol/L)')
subplot(1,2,2)
plot(tp,yp(:,2))
xlabel('Time')
ylabel('Temp (\circC)')
% Add more here?  It would be interesting to figure out what TIME it hits
% 99% depletion of the reactant.


function dy = sys(t,y)
% y(1) = C
% y(2) = T
%
dy = [-exp(-10/(y(2)+273))*y(1);
    1000*exp(-10/(y(2)+273))*y(1)-10*(y(2)-20)];
end

function [tp,yp] = rk4sys(dydt,tspan,y0,h,varargin)
% rk4sys: fourth-order Runge-Kutta for a system of ODEs
% [t,y] = rk4sys(dydt,tspan,y0,h,p1,p2,...): integrates
% a system of ODEs with fourth-order RK method
% input:
% dydt = name of the M-file that evaluates the ODEs
% tspan = [ti, tf]; initial and final times with output
% generated at interval of h, or
% = [t0 t1 ... tf]; specific times where solution output
% y0 = initial values of dependent variables
% h = step size
% p1,p2,... = additional parameters used by dydt
% output:
% tp = vector of independent variable
% yp = vector of solution for dependent variables
if nargin<4,error('at least 4 input arguments required'), end
if any(diff(tspan)<=0),error('tspan not ascending order'), end
n = length(tspan);
ti = tspan(1);tf = tspan(n);
if n == 2
t = (ti:h:tf)'; n = length(t);
if t(n)<tf
t(n+1) = tf;
n = n+1;
end
```

```
else
t = tspan;
end
tt = ti; y(1,:) = y0;
np = 1; tp(np) = tt; yp(np,:) = y(1,:);
i=1;
while(1)
tend = t(np+1);
hh = t(np+1) - t(np);
if hh>h,hh = h;end
while(1)
if tt+hh>tend,hh = tend-tt;end
k1 = dydt(tt,y(i,:),varargin{:})';
ymid = y(i,:) + k1.*hh./2;
k2 = dydt(tt+hh/2,ymid,varargin{:})';
ymid = y(i,:) + k2*hh/2;
k3 = dydt(tt+hh/2,ymid,varargin{:})';
yend = y(i,:) + k3*hh;
k4 = dydt(tt+hh,yend,varargin{:})';
phi = (k1+2*(k2+k3)+k4)/6;
y(i+1,:) = y(i,:) + phi*hh;
tt = tt+hh;
i=i+1;
if tt>=tend,break,end
end
np = np+1; tp(np) = tt; yp(np,:) = y(i,:);
if tt>=tf,break,end
end
end
```





Max Temp ~ 94.62 °C

95% conversion ~ 3.1 minutes

Problem 4 – Cylindrical coordinates – Problem 20.17 from textbook.

```
% PSET11_Prob 4
% Coded by Nigel F. Reuel on 11.13.2017
% This code solves problem 20.17 from the textbook
%
ro = 3; %cm
vfunc = @(r) 2*(1-r./ro).^(1/6);
Qfunc = @(r) vfunc(r)*2*pi.*r;
I = trap(Qfunc,0,3,1000) % assuming units of cm^3/s
% NOTE to grader - they may use fewer trapezoid points to calculate the
% integral.  I use 1000 here to be close.
%
% Let's check work with integral function (not required)
Icheck = integral(Qfunc,0,3)
% Looks good!



% Chapra conveniently gives us the composite trapezoid function within the
% Romberg code.  You could also make this yourself using the trapz function
% in MATLAB.
function I = trap(func,a,b,n,varargin)
% trap: composite trapezoidal rule quadrature
% I = trap(func,a,b,n,pl,p2,...):
% composite trapezoidal rule
% input:
% func = name of function to be integrated
% a, b = integration limits
% n = number of segments (default = 100)
% pl,p2,... = additional parameters used by func
% output:
% I = integral estimate
if nargin<3,error('at least 3 input arguments required'),end
if ~(b>a),error('upper bound must be greater than lower'),end
if nargin<4|isempty(n),n=100;end
x = a; h = (b - a)/n;
s=func(a,varargin{:});
for i = 1 : n-1
x = x + h;
s = s + 2*func(x,varargin{:});
end
s = s + func(b,varargin{:});
I = (b - a) * s/(2*n);
end
```

I (Q) = 44.7285

Icheck (Q) = 44.7418 (checking the answer against the integral function)

Problem 5 - Integration from formulas

Part 1) Problem 20.6 from textbook. For part (c), use the integral function.

Part 2) Problem 20.8 from textbook – Plot the mass flow rate (mg/min) vs. time (min) curve for this problem. For part (b) use the 'integral' and 'quadgk' function and compare values.

