Programming for Geophysics

Bill Harlan

May 21, 2008


WHILE YOU WERE DRONING I SLAMMED OUT SOME BETA CODE AND PUT IT ON THE INTERNET FOR COMMENTS.

Some ways to program as a geophysicist

- Avoid it. Borrow code. Find partners. Use Matlab.

Some ways to program as a geophysicist

- Avoid it. Borrow code. Find partners. Use Matlab.
- Become a guru/sysadmin. Help others publish first.

Some ways to program as a geophysicist

- Avoid it. Borrow code. Find partners. Use Matlab.
- Become a guru/sysadmin. Help others publish first.
- Concentrate on numerics. Fortran. One program per dataset.

Some ways to program as a geophysicist

- Avoid it. Borrow code. Find partners. Use Matlab.
- Become a guru/sysadmin. Help others publish first.
- Concentrate on numerics. Fortran. One program per dataset.
- Build a personal library. Generalize your code for reuse.

Learn fundamentals deliberately, not as you go

- Take a course, read books
  - Data structures, algorithms, object-oriented, functional

Learn fundamentals deliberately, not as you go

- Take a course, read books
  - Data structures, algorithms, object-oriented, functional
- Learn as a branch of math, not engineering.
  - Concentrate on reusable abstractions, not popular toolkits.
  - Master simplicity, not complexity.

Learn fundamentals deliberately, not as you go

- Take a course, read books
  - Data structures, algorithms, object-oriented, functional
- Learn as a branch of math, not engineering.
  - Concentrate on reusable abstractions, not popular toolkits.
  - Master simplicity, not complexity.
- Do not get carried away.

Learn best software practices

- Show and share
- Source control
- Tests
- Small changes (refactoring)
- Appropriate generalization

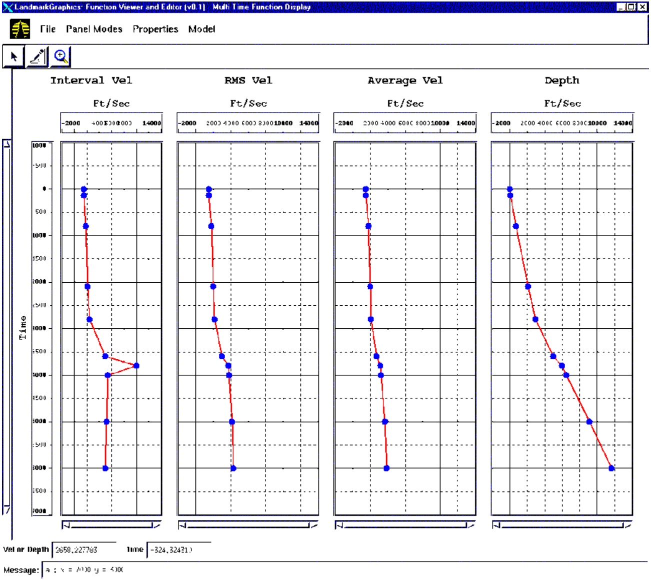
Examples of generalization/abstraction

- Seismic data objects with flexible dimensions
- Separate velocity models from ray tracers
- Different imaging conditions with different extrapolators

Typical geophysical inversions

- Data simulated by series of non-linear operations
- Inversion is both over- and under-determined
- No model parameters fit data perfectly
- Many models fit data equally well
- Non-linearity is well-behaved

Sensitivity of interval velocity to RMS errors



Dix inversion

- Forward equation cannot fit arbitrary data:

$$V_j^{\mathrm{rms}} = \sqrt{\frac{1}{j}\sum_{k=1}^{j}(V_k^{\mathrm{int}})^2}$$

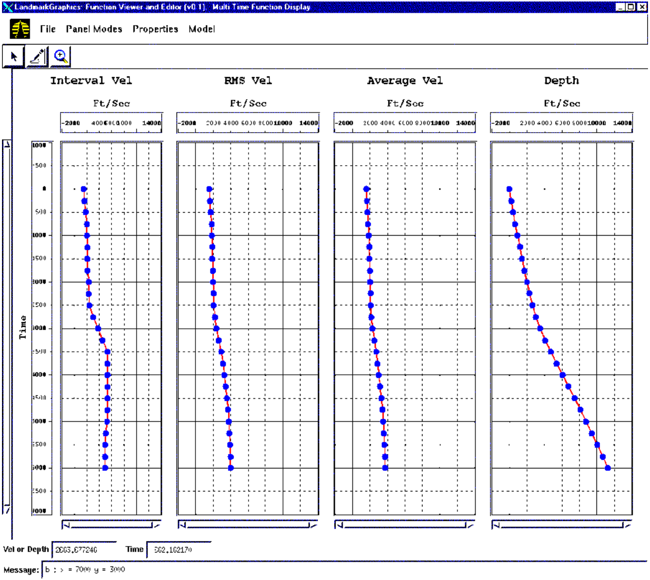
- Explicit inverse may not be physical:

$$V_j^{\mathrm{int}} = \sqrt{j(V_j^{\mathrm{rms}})^2 - (j-1)(V_{j-1}^{\mathrm{rms}})^2}$$

- Instead minimize damped least-squares:

$$\sum_j \left\{ (V_j^{\mathrm{rms}})^{-2} - \left[\frac{1}{j}\sum_{k=1}^{j}(V_k^{\mathrm{int}})^2\right]^{-1} \right\}^2 + \epsilon \sum_k (V_k^{\mathrm{int}})^{-2}$$

Damp interval velocity roughness



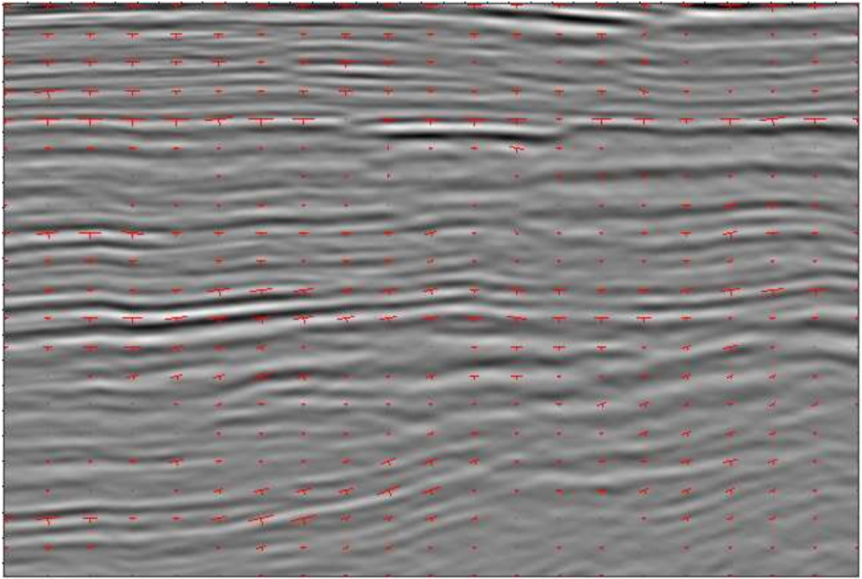
Defining an inversion

- Do not define your solution by the way you solve it.
- Want to improve the solution without redefining the problem.
- Instead, identify an objective function (or probabilities).
  E.g., define rays by minimum time.

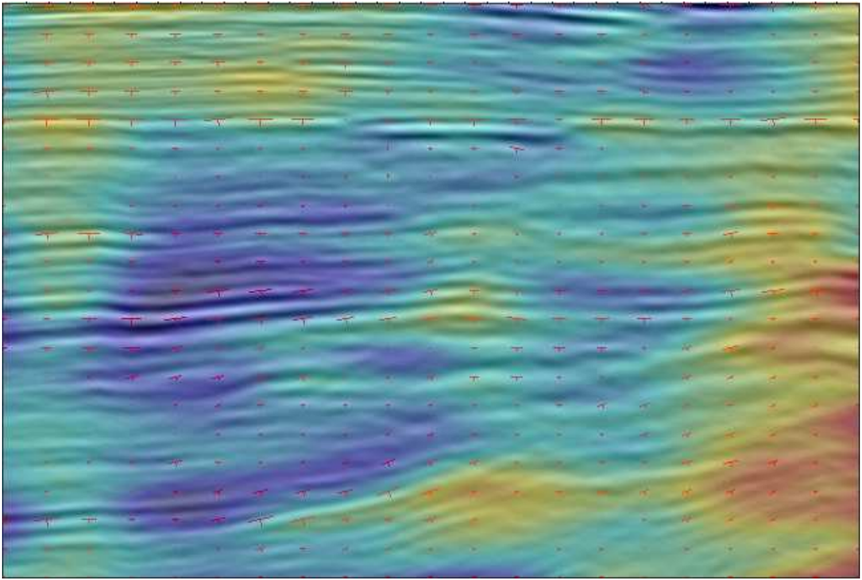
Lomask's flattening, redone

- Estimate vertical stretch that flattens reflections.
- Original: Custom regression, phase-unwrapping
- New version: A few hundred lines of code specific to inversion
- JTK reused: structure tensors, Gaussian filters, Gauss-Newton

