Project 3

Due on Friday, May 19th, 2017 (Three weeks)

Consider the following ordinary differential equation for u:

$$u(x) - u''(x) = f(x) \quad x \in [0,1], \tag{1}$$

with periodic boundary conditions, i.e.,

$$\begin{aligned} u(0) &= u(1) \\ u'(0) &= u'(1). \end{aligned} \tag{2}$$

In this problem we want to approximate the solution to this equation numerically using the Fast Fourier Transform (FFT). Since we do not know what the solution is, you need to make sure that your program is giving the right answer.

(i) Consider $u(x) = \cosh(\sin(2\pi x))$, and compute $f(x)$ so that u is the solution to equation (1).

(ii) Approximating the second derivative using second order centered differences, one gets:

$$\begin{aligned} u_j - \frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} &= f_j, \quad j = 0, \cdots, n-1, && (3) \\ u(n) &= u(0), && (4) \\ u(-1) &= u(n-1). && (5) \end{aligned}$$

Show that the following vectors are eigenvectors of the resulting matrix, and find the corresponding eigenvalues:

$$v_j^k = e^{-2\pi ijk/n}, \quad j = 0, \cdots, n-1. \tag{6}$$

(iii) Based on (ii), u_j can be approximated as

$$u_j = \sum_{k=0}^{n-1} c_k e^{-2\pi ijk/n}, \quad j = 0, \cdots, n-1,$$

where c_k are coefficients to be determined. Solve the system of equations (3) using the FFT, for $n = 2^l$, $l = 6, 7, 8, 9$. An intrinsic function

fft in matlab can be used to conduct the discrete fourier transform. Compute the error of your algorithm, and show that the solution obtained is second order accurate.

Solution.

(i) Consider $u(x) = \cosh(\sin(2\pi x))$, where $x \in [0, 1]$. If we change the interval $[0, 1]$ to $[-\pi, \pi]$, we can get $u(x) = \cosh(\sin(x))$, where $x \in [-\pi, \pi]$. Noting that

$$\sinh x = \frac{e^x - e^{-x}}{2}, \quad \cosh x = \frac{e^x + e^{-x}}{2}.$$

$$(\sinh x)' = \cosh x, \quad (\cosh x)' = \sinh x.$$

Thus,

$$\begin{aligned} u'(x) &= \sinh(\sin(x))\cos(x), \\ u''(x) &= \cos^2(x)\cosh(\sin(x)) - \sinh(\sin(x))\sin(x). \end{aligned}$$

Then,

$$\begin{aligned} f(x) &= u(x) - u''(x) \\ &= \cosh(\sin(x)) - \cos^2(x)\cosh(\sin(x)) + \sinh(\sin(x))\sin(x) \\ &= \sin^2(x)\cosh(\sin(x)) + \sinh(\sin(x))\sin(x). \end{aligned}$$

We also note that $u(x)$ is a periodic function with period $L = \frac{1}{2}$, since

$$\begin{aligned} \cosh(\sin(2\pi(x + \frac{1}{2}))) &= \cosh(\sin(2\pi x + \pi)) \\ &= \cosh(-\sin(2\pi x)) \\ &= \cosh(\sin(2\pi x)) \\ &= u(x). \end{aligned}$$

It is easy to check $u(x)$ satisfy with periodic boundary conditions, i.e.,

$$\begin{aligned} u(0) &= u(1) \\ u'(0) &= u'(1). \end{aligned}$$

(ii) Approximating the second derivative using second order centered differences, one gets:

$$\begin{aligned} u_j - \frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} &= f_j, \quad j = 0, \cdots, n-1, \\ u(n) &= u(0), \\ u(-1) &= u(n-1). \end{aligned}$$

