Robert Heeter
BIOE 391 Numerical Methods – Due 25 March 2022

Problem Set 8 Solutions

1.

- **a. Output**
  PART A (analytical): I = 3.018316
  Analytical integral: $\boldsymbol{x + e^{-x}}$

- **b. Output**
  PART B (single trapezoidal rule (n = 1)): I = 1.963369 (true error = 34.951511%)

- **c. Output**
  PART C,i (composite trapezoidal rule with n = 2): I = 2.711014 (true error = 10.181236%)
  PART C,ii (composite trapezoidal rule with n = 4): I = 2.937840 (true error = 2.666230%)

- **d. Output**
  PART D (single Simpson’s 1/3 rule (n = 2)): I = 2.960229 (true error = 1.924478%)

- **e. Output**
  PART E (composite Simpson's 1/3 rule (n = 4)): I = 3.013449 (true error = 0.161229%)

- **f. Output**
  PART F (single Simpson's 3/8 rule (n = 3)): I = 2.991221 (true error = 0.897664%)

- **g. Output**
  PART G (composite Simpson's rule (n = 5)): I = 3.015814 (true error = 0.082874%)

2.

- **a. Output**
  PART A (analytical): I = 0.698806
  Analytical integral: $\boldsymbol{-e^{-x}}$

- **b. Output**
  PART B (composite trapezoidal rule): I = 0.701262 (true error = 0.351432%)

- **c. Output**
  PART C (combined trapezoidal and Simpson's rules): I = 0.698897 (true error = 0.013106%)

3. Output
Mass = 9518.500000 mg

4. Output
Work = 409.066643 kJ

5.

- **a. Output**
  PART A (Romberg integration): I = 504.693241 (approx. rel. error = 0.029054%, iterations = 3)

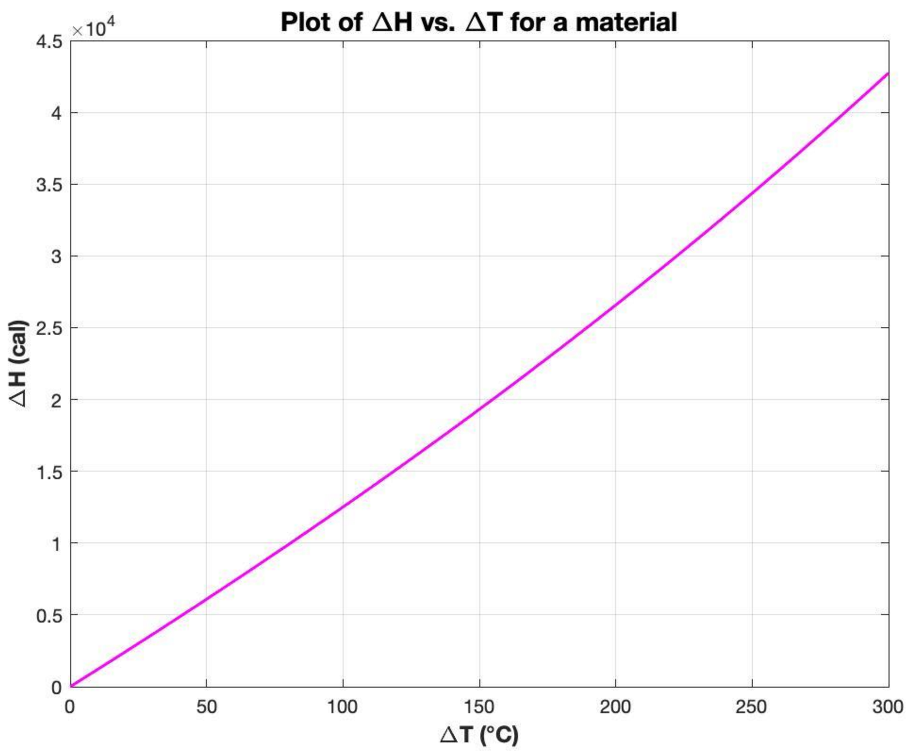
- **b. Output**

PART B (Two-point Gauss quadrature): I = 406.295022

c. **Output**
PART C (In-built integral function): I = 504.535992

6. **Output**
Final deltaH = 42732.000000 cal

Figure



7.

a. **Output**
PART A (Romberg integration): M = 335.959198 mg (approx. rel. error = 0.001443%, iterations = 3)

