1. Consider the matrix A and vector b given by

$$A = \begin{pmatrix} 1 & -1 & \alpha \\ -1 & 2 & -\alpha \\ \alpha & 1 & 1 \end{pmatrix} \qquad \text{and} \qquad b = \begin{pmatrix} -2 \\ 3 \\ 2 \end{pmatrix}.$$

a. Gaussian elimination begins with $R_2 + R_1$ and $R_3 - \alpha R_1$ to row reduce the first column, so $L_{21} = -1$ and $L_{31} = \alpha$. This gives the extended matrix system:

$$\left[\begin{array}{ccccc} 1 & -1 & \alpha & \vdots & -2 \\ -1 & 2 & -\alpha & \vdots & 3 \\ \alpha & 1 & 1 & \vdots & 2 \end{array}\right] \longrightarrow \left[\begin{array}{ccccc} 1 & -1 & \alpha & \vdots & -2 \\ 0 & 1 & 0 & \vdots & 1 \\ 0 & 1+\alpha & 1-\alpha^2 & \vdots & 2+2\alpha \end{array}\right]$$

The second column is row reduced by $R_3 - (1+\alpha)R_2$, so $L_{32} = \alpha + 1$, or

$$\longrightarrow \left[\begin{array}{ccccc} 1 & -1 & \alpha & \vdots & -2 \\ 0 & 1 & 0 & \vdots & 1 \\ 0 & 0 & 1-\alpha^2 & \vdots & 1+\alpha \end{array}\right]$$

This results in L and U having the following compact representation:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ \alpha & \alpha+1 & 1 \end{pmatrix} \qquad \text{and} \qquad U = \begin{pmatrix} 1 & -1 & \alpha \\ 0 & 1 & 0 \\ 0 & 0 & 1-\alpha^2 \end{pmatrix}.$$

b. From the LU factorization in Part a, we solve $Ax = b$ and readily obtain (by back substitution):

$$x = \begin{pmatrix} \frac{1}{\alpha-1} \\ 1 \\ \frac{1}{1-\alpha} \end{pmatrix},$$

provided $\alpha \neq \pm 1$. If $\alpha = -1$, then there are infinitely many solutions as x_3 is arbitrary, giving

$$x = \begin{pmatrix} -1+c \\ 1 \\ c \end{pmatrix},$$

while $\alpha = 1$ implies there are no solutions.

c. To find the condition number, $\kappa_1(A)$, we find the 1-norm of the matrices A and A^{-1}. Since

$$A = \begin{pmatrix} 1 & -1 & \alpha \\ -1 & 2 & -\alpha \\ \alpha & 1 & 1 \end{pmatrix} \quad \text{and} \quad A^{-1} = \begin{pmatrix} \frac{-(\alpha+2)}{(\alpha^2-1)} & \frac{-1}{(\alpha-1)} & \frac{\alpha}{(\alpha^2-1)} \\ 1 & 1 & 0 \\ \frac{(2\alpha+1)}{(\alpha^2-1)} & \frac{1}{(\alpha-1)} & \frac{-1}{(\alpha^2-1)} \end{pmatrix},$$

the 1-norm of these matrices comes from the max of the sum of the absolute value of the column elements or

$$\|A\|_1 = \max\{2+|\alpha|, 4, 1+2|\alpha|\}$$

and

$$\|A^{-1}\|_1 = \max\left(\frac{|\alpha|+1}{|\alpha^2-1|}, 1+2\left(|\alpha-1|\right)^{-1}, 1+\left|\frac{2+\alpha}{\alpha^2-1}\right| + \left|\frac{2\,\alpha+1}{\alpha^2-1}\right|\right).$$

The condition number depends on α from above with

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1.$$

d. With $\alpha = 1.01$, we can begin use our program `lutx` to find the $LU = PA$ factorization. The result gives the matrices:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -0.99 & 1 & 0 \\ 0.99 & -0.6656 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 1.01 & 1 & 1 \\ 0 & 2.99 & -0.0199 \\ 0 & 0 & 0.006656 \end{pmatrix}, \quad \text{and} \quad P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

From MatLab, we find the condition number for A is

$$\kappa_1(A) = 1204.$$

Alternately, we use the norms above to find that $\|A\|_1 = 4$ and $\|A^{-1}\|_1 = 301$, so again $\kappa_1(A) = 1204$, which suggests we could easily have problems in 3-digit arithmetic. The exact solution to $Ax = b$ is:

$$x = \begin{pmatrix} 100 \\ 1 \\ -100 \end{pmatrix}.$$

Next we perform 3-digit arithmetic using Gaussian elimination with partial pivoting. The first operation is $R_1 \leftrightarrow R_3$, which can be written:

$$[A:b] = \left[\begin{array}{ccccc} 1 & -1 & 1.01 & \vdots & -2 \\ -1 & 2 & -1.01 & \vdots & 3 \\ 1.01 & 1 & 1 & \vdots & 2 \end{array}\right] \leftrightarrow \left[\begin{array}{ccccc} 1.01 & 1 & 1 & \vdots & 2 \\ -1 & 2 & -1.01 & \vdots & 3 \\ 1 & -1 & 1.01 & \vdots & -2 \end{array}\right]$$

We perform the two pivots $R_2 + \frac{1}{1.01}R_1$ and $R_3 - \frac{1}{1.01}R_1$, which with 3-digit arithmetic gives

$$\left[\begin{array}{ccccc} 1.01 & 1 & 1 & \vdots & 2 \\ 0 & 2.99 & -0.02 & \vdots & 4.98 \\ 0 & -1.99 & 0.02 & \vdots & -3.98 \end{array}\right]$$

Next we perform the pivot $R_3 + \frac{1.99}{2.99}R_2$., which with 3-digit arithmetic gives:

$$\left[\begin{array}{ccccc} 1.01 & 1 & 1 & \vdots & 2 \\ 0 & 2.99 & -0.02 & \vdots & 4.98 \\ 0 & 0 & 0.0067 & \vdots & -0.66 \end{array}\right]$$

From this point, we perform back substitution with 3-digit arithmetic, yielding

$$x_3 = -98.5, \qquad x_2 = 1.01, \qquad \text{and} \qquad x_1 = 98.5.$$

The 1-norm of the absolute error is

$$|1.5 + 0.01 + 1.5| = 3.01.$$

2. a. The steady state diffusion in a quiescent fluid body with a first order chemical reaction may be modeled by the following boundary value problem:

$$D\frac{d^2w}{dx^2} - Kw = 0, \quad \text{with} \quad w(0) = 0, \quad w(1) = C, \quad \text{and} \quad x \in (0,1),$$

where w is the concentration of the substance, D is the diffusivity, K is the reaction rate, and C is the fixed boundary concentration at $x = 0$. For this problem, we assume that $D = 0.01$ cm^2/s, $K = 0.1$ s^{-1}, and $C = 5.0$ g/cm^3. It can be shown that the continuous solution to this boundary value problem is

$$w(x) = \frac{5\,\sinh(\sqrt{10}(1-x))}{\sinh(\sqrt{10})}.$$

With the diffusion operator discretized with a second order difference, boundary value problem can written as a matrix system, $AW = b$, where

$$A = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & -q & 1 & 0 & \cdots & 0 \\ 0 & 1 & -q & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -q & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix} \qquad \text{and} \qquad b = \begin{pmatrix} C \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

with $q = 2 + Kh^2/D$. Thus, if $n = 4$, then $h = 0.25$ and $q = 2.625$.

