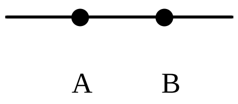
差分算子

有限差分波动方程的数值模拟的关键在于微分算子的差分近似。SMART 中的差分算子均是利用褶积算子（conv）实现的。SMART 的程序均是以 gf 为开头的，差分算子也不例外。但是有几个差分算子使用非常频繁，所以其前缀 gf 省略了。它们是 Lx1，Lz1，sbLx，sbLz，sfLz 和 sfLz。差分算子的命名规则是这样的：（以 gfLx2_3D_opt 为例）gf 为 SMART 函数前缀，L 表示算子，x，y 和 z 表示方向，后面的数字表示导数的阶数（交错网格只针对 1 阶导数，因此无数字后缀），3D 表示 3 维差分算子（2D 的后缀省略），后缀 opt 表示利用的优化差分系数。在交错网格里又分为向前和向后差分，向前和先后是相对的。比如



在计算 A 的时候，B 的位置相对滞后则利用 sbLx，在计算 B 的时候 A 的位置相对超前则利用 sfLx。这只是一种相对的概念。在后面的正演示例中会详细的讨论这些问题。这一部分先主要介绍一些基本的概念，比如差分近似，高阶差分系数的计算，频散分析和稳定性分析等。下面是规则网格和交错网格、旋转交错网格差分算子列表。

表 3-1 规则网格有限差分算子

函数名称	简单描述
Lx1	1 阶导数 12 阶规则网格 FD 算子，gfLx1_opt 的简写
Lz1	1 阶导数 12 阶规则网格 FD 算子，gfLz1_opt 的简写
gfLx1	1 阶导数 2N 阶规则网格 FD 算子
gfLz1	1 阶导数 2N 阶规则网格 FD 算子
gfLx1_opt	1 阶导数 12 阶规则网格优化差分系数 FD 算子
gfLz1_opt	1 阶导数 12 阶规则网格优化差分系数 FD 算子
gfLx2	2 阶导数 8 阶规则网格 FD 算子
gfLz2	2 阶导数 8 阶规则网格 FD 算子
gfLx2_opt	2 阶导数 8 阶规则网格优化差分系数 FD 算子
gfLz2_opt	2 阶导数 8 阶规则网格优化差分系数 FD 算子
gfLx2_3D_opt	3 维 2 阶导数 8 阶规则网格优化差分系数 FD 算子
gfLy2_3D_opt	3 维 2 阶导数 8 阶规则网格优化差分系数 FD 算子
gfLz2_3D_opt	3 维 2 阶导数 8 阶规则网格优化差分系数 FD 算子

注：这部分内容摘自笔者写的另一份文档内容（Seismic Modelling and Research with MATLAB for Absolute Beginner；虽然是用英语写的，却是标准的中式英语，相信大家都能看得懂。如看到有什么语法、拼写等错误，语句混乱表意不清等问问题，请无视之😁，作者太懒，无解！）

表 3-2 交错网格有限差分算子

函数名称	简单描述
sbLx	1 阶导数 12 阶向后 SGFD 算子，gfsbLx1_opt 的简写
sbLz	1 阶导数 12 阶向后 SGFD 算子，gfsbLz1_opt 的简写
sfLx	1 阶导数 12 阶向前 SGFD 算子，gfsfLx1_opt 的简写
sfLz	1 阶导数 12 阶向前 SGFD 算子，gfsfLx1_opt 的简写
gfsbLx1	1 阶导数 2N 阶向后 SGFD 算子
gfsbLz1	1 阶导数 2N 阶向后 SGFD 算子
gfsfLx1	1 阶导数 2N 阶向前 SGFD 算子
gfsfLz1	1 阶导数 2N 阶向前 SGFD 算子
gfsbLx1_opt	1 阶导数 12 阶优化差分系数向后 SGFD 算子
gfsbLz1_opt	1 阶导数 12 阶优化差分系数向后 SGFD 算子
gfsfLx1_opt	1 阶导数 12 阶优化差分系数向前 SGFD 算子
gfsfLz1_opt	1 阶导数 12 阶优化差分系数向前 SGFD 算子
gfsbLx1_3D_opt	3 维 1 阶导数 12 阶优化差分系数向后 SGFD 算子
gfsbLy1_3D_opt	3 维 1 阶导数 12 阶优化差分系数向后 SGFD 算子
gfsbLz1_3D_opt	3 维 1 阶导数 12 阶优化差分系数向后 SGFD 算子
gfsfLx1_3D_opt	3 维 1 阶导数 12 阶优化差分系数向前 SGFD 算子
gfsfLy1_3D_opt	3 维 1 阶导数 12 阶优化差分系数向前 SGFD 算子
gfsfLz1_3D_opt	3 维 1 阶导数 12 阶优化差分系数向前 SGFD 算子
rsgbfd	1 阶导数 12 阶优化差分系数向后 RSGFD 算子，需要 CUDA 支持
rsgffd	1 阶导数 12 阶优化差分系数向前 RSGFD 算子，需要 CUDA 支持

Contents

1 Finite Difference Method for Wave Propagation **1**

1.1 Finite difference Approximations 2

1.1.1 One-sided approximation 2

1.1.2 Centered approximation 3

1.1.3 High order approximation 3

1.1.4 Staggered-grid finite difference approximation 5

1.1.5 Rotated-staggered finite difference 7

1.1.6 Numerical Dispersion 8

1.1.7 Stability analysis 13

Chapter 1

Finite Difference Method for Wave Propagation

Personally speaking, finite difference method is the easiest numerical method among the various modeling approaches. Before the detailed discussion, let me show you an acoustic example. And you will find how simple it is. In two dimension, the acoustic equation with excitation source $f(t)$ at a position (x_0, z_0) reads as

$$\frac{\partial^2 p}{\partial t^2} = v^2 \left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial z^2} \right) + f(t)\delta(x - x_0, z - z_0) \tag{1-1}$$

where v is velocity. The approximation of derivatives by finite differences plays a central role in finite difference methods for the numerical solution of wave equations. The discrete approximation of the acoustic equation is

$$\frac{p^{n+1} - 2p^n + p^{n-1}}{\Delta t^2} = v^2 \left(D_{xx}[p^n] + D_{zz}[p^n] \right) + f(t)\delta(x - x_0, z - z_0) \tag{1-2}$$

The left-hand side of equation (1-2) represents the temporal 2nd-order central differences, and D_{xx} and D_{zz} (their specific forms is discussed in next section) in right-hand side represent the high-order space derivatives along $x-$ and $z-$ axis, respectively. From the discrete approximation equation (1-2), we get the time march scheme (Δt in the source term is omitted)

$$p^{n+1} = 2p^n - p^{n-1} + v^2 \Delta t^2 \left(D_{xx}[p^n] + D_{zz}[p^n] \right) + f(t)\delta(x - x_0, z - z_0) \tag{1-3}$$

