Solving the Wave Equation by the Finite Element Method

Consider the wave equation with Dirichlet boundary conditions

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = f(x,t)$$

$$u(0,t) = u(l,t) = 0$$

$$u(x,0) = \psi(x)$$

$$\frac{\partial u}{\partial t}(x,0) = \gamma(x)$$

To apply the Galerkin method to this equation we start by multiplying both sides of the PDE by test functions $v(x)$ from $C_D^2[0,l]$ and integrating to make a weak form of the PDE.

$$\int_0^l \left(\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2}\right) v(x)\, \mathrm{d}x = \int_0^l f(x,t)\ v(x)\, \mathrm{d}x$$

As usual, we apply integration by parts one time to convert the form of the second term on the left:

$$\int_0^l \left(-c^2 \frac{\partial^2 u}{\partial x^2}\right) v(x)\, \mathrm{d}x = -c^2 \left(\frac{\partial u}{\partial x}\ v(x)\right)\Big|_0^l + c^2 \int_0^l \frac{\partial u}{\partial x} \frac{\partial v}{\partial x}\, \mathrm{d}x$$

Since $v(x)$ vanishes at the boundary we have

$$\int_0^l \frac{\partial^2 u}{\partial t^2}\ v(x) + c^2 \frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\, \mathrm{d}x = \int_0^l f(x,t)\ v(x)\, \mathrm{d}x$$

This is the weak form of the wave equation. As before, we will select N functions $\varphi_i(x)$ that form a basis for a subspace V_N of $C_D^2[0,l]$. This time around we have to assume that the approximate solution $u_N(x,t)$ takes the form

$$u_N(x,t) = \sum_{j=1}^{N} \alpha_j(t)\ \varphi_j(x)$$

Since the solution depends on both x and t we have to assume that the coefficients of this combination are functions of t.

Substituting this approximate solution with test function $v(x) = \varphi_i(x)$ into the weak form gives

$$\int_0^l \sum_{j=1}^{N} \alpha_j''(t)\ \varphi_j(x)\ \varphi_i(x) + c^2 \sum_{j=1}^{N} \alpha_j(t) \frac{\mathrm{d}\varphi_j(x)}{\mathrm{d}x}\frac{\mathrm{d}\varphi_i(x)}{\mathrm{d}x}\, \mathrm{d}x = \int_0^l f(x,t)\ \varphi_i(x)\, \mathrm{d}x$$

If we introduce *mass matrix* M whose i,j entry is

$$M_{i,j} = \int_0^l \varphi_j(x)\ \varphi_i(x)\, \mathrm{d}x$$

a stiffness matrix K whose i,j entry is

$$K_{i,j} = \int_0^l c^2 \frac{\mathrm{d}\,\varphi_j(x)}{\mathrm{d}\,x} \frac{\mathrm{d}\,\varphi_i(x)}{\mathrm{d}\,x}\,\mathrm{d}\,x$$

a vector $\mathbf{f}(t)$ whose j entry is

$$\mathbf{f}_j(t) = \int_0^l f(x,t)\ \varphi_j(x)\,\mathrm{d}\,x$$

and a vector $\alpha(t)$ whose j entry is $\alpha_j(t)$ we can write the equation above as a system of ODEs for the vector $\alpha(t)$ of unknown coefficients $\alpha_j(t)$:

$$M\,\alpha''(t) + K\,\alpha(t) = \mathbf{f}(t)$$

This system of equations is second order in t, so we will require two initial conditions. One initial condition is given by the requirement that

$$u_N(x,0) = \sum_{j=1}^{N} \alpha_j(0)\ \varphi_j(x) \approx \sum_{j=1}^{N} \psi(x_j)\ \varphi_j(x)$$

The second initial condition comes from

$$\frac{\partial u_N}{\partial t}(x,0) = \sum_{j=1}^{N} \alpha_j'(0)\ \varphi_j(x) \approx \sum_{j=1}^{N} \gamma(x_j)\ \varphi_j(x)$$

The system above can be solved exactly in some cases. The Mathematica notebook that accompanies this lecture shows a couple of examples of using DSolve to solve systems like this.

An alternative technique is to use a numerical solution method. The next section will discuss how to apply standard numerical solutions techniques to this problem.

Numerical solution of second order systems

Consider a typical first-order system of equations. In standard form we can write this system

$$\mathbf{y}'(t) = \mathbf{f}(t,\mathbf{y}(t))$$

$$\mathbf{y}(0) = \mathbf{y}_0$$

All of the numerical solution methods we studied previously generalize in a straightforward way to systems. In particular, the standard Runge-Kutta method of order four

$$\mathbf{k}_1 = \mathbf{f}(t_k,\mathbf{y}_k)$$

$$\mathbf{k}_2 = \mathbf{f}(t_k + h/2,\mathbf{y}_k + h/2\ \mathbf{k}_1)$$

$$\mathbf{k}_3 = \mathbf{f}(t_k + h/2,\mathbf{y}_k + h/2\ \mathbf{k}_2)$$

$$\mathbf{k}_4 = \mathbf{f}(t_k + h, \mathbf{y}_k + h\,\mathbf{k}_3)$$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{3}\left(\mathbf{k}_1 + 2\,\mathbf{k}_2 + 2\,\mathbf{k}_3 + \mathbf{k}_4\right)$$

carries over to systems quite readily.

