An Introduction to Partial Differential Equations in the Undergraduate Curriculum

David Arnold

LECTURE 13
Sturm-Liouville — Numerics

13.1. Outline of Lecture

- Revisiting an old problem
- Making connections
- Second derivative approximation
- Transforming to a matrix problem
- Automating the Process

13.2. Revisiting an Old Problem

In our stay at Park City, we've encountered numerous problems similar to the following.

$$u_t = ku_{xx}, \quad 0 < x < \pi, \quad t > 0, \tag{13.1}$$

$$u(0,t) = u(\pi,t) = 0, \quad t > 0, \tag{13.2}$$

$$u(x,0) = x^3(\pi - x), \quad 0 < x < \pi. \tag{13.3}$$

We've assumed that we can find a solution of the form $u(x,t) = y(x)g(t)$, where the variables "separate," [1] then substituted it into the

[1]Unfortunately, I've yet to hear a satisfactory explanation as to why one would want to separate the variables as in $u(x,t) = y(x)g(t)$. The only explanation I've heard is "Look, it works!" However, I am left yearning for more. In my discussions with Dr. Polking, he has tried to explain that it has to do with the inherent symmetries of the problem, but I'll need to pursue this thought further when I get home.

partial differential equation (13.1)

$$\big[y(x)g(t)\big]_t = k\big[y(x)g(t)\big]_{xx}$$

to find that

$$y(x)g'(t) = ky''(x)g(t).$$

A little more algebra provides

$$\frac{g'(t)}{kg(t)} = \frac{y''(x)}{y(x)} = -\lambda, \tag{13.4}$$

where λ is a constant.[2]

The first ratio in (13.4) leads to the differential equation

$$g'(t) + \lambda k g(t) = 0 \qquad \text{with solution} \qquad g(t) = e^{-\lambda k t}. \tag{13.5}$$

The second ratio in (13.4), together with the Dirichlet boundary conditions (13.2), leads to the Sturm-Liouville problem

$$-y''(x) = \lambda y(x), \quad y(0) = y(\pi) = 0. \tag{13.6}$$

The values of λ for which this equation has nontrivial (nonzero) solutions $y(x)$ are called *eigenvalues* and the associated solutions $y(x)$ are called *eigenfunctions*. Several times during our PDE lectures we saw that the eigenvalue-eigenfunction pairs for the Sturm-Liouville problem (13.6) are[3]

$$\lambda_n = n^2 \qquad \text{and} \qquad y_n(x) = \sin nx. \tag{13.7}$$

[2]I found I didn't understand most of the arguments given that these rates had to equal a constant (one exception was a neat little argument Katherine Socha shared with me), so I tried to devise my own. Suppose that that $r(t) = s(x)$ for all t and x in the domain of each respective function. For purposes of contradiction, suppose that r is not constant. Then there exists $t_1 \neq t_2$ in the domain of r such that $r(t_1) \neq r(t_2)$. Pick some x_0 in the domain of s. Because $s(x) = r(t)$ for all x and t, I know that $s(x_0) = r(t_1)$ and $s(x_0) = r(t_2)$. Because $r(t_1) \neq r(t_2)$, this contradicts the fact that s is a function. Therfore, r must be constant. A similar argument shows that s is constant. Because $r(t) = s(x)$ for all t and x in the respective domains, the function s must equal the same constant.

[3]It is important to note that if $y(x)$ is an eigenfunction of (13.6), then so to is $cy(x)$, where c is any constant. Check this! Therefore, we could just as easily use $y_n(x) = (3/4)\sin nx$ as our eigenfunctions. This fact will be important later in the narrative.

Thus, the first few eigenvalue-eigenfunction pairs are:

$$\begin{aligned} \lambda_1 = 1 \quad &\leftrightarrow \quad y_1(x) = \sin x \\ \lambda_2 = 4 \quad &\leftrightarrow \quad y_2(x) = \sin 2x \\ \lambda_3 = 9 \quad &\leftrightarrow \quad y_3(x) = \sin 3x \\ &\vdots \end{aligned}$$

13.3. Making Connections

At this point I immediately began to wonder if there was a connection between the eigenvalues and eigenfunctions of the Sturm-Liouville problem and the eigenvalues and eigenvectors of matrices that I encountered in my linear algebra course.

Secondly, I began to wonder why many of the PDEs encountered at PCMI seemed to lead to this exact Sturm-Liouville problem (equation (13.6)), or at least to something remarkably similar. After sharing this observation with several of the resident PDE experts at PCMI, I was informed that there are very few Sturm-Liouville problems that possess exact analytical solutions. This is one of the reasons that we seemed to be doing the same problem over and over again.

However, the general Sturm-Liouville problem,

$$-(p(x)y')' + q(x)y = \lambda r(x)y, \tag{13.8}$$

with a variety of different boundary conditions (Direchlet, Neumann, Robin, Mixed, etc), must surely possess a wealth of different looking eigenvalue-eigenfunction pairs. How are we going to solve the general Sturm-Liouville problem if only a few of them seem to possess solutions that can be found with analytical techniques?

I found the answers to these questions in a wonderful paper entitled *The Schrodinger Equation*, written by Dr. John Polking at Rice University (`http://math.rice.edu/~polking`).

13.3.1. Approximating the Second Derivative

Recall that under certain conditions, we can use Taylor's Theorem to write

$$y(x+h) = y(x) + y'(x)h + \frac{y''(x)}{2!}h^2 + \cdots$$

and

$$y(x-h) = y(x) - y'(x)h + \frac{y''(x)}{2!}h^2 + \cdots .$$

Adding these equations,

$$y(x+h) + y(x-h) \approx 2y(x) + y''(x)h^2,$$

where we've thrown away the remaining higher order terms. Solving for $y''(x)$ provides the approximation

$$y''(x) \approx \frac{y(x-h) - 2y(x) + y(x+h)}{h^2}. \tag{13.9}$$

Substituting this approximation for the second derivative into the Sturm-Liouville equation (13.6), we get

$$-\frac{y(x-h) - 2y(x) + y(x+h)}{h^2} \approx \lambda y(x). \tag{13.10}$$

13.3.2. Partitioning $[0, \pi]$

We now partition the interval (see Figure 1) on which we seek the solution of our Sturm-Liouville problem. The distance between consecutive

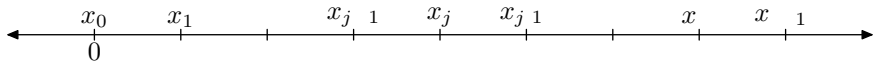

Figure 1. Partitioning the solution interval $[0, \pi]$.

points on this interval is given by the following calculation.

$$h = \frac{\pi - 0}{N+1} = \frac{\pi}{N+1}$$

Due to the boundary conditions given in (13.6), the solution is known at each endpoint of this interval.

$$y(x_0) = y(0) = 0$$
$$y(x_{N+1}) = y(\pi) = 0$$

However, the solution is unknown at each of the N interior points

$$x_j = jh, \quad j = 1, 2, \ldots, N.$$

If we evaluate the difference equation (13.10) at x_j, then

$$-h^{-2}\big[y(x_j - h) - 2y(x_j) + y(x_j + h)\big] \approx \lambda y(x_j).$$

Because h is the increment between consecutive points on our partition, $x_j - h = x_{j-1}$ and $x_j + h = x_{j+1}$. Hence,

$$-h^{-2}\big[y(x_{j-1}) - 2y(x_j) + y(x_{j+1})\big] \approx \lambda y(x_j).$$

We now replace $y(x_j)$ with y_j and iterate the equation

$$-h^{-2}\big[y_{j-1} - 2y_j + y_{j+1}\big] = \lambda y_j \tag{13.11}$$

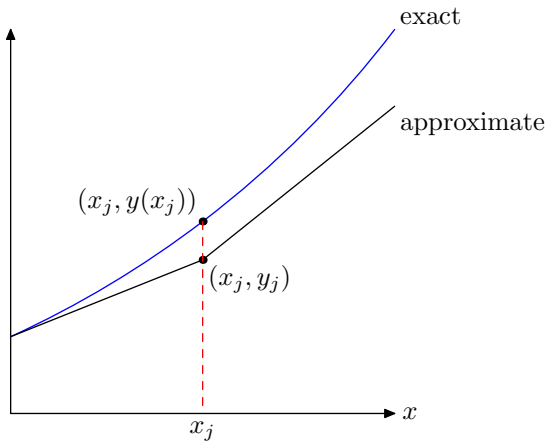

Figure 2. Note that y_j is an approximation of $y(x_j)$.

in an attempt to find an approximate solution to the Sturm-Liouville problem (13.6).

There is an important distinction to note before we continue. Note that $y(x_j)$ represents the *exact* value of the Sturm-Liouville solution evaluated at x_j, but the variable y_j represents the approximation of $y(x_j)$ at x_j. This is depicted nicely in Figure 2, where the exact and approximate solutions of an imaginary Sturm-Liouville problem are shown.

At this point, we will evaluate equation (13.11) at each interior point of the partition on $[0, \pi]$; that is, at x_j for $j = 1, 2, \ldots, N$. It is instructive to do this for a small value of N, say $N = 5$.

For $j = 1$, equation (13.11) becomes

$$-h^{-2}\big[y_0 - 2y_1 + y_2\big] = \lambda y_1. \tag{13.12}$$

Similarly, for $j = 2$, 3, and 4, we get

$$\begin{aligned} -h^{-2}\big[y_1 - 2y_2 + y_3\big] &= \lambda y_2, \\ -h^{-2}\big[y_2 - 2y_3 + y_4\big] &= \lambda y_3, \\ -h^{-2}\big[y_3 - 2y_4 + y_5\big] &= \lambda y_4. \end{aligned}$$

Finally, for $j = 5$, we get

$$-h^{-2}\big[y_4 - 2y_5 + y_6\big] = \lambda y_5. \tag{13.13}$$

However, the boundary conditions require

$$\begin{aligned} y_0 &= y(x_0) = y(0) = 0, \\ y_6 &= y(x_6) = y(\pi) = 0, \end{aligned}$$

so equation (13.12) becomes $-h^{-2}\big[-2y_1+y_2\big]=\lambda y_1$, while equation (13.13) becomes $-h^{-2}\big[y_4-2y_5\big]=\lambda y_5$. This leads to the following system of five equations in five unknowns.

$$\begin{aligned}
2h^{-2}y_1-h^{-2}y_2&=\lambda y_1,\\
-h^{-2}y_1+2h^{-2}y_2-h^{-2}y_3&=\lambda y_2,\\
-h^{-2}y_2+2h^{-2}y_3-h^{-2}y_4&=\lambda y_3,\\
-h^{-2}y_3+2h^{-2}y_4-h^{-2}y_5&=\lambda y_4,\\
-h^{-2}y_4+2h^{-2}y_5&=\lambda y_5.
\end{aligned}$$

This system is unenlightening until it is placed in matrix form.

$$\begin{bmatrix} 2h^{-2} & -h^{-2} & 0 & 0 & 0\\ -h^{-2} & 2h^{-2} & -h^{-2} & 0 & 0\\ 0 & -h^{-2} & 2h^{-2} & -h^{-2} & 0\\ 0 & 0 & -h^{-2} & 2h^{-2} & -h^{-2}\\ 0 & 0 & 0 & -h^{-2} & 2h^{-2} \end{bmatrix}\begin{bmatrix} y_1\\ y_2\\ y_3\\ y_4\\ y_5 \end{bmatrix}=\lambda\begin{bmatrix} y_1\\ y_2\\ y_3\\ y_4\\ y_5 \end{bmatrix} \tag{13.14}$$

System (13.14) has the form

$$M\mathbf{y}=\lambda\mathbf{y},$$

where

$$M=\begin{bmatrix} 2h^{-2} & -h^{-2} & 0 & 0 & 0\\ -h^{-2} & 2h^{-2} & -h^{-2} & 0 & 0\\ 0 & -h^{-2} & 2h^{-2} & -h^{-2} & 0\\ 0 & 0 & -h^{-2} & 2h^{-2} & -h^{-2}\\ 0 & 0 & 0 & -h^{-2} & 2h^{-2} \end{bmatrix}\quad\text{and}\quad\mathbf{y}=\begin{bmatrix} y_1\\ y_2\\ y_3\\ y_4\\ y_5 \end{bmatrix}.$$

Aha! We've established a relationship between the problem of finding eigenvalues and eigenfunctions of the Sturm-Liouville problem, $-y''(x)=\lambda y(x)$, $y(0)=y(\pi)=0$, to a problem of finding the eigenvalues and eigenvectors of the matrix equation $M\mathbf{y}=\lambda\mathbf{y}$.

