CBE 502 - HW7 - Finite Elements

Tom Bertalan

December 26, 2012

Contents

List of Figures

1 Analytical Solution by Green's Function

1.1 Problem

$$L[u(x)] = -f(x)$$
$$L[u(x)] = \frac{\partial^2 u}{\partial x^2} \tag{1}$$
$$-f(x) = A\sin(\omega x) + mx$$

Boundary conditions:

$$u(x)\big|_{x=0} = 1$$
$$\frac{\partial u(x)}{\partial x}\bigg|_{x=1} = \epsilon \tag{2}$$

Constants:

$$A = 18.0$$
$$\omega = 10.0$$
$$m = 4.0 \tag{3}$$
$$\epsilon = 0.5$$

1

Use a change of variables to create homogeneous boundary conditions.

$$v(x) = u(x) + ax + b$$
$$v'(x) = u'(x) + a$$

(4)

To make the new boundary conditions homogeneous (Dirichlet at $x = 0$ and Neumann at $x = 1$), choose $b = -1$ and $a = -\epsilon$.

$$-f(x) = L[u] = \frac{\partial^2 v}{\partial x^2} + 0 + 0 = L[v]$$

That is, the problem does not change with the change-of-variables, only the boundary conditions.

1.2   Solution by Properties of the Green's Function

$G(x,t)$ satisfies the homogeneous problem (that is, $\partial^2 G(x,t)/\partial x^2 = 0$):

$$G(x,t) = \begin{cases} C_{1,1}(t)x + C_{1,2}(t), & 0 \le x \le t \\ C_{2,1}(t)x + C_{2,2}(t), & t \le x \le 1 \end{cases} = \begin{cases} C_{2,1}(t)x + C_{2,2}(t), & 0 \le t \le x \\ C_{1,1}(t)x + C_{1,2}(t), & x \le t \le 1 \end{cases}$$

(5)

$G(x,t)$ satisfies homogeneous boundary conditions:

$$G(0,t) = 0 = C_{1,1} \cdot 0 + C_{1,2} \to C_{1,2} = 0$$
$$G'(0,t) = 0 = C_{2,1} \qquad \to C_{2,1} = 0$$

(6)

$G(x,t)$ is piecewise, but fully continuous:

$$\lim_{x \to t^-} G(x,t) = \lim_{x \to t^+} G(x,t)$$
$$C_{1,1}(t) \cdot t = C_{2,2}(t)$$

(7)

$G'(x,t)$ has a jump discontinuity of $1/p(x)$, where $p(x) = 1$ is taken from the standard from of the second-order operator $L[v(x)]$:

$$\frac{\partial G}{\partial x}\bigg|_{t+} - \frac{\partial G}{\partial x}\bigg|_{t-} = 1$$
$$0 - C_{1,1}(t) = 1$$

(8)

From (7) and (8), we find that $C_{1,1} = -1$ and $C_{2,2}(t) = -t$. So, the completed Green's function for this operator L is:

$$G(x,t) = \begin{cases} -x & , & 0 \le x \le t \\ -t & , & t \le x \le 1 \end{cases} = \begin{cases} -t & , & 0 \le t \le x \\ -x & , & x \le t \le 1 \end{cases}$$

(9)

The solution (with the change-of-variables) is then given by integrating the product of the Green's function and the forcing function:

$$v(x) = \int_{x=a}^{x=b} -f(t) \quad G(x,t) \quad dt$$
$$= \int_0^x -f(t)(-t)dt + \int_x^1 -f(t)(-x)dt$$
$$= \frac{m\omega^2 x(-3 + x^2) + 6A\omega x \cos(\omega) - 6A\sin(\omega x)}{6\omega^2}$$

(10)

Check:

$$v(x)|_{x=0} = 0 \quad \checkmark$$
$$\frac{\partial v(x)}{\partial x}\bigg|_{x=1} = 0 \quad \checkmark$$
$$\frac{\partial^2 v(x)}{\partial x^2} - (-f(x)) = 0 \quad \checkmark$$

(11)

2

The true solution can be obtained by inverting the change-of-variables:

$$
\begin{aligned}
u(x) &= v(x) + \epsilon x + 1 \\
&= 1 + \epsilon x - \frac{mx}{2} + \frac{mx^3}{6} + \frac{Ax\cos(\omega)}{\omega} - \frac{A\sin(\omega x)}{\omega^2}
\end{aligned}
\tag{12}
$$

Check,

$$
\begin{aligned}
u(x)|_{x=0} &= 1 \quad \checkmark \\
\left.\frac{\partial u(x)}{\partial x}\right|_{x=1} &= \epsilon \quad \checkmark \\
\frac{\partial^2 u(x)}{\partial x^2} - (-f(x)) &= 0 \quad \checkmark
\end{aligned}
\tag{13}
$$

Algebra is verified in Section 5.

2 Finite Element Solution

Galerkin Form

$$
L[\tilde{u}(x)] - f(x) = error(x) \approx \vec{0}(x)
\tag{14}
$$

$$
\tilde{u}(x) = \sum_{i=1}^{N} u_i \phi^i(x)
\tag{15}
$$

$$
\int_a^b \left[L[\tilde{u}(x)] - f(x) \right] \phi^i(x) dx = 0
\tag{16}
$$

Split this into two separate integrals, then in the left integral, let $\mu = \phi^i(x)$ and $d\eta = \frac{\partial^2 \tilde{u}}{\partial x^2} dx$. Integrate by parts ($\int \mu d\eta = \mu\eta - \int \eta d\mu$). This reduces the derivatives in the operator from second to first order, which enables the use of linear basis functions.

