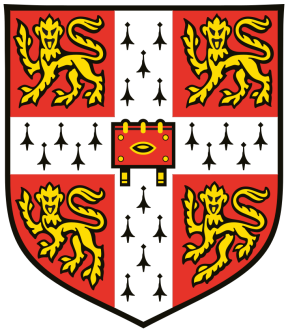
1.1

University of Cambridge

CATAM

Part II

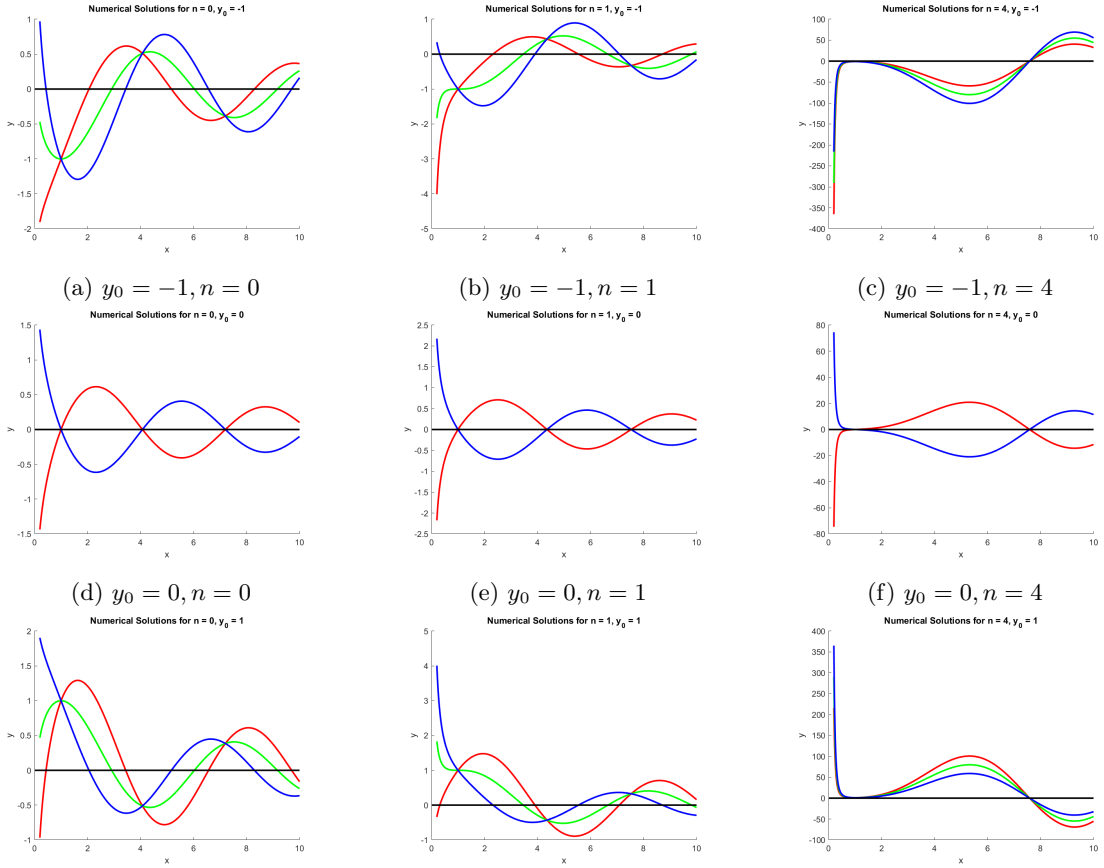
Fourier Transforms of Bessel Functions



UNIVERSITY OF
CAMBRIDGE

Question 1

Throughout this project, I have employed Matlab's ODE45 routine to investigate $n = 0, 1, 4$ for various initial y and y'. I have included some plots in figure 1 below. Different columns represent different values of n, while different rows represent different values of y_0. Each image has 3 different initial gradients, with $y'(0) = -1, 0, 1$ being represented in blue, green and red respectively. In all of these plots, we have taken $x_0 = 1$ and integrated both forwards and backwards.



(a) $y_0 = -1, n = 0$ (b) $y_0 = -1, n = 1$ (c) $y_0 = -1, n = 4$

(d) $y_0 = 0, n = 0$ (e) $y_0 = 0, n = 1$ (f) $y_0 = 0, n = 4$

(g) $y_0 = 1, n = 0$ (h) $y_0 = 1, n = 1$ (i) $y_0 = 1, n = 4$

Figure 1: Various Numerical Solutions to Bessel's Equation with different values for n, y_0 and y_0'.

We observe a few properties:

1. The function y almost always diverges at $x = 0$.

2. Solutions always look oscillatory and in the long term they decay like $1/\sqrt{x}$, as illustrated in figure 2.

3. All solutions with some initial y_0 intersect at the same points. This is true if we include more than three initial gradients, as illustrated in figure 3 below.
   To explain this, note that in the case $y_0 = 0$, all nontrivial solutions are scalar multiples of the solution f with $y_0 = 0, y_0' = 1$. This is since a solution is uniquely defined by y_0 and y_0', so if we seek a solution y with $y_0 = 0, y_0' = \lambda$, then λf satisfies these initial conditions and hence solves the initial value problem.

