Acoustic Wave Propagator—A Split Region Implementation

Neil Riste
Bradley McGrath
Jingbo Wang
Jie Pan

In this article we develop and implement a split region technique that solves the time-dependent acoustic wave equation with greatly increased efficiency. This method uses a sophisticated Chebyshev propagation scheme in areas where there are interfaces and medium variations, and a simple free space propagator where the medium is homogeneous. *Mathematica* provides a cohesive and interactive environment, where the mathematical functions and visualization tools required for this work are already built in. The interactive interface allows users to modify the code and study specific problems with ease.

■ The Acoustic Wave Propagator

□ Theory

In an earlier paper by Pan and Wang [1], an explicit acoustic wave propagator (AWP) was proposed to describe the time-domain evolution of mechanical waves in various media. This method was based on a similar scheme that was originally developed by Tal-Ezer and Kosloff in [2, 3, 4, 5, 6], who studied seismic wave propagation and a variety of gas-phase reactive scattering and related chemical processes. The AWP method has been successfully applied to study both the propagation of a flexural wave in a thin plate by Peng, Pan, and Sum [7] and an acoustic wave in a room by Sun, Wang, and Pan [8] and [9]. However, this method requires significant computer resources since the AWP propagation scheme utilizes a large set of modified Chebyshev polynomials with Bessel functions of the first kind as the expansion coefficients.

In this article we develop and implement a split region technique in *Mathematica* that uses the sophisticated Chebyshev propagation scheme in areas where there are interfaces and medium variations, but a simple and much more efficient free space propagator where the medium is homogeneous.

First, we give a brief outline of the AWP and then introduce the Chebyshev propagation scheme. This method is then implemented in *Mathematica* code. In a later section, the free space propagator is defined, along with a method of integration that uses Fourier transforms. Finally, the split region technique is constructed and propagation is demonstrated across a splitting region with a boundary.

The motion of acoustic waves in air and solids can be described by a partial differential equation known as the acoustic wave equation:

$$\frac{\partial}{\partial t}\Phi(x, t) = -\hat{\mathcal{H}}\,\Phi(x, t). \tag{1}$$

Integrating this with respect to time, yields a formal solution to this equation:

$$\Phi(x, t) = e^{-(t - t_0)\hat{\mathcal{H}}}\,\Phi(x, t_0), \tag{2}$$

where x denotes the spatial coordinates collectively and t stands for time, with t_0 being the initial starting time. Φ is a state vector, while $\hat{\mathcal{H}}$ is the system Hamiltonian that describes the physical properties of the propagation and the boundary medium. In the case of a one-dimensional duct, Φ describes the sound pressure $p(x, t)$ and the particle velocity $v(x, t)$:

$$\Phi = \begin{pmatrix} p(x,\ t) \\ v(x,\ t) \end{pmatrix}, \tag{3}$$

and $\hat{\mathcal{H}}$ is of the form:

$$\hat{\mathcal{H}} = \begin{pmatrix} 0 & \rho\, c^2 \frac{\partial}{\partial x} \\ \frac{1}{\rho}\,\frac{\partial}{\partial x} & 0 \end{pmatrix}, \tag{4}$$

where c is the speed of sound within the medium, and ρ is the density of the medium.

The AWP is defined as:

$$\hat{\mathcal{U}} = e^{-(t - t_0)\hat{\mathcal{H}}}. \tag{5}$$

However, this exponential operator is impractical in its current form, and thus it must be expanded as a finite polynomial. In this work we use a Chebyshev polynomial expansion. To ensure the convergence of this expansion, the system Hamiltonian $\hat{\mathcal{H}}$ must be normalized:

$$\hat{\mathcal{H}}' = \frac{\hat{\mathcal{H}}}{\sqrt{\lambda_{\max}}}, \tag{6}$$

where $\lambda_{\max}$ is the maximum eigenvalue of the system operator $\hat{\mathcal{H}}$. In the case of sound pressure in a one-dimensional duct, this value is:

$$\lambda_{\max} = \left(\frac{c\,\pi}{\Delta x}\right)^2. \tag{7}$$

If we let:

$$R = (t - t_0)\sqrt{\lambda_{\max}}\,, \tag{8}$$

then the AWP may be expressed as:

$$\hat{\mathcal{U}} = e^{-(t-t_0)\hat{\mathcal{H}}} = e^{-R\hat{\mathcal{H}}'}. \tag{9}$$

