EAA/UKAN Computational Acoustics Tutorial

Stefan Bilbao, Acoustics and Audio Group, University of Edinburgh

8 July 2021

In this tutorial, we'll look at a series of Matlab codes solving variants of the 1D and 2D wave equation, and various features of interest.

1 The 1D Wave Equation

Here, we'll be solving

$$\frac{1}{c^2}\partial_t^2\Psi = \partial_x^2\Psi \tag{1}$$

defined for $x \in [0, L]$, and given initial conditions.

1.1 Raw Solver

The Matlab code:

`weq1d_0_raw.m`

is a solver depending on the following parameter set:

- SR: sample rate (Hz)
- L: domain length (m)
- c: wave speed (m/s)
- Tf: simulation duration (s)
- xi: center of initial distribution (m)
- wid; width of initial distribution (m)
- psi0amp: amplitude of initial distribution
- lambda: Courant number

It implements the following finite difference scheme:

$$\Psi_l^{n+1} = 2\Psi_l^n - \Psi_l^{n-1} + \lambda^2\left(\Psi_{l+1}^n - 2\Psi_l^n + \Psi_{l-1}^n\right) \tag{2}$$

under Dirichlet boundary conditions.

After a walkthrough of the operation of this code (which is short), try playing with all the parameters listed above, except for the Courant number, which you leave set as 1 for the moment. We'll return to this shortly.

1.2 Energy calculation

The Matlab code:

`weq1d_1_energy.m`

is the same as the previous code, but now complemented by a calculation of the numerical energy of the scheme (to within a constant factor). The

`egy_on`

flag enables the generation of relative variation of the numerical energy. Try varying any of the defining parameters, to illustrate that the energy variation is on the order of machine precision.

1.3 Vector Matrix Form

The Matlab code:

`weq1d_2_matrixform.m`

is an implementation employing sparse matrix representations of the difference operations, to get a two step update of the form:

$$\Psi^{n+1} = \mathbf{B}\Psi^{n} - \Psi^{n-1} \tag{3}$$

1.4 Instability

The Matlab code:

`weq1d_3_unstable.m`

is an implementation with the Courant condition violated. Try playing with λ above 1 to see ho long it takes the instability to develop.

1.5 Numerical Dispersion

The Matlab code:

`weq1d_4_dispersion.m`

is intended to illustrate effects of dispersion, and how it depends on λ. You can see this both in the time domain (through the

`plot_on`

flag), or, through a spectral plot of output (using the

`specplot_on`

flag).

In this code, we have drawn an output, from location xo.

Try adjusting λ away from 1 (but below it) and look at the effects in terms of both the time response and the spectrum.

1.6 Source

The Matlab code:

```
weq1d_5_source.m
```

simulated the wave equation with point source (velocity), located at $x = x_i$.

$$\frac{1}{c^2}\partial_t^2\Psi = \partial_x^2\Psi + v_s(t)\delta(x - x_i) \tag{4}$$

In this code, we'll assume a Gaussian signal for the source term. Initial conditions are assumed zero now.

Try playing with the source location and the time constant sig.

1.7 Source: pressure and velocity plots

The Matlab code:

```
weq1d_6_source_pv.m
```

is the same as the previous code, except that now we plot physical quantities pressure and velocity, derived from the velocity potential through:

$$p = \rho\partial_t\Psi \qquad\qquad v = -\partial_x\Psi \tag{5}$$

Note that we need to introduce a new constant ρ, the air density here.

1.8 Neumann conditions

The Matlab code:

```
weq1d_7_neumann.m
```

is an implementation with Neumann conditions. Notice that this amounts to an adjustment of values in the corners of the Laplacian matrix. Otherwise the code is unchanged.

2 The 2D Wave Equation

Here, we'll be solving

$$\frac{1}{c^2}\partial_t^2\Psi = \left(\partial_x^2 + \partial_y^2\right)\Psi \tag{6}$$

defined for $x \in [0, L]$ and $y \in [0, L]$, and given initial conditions.

2.1 Raw Solver

The Matlab code:

`weq2d_0_raw.m`

is a solver depending on the following parameter set:

- SR: sample rate (Hz)
- L: domain side length (m)
- c: wave speed (m/s)
- Tf: simulation duration (s)
- ctr: center coordinates of initial distribution (m)
- wid; radius of initial distribution (m)
- psi0amp: amplitude of initial distribution
- lambda: Courant number

It implements the following "five-point" finite difference scheme:

$$\Psi_{l,m}^{n+1} = 2\Psi_{l,m}^{n} - \Psi_{l,m}^{n-1} + \lambda^2 \left(\Psi_{l+1,m}^{n} + \Psi_{l-1,m}^{n} + \Psi_{l,m+1}^{n} + \Psi_{l,m-1}^{n} - 4\Psi_{l,m}^{n}\right) \quad (7)$$

under Dirichlet boundary conditions.

After a walkthrough of the operation of this code (which is short), try playing with all the parameters listed above, except for the Courant number, which you leave set as $1/\sqrt{2}$ for the moment. We'll return to this shortly.

In particular: you can see anisotropic numerical dispersion easily by making the radius of the initial distribution small (say wid = 0.02). But, you can then counteract this effect by raising the sample rate! This is the classic trade-off.

2.2 Energy

The Matlab code:

`weq2d_1_energy.m`

is the same as the previous code, except that now it computes the numerical energy in the loop.

2.3 Matrix/vector Form

The Matlab code:

`weq2d_2_matrixform.m`

is the same as the previous code, except that now it computes according to a matrix vector mulitplication. To this end, the grid functions are now reshaped to be vectors.

The difference matrices can be constructed easily using the kron operator (implementing a Kronecker product).

2.4 Neumann conditions

The Matlab code:

```
weq2d_3_neumann.m
```

is a slight modification to implement Neumann conditions. This can be done through the adjustment of the matrix Laplacian corner elements.

2.5 Point source

The Matlab code:

```
weq2d_4_source.m
```

includes a Gaussian point source at coordinates (x_i, y_i) according to

$$\frac{1}{c^2}\partial_t^2\Psi = \nabla^2\Psi + v_s(t)\delta(x - x_i)\delta(y - y_i) \tag{8}$$

2.6 Dipole source

The Matlab code:

```
weq2d_5_dipole.m
```

includes a Gaussian source signal driving a dipole term aligned with the x direction, according to

$$\frac{1}{c^2}\partial_t^2\Psi = \nabla^2\Psi + v_s(t)\partial_x\delta(x - x_i)\delta(y - y_i) \tag{9}$$