The matrix and vector become

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -2.625 & 1 & 0 & 0 \\ 0 & 1 & -2.625 & 1 & 0 \\ 0 & 0 & 1 & -2.625 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and $b = [5, 0, 0, 0, 0]^T$ This matrix A can be inserted into the MatLab code `lutx`. The resulting LU-factorization is

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & -0.3810 & 1 & 0 & 0 \\ 0 & 0 & -0.4456 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -2.6250 & 1 & 0 & 0 \\ 0 & 0 & -2.2440 & 1 & 0 \\ 0 & 0 & 0 & -2.1794 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The solution of $AW = b$, using MatLab's `bA` is

$$W = \begin{pmatrix} 5.0 \\ 2.29423399 \\ 1.02236422 \\ 0.38947208 \\ 0.0 \end{pmatrix}.$$

The actual solutions, $w(0.25) = 2.25221659$, $w(0.5) = 0.98692744$, and $w(0.75) = 0.37127130$. It follows that the absolute errors are 0.04201740, 0.03543678, and 0.01820078 at $x = 0.25$, 0.5, and 0.75, respectively.

b. With $n = 8$, the diagonal of the matrix A has $a_{ii} = -2.15625$ in this 9×9 matrix. The solution is below on the left. With $n = 16$, the diagonal of the matrix A has $a_{ii} = -2.0390625$ in this 17×17 matrix. The solution is below on the right.

$$W_8 = \begin{pmatrix} 5.0 \\ 3.36843745 \\ 2.26319325 \\ 1.51157299 \\ 0.99613601 \\ 0.63634529 \\ 0.37598352 \\ 0.17436917 \\ 0 \end{pmatrix}. \qquad W_{16} = \begin{pmatrix} 5.0 \\ 4.10102924 \\ 3.36225494 \\ 2.75481872 \\ 2.25499260 \\ 1.84325214 \\ 1.50351371 \\ 1.22250628 \\ 0.98925301 \\ 0.79464244 \\ 0.63107258 \\ 0.49215399 \\ 0.37246018 \\ 0.26731558 \\ 0.17261300 \\ 0.08465312 \\ 0 \end{pmatrix}.$$

The table below gives the values at $x = 0.25, 0.5, 0.75$ for the exact solution, the approximations, and the absolute errors.

x	$w(x)$	W_4	Err_4	W_8	Err_8	W_{16}	Err_{16}
0.25	2.252217	2.294234	0.042017	2.263193	0.010977	2.254993	0.002776
0.5	0.986927	1.022364	0.035437	0.996136	0.009209	0.989253	0.002326
0.75	0.371271	0.389472	0.018201	0.375984	0.004712	0.372460	0.001189

We see that halving the stepsize quarters the error, so this numerical approximation is $O(h^2)$. Using the values at $x = 0.5$, we see that $Err_4/(0.25)^2 = 0.56699$, $Err_8/(0.125)^2 = 0.58935$, and $Err_{16}/(0.0625)^2 = 0.0.59535$. These values are roughly constant, confirming the method is $O(h^2)$. Below is the graph of the actual solution with the three approximations above. We observe that all approximations are fairly accurate.

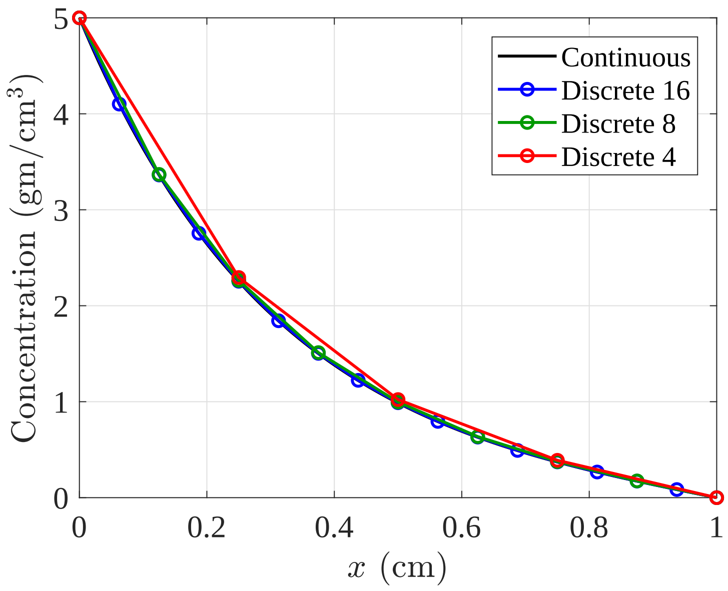


Figure 1: Plot of exact solution with the approximations for $n = 4$, $n = 8$, and $n = 16$.

c. Jacobi's iterative method for this problem takes the matrix A above and decomposes into a diagonal matrix D, a strictly lower triangular matrix L, and a strictly upper triangular matrix U, so that $A = D - L - U$. The matrix $T_j = D^{-1}(L + U)$ and vector $c_j = D^{-1}b$, then a convergent iterative scheme was produced by

$$w^{(k+1)} = T_j w^{(k)} + c_j, \qquad k = 0, 1, ...,$$

where x^0 is an initial vector. Let $x^0 = [0, ..., 0]^T$ (initially clear solution). The program below perform k steps of this iteration.

```
function [ w ] = prob2Cc_jacobi(w0,k,n)
% Compute Jacobi Iterate - perform k iterations

h = 1/n;
D = 0.01;
K = 0.1;
C = 5.0;
q = 2+K*h^2/D;

Dc = -q*eye(n+1);Dc(1,1)  = 1;Dc(n+1,n+1)=1;
Lc = diag(ones(n,1),-1);Lc(n+1,n)=0;
Uc = diag(ones(n,1),1);Uc(1,2)=0;
Tc = inv(Dc)*(-Lc-Uc);
cc = inv(Dc)*[C;zeros(n,1)];

w = w0;
for i = 1:k
```

```
        w = Tc*w + cc;
    end

end
```

Below are the iterates with $k = 4, 10, 20$, and 50.

$$w^{(4)} = \begin{pmatrix} 5.0 \\ 2.817579 \\ 1.075404 \\ 0.498738 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad w^{(10)} = \begin{pmatrix} 5.0 \\ 3.216946 \\ 1.936540 \\ 1.167132 \\ 0.580088 \\ 0.321872 \\ 0.113947 \\ 0.052845 \\ 0 \end{pmatrix}$$

$$w^{(20)} = \begin{pmatrix} 5.0 \\ 3.339076 \\ 2.199882 \\ 1.441000 \\ 0.907273 \\ 0.566213 \\ 0.313624 \\ 0.145449 \\ 0 \end{pmatrix} \qquad w^{(50)} = \begin{pmatrix} 5.0 \\ 3.368154 \\ 2.262581 \\ 1.510888 \\ 0.995271 \\ 0.635660 \\ 0.375372 \\ 0.174085 \\ 0 \end{pmatrix}$$

Below is a graph showing the first 10 iterates of $w^{(k)}$ along with the exact solution.

