

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:

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

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

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

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 $v(x, t)$:

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

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

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

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

The AWP is defined as:

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

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:

$$\hat{\mathcal{H}}' = \frac{\hat{\mathcal{H}}}{\sqrt{\lambda_{\max}}}, \quad (6)$$

where λ_{\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:

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

If we let:

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

then the AWP may be expressed as:

$$\hat{\mathcal{U}} = e^{-(t-t_0)\hat{\mathcal{H}}} = e^{-R\hat{\mathcal{H}}}. \quad (9)$$

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

$$X' = iX. \quad (10)$$

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

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

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

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

where $c_0 = 1$ and $c_n = 2$ for $n > 0$, and \mathcal{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:

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

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

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

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:

$$\hat{\mathcal{U}} = e^{-(t-t_0)\hat{\mathcal{H}}} = e^{-R\hat{\mathcal{H}}} = \sum_{n=0}^{\infty} c_n \mathcal{J}_n(R) \mathcal{T}_n(\hat{\mathcal{H}}), \quad (15)$$

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

$$\Phi(x, t) = \hat{\mathcal{U}}\Phi(x, t_0) = \sum_{n=0}^{\infty} c_n \mathcal{J}_n(R) \mathcal{T}_n(\hat{\mathcal{H}})\Phi(x, t_0), \quad (16)$$

which appears in full as:

$$\begin{pmatrix} p(x, t) \\ v(x, t) \end{pmatrix} = \sum_{n=0}^{\infty} c_n \mathcal{J}_n((t - t_0) \sqrt{\lambda_{\max}})$$

$$\mathcal{T}_n \left(\frac{1}{\sqrt{\lambda_{\max}}} \begin{pmatrix} 0 & \rho c^2 \frac{\partial}{\partial x} \\ \frac{1}{\rho} \frac{\partial}{\partial x} & 0 \end{pmatrix} \right) \begin{pmatrix} p(x, t_0) \\ v(x, t_0) \end{pmatrix}. \quad (17)$$

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 .

$$In[1]:= \text{PositionGrid}[L_, n_] := \text{Table}\left[\frac{-L}{2} + \frac{q L}{n}, \{q, 0, n - 1\}\right] // N$$

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

$$In[2]:= \text{KSpaceGrid}[L_, n_] := \text{Table}\left[\frac{-\pi n}{L} + \frac{2 \pi q}{L}, \{q, 0, n - 1\}\right] // N$$

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

$$In[3]:= \text{kmax}[L_, n_] := \frac{\pi n}{L} // N$$

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

$$In[4]:= \text{PositionNorm}[\psi\text{grid}_, L_, n_] := \text{Plus}@@(\text{Abs}[\psi\text{grid}])^2 \frac{L}{n}$$

$$In[5]:= \text{KSpaceNorm}[\Psi\text{grid}_, L_, n_] := \text{Plus}@@(\text{Abs}[\Psi\text{grid}])^2 \frac{2 \pi}{L}$$

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

```

In[6]:= PositionPlot[ψgrid_, L_, n_, opts___] :=
  ListPlot[Thread[{PositionGrid[L, n], Re[ψgrid]}],
    opts, PlotJoined → True, PlotRange → All]

In[7]:= KSpacePlot[Ψgrid_, L_, n_, opts___] :=
  ListPlot[Thread[{KSpaceGrid[L, n], Re[Ψgrid]}],
    opts, PlotJoined → True, PlotRange → All]

```

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

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

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 $L/\sqrt{2\pi n}$.

```

In[8]:= ToKSpaceGrid[ψgrid_, L_, n_] :=  $\frac{L}{\sqrt{2\pi n}}$ 
  RotateLeft[InverseFourier[RotateLeft[ψgrid, n/2]], n/2] // N

In[9]:= ToPositionGrid[Ψgrid_, L_, n_] :=
   $\frac{\sqrt{2\pi n}}{L}$  RotateLeft[Fourier[RotateLeft[Ψgrid, n/2]], n/2] // 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:

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

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

```

In[10]:= Gaussian[x_] := ( $\sqrt{\pi}$  $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 (x_0), 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.

```

In[11]:= $c = 344.0;
  $ρ = 1.21;

```

```

 $\sigma$  = 1.5;
 $x_0$  = 70;
 $L$  = 500;
 $time$  = 0.4;
 $num$  = 1024;
 $mint$  = 256;

```

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

```

In[19]:= Off[CompiledFunction::cfte, CompiledFunction::cfex,
           CompiledFunction::cfta, CompiledFunction::cfse]

```

Defining the Propagator

Both components of the system Hamiltonian \hat{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 ψ_{grid} , and computes its derivative via Fourier transformations.

```

In[20]:= DelGrid[ $\psi_{grid}$ _, L_, n_] := ToPositionGrid[
           i KSpaceGrid[L, n] ToKSpaceGrid[ $\psi_{grid}$ , 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.