b. **Output**
PART B (In-built integral function): M = 335.962530 mg

8. **Output**
Q = 44.741803 cm^3/unit time

9. **Output**
Average value from 1 to 5 = 0.281768

Robert Heeter
BIOE 391 Numerical Methods – Due 25 March 2022

Complete MATLAB Code

```
% Robert Heeter
% BIOE 391 Numerical Methods
% HOMEWORK 8 MATLAB SCRIPT

clc, clf, clear, close all

%% P1. PROBLEM 19.2
disp('P1. PROBLEM 19.2');

% PART A
fx = @(x) 1-exp(-1.*x); % function
x1 = 0;
x2 = 4;
s_a = (x2+exp(-1*x2))-(x1+exp(-1*x1)); % analytical solution

% PART B
s_b = trap_integ(fx,0,4,1); % single trapezoidal rule (n = 1)
er_b = (abs(s_b-s_a)/s_a)*100; % true percent relative error

% PART C
s_c1 = trap_integ(fx,0,4,2); % composite trapezoidal rule with n = 2
er_c1 = (abs(s_c1-s_a)/s_a)*100; % true percent relative error

s_c2 = trap_integ(fx,0,4,4); % composite trapezoidal rule with n = 4
er_c2 = (abs(s_c2-s_a)/s_a)*100; % true percent relative error

% PART D
x = [0 2 4];
s_d = simpson13(fx,x); % single Simpson's 1/3 rule (n = 2)
er_d = (abs(s_d-s_a)/s_a)*100; % true percent relative error

% PART E
x1 = [0 1 2];
x2 = [2 3 4];
s_e = simpson13(fx,x1) + simpson13(fx,x2); % composite Simpson's 1/3 rule (n = 4)
er_e = (abs(s_e-s_a)/s_a)*100; % true percent relative error

% PART F
x = [0 4/3 8/3 4];
s_f = simpson38(fx,x); % single Simpson's 3/8 rule (n = 3)
er_f = (abs(s_f-s_a)/s_a)*100; % true percent relative error

% PART G
x1 = [0 4/5 8/5];
x2 = [8/5 12/5 16/5 4];
s_g = simpson13(fx,x1) + simpson38(fx,x2); % composite Simpson's rule (n = 5)
er_g = (abs(s_g-s_a)/s_a)*100; % true percent relative error

% Display results
fprintf('PART A (analytical): I = %f\n\n',s_a);
fprintf('PART B (single trapezoidal rule (n = 1)): I = %f (true error = %f%%)\n\n',s_b,er_b);
fprintf('PART C,i (composite trapezoidal rule with n = 2): I = %f (true error = %f%%)\n',s_c1,er_c1);
fprintf('PART C,ii (composite trapezoidal rule with n = 4): I = %f (true error =
%f%%)\n\n',s_c2,er_c2);
fprintf('PART D (single Simpsons 1/3 rule (n = 2)): I = %f (true error = %f%%)\n\n',s_d,er_d);
fprintf('PART E (composite Simpsons 1/3 rule (n = 4)): I = %f (true error = %f%%)\n\n',s_e,er_e);
fprintf('PART F (single Simpsons 3/8 rule (n = 3)): I = %f (true error = %f%%)\n\n',s_f,er_f);
fprintf('PART G (composite Simpsons rule (n = 5)): I = %f (true error = %f%%)\n\n',s_g,er_g);


%% P2. PROBLEM 19.5
disp('P2. PROBLEM 19.5');
```

Robert Heeter

BIOE 391 Numerical Methods – Due 25 March 2022

```
% PART A
fx = @(x) exp(-1.*x); % function
x1 = 0;
x2 = 1.2;
s_a = (-1*exp(-1*x2))-(-1*exp(-1*x1)); % analytical solution
x = [0 0.1 0.3 0.5 0.7 0.95 1.2];

% PART B
s_trap = 0;
for i = 1:6
    s_trap = s_trap + trap_integ(fx,x(i),x(i+1),1);
end
er_trap = (abs(s_trap-s_a)/s_a)*100; % true percent relative error

% PART C
s_comb = trap_integ(fx,x(1),x(2),1) + simpson38(fx,x(2:5)) + simpson13(fx,x(5:7));
er_comb = (abs(s_comb-s_a)/s_a)*100; % true percent relative error

% Display results
fprintf('PART A (analytical): I = %f\n\n',s_a);
fprintf('PART B (composite trapezoidal rule): I = %f (true error = %f%%)\n\n',s_trap,er_trap);
fprintf('PART C (combined trapezoidal and Simpsons rules): I = %f (true error =
%f%%)\n\n',s_comb,er_comb);


%% P3. PROBLEM 19.16
disp('P3. PROBLEM 19.16');

t = [0 10 20 30 35 40 45 50]; % time (min)
Q = [4 4.8 5.2 5.0 4.6 4.3 4.3 5.0]; % flow rate (m^3/min)
c = [10 35 55 52 40 37 32 34]; % concentration (mg/m^3)

M = trapz(t,(Q.*c)); % mass equation

fprintf('Mass = %f mg\n\n',M); % display results


%% P4. PROBLEM 19.28
disp('P4. PROBLEM 19.28');

Pi = 2550; % initial pressure (kPa)
Pf = 210; % final pressure (kPa)
Vf = 0.75; % final volume (m^3)
c = Pf.*Vf.^1.3; % constant
Vi = (c/Pi)^(1/1.3); % initial pressure (m^3)

V = Vi:0.001:Vf; % volume vector
P = c./(V.^1.3); % pressure vector

W = trapz(V,P); % work equation

fprintf('Work = %f kJ\n\n',W); % display results


%% P5. PROBLEM 20.3
disp('P5. PROBLEM 20.3');

fx = @(x) x.*exp(2.*x); % function

% PART A
[I_rom,ea,iter] = romberg(fx,0,3,0.5); % use romberg function (below) with es = 0.5%

% PART B
I_gauss = gaussquad2(fx,0,3); % use gaussquad2 function (below) for two-point Gauss quadrature

% PART C
```