The following program code demonstrates a complete, simple modeling program. From this code, you will learn how single program statements come together to form a complete modeling program. A complete modeling program consists of these parts: model input, grid and wavelet configuration, wavefield extrapolation, results output. In next sections, these ingredient will be discussed detailedly. In SMART, `fzeros` and `fones` are the single vesion of zeros and ones for CPU. `gzeros` and `gones` are the single vesion of gpuArray.zeros and gpuArray.ones for GPU.

```matlab
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% * Program: Numerical solution for 2nd-order acoustic equation
% * Author:  C.F. Guo
% * Date:     2015-11-11
% * Version:  1.0
% * * * * * * * * * * * * * * * * * * * * * * * * * * *  * * * *

```

```
%*************** basic configuration *****************
vp=gfin('vp.su','su');            % velocity model
[Nz,Nx]=size(vp); Nt=450;    % grid dimension and sampling length
dx=10; dz=10;  dt=0.001;       % spatial interval and time interval
p2=gzeros(Nz,Nx);              % wavefield variables, gzeros, gones for GPU
p1=gzeros(Nz,Nx);              % wavefield variables,
fzeros, fones for CPU
p0=gzeros(Nz,Nx);
sz=Nz/2; sx=Nx/2;          % source position
loc=gfloc(Nz,Nx,sz,sx);  % Delta function with the Gauss smoothing filter
fm=30;          % domain frequence
lag=40;          % the peak of wave delays for lag*dt
wlet=gfwletricker(Nt,fm,dt,lag);  % Ricker wavelet
tv=dt*dt*vp.*vp;

%*************** Time iteration *****************
for it=1:Nt
p2=2*p1−p0+tv.*(gfLx2(p1,dx)+gfLz2(p1,dz))+wlet(it)*loc;
p0=p1;
p1=p2;
% displays the snapshots at each time step
% imagesc(p2);
% getframe;
end

%*************** Output *************************
gfout(p2,'example_1.segy');
```

1.1 Finite difference Approximations

1.1.1 One-sided approximation

Let $u(x)$ represents a function of one variable that will always be assumed to be smooth, meaning that we can differentiate the function several times and each derivative is a well-defined bounded function over an interval containing a particular point of interest $\bar{x}$.Suppose we want to approximate $u'(\bar{x})$ by a finite difference approximation based on only on values of u at a finite number of points near $u'(\bar{x})$. One obvious choice would be to use

$$D_{+}u(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x})}{h} \tag{1-4}$$

for some small value of h. This is motivated by the standard definition of the derivative as the limiting value of this expression as $h \to 0$. Note that $D_{+}u(\bar{x})$ is the slope of the line interpolating u at the points $\bar{x}$ and $\bar{x}+h$ (see figure 1-1). The expression (1-4) is called as forward difference, a one-sided approximation to u' since u is evaluated only at values of $x \geqslant \bar{x}$. Another one-sided approximation (backward difference) would be

$$D_{-}u(\bar{x}) = \frac{u(\bar{x}) - u(\bar{x}-h)}{h}. \tag{1-5}$$

1.1.2 Centered approximation

Another possibility is to use the central (sometimes, use *central*) difference approximation

$$D_0 u(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x}-h)}{2h}. \tag{1-6}$$

This is the slope of the line interpolating u at $u(\bar{x}-h)$ and $u(\bar{x}+h)$ and is simply the average of the two one-sided approximation defined above. From figure 1-1 it should be clear that we would expect $D_0 u(\bar{x})$ to give a better approximation than either of the one-sided approximations. In fact this give a second order accurate approximation–the error is proportional to h^2 and hence is much smaller than the error in a first order approximation when h is small.

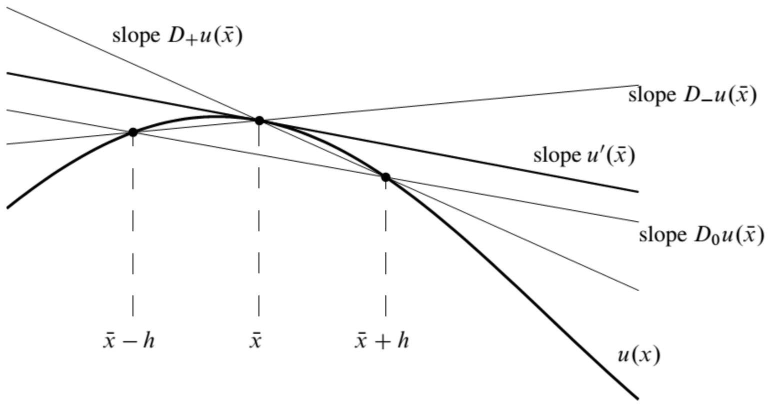


Figure 1-1 Various approximations to u' interpreted as the slope of secant lines (See LeVeque, 2007)

Finite difference approximations to higher order derivatives can also be obtained using any of the approaches outlined above. Repeatedly differencing approximations to lower order derivatives is a particularly simple approach. Approximations to the second derivative $u''(x)$ can be obtained in an analogous manner.The standard second order central approximation is given by

$$D^2 u(\bar{x}) = \frac{1}{h^2}\left[u(\bar{x}+h) - 2u(\bar{x}) + u(\bar{x}-h)\right]. \tag{1-7}$$

1.1.3 High order approximation

We can use Taylor series to derive an appropriate high order approximation to $k-$order derivative. We assume $u(x)$ is sufficiently smooth, namely, at least $n+1$ times continuously differentiable in the interval containing $\bar{x}$ and all the stencil points, so that the Taylor series expansions below are valid. Taylor series expansions of u at each point $x_{\pm i}(\bar{x} \pm ih)$ in the stencil about $u(\bar{x})$ yield

$$u(x_{+i}) = u(\bar{x}) + (ih)u'(\bar{x}) + \cdots + \frac{(ih)^k}{k!}u^{(k)}(\bar{x}) + \cdots \tag{1-8}$$

$$u(x_{-i}) = u(\bar{x}) + (-ih)u'(\bar{x}) + \cdots + \frac{(-ih)^k}{k!}u^{(k)}(\bar{x}) + \cdots \tag{1-9}$$

for $i = 1, \cdots, n$. We want to find a linear combination of these values that agrees with $u^{(k)}(\bar{x})$ as well as possible. If we only consider the central scheme, the high order finite difference approximation to the k-order derivative can be expressed as (Note that $u(x_{+i}) - u(x_{-i})$ gives odd order derivatives and $u(x_{+i}) + u(x_{-i})$ gives even order derivatives):

$$h^k u^{(k)}(\bar{x}) \approx \sum_{m=1}^{n} c_m \left[u(x_{+m}) - u(x_{-m})\right] \tag{1-10}$$

Inserting expression (1-8) and (1-9) into expression (1-10) yields

$$\sum_{i=1}^{n} 2c_i \begin{pmatrix} (ih)u'(\bar{x}) \\ \frac{(ih)^3}{3!}u^{(3)}(\bar{x}) \\ \vdots \\ \frac{(ih)^{2n-1}}{(2n-1)!}u^{(2n-1)}(\bar{x}) \end{pmatrix} = \begin{pmatrix} u'(\bar{x})h \\ u^{(3)}(\bar{x})h^3 \\ \vdots \\ u^{(2n-1)}(\bar{x})h^{2n-1} \end{pmatrix} \tag{1-11}$$

