Lecture 36: Boundary value problems

We want to adapt the approach we introduced last lecture to solving *boundary value problems*. Consider the following simple example:

$$u(0) = a$$
$$u(1) = b$$
$$u''(x) - a(x)u(x) = f(x)$$

Here is an example solution:

In [18]:

```
using ApproxFun

B=dirichlet()
x=Fun([0.,1.])
u=[B;Derivative([0.,1.])^2+1000x^2]\[1.,2.]
ApproxFun.plot(u)
```

Out[18]:

```
Fun([-1.56831,1.42181,0.396509,-2.73726,2.49329,0.521205,-1.24882
,1.76698,2.93991,0.328768  …  5.95226e-16,-8.27141e-15,-2.50338e-
15,-2.2721e-16,8.4236e-17,3.75925e-17,5.91565e-18,-4.23325e-19,-4
.3985e-19,-9.91228e-20],Chebyshev(【0.0,1.0】))
```

Unlike initial value problems, where the conditions $u(0) = a$ and $u'(0)$ are specified at a single point, in boundary value problems the conditions are specified at two *different* points. This means we can't view their solution as "time-stepping": we have to solve the problem globally. We will do so by using the approach advocated last lecture of constructing discrete derivatives.

Discrete Second Derivative

Recall the midpoint discrete derivative

$$D_n : \begin{array}{c}\text{Values at}\\ x_0, \ldots, x_n\end{array} \to \begin{array}{c}\text{Values at}\\ x_{1/2}, \ldots, x_{n-1/2}\end{array}$$

which is an $n \times n + 1$ matrix with entries

$$D_n \triangleq \frac{1}{h}\begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}$$

where $h = 1/n$ and $x_k = kh$. We will construct an approximate second derivative by now creating another midpoint discrete derivative

$$D_{n-1} : \begin{matrix}\text{Values at} \\ x_{1/2}, \ldots, x_{n-1/2}\end{matrix} \rightarrow \begin{matrix}\text{Values at} \\ x_1, \ldots, x_{n-1}\end{matrix},$$

which is $n - 1 \times n$.

Because the spacing between the nodes is still $h = 1/n$, when we approximate data at $x_{1/2}, \ldots, x_{n-1/2}$ by trapezoids, differentiate, and evaluate at the grid $x_1, \ldots, x_{n-1}$ we get the entries:

$$D_{n-1} \triangleq \frac{1}{h} \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}$$

where h is still $1/n$.

The $n - 1 \times n + 1$ discrete second derivative is then specified by

$$D_n^2 \triangleq D_{n-1} D_n.$$

This satisfies

$$D_n^2 : \begin{matrix}\text{Values at} \\ x_0, \ldots, x_n\end{matrix} \rightarrow \begin{matrix}\text{Values at} \\ x_1, \ldots, x_{n-1}\end{matrix},$$

that is, we map from all the nodes to the interior nodes. The entries are given by matrix multiplication as

$$D_n^2 \triangleq \frac{1}{h^2} \begin{pmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & -1 \end{pmatrix}$$

Remark Matrices with constant diagonals are called *Toeplitz matrices*. They have been studied extensively, with a special emphasis on studying the eigenvalues and their behaviour as the dimension tends to infinity.

We can construct this matrix as `D2` here:

In [1]:

```
# discrete first derivative
function D(h,n)
    ret=zeros(n,n+1)
    for k=1:n
        ret[k,k]=-1/h
        ret[k,k+1]=1/h
    end
    ret
end

# discrete second derivative
D2(h,n) = D(h,n-1)*D(h,n)
```

Out[1]:

```
D2 (generic function with 1 method)
```

We verify that `D2*f(x)` gives an approximation of the second derivative at the interior nodes:

In [2]:

```
n=10
h=1/n

f=x->cos(x)
fpp=x->-cos(x)   # second derivative of f

x=linspace(0.,1.,n+1) # domain nodes
r=x[2:end-1]          # range nodes


norm(D2(h,n)*f(x)  - fpp(r),Inf)
```

Out[2]:

```
0.0008288937970136745
```

Exercise Estimate the rate of convergence by finding α so that the error decays like Cn^{α}.

