

HPC Summer School, Day 1: Finite Difference Modeling and Reverse Time Migration

William W. Symes

Rice University

Agenda

Wave physics, wave equations, and waves

Finite difference methods for the wave equation: accuracy, stability, dispersion

`wave.c`: a simple finite difference modeling application

Imaging with waves: reverse time migration

`rtm.c`: a simple reverse time migration application

An exercise: decreasing dispersion by increasing order

Basic physics of linear waves

Acoustics

- ▶ position $\mathbf{x} = (z, x, y)$, time t
- ▶ dynamic fields: pressure $p(\mathbf{x}, t)$, particle velocity $\mathbf{v}(\mathbf{x}, t)$
- ▶ material parameters: material density $\rho(\mathbf{x})$, bulk modulus $\kappa(\mathbf{x})$
- ▶ energy source: $f_c(\mathbf{x}, t)$ (constitutive law defect)

linked by

$$\rho(\mathbf{x}) \frac{\partial \mathbf{v}}{\partial t}(\mathbf{x}, t) = -\nabla p(\mathbf{x}, t)$$

(Newton's law) and

$$\frac{1}{\kappa(\mathbf{x})} \frac{\partial p}{\partial t}(\mathbf{x}, t) = -\nabla \cdot \mathbf{v}(\mathbf{x}, t) + f_c(\mathbf{x}, t)$$

(constitutive law)

Basic physics of linear waves

acoustics = special case of

- ▶ gas dynamics (sound waves in gases and fluids)
- ▶ linear elastodynamics:

$$\rho \frac{\partial \mathbf{v}}{\partial t} = \nabla \cdot \mathbf{s}, \quad \frac{\partial \mathbf{s}}{\partial t} = \mathbf{C} :: \mathbf{v}$$

\mathbf{s} = stress, \mathbf{C} = Hooke tensor (+ energy source representation)

- ▶ Elastodynamics more accurate rep'n of Earth dynamics
- ▶ but most industrial seismic processing based on acoustic model
- ▶ recent interest in quasiacoustic anisotropic approximations to elastic p-waves

Basic physics of linear waves

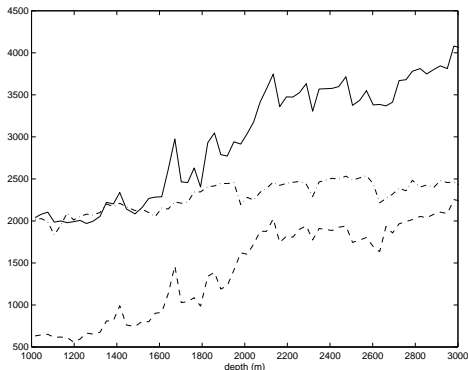
Second order form:

- ▶ differentiate const. law wrt t
- ▶ introduce $c = \sqrt{\kappa/\rho}$ (compressional (p-)wavevelocity)
- ▶ $f = \frac{\partial f_c}{\partial t}$
- ▶ eliminate \mathbf{v}

$$\frac{1}{\rho c^2} \frac{\partial^2 p}{\partial t^2} - \nabla \cdot \frac{1}{\rho} \nabla p = f$$

Sedimentary rocks, reflection geometry

Acoustics properties - log, North Sea (thanks Mobil - Keys & Foster 1998)



p-wave velocity (top curve, m/s) varies substantially, density (middle curve, gm/cm³) not so much, dominant variation *vertical*

⇒ *constant density* approximation

Sedimentary rocks, reflection geometry

Constant density acoustics:

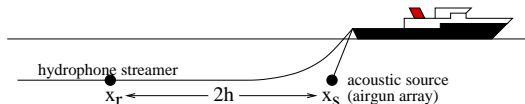
$$\frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} - \nabla^2 p = \rho f$$

- ▶ simplest to understand - spatial operator is spatially homogeneous
- ▶ basis for most processing
- ▶ increasingly popular: anisotropic generalizations

Sedimentary rocks, reflection geometry

Geometry of seismic reflection experiment:

- ▶ sources highly localize wrt other scales - characterized by *source position* \mathbf{x}_s
- ▶ receivers (typically *antennae*) highly localized, *receiver position* \mathbf{x}_r
- ▶ source, receiver depths near-constant over survey - sample *depth plane*
- ▶ source-receiver distance = *offset* = $2h$
- ▶ marine surveys: source, receiver characteristics *highly reproducible* (eg. Dragoset et al. 1987)



Sedimentary rocks, reflection geometry

Modeling consequences:

- ▶ simplest source representation: *point radiator*
 $f(\mathbf{x}, t) = w(t)\delta(\mathbf{x} - \mathbf{x}_s)$, $w(t) = \text{source wavelet}$
- ▶ receiver representation: dual to source - *point sampling*
- ▶ seismic trace at \mathbf{x}_r , \mathbf{x}_s , $t = p(\mathbf{x}_r, t)$ with
 $f(\mathbf{x}, t) = w(t)\delta(\mathbf{x} - \mathbf{x}_s)$
- ▶ experiment is *causal*: $p = 0$, $t < t_{\text{src}}$ - homogeneous initial condition
- ▶ Caveat emptor: in reality, both sources and receivers are antennae with nontrivial *radiation pattern*

Sedimentary rocks, reflection geometry

Simple solutions, homogeneous infinite space ($c = \text{const.}$):

- ▶ 3D: outgoing spherical wave

$$p(\mathbf{x}, t) = \frac{f\left(t - \frac{r}{c}\right)}{4\pi r}, \quad r = |\mathbf{x} - \mathbf{x}_s|$$

- ▶ 2D: Poisson's formula

$$p(\mathbf{x}, t) = \frac{1}{\pi c^4} \int_0^{\sqrt{t-r/c}} \frac{f'(t - \tau^2 - r/c)}{\sqrt{\tau^2 + 2r/c}} d\tau$$

(looks like outgoing circular wave)

Sedimentary rocks, reflection geometry

Boundary conditions:

- ▶ Physical boundary - sea surface is nearly pressure-free (water/air impedance contrast very large), so acts like perfect reflector: $p = 0$ [land surface - not so perfect, also elastic effects may be nonnegligible...]
- ▶ on wavelength scale ($30 \text{ Hz} \quad 1500 \text{ m/s} = 50 \text{ m}$) sea surface often roughly flat - $z = 0$
- ▶ typical simulation domain = box
 $0 \leq z \leq z_{\max}, x_{\min} \leq x \leq x_{\max}, \text{ sim for } y$
- ▶ other boundaries are *artificial*, should (mostly) absorb waves - large literature on modeling of *absorbing boundary conditions*

Sedimentary rocks, reflection geometry

Boundary conditions, cont'd: Simple equivalence of free surface:
method of images - if $p_{\text{full}}(\mathbf{x}, t)$ is full space solution, then solution in half-space $z \geq 0$ with $p(z = 0) = 0$ is

$$p(\mathbf{x}, t) = p_{\text{full}}(\mathbf{x}, t) - p_{\text{img}}(\mathbf{x}, t)$$

image or *free surface ghost* solution

$$p_{\text{img}}(x, y, z, t) = p_{\text{full}}(x, y, -z, t)$$

Leads to directional tuning of amplitudes even for point radiator

DEMO test1:wavehom.showmovie

Agenda

Wave physics, wave equations, and waves

Finite difference methods for the wave equation: accuracy, stability, dispersion

`wave.c`: a simple finite difference modeling application

Imaging with waves: reverse time migration

`rtm.c`: a simple reverse time migration application

An exercise: decreasing dispersion by increasing order

Critical references for finite difference modeling:

- ▶ Moczo et al., *Advances in Geophysics* **48** (2006), pp. 421-516
- ▶ *Geophysics*, special issue on modeling, Sept-Oct 2007
- ▶ G. Cohen, *Higher order numerical methods for transient wave equations*, Springer, 2002
- ▶ R. Leveque, *Finite difference methods for ordinary and partial differential equations*, SIAM, 2007