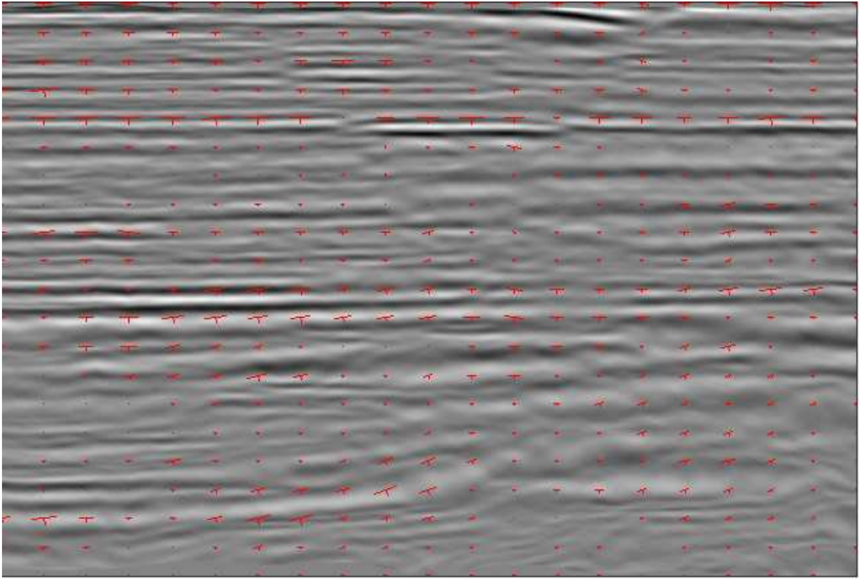
Local dips from structure tensors



Estimated vertical shifts in color



Flattened with vertical shifts



The problem, the data, and the solution

Flatten seismic structure with vertical shift $\tau(x, y, t)$:

$$\text{flat}(x, y, t) = \text{structure}[x, y, t + \tau(x, y, t)].$$

Data are slopes p_x, p_y measured from structure tensors.

$$\begin{aligned} \text{Want} \quad \frac{\partial}{\partial x}\tau(x, y, t) &\approx p_x(x, y, t) \\ \text{and} \quad \frac{\partial}{\partial y}\tau(x, y, t) &\approx p_y(x, y, t). \end{aligned}$$

$$\begin{aligned} \min_{\tau(x,y,t)} \iiint \Big(\|\tfrac{\partial}{\partial x}\tau(x, y, t) - p_x(x, y, t)\|^2 \\ + \|\tfrac{\partial}{\partial y}\tau(x, y, t) - p_y(x, y, t)\|^2 \\ + \epsilon\|\tau(x, y, t)\|^2 \Big)\, dx\, dy\, dt \end{aligned}$$

Looks like damped least-squares

The best model m fits the data d with a function f(d) by minimizing the vector norms

$$\|\mathrm{d} - \mathrm{f}(\mathrm{m})\|_d^2 + \|\mathrm{m}\|_m^2$$

$$\text{or} \quad [\mathrm{d} - \mathrm{f}(\mathrm{m})]^* \,\underset{\sim}{\mathrm{C}}_d^{-1} \,[\mathrm{d} - \mathrm{f}(\mathrm{m})] + \mathrm{m}^* \underset{\sim}{\mathrm{C}}_m^{-1} \mathrm{m}.$$

Optional covariances:

$$\underset{\sim}{\mathrm{C}}_d \equiv E(\mathrm{d}\,\mathrm{d}^*) \text{ and } \underset{\sim}{\mathrm{C}}_m \equiv E(\mathrm{m}\,\mathrm{m}^*).$$

Gauss-Newton inversion

- Finds m to minimize

$$[\mathrm{d} - \mathrm{f}(\mathrm{m})]^* \underset{\sim}{C}_d^{-1} [\mathrm{d} - \mathrm{f}(\mathrm{m})] \; + \; [\mathrm{m} - \mathrm{m}_0]^* \underset{\sim}{C}_m^{-1} [\mathrm{m} - \mathrm{m}_0]$$

- Algorithm:
  1. Accepts starting model m_0
  2. Approximates $\mathrm{f}(\mathrm{m}_0 + \Delta\mathrm{m}) \approx \mathrm{f}(\mathrm{m}_0) + \underset{\sim}{F} \cdot \Delta\mathrm{m}$
  3. Conjugate-gradients minimizes quadratic for $\Delta\mathrm{m}$
  4. Line search scales perturbation: $\mathrm{m}_0 + \alpha\Delta\mathrm{m}$
  5. Adds perturbation to reference model for new m_0
  6. Returns to step 2

Required operations

- Simulate data from model:
  $d = f(m)$
- Perturb data with model perturbation:
  $\Delta d = \underset{\sim}{F}(m_0) \cdot \Delta m \;\approx\; f(m_0 + \Delta m) - f(m_0)$
- Perturb model with transpose:
  $\underset{\sim}{F}(m_0)^* \cdot \Delta d$

What is that transpose?

Use definition: $d^*(\underset{\sim}{F}m) \equiv (\underset{\sim}{F}^*d)^*m$

- Discrete: swap summations
- Continuous: integrate by parts

Examples:

- Smoothing $\rightarrow$ Smoothing
- Convolution $\rightarrow$ Correlation
- Derivative $\rightarrow$ Negative derivative
- Plane-wave modeling $\rightarrow$ Slant stacks
- Seismic modeling $\rightarrow$ Migration

Inversion sees three abstract operations

```
Vector data  = forward(Vector model)
Vector data  = linearized(Vector model, Vector refModel)
Vector model = transpose(Vector data, Vector refModel)
```

Required operations for both data and model

```
Vector {
  scale(float scalar)           [required]
  add(Vector other)
  dot(Vector other)

  multiplyInverseCovariance()  [optional]
  applyHardConstraint()
}
```

Constrained Dix inversion

- Solve for smooth interval slowness m.
- Minimize errors in squared stacking slowness d
- Forward transform:

$$1/d_j = (1/j) \sum_{k=1}^{j} (1/m_k^2)$$

- Linearized transform:

$$\Delta d_j = (2\, d_j^2/j) \sum_{k=1}^{j} (1/m_k^3) \Delta m_k$$

- Transpose transform:

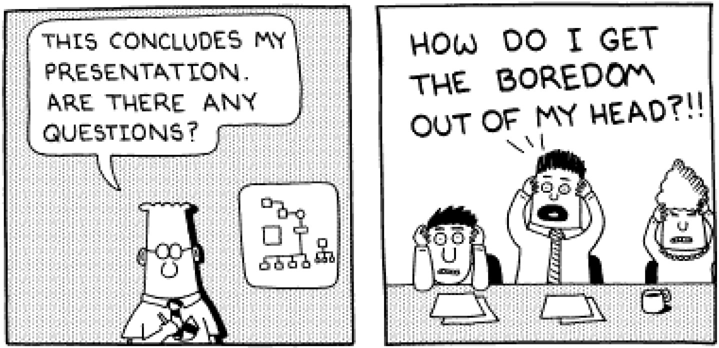
$$\Delta m_k = (1/m_k^3) \sum_{j=k}^{\infty} (2\, d_j^2/j) \Delta d_j$$

Other applications

- Tomography: reflection, cross-well, diving, amplitude
- Generalized Radon transforms
- Surface-consistent deconvolution
- Normal moveout corrections
- Automatic moveout picking
- Coherency, wavelet/phase attributes
- Tests for simulations

Conclusions

- More time on "computer science" quickly saves time
- Look for opportunities to generalize


THIS CONCLUDES MY PRESENTATION. ARE THERE ANY QUESTIONS?
HOW DO I GET THE BOREDOM OUT OF MY HEAD?!!

Alternative to covariance

- Insert simplification filter $\mathrm{d} = \mathrm{f}(\underset{\sim}{\mathrm{S}} \cdot \mathrm{m})$
  where $\underset{\sim}{\mathrm{S}}^{*}\underset{\sim}{\mathrm{C}}_{m}^{-1}\underset{\sim}{\mathrm{S}} \approx \underset{\sim}{\mathrm{I}}$
- If $\underset{\sim}{\mathrm{C}}_{m}^{-1} \cdot \mathrm{m}$ roughens, then $\underset{\sim}{\mathrm{S}} \cdot \mathrm{m}$ smooths.
- Faster than covariance constraint
- Can change dynamically during optimization