```
%PSET10_Prob4
disp('#### Part 1 ####')
% Coded by Nigel F. Reuel on 4.23.2019
% Problem 20.6
% For this problem, solve the integral portion first, then do the root
Fsqr = @(t) RMSfunc(t).^2;
% Plot the function
fplot(Fsqr,[0 1])
disp('Part (a) Romberg')
IRMS_romberg = sqrt(romberg(Fsqr,0,1,0.1))
% Change variables to use for GL formulas
% We need to move our x points: x = (1+xd)/2 and dx = 1/2*dxd
Fsqr_GL = @(t) 1/2*RMSfunc((1+t)/2).^2;
% Now we can use the tabluated functions:
disp('Part (b) two point GL formula')
IRMS_2GL = sqrt(Fsqr_GL(-1/sqrt(3))+Fsqr_GL(1/sqrt(3)))
disp('Part (b) three point GL formula')
IRMS_3GL = sqrt(5/9*Fsqr_GL(-
sqrt(3/5))+8/9*Fsqr_GL(0)+5/9*Fsqr_GL(sqrt(3/5)))
% Use built in matlab function
disp('(c) Built in matlab function') %, <--- Dont't know what is wrong here
IRMS_matlab = sqrt(integral(Fsqr,0,1))

disp('#### Part 2 ####')
% Coded by Nigel F. Reuel on 11.11.2016
% This code solves problem 20.8 from the textbook
%
Q = @(t) 9 + 5*cos(0.4*t)^2;
c = @(t) 5*exp(-0.5*t)+2*exp(0.15*t);
QC = @(t) (9 + 5.*cos(0.4.*t).^2).*( 5.*exp(-0.5.*t)+2.*exp(0.15.*t));
%QC = @(t) Q(t)*c(t); From Yee
%
disp('Part (a) Mass (mg) calculated from romberg integration:')
[M,ea,iter]=romberg(QC,2,8,0.001,1000)
fplot(QC,[2 8])
ylabel('mg/min')
xlabel('min')
disp('Part (b) Mass (mg) calculated from integral function:')
M_integral = integral(QC,2,8)
disp('Part (b) Mass (mg) calculated from quadgk function:')
M_integral = quadgk(QC,2,8)

function Sol = RMSfunc(t)
for ii = 1:length(t)
    if t(ii) < 0
        Sol(ii) = 0;
    elseif t(ii) >= 0 && t(ii) <= 1/2
```

```
        Sol(ii) = 8*exp(-t(ii))*sin(2*pi*t(ii));
    elseif t(ii) > 1/2
        Sol(ii) = 0;
    end
end
end

function [q,ea,iter]=romberg(func,a,b,es,maxit,varargin)
% romberg: Romberg integration quadrature
% q = romberg(func,a,b,es,maxit,p1,p2,...):
% Romberg integration.
% input:
% func = name of function to be integrated
% a, b = integration limits
% es = desired relative error (default = 0.000001%)
% maxit = maximum allowable iterations (default = 30)
% pl,p2,... = additional parameters used by func
% output:
% q = integral estimate
% ea = approximate relative error (%)
% iter = number of iterations
if nargin<3,error('at least 3 input arguments required'),end
if nargin<4|isempty(es), es=0.000001;end
if nargin<5|isempty(maxit), maxit=50;end
n = 1;
I(1,1) = trap(func,a,b,n,varargin{:});
iter = 0;
while iter<maxit
iter = iter+1;
n = 2^iter;
I(iter+1,1) = trap(func,a,b,n,varargin{:});
for k = 2:iter+1
j = 2+iter-k;
I(j,k) = (4^(k-1)*I(j+1,k-1)-I(j,k-1))/(4^(k-1)-1);
end
ea = abs((I(1,iter+1)-I(2,iter))/I(1,iter+1))*100;
if ea<=es, break; end
end
q = I(1,iter+1);
end

function I = trap(func,a,b,n,varargin)
% trap: composite trapezoidal rule quadrature
% I = trap(func,a,b,n,pl,p2,...):
% composite trapezoidal rule
% input:
% func = name of function to be integrated
% a, b = integration limits
% n = number of segments (default = 100)
% pl,p2,... = additional parameters used by func
% output:
% I = integral estimate
if nargin<3,error('at least 3 input arguments required'),end
if ~(b>a),error('upper bound must be greater than lower'),end
if nargin<4|isempty(n),n=100;end
x = a; h = (b - a)/n;
```