Let's use Matlab to find the eigenvalues and eigenvectors of the coefficient matrix of system (13.14). It is helpful to factor h^{-2} from the matrix M and note that

$$M=h^{-2}\begin{bmatrix} 2 & -1 & 0 & 0 & 0\\ -1 & 2 & -1 & 0 & 0\\ 0 & -1 & 2 & -1 & 0\\ 0 & 0 & -1 & 2 & -1\\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}.$$

Note that the resulting matrix is *tridiagonal*. Each entry on the main diagonal is a 2, while each entry in the first sub and supra diagonals is a -1. It is an easy matter to construct such a matrix in Matlab. To

form a matrix with five 2's on its main diagonal, enter the following command.

```
>> diag([2,2,2,2,2])
ans =
     2     0     0     0     0
     0     2     0     0     0
     0     0     2     0     0
     0     0     0     2     0
     0     0     0     0     2
```

To form a diagonal matrix with 1's on the first subdiagonal, enter the following command.

```
>> diag([1,1,1,1],-1)
ans =
     0     0     0     0     0
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
```

Similarly, the command `diag([1,1,1,1],1)` will create a matrix with 1's on the first supra diagonal. This should give us enough information to build the matrix M.

```
>> M=diag([2,2,2,2,2])-diag([1,1,1,1],-1)-diag([1,1,1,1],1)
M =
     2    -1     0     0     0
    -1     2    -1     0     0
     0    -1     2    -1     0
     0     0    -1     2    -1
     0     0     0    -1     2
```

It should now be an easy matter to calculate the eigenvalues and eigenvectors of matrix M. Of course, we must first scale matrix M by h^{-2}.

```
>> N=5;
>> h=(pi-0)/(N+1);
>> M=diag([2,2,2,2,2])-diag([1,1,1,1],-1)-diag([1,1,1,1],1);
>> M=h^(-2)*M;
>> [v,e]=eig(M)
v =
   -0.2887   -0.5000    0.5774   -0.5000    0.2887
   -0.5000   -0.5000    0.0000    0.5000   -0.5000
   -0.5774    0.0000   -0.5774    0.0000    0.5774
   -0.5000    0.5000   -0.0000   -0.5000   -0.5000
```

```
   -0.2887    0.5000    0.5774    0.5000    0.2887
e =
    0.9774         0         0         0         0
         0    3.6476         0         0         0
         0         0    7.2951         0         0
         0         0         0   10.9427         0
         0         0         0         0   13.6129
```

Equation (13.7) predicts that the first five eigenvalues should be $\lambda_n = n^2$, $n = 1, 2, \ldots, 5$; that is, 1, 4, 9, 16, and 25. The eigenvalues of matrix M are reported down the main diagonal of matrix `e`. They are 0.9774 (close to 1), 3.6476 (somewhat close to 4), 7.2951 (not very close to 9), then 10.9427, and 13.6129, which are not close to the expected values of 16 and 25.

In matrix `v`, the first column is

$$\begin{bmatrix} -0.2887 \\ -0.5000 \\ -0.5774 \\ -0.5000 \\ -0.2887 \end{bmatrix},$$

which is the eigenvector associated with the first eigenvalue on the diagonal of matrix `e`, namely, 0.9774. This eigenvector is an approximation of the first eigenfunction of the Sturm-Liouville problem (13.6), as we shall soon see.

13.3.3. Finer Partitions

Now, how can we make these approximations better? You probably have guessed that increasing the value of N will create a finer partition of the interval $[0, \pi]$ shown in Figure 1. The hope is that this finer partition will generate a more accurate approximation of the eigenvalues and eigenfunctions of the Sturm-Liouville problem (13.6).

The general system of N equations has matrix form

$$\begin{bmatrix} 2h^{-2} & -h^{-2} & 0 & \cdots & 0 \\ -h^{-2} & 2h^{-2} & h^{-2} & \cdots & 0 \\ 0 & -h^{-2} & 2h^{-2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2h^{-2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} = \lambda \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix}. \tag{13.15}$$

The only thing standing in our way of analyzing this system is the crafting of the coefficient matrix in Matlab. As N increases, the dimensions of the coefficient matrix increase, and it soon becomes painful

to enter the matrix M as we did for smaller dimensions. Fortunately, Matlab has a beautiful collection of routines for building matrices.

Consider the following command, which crafts a row of five ones.

```
>> ones(1,5)
ans =
     1     1     1     1     1
```

If we multiply this vector by 2, we have a row of twos.

```
>> 2*ones(1,5)
ans =
     2     2     2     2     2
```

Similarly, we can craft a row of four -1's with this command.

```
>> -1*ones(1,4)
ans =
    -1    -1    -1    -1
```

Thus, we could have built our 5×5 matrix M with this command.

```
>> M=diag(2*ones(1,5))-diag(ones(1,4),-1)-diag(ones(1,4),1)
M =
     2    -1     0     0     0
    -1     2    -1     0     0
     0    -1     2    -1     0
     0     0    -1     2    -1
     0     0     0    -1     2
```

Using this technique, we should find it easy to create a tridiagonal matrix of arbitrary size with 2's down the main diagonal and -1's down the first sub and supra diagonals. Let's increase N to 20.

```
>> N=20;
>> h=(pi-0)/(N+1);
>> M=diag(2*ones(1,N))-diag(ones(1,N-1),-1)-diag(ones(1,N-1),1);
>> M=h^(-2)*M;
>> [v,e]=eig(M);
```

Unlike Maple, the semicolons at the end of each of these commands is not required. Rather, in Matlab, the semicolon is used to suppress the output of a command. Try removing the semicolon on the last command, `[v,e]=eig(M)` to see the output of this command.

In this case, `e` is a 20×20 matrix. This can be checked with Matlab's `size` command.

```
>> size(e)
ans =
    20    20
```

As we've seen in the $N = 5$ case, the eigenvalues are located on the diagonal of the matrix `e`. We can extract the diagonal of eigenvalues with the multifaceted `diag` command.

```
>> d=diag(e);
```

The variable `d` is now a vector containing the eigenvalues.

```
>> size(d)
ans =
    20     1
```

Let's list the first five eigenvalues using Matlab's indexing capability.

```
>> d(1:5)
ans =
    0.9981
    3.9702
    8.8499
   15.5282
   23.8559
```

Note that these values are much closer to the predicted eigenvalues 1, 4, 9, 16, and 25.

