ESAM445 HW2 Computing Report

Mingfu Liang, Student ID:3146919, NetID:MLQ4767

May 2019

1. Introduction

In this computing assignment, we are going to solve the Model boundary value problem as follows:

$$-u_{xx} + K\pi^2 u = 0, u(0) = u(1) = 0. \tag{1}$$

The numerical methods using in this assignment is the Multigrid method as descripe in Page 38 in W.L. Briggs book, A Multigrid Tutorial. We are going to reproduce the Fig. 3.5 of the book.

1.1 Determine all nontrivial solutions (eigenfunctions) when $K \neq 0$ is an eigenvalue

When $K \neq 0$, Eq. (1) is a constant coefficient equations and let $u(x) = e^{tx}$ and t is arbitrary constant, then we plug it into Eq. (1) and we get the solution which are

$$u(x) = \begin{cases} c_1 e^{\pi x\sqrt{K}} + c_2 e^{-\pi x\sqrt{K}}, K > 0 \\ c_3 \cos\left(\pi x\sqrt{-K}\right) + c_4 \sin\left(\pi x\sqrt{-K}\right), K < 0 \end{cases}$$

Now based on the boundary conditions, we can deduce that when $K > 0$ there will only have trivial solution. When $K < 0$, we get the nontrivial solutions (eigenfunctions) of Eq. (1) as $u(x) = \sin(n\pi x), n = 1, 2...$ (constant are omitted) and the corresponding eigenvalue $K = -n^2, n = 1, 2, ...$

1.2 Identify the eigenvalues and eigenvectors of the discrete system

Consider the discretized problem of Eq. (1)

$$-\frac{u_{j+1} + u_{j-1} - 2u_j}{h^2} + K\pi^2 u_j = \lambda u_j,\ j = 1, 2, ..., N,\ h = \frac{1}{N+1}$$

where it is understood that $u_0 = u_{N+1} = 0$. This system leads to an $N \times N$ matrix so there can be only N eigenvalues and eigenvectors. To find the eigenvalues and eigenvectors try to micmic the continuous problem by looking at vectors of the form as

$$u_j = \sin(m\pi x_j),\ x_j = jh,\ j = 1, 2, ..., N,\ m = 1, 2, 3, ...$$

where we should expect the upper limit on m to be $m = N$ but this has to be shown. First it is obvious that this guessing eigenvector form can satisfy the boundary conditions in Eq. (1).

Now plug the guessing vectors into the discretized system and we have

$$-\frac{\sin(m\pi h(j+1)) + \sin(m\pi h(j-1)) - 2\sin(m\pi hj)}{h^2} + K\pi^2 \sin(m\pi hj) = \lambda \sin(m\pi hj).$$

Using the basic trig identities that

$$\sin(m\pi h(j \pm 1)) = \sin(m\pi hj)\cos(m\pi h) \pm \cos(m\pi hj)\sin(m\pi h),$$

we have

$$\left[\frac{2}{h^2}\left[1 - \cos(m\pi h)\right] + K\pi^2\right] \sin(m\pi hj) = \lambda \sin(m\pi hj).$$

Therefore we show that $u_j = \sin(m\pi x_j)$, $x_j = jh$, $j = 1, 2, ..., N$, $m = 1, 2, 3, ...$ does in fact define an eigenvector for the discretized problem with eigenvalue

$$\lambda_m = \left[\frac{2}{h^2}\left[1 - \cos(m\pi h)\right] + K\pi^2\right] = \frac{4}{h^2}\sin^2\left(\frac{m\pi h}{2}\right) + K\pi^2$$

using the trig formula that

$$1 - \cos(x) = 2\sin^2\left(\frac{x}{2}\right)$$

2. Reproduction of the implementation of the multigrid method for Eq. (1) when $K = 0$ as discussed in Briggs Book

Now we are going to follow the implementation of the multigrid grid method for Eq. (1) when $K = 0$ as discussed in Briggs. In the book, the relaxation method is weighted Jacobi method with $\omega = \frac{2}{3}$ on a grid with $n = 64$ points. The initial guess should be

$$v_j^h = \frac{1}{2}\left[\sin\left(\frac{16j\pi}{n}\right) + \sin\left(\frac{40j\pi}{n}\right)\right],$$

consisting of the $k = 16$ and $k = 40$ modes and h represents the fine-grid.

2.1 Discuss and explain the computational results presented in each of the figures of Figure 3.5 in Briggs Book

Table 1: 2-norm of the error vector for each step in the Fig. 3.5 in the Book

	Top Left	**Top Right**	**Middle Left**	**Middle Right**	**Bottom Left**	**Bottom Right**
$K = 0$	4.0000	2.2869	1.4740	0.6514	0.3348	0.1368
$K = 2$	4.0000	2.2838	1.4678	0.6497	0.3345	0.1365

As shown in Figure 1, 2, 3, 4, 5 and 6, they are the reproduction of the top left, top right, middle left, middle right, bottom left, bottom right figures in Fig. 3.5 in the Book respectively. From Figure 1 we can see the initial guess with its two modes when $K = 0$, and the 2-norm of the error is 4.0000 as shown in the Table 1 when $K = 0$. As shown in Figure 2, the approximation v^h after one relaxation sweep on the fine grid is superimposed on the initial guess. We can see that much of the oscillatory component of the initial guess has already been removed, and the 2-norm of the error is 2.2869 as shown in the Table 1 when $K = 0$ and the 2-norm of the error has been diminished to 57.10% of the norm of the initial error, which is the same as the one mentioned in the Book. As shown in Figure 3, the approximation after three relaxation sweeps on the fine grid are shown, again superimposed on the initial guess. The solution (in this case, the error) has become smoother and its 2-norm is 1.4740 as shown in the Table 1 when $K = 0$, which is 36.7% of the initial error norm, which is the same as the one mentioned in the Book. Further relaxations on the fine grid would provide only a slow improvement at this point, therefore it's time to move to the coarse grid.

Figure 4 shows the fine-grid error after one relaxation sweep on the coarse-grid residual equation, superimposed on the initial guess. We have achieved another reduction in the error by moving to the coarse grid and the 2-norm of the error is 0.6514 as shown in the Table 1 when $K = 0$, which is 16.24% of the initial error norm. This improvement occurs since the smooth error components, inherited from the fine grid, appear oscillatory on the coarse grid and are quickly removed. This figure is different with the middle right figure in Fig. 3.5 in the Book and we will clarify that it should be the correct figure and the one in the book should be wrong. First let's continue to the Figure 5, it shows the error after three coarse-grid relaxation sweeps and it is the same as the bottom left figure in Fig. 3.5 in the Book. We can see that the oscillatory are effectively reduced and the 2-norm of the error is 0.3348 as shown in the Table 1 when $K = 0$, which is of 8.36% of the initial error. Since the results are completely the same as mentioned in the book and the Figure 4 and 5 are only differ from the relaxation times, therefore we can tell that the middle right figure in Fig. 3.5 in the Book is wrong and the Figure 4 should be the correct one. The discussion about the Figure 5, the bottom left figure in the Fig. 3.5 in the book, will be presented in next subsection.

The coarse-grid approximation to the error is now used to correct the fine-grid approximation. Figure 6 shows the approximations after three additional fine-grid relaxations, the 2-norm of the error is 0.1368 as shown in the Table 1 when $K = 0$, which is 3.41% of the initial error norm.

To sum up, all the reproduction are consistent with the Fig. 3.5 in the Book except the middle right figure in Fig. 3.5 in the Book, and we elaborate that the Figure 4 should be the correct one.