The method is similar to project 2, consider the matrix

$$\mathbf{A} = \begin{pmatrix} 1+\frac{2}{h^2} & -\frac{1}{h^2} & 0 & \cdots & 0 & -\frac{1}{h^2} \\ -\frac{1}{h^2} & 1+\frac{2}{h^2} & -\frac{1}{h^2} & \cdots & \cdots & 0 \\ 0 & -\frac{1}{h^2} & 1+\frac{2}{h^2} & -\frac{1}{h^2} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & -\frac{1}{h^2} \\ -\frac{1}{h^2} & 0 & \cdots & \cdots & -\frac{1}{h^2} & 1+\frac{2}{h^2} \end{pmatrix}_{n\times n}$$

According to the definition of an eigenvalue and an eigenvector of the matrix, it is easy to check the vector $\mathbf{v}^{(k)}$ given by

$$v_j^{(k)} = e^{-2\pi ijk/n}, \quad j = 0, \cdots, n-1,$$

is an eigenvector of the matrix $\mathbf{A}$, for each $k = 0, 1, \ldots, n-1$,

$$\omega = e^{-ih} = e^{-\frac{2\pi i}{n}},$$

is the n-th root of unity with i the imaginary unit. Noting that $\omega^{-1} = \bar{\omega}$. Then,

$$\begin{aligned} e^{-\frac{2\pi ijk}{n}} &= \omega^{kj}, \quad 0 \le j, k \le n-1, \\ \mathbf{v}^{(k)} &= (1, \omega^k, \omega^{2k}, \cdots, \omega^{(n-1)k})^T, \quad 0 \le k \le n-1, \end{aligned}$$

where $(\cdot)^T$ denotes the transpose of a matrix or a vector.

When $\ell \neq 0, n-1$, where ℓ denotes the index of rows, we can take the ℓ-th component of $\mathbf{A}\mathbf{v}^{(k)}$, i.e., we need to show that

$$\begin{aligned} (\mathbf{A}\mathbf{v}^{(k)})_\ell &= -\frac{1}{h^2}\omega^{(\ell-2)k} + (1+\frac{2}{h^2})\omega^{(\ell-1)k} - \frac{1}{h^2}\omega^{\ell k} \\ &\overset{(?)}{=} \lambda_k \mathbf{v}_\ell^{(k)} = \lambda_k \omega^{(\ell-1)k}, \end{aligned}$$

after simplification,

$$(1+\frac{2}{h^2}-\lambda_k)\omega^{\ell k}=\frac{1}{h^2}(\omega^{(\ell+1)k}+\omega^{(\ell-1)k})$$

$$\Rightarrow \quad \lambda_k=1+\frac{2}{h^2}-\frac{1}{h^2}(\omega^k+\omega^{-k})=1+\frac{2}{h^2}-\frac{1}{h^2}(\omega^k+\omega^{n-k}).$$

Let us check the first and the last component, for $\ell=0$, we need to show

$$\begin{aligned}(\mathbf{A}\mathbf{v}^{(k)})_0 &= 1+\frac{2}{h^2}-\frac{1}{h^2}\omega^k-\frac{1}{h^2}\omega^{(n-1)k}\\ &= 1+\frac{2}{h^2}-\frac{1}{h^2}\omega^k-\frac{1}{h^2}\omega^{-k} \quad (\text{ due to } \omega^n=1)\\ &= \lambda_k\\ &= \lambda_k v_0^{(k)}.\end{aligned}$$

for $\ell=n-1$, we need to show

$$\begin{aligned}(\mathbf{A}\mathbf{v}^{(k)})_{n-1} &= -\frac{1}{h^2}-\frac{1}{h^2}\omega^{(n-2)k}+(1+\frac{2}{h^2})\omega^{(n-1)k}\\ &\overset{(?)}{=} \lambda_k\mathbf{v}_{n-1}^{(k)}=\lambda_k\omega^{(n-1)k},\end{aligned}$$

$$\Rightarrow \quad (1+\frac{2}{h^2}-\lambda_k)\omega^{(n-1)k}=\frac{1}{h^2}(\omega^{nk}+\omega^{(n-2)k})$$

$$\Rightarrow \quad \lambda_k=1+\frac{2}{h^2}-\frac{1}{h^2}(\omega^k+\omega^{-k})$$

Therefore, the vectors $\mathbf{v}^{(k)}$ with components

$$v_j^k=e^{-2\pi ijk/n}, \quad j=0,\cdots,n-1,$$

and the corresponding eigenvalue

$$\lambda_k=1+\frac{2}{h^2}-\frac{1}{h^2}(\omega^k+\omega^{-k}), \quad k=0,\ldots,n-1,$$

are the eigenpairs of $\mathbf{A}$.

