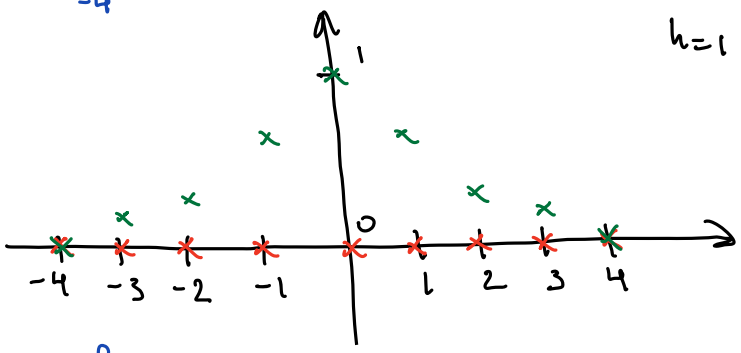
Problem 1

(a) $\int_{-\infty}^{\infty} e^{-x^2}dx \approx \int_{-4}^{4} e^{-x^2}dx$ — b/c the integrand is small for $x \notin [-4,4]$



$I_1 = \int_{-4}^{4} e^{-x^2}dx = 1\cdot\left(\frac{f(-4)}{2} + f(-3)+f(-2)+f(-1)+f(0)+f(1)+f(2)+f(3)+\right.$

$\left. + \frac{f(4)}{2}\right) \overset{\text{symmetry}}{=} f(0) + 2f(1) + 2f(2) + 2f(3) + f(4) = 1 + 2\cdot 0.367879 +$

$+ 2\cdot 0.018316 + 2\cdot 0.000123 + 2\cdot 0 = 1.772636$

$I_{0.5} = 0.5\cdot(f(0) + 2f(0.5) + 2f(1) + 2f(1.5) + 2f(2) + 2f(2.5) +$

$+ 2f(3) + 2f(3.5) + f(4)) = 0.5f(0) + f(0.5) + f(1) + f(1.5) + f(2) +$

$f(2.5) + f(3) + f(3.5) + f(4) = 0.5 + 0.778801 + 0.367879 + 0.105399 +$

$+ 0.018316 + 0.001930 + 0.000123 + 0.000005 = 1.772453$

Since order of accuracy for trapezoidal rule is (2).

$$\text{rate of convergence} = \frac{|I_{0.5} - I|}{|I_1 - I|^{(2)}} = \frac{|1.772453 - 1.772454|}{|1.772636 - 1.772454|^{(2)}} \approx 30$$

Having this high rate of convergence indicates that for our specific function the order was way better than the usual quadratic order. Hence, indeed for smooth functions the integral is computed to very high order of accuracy.

(b) Let $a=1$, $b=n+1$, $h=1$ and $f(x)=x$.

E-M formula:

$$\int_a^b f(x)dx = h\left[\frac{f(a)}{2} + f(a+h) + \dots + f(b-h) + \frac{f(b)}{2}\right] -$$

$$- \sum_{r=1}^{p} \frac{h^{2r} B_{2r}}{(2r)!} \left[f^{(2r-1)}(b) - f^{(2r-1)}(a) \right] + O(h^{2p+2})$$

Since $f(x) = x$ is smooth $p = \infty$ and $O(h^{2p+2}) = 0$.

$f'(x) = 1$, $f^{(k)}(x) = 0$ for $k > 1$. Hence, for our specific case:

$$\int_1^{n+1} x\,dx = \frac{1}{2} + 2 + \dots + n + \frac{n+1}{2} - \frac{B_2}{2}[1 - 1]$$

$$\frac{(n+1)^2}{2} - \frac{1}{2} - \frac{n}{2} = 1 + 2 + \dots + n$$

Therefore: $S = 1 + 2 + \dots + n = \frac{n^2 + 2n + 1 - 1 - n}{2} = \frac{n^2 + n}{2} = \frac{n(n+1)}{2}$

Problem 2

$$A^* = A(h) + c_n h^n + c_{n+1} h^{n+1} + \dots$$

$$R(h,k) = \frac{k^n A(h) - A(kh)}{k^n - 1}$$

(a) $k^n A^* = k^n A(h) + k^n c_n h^n + k^n c_{n+1} h^{n+1} + \dots$

$A^* = A(kh) + c_n (kh)^n + c_{n+1} (kh)^{n+1} + \dots$ } −

$$(k^n - 1) A^* = k^n A(h) - A(kh) + (1-k) c_{n+1} k^n h^{n+1} + \dots$$

$$+ (1 - k^2) c_{n+2} k^n h^{n+2} + \dots$$

$$A^* = \frac{k^n A(h) - A(kh)}{k^n - 1} + \frac{k^n - k^{n+1}}{k^n - 1} c_{n+1} h^{n+1} + \frac{k^n - k^{n+2}}{k^n - 1} c_{n+2} h^{n+2} + \dots$$

$$R(h,k) = A^* + \frac{k^{n+1} - k^n}{k^n - 1} c_{n+1} h^{n+1} + \frac{k^{n+2} - k^n}{k^n - 1} c_{n+2} h^{n+2} + \dots$$

$$R(h,k) = A^* + O(h^{n+1})$$

Also:

$$A^* = A(k^2 h) + c_n (k^2 h)^n + c_{n+1} (k^2 h)^{n+1} + \dots$$

Then:

$$A(h) + c_n h^n + c_{n+1} h^{n+1} + \dots = A(kh) + c_n k^n h^n + c_{n+1} k^{n+1} h^{n+1} + \dots$$

$$A(h) - A(kh) = (k^n - 1) c_n h^n + (k^{n+1} - 1) c_{n+1} h^{n+1} + \dots$$

$$A(kh) - A(k^2 h) = (k^n - 1) c_n (kh)^n + (k^{n+1} - 1) c_{n+1} (kh)^{n+1} + \dots$$

Since $c_n h^n$ is the dominant term:

$$A(h) - A(kh) \approx (k^n - 1) c_n h^n$$
$$A(kh) - A(k^2 h) \approx (k^n - 1) c_n (kh)^n$$

$$\frac{A(h) - A(kh)}{A(kh) - A(k^2 h)} \approx \frac{1}{k^n} = k^{-n}$$

(b) $T_{m,n} = A_m(h)$, where $h = \frac{b-a}{2^{m-1}}$

$T_{m-1,n} = A_{m-1}(h)$, where $h = \frac{b-a}{2^{m-2}}$

$T_{m+1,n} = A_{m+1}(h)$, where $h = \frac{b-a}{2^m}$

Let $T_{m-1,n} = A(h)$, $h = \frac{b-a}{2^{m-2}}$

Then $T_{m,n} = A(kh)$, where $k = \frac{1}{2}$, s.t. $kh = \frac{b-a}{2^{m-1}}$

and $T_{m+1,n} = A(k^2 h)$, where $k^2 h = \frac{b-a}{2^m}$

Hence: $R_{m,n} = \frac{T_{m,n} - T_{m-1,n}}{T_{m+1,n} - T_{m,n}} = \frac{A(kh) - A(h)}{A(k^2 h) - A(kh)} \approx k^{-2n} =$ (because the order is $2M$)

$$= \left(\frac{1}{2}\right)^{-2n} = 4^n$$

(c) At the end of the pdf

(d) At the end of the pdf

Convergence rates are different because for $[0,1]$ sqrt function is not smooth, hence the rate is way slower. Whereas for $[1,2]$ it is smooth and the rate is 4^n as expected.

Problem 3

(a) At the end of the pdf

From the last log graph of the error against N, which looks linear, we can say that the error decreases

exponentially, which by definition means that the accuracy increases better than any polynomial, hence spectrally.

(b) I have tried $\alpha = 0, \beta = 1$; $\alpha = 1, \beta = 0$; $\alpha = 0, \beta = 0$; $\alpha = 1, \beta = 1$ and program worked fine for all of them and since other cases are just the matter of scaling, we can conclude that there are ND values of α, β for which program fails.

(c) At the end of the pdf

I have plotted the log error of of this one too and seems like the rate of convergence is a bit slower, but I feel like it is still spectral since sine is C^{∞}, or the error can decay proportionally to some exponent.

Problem 4

(a) At the end of the pdf

$$d_k = \frac{2}{N} \cdot \sum_{i=0}^{N-1} f(x_i) \cdot \cos(k \cdot \cos^{-1}(x_i))$$

(b) At the end of the pdf

$$\int d_k T_k(x) = \frac{d_k}{2} \left(\frac{T_{k+1}(x)}{k+1} + \frac{T_{k-1}(x)}{k-1} \right) \Rightarrow$$

$$\Rightarrow \beta_k = \frac{d_{k-1} - d_{k+1}}{2k} \quad \text{for} \quad k \geq 1$$

(c) At the end of the pdf

(d) At the end of the pdf

Problem 2.

(c) Romberg integration code:

```
function [I, T, R] = romb(N,a,b,f)
T = zeros(N,N);
R = zeros(N,N);

for m = 1:N
    M = 2^(m-1);
    h = (b-a)/M;
    x = f(a:h:b);
    x(1) = x(1)/2;
    x(end) = x(end)/2;
    T(m,1) = h*(sum(x));
    for n = 2:m
        T(m,n) = T(m,n-1) + (T(m,n-1)-T(m-1,n-1))/(4^(n-1)-1);
    end
end

for m = 1:N-1
    for n = 1:m-1
        R(m,n) = (T(m,n)-T(m-1,n))/(T(m+1,n)-T(m,n));
    end
end

I = T(N,N);
```

Below are the T and R matrices for N=6:

Command Window

```
>> [I, T, R] = romb(6, 0, 1, @exp);
>> vpa(I)

ans =

1.718282

>> vpa(T)

ans =

[ 1.859141,        0,        0,        0,        0,        0]
[ 1.753931, 1.718861,        0,        0,        0,        0]
[ 1.727222, 1.718319, 1.718283,        0,        0,        0]
[ 1.720519, 1.718284, 1.718282, 1.718282,        0,        0]
[ 1.718841, 1.718282, 1.718282, 1.718282, 1.718282,        0]
[ 1.718422, 1.718282, 1.718282, 1.718282, 1.718282, 1.718282]

>> vpa(R)

ans =

[        0,        0,        0,        0, 0, 0]
[ 3.939087,        0,        0,        0, 0, 0]
[ 3.984476, 15.63429,        0,        0, 0, 0]
[   3.9961, 15.90684, 62.44563,        0, 0, 0]
[ 3.999024,  15.9766, 63.60401, 249.6836, 0, 0]
[        0,        0,        0,        0, 0, 0]
```

As we can see from the R matrix, its nonzero entries are indeed the powers of 4 (4, 16, 64, 256) that all depend on n (number of the corresponding column), thus validating the estimate.

(d)

MI = the number of mesh intervals

ES = the number of exploration steps

R = ratios

E = errors

Results for [0,1]:

```
>> vpa(MI)

ans =

[  1.0,    0,    0,    0,    0,    0]
[  2.0,  2.0,    0,    0,    0,    0]
[  4.0,  4.0,  4.0,    0,    0,    0]
[  8.0,  8.0,  8.0,  8.0,    0,    0]
[ 16.0, 16.0, 16.0, 16.0, 16.0,    0]
[ 32.0, 32.0, 32.0, 32.0, 32.0, 32.0]

>> vpa(ES)

ans =

[ 1.0,   0,   0,   0,    0,    0]
[ 1.0, 2.0,   0,   0,    0,    0]
[ 1.0, 2.0, 4.0,   0,    0,    0]
[ 1.0, 2.0, 4.0, 8.0,    0,    0]
[ 1.0, 2.0, 4.0, 8.0, 16.0,    0]
[ 1.0, 2.0, 4.0, 8.0, 16.0, 32.0]

>> vpa(R)

ans =

[        0,        0,        0,        0, 0, 0]
[ 2.606451,        0,        0,        0, 0, 0]
[ 2.675907, 2.816273,        0,        0, 0, 0]
[ 2.723765, 2.825905, 2.827971,        0, 0, 0]
[ 2.756165, 2.827958, 2.828398, 2.828418, 0, 0]
[        0,        0,        0,        0, 0, 0]

>> vpa(E)

ans =

[   0.1666667,    0.6666667,    0.6666667,   0.6666667,    0.6666667,    0.6666667]
[  0.06311328,   0.02859548,    0.6666667,   0.6666667,    0.6666667,    0.6666667]
[  0.02338362,    0.0101404,  0.008910063,   0.6666667,    0.6666667,    0.6666667]
[ 0.008536445,  0.003587387,  0.003150519, 0.003059098,    0.6666667,    0.6666667]
[  0.00308547,  0.001268478,  0.001113884, 0.001081557,  0.001073802,    0.6666667]
[  0.00110773, 0.0004484839, 0.0003938176, 0.000382388, 0.0003796462, 0.0003789676]
```

