Math/CS 467/667 Programming Project 2

Approximation of Derivatives by FFT

1. Let $f : \mathbf{R} \to \mathbf{R}$ be a differentiable function with period 2. Approximate f on the interval $[-1, 1]$ as $f \approx A$ where

$$A(x) = \sum_{j=-N/2+1}^{N/2} y_j e^{i\pi jx} \quad \text{and} \quad y_j = \frac{1}{N} \sum_{\ell=-N/2+1}^{N/2} f\Big(\frac{2\ell}{N}\Big) e^{-2\pi i \ell j/N}.$$

Differentiate A to obtain approximations for f', f'' and f'''.

Differentiating yields

$$A'(x) = i\pi \sum_{j=-N/2+1}^{N/2} j y_j e^{i\pi jx},$$

$$A''(x) = -\pi^2 \sum_{j=-N/2+1}^{N/2} j^2 y_j e^{i\pi jx}$$

and

$$A'''(x) = -i\pi^3 \sum_{j=-N/2+1}^{N/2} j^3 y_j e^{i\pi jx}.$$

2. Modify the approximation in part 1 by setting $y_{N/2} = 0$ to obtain $\widetilde{A}$. Explain why $\widetilde{A}$ is guaranteed to be real for all values of x.

For notational convenience denote

$$B'(x) = \widetilde{A}'(x), \qquad B''(x) = \widetilde{A}''(x) \qquad \text{and} \qquad B'''(x) = \widetilde{A}'''(x).$$

Since $y_{N/2} = 0$ the resulting functions are simply the sums that end at $N/2 - 1$. Thus

$$B(x) = \sum_{j=-N/2+1}^{N/2-1} y_j e^{i\pi jx},$$

$$B'(x) = i\pi \sum_{j=-N/2+1}^{N/2-1} j y_j e^{i\pi jx},$$

$$B''(x) = -\pi^2 \sum_{j=-N/2+1}^{N/2-1} j^2 y_j e^{i\pi jx}$$

and

$$B'''(x) = -i\pi^3 \sum_{j=-N/2+1}^{N/2-1} j^3 y_j e^{i\pi jx}.$$

Real means that

$$B'(x) = \overline{B'(x)}, \qquad B''(x) = \overline{B''(x)} \qquad \text{and} \qquad B'''(x) = \overline{B'''(x)}$$

where the overlining indicates complex conjugation. Since f is real, then by definition

$$\begin{aligned}\overline{y_j} &= \frac{1}{N} \sum_{\ell=-N/2+1}^{N/2} \overline{f\Big(\frac{2\ell}{N}\Big)}\,\overline{e^{-2\pi i\ell j/N}} = \frac{1}{N} \sum_{\ell=-N/2+1}^{N/2} f\Big(\frac{2\ell}{N}\Big) e^{2\pi i\ell j/N} \\ &= \frac{1}{N} \sum_{\ell=-N/2+1}^{N/2} f\Big(\frac{2\ell}{N}\Big) e^{-2\pi i\ell(-j)/N} = y_{-j}.\end{aligned}$$

Consequently, setting $k = -j$ in the sum yields

$$\begin{aligned}\overline{B'(x)} &= \overline{i\pi} \sum_{j=-N/2+1}^{N/2-1} j\overline{y_j}\,\overline{e^{i\pi jx}} = -i\pi \sum_{j=-N/2+1}^{N/2-1} j y_{-j} e^{-i\pi jx} \\ &= i\pi \sum_{j=-N/2+1}^{N/2-1} (-j) y_{-j} e^{i\pi(-j)x} = i\pi \sum_{k=-N/2+1}^{N/2-1} k y_k e^{i\pi kx} = B'(x)\end{aligned}$$

so $B'(x)$ is real for any value of x. Similarly

$$\begin{aligned}\overline{B''(x)} &= -\pi^2 \sum_{j=-N/2+1}^{N/2-1} j^2\overline{y_j}\,\overline{e^{i\pi jx}} = -\pi^2 \sum_{j=-N/2+1}^{N/2-1} j^2 y_{-j} e^{-i\pi jx} \\ &= -\pi^2 \sum_{j=-N/2+1}^{N/2-1} (-j)^2 y_{-j} e^{i\pi(-j)x} = -\pi^2 \sum_{k=-N/2+1}^{N/2-1} k^2 y_k e^{i\pi kx} = B''(x)\end{aligned}$$

and

$$\begin{aligned}\overline{B'''(x)} &= \overline{-i\pi^3} \sum_{j=-N/2+1}^{N/2-1} j^3\overline{y_j}\,\overline{e^{i\pi jx}} = i\pi^3 \sum_{j=-N/2+1}^{N/2-1} j^3 y_{-j} e^{-i\pi jx} \\ &= -i\pi^3 \sum_{j=-N/2+1}^{N/2-1} (-j)^3 y_{-j} e^{i\pi(-j)x} = -\pi^3 \sum_{k=-N/2+1}^{N/2-1} k^3 y_k e^{i\pi kx} = B'''(x)\end{aligned}$$

show that $B''(x)$ and $B'''(x)$ are real valued.