The next step is to make a simple, if somewhat nonintuitive, change of variables. We let:

$$X' = i\mathrm{X}. \tag{10}$$

Then we expand the exponential operator in terms of the Chebyshev polynomials $T_n(X')$:

$$e^{-RX} = e^{iRX'} = \sum_{n=0}^{\infty} b_n(R)\,T_n(X'). \tag{11}$$

Using the orthogonality relationship for the Chebyshev polynomials, we find that the coefficients $b_n(R)$ are:

$$b_n(R) = \frac{c_n}{\pi}\int_{-1}^{1}\frac{e^{iRX'}\,T_n(X')}{\sqrt{1 - X'^2}}\,dX' = \frac{c_n}{2\pi}\int_{-\pi}^{\pi} e^{iR\cos(\theta) + in\theta}\,d\theta = i^n\,c_n\,J_n(R), \tag{12}$$

where $c_0 = 1$ and $c_n = 2$ for $n > 0$, and J_n is a Bessel function of the first kind. However, there are complex numbers involved in this expansion, while the state vector and operator are real. Hence, we define a new set of modified Chebyshev polynomials, as given by:

$$\mathcal{T}_n(X) = i^n\,T_n(i\,X). \tag{13}$$

It can be shown that these satisfy the following recursion relation:

$$\mathcal{T}_{n+1}(X) = -2\,X\,\mathcal{T}_n(X) + \mathcal{T}_{n-1}(X), \tag{14}$$

with $\mathcal{T}_0(X) = 1$ and $\mathcal{T}_1(X) = -X$. It is now possible for us to write the AWP, $\hat{\mathcal{U}}$, in the form:

$$\hat{\mathcal{U}} = e^{-(t-t_0)\hat{\mathcal{H}}} = e^{-R\hat{\mathcal{H}}'} = \sum_{n=0}^{\infty} c_n\,J_n(R)\,\mathcal{T}_n(\hat{\mathcal{H}}'), \tag{15}$$

which only involves real-valued operations. We now obtain our state vector with an expanded AWP:

$$\Phi(x,\,t) = \hat{\mathcal{U}}\,\Phi(x,t_0) = \sum_{n=0}^{\infty} c_n\,J_n(R)\,\mathcal{T}_n(\hat{\mathcal{H}}')\,\Phi(x,\,t_0), \tag{16}$$

which appears in full as:

$$\begin{pmatrix} p(x,\ t) \\ v(x,\ t) \end{pmatrix} = \sum_{n=0}^{\infty} c_n\, J_n\left((t - t_0)\sqrt{\lambda_{\max}}\right)\mathcal{T}_n\left(\frac{1}{\sqrt{\lambda_{\max}}}\begin{pmatrix} 0 & \rho\, c^2 \frac{\partial}{\partial x} \\ \frac{1}{\rho}\frac{\partial}{\partial x} & 0 \end{pmatrix}\right)\begin{pmatrix} p(x,\ t_0) \\ v(x,\ t_0) \end{pmatrix}. \tag{17}$$

The benefit of this method is that the Bessel functions decay exponentially with the coefficient index n when $n > R$. These properties are very useful for numerical computation, as they allow expansions of the exponential function to be accurately calculated for arbitrarily large values of R (that is, arbitrarily large time steps).

□ Numerical Implementation

■ *Discretising the Problem*

In this section we closely follow the work of Falloon and Wang [10], with the necessary changes for the AWP. In particular, we need to handle two components representing the sound wave, instead of just one as in the electronic wavefunction.

The first step is to create a one-dimensional grid of length L that contains n points, which stands for our spatial dimension x.

```
PositionGrid[L_, n_] := Table[-L/2 + q L/n, {q, 0, n - 1}] // N
```

We also a need a similar grid in k-space, as the evaluation of derivatives involves taking the Fourier transform of the functions:

```
KSpaceGrid[L_, n_] := Table[-π n/L + 2 π q/L, {q, 0, n - 1}] // N
```

This grid has a grid spacing of $\Delta k = 2\pi / L$, and consequently can represent a maximum wave-number of $k_{\max} = \pi\, n / L$:

```
kmax[L_, n_] := π n/L // N
```

A norm function is defined for both the position and the k-space grids:

```
PositionNorm[ψgrid_, L_, n_] := Plus @@ (Abs[ψgrid])^2 L/n

KSpaceNorm[Ψgrid_, L_, n_] := Plus @@ (Abs[Ψgrid])^2 2 π/L
```

The following functions are defined to plot the pressure and velocity distributions in both the position and *k*-spaces.

```
PositionPlot[ψgrid_, L_, n_, opts___] :=
 ListLinePlot[Thread[{PositionGrid[L, n], Re[ψgrid]}],
  opts, PlotRange → All]

KSpacePlot[Ψgrid_, L_, n_, opts___] :=
 ListLinePlot[Thread[{KSpaceGrid[L, n], Re[Ψgrid]}],
  opts, PlotRange → All]
```

