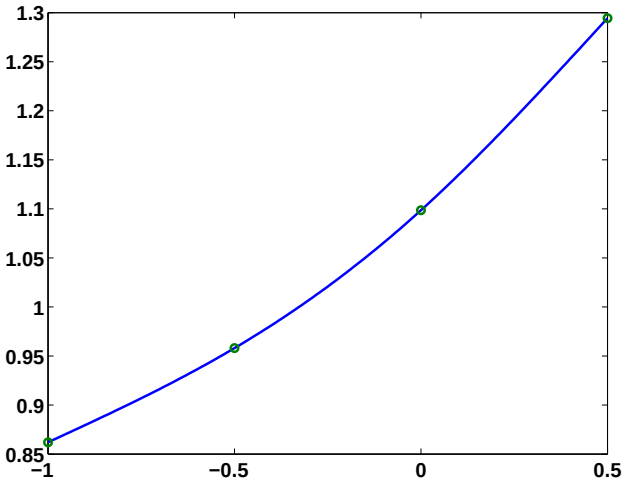
Homework 5 - Solutions

BF 3.4.4. d. The spline interpolation routine below produces the following coefficients:

x_i	a_i	b_i	c_i	d_i
-1.0	0.8619948	0.175637848	0	0.065650928
-0.5	0.95802009	0.224876044	0.098476392	0.02828072
0	1.0986123	0.344562976	0.140897472	-0.093931648

which gives the natural cubic spline:

$$
\begin{aligned}
S_0(x) &= 0.8619948 + 0.175637848(x+1) + 0.065650928(x+1)^3, && x \in [-1, -0.5],\\
S_1(x) &= 0.95802009 + 0.224876044(x+0.5) + 0.098476392(x+0.5)^2 \\
&\quad + 0.02828072(x+0.5)^3, && x \in [-0.5, 0],\\
S_2(x) &= 1.0986123 + 0.344562976\,x + 0.140897472\,x^2 - 0.093931648\,x^3, && x \in [0, 0.5].
\end{aligned}
$$



Main Code

```
%function z = spline1(n, x, f)

clear all
load ex344d
n =length(f);

for i = 1:n-1
    h(i) = x(i+1)-x(i);
    a = f;
end

for i = 2:n-1
```

```
    alpha(i-1) = (3./h(i)).*(a(i+1)-a(i))-(3./h(i-1)).*(a(i)-a(i-1));
end
rhs = [0 alpha 0]';

for i = 1:n-2
    l(i) = 2*(h(i)+h(i+1));
end
T = [0 h(1:n-2) 0 ; 1 l(1:n-2) 1 ; 0 h(2:n-1) 0]';

c = trisolve(T, rhs);
for j = 1:n-1
    b(j) = (a(j+1)-a(j))./h(j)-h(j)*(c(j+1)+2*c(j))./3;
    d(j) = (c(j+1)-c(j))./(3*h(j));
end

y = min(x):0.01:max(x);
Sy = evalspline(a,b,c,d,x,y);
plot(y, Sy, '-', x, f, 'o')
print -depsc spline_free.eps
```

Evaluation Subroutine

```
%
% function S = evalspline(a,b,c,d,x,y)
%
% Inputs:
%   a-b-c-d the coefficients that define the spline
%   x are the node-points
%   y the point(s) where you want to evaluate the spline
%

function S = evalspline(a,b,c,d,x,y)

if( length(a) ~= length(x) )
  error(['The first argument (a) must be the same length as the 5th' ...
 ' argument (x)']);
end

if( length(b) < length(x)-1 )
  error(['The second argument (b) cannot be more then one element' ...
 ' shorter than the 5th argument (x)']);
end

if( length(c) ~= length(x) )
  error(['The third argument (a) must be the same length as the 5th' ...
 ' argument (x)']);
end

if( length(d) < length(x)-1 )
```

```
  error(['The second argument (b) cannot be more then one element' ...
 ' shorter than the 5th argument (x)']);
end

if( min(y) < min(x) )
  error(['The values in the 6th argument (y) should not be less than' ...
 ' the values in the 5th argument (x)']);
end

if( max(y) > max(x) )
  error(['The values in the 6th argument (y) should not be greather' ...
 ' than the values in the 5th argument (x)']);
end

S = zeros(size(y));

for k = 1:length(y)
  n = min([length(b) max(find(x<=y(k)))]);
  S(k) = polyval([d(n) c(n) b(n) a(n)],(y(k)-x(n)));
end
```

