Q1. Using the fact that Simpson's rule is exact for $f(x) = x^n$ for $n = 0, 1, 2$ and 3, we can write the following:

$$\begin{aligned}
\int_{x_0}^{x_2} dx &= a_0 + a_1 + a_2 \\
\int_{x_0}^{x_2} x dx &= a_0 x_0 + a_1 x_1 + a_2 x_2 \\
\int_{x_0}^{x_2} x^2 dx &= a_0 {x_0}^2 + a_1 {x_1}^2 + a_2 {x_2}^2 \\
\int_{x_0}^{x_2} x^3 dx &= a_0 {x_0}^3 + a_1 {x_1}^3 + a_2 {x_2}^3
\end{aligned}$$

Evaluating the integrals on the left hand side, we have

$$\begin{aligned}
x_2 - x_0 &= a_0 + a_1 + a_2 \\
\frac{{x_2}^2 - {x_0}^2}{2} &= a_0 x_0 + a_1 x_1 + a_2 x_2 \\
\frac{{x_2}^3 - {x_0}^3}{3} &= a_0 {x_0}^2 + a_1 {x_1}^2 + a_2 {x_2}^2 \\
\frac{{x_2}^4 - {x_0}^4}{4} &= a_0 {x_0}^3 + a_1 {x_1}^3 + a_2 {x_2}^3
\end{aligned}$$

The above system of equations can be written in matrix form (the bottom 3 equations) as shown below.

$$\begin{bmatrix} x_0 & x_1 & x_2 \\ {x_0}^2 & {x_1}^2 & {x_2}^2 \\ {x_0}^3 & {x_1}^3 & {x_2}^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} ({x_2}^2 - {x_0}^2)/2 \\ ({x_2}^3 - {x_0}^3)/3 \\ ({x_2}^4 - {x_0}^4)/4 \end{bmatrix}$$

The solution to the system of equations is then given by

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} x_0 & x_1 & x_2 \\ {x_0}^2 & {x_1}^2 & {x_2}^2 \\ {x_0}^3 & {x_1}^3 & {x_2}^3 \end{bmatrix}^{-1} \begin{bmatrix} ({x_2}^2 - {x_0}^2)/2 \\ ({x_2}^3 - {x_0}^3)/3 \\ ({x_2}^4 - {x_0}^4)/4 \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -(x_0 - x_2)(3{x_0}^2 + 2x_0x_2 - 4x_1x_0 + {x_2}^2 - 2x_1x_2)/12x_0(x_0 - x_1) \\ -(x_0 + x_2)(x_0 - x_2)^3/12x_1(x_0 - x_1)(x_1 - x_2) \\ (x_0 - x_2)({x_0}^2 + 2x_0x_2 - 2x_1x_0 + 3{x_2}^2 - 4x_1x_2)/12x_2(x_1 - x_2) \end{bmatrix}$$

Assuming x_0, x_1 and x_2 are equispaced, we make the following substitution.

$$x_1 = x_0 + h \text{ and } x_2 = x_0 + 2h$$

Consequently, simplifying the above expressions for a_0, a_1 and a_2, we obtain

$$\boxed{a_0 = \frac{h}{3}, a_1 = \frac{4h}{3} \text{ and } a_2 = \frac{h}{3}}$$

VERIFICATION: Substituting the values of a_0, a_1 and a_2 into the first equation, we have

$$x_2 - x_0 = \frac{h}{3} + \frac{4h}{3} + \frac{h}{3}$$
$$x_2 - x_0 = 2h$$

Since $x_2 = x_0 + 2h$, this simplifies to $2h = 2h$. Thus, the obtained values of a_0, a_1, a_2 are consistent with all equations.