The position and *k* distribution functions form a Fourier transform pair and are related by the transformations:

$$\begin{aligned}\Psi(k,\ t) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \psi(x,\ t)\, e^{-i k x}\, dx,\\ \psi(x,\ t) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \Psi(k,\ t)\, e^{i k x}\, dk.\end{aligned} \tag{18}$$

For discretised functions, such as those being used here, the Fast Fourier Transform (FFT) algorithm may be used. The built-in Fourier functions of *Mathematica* utilise this algorithm. The functions defined next allow us to transform distributions between the position and *k*-spaces. The `RotateLeft` command allows us to easily shift our distributions from the interval $[0, L]$ to $[-L/2, L/2]$, in order to provide a centred transform. These distributions are normalised by the factor $L \big/ \sqrt{2\pi n}$.

```
ToKSpaceGrid[ψgrid_, L_, n_] :=
    L
 ------- RotateLeft[InverseFourier[RotateLeft[ψgrid, n / 2]],
 √2 π n
    n / 2] // N

ToPositionGrid[Ψgrid_, L_, n_] :=
 √2 π n
 ------- RotateLeft[Fourier[RotateLeft[Ψgrid, n / 2]], n / 2] //
    L
  N
```

To test the AWP, we use Gaussian distributions for both the pressure and the velocity components of the wave description. A Gaussian distribution is of the form:

$$\psi(x) = \left(\sqrt{\pi}\ \sigma\right)^{-\frac{1}{2}} e^{\left(-\frac{x^2}{2\sigma^2}\right)}, \tag{19}$$

where its width is determined by σ. This defines a Gaussian function of this form:

```
Gaussian[x_] := (√π $σ)^(-1/2) e^(-x²/(2 $σ²)) // N // Chop
```

The speed of sound in the medium (c) and the density of the material (ρ) should not change throughout the calculation of the propagation. Other parameters that should not change are the initial pressure pulse spread (σ) and position ($x0$), the duct length (L), the propagation time (*time*), and the numbers of grid points for the entire region (*num*) and the Chebyshev region (*nint*). These have been defined using global variables, denoted with the *Mathematica* `$name` notation.

```
$c = 344.0;
$ρ = 1.21;
$σ = 1.5;
$x0 = 70;
$L = 500;
$time = 0.4;
$num = 1024;
$nint = 256;
```

Here we turn off some unimportant *Mathematica* warning messages.

```
Off[CompiledFunction::cfte, CompiledFunction::cfex,
 CompiledFunction::cfta, CompiledFunction::cfse]
```

■ *Defining the Propagator*

Both components of the system Hamiltonian $\hat{\mathcal{H}}'$ in the AWP apply a first-order derivative to each component of the state vector $\Phi(x, t)$. Thus, a good starting place for the implementation of the propagator is to define a discrete differentiation operator. `DelGrid` takes a given function, labelled as `ψgrid`, and computes its derivative via Fourier transformations.

```
DelGrid[ψgrid_, L_, n_] :=
 ToPositionGrid[ⅈ KSpaceGrid[L, n] ToKSpaceGrid[ψgrid, L, n],
   L, n] // Chop
```