Results for [1,2]:

```
>> vpa(MI)

ans =

[  1.0,    0,    0,    0,    0,    0]
[  2.0,  2.0,    0,    0,    0,    0]
[  4.0,  4.0,  4.0,    0,    0,    0]
[  8.0,  8.0,  8.0,  8.0,    0,    0]
[ 16.0, 16.0, 16.0, 16.0, 16.0,    0]
[ 32.0, 32.0, 32.0, 32.0, 32.0, 32.0]

>> vpa(ES)

ans =

[ 1.0,    0,    0,    0,    0,    0]
[ 1.0, 2.0,    0,    0,    0,    0]
[ 1.0, 2.0, 4.0,    0,    0,    0]
[ 1.0, 2.0, 4.0, 8.0,    0,    0]
[ 1.0, 2.0, 4.0, 8.0, 16.0,    0]
[ 1.0, 2.0, 4.0, 8.0, 16.0, 32.0]
```

```
>> vpa(R)

ans =

[        0,        0,        0,        0, 0, 0]
[ 3.894481,        0,        0,        0, 0, 0]
[ 3.969246, 13.61861,        0,        0, 0, 0]
[ 3.991917, 15.18853, 44.57294,        0, 0, 0]
[ 3.997951,  15.7725, 56.25893, 141.1883, 0, 0]
[        0,        0,        0,        0, 0, 0]

>> vpa(E)

ans =

[    0.01184464,        1.218951,        1.218951,     1.218951,     1.218951,     1.218951]
[    0.00302559,   0.00008590851,        1.218951,     1.218951,     1.218951,     1.218951]
[  0.0007610923,   0.00000625964, 0.0000009497159,     1.218951,     1.218951,     1.218951]
[  0.0001905814, 0.0000004111096,     2.120759e-8,  6.469358e-9,     1.218951,     1.218951]
[ 0.00004766489,      2.60472e-8,    3.763758e-10, 4.572165e-11, 2.053091e-11,     1.218951]
[ 0.00001191745,     1.633671e-9,    6.102008e-12, 2.247091e-13, 4.618528e-14, 2.620126e-14]
```

First, as expected number of ME and ES do not change due to the domain. However, notice that the ratios are different. Especially, notice that for [0,1] the ratios do not match with the theory in the first glance. But in fact, it matches, because theory assumes that the function is differentiable in the integration interval, while sqrt(x) is not for [0,1], thus causing the ratios not to be the powers of 4. Since [1,2] matches the theory it converges spectrally and much faster than [0,1] as it can be seen from the above results.

Problem 3.

(a)

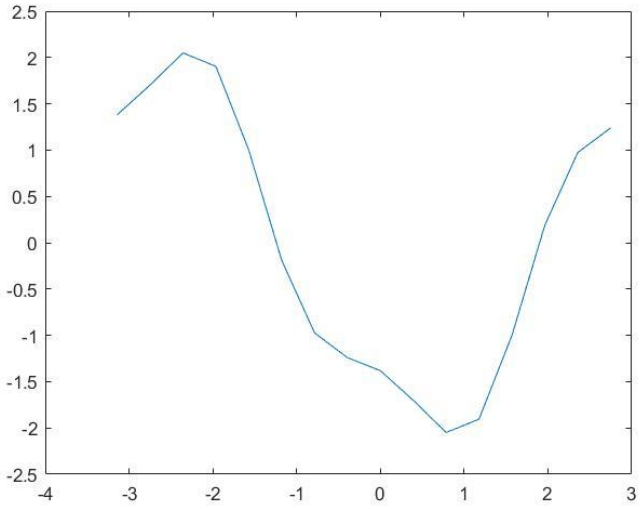
The exact f, that was found analytically:

```
function f = exactf(x)
f = -cos(x).*cos(cos(x)) - cos(cos(x)).*sin(x) - sin(cos(x)) - (sin(x)).^2 .* sin(cos(x));
```