```

In[21]:= Propagator =
           Compile[{{pgrid, _Real, 1}, {vgrid, _Real, 1}, {L, _Real}, {n, _Real},
                  {t, _Real}}, Module[{sqr $\lambda$ max =  $c$  kmax[L, n], R, M,  $\phi_0$ ,
                   $\phi_{v0}$ ,  $\phi_{p1}$ ,  $\phi_{v1}$ ,  $\phi_{p2}$ ,  $\phi_{v2}$ , psum, vsum}, R = sqr $\lambda$ max t;
                  M = Ceiling[1.2 R + 30];
                   $\phi_{p0}$  = pgrid;
                   $\phi_{v0}$  = vgrid;
                   $\phi_{p1}$  = -  $\frac{1}{sqr\lambda max}$   $\rho$   $c^2$  DelGrid[vgrid, L, n];
                   $\phi_{v1}$  = -  $\frac{1}{sqr\lambda max}$   $\frac{1}{\rho}$  DelGrid[pgrid, L, n];
                  psum = BesselJ[0, R]  $\phi_{p0}$  + 2 BesselJ[1, R]  $\phi_{p1}$ ;
                  vsum = BesselJ[0, R]  $\phi_{v0}$  + 2 BesselJ[1, R]  $\phi_{v1}$ ;
                  Do[ $\phi_{p2}$  = -2  $\frac{1}{sqr\lambda max}$   $\rho$   $c^2$  DelGrid[ $\phi_{v1}$ , L, n] +  $\phi_{p0}$ ;
                      $\phi_{v2}$  = -2  $\frac{1}{sqr\lambda max}$   $\frac{1}{\rho}$  DelGrid[ $\phi_{p1}$ , L, n] +  $\phi_{v0}$ ;
                     psum += 2 BesselJ[q, R]  $\phi_{p2}$ ;
                     vsum += 2 BesselJ[q, R]  $\phi_{v2}$ ;

```

```

 $\phi_{p0} = \phi_{p1}$ ;
 $\phi_{p1} = \phi_{p2}$ ;
 $\phi_{v0} = \phi_{v1}$ ;
 $\phi_{v1} = \phi_{v2}$ ; ,
{q, 2, M}];
{psum, vsum}],
{{KSpaceGrid[_], _Real, 1}, {ToKSpaceGrid[_], _Real, 1},
{ToPositionGrid[_], _Real, 1}, {DelGrid[_], _Real, 1}}];

```

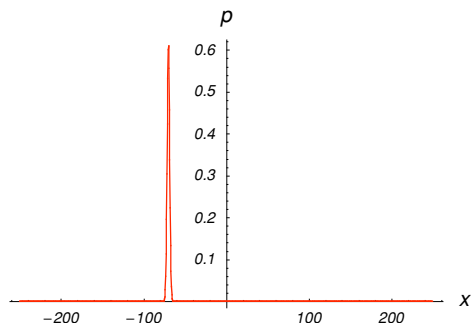
Testing the Propagator

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:

```

In[22]:= initpgrid = Gaussian[PositionGrid[$L, $num] + $x0];
initpplot = PositionPlot[initpgrid,
  $L, $num, PlotStyle -> RGBColor[1, 0, 0], AxesLabel ->
  {StyleForm["x", FontSize -> 8], StyleForm["p", FontSize -> 8]}]

```

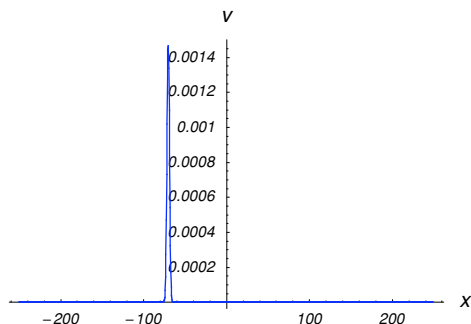


Then the velocity:

```

In[24]:= initvgrid =  $\frac{1}{\rho c}$  Gaussian[PositionGrid[$L, $num] + $x0];
initvplot = PositionPlot[initvgrid, $L,
  $num, PlotStyle -> RGBColor[0, 0, 1], AxesLabel ->
  {StyleForm["x", FontSize -> 8], StyleForm["v", FontSize -> 8]}]