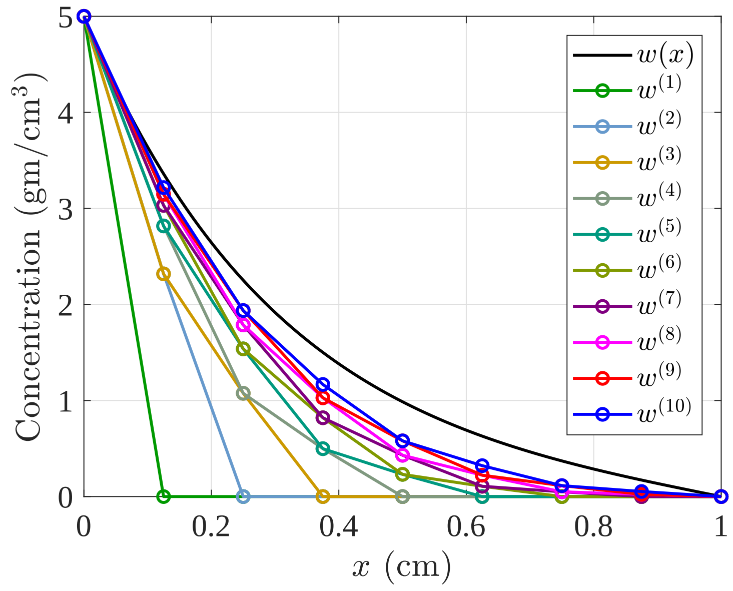


Figure 2: Plot of exact solution with the approximations for first 10 iterates.

3. a. The exact integral is found by the substitution $u = x^4$ (so $du = 4x^3dx$)), which gives

$$\begin{aligned} 0.4\int_0^3 x^3\cos(x^4)\,dx &= 0.1\int_0^{81}\cos(u)\,du \\ &= 0.1\sin(u)\Big|_0^{81} = 0.1\sin(81) = -0.06298879943. \end{aligned}$$

Below is a graph of the function, showing the increasingly rapid oscillations and the growing amplitude:

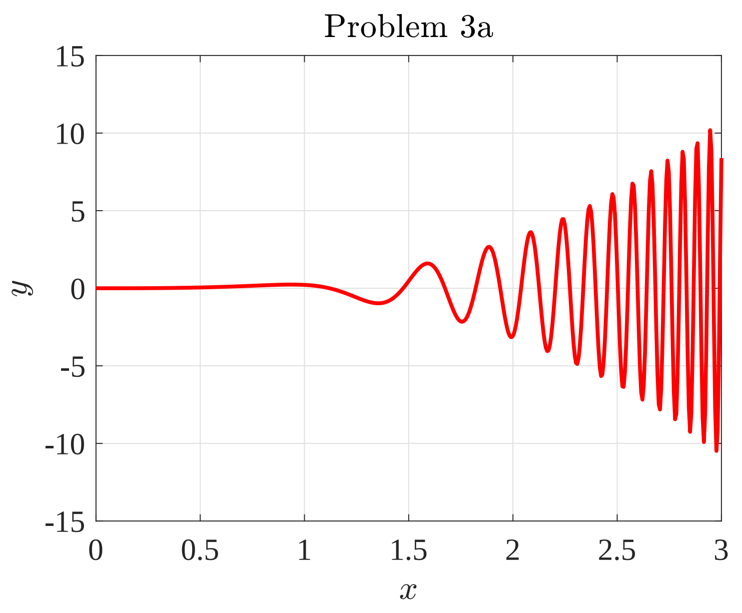


Figure 3: Graph of $f(x) = 0.4\,x^3\cos(x^4)$.

b. The MatLab program below can be used to compute the Composite Midpoint rule with $n = 12$ or $h = 0.25$.

```
function M = compmidptth2_4A(a,b,N)
% Composite Midpoint Rule for function f(x)
% on [a,b] using N steps
f = @(x) 0.2*x.^3.*sin(x.^4);
h = (b-a)/N;
i = 1:N;
ci = a+0.5*(2*i-1)*h;
M = h*sum(f(ci));
end
```

The Composite Midpoint Rule with $n = 12$ or $h = 0.25$ gives:

$$0.4\int_0^3 x^3\cos(x^4)\,dx \approx 2.0531341407 \quad \text{with} \quad abs\ err = 2.1161229401,$$

which gives a very poor approximation.

The MatLab program below can be used to compute the Composite Simpson's rule with $n = 12$ or $h = 0.25$.

```
function S = compsimpth2_4A(a,b,N)
% Composite Simpson's Rule for function f(x)
% on [a,b] using 2N steps
f = @(x) 0.2*x.^3.*sin(x.^4);
h = (b-a)/(2*N);
i = 0:N-1;
xi = a+2*i*h;
xi1 = a+2*(i+0.5)*h;
xi2 = a+2*(i+1)*h;
S = (h/3)*sum(f(xi)+4*f(xi1)+f(xi2));
end
```

The Composite Simpson's Rule with $n = 12$ or $h = 0.25$ gives:

$$0.4\int_0^3 x^3\cos(x^4)\,dx \approx 3.2220584345 \quad \text{with} \quad \mathit{abs\ err} = 3.2850472339,$$

which still gives a very poor approximation.

c. This part uses the Composite Midpoint and Composite Simpson's Rules starting with $n = 12$ or $h = 0.25$ to approximate the integral above and halving the stepsize until the accuracy is less than a tolerance of 10^{-6}. As this function oscillates quite rapidly, it requires a large number of steps to accurately approximate the solution.
The Composite Midpoint rule is analyzed with the MatLab program below:

```
function [H,Eh,S,N] = compmidtolth2_3C(a,b,tol)
% Composite Midpoint Rule for function f(x)
% on [a,b] doubling steps til within tolerance
f = @(x) 0.4*x.^3.*cos(x.^4);
ex = -0.062988799427445;
S0 = 0;
N = 12;
S = 1;
H = [];
Eh = [];
while (abs(S-S0)>tol)
    S0 = S;
    h = (b-a)/N;
    H = [H,h];
    i = 1:N;
    ci = a+0.5*(2*i-1)*h;
    S = h*sum(f(ci));
    eh = abs(S-ex);
    Eh = [Eh,eh];
    N = 2*N;
end
N = N/2;
p = polyfit(log(H),log(Eh),1)
nn = length(H);
```

```
lnH = log(H);
lnE = log(Eh);
plot([lnH(1),lnH(nn)],[(p(1)*lnH(1)+p(2)),(p(1)*lnH(nn)+p(2))],'b-');
hold on
plot(lnH,lnE,'bo');grid

fontlabs = 'Times New Roman';
xlabel('$\ln(h)$','FontSize',16,'FontName',fontlabs, ...
    'interpreter','latex');
ylabel('$\ln(err)$','FontSize',16,'FontName',fontlabs, ...
    'interpreter','latex');
mytitle = 'Problem 3c: Midpoint';
title(mytitle,'FontSize',16,'FontName', ...
    'Times New Roman','interpreter','latex');
set(gca,'FontSize',16);

print -depsc th2_3Cm_gr.eps
end
```

This program requires $n = 49152$ steps with a stepsize of $h = 0.00006103515625$ to obtain the accuracy to the desired tolerance (absolute error of 1.1534×10^{-7}. The graph below shows the log-log plot of the stepsize and the absolute error. The MatLab program `polyfit` gives the best fit slope to be $p = 2.1245$, which is consistent with the expected $p = 2$.

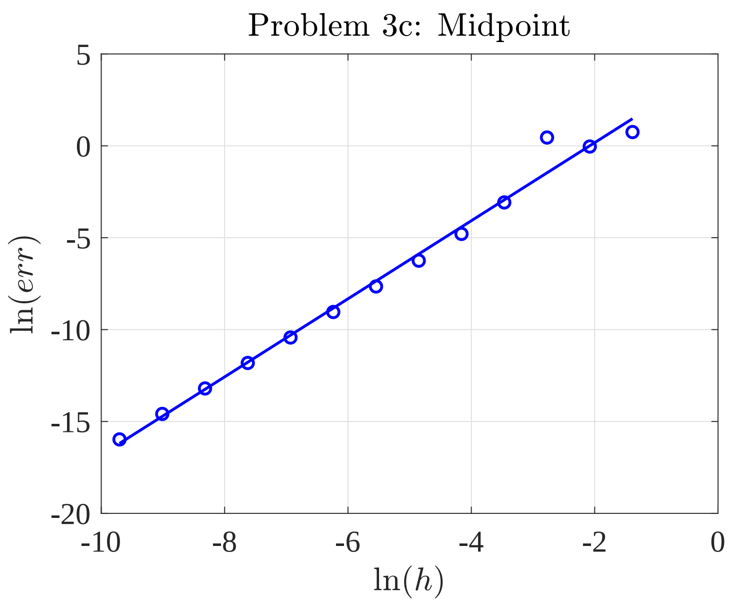


Figure 4: Graph of log of stepsize vs log of absolute error for Midpoint rule.

The Composite Simpson's rule is analyzed with the MatLab program below:

```
function [H,Eh,S,N] = compsimptolth2_3C(a,b,tol)
% Composite Simpson's Rule for function f(x)
% on [a,b] doubling steps til within tolerance
f = @(x) 0.4*x.^3.*cos(x.^4);
ex = -0.062988799427445;
S0 = 0;
```

```
N = 6;
S = 1;
H = [];
Eh = [];
while (abs(S-S0)>tol)
    S0 = S;
    h = (b-a)/(2*N);
    H = [H,h];
    i = 0:N-1;
    xi = a+2*i*h;
    xi1 = a+2*(i+0.5)*h;
    xi2 = a+2*(i+1)*h;
    S = (h/3)*sum(f(xi)+4*f(xi1)+f(xi2));
    eh = abs(S-ex);
    Eh = [Eh,eh];
    N = 2*N;
end
p = polyfit(log(H),log(Eh),1)
nn = length(H);
lnH = log(H);
lnE = log(Eh);
plot([lnH(1),lnH(nn)],[(p(1)*lnH(1)+p(2)),(p(1)*lnH(nn)+p(2))],'b-');
hold on
plot(lnH,lnE,'bo');grid

fontlabs = 'Times New Roman';
xlabel('$\ln(h)$','FontSize',16,'FontName',fontlabs, ...
    'interpreter','latex');
ylabel('$\ln(err)$','FontSize',16,'FontName',fontlabs, ...
    'interpreter','latex');
mytitle = 'Problem 3c: Simpson';
title(mytitle,'FontSize',16,'FontName', ...
    'Times New Roman','interpreter','latex');
set(gca,'FontSize',16);

print -depsc th2_3Cs_gr.eps
end
```

This program requires $n = 3072$ steps with a stepsize of $h = 0.0009765625$ to obtain the accuracy to the desired tolerance. The graph below shows the log-log plot of the stepsize and the absolute error. The MatLab program `polyfit` gives the best fit slope to be $p = 3.5676$, which is below the expected $p = 4$ because of the initial points on the graph.

d. The Adaptive Composite Simpson's Rule program used the same core program `ACSR` provided in class, while slight modifications were made to the `AdaptiveCSRsin` code at the beginning as seen below:

```
%
% Adaptive CSR
%
% This is the setup/driver part...
% Play with the tolerance for different results.
%

tol = 10^(-6);
a   = 0;
b   = 3;
```

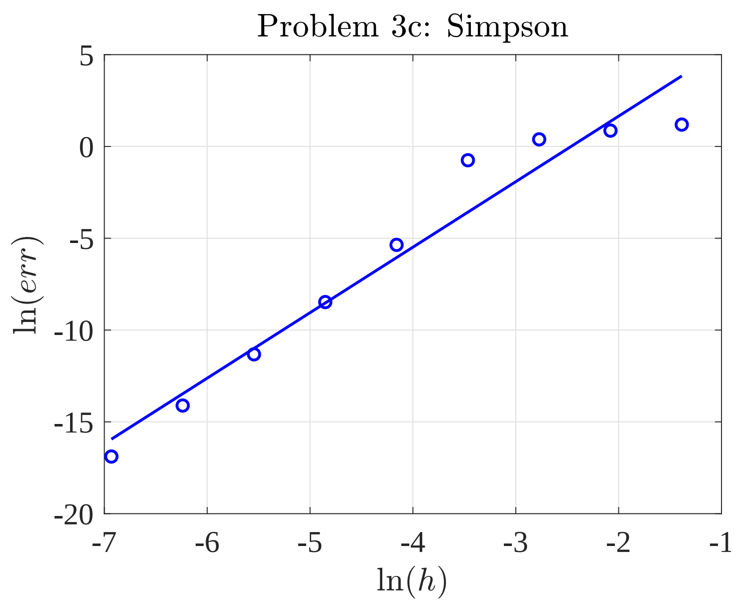


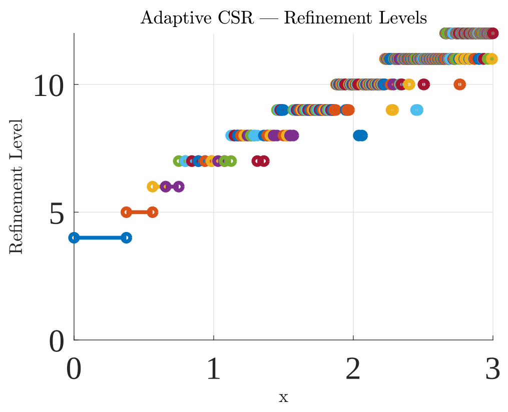
Figure 5: Graph of log of stepsize vs log of absolute error for Simpson's rule.