A second order differential equation takes the form

$$y''(t) = f(t, y(t), y'(t))$$

$$y(t_0) = \alpha$$

$$y'(t_0) = \beta$$

This equation can be converted to a system of two first order equations by introducing functions $u_1(t) = y(t)$ and $u_2(t) = y'(t)$. This converts the equation above into a system of equations

$$\mathbf{y}'(t) = \begin{bmatrix} u_2(t) \\ u_1(t) \end{bmatrix}' = \begin{bmatrix} f(t, u_1(t), u_2(t)) \\ u_2(t) \end{bmatrix} = \mathbf{f}(t, \mathbf{y}(t))$$

$$\mathbf{y}(t_0) = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

which can then be solved by any of our standard methods. The only complication is doing the necessary bookkeeping to construct the functions and vectors.

Let us now turn these methods to the problem at hand in section 7.3.

$$M\,\boldsymbol{\alpha}''(t) + K\,\boldsymbol{\alpha}(t) = \mathbf{g}(t)$$

$$\alpha_j(0) = \psi(x_j)$$

$$\alpha_j'(0) = \gamma(x_j)$$

(Here I have renamed the forcing function from $\mathbf{f}(t)$ to $\mathbf{g}(t)$ to prevent confusion with the function $\mathbf{f}(t, \mathbf{y}(t))$ I used above.) The first step is to put the system in standard form.

$$\boldsymbol{\alpha}''(t) = -M^{-1}\,K\,\boldsymbol{\alpha}(t) + M^{-1}\,\mathbf{g}(t)$$

$$\boldsymbol{\alpha}(0) = \boldsymbol{\psi}$$

$$\boldsymbol{\alpha}'(0) = \boldsymbol{\gamma}$$

The next step is to apply the trick from above to convert the second order system to a first order system. We introduce vectors

$$\mathbf{y}'(t) = \begin{bmatrix} \boldsymbol{\alpha}'(t) \\ \boldsymbol{\alpha}(t) \end{bmatrix}' = \begin{bmatrix} -M^{-1}\, K\, \boldsymbol{\alpha}(t) + M^{-1}\, \mathbf{g}(t) \\ \boldsymbol{\alpha}'(t) \end{bmatrix} = \mathbf{f}(t, \mathbf{y}(t))$$

$$\mathbf{y}(t_0) = \begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\psi} \end{bmatrix}$$

The only special trick needed to handle this converted system is making the expression for $\mathbf{f}(t,\mathbf{y}(t))$ look more like a matrix-vector equation. We can do this by introducing a matrix

$$A = \begin{bmatrix} 0 & -M^{-1}\, K \\ I & 0 \end{bmatrix}$$

and a vector

$$\mathbf{v}(t) = \begin{bmatrix} M^{-1}\, \mathbf{g}(t) \\ 0 \end{bmatrix}$$

The matrix A is a block-structure matrix made up of other matrices. The term $M^{-1}\, K$ is the original n by n matrix that appeared in the second order system, and the identity matrix I is an n by n identity matrix. This produces a (2 n) by (2 n) matrix A. Likewise, the vector $\mathbf{v}(t)$ is the result of stacking the original n component vector $M^{-1}\, \mathbf{g}(t)$ on top of a vector of n zeroes.

We have now rewritten our second order system into a larger first order system.

$$\mathbf{y}'(t) = A\, \mathbf{y}(t) + \mathbf{v}(t) = \mathbf{f}(t, \mathbf{y}(t))$$

$$\mathbf{y}(t_0) = \begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\psi} \end{bmatrix}$$

We can now use, say, the Runge-Kutta formulas

$$\mathbf{k}_1 = \mathbf{f}(t_k, \mathbf{y}_k)$$

$$\mathbf{k}_2 = \mathbf{f}(t_k + h/2, \mathbf{y}_k + h/2\, \mathbf{k}_1)$$

$$\mathbf{k}_3 = \mathbf{f}(t_k + h/2, \mathbf{y}_k + h/2\, \mathbf{k}_2)$$

$$\mathbf{k}_4 = \mathbf{f}(t_k + h, \mathbf{y}_k + h\, \mathbf{k}_3)$$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{3}\left(\mathbf{k}_1 + 2\, \mathbf{k}_2 + 2\, \mathbf{k}_3 + \mathbf{k}_4\right)$$

In the sequence of vectors $\mathbf{y}_k$ that we generate the bottom n components will be the values of $\boldsymbol{\alpha}_k$ that we wanted to compute. The accompanying Mathematica notebook will show a couple of examples of this in practice.