Basic concept

- ▶ *replace* continuum fields ($p(\mathbf{x}, t), \dots$) with sampled discrete fields

$$p(\mathbf{x}, t) \rightarrow p_{n,m,l}^k \simeq p(n\Delta z, m\Delta x, l\Delta y, k\Delta t)$$

- ▶ *replace* derivatives by divided differences:

$$\frac{\partial p}{\partial z} \rightarrow \frac{p_{n+1,m,l}^k - p_{n-1,m,l}^k}{2\Delta z}, \quad \frac{\partial^2 p}{\partial z^2} \rightarrow \frac{p_{n+1,m,l}^k + p_{n-1,m,l}^k - 2p_{n,m,l}^k}{\Delta z^2},$$

and so on.

- ▶ solve resulting system of algebraic equations for $p_{n,m,l}^k$

Basic concept

Time to 'fess up: this course deals (explicitly) only with 2D problems, so

$$p(\mathbf{x}, t) \rightarrow p_{n,m}^k \simeq p(n\Delta z, m\Delta x, k\Delta t)$$

Everything carries over nearly without change to 3D - with a lot more flops, bytes!

Truncation error

Refers to difference between derivative and divided difference (“finite difference”, FD) formulae, and ranks FD operators by asymptotic *order of accuracy*

Based on Taylor’s formula:

$$p(x + \Delta x) = p(x) + \frac{\partial p}{\partial x}(x)\Delta x + \frac{1}{2} \frac{\partial^2 p}{\partial x^2} + \dots$$

so

$$\frac{\partial p}{\partial x}(x) = \frac{p(x + \Delta x) - p(x)}{\Delta x} + e(x, \Delta x),$$

$$e(x, \Delta x) = \frac{1}{2} \frac{\partial^2 p}{\partial x^2}(x)\Delta x + O(\Delta x^2)$$

where “...” represents terms from rest of Taylor series, all having at least two factors of Δx - *first order accurate*

Truncation error

Also true that

$$\frac{\partial p}{\partial x}(x) = -\frac{p(x - \Delta x) - p(x)}{\Delta x} + e(x, \Delta x),$$

$$e(x, \Delta x) = -\frac{1}{2} \frac{\partial^2 p}{\partial x^2}(x) \Delta x + O(\Delta x^2)$$

average these to get

$$\begin{aligned} \frac{\partial p}{\partial x}(x) &= \frac{1}{2} \left(\frac{p(x + \Delta x) - p(x)}{\Delta x} - \frac{p(x - \Delta x) - p(x)}{\Delta x} \right) + e(x, \Delta x) \\ &= \frac{p(x + \Delta x) - p(x - \Delta x)}{2\Delta x} + e(x, \Delta x) \end{aligned}$$

= centered difference formula for first derivative - 2nd order accurate, $e = O(\Delta x^2)$

Truncation error

Similar story for 2nd derivative: subtract the two first order first derivative formulas on the last two slides, divide by Δx to get **standard centered 2nd order operator**

$$\frac{\partial^2 p}{\partial x^2}(x) = \frac{p(x + \Delta x) - 2p(x) + p(x - \Delta x)}{\Delta x^2} + e(x, \Delta x)$$

$$e(x, \Delta x) = O(\Delta x^2)$$

e depends on four term Taylor series w/ remainder - leading term proportional to $\partial^4 p / \partial x^4$.

