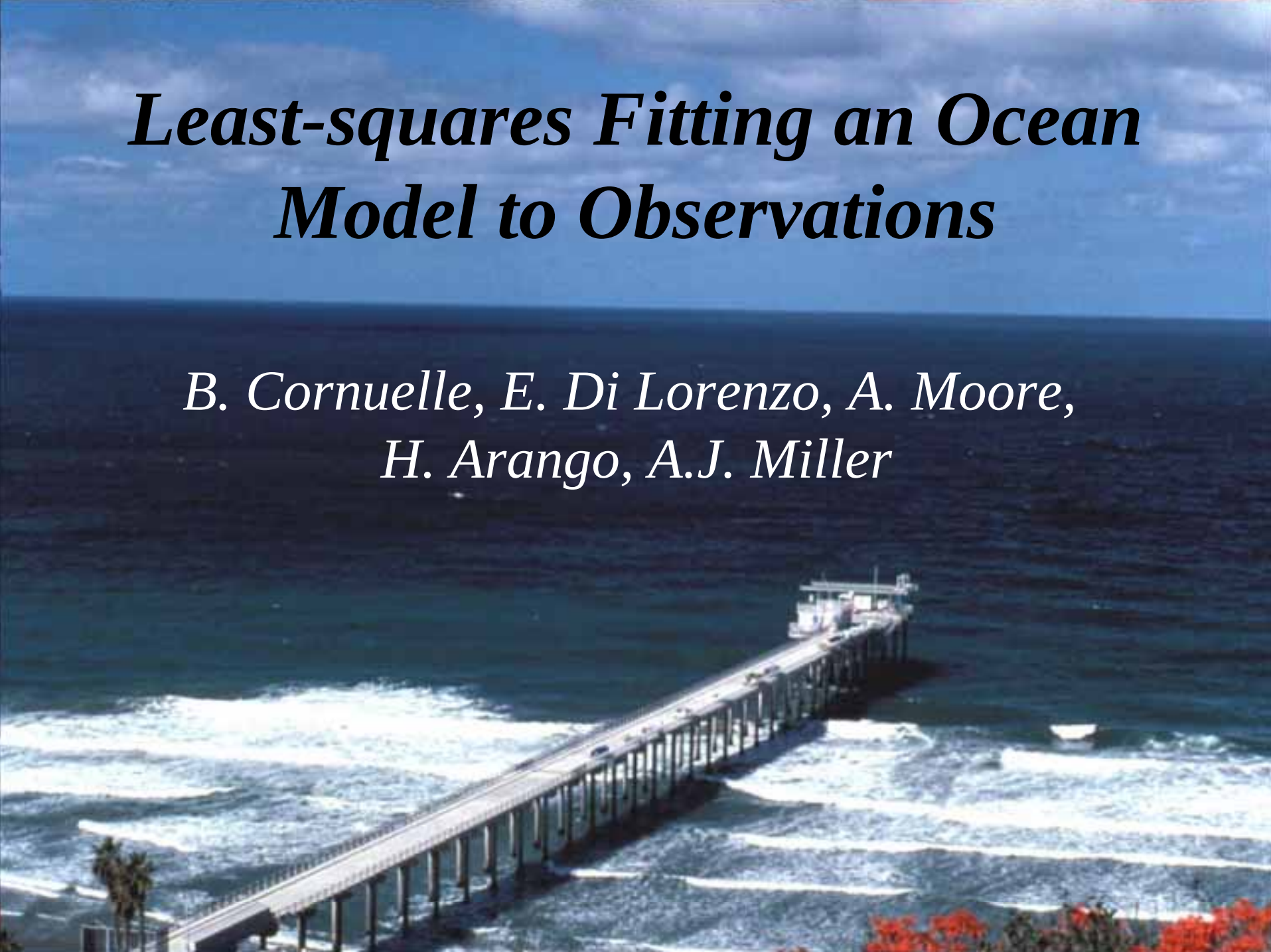
Least-squares Fitting an Ocean Model to Observations

B. Cornuelle, E. Di Lorenzo, A. Moore, H. Arango, A.J. Miller



Motivations

- Merge datasets together
- Use ocean dynamics as constraints
- Evaluate model (and data) quality
- Estimate forcing and bdry conds

References

Books:

Bennett, Andrew F.

Inverse methods in physical oceanography,
Cambridge University Press, 1992.
Inverse modeling of the ocean and atmosphere,
Cambridge University Press, 2002.

Lanczos, Cornelius

Linear differential operators,
Dover, 1964.
The variational principles of mechanics
Dover, 1970.

Menke, William.

Geophysical data analysis : discrete inverse theory,
Academic Press, 1989.

References, continued

Parker, Robert L.
Geophysical inverse theory
Princeton University Press, 1994.

Tarantola, Albert.
Inverse problem theory : methods for data fitting and model parameter estimation,
Elsevier, 1987

Wunsch, Carl.
The ocean circulation inverse problem,
Cambridge University Press, 1996.

Journal Articles:

Le Dimet, F.-X.; Talagrand, O.
Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects
Tellus A, **38**, pp. 97, 1986.

Talagrand, Olivier and Philippe Courtier,
Variational assimilation of meteorological observations with the adjoint vorticity equation. I: Theory
Quarterly Journal of the Royal Meteorological Society, **113**, pp. 1311-1328, 1987.

Weaver, Andrew, J. Vialard, and D.L.T. Anderson,
Three- and Four-Dimensional Variational Assimilation with a general circulation model of the tropical Pacific Ocean. Part I: Formulation, internal diagnostics, and consistency checks
Monthly Weather Review, **131**, pp. 1360-1378, 2003.

Outline

1) Summary of Bayesian statistics, cost functions
2) Least-squares inversion formulas; over- and under-determined
3) Gradient methods for approximate inversion
4) Priors and Smoothing
5) Nonlinearity: unbounded sensitivity (no)
6) Not a speck of science!

Quotes out of context

- Imitation (stealing) is the sincerest form of flattery
- Everything should be made as simple as possible, but no simpler (A. Einstein)
- Foolish consistency is the hobgoblin of small minds (R.W. Emerson)
- Willem, a grown man like you, working on linear problems!? (J. Keller?)

Notation Pitfalls/Issues

- Several conflicting conventions; have to compromise.
- Control theory (e.g. Wunsch)
- Geophysical inverse theory (e.g. Menke)
- Bennett
- Weaver
- Objective mapping (old)
- Sometimes leads to duplication; sorry!

Control Theory (matrices)

- P = covariance of parameter uncertainty
- R = covariance of data uncertainty (error)
- Q = covariance of model error at each step
- A = transition matrix (Andy's "R" propagator)
- H = Sampling matrix
- x = unknown state vector
- y = data

Geophysical Inverse Theory (matrices)