If we only consider the 1st-order derivative, this system of linear equations can be reduced into

$$\begin{pmatrix} 1^1 & 2^1 & \cdots & n^1 \\ 1^3 & 2^3 & \cdots & n^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1^{2n-1} & 2^{2n-1} & \cdots & n^{2n-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{1-12}$$

In MATLAB it is very easy to set up and solve this Vandermonde-like coefficient matrix. This part text is taken from LeVeque (2007). In this book, LeVeque (2007) provides a very stable function to compute coefficients for finite difference approximation for the derivative of order k. The value of these coefficients depends on h, which is unpractical. If we want to obtain the conventional coefficients, the h should be set to 1. The following code illustrates how to use `fdcoeffF` to obtain coefficients for central and staggered-grid finite difference approximation for 1st-order derivative. I also present two functions to compute coefficients for the 1st-order derivative: `gfcoef` and `gfstcoef`.

```
% Compute coefficients for finite difference approximation for the derivative of order k
% Time: 2015/11/12
k=1:    % The order of the derivative
N=5;  % 2N−order approximation
x=−N:1:N
cc=fdcoeffF(k,0,x); % coefficients for central finite difference approximation
x=−N+0.5:1:N−0.5;
cs=fdcoeffF(k,0,x); % coefficients for staggered−grid finite difference approximation
```

Now, let's look at how to implement finite difference in MATLAB . Equation 1-10 indicates that derivative can be expressed as the sum of the values of it's neighbouring points. This allows us use convolution to implement finite difference. Since MATLAB has the vector/matrix representation of its data, vectorization can help to make our MATLAB codes run faster. The key for vectorization is to minimize the usage of a `for-loop` and use the built-in functions as much as possible.The following code illustrates the implementation of finite difference for 1st-order derivative along $x-$ direction with the help of convolution. The trick for finite difference implementation in MATLAB is to replace the `for-loop` structures with efficient `conv` function to improve computational performance. Figure 1-2 shows the peaks and it's first-order derivative in $x-$ axis.

```
% Centered finite difference for 1st−order derivative along x−axis
% Time: 2015/11/13
M=6;
dx=10;
cm=gfcoef(M);
x=−[−cm(end:−1:1),0,cm]; % conv reverses the coefficients sequence
inp=peaks(200);    % test data
outp=conv2(inp,x,'same')/dx;
```

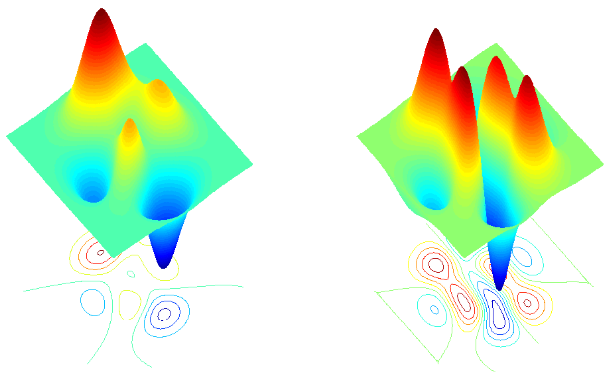


Figure 1-2 Peaks (left) and it's derivative (right) in $x-$ axis

Table 1-1 displays some commonly-used central finite difference operators. Since we seldom use the 2nd-order wave equations, the order of finite difference operators for 2nd-order derivatives is fixed. You can use `fdcoeffF` to construct your own any high-order finite difference operators for any order derivatives. The optimized coefficients will be discussed in §2.4

Table 1-1 CGFD operations

Function Name	Description	Supplement
`Lx1`	12-order FD for the 1st-derivative	same as `gfLx1_opt`
`Lz1`	12-order FD for the 1st-derivative	same as `gfLz1_opt`
`gfLx1`	2N-order FD for the 1st-derivative	
`gfLz1`	2N-order FD for the 1st-derivative	
`gfLx1_opt`	12-order FD for the 1st-derivative	optimized coef.
`gfLz1_opt`	12-order FD for the 1st-derivative	optimized coef.
`gfLx2`	8-order FD for the 2nd-derivative	
`gfLz2`	8-order FD for the 2nd-derivative	
`gfLx2_opt`	8-order FD for the 2nd-derivative	optimized coef.
`gfLz2_opt`	8-order FD for the 2nd-derivative	optimized coef.
`gfLx2_3D_opt`	8-order FD for the 3D 2nd-derivative	optimized coef.
`gfLy2_3D_opt`	8-order FD for the 3D 2nd-derivative	optimized coef.
`gfLz2_3D_opt`	8-order FD for the 3D 2nd-derivative	optimized coef.

1.1.4 Staggered-grid finite difference approximation

Compared with conventional central finite difference (CFD) methods, staggered-grid finite-difference (SGFD) methods have been more widely used in numerically solving wave equations for their greater precision. To my knowledge, staggered-grid finite-difference scheme for wave propagation modeling in seismic exploration filed is first introduced in Madariaga (1976). Later, Virieux (1984, 1986) advanced staggered-grid finite-difference schemes for simulating SH and P-SV wave propagation in 2-D elastic media (See Liu (2014) for detailed introduction about the evolution of SGFD).

Figure 1-3 shows the schematic diagram of one kind of staggered grids in which the normal stress are defined at grid points, the shear stress is defined at the centre of the grid and the partial-velocities are defined at the half of the grid (also refer to figure 1-4-C). We also can put the stress variables (normal and stress variables) at the half of the grid, and define the partial-velocities at the centre of grid and grid points (refer to figure 1-4-E, we can treat p as stress wavefields).

The finite difference approximation of first-order derivative on a staggered grid is similar to expression 1-10, represented as

$$u'(x_i) \approx \frac{1}{h}\sum_{m=1}^{n} c_m \left[u\left(x_{i+m-\frac{1}{2}}\right) - u\left(x_{i-m+\frac{1}{2}}\right)\right], \qquad (1\text{-}13)$$

where c_m are the staggered-grid finite difference coefficients. The construction of SGFD operators are a little bit complex than central FD, because the partial-velocity and stress wavefields need different treatment. If we adopt the definition shown in figure 1-3, as for v_x and v_z, τ_{xx} and τ_{zz} lead for a half grid, τ_{xz} lags for a half grid, vice versa. Taking $v_x(x_i, z_j)$ as an example, we have to use