3. Let $f(x) = \exp(\sin \pi x)$. Compute f', f'' and f''' exactly.

The Maple script

```
restart;
with(codegen):
f:=x->exp(sin(Pi*x));
df:=unapply(simplify(diff(f(x),x)),x);
ddf:=unapply(simplify(diff(df(x),x)),x);
dddf:=unapply(simplify(diff(ddf(x),x)),x);
C(f,optimized,filename="f.i");
C(df,optimized,filename="f.i");
C(ddf,optimized,filename="f.i");
C(dddf,optimized,filename="f.i");
```

produces the output

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> with(codegen):
Warning, the protected name MathML has been redefined and unprotected
> f:=x->exp(sin(Pi*x));
                            f := x -> exp(sin(Pi x))

> df:=unapply(simplify(diff(f(x),x)),x);
                     df := x -> cos(Pi x) Pi exp(sin(Pi x))

> ddf:=unapply(simplify(diff(df(x),x)),x);
                           2                                       2
            ddf := x -> -Pi  exp(sin(Pi x)) (sin(Pi x) - cos(Pi x) )

> dddf:=unapply(simplify(diff(ddf(x),x)),x);
                   3                                                  2
   dddf := x -> -Pi  exp(sin(Pi x)) cos(Pi x) (3 sin(Pi x) - cos(Pi x)  + 1)

> C(f,optimized,filename="f.i");
> C(df,optimized,filename="f.i");
> C(ddf,optimized,filename="f.i");
> C(dddf,optimized,filename="f.i");
> quit
bytes used=2782620, alloc=2358864, time=0.06
```

which shows that

$$f'(x) = \pi(\cos \pi x)\exp(\sin \pi x)$$
$$f''(x) = \pi^2(\cos^2 \pi x - \sin \pi x)\exp(\sin \pi x)$$

and

$$f'''(x) = \pi^3(\cos \pi x)(\cos^2 \pi x - 3 \sin \pi x - 1) \exp(\sin \pi x)$$

Automatically generated C code for the above derivatives is

```c
/* The options were    : operatorarrow */
#include <math.h>
double f(x)
double x;
{
  double t2;
  {
    t2 = sin(0.3141592653589793E1*x);
    return(exp(t2));
  }
}

/* The options were    : operatorarrow */
#include <math.h>
double df(x)
double x;
{
  double t1;
  double t2;
  double t4;
  double t5;
  {
    t1 = 0.3141592653589793E1*x;
    t2 = cos(t1);
    t4 = sin(t1);
    t5 = exp(t4);
    return(t2*0.3141592653589793E1*t5);
  }
}

/* The options were    : operatorarrow */
#include <math.h>
double ddf(x)
double x;
{
  double t1;
  double t2;
  double t3;
  double t4;
  double t6;
  double t7;
```

```
  {
    t1 = 0.3141592653589793E1*0.3141592653589793E1;
    t2 = 0.3141592653589793E1*x;
    t3 = sin(t2);
    t4 = exp(t3);
    t6 = cos(t2);
    t7 = t6*t6;
    return(-t1*t4*(t3-t7));
  }
}

/* The options were    : operatorarrow */
#include <math.h>
double dddf(x)
double x;
{
  double t1;
  double t3;
  double t4;
  double t5;
  double t7;
  double t9;
  {
    t1 = 0.3141592653589793E1*0.3141592653589793E1;
    t3 = 0.3141592653589793E1*x;
    t4 = sin(t3);
    t5 = exp(t4);
    t7 = cos(t3);
    t9 = t7*t7;
    return(-t1*0.3141592653589793E1*t5*t7*(3.0*t4-t9+1.0));
  }
}

```

4. Define

$$A'_\ell = A'\Big(\frac{2\ell}{N}\Big), \qquad A''_\ell = A''\Big(\frac{2\ell}{N}\Big), \qquad A'''_\ell = A'''\Big(\frac{2\ell}{N}\Big),$$
$$\widetilde{A}'_\ell = \widetilde{A}'\Big(\frac{2\ell}{N}\Big), \qquad \widetilde{A}''_\ell = \widetilde{A}''\Big(\frac{2\ell}{N}\Big), \qquad \widetilde{A}'''_\ell = \widetilde{A}'''\Big(\frac{2\ell}{N}\Big).$$

Write a program that uses the the FFT and inverse FFT to compute these approximations for $N = 4, 8, 16$. Display your results in a table form. Are the imaginary parts of A'_ℓ, A''_ℓ and A'''_ℓ zero? How about the imaginary parts of $\widetilde{A}'_\ell$, $\widetilde{A}''_\ell$ and $\widetilde{A}'''_\ell$? Which are better approximations? What role does rounding error play?

The program

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <complex.h>

#include "fft.h"

extern double f(double x);
extern double df(double x);
extern double ddf(double x);
extern double dddf(double x);

#include "f.i"

int main(){
    for(int N=4;N<=16;N=N*2){
        complex F[N],Y[N];
        for(int l=-N/2+1;l<=N/2;l++){
            int n; if(l<0) n=l+N; else n=l;
            F[n]=f(2.0*l/N);
        }
        fft(N,F,Y);
        complex dA[N],ddA[N],dddA[N];
        complex dB[N],ddB[N],dddB[N];  //  B = tilde A
        complex T[N];

        for(int j=-N/2+1;j<=N/2;j++){
            int n; if(j<0) n=j+N; else n=j;
            T[n]=I*M_PI*j*Y[n];
        }
        fift(N,T,dA);
        T[N/2]=0;
        fift(N,T,dB);

        for(int j=-N/2+1;j<=N/2;j++){
            int n; if(j<0) n=j+N; else n=j;
            T[n]=-M_PI*M_PI*j*j*Y[n];
        }
        fift(N,T,ddA);
        T[N/2]=0;
        fift(N,T,ddB);

        for(int j=-N/2+1;j<=N/2;j++){
            int n; if(j<0) n=j+N; else n=j;
```

```
            T[n]=-I*M_PI*M_PI*M_PI*j*j*j*Y[n];
        }
        fift(N,T,dddA);
        T[N/2]=0;
        fift(N,T,dddB);

        printf("%s#Values of dA, ddA and dddA for N=%d\n",
            N>4?"\n":"",N);
        printf("#%2s  %11s %11s  %11s %11s  %11s %11s\n",
            "l","real(dA)","imag(dA)","real(ddA)","imag(ddA)",
            "real(dddA)","imag(dddA)");
        for(int l=-N/2+1;l<=N/2;l++){
            int n; if(l<0) n=l+N; else n=l;
            printf("%3d  %11.4e %11.4e  %11.4e %11.4e  %11.4e %11.4e\n",
                l,dA[n],ddA[n],dddA[n]);
        }

        printf("\n#Values of dB, ddB and dddB for N=%d\n",N);
        printf("#%2s  %11s %11s  %11s %11s  %11s %11s\n",
            "l","real(dB)","imag(dB)","real(ddB)","imag(ddB)",
            "real(dddB)","imag(dddB)");
        for(int l=-N/2+1;l<=N/2;l++){
            int n; if(l<0) n=l+N; else n=l;
            printf("%3d  %11.4e %11.4e  %11.4e %11.4e  %11.4e %11.4e\n",
                l,dB[n],ddB[n],dddB[n]);
        }

    }
    return 0;
}
```