Tridiagonal solver

```
%
% Solver for tri-diagonal matrices.
%
% T must be in compact form: the sub-diagonal elements in the first
% column, the diagonal in the second column, and the super-diagonal
% in the third column.
%
% Note that T(1,1) and T(3,n) are never accessed, i.e. the
% subdiagonal entries start on the second row, and the
% superdiagonal elements end on the (n-1)st row.
%
function [x]=trisolve(T,b)

if nargin ~= 2
  error('trisolve:: two input arguments required.')
end
if nargout ~= 1
  error('trisolve: one output argument required.')
end

[n,m] = size(T);
if m ~= 3
  error('trisolve: matrix must have three columns.')
end
```

```
[N,M] = size(b);

if N ~= n
  error('trisolve: rhs and matrix must have same number of rows.');
end

work = zeros(n,1);

work(1) = T(1,2);
x(1,:)  = b(1,:);

% Forward sweep.
for i=2:n
  beta    = T(i,1)/work(i-1);
  x(i,:)  = b(i,:) - beta*x(i-1,:);
  work(i) = T(i,2) - beta*T(i-1,3);
end

x(n,:) = x(n,:)/work(n);

% Backward sweep.
for i=n-1:-1:1
  x(i,:) = (x(i,:) - T(i,3)*x(i+1,:)) / work(i);
end
```

BF 3.4.6. d. The data for the above problem were generated from the function $f(x) = \ln(e^x + 2)$. We evaluate the $f(x)$ and $S_2(x)$ at $x = 0.25$ and find the absolute error giving:

$$\begin{aligned} S_2(0.25) &= 1.192091454 \\ f(0.25) &= 1.189069931 \\ |S_2(0.25) - f(0.25)| &= 0.003021523. \end{aligned}$$

Similarly, we evaluate the derivative of the spline and the function to obtain:

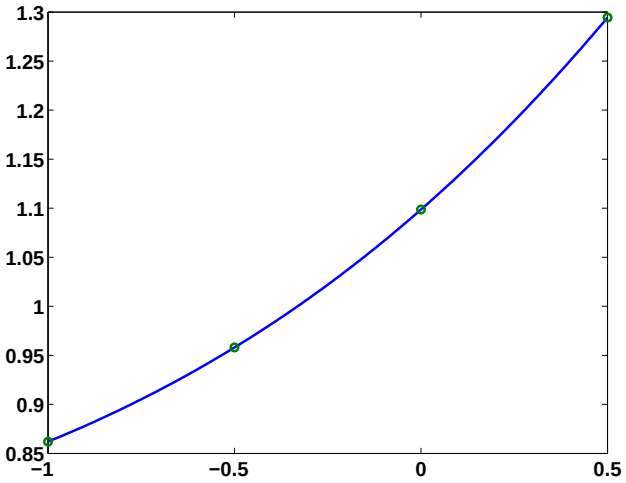
$$\begin{aligned} S_2{}'(0.25) &= 0.3973995280 \\ f'(0.25) &= 0.3909913152 \\ |S_2{}'(0.25) - f'(0.25)| &= 0.0064082128. \end{aligned}$$