$$
\begin{aligned}
\int_a^b L[\tilde{u}(x)]\phi^i(x)dx - \int_a^b f(x)\phi^i(x)dx &= 0 \\
\int_0^1 \frac{\partial^2 \tilde{u}}{\partial x^2}\phi^i dx - \int_0^1 f(x)\phi^i(x)dx &= 0
\end{aligned}
\tag{17}
$$

$$
\begin{aligned}
\phi^i(x)\left.\frac{\partial \tilde{u}}{\partial x}\right|_0^1 - \int_0^1 \left[\frac{\partial \tilde{u}}{\partial x}\frac{\partial \phi^i}{\partial x} + f(x)\phi^i(x) \right] & \\
\phi^i(x)\left.\frac{\partial \tilde{u}}{\partial x}\right|_0^1 - \int_0^1 \frac{\partial \tilde{u}}{\partial x} &= \\
\frac{\partial \phi^i}{\partial x}dx &= \int_0^1 f(x)\phi^i(x)
\end{aligned}
\tag{18}
$$

Now use (15) to express $\partial \tilde{u}/\partial x$ in terms of basis functions. Bring the derivative inside the sum:

$$
\phi^i(x)\sum u_j \frac{\partial \phi^i}{\partial x} - \int_0^1 \frac{\partial \phi^i}{\partial x}\left(\sum u_j \frac{\partial \phi^j}{\partial x} \right) dx = \int_0^1 f(x)\phi^i(x)dx
\tag{19}
$$

Since all other basis functions are zero at the right side of the domain, the term on the left applies only to the last basis function. This allows the use of a Dirac delta function. Combined with the right (Neumann) boundary condition, this changes the term to $\epsilon \delta_{iN}$.

The derivative of ϕ^i can be brought inside the adjacent sum. Its product with the derivative of ϕ^j is zero for values of j but $j = i-1$, $j = i$, and $j = i+1$, by the near-orthogonality of the basis functions. With our linear basis functions, the right-hand-side intergrals must be calculated in piecewise fashion, with the first piece being from the left edge of each hat function to its point, and the second piece being from the point to the right edge.

This simplifies the equation somewhat.

$$\epsilon \delta_{iN} - \int_0^1 \frac{\partial \phi^i}{\partial x} \left(\sum_{j=i-1}^{i+1} u_j \frac{\partial \phi^j}{\partial x} \right) dx = \int_0^1 f(x) \phi^i(x) dx$$

$$\sum_{j=1}^{N} \left(\int_0^1 \frac{\partial \phi^i}{\partial x} \frac{\partial \phi^j}{\partial x} dx \right) u_j = \epsilon \delta_{iN} - \int_0^1 f(x) \phi^i(x) dx \tag{20}$$

$$\underline{\underline{K}} \cdot \underline{u} = \underline{F}$$

Here, $\underline{\underline{K}}$ is a tridiagonal square matrix. To satisfy the left boundary condition, the first row of $\underline{\underline{K}}$ is $1, 0, 0, ...$, and the first element of $\underline{F}$ is 1.

With linear basis functions, most of the main diagonal of $\underline{\underline{K}}$ (besides the first element) is repetitions of $2/h$. Here, $h = (1-0)/(N-1)$ is half the width of one of the N basis functions. However, the last value in the main diagonal is only $1/h$, because the product $\frac{\partial \phi^N}{\partial x} \frac{\partial \phi^N}{\partial x}$ only involves the left slope of the hat for the final basis function. Besides the second element in the first row, which is zero to satisfy the left boundary condition, the two off-diagonals are repetitions of $-1/h$, from the product of the (opposite-sign) slopes of the overlapping portions of adjacent basis functions.

In Section 3, we solve this system with the python library *Numpy* ("numerical python") for various numbers of basis functions, N.

3 Results
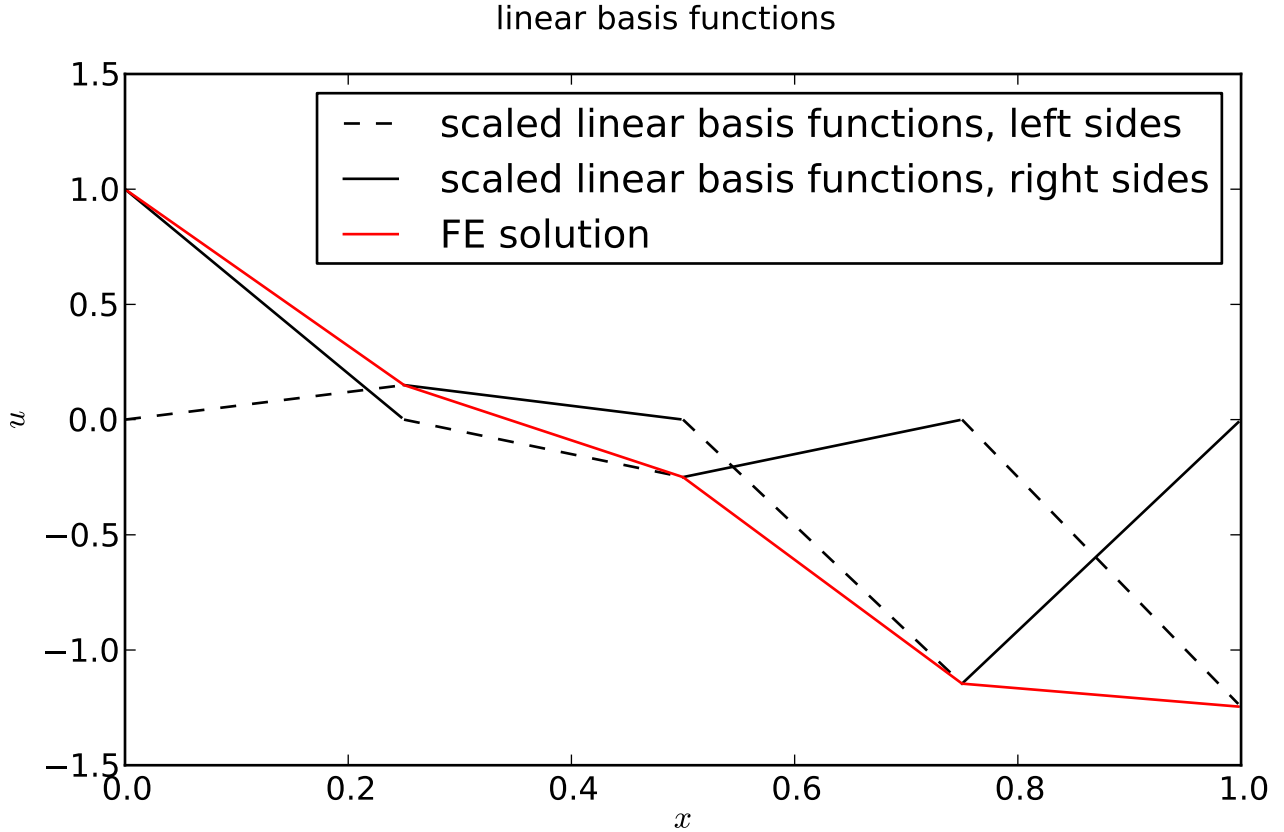
Figure 1: Five basis functions. Unscaled basis functions are hats of width $2h$ and unit height. The FE (finite element) solution is the superposition of the scaled basis functions shown here: $u(x) = \sum_{j=1}^{N} \phi^j(x)$