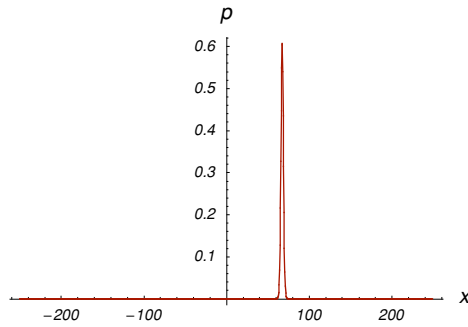
```



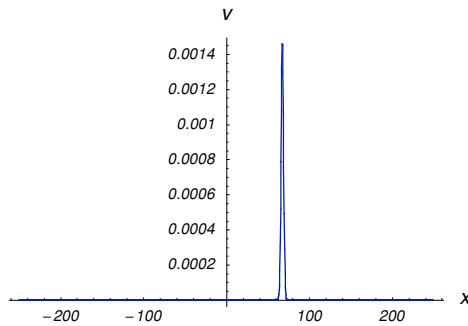
Now we run the Chebyshev expansion propagation scheme for these distributions. This returns the pressure and velocity distributions of the acoustic wave after it has propagated for $t = 3.5 \times 10^{-2}$ seconds.

```
In[26]:= result = Propagator[initpgrid, initvgrid, $L, $num, $time];
```

```
In[27]:= finalpplot = PositionPlot[result[[1]], $L,
      $num, PlotStyle → RGBColor[0.6, 0, 0], AxesLabel →
      {StyleForm["x", FontSize → 8], StyleForm["p", FontSize → 8]}]
```

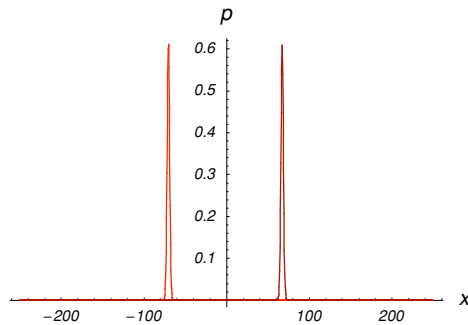


```
In[28]:= finalvplot = PositionPlot[result[[2]], $L,
      $num, PlotStyle → RGBColor[0, 0, 0.6], AxesLabel →
      {StyleForm["x", FontSize → 8], StyleForm["v", FontSize → 8]}]
```

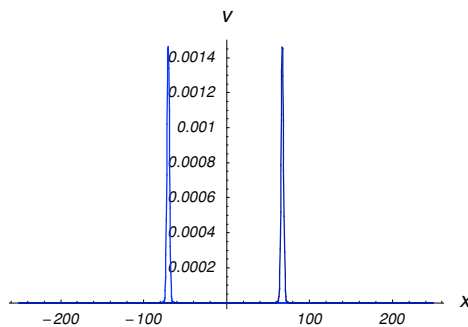


Now we compare the initial and final pressure and velocity distributions and can see that the acoustic wave has indeed propagated to the right.

```
In[29]:= Show[initpplot, finalpplot, AxesLabel →
  {StyleForm["x", FontSize → 8], StyleForm["p", FontSize → 8]]
```



```
In[30]:= Show[initvplot, finalvplot, AxesLabel →
  {StyleForm["x", FontSize → 8], StyleForm["v", FontSize → 8]]
```



The plots show a Gaussian sound pulse propagating to the right, travelling a distance of approximately 12 metres in $t = 3.5 \times 10^{-2}$ seconds. This corresponds to a wave speed of approximately 340 m/s^{-1} , the expected speed of sound in the material. This observation agrees perfectly with expectations—for a sound pulse like that defined earlier, the exact solution is a Gaussian pulse travelling to the right at the speed of sound in the medium. A more exact measurement of the speed and distance travelled gives:

```
In[31]:= Displacement[result_, n_] :=
  (  $\frac{\$L}{n}$  Flatten[Position[result[[1]], Max[Re[result[[1]]]]] ] -  $\frac{\$L}{2}$  ) -
  (  $\frac{\$L}{n}$  Flatten[Position[initpgrid, Max[Re[initpgrid]]] ] -  $\frac{\$L}{2}$  ) // N
```