where `fft.c` is given by

```
#include <complex.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "fft.h"

void dft(int N,complex x[N],complex y[N]){
    for(int j=0;j<N;j++){
        y[j]=0.0;
        for(int l=0;l<N;l++){
            y[j]+=cexp(-2.0*M_PI*I/N*l*j)*x[l];
```

```
        }
        y[j]/=N;
    }
}
void dift(int N,complex y[N],complex x[N]){
    for(int j=0;j<N;j++){
        x[j]=0.0;
        for(int l=0;l<N;l++){
            x[j]+=cexp(2.0*M_PI*I/N*l*j)*y[l];
        }
    }
}
void fftwork(int N,int s,complex *x,complex *y){
    if(N==1){
        y[0]=x[0];
        return;
    }
    if(N%2){
        printf("Error N was not divisible by 2!\n");
        exit(1);
    }
    int K=N/2;
    fftwork(K,2*s,x,y);
    fftwork(K,2*s,x+s,y+K);
    for(int j=0;j<K;j++){
        complex ye=y[j],yo=y[j+K];
        complex omega=cexp(-2*M_PI*I/N*j);
        y[j]=ye+omega*yo; y[j+K]=ye-omega*yo;
    }
}
void fft(int N,complex x[N],complex y[N]){
    fftwork(N,1,x,y);
    for(int j=0;j<N;j++) y[j]/=N;
}
void fiftwork(int N,int s,complex *x,complex *y){
    if(N==1){
        y[0]=x[0];
        return;
    }
    if(N%2){
        printf("Error N was not divisible by 2!\n");
        exit(1);
    }
    int K=N/2;
```

```
    fiftwork(K,2*s,x,y);
    fiftwork(K,2*s,x+s,y+K);
    for(int j=0;j<K;j++){
        complex ye=y[j],yo=y[j+K];
        complex omega=cexp(2*M_PI*I/N*j);
        y[j]=ye+omega*yo; y[j+K]=ye-omega*yo;
    }
}
void fift(int N,complex x[N],complex y[N]){
    fiftwork(N,1,x,y);
}
```

and `fft.h` is

```
#ifndef _FFT_H
#define _FFT_H

#include <complex.h>

extern void dft(int N,complex x[N],complex y[N]);
extern void dift(int N,complex y[N],complex x[N]);
extern void fft(int N,complex x[N],complex y[N]);
extern void fift(int N,complex x[N],complex y[N]);

#endif
```