   Therefore, all the solutions passing through $(x_0, 0)$ intersect at the same points, their roots. Let's call these $x-$values $\lambda_1, \lambda_2, \lambda_3, \ldots$. Now, what if $y_0 \neq 0$? Well, suppose that f and g are both solutions passing through the point x_0, y_0. Then, $f - g$ passes through $(x_0, 0)$ so is 0 at

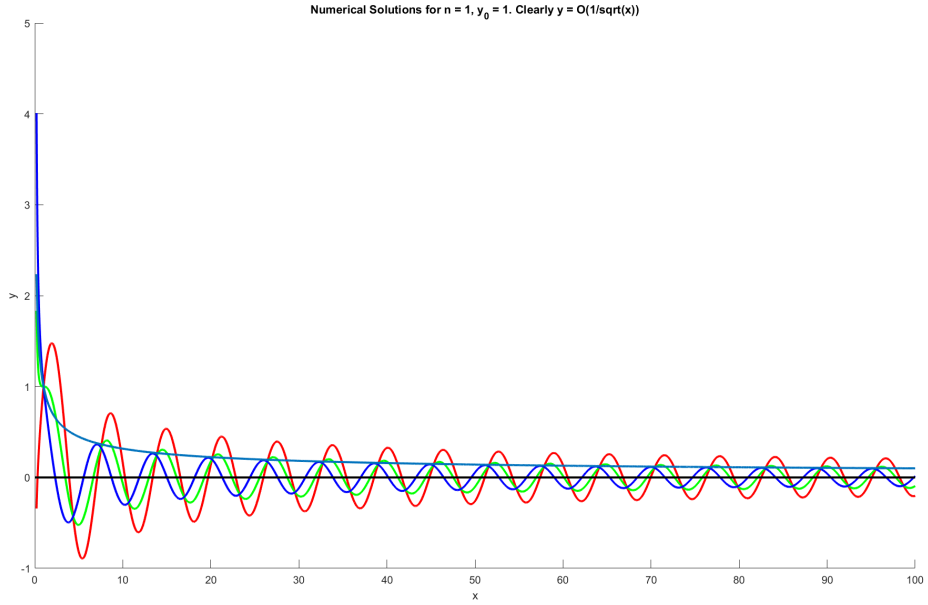


Figure 2: Asymptotically amplitudes decay like $1/\sqrt{x}$.

$\lambda_1, \lambda_2, \lambda_3, \ldots$. Since f and g were arbitrary, all solutions passing through (x_0, y_0) intersect at $\lambda_1, \lambda_2, \lambda_3, \ldots$

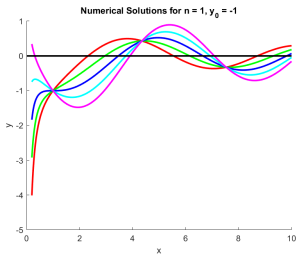
4. As $n \to \infty$, $\lambda_2 \to \infty$, initial amplitudes $\to \infty$ and $y \to \infty$ much faster as $x \to \infty$. At $x_0 = 0$, Bessel's equation gives $n^2 y_0 = 0$, so we have two cases to consider:

   $\boxed{\text{n} \neq 0\text{:}}$

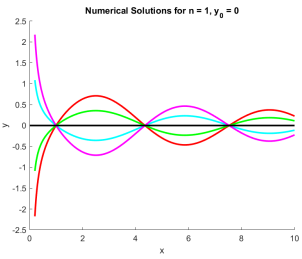
$$u_{k+1} = u_k + h v_k,$$
$$v_{k+1} = v_k + h f(x_k, u_k, v_k),$$

where

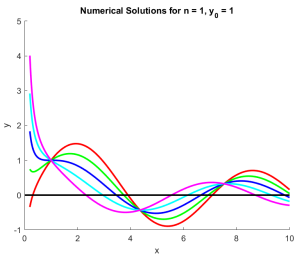
$$f(x, u, v) = \frac{(n^2 - x^2)u - xv}{x^2}.$$



(a) $y_0 = -1, n = 1$



(b) $y_0 = 0, n = 1$



(c) $y_0 = 1, n = 1$

Figure 3: All points passing through (x_0, y_0) intersect at the same points

So for $u_0 = 0, v_0 = y_0'$, we have

$$u_1 = hy_0$$
$$v_1 = y_0' + h \lim_{x\to 0} f(x, y, y_0').$$

Note that $\lim_{x\to 0} f(x, 0, y_0') = \lim_{x\to 0} -\frac{y_0'}{x} \to -\infty$. So $v_1 = y_1' = -\infty$. Hence, there is no solution.

