Coursework report presented for
M4N9

Computational Linear Algebra
Project 1

Yadu *Bhageria*

Mathematics Department
Imperial College London
United Kingdom

November 28, 2016

Contents

1 Q1

The linear system to be solved is

$$Au_N = f_N$$

This can be done by QR factorising A by using the householder triangularization method (given in the house.m file). The output gives a matrix W that contains the Householder reflections and another matrix that is upper triangular, R. From this we can compute the matrix, Q. This gives

$$\begin{aligned} A &= QR \\ Au_N &= QRu_N \\ Ru_N &= Q^\star f_N \\ Ru_N &= b \end{aligned}$$

which can be solved using backwards substitution. Note the matrix Q never has to be computed explicitly from W as we can directly compute $Q^\star f_N$ using W and f_N.

1.1 Algorithm

So the algorithm used can be written as

```
$[m, n] = size(A)$                          ▷ Compute size of A matrix
$[W, R] = \text{house}(A)$                  ▷ Compute householder reflection and R matrices using house.m
for $k = 1 : n$ do                          ▷ Compute $Q^\star b$
    $b_{k:m} = b_{k:m} - 2Wb_{k:n,k}(W^\star_{k:n,k} b_{k:m})$
end for
for $i = n : -1 : 1$ do                     ▷ Compute $R^\star b$, i.e. backwards substitution
    $b_i = \frac{b_i - R_{i,i+1:n} b_{i+1:n}}{R_{i,i}}$
end for
```

1.2 Results

Using this code below is the outputted table of errors from the MATLAB code.

```
Error_Table =

     N        Error
    ---    ----------
     16    0.022955
     32    0.0057055
     64    0.0014243
    128    0.00035595
```

Dividing the errors by $\Delta^2 x$ it can be seen that the errors are indeed decreasing $\sim \Delta^2 x$. This gives

```
ScaledError_Table =

     N      Scaled_Error
    ---     ------------
     16     5.8764
     32     5.8424
     64     5.8340
    128     5.8318
```

2 Q2

2.1 Showing given eigenvectors work and finding corresponding eigenvalues

Eigenvectors z_k with corresponding eigenvalues λ_k of D_2 will satisfy the following eigenvector-eigenvalue equation.

$$D_2 z_k = \lambda_k z_k \tag{$\star$1}$$

where $z_k \in \mathbb{C}^N$ and $\lambda_k \in \mathbb{C}$. We can write Eq. ($\star$1) using the given Eq. (2) as

$$\frac{z_{k,j+1} - 2z_{k,j} + z_{k,j-1}}{\Delta x^2} = \lambda_k z_{k,j} \quad \text{for } j = 1 \ldots N \tag{$\star$2}$$

Note $z_{k,j}$ corresponds to the j^{th} entry in the k^{th} eigenvector of D_2. Also note that the indices loop around, i.e. $z_{k,1} = z_{k,N+1}$ and $z_{k,0} = z_{k,N}$.

Assume

$$z_k = \frac{1}{\sqrt{N}} \begin{pmatrix} e^{i(k-1)x_1} \\ \vdots \\ e^{i(k-1)x_N} \end{pmatrix}$$

Combining this and $\Delta x = \frac{2\pi}{N}$ and $x_j = (j-1)\Delta x = \frac{2\pi}{N}(j-1)$, the LHS of Eq. ($\star$2) gives

$$\begin{aligned}
\frac{z_{k,j+1} - 2z_{k,j} + z_{k,j-1}}{\Delta x^2} &= \frac{e^{i(k-1)x_{j+1}} - 2e^{i(k-1)x_j} + e^{i(k-1)x_{j-1}}}{\Delta x^2 \sqrt{N}} \\
&= \frac{e^{i(k-1)(j)\frac{2\pi}{N}} - 2e^{i(k-1)(j-1)\frac{2\pi}{N}} + e^{i(k-1)(j-2)\frac{2\pi}{N}}}{\Delta x^2 \sqrt{N}} \\
&= \frac{(e^{i\frac{2\pi}{N}})^{(k-1)(j)} - 2(e^{i\frac{2\pi}{N}})^{(k-1)(j-1)} + (e^{i\frac{2\pi}{N}})^{(k-1)(j-2)}}{\Delta x^2 \sqrt{N}}
\end{aligned}$$

