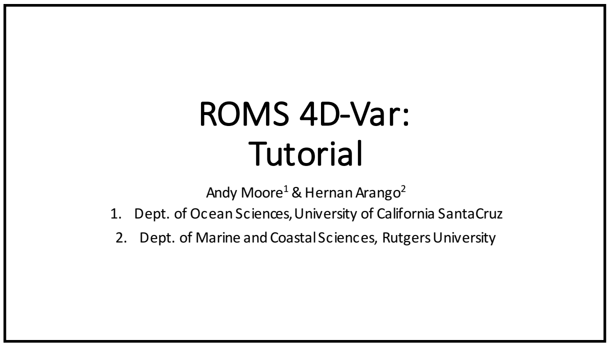
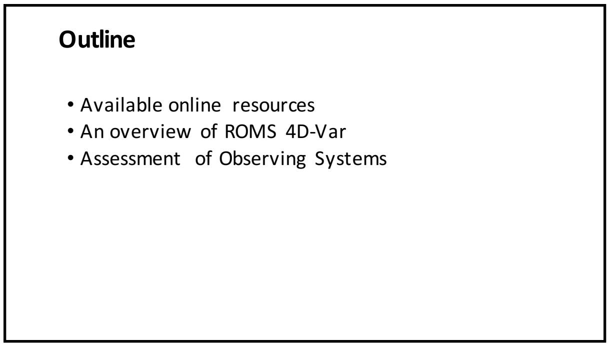


ROMS 4D-Var: Tutorial

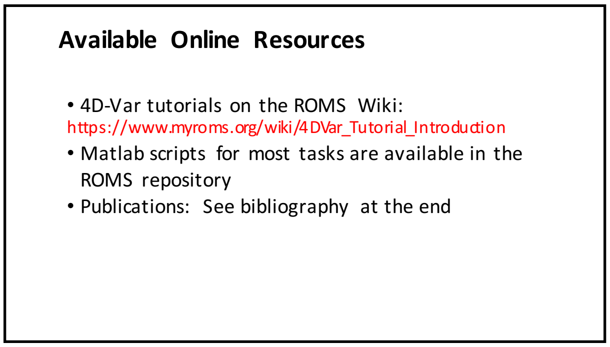
Andy Moore[1] & Hernan Arango[2]

1. Dept. of Ocean Sciences, University of California Santa Cruz
2. Dept. of Marine and Coastal Sciences, Rutgers University



Outline

- Available online resources
- An overview of ROMS 4D-Var
- Assessment of Observing Systems



Available Online Resources

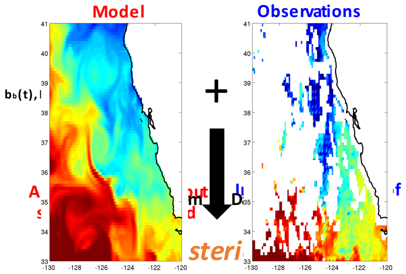
- 4D-Var tutorials on the ROMS Wiki: https://www.myroms.org/wiki/4DVar_Tutorial_Introduction
- Matlab scripts for most tasks are available in the ROMS repository
- Publications: See bibliography at the end

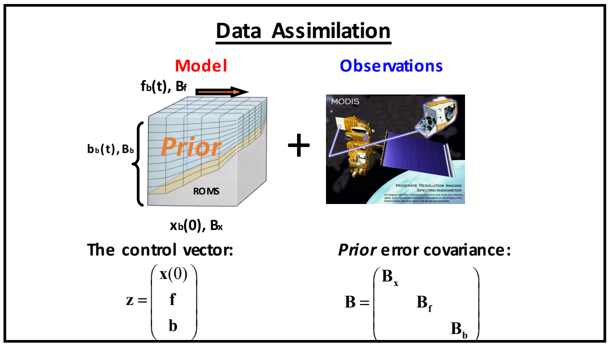
An Overview of ROMS 4D-Var

- Basics of data assimilation
- Important ingredients of ROMS 4D-Var
- Covariance models
- Preconditioning
- Conjugate gradients
- New developments

Reverend Thomas Bayes
(1702-1761)

Data Assimilation





Data Assimilation

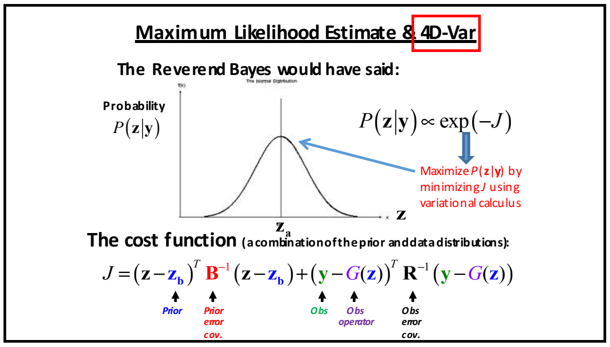
Model **Observations**

The control vector:

$$\mathbf{z} = \begin{pmatrix} \mathbf{x}(0) \\ \mathbf{f} \\ \mathbf{b} \end{pmatrix}$$

Prior error covariance:

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_x & & \\ & \mathbf{B}_f & \\ & & \mathbf{B}_b \end{pmatrix}$$



Maximum Likelihood Estimate & 4D-Var

The Reverend Bayes would have said:

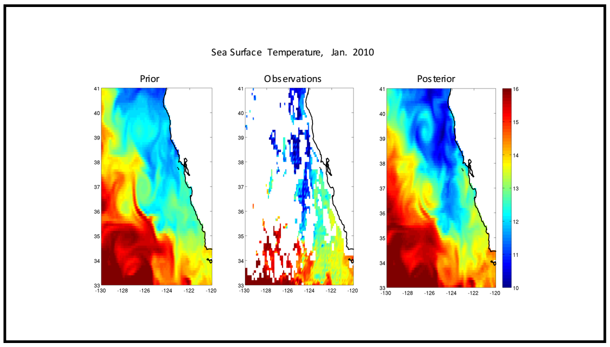
$$P(\mathbf{z}|\mathbf{y}) \propto \exp(-J)$$

Maximize $P(\mathbf{z}|\mathbf{y})$ by minimizing J using variational calculus

The cost function (a combination of the prior and data distributions):

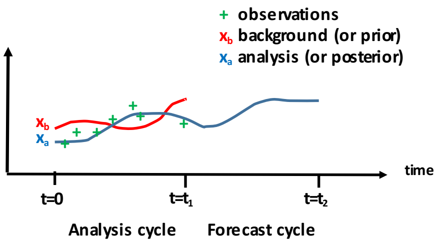
$$J = (\mathbf{z}-\mathbf{z}_b)^T \mathbf{B}^{-1} (\mathbf{z}-\mathbf{z}_b) + (\mathbf{y}-G(\mathbf{z}))^T \mathbf{R}^{-1} (\mathbf{y}-G(\mathbf{z}))$$

Prior *Prior error cov.* *Obs* *Obs operator* *Obs error cov.*



Sea Surface Temperature, Jan. 2010

4-dimensional variational (4D-Var) data assimilation



Notation & Nomenclature

$$\mathbf{x}=\begin{bmatrix}\mathbf{T}\\ \mathbf{S}\\ \mathbf{u}\\ \mathbf{v}\\ \zeta\end{bmatrix}\quad \mathbf{z}=\begin{bmatrix}\mathbf{x}(0)\\ \mathbf{f}(t)\\ \mathbf{b}(t)\\ \boldsymbol{\eta}(t)\end{bmatrix}\quad \mathbf{y}=\begin{bmatrix}y_1\\ \vdots\\ \vdots\\ y_N\end{bmatrix}\quad \mathbf{d}=\left(\mathbf{y}-G(\mathbf{x}^{\mathrm{b}})\right)$$

State vector — Control vector — Observation vector

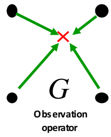
Prior

Innovation vector

η(t) = Correction for model error

η(t)=0 : Strong constraint

η(t)≠0 : Weak constraint



In 4D-Var, G includes NL ROMS

The Linear Optimal Estimate

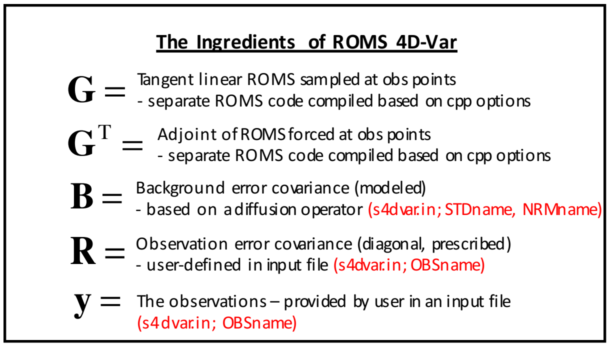
Analysis: $$\mathbf{z}_{\mathbf{a}}=\mathbf{z}_{\mathbf{b}}+\mathbf{Kd}$$

Gain (dual) (#if defined W4DPSAS & defined RPCG):

$$\mathbf{K}=\mathbf{BG}^{\mathrm{T}}(\mathbf{GBG}^{\mathrm{T}}+\mathbf{R})^{-1}$$

Gain (primal) (#ifdef IS4DVAR):

$$\mathbf{K}=(\mathbf{B}^{-1}+\mathbf{G}^{T}\mathbf{R}^{-1}\mathbf{G})^{-1}\mathbf{G}^{T}\mathbf{R}^{-1}$$



The Ingredients of ROMS 4D-Var

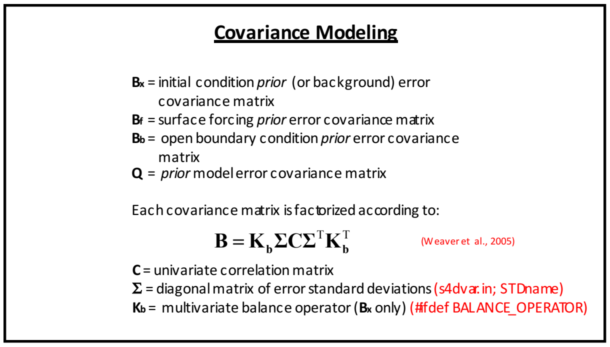
$\mathbf{G} =$ Tangent linear ROMS sampled at obs points
- separate ROMS code compiled based on cpp options

$\mathbf{G}^{\mathrm{T}} =$ Adjoint of ROMS forced at obs points
- separate ROMS code compiled based on cpp options

$\mathbf{B} =$ Background error covariance (modeled)
- based on a diffusion operator (s4dvar.in; STDname, NRMname)

$\mathbf{R} =$ Observation error covariance (diagonal, prescribed)
- user-defined in input file (s4dvar.in; OBSname)

$\mathbf{y} =$ The observations – provided by user in an input file (s4dvar.in; OBSname)



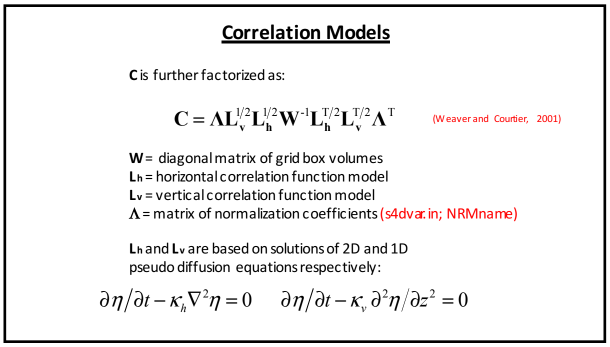
Covariance Modeling

$\mathbf{B}_x$ = initial condition *prior* (or background) error covariance matrix
$\mathbf{B}_f$ = surface forcing *prior* error covariance matrix
$\mathbf{B}_b$ = open boundary condition *prior* error covariance matrix
$\mathbf{Q}$ = *prior* model error covariance matrix

Each covariance matrix is factorized according to:

$$\mathbf{B} = \mathbf{K}_{\mathbf{b}}\boldsymbol{\Sigma}\mathbf{C}\boldsymbol{\Sigma}^{\mathrm{T}}\mathbf{K}_{\mathbf{b}}^{\mathrm{T}}$$ (Weaver et al., 2005)

$\mathbf{C}$ = univariate correlation matrix
$\boldsymbol{\Sigma}$ = diagonal matrix of error standard deviations (s4dvar.in; STDname)
$\mathbf{K}_b$ = multivariate balance operator ($\mathbf{B}_x$ only) (#ifdef BALANCE_OPERATOR)



Correlation Models

$\mathbf{C}$ is further factorized as:

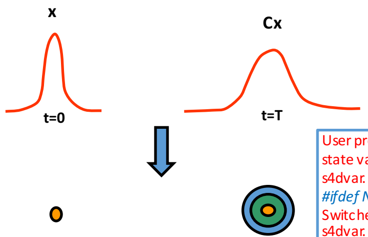
$$\mathbf{C} = \boldsymbol{\Lambda}\mathbf{L}_{\mathbf{v}}^{1/2}\mathbf{L}_{\mathbf{h}}^{1/2}\mathbf{W}^{-1}\mathbf{L}_{\mathbf{h}}^{\mathrm{T}/2}\mathbf{L}_{\mathbf{v}}^{\mathrm{T}/2}\boldsymbol{\Lambda}^{\mathrm{T}}$$ (Weaver and Courtier, 2001)

$\mathbf{W}$ = diagonal matrix of grid box volumes
$\mathbf{L}_h$ = horizontal correlation function model
$\mathbf{L}_v$ = vertical correlation function model
$\boldsymbol{\Lambda}$ = matrix of normalization coefficients (s4dvar.in; NRMname)

$\mathbf{L}_h$ and $\mathbf{L}_v$ are based on solutions of 2D and 1D pseudo diffusion equations respectively:

$$\partial\eta/\partial t - \kappa_h \nabla^2 \eta = 0 \qquad \partial\eta/\partial t - \kappa_v\, \partial^2\eta/\partial z^2 = 0$$

Correlation Models



User provides *L* for each state variable:
s4dvar.in; Hdecay, Vdecay
#ifdef NORMALIZATION
Switches to compute Λ are also in s4dvar.in; LdefNRM, LwrtNRM, Cnorm

Correlation length, *L*: $L^2 \approx 2\kappa T$

Covariance Modeling

$$\mathbf{C} = \mathbf{\Lambda} \mathbf{L}_{\mathbf{v}}^{1/2} \mathbf{L}_{\mathbf{h}}^{1/2} \mathbf{W}^{-1} \mathbf{L}_{\mathbf{h}}^{T/2} \mathbf{L}_{\mathbf{v}}^{T/2} \mathbf{\Lambda}^{T}$$

Λ ensures that the range of **C** is ±1.

Suppose that **x** is divided into a balanced and unbalanced contribution: $\mathbf{x}=\mathbf{x}_b+\mathbf{x}_u$

Examples of balance: geostrophy, hydrostatic

$$(\mathbf{B}_x)_{\mathbf{u}} = \mathbf{\Sigma} \mathbf{C} \mathbf{\Sigma}^T$$

$$\mathbf{B}_x = \mathbf{K}_b (\mathbf{B}_x)_{\mathbf{u}} \mathbf{K}_b^T$$

The Linear Optimal Estimate

Analysis: $\mathbf{z}_a = \mathbf{z}_b + \mathbf{K}\mathbf{d}$

Gain (dual) (#if defined W4DPSAS & defined RPCG):

$$\mathbf{K} = \mathbf{B}\mathbf{G}^T \boxed{(\mathbf{G}\mathbf{B}\mathbf{G}^T + \mathbf{R})^{-1}}$$

Gain (primal) (#ifdef IS4DVAR):

$$\mathbf{K} = \boxed{(\mathbf{B}^{-1} + \mathbf{G}^T \mathbf{R}^{-1} \mathbf{G})^{-1}} \mathbf{G}^T \mathbf{R}^{-1}$$

Matrix inverse estimated using a preconditioned conjugate gradient method

Preconditioning

Analysis: $$\mathbf{z}_a = \mathbf{z}_b + \mathbf{Kd}$$

Gain (dual) (#if defined W4DPSAS & defined RPCG):

$$\mathbf{K} = \mathbf{BG}^T\left(\mathbf{R}^{-1}\mathbf{GBG}^T + \mathbf{I}\right)^{-1}\mathbf{R}^{-1}$$

Gain (primal) (#ifdef IS4DVAR):

$$\mathbf{K} = \mathbf{B}^{1/2}\left(\mathbf{I} + \mathbf{B}^{-T/2}\mathbf{G}^T\mathbf{R}^{-1}\mathbf{GB}^{-1/2}\right)^{-1}\mathbf{B}^{1/2}\mathbf{G}^T\mathbf{R}^{-1}$$

Lanczos Formulation of Conjugate Gradient Method

Dual form (#if defined W4DPSAS & defined RPCG):

$$\left(\mathbf{R}^{-1}\mathbf{GBG}^T + \mathbf{I}\right) \simeq \mathbf{V}_m\mathbf{T}_m\mathbf{V}_m^T\mathbf{GBG}^T$$

s4dvar.in; MODname

Matrix of Lanczos vectors

$$\boxed{\tilde{\mathbf{V}}_m = \mathbf{G}^T\mathbf{V}_m}$$

ocean.in; ADJname

Primal form (#ifdef IS4DVAR):

$$\left(\mathbf{I} + \mathbf{B}^{-T/2}\mathbf{G}^T\mathbf{R}^{-1}\mathbf{GB}^{-1/2}\right) \simeq \tilde{\mathbf{V}}_m\mathbf{T}_m\tilde{\mathbf{V}}_m^T$$

You don't care about any of this UNLESS you plan to use the 4D-Var post-processing tools, then you must to save the appropriate output files!!

Summary of ROMS 4D-Var Input and Output Files

Input files:

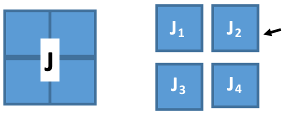
- INIname – background initial conditions (ocean.in)
- STDname – background error stds (s4dvar.in)
- NRMname – background error covariance normalization factors (s4dvar.in)
- OBSname – observations (s4dvar.in)

Output files:

- FWDname – background circulation estimate history file (ocean.in)
- HISname – analysis circulation estimate history file (ocean.in)
- ADJname – Lanczos vectors for primal 4D-Var (ocean.in)
- MODname – Diagnostics for 4D-Var & Lanczos vectors for dual 4D-Var (s4dvar.in)

New Developments

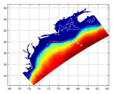
- DART-ROMS: Community code Ensemble Kalman Filter for ROMS
- Long window 4D-Var
- DD-4D-Var (NASDAC – Arcucci et al.)



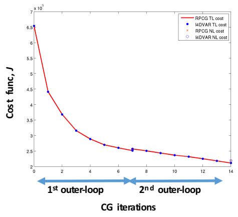
Add boundary conditions for each tile to cost function.

Time interval can be treated in the same way.

Dual 4D-Var vs Primal 4D-Var



Mid-Atlantic Bight, 7km, 2014 test case
2 outer-loops
7 inner-loops
3 day windows
SSH, SST, in situ, HF radar obs

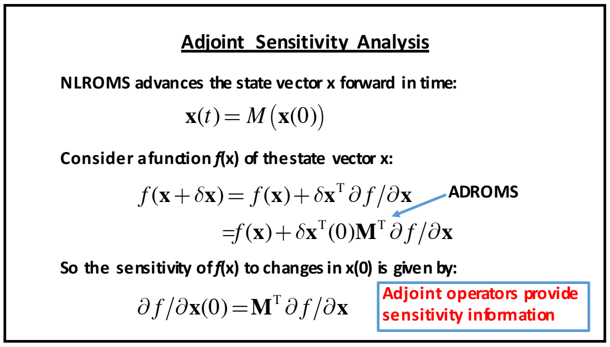


$$J=\left(\mathbf{z}-\underset{\substack{\uparrow\\ \text{Prior}}}{\mathbf{z}_b}\right)^T \underset{\substack{\uparrow\\ \text{Prior}\\ \text{error}\\ \text{cov.}}}{\mathbf{B}^{-1}}\left(\mathbf{z}-\mathbf{z}_b\right)+\left(\underset{\substack{\uparrow\\ \text{Obs}}}{\mathbf{y}}-\underset{\substack{\uparrow\\ \text{Obs}\\ \text{operator}}}{G}(\mathbf{z})\right)^T \underset{\substack{\uparrow\\ \text{Obs}\\ \text{error}\\ \text{cov.}}}{\mathbf{R}^{-1}}\left(\mathbf{y}-G(\mathbf{z})\right)$$

Dual 4D-Var is ~15-25% faster than primal 4D-Var in ROMS
Dual 4D-Var has more post-processing utility in ROMS
Only Dual 4D-Var supports the weak constraint option

Assessment of Observing Systems

- Adjoints for sensitivity analysis
- Quantifying observation impacts on analyses & forecasts
- Examples
- Practical matters
- Array modes



Adjoint Sensitivity Analysis

NLROMS advances the state vector x forward in time:

$$\mathbf{x}(t) = M\left(\mathbf{x}(0)\right)$$

Consider a function *f*(x) of the state vector x:

$$f(\mathbf{x}+\delta\mathbf{x}) = f(\mathbf{x}) + \delta\mathbf{x}^{\mathrm{T}}\,\partial f/\partial\mathbf{x}$$

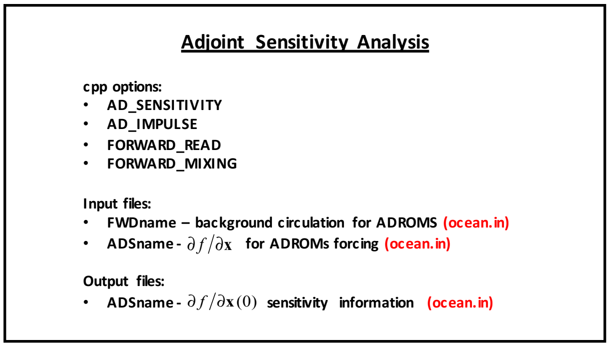
ADROMS

$$= f(\mathbf{x}) + \delta\mathbf{x}^{\mathrm{T}}(0)\mathbf{M}^{\mathrm{T}}\,\partial f/\partial\mathbf{x}$$

So the sensitivity of *f*(x) to changes in x(0) is given by:

$$\partial f/\partial\mathbf{x}(0) = \mathbf{M}^{\mathrm{T}}\,\partial f/\partial\mathbf{x}$$

Adjoint operators provide sensitivity information



Adjoint Sensitivity Analysis

cpp options:

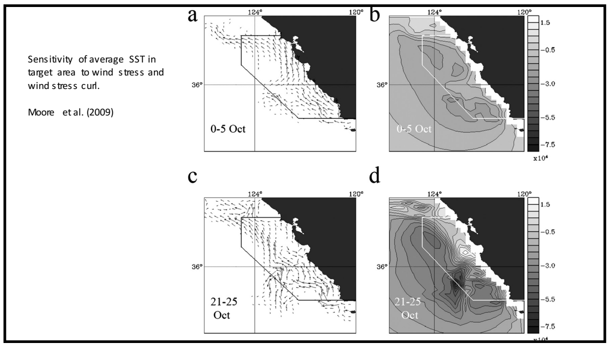
- AD_SENSITIVITY
- AD_IMPULSE
- FORWARD_READ
- FORWARD_MIXING

Input files:

- FWDname – background circulation for ADROMS (ocean.in)
- ADSname - $\partial f/\partial\mathbf{x}$ for ADROMs forcing (ocean.in)

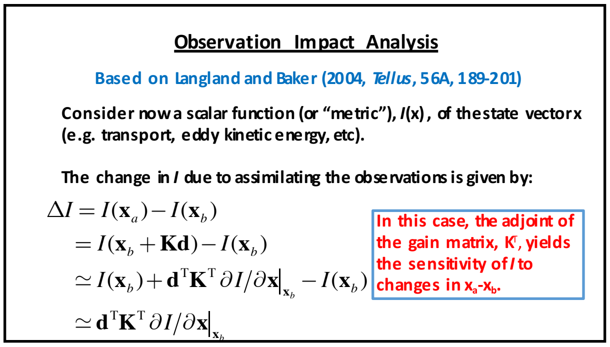
Output files:

- ADSname - $\partial f/\partial\mathbf{x}(0)$ sensitivity information (ocean.in)



Sensitivity of average SST in target area to wind stress and wind stress curl.

Moore et al. (2009)



Observation Impact Analysis

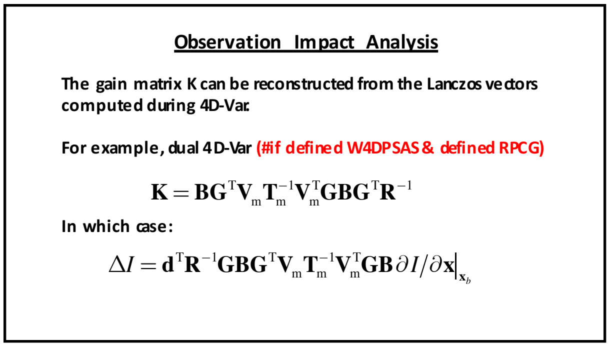
Based on Langland and Baker (2004, *Tellus*, 56A, 189-201)

Consider now a scalar function (or “metric”), *I*(x), of the state vector x (e.g. transport, eddy kinetic energy, etc).

The change in *I* due to assimilating the observations is given by:

$$\begin{aligned}\Delta I &= I(\mathbf{x}_a) - I(\mathbf{x}_b)\\ &= I(\mathbf{x}_b + \mathbf{K}\mathbf{d}) - I(\mathbf{x}_b)\\ &\simeq I(\mathbf{x}_b) + \mathbf{d}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\,\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b} - I(\mathbf{x}_b)\\ &\simeq \mathbf{d}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\,\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b}\end{aligned}$$

In this case, the adjoint of the gain matrix, K^T, yields the sensitivity of *I* to changes in x_a-x_b.



Observation Impact Analysis

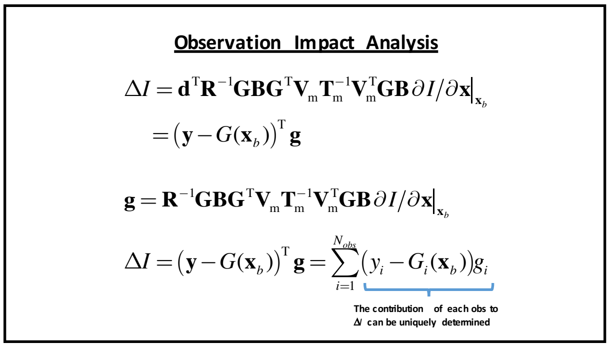
The gain matrix K can be reconