1 Numerical Methods

1.1 Fourier Transforms of Bessel Functions (6 units)

This project assumes only material contained in Part IA and Part IB core courses. Other than that, the project is self contained (although the Part II courses on Numerical Analysis, Further Complex Methods and Asymptotic Methods may provide relevant but non-essential background).

1 Introduction

Bessel’s equation of order \( n \) is the linear second-order equation

\[
x^2 y'' + xy' + (x^2 - n^2)y = 0.
\]

(1)

Bessel functions of the first kind are solutions of (1) which are finite at \( x = 0 \). They are usually written \( J_n(x) \).

**Question 1** Investigate (1) for \( n = 0, 1, 4 \) using a Runge–Kutta (or similar) method commencing the integration for a positive value of \( x \) and arbitrary values of \( y \) and \( y' \). You may use a library routine to solve (1), for example if using MATLAB you can use the built-in `ode45` routine. Integrate forwards and backwards in \( x \) for a few such initial conditions, plotting \( y \). Describe what you observe, and illustrate interesting behaviour by five or so plots in your write-up.

Now try starting at \( x = 0 \). What happens, and why?

**Question 2** The series solution for \( J_n(x) \) is

\[
J_n(x) = \sum_{r=0}^{\infty} \frac{(-1)^r (1/2)^{2r+n}}{r!(n+r)!}.
\]

(2)

Write a program to sum this series, and plot \( J_n(x) \) for \( n = 0, 1, 4 \) for a range of \( x \), e.g., for \( 0 \leq x \leq 100 \). Identify a range of \( x \) for which this summation method is not accurate and explain why.

2 The Discrete Fourier Transform

The Fourier Transform \( \hat{F}(k) \) of a function \( F(x) \) may be defined as

\[
\hat{F}(k) = \int_{-\infty}^{+\infty} F(x) \exp(-2\pi ikx) \, dx.
\]

(3)

If \( F(x) \) is a function which is only appreciably non-zero over a limited range of \( x \), say \( 0 < x < X \), then it is possible to approximate \( \hat{F}(k) \) by means of finite sums. Suppose

\[
F_r = F(r\Delta x) \quad \text{for} \quad r = 0, \ldots, (N-1) \, , \quad \text{where} \quad \Delta x = X/N .
\]

(4)

An approximation to (3), known as the Discrete Fourier Transform (DFT), is

\[
\hat{F}_s = \frac{X}{N} \sum_{r=0}^{N-1} F_r \omega_N^{-rs} \, , \quad \text{where} \quad \omega_N = e^{2\pi i/N} .
\]

(5)
The exact inverse of (5) is
\[ F_r = \frac{1}{X} \sum_{s=0}^{N-1} \hat{F}_s \omega_N^{rs}. \] (6)

In order to deduce the relationship between the \( \hat{F}_s \) and \( \hat{F}(k) \), we first note from (5) that \( \hat{F}_s \) represents values of the Fourier Transform spaced by the “wavenumber” interval \( \Delta k = 1/X \).

Also \( \hat{F}_s \) is periodic in \( s \) with period \( N \); this corresponds to a “wavenumber” periodicity
\[ K = N \Delta k = N/X = 1/\Delta x. \] (7)

Now it is to be expected that (5) will fail to approximate to (3) when the exponential function oscillates significantly between sample points, that is when
\[ |k| \gtrsim \frac{1}{2\Delta x} = \frac{1}{2}K. \] (9)

This, together with its periodicity, suggests that \( \hat{F}_s \) will be related to \( \hat{F}(k) \) by
\[ \hat{F}_s \approx \begin{cases} \hat{F}(s\Delta k) & s = 0, \ldots, \frac{1}{2}N - 1, \\ \hat{F}(s\Delta k - K) & s = \frac{1}{2}N, \ldots, N - 1. \end{cases} \] (10)

Thus (6) is an approximation to
\[ F(x) \approx \int_{-K/2}^{+K/2} \hat{F}(k) \exp(2\pi ikx) \, dk. \] (11)

Because of the periodicity, the \( \hat{F}_s \) are usually thought of as a series with \( s = 0, \ldots, N - 1 \), the upper half being mentally re-positioned to correspond to negative “wavenumber”. Note that if \( F(x) \) is real, and * denotes a complex conjugate, then
\[ \hat{F}(k) = \hat{F}^*(-k). \] (12)

**Question 3** Carefully discuss under what limiting processes for \( N \) and \( X \) (possibly after a suitable change in origin in \( x \)), does the DFT tend to the Fourier Transform?

### 3 The Fast Fourier Transform

The DFT is best evaluated by the Fast Fourier Transform (FFT) method. This is a quick method of evaluating sums of the form
\[ \lambda_s = \sum_{r=0}^{N-1} \mu_r \omega_N^{rs}, \quad s = 0, \ldots, N - 1, \quad \sigma = \pm 1, \] (13)

where \( N \) is an integer and the \( \mu_r \) are a known sequence. The “fast” in FFT depends on \( N \) being a power of a small prime, or combination of small primes; for simplicity we will assume that \( N = 2^M \).

A brief outline of the FFT method is given in the Appendix. However, it is not necessary to understand the implementation details, since you may use the MATLAB one-dimensional Fast Fourier Transform function \texttt{fft} (which has inverse \texttt{ifft}), or an equivalent routine in any other package. Alternatively you may write your own routine; however do **not** simply compute the series (5) simplistically.