linear basis functions

Figure 2: Twenty basis functions.

linear basis functions

Figure 3: One hundred basis functions.
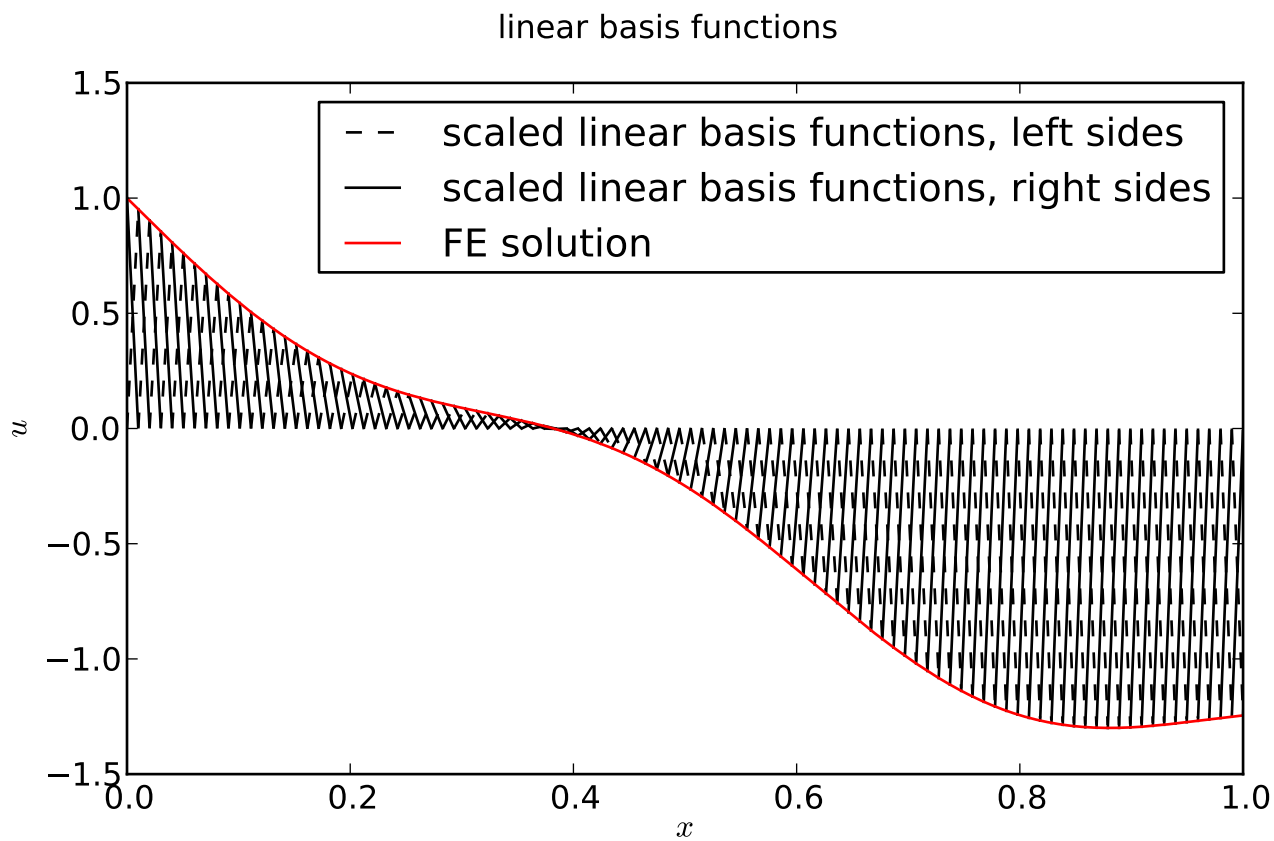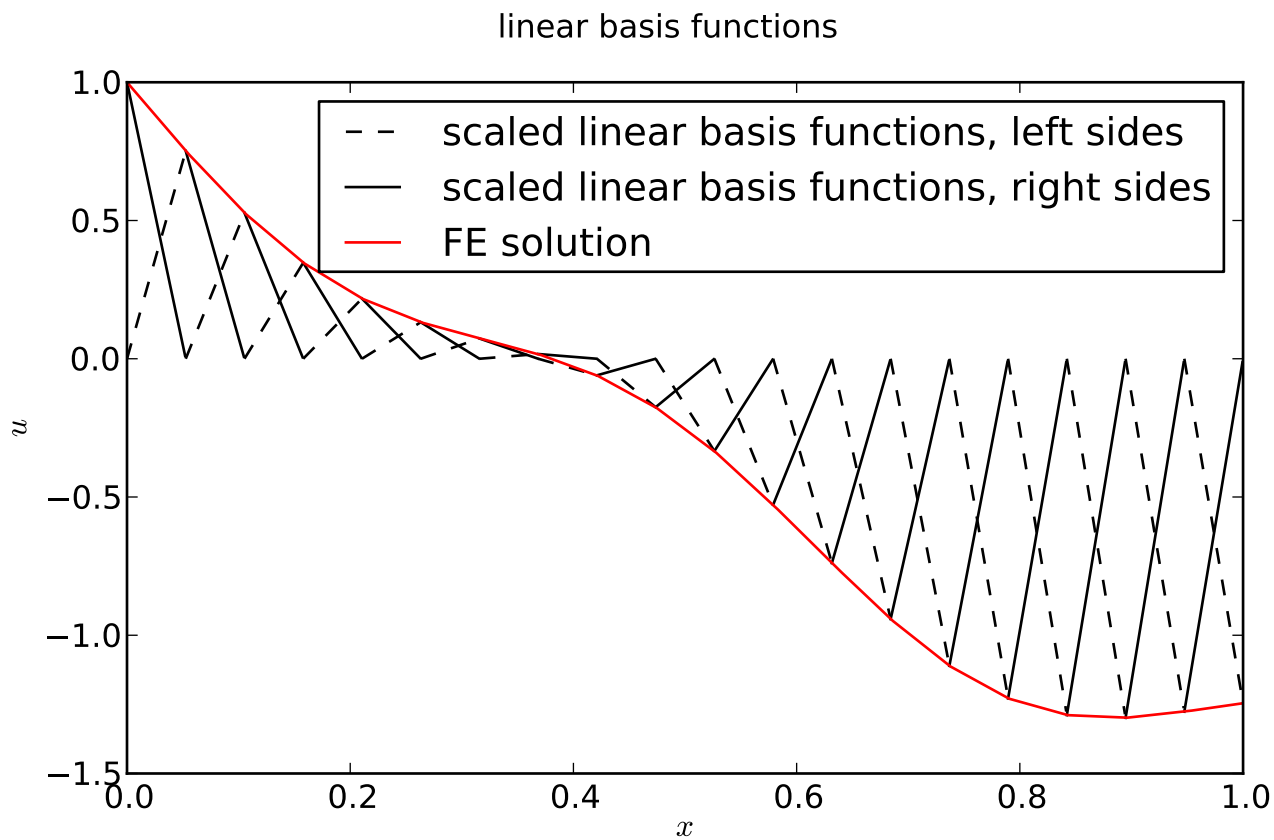
Figure 4: Solution and forcing, 5 basis functions.

