Wave Equation in One Space Dimension

This project is due at 11 PM on Thursday, April 14.

You are to implement and test a solver for the equation

$$d(x)\partial_t^2 u(x,t) - \partial_x(e(x)\partial_x u(x,t)) = 0. \tag{1}$$

Your code should approximate the solution 0f (1) on $Q = I{\times}J$, where $I = (a,b)$, $J = (0,T)$, $a < b$, and $0 < T$. The differential problem will have initial conditions

$$u(x,0) = u_0(x) \text{ and } \partial_t u(x,0) = v_0(x), \tag{2}$$

for all $x \in I$; the functions u_0 and v_0 are given. In addition there will be conditions on the lateral boundary, $\{a,b\}{\times}J$. The data on the left will be $u_{lft}(t)$ and the data on the right will be $u_{rgt}(t)$. If the the boundary condition is a Dirichlet condition the data gives the value of the solution for $t \in J$. If the boundary condition is Neumann, the data gives the "force", $-e()\partial_x u$. Other types of boundary conditions are possible, but these are the ones we will use here.

Continuous-time Galerkin

We choose to represent the approximate solution U at each time t, as a continuous piecewise linear function on I. We get a system of second order ordinary differential equations by applying a Galerkin method. Here we examine first the case of Neumann boundary conditions with the data given by zero; the modifications for Dirichlet conditions will be discussed below.

Take $\sigma = \{a = x_0 < x_1 < \ldots < x_N = b\}$ to be a mesh on I and let $\mathcal{M}$ be the space of continuous piecewise linear function over σ. The the continuous-time Galerkin approximation is given by a function U such that $U(t)$ is in $\mathcal{M}$ for each $t \in J$ and such that

$$MU'' + SU = 0, \tag{3}$$

where M and S are the mass and stiffness matrices associated with the space $\mathcal{M}$ and the coefficients $d()$ and $e()$. Here $U(t)$ is the $N+1$-vector that gives the vales of U at the points x_i.

For completeness we note that $U(x,t) = \sum_0^N U_i(t)\varphi_i(x)$ where $\varphi_i \in \mathcal{M}$, with $\varphi_i(x_j) = 1$ if $i = j$ and $\varphi_i(x_j) = 0$ if $i \neq j$. The mass matrix $M = (m_{i,j})$ and

the stiffness matrix $S = (s_{i,j})$, where

$$m_{i,j} = \int_a^b d(x)\varphi_j(x)\varphi_i(x)dx,$$
$$s_{i,j} = \int_a^b e(x)\varphi_j'(x)\varphi_i'(x)dx,$$

and i and j go from 0 to N.

Discrete-time Galerkin

The approximate solution is given by a pair of functions U and V at each time where it is defined. These two functions are continuous piecewise linears in $\mathcal{M}$. The function $U(x)$ is an approximation of $u(x,t)$ and $V(x)$ is an approximation of $\partial_t u(x,t)$.

For this project we will use a constant time step dt.

At the initial time, $t = 0$, U and V will interpolate the initial data, u_o and v_0, respectively, at the mesh points in the partition.

If U and V are our approximations at time t, then dU and dV will denote the change in these functions to advance time to $t + dt$. The relations that are satisfied by these are

$$\begin{aligned} dU &= dt(V + \frac{1}{2}dV) \\ \frac{1}{dt}MdV &+ S(U + \frac{1}{2}dU) = 0. \end{aligned} \tag{4}$$

Thinking of U, V, dU, and dV as vectors associated with the mesh points, the first relation holds at every mesh point. The second relation holds at each interior mesh point. At a Neumann boundary point with nonzero data, the 0 is modified by adding or subtracting the boundary data, evaluated at $t + dt/2$. At a Dirichlet boundary point, the value of dU is computed from the data, and dV at that boundary point is computed from the first relation in (4).

The relations in (4) can be manipulated to give

$$(\frac{1}{dt}M + \frac{dt}{4}S)dV = -S(U + \frac{dt}{2}V). \tag{5}$$

This is $(N+1)\times(N+1)$ set of linear equations. The matrices M and S are tridiagonal. At a Dirichlet boundary condition, the equation and rhs are modified to say that $dV = 2((1/dt)dU - V)$, where dU is computed from the given values on the boundary.

Testing and Demonstrating Your Code

One of the objectives in building a code to approximate the solutions of the wave equations is to learn about the behavior of those solutions. I will specify several examples so that it is easier to compare the results that you get to those that are expected.

Cases 0 and 1 show the difference in the way that Dirichlet and Neumann boundary conditions reflect disturbances that hit the boundary.

Cases 0, 2, and 3 show something profound about the differences between one, two, and three spacial dimensions. Even though the code that you have is written for one space, you can use it to examine the behavior of axisymmetric solutions in two space and the behavior of spherically symmetric solutions in three space.

In all of these examples the interval I is of length 10, $dx = 1/50$, and $dt = 1/100$. There is a named functions that are used in defining the examples:

$$flick(t) = (t(2-t))^3 \text{ on } [0,2] \text{ and } 0 \text{ otherwise.}$$

Test Cases

Case 0 $I = (1, 11)$, $d(x) = e(x) = 1$, $u_0(x) = v_0(x) = 0$, $u(a,t) = flick(t)$, $u(b,t) = 0$. Plot the approximate solution U at $t = 5$ and 13.