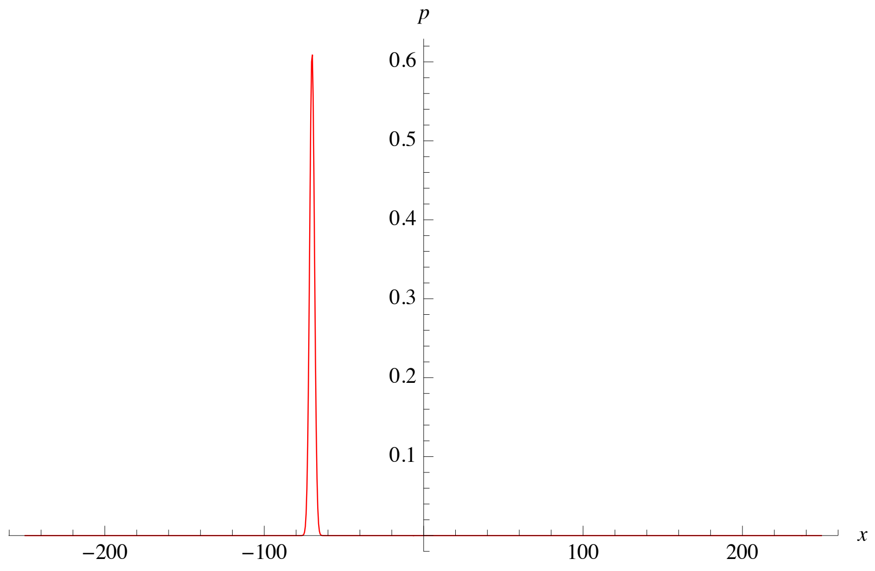
The function `Propagator` implements the Chebyshev propagation scheme. First a number of definitions are made, then a `Do` loop is constructed to perform and repeat the Chebyshev expansion the number of times determined by the term `M`. The pressure and velocity distributions are handled simultaneously, due to their related definitions. The whole set of commands is contained within a `Module` function so that all definitions remain local. Finally, we use `Compile` on the overall function, which keeps the calculation time down by keeping all values numerical.

```
Propagator =
  Compile[{{pgrid, _Real, 1}, {vgrid, _Real, 1},
    {L, _Real}, {n, _Real}, {t, _Real}},
   Module[{sqrλmax = $c kmax[L, n], R, M, ϕp0, ϕv0, ϕp1,
     ϕv1, ϕp2, ϕv2, psum, vsum}, R = sqrλmax t;
    M = Ceiling[1.2 R + 30];
    ϕp0 = pgrid;
    ϕv0 = vgrid;
    ϕp1 = - 1/sqrλmax $ρ $c^2 DelGrid[vgrid, L, n];
    ϕv1 = - 1/sqrλmax 1/$ρ DelGrid[pgrid, L, n];
    psum = BesselJ[0, R] ϕp0 + 2 BesselJ[1, R] ϕp1;
    vsum = BesselJ[0, R] ϕv0 + 2 BesselJ[1, R] ϕv1;
    Do[ϕp2 = -2 1/sqrλmax $ρ $c^2 DelGrid[ϕv1, L, n] + ϕp0;
      ϕv2 = -2 1/sqrλmax 1/$ρ DelGrid[ϕp1, L, n] + ϕv0;
      psum += 2 BesselJ[q, R] ϕp2;
      vsum += 2 BesselJ[q, R] ϕv2;
      ϕp0 = ϕp1;
      ϕp1 = ϕp2;
      ϕv0 = ϕv1;
      ϕv1 = ϕv2;,
      {q, 2, M}];
    {psum, vsum}], {{KSpaceGrid[__], _Real, 1},
    {ToKSpaceGrid[__], _Real, 1},
    {ToPositionGrid[__], _Real, 1},
    {DelGrid[__], _Real, 1}}];
```

■ Testing the Propagator

Now we carry out a few tests of this propagation scheme. Here the initial pressure and velocity distributions are defined, followed by their respective derivatives. First the pressure:

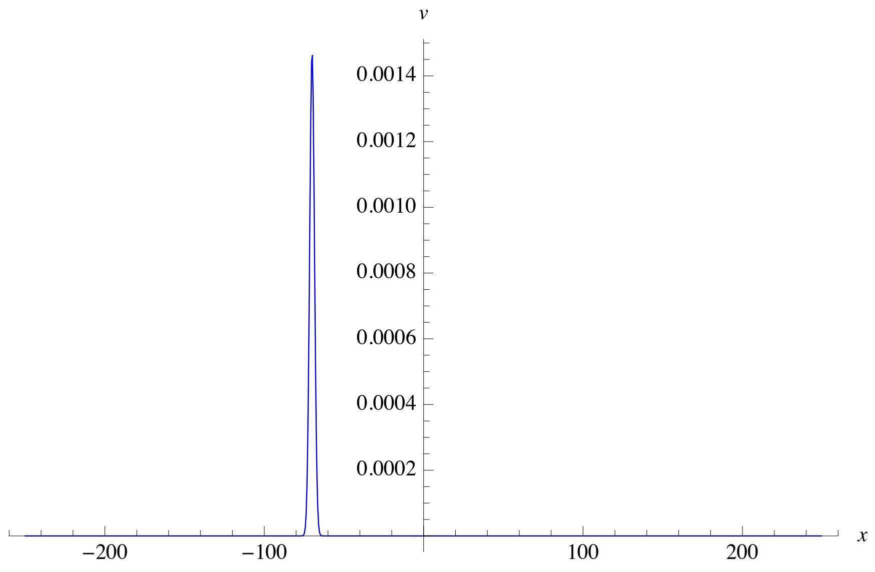
```
initpgrid = Gaussian[PositionGrid[$L, $num] + $x0];
initpplot = PositionPlot[initpgrid, $L, $num,
  PlotStyle → RGBColor[1, 0, 0], AxesLabel → {x, p}]
```



Then the velocity:

$$\texttt{initvgrid} = \frac{1}{\$\rho\ \$c}\ \texttt{Gaussian[PositionGrid[\$L, \$num] + \$x0];}$$

```
initvplot = PositionPlot[initvgrid, $L, $num,
  PlotStyle → RGBColor[0, 0, 1], AxesLabel → {x, v}]
```


v
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
x
−200
−100
100
200