BF 3.4.8. d. The clamped spline interpolation routine below produces the following coefficients:

x_i	a_i	b_i	c_i	d_i
-1.0	0.8619948	0.1553624	0.0653747467	0.0160032267
-0.5	0.95802009	0.232739567	0.0893795867	0.01502024
0	1.0986123	0.33338433	0.1119099467	0.0087579733

which gives the clamped cubic spline:

$$
\begin{aligned}
S_0(x) &= 0.8619948 + 0.1553624(x+1) + 0.0653747467(x+1)^2 \\
&\quad + 0.0160032267(x+1)^3, && x \in [-1, -0.5], \\
S_1(x) &= 0.95802009 + 0.232739567(x+0.5) + 0.0893795867(x+0.5)^2 \\
&\quad + 0.01502024(x+0.5)^3, && x \in [-0.5, 0], \\
S_2(x) &= 1.0986123 + 0.33338433\,x + 0.1119099467\,x^2 \\
&\quad + 0.0087579733\,x^3, && x \in [0, 0.5].
\end{aligned}
$$



Main Code

```
%function z = spline_clp(n, x, f)

clear all
load ex344d
fp0 = 0.1553624;
fpn = 0.45186276;
n =length(f);

for i = 1:n-1
    h(i) = x(i+1)-x(i);
    a = f;
end

alpha1 = 3*(a(2)-a(1))/h(1)-3*fp0;
alphan = 3*fpn -3*(a(n)-a(n-1))/h(n-1);

for i = 2:n-1
    alpha(i-1) = (3./h(i)).*(a(i+1)-a(i))-(3./h(i-1)).*(a(i)-a(i-1));
end
rhs = [alpha1 alpha alphan]';

t11 = 2*h(1);
```

```
tnn = 2*h(n-1);
for i = 1:n-2
    l(i) = 2*(h(i)+h(i+1));
end
T = [0 h(1:n-1) ; t11 l(1:n-2) tnn ; h(1:n-1) 0]';

c = trisolve(T, rhs);
for j = 1:n-1
    b(j) = (a(j+1)-a(j))./h(j)-h(j)*(c(j+1)+2*c(j))./3;
    d(j) = (c(j+1)-c(j))./(3*h(j));
end

y = min(x):0.01:max(x);
Sy = evalspline(a,b,c,d,x,y);
plot(y, Sy, '-', x, f, 'o')
print -depsc spline_clpd.eps
```

BF 4.1.5. The three-point formulae for derivatives are

$$\begin{aligned} f'(x_0) &= \frac{1}{2h}\left[-3f(x_0)+4f(x_0+h)-f(x_0+2h)\right]+\frac{h^2}{3}f^{(3)}(\xi_0), \\ f'(x_0) &= \frac{1}{2h}\left[-f(x_0-h)+f(x_0+h)\right]+\frac{h^2}{6}f^{(3)}(\xi_1), \\ f'(x_0) &= \frac{1}{2h}\left[f(x_0-2h)-4f(x_0-h)+3f(x_0+2h)\right]+\frac{h^2}{3}f^{(3)}(\xi_2), \end{aligned}$$

x	f(x)	f'(x)
8.1	16.94410	3.092050
8.3	17.56492	3.116150
8.5	18.19056	3.139975
8.7	18.82091	3.163525

BF 4.4.2(a). For trapezoid rule, we see that with $n = 4$

$$\int_{-0.5}^{0.5} \cos^2(x)dx \approx 0.91193343.$$

BF 4.4.2(c). We see that with $n = 8$

$$\int_{0.75}^{1.75} (\sin^2(x)) - 2x\sin(x) + 1)dx \approx -0.48932272.$$

The trapezoid rule is given by the MatLab program:

```
% Trapezoid Method for BF 4.4;
%
f  = inline('(cos(x))^2');
n = 4;
```

```
a = -0.5;
b = 0.5;

h = (b-a)/n;

fprintf('-----------------------------------------------\n');
fprintf('  n              h           sum        \n');
fprintf('-----------------------------------------------\n');

sum = h*(f(a)+f(b))/2;
for j = 1:(n-1)
    x = a+j*h;
    sum = sum + h*f(x);
end
fprintf('%3d   %8.6e   % 12.8f \n',n, h, sum);
```