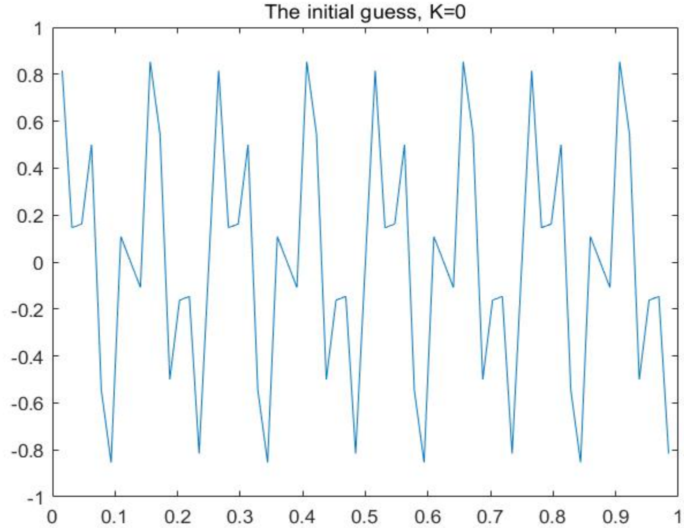

Figure 1: Visualization of the Initial Guess when $K = 0$ as the reproduction of Top Left subfigure of Figure 3.5 in Briggs

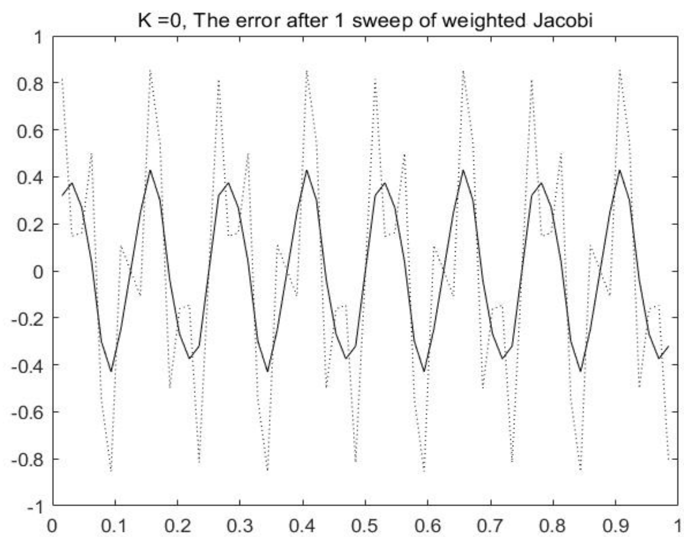

Figure 2: The error after one sweep of weighted Jacobi as the reproduction of Top Right subfigure of Figure 3.5 in Briggs

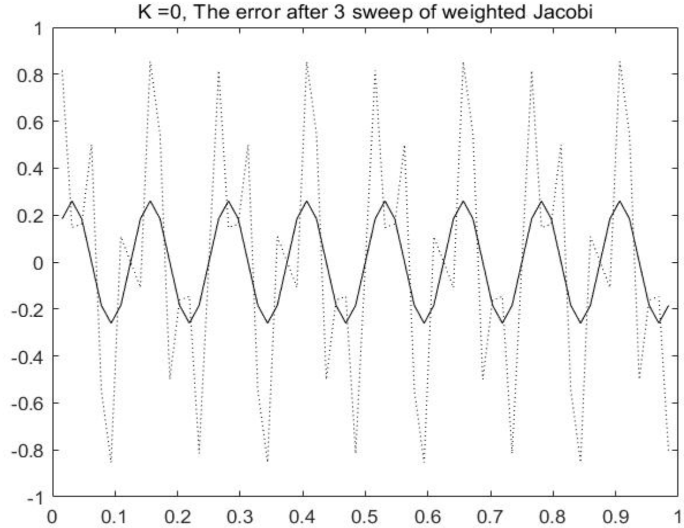

Figure 3: The error after three sweep of weighted Jacobi as the reproduction of Middle Left subfigure of Figure 3.5 in Briggs

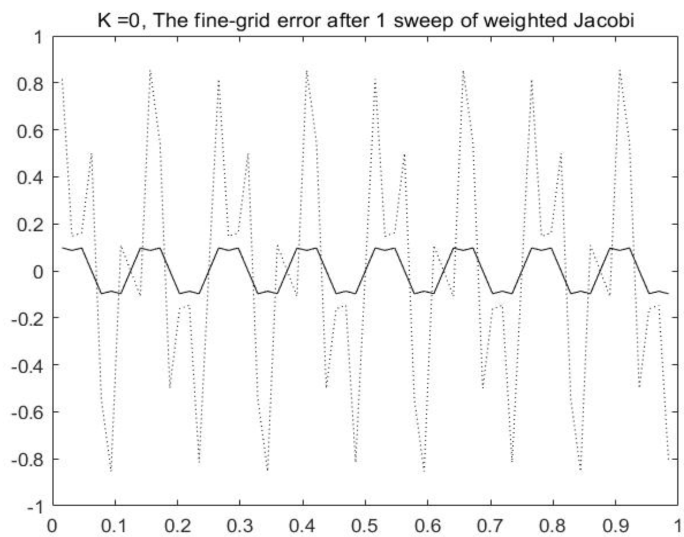

Figure 4: The fine-grid error after one sweep of weighted Jacobi on the coarse-grid problem as the reproduction of Middle Right subfigure of Figure 3.5 in Briggs. Note that here the reproduction is different from the one in Figure 3.5.

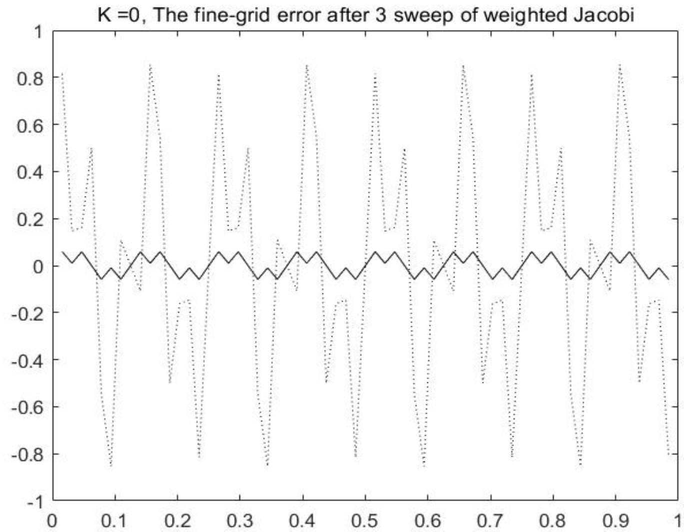

Figure 5: The fine-grid error after three sweep of weighted Jacobi on the coarse-grid problem as the reproduction of Bottom left subfigure of Figure 3.5 in Briggs.

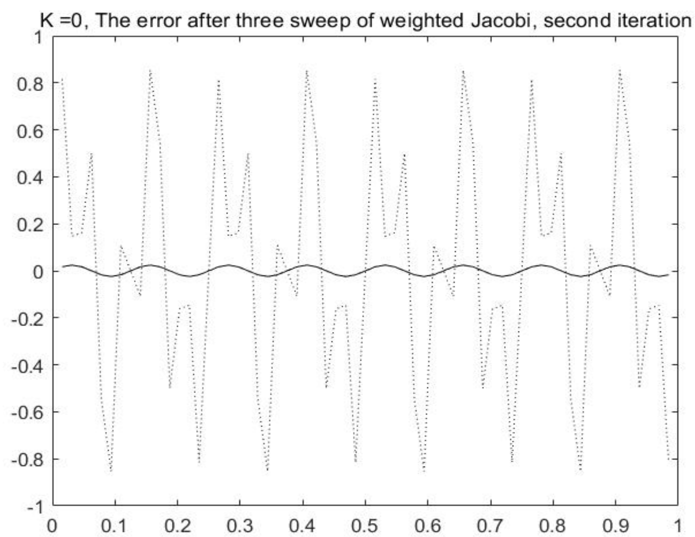

Figure 6: The fine-grid error after the coarse-grid correction is followed by three weighted Jacobi sweeps on the fine grid as the reproduction of Bottom Right subfigure of Figure 3.5 in Briggs.

2.2 Explain the results presented in his bottom left figure.

In the class, the bottom left figure seems wrong since we thought the Multigrid method should make the approximations smoother but we see the approximation become more oscillatory then before. However, in my opinion, I think this numerical result is correct and I will explain it in detailed. First I need to mention that the initial guess we used in this example consists with two modes, $k = 16$ and $k = 40$. Since we implement the multigrid method on a grid with $n = 64$ points, which means the second mode $k = 40 > 32 = \frac{n}{2}$. As mentioned on the page 32 in the Book, fine-grid modes with $k > \frac{n}{2}$ undergo a more curious transformation that the kth mode on Ω^h becomes the $(n-k)$th mode on Ω^{2h} when

$k > \frac{n}{2}$. In other words, the oscillatory modes of Ω^h are misrepresented as relatively smooth modes on Ω^{2h}. Therefore, when we do relaxation on the coarse-grid residual equation, the relatively smooth mode in the fine-grid, $k = 16$, will become more oscillatory in coarse-grid and its error be effectively reduced, while the the relatively oscillatory mode in the fine-grid, $k = 40$, will be misrepresented as relatively smooth modes and its error will be not much reduced. Therefore the approximations will be dominated by the $k = 40$ mode and become more oscillatory. This explain why in the Figure 5, the bottom left figure in Fig. 3.5 in the book, is more oscillatory.