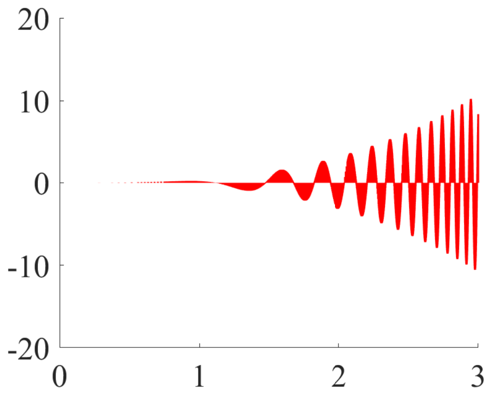
```

f = @(x) 0.4*x.^3.*cos(x.^4);

figure(1); xv = 0:0.001:3;
```

The output generates all the intervals of integration, which can be easily copied into a spreadsheet, where the number of rows can be counted. (Better coding would allow inserting an index counter, but I deemed it not worth the effort, given an existing program that suffices.) The number of steps was reduced substantially to only $n = 448$, since the early portion of the graph could use fairly large stepsizes. The reduction is to about 14.6% of the number of steps use in the Composite Simpson's rule. Below are graphic output from the Adaptive scheme used.





4 a. The exact integral is found by integration by parts twice, which gives

$$\begin{aligned}
\int_0^4 2\,x^2 e^{-0.6x}\,dx &= -\frac{10}{3}x^2 e^{-0.6x}\bigg|_0^4 + \frac{20}{3}\int_0^4 x\,e^{-0.6x}\,dx \\
&= -\frac{160}{3}e^{-12/5} - \frac{100}{9}xe^{-0.6x}\bigg|_0^4 + \frac{100}{9}\int_0^4 e^{-0.6x}\,dx \\
&= -\frac{160}{3}e^{-12/5} - \frac{400}{9}e^{-12/5} - \frac{500}{27}e^{-0.6x}\bigg|_0^4 \\
&= -\frac{3140}{27}e^{-12/5} + \frac{500}{27} \;=\; 7.9683565434.
\end{aligned}$$

Below is a graph of this function.

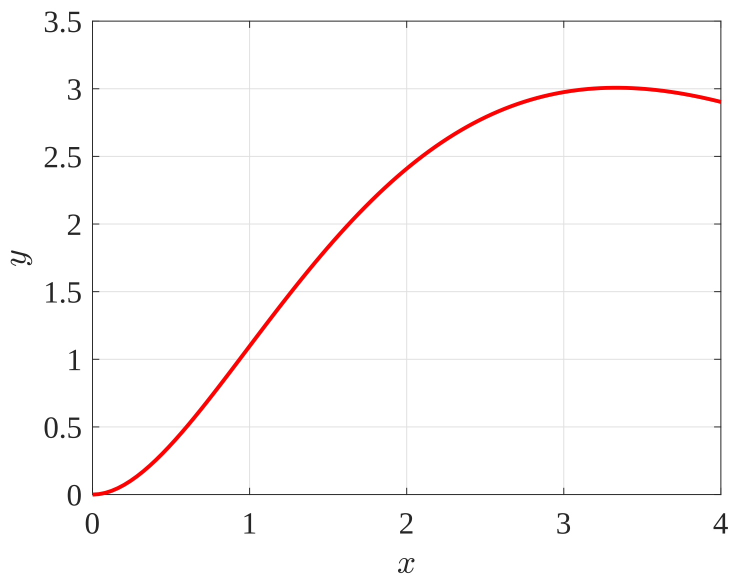


Figure 6: Graph of $f(x) = 2\,x^2 e^{-0.6x}$.

b. The MatLab program below can be used to compute the integral with Gaussian quadrature with 2, 3, 4, and 5 points. Only lines 2 and 3 need to be changed with the appropriate value.

```
1  [L,Lroots,GQw] = LegendrePolynomials(8);
2  gr = Lroots{5};
3  wt = GQw{5};
4  a = 0; b = 4;
5  f = @(x) 2*x.^2.*exp(-0.6*x);
6  gx = ((b-a)*gr + (b+a))/2;
7  wx = wt*(b-a)/2;
8  S = sum(wx.*f(gx))
```

For Gaussian quadrature with 2 points the program gives:

$$\int_0^4 2\,x^2 e^{-0.6x}\,dx \approx 7.7181496432 \quad \text{with} \quad abs\ err = 0.2502069002.$$

For Gaussian quadrature with 3 points the program gives:

$$\int_0^4 2\,x^2 e^{-0.6x}\,dx \approx 7.9564402246 \quad \text{with} \quad abs\ err = 0.0119163188.$$

For Gaussian quadrature with 4 points the program gives:

$$\int_0^4 2\,x^2 e^{-0.6x}\,dx \approx 7.9681879155 \quad \text{with} \quad abs\ err = 1.686279 \times 10^{-4}.$$

For Gaussian quadrature with 5 points the program gives:

$$\int_2 x^2 e^{-0.6x}\,dx \approx 7.9683553612 \quad \text{with} \quad abs\ err = 1.1822 \times 10^{-6}.$$

c. The MatLab program below can be used to compute the Composite Simpson's rule with $n = 4$ or $h = 1$.

```
1  function S = compsimpth2_4C(a,b,N)
2  % Composite Simpson's Rule for function f(x)
3  % on [a,b] using 2N steps
4  f = @(x) 2*x.^2.*exp(-0.6*x);
5  h = (b-a)/(2*N);
6  i = 0:N-1;
7  xi = a+2*i*h;
8  xi1 = a+2*(i+0.5)*h;
9  xi2 = a+2*(i+1)*h;
10 S = (h/3)*sum(f(xi)+4*f(xi1)+f(xi2));
11 end
```

The Composite Simpson's Rule with $n = 4$ or $h = 1$ gives:

$$\int_0^4 2\,x^2 e^{-0.6x}\,dx \approx 8.0046983122 \quad \text{with} \quad abs\ err = 0.0363417688.$$

d. The Composite Simpson's rule is analyzed with the MatLab program almost identical to Problem 3c. The only differences between these are the tolerance entered, which is $tol = 1.1822 \times 10^{-6}$, $N = 2$ in Line 7, and Line 4 of the code:

```
4  f = @(x) 2*x.^2.*exp(-0.6*x);
```

This program gives:

$$\int_0^4 2\,x^2 e^{-0.6x}\,dx \approx 7.96835658.$$

and requires $n = 128$ steps with a stepsize of $h = 0.03125$ to obtain the accuracy to the desired tolerance. The graph below shows the log-log plot of the stepsize and the absolute error. The MatLab program `polyfit` gives the best fit slope to be $p = 3.9678$, which is very close to the expected $p = 4$.