$\boxed{\text{n = 0:}}$ Now y_0 is not necessarily 0. Therefore

$$f(x, y_0, y_0') = -(y_0 + \frac{y_0'}{x}) \to -\infty \text{ as } x \to 0, \text{ so there is no solution.}$$

Question 2

Let $E_N(x)$ denote the error at x for truncating at N terms.

$$E_N(x) = \left| \sum_{r=N+1}^{\infty} \frac{(-1)^r (\frac{1}{2}x)^{2r+n}}{r!(n+r)!} \right| = \left| \sum_{r=N+1}^{\infty} (-1)^r a_n \right|.$$

This will be an alternating series with coefficients decreasing in magnitude for sufficiently large N. When does this begin? Observe that

$$\frac{|a_{N+2}|}{|a_{N+1}|} = \frac{(\frac{1}{2}x)^{2(N+2)+n}}{(\frac{1}{2}x)^{2(N+1)+n}} \cdot \frac{(N+1)!(n+N+1)!}{(N+2)!(n+N+2)!}$$
$$= \frac{(x/2)^2}{(N+2)(N+n+2)},$$

$$\sup_{x\in[0,X]} \left| \frac{a_{n+2}}{a_{n+1}} \right| \leq \frac{(X/2)^2}{(N+2)^2} < 1.$$

We thus require that $X/2 < N + 2$ in order for the series $\sum_{n\geq N+1}(-1)^n a_n$ to be an alternating series with decreasing magnitudes a_n. Therefore, by the alternating series bound we have

$$\left| \sum_{n\geq N+1} (-1)^n a_n \right| \leq a_{N+1}.$$

So to determine N we will use the following process:

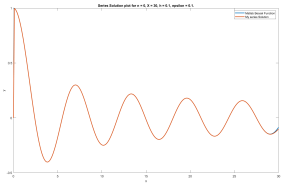
```
N = X/floor(2) - 2
while a_N > e : # e is some specified error threshold
    N = N+1
end
```

This terminates since $a_n \to 0$. For this N we therefore have

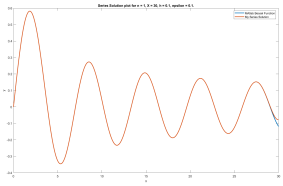
$$\sup_{x\in[0,X]} \left| J_n(x) - \sum_{r\leq N} \frac{(-1)^r (\frac{1}{2}x)^{2r+n}}{r!(n+r)!} \right| \leq \epsilon$$

So we define a function

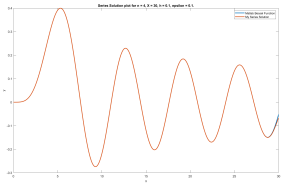
```
function [X,J] = series_soln(n,rangle,h,eps)
```



(a) Series Solution for $n = 0$.



(b) Series Solution for $n = 1$.



(c) Series Solution for $n = 0$.

Figure 4: Plots of Series Solutions for $n = 0, 1, 4$.

where n is the order of $J_n(x)$, range $= X$, h is the stepsize and eps $= \epsilon$ is the error threshold. In figure (4) we have plotted the cases we have plotted the cases of $n = 0, 1, 4$, range $= 30$, $h = 0.1$, $\epsilon = 0.1$.

Our function doesn't work for range ≥ 40. We plot this case in (5) below, but I'm not sure exactly why it happens here. I'm sure that my mathematics is correct so I believe that this is due to limitations of computing power but I do not see why. The order of magnitude for terms $x = 45$ is less than 10^{20}, and even when we work to 100 decimal digits we get $J_1(45) \cong -48$.

To find the order of the terms we're summing, note that for $x = 2\lambda$, we have

$$\frac{a_{k+1}(x)}{a_k(x)} = \frac{\lambda^2}{k(k+n+1)}.$$

At the maximum index, m, this becomes less than 1.

$$\lambda^2 = k(k+n+1) \implies k + \frac{n+1}{2} \approx \lambda \text{ (for small n)}.$$

So for $n = 1$, $k = \lambda - 1$ gives an approximate index at which $a_k(x)$ is maximised. To find the value of this maximum, note that