FFT implementation:

```
function [f,fx,err] = fftpdesolver(a,b,N,alpha, betha)
x = sin(cos((a-(b-a)/2:(b-a)/N:b-(b-a)/2-(b-a)/N)));
y = fft(x);
index = (2*pi/(b-a))*[-N/2:N/2-1];
index = fftshift(index);
yx = i*index.*y;
yxx = -index.*index.*y;
f = real(ifft(yxx + alpha*yx - betha*y));
fx = exactf((a-(b-a)/2:(b-a)/N:b-(b-a)/2-(b-a)/N));
err = immse(f,fx);
plot((a-(b-a)/2:(b-a)/N:b-(b-a)/2-(b-a)/N), f);
%plot((a-(b-a)/2:(b-a)/N:b-(b-a)/2-(b-a)/N), fx);
```

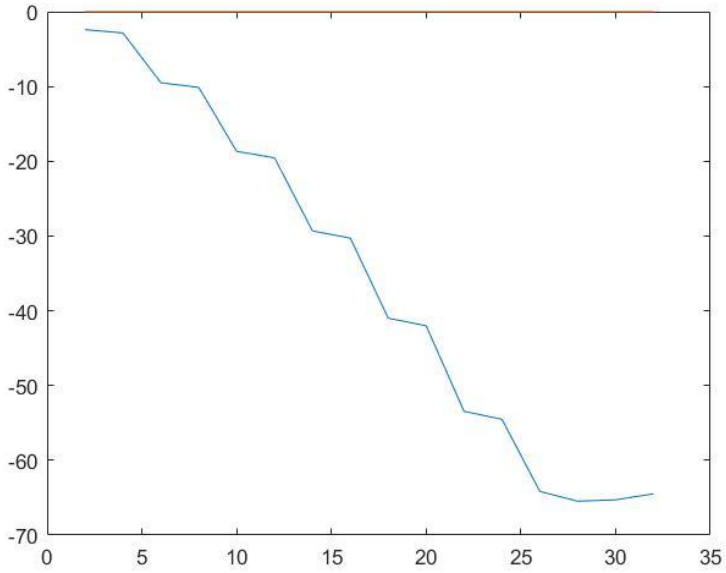
The plot of the resulting function:



The code to check the convergence rate:

```
N = 2:2:32;
err = zeros(length(N));
for n = 1:length(N)
    [f,fx,er] = fftpdesolver(0,2*pi, N(n), 1, 1);
    err(n) = log(vpa(er));
end
plot(N,vpa(err));
```

Plot of the log of the error:



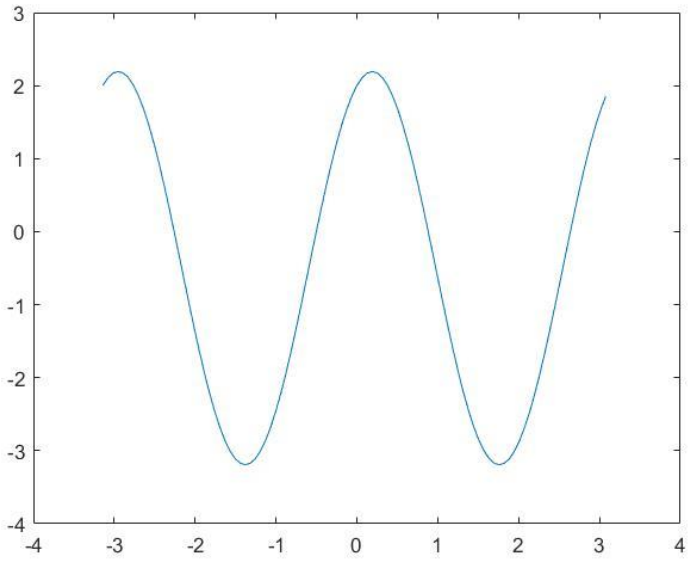
It can be seen from above that the function decays almost linearly, if we fit the line, which mean the error itself decays exponentially, thus meaning that convergence rate is faster than any polynomial and hence by definition spectral.

(b) No values of alpha and beta.