BF 4.4.4(a). For Simpson's rule, we see that with $n = 4$

$$\int_{-0.5}^{0.5} \cos^2(x)dx \approx 0.92088605.$$

BF 4.4.4(c). We see that with $n = 8$

$$\int_{0.75}^{1.75} (\sin^2(x)) - 2x\sin(x) + 1)dx \approx -0.48901182.$$

The Simpson's rule is given by the MatLab program:

```
% Simpson Method;
%
f  = inline('(cos(x))^2');
n = 4;
a = -0.5;
b = 0.5;

h = (b-a)/n;
k = n/2;

fprintf('-----------------------------------------------\n');
fprintf('  n              h           sum        \n');
fprintf('-----------------------------------------------\n');

sum = h*(f(a)+f(b))/3;
x = a+h;
sum = sum + (4*h/3)*f(x);
for j = 1:(k-1)
    x1 = a+(2*j)*h;
    x2 = a+(2*j+1)*h;
    sum = sum + (2*h/3)*f(x1)+(4*h/3)*f(x2);
end
fprintf('%3d   %8.6e   % 12.8f \n',n, h, sum);
```

BF 4.4.12. We integrate and obtain

$$\int_0^{\pi} x^2 \cos(x) dx = -2\pi = -6.283185308.$$

The codes above to increase n until the accuracy is within tolerance (10^{-4}).

a. The Composite Trapezoid Rule requires $n = 228$ with $h = 0.013779$ to obtain the approximation:

$$\int_0^{\pi} x^2 \cos(x) dx \approx -6.28328472.$$

b. The Composite Simpson's Rule requires $n = 18$ with $h = 0.174533$ to obtain the approximation:

$$\int_0^{\pi} x^2 \cos(x) dx \approx -6.28308755.$$

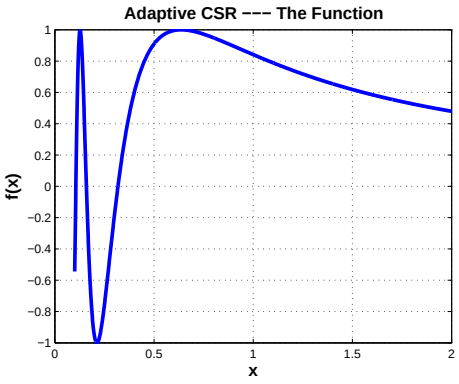
c. The Composite Midpoint Rule requires $n = 320$ with $h = 0.0097565$ to obtain the approximation:

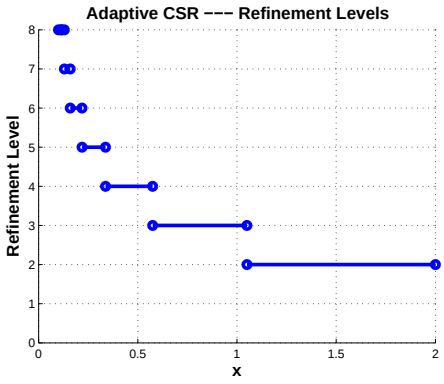
$$\int_0^{\pi} x^2 \cos(x) dx \approx -6.28308562.$$

BF 4.6.6(a). We want to compute

$$\int_{0.1}^{2} \sin\left(\frac{1}{x}\right) dx = 1.145580834,$$

where Maple gives us the value above. We use the Adaptive Quadrature with the Composite Simpson's rule. The tolerance is set at 10^{-3} for the program (given in class). The graphs below show the function and the refinement levels to obtain the solution to the level of desired accuracy. The computed value of the integral is 1.14549928, with an error of 1.611802×10^{-4}.