Robert Heeter
BIOE 391 Numerical Methods – Due 25 March 2022

```
I_int = integral(fx,0,3); % use in-built integral function

% Display results
fprintf('PART A (Romberg integration): I = %f (approx. rel. error = %f%%, iterations =
%d)\n\n',I_rom,ea,iter);
fprintf('PART B (Two-point Gauss quadrature): I = %f\n\n',I_gauss);
fprintf('PART C (In-built integral function): I = %f\n\n',I_int);


%% P6. PROBLEM 20.7
disp('P6. PROBLEM 20.7');

m = 1000; % mass (g)
deltaT = 0:300; % temperature differential vector (Â°C)
Ti = -100; % initial temperature (Â°C)
Tf = Ti + deltaT; % final temperature vector (Â°C)

Cp = @(T) 0.132 + (1.56e-4.*T) + ((2.64e-7).*T.^2); % heat capacity equation

deltaH = zeros(size(Tf)); % preallocate
for i = 1:length(Tf)
    deltaH(i) = integral(Cp,Ti,Tf(i))*m; % heat equation
end

% Plot results
figure
plot(deltaT,deltaH,'-m','LineWidth',1.5)
xlabel('{\Delta}T (Â°C)','FontSize',12,'FontWeight','bold');
ylabel('{\Delta}H (cal)','FontSize',12,'FontWeight','bold');
title('Plot of {\Delta}H vs. {\Delta}T for a material','FontSize',14,'FontWeight','bold');
grid on

% Display results
fprintf('Final deltaH = %f cal\n\n',deltaH(end));


%% P7. PROBLEM 20.8
disp('P7. PROBLEM 20.8');

Qt = @(t) 9 + 5.*(cos(0.4.*t).^2); % flow rate equation (m^3/min)
ct = @(t) 5.*exp(-0.5.*t) + 2.*exp(0.15.*t); % concentration equation (mg/m^3)

% PART A
[M_rom,ea,iter] = romberg(@(t) Qt(t).*ct(t),2,8,0.1); % use romberg function (below) with es = 0.1%

% PART B
M_int = integral(@(t) Qt(t).*ct(t),2,8); % use in-built integral function

% Display results
fprintf('PART A (Romberg integration): M = %f mg (approx. rel. error = %f%%, iterations =
%d)\n\n',M_rom,ea,iter);
fprintf('PART B (In-built integral function): M = %f mg\n\n',M_int);


%% P8. PROBLEM 20.17
disp('P8. PROBLEM 20.17');

r0 = 3; % radius of pipe (cm)
vr = @(r) (2.*(1-(r./r0)).^(1/6)).*2.*pi.*r; % radial velocity distribution

Q = integral(vr,0,r0); % use in-built integral function

fprintf('Q = %f cm^3/unit time\n\n',Q); % display results


%% P9. PROBLEM 20.29
```

Robert Heeter
BIOE 391 Numerical Methods – Due 25 March 2022

```
disp('P9. PROBLEM 20.29');

fx = @(x) 2./(1+x.^2); % function
I = gaussquad2(fx,1,5); % use gaussquad2 function (below) for two-point Gauss quadrature
avg_val = I/(5-1); % average value of function over interval from 1 to 5

fprintf('Average value from 1 to 5 = %f\n\n',avg_val); % display results


%% Additional Functions

function I = trap_integ(func,a,b,n,varargin)
% ABOUT: Composite trapezoidal rule quadrature, from textbook .m file.
% INPUTS: func = function; a = lower bound; b = upper bound; n = number of
% segments
% OUTPUTS: 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


function I = simpson13(func,x)
% ABOUT: Simpson's 1/3 rule equation.

I = (x(3)-x(1))*(func(x(1))+(4*func(x(2)))+func(x(3)))/6;

end


function I = simpson38(func,x)
% ABOUT: Simpson's 3/8 rule equation.

I = (x(4)-x(1))*(func(x(1))+(3*func(x(2)))+(3*func(x(3)))+func(x(4)))/8;

end


function I = gaussquad2(func,a,b)
% ABOUT: Two-point Gaussian quadrature using the Gauss-Legendre formula.

xd1 = -1/sqrt(3); % weighting factors
xd2 = 1/sqrt(3);
I1 = func(((b+a)/2)+((b-a)*xd1/2))*((b-a)/2); % first term in formula
I2 = func(((b+a)/2)+((b-a)*xd2/2))*((b-a)/2); % second term in formula
I = I1 + I2;
```

```
end


function [q,ea,iter] = romberg(func,a,b,es,maxit,varargin)
% ABOUT: Romberg integration quadrature.
% INPUTS: func = function; a = lower bound; b = upper bound; es = desired
% relative error (%); maxit = maximum number of iterations
% OUTPUTS: 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_integ(func,a,b,n,varargin{:});
iter = 0;

while iter < maxit
  iter = iter+1;
  n = 2^iter;
  I(iter+1,1) = trap_integ(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
```