To further illustrate my opinion, first I visualize the two modes in the initial guess in Figure 7, which we can see that the mode $k = 40$ is more oscillatory than mode $k = 16$. Then I visualize the error after three sweeps of weigthed Jacobi relaxation in the fine-grid and the error is superimposed to the smooth mode $k = 16$ and oscillatory mode $k = 40$ as shown in Figure 8 and 9 respectively. We can see that after three relaxation in the fine-grid, the shape of approximations are very similar to the shape of the smooth mode $k = 16$. This is correct to what we expect since in the fine-grid the error of the oscillatory mode $k = 40$ should be effectively reduce. Then we transfer to the coarse-grid and do three relaxation on the residual equation. I also visualize the error after three sweeps of weighted Jacobi in the coarse-grid and the error is superimposed to the smooth mode $k = 16$ and oscillatory mode $k = 40$ as shown in Figure 10 and 11 respectively. We can see that the shape of the approximations is not consistent with the shape of the smooth mode $k = 16$. In particular we can see some opposite peaks and valleys in the Figure 10 between the approximations and the smooth mode $k = 16$. Additionally, from Figure 11 we can see that those opposite peaks and valleys is consistent with the peaks and valleys of the oscillatory mode $k = 40$ with a little skewing, which means they do not totally match but the trend are the same. It implies that the opposite peaks and valleys are due to the effect of the oscillatory mode $k = 40$ since we know that in the coarse-grid relaxation the oscillatory mode $k = 40$ will be relatively smooth and can not be effectively reduce.

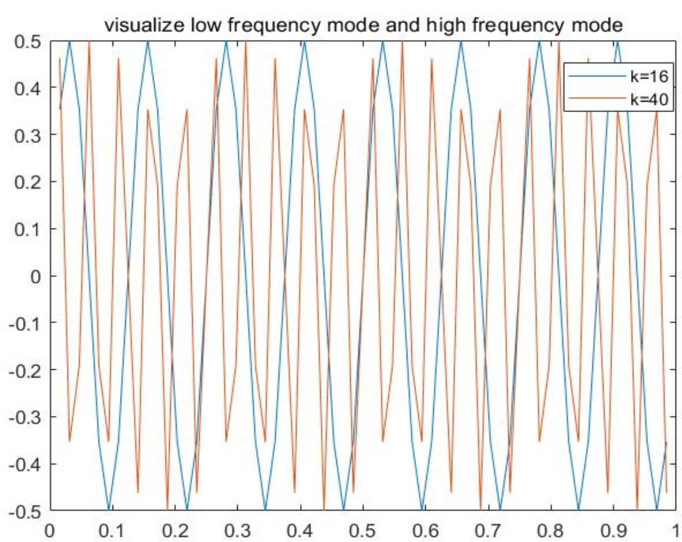

Figure 7: Visualization of the oscillatory mode $k = 40$ and the smooth mode $k = 16$ in the initial guess

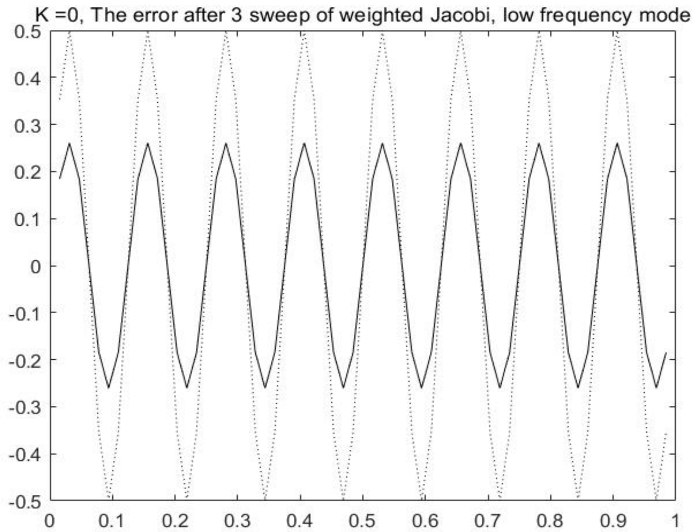

Figure 8: Visualization of the error after three sweeps of weighted Jacobi and also the smooth mode $k = 16$ in the initial guess in the same figure.

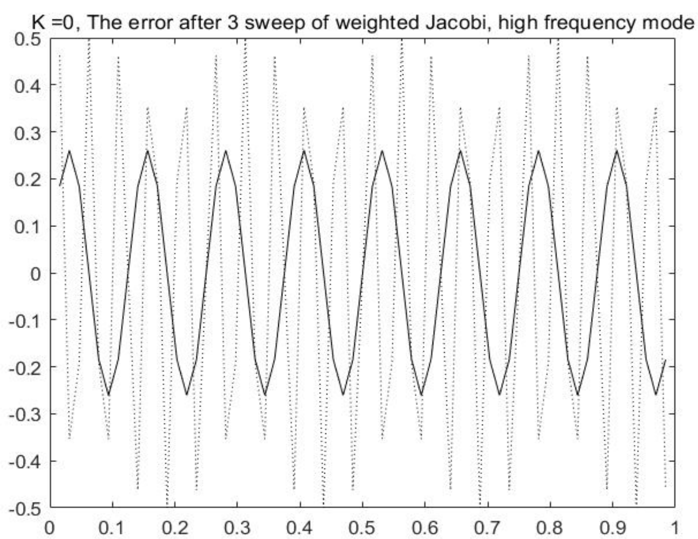

Figure 9: Visualization of the error after three sweeps of weighted Jacobi and also the oscillatory mode $k = 40$ in the initial guess in the same figure.

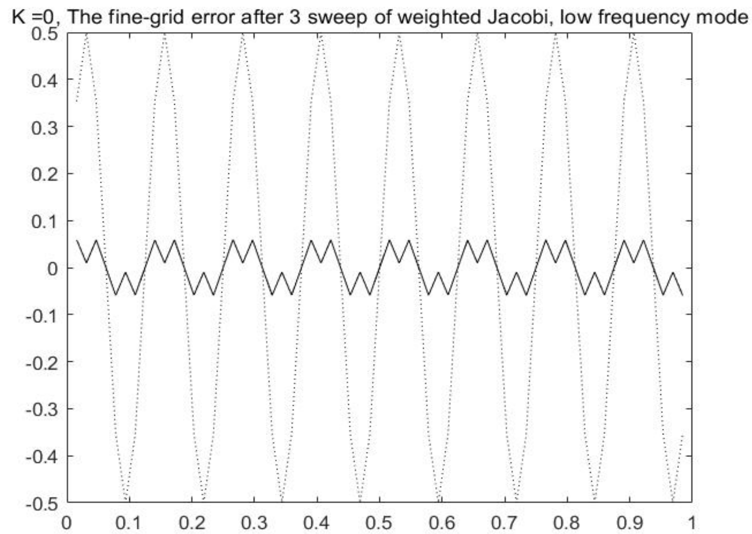

Figure 10: Visualization of the find-grid error after three sweep of weighted Jacobi on the coarse-grid problem and also the smooth mode $k = 16$ in the initial guess in the same figure.

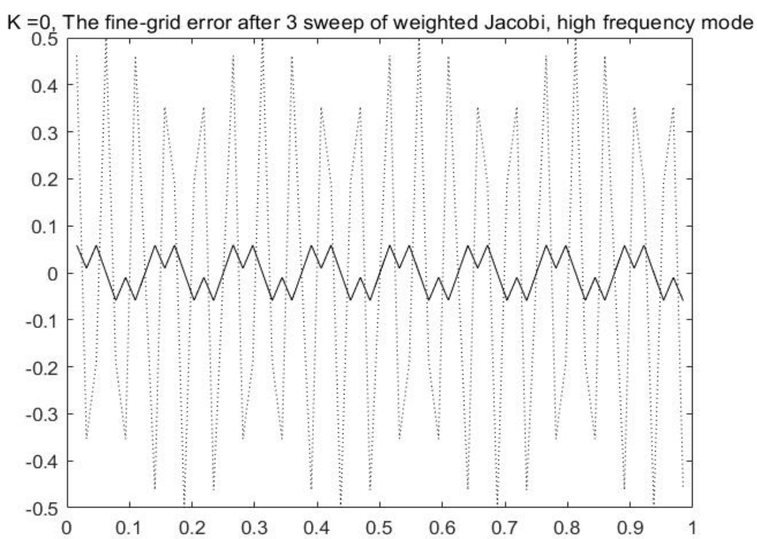

Figure 11: Visualization of the find-grid error after three sweep of weighted Jacobi on the coarse-grid problem and the oscillatory mode $k = 40$ in the initial guess in the same figure.