Figure 5: Solution and forcing, 20 basis functions.
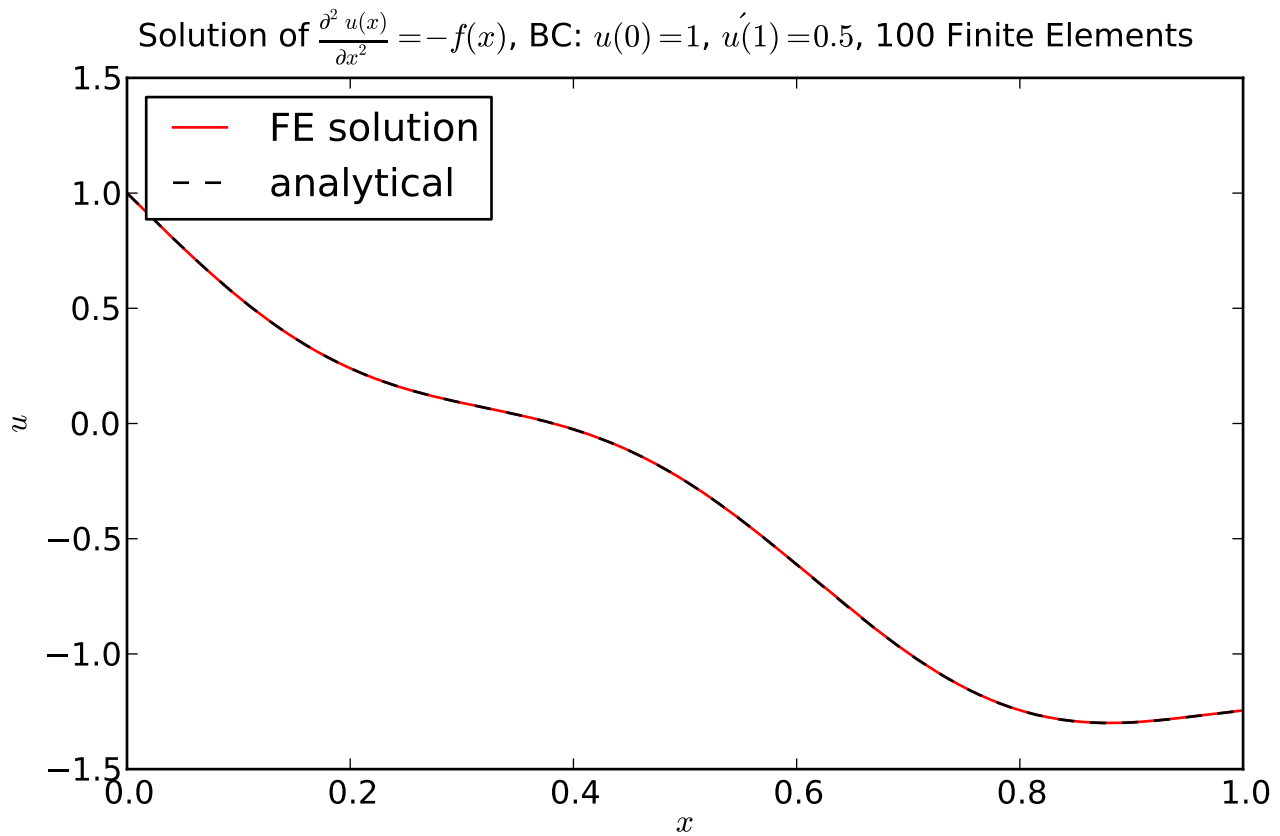
Figure 6: Solution and forcing, 100 basis functions.
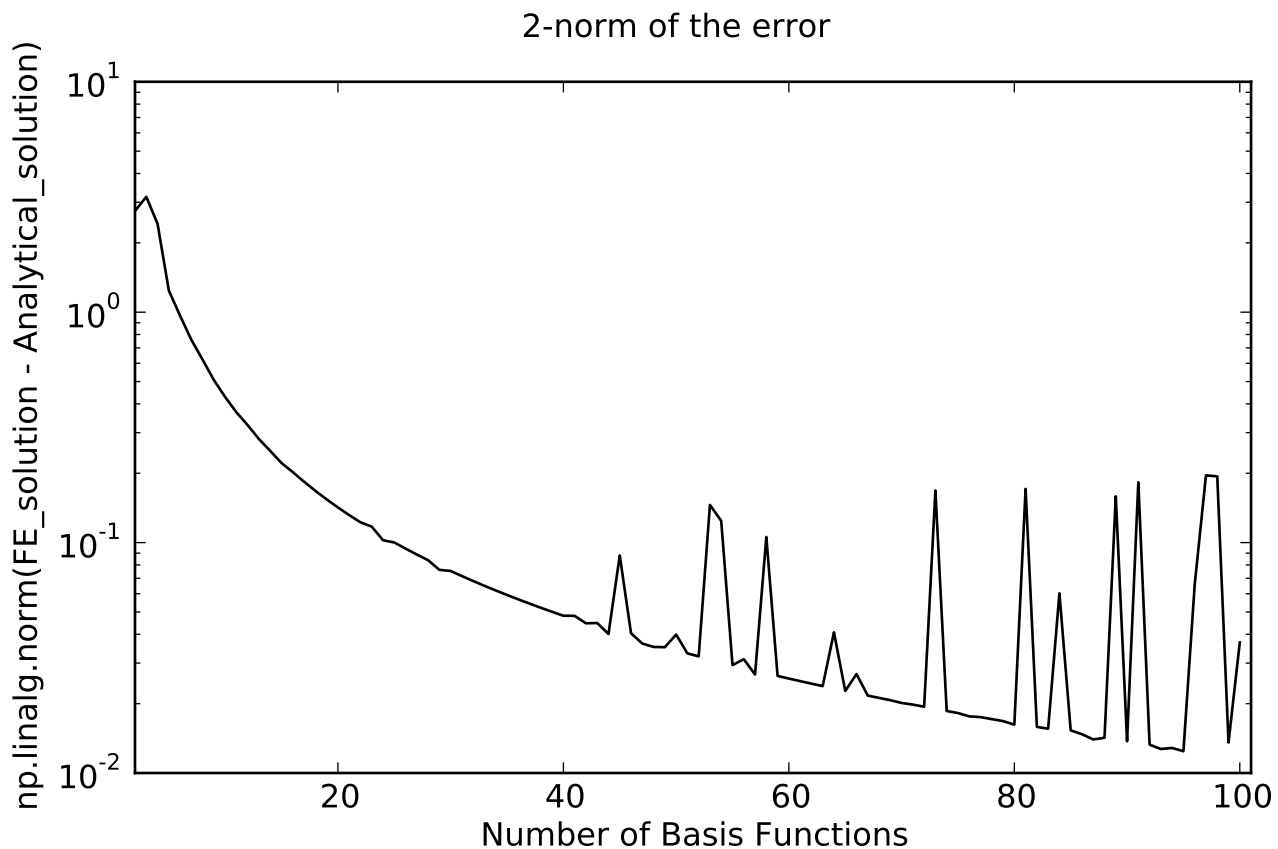
Figure 7: Rate of error reduction with increasing number of basis functions (decreasing discretization distance).

4 Code

hw7.py

```python
import numpy as np
import matplotlib.pyplot as plt
width = 7.5
figsize = (width, width * 6.0 / 10.0)


def do_hw(N, save=False):

    h = (1.0 - 0) / (N - 1)

    def  phi_l(i):
        a = (i - 1) * h
        return lambda x: (x - a) / h

    def phi_r(i):
        a = (i - 1) * h
        b = a + h
        c = b + h
        return lambda x: -(x - c) / h

    cos = np.cos
    sin = np.sin

    if save:
        fig = plt.figure(figsize=figsize)
        ax = fig.add_subplot(1, 1, 1)

        fig2 = plt.figure(figsize=figsize)
        ax2 = fig2.add_subplot(1, 1, 1)
        ax2.set_xlim((0,1))
        #ax2.set_ylim((0,1))
        fig2.suptitle('linear basis functions')
        ax2.set_xlabel(r'$x$')
        ax2.set_ylabel(r'$u$')

    epsilon = 0.5
    omega = 10.0
    m = 4.0
    amp = 18.0
    forcing = lambda x: -amp * sin(x * omega) - m * x
    #forcing = lambda x: 0

    analytical = lambda x: 1 + epsilon*x - (m*x)/2 + (m*x**3)/6 + \
        (amp*x*cos(omega))/omega - amp*sin(omega*x)/omega**2

    def integrate(f, a, b, precision=1000):
        dx = (float(b) - float(a)) / float(precision)
        sum_ = 0
        for x in np.arange(a, b, dx):
            sum_ += f(x) * dx
        return sum_



    main = 2 / h * np.eye(N)
    tri = -1 / h * np.eye(N-1)
```
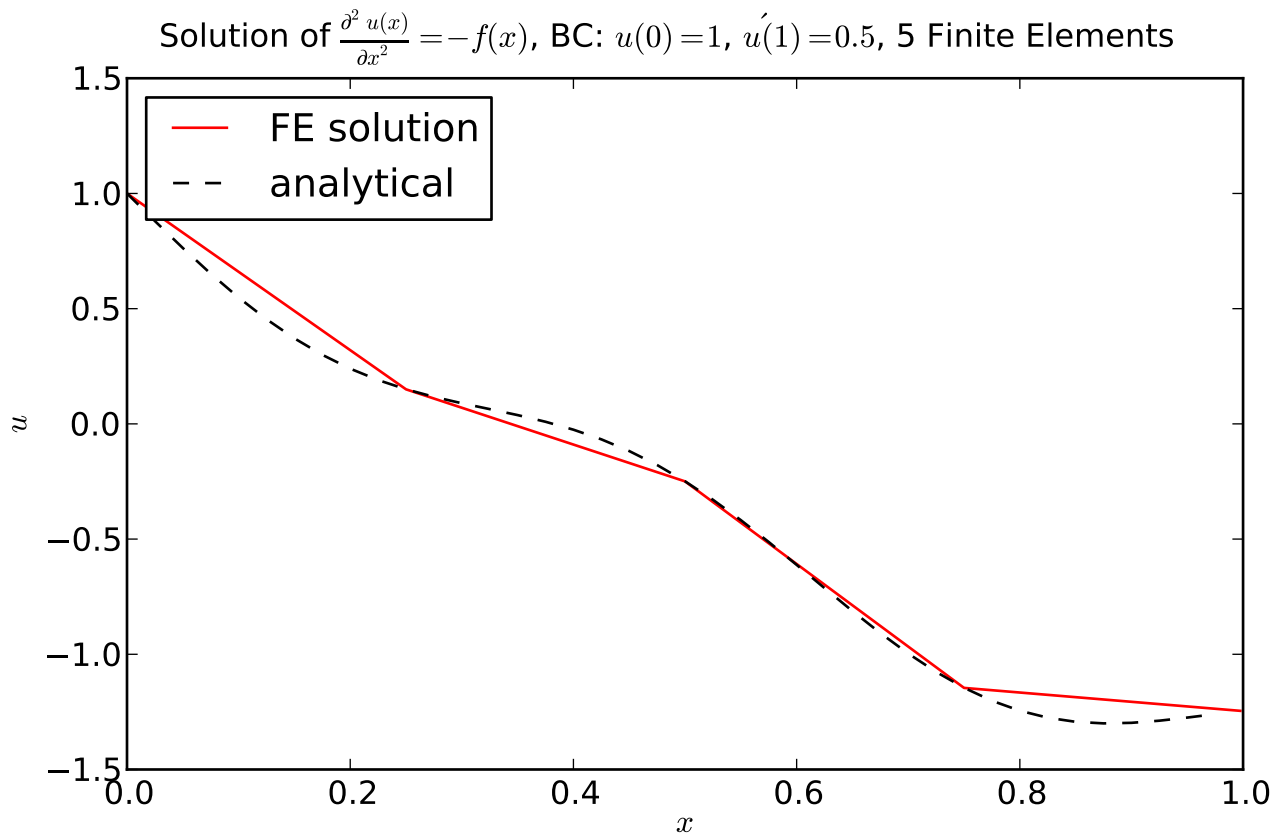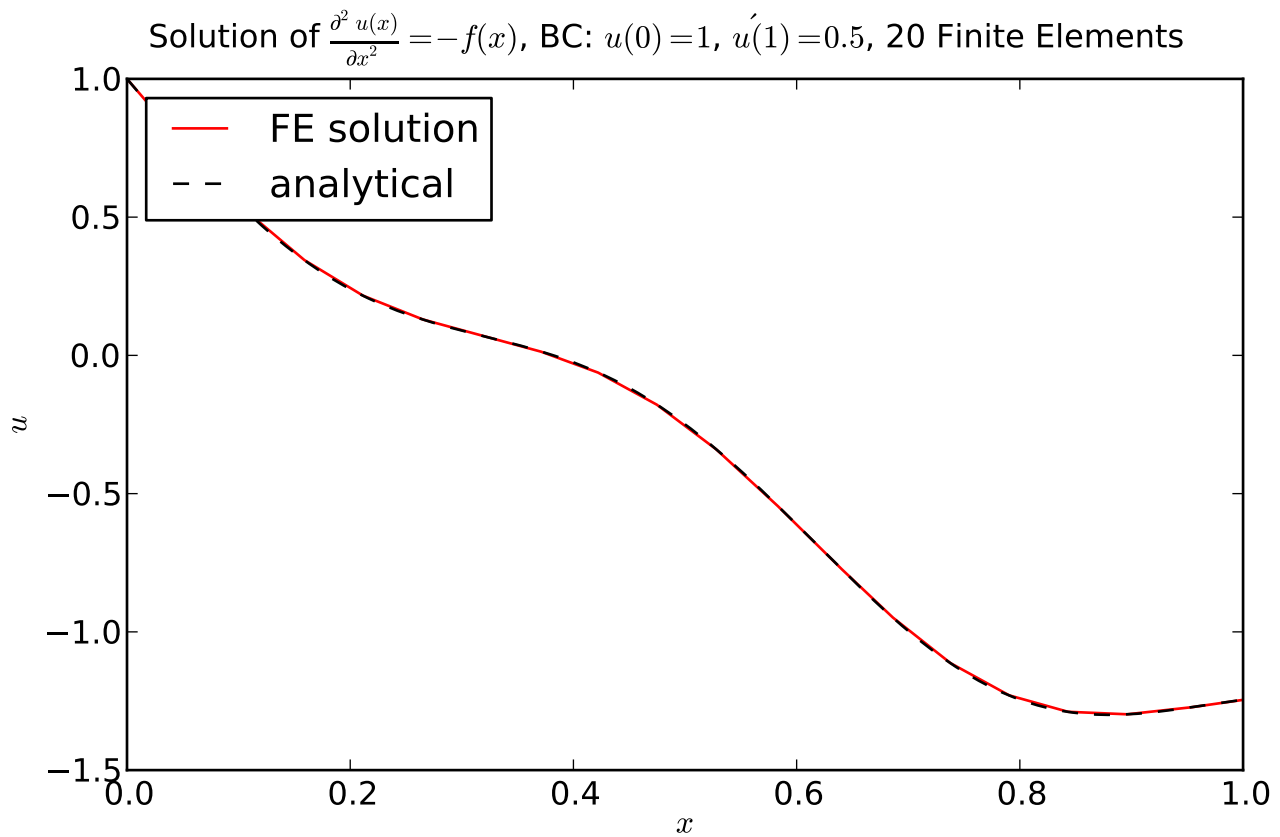
```python
        tril = np.vstack((np.zeros((1, N)), np.hstack((tri, np.zeros((N-1, 1)) ) ) ))
        triu = np.vstack((np.zeros((1, N)), np.hstack((tri, np.zeros((N-1, 1)) ) ) )).T
        A = main + tril + triu
        A[0, 0] = 1
        A[0, 1] = 0
        A[-1, -1] = A[-1, -1] * 0.5
#       print "A:",
#       print A

        rhs = []
        for i in range(N):
            a = (i - 1) * h
            b = a + h
            c = b + h
            xl_l = list(np.arange(a, b, (b - a) / 100))
            yl_l = [phi_l(i)(x) for x in xl_l]

            xl_r = list(np.arange(b, c, (c - b) / 100))
            yl_r = [phi_r(i)(x) for x in xl_r]