produces the output

```
#Values of dA, ddA and dddA for N=4
# l     real(dA)    imag(dA)    real(ddA)   imag(ddA)   real(dddA)  imag(dddA)
 -1  -3.4879e-16  1.7061e+00   8.7879e-01  9.8608e-32   3.4424e-15 -6.7356e+01
  0   3.6920e+00 -1.7061e+00   1.0720e+01  0.0000e+00  -3.6439e+01  6.7356e+01
  1   3.4879e-16  1.7061e+00  -2.2319e+01 -9.8608e-32  -3.4424e-15 -6.7356e+01
  2  -3.6920e+00 -1.7061e+00   1.0720e+01  0.0000e+00   3.6439e+01  6.7356e+01

#Values of dB, ddB and dddB for N=4
# l     real(dB)    imag(dB)    real(ddB)   imag(ddB)   real(dddB)  imag(dddB)
 -1  -3.4879e-16  2.1356e-32   1.1599e+01  9.8608e-32   3.4424e-15 -2.1078e-31
  0   3.6920e+00  2.2606e-16   1.0957e-15  0.0000e+00  -3.6439e+01 -2.2311e-15
  1   3.4879e-16 -2.1356e-32  -1.1599e+01 -9.8608e-32  -3.4424e-15  2.1078e-31
  2  -3.6920e+00 -2.2606e-16  -1.0957e-15  0.0000e+00   3.6439e+01  2.2311e-15

#Values of dA, ddA and dddA for N=8
# l     real(dA)    imag(dA)    real(ddA)   imag(ddA)   real(dddA)  imag(dddA)
 -3  -1.1039e+00 -6.8791e-02   5.9339e+00 -4.4409e-16  -1.6006e+01  1.0863e+01
 -2   0.0000e+00  6.8791e-02   3.5579e+00  4.3232e-16  -3.1554e-30 -1.0863e+01
 -1   1.1039e+00 -6.8791e-02   5.9339e+00  2.6837e-31   1.6006e+01  1.0863e+01
  0   3.1280e+00  6.8791e-02   9.8555e+00  0.0000e+00   2.5249e+00 -1.0863e+01
  1   4.5162e+00 -6.8791e-02  -4.2050e+00  4.4409e-16  -1.1871e+02  1.0863e+01
  2   1.3951e-15  6.8791e-02  -2.6727e+01 -4.3232e-16  -5.5078e-14 -1.0863e+01
  3  -4.5162e+00 -6.8791e-02  -4.2050e+00  2.6837e-31   1.1871e+02  1.0863e+01
```

```
  4  -3.1280e+00  6.8791e-02   9.8555e+00  0.0000e+00  -2.5249e+00 -1.0863e+01

#Values of dB, ddB and dddB for N=8
# l     real(dB)    imag(dB)    real(ddB)   imag(ddB)   real(dddB)  imag(dddB)
 -3  -1.1039e+00  1.1758e-16   5.0695e+00 -4.4409e-16  -1.6006e+01  4.1242e-15
 -2   0.0000e+00  4.2713e-32   4.4224e+00  4.3232e-16  -3.1554e-30 -1.6862e-30
 -1   1.1039e+00 -1.1758e-16   5.0695e+00  2.6837e-31   1.6006e+01 -7.6769e-15
  0   3.1280e+00  1.2272e-16   1.0720e+01  0.0000e+00   2.5249e+00  4.9082e-15
  1   4.5162e+00 -3.2651e-16  -5.0695e+00  4.4409e-16  -1.1871e+02  4.1242e-15
  2   1.3951e-15 -4.2713e-32  -2.5862e+01 -4.3232e-16  -5.5078e-14  1.6862e-30
  3  -4.5162e+00  3.2651e-16  -5.0695e+00  2.6837e-31   1.1871e+02 -5.7149e-16
  4  -3.1280e+00 -1.2272e-16   1.0720e+01  0.0000e+00  -2.5249e+00 -4.9082e-15

#Values of dA, ddA and dddA for N=16
# l     real(dA)    imag(dA)    real(ddA)   imag(ddA)   real(dddA)  imag(dddA)
 -7  -1.9796e+00 -5.0068e-06   8.3215e+00  8.4377e-15  -1.9568e+01  3.1625e-03
 -6  -1.0953e+00  5.0068e-06   5.8743e+00 -9.7700e-15  -1.7527e+01 -3.1625e-03
 -5  -4.7726e-01 -5.0068e-06   4.1935e+00  7.5495e-15  -9.0347e+00  3.1625e-03
 -4  -1.0028e-15  5.0068e-06   3.6308e+00 -5.4679e-15   2.2935e-13 -3.1625e-03
 -3   4.7726e-01 -5.0068e-06   4.1935e+00  3.9968e-15   9.0347e+00  3.1625e-03
 -2   1.0953e+00  5.0068e-06   5.8743e+00 -7.9936e-15   1.7527e+01 -3.1625e-03
 -1   1.9796e+00 -5.0068e-06   8.3215e+00  9.8810e-15   1.9568e+01  3.1625e-03
  0   3.1416e+00  5.0068e-06   9.8696e+00 -5.2712e-15   3.6572e-04 -3.1625e-03
  1   4.2556e+00 -5.0068e-06   6.8139e+00 -2.2204e-15  -5.4371e+01  3.1625e-03
  2   4.5053e+00  5.0068e-06  -4.1456e+00  9.7700e-15  -1.1656e+02 -3.1625e-03
  3   3.0285e+00 -5.0068e-06  -1.9329e+01 -1.3767e-14  -1.0836e+02  3.1625e-03
  4  -3.9239e-16  5.0068e-06  -2.6828e+01  1.4234e-14   4.8667e-13 -3.1625e-03
  5  -3.0285e+00 -5.0068e-06  -1.9329e+01 -1.0214e-14   1.0836e+02  3.1625e-03
  6  -4.5053e+00  5.0068e-06  -4.1456e+00  7.9936e-15   1.1656e+02 -3.1625e-03
  7  -4.2556e+00 -5.0068e-06   6.8139e+00 -3.6637e-15   5.4371e+01  3.1625e-03
  8  -3.1416e+00  5.0068e-06   9.8696e+00 -3.4948e-15  -3.6572e-04 -3.1625e-03

#Values of dB, ddB and dddB for N=16
# l     real(dB)    imag(dB)    real(ddB)   imag(ddB)   real(dddB)  imag(dddB)
 -7  -1.9796e+00  4.4830e-16   8.3214e+00  8.4377e-15  -1.9568e+01 -1.8541e-13
 -6  -1.0953e+00 -1.9970e-16   5.8744e+00 -9.7700e-15  -1.7527e+01  8.1801e-14
 -5  -4.7726e-01 -1.1523e-16   4.1934e+00  7.5495e-15  -9.0347e+00 -9.9930e-15
 -4  -1.0028e-15  2.2204e-16   3.6309e+00 -5.4679e-15   2.2935e-13 -2.1316e-14
 -3   4.7726e-01  2.2626e-16   4.1934e+00  3.9968e-15   9.0347e+00  2.0651e-14
 -2   1.0953e+00 -2.4439e-16   5.8744e+00 -7.9936e-15   1.7527e+01 -5.6932e-14
 -1   1.9796e+00 -6.7035e-16   8.3214e+00  9.8810e-15   1.9568e+01  1.6498e-13
  0   3.1416e+00  8.6390e-16   9.8697e+00 -5.2712e-15   3.6572e-04 -2.7484e-13
  1   4.2556e+00 -8.8397e-16   6.8138e+00 -2.2204e-15  -5.4371e+01  3.1197e-13
  2   4.5053e+00  6.8848e-16  -4.1455e+00  9.7700e-15  -1.1656e+02 -2.6636e-13
  3   3.0285e+00 -3.3728e-16  -1.9329e+01 -1.3767e-14  -1.0836e+02  1.3922e-13
  4  -3.9239e-16 -2.2204e-16  -2.6828e+01  1.4234e-14   4.8667e-13  2.1316e-14
  5  -3.0285e+00  2.2626e-16  -1.9329e+01 -1.0214e-14   1.0836e+02 -1.4988e-13
  6  -4.5053e+00 -2.4439e-16  -4.1455e+00  7.9936e-15   1.1656e+02  2.4150e-13
  7  -4.2556e+00  1.1060e-15   6.8138e+00 -3.6637e-15   5.4371e+01 -2.9155e-13
  8  -3.1416e+00 -8.6390e-16   9.8697e+00 -3.4948e-15  -3.6572e-04  2.7484e-13
```