3. Set $K = 2$ and reproduce Figure 3.5 for this case

Now we set $K = 2$ and reproduce Figure 3.5 in the Book. Figure 12, 13, 14, 15, 16 and 17 are the reproduction of Fig. 3.5 in the Book when $K = 2$. It is hard to tell the difference only based on the observation of these six figures since they are very similar to those when $K = 0$. Therefore let's look at the difference between the norm of the residual vector and error vector at similar steps in the V-cycle when $K = 0$ and $K = 2$ as shown in the Table 2 and 3. Here **Top Left**, **Top Right**, **Middle Left**, **Middle Right**, **Bottom Left** and **Bottom Right** represent the step in the V-cycle for each figure of

Fig. 3.5 in the Book. Since in the Book it use 2-norm for error vector for the numerical example, therefore I also use 2-norm for the error vector and I use infinite norm for the residual vector as the last computing assignment. We can see that when $K = 2$ the 2-norm of the error vector is less than the 2-norm of the error vector when $K = 0$ at the same steps in the V-cycle as shown in Table 3. It implies that when $K = 2$ the convergence is faster than when $K = 0$. It makes sense since we know from the Professor Bayliss's textbook that we should expect more rapid convergence the more diagonally dominant the matrix A is, where matrix A represent coefficient matrix of the discretized system. And when $K = 2$, the matrix A should be more diagonally dominant since based on the discussion of Section 1.2 we know that when the Eq. (1) is discretized, each diagonal entry of the matrix A when $K = 2$ is $2h^2\pi^2 + 2$ while the diagonal entry of matrix A is only 2 when $K = 0$.

Table 2: Infinite norm of the residual vector for each step in the Fig. 3.5 in the Book

	Top Left	**Top Right**	**Middle Left**	**Middle Right**	**Bottom Left**	**Bottom Right**
$K = 0$	6.0807e+03	1.2786e+03	627.1355	444.6002	444.6002	59.4907
$K = 2$	6.0968e+03	1.2882e+03	629.6687	444.7178	443.9599	59.8136

Table 3: 2-norm of the error vector for each step in the Fig. 3.5 in the Book

	Top Left	**Top Right**	**Middle Left**	**Middle Right**	**Bottom Left**	**Bottom Right**
$K = 0$	4.0000	2.2869	1.4740	0.6514	0.3348	0.1368
$K = 2$	4.0000	2.2838	1.4678	0.6497	0.3345	0.1365

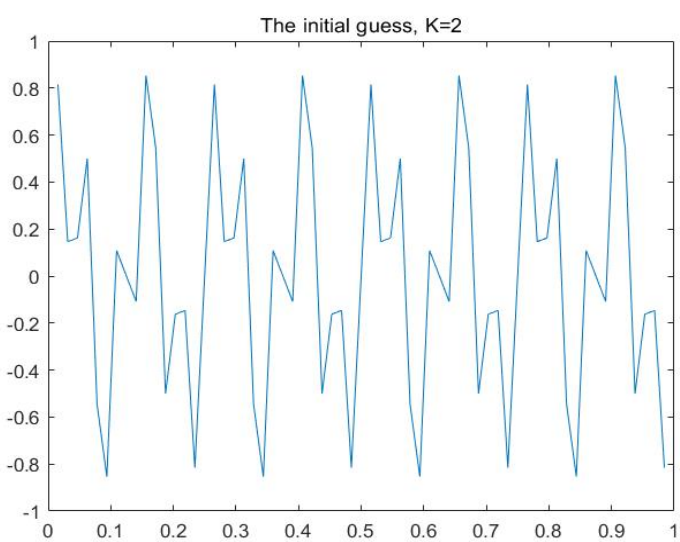

Figure 12: Visualization of the Initial Guess when $K = 2$

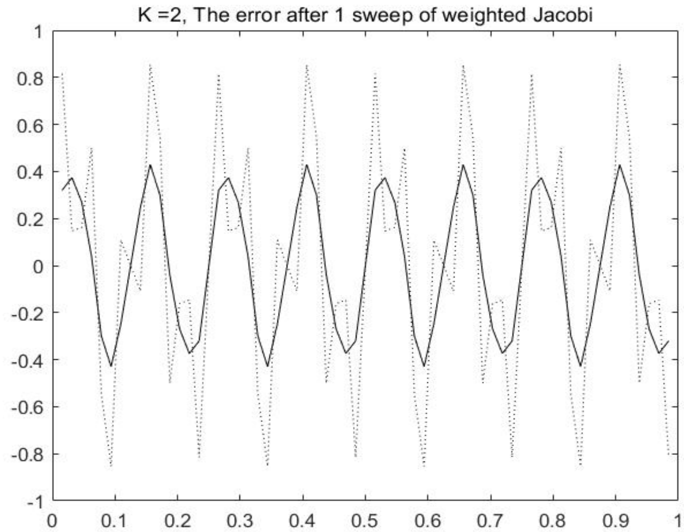

Figure 13: The error after one sweep of weighted Jacobi when $K = 2$

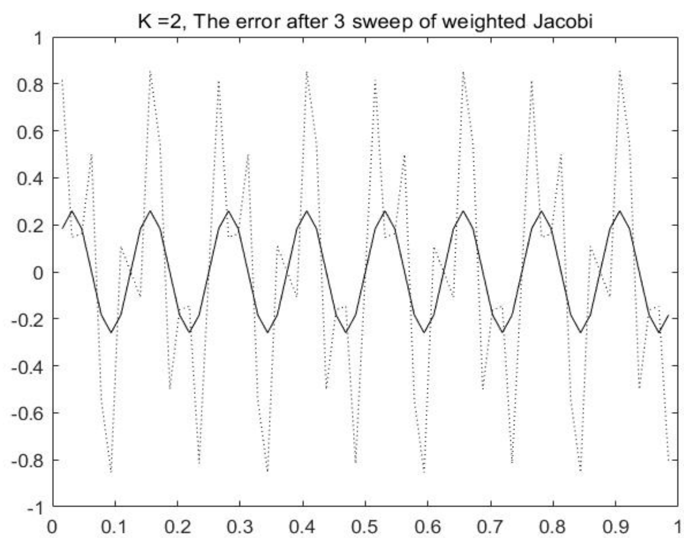

Figure 14: The error after three sweep of weighted Jacobi when $K = 2$

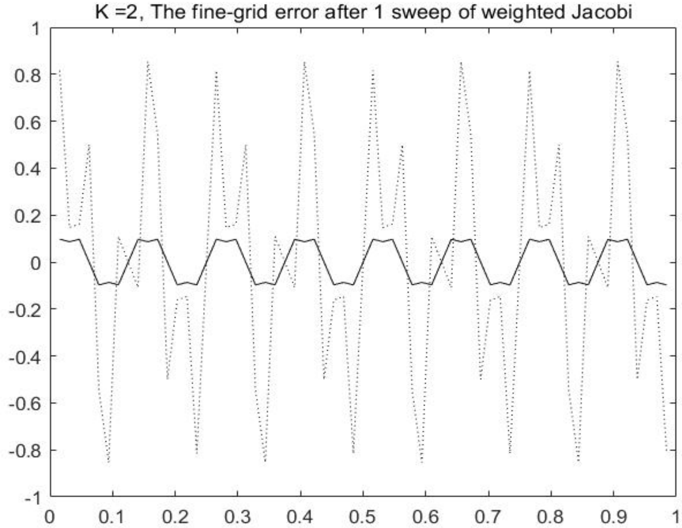

Figure 15: The fine-grid error after one sweep of weighted Jacobi on the coarse-grid problem when $K = 2$.

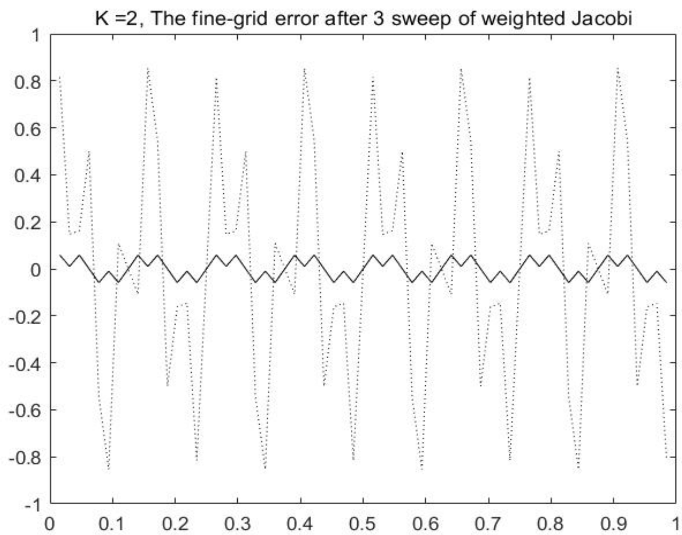

Figure 16: The fine-grid error after three sweep of weighted Jacobi on the coarse-grid problem when $K = 2$.