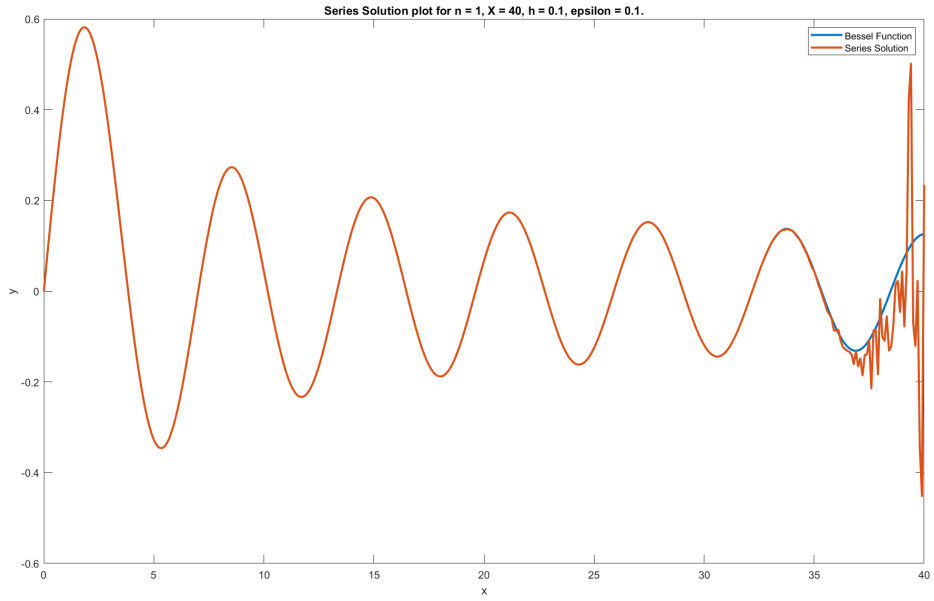


Figure 5: Our method fails for $X \geq 40$

$$
\begin{aligned}
a_{\lambda-1}(x) &= \frac{(\lambda)^{2(\lambda-1)+1}}{(\lambda-1)!(\lambda)!} \\
&= \frac{\lambda^{2\lambda}}{(\lambda!)^2} \\
\text{Stirling} \to &\approx \frac{\lambda^{2\lambda-1}}{2\pi\lambda(\lambda/e)^{2\lambda}} \\
&= \frac{e^{2\lambda}}{2\pi\lambda^2} \\
&= \frac{2e^x}{\pi x^2}.
\end{aligned}
$$

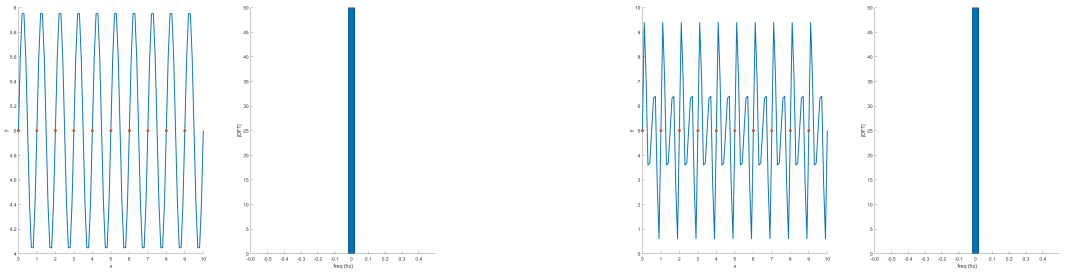
So for $x = 50$, the maximum terms are of order 10^{18}. I am not sure why my program struggles with this sum.

Question 3

For $\hat{\mathcal{F}} \to F$ we will definitely need $X \to \infty$ and $N \to \infty$, but these conditions alone are not sufficient. suppose we have the functions

$$
\begin{aligned}
F(x) &= 5 + \sin 2\pi x \\
G(x) &= 5 + \sin 2\pi x + 3\sin 4\pi x + \sin 6\pi x,
\end{aligned}
$$

and we sample these on the intervals $[0, N]$ at the points $\{0, 1, \ldots, N\}$. Then as $N \to \infty$, both X and $N \to \infty$, yet the DFT is unable to distinguish F and G, so cannot in any sense converge to the Fourier transform.



(a) DFT of F

(b) DFT of G

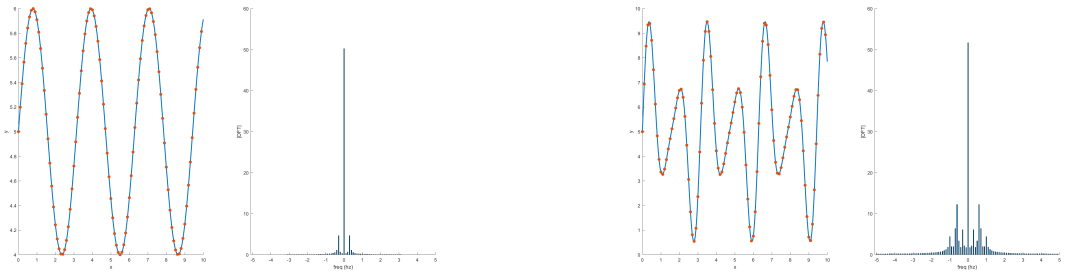
Figure 6: No matter how large X and N are, if Δx is too large, our DFT will fail to distinguish functions which differ in high frequencies.

Thus we need:

- $X \to \infty$, (range tends to ∞)
- $N \to \infty$, (num samples tends to ∞)
- $\limsup \frac{X}{N} \leq \frac{1}{2k_{\max}}$ (sampling frequencies fast enough to resolve all frequencies).

Changing Δx in this way, our DFT distinguishes F and G.
Rigorously, we let $[a, b]$ be an interval such that

$$
\int_{\mathbb{R}\setminus[a,b]} |F(x)|\, dx < \epsilon/2.
$$



(a) DFT of F (b) DFT of G

Figure 7: DFT with $\Delta x = 0.1$ distinguishes between F and G with ease.

By translating we may, without loss of generality, assume that $[a, b] = [0, X]$, and then

$$\begin{aligned}\hat{\mathcal{F}}_s &= \sum_{r=0}^{N-1} F_r\, \omega_N^{-rs} \Delta x \\ &= \sum_{r=0}^{N-1} F(r\Delta x) \exp\left(-2\pi i r \Delta x s / X\right) \Delta x \quad \xrightarrow[N\to\infty]{} \quad \int_0^x F(x) \exp(-2\pi i s x / X)\, dx\end{aligned}$$

So for N sufficiently large, $|\hat{\mathcal{F}}_s - \int_0^X F(x) \exp(-2\pi i s x/X)\, dx| < \epsilon/2$. Therefore

$$\begin{aligned}\left|\hat{\mathcal{F}}_s - \hat{F}(s/X)\right| &\le |\hat{\mathcal{F}}_s - \int_0^X F(x) \exp(-2\pi i s x/X)\, dx| + \left|\int_{\mathbb{R}\setminus[0,X]} F(x) \exp(-2\pi i s x/X)\, dx\right| \\ &\le \left|\hat{\mathcal{F}}_s - \int_0^X F(x) \exp(-2\pi i s x/X)\, dx\right| + \int_{\mathbb{R}\setminus[0,X]} |F(x)|\, dx \\ &< \epsilon.\end{aligned}$$

So if we let $k = s/X$, then $\hat{\mathcal{F}_{kX}} \quad \xrightarrow[X,N\to\infty,\Delta x \text{ small}]{} \quad \hat{F(k)}$.