By setting $\omega = e^{i\frac{2\pi}{N}}$ it simplifies to

$$\frac{z_{k,j+1} - 2z_{k,j} + z_{k,j-1}}{\Delta x^2} = \frac{\omega^{(k-1)(j)} - 2\omega^{(k-1)(j-1)} + \omega^{(k-1)(j-2)}}{\Delta x^2 \sqrt{N}}$$

Furthermore by setting $\gamma = \omega^{(k-1)} = e^{i\frac{2\pi}{N}(k-1)}$ it follows that

$$\begin{aligned}\frac{z_{k,j+1} - 2z_{k,j} + z_{k,j-1}}{\Delta x^2} &= [\gamma - 2 + \gamma^{-1}]\frac{\gamma^{(j-1)}}{\Delta x^2\sqrt{N}} \\ &= [e^{i(k-1)\frac{2\pi}{N}} - 2 + e^{-i(k-1)\frac{2\pi}{N}}]\frac{e^{i(k-1)(j-1)\frac{2\pi}{N}}}{\Delta x^2\sqrt{N}} \\ &= \frac{[2\cos((k-1)\frac{2\pi}{N}) - 2]}{\Delta x^2}\frac{e^{i(k-1)(j-1)\frac{2\pi}{N}}}{\sqrt{N}} \\ &= \frac{2[\cos((k-1)\frac{2\pi}{N}) - 1]}{\Delta x^2} z_{k,j} \\ &= \lambda_k z_{k,j}\end{aligned}$$

Thus satisfying Eq. ($\star$2) proving that the assumed z_k's are indeed eigenvectors of D_2 with corresponding eigenvalues λ_k given by

$$\lambda_k = \frac{2[\cos((k-1)\frac{2\pi}{N}) - 1]}{\Delta x^2}$$

2.2 Particular case $k = 1$

For the particular case when $k = 1$

$$z_1 = \frac{1}{\sqrt{N}}\begin{pmatrix} e^{i(0)x_1} \\ \vdots \\ e^{i(0)x_N}\end{pmatrix} = \frac{1}{\sqrt{N}}\begin{pmatrix}1 \\ \vdots \\ 1\end{pmatrix} \text{ and } \lambda_1 = \frac{2[\cos(0) - 1]}{\Delta x^2} = 0$$

Thus the LHS of Eq. ($\star$1) gives

$$z_1 D_2 = \frac{1}{\sqrt{N}}\begin{pmatrix}1 \\ \vdots \\ 1\end{pmatrix} D_2 = \frac{1}{\sqrt{N}}\begin{pmatrix}1-2+1 \\ \vdots \\ 1-2+1\end{pmatrix} = \mathbf{0}$$

the RHS of Eq. ($\star$1) gives

$$\lambda_1 z_1 = 0 \cdot z_1 = \mathbf{0}$$

Since LHS = RHS, we have that these satisfy the eigenvalue-eigenvector equation.

3 Q3

Consider two of the eigenvectors of D_2, z_k and z_l. Then

$$\begin{aligned}
\langle z_k, z_l \rangle &= z_k^\star z_l \\
&= \frac{1}{N}(\mathrm{e}^{-i(k-1)x_1}, \dots, \mathrm{e}^{-i(k-1)x_N}) \begin{pmatrix} \mathrm{e}^{i(l-1)x_1} \\ \vdots \\ \mathrm{e}^{i(l-1)x_N} \end{pmatrix} \\
&= \frac{1}{N}\sum_{j=1}^{N} \mathrm{e}^{-i(k-1)x_j}\mathrm{e}^{i(l-1)x_j} \\
&= \frac{1}{N}\sum_{j=1}^{N} \mathrm{e}^{i(l-k)x_j} \\
&= \frac{1}{N}\sum_{j=1}^{N} \mathrm{e}^{i\frac{2\pi}{N}(j-1)(l-k)}
\end{aligned}$$

Again setting $\omega = \mathrm{e}^{i\frac{2\pi}{N}}$ it follows that

$$\begin{aligned}
&= \frac{1}{N}\sum_{j=1}^{N} \omega^{(j-1)(l-k)} \\
&= \frac{1}{N}[1 + \omega^{(l-k)} + \omega^{2(l-k)} + \dots + \omega^{(N-1)(l-k)}] \\
&= \begin{cases} \frac{1}{N}N & \text{if } k = l \\ \frac{1}{N}[\frac{\omega^{N(l-k)}-1}{\omega^{(l-k)}-1}] & \text{if } k \neq l \end{cases}
\end{aligned}$$

Note that $\omega^{N(l-k)} = \mathrm{e}^{i\frac{2\pi}{N}N(l-k)} = \mathrm{e}^{i2\pi(l-k)} = 1 \quad \because (l-k) \in \mathbb{Z}$. And so

$$\langle z_k, z_l \rangle = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{if } k \neq l \end{cases}$$

Thus $\{z_k\}$ form an orthonormal basis with respect to the standard inner product.

4 Q4

The equation to be solved is

$$-\frac{d^2u}{dx^2} = f(x)$$

which using the second order difference operator, D_2, for a periodic function $f(x)$ can be written as a linear system of equations of the form

$$-D_2 u_N = f_N \tag{$\star$3}$$

This however does not enforce the zero-mean condition by itself. Thus starting with the decomposition

$$D_2 = Z\Lambda Z^{-1}$$

and noting that

$$Z^{-1} = Z^\star \quad \because \langle z_i, z_j\rangle = \delta_{ij} \implies Z^\star Z = I$$

Eq. ($\star$3) can be written as

$$\begin{aligned} -D_2 u_N &= -Z\Lambda Z^\star u_N = f_N \\ \implies \Lambda Z^\star u_N &= -Z^\star f_N \end{aligned}$$

Now writing

$$u^{(1)} = Z^\star u_N \quad \text{and} \quad b^{(1)} = Z^\star f_N$$

Eq. ($\star$3) gives

$$\Lambda u^{(1)} = -b^{(1)} \tag{$\star$4}$$

where $u_1^{(1)} = z_1 u_N = \frac{1}{\sqrt{N}}\Sigma_{i=1}^N u_i = 0$ due to the zero-mean condition and similarly $b_1^{(1)} = z_1 f_N = \frac{1}{\sqrt{N}}\Sigma_{i=1}^N f_i = 0$.

This means that the first equation from linear system of equations given by Eq. ($\star$4), when enforcing the zero-mean condition, is stating

$$\lambda_1 u_1^{(1)} = \lambda_1 \cdot 0 = 0 = b_1^{(1)}$$

This is redundant and thus we can simply ignore it. and solve the other $N-1$ equations.

This can be done by writing

$$\begin{aligned}\Lambda^{(-1)} &= diag(0, \frac{1}{\lambda_2}, \frac{1}{\lambda_3}, \dots, \frac{1}{\lambda_N}) \\ \implies u^{(1)} &= -\Lambda^{(-1)} b^{(1)} = -\Lambda^{(-1)} Z^\star f_N \\ \implies Z^\star u_N &= -\Lambda^{(-1)} Z^\star f_N \\ \implies u_N &= -Z\Lambda^{(-1)} Z^\star f_N\end{aligned}$$

4.1 Algorithm

Given the Z matrix, eigenvalues $\{\lambda_k\}_{k=1:N}$ and the vector f containing values for f_N

```
$u = Z^\star f$
**for** $k = 2 : N$ **do**
    $u_k = u_k / \lambda_k$
**end for**
$u = -Zu$
```

4.2 Results

This method gives almost exactly the same error as the previous method.

```
Error_Table =

     N              Error              Real_Error
    ---     ----------------------     ----------

     16     0.022955   - 2.2332e-18i    0.022955
     32     0.0057055  + 6.8926e-18i    0.0057055
     64     0.0014243  - 1.0167e-17i    0.0014243
    128     0.00035595 - 4.8256e-19i    0.00035595
```

So the error is the same for the real part and there a perturbation on the order of ϵ_m.