```
In[32]:= Displacement[result, $num]
```

```
Out[32]= {137.207}
```

```
In[33]:= % / $time
```

```
Out[33]= {343.018}
```

This measurement confirms that the propagation speed of the sound wave agrees with the specified speed of sound in air.

■ The Region Splitting Propagator

The aim of our work has been to develop an algorithm that propagates an acoustic wave through a region that is mainly free space, but also has regions where there are changes in the media and obstructions to the wave's propagation. In principle, the Chebyshev expansion scheme is sufficient to perform any calculation that is necessary in this situation, but for a large grid, the computational effort that it requires becomes prohibitive. The exact solution that is available for regions of free, homogeneous space involves less computational effort, but is not appropriate for regions where there are changes in the media or obstructions to the wave. Because of this, we construct an algorithm where the space is divided into multiple regions—regions of free space are handled by the exact solution, and for those where there are variations, the Chebyshev scheme is used.

□ The Free Space Propagator

An exact solution to the problem of the propagation of a sound wave through free space is available and applicable to any situation that is free of medium changes and interfaces.

It can be shown that the pressure (p) and particle velocity (v) waves that satisfy the equation:

$$\frac{\partial}{\partial t} \Phi(x, t) = -\hat{\mathcal{H}} \Phi(x, t), \quad (20)$$

with the parameters described earlier, also satisfy the second-order wave equation:

$$\frac{\partial^2 p}{\partial t^2} - c^2 \frac{\partial^2 p}{\partial x^2} = 0, \quad (21)$$

which has the initial conditions:

$$\begin{aligned} p(x, 0) &= \phi(x), \\ \frac{\partial}{\partial t} p(x, 0) &= \psi(x). \end{aligned} \quad (22)$$

The solution to this partial differential equation is of the form:

$$p(x, t) = \frac{1}{2} [\phi(x + ct) + \phi(x - ct)] + \frac{1}{2c} \int_{x-ct}^{x+ct} \psi(\xi) d\xi. \quad (23)$$

Due to the excellent accuracy and efficiency of the numerical differentiation technique based upon the FFT [11], we seek a similar technique for the integra-

tion required in the previous equation. We substitute the function with its transform:

$$\begin{aligned}
 \int_{x-at}^{x+at} \psi(\xi, t) d\xi &= \int_{x-at}^{x+at} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \Psi(k, t) e^{ik\xi} dk d\xi \\
 &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \Psi(k, t) \left(\frac{1}{ik} \right) e^{ik\xi} \Bigg|_{x-at}^{x+at} dk \\
 &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \Psi(k, t) \left(-\frac{i}{k} \right) (e^{ik at} - e^{-ik at}) e^{ikx} dk \\
 &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \Psi(k, t) \left(\frac{2 \sin(ka t)}{k} \right) e^{ikx} dk.
 \end{aligned} \tag{24}$$

Thus the integral can be computed numerically by taking the transform of the original function, multiplying it by $2 \sin(ka t)/k$, and then taking the inverse transform. We implement the Fourier integration method as the function `FourierIntegrate`:

```

In[34]:= FinvSMG[L_, n_, t_, α_] :=
Module[{result, kgrid}, kgrid = KSpaceGrid[L, n];
result = Join[Table[ $\frac{\text{Sin}[\alpha t \text{kgrid}[[i]]}{\text{kgrid}[[i]]}$  // Chop, {i, 1,  $\frac{n}{2}$ }],
{α t}, Table[ $\frac{\text{Sin}[\alpha t \text{kgrid}[[i]]}{\text{kgrid}[[i]]}$  // Chop, {i,  $\frac{n}{2} + 2, n$ }],
result]

In[35]:= FourierIntegrate[ψgrid_, L_, n_, t_, α_] := ToPositionGrid[
2 FinvSMG[L, n, t, α] ToKSpaceGrid[ψgrid, L, n], L, n] // Chop

```

The function `FreePropagator` implements the free space propagation of an acoustic wave in a homogeneous medium. First, the constants and parameters of the system are defined, then the correct shift of grid spacings in the first two terms of equation (23), `nshift`, is calculated, and the initial pressure and velocity distributions are converted to the left and right travelling forms. Finally, the resulting pressure and velocity distributions are calculated, as well as the proper time of propagation, `actualt`. All of these commands are gathered into a single `Module` function, and `Compile` is used to ensure that they run as efficiently as possible.

```

In[36]:= FreePropagator = Compile[{{pgrid, _Real, 1},
{vgrid, _Real, 1}, {L, _Real}, {n, _Real}, {t, _Real}},
Module[{initpdvt, initvdvt, pgridrt, pgridlt, vgridrt, vgridlt,
nshift, presult, vresult, actualt}, nshift = Floor[ $\frac{t n \$c}{L}$ ];
initpdvt = -\$ρ \$c2 DelGrid[vgrid, L, n] // Chop;

```