(iii) Based on (ii), u_j can be approximated as

$$u_j = \sum_{k=0}^{n-1} c_k e^{-2\pi ijk/n}, \quad j = 0, \cdots, n-1,$$

i.e., $\mathbf{u} = \mathbf{F}\mathbf{c}$, where

$$\mathbf{F} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \omega^{n-1} & \cdots & \omega^{(n-1)^2} \end{pmatrix}_{n\times n}$$

Suppose we have $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})^T$, define the Discrete Fourier Transform (DFT) of $\mathbf{u}$ as $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})^T \triangleq \hat{\mathbf{u}}$, where

$$c_k = \sum_{j=0}^{n-1} u_j e^{2\pi ijk/n}, \quad j = 0, \cdots, n-1.$$

We simplify the system (3) as follows,

$$(h^2+2)u_j - u_{j+1} - u_{j-1} = h^2 f_j.$$

Replacing $u_j = \sum_{k=0}^{n-1} c_k e^{-2\pi ijk/n}$ in the differential equation as above, we obtain that

$$(h^2+2)\sum_{k=0}^{n-1} \hat{u_k}\omega^{jk} - \sum_{k=0}^{n-1} \hat{u_k}\omega^{(j+1)k} - \sum_{k=0}^{n-1} \hat{u_k}\omega^{(j-1)k} = h^2 \sum_{k=0}^{n-1} \hat{f}_k \omega^{jk}.$$

Combining terms and doing an algebraic manipulation then results in this:

$$\sum_{k=0}^{n-1} \hat{u_k}\omega^{jk}(h^2 + 2 - \omega^k - \bar{\omega}^k) = \sum_{k=0}^{n-1} h^2 \hat{f}_k \omega^{jk}.$$

Taking the inverse DFT of both sides and dividing by $h^2 + 2 - \omega^k - \bar{\omega}^k$, which we assume is never 0, we find that

$$\hat{u_k} = h^2(h^2 + 2 - \omega^k - \bar{\omega}^k)^{-1}\hat{f}_k.$$

Thus, we have found the DFT of u. Inverting this then recovers u itself. In the following, we will discuss methods for fast computation of the DFT and its inverse.

Let us consider the DFT of a periodic sequence u with period $n = 2m$. The $\hat{u_k}$ are calculated via

$$\hat{u_k} = \sum_{j=0}^{2m-1} u_j \bar{\omega}^{jk}.$$

Splitting the sum above into a sum over even and odd integers yields

$$\begin{aligned}\hat{u_k} &= \sum_{j=0}^{m-1} u_{2j}\hat{\omega}^{2jk} + \sum_{j=0}^{m-1} u_{2j+1}\hat{\omega}^{(2j+1)k} \\ &= \sum_{j=0}^{m-1} u_{2j}\hat{W}^{jk} + \hat{\omega}^k\big(\sum_{j=0}^{m-1} u_{2j+1}\hat{W}^{jk}\big),\end{aligned}$$

where $W := e^{\frac{2\pi i}{m}} = \omega^2$. If we call an intrinsic function fft in matlab, then the program is that

```
%
% Project 3
% FFT method
%
clear
f =@(x) cosh(sin(x))...%convert [0,1] to [-pi,pi], we need to make change.
    -cos(x).*cos(x).*cosh(sin(x))...
    +sinh(sin(x)).*sin(x); % define the right hand function
L=[6 7 8 9];
n = 2.^L;
figure(1)
for j=1:4
xk = -pi*ones(n(j),1)+(0:n(j)-1)'*2*pi/n(j); % grid points
h(j)=2*pi/n(j);
w=exp(-1i*h(j));
fk=fft(f(xk));
for k=1:n(j)
```

```
uk(k)=h(j)*h(j)*fk(k)/(h(j)*h(j)+2-w.^(k-1)-w.^(n(j)-k+1));
end
u=ifft(uk);
subplot(2,2,j),
plot(xk,u,'k.-'),hold on,grid on
y =@(x) cosh(sin(x));%exact solution
yk=y(xk);
plot(xk,yk,'k--'),hold on
error(j)=norm(u'-yk);
end
format long
error,h
figure(2)
loglog(h,error,'ks-'),hold on,grid on
xlabel('h'),ylabel('error of fft'),hold on,
title({['Log-Log plot of the error'];
['Due on 2017/5/12, Changjian Xie']}),
hold on
%calculate the slope as follow
r=sum((log(h)).^2);
s=sum(log(h));
t=sum((log(h)).*log(error));
p=sum(log(error));
alpha=(p*s-6*t)/(s^2-6*r)
```

the result is as follows,

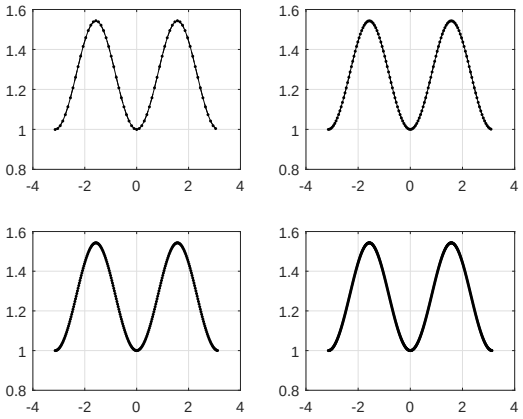

Figure 1: Using the values $n = 2^L$, where $L = 6, 7, 8, 9$. Do a plot of the numerical solution using FFT, compared with the exact solution.

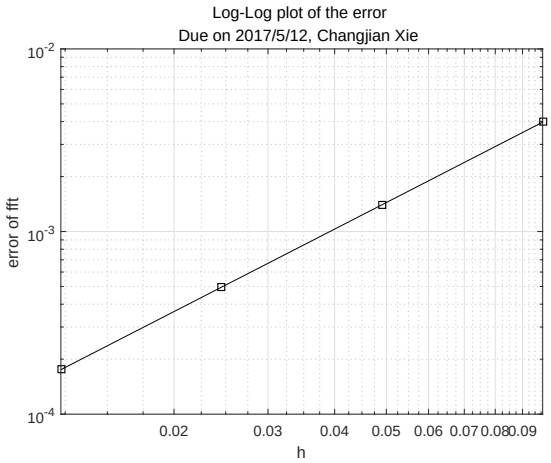

Figure 2: Using the values $n = 2^L$, where $L = 6, 7, 8, 9$, and $h = \frac{2\pi}{n}$. Do a Log-Log plot of the error.

The result of error is as follows,

h	error
0.098174770424681	0.003970390361341
0.049087385212341	0.001402338801571
0.024543692606170	0.000495677469557
0.012271846303085	0.000175237330099

We obtain the slope of straight line as follows using Least Square Method (LSM), that is, the order of error is that

α_1
2.025709101424918

Remark

If we have only the data $\{(x_j, y_j)\}_{j=0}^{2m-1}$, and we don't know the exact analytical solution. We can do the interpolating trigonometric polynomial using FFT, the following is the detail.

We need to transform $[0, 1]$ to $[-\pi, \pi]$ by $z_j = 2\pi(x_j - \frac{1}{2})$, where $x_j \in [0, 1]$, $z_j \in [-\pi, \pi]$, then we get point $\{z_j, u(\frac{1}{2} + \frac{z_j}{2\pi})\}$. We observe that $x_j = \frac{j}{n}$, $z_j = -\pi + (\frac{2j}{n})\pi$, so it doesn't matter change z_j to x_j, i.e., $x_j = -\pi + (\frac{j}{m})\pi$, $j = 0, 1, 2, \ldots, 2m-1$, then we transform $[0, 1]$ to the interval $[-\pi, \pi]$, we get

$$u_{2m} = u(x_{2m}) = u(\pi) = 1 = u_0 = 1,$$

$$u_{-1} = u(x_{-1}) = u(-\pi + \frac{-1}{m}\pi) = \cosh(\sin(\frac{-1}{m})),$$

$$u_{2m-1} = u(x_{2m-1}) = \cosh(\sin(2\pi - \frac{1}{m}\pi)),$$

$$y_j = u(x_j) = \cosh(\sin(\frac{j}{m}\pi)),$$

where $x_j \in [-\pi, \pi]$, $j = 0, 1, \ldots, 2m-1$, $h = x_{j+1} - x_j = \frac{1}{m}\pi = \frac{2\pi}{n}$. For these data $\{(x_j, y_j)\}_{j=0}^{2m-1}$, we determine the form of the discrete least squares polynomial of degree n on the $2m$ data points $\{(x_j, y_j)\}_{j=0}^{2m-1}$, where $x_j = -\pi + (\frac{j}{m})\pi$, for each $j = 0, 1, 2, \ldots, 2m-1$. The interpolating trigonometric polynomial in T_m on these $2m$ data points is nearly

the same as the least squares polynomial. This is because the least squares trigonometric polynomial minimizes the error term

$$E(S_m) = \sum_{j=0}^{2m-1} (y_j - S_m(x_j))^2,$$

and for the interpolating trigonometric polynomial, this error is 0, hence minimized, when the $S_m(x_j) = y_j$, for each $j = 0, 1, \ldots, 2m-1$. This requires the interpolating polynomial to be written as

$$S_m(x) = \frac{a_0 + a_m \cos mx}{2} + \sum_{k=0}^{m-1} (a_k \cos kx + b_k \sin kx),$$

if we want the form of the constants a_k and b_k to agree with those of the discrete least squares polynomial; that is,

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos kx_j, \quad \text{for } k = 0, 1, \ldots, m,$$

$$b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \sin kx_j, \quad \text{for } k = 1, \ldots, m-1.$$

Instead of directly evaluating the constants a_k and b_k, the fast Fourier transform procedure computes the complex coefficients c_k in

$$\frac{1}{m} \sum_{k=0}^{2m-1} c_k e^{ikx},$$

where

$$c_k = \sum_{j=0}^{2m-1} y_j e^{\frac{ik\pi j}{m}}, \quad k = 0, 1, \ldots, 2m-1, \tag{7}$$

i.e. $y_j = u_j = \sum_{j=0}^{2m-1} c_k e^{\frac{-ik\pi j}{m}}$. Once the constants c_k have been determined, a_k and b_k can be recovered by using Eulers Formula,

$$e^{iz} = \cos z + i \sin z.$$

For each $k = 0, 1, \ldots, m$, we have

$$\begin{aligned}\frac{1}{m}c_k(-1)^k &= \frac{1}{m}c_k e^{-i\pi k} = \frac{1}{m}\sum_{j=0}^{2m-1} y_j e^{\frac{ik\pi j}{m}} e^{-i\pi k} \\ &= \frac{1}{m}\sum_{j=0}^{2m-1} y_j e^{ik(-\pi+\frac{\pi j}{m})} \\ &= \frac{1}{m}\sum_{j=0}^{2m-1} y_j(\cos k(-\pi+\frac{\pi j}{m}) + i\sin k(-\pi+\frac{\pi j}{m})) \\ &= \frac{1}{m}\sum_{j=0}^{2m-1} y_j(\cos kx_j + i\sin kx_j).\end{aligned}$$

