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

Some ways to program as a geophysicist

- Become a guru/sysadmin. Help others publish first.
Some ways to program as a geophysicist

- Become a guru/sysadmin. Help others publish first.
- Concentrate on numerics. Fortran. One program per dataset.
Some ways to program as a geophysicist

- 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:**
  \[ V_{\text{rms}}^j = \sqrt{\frac{1}{j} \sum_{k=1}^{j} (V_{\text{int}}^k)^2} \]

- **Explicit inverse may not be physical:**
  \[ V_{\text{int}}^j = \sqrt{j(V_{\text{rms}}^j)^2 - (j-1)(V_{\text{rms}}^{j-1})^2} \]

- **Instead minimize damped least-squares:**
  \[ \sum_j \left\{ (V_{\text{rms}}^j)^{-2} - \left[ \frac{1}{j} \sum_{k=1}^{j} (V_{\text{int}}^k)^2 \right]^{-1} \right\}^2 + \epsilon \sum_k (V_{\text{int}}^k)^{-2} \]
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.

Want

$$\frac{\partial}{\partial x} \tau(x, y, t) \approx p_x(x, y, t)$$

and

$$\frac{\partial}{\partial y} \tau(x, y, t) \approx p_y(x, y, t).$$

$$\min_{\tau(x,y,t)} \int \int \int (\| \frac{\partial}{\partial x} \tau(x, y, t) - p_x(x, y, t) \|^2$$

$$+ \| \frac{\partial}{\partial y} \tau(x, y, t) - p_y(x, y, t) \|^2$$

$$+ \epsilon \| \tau(x, y, t) \|^2 ) \, dx \, dy \, dt$$
Looks like damped least-squares

The best model $\mathbf{m}$ fits the data $\mathbf{d}$ with a function $\mathbf{f}(\mathbf{d})$ by minimizing the vector norms

$$\| \mathbf{d} - \mathbf{f}(\mathbf{m}) \|_{\mathbf{d}}^2 + \| \mathbf{m} \|_{\mathbf{m}}^2$$

or

$$[\mathbf{d} - \mathbf{f}(\mathbf{m})]^* \mathbf{C}_d^{-1} [\mathbf{d} - \mathbf{f}(\mathbf{m})] + \mathbf{m}^* \mathbf{C}_m^{-1} \mathbf{m}.$$  

Optional covariances:

$$\mathbf{C}_d \equiv E(\mathbf{d} \mathbf{d}^*) \quad \text{and} \quad \mathbf{C}_m \equiv E(\mathbf{m} \mathbf{m}^*).$$
Gauss-Newton inversion

► Finds $\mathbf{m}$ to minimize

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

► Algorithm:

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

- Simulate data from model:
  \[ d = f(m) \]

- Perturb data with model perturbation:
  \[ \Delta d = \tilde{F}(m_0) \cdot \Delta m \approx f(m_0 + \Delta m) - f(m_0) \]

- Perturb model with transpose:
  \[ \tilde{F}(m_0)^* \cdot \Delta d \]
What is that transpose?

Use definition: \( d^* (\mathcal{F} m) \equiv (\mathcal{F}^* d)^* m \)

- 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:

\[
\frac{1}{d_j} = \frac{1}{j} \sum_{k=1}^{j} \frac{1}{m_k^2}
\]

- Linearized transform:

\[
\Delta d_j = \left(2 \frac{d_j^2}{j}\right) \sum_{k=1}^{j} \frac{1}{m_k^3} \Delta m_k
\]

- Transpose transform:

\[
\Delta m_k = \frac{1}{m_k^3} \sum_{j=k}^{\infty} \left(2 \frac{d_j^2}{j}\right) \Delta d_j
\]
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 $\mathbf{d} = \mathbf{f}(\mathbf{S} \cdot \mathbf{m})$
  where $\mathbf{S}^* \mathbf{C}_m^{-1} \mathbf{S} \approx \mathbf{I}$
- If $\mathbf{C}_m^{-1} \cdot \mathbf{m}$ roughens, then $\mathbf{S} \cdot \mathbf{m}$ smooths.
- Faster than covariance constraint
- Can change dynamically during optimization