Truncation error

More arithmetic, using six term Taylor series \Rightarrow standard centered
4th order operator

$$\frac{\partial^2 p}{\partial x^2}(x) = \frac{-\frac{1}{12}p(x + 2\Delta x) + \frac{4}{3}p(x + \Delta x) - \frac{5}{2}p(x) + \frac{4}{3}p(x - \Delta x) - \frac{1}{12}p(x - 2\Delta x)}{\Delta x^2}$$
$$e(x, \Delta x) = O(\Delta x^4)$$

Leading term in e proportional to $\partial^6 p / \partial x^6$

Truncation error

Can concoct difference formulas of arbitrary order of accuracy - standard formulas shown so far use minimal number of points for given accuracy (see Moczo et al 06, Cohen 02)

What difference does order make?

error formulae depend on attributes (Taylor series) of function being approximated ($p(\mathbf{x}, t)$)

\Rightarrow 2nd order *always* beats 1st order for *small enough* Δx - more generally, higher order *always* beats lower order for *small enough* Δx

Caveat: derivatives up to the appropriate order must exist... more on this later

Standard 2nd order scheme

- ▶ compute only points on $(\Delta z, \Delta x, \Delta t)$ grid - use notation $p_{n,m}^k \simeq p(n\Delta z, m\Delta x, k\Delta t)$
- ▶ substitute standard 2nd order centered difference formula for 2nd derivatives in const density acoustic wave equation

$$\frac{p_{n,m}^{k+1} - 2p_{n,m}^k + p_{n,m}^{k-1}}{\Delta t^2} =$$
$$(c^2)_{n,m} \left(\frac{p_{n+1,m}^k - 2p_{n,m}^k + p_{n-1,m}^k}{\Delta z^2} + \right.$$
$$\left. \frac{p_{n,m-1}^k - 2p_{n,m}^k + p_{n,m+1}^k}{\Delta x^2} + w^k \delta_{n,m} \right)$$

Standard 2nd order scheme

Here δ is discrete approximation to Dirac delta function: if $z_s = n_s \Delta z, x_s = m_s \Delta x$, then

$$\delta_{n,m} = \begin{cases} \frac{1}{\Delta z \Delta x}, & n = n_s, m = m_s \\ 0, & \text{else} \end{cases}$$

$$w^k = w(k \Delta t)$$

$$(c^2)_{n,m} = c^2(n \Delta z, m \Delta x)$$

Standard 2nd order scheme

Formula to be used to advance p in t - clear denominators, rearrange:

$$p_{n,m}^{k+1} = 2p_{n,m}^k - p_{n,m}^{k-1} + (c^2)_{n,m} [r_z(p_{n+1,m}^k + p_{n-1,m}^k) + r_x(p_{n,m-1}^k + p_{n,m+1}^k) + sp_{n,m}^k + \Delta t^2 w^k \delta_{n,m}]$$

$$r_z = \frac{\Delta t^2}{\Delta z^2}, r_x = \frac{\Delta t^2}{\Delta x^2}, s = -2 \left(\frac{\Delta t^2}{\Delta z^2} + \frac{\Delta t^2}{\Delta x^2} \right)$$

RHS involves same sum of five neighboring p -values at every (n, m) - **standard five-point stencil**. Provides explicit computation of field update $p_{n,m}^k$: **explicit scheme**.

Stability and Convergence

Truncation error e of scheme = difference between LHS, RHS
when $p_{n,m}^k = p(n\Delta z, m\Delta x, k\Delta t)$

Will use *one-parameter family of grids*: $\Delta z = a_z \Delta t$, $\Delta x = a_x \Delta t$
with fixed a_z, a_x (reason for this to come)

Main Theorem: provided **stability** condition (to be discussed next)
holds, if $e = O(\Delta t^p)$, $p > 0$ then
 $p(n\Delta z, m\Delta x, k\Delta t) - p_{n,m}^k = O(\Delta t^p)$

Role of truncation error: to estimate how well FD solution
approximates wave equation solution, compute how well wave
equation solution satisfies FD equation!

Stability and Convergence

Scheme is **stable** if the size of the solution to an inhomogeneous FD problem (with $\text{LHS} = \text{RHS} + \text{inhom. term}$) is proportional to the size of the inhomogeneous term - const. of proportionality indep. of choice of inhom. term for fixed simulation time in *physical* units, i.e. $0 \leq k\Delta t \leq t_{\max}$.

Why this forces convergence: difference $p(n\Delta z, m\Delta x, k\Delta t) - p_{n,m}^k$ solves inhom. problem with $\text{LHS} = \text{RHS} + e$, i.e. inhom. term = truncation error. If scheme is stable, then $p(n\Delta z, m\Delta x, k\Delta t) - p_{n,m}^k$ has to go to zero with Δt at the same rate as e .

Generally, explicit schemes are stable $\Leftrightarrow \Delta t < \text{const.} \max(\Delta z, \Delta x)$. For nD 2nd order centered scheme, $\text{const.} = 1/(c_{\max} \sqrt{n})$ ("**CFL condition**"). Violation \Rightarrow rapid overflow.

Dispersion

For $v = \text{const.}$, both wave equation and FD scheme have *plane wave* solutions

$$p(z, x, t) = e^{i(\zeta z + \xi x + \omega t)}$$

(equations are real, so both real and imag parts of complex solution are solutions!)

Dispersion relation = necessary relation between ζ, ξ, ω for above to be solution

Phase velocity of plane wave $v_{\text{ph}} = \frac{\omega}{\sqrt{\zeta^2 + \xi^2}}$

Exercise: for plane wave solution of wave equation, $v_{\text{ph}} = c$ - independent of ω , **nondispersive**

Dispersion

plane wave solutions of FD scheme:

$$p_{n,m}^k = e^{i(n\zeta\Delta z + m\xi\Delta x + k\omega\Delta t)}$$

phase velocity v_{FD} generally depends on ζ, ξ, ω and $\neq c$ - **dispersive**

Dispersion worse for small **gridpoints per wavelength**

$G = 1/\max(2\pi\zeta\Delta z, 2\pi\xi\Delta x)$, hence for larger ω .

Rules of thumb (incomplete!!!): to keep phase error $\leq 5\%$ over 100 wavelengths, need $G \geq 10$ for 2nd order scheme. Higher order schemes typically exhibit less dispersion and higher accuracy for given G .

See references for much more info. **DEMO**
test2:wavehom.showmovie

Agenda

Wave physics, wave equations, and waves

Finite difference methods for the wave equation: accuracy, stability, dispersion

`wave.c`: a simple finite difference modeling application

Imaging with waves: reverse time migration

`rtm.c`: a simple reverse time migration application

An exercise: decreasing dispersion by increasing order

Implementing the 2nd order scheme

$$p_{n,m}^{k+1} = 2p_{n,m}^k - p_{n,m}^{k-1} +$$

$$(c^2)_{n,m}[r_z(p_{n+1,m}^k + p_{n-1,m}^k) + r_x(p_{n,m-1}^k + p_{n,m+1}^k) + sp_{n,m}^k + \Delta t^2 w^k \delta_{n,m}]$$

Observations:

- ▶ full 3D space-time field $p_{n,m}^t$ not needed for either trace recording (as in physical world!) or for making movies of wavefield - only need single time level (k)
- ▶ to perform update formula, need three time levels ($k-1, k, k+1$)
- ▶ index ranges: $0 \leq n < n_{\max}$ etc. If $p = 0$ at boundary points $n = 0, \dots$ and only $n = 1, \dots, n_{\max} - 2$ are updated, then pressure-free boundary condition automatically maintained (we are not trying for absorbing boundary conditions here!)