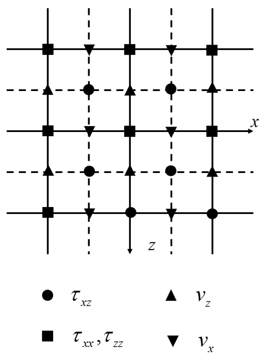


Figure 1-3 Schematic of staggered grid (Han)

$$Dxx[\tau_{xx}(x_i, z_j)] = \frac{1}{h}\sum_{m=1}^{n} c_m \left[\tau_{xx}(x_{i+m}, z_j) - \tau_{xx}(x_{i-m+1}, z_j)\right], \qquad (1\text{-}14)$$

and

$$Dzz[\tau_{xz}(x_i, z_j)] = \frac{1}{h}\sum_{m=1}^{n} c_m \left[\tau_{xz}(x_i, z_{j+m-1}) - \tau_{xx}(x_i, z_{j-m})\right] \qquad (1\text{-}15)$$

to update $v_x(x_i)$ by $v_x^{n+1}(x_i) = v_x^n(x_i) + (dt/\rho)(Dxx[\tau_{xx}] + Dzz[\tau_{xz}])$ as shown in the following code. The prefix `sf` is the abbreviation of *staggered-grid forward* finite difference. If $n = 1$, equation (1-14) reduce to the following forward finite difference (see expression 1-4) scheme

$$Dxx[\tau_{xx}(x_i, z_j)] = \frac{\tau_{xx}(x_{i+1}, z_j) - \tau_{xx}(x_i, z_j)}{h}. \qquad (1\text{-}16)$$

So, I use the prefix `sf` to describe finite differences like expression (1-14) and the prefix `sb` to finite differences like expression (1-15).

```
% Elastic wave propagation
% Time: 2015/11/13
......
for it=1:Nt
 Vx=Vx+cp.*(sfLx(Txx,dx)+sbLz(Txz,dz));
 Vz=Vz+cp.*(sfLz(Tzz,dz)+sbLx(Txz,dx))+loc*wlet(it);

 temp1=sbLx(Vx,dx);
 temp2=sbLz(Vz,dz);
 Txx=Txx+c11.*temp1+c13.*temp2;
 Tzz=Tzz+c13.*temp1+c33.*temp2;
 Txz=Txz+c44.*sfLx(Vz,dx)+c44.*sfLz(Vx,dz);
end
.....
```

Now, we turn to the staggered grids in other research filed. Figure 1-4 illustrates five different grids, the Arakawa grid system used for Earth system models for meteorology and oceanography, designed by Arakawa and Lamb (1977). The Arakawa Staggered C-Grid is equivalent to staggered grid shown in 1-3, and the Arakawa Staggered B-Grid is called rotated staggered grid (Saenger et al., 2000) in seismic modeling. I have not found a efficient way to implement rotated staggered-grid finite difference (RSGFD) within MATLAB alone. In the next section, I'll show how to implement RSGFD with the help of CUDA. Table 1-2 lists some commonly-used staggered-grid finite difference operators.

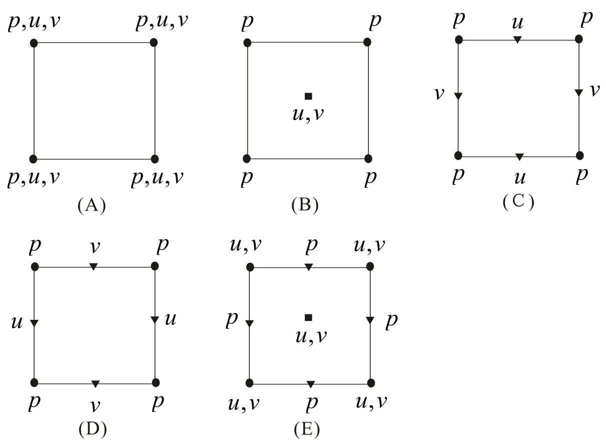


Figure 1-4 The five different grids by Arakawa (Purser and Leslie, 1988)

1.1.5 Rotated-staggered finite difference

The differential operators in rectangular coordinate system can be expressed by a linear combination of derivatives along the diagonal directions (Saenger et al., 2000)

$$\begin{aligned}\tilde{x} &= \frac{\Delta x}{\Delta r}x - \frac{\Delta z}{\Delta r}z \\ \tilde{z} &= \frac{\Delta x}{\Delta r}x + \frac{\Delta z}{\Delta r}z\end{aligned} \tag{1-17}$$

where $\tilde{x}$ and $\tilde{z}$ are the rotated space coordinates along diagonal directions, Δx and Δz are the spatial steps and $\Delta r = \sqrt{\Delta x^2 + \Delta z^2}$. Since the new derivative direction $\tilde{x}$ and $\tilde{z}$ replace the old direction x and z, it is necessary to express derivatives along old directions by a linear combination of derivative along the new directions

$$\begin{aligned}\frac{\partial}{\partial x} &= \frac{\Delta r}{2\Delta x}\left(\frac{\partial}{\partial \tilde{z}} + \frac{\partial}{\partial \tilde{x}}\right) \\ \frac{\partial}{\partial z} &= \frac{\Delta r}{2\Delta z}\left(\frac{\partial}{\partial \tilde{z}} - \frac{\partial}{\partial \tilde{x}}\right)\end{aligned} \tag{1-18}$$

Table 1-2 SGFD operations

Function Name	Description	Supplement
`sbLx`	12-order backward SGFD for the 1st-der.	`gfsbLx1_opt`
`sbLz`	12-order backward SGFD for the 1st-der.	`gfsbLz1_opt`
`sfLx`	12-order forward SGFD for the 1st-der.	`gfsfLx1_opt`
`sfLz`	12-order forward SGFD for the 1st-der.	`gfsfLz1_opt`
`gfsbLx1`	2N-order backward SGFD for the 1st-der.	
`gfsbLz1`	2N-order backward SGFD for the 1st-der.	
`gfsfLx1`	2N-order forward SGFD for the 1st-der.	
`gfsfLz1`	2N-order forward SGFD for the 1st-der.	
`gfsbLx1_opt`	12-order backward SGFD for the 1st-der.	optimized coef.
`gfsbLz1_opt`	12-order backward SGFD for the 1st-der.	optimized coef.
`gfsfLx1_opt`	12-order forward SGFD for the 1st-der.	optimized coef.
`gfsfLz1_opt`	12-order forward SGFD for the 1st-der.	optimized coef.
`gfsbLx1_3D_opt`	12-order backward SGFD for the 3D 1st-der.	optimized coef.
`gfsbLy1_3D_opt`	12-order backward SGFD for the 3D 1st-der.	optimized coef.
`gfsbLz1_3D_opt`	12-order backward SGFD for the 3D 1st-der.	optimized coef.
`gfsfLx1_3D_opt`	12-order forward SGFD for the 3D 1st-der.	optimized coef.
`gfsfLy1_3D_opt`	12-order forward SGFD for the 3D 1st-der.	optimized coef.
`gfsfLz1_3D_opt`	12-order forward SGFD for the 3D 1st-der.	optimized coef.
`rsgffd`	12-order forward rotated SGFD	CUDA required
`rsgbfd`	12-order backward rotated SGFD	CUDA required