The imaginary parts of A'_ℓ and A'''_ℓ are comparatively large while the imaginary parts of A''_ℓ, B'_ℓ, B''_ℓ and B'''_ℓ are all essential zero to within the limits of rounding error. In particular, since double precision floating point arithmetic has about 15 significant digits, then any number less that 1^{-14} is numerically zero in comparison to a number of unit magnitude. It is interesting that A''_ℓ is also real valued. Since

$$y_{N/2} = \frac{1}{N}\sum_{\ell=-N/2+1}^{N/2} f\Big(\frac{2\ell}{N}\Big)e^{-\pi i\ell} = \frac{1}{N}\sum_{\ell=-N/2+1}^{N/2} f\Big(\frac{2\ell}{N}\Big)(-1)^\ell$$

is a sum of real terms, then $y_{N/2}$ is real. Consequently, the term for $j = N/2$ in the sum definition of $A''(x)$ given by

$$A''_\ell - B''_\ell = -\pi^2(N/2)^2 y_{N/2} e^{i\pi(N/2)(2\ell/N)} = -\pi^2(N/2)^2 y_{N/2}(-1)^\ell$$

is also real. Since B''_ℓ has already been shown to be real it follows that A''_ℓ is also real. Note however, that we have only shown $A''(x)$ is real when $x = 2\ell/N$. For other values of x it is still the case that $A''(x)$ may be complex.

5. Compute the errors

$$E_k = \bigg(\frac{1}{N}\sum_{\ell=-N/2+1}^{N/2}\Big|A^{(k)}_\ell - f^{(k)}\Big(\frac{2\ell}{N}\Big)\Big|^2\bigg)^{1/2}$$

and

$$\widetilde{E}_k = \bigg(\frac{1}{N}\sum_{\ell=-N/2+1}^{N/2}\Big|\widetilde{A}^{(k)}_\ell - f^{(k)}\Big(\frac{2\ell}{N}\Big)\Big|^2\bigg)^{1/2}$$

for $k = 1, 2, 3$ and $N = 4, 8, \ldots, 65536$. Comment on the quality of the approximations.

The program

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <complex.h>
#include <sys/resource.h>

#include "fft.h"

extern double f(double x);
extern double df(double x);
extern double ddf(double x);
extern double dddf(double x);

#include "f.i"

double rmsdist(int N,complex x[N],complex y[N]){
```

```
    double r=0.0;
    for(int n=0;n<N;n++){
        complex t=x[n]-y[n];
        r=r+t*conj(t);
    }
    return sqrt(r/N);
}

int main(){
    {
        struct rlimit rlim={RLIM_INFINITY,
                            RLIM_INFINITY };
        setrlimit(RLIMIT_STACK,&rlim);
    }
    for(int N=4;N<=65536;N=N*2){
        complex F[N],dF[N],ddF[N],dddF[N],Y[N];
        for(int l=-N/2+1;l<=N/2;l++){
            int n; if(l<0) n=l+N; else n=l;
            F[n]=f(2.0*l/N);
            dF[n]=df(2.0*l/N);
            ddF[n]=ddf(2.0*l/N);
            dddF[n]=dddf(2.0*l/N);
        }
        fft(N,F,Y);
        complex dA[N],ddA[N],dddA[N];
        complex dB[N],ddB[N],dddB[N];  //  B = tilde A
        complex T[N];

        for(int j=-N/2+1;j<=N/2;j++){
            int n; if(j<0) n=j+N; else n=j;
            T[n]=I*M_PI*j*Y[n];
        }
        fift(N,T,dA);
        T[N/2]=0;
        fift(N,T,dB);

        for(int j=-N/2+1;j<=N/2;j++){
            int n; if(j<0) n=j+N; else n=j;
            T[n]=-M_PI*M_PI*j*j*Y[n];
        }
        fift(N,T,ddA);
        T[N/2]=0;
        fift(N,T,ddB);

```

```
        for(int j=-N/2+1;j<=N/2;j++){
            int n; if(j<0) n=j+N; else n=j;
            T[n]=-I*M_PI*M_PI*M_PI*j*j*j*Y[n];
        }
        fift(N,T,dddA);
        T[N/2]=0;
        fift(N,T,dddB);

        if(N==4)
            printf("#%4s %11s %11s %11s %11s %11s %11s\n",
            "N","E1","tilde-E1","E2","tilde-E2","E3","tilde-E3");
        printf("%5d %11.4e %11.4e %11.4e %11.4e %11.4e %11.4e\n",
            N,rmsdist(N,dA,dF),rmsdist(N,dB,dF),
            rmsdist(N,ddA,ddF),rmsdist(N,ddB,ddF),
            rmsdist(N,dddA,dddF),rmsdist(N,dddB,dddF));