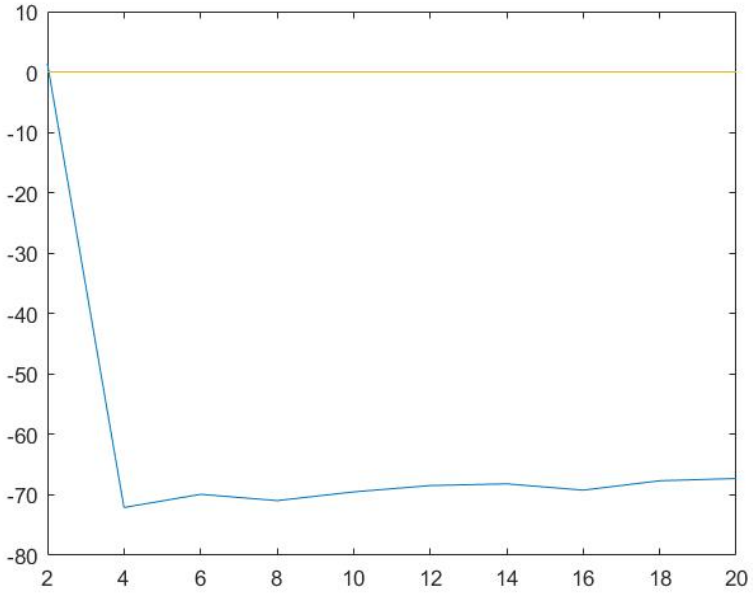
(c) Implementation:

```
function [f,fx,err] = fftpdesolver2(a,b,N,alpha, betha)
x = sin((a-(b-a)/2:(b-a)/N:b-(b-a)/2-(b-a)/N)).^2;
y = fft(x);
index = (2*pi/(b-a))*[-N/2:N/2-1];
index = fftshift(index);
yx = i*index.*y;
yxx = -index.*index.*y;
f = real(ifft(yxx + alpha*yx - betha*y));
fx = exactf((a-(b-a)/2:(b-a)/N:b-(b-a)/2-(b-a)/N));
err = immse(f,fx);
plot((a-(b-a)/2:(b-a)/N:b-(b-a)/2-(b-a)/N), f);
%plot((a-(b-a)/2:(b-a)/N:b-(b-a)/2-(b-a)/N), fx);
```

The resulting function which the same as the analytical solution:



And using the same method as in part a, the log plot of the error was given:



There is a fast decay at the beginning, but it's mostly because the initial guessed were incorrect. Then, log becomes more or less linear, which still looks like a spectral behavior.

Problem 4.