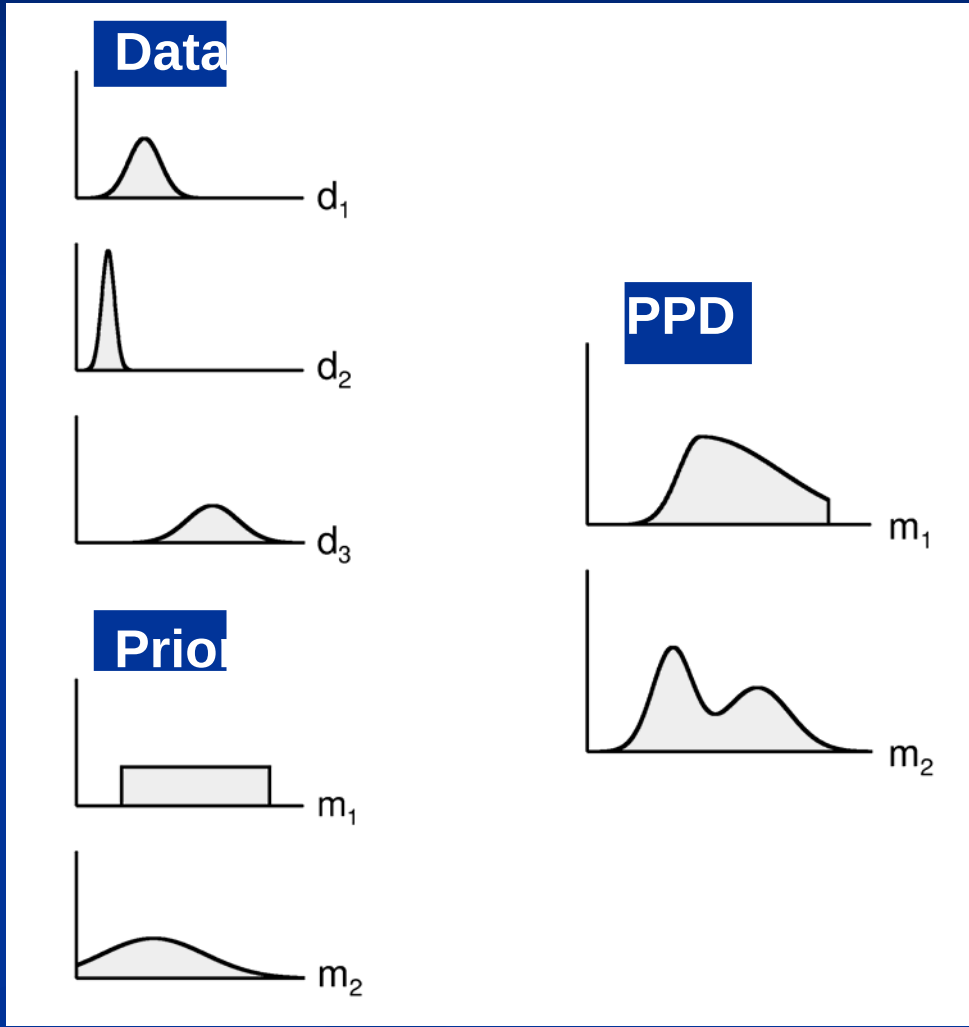
- P = covariance of parameter uncertainty
- R = covariance of data uncertainty (error)
- G = Sampling matrix (G = H*A)
- m = unknown parameters
- d = data

Bennett (time and space continuous)

- C = covariance of parameter uncertainty
- $\mathbf{C}_\varepsilon$ = covariance of data error (used in the following)
- R = representer matrix (OA: “data-data covariance”)
- P = R + $\mathbf{C}_\varepsilon$ (stabilized representer matrix)
- r = representer functions
- L = Sampling functional
- u = unknown parameters
- d = data
- $\hat{\beta}$ = representer coefficients (or “rotated data”)

Bayesian Estimation; MAP

- Combine data & prior information to define *Posterior Probability Density (PPD)*
- *PPD* quantifies model probability over M-D parameter space



Bayes Theorem

- *Bayes Theorem:*

$$P(\mathrm{m}\,|\,\mathrm{d})\;P(\mathrm{d}) \;=\; P(\mathrm{d}\,|\,\mathrm{m})\;P(\mathrm{m})$$

PPD — Likelihood — prior

- *Likelihood:* data uncertainty distribution, interpreted as function of m (for measured d). Typically

$$P(\mathrm{d}\,|\,\mathrm{m}) \propto \exp[-E(\mathrm{m},\mathrm{d})]$$

e.g.: $P(\mathbf{d}\,|\,\mathbf{m}) \propto \exp[-(\mathbf{d} - \mathrm{G}(\mathbf{m}))^{\mathrm{T}}\mathbf{C}_{\varepsilon}^{-1}(\mathbf{d} - \mathbf{G}(\mathbf{m}))]$ Data misfit

- *Prior:* existing knowledge of m

Stan Dosso; U. Victoria BC CA

A posteriori probability distribution: PPD

- Bayes Theorem: $P(\mathrm{m}\,|\,\mathrm{d}) \;\propto\; \exp[-E(\mathrm{m},\mathrm{d})]\; P(\mathrm{m})$

- PPD:

$$P(\mathrm{m}\,|\,\mathrm{d}) \;=\; \frac{\exp[-\phi(\mathrm{m},\mathrm{d})]}{\int_M \exp[-\phi(\mathrm{m}',\mathrm{d})]\; d\mathrm{m}'}$$

- **Log likelihood**

$$\phi(\mathrm{m},\mathrm{d}) = E(\mathrm{m},\mathrm{d}) - \log_{\mathrm{e}} P(\mathrm{m})$$

(generalized misfit)

→ Interpret *M*-dimensional distribution?

Stan Dosso; U. Victoria BC CA

PPD Interpretation

- *M*-D PPD interpreted by properties defining parameter estimates, uncertainties, inter-relationships

$$\hat{\mathrm{m}}_{\mathrm{MAP}} = \mathrm{Arg}_{\mathrm{max}} \left\{ P(\mathrm{m} \mid \mathrm{d}) \right\}$$

Maximum ("mode") MAP

$$\hat{\mathrm{m}}_{\mathrm{Mean}} = \int \mathrm{m}' \; P(\mathrm{m}' \mid \mathrm{d}) \; d\mathrm{m}'$$

Mean (1st moment)

$$P(m_i \mid \mathrm{d}) = \int \delta(m_i - m_i') \; P(\mathrm{m}' \mid \mathrm{d}) \; d\mathrm{m}'$$

Marginal Distribution

$$\mathbf{C}_{ij} = \int (m'_i - \hat{m}_i)(m'_j - \hat{m}_j) \; P(\mathrm{m}' \mid \mathrm{d}) \; d\mathrm{m}'$$

Covariance (2nd moment)

PPD Optimization / Integration

- MAP requires maximizing PPD $\rightarrow$ minimize $\phi(\mathrm{m})$
  e.g. using global or (adaptive) hybrid inversion

- Marginals, covariance, *etc* require integrating PPD

$$I = \int f(\mathrm{m}')\ P(\mathrm{m}'|\,\mathrm{d})\ d\mathrm{m}'$$

- For nonlinear problems, numerical integration required via *Importance Sampling* and *Markov Chain / Monte Carlo* methods

Analytical Forms for PDF: Cost Functions

- *Given the form of the PDF, the MAP estimator becomes a norm to be maximized*

$$P(\mathbf{m}\,|\,\mathbf{d}) \;\propto\; \exp[-E(\mathbf{m},\mathbf{d})]\;P(\mathbf{m})$$

- *Example: Gaussian statistics:*

$$E(\mathbf{m},\mathbf{d}) = (\mathbf{d} - \mathbf{G}(\mathbf{m}))^{\mathrm{T}}\mathbf{C}_{\varepsilon}^{-1}(\mathbf{d} - \mathbf{G}(\mathbf{m}))$$

$$\log_e P(\mathbf{m}) = \mathbf{m}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{m}$$

$$\phi(\mathbf{m},\mathbf{d}) = (\mathbf{d} - \mathbf{G}(\mathbf{m}))^{\mathrm{T}}\mathbf{C}_{\varepsilon}^{-1}(\mathbf{d} - \mathbf{G}(\mathbf{m})) + \mathbf{m}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{m}$$