Clearly, we can continue in this manner, increasing N to create a finer partition of the interval $[0, \pi]$ in order to better approximate the eigenvalues and eigenfunctions of the Sturm-Liouville problem (13.6), but only at the cost of increased computer time. For example, try running the algorithm with $N = 1000$.

```
>> N=1000;
>> h=(pi-0)/(N+1);
>> M=diag(2*ones(1,N))-diag(ones(1,N-1),-1)-diag(ones(1,N-1),1);
>> M=h^(-2)*M;
>> tic, [v,e]=eig(M); toc
elapsed_time =
   69.9030
```

Matlab provides the construct `tic ... toc` to enable the user to see the amount of time required for the included command to complete,

in this case, approximately 70 seconds.[4] Let's look at the first five eigenvalues.

```
>> d=diag(e);
>> d(1:5)
ans =
    1.0000
    4.0000
    8.9999
   15.9998
   24.9995
```

These are very close to the predicted eigenvalues 1, 4, 9, 16, and 25.

13.3.4. The Eigenfunctions

Each column of the matrix `v` contains the eigenvector associated with the eigenvalue in the corresponding diagonal position of the matrix `e`. Each of these eigenvectors is an approximation of the corresponding eigenfunction of the Sturm-Louville problem.

Note that each eigenvector is a solution of the general equation (13.15), so that each eigenvector contains approximations y_1, y_2, ..., y_N to the eigenfunction only at the points x_1, x_2, ..., x_N. The eigenvectors do not contain an approximation of the eigenfunction at x_0 nor at x_{N+1}. However, we have the Dirchlet conditions at each endpoint; that is,

$$y_0 = y(x_0) = y(0) = 0,$$
$$y_N = y(x_n) = y(\pi) = 0.$$

Consequently, each of the eigenvectors in `v` needs a zero prepended to the beginning of the vector, and in addtion, a zero appended to the end of the vector. We can easily accomplish this amendment for each eigenvector in the matrix `v` with the following commands.

```
>> VV=[zeros(1,N); v; zeros(1,N)];
```

This command warrants some explanation. The command `zeros(1,N)` creates a $1 \times N$ row vector where each entry is a zero. Further, semicolons delimit rows when building matrices. Thus, the command

[4]One can greatly improve the performance by taking advantage of Matlab's sparse matrix routines. A sparse matrix contains very few nonzero entries. In the case of matrix M, which is 1000×1000, the main diagonal contains 1000 nonzero entries, while the diagonals immediately above and below the main diagonal each contain 999 nonzero entries. A quick calculation reveals that 99.7% of the entries of matrix M are zeros. Try the command `opts.disp=0`, followed by the command `tic, [v,e]=eigs(M,20,'SM',opts); toc` to see the improvement in performance. On our system, the calculation took about 0.49 seconds.

`[zeros(1,N); v; zeros(1,N)]` builds a new matrix by creating a first row with all zeros, followed by the rows of the matrix `v`, then concluding by appending another row of zeros. Therefore, each column of `VV` now contains a zero as its first entry, followed by an eigenvector of equation (13.15), then a final zero. In this way, we attach the Dirichlet conditions at each end of our eigenvector approximation for the eigenfunction of the Sturm-Liouville problem (13.6).

Now, let's plot a few of the eigenvector approximations of the eigenfunctions. Note that each column of `VV` at this point contains entries y_0, y_1, $\ldots y_{N+1}$, thus $N+2$ entries in all. We want to plot these entries versus x_0, x_1, $\ldots x_{N+1}$. Therefore, the first task is to create a vector containing these x-values of the partition on $[0, \pi]$.

In Matlab, the construct `start:increment:finish` is used to create sequences of numbers beginning with `start`, ending with `finish`, and proceeding in steps of `increment`. For example, the command

```
>> 0:0.5:2
ans =
         0    0.5000    1.0000    1.5000    2.0000
```

is used to create a vector of numbers that start at 0, end at 2, proceeding in steps of 0.5. If no increment is used, a default increment of 1 is ued.

```
>> 0:5
ans =
     0     1     2     3     4     5
```

The points in our partition are given by the formula $x_j = jh$, which was earlier computed with $h = \pi/(N+1)$. Thus, the command

```
>> xx=(0:N)*h;
```

should create a vector containing x_0, x_1, $\ldots$, x_N, then the command

```
>> xx=[xx,pi];
```

should append the endpoint of the interval $[0, \pi]$. Alternatively, you might try `xx=(0:N+1)*h`, but the technique used above helps eliminate roundoff error at the right endpoint of the interval $[0, \pi]$.

It is now a simple matter to plot the the approximations of the eigenfunctions versus `xx`. For example, the following command was used to plot the first two eigenfunctions shown in Figure 3.

```
>> plot(xx,VV(:,[1,2]))
>> grid on
>> xlabel('x')
>> legend('\lambda = 1', '\lambda = 4',4)
```

A word of explanation is in order. The syntax `VV(:,[1,2])` is Matlab indexing that some read as "Matrix VV, every row, first and second columns." We could just as easily have entered `VV(1:N+2,[1,2])`, which calls for rows 1 through $N+2$, first and second columns, but the colon syntax, meaning "every row," is more convenient.

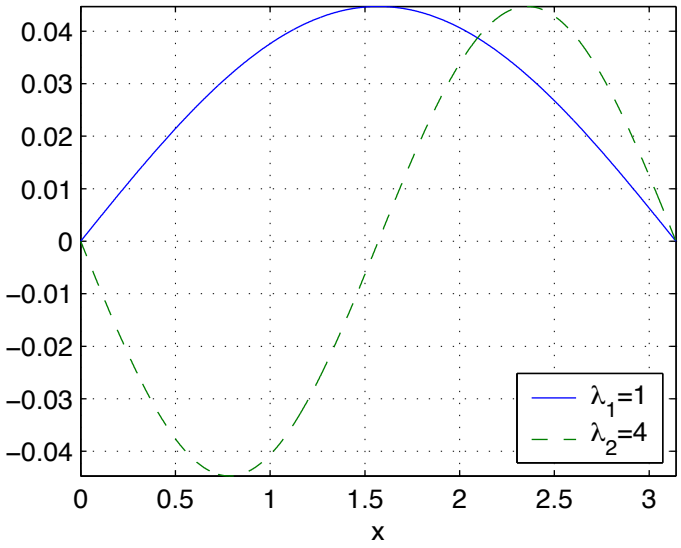

Figure 3. Eigenfunctions associated with $\lambda = 1$ and $\lambda = 4$.

The astute reader will remember that we are expecting the first two eigenvalue-eigenfunction pairs.

$$\begin{aligned} \lambda_1 = 1 \quad &\leftrightarrow \quad y_1(x) = \sin x \\ \lambda_2 = 4 \quad &\leftrightarrow \quad y_2(x) = \sin 2x \end{aligned}$$

Both $y_1(x) = \sin x$ and $y_2(x) = \sin 2x$ have amplitude 1, but the eigenvector approximations in Figure 3 appear to have an amplitude approximately equal to 0.045. Moreover, the second eigenvector approximation appears to be inverted.