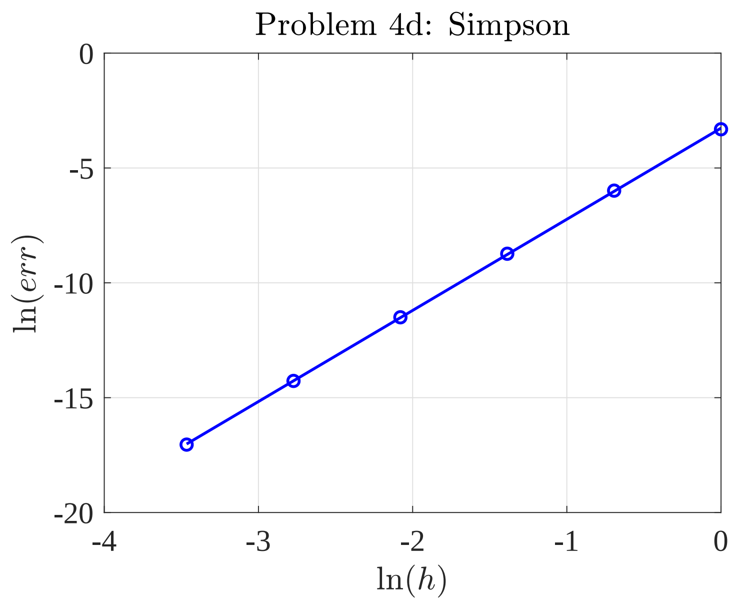


Figure 7: Graph of log of stepsize vs log of absolute error for Simpson's rule.

5. a. We have the two approximations to the integral with the Trapezoid rule on the interval $[a, b]$, where $\Delta x = \frac{b-a}{N}$. The first is

$$\int_a^b f(x)dx = T_{\Delta x}f + K_2(\Delta x)^2 + K_4(\Delta x)^4 + \cdots,$$

where

$$T_{\Delta x}f = \frac{\Delta x}{2}\left(f(a) + f(b) + 2\sum_{j=1}^{N-1} f(x_j)\right), \; x_j = a + j\Delta x.$$

By cutting the mesh size in half, we obtain:

$$\int_a^b f(x)dx = T_{\Delta x/2}f + \frac{K_2}{4}(\Delta x)^2 + \frac{K_4}{16}(\Delta x)^4 + \cdots$$

Multiplying the second approximation by 4 and subtracting the first approximation gives:

$$\begin{aligned}
3\int_a^b f(x)dx &= 4T_{\Delta x/2}f + K_2(\Delta x)^2 + \frac{K_4}{4}(\Delta x)^4 \\
&\quad - \left(T_{\Delta x}f + K_2(\Delta x)^2 + K_4(\Delta x)^4\right) + \cdots \\
3\int_a^b f(x)dx &= 4T_{\Delta x/2}f - T_{\Delta x}f - \frac{3K_4}{4}(\Delta x)^4 + \cdots \\
\int_a^b f(x)dx &= \frac{1}{3}\left(4T_{\Delta x/2}f - T_{\Delta x}f\right) - \frac{K_4}{4}(\Delta x)^4 + \cdots
\end{aligned}$$

So, you now have two approximations, one better than the other. Using these two approximations, show how you get the approximation

$$\int_a^b f(x)dx = \frac{1}{3}\left(4T_{\Delta x/2}f - T_{\Delta x}f\right) - \frac{K_4}{4}(\Delta x)^4 + \cdots$$

Thus, two second order methods are combined in a linear manner to produce a fourth order method (Richardson Extrapolation).

b. Below is a code which implements this new method (with the function in Part c).

```
function S = comptrapth2_5C(a,b,N)
% Modified Trapezoid Rule for function f(x)
% on [a,b] doubling steps til within tolerance
f = @(x) 0.5 + sin(x.^2).*exp(-0.4*x);
ex = 5.45052322777848;
h1 = (b-a)/N;
i = 0:N-1;
xi = a+i*h1;
xi1 = a+(i+1)*h1;
S1 = (h1/2)*sum(f(xi)+f(xi1));

h2 = (b-a)/(2*N);
j = 0:2*N-1;
xj = a+j*h2;
xj1 = a+(j+1)*h2;
S2 = (h2/2)*sum(f(xj)+f(xj1));

S = (1/3)*(4*S2 - S1);
eh = abs(S-ex)
end
```

c. A graph of the function shows a damped oscillation as seen below:

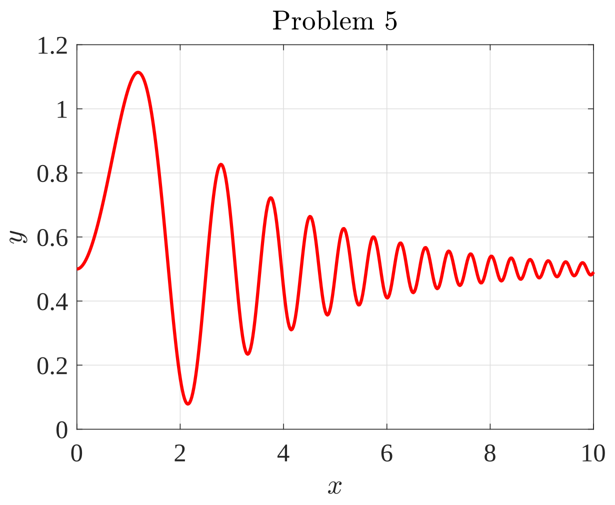


Figure 8: Graph of $f(x) = 0.5 + \sin(x^2)e^{-0.4x}$ on $[0, 10]$.

The method from Part a is tested on the integral:

$$\int_0^{10} 0.5 + \sin(x^2)e^{-0.4x}dx,$$

with $h = 0.5$ ($N = 20$). For this integral, Maple gives the numerical value of 5.450523228. The higher order Trapezoid method (from the program above) gives an approximation of 5.493486864, which has an absolute error of 0.042963636. If the CSR is implemented we obtain exactly the same answer (since they are effectively the same method). Note we must compare with $h = 0.25$ on the CSR for equivalency, since the modified Trapezoid uses a half-step. The clock times were 0.006565 for the modified Trapezoid rule compared to 0.004529 for the Simpson's rule, showing slightly more calculations for the modified Trapezoid rule and making it slightly less efficient.

d. The code above is expanded to find the solution to a particular tolerance and produce the log-log plot, which proves the order of convergence.

```
function [H,Eh,S,N] = comptraptolth2_5C(a,b,tol)
% Modified Trapezoid Rule for function f(x)
% on [a,b] doubling steps til within tolerance
f = @(x) 0.5 + sin(x.^2).*exp(-0.4*x);
ex = 5.45052322777848;
S0 = 0;
N = 10;
S = 1;
H = [];
Eh = [];
while (abs(S-S0)>tol)
    S0 = S;
    h1 = (b-a)/N;
    H = [H,h1];
    i = 0:N-1;
    xi = a+i*h1;
    xi1 = a+(i+1)*h1;
    S1 = (h1/2)*sum(f(xi)+f(xi1));

    h2 = (b-a)/(2*N);
    j = 0:2*N-1;
    xj = a+j*h2;
    xj1 = a+(j+1)*h2;
    S2 = (h2/2)*sum(f(xj)+f(xj1));

    S = (1/3)*(4*S2 - S1);
    eh = abs(S-ex);
    Eh = [Eh,eh];
    N = 2*N;
end
N = N/2;
p = polyfit(log(H),log(Eh),1)
nn = length(H);
lnH = log(H);
lnE = log(Eh);
plot([lnH(1),lnH(nn)],[(p(1)*lnH(1)+p(2)),(p(1)*lnH(nn)+p(2))],'b-');
hold on
plot(lnH,lnE,'bo');grid

```

```
fontlabs = 'Times New Roman';
xlabel('$\ln(h)$','FontSize',16,'FontName',fontlabs, ...
    'interpreter','latex');
ylabel('Absolute Error','FontSize',16,'FontName',fontlabs, ...
    'interpreter','latex');
mytitle = '';
title(mytitle,'FontSize',16,'FontName', ...
    'Times New Roman','interpreter','latex');
set(gca,'FontSize',16);

print -depsc th2_5C_gr.eps
end
```

We implement the code above using `comptraptolth2_5C(0,10,1e-8)`. It finds that the stepsize needs to be reduced to $\Delta x = 0.0078125$ ($N = 1280$) to obtain the tolerance of 10^{-8}. The graph below shows the log-log plot of the stepsize and the absolute error. The MatLab program `polyfit` gives the best fit slope to be $p = 4.3091$, which is consistent with the expected $p = 4$ (some wandering initially).

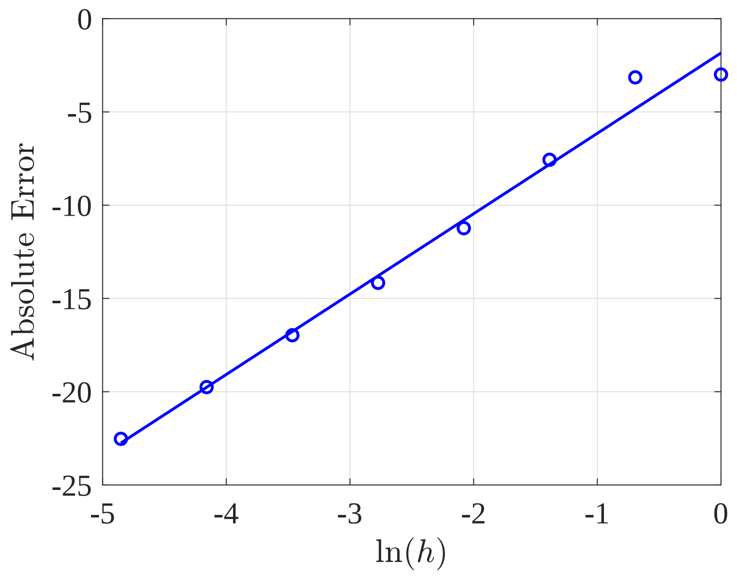


Figure 9: Graph of log of stepsize vs log of absolute error for the modified Trapezoid rule.

When the Composite Simpson's rule is modified for the function above, its graphic output is identical. The program is almost identical to the one in Problem 4, so is omitted. Both programs go to the same stepsize to obtain the tolerance set. (Note that the Simpson's $h = \Delta x/2$ for the modified Trapezoid rule.) The graphs of the log-log plots of stepsize vs absolute error are the same ultimately moving towards the $p = 4$.

6. a. This examines the risk of cadmium over a lifetime of accumulation. The risk increases with smoking adding additional amounts of cadmium over the normal entering through food. The average amount of Cd in the kidney (in mg) acquired mostly through food satisfies:

$$C(t) = \frac{74}{39}\left(1 - e^{-0.039t}\right).$$

The exposure to Cd, $E(t)$, is found by the integral:

$$E(t) = \int_0^t C(s)ds.$$

Below is table of the approximations using the Composite Trapezoid and Simpson's rules at ages 30, 50, and 70 with a stepsize of $h = 5$.

t	Exact	Trapezoid	Abs Err	Simpson	Abs Err
30	23.370910	23.264659	0.106251	23.370641	0.00026830
50	53.141539	53.009390	0.132149	53.141206	0.00033370
70	87.341372	87.197352	0.144021	87.341009	0.00036368

The programs for this table are below.

```
function T = comptrapth2_6Ca(a,b,N)
% Composite Trapezoid Rule for function f(x)
% on [a,b] using N steps
f = @(x) (74/39)*(1 - exp(-0.039*x));
h = (b-a)/N;
i = 0:N-1;
xi = a+i*h;
xi1 = a+(i+1)*h;
T = (h/2)*sum(f(xi)+f(xi1));
end
```

```
function S = compsimpth2_6Ca(a,b,N)
% Composite Simpson's Rule for function f(x)
% on [a,b] using 2N steps
f = @(x) (74/39)*(1 - exp(-0.039*x));
h = (b-a)/(2*N);
i = 0:N-1;
xi = a+2*i*h;
xi1 = a+2*(i+0.5)*h;
xi2 = a+2*(i+1)*h;
S = (h/3)*sum(f(xi)+4*f(xi1)+f(xi2));
end
```

b. To find the age with a tolerance of 10^{-2} of the average nonsmoker when he or she achieves an exposure level of 100 mg-yr, begin by setting the tolerance of a Composite Simpson's Rule quite low, say 10^{-6} to assure that the integral is close to the exact value over any specified interval. Next we connect this to a bisection method with a tolerance below the critical level of 10^{-2} (preferably an order of magnitude lower). Combining these techniques for this model we find the level of 100 mg yr is achieved at age:

$$\text{age}_{100} = 77.075 \text{ years}.$$

Below is the bisection script followed by the function needed for the Simpson's rule.

```
%BISECTION METHOD - Modify function below, then can
%   find its roots in [a,b] to tolerance tol
a = 50; b = 100;
tol = 0.0005;
S0 = compsimptolth2_6Cb(0,a,1e-6);
S1 = compsimptolth2_6Cb(0,b,1e-6);
f0 = S0 - 100;
f1 = S1 - 100;
if (f0*f1 > 0)
    fprintf('Error in choice of a and b\n');
    return
end
while (abs(b-a) ≥ tol)
    m = (a+b)/2;
    SM = compsimptolth2_6Cb(0,m,1e-6);
    fm = SM - 100;
    if (fm == 0)
        break;
    elseif(f1*fm < 0)
        a = m;
    else
        b = m;
    end
end
root = m
```

```
function [S] = compsimptolth2_6Cb(a,b,tol)
% Composite Simpson's Rule for function f(x)
% on [a,b] doubling steps til within tolerance
f = @(x) (74/39)*(1 - exp(-0.039*x));
S0 = 0;
N = 10;
S = 1;
while (abs(S-S0)>tol)
    S0 = S;
    h = (b-a)/(2*N);
    i = 0:N-1;
    xi = a+2*i*h;
    xi1 = a+2*(i+0.5)*h;
    xi2 = a+2*(i+1)*h;
    S = (h/3)*sum(f(xi)+4*f(xi1)+f(xi2));
    N = 2*N;
end
end
```

c. This part assumes someone smokes regularly after age 16 and more than doubles his or her intake of Cd. The functions change at $t = 16$, so the exposure integral uses $C(t)$ until $t = 16$, then for $t \geq 16$ the function becomes:

$$C_1(t) = \frac{74}{39}\left(1.9 + \left((1 - e^{-16k}) - 1.9\right) e^{-k(t-16)}\right),$$

where $k = 0.039$. The exposure, $E_1(t)$, for the smoker satisfies:

$$E_1(t) = \int_0^{16} C(s)ds + \int_{16}^{t} C_1(s)ds.$$

Below is table of the approximations using the Composite Trapezoid and Simpson's rules at ages 30, 50, and 70 for a smoker with a stepsize of $h = 1$.

t	Exact	Trapezoid	Abs Err	Simpson	Abs Err
30	28.855765	28.849177	0.006588	28.855764	6.679E-07
50	79.043172	79.033807	0.009365	79.043171	9.495E-07
70	141.099697	141.089058	0.010639	141.099695	1.0786E-06

This table uses a script that calls the Composite Trapezoid and Simpson's rules codes for $0 \leq t \leq 16$ and calls a variant of these rules. The script and top parts of the other codes are shown below.

```
% Smoker Cadmium script: Enter some age > 16
a = 70;
T0 = comptrapth2_6Ca(0,16,16);
N = a - 16;
T1 = comptrapth2_6Cc(16,a,N);
T = T0 + T1
```

```
function T = comptrapth2_6Cc(a,b,N)
% Composite Trapezoid Rule for function f(x)
% on [a,b] using N steps
f = @(x) (74/39)*(1.9 + ((1 - exp(-16*0.039)) - 1.9)*...
    exp(-0.039*(x-16)));
```

```
% Smoker Cadmium script: Enter some age > 16
a = 70;
T0 = compsimpth2_6Ca(0,16,8);
N = (a - 16)/2;
T1 = compsimpth2_6Cc(16,a,N);
T = T0 + T1
```

```
function S = compsimpth2_6Cc(a,b,N)
% Composite Simpson's Rule for function f(x)
% on [a,b] using 2N steps
f = @(x) (74/39)*(1.9 + ((1 - exp(-16*0.039)) - 1.9)*...
    exp(-0.039*(x-16)));
```

e. To find the age with a tolerance of 10^{-2} of the smoker described above when he or she achieves an exposure level of 100 mg-yr, begin by using the Composite Simpson's Rule for the nonsmoker to age 16. Then proceed as Part b with the code modified to use the smoker cadmium function. In this case we find the level of 100 mg yr is achieved at age:

$$\text{age}_{100} = 57.050 \text{ years.}$$

Below is the bisection script followed by the function needed for the Simpson's rule (first lines, which are followed as above in Part b.

```matlab
%BISECTION METHOD - Modify function below, then can
%   find its roots in [a,b] to tolerance tol
a = 50; b = 100;
tol = 0.0005;
B0 = compsimptolth2_6Cb(0,16,1e-6);
S0 = compsimptolth2_6Ce(16,a,1e-6);
S1 = compsimptolth2_6Ce(16,b,1e-6);
f0 = B0 + S0 - 100;
f1 = B0 + S1 - 100;
if (f0*f1 > 0)
    fprintf('Error in choice of a and b\n');
    return
end
while (abs(b-a) ≥ tol)
    m = (a+b)/2;
    SM = compsimptolth2_6Ce(16,m,1e-6);
    fm = B0 + SM - 100;
    if (fm == 0)
        break;
    elseif(f1*fm < 0)
        a = m;
    else
        b = m;
    end
end
root = m
```

```matlab
function [S] = compsimptolth2_6Ce(a,b,tol)
% Composite Simpson's Rule for function f(x)
% on [a,b] doubling steps til within tolerance
f = @(x) (74/39)*(1.9 + ((1 - exp(-16*0.039)) - 1.9)*...
    exp(-0.039*(x-16)));
```

e. Below is a graph of the two models, $E(t)$ and $E_1(t)$, for exposure to Cd for $t \in [0, 70]$. The graphs show aging increases the Cd exposure in all people (like many heavy metals). The increase in Cd by a factor of 1.9 results in significant increases of Cd exposure for the smoker. As seen above, the nonsmoker takes about 77 years to achieve an exposure of 100 mg yr, while the smoker achieves this at the significantly lower age near 57. At age 30, the smoker has about a 23.5% higher exposure than the nonsmoker. However, by age 60 the smoker has 56.1% higher exposure, so significantly more risk due to Cd. Thus, the cumulative effects of smoking take a serious toll with aging.

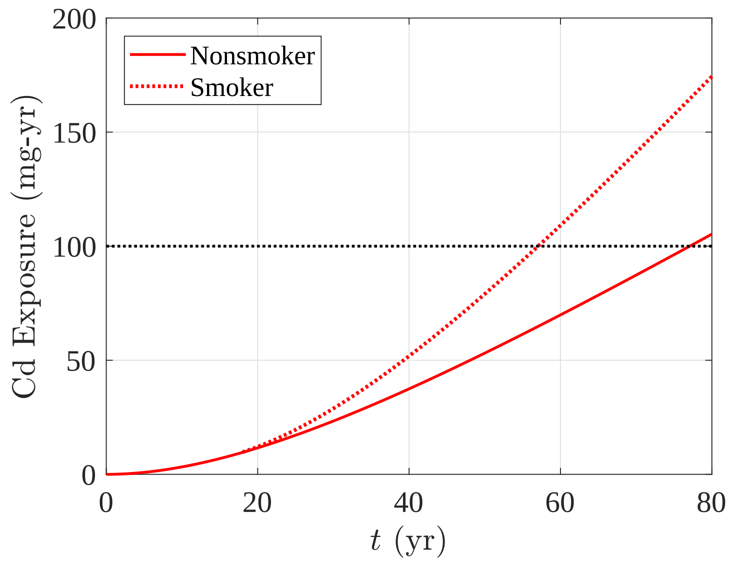


Figure 10: Graph of Cadmium exposure - Smoking vs Nonsmoking.