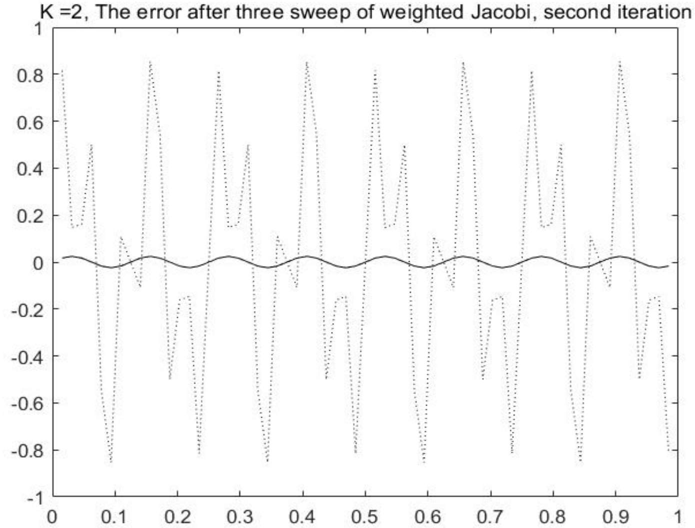

Figure 17: The fine-grid error after the coarse-grid correction is followed by three weighted Jacobi sweeps on the fine grid when $K = 2$.

4. Consider the Neumann problem for our ODE and redo parts (a) and (b) above for the Neumann problem

In this section and next we consider the Eq. 2 with Neumann conditions on all the boundaries and use multigrid method to reproduce the Fig. 3.5 in Briggs Book.

$$-u_{xx} + K\pi^2 u = 0, \frac{du}{dx}(0) = \frac{du}{dx}(1) = 0. \tag{2}$$

4.1 Determine all nontrivial solutions (eigenfunctions) when $K \neq 0$ is an eigenvalue for Eq. (2)

When $K \neq 0$, Eq. (2) is a constant coefficient equations and let $u(x) = e^{tx}$ where t is arbitrary constant, then we plug it into Eq. (2) and we get the solution which are

$$u(x) = \begin{cases} c_1 e^{\pi x\sqrt{K}} + c_2 e^{-\pi x\sqrt{K}}, K > 0 \\ c_3 \cos\left(\pi x\sqrt{-K}\right) + c_4 \sin\left(\pi x\sqrt{-K}\right), K < 0 \end{cases}$$

Now based on the boundary conditions, we can deduce that when $K > 0$ there will only have trivial solutions. When $K < 0$, we get the nontrivial solutions (eigenfunctions) of (1) as $u(x) = \cos(n\pi x), n = 0, 1, 2...$ (constant are omitted) and the corresponding eigenvalue $K = -n^2, n = 0, 1, 2, ...$

4.2 Identify the eigenvalues and eigenvectors of the discrete system for Eq. (2)

Consider the discretized problem of Eq. (2)

$$-\frac{u_{j+1} + u_{j-1} - 2u_j}{h^2} + K\pi^2 u_j = \lambda u_j,\ j = 1, 2, ..., N,\ h = \frac{1}{N+1}$$

where it is understood that $\frac{du_0}{dx} = \frac{du_{N+1}}{dx} = 0$. This system leads to an $N \times N$ matrix so there can be only N eigenvalues and eigenvectors. To find the eigenvalues and eigenvectors try to micmic the continuous problem by looking at vectors of the form as

$$u_j = \cos\left(m\pi x_j\right), \; x_j = jh, \; j = 1, 2, ..., N, \; m = 1, 2, 3, ...$$

where we should expect the upper limit on m to be $m = N$ but this has to be shown. First it is obvious that this guessing eigenvector form can satisfy the Neumann conditions in Eq. 2.

Now plug the guessing solution into the discretized system and we have

$$-\frac{\cos\left(m\pi h\left(j+1\right)\right) + \cos\left(m\pi h\left(j-1\right)\right) - 2\cos\left(m\pi hj\right)}{h^2} + K\pi^2\cos\left(m\pi hj\right) = \lambda\cos\left(m\pi hj\right).$$

Using the basic trig identities that

$$\cos\left(m\pi h\left(j \pm 1\right)\right) = \cos\left(m\pi hj\right)\cos\left(m\pi h\right) \mp \sin\left(m\pi hj\right)\sin\left(m\pi h\right),$$

we have

$$\left[\frac{2}{h^2}\left[1 - \cos\left(m\pi h\right)\right] + K\pi^2\right]\cos\left(m\pi hj\right) = \lambda\cos\left(m\pi hj\right).$$

Therefore we show that $u_j = \cos\left(m\pi x_j\right), \; x_j = jh, \; j = 1, 2, ..., N, \; m = 1, 2, 3, ...$ does in fact define an eigenvector for the discretized problem with eigenvalue

$$\lambda_m = \left[\frac{2}{h^2}\left[1 - \cos\left(m\pi h\right)\right] + K\pi^2\right] = \frac{4}{h^2}\sin^2\left(\frac{m\pi h}{2}\right) + K\pi^2$$

using the trig formula that

$$1 - \cos\left(x\right) = 2\sin^2\left(\frac{x}{2}\right)$$

4.3 Are the solutions unique? How to make the solution unique?

Given the Neumann boundary condition for all the boundaries in Eq. (2), there will be infinite number of solutions when $K < 0$, which means the problem is ill-posed and the solutions are not unique. To make the solutions unique, one of the boundaries must be Dirichlet. For example we can set $u(0) = 0$ to make the solutions unique.

5. Parallel the implementation of the multigrid V-cycle in Briggs for the Neumann Problem when $K = 0$.

As discuss in Section 4.3, when $K = 0$ we also need to ensure the solution is unique since arbitrary constant can be the solution of Eq. (2). Therefore in numerical approach, first I introduce two fictitious points u_{-1} and u_{n+1} as mentioned in page 12 in the Professor Bayliss's textbook since the solution at $x = 0$ and $x = 1$ are not specified. When $K = 0$, the equation becomes $-u_{xx} = 0$ and based on the boundary condition that $\frac{du}{dx}\left(0\right) = 0$, we have two equations involving u_0 and u_{-1} as

$$-\left(\frac{u_1+u_{-1}-2u_0}{h^2}\right)=0,\ \frac{u_1-u_{-1}}{2h}=0. \tag{3}$$

The first equation in Eq. (3) is just the $-u_{xx}=0$ discretized using the second order central difference approximation to the second derivative and the fictitious point u_{-1}. The second equation in Eq. (3) is the second order central difference approximation to the Neumann condition at $x=0$. We can deduce from Eq. (3) that $u_0=u_1$ and similarly we know that $u_{n-1}=u_n$. When $K=0$, arbitrary constant can satisfy the Eq. (2) and since we discuss in Section 4.3 that we at least need one of the boundaries to be Dirichlet such that our solution is unique, therefore I set $u_0=0$ in the initial guess and therefore $u_1=u_0=0$. Note, that I also keep setting $u_0=u_1=0$ at each sweep of relaxations in fine-grid and coarse-grid when I implement the Multigrid method, which is the new implementation in the Multigrid method for this case. The Figure 18, 19, 20, 21, 22 and 23 are the reproduction of Figure 3.5 in the book with new implementation of the multigrid method for this case when $K=0$. The initial guess we use here is the same as

$$v_j^h=\frac{1}{2}\left[\sin\left(\frac{16j\pi}{n}\right)+\sin\left(\frac{40j\pi}{n}\right)\right],$$

consisting of the $k=16$ and $k=40$ modes.

We know that since we do not specify the Dirichlet condition at $x=1$, we should expect that the approximation will be diverge when x is close to 1. The Figure 18-23 show that although the error are reduced effectively as we transform between fine-grid and coarse-grid using the multigrid method, when x is close to 1, we can always see the approximations are impacted by the initial guess we used: The approximation are going down when x is close to 1, which is similar to the trend of the initial guess. The results show that the approximations will mimic the initial guess and it implies that a better initial guess should be considered since using the same initial guess in the Book in this case is not good. We don't want to see any divergence happen near the boundary at $x=0$ and $x=1$. To illustrate my opinion, I give another initial guess as

$$v_j^h=\frac{1}{2}\left[\cos\left(\frac{20j\pi}{n}\right)+\sin\left(\frac{40j\pi}{n}\right)\right], \tag{4}$$

consisting of the $k=20$ and $k=40$ modes. The final result with initial guess (4) is shown in Figure 24 at the same relaxation step as Figure 23 and we can see that when the initial guess is better, the approximation is also better since the approximations will mimic the initial guess's shape.

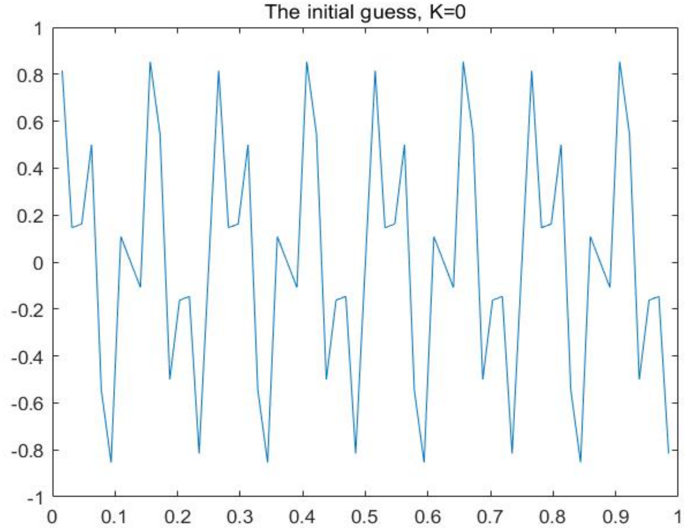

Figure 18: Visualization of the Initial Guess when $K = 0$ with the same initial guess in Briggs Book.