Then for $n = 4$, we have

$$\int_{x_0}^{x_2} x^4 dx = a_0 {x_0}^4 + a_1 {x_1}^4 + a_2 {x_2}^4 + 24k$$

since $f^{(4)}(x) = 24 \ \forall \ x \in \mathbb{R}$

Evaluating the integral on LHS, we have

$$k = \left(\left({x_2}^5 - {x_1}^5\right)/5\right) - a_0 {x_0}^4 - a_1 {x_1}^4 - a_2 {x_2}^4\big)/24$$

Substituting for $x_0, x_1, x_2, a_0, a_1, a_2$ and simplifying, we get

$$\boxed{k = -\frac{h^5}{90}}$$

Thus, the final form of Simpson's rule is obtained.

$$\boxed{\int_{x_0}^{x_2} f(x)\,dx = \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right] - \frac{h^5}{90} f^{(4)}(\xi)}$$

Note that $h = x_1 - x_0 = x_2 - x_1$ in the above formula.

Q2. Using the technique of Romberg Integration, the following results are obtained.

Integral	$R_{n,n}$	n	# of func. eval.
$\int_0^1 x^{1/3}dx$	0.749995	12	2049
$\int_0^1 x^2 e^{-x}dx$	0.160603	4	9

For the same number n, applying the Composite Trapezoidal Rule (using 2^{n-1} subintervals) results in a value that is the same as $R_{n,1}$. The values are tabulated below along with the true value of the integrals obtained analytically and the corresponding errors. The step size for the Composite Trapezoidal Rule is $h = 1/(2^{n-1})$ for both integrals.

Integral	$R_{n,n}$	$R_{n,1}$	True Value	$\epsilon_{n,n}$	$\epsilon_{n,1}$	Step Size (h)
$\int_0^1 x^{1/3}dx$	0.749995	0.749989	0.750000	0.000005	0.000011	0.000488
$\int_0^1 x^2 e^{-x}dx$	0.160603	0.161080	0.160603	0.000000	0.000477	0.125000

Note that $\epsilon_{n,n}$ is the absolute error in $R_{n,n}$ and similarly $\epsilon_{n,1}$ is the absolute error in $R_{n,1}$.

The scaling relationships between the errors and step size can be seen in the table below.

Integral	$\log_h \epsilon_{n,n}$	$\log_h \epsilon_{n,1}$	$\frac{\log_h \epsilon_{n,n}}{\log_h \epsilon_{n,1}}$
$\int_0^1 x^{1/3}dx$	1.6	1.5	1.1
$\int_0^1 x^2 e^{-x}dx$	7.8	3.7	2.1

Theoretical analysis tells us that the Composite Trapezoidal Method (CTM) gives has an error term of order $O(h^2)$ whereas Romberg Integration has an error term of order $O(h^{2n})$. Thus, Romberg integration in theory should have an error term that is n orders of h smaller than CTM. This is not seen to be true for any of the integrals. In fact, for the first integral, Romberg's Method performs only as good as CTM. One possible cause might be that the computation of this particular integral involves computations with small numbers, which we know are highly error prone. Also, we notice that for the same tolerance value of 10^{-5}, the first integral, while seemingly being simpler compared to the second integral takes thrice the number of iterations. This can be attributed to the small numbers involved too. In fact, when the same integral is computed with bounds 1 and 2, for the same tolerance level, Romberg's Method requires only 4 iterations! Additionally, the error is almost two orders smaller than the error obtained by CTM.

Integral	n	$R_{n,n}$	$R_{n,1}$	$\log_h \epsilon_{n,n}$	$\log_h \epsilon_{n,1}$	$\frac{\log_h \epsilon_{n,n}}{\log_h \epsilon_{n,1}}$
$\int_1^2 x^{1/3}dx$	4	1.139882	1.139724	7.8	4.2	1.9

Q3. (a) The following results are obtained when Gaussian Quadrature is used to compute the integral $\int_0^1 x^{1/3} dx$.

n	GQ_n	# of func. eval.
2	0.759778	2
3	0.753855	3
4	0.751946	4
5	0.751132	5

The values obtained by Romberg Integration for 2 and 4 intervals are tabulated below.