Above equation indicates even if only one derivative in x or z direction we need, we have to calculate the both derivatives. The derivatives in $\tilde{x}$ and $\tilde{z}$ directions are (Saenger et al., 2000)

$$\begin{aligned}
\frac{\partial}{\partial \tilde{x}} &= \frac{1}{\Delta r}\sum_{m=1}^{M} a_m \left[u\left(x+\frac{(2m-1)}{2}\Delta x, z-\frac{(2m-1)}{2}\Delta z, t\right) - u\left(x-\frac{(2m-1)}{2}\Delta x, z+\frac{(2m-1)}{2}\Delta z, t\right)\right] \\
\frac{\partial}{\partial \tilde{z}} &= \frac{1}{\Delta r}\sum_{m=1}^{M} a_m \left[u\left(x+\frac{(2m-1)}{2}\Delta x, z+\frac{(2m-1)}{2}\Delta z, t\right) - u\left(x-\frac{(2m-1)}{2}\Delta x, z-\frac{(2m-1)}{2}\Delta z, t\right)\right]
\end{aligned} \tag{1-19}$$

where $a_m (m = 1, 2, ..., N)$ are the SGFD coefficients.

1.1.6 Numerical Dispersion

Spatial Dispersion

Numerical dispersion is one of crucial factors in evaluation of a numerical method. In physics, dispersion is the phenomenon in which the phase velocity of a wave depends on its frequency. Dispersion occurs when pure plane waves of different wavelengths have different propagation velocities, so that a wave packet of mixed wavelengths tends to spread out in space (like a tail, see figure 1-5).The discretization inherent in the finite-difference method results in the numerical dispersion of a propagating wave. figure 1-5 illustrates the numerical dispersion, the red line represents 4th-order central finite difference and the blue line represents 4th-order staggered-grid finite difference. Obviously, the staggered-grid finite-difference method provides more accurate result.

Numerical dispersion analysis is an useful tool to judge a numerical solution method. Before that we first briefly revisit the harmonic plane wave and some important relations. The plane waves being considered can be described by

$$P(x,t) = A_0 e^{2\pi i \frac{x-vt}{\lambda}} = A_0 e^{i(kx-\omega t)}, \tag{1-20}$$

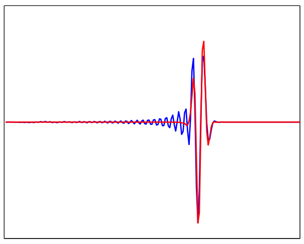


Figure 1-5 Comparison of numerical dispersion between CFD (blue) and SGFD (red)

where A_0 is the amplitude of the wave.
The velocity of a plane wave, v, is a function of the wave's wavelength λ:

$$v = v(\lambda).$$

The wave's velocity, wavelength, and frequency, f, are related by the identity

$$v(\lambda) = \lambda\, f(\lambda).$$

Dispersion relations are more commonly expressed in terms of the angular frequency $\omega = 2\pi f$ and wavenumber $k = 2\pi/\lambda$. Rewriting the relation above in these variables gives

$$\omega(k) = v(k)k \tag{1-21}$$

where we now view f as a function of k. Numerical dispersion involves temporal dispersion, spatial dispersion and anisotropy. Without loss of generality, we first consider the one dimension problem with acoustic wave equation,

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \tag{1-22}$$

by using controlling variable method. Now we assume the temporal derivative is solved accurately, only the spatial derivatives are solved by finite-difference method. Plugging the harmonic plane wave solution (1-20) into equation (1-22), gives

$$\left(v_n^2 k^2 =\right)\omega^2 = (cD_{xx}^c)^2 = \left(c\frac{2}{h}\sum_{m=1}^{n} c_m \sin(mhk)\right)^2 \approx c^2k^2 \tag{1-23}$$

for central finite difference, and

$$\left(v_n^2 k^2 =\right)\omega^2 = (cD_{xx}^s)^2 = \left(c\frac{2}{h}\sum_{m=1}^{n} s_m \sin\left(\frac{h}{2}(2m-1)k\right)\right)^2 \approx c^2k^2 \tag{1-24}$$

for staggered grid finite difference. Here v_n is the numerical velocity, c_m and s_m represent the central and staggered-grid finite difference coefficients, respectively. The spatial numerical dispersion can be expressed virtually as (see figure 1-6)

$$k \approx \frac{2}{h}\sum_{m=1}^{n} s_m \sin\left(\frac{h}{2}(2m-1)k\right), \tag{1-25}$$

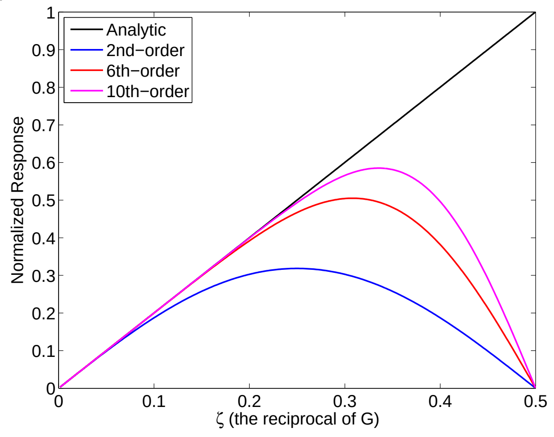
or the ratio of the numerical wavenumber to the exact wavenumber (relative wavenumber),

$$R_k = \frac{2}{kh}\sum_{m=1}^{n} s_m \sin\left(\frac{h}{2}(2m-1)k\right) = \sum_{m=1}^{n} s_m \frac{\sin(\pi\zeta(2m-1))}{\pi\zeta}, \tag{1-26}$$

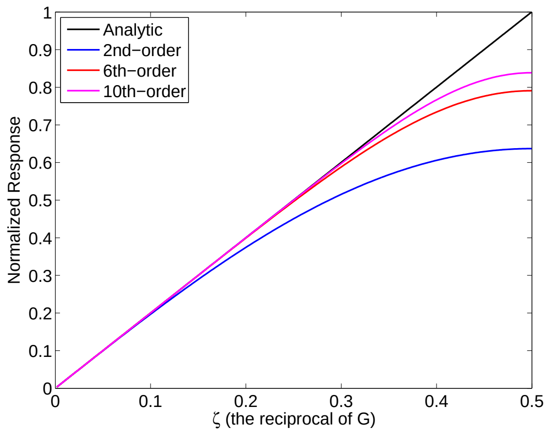
where $\zeta = h/\lambda$ represents the reciprocal of the number of nodal points (G) per wavelength, or the ratio of the numerical velocity to the true velocity (relative velocity, see figure 1-7),

$$R_{vel} = \frac{v_n}{c}\left(\frac{\omega/k}{c}\right) = \frac{2}{kh}\sum_{m=1}^{n} s_m \sin\left(\frac{h}{2}(2m-1)\,k\right) = \sum_{m=1}^{n} s_m \frac{\sin(\pi\zeta(2m-1))}{\pi\zeta}. \tag{1-27}$$

As shown in figure 1-6 and 1-7, using a central FD scheme to approximate the first-order derivatives leads to numerical dispersion in the high wavenumber region. The convergence of a conventional central FD is very slow due to the singularity existing at the Nyquist wavenumber($\zeta = 0.5$). Staggered-grid scheme appears to provide more accurate results. However, such a scheme requires interpolating the model parameters onto a different computational grid, which causes numerical inaccuracy. Zhang et al. (2011) points out that, for higher accuracy, staggered-grid methods usually assume uniform sampling in all three spatial dimensions, which reduces the flexibility of their application and increases computational cost. But I haven't heard a similar point of view, I still prefer to use the staggered-grid difference difference. The SGFD dispersion curve looks like the first half of the CFD dispersion curve with a stretched axis, $\zeta' = \zeta/2$ (PhD Zhao told me this). In a word, whichever scheme we choose, the numerical velocities are both slower than the exact velocity, and the error grows as nodal points per wavelength decreases.