Question 4

$$I_1 = \int_0^X F(x)\cos(2\pi k x)\, dx - i \int_0^X F(x)\sin(2\pi k x)\, dx.$$

Since $F(x)$, $\cos(x)$ and $\sin(x)$ are real on $[0, X]$ we have

- $\mathrm{Re}(I_1) = \int_0^X F(x)\cos(2\pi k x)\, dx$
- $\mathrm{Im}(I_1) = -\int_0^X F(x)\sin(2\pi k x)\, dx$.

For I_2, we have

$$\begin{aligned}I_2 &= \int_{-X}^X F(x)\cos(2\pi k x)\, dx - i \int_{-X}^X F(x)\sin(2\pi k x)\, dx \\ &= 2\int_0^X F(x)\cos(2\pi k x)\, dx + 0,\end{aligned}$$

since $F(x)\cos(2\pi k x)$ is even and $F(x)\sin(2\pi k x)$ is odd. Therefore:

- $\mathrm{Re}(I_2) = 2\int_0^X F(x)\cos(2\pi k x)\, dx = 2\mathrm{Re}(I_1)$
- $\mathrm{Im}(I_2) = 0$.

Next, suppose that we desire an approximation to I_2 and $F_N \neq F_0$. The procedure above yields

$$\begin{aligned} I_2 &\approx 2\mathrm{Re}\left(\sum_{r=0}^{N-1} F_r \omega_N^{-rs} \Delta x\right) \\ &= \sum_{r=0}^{N-1} F_r(\omega_N^{-rs} + \omega_N^{rs})\Delta x \quad + \sum_{r=1-N}^{0} F_r \omega_N^{-rs} \Delta x. \end{aligned}$$

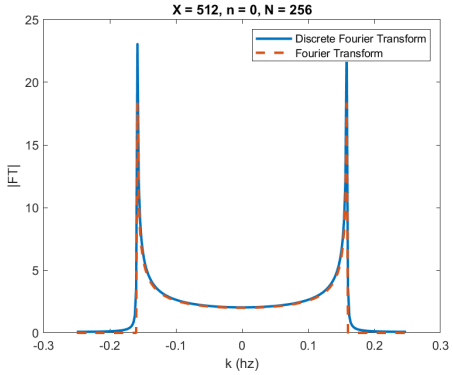
If $F_N \neq F_0$ this doesn't give the rectangular approximation to the Riemann integral for I_2, as the F_0 rectangle is counted twice and F_N is not counted at all. So if we set $F_0 = \frac{1}{2}(F_0 + F_N)$ we get

$$I_2 = 2\mathrm{Re}(I_1) \approx \sum_{r=-N}^{N-1} F_r \omega_N^{-rs} \Delta x,$$

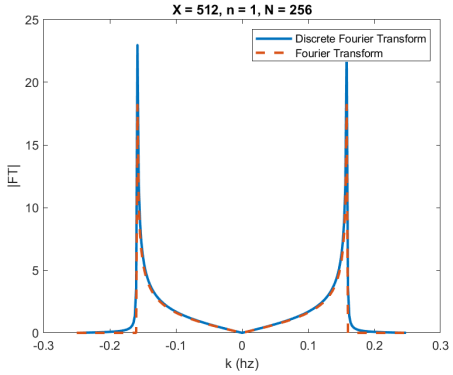
as desired. If F is real and odd then

- $\mathrm{Re}(I_2) = 0$
- $\mathrm{Im}(I_2) = 2\mathrm{Im}(I_1)$.

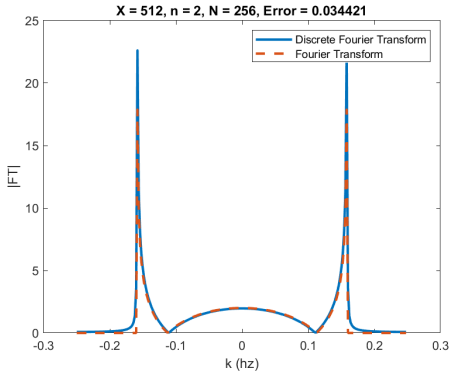
Question 5



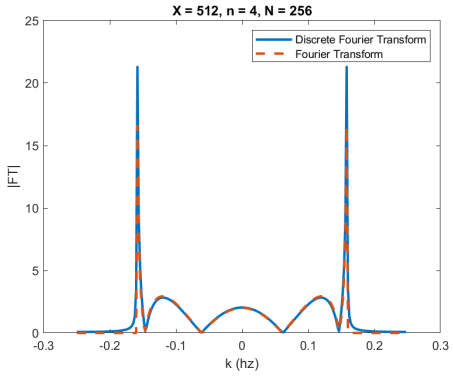
(a) $n = 0$



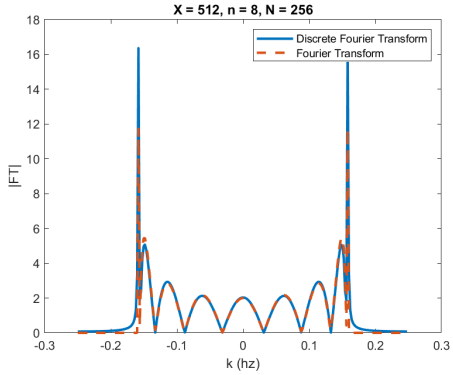
(b) $n = 1$



(c) $n = 2$



(d) $n = 4$



(e) $n = 8$

Figure 8: Plots of the DFT and the FT of J_n for $n = 0, 1, 2, 4, 8$. Here we have $X = 512$, $N = 256$, $\Delta x = 1/2$.

The code for question 5 is included in the appendix.
We have chosen X so large in figure (8) because this allows us to resolve frequencies very precisely ($\Delta k = 1/X$).

Discussion :

First of all, figures 9.1-9.5 show striking similarity between the FT and DFT. Both capture spikes at $\pm 1/2\pi$, which is consistent with the asymptotic form

$$J_n(x) \sim \sqrt{\frac{2}{\pi x}} \cos(x - \frac{n\pi}{2} - \frac{\pi}{4}).$$

At these points, our theoretical formula seems nearly singular, and our DFT replicates this behaviour well.

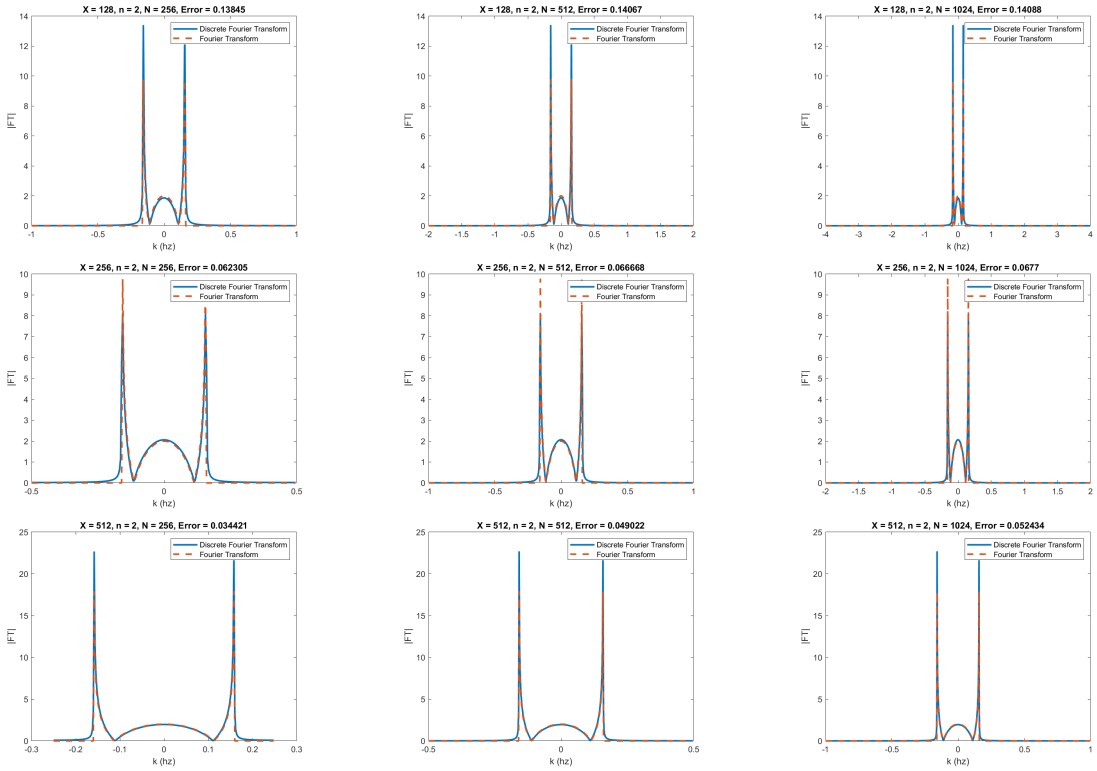


Figure 9: Variation of results for different values of N and X, all for $n = 2$

We calculate the error by evaluating the FT and DFT at 0 and looking at their difference. Surprisingly, these errors are mostly independent of N but do seem to be decreasing as x increases. For a systematic variation of X and N, see figure (9) below.