n	$R_{n,n}$	# of func. eval.
2	0.695800	3
4	0.742500	9

(b) The following results are obtained when Gaussian Quadrature is used to compute the integral $\int_0^1 x^2 e^{-x} dx$.

n	GQ_n	# of func. eval.
2	0.159410	2
3	0.160595	3
4	0.160603	4
5	0.160603	5

The values obtained by Romberg Integration for 2 and 4 intervals are tabulated below.

n	$R_{n,n}$	# of func. eval.
2	0.162402	3
4	0.160603	9

It can be clearly seen that for both the integrals, Gaussian Quadrature outperforms Romberg Integration. However, the difference is more pronounced for the first integral. Also, for the same n, Romberg Integrations requires more number of function evaluations. The number of function evaluations increases exponentially with n for Romberg Integration, but only linearly in the case of Gaussian Quadrature. Hence, Gaussian Quadrature is clearly the better method in both the cases.

Q4. The initial value problem to be solved is

$$\frac{dy}{dt} = f(y,t)$$
$$y(1) = -1$$

where

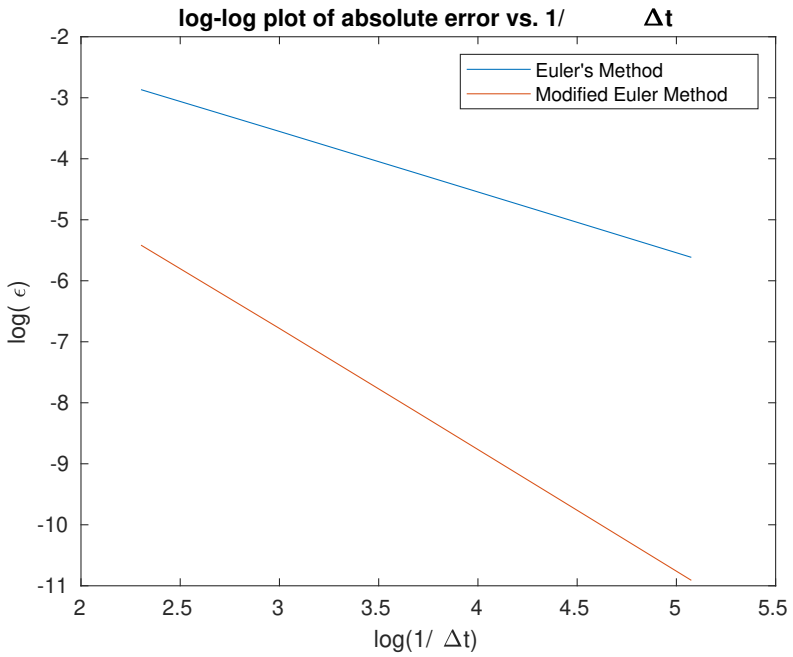
$$f(y,t) = \frac{1}{t^2} - \frac{y}{t} - y^2$$

and $1 \leq t \leq 2$.

The computed values of $y(2)$ are reported for both Euler's Method and Modified Euler's Method along with their respective absolute errors in the table below. The true value is $y(2) = -0.500000$.