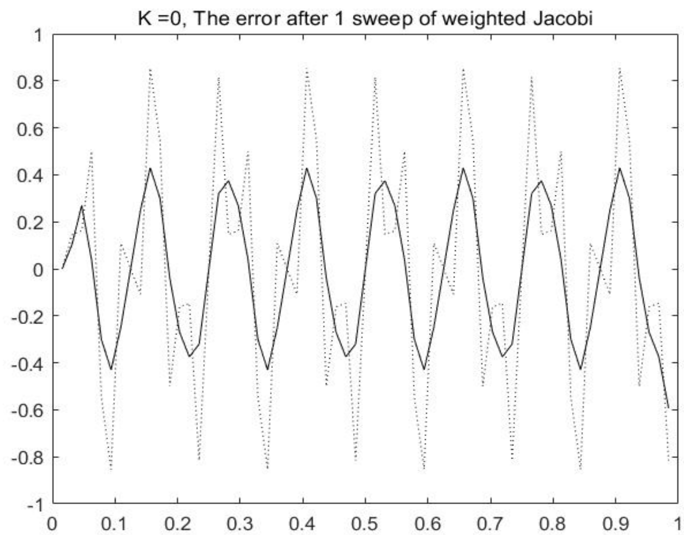

Figure 19: The error after one sweep of weighted Jacobi when $K = 0$ for Eq. (2) with the same initial guess in Briggs Book.

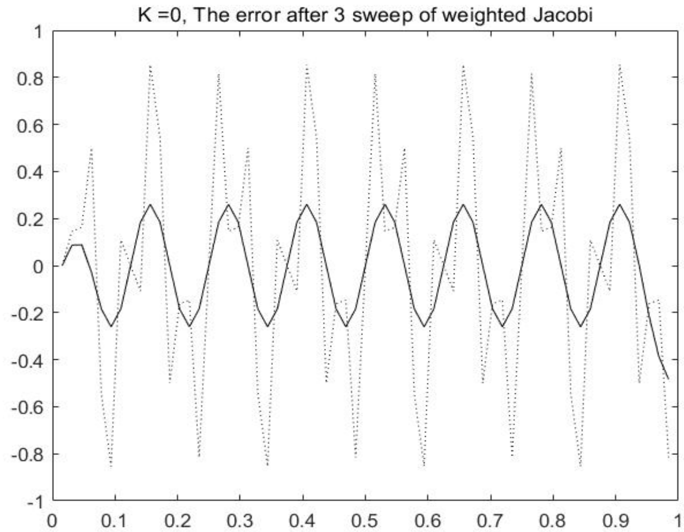

Figure 20: The error after three sweep of weighted Jacobi when $K = 0$ for Eq. (2) with the same initial guess in Briggs Book.

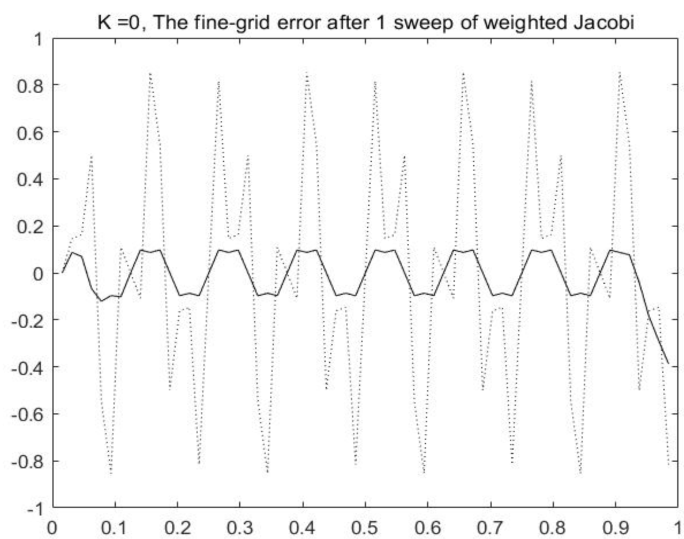

Figure 21: The fine-grid error after one sweep of weighted Jacobi on the coarse-grid problem when $K = 0$ for Eq. (2) with the same initial guess in Briggs Book.

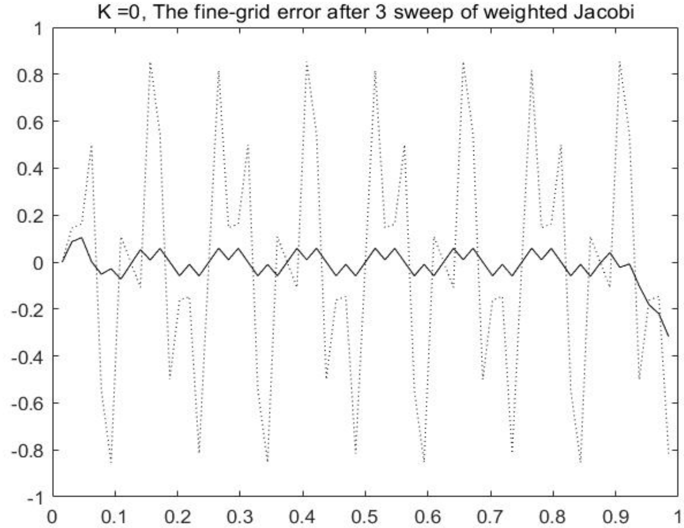

Figure 22: The fine-grid error after three sweep of weighted Jacobi on the coarse-grid problem when $K = 0$ for Eq. (2) with the same initial guess in Briggs Book.

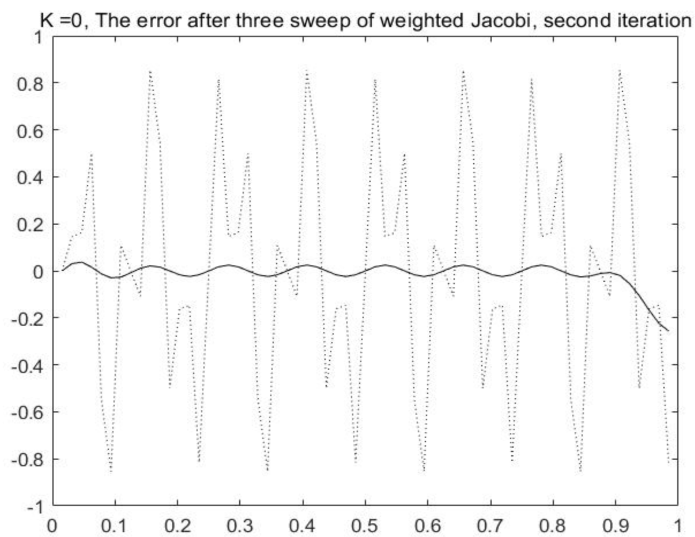

Figure 23: The fine-grid error after the coarse-grid correction is followed by three weighted Jacobi sweeps on the fine grid when $K = 0$ for Eq. (2) with the same initial guess in Briggs Book.

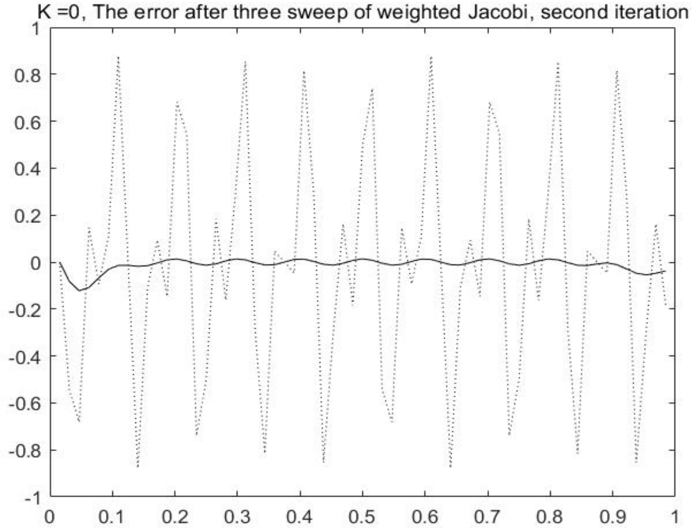

Figure 24: The fine-grid error after the coarse-grid correction is followed by three weighted Jacobi sweeps on the fine grid when $K = 0$ for Eq. (2) with different initial guess as Eq. (4).

6. MATLAB code

6.1 Mutligrid Method

```
function [error_norm_vector, residual_norm_vector]= ESAM445_MultiGrid_MingfuLiang(K,N,Omega,SmoothMode, OscillatoryMode, BoundaryCondition)
% Author: Mingfu Liang
% Date: 2019/05/28
%
% Implementation of Multigrid method using weight Jacobi method for relaxation
% to solving boundary value equation with different boundary conditions.
%
%                         - u_{xx} + K * pi * u =0
%
% Input:
%          K:
%                Input  0 or 2 to choose different equation.
%
%                In this homewore we only have two choice of K as  0, 2.
%
%          N:
%                 Input the size of the grid.
%
```

```
%                   the size of the grid we are going to create. In this
%                   homework we use N=64 to reproduce the numerical example in
%                   Briggs' Book.
%
%            omega:
%                   Input value between 0 to 1 for weight
%
%                  The weight parameter for weight Jacobi method.
%
%            SmoothMode:
%                   The smooth mode in the initial guess.
%
%                  In this homework the smooth mode in the initial guess is 16
%
%            OscillatoryMode:
%                   The oscillatory mode in the initial guess.
%
%                  In this homework the oscillatory mode in the initial guess
%                  is 40
%
%            BoundaryCondition:
%                   Input the choice of boundary condition.
%
%                  In this homework we have two the boundary condition choice:
%
%                  input 1 to choose Dirichlet conditions
%                  input 2 to choose Neumann conditons
%
%   Output:
%             error_norm_vector:
%                   The norm of the error vector in each step
%
%             residual_norm_vector:
%                   The norm of the residual vector in each step
%
% Example:
%
%             ESAM445_MultiGrid_MingfuLiang(0,64,2/3,16,40,1)
%
%             means that you are going to set the grid size N=64 and omega
```

```
%              =2/3 to do the Weight Jacobi method in each relaxation step
%              in the V-cycle in multigrid method  when K=0 and using the
%              Dirichlet conditions, which represent Eq (1) in the assignment.
%

boundary_condition = BoundaryCondition; % if boundary condition is equal to 1 means we use the Dirichlet Value
                            % if boundary condition is equal to 2 means we use the Neumann condition

w = Omega; % the parameter omega in weight Jacobi method
n = N; % the number of grid point
k1=SmoothMode; % low frequency mode in initial guess, in Book page 37
k2=OscillatoryMode; % high frequency mode in initial guess, in Book page 37
%K=2; % the parameter for the boundary value problem, it can take 0 or 2 in this assignment
res_vector_ind=1; % use to indicate the row index of the vector
err_norm_count = 2; % use to count for storaging the error vector norm in the err_vector
vector_norm_count=2; % use to count for storaging the residual vector norm in the res_norm_vector
h  = 1/n;          % grid spacing.
u  = zeros(1,1+n);     % storage for solution, here I use n+1 since I want to be consistent with the notebook
                                              % we denote that 0,1,2,...,n, which means
                                              % that if we have n=16, we actually have n+1
                                              % points although the at 0 and n+1 it should
                                              % be zero as the boundary condition in this
                                              % Homework since we are setting at all the
                                              % boundary u=0.
f=u; % initialize the f(x), which should be zero in this assignment
res= u;% storage for residual, here residual is a matrix
u_update  = u;          % intialize the solution update in each iteration
i=1:n+1 ; % use for parallelly generate the grid point vector
grid_point = (i-1).*h; % generate the grid point vector
```

```
j =2:n; % the index for the parallel computation in weighted Jacobi

%%%%%%%%%%%%%%%%%%%%
u_low = u; % initialize the low frequency mode vector
u_high =u; % initialize the high frequency mode vector
u_low(1,j)=sin(k1.*(j-1)*pi/n)/2; % generate the low frequency mode vector in a parallelism manner
u_high(1,j)=sin(k2.*(j-1)*pi/n)/2; % generate the high frequency mode vector in a parallelism manner

%%%%%%%%%% visualize the low frequency mode and high frequency mode %%%%
figure;
plot(grid_point(2:1:end-1),u_low(2:1:end-1));
hold on;
plot(grid_point(2:1:end-1),u_high(2:1:end-1));
hold off;
legend1=['k=',num2str(k1)];
legend2=['k=',num2str(k2)];
legend(legend1,legend2);
title('visualize low frequency mode and high frequency mode');

%%%%%%%%%% initialize the initial guess, error vector and residual vector

u(1,j)=(sin(k1.*(j-1)*pi/n)+sin(k2.*(j-1)*pi/n))/2; % initialize the initial guess

if boundary_condition ==2
    u(1,1)=0;
    u(1,2)=u(1,1);
    u(1,n+1)=u(1,n);
end

u_initial = u; % copy the initial guess for later reuse
err_inital(1,j) = 0-u_initial(1,j); % get the corresponding error vector for each individual solution
err_vector(1,1)=norm(err_inital,2); % get the norm of the initial error vector and storage it in the err_vector
res_norm = f(1,j) - (1/h^2)*( -u(1,j-1) - u(1,j+1) + (2+(h^2)*pi*pi*K)*u(1,j) ); % get the corresponding residual vector for each individual solution
res_norm_vector(1,1)= norm(res_norm,2); % get the norm of the initial
```

```
    residual vector and storage it in the err_vector

%%%%%%% plot initial guess %%%%%%%%%%%%%%%%%%%
figure;
plot(grid_point(2:1:end-1),u_initial(2:1:end-1));
title(['The initial guess, K=',num2str(K)]);

%%%%%%% use weight jacobi relax three time %%%%%%%%%%%

for relax_time =1:3
    % use weighted Jacobi to do relaxation in fine-grid in a parallelism
    % manner
    u_update(1,j) = (1-w)*(u(1,j))+ ...
                    w*(1/(2 + (h^2)*pi*pi*K))*( u(1,j-1) + u(1,j+1) + h^2 * f(1,j));
    u = u_update; % update the approximation

    if boundary_condition ==2
        u(1,1)=0;
        u(1,2)=u(1,1);
        u(1,n+1)=u(1,n);
    end

    %%%%%%%%%%%%% evaluate the norm of error vector %%%%%%%%%%%%%%%
    if relax_time ==1 || relax_time ==3
        err(1,j)=0-u(1,j); % get the error vector
        err_vector(res_vector_ind,err_norm_count )=norm(err,2); % storage the norm of the error vector
        err_norm_count = err_norm_count +1; %
        res_norm = f(1,j) - (1/h^2)*( -u(1,j-1) - u(1,j+1) + (2+(h^2)*pi*pi*K)*u(1,j) );
        res_norm_vector(1,vector_norm_count)= norm(res_norm,2);
        vector_norm_count = vector_norm_count + 1;
    end

    %%%%%%%%%%%% plot the top right figure in Fig. 3.5 in the book %%%%%
    if relax_time ==1
        figure;
        plot(grid_point(2:1:end-1),u_initial(2:1:end-1),"k:");
        hold on
```

```
        plot(grid_point(2:1:end-1),u(2:1:end-1),"k");
        hold off
        title(['K =',num2str(K),', The error after ', num2str(relax_time) ,' sweep of weighted Jacobi'])
    end

end

%%%%%%%%%%%% plot the middle left figure in Fig. 3.5 in the book %%%%%
 figure;
 plot(grid_point(2:1:end-1),u_initial(2:1:end-1),"k:");
 hold on
 plot(grid_point(2:1:end-1),u(2:1:end-1),"k");
 hold off
 title(['K =',num2str(K),', The error after ', num2str(relax_time) ,' sweep of weighted Jacobi'])

 %%%%% visualize the error with different mode in the same plot%%%%%%%%
 figure;
 plot(grid_point(2:1:end-1),u_low(2:1:end-1),"k:");
 hold on
 plot(grid_point(2:1:end-1),u(2:1:end-1),"k");
 hold off
 title(['K =',num2str(K),', The error after ', num2str(relax_time) ,' sweep of weighted Jacobi, low frequency mode'])

 figure;
 plot(grid_point(2:1:end-1),u_high(2:1:end-1),"k:");
 hold on
 plot(grid_point(2:1:end-1),u(2:1:end-1),"k");
 hold off
 title(['K =',num2str(K),', The error after ', num2str(relax_time) ,' sweep of weighted Jacobi, high frequency mode'])

%%%%%%% Compute r^{2h} =I^{2h}_{h} * r^{h} %%%%%%%%%%%
res(1,j) = f(1,j) - (1/h^2)*( -u(1,j-1) - u(1,j+1) + (2+(h^2)*pi*pi*K)*u(1,j) ); % get the corresponding residual vector for each individual solution in fine-grid
res_coarse = zeros(1,n/2+1); % initialize the coarse grid residual vector
res_coarse(1,1)=res(1,1); % for the j=0, just copy the residual from fine-
```

```
grid residual vector
res_coarse(1,end)=res(1,end); % for the j=n+1, just copy the residual from fine-grid residual vector
res_coarse(1,2:32)=(1/4)*(res(1,2:2:end-2)+2*res(1,3:2:end-1)+res(1,4:2:end)); % interpolation fine to coarse grid, here use full weighting as a restriction operator.
e_2h = res_coarse * 0; % initialize the error vector in coarse grid
e_2h_update = e_2h; % initialize the error vector in coarse grid

%%%%%%% Relax three times on A^{2h}*e^{2h}=r^{2h} %%%%%%

j_2h = 2:n/2; % coarse grid index
h_c = 2*h; % coarse grid interval

for relax_time =1:3
    % use the weighted Jacobi to relax the residual equation in a
    % parallelism manner
    e_2h_update(1,j_2h) = (1-w)*(e_2h(1,j_2h))+ ...
                        w*(1/(2 + (h_c^2)*pi*pi*K))*( e_2h(1,j_2h-1) + e_2h(1,j_2h+1) + h_c^2 * res_coarse(1,j_2h));
    e_2h = e_2h_update; % update the error vector

    if boundary_condition ==2
        e_2h(1,1)=0;
        e_2h(1,2)=e_2h(1,1);
        e_2h(1,n/2+1)=e_2h(1,n/2);
    end
    %%%%%%%%%%% check fine-grid error %%%%%%%%%%%%%%%%%%%%%%
    e_h = res*0;
    e_h(1,1:2:end) = e_2h; % transfer back to the fine-grid
    e_h(1,2:2:end) = (e_h(1,1:2:end-2) +e_h(1,3:2:end) )/2; % transfer back to the fine-grid using interpolation
    u_new = u + e_h;
    if relax_time ==1 || relax_time ==3
        err(1,j)=0-u_new(1,j);
        err_vector(res_vector_ind,err_norm_count )=norm(err,2);
        err_norm_count = err_norm_count +1;
        res_norm = f(1,j) - (1/h^2)*( -u_new(1,j-1) - u_new(1,j+1) + (2+(h^2)*pi*pi*K)*u_new(1,j) );
        res_norm_vector(1,vector_norm_count)= norm(res_norm,2);
```

```
            vector_norm_count = vector_norm_count + 1;
        end

        %%%%%%%%%%%%% plot the middle right figure in Fig. 3.5 in the book %%%%%
        if relax_time ==1
            figure;
            plot(grid_point(2:1:end-1),u_initial(2:1:end-1),"k:");
            hold on
            plot(grid_point(2:1:end-1),u_new(2:1:end-1),"k");
            hold off
            title(['K =',num2str(K),', The fine-grid error after ', num2str(relax_time) ,' sweep of weighted Jacobi'])
        end

    end

%%%%%%%%%%%%% plot the bottom left figure in Fig. 3.5 in the book %%%%%
u = u_new;
figure;
plot(grid_point(2:1:end-1),u_initial(2:1:end-1),"k:");
hold on
plot(grid_point(2:1:end-1),u_new(2:1:end-1),"k");
hold off
title(['K =',num2str(K),', The fine-grid error after ', num2str(relax_time),' sweep of weighted Jacobi'])

%%%%% visualize the fine-grid error with different mode in the same plot %
 figure;
 plot(grid_point(2:1:end-1),u_low(2:1:end-1),"k:");
 hold on
 plot(grid_point(2:1:end-1),u(2:1:end-1),"k");
 hold off
 title(['K =',num2str(K),', The fine-grid error after ', num2str(relax_time),' sweep of weighted Jacobi, low frequency mode'])

 figure;
 plot(grid_point(2:1:end-1),u_high(2:1:end-1),"k:");
 hold on
 plot(grid_point(2:1:end-1),u(2:1:end-1),"k");
```

```
 hold off
 title(['K =',num2str(K),', The fine-grid error after ', num2str(relax_time),' sweep of weighted Jacobi, high frequency mode'])

%%%%%%%%%%%%%%%%%%%%%% second iteration %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

j =2:n;
u_update = zeros(1,n+1);

for relax_time =1:3

    u_update(1,j) = (1-w)*(u(1,j))+ ...
                    w*(1/(2 + (h^2)*pi*pi*K))*( u(1,j-1) + u(1,j+1) + h^2 * f(1,j));
    u = u_update;

   if boundary_condition ==2
        u(1,1)=0;
        u(1,2)=u(1,1);
        u(1,n+1)=u(1,n);
    end

    %%%%%%%%%%%%%% evaluate the norm of error vector %%%%%%%%%%%%%%%%%%
    if relax_time ==3
        err(1,j)=0-u(1,j);
        err_vector(res_vector_ind,err_norm_count )=norm(err,2);
        err_norm_count = err_norm_count +1;
        res_norm = f(1,j) - (1/h^2)*( -u(1,j-1) - u(1,j+1) + (2+(h^2)*pi*pi*K)*u(1,j) );
        res_norm_vector(1,vector_norm_count)= norm(res_norm,2);
        vector_norm_count = vector_norm_count + 1;
    end
end

%%%%%%%%%%%%% plot the bottom right figure in Fig. 3.5 in the book %%%%%%
 figure;
 plot(grid_point(2:1:end-1),u_initial(2:1:end-1),"k:");
 hold on
 plot(grid_point(2:1:end-1),u(2:1:end-1),"k");
 title(['K =',num2str(K),', The error after three sweep of weighted Jacobi,
```

```
second iteration'])
hold off

%%% visualize the fine-grid error with different mode in the same plot %%
figure;
plot(grid_point(2:1:end-1),u_low(2:1:end-1),"k:");
hold on
plot(grid_point(2:1:end-1),u(2:1:end-1),"k");
title(['K =',num2str(K),', The error after three sweep of weighted Jacobi, second iteration, low mode'])
hold off

figure;
plot(grid_point(2:1:end-1),u_high(2:1:end-1),"k:");
hold on
plot(grid_point(2:1:end-1),u(2:1:end-1),"k");
title(['K =',num2str(K),', The error after three sweep of weighted Jacobi, second iteration, high mode'])
hold off

%%%%%%% Compute r^{2h} =I^{2h}_{h} * r^{h} %%%%%%%%%%%
res(1,j) = f(1,j) - (1/h^2)*( -u(1,j-1) - u(1,j+1) + (2+(h^2)*pi*pi*K)*u(1,j) ); % get the corresponding residual vector for each individual solution in fine-grid
res_coarse = zeros(1,n/2+1); % initialize the coarse grid residual vector
res_coarse(1,1)=res(1,1); % for the j=0, just copy the residual from fine-grid residual vector
res_coarse(1,end)=res(1,end); % for the j=n+1, just copy the residual from fine-grid residual vector
res_coarse(1,2:32)=(1/4)*(res(1,2:2:end-2)+2*res(1,3:2:end-1)+res(1,4:2:end)); % interpolation fine to coarse grid, here use full weighting as a restriction operator.
e_2h = res_coarse * 0; % initialize the error vector in coarse grid
e_2h_update = e_2h; % initialize the error vector in coarse grid

%%%%%%% Relax three times on A^{2h}*e^{2h}=r^{2h} %%%%%%

j_2h = 2:n/2; % coarse grid index
h_c = 2*h; % coarse grid interval