Max(log likelihood) = minimum cost function

Aside: Quadratic norms

- *The quadratic product defines a parabolic surface (manifold) in the shape of an elipsoid of rotation*
- *Diagonalizing the weighting matrix identifies principal axes*
- *Example: simple function (or functional if continuous)*

$$f(\mathbf{m}) = \mathbf{m}^T \mathbf{P}^{-1} \mathbf{m}$$ Decompose into eigenvectors

$$f(\mathbf{m}) = \mathbf{m}^T (\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T)^{-1} \mathbf{m}$$ where $$\mathbf{U}\mathbf{U}^T = \mathbf{I} = \mathbf{U}^T\mathbf{U}$$

$$f(\mathbf{m}) = (\mathbf{U}^T\mathbf{m})^T (\boldsymbol{\Lambda})^{-1} (\mathbf{U}^T\mathbf{m})$$

$$\mathbf{m}' = \mathbf{U}^T\mathbf{m}$$ = Independent variables

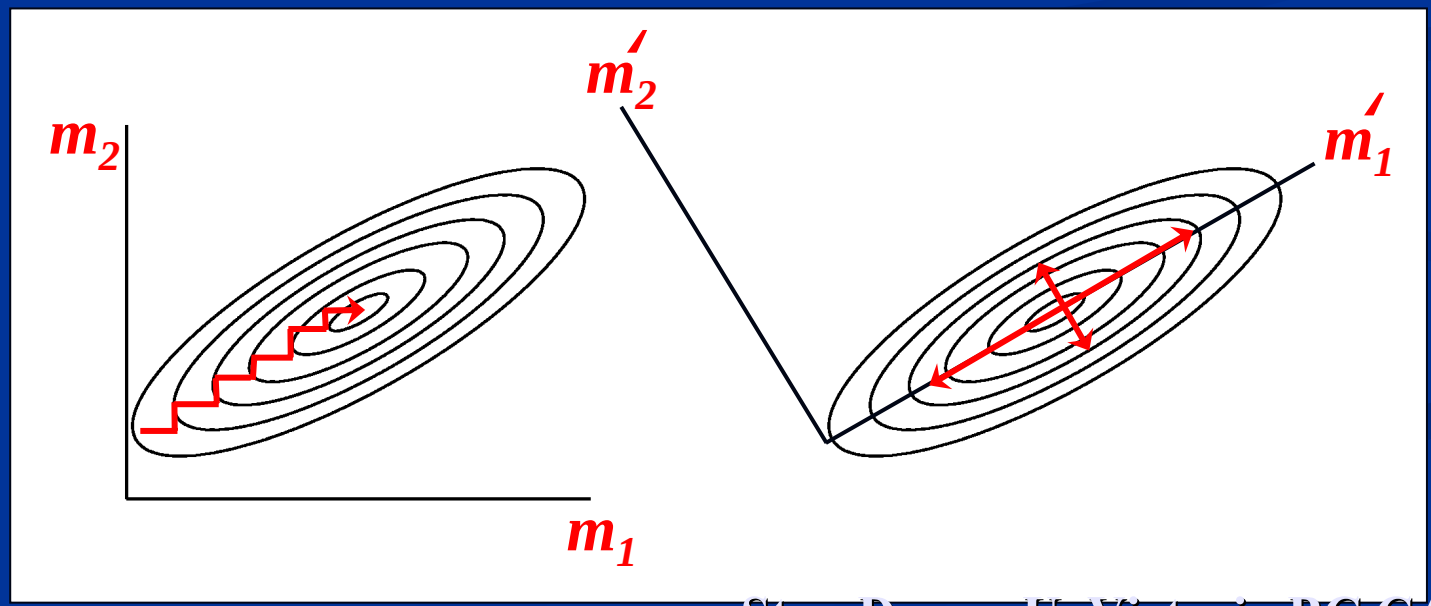
Or: orthonormalize: e.g. Cholesky factorization (caution!)

$$f(\mathbf{m}) = \mathbf{m}^T \mathbf{P}^{-T/2} \mathbf{P}^{-1/2} \mathbf{m} \qquad f(\mathbf{m}) = (\mathbf{P}^{-1/2}\mathbf{m})^T (\mathbf{P}^{-1/2}\mathbf{m})$$

Re-parameterization

- Sampling along parameter axes inefficient for correlated parameters → *rotate to principal component parameter space* by diagonalizing covariance

$$\mathbf{P} = \mathbf{U}\Lambda\mathbf{U}^T \quad \Rightarrow \quad \mathrm{m}' = \mathbf{U}^T\mathrm{m}$$



Least Squares Minimization

$$\mathbf{d}(t_1)=\mathbf{G}(\mathbf{x}(t_0))+\mathbf{r}(t_1)$$

- Assume Gaussian PDFs for x, r (can be dangerous!!)
- Minimize difference between model solution and observations and size of control ("background") in a least-squares sense.
- Cost function

$$J=\underbrace{\frac{1}{2}\left(\mathbf{G}(\mathbf{x}(t_0))-\mathbf{d}(t_1)\right)^{T}\mathbf{C}_{\varepsilon}^{-1}\left(\mathbf{G}(\mathbf{x}(t_0))-\mathbf{d}(t_1)\right)}_{\text{Observation term } (J_o)}$$

$$+\underbrace{\frac{1}{2}\left(\mathbf{x}(t_o)-\mathbf{x}_b(t_0)\right)^{T}\mathbf{P}^{-1}\left(\mathbf{x}(t_0)-\mathbf{x}_b(t_0)\right)}_{\text{Background term } (J_b)}$$

If the model is linear, then $\mathbf{d}(t_1) = \mathbf{G}_1\mathbf{x}(t_0) + \mathbf{r}(t_1)$

For simplicity, define x as a CORRECTION, so $\mathbf{x}_b(t_0) = 0$

The cost function is

$$J = (\mathbf{d}(t_1) - \mathbf{G}_1\mathbf{x}(t_0))^T \mathbf{C}_\varepsilon^{-1} (\mathbf{d}(t_1) - \mathbf{G}_1\mathbf{x}(t_0)) + \mathbf{x}(t_0)^T \mathbf{P}^{-1}\mathbf{x}(t_0)$$

R is the covariance of data error, **P** is the covariance of model error

The gradient of J with respect to initial x is:

$$\frac{\partial J}{\partial \mathbf{x}(t_0)} = \mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1} (\mathbf{d}(t_1) - \mathbf{G}_1\mathbf{x}(t_0)) + \mathbf{P}^{-1}\mathbf{x}(t_0)$$

$$\frac{\partial J}{\partial \mathbf{x}(t_0)} = 0$$ At the minimum (solution)

Time-dependence using a model

Assume a linearized model:

$$\mathbf{x}(n+1) = \mathbf{A}(n) \cdot \mathbf{x}(n) + \mathbf{L}(n) \cdot \mathbf{u}(n) + \mathbf{q}(n)$$

$\mathbf{x}(n)$ is the model state vector at time t_n

$\mathbf{A}(n)$ is the linearized transition matrix

$\mathbf{u}(n)$ is the forcing and boundary condition control parameters

$\mathbf{L}(n)$ transforms $\mathbf{u}(n)$ to model forcing and boundary conditions

$\mathbf{q}(n)$ represents the model errors

Data at time n are available through a sampling matrix $\mathbf{H}(n)$:

$$\mathbf{y}(n) = \mathbf{H}(n) \cdot \mathbf{x}(n) + \mathbf{r}(n)$$

In other words, the residuals are:

$$\mathbf{r}(n) = \mathbf{y}(n) - \mathbf{H}(n) \cdot \mathbf{x}(n)$$

Using the model, data timestep 1 samples the state vector:

$$\mathbf{y}(1) = \mathbf{H}(1) \cdot \mathbf{x}(1) + \mathbf{r}(1)$$

Or, using the dynamics, it samples the initial conditions and forcing:

$$\mathbf{y}(1) = \mathbf{H}(1) \cdot (A(0) \cdot x(0) + L(0) \cdot u(0) + q(0)) + r(1)$$

This can be continued forward in time to step 2, etc.