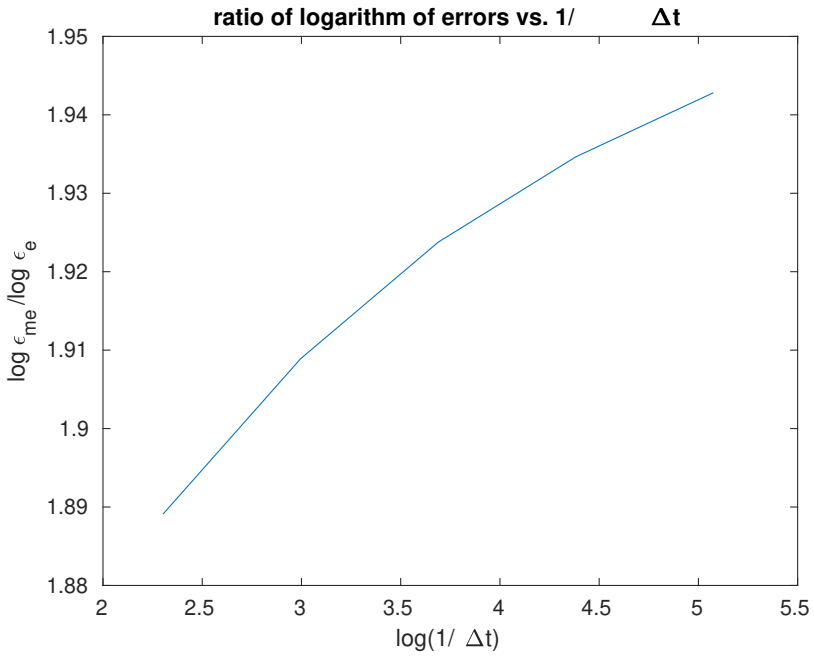
n	$y(2)$ (Euler)	$y(2)$ (Mod. Euler)	ϵ_{Euler}	$\epsilon_{\text{Mod. Euler}}$
0	-0.443139	-0.495557	0.056861	0.004443
1	-0.471220	-0.498856	0.028780	0.001144
2	-0.485516	-0.499710	0.014484	0.000290
3	-0.492733	-0.499927	0.007267	0.000073
4	-0.496360	-0.499982	0.003640	0.000018

The plot of the absolute error in $y(2)$ vs. $1/\Delta t$ is plotted on a log-log scale.



We can clearly see that the Modified Euler's method is more effective than Euler's Method; the difference between the two methods increases as the step size (i.e. Δt) decreases.

From the plot above, it can also be seen that the logarithm of the error of Modified Euler's Method is roughly twice of logarithm of error of Euler's Method which should be expected since Euler's method has an error term of order $O(h)$ whereas Modified Euler's Method has an error term of order $O(h^2)$. To see this more clearly, we can look at the plot below.



It can be seen in the plot above that the ratio of the log of errors approaches 2 as the step size decreases. This behaviour is expected from theoretical analysis too. The trend agrees with our knowledge of the order of accuracy of these methods.

MATLAB Code - Q1 (hw5p1.m)

```
%
% DS 288 Numerical Methods
% Homework 5, Problem 1
% Nikhil Jayswal, SR - 16961
%

clc; clear;
close all

syms x0 h
x1 = x0 + h;
x2 = x0 + 2*h;
% uncomment below and comment above 2 lines
% to see solution in terms of x0 x1 and x2
% syms x1 x2

A = [  x0   x1   x2;
     x0^2 x1^2 x2^2;
     x0^3 x1^3 x2^3];

b = [(x2^2 - x0^2)/2;
     (x2^3 - x0^3)/3;
     (x2^4 - x0^4)/4];

coeffs = A\b;
a0 = coeffs(1)
a1 = coeffs(2)
a2 = coeffs(3)

% verification a0 + a1 + a2 = 2h
% a0 + a1 + a2

% value of k
k = (x2^5 - x0^5)/5 - a0*x0^4 - a1*x1^4 - a2*x2^4;
k = k/24;
k = simplify(k)
```