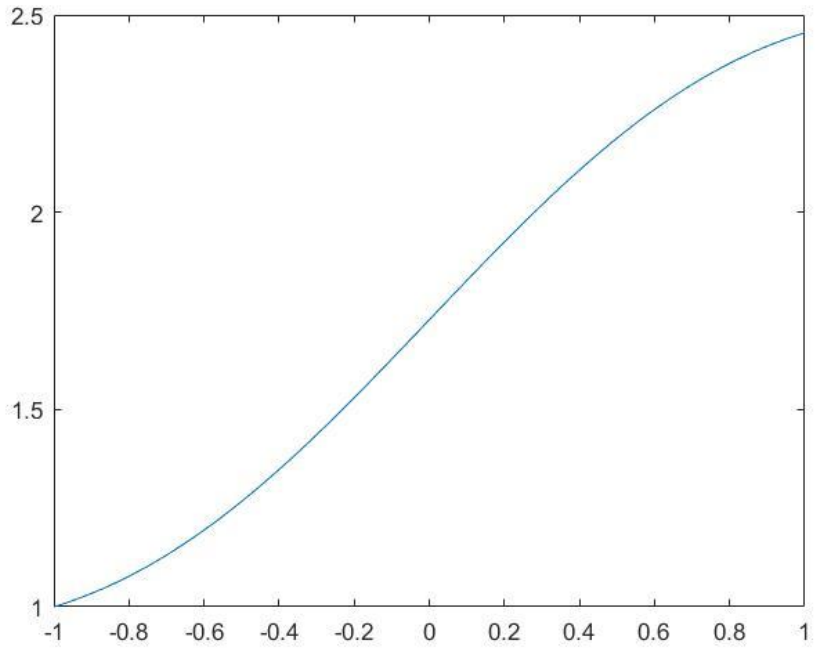
(a) See the code below

(b) See the code below

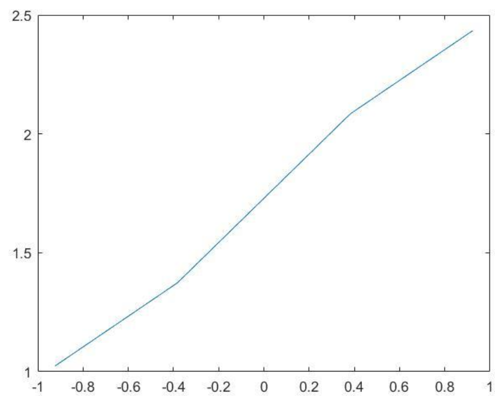
(c) See the code below

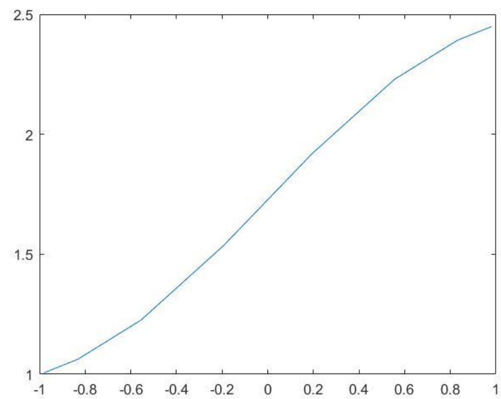
```
function u = fchebt(N)
t = -cos(((0:N-1)+1/2)/N*pi);
ft = cos(t).^2;
alpha = zeros([1 N]);
for n = 1:N
    alpha(n) = 2/N*dot(ft,cos((n-1)*acos(t)));
end
betha = zeros([1 N]);
c = 0.0;
sign = -1;
for n = 2:N-1
    betha(n) = (alpha(n-1)-alpha(n+1))/2/(n-1);
    c = c - sign*betha(n);
    sign = (-1)*sign;
end
betha(N) = alpha(N-1)/2/(N-1);
c = c - sign*betha(N);
betha(1) = c;
u = zeros([1 N]);

for n = 1:N
    u(n) = dot(betha,cos((0:N-1)*acos(t(n))))+1;
end
plot(t, u)
```

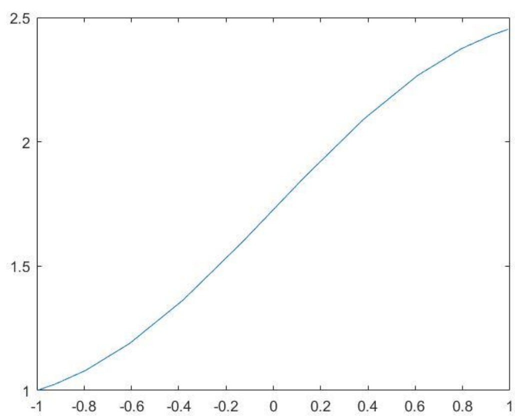
(d) Analytic Solution:

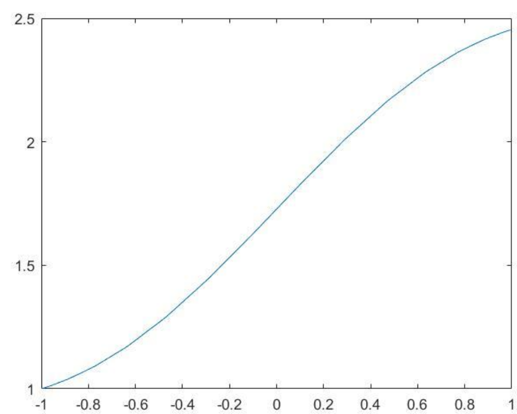


Numerical solutions for different number of points:



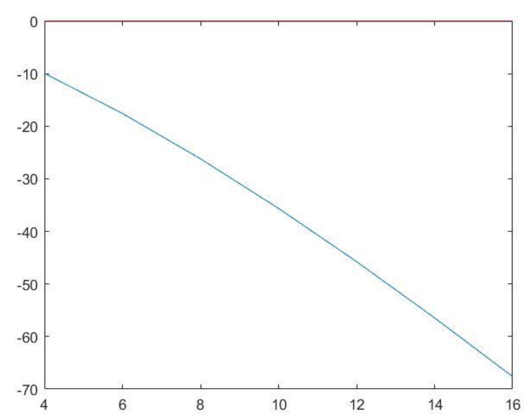






Code for plotting the log of the error with the plot of the Log(Error):

```
N = 4:2:16;
err = zeros(length(N));
for n = 1:length(N)
    [u,er] = fchebt(N(n));
    err(n) = log(vpa(er));
end
plot(N,vpa(err));
```



From the above plot it can be seen that, the numerical solution converges to the analytical one and its error's log plot is decaying linearly, which means that the error itself is decaying exponentially, which by definition means that the convergence rate is spectral and the program is spectrally accurate.