#            if i==N-1:
#                ax2.plot(xl_l, yl_l, 'k:', label='linear basis functions, left sides')
#                ax2.plot(xl_r, yl_r, 'k-.', label='linear basis functions, right sides')
#            else:
#                ax2.plot(xl_l, yl_l, 'k:')
#                ax2.plot(xl_r, yl_r, 'k-.')

            f_l = lambda x: forcing(x) * phi_l(i)(x)
            f_r = lambda x: forcing(x) * phi_r(i)(x)
            if i==N-1:
                rhs.append(integrate(f_l, a, b) + epsilon)
            else:
                rhs.append(integrate(f_l, a, b) + integrate(f_r, b, c))

        rhs = np.array(rhs).reshape((N, 1))
        rhs[0] = 1
#       print "rhs:",
#       print rhs
        u = np.linalg.solve(A, rhs)

        # plot scaled basis functions:
        for i in range(N):
            a = (i - 1) * h
            b = a + h
            c = b + h
            phi_i_l = lambda x: (x - a) / h * u[i]
            xl_l = list(np.arange(a, b, (b - a) / 100))
            yl_l = [phi_i_l(x) for x in xl_l]

            phi_i_r = lambda x: -(x - c) / h * u[i]
            xl_r = list(np.arange(b, c, (c - b) / 100))
            yl_r = [phi_i_r(x) for x in xl_r]

            if save:
                if i==N-1:
                    ax2.plot(xl_l, yl_l, 'k--', label='scaled linear basis functions, left sides')
                    ax2.plot(xl_r, yl_r, 'k-', label='scaled linear basis functions, right sides')
                else:
                    ax2.plot(xl_l, yl_l, 'k--')
                    ax2.plot(xl_r, yl_r, 'k-')
```

```
        #v = -u + 2
#    print "u:",
#    print u
#    print "resid:",
#    print np.dot(A, u) - rhs

    xl = list(np.arange(0, 1, h))
    xl_fine = list(np.arange(0, 1, h / 10.0))
    if len(xl) < len(u):
        xl.append(1.0)

    FE_domains = []
    FE_soln = []
    for i in range(1, N):
        # in each node, two basis functions apply.
        phi_lower = phi_r(i-1)
        phi_upper = phi_l(i)
        node_soln_function = lambda x: u[i-1] * phi_lower(x) + u[i] * phi_upper(x)
        # The node streches from a to b
        a = (i - 1) * h
        b = a + h
        node_domain = list(np.arange(a, b, (b - a) / 100))
        node_soln = [node_soln_function(x) for x in node_domain]
        # Add these to the growing solution array
        FE_soln.extend(node_soln)
        FE_domains.extend(node_domain)
    if save:
        ax2.plot(FE_domains, FE_soln, 'r-', label='FE solution')
        ax.plot(FE_domains, FE_soln, 'r-', label='FE solution')

#      ax2.plot(xl_fine, [analytical(x) for x in xl_fine], 'r-', label='analytical solution')
#      ax.scatter(xl, u, label=r'$N=%i$ finite element coeffecients' % N, color='k', marker='d')
#        ax.plot(xl_fine, [-forcing(x) for x in xl_fine], 'k--', label=r'forcing function $-f(x)=%.2f s

        ax.plot(xl_fine, [analytical(x) for x in xl_fine], 'k--', label='analytical')
        fig.suptitle(r'Solution of $\frac{\partial^2 u(x) }{ \partial x^2 }= -f(x)$, BC: $u(0)=1$, $u\'(1)=
        ax.legend(loc='upper left')
        ax2.legend(loc='upper right')

        ax.set_xlim((0, 1))
        ax.set_xlabel(r'$x$')
        ax.set_ylabel(r'$u$')


        #fig.tight_layout()
        fig.savefig('hw7-solution_and_forcing-N%i.pdf' % N)
        fig2.savefig('hw7-basis_functions-N%i.pdf' % N)
        #plt.show()

    error = []
    for (FE, a) in zip(FE_soln, [analytical(x) for x in FE_domains]):
        error.append(abs(FE - a))
    return error

if __name__=="__main__":
    fig3 = plt.figure(figsize=figsize)
    ax3 = fig3.add_subplot(1, 1, 1)
    to_save = [5, 20, 100]
```

```python
    N_list = []
    norm_list = []
    Nmin = 2
    Nmax = 101
    for N in range(Nmin, Nmax):
        print 'N is', N,
        if N in to_save:
            error = do_hw(N, save=True)
        else:
            error = do_hw(N, save=False)
        print "; norm is ", np.linalg.norm(np.array(error))
        N_list.append(N)
        norm_list.append(np.linalg.norm(error))
#        ax3.plot(error, 'k')
    ax3.set_yscale('log')
    fig3.suptitle('2-norm of the error')
    ax3.set_xlim((Nmin, Nmax))
    ax3.set_xlabel('Number of Basis Functions')
    ax3.set_ylabel('np.linalg.norm(FE_solution - Analytical_solution)')
    ax3.plot(N_list, norm_list, 'k')
    fig3.savefig('hw7-error_rate.pdf')
```