Case 1 as in Case 0, except it uses a boundary condition $\partial_x u(b,t) = 0$.

Case 2 like Case 0, except $d(x) = e(x) = x$. In this case you should plot $U(x)\sqrt{x}$ instead of $U(x)$

Case 3 like Case 0, except $d(x) = e(x) = x^2$. In this case you should plot $U(x)x$ instead of $U(x)$

Implementation notes

At each time step you need to solve a system of linear equations; the matrix involved is tridiagonal. These equations should be solved using that sparseness rather than using a full matrix solver. This can be done with Scipy tools.

I suggest that the generation of the S and M matrices is simpler if it is done interval by interval. The pseudo code for the mass matrix, M, with coefficient d(x) might be

```
M[0,0] = 0
for i=1, ..., N
   dx = x[i]-x[i-1]
   xmid = (x[i]+x[i-1])/2
   coef = d(xmid)
   M[i-1,i-1] += coef*dx/3
   M[[i-1,i]   = coef*dx/6
   M[[i,i-1]   = coef*dx/6
   M[[i,i]     = coef*dx/6
```

Of course, how you represent `M[i,j]` depends on how you store tridiagonal matrices.

The pseudo code for the stiffness matrix, S, with coefficient e(x) might be

```
S[0,0] = 0
for i=1, ..., N
   dx = x[i]-x[i-1]
   xmid = (x[i]+x[i-1])/2
   coef = e(xmid)
   S[i-1,i-1] += coef/dx
   S[[i-1,i]   = -coef/dx
   S[[i,i-1]   = -coef/dx
   S[[i,i]     = coef/dx
```

Conservation of Energy

Consider the case in which both boundary conditions are Neumann and suppose that the data u_{lft} and u_{rgt} are identically zero on the time interval we

examine.

If we multiply (1) by $\partial_t u$ and integrate from a to b we get

$$\begin{aligned}
0 &= \int_a^b d(x)\partial_t^2 u(x,t)\partial_t u(x,t) - \partial_x(e(x)\partial_x u(x,t))\partial_t u(x,t)dx \\
&= \int_a^b d(x)\partial_t^2 u(x,t)\partial_t u(x,t) + e(x)\partial_x u(x,t)\partial_t\partial_x u(x,t)dx \\
&= \frac{d}{dt}\frac{1}{2}\int_a^b d(x)(\partial_t u(x,t))^2 + e(x)(\partial_x u(x,t))^2 dx \\
&= \frac{d}{dt}\mathcal{E}(t),
\end{aligned}$$

where the last two lines define the energy, $\mathcal{E}(t)$.

Now we mimic the above calculation in the discrete time case by taking the dot product of the second equation in (4) with the vector (1/dt)dU to get that

$$\frac{1}{dt^2}dU^T M dV + \frac{1}{dt}dU^T S(U + \frac{1}{2}dU) = 0. \tag{6}$$

We can rewrite the first term in (6) as follows

$$\begin{aligned}
\frac{1}{dt^2}dU^T M dV &= \frac{1}{dt}(V + \frac{1}{2}dV)^T M dV \\
&= \frac{1}{2}\frac{1}{dt}((V + dV) + V)^T M((V + dV) - V) \\
&= \frac{1}{2}\frac{1}{dt}[(V + dV)^T M(V + dV) - V^T MV],
\end{aligned}$$

where we used the symmetry of M to see that

$$(A + B)^T M(A - B) = A^T MA - B^T MB.$$

Similarly we can rewrite the second term in (6) as follows

$$\begin{aligned}
\frac{1}{dt}dU^T S(U + \frac{1}{2}dU) &= \frac{1}{2}\frac{1}{dt}((U + dU) - U)^T S((U + dU) + U) \\
&= \frac{1}{2}\frac{1}{dt}[(U + dU)^T S(U + dU) - U^T SU].
\end{aligned}$$

Thus, in this case we see that the discrete energy,

$$\begin{aligned}\mathcal{E}_d &= \frac{1}{2}\frac{1}{dt}[V^T M V + U^T S U] \\ &\approx \int_a^b d(x)V(x)^2 + e(x)(\partial_x U)^2 dx\end{aligned}$$

is conserved.

If there is non-zero data for the Neumann boundary condition(s), the energy will in general not be conserved.

If one or both of the boundary conditions are Dirichlet boundary conditions and U does not change over the step at the boundary point(s), the above calculation can be modified a little to see that $\mathcal{E}_d$ is unchanged over the step. If the first component in the second equation in (4) is modified because of a Dirichlet boundary condition at a, the $dU[0]$ that multiplies that term is zero. Hence (6) still holds in this case. Similarly, if there is a Dirichlet boundary condition at b and $dU[N] = 0$ equation (6) is still correct.

The fact that the discrete scheme conserves energy provides a check for your code. During any periods in which Neumann boundary conditions are zero and Dirichlet conditions are unchanging, the discrete energy should not change.

Last modified 4/13/22 – tfd