To allay our fears that things are not working as they should, we need only remember that any scalar multiple of an eigenfunction is again an eigenfunction. That is, the eigenfunction approximations pictured in Figure 3 should be scalar multiples of $y_1(x) = \sin x$ and $y_2(x) = \sin 2x$. To see this, we will scale our eigenvector approximations, then compare them with $y_1(x) = \sin x$ and $y_2(x) = \sin 2x$.

We wish our first eigenvector approximation to have amplitude 1. This is easily accomplished by dividing each entry of the eigenvector by

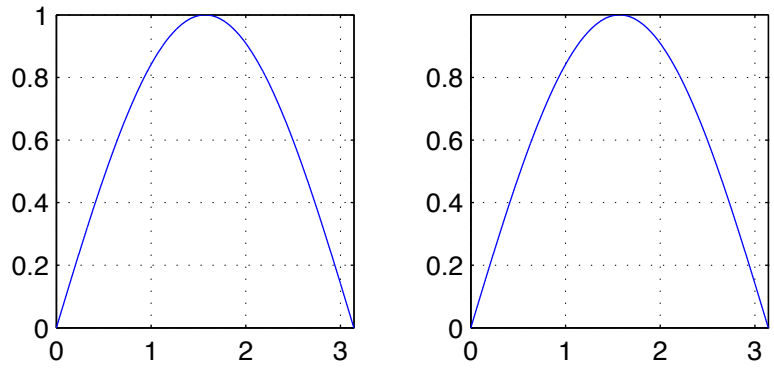

Figure 4. Comparing a scalar multiple of the first eigenvector versus $y_1(x) = \sin x$ on $[0, \pi]$.

the magnitude of the largest entry present. The following commands were used to produce the image on the left in Figure 4.[5]

```
>> plot(xx,VV(:,1)/max(VV(:,1)))
>> grid on
>> xlabel('x')
```

For comparison, these commands were used to plot $y_1(x) = \sin x$ on the interval $[0, \pi]$. This image is shown on the right in Figure 4.

```
>> plot(xx,sin(xx))
>> grid on
>> xlabel('x')
```

13.4. Neumann Conditions

Let's revisit the Sturm-Liouville equation (13.6), but this time let's attach a Neumann condition at the right endpoint.

$$-y''(x) = \lambda y(x), \qquad y(0) = 0, \qquad y'(\pi) = 0 \tag{13.16}$$

It is not difficult to show that the eigenvalues and eigenfunctions of the Sturm-Liouville problem (13.16) are given by

$$\lambda_n = (n - 1/2)^2 \qquad \text{and} \qquad y_n(x) = \sin((n - 1/2)x),$$

for $n = 1, 2, 3, \ldots$. Accordingly, the first five eigenvalues should be 0.25, 2.25, 6.25, 12.25, and 20.25.

[5]Another idea would be to divide each eigenvector by the infimum norm with the command `VV(:,1)/norm(VV(:,1),inf)`.

We still have the same Sturm-Liouville equation, $-y''(x) = \lambda y(x)$, so we will still use equation (13.11), repeated here for convenience.

$$-h^{-2}\big[y_{j-1} - 2y_j + y_{j+1}\big] = \lambda y_j$$

Because of the Neumann condition $y'(\pi) = 0$, we do not know the value of the solution at the right endpoint; i.e., we do not know $y(\pi)$. Therefore, if we again partition the interval as in Figure 1 with $N = 5$, we will need to compute y_1, y_2, …, y_6. This will necessitate the addition of an extra equation. With $j = 1, 2, \ldots, 6$,

$$\begin{aligned}
-h^{-2}\big[y_0 - 2y_1 + y_2\big] &= \lambda y_1,\\
-h^{-2}\big[y_1 - 2y_2 + y_3\big] &= \lambda y_2,\\
-h^{-2}\big[y_2 - 2y_3 + y_4\big] &= \lambda y_3,\\
-h^{-2}\big[y_3 - 2y_4 + y_5\big] &= \lambda y_4,\\
-h^{-2}\big[y_4 - 2y_5 + y_6\big] &= \lambda y_5,\\
-h^{-2}\big[y_5 - 2y_6 + y_7\big] &= \lambda y_6.
\end{aligned} \tag{13.17}$$

However, this gives us six equations in eight unknowns. Because of the Dirichlet condition at the left endpoint of $[0, \pi]$, we know that

$$y_0 = y(x_0) = y(0) = 0,$$

so that eliminates the unknown y_0. However, we will need to obtain an estimate for y_7.

We can use a forward difference estimate for the first derivative, defined by

$$y'(x) \approx \frac{y(x+h) - y(x)}{h}.$$

Using h as the step size in our partition of $[0, \pi]$, we can write

$$y'(x_j) \approx \frac{y(x_{j+1}) - y(x_j)}{h}.$$

Using y_j as an approximation of $y(x_j)$, we write

$$y'_j = \frac{y_{j+1} - y_j}{h},$$

or, equivalently,

$$y_{j+1} = y_j + hy'_j.$$

With $j = 6$, this gives us $y_7 = y_6 + hy'_6$. However, because of the Neumann condition on the right endpoint of $[0, \pi]$, we know that

$$y'_6 = y'(x_6) = y'(\pi) = 0.$$

Thus, we get $y_7 = y_6$.

Substituting $y_0 = 0$ and $y_7 = y_6$ into the first and last equations of system (13.17) gives us six equations in six unknowns.

$$\begin{aligned}
-h^{-2}\big[-2y_1 + y_2\big] &= \lambda y_1,\\
-h^{-2}\big[y_1 - 2y_2 + y_3\big] &= \lambda y_2,\\
-h^{-2}\big[y_2 - 2y_3 + y_4\big] &= \lambda y_3,\\
-h^{-2}\big[y_3 - 2y_4 + y_5\big] &= \lambda y_4,\\
-h^{-2}\big[y_4 - 2y_5 + y_6\big] &= \lambda y_5,\\
-h^{-2}\big[y_5 - y_6\big] &= \lambda y_6.
\end{aligned}$$

Again, this doesn't mean much until we see this system in matrix form.

$$\begin{bmatrix} 2h^{-2} & -h^{-2} & 0 & 0 & 0 & 0\\ -h^{-2} & 2h^{-2} & -h^{-2} & 0 & 0 & 0\\ 0 & -h^{-2} & 2h^{-2} & -h^{-2} & 0 & 0\\ 0 & 0 & -h^{-2} & 2h^{-2} & -h^{-2} & 0\\ 0 & 0 & 0 & -h^{-2} & 2h^{-2} & -h^{-2}\\ 0 & 0 & 0 & 0 & -h^{-2} & h^{-2} \end{bmatrix} \begin{bmatrix} y_1\\ y_2\\ y_3\\ y_4\\ y_5\\ y_6 \end{bmatrix} = \lambda \begin{bmatrix} y_1\\ y_2\\ y_3\\ y_4\\ y_5\\ y_6 \end{bmatrix} \tag{13.18}$$

Note that this matrix equation again has the form $M\mathbf{v} = \lambda\mathbf{v}$. Once again, finding the eigenvaues and eigenfunctions of the Sturm-Liouville problem (13.16) has been reduced to finding the eigenvalues and eigenvectors of a matrix.