```

initvdvt = -  $\frac{1}{\rho}$  DelGrid[pgrid, L, n] // Chop;

pgridrt = Table[0.0, {i, 1, n}];
Do[pgridrt[[i]] = pgrid[[i - nshift]], {i, 1 + nshift, n}];
pgridlt = Table[0.0, {i, 1, n}];
Do[pgridlt[[i]] = pgrid[[i + nshift]], {i, 1, n - nshift}];
vgridrt = Table[0.0, {i, 1, n}];
Do[vgridrt[[i]] = vgrid[[i - nshift]], {i, 1 + nshift, n}];
vgridlt = Table[0.0, {i, 1, n}];
Do[vgridlt[[i]] = vgrid[[i + nshift]], {i, 1, n - nshift}];

actualt = nshift  $\frac{L}{n \rho c}$ ;

presult =  $\frac{1}{2}$  (pgridrt + pgridlt) +
 $\frac{1}{2 \rho c}$  FourierIntegrate[initpdvt, L, n, actualt,  $\rho c$ ] // Chop;

vresult =  $\frac{1}{2}$  (vgridrt + vgridlt) +
 $\frac{1}{2 \rho c}$  FourierIntegrate[initvdvt, L, n, actualt,  $\rho c$ ] // Chop;

{actualt, presult, vresult}], {{DelGrid, _Real, 1},
{KSpaceGrid[_], _Real, 1}, {ToPositionGrid[_], _Real, 1},
{ToKSpaceGrid[_], _Real, 1}, {FinvSMG[_], _Real, 1},
{FourierIntegrate[_], _Complex, 1}}];

```

The FreePropagator function is used to calculate the propagation of the initial pressure and velocity distributions.

```

In[37]:= CompGresult = FreePropagator[initpgrid, initvgrid, $L, $num, $time];

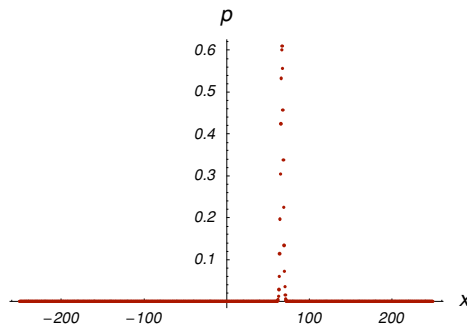
```

Here is the final pressure distribution.

```

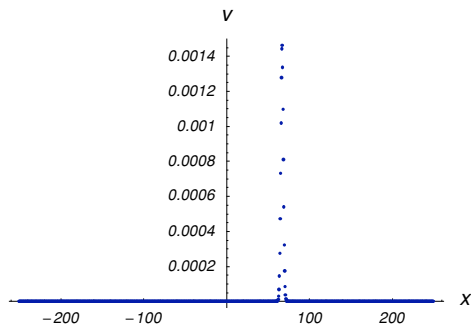
In[38]:= CFP =
ListPlot[Transpose[{PositionGrid[$L, $num], Re[CompGresult[[2]]}],
PlotRange -> All, PlotStyle -> RGBColor[0.6, 0, 0], AxesLabel ->
{StyleForm["x", FontSize -> 8], StyleForm["p", FontSize -> 8]}]

```



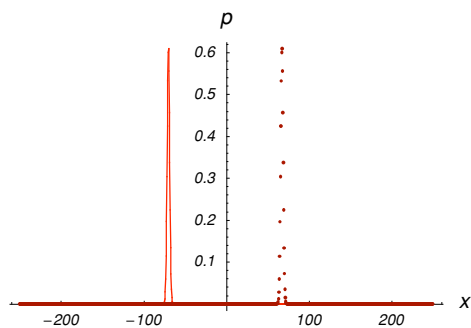
Here is the final velocity distribution.

```
In[39]:= CFV =
ListPlot[Transpose[{PositionGrid[$L, $num], Re[CompGresult[[3]]}],
PlotRange -> All, PlotStyle -> RGBColor[0, 0, 0.6], AxesLabel ->
{StyleForm["x", FontSize -> 8], StyleForm["v", FontSize -> 8]}]
```

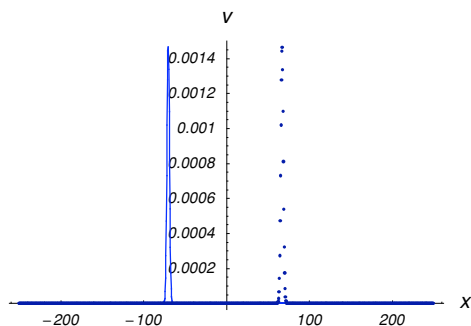


Here we compare the initial and final pressure distributions and the initial and final velocity distributions.