Dividing the errors by $\Delta^2 x$ it can be seen that the errors are indeed decreasing $\sim \Delta^2 x$. This gives

```
ScaledError_Table =

     N         Scaled_Error       Real_Scaled_Error
    ---    ------------------     -----------------

     16    5.8764 - 5.7169e-16i     5.8764
     32    5.8424 + 7.058e-15i      5.8424
     64    5.834  - 4.1646e-14i     5.834
    128    5.8318 - 7.9062e-15i     5.8318
```

5 Q5

The backwards stability conditions that need to be satisfied by the algorithm in part 4 are:

1. For the first step of $b^{(1)} = Z^\star f$, the condition is:

$$(Z + \delta Z)\tilde{b}^{(1)} = f$$

with $\delta Z = Z - \tilde{Z}$ where $\|\delta Z\| = \mathcal{O}(\epsilon)$. Note that since Z is unitary, $\|Z\| = 1$.

2. For the next step of $u_i^{(1)} = b_i^{(1)} / \lambda_i$ for $i = 2 : N$

$$(\lambda_i + \delta\lambda_i)\tilde{u}_i^{(1)} = \tilde{b}_i^{(1)}$$

with $\delta\lambda_i = \lambda_i - \tilde{\lambda}_i$ where $\frac{\|\delta\lambda_i\|}{\|\lambda_i\|} = \mathcal{O}(\epsilon)$ for $i = 2 : N$.

3. For the final step of $u_N = -Z^\star u^{(1)}$

$$(Z^\star + \delta Z^\star)\tilde{u}_N = -\tilde{u}^{(1)}$$

with $\delta Z^\star = Z^\star - \tilde{Z}^\star$ where $\|\delta Z^\star\| = \mathcal{O}(\epsilon)$.

Note that the fundamental axion of floating point operations is also valid for complex values numbers with a slightly enlargement adjustment to the value of ϵ_m [1]. Thus the analysis of complex valued numbers can proceed in exactly the same way as that for real valued numbers.

Step (1) is backwards stable as it is simply a matrix-vector multiplications and such a step also occurs in QR factorization.

Step (2) is also backwards stable as it simply a floating point operation of a real number with a complex number. Furthermore it is worth noting that $0 < |\lambda_i| \leq \frac{4}{\Delta x^2}$ for $i = 2 : N$. So there are no singularities involved and λ_i are bounded in magnitude.

Step (3) is also backwards stable as it is the negative of a matrix-vector multiplication. We know that matrix-vector multiplication is backwards stable and taking the negative of values is not a FLOP and thus is also backwards stable.

In lectures it is shown that the steps involved in using QR factorization to solve a system of equations are backwards stable then therefore so is the entire algorithm. Thus since each of the above steps are backwards stable for the algorithm in Q4 then the entire algorithm should also be backwards stable.

[1] Numerical Linear Algebra - Trefethen and Bau

6 Q6

In pursuit of clarity it is worth mentioning that the operation counts for setting up the problem, computing Δx, x, f_N, A, Z, and $\{\lambda_k\}_{k=1:N}$ are not included in the following subsections.

Also, complex numbers can be treated as 2 real numbers and thus addition takes 2 FLOPs and multiplication takes 6 FLOPs $\epsilon_{\text{machine}}$.[2]

6.1 Part 1 Operation Count

1. From lectures we know that the leading order terms of the number of operations needed for the householder algorithm for a matrix $A \in \mathbb{R}^{m\times n}$ is $2mn^2 - \frac{2}{3}n^3$. In this case the operation count is simplified to $2N^3 - \frac{2}{3}N^3 = \frac{4}{3}N^3$ as $N = n = m$.

2. $2N^2 + 2N$ FLOPs are needed in the calculation of $Q^{\star}b$. This is because $2(N-k+1)$ multiplications are needed and $N-k+1$ additions and subtractions are needed for $k = 1 : N$. i.e.
$$\sum_{k=1}^{N} 4(N-k+1) = 2N^2 + 2N$$

3. N^2 FLOPs are needed for the calculation of $R *^{\star} b$ i.e. backwards substitution. This is because $N-k$ subtractions and multiplications are needed and 1 division. i.e.
$$\sum_{k=1}^{N} 2(N-k) + 1 = N^2$$

In total this gives

$$(\frac{4}{3}N^3) + (2N^2 + 2N) + (N^2) = \frac{4}{3}N^3 + 3N^2 + 2N = \frac{4}{3}N^3 + \mathcal{O}(N^2)$$