Implementing the 2nd order scheme

First pass implementation: three arrays p^0 (previous k), p^1 (current k), p^2 (next k)

For $k = 0, \dots, k_{\max} - 1$

For $m = 1, \dots, m_{\max} - 2$

For $n = 1, \dots, n_{\max} - 2$

$$p_{n,m}^2 = 2p_{n,m}^1 - p_{n,m}^0 +$$

$$(c^2)_{n,m} [r_z(p_{n+1,m}^1 + p_{n-1,m}^1) + r_x(p_{n,m-1}^1 + p_{n,m+1}^1) + sp_{n,m}^1 + \Delta t^2 w^k \delta_{n,m}]$$

[record p^1 - either trace samples or movie frame]

[overwrite $p^1 \rightarrow p^0$, $p^2 \rightarrow p^1$]

For $m = 1, \dots, m_{\max} - 2$

For $n = 1, \dots, n_{\max} - 2$

$$p_{n,m}^0 = p_{n,m}^1, p_{n,m}^1 = p_{n,m}^2$$

[next k]

Implementing the 2nd order scheme

Critique:

- ▶ update loop: each value in p^0 used *once* on RHS of scheme (not assigned to, no dependence across loop)
- ▶ overwrite loop: lots of data motion not implicit in math

Waste of both storage and data motion. Classic fix - takes advantage of assignment syntax, access to addresses

- ▶ use only *two* arrays p^0, p^1
- ▶ replace LHS with assignment to p^0
- ▶ swap addresses of p^0, p^1 rather than copy values

Implementing the 2nd order scheme

wave.c coded in C - switch to C syntax:

- ▶ $\Delta z \rightarrow dz, \Delta x \rightarrow dx, \Delta t \rightarrow dt$
- ▶ $n \rightarrow iz, m \rightarrow ix, k \rightarrow it$
- ▶ $n_{\max} \rightarrow nz, m_{\max} \rightarrow nx, k_{\max} \rightarrow nt$
- ▶ $p^0 \rightarrow p0, p^1 \rightarrow p1, (c^2) \rightarrow v$ (only square is used!)

Implementing the 2nd order scheme

hpcss/src/step.c:step_forward - implements update operator

```
for (ix=1;ix<nx-1;ix++) {  
    for (iz=1;iz<nz-1;iz++) {  
        ioff=iz+ix*nz;  
        p0[ioff]=two*p1[ioff]-p0[ioff]  
            +v[ioff]*(rz*(p1[ioff+1]+p1[ioff-1]) +  
            rx*(p1[ioff+nz]+p1[ioff-nz]) -  
            s*p1[ioff]);  
    }  
}
```

Then swap pointers (in hpcss/main/wave.c, after call to step_forward):

```
tmp=p0; p0=p1; p1=tmp;
```

Implementing the 2nd order scheme

Source: C notation $n_s \rightarrow \text{isz}$, $m_s \rightarrow \text{isx}$

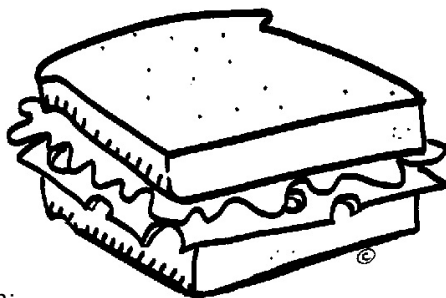
`wave.c` offers only Ricker wavelet (2nd deriv of Gaussian) source option, peak frequency = `freq`, via function call:

```
p0[isz+isx*nz]+=fgetrick(it*dt,freq);
```

Initialization: assignment functions

```
nxz = nx*nz;  
fzeros(p0,nxz);  
fzeros(p1,nxz);
```

wave.c: structure



Code sandwich:

- ▶ `meat = step_forward`
- ▶ top bread = load survey geometry parameters from command line, binary velocity data from disk file (optional), open files for output, allocate memory, sanity checks, start loop over sources, initialize fields, start time loop
- ▶ bottom bread = transfer data at each time step to trace buffer and/or movie file, time loop bottom, flush trace data to file, source loop bottom, clean up.

wave.c: capabilities and limitations

+:

- ▶ self-documenting after Seismic Unix - uses SU self-doc utility (`main/wave.x` - doc's all command line args for all commands)
- ▶ arbitrary length of receiver line
- ▶ arbitrary number of (evenly spaced) point sources (“survey”)
- ▶ arbitrary peak frequency
- ▶ arbitrary velocity structure

wave.c: capabilities and limitations

-:

- ▶ receiver spacing = grid Δx , receiver at every gridpoint in “cable”
- ▶ fixed spread for all sources,
- ▶ source spacing = multiple of grid Δx , shots on gridpoints
- ▶ source, receiver depths multiples of grid Δz
- ▶ sample rate of traces = simulation Δt
- ▶ output files are flat binary (no SEG Y, SEG D, SEP/RSF,...)
- ▶ not every possible sanity check
- ▶ highly non-optimal FD scheme
- ▶ no absorbing boundary conditions

Upshot: not industrial strength, even as 2D const acoustic sim - just a testbed!

Optimization

- ▶ elementary opts - reduction of storage, data motion already explained
- ▶ many others - tomorrow (parallel), Wednesday (parallel & serial)

Loop vectorization: available for the price of a compiler flag

- ▶ takes advantage of limited SIMD capabilities of recent Intel, AMD hardware via SSE
- ▶ requirements: regular strides, aligned arrays (see commented-out `usermalloc` code in `src/utils/utils.c`, no recursion in loop - once address is written, should not be read again (restrict assertion necessary)
- ▶ inner (`iz`) loop in `src/step.c:step_forward` vectorizes - roughly 40% speedup on recent Opteron

Parallelization over Sources

Survey is a **multisimulation**, all sources independent

⇒ “embarrassing” parallelization

Many waves to implement, using MPI, scripting languages

`main/sqwave.c`: MPI implementation of source parallel simulator

Tomorrow - much trickier parallelization of single-source simulation

DEMO `mpiwave2:sqtrace.bin` and `trace.bin`

Agenda

Wave physics, wave equations, and waves

Finite difference methods for the wave equation: accuracy, stability, dispersion

`wave.c`: a simple finite difference modeling application

Imaging with waves: reverse time migration

`rtm.c`: a simple reverse time migration application

An exercise: decreasing dispersion by increasing order

Critical references for imaging and reverse time migration

- ▶ SEG reprint volume on Migration (“the classics”)
- ▶ Yilmaz, Seismic Data Processing (2nd ed.), SEG 2001
- ▶ WS, Optimal checkpointing for RTM, *Geophysics* 2008

Basic physics of imaging

Concept, first expressed clearly by Claerbout in early 70's (but implicit much earlier):

- ▶ incident wave (source wavefield, p) and reflected wave (receiver wavefield, q) coincide in space and time at reflectors
- ▶ true for every source and receiver - sum over sources and receivers interferes *constructively* at reflectors, *destructively* elsewhere
- ▶ reflectors appear as strong components in *image* = zero-lag time correlation of source (p) and receiver (q) wavefields:
RTM formula

$$I(z, x) = \sum_{x_s} \int dt p(z, x, t; x_s) q(z, x, t; x_s)$$

Basic physics of imaging

Receiver wavefield $q(z, x, t; x_s)$ is solution of wave equation

$$\frac{1}{c^2} \frac{\partial^2 q}{\partial t^2} - \nabla^2 q = \sum_{x_r} d(x_r, t; x_s) \delta(z - z_r) \delta(x - x_r)$$

with *anti-causal* initial condition: $q = 0, t \gg 0$. Solved **backwards in time**.

“Receivers act as sources”: each data trace $d(x_r, t; x_s)$ serves as a radiating pulse

Basic physics of imaging

Mathematically coherent treatment: Cohen & Bleistein 1977, Beylkin 1985, Rakesh 1988, many others

Upshot: migration is an approximate inverse to the Born (linearized) modeling operator - models *single scattering*

Born modeling operator is “nearly” unitary - its adjoint differs from its inverse by computable scaling transformations in space and frequency

RTM = *adjoint state method* applied to Born modeling operator - computes adjoint

Implementation issues in reverse time migration

“Source wavefield” = $p(\mathbf{x}, t; x_s)$, approximate using FD scheme explained previously