Perturbations to the data at an arbitrary time n depend on perturbations to the initial conditions according to the matrix:

$$\mathbf{y}(n) = \mathbf{H}(n) \cdot \mathbf{A}(n-1) \cdot \mathbf{A}(n-2) \cdot \quad \ldots \quad \cdot \mathbf{A}(1) \cdot \mathbf{A}(0) \cdot \mathbf{x}(0) \equiv G_1 \cdot \mathbf{x}(0)$$

Adjust the free parameters to make the model match the observations. Ignoring forcing, the objective function to be minimized is:

$$J = \mathbf{r}(n)^T \cdot \mathbf{C}_\varepsilon^{-1} \cdot \mathbf{r}(n) + \mathbf{x}(0)^T \cdot \mathbf{P}^{-1} \cdot \mathbf{x}(0)$$

which minimizes weighted residual variance and weighted variance of the modifications to the initial conditions.

- $\mathbf{P}$ is the covariance of uncertainty in the model initial state, $\mathbf{x}(0)$
- $\mathbf{R}$ is the covariance of uncertainty in the residuals, $\mathbf{r}(n)$

So, the cost function is:

$$J = (\mathbf{y}(n) - \mathbf{G}_1 \cdot \mathbf{x}(0))^T \cdot \mathbf{C}_\varepsilon^{-1} \cdot (\mathbf{y}(n) - \mathbf{G}_1 \cdot \mathbf{x}(0)) + \mathbf{x}(0)^T \cdot \mathbf{P}^{-1} \cdot \mathbf{x}(0)$$

The gradient is:

$$\frac{\partial J}{\partial \mathbf{x}(0)} = \mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1} \cdot (\mathbf{y}(n) - \mathbf{G}_1 \cdot \mathbf{x}(0)) + \mathbf{P}^{-1} \cdot \mathbf{x}(0)$$

Rewriting the first term on the RHS using the model:

$$= [\mathbf{H}(n) \cdot \mathbf{A}(n-1) \cdot \quad \ldots \quad \cdot \mathbf{A}(1) \cdot \mathbf{A}(0)]^T \mathbf{C}_\varepsilon^{-1} \cdot (\mathbf{y}(n) - \mathbf{G}_1 \cdot \mathbf{x}(0))$$

$$= \mathbf{A}^T(0) \cdot \mathbf{A}^T(1) \cdot \quad \ldots \quad \cdot \mathbf{A}^T(n-1) \cdot \mathbf{H}^T(n) \mathbf{C}_\varepsilon^{-1} \cdot (\mathbf{y}(n) - \mathbf{G}_1 \cdot \mathbf{x}(0))$$

which is the adjoint model integrated backward in time, starting from the backprojection (adjoint) of the observation(s).

The solution is the value that sets the gradient to zero

$$\mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1}(\mathbf{d}(t_1) - \mathbf{G}_1 \hat{\mathbf{x}}(t_0)) + \mathbf{P}^{-1}\hat{\mathbf{x}}(t_0) = 0$$

Collecting terms:

$$\underbrace{(\mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1}\mathbf{G}_1 + \mathbf{P}^{-1})}_{\text{“Hessian matrix”}}\hat{\mathbf{x}}(t_0) = \mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1}\mathbf{d}(t_1)$$

Formal solution (do the matrix inverse)

$$\hat{\mathbf{x}}(t_0) = (\mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1}\mathbf{G}_1 + \mathbf{P}^{-1})^{-1}\mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1}\mathbf{d}(t_1)$$

Above is model space inverse (overdetermined)
Below is data space inverse (underdetermined)

$$\hat{\mathbf{x}}(t_0) = \underbrace{\mathbf{P}\ \mathbf{G}_1^T}_{\text{Model-data Covariance}}\underbrace{(\mathbf{G}_1 \mathbf{P}\ \mathbf{G}_1^T + \mathbf{C}_\varepsilon)^{-1}\mathbf{d}(t_1)}_{\text{“Rotated data” Representer Coeffs}} \quad \text{(OA)}$$

Doing the inverse gives error estimates

$$\hat{\mathbf{P}}(t_0) = \mathbf{P} - \mathbf{P}\ \mathbf{H}_0^T(\mathbf{H}_0\mathbf{P}\ \mathbf{H}_0^T + \mathbf{C}_\varepsilon)^{-1}\mathbf{H}_0\mathbf{P}$$

Above is model space inverse (overdetermined)
Below is data space inverse (underdetermined)

$$\hat{\mathbf{P}}(t_0) = (\underbrace{\mathbf{G}_1^T\mathbf{C}_\varepsilon^{-1}\mathbf{G}_1 + \mathbf{P}^{-1}}_{\text{“Hessian matrix”}})^{-1}$$

So the Hessian matrix is the inverse of the MAP estimate uncertainty covariance matrix

Check: as $\mathbf{C}_\varepsilon \to \infty$ $\quad \hat{\mathbf{P}}(t_0) \to \mathbf{P}$

Linear Inverse Problems

$$\mathbf{d} = \mathbf{H}\mathbf{m}$$

- Under-determined:

$$\begin{bmatrix} d_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_N \end{bmatrix}$$

- Over-determined:

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} m_1 \end{bmatrix}$$

Pseudo-inverses

- Under-determined problem: $\begin{bmatrix} \hat{m}_1 \\ \hat{m}_2 \\ \vdots \\ \hat{m}_N \end{bmatrix} = P H^T \left(H P H^T + R \right)^{-1} \left[d_1 \right]$

- Over-determined problem: $\left[\hat{m}_1 \right] = \left(H^T R^{-1} H + P^{-1} \right)^{-1} H^T R^{-1} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}$

$\hat{m}_n$	d_n	H	P	R
=	=	=	=	=
Estimate of unkown parameter(s)	Known data	Sampling Matrix	Model parameter uncertainty covariance	Noise covariance

Under-determined problem

$$\begin{bmatrix} \hat{m}_1 \\ \hat{m}_2 \\ \vdots \\ \hat{m}_N \end{bmatrix} = P H^T \left(H P H^T + R \right)^{-1} \left[d_1 \right]$$

$$H = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \qquad P = I \qquad R = \begin{bmatrix} \sigma^2 & & & \\ & \sigma^2 & & \\ & & \ddots & \\ & & & \sigma^2 \end{bmatrix} = \sigma^2 I$$

$$\therefore \begin{bmatrix} \hat{m}_1 \\ \hat{m}_2 \\ \vdots \\ \hat{m}_N \end{bmatrix} = I \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} I \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \sigma^2 I \right)^{-1} \left[d_1 \right]$$

$$= \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \left(N + \sigma^2 \right)^{-1} \left[d_1 \right]$$

Over-determined problem

$$\left[\hat{m}_1\right] = \left(H^T R^{-1} H + P^{-1}\right)^{-1} H^T R^{-1} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}$$

$$H = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \qquad P = I \qquad R = \begin{bmatrix} \sigma^2 & & & \\ & \sigma^2 & & \\ & & \ddots & \\ & & & \sigma^2 \end{bmatrix} = \sigma^2 I$$

$$\therefore \left[\hat{m}_1\right] = \left(\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \frac{I}{\sigma^2} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + I \right)^{-1} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \frac{I}{\sigma^2} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}$$

$$= \left(\frac{N}{\sigma^2} + 1 \right)^{-1} \frac{1}{\sigma^2} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \qquad = \left(N + \sigma^2\right)^{-1} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}$$