6.2 Part 4 Operation Count

1. $4N^2 - 2N$ FLOPs are needed to compute $u^{(1)} = Z^{-1}f_N$. This is becasue there are $2N$ multiplications between a complex and real number and $2(N-1)$ additions ($N-1$ real and complex ones respectively) for each value in the array which is of size N. i.e.
$$\sum_{k=1}^{N} 4N - 2 = 4N^2 - 2N$$

2. $2N - 2$ FLOPs to compute $u^{(2)} = \Lambda^{-1}u^{(1)}$ for rows from $2/dotsN$ where a complex number is being divided by a real number. i.e.
$$\sum_{k=2}^{N} 2 = 2N - 2$$

[2]Numerical Linear Algebra - Trefethen and Bau

3. $8N^2 - 2N$ FLOPS are needed to compute $u_N = Zu^{(2)}$. 6 computations are needed for a complex and complex number multiplication and there are N such numbers and thus a further $2(N-1)$ additions. This is done N times. i.e.

$$\sum_{k=1}^{N} 2(N-1) + \sum_{j=1}^{N} 6 = \sum_{k=1}^{N} 8N - 2 = 8N^2 - 2N$$

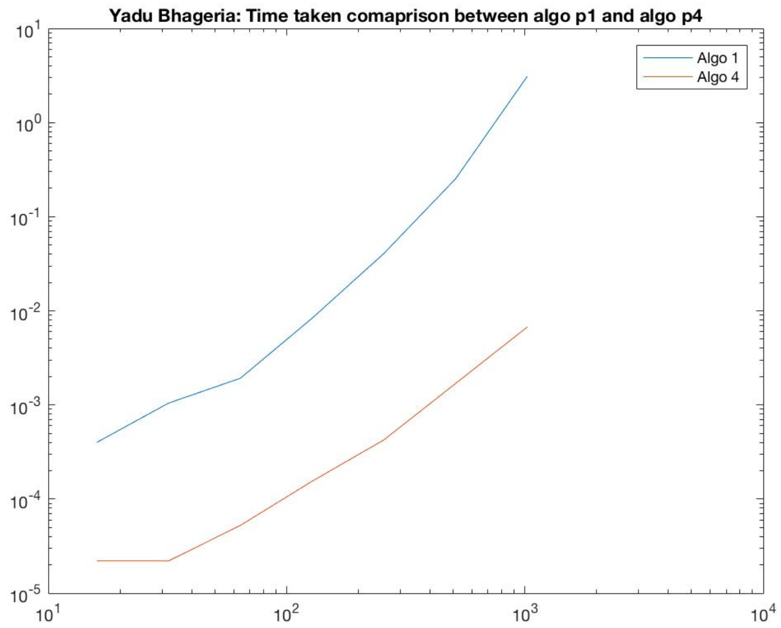
Thus the total number of FLOPS needed are

$$(4N^2 - 2N) + (2N - 2) + (8N^2 - 2N) = 12N^2 - 2N - 2 = 12N^2 + \mathcal{O}(N)$$

6.3 Comparison

By comparing the theoretical number of FLOPs it is clear than in the limit $N \to \infty$ the algorithm in part 4 ($\mathcal{O}(N^2)$) should be much faster than the algorithm in part 1 ($\mathcal{O}(N^3)$).

Using the tic and toc functions in MATLAB I get the following results which agree with the theoretical predictions.



7 Q7

The coefficients in the fft() function are precisely those stored in the $Z^\star$ matrix scaled by a factor of $\frac{1}{\sqrt{N}}$. And the coefficients in the ifft() function are precisely those stored in the Z matrix scaled by a factor of $\sqrt{N}$. Since in the algorithm each of Z and $Z^\star$ are applied once the effect cancels out. It is precisely the same as multiplying by $\frac{1}{\sqrt{N}}$ twice due to applying Z and $Z^\star$ and multiplying by $\frac{1}{N}$ in the ifft() function.

Thus the vector matrix multiplication of $b^{(1)} = Z^\star f_N$ can be replaced by $b^{(1)} = \text{fft}(f_N)$ and the vector matrix multiplication of $u_N = -Zu^{(1)}$ can be replaced by $u_N = -\ \text{ifft}(u^{(1)})$ where $u^{(1)} = \Lambda^{(-1)} b^{(1)} = \Lambda^{(-1)} Z^\star f_N$.

7.1 Algorithm

This gives the new algorithm as

```
$u = \text{fft}(f)$
**for** $k = 2 : N$ **do**
    $u_k = u_k / \lambda_k$
**end for**
$u = -\ \text{ifft}(u)$
```

7.2 Comparison

7.2.1 Error

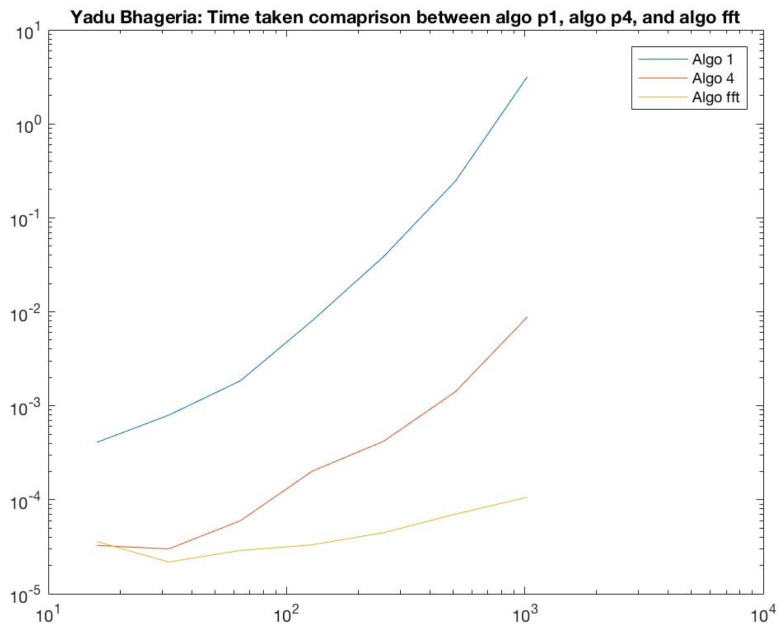
The error remains the same using this method with a perturbation on the order of ϵ_m

```
Error_Table =

     N             Error               Real_Error
    ---     ----------------------     ----------

     16     0.022955   + 3.9011e-30i    0.022955
     32     0.0057055  - 7.8687e-29i    0.0057055
     64     0.0014243  + 7.4363e-18i    0.0014243
    128     0.00035595 - 1.7416e-17i    0.00035595

ScaledError_Table =

     N         Scaled_Error       Real_Scaled_Error
    ---     ------------------    -----------------

     16     5.8764 + 9.9867e-28i     5.8764
     32     5.8424 - 8.0576e-26i     5.8424
     64     5.834  + 3.0459e-14i     5.834
    128     5.8318 - 2.8535e-13i     5.8318
```

7.2.2 Runtime Comparison

Number of operations needed for the Fast Fourier Transform algorithm will be $\mathcal{O}(N \log N)$. fft() and ifft() increase $\mathcal{O}(N \log N)$ as N increases and doing $u_k = u_k / \lambda_k$ increase $\mathcal{O}(N)$ as N increases.

As can be seen, the algorithm utilizing Fast Fourier Transforms is much faster the the previous two algorithms and increases on a smaller order than either of them.



8 Appendix

8.1 Code

All the code used for this project is submitted with it on blackboard. Here I outline what files are needed for each part.

8.1.1 Q1

The algorithm and implementation in Q1 uses the house.m, construct_A.m, algo_p1.m and error_p1.m files. The error_p1.m file is the script that produces the error tables and can also produce a plot of the various estimations.

8.1.2 Q4

The algorithm and implementation in Q4 uses the construct_Z.m, construct_lambda.m, algo_p4.m and error_p4.m files. The error_p4.m file is the script that produces the error tables and can also produce a plot of the various estimations.

8.1.3 Q6

The plot producing file timetaken_p1p4.m calls the time_p1p4algos.m which in turn calls house.m, construct_A.m, algo_p1.m, construct_Z.m, construct_lambda.m, and algo_p4.m files.

8.1.4 Q7

The algorithm and implementation in Q7 uses construct_lambda.m, algo_fft.m and error_fft.m files with the latter being the script producing error tables.

The plot producing file timetaken_all.m calls the time_allalgos.m which in turn calls house.m, construct_A.m, algo_p1.m, construct_Z.m, construct_lambda.m, algo_p4.m, and algo_fft.m files.