Again, it is helpful to factor out the h^{-2} from the coefficient matrix M.

$$M = h^{-2}\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0\\ -1 & 2 & -1 & 0 & 0 & 0\\ 0 & -1 & 2 & -1 & 0 & 0\\ 0 & 0 & -1 & 2 & -1 & 0\\ 0 & 0 & 0 & -1 & 2 & -1\\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

We need only make a minor adjust to our Matlab strategy to build this matrix.

```
>> M=diag([2*ones(1,5),1])-diag(ones(1,5),-1)-diag(ones(1,5),1)
M =
     2    -1     0     0     0     0
    -1     2    -1     0     0     0
     0    -1     2    -1     0     0
     0     0    -1     2    -1     0
     0     0     0    -1     2    -1
     0     0     0     0    -1     1
```

The construct `[2*ones(1,5),1]` creates a vector with five 2's, then appends one additional 1, which is exactly what we need on the main diagonal.

The general system of $N+1$ equations has matrix form

$$\begin{bmatrix} 2h^{-2} & -h^{-2} & 0 & \cdots & 0 \\ -h^{-2} & 2h^{-2} & h^{-2} & \cdots & 0 \\ 0 & -h^{-2} & 2h^{-2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & h^{-2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N+1} \end{bmatrix} = \lambda \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N+1} \end{bmatrix}. \tag{13.19}$$

Note that we're solving for $y_1, y_2, \ldots, y_{N+1}$ this time.

Let's jump right to the case with $N = 1000$.

```
>> N=1000;
>> h=pi/(N+1);
>> M=diag([2*ones(1,N),1])-diag(ones(1,N),-1)-diag(ones(1,N),1);
>> M=h^(-2)*M;
>> tic, [v,e]=eig(M); toc
elapsed_time =
   67.3190
```

Strip the eigenvalues from the main diagonal of `e`.

```
>> d=diag(e);
>> d(1:5)
ans =
    0.2498
    2.2477
    6.2437
   12.2376
   20.2294
```

Note the close agreement with the exact eigenvalues determined by $\lambda_n = (n - 1/2)^2$; i.e., 0.25, 2.25, 6.25, 12.25, and 20.25.

The Dirichlet condition $y(0) = 0$ at the left endpoint requires that we prepend a zero to each eigenvector. This is easily accomplished by adding a first row of zeros to the matrix `v`.

```
>> VV=[zeros(1,N+1);v];
```

We create a vector of the mesh points on the partition of $[0, \pi]$ as before.

```
>> xx=(0:N)*h;
>> xx=[xx,pi];
```

We can now plot the eigenvectors in `VV` versus `xx`. For example, to plot the first two eigenvectors, execute the following commands to obtain a plot similar to that shown in Figure 5.

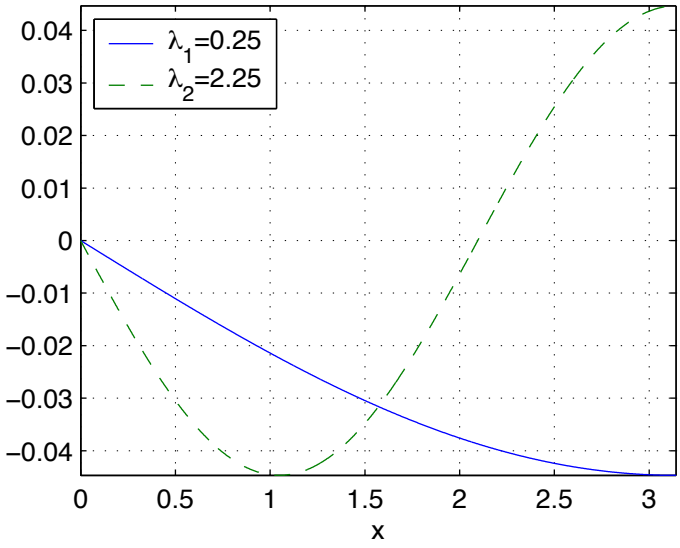

Figure 5. The first two eigenfunctions.

```
>> plot(xx,VV(:,[1,2]))
>> grid on
>> xlabel('x')
>> legend('\lambda_1=0.25','\lambda_2=2.25',2)
```

In Figure 5, note that each eigenfunction begins at $(0, 0)$. Also, at the right endpoint, note that each eigenfunction levels, eventually reaching slope zero, as required by the Neumann condition at the right endpoint $(y'(\pi) = 0)$.

13.5. One Final Example

Consider the Sturm-Liouville problem

$$-y''(x) + 3xy(x) = \lambda y(x), \quad y'(0) = 0, \quad y(\pi) = 0. \tag{13.20}$$

You might note that this has the form $-(p(x)y')' + q(x)y = \lambda r(x)y$, with $p(x) = 1$, $q(x) = 3x$, and $r(x) = 1$. You might also try your hand at solving this problem analytically. We will provide a numerical solution using the same methods as in our previous examples.

