

# Solution of a boundary-value problem through orthogonal projection

We will use orthogonal projection to solve the equation

$$R[u] = \frac{d}{dx} \left[ e^{2x} \frac{du}{dx} \right] - e^x = 0$$

with boundary conditions  $u(0) = 0$  and  $u(1) = 0$ . The solution will be a member of some inner product space  $V$  of functions meeting certain differentiability and integrability conditions (which we'll talk about later; for now, any "reasonable" function will work).

Recall there's a theorem stating that  $R[u] = 0$  iff  $\langle v, R[u] \rangle = 0 \forall v \in V$ , that is, the solution is the point  $u$  such that the residual  $R[u]$  is orthogonal to every member of  $V$ . The method of orthogonal projection (Galerkin's method) replaces this exact condition with a requirement that the residual be orthogonal to every member of a *finite-dimensional* subspace  $V^M$ , and assumes that the solution can be well-approximated by a member of that subspace.

We're searching for a function that obeys boundary conditions  $u(0) = u(1) = 0$ . Notice that the set of all such functions is a subspace of  $V$ . We'll let  $V_0$  be the subspace of BC-compatible functions. We then want a finite-dimensional subspace of  $V_0$ , which we'll call  $V_0^M$ .

Let  $\Phi = \{\phi\}$  be a basis for  $V_0^M$ . Each  $\phi$  must satisfy the BCs.

We assume  $u(x) = \sum_{n=1}^M u_n \phi_n(x)$ , and then require  $\langle \phi_n, R[\sum_{m=1}^M u_m \phi_m] \rangle = 0$  for  $n = 1, 2, \dots, M$ .

The structure of the resulting equation set will be clarified by writing  $R[u] = L[u] - f(x)$ . In our current problem, we have  $L[u] = \frac{d}{dx} \left[ e^{2x} \frac{du}{dx} \right]$  and  $f(x) = e^x$ . Then, the equation

$$\langle \phi_n, R[\sum_{m=1}^M u_m \phi_m] \rangle = 0$$

becomes

$$\langle \phi_n, L[\sum_{m=1}^M u_m \phi_m] \rangle = \langle \phi_n, f \rangle. \text{ When } L \text{ is a linear operator, we can distribute } L \text{ to find}$$

$$\sum_{m=1}^M u_m \langle \phi_n, L[\phi_m] \rangle = \langle \phi_n, f \rangle.$$

Writing  $A_{nm} = \langle \phi_n, L[\phi_m] \rangle$  and  $b_n = \langle \phi_n, f \rangle$  we get a matrix-vector equation  $Au = b$ .

## ■ Compute an exact solution for comparison to our approximate calculations

This problem can be solved exactly; most boundary-value problems can only be solved approximately.

```
exactSoln = DSolve[{D[Exp[2 x] D[u[x], x], x] == -Exp[x], u[0] == 0, u[1] == 0}, u[x], x]
```

$$\left\{ \left\{ u(x) \rightarrow -\frac{e^{-2x}(-1+e^x)(-e+e^x)}{1+e} \right\} \right\}$$

```
uExact[x_] = u[x] /. exactSoln[[1]]
```

$$-\frac{e^{-2x}(-1+e^x)(-e+e^x)}{1+e}$$

### Define a basis for the approximating subspace

We want a linearly independent set of simple functions that satisfy the BCs. Here's a simple choice:

```
phi[n_, x_] := Sin[Pi n x]
sin(n π x)
```

In problems on complicated multidimensional domains, it's usually not possible to find a basis using such simple functions. The **Finite Element Method** is an approach to building suitable bases out of piecewise-differentiable combinations of simple functions. To use a piecewise basis we'll need to look more carefully at the differentiability and integrability requirements for the space of functions. For now, we proceed with a 1D problem allowing a simple basis.

#### ■ Define an inner product

We'll use the unweighted inner product  $\int_0^1 f(x) g(x) dx$ .

```
innerProduct[f_, g_] := Integrate[f g, {x, 0, 1}]
```

#### ■ Define the LHS operator and RHS function

```
L[u_] := D[Exp[2 x] D[u, x], x]
f[x_] := Exp[x]
```

#### ■ Define functions which calculate the matrix and vector

```
A[M_] := Table[innerProduct[phi[n, x], op[phi[m, x]]],
  {n, 1, M}, {m, 1, M}]
b[M_] := Table[innerProduct[phi[n, x], -Exp[x]], {n, 1, M}]
```

#### ■ Show some typical examples

```
A[3] // N

$$\begin{pmatrix} -17.2147 & 13.7232 & -5.5193 \\ 13.7232 & -64.6153 & 37.7991 \\ -5.5193 & 37.7991 & -143.459 \end{pmatrix}$$

b[3] // N
{-1.07468, 0.266717, -0.39013}
```

#### ■ Define a function that will set up and solve the linear system of equations

This function creates matrix  $A$  and vector  $b$ , then solves the equation  $A u = b$ .

```
uCoeff[M_] := LinearSolve[A[M] // N, b[M] // N]
```

This function forms the approximate solution  $u(x) = \sum_{m=1}^M u_m \phi_m(x)$ .

```
uSoln[M_, x_] := Sum[uCoeff[M][[m]] phi[m, x], {m, 1, M}]
```

## Display a few solutions

```
u1[x_] = uSoln[1, x]
```

```
0.062428 sin( $\pi x$ )
```

```
u2[x_] = uSoln[2, x]
```

```
0.0711905 sin( $\pi x$ ) + 0.0109919 sin(2  $\pi x$ )
```

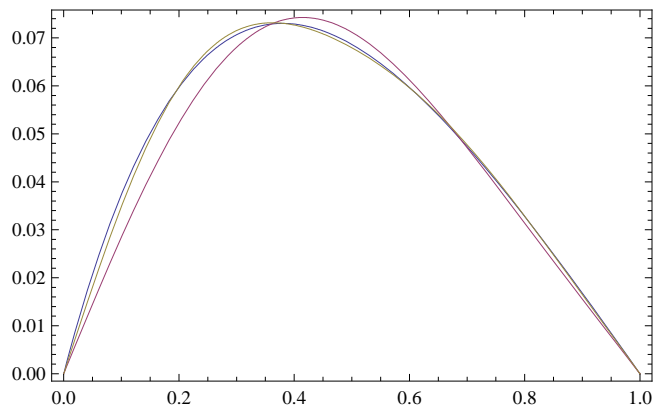
```
u4[x_] = uSoln[4, x]
```

```
0.0721277 sin( $\pi x$ ) + 0.0133536 sin(2  $\pi x$ ) + 0.00417192 sin(3  $\pi x$ ) + 0.00138055 sin(4  $\pi x$ )
```

```
u6[x_] = uSoln[6, x]
```

```
0.072229 sin( $\pi x$ ) + 0.0134936 sin(2  $\pi x$ ) + 0.00432239 sin(3  $\pi x$ ) +  
0.00180426 sin(4  $\pi x$ ) + 0.000923871 sin(5  $\pi x$ ) + 0.000396923 sin(6  $\pi x$ )
```

```
Plot[{uExact[x], u2[x], u4[x]}, {x, 0, 1}, Frame -> True]
```



```
Plot[{uExact[x] - u2[x], uExact[x] - u4[x], uExact[x] - u6[x]}, {x, 0, 1}, PlotRange -> All]
```