Can avoid the inverse by solving for just the single vector:

model space (overdetermined)

$$(\mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1} \mathbf{G}_1 + \mathbf{P}^{-1}) \hat{\mathbf{x}}(t_0) = \mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1} \mathbf{d}(t_1)$$

data space (underdetermined) is more tricky:

$$\hat{\mathbf{x}}(t_0) = \mathbf{P}\ \mathbf{G}_1^T (\mathbf{G}_1 \mathbf{P}\ \mathbf{G}_1^T + \mathbf{C}_\varepsilon)^{-1} \mathbf{d}(t_1)$$

Can break it up into two pieces

$$\hat{\mathbf{x}}(t_0) = \mathbf{P}\ \mathbf{G}_1^T \hat{\boldsymbol{\beta}}$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{G}_1 \mathbf{P}\ \mathbf{G}_1^T + \mathbf{C}_\varepsilon)^{-1} \mathbf{d}(t_1)$$

"Rotated data"

Representer Coeffs

Instead of doing the formal inversion, solve

$$(\mathbf{G}_1 \mathbf{P}\ \mathbf{G}_1^T + \mathbf{C}_\varepsilon)\ \hat{\boldsymbol{\beta}} = \mathbf{d}(t_1)$$

Can't solve if don't have the matrix all at once. Since G includes the TLM, and G-transpose includes the adjoint model, need methods that operate with one row or column at a time. "Row-action methods" e.g., Conjugate-gradient.

Model space (overdetermined)

$$(\mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1} \mathbf{G}_1 + \mathbf{P}^{-1}) \hat{\mathbf{x}}(t_0) = \mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1} \mathbf{d}(t_1)$$

Newton's method (one-step) solution (requires Hessian inverse)

$$\hat{\mathbf{x}}(t_0) = (\mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1} \mathbf{G}_1 + \mathbf{P}^{-1})^{-1} \mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1} \mathbf{d}(t_1)$$

This can be viewed as a rotation and scaling of the gradient at x=0

$$\frac{\partial J}{\partial \mathbf{x}(t_0)} = \mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1} (\mathbf{d}(t_1) - \mathbf{G}_1 \mathbf{x}(t_0)) + \mathbf{P}^{-1} \mathbf{x}(t_0)$$

"4DVAR" or "Adjoint method"

Since can't afford to do the inverse and go to the solution in one iteration, try to go down the gradient

$$\frac{\partial J}{\partial \mathbf{x}(t_0)} = \mathbf{G}_1^T \mathbf{C}_\varepsilon^{-1}(\mathbf{d}(t_1) - \mathbf{G}_1 \mathbf{x}(t_0)) + \mathbf{P}^{-1}\mathbf{x}(t_0)$$

Make a series of guesses for x:

$$\mathbf{x}(t_0)_1 = \mathbf{x}(t_0)_0 + \mathbf{K}_0 \frac{\partial J}{\partial \mathbf{x}(t_0)}$$

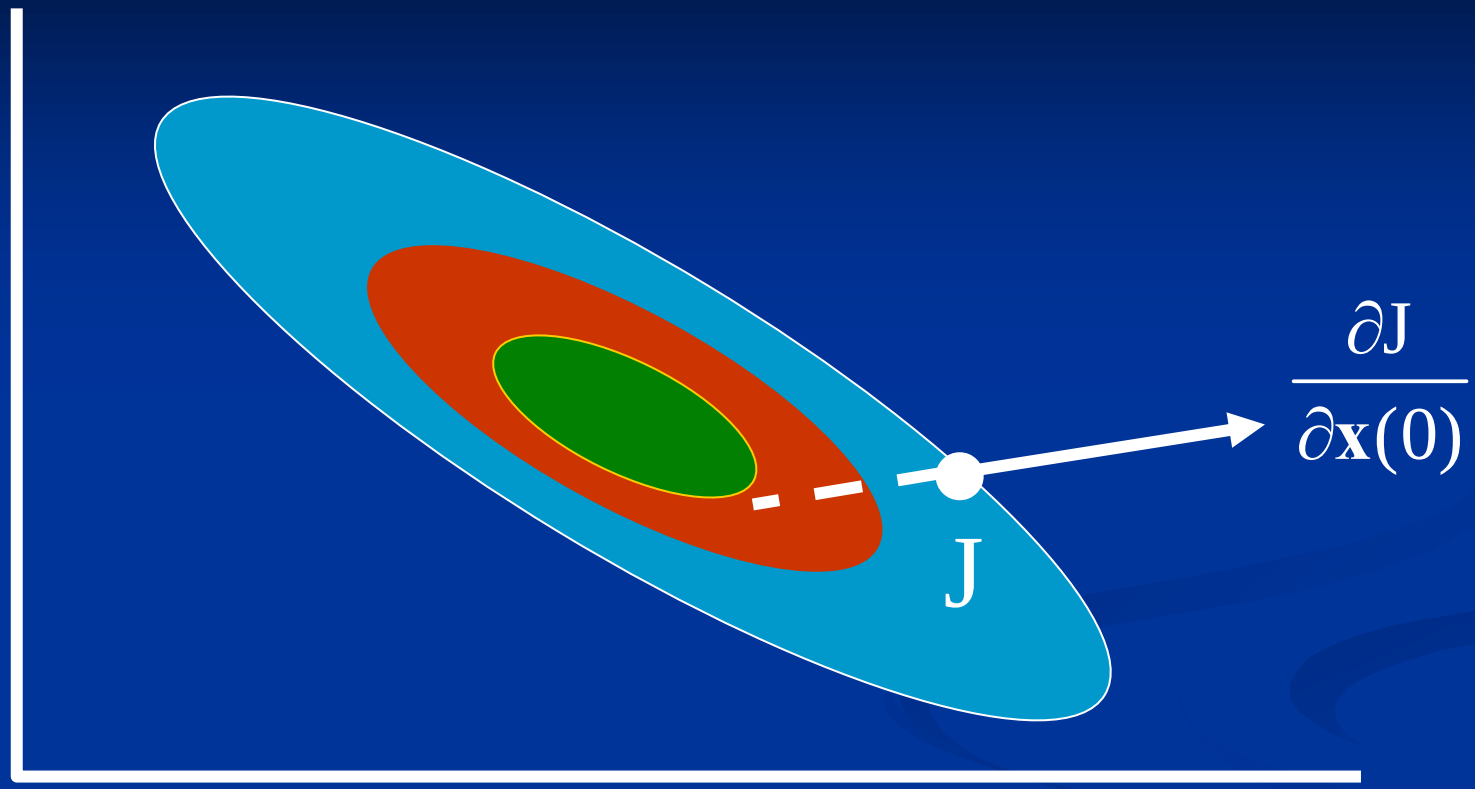
Where K is a matrix that rotates and scales the descent direction. Ideally, this would be the inverse of the Hessian, but various approximate techniques choose this differently. The simplest way is to have it be a scalar, determined by a line search in the down-gradient direction.

Pre-conditioning is the use of matrices K that accelerate the descent. Approximation of the Hessian matrix is an option, and can be built up during a set of iterations. The control parameter covariance matrix, P, is a reasonable start

$$\mathbf{x}(t_0)_1 = \mathbf{x}(t_0)_0 + a_0 \mathbf{P} \frac{\partial J}{\partial \mathbf{x}(t_0)}$$

Subsequent iterations take account of previous descent directions. Many algorithms are available; ROMS uses a proprietary algorithm.