BF 4.7.1(b). We use Maple to show that

$$\int_0^1 x^2 e^{-x} dx = 2 - 5e^{-1} = 0.160602794.$$

To use Gaussian quadrature, we first convert the integral to be defined on $t \in [-1, 1]$, so we have

$$\int_0^1 x^2 e^{-x} dx = \frac{1}{2} \int_{-1}^1 \left(\frac{t+1}{2}\right)^2 e^{-\left(\frac{t+1}{2}\right)} dt.$$

Using Gaussian quadrature with $n = 2$, we have

$$\int_0^1 x^2 e^{-x} dx \approx \frac{1}{2} \left[\left(\frac{r_1+1}{2}\right)^2 e^{-\left(\frac{r_1+1}{2}\right)} + \left(\frac{r_2+1}{2}\right)^2 e^{-\left(\frac{r_2+1}{2}\right)} \right] = 0.1594104309,$$

where $-r_1 = 0.5773502692 = r_2$. The absolute error is 0.0011924.

BF 4.7.1(g). We use Maple to show that

$$\int_3^{3.5} \frac{x}{\sqrt{x^2-4}} dx = \sqrt{x^2-4}\Big|_3^{3.5} = 0.6362133458.$$

To use Gaussian quadrature, we first convert the integral to be defined on $t \in [-1, 1]$, so we have

$$\int_3^{3.5} \frac{x}{\sqrt{x^2-4}} dx = \frac{1}{4} \int_{-1}^1 \frac{(t+13)/4}{\sqrt{(t+13)^2/16-4}} dt.$$

Using Gaussian quadrature with $n = 2$, we have

$$\int_3^{3.5} \frac{x}{\sqrt{x^2-4}} dx \approx \frac{1}{4} \left[\frac{(r_1+13)/4}{\sqrt{(r_1+13)^2/16-4}} + \frac{(r_2+13)/4}{\sqrt{(r_2+13)^2/16-4}} \right] = 0.6361965650,$$

where $-r_1 = 0.5773502692 = r_2$. The absolute error is 0.00001678.

BF 4.7.2(b). We use the conversions above to get the form for the Gaussian quadrature. Using Gaussian quadrature with $n = 3$, we have

$$\begin{aligned} \int_0^1 x^2 e^{-x} dx &\approx \frac{1}{2} \left[c_1 \left(\frac{r_1+1}{2}\right)^2 e^{-\left(\frac{r_1+1}{2}\right)} + c_0 \left(\frac{1}{2}\right)^2 e^{-\left(\frac{1}{2}\right)} + c_1 \left(\frac{r_2+1}{2}\right)^2 e^{-\left(\frac{r_2+1}{2}\right)} \right] \\ &= 0.1605953868, \end{aligned}$$

where $-r_1 = 0.7745966692 = r_2$ and $c_1 = 0.5555555556$ and $c_0 = 0.8888888889$. The absolute error is 0.0000074073.

BF 4.7.2(g). We use the conversions above to get the form for the Gaussian quadrature. Using Gaussian quadrature with $n = 3$, we have

$$\begin{aligned} \int_3^{3.5} \frac{x}{\sqrt{x^2-4}} dx &\approx \frac{1}{4} \left[\frac{c_1(r_1+13)/4}{\sqrt{(r_1+13)^2/16-4}} + \frac{c_0 13/4}{\sqrt{(13)^2/16-4}} + \frac{c_1(r_2+13)/4}{\sqrt{(r_2+13)^2/16-4}} \right] \\ &= 0.6362131960, \end{aligned}$$

where $-r_1 = 0.7745966692 = r_2$ and $c_1 = 0.5555555556$ and $c_0 = 0.8888888889$. The absolute error is 1.499×10^{-7}.