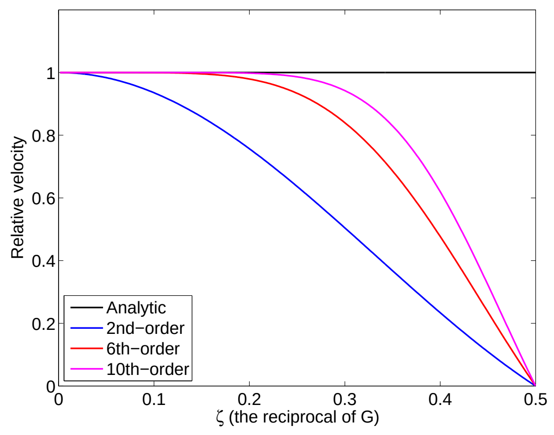


(a) Dispersion curves for CFD

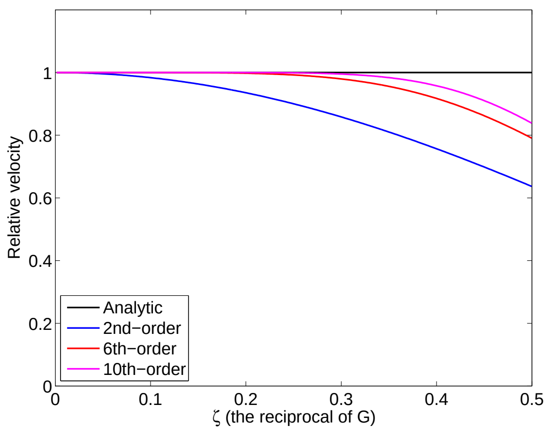


(b) Dispersion curves for SGFD

Figure 1-6 Dispersion curves for two finite difference schemes



(a) Dispersion curves for CFD



(b) Dispersion curves for SGFD

Figure 1-7 Dispersion curves for two finite difference schemes

Aliasing and dispersion

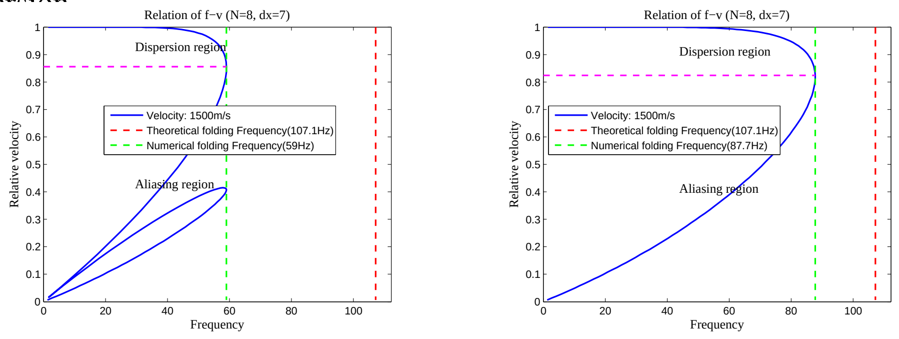
Sampling theorem tells us that, in theory, if we want to perfectly reconstruct the original continuous signal of finite bandwidth from the samples, the sampling frequency f_s is necessarily twice larger than the bandlimit f_m, i.e.

$$f_s \geqslant 2f_m, \tag{1-28}$$

and $f_s/2$ is called Nyquist frequency. Any frequency component above $f_s/2$ is indistinguishable from a lower-frequency component, called an *alias*,associated with one of the copies. Unfortunately, the Nyquist frequency connected to finite difference method is smaller than theoretical value, unless the order is infinite. Next, the aliasing is investigated from the dispersion relation. From equation 1-24, we obtain the $f-k$ dispersion relation,

$$f = \frac{c}{\pi h}\sum_{m=1}^{n} s_m \sin\left(\frac{h}{2}(2m-1)\,k\right), \tag{1-29}$$

which allows us to calculate the frequency-depended $v(f)$ by using $v(f) = 2\pi f/k$. we can use $f - v(f)$ curves to depict aliasing and dispersion, which can help us to determine grid parameters for a given bandlimit source or implement an anti-aliasing filtering. For example, if the spatial steps and the order of the finite difference are fixed, low pass filtering is usually carried out on source (or seismic record) to attenuate the high frequency components to avoid aliasing and dispersion. Figure 1-9 illustrate the affect of source frequency and grid step to numerical dispersion. Obviously, high frequency, large grid step and low velocity result in heavy disp



(a) Aliasing and dispersion curves for CFD (b) Aliasing and dispersion curves for SGFD

Figure 1-8 Aliasing and dispersion curves for two finite difference schemes

Time Dispersion

Now, we go further to the time dispersion. Plugging the harmonic plane wave solution (1-20) into equation (1-22) and assuming the spatial derivatives are solved accurately, gives

$$\frac{4}{\Delta t^2}\sin^2\left(\frac{\omega\Delta t}{2}\right) = c^2k^2 \approx \omega^2 \left(= v_n^2k^2\right) \tag{1-30}$$

for 2nd-order time finite difference scheme. Then

$$\frac{1}{R_t} = \frac{c}{v_n} = \frac{1}{\pi\Delta t/T}\sin\left(\pi\Delta t/T\right). \tag{1-31}$$

Because of restriction of stability condition, the time ratio Gt=$\Delta t/T$ is usually far smaller than 0.5, therefore time dispersion results in negligible inaccuracy (at least I think so). But in order to improve computational efficiency, a series of methods have been proposed to improve time finite difference accuracy for a large time step (e.g. Firouzi et al., 2012; Fang et al., 2014).