So, given c_k we have

$$a_k + ib_k = \frac{(-1)^k}{m}c_k. \tag{8}$$

For notational convenience, b_0 and b_m are added to the collection, but both are 0 and do not contribute to the resulting sum. The operation-reduction feature of the fast Fourier transform results from calculating the coefficients c_k in clusters, and uses as a basic relation the fact that for any integer n,

$$e^{n\pi i} = \cos n\pi + i\sin n\pi = (-1)^n.$$

Suppose $m = 2^p$ for some positive integer p. For each $k = 0, 1, \ldots, m?1$ we have

$$c_k + c_{m+k} = \sum_{j=0}^{2m-1} y_j e^{\frac{ik\pi j}{m}} + \sum_{j=0}^{2m-1} y_j e^{\frac{i(m+k)\pi j}{m}} = \sum_{j=0}^{2m-1} y_j e^{\frac{ik\pi j}{m}}(1 + e^{\pi ij}),$$

but

$$1 + e^{i\pi j} = \begin{cases} 2, & \text{if } j \text{ is even,} \\ 0 & \text{if } j \text{ is odd,} \end{cases}$$

so there are only m nonzero terms to be summed. If j is replaced by $2j$ in the index of the sum, we can write the sum as

$$c_k + c_{m+k} = 2\sum_{j=0}^{m-1} y_{2j} e^{\frac{ik\pi(2j)}{m}},$$

that is

$$c_k + c_{m+k} = 2\sum_{j=0}^{m-1} y_{2j} e^{\frac{ik\pi j}{\frac{m}{2}}}. \tag{9}$$

In a similar manner,

$$c_k - c_{m+k} = 2e^{\frac{ik\pi}{m}} \sum_{j=0}^{m-1} y_{2j+1} e^{\frac{ik\pi j}{\frac{m}{2}}}. \tag{10}$$

There are $2m$ coefficients $c_0, c_1, \ldots, c_{2m-1}$ to be calculated. Using the basic formula (7) requires $2m$ complex multiplications per coefficient, for a total of $(2m)^2$ operations. Equation (9) requires m complex multiplications for each $k = 0, 1, \ldots, 2m-1$ and (10) requires $m+1$ complex multiplications for each $k = 0, 1, \ldots, m-1$. Using these equations to compute $c_0, c_1, \ldots, c_{2m-1}$ reduces the number of complex multiplications from $(2m)^2 = 4m^2$ to

$$m \cdot m + m(m+1) = 2m^2 + m.$$

The sums in (9) and (10) have the same form as the original and m is a power of 2, so the reduction technique can be reapplied to the sums in (9) and (10). Each of these is replaced by two sums from $j = 0$ to $j = \frac{m}{2} - 1$. This reduces the $2m^2$ portion of the sum to

$$2[\frac{m}{2} \cdot \frac{m}{2} + \frac{m}{2} \cdot (\frac{m}{2} + 1)] = m^2 + m.$$

So a total of

$$(m^2 + m) + m = m^2 + 2m$$

complex multiplications are now needed, instead of $(2m)^2$.

Applying the technique one more time gives us 4 sums each with $\frac{m}{4}$ terms and reduces the m^2 portion of this total to

$$4[(\frac{m}{4})^2 + \frac{m}{4}(\frac{m}{4} + 1)] = \frac{m^2}{2} + m,$$

for a new total of $\frac{m^2}{2} + 3m$ complex multiplications. Repeating the process r times reduces the total number of required complex multiplications to

$$\frac{m^2}{2^{r-2}} + mr.$$

The process is complete when $r = p+1$, because we then have $m = 2^p$ and $2m = 2^{p+1}$. As a consequence, after $r = p + 1$ reductions of this type, the number of complex multiplications is reduced from $(2m)^2$ to

$$\frac{(2^p)^2}{2^{p-1}} + m(p+1) = 2m + pm + m = 3m + m\log_2 m = O(m\log_2 m).$$

Because of the way the calculations are arranged, the number of required complex additions is comparable.