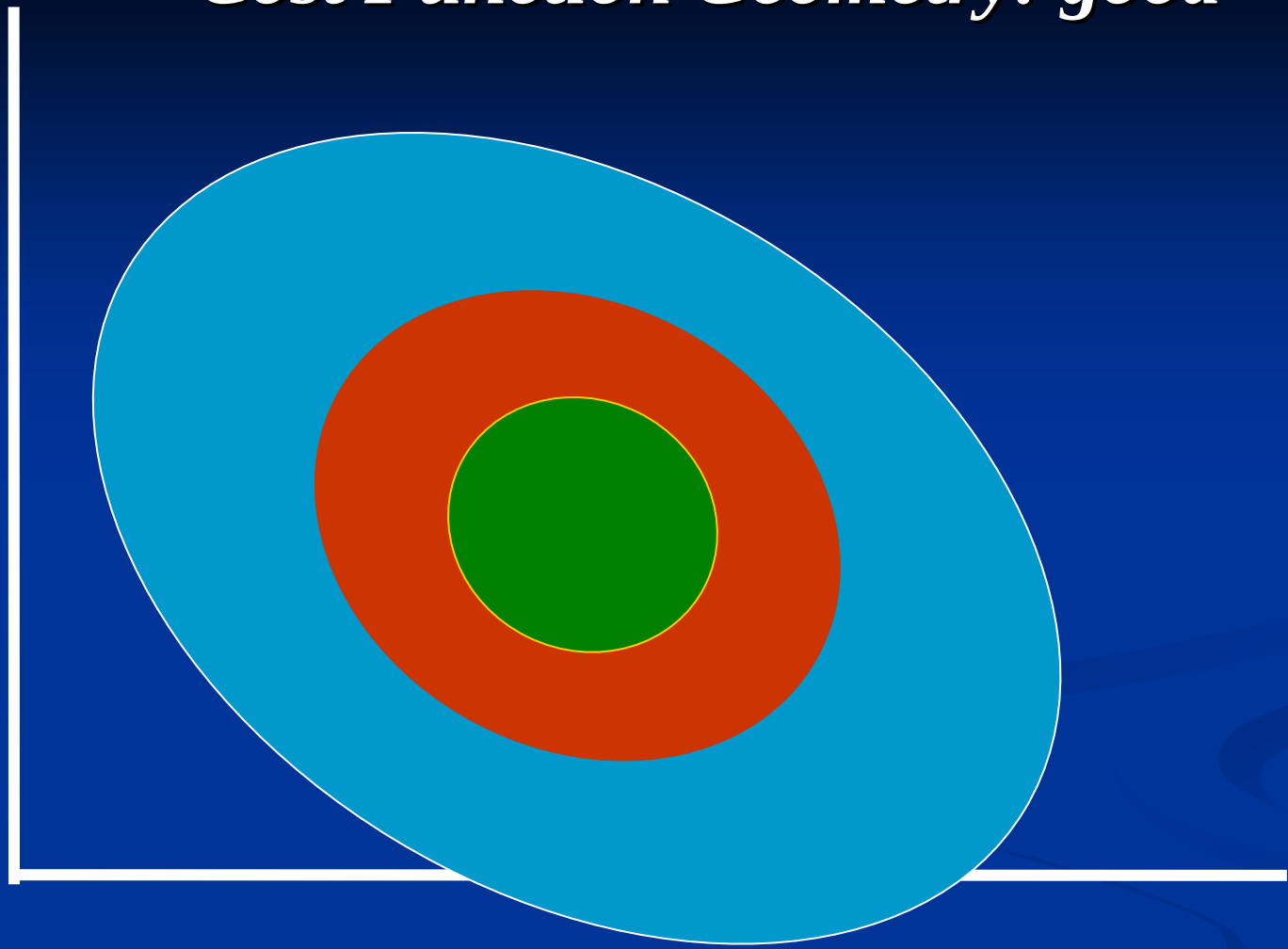
Cost Function Minimization



Minimize J by iterating

- *Forward model run* to evaluate J
- *Adjoint model run* to compute gradient of J

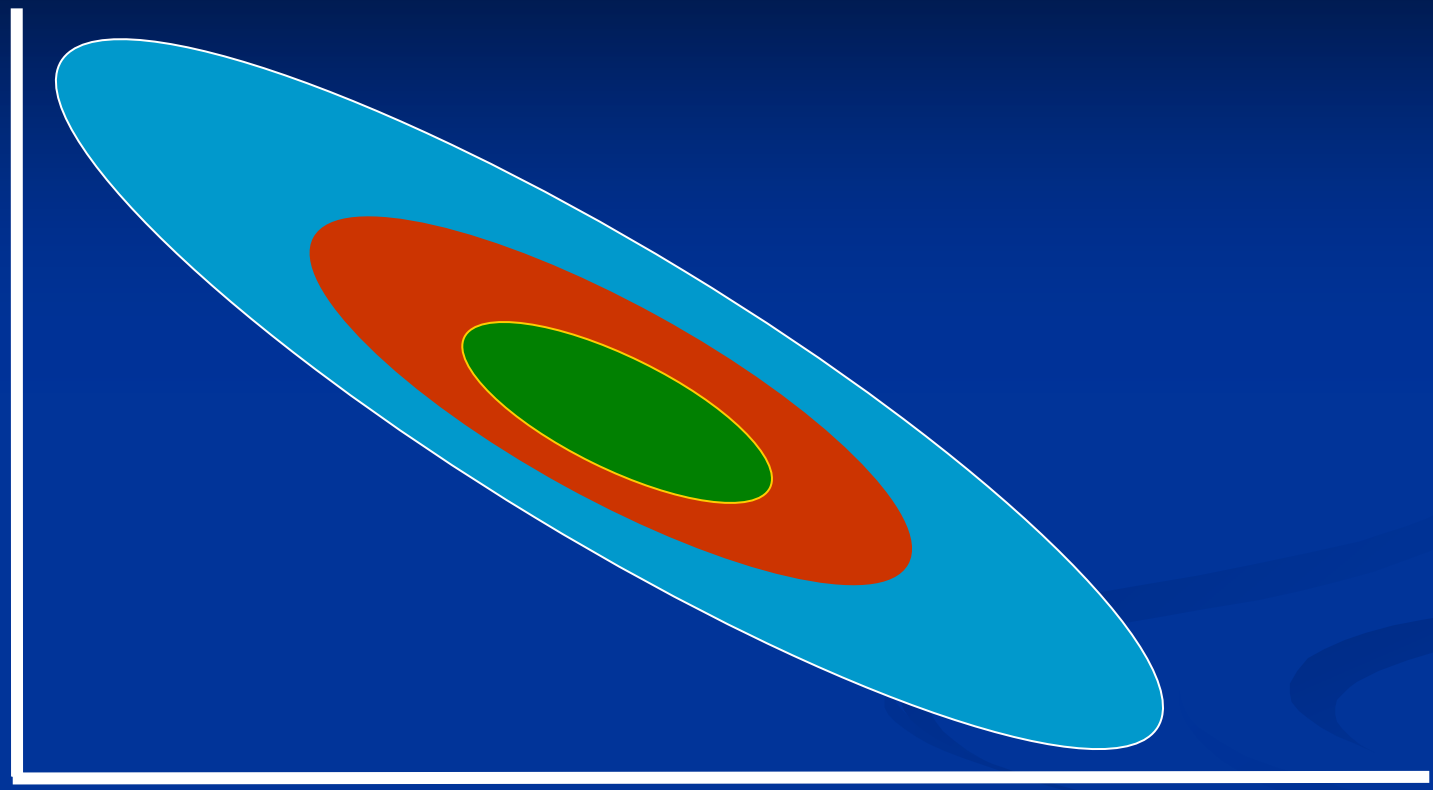
Cost Function Geometry: good



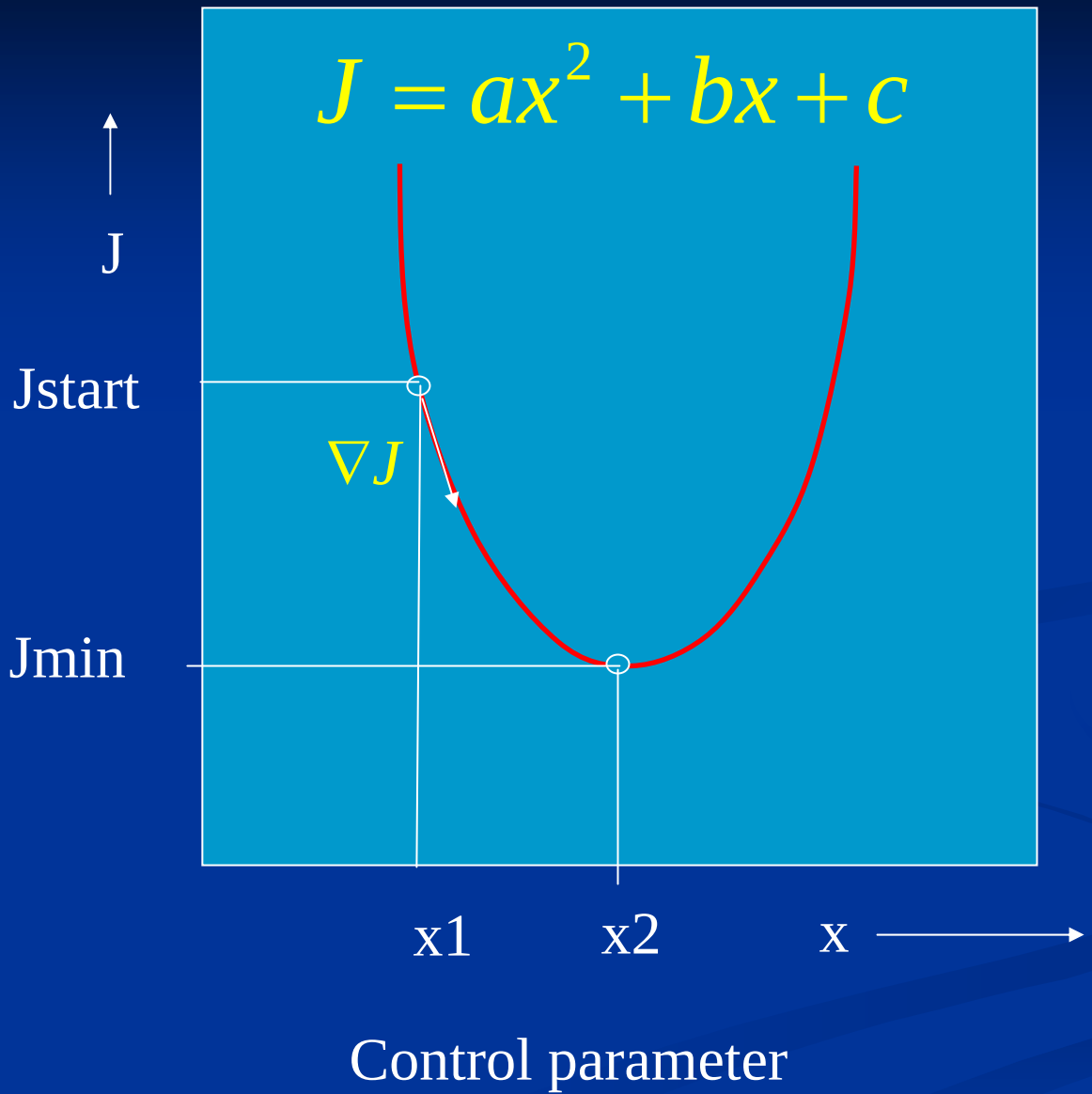
Strong gradients can make for slow convergence

- Pre-conditioning should make the cost function more circular

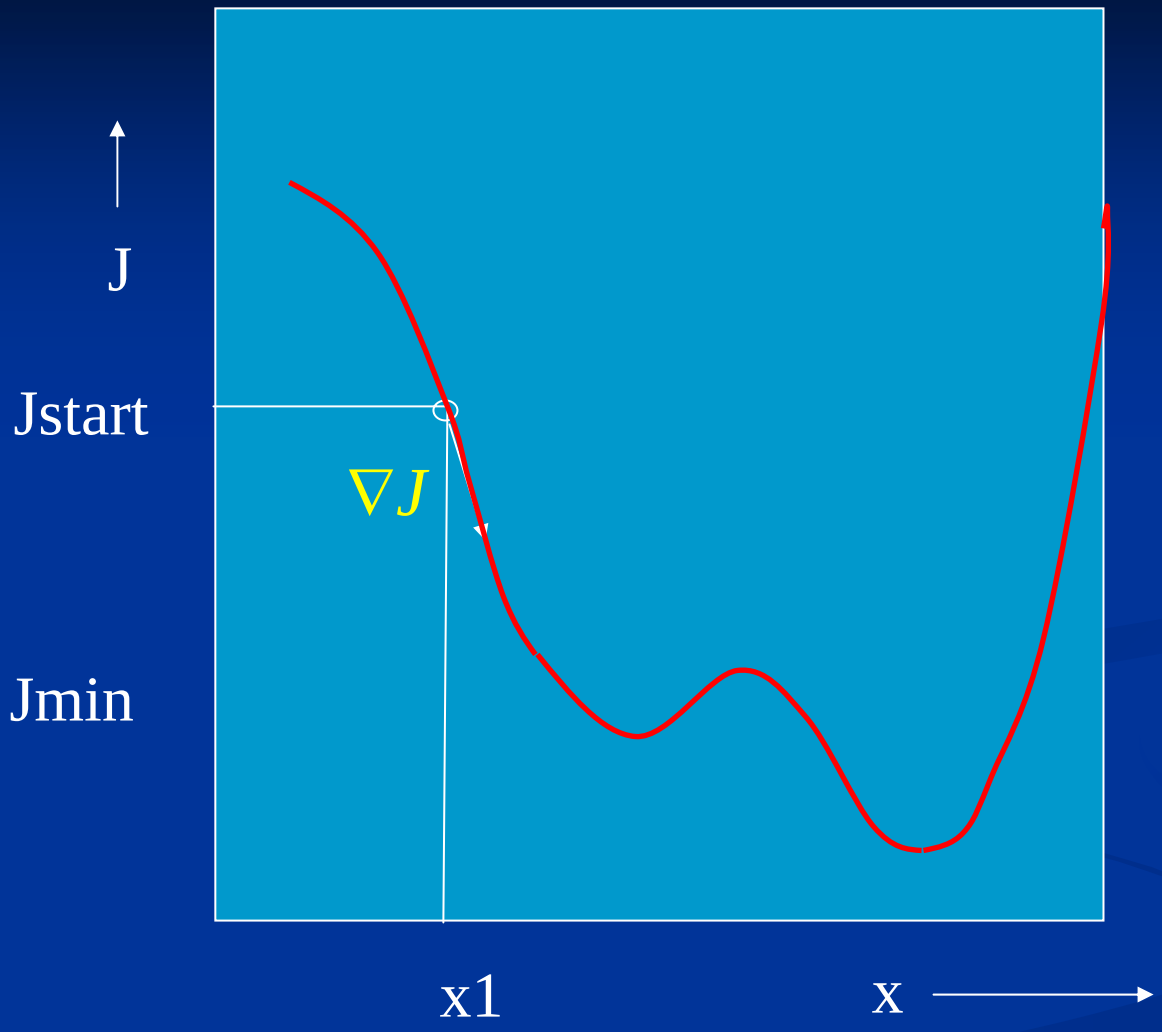
Cost Function Geometry: bad



Line search: minimize J along gradient



Success depends on linearity



data space (underdetermined)

$$\hat{\mathbf{x}}(t_0) = \mathbf{P}\,\mathbf{G}_1^T(\mathbf{G}_1\mathbf{P}\,\mathbf{G}_1^T + \mathbf{C}_\varepsilon)^{-1}\mathbf{d}(t_1)$$

Solve in two pieces, (repeat from before)

$$\hat{\mathbf{x}}(t_0) = \mathbf{P}\,\mathbf{G}_1^T\hat{\boldsymbol{\beta}}$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{G}_1\mathbf{P}\,\mathbf{G}_1^T + \mathbf{C}_\varepsilon)^{-1}\mathbf{d}(t_1)$$

Need to solve the problem:

$$(\mathbf{G}_1\mathbf{P}\,\mathbf{G}_1^T + \mathbf{C}_\varepsilon)\;\boldsymbol{\beta} = \mathbf{d}(t_1)$$

Can solve this using iteration as well;

$$(\mathbf{G}_1\mathbf{P}\,\mathbf{G}_1^T + \mathbf{C}_\varepsilon) \equiv \mathbf{S}$$ (stabilized representer matrix)

$$J_\beta = (\mathbf{d} - \mathbf{S}\boldsymbol{\beta})^T(\mathbf{d} - \mathbf{S}\boldsymbol{\beta})$$

$$\frac{\partial J}{\partial \boldsymbol{\beta}} = \mathbf{S}^T(\mathbf{d} - \mathbf{S}\boldsymbol{\beta})$$

Indirect representer method (Egbert)

$$\frac{\partial J}{\partial \boldsymbol{\beta}} = \mathbf{S}^T(\mathbf{d} - \mathbf{S}\boldsymbol{\beta}) = (\mathbf{G}_1 \mathbf{P}\ \mathbf{G}_1^T + \mathbf{C}_\varepsilon)^T(\mathbf{d} - \mathbf{S}\boldsymbol{\beta})$$

Solve with gradient descent iteration

$$\boldsymbol{\beta}_2 = \boldsymbol{\beta}_1 + \mathbf{K}_1 \frac{\partial J}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta}_1 + \mathbf{K}_1(\mathbf{G}_1 \mathbf{P}\ \mathbf{G}_1^T + \mathbf{C}_\varepsilon)^T(\mathbf{d} - \mathbf{S}\boldsymbol{\beta}_1)$$

Once have a solution, plug it in to get controls:

$$\hat{\mathbf{x}}(t_0) = \mathbf{P}\ \mathbf{G}_1^T \hat{\boldsymbol{\beta}}$$

Make data estimate from estimated controls

$$\hat{\mathbf{d}} = \mathbf{G}_1 \hat{\mathbf{x}}(t_0) = \mathbf{G}_1 \mathbf{P}\ \mathbf{G}_1^T \hat{\boldsymbol{\beta}}$$

Aside: Smoothing tricks

- *The covariance, **P**, has off-diagonal terms, and can be expensive and annoying to compute.*
- *Derber and Rosati (1989) for P; Bennett (2002) for inv(P)*

$f(\mathbf{m}) = \mathbf{m}^T \mathbf{P}^{-1} \mathbf{m}$ P is smoothing, so inv(P) is roughening

If m is a 2-d field, and P is approximately Gaussian in x and y, can approximate the norm by $f(\mathbf{m}) \approx a\mathbf{m}^T\mathbf{m} + b(\mathbf{D}_2\mathbf{m})^T(\mathbf{D}_2\mathbf{m})$

where $\mathbf{D}_2$ is the second derivative operator on m

$f(\mathbf{m}) = \mathbf{P}\mathbf{m}$ P is smoothing. If P is a Gaussian, it can be approximated by solving the diffusion problem with m as an initial condition.

Other topics

- Sensitivity and nonlinearity
- Re-linearization: e.g. S4DVAR vs. IS4DVAR
  - Inner loops vs. outer loops
- Kalman filters: extended, ensemble, particle, etc.
- Data functionals for constraints (“bring the model to the data”) e.g.:
  - Constrain monthly spatial averages of the model to WOA atlas climatology (Levitus)
  - Constrain averages from the first month to match the same month in subsequent years

Kalman filter: partitioned inverse in time

Only do inverse for data within a short time range,
so inverse is feasible in data space, just sample:

$$\hat{\mathbf{x}}(t_0) = \mathbf{P}\ \mathbf{H}_0^T(\mathbf{H}_0\mathbf{P}\ \mathbf{H}_0^T + \mathbf{C}_\varepsilon)^{-1}\mathbf{d}(t_0)$$

Also compute uncertainty of control parameter estimate:

$$\hat{\mathbf{P}}(t_0) = \mathbf{P} - \mathbf{P}\ \mathbf{H}_0^T(\mathbf{H}_0\mathbf{P}\ \mathbf{H}_0^T + \mathbf{C}_\varepsilon)^{-1}\mathbf{H}_0\mathbf{P}$$

Propagate estimate and uncertainty forward to next time range:

$$\tilde{x}(t_1) = \mathbf{A}\hat{x}(t_0)$$

$$\tilde{\mathbf{P}}(t_1) \equiv \tilde{\mathbf{P}}_1 = \mathbf{A}\hat{\mathbf{P}}(t_0)\mathbf{A}^T + \mathbf{Q}$$

Note: includes model errors

Invert again with new data

$$\hat{\mathbf{x}}(t_1) = \tilde{\mathbf{x}}(t_1) + \tilde{\mathbf{P}}_1\mathbf{H}_1^T(\mathbf{H}_1\tilde{\mathbf{P}}_1\mathbf{H}_1^T + \mathbf{C}_\varepsilon)^{-1}(\mathbf{d}(t_1) - \mathbf{H}_1\tilde{\mathbf{x}}(t_1))$$

Kalman gain — innovation

Kalman filter: approximations

Not feasible to estimate or propagate the P matrix

$$\hat{\mathbf{P}}(t_0) = \mathbf{P} - \mathbf{P}\ \mathbf{H}_0^T(\mathbf{H}_0\mathbf{P}\ \mathbf{H}_0^T + \mathbf{C}_\varepsilon)^{-1}\mathbf{H}_0\mathbf{P}$$

$$\tilde{\mathbf{P}}(t_1) = \mathbf{A}\hat{\mathbf{P}}(t_0)\mathbf{A}^T + \mathbf{Q}$$

Approximate approaches:

3DVAR: approximate P and don't change it

$$\hat{\mathbf{x}}(t_1) = \tilde{\mathbf{x}}(t_1) + \mathbf{P}\ \mathbf{H}_1^T(\mathbf{H}_1\mathbf{P}\mathbf{H}_1^T + \mathbf{C}_\varepsilon)^{-1}(\mathbf{d}(t_1) - \mathbf{H}_1\tilde{\mathbf{x}}(t_1))$$

Reduced State Space
Ensemble KF: sample P in a subspace
Other refinements: particle KF, etc.

Thank you!

Whew!