Again, if we estimate the second derivative with

$$y''(x_j) = \frac{y(x_{j-1}) - 2y(x_j) + y(x_{j+1})}{h^2},$$

then the equation $-y''(x) + 3xy(x) = \lambda y(x)$ becomes

$$-h^{-2}\big[y(x_{j-1}) - 2y(x_j) + y(x_{j+1})\big] + 3x_j y(x_j) = \lambda y(x_j).$$

With $y_j \approx y(x_j)$ and a little algebra,

$$-h^{-2}y_{j-1} + (2h^{-2} + 3x_j)y_j - h^{-2}y_{j+1} = \lambda y_j. \tag{13.21}$$

In this example the value of y is unknown at the left endpoint of $[0, \pi]$, but known at the right endpoint. Thus, with $N = 5$, we will want to compute the value of y at x_0, x_1, $\ldots$, x_5. Consequently, we substitute $j = 0, 1, \ldots, 5$ in equation (13.21) to obtain six equations in eight unknowns.

$$\begin{aligned}
-h^{-2}y_{-1} + (2h^{-2} + 3x_0)y_0 - h^{-2}y_1 &= \lambda y_0 \\
-h^{-2}y_0 + (2h^{-2} + 3x_1)y_1 - h^{-2}y_2 &= \lambda y_1 \\
-h^{-2}y_1 + (2h^{-2} + 3x_2)y_2 - h^{-2}y_3 &= \lambda y_2 \\
-h^{-2}y_2 + (2h^{-2} + 3x_3)y_3 - h^{-2}y_4 &= \lambda y_3 \\
-h^{-2}y_3 + (2h^{-2} + 3x_4)y_4 - h^{-2}y_5 &= \lambda y_4 \\
-h^{-2}y_4 + (2h^{-2} + 3x_5)y_5 - h^{-2}y_6 &= \lambda y_5
\end{aligned} \tag{13.22}$$

However, the Dirichlet condition at the right endpoint of $[0, \pi]$ provides

$$y_6 = y(x_6) = y(\pi) = 0. \tag{13.23}$$

We need to find an estimate for y_{-1}. We're given a Neumann condition at the left endpoint of the interval. We'll use a backward difference to estimate the first derivative of y at the left endpoint of $[0, \pi]$.

$$y'(x) = \frac{y(x) - y(x - h)}{h}$$

Again, h is the step size on our partition of $[0, \pi]$, so we may write

$$y'(x_j) = \frac{y(x_j) - y(x_{j-1})}{h}.$$

Use y_j as an approximation of $y(x_j)$ and write

$$y'_j = \frac{y_j - y_{j-1}}{h},$$

or equivalently,

$$y_{j-1} = y_j - hy'_j.$$

With $j = 0$, this last equation gives us $y_{-1} = y_0 - hy'_0$. However, the Neumann condition on the left endpoint of $[0, \pi]$ provides $y'_0 = y'(x_0) = y'(0) = 0$, so

$$y_{-1} = y_0. \tag{13.24}$$

Substituting (13.24) and (13.23) into system (13.22), we arrive at six equations in six unknowns y_0, y_1, ..., y_5.

$$
\begin{aligned}
(h^{-2}+3x_0)y_0 - h^{-2}y_1 &= \lambda y_0 \\
-h^{-2}y_0 + (2h^{-2}+3x_1)y_0 - h^{-2}y_2 &= \lambda y_1 \\
-h^{-2}y_1 + (2h^{-2}+3x_2)y_1 - h^{-2}y_3 &= \lambda y_2 \\
-h^{-2}y_2 + (2h^{-2}+3x_3)y_2 - h^{-2}y_4 &= \lambda y_3 \\
-h^{-2}y_3 + (2h^{-2}+3x_4)y_3 - h^{-2}y_5 &= \lambda y_4 \\
-h^{-2}y_4 + (2h^{-2}+3x_5)y_5 &= \lambda y_5
\end{aligned}
\tag{13.25}
$$

This system can be written in the matrix form $M\mathbf{y} = \lambda\mathbf{y}$, with coefficient matrix

$$
\begin{bmatrix}
h^{-2}+3x_0 & -h^{-2} & 0 & 0 & 0 & 0 \\
-h^{-2} & 2h^{-2}+3x_1 & -h^{-2} & 0 & 0 & 0 \\
0 & -h^{-2} & 2h^{-2}+3x_2 & -h^{-2} & 0 & 0 \\
0 & 0 & -h^{-2} & 2h^{-2}+3x_3 & -h^{-2} & 0 \\
0 & 0 & 0 & -h^{-2} & 2h^{-2}+3x_4 & -h^{-2} \\
0 & 0 & 0 & 0 & -h^{-2} & 2h^{-2}+3x_5
\end{bmatrix}.
$$

It is helpful to factor out h^{-2}. Then the coefficient matrix becomes

$$
h^{-2}\begin{bmatrix}
1+3h^2x_0 & -1 & 0 & 0 & 0 & 0 \\
-1 & 2+3h^2x_1 & -1 & 0 & 0 & 0 \\
0 & -1 & 2+3h^2x_2 & -1 & 0 & 0 \\
0 & 0 & -1 & 2+3h^2x_3 & -1 & 0 \\
0 & 0 & 0 & -1 & 2+3h^2x_4 & -1 \\
0 & 0 & 0 & 0 & -1 & 2+3h^2x_5
\end{bmatrix}.
$$

As one might expect, building this matrix in Matlab is a bit more involved, but you will see that it is not difficult. We will again work with $N = 5$.

```
>> N=5;
```

Determine the step size of the partiton 13.3.2.

```
>> h=pi/(N+1);
```

Now, determine the points of the partition. Note that the values x_0, x_1, ..., x_5 run down the main diagonal of the coefficient matrix, so those are the ones we need at the moment.

```
>> xx=(0:N)*h;
```

Let's first create a diagonal matrix with main diagonal containing a 1, followed by five 2's.

```
>> diag([1,2*ones(1,N)])
ans =
     1     0     0     0     0     0
```

```
     0     2     0     0     0     0
     0     0     2     0     0     0
     0     0     0     2     0     0
     0     0     0     0     2     0
     0     0     0     0     0     2
```

We will need to add to this to a matrix having entries $3h^2x_0, \ldots, 3h^2x_5$ on its main diagonal. Again, this is an easy construct using Matlab's `diag` command. Remember that the vector `xx` already contains the entries $x_0, \ldots, x_5$.

```
>> diag(3*h^2*xx)
ans =
         0         0         0         0         0         0
         0    0.4306         0         0         0         0
         0         0    0.8613         0         0         0
         0         0         0    1.2919         0         0
         0         0         0         0    1.7226         0
         0         0         0         0         0    2.1532
```

You might want to check these entries with a calculator.