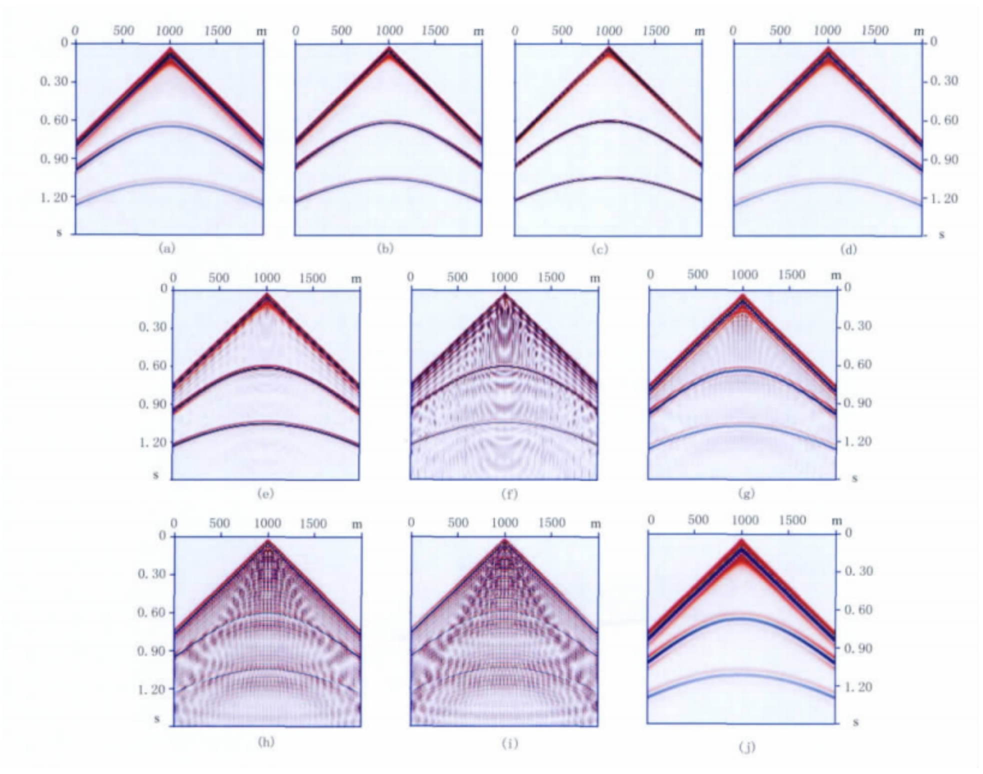


Figure 1-9 Snapshots for three layers model: (a), (b) and (c) for 15, 25 and 35Hz Ricker wavelet on 5m grid; (d), (e) and (f) for 15, 25 and 35Hz Ricker wavelet on 10m grid; (g), (h), (i) and (j) for 15, 25, 35 and 10Hz Ricker wavelet on 20m grid.(see Guo and Wu, 2012)

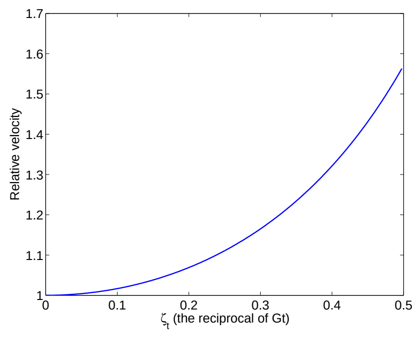


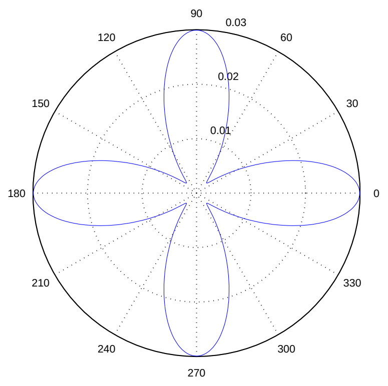
Figure 1-10 Dispersion curves for 2nd-order time finite difference approximation

Dispersion Anisotropy

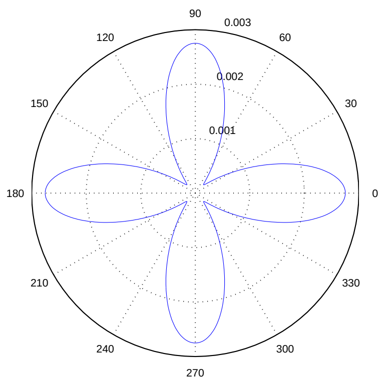
Numerical dispersion anisotropy means that the relative phase velocity errors are different as the propagation direction changes. In two dimension, the relative phase velocity error reads as

$$E_{vel} = R_{vel} - 1 = \frac{v_n}{c} - 1 = \sqrt{\left(\sum_{m=1}^{n} s_m \frac{\sin(\pi\zeta_x(2m-1))}{\pi\zeta}\right)^2 + \left(\sum_{m=1}^{n} s_m \frac{\sin(\pi\zeta_z(2m-1))}{\pi\zeta}\right)^2} - 1, \tag{1-32}$$

where $\zeta^2 = \zeta_x^2 + \zeta_z^2$. Some methods (I has read a related paper) have been proposed to suppress this numerical dispersion anisotropy. But I think it's absolutely unnecessary. Because the largest relative phase velocity errors appear at $x-$ and $z-$ axis (see figure 1-11), the methods to reduce numerical dispersion for 1D problem (propagation direction is zero or ninety degree) suppress this anisotropy phenomenon, implicitly.



(a) Dispersion curves for CFD



(b) Dispersion curves for SGFD

Figure 1-11 Dispersion curves for two finite difference schemes

1.1.7 Stability analysis

The stability of numerical schemes is closely associated with numerical error. A finite difference scheme is stable if and only if the errors made at one time step of the calculation do not cause the errors to increase as the computations are continued. If the errors decay and eventually damp out, the numerical scheme is said to be stable. If, on the contrary, the errors grow with time the numerical scheme is said to be unstable. The stability of numerical schemes can be investigated by performing von Neumann stability analysis. In numerical analysis, von Neumann stability analysis (also known as Fourier stability analysis) is a procedure used to check the stability of finite difference schemes as applied to linear partial differential equations. The analysis is based on the Fourier decomposition of numerical error. In certain cases, von Neumann stability is necessary and sufficient for stability in the sense of case: the PDE and the finite difference scheme models are linear. I take the first-order acoustic equation as an example to illustrate the stability analysis via von Neumann method. Recall the 2D velocity-stress equation

$$\frac{\partial \mathbf{Q}}{\partial t} = \mathbf{A}\mathbf{Q} \tag{1-33}$$

where $\mathbf{Q} = (p, v_x, x_z)$, and $\mathbf{A}$ is the differential matrix, represented as

$$\mathbf{A} = -\frac{1}{\rho}\begin{pmatrix} 0 & \rho^2 v^2 \frac{\partial}{\partial x} & \rho^2 v^2 \frac{\partial}{\partial z} \\ \frac{\partial}{\partial x} & 0 & 0 \\ \frac{\partial}{\partial z} & 0 & 0 \end{pmatrix}. \tag{1-34}$$

Taking the derivative of equation (1-33) with respect to t, gives

$$\frac{\partial^2 \mathbf{Q}}{\partial t^2} = \mathbf{A}^2 \mathbf{Q} \tag{1-35}$$

with

$$\mathbf{A^2} = v^2 \begin{pmatrix} \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2} & 0 & 0 \\ 0 & \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial x \partial z} \\ 0 & \frac{\partial^2}{\partial x \partial z} & \frac{\partial^2}{\partial z^2} \end{pmatrix}. \tag{1-36}$$