$\boldsymbol{N}$	4	8	16	32	64	128	256	512	1024	2048	4096
$\boldsymbol{t}$	19.305	32.254	53.754	74.167	86.018	69.724	92.643	86.136	88.236	106.837	163.061

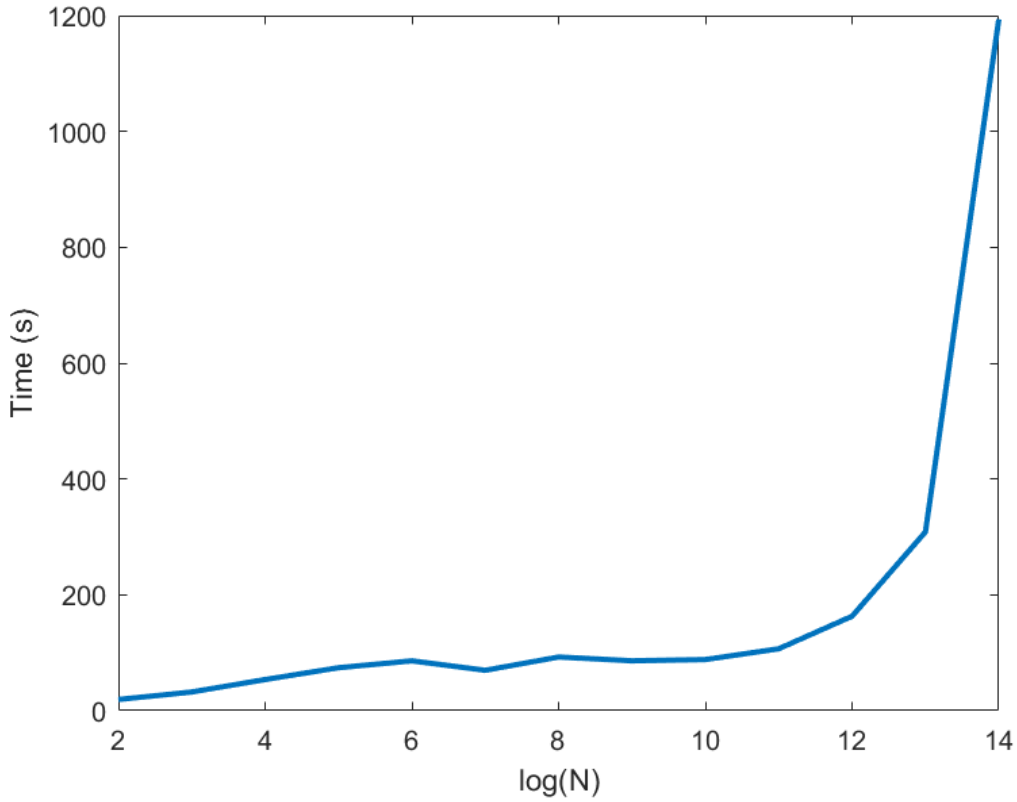


Figure 10: Complexity of program for different values of $\log(N)$.