```

```matlab
for relax_time =1:3
    % use the weighted Jacobi to relax the residual equation in a
    % parallelism manner
    e_2h_update(1,j_2h) = (1-w)*(e_2h(1,j_2h))+ ...
                       w*(1/(2 + (h_c^2)*pi*pi*K))*( e_2h(1,j_2h-1) + e_2h(1,j_2h+1) + h_c^2 * res_coarse(1,j_2h));
    e_2h = e_2h_update; % update the error vector

     if boundary_condition ==2
         e_2h(1,1)=0;
         e_2h(1,2)=e_2h(1,1);
         e_2h(1,n/2+1)=e_2h(1,n/2);
    end
    %%%%%%%%%% check fine-grid error %%%%%%%%%%%%%%%%%%%%
    e_h = res*0;
    e_h(1,1:2:end) = e_2h; % transfer back to the fine-grid
    e_h(1,2:2:end) = (e_h(1,1:2:end-2) +e_h(1,3:2:end) )/2; % transfer back to the fine-grid using interpolation
    u_new = u + e_h;
     if relax_time ==3
         err(1,j)=0-u_new(1,j);
         err_vector(res_vector_ind,err_norm_count )=norm(err,2);
         err_norm_count = err_norm_count +1;
         res_norm = f(1,j) - (1/h^2)*( -u_new(1,j-1) - u_new(1,j+1) + (2+(h^2)*pi*pi*K)*u_new(1,j) );
         res_norm_vector(1,vector_norm_count)= norm(res_norm,2);
         vector_norm_count = vector_norm_count + 1;
    end

end

 %%%% output the error reduction with respect to the initial error %%%%
 error_reduction_to_initial_error = (err_vector(1,2:end))./(err_vector(1,1))
 error_norm_vector=err_vector;
 residual_norm_vector =res_norm_vector;
end
```