Replacing the time derivative of equation 1-35 with a second-order FD scheme, and applying spatial and temporal forward Fourier transform to wavefield $\mathbf{Q}$, we obtain

$$\left[\frac{4}{\Delta t^2}\sin^2(\frac{\omega \Delta t}{2})\mathbf{I} - \tilde{\mathbf{A}}^2\right]\tilde{\mathbf{Q}} = \mathbf{0}, \tag{1-37}$$

where $\tilde{\mathbf{Q}} = \iiint \mathbf{Q}(t, x, z) e^{-j(\omega t + k_x x + k_z z)} dt dx dz$, $\mathbf{I}$ is is identity matrix, and

$$\tilde{\mathbf{A}}^{\mathbf{2}} = v^2 \begin{pmatrix} k_x^2 + k_z^2 & 0 & 0 \\ 0 & k_x^2 & k_x k_z \\ 0 & k_x k_z & k_z^2 \end{pmatrix}. \tag{1-38}$$

Since equation 1-35 is hyperbolic equation system, there exists a nonsingular matrix $\mathbf{S}$ such that $\tilde{\mathbf{A}}^2 = \mathbf{S}^{-1}\Lambda\mathbf{S}$, where Λ is a positive diagonal matrix composed of eigenvalues λ_i of matrix $\tilde{\mathbf{A}}^{\mathbf{2}}$ (Dong et al., 2000), and then

$$\mathbf{S}^{-1}\left[\frac{4}{\Delta t^2}\sin^2(\frac{\omega \Delta t}{2})\mathbf{I} - \Lambda\right]\mathbf{S}\tilde{\mathbf{Q}} = \mathbf{0} \tag{1-39}$$

Letting $\mathbf{G} = \mathbf{S}^{-1}\mathbf{T}\mathbf{S} = \mathbf{S}^{-1}\left[\frac{4}{\Delta t^2}\sin^2(\frac{\omega \Delta t}{2})\mathbf{I} - \Lambda\right]\mathbf{S}$, then $\det(\mathbf{G}) = \det(\mathbf{T})$. If equation (1-39) has non-zero solution, the following expression must be satisfied

$$\left[\frac{4}{\Delta t^2}\sin^2(\frac{\omega \Delta t}{2}) - \lambda_i\right] = 0 \tag{1-40}$$

Since $0 \leqslant \sin^2(\frac{\omega \Delta t}{2}) \leqslant 1$, we obtain the stability condition as

$$0 \leqslant \frac{\Delta t^2}{4}\lambda_i \leqslant 1 \tag{1-41}$$

The eigenvalues of matrix $\tilde{\mathbf{A}}^2$ are $\lambda_1 = \lambda_2 = v^2\left(k_x^2 + k_z^2\right)$ and $\lambda_3 = 0$. Now, we extend the above analysis to three dimension, we obtain

$$\frac{\Delta t^2 v^2}{4}\left(k_x^2 + k_y^2 + k_z^2\right) \leqslant 1 \tag{1-42}$$

Substituting k_i with finite difference approximation 1-25, and using $c\sin(x) \leqslant |c|$, gives

$$\Delta t V\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}} \leqslant \frac{1}{\sum\limits_{m=1}^{n} |c_m|}, \tag{1-43}$$

for central finite difference scheme, where c_m are central finite difference coefficients, and

$$\Delta t V\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}} \leqslant \frac{1}{\sum\limits_{m=1}^{n} |s_m|}, \tag{1-44}$$

for staggered-grid finite difference scheme, where s_m are staggered-grid finite difference coefficients. Next, I introduce a simpler procedure (essentially same as the von Neumann method) to evaluate the stability condition. From the harmonic plane wave 1-20, we can derive

$$P^{n+1} + P^{n-1} = 2\cos\left(\omega\Delta t\right)P^n. \tag{1-45}$$

There is no approximation made so far. Equation (1-45) is unconditionally stable. The second-order time stepping scheme is the second-order truncation to equation (1-45) after expanding the cosine function term using the Taylor series, i.e.

$$P^{n+1} + P^{n-1} = 2\left[1 - \frac{(\omega\Delta t)^2}{2}\right]P^n. \tag{1-46}$$

To make it stable, $\left[1 - \frac{(\omega\Delta t)^2}{2}\right]$ must be bounded in $[-1, 1]$, Equivalently, the following condition must be satisfied

$$\omega\Delta t \leqslant 2 \tag{1-47}$$

Inserting the following dispersion relation

$$\omega^2 = c^2(D_{xx}^2 + D_{yy}^2 + D_{zz}^2) \tag{1-48}$$

into above inequality gives the same results shown in equation (1-43) and (1-44).

Bibliography

Arakawa, A., and V. R. Lamb, 1977, Computational design of the basic dynamical processes of the ucla general circulation model: Methods in computational physics, **17**, 173–265.

Dong, L., Z. MA, and J. Cao, 2000, Stability of the staggered-grid high-order difference method for first-order elastic wave equation: Chinese Journal of Geophysics, **43**, 904–913.

Fang, G., S. Fomel, Q. Du, and J. Hu, 2014, Lowrank seismic-wave extrapolation on a staggered grid: Geophysics, **79**, T157–T168.

Firouzi, K., B. Cox, B. Treeby, and N. Saffari, 2012, A first-order k-space model for elastic wave propagation in heterogeneous media: The Journal of the Acoustical Society of America, **132**, 1271–1283.

Guo, N., and G. Wu, 2012, High-order finite difference method in reverse-time migration with variable grids based on pml boundary condtion: Oil geophysical prospecting, **47**, 256–265.

LeVeque, R. J., 2007, Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems: Siam, **98**.

Liu, Y., 2014, Optimal staggered-grid finite-difference schemes based on least-squares for wave equation modelling: Geophysical Journal International, ggu032.

Madariaga, R., 1976, Dynamics of an expanding circular fault: Bulletin of the Seismological Society of America, **66**, 639–666.

Purser, R., and L. Leslie, 1988, A semi-implicit, semi-lagrangian finite-difference scheme using hligh-order spatial differencing on a nonstaggered grid: Monthly Weather Review, **116**, 2069–2080.

Saenger, E. H., N. Gold, and S. A. Shapiro, 2000, Modeling the propagation of elastic waves using a modified finite-difference grid: Wave motion, **31**, 77–92.

Virieux, J., 1984, Sh-wave propagation in heterogeneous media: velocity-stress finite-difference method: Geophysics, **49**, 1933–1942.

———, 1986, P-sv wave propagation in heterogeneous media: Velocity-stress finite-difference method: Geophysics, **51**, 889–901.

Zhang, Y., H. Zhang, and G. Zhang, 2011, A stable tti reverse time migration and its implementation: Geophysics, **76**, WA3–WA11.