    }
    return 0;
}
```

produces the output

# N	E1	tilde-E1	E2	tilde-E2	E3	tilde-E3
4	1.7500e+00	3.8920e-01	2.7091e+00	1.1071e+01	7.2116e+01	2.5766e+01
8	6.9469e-02	9.6818e-03	6.1460e-02	8.6664e-01	1.1015e+01	1.8229e+00
16	5.0221e-06	3.9277e-07	2.4771e-06	1.2586e-04	3.1732e-03	2.5982e-04
32	9.7383e-15	7.9807e-15	4.0878e-13	2.9389e-13	1.9035e-11	1.2787e-11
64	1.9699e-14	1.6232e-14	1.6501e-12	1.2103e-12	1.5136e-10	1.0092e-10
128	3.4891e-14	3.4891e-14	5.6461e-12	5.6461e-12	9.9190e-10	9.9190e-10
256	6.5037e-14	6.5037e-14	1.9939e-11	1.9939e-11	6.7432e-09	6.7432e-09
512	1.1979e-13	1.1979e-13	7.4613e-11	7.4613e-11	5.2421e-08	5.2421e-08
1024	3.4041e-13	3.4041e-13	4.6722e-10	4.6722e-10	6.8584e-07	6.8584e-07
2048	5.6296e-13	5.6296e-13	1.3485e-09	1.3485e-09	3.6094e-06	3.6094e-06
4096	1.2525e-12	1.2525e-12	6.4740e-09	6.4740e-09	3.6486e-05	3.6486e-05
8192	2.4351e-12	2.4351e-12	2.4203e-08	2.4203e-08	2.6545e-04	2.6545e-04
16384	4.6974e-12	4.6974e-12	9.1677e-08	9.1677e-08	1.9636e-03	1.9636e-03
32768	1.0509e-11	1.0509e-11	4.4714e-07	4.4714e-07	2.0721e-02	2.0721e-02
65536	2.5330e-11	2.5330e-11	2.2232e-06	2.2232e-06	2.1131e-01	2.1131e-01

We comment that B'_ℓ is more accurate than A'_ℓ and B'''_ℓ is more accurate than A'''_ℓ. On the other hand, A''_ℓ appears more accurate than B''_ℓ especially for small values of N, such as, for example $N = 16$. When N is large the error in the approximations given by A and the error from B are essentially the same; however, it should be noted that the quality of the approximations decrease when N gets very large. This is especially noticeable for the third derivative for which the quality of approximation is best when $N = 32$ and remarkably inaccurate when $N = 65536$.

Just as the derivative approximations formed by finite differences decrease in quality when h is too small, the reason that the error increases when N is large is likely due to rounding error in the floating point arithmetic. For example, the amplitude of highest

Fourier mode is multiplied by j^3 for $j = N/2 - 1$ when approximating f'''. Since $j^3 \approx 3.5 \times 10^{13}$ for $N = 65536$ and the double precision arithmetic is accurate to at most 10^{-15}, this simple estimate implies the resulting error in the highest mode of the derivative approximation could be as much as 3.5×10^{-2}. While it is possible filtering of the higher modes could improve the higher order derivative approximations when N is large, we do not explore this line of inquiry here.

6. Repeat parts 3 through 5 above for the function $f(x) = \exp(x^2)$. Compare the size of the errors and rate of convergence as $N \to \infty$ in this case to the previous one. Explain any differences or similarities.

Modify the Maple script as

```
restart;
with(codegen):
f:=x->exp(x^2);
df:=unapply(simplify(diff(f(x),x)),x);
ddf:=unapply(simplify(diff(df(x),x)),x);
dddf:=unapply(simplify(diff(ddf(x),x)),x);
C(f,optimized,filename="f6.i");
C(df,optimized,filename="f6.i");
C(ddf,optimized,filename="f6.i");
C(dddf,optimized,filename="f6.i");
```

to produce a new file `f6.i` that can be included in the previous two programs in place of `f.i` . The output of the script is

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> with(codegen):
Warning, the protected name MathML has been redefined and unprotected
> f:=x->exp(x^2);
                                                2
                              f := x -> exp(x )

> df:=unapply(simplify(diff(f(x),x)),x);
                                                  2
                            df := x -> 2 x exp(x )

> ddf:=unapply(simplify(diff(df(x),x)),x);
                                              2           2
                       ddf := x -> 2 exp(x ) (1 + 2 x )

> dddf:=unapply(simplify(diff(ddf(x),x)),x);
                                                2           2
                     dddf := x -> 4 x exp(x ) (3 + 2 x )
```

```
> C(f,optimized,filename="f6.i");
> C(df,optimized,filename="f6.i");
> C(ddf,optimized,filename="f6.i");
> C(dddf,optimized,filename="f6.i");
> quit
bytes used=2010624, alloc=1703624, time=0.05
```