```
In[40]:= Show[initpplot, CFP, AxesLabel ->
{StyleForm["x", FontSize -> 8], StyleForm["p", FontSize -> 8]}]
```



```
In[41]:= Show[initvplot, CFV, AxesLabel ->
{StyleForm["x", FontSize -> 8], StyleForm["v", FontSize -> 8]}]
```



□ The Split Region Propagator

The splitting technique that we use involves multiplying the total acoustic wave distribution by what is essentially a step function, where the step occurs at the intended boundary between regions. This method divides the wave distribution into two sections that can be propagated separately, by the most appropriate technique for that region. Due to the linear nature of the splitting, these two sections can be easily recombined by adding the two resulting distributions together. The whole process can then be repeated if more than one time step is desired.

Ideally, the splitting function would be a step function:

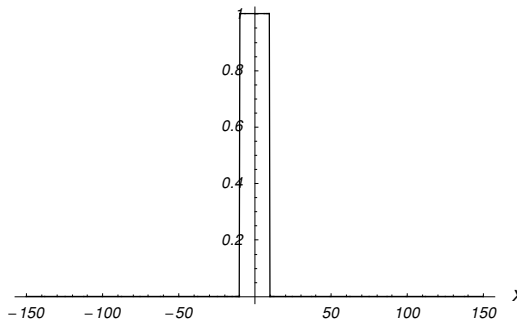
$$f(x) = \begin{cases} 1 & -x_0 \leq x \leq x_0 \\ 0 & |x| > x_0 \end{cases}, \quad (25)$$

where one region lies between $-x_0$ and x_0 , and the other is everything outside that range. This is often called a “top hat function”. However, the discontinuities that such a splitting function would introduce into the wave distribution would create large inaccuracies in the numerical techniques that are to be used. To avoid this consequence, a more gentle splitting function is used, where the step function is convolved with a Gaussian curve, so that the splitting actually occurs over some width σ , centred on $\pm x_0$. This eliminates the discontinuities of the pure step function and the associated difficulties and errors. We first define the step function:

```

In[42]:= SplittingFunction[x_, x0_] := { 1  -x0 ≤ x ≤ x0
                                         0  |x| > x0  };
Attributes[SplittingFunction] = {Listable};
$splittingwidth = 10;
Plot[SplittingFunction[x, $splittingwidth],
     {x, -150, 150}, AxesLabel → {StyleForm["x", FontSize → 8], ""}]

```



Now the function `SplittingGrid` is defined to create a grid of values from the convolution of the step function and the Gaussian. All values of this lie in the range from 0 to 1. The wave distribution is multiplied by this grid to achieve the required splitting. This calculation makes use of the fact that the transform of the convolution of two functions is the product of the transforms of the individual functions. Later, the new smooth top hat function is plotted.

```

In[46]:= SplittingGrid[L_, n_, x0_,  $\sigma$ _: 50] :=
Module[{xgrid = PositionGrid[L, n], fgrid, kernel},
  fgrid = SplittingFunction[xgrid, x0];
  kernel = Chop[RotateLeft[ $\frac{L e^{-\frac{xgrid^2}{\sigma}}}{n \sqrt{\pi \sigma}}$ ,  $\frac{n}{2}$ ]];
  Chop[InverseFourier[ $\sqrt{n}$  Fourier[fgrid] Fourier[kernel]]]]

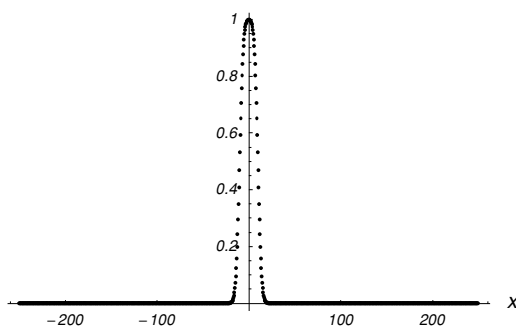
```

For the subsequent calculations, we use the following split region barrier. It is important to note that there are as yet no variations in the medium—the speed of sound and the density remain the same across all three regions.

```

In[47]:= $smoothness = 20;
fgrid = SplittingGrid[$L, $num, $splittingwidth, $smoothness];
ListPlot[Transpose[{PositionGrid[$L, $num], fgrid}],
  PlotRange -> All, AxesLabel -> {StyleForm["x", FontSize -> 8], ""}]

```



Now we define the function `SplittingPropagator`, which implements the split region propagation technique. The `SplittingGrid` function (here it is labelled as `fgrid`) is used to split the distributions, but it is the quantities `nlower`, `npad`, and `nupper` that divide the propagator into separate regions via a `Take` selection. In the outer regions, where there is only free space, the `FreePropagator` function is used, while in the central region, the Chebyshev method is used in the form of the `Propagator` function. A `Do` command is used to repeat this propagation a number of times determined by `steps`.