Consider the following algorithm of Fast Fourier Transform:

Fast Fourier Transform

To compute the coefficients in the summation

$$\frac{1}{m}\sum_{k=0}^{2m-1} c_k e^{ikx} = \frac{1}{m}\sum_{k=0}^{2m-1} c_k(\cos kx + i\sin kx), \text{ where } i = \sqrt{-1},$$

for the data $\{(x_j, y_j)\}_{j=0}^{2m-1}$, where $m = 2^p$, and $x_j = -\pi + \frac{j\pi}{m}$ for $j = 0, 1, \ldots, 2m-1$:

INPUT $m, p; y_1, y_2, \ldots, y_{2m-1}$.

OUTPUT complex numbers $c_0, \ldots, c_{2m-1}$; real numbers $a_0, \ldots, a_m; b_1, \ldots, b_{m-1}$.

Step 1 Set

$$\begin{aligned} M &= m; \\ q &= p; \\ \varsigma &= e^{\frac{\pi i}{m}}. \end{aligned}$$

Step 2 For $j = 0, 1, \ldots, 2m-1$, set $c_j = y_j$.

Step 3 For $j = 1, \ldots, M$, set

$$\begin{aligned} \xi_j &= \zeta^j; \\ \xi_{j+m} &= -\xi_j. \end{aligned}$$

Step 4 Set $K = 0$; $\xi_0 = 1$.

Step 5 For $L = 1, 2, \ldots, p+1$ do Steps $6 - 12$.

Step 6 While $K < 2m - 1$ do Steps $7 - 11$.

Step 7 For $j = 1, 2, \ldots M$ do Steps $8 - 10$.

Step 8 Let $K = k_p \cdot 2^p + k_{p-1} \cdot 2^{p-1} + \cdots + k_1 \cdot 2 + k_0$;
(Decompose k).
set

$$\begin{aligned} K_1 &= \frac{K}{2^q} = k_p \cdot 2^{p-q} + \cdots + k_{q+1} \cdot 2 + k_q; \\ K_2 &= k_q \cdot 2^p + k_{q+1} \cdot 2^{p-1} + \cdots + k_p \cdot 2^q. \end{aligned}$$

Step 9 Set

$$\begin{aligned} \eta &= c_{K+M} \xi_{K_2}; \\ c_{K+M} &= c_K - \eta; \\ c_K &= c_K + \eta. \end{aligned}$$

Step 10 Set $K = K + 1$.

Step 11 Set $K = K + M$.

Step 12 Set

$$\begin{aligned} K &= 0; \\ M &= \frac{M}{2}; \\ q &= q - 1. \end{aligned}$$

Step 13 While $K < 2m - 1$ do Steps $14 - 16$.

Step 14 Let $K = k_p \cdot 2^p + k_{p-1} \cdot 2^{p-1} + \cdots + k_1 \cdot 2 + k_0$; (Decompose k).
set $j = k_0 \cdot 2^p + k_1 \cdot 2^{p-1} + \cdots + k_{p-1} \cdot 2 + k_p$.

Step 15 If $j > K$ then interchange c_j and c_k.

Step 16 Set $K = K + 1$.

Step 17 Set

$$\begin{aligned} a_0 &= \frac{c_0}{m}; \\ a_m &= \mathrm{Re}(e^{-i\pi m}\frac{c_m}{m}). \end{aligned}$$

Step 18 For $j = 1, \ldots, m-1$ set

$$\begin{aligned} a_j &= \mathrm{Re}(e^{-i\pi j}\frac{c_j}{m}); \\ b_j &= \mathrm{Im}(e^{-i\pi j}\frac{c_j}{m}). \end{aligned}$$

Step 19 OUTPUT$(c_0, \ldots, c_{2m-1}; a_0, \ldots, a_m; b_1, \ldots, b_{m-1})$; STOP.