```
s=func(a,varargin{:});
for i = 1 : n-1
x = x + h;
s = s + 2*func(x,varargin{:});
end
s = s + func(b,varargin{:});
I = (b - a) * s/(2*n);
end
```

Part 1

Part (a) Romberg

IRMS_romberg = 3.1411

Part (b) two point GL formula

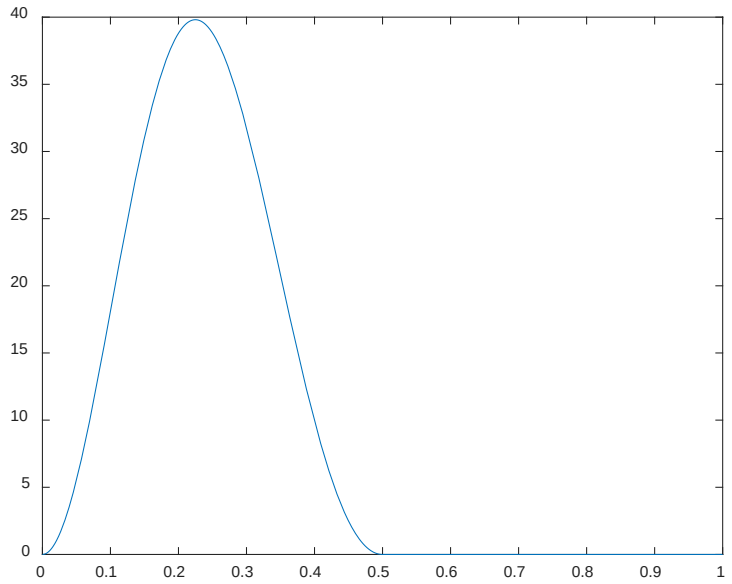
IRMS_2GL = 4.4447

Part (b) three point GL formula

IRMS_3GL = 2.4501

(c) Built in matlab function

IRMS_matlab = 3.1407



Part 2

Part (a) Mass (mg) calculated from romberg integration:

M = 335.9625

ea = 3.8843e-06

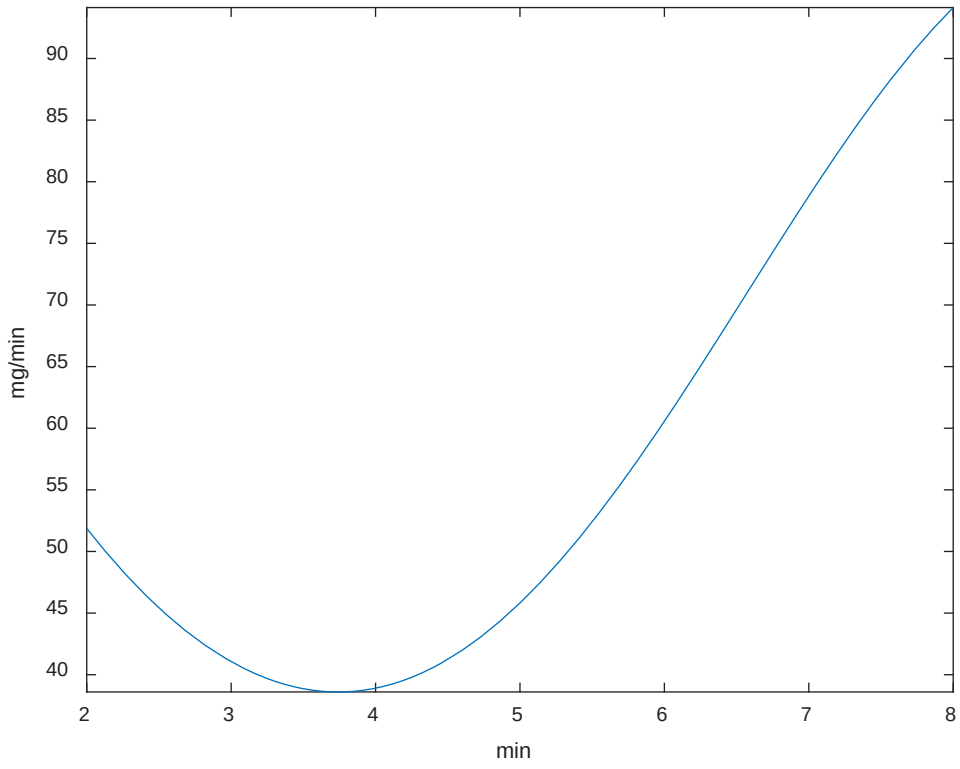
iter = 4

Part (b) Mass (mg) calculated from integral function:

M_integral = 335.9625

Part (b) Mass (mg) calculated from quadgk function:

M_integral = 335.9625



Group work credit – as we have done throughout the semester, show collaboration on slack by commenting and helping each other in your group.