BF 4.8.1(a). We use Maple to solve

$$\int_{2.1}^{2.5}\int_{1.2}^{1.4} xy^2 dydx = \frac{x^2}{2}\Big|_{2.1}^{2.5}\frac{y^3}{3}\Big|_{1.2}^{1.4} = 0.3115733334.$$

Next we approximate the double integral with Simpson's rule using $m = n = 4$. The program below produces

$$\int_{2.1}^{2.5}\int_{1.2}^{1.4} xy^2 dydx \approx 0.31157333.$$

We note that there is no error between the actual value of the double integral and the Simpson rule approximation, which should be true as Simpson's rule solves polynomials degree less than or equal to 3. (The integrand is linear in x and quadratic in y, so the values will be exact in each direction.)

```
f  = inline('x*y^2');
c = inline('1.2');
d = inline('1.4');
a = 2.1;
b = 2.5;
m = 4;
n = 4;
hx = (b-a)/n;
An = 0;
Ae = 0;
Ao = 0;
for i = 0:n
    x = a + i*hx;
    ya = c(x);
    yb = d(x);
    hy = (d(x) - c(x))/m;
    Bn = f(x,ya) + f(x,yb);
    Be = 0;
    Bo = 0;
    for j = 1:m-1
        y = ya + j*hy;
        z = f(x,y);
        if rem(j,2)==0
            Be = Be + z;
        else
            Bo = Bo + z;
        end
    end
    A1 = (Bn + 4*Bo + 2*Be)*hy/3;
    if i == 0 | i == n
        An = An + A1;
    else
        if rem(i,2)==0
            Ae = Ae + A1;
```

```
        else
            Ao = Ao + A1;
        end
    end
end
Dint = (An + 4*Ao + 2*Ae)*hx/3;
fprintf(1,'\n Double Integral is %12.8f \n',Dint);
```

BF 4.8.1(c). We use Maple to solve

$$\int_2^{2.2}\int_x^{2x}(x^2+y^3)dydx = \int_2^{2.2}\left(\frac{x^2y}{2}+\frac{y^4}{4}\right)\Bigg|_x^{2x} = \int_2^{2.2}\left(x^3+\frac{15}{4}x^4\right)dx = 16.50864.$$

Next we approximate the double integral with Simpson's rule using $m = n = 4$. The program below produces

$$\int_2^{2.2}\int_x^{2x}(x^2+y^3)dy\,dx \approx 16.50864063.$$

Again we note that there is almost no error between the actual value of the double integral and the Simpson rule approximation.

BF 4.8.2(a). The notes above about Simpson's rule solving exactly any polynomial of degree less than or equal to 3 means that we can use $n = m = 2$, the smallest possible values of n and m to get the actual value of the integral, which requires 9 function evaluations.

BF 4.8.2(c). The notes above about Simpson's rule solving exactly any polynomial of degree less than or equal to 3 means that we can use $m = 2$, the smallest possible values of m to get the actual value of the integral in the y direction. With $n = 2$, the approximation is not quite sufficiently accurate (in the x direction it is a polynomial of order 4), but increasing to $n = 4$ gives the approximate double integral accurate to 10^{-6} of the actual value. Thus, if we require $m = n$, then $m = n = 4$, which requires 25 function evaluations.

BF 4.9.2(a). This integral has a singularity at $x = 1$, so we transform the integral with $u = 1 - x$, so

$$\int_0^1 \frac{e^{-x}}{\sqrt{1-x}}dx = \int_0^1 \frac{e^{u-1}}{\sqrt{u}}du = \frac{1}{e}\left[\int_0^1 \frac{P_4(u)}{\sqrt{u}}du + \int_0^1 G(u)du\right],$$

which is singular at $u = 0$. With

$$P_4(u) = 1+u+\frac{u^2}{2}+\frac{u^3}{6}+\frac{u^4}{24} \qquad \text{and} \qquad G(u) = \begin{cases} \frac{e^u - P_4(u)}{\sqrt{u}}, & u \neq 0, \\ 0, & u = 0, \end{cases}$$