MATLAB Code - Q2 (hw5p2.m)

```
%
% DS 288 Numerical Methods
% Homework 5, Problem 2
% Nikhil Jayswal, SR - 16961
%

clc; clear;
close all

% Part (a)
f = @(x) x^(1/3);
a = 0;
b = 1;
% to see how small numbers affect # of iterations req.
% a = 1;
% b = 2;
h = b - a;

tol = 1e-5;
R = [0];
nfeval = 0;

i = 1;
j = 1;
R(i, j) = (h/2)*(f(a) + f(b));
nfeval = nfeval + 2;

while (1 > 0)
    ansR = R(i, i);
    i = i + 1;
    % calculate R(i, 1)
    fsum = 0;
    for k = 1:2^(i - 2)
            fsum = fsum + f(a + (2*k-1)*(h/2^(i-1)));
    end
    R(i, 1) = 0.5*(R(i-1, 1) + (h/2^(i-2))*fsum);
    nfeval = nfeval + k;
    % calculate R(i, j) for j = 2,..,i
    for j = 2:i
```

```
        R(i, j) = R(i, j-1) + (R(i, j-1) - R(i-1, j-1))/(4^(j-1) - 1);
    end
    % check for termination
    if abs(R(i, j) - ansR) < tol
        ansR = R(i, j);
        break;
    end
end

fprintf('Approximate value of integral = %.6f\n', ansR)
fprintf('Iterations required = %d\n', i)
fprintf('Total number of function evaluations = %d\n', nfeval)
fprintf('Value obtained by Composite Trapezoidal Rule = %.6f\n\n', R(i,1))

% Part (b)
f = @(x) exp(-x)*x^2;
a = 0;
b = 1;
h = b - a;

tol = 1e-5;
R = [0];
nfeval = 0;

i = 1;
j = 1;
R(i, j) = (h/2)*(f(a) + f(b));
nfeval = nfeval + 2;

while (1 > 0)
    ansR = R(i, i);
    i = i + 1;
    % calculate R(i, 1)
    fsum = 0;
    for k = 1:2^(i - 2)
            fsum = fsum + f(a + (2*k-1)*(h/2^(i-1)));
    end
    R(i, 1) = 0.5*(R(i-1, 1) + (h/2^(i-2))*fsum);
    nfeval = nfeval + k;
    % calculate R(i, j) for j = 2,..,i
    for j = 2:i
```

```
        R(i, j) = R(i, j-1) + (R(i, j-1) - R(i-1, j-1))/(4^(j-1) - 1);
    end
    % check for termination
    if abs(R(i, j) - ansR) < tol
        ansR = R(i, j);
        break;
    end
end

fprintf('Approximate value of integral = %.6f\n', ansR)
fprintf('Iterations required = %d\n', i)
fprintf('Total number of function evaluations = %d\n', nfeval)
fprintf('Value obtained by Composite Trapezoidal Rule = %.6f\n\n', R(i,1))
```

MATLAB Code - Q3 (hw5p3.m)

```
%
% DS 288 Numerical Methods
% Homework 5, Problem 3
% Nikhil Jayswal, SR - 16961
%

clc; clear;
close all

% data required for Gaussian Quadrature
% roots and coefficients
% source - https://pomax.github.io/bezierinfo/legendre-gauss.html
roots = [-0.5773502691896257;  0.5773502691896257; ...
          0.0000000000000000; -0.7745966692414834; ...
          0.7745966692414834; -0.3399810435848563; ...
          0.3399810435848563; -0.8611363115940526; ...
          0.8611363115940526;  0.0000000000000000; ...
         -0.5384693101056831;  0.5384693101056831; ...
         -0.9061798459386640;  0.9061798459386640];

coeffs = [1.0000000000000000;  1.0000000000000000; ...
          0.8888888888888888;  0.5555555555555556; ...
          0.5555555555555556;  0.6521451548625461; ...
          0.6521451548625461;  0.3478548451374538; ...
          0.3478548451374538;  0.5688888888888889; ...
          0.4786286704993665;  0.4786286704993665; ...
          0.2369268850561891;  0.2369268850561891;];

% Part (a)
f = @(x) x^(1/3);
a = 0;
b = 1;
n = 2:5;
qdtr = zeros(size(n));
nfeval = zeros(size(n));

% use Gaussian Quadrature to compute the integral
for i = 1:length(n)
    qdtr(i) = 0;
```

```
    if i~=1
        index = sum(n(1:i-1));
    else
        index = 0;
    end
    for j = 1:n(i)
        z = 0.5*((b - a)*roots(index+j) + a + b);
        qdtr(i) = qdtr(i) + ((b - a)/2)*coeffs(index+j)*f(z);
        nfeval(i) = nfeval(i) + 1;
    end

end

% print results
fprintf('     Integrand = x^(1/3)\n\n')
fprintf(' n  \t\t  Quadrature  \t\t  nfeval \n')
fprintf('--- \t\t ------------ \t\t --------\n')
for i = 1:length(qdtr)
    fprintf(' %d \t\t   %.6f \t\t    %d\n', n(i), qdtr(i), nfeval(i))
end

% Part (b)
f = @(x) exp(-x)*x^2;
a = 0;
b = 1;
n = 2:5;
qdtr = zeros(size(n));
nfeval = zeros(size(n));

% use Gaussian Quadrature to compute the integral
for i = 1:length(n)
    qdtr(i) = 0;
    if i~=1
        index = sum(n(1:i-1));
    else
        index = 0;
    end
    for j = 1:n(i)
        z = 0.5*((b - a)*roots(index+j) + a + b);
        qdtr(i) = qdtr(i) + ((b - a)/2)*coeffs(index+j)*f(z);
        nfeval(i) = nfeval(i) + 1;
```

```
    end
end

% print results
fprintf('\n\n')
fprintf('     Integrand = exp(-x)*x^2\n\n')
fprintf(' n  \t\t  Quadrature  \t\t  nfeval \n')
fprintf('--- \t\t ------------ \t\t --------\n')
for i = 1:length(qdtr)
    fprintf(' %d \t\t   %.6f \t\t    %d\n', n(i), qdtr(i), nfeval(i))
end
fprintf('\n')
```