5 Algebra by Mathematica

5.1 Generalized in A, ω, m, ϵ

In[1]:= `(*omega := 10.0;*)`
`(*m := 4.0;*)`
`(*epsilon := 0.5;*)`
`(*amp = 18.0*)`
`f[x_] := -amp * Sin[omega * x] - m * x`
`oldf[x_] := -4 * x`

In[3]:= `v[x_] :=  Integrate[ -f[t]*(-t), {t, 0, x} ] + Integrate[ -f[t]*(-x), {t, x, 1} ];`
`Simplify[v[x]]`

Out[4]= $\dfrac{m\ \text{omega}^2\ x\ \left(-3 + x^2\right) + 6\ \text{amp}\ \text{omega}\ x\ \text{Cos}[\text{omega}] - 6\ \text{amp}\ \text{Sin}[\text{omega}\ x]}{6\ \text{omega}^2}$

In[5]:= `u[x_] := v[x] + epsilon * x + 1`

In[6]:= `Simplify[u[x]]`

Out[6]= $1 + \text{epsilon}\ x - \dfrac{m\ x}{2} + \dfrac{m\ x^3}{6} + \dfrac{\text{amp}\ x\ \text{Cos}[\text{omega}]}{\text{omega}} - \dfrac{\text{amp}\ \text{Sin}[\text{omega}\ x]}{\text{omega}^2}$

In[7]:= `Simplify[D[u[x], x]]`

Out[7]= $\text{epsilon} - \dfrac{m}{2} + \dfrac{m\ x^2}{2} + \dfrac{\text{amp}\ \text{Cos}[\text{omega}]}{\text{omega}} - \dfrac{\text{amp}\ \text{Cos}[\text{omega}\ x]}{\text{omega}}$

In[8]:= `v[0] // N`
`D[v[x], x] /. x -> 1 // N`
`Simplify[D[D[v[x], x], x] + f[x]]`

Out[8]= `0.`

Out[9]= $-1.\ m - 1.\ \text{amp}\ \text{Sin}[\text{omega}] + \dfrac{0.5\ (2.\ m\ \text{omega} + 2.\ \text{amp}\ \text{omega}\ \text{Sin}[\text{omega}])}{\text{omega}}$

Out[10]= `0`

In[11]:= `u[0]`
`D[u[x], x] /. x -> 1`
`Simplify[D[D[u[x], x], x] + f[x]]`

Out[11]= `1`

Out[12]= $\text{epsilon} - m - \text{amp}\ \text{Sin}[\text{omega}] + \dfrac{2\ m\ \text{omega} + 2\ \text{amp}\ \text{omega}\ \text{Sin}[\text{omega}]}{2\ \text{omega}}$

Out[13]= `0`

5.2 Particularized, to Show Satisfied BCs and Nonhomogeneous Equation

```
In[14]:= omega := 10.0;
        m := 4.0;
        epsilon := 0.5;
        amp = 18.0
        f[x_] := -amp * Sin[omega * x] - m * x
        oldf[x_] := -4 * x
```

Out[17]= 18.

```
In[20]:=  v[x_] :=   Integrate[ -f[t] * (-t), {t, 0, x} ] + Integrate[ -f[t] * (-x), {t, x, 1} ];
        Simplify[v[x]]
```

Out[21]= $0. - 3.51033 \, x + 0.666667 \, x^3 - 0.18 \, \text{Sin}[10. \, x]$

```
In[22]:= u[x_] := v[x] + epsilon * x + 1
```

```
In[23]:= Simplify[u[x]]
```

Out[23]= $1. - 3.01033 \, x + 0.666667 \, x^3 - 0.18 \, \text{Sin}[10. \, x]$

```
In[24]:= Simplify[D[u[x], x]]
```

Out[24]= $-3.01033 + 2. \, x^2 - 1.8 \, \text{Cos}[10. \, x]$

```
In[25]:= v[0] // N
        D[v[x], x] /. x → 1 // N
        Simplify[D[D[v[x], x], x] + f[x]]
```

Out[25]= 0.

Out[26]= -8.88178×10^{-16}

Out[27]= $0. - 3.55271 \times 10^{-15} \, \text{Sin}[10. \, x]$

```
In[28]:= u[0]
        D[u[x], x] /. x → 1
        Simplify[D[D[u[x], x], x] + f[x]]
```

Out[28]= 1.

Out[29]= 0.5

Out[30]= $0. - 3.55271 \times 10^{-15} \, \text{Sin}[10. \, x]$

13