The integral with $P_4(u)$ is easily handled, giving

$$\int_0^1 \frac{P_4(u)}{\sqrt{u}}du = 2.923544974.$$

We apply Simpson's rule with $n = 6$ to the integral for $G(u)$, so

$$\begin{aligned}\int_0^1 G(u)du &\approx \frac{1}{18}(G(0)+4G(1/6)+2G(1/3)+4G(1/2)+2G(2/3)+4G(5/6)+G(1)) \\ &= 0.0017606160.\end{aligned}$$

$$\int_0^1 \frac{e^{-x}}{\sqrt{1-x}}dx = \frac{1}{e}\left[\int_0^1 \frac{P_4(u)}{\sqrt{u}}du + \int_0^1 G(u)du\right] \approx \frac{1}{e}(2.923544974+0.0017606160) = 1.076159786.$$

BF 4.9.3(c). With the transformation of $x = \frac{1}{t}$, we have

$$\int_1^\infty \frac{\cos(x)}{x^3}dx = \int_1^0 \frac{\cos(1/t)}{t^{-3}}(-t^{-2})dt = \int_0^1 t\cos(1/t)dt.$$

Clearly, $\lim_{t\to 0} t\cos(1/t) = 0$, so we directly apply the Composite Simpson's rule with $n = 6$. With $G(t) = t\cos(1/t)$, the result is

$$\begin{aligned}\int_0^1 t\cos(1/t)dt &= \frac{1}{18}(G(0) + 4G(1/6) + 2G(1/3) + 4G(1/2) + 2G(2/3) + 4G(5/6) + G(1)) \\ &\approx 0.05501681169.\end{aligned}$$

The actual value is 0.0181176220, so there a significant error in this calculation as one might expect from the rapid oscillations of the cosine function near $t = 0$.

Problem 2. A number of these integrals have singularities, so it is useful to use a quadrature technique that does not evaluate at the endpoints. Gaussian quadrature has the approximately the same accuracy as the Simpson's method. Below is a Composite Gaussian quadrature program that does successive halving of the interval size until the tolerance is met.

```
f  = inline('sin(x)/x');
a = 0;
b = 2*pi;
tol = 10^(-6);
max_iter = 20;
n = 1;
fprintf('-----------------------------------------------\n');
fprintf('  n              h           sum         \n');
fprintf('-----------------------------------------------\n');
h = b-a;
u = a + h*(1-1/sqrt(3))/2;
v = a + h*(1+1/sqrt(3))/2;
sum = h*(f(u)+f(v))/2;
sum1 = sum;
fprintf('%3d   %8.6e   %12.8f \n',n, h, sum);
while(n < max_iter)
    m = 2^n;
    sum = 0;
    h = (b-a)/m;
    for j = 1:m
        u = a + (j-1)*h + h*(1-1/sqrt(3))/2;
        v = a + (j-1)*h + h*(1+1/sqrt(3))/2;
        sum = sum + h*(f(u)+f(v))/2;
    end
    diff = abs(sum - sum1);
```

```
    fprintf('%3d   %8.6e   %12.8f  \n',m, h, sum);
    if (diff < tol)
        return;
    end
    sum1 = sum;
    n = n+1;
end
```

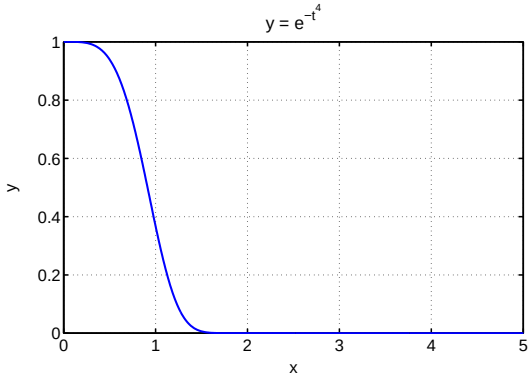
Part 2A. We split the integral into two parts and transform the second integral with $t = \frac{1}{x}$, so

$$\begin{aligned}\int_0^\infty e^{-t^4}dt &= \int_0^1 e^{-t^4}dt + \int_1^\infty e^{-t^4}dt \\ &= \int_0^1 e^{-t^4}dt + \int_0^1 \frac{e^{-x^{-4}}}{x^2}dx\end{aligned}$$