Multiplication operator

We now set up the multiplication operator correspo representing multiplication by $a(x)$. This is the $n-1 \times n+1$ matrix

$$A_n : \begin{array}{c}\text{Values at}\\ x_0, \ldots, x_n\end{array} \rightarrow \begin{array}{c}\text{Values at}\\ x_1, \ldots, x_{n-1}\end{array}$$

with entries given by

$$A_n \triangleq \begin{pmatrix} 0 & a(x_1) & & & \\ & & a(x_2) & & \\ & & & \ddots & \\ & & & & a(x_{n-1}) & 0 \end{pmatrix}.$$

We can set this up as follows:

In [3]:

```
function A(a::Function,h,n)
    ret=zeros(n-1,n+1)
    for k=1:n-1
        ret[k,k+1]=a(k*h)
    end
    ret
end
```

Out[3]:

```
A (generic function with 1 method)
```

In this case, the operator is in fact, exact:

In [4]:

```
a=x->sin(x)

A(a,h,n)
norm(A(a,h,n)*f(x)  - a(r).*f(r),Inf)
```

Out[4]:

```
1.1102230246251565e-16
```

So the operator $L = D^2 - a(x)$ is discretized as

$$L_n = D_n^2 - A_n$$

which is a map

$$L_n : \begin{matrix}\text{Values at} \\ x_0, \ldots, x_n\end{matrix} \rightarrow \begin{matrix}\text{Values at} \\ x_1, \ldots, x_{n-1}\end{matrix}$$

In [6]:

```
L=D2(h,n)  - A(a,h,n)
```

Out[6]:

```
9x11 Array{Float64,2}:
 100.0  -200.1    100.0       0.0     …     0.0        0.0        0.0
      0.0
   0.0   100.0  -200.199   100.0           0.0        0.0        0.0
      0.0
   0.0     0.0   100.0    -200.296         0.0        0.0        0.0
      0.0
   0.0     0.0     0.0     100.0           0.0        0.0        0.0
      0.0
   0.0     0.0     0.0       0.0           0.0        0.0        0.0
      0.0
   0.0     0.0     0.0       0.0      …   100.0       0.0        0.0
      0.0
   0.0     0.0     0.0       0.0        -200.644   100.0        0.0
      0.0
   0.0     0.0     0.0       0.0         100.0    -200.717   100.0
      0.0
   0.0     0.0     0.0       0.0           0.0      100.0    -200.7
83  100.0
```

Boundary conditions

We need to represent $u(0)$ and $u(1)$ where u is given at the grid $x_0, \ldots, x_n$. We see that this is accomplished via the $1 \times n + 1$ row vectors

$$B_n^0 \triangleq [1, 0, \cdots, 0]$$

In [7]:

```
B0 = [1 zeros(1,n)]

B0*f(x)  - f(0)
```

Out[7]:

```
1-element Array{Float64,1}:
 0.0
```

and

$$B_n^1 \triangleq [0, 0, \cdots, 1]$$

In [8]:

```
B1 = [zeros(1,n) 1]
B1*f(x) - f(1)
```

Out[8]:

```
1-element Array{Float64,1}:
 0.0
```

Constructing the discrete boundary value problem

We now discretize the *operator*

$$Mu = \begin{pmatrix} u(0) \\ u'' - a(x)u \\ u(1) \end{pmatrix}.$$

by

$$M_n = \begin{pmatrix} B_n^0 \\ D_n^2 - A_n \\ B_n^1 \end{pmatrix}$$

We put the boundary conditions at the top and bottom so that `M_n` is tridiagonal (that is, only has three non-zero bands).

In [9]:

```
L=D2(h,n)  - A(a,h,n)
M=[B0;
    L;
    B1]
```

Out[9]:

```
11x11 Array{Float64,2}:
   1.0      0.0      0.0        0.0     …      0.0        0.0        0.0
      0.0
 100.0   -200.1    100.0        0.0            0.0        0.0        0.0
      0.0
   0.0    100.0  -200.199    100.0             0.0        0.0        0.0
      0.0
   0.0      0.0    100.0    -200.296           0.0        0.0        0.0
      0.0
   0.0      0.0      0.0     100.0             0.0        0.0        0.0
      0.0
   0.0      0.0      0.0        0.0     …      0.0        0.0        0.0
      0.0
   0.0      0.0      0.0        0.0          100.0        0.0        0.0
      0.0
   0.0      0.0      0.0        0.0         -200.644   100.0         0.0
      0.0
   0.0      0.0      0.0        0.0          100.0    -200.717   100.0
      0.0
   0.0      0.0      0.0        0.0            0.0     100.0    -200.7

83  100.0
   0.0      0.0      0.0        0.0     …      0.0        0.0        0.0
```

We test that it approximates M:

In [10]:

```
norm(M*f(x)  - [f(0.); fpp(r)-a(r).*f(r); f(1.)],Inf)
```

Out[10]:

```
0.0008288937970233334
```

Exercise Estimate the rate of convergence.

We now approximate the boundary value problem

$$Mu = \begin{pmatrix} a \\ f(x) \\ b \end{pmatrix}$$

by solving the discretized problem

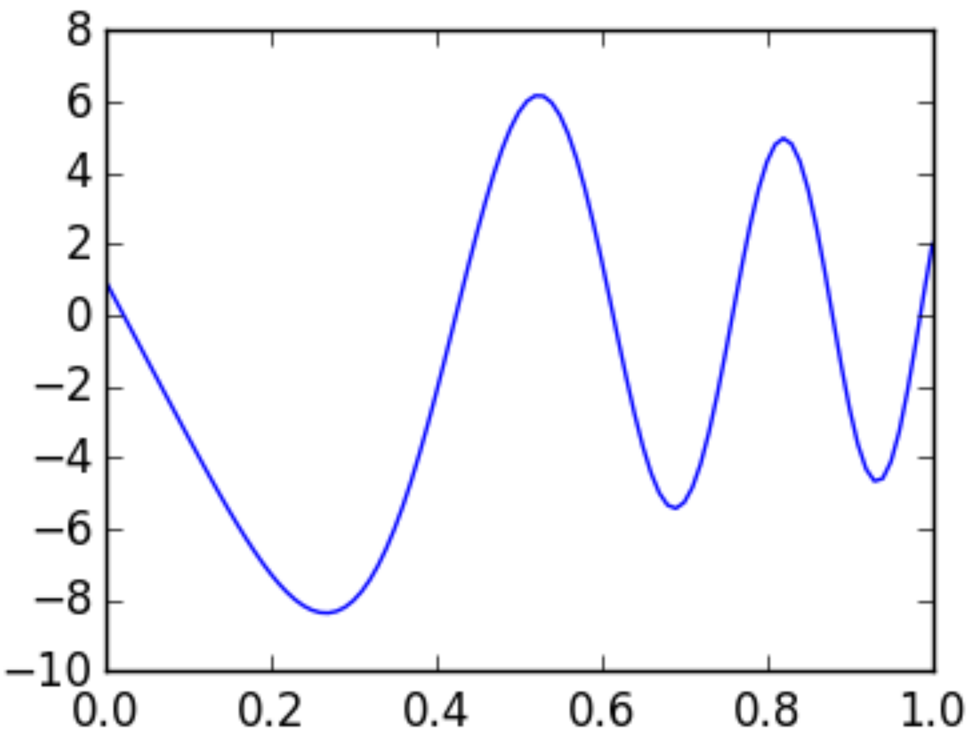
$$M_n \mathbf{w} = \begin{pmatrix} a \\ f(\mathbf{x}[2 : end - 1]) \\ b \end{pmatrix}$$

to find $w_k = \mathbf{e}_k^\top \mathbf{w}$ that approximates $u(x_k)$.

Here we solve the same equation as at the top:

In [13]:

```
using PyPlot

n=100
h=1/n

a=x->-1000x^2



x=linspace(0.,1.,n+1)

B0 = [1 zeros(1,n)]
B1 = [zeros(1,n) 1]

L=D2(h,n)  - A(a,h,n)
M=[B0;
    L;
    B1]

w=M\[1.;zeros(n-1);2.]

plot(x,w);
```



We observe empirically that the method converges:

In [19]:

```
n=4000
h=1/n

a=x->-1000x^2



x=linspace(0.,1.,n+1)

B0 = [1 zeros(1,n)]
B1 = [zeros(1,n) 1]

L=D2(h,n)  - A(a,h,n)
M=[B0;
    L;
    B1]

w=M\[1.;zeros(n-1);2.]

norm(w-u(x),Inf)  # the exact solution u was calulated above
```

Out[19]:

```
0.0003873070855124894
```

Exercise What property of M_n guarantees that we converge to the solution u as fast as

$$\|M_n f(\mathbf{x}) - (Mf)(\mathbf{x}[2 : end - 1])\|_\infty$$

converges?