MATLAB Code - Q4 (hw5p4.m)

```
%
% DS 288 Numerical Methods
% Homework 5, Problem 4
% Nikhil Jayswal, SR - 16961
%

clc; clear;
close all

% function f(y, t)
f = @(t, y) (1/t^2) - (y/t) - (y^2);

% step sizes
n = 0:4;
delt = 0.1*2.^(-n);

% bounds
tlower = 1;
tupper = 2;

% initial condition
w0 = -1;

% output (y(2))
w2 = zeros(size(n));

% Euler's method
for i = 1:length(delt)
    step = delt(i);
    t = tlower:step:tupper;
    w = zeros(size(t));
    w(1) = w0;
    for j = 2:length(t)
        w(j) = w(j-1) + f(t(j-1), w(j-1))*step;
    end
    w2(i) = w(j);
end

% plot errors vs 1/delt
```

```
trueval = -1/2;
errorEuler = abs(w2-trueval);
plot(log(1./delt), log(errorEuler))
hold on

% Modified Euler's Method
% initial condition
w0 = -1;

% output (y(2))
w2 = zeros(size(n));

for i = 1:length(delt)
    step = delt(i);
    t = tlower:step:tupper;
    w = zeros(size(t));
    w(1) = w0;
    for j = 2:length(t)
        prev_slope = f(t(j-1), w(j-1));
        after_slope = f(t(j), w(j-1) + prev_slope*step);
        avg_slope = (prev_slope + after_slope)/2;
        w(j) = w(j-1) + step*avg_slope;
    end
    w2(i) = w(j);
end

% plot errors vs 1/delt
trueval = -1/2;
errorModEuler = abs(w2-trueval);
plot(log(1./delt), log(errorModEuler))
xlabel('log(1/\Deltat)')
ylabel('log(\epsilon)')
title('log-log plot of absolute error vs. 1/\Deltat')
legend('Euler''s Method', 'Modified Euler Method')
hold off

% plot log(error - Euler)/log(error - Modified Euler) vs 1/delt
figure
ratio = log(errorModEuler)./log(errorEuler);
plot(log(1./delt), ratio)
xlabel('log(1/\Deltat)')
```

```
ylabel('log\epsilon_{me}/log\epsilon_e')
title('ratio of logarithm of errors vs. 1/\Deltat')
```