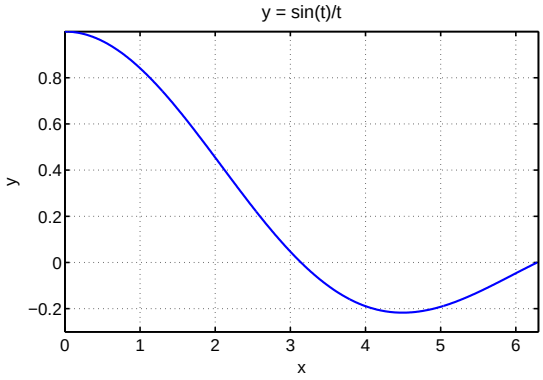
Each of these integrals are solved using the algorithm above. We obtain

$$\int_0^1 e^{-t^4}dt + \int_0^1 \frac{e^{-x^{-4}}}{x^2}dx \approx 0.84483859 + 0.06156386 = 0.90640245.$$

We iterated these integrals until the tolerance was 10^{-6}. The stepsize for the first was $h = 1/32$, and the second was $h = 1/16$. A graph of the integrand is seem below.



Problem 2A



Problem 2B

Part 2B. We used the Gaussian quadrature algorithm above for the integral below and iterated until the tolerance was 10^{-6}. The final stepsize was $h = 1/32$, and the resulting integral is approximated by

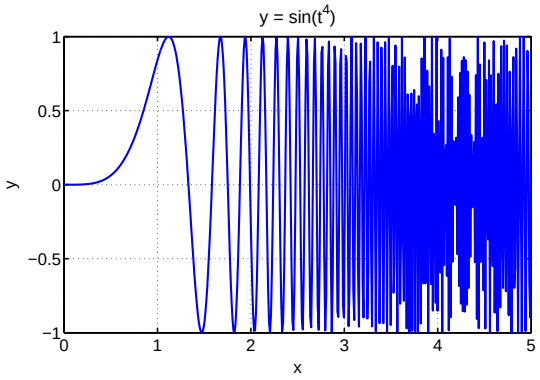
$$\int_0^{2\pi} \frac{\sin(t)}{t}\, dt \approx 1.41815162.$$

A graph of the integrand is seem above.

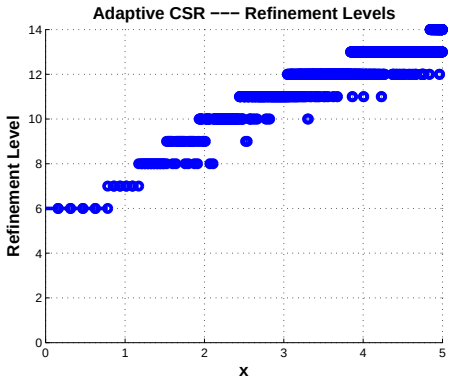
Part 2C. The integrand for this integral oscillates wildly. We tried both the Gaussian quadrature formula above and the adaptive CSR. The latter should be more efficient computing the integral as the integrand for smaller t is well-behaved. The Gaussian quadrature needed to use a stepsize of $h = 1/4096$ to be within a tolerance of 10^{-5}, while the graph below shows the stepsizes for the adaptive CSR for a tolerance of 10^{-5} The resulting integral is approximated by

$$\int_0^5 \sin(t^4)\, dt \approx 0.34883349 \quad (\text{for ACSR } = 0.34883335).$$

A graph of the integrand is seem below.



Problem 2C



Problem 2C steps