Note that MATLAB’s \texttt{fft} function will work for any value of \( N \) although it works best when \( N \) is a power of 2. Further details can be found on the \texttt{fft} MATLAB help page.
4 Fourier Transforms of Bessel Functions

**Question 4** Show analytically that if $F(x)$ is a real even function and

$$I_1 = \int_0^X F(x) \exp(-2\pi ikx) \, dx, \quad I_2 = \int_{-X}^X F(x) \exp(-2\pi ikx) \, dx,$$  

then

$$\text{Im}(I_2) = 0, \quad \text{Re}(I_2) = 2\text{Re}(I_1).$$  

With the definitions of §2 and §3, the FFT algorithm is ideally suited to approximating $I_1$ rather than $I_2$. Hence if an approximation to $I_2$ is desired, an approximation to $I_1$ could first be calculated, and then the relations (14b) could be used. If this procedure for calculating $I_2$ is adopted, and $F_N \neq F_0$, explain why $F_0$ should be replaced by $\frac{1}{2}(F_0 + F_N)$ before calculating the DFT. What is the equivalent result to (14b) if $F(x)$ is a real odd function?

**Question 5** Using an FFT, and the results of question 4, find numerically the Fourier Transform of $J_n(x)$:

$$\hat{J}_n(k) = \int_{-\infty}^{+\infty} J_n(x) \exp(-2\pi ikx) \, dx.$$  

Compare it with the theoretical formula

$$\hat{J}_n(k) = 2(-i)^n(1 - 4\pi^2k^2)^{-1/2} T_n(2\pi k),$$  

where $T_n(\mu)$ is the Chebyshev polynomial of order $n$ defined by

$$T_n(\mu) = \begin{cases} \cos n\theta, & \mu = \cos \theta; \\ 0, & |\mu| > 1. \end{cases}$$

To obtain $J_n(x)$, you may either devise a method of your own (e.g., a combination of questions 1 and 2), or you may use the MATLAB procedure `besselj`.

You should obtain results for $n = 0, 1, 2, 4,$ and $8$. Choose sufficient points in the transform to resolve the functions to your satisfaction subject to reasonable time constraints on whatever computer you are using.

Plots of $J_n(x)$ for a couple of representative values of $n$ should be included in your write-up. You should also include plots of $\hat{J}_n$ and $\hat{J}_n$ on the same graph. Choose a range of $k$ which allows you to see the detailed behaviour in the interval $-1 \leqslant \pi k \leqslant 1$.

Comment on your results. *Inter alia* you should remark on how the FFT deals with any values of $k$ which might be expected from the theoretical result to give problems, and you should describe the effects of varying $N$ and $X$; in particular you should *systematically* examine how the numerical errors change as $N$ and/or $X$ are varied, e.g. in the light of your answer to question 3.

You should also find a way to demonstrate from your computational results how the execution time necessary to calculate the transform varies with $N$. 

---

July 2017/Part II/1.1 Page 3 of 4 © University of Cambridge
Appendix: The Fast Fourier Transform

The Fast Fourier Transform (FFT) technique is a quick method of evaluating sums of the form

$$\lambda_r = \sum_{s=0}^{N-1} \mu_s \omega_N^{rs}, \quad r = 0, \ldots, N - 1, \quad \sigma = \pm 1,$$

(18)

where \( N \) is an integer, \( \mu_s \) is a known sequence and \( \omega_N = e^{2\pi i/N} \). The “fast” in FFT depends on \( N \) being a power of a small prime, or combination of small primes; for simplicity we will assume that \( N = 2^M \). Write

$$\lambda_r \leftrightarrow \mu_s, \quad r, s = 0, \ldots, N - 1$$

(19)

to denote that (18) is satisfied. Introduce the half-length transforms

$$\lambda_r \leftrightarrow \mu_{2s}$$
$$\lambda_r \leftrightarrow \mu_{2s+1}$$

(20)

then it may be shown that

$$\lambda_r = \lambda_{r}^{E} + \omega_N^{2r} \lambda_{r}^{O}$$
$$\lambda_{r+N/2} = \lambda_{r}^{E} - \omega_N^{2r} \lambda_{r}^{O}$$

(21)

Hence if the half-length transforms are known, it costs \( \frac{1}{2} N \) products to evaluate the \( \lambda_r \).

To execute an FFT, start from \( N \) vectors of unit length (i.e., the original \( \mu_s \)). At the \( s \)th stage, \( s = 1, 2, \ldots, M \), assemble \( 2^{M-s} \) vectors of length \( 2^s \) from vectors of length \( 2^{s-1} \) – this “costs” \( 2^{M-s} \times \frac{1}{2}(2^s) = 2^{M-1} = \frac{1}{2} N \) products for each stage. The complete discrete Fourier transform has been formed after \( M \) stages, i.e., after \( O(\frac{1}{2} N \log_2 N) \) products. For \( N = 1024 = 2^{10} \), say, the cost is \( \approx 5 \times 10^3 \) products, compared to \( \approx 10^6 \) products in naive matrix multiplication.

A description and short history of the FFT are given in Chapter 12 of the book *Numerical Recipes* by Press et al.