```

In[50]:= SplittingPropagator[pgrid_,
  vgrid_, fgrid_, L_, n_, nint_, tprop_, steps_] :=
Module[{Lint =  $\frac{L nint}{n}$ , nlower =  $\frac{(n - nint + 2)}{2}$ , npad =  $\frac{(n - nint)}{2}$ ,
  nupper =  $\frac{(n + nint)}{2}$ ,  $\phi$ total = {pgrid, vgrid},
  freeresult,  $\phi$ free,  $\phi$ int, pint, vint, actualt},
  Do[freeresult = FreePropagator[ $\phi$ total[[1]] Chop[1 - fgrid],
     $\phi$ total[[2]] Chop[1 - fgrid], L, n, tprop/steps];
   $\phi$ free = Take[freeresult, {2, 3}];
  pint = Take[ $\phi$ total[[1]] fgrid, {nlower, nupper}];

```

```

vint = Take[ $\phi$ total[[2]] fgrid, {nlower, nupper}];
actualt = freeresult[[1]];
 $\phi$ int = Propagator[pint, vint, nint, actualt];
 $\phi$ total =  $\phi$ free + Transpose[Join[Table[{0, 0}, {npad}],
    Transpose[ $\phi$ int], Table[{0, 0}, {npad}]]];, {steps}];  $\phi$ total]

```

And now we propagate the acoustic wave through both free space and the splitting region, using `SplittingPropagator`. Note that the grid should be large enough that a shift of ct in either direction does not move any significant (that is, nonzero) data outside the boundaries of the grid. We arbitrarily choose the propagation time to be 0.5 second and split the propagation into five time steps with 0.1 second each.

```

In[51]:= splitresult5 = SplittingPropagator[initpgrid,
    initvgrid, fgrid, $L, $num, $nint, $time, 5]; // Timing

```

```

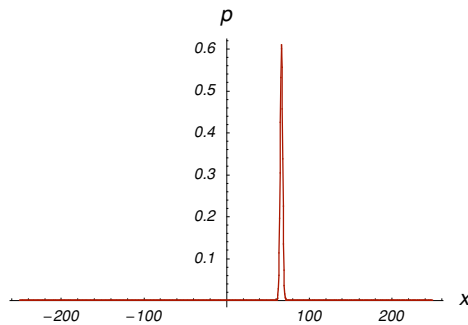
Out[51]:= {20.8254 Second, Null}

```

```

In[52]:= ListPlot[Transpose[{PositionGrid[$L, $num], Abs[splitresult5[[1]]}],
    PlotRange -> All, PlotStyle -> RGBColor[0.6, 0, 0],
    PlotJoined -> True, AxesLabel ->
    {StyleForm["x", FontSize -> 8], StyleForm["p", FontSize -> 8]}]

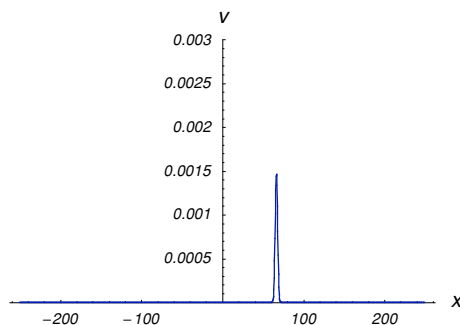
```



```

In[53]:= ListPlot[Transpose[{PositionGrid[$L, $num], Abs[splitresult5[[2]]}],
    PlotRange -> {0, 0.003}, PlotStyle -> RGBColor[0, 0, 0.6],
    PlotJoined -> True, AxesLabel ->
    {StyleForm["x", FontSize -> 8], StyleForm["v", FontSize -> 8]}]

```



The splitting propagator scheme works as expected. The sound wave propagates through the splitting and Chebyshev interaction regions without introducing any discontinuity and the norm is conserved throughout. Again, the following measure confirms that the propagation speed of the sound wave agrees well with the specified speed of sound in air.