Appendix

```
function [d] = bes(y,x,n)
% When we write Bessel's Equation down as a system of 1st order ODES we
% have: u = y, v = y'. We then have eqns:
% u' = v
% v' = ((n^2 - x^2)u -xv)/(x^2).
% In our scheme, y(1) = u, y(2) = v, x = x and order = n
% The vector d(x,y) = [u'(x), v'(x)] = [y'(x), y''(x)]
d = [y(2); ((n^2-x^2)*y(1)-x*y(2))/x^2];
end


function [] = solve(n,y0,LineSpec)
% This function implements the ode45 ode solver to get solutions of
% Bessel's Equations.
% n = order of Bessel Equation
% y0 = [u,v] where u = y(1), v = y'(1), for our numerical soln y
[t,y] = ode45(@(x,y) bes(y,x,n),1:0.01:100,y0);  %Commencing forward integration
[s,z] = ode45(@(x,z) bes(z,x,n),1:-0.01:0.2,y0);  %Commencing backward integration
p1 = plot(t,y(:,1),LineSpec);       % plotting forwards soln
p2 = plot (s,z(:,1),LineSpec);      % plotting backwards soln
%legend([p1],{"x_0 = 1, y_0 = " + y0(1) + ", (dy/dx)_0 = " + y0(2)})
```

```
function [] = diff_ics(c,n,save)
% This script will help us numerically solve Bessel's Equation of order n
% for 3 diff ICS, y(x0) = c, y'(x0) = {-1,0,1}
% save = 0 or 1, if save = 1 then save images

hold on     % This ensures all graphs displayed superimposed
solve(n,[c,1],'r')
solve(n,[c,0],'g')
solve(n,[c,-1],'b')
plot([0,100],[0,0],'k')  % Plot x axis in black
xlabel('x')
ylabel('y')
title("Numerical Solutions for n = " + n + ", y_0 = " + c)

if save == 1
    saveas(gcf,"testy"+c+"n"+n+".png")
    clf
end
end


% Little script to print out and save all 9 possible images of diff ICs
for c = [-1,-0.5,0,0.5,1]
    for n = [1]
        diff_ics(c,n)
    end
end



function [X,J] = series_soln(n,range,h,eps)
% This function truncates a sum of the series solution of Bessel's equation
% of order n. The number of terms is chosen to ensure that within
% [0,range], the error is bounded by the input eps. Here h represents the
% resolution we want to view our domain in

N = floor(range/2)-2;
a = (0.5*range)^(2*(N+1)+n)/(factorial(N+1)*factorial(N+n+1));

while a > eps
    N = N+1;
    a = ((0.5*range))^(2*(N+1)+n)/(factorial(N+1)*factorial(N+n+1));
end

% Next we compute the solution J
X = 0:h:range;  % these are the x vals at which we evaluate our series soln
J = 0.*X;       % initialising soln
k = 2;          % k will index the values of our solution J at X value X(k)

for i = h:h:range% i loops through all values in domain we wish to calculate

    for j = 0:N     % j adds each term in the series we wish to calculate
```

```
        taylor_term = (-1)^(j)*(i/2)^(2*j+n)/(factorial(j)*factorial(n+j));
        J(k) = J(k) + taylor_term;

    end

k = k+1;

end

plot(X,besselj(n,X),X,J)
xlabel('x')
ylabel('y')
title("Series Solution plot for n = "+n+", X = "+range+", h = "+h+", epsilon = "+eps+".")
N

end



function [Y] = DFT(X,F,N)
% We take in a function F defined over a set [0, R] with very small
% stepsize. We then choose how many samples we want to take from this
% interval, N. higher N corresponds to higher sampling freq.
% This is NOT the code used for Question 5, but rather code used to get an
% intuition for results and develop plots for some of the figures generated
% NOTE: N must divide length(X)-1

A = zeros(N,N); % A will be our DFT matrix
omega_N = exp(2*pi*1i/N);   % omega_N is the Nth root of unity
m = length(X);
step = (m-1)/N;
R = X(m);   % R is what the notes call X

for j = 0:N-1   % Now we define matrix A
    for k = 0:N-1
        A(j+1,k+1) = (R/N)*omega_N^(-j*k);
    end
end

% Next we take N evenly spaced samples from the vector F and compute its
% DFT Y.
U = X(1:step:m-step)';  % U is a column vector of our sampled x vals
V = F(1:step:m-step)';  % V is a column vector of our sampled y vals

Y = A*V;

L = floor(N/2);
B = zeros(N,N);
B(L+1:N,1:L) = eye(L);
B(1:L,L+1:N) = eye(L);

Y = B*Y;
```

```
% Next plot [0,1/X,2/X,...,(N-1)/X] and Y

freqs = (1/R).*(-L:L-1);

tiledlayout(1,2);

nexttile
hold on
plot(X,F);
scatter(U,V,40,'filled');
xlabel('x')
ylabel('y')

nexttile
hold on
bar(freqs,abs(Y),0.4);
xlabel('freq (hz)')
ylabel('|DFT|')

end



function [] = Q5_Code(X,n,N,save)
% X is the set of input values, will usually be some interval [0,R]
% n is the order of the Bessel Function
% N is the number of samples we take. Needs to be even.
% if save == 1, we save the image

m = length(X);  % This shorthand variable will come in handy later
F = besselj(n,X);   % Bessel func of first kind, order n, evaluated at X
R = X(m);   % R is what notes call X
% F = 3*sin(X) + 4*sin(2*X); % This is just here to test everything
x_step = floor(m/N);    % this is not delta_x, but rather the number of entries
% we have to skip when we are sampling F
V = F(1:x_step:m);   % V is our sampled signal
V(1) = 0.5*(V(1)+V(N+1)); % Always need to do this as interval sym abt 0
V = V(1:N);

% Next we will need to build the matrix to rearrange the DFT vector
B = zeros(N,N);
L = N/2;
B(L+1:N,1:L) = eye(L);
B(1:L,L+1:N) = eye(L);

ft_F = fft(V);    % FT of F, evaluated at (-N/2X),(-N/2X + delta_k), ...
ft_F = B*(ft_F)';   % This cyclically permutes the vector N/2 steps forward
if mod(n,2) == 1
    ft_F = 2*imag(ft_F);  % Since the domain of F is [-R,R], we have to do this
else
    ft_F = 2*real(ft_F);  % Since the domain of F is [-R,R], we have to do this
end
```

```
ft_F = (R/N).*ft_F;

delta_k = 1/R;  % Step size in freq domain
freqs = -(N/(2*R)):delta_k:(N-1)/(2*R); % Frequency domain

% Next, we define T, the theoretical formula for the FT of J_n
T = 0.*freqs';

i = 1;   %This will be our counter
for k = freqs
    if abs(k) >= 1/(2*pi)
        T(i) = 0;
    else
    T(i) = ((2*(-1i)^n)*(1-(2*pi*k)^2)^(-0.5))*chebyshevT(n,2*pi*k);
    end
    i = i+1;
end

plot(freqs,abs(ft_F),freqs,abs(T),'--');
xlabel('k (hz)')
ylabel('|FT|')
legend('Discrete Fourier Transform','Fourier Transform');
er_vec = abs(abs(T)-abs(ft_F));
error = er_vec(N/2 + 1);   % This is Error at 0
title("X = "+ R +", n = "+n+", N = "+N+", Error = "+error);

if save == 1
    saveas(gcf,"X"+R+"n"+n+"N"+N+".png")
    clf
end

end



% Little script to print out and save all 5 possible images of DFTS
for n = [0,1,2,4,8]
    Q5_Code(X,n,N,1)
end


% Little script to print out and save all possible variations of fig 10
for N = [128,256,512,1024]
    for R = [64,128,256.512]
        Q5_Code(0:1/16:R,2,N,1)
    end
end
```