“Receiver wavefield” = $q(\mathbf{x}, t; x_s)$, approximate using same FD scheme running backwards, with input data traces $d(x_r, t)$ providing source

Imaging condition:

$$I(n\Delta z, m\Delta x) \simeq \sum_{x_s} \Delta t \sum_k p_{n,m}^k q_{n,m}^k$$

RHS for q eqn at timestep k =

$$\sum_m d_m^k \delta_{n_s, m}, \quad d_m^k = d(m\Delta x, k\Delta t)$$

Implementation issues in reverse time migration

p is computed in order of increasing k , q in order of decreasing k , yet you **need them at the same k !** Remedies:

- ▶ Precompute p^k for $k = 0, \dots, k_{\max}$, save to disk, read in as required as q^k updated. Lots of disk i/o.
- ▶ Same, but only save every Δk th step, and only compute imaging condition every Δk th step during backwards q^k loop - Δt far below Nyquist. Used in some production RTM.
- ▶ *Time-reverse* p^k evolution - compute for $k = 0, \dots, k_{\max}$, recompute same fields in synchrony with q^k . Flop premium of 50% but either no i/o (free surfaces all around) or less per time step (absorbing BC, save boundary data). Pretty good, used in some production code.
- ▶ Optimal checkpointing - $\log k_{\max}$ storage, $\log k_{\max}$ additional simulations, only forward time steps, works also for dissipative models like viscoelasticity - see article by WS in 2007 special issue of *Geophysics*

Agenda

Wave physics, wave equations, and waves

Finite difference methods for the wave equation: accuracy, stability, dispersion

`wave.c`: a simple finite difference modeling application

Imaging with waves: reverse time migration

`rtm.c`: a simple reverse time migration application

An exercise: decreasing dispersion by increasing order

rtm.c: structure

Uses same infrastructure as `wave.c`, evolution loops all `step_forward` - only additional cmd line are `image` (required) and `receiver wavefield movie` (optional) filenames.

Self-doc provided by `wave.x`

Major differences:

- ▶ data file (trace keyword) is *input*, not output - must exist
- ▶ source wavefield either
 - ▶ written to disk if value assigned to source keyword, then read back, or
 - ▶ time-reversed otherwise (no i/o)

DEMO test1:rtm.view, rtmnm.view test3:rtmnm.view

Agenda

Wave physics, wave equations, and waves

Finite difference methods for the wave equation: accuracy, stability, dispersion

`wave.c`: a simple finite difference modeling application

Imaging with waves: reverse time migration

`rtm.c`: a simple reverse time migration application

An exercise: decreasing dispersion by increasing order

The (2,4) scheme

2nd order in time, 4th order in space - with $D_z^2 = 4$ th order approx. to 2nd z-deriv, etc., as explained in first part of course,

$$p_{n,m}^{k+1} = 2p_{n,m}^k - p_{n,m}^{k-1} + \\ (c^2)_{n,m} \Delta t^2 [D_z^2 p_{n,m}^k + D_x^2 p_{n,m}^k + w^k \delta_{n,m}]$$

Properties:

- ▶ substantially less dispersion than (2,2) scheme, for only a few more flops
- ▶ 5% phase velocity error over 100 wavelengths with $G \simeq 6$
- ▶ used in many industry, academic codes - eg. E3D (Larson)

The (2,4) scheme

Exercise: upgrade `wave.c` to (2,4) scheme

Suggestions:

Can reuse virtually entire infrastructure (“bread”)

Write new `step_forward` to include *method of images* updates at boundary: either

- ▶ keep physical boundary at $n = 0$ etc., write special stencil for $n = 1$ (row/col just inside boundary) with implicit cond'n $p(-1, \dots) = -p(1, \dots)$ etc.
- ▶ allocate larger arrays with one more row and column on each side, indexed with $n = -1$ etc. - must compute difference between *coordinate indices* n etc. and offset in array row/col. Then method of images \Rightarrow run around boundary explicitly setting $p(-1, \dots) = p(1, \dots)$