which indicates the exact first, second and third derivatives of $\exp(x^2)$ for the modified part 3. Now, the program for the modified part 4 produces

```
#Values of dA, ddA and dddA for N=4
# l      real(dA)    imag(dA)    real(ddA)   imag(ddA)   real(dddA)  imag(dddA)
 -1   -2.6991e+00 -1.8068e+00   1.1352e+01  0.0000e+00   2.6639e+01  7.1329e+01
  0    0.0000e+00  1.8068e+00  -2.8729e+00  0.0000e+00   0.0000e+00 -7.1329e+01
  1    2.6991e+00 -1.8068e+00   1.1352e+01  0.0000e+00  -2.6639e+01  7.1329e+01
  2    0.0000e+00  1.8068e+00  -1.9832e+01 -0.0000e+00   0.0000e+00 -7.1329e+01

#Values of dB, ddB and dddB for N=4
# l      real(dB)    imag(dB)    real(ddB)   imag(ddB)   real(dddB)  imag(dddB)
 -1   -2.6991e+00  1.6526e-16  -0.0000e+00 -0.0000e+00   2.6639e+01 -1.6311e-15
  0    0.0000e+00  0.0000e+00   8.4794e+00  0.0000e+00   0.0000e+00  0.0000e+00
  1    2.6991e+00 -1.6526e-16   0.0000e+00  0.0000e+00  -2.6639e+01  1.6311e-15
  2    0.0000e+00  0.0000e+00  -8.4794e+00  0.0000e+00   0.0000e+00  0.0000e+00

#Values of dA, ddA and dddA for N=8
# l      real(dA)    imag(dA)    real(ddA)   imag(ddA)   real(dddA)  imag(dddA)
 -3   -4.5391e+00 -1.0167e+00   1.9720e+01 -3.1086e-15   1.9586e+02  1.6055e+02
 -2   -3.6901e-01  1.0167e+00  -1.4236e+00  0.0000e+00  -1.3434e+02 -1.6055e+02
 -1   -9.2555e-01 -1.0167e+00   5.8315e+00  3.1086e-15   5.3205e+01  1.6055e+02
  0    0.0000e+00  1.0167e+00  -1.0086e+00 -2.1915e-15   0.0000e+00 -1.6055e+02
  1    9.2555e-01 -1.0167e+00   5.8315e+00  3.1086e-15  -5.3205e+01  1.6055e+02
  2    3.6901e-01  1.0167e+00  -1.4236e+00  0.0000e+00   1.3434e+02 -1.6055e+02
  3    4.5391e+00 -1.0167e+00   1.9720e+01 -3.1086e-15  -1.9586e+02  1.6055e+02
  4    0.0000e+00  1.0167e+00  -4.7248e+01  2.1915e-15   0.0000e+00 -1.6055e+02

#Values of dB, ddB and dddB for N=8
# l      real(dB)    imag(dB)    real(ddB)   imag(ddB)   real(dddB)  imag(dddB)
 -3   -4.5391e+00  3.3267e-16   6.9444e+00 -3.1086e-15   1.9586e+02 -3.2789e-14
 -2   -3.6901e-01 -3.2619e-16   1.1352e+01  0.0000e+00  -1.3434e+02  3.2322e-14
 -1   -9.2555e-01  3.3346e-16  -6.9444e+00  3.1086e-15   5.3205e+01 -2.4054e-14
  0    0.0000e+00  0.0000e+00   1.1767e+01 -2.1915e-15   0.0000e+00  0.0000e+00
  1    9.2555e-01 -1.1141e-16  -6.9444e+00  3.1086e-15  -5.3205e+01  2.4054e-14
  2    3.6901e-01  3.2619e-16   1.1352e+01  0.0000e+00   1.3434e+02 -3.2322e-14
  3    4.5391e+00 -5.5472e-16   6.9444e+00 -3.1086e-15  -1.9586e+02  3.2789e-14
  4    0.0000e+00  0.0000e+00  -3.4472e+01  2.1915e-15   0.0000e+00  0.0000e+00

#Values of dA, ddA and dddA for N=16
# l      real(dA)    imag(dA)    real(ddA)   imag(ddA)   real(dddA)  imag(dddA)
 -7   -5.8129e+00 -5.2695e-01   3.3905e+01 -6.2172e-15   9.2916e+02  3.3285e+02
 -6   -1.4782e+00  5.2695e-01  -1.4418e+00 -1.1102e-15  -6.7033e+02 -3.3285e+02
 -5   -2.5959e+00 -5.2695e-01   1.0037e+01  3.4417e-15   4.2946e+02  3.3285e+02
 -4   -7.7469e-01  5.2695e-01   7.5036e-01  1.3612e-15  -3.1740e+02 -3.3285e+02
 -3   -1.2066e+00 -5.2695e-01   5.2505e+00 -1.7764e-15   2.0444e+02  3.3285e+02
```

```
 -2  -3.1846e-01  5.2695e-01   5.0630e-01 -1.3878e-15  -1.3489e+02 -3.3285e+02
 -1  -3.5683e-01 -5.2695e-01   3.7821e+00  1.4211e-14   6.1953e+01  3.3285e+02
  0  -5.4498e-16  5.2695e-01   3.7389e-01 -1.0273e-14   1.8481e-13 -3.3285e+02
  1   3.5683e-01 -5.2695e-01   3.7821e+00  8.8818e-16  -6.1953e+01  3.3285e+02
  2   3.1846e-01  5.2695e-01   5.0630e-01  1.1102e-15   1.3489e+02 -3.3285e+02
  3   1.2066e+00 -5.2695e-01   5.2505e+00  1.8874e-15  -2.0444e+02  3.3285e+02
  4   7.7469e-01  5.2695e-01   7.5036e-01 -5.7442e-15   3.1740e+02 -3.3285e+02
  5   2.5959e+00 -5.2695e-01   1.0037e+01  7.1054e-15  -4.2946e+02  3.3285e+02
  6   1.4782e+00  5.2695e-01  -1.4418e+00  1.3878e-15   6.7033e+02 -3.3285e+02
  7   5.8129e+00 -5.2695e-01   3.3905e+01 -1.9540e-14  -9.2916e+02  3.3285e+02
  8   5.4498e-16  5.2695e-01  -1.0595e+02  1.4656e-14  -1.8481e-13 -3.3285e+02

#Values of dB, ddB and dddB for N=16
# l     real(dB)    imag(dB)    real(ddB)   imag(ddB)   real(dddB)  imag(dddB)
 -7  -5.8129e+00  8.3941e-16   2.0662e+01 -6.2172e-15   9.2916e+02 -2.0878e-13
 -6  -1.4782e+00 -3.0542e-16   1.1802e+01 -1.1102e-15  -6.7033e+02  2.3512e-13
 -5  -2.5959e+00  6.0388e-16  -3.2070e+00  3.4417e-15   4.2946e+02 -1.8912e-13
 -4  -7.7469e-01 -6.1561e-17   1.3994e+01  1.3612e-15  -3.1740e+02  8.1182e-14
 -3  -1.2066e+00 -2.7082e-16  -7.9931e+00 -1.7764e-15   2.0444e+02 -1.2352e-13
 -2  -3.1846e-01 -6.3827e-16   1.3750e+01 -1.3878e-15  -1.3489e+02  1.6279e-13
 -1  -3.5683e-01  6.0388e-16  -9.4616e+00  1.4211e-14   6.1953e+01 -1.3228e-13
  0  -5.4498e-16  4.4409e-16   1.3618e+01 -1.0273e-14   1.8481e-13 -8.5265e-14
  1   3.5683e-01 -4.8771e-17  -9.4616e+00  8.8818e-16  -6.1953e+01  1.8591e-14
  2   3.1846e-01  2.7647e-17   1.3750e+01  1.1102e-15   1.3489e+02 -1.0594e-13
  3   1.2066e+00 -2.8430e-16  -7.9931e+00  1.8874e-15  -2.0444e+02  3.8252e-14
  4   7.7469e-01  6.1561e-17   1.3994e+01 -5.7442e-15   3.1740e+02 -8.1182e-14
  5   2.5959e+00 -2.7082e-16  -3.2070e+00  7.1054e-15  -4.2946e+02  2.7439e-13
  6   1.4782e+00  9.1604e-16   1.1802e+01  1.3878e-15   6.7033e+02 -2.9196e-13
  7   5.8129e+00 -1.1725e-15   2.0662e+01 -1.9540e-14  -9.2916e+02  3.2247e-13
  8   5.4498e-16 -4.4409e-16  -9.2709e+01  1.4656e-14  -1.8481e-13  8.5265e-14
```