We can use these ideas, along with some former ideas, to construct our coefficient matrix M.

```
>> M=diag([1,22*ones(1,N)]);
>> M=M+diag(3*h^2*xx);
>> M=M-diag(ones(1,N),-1)-diag(ones(1,N),1)
M =
    1.0000   -1.0000         0         0         0         0
   -1.0000    2.4306   -1.0000         0         0         0
         0   -1.0000    2.8613   -1.0000         0         0
         0         0   -1.0000    3.2919   -1.0000         0
         0         0         0   -1.0000    3.7226   -1.0000
         0         0         0         0   -1.0000    4.1532
```

Let's skip right to the case we've used before and set $N = 1000$. It is a simple matter to repeat the steps taken above with this new value of N.

```
>> N=1000;
>> h=pi/(N+1);
>> xx=(0:N)*h;
>> M=diag([1,2*ones(1,N)]);
>> M=M+diag(3*h^2*xx);
>> M=M-diag(ones(1,N),-1)-diag(ones(1,N),1);
```

In addition, we want to remember to scale M by h^{-2}.

```
>> M=h^(-2)*M;
```

Finally, find the eigenvalues and eigenvectors.

```
>> [v,e]=eig(M);
```

As we see in system (13.25), in this case our eigenvectors contain the approximations $y_0, \ldots, y_N$ at the points $x_0, \ldots, x_N$. The Dirichlet condition at the right endpoint of the solution interval $[0, \pi]$ requires that $y_{N+1} = y(x_{N+1}) = y(\pi) = 0$. Therefore, we append a row of zeros to the eigenvector matrix `v`.

```
>> VV=[v;zeros(1,N+1)];
```

To obtain plots of the eigenfunctions, we must append $x_{N+1} = \pi$ to our partition of $[0, \pi]$.

```
>> xx=[xx,pi];
```

We can now plot the first two eigenfunctions. The following commands were used to create the eigenfunctions shown in Figure 6.

```
>> plot(xx,VV(:,[1,2]))
>> xlabel('x')
>> legend('\lambda_1','\lambda_2',4)
>> grid on
```

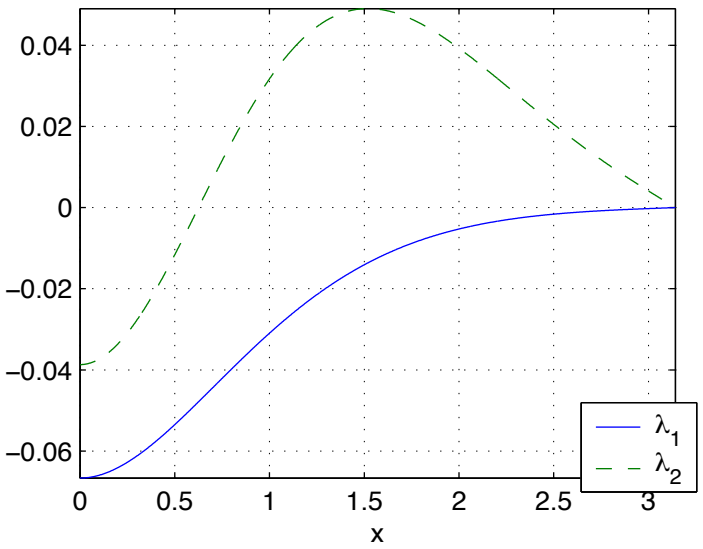

Figure 6. The first two eigenfunctions for $-y''(x) + 3xy(x) = \lambda y(x)$.

Note that both eigenfunctions in Figure 6 satisfy the Dirichlet condition $y(\pi) = 0$ at the right endpoint of the interval $[0, \pi]$. Furthermore, note that the Neumann condition $y'(0) = 0$ is satisfied at the left

endpoint of the solution interval, as both eigenfunctions seem to have tangent lines with slope zero at $x = 0$.

It's interesting to examine eigenfunctions two at a time and note that they display the same endpoint conditions described in the previous paragraph. For example, the following code will plot the third and fourth eigenfunctions, as shown in Figure 7.

```
>> plot(xx,VV(:,[3,4]))
>> xlabel('x')
>> legend('\lambda_3','\lambda_4')
>> grid on
```

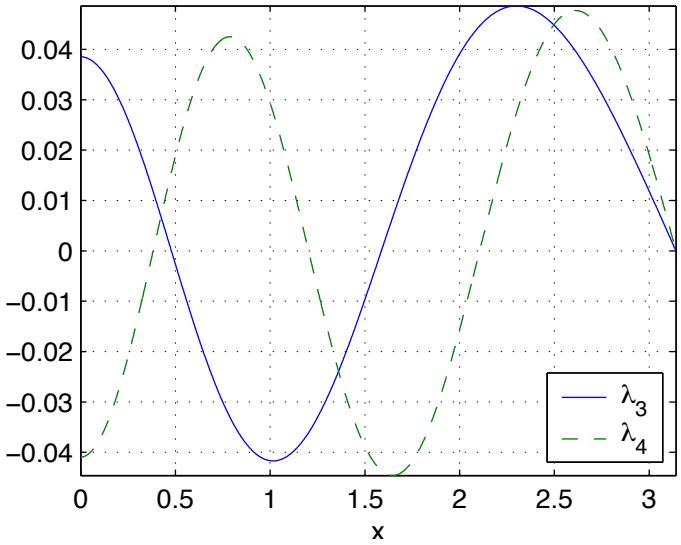

Figure 7. Third and fourth eigenfunctions of $-y''(x) + 3xy(x) = \lambda y(x)$.

13.6. Automating the Solution of Sturm-Liouville Problems

One can take the ideas developed in this paper and automate the process of extracting the eigenvalues and eigenfunctions of the Sturm-Liouville problem. Dr. John Polking from Rice University has done this in a suite of Matlab programs. The programs in the suite are named: `sls`, `slview`, and `slseries`.

The first of these programs, `sls`, will find the eigenvalues and eigenfunctions of an arbitrary regular Sturm-Liouville problem on the interval $I = [a, b]$, having the form

$$-(p(x)y')' + q(x)y = \lambda r(x)y, \tag{13.26}$$

with boundary conditions

$$\begin{aligned} \alpha_1 y'(a) + \alpha_1 y(a) &= 0, \\ \beta_1 y'(b) + \beta_2 y(b) &= 0. \end{aligned} \tag{13.27}$$

The second program in the suite, `slview`, allows the user to view the output of the `sls` routine. This routine will plot the eigenfunctions of the Sturm-Liouville problem, and an optional parameter K allows the users to view the eigenfunctions in groups of K. The default behavior is to view the plots of consecutive pairs of eigenfunctions.

The third and final routine in the suite, `slseries`, allows the user to view the expansion of an initial condition $f(x)$ in terms of the eigenfunctions of the Sturm-Liouville problem.

These and other programs written by Dr. Polking are avalaible for download at:

`http://www.math.hmc.edu/%7Eajb/PCMI/polking/polking.html`