```
In[54]:= Displacement[splitresult5, $num]
```

```
Out[54]= {136.719}
```

```
In[55]:= % / $time
```

```
Out[55]= {341.797}
```

Note that it is vital to ensure that the sizes of the various regions are appropriate—the interaction region must be larger than the splitting region, as the interaction wave packet must have enough space to get clear of the splitting zone (inside the barrier region defined by `fgrid`) before leaving the interaction zone (defined by `nint`). Otherwise the wave packet is wrapped around inside the interaction zone due to the Fourier methods used in the calculation, thus never leaving that area.

It is clear that the sizes of the splitting region, the interaction zone, and the time step need careful determination. The buffer zone in the interaction region surrounding the split region needs to be large enough that portions of the wave packet that are at the edge of the split region at the start of the time step do not travel further than the edge of the buffer region, otherwise they are artificially wrapped, thus destroying the accuracy of the scheme. The size of the buffer region needs to be greater than the distance travelled by the wave in the length of the time step:

$$L_{\text{buf}} = c \delta t.$$

This length is taken from the point where the splitting grid effectively falls to zero.

■ Conclusion

In this article we developed a highly efficient technique that solves the time-dependent acoustic wave equation using split regions, where a free space solution is used in free space, and the more sophisticated Chebyshev expansion solution is used in the interaction regions. The next stage is to incorporate variations and boundaries in the medium in order to simulate a “real-life” wave passing through air, liquids, and solids. This should only require minor adjustments to the split region technique as it is presented in this article. Preliminary work indicates the reflection and transmission of a wave passing through a barrier, as well as a change in speed as the wave passes through a denser medium.

■ Acknowledgment

The authors would like to acknowledge support from the Australian Research Council and the National Science Foundation of China (Project No. 60340420325).

For this work we used *Mathematica* 5.0 and a Pentium 4 computer running Windows XP with a 2.8 GHz CPU and 512 MB of RAM.

■ References

- [1] J. Pan and J. B. Wang, "Acoustic Wave Propagator," *Journal of the Acoustical Society of America*, **108**(2), 2000 pp. 481–487.
- [2] H. Tal-Ezer and R. Kosloff, "An Accurate and Efficient Scheme for Propagating the Time Dependent Schrödinger Equation," *Journal of Chemical Physics*, **81**(9), 1984 pp. 3967–3971.
- [3] H. Tal-Ezer, "A Pseudospectral Legendre Method for Hyperbolic Equations with an Improved Stability Condition," *Journal of Computational Physics*, **67**(1), 1986 pp. 145–172.
- [4] H. Tal-Ezer, "Spectral Methods in Time for Hyperbolic Equations," *SIAM Journal on Numerical Analysis*, **23**(1), 1986 pp. 11–26.
- [5] H. Tal-Ezer, "An Accurate Scheme for Seismic Forward Modeling," *Geophysical Prospecting*, **35**, 1987 pp. 479–490.
- [6] H. Tal-Ezer, "Polynomial Approximation of Functions of Matrices and Applications," *Journal of Scientific Computing*, **4**(1), 1989 pp. 25–60.
- [7] S. Z. Peng, J. Pan, and K. S. Sum, "Acoustical Wave Propagator Method for Time Domain Analysis of Flexural Wave Scattering and Dynamic Stress Concentration in a Heterogeneous Plate with Multiple Cylindrical Patches," *Tenth International Congress on Sound and Vibration (ICSV10)*, Stockholm, Sweden, 2003.
- [8] H. M. Sun, J. B. Wang, and J. Pan, "An Effective Algorithm for Simulating Acoustical Wave Propagation," *Computer Physics Communications*, **151**, 2003 pp. 241–249.
- [9] H. M. Sun, J. B. Wang, and J. Pan, "Acoustical Wave Propagator Method for Two-dimensional Sound Propagation," *Third International Conference on Acoustics*, University of Cadiz, Spain, 2003.
- [10] P. Falloon and J. B. Wang, "Electronic Wave Propagation with *Mathematica*," *Computer Physics Communications*, **134**(2), 2001 pp. 167–182.
- [11] J. B. Wang, "Numerical Differentiation Using Fourier," *The Mathematica Journal*, **8**(3), 2002 pp. 383–388.

About the Authors

Neil Riste and Bradley McGrath are Ph.D. students at The University of Western Australia, studying and theoretically modelling the dynamical response of both quantum and classical systems to external disturbance.

Associate Professor Jingbo Wang leads the Quantum Dynamics Theory Group at The University of Western Australia and has a wide range of interests, from atomic physics, molecular and chemical physics, spectroscopy, acoustics, chaos, nanostructured electronic devices, and mesoscopic physics to quantum information and computation.

Professor Jie Pan is the director of the Center for Acoustics, Dynamics and Vibration at The University of Western Australia. His research areas are room acoustics, structural acoustics, and active noise and vibration control.

Neil Riste^{1,2}

Bradley McGrath¹

Jingbo Wang^{1,2}

Jie Pan²

¹ *School of Physics*

² *School of Mechanical Engineering*

The University of Western Australia

35 Stirling Highway

Crawley WA 6009, Australia

wang@physics.uwa.edu.au