Note that the first line of each table is is noticeably different as N ranges over the values 4, 8 and 16. The line in the middle of the table at $\ell = 0$ gives the correct value of $f'(0) = 0$ for each of the values of N, but the value of

$$f''(0) = 2(1 + 2 \cdot 0^2)\exp(0^2) = 2$$

is wrong for all values of N.

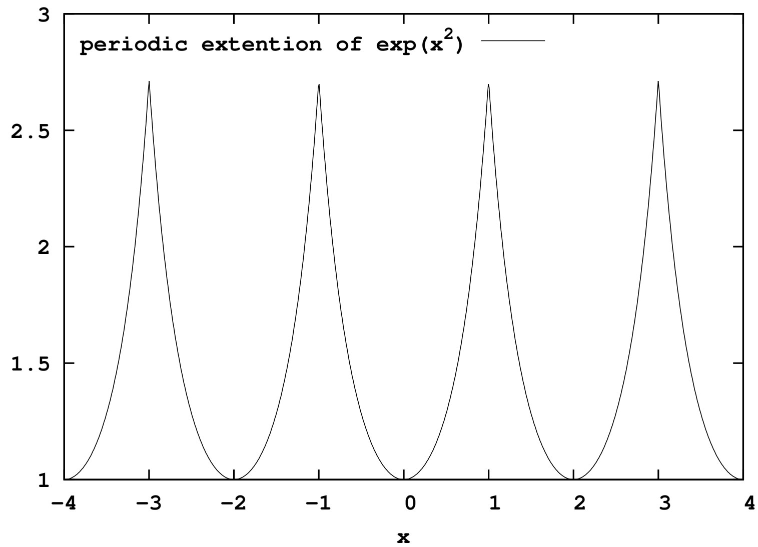
The output from the program for the modified part 5 is

```
#   N          E1    tilde-E1          E2    tilde-E2          E3    tilde-E3
    4  3.4139e+00  2.8966e+00  1.8990e+01  1.3097e+01  8.0382e+01  3.7061e+01
    8  2.4259e+00  2.2026e+00  2.3528e+01  1.9241e+01  2.0666e+02  1.3012e+02
   16  1.7149e+00  1.6320e+00  3.1876e+01  2.8899e+01  5.6939e+02  4.6196e+02
   32  1.2123e+00  1.1828e+00  4.4555e+01  4.2485e+01  1.5986e+03  1.4504e+03
   64  8.5716e-01  8.4673e-01  6.2822e+01  6.1373e+01  4.5128e+03  4.3069e+03
  128  6.0609e-01  6.0241e-01  8.8777e+01  8.7758e+01  1.2758e+04  1.2470e+04
  256  4.2857e-01  4.2727e-01  1.2553e+02  1.2481e+02  3.6080e+04  3.5675e+04
  512  3.0304e-01  3.0258e-01  1.7751e+02  1.7700e+02  1.0205e+05  1.0148e+05
 1024  2.1428e-01  2.1412e-01  2.5104e+02  2.5068e+02  2.8863e+05  2.8782e+05
 2048  1.5152e-01  1.5146e-01  3.5502e+02  3.5477e+02  8.1637e+05  8.1523e+05
 4096  1.0714e-01  1.0712e-01  5.0207e+02  5.0189e+02  2.3090e+06  2.3074e+06
```

8192	7.5761e-02	7.5754e-02	7.1004e+02	7.0991e+02	6.5310e+06	6.5287e+06
16384	5.3571e-02	5.3569e-02	1.0041e+03	1.0041e+03	1.8472e+07	1.8469e+07
32768	3.7881e-02	3.7880e-02	1.4201e+03	1.4200e+03	5.2248e+07	5.2243e+07
65536	2.6786e-02	2.6785e-02	2.0083e+03	2.0082e+03	1.4778e+08	1.4777e+08

Note that the approximations of the second and third derivatives are completely wrong while the first derivative is barely good to two decimal points.

Although the function $\exp(x^2)$ appears quite similar to the previous one, remember that the theory of approximation by Fourier series assumes the function is both periodic and smooth. While $\exp(x^2)$ does not appear periodic, our code is written in such a way that it periodically extends the function defined on the interval $[-1, 1]$ to the entire real line. In this case, the periodically extended function has a graph which looks like



At this point the problem can be seen: the periodically extended function (though originally smooth) now has a cusp-like corner at each even integer value of x. The resulting function is continuous; however, a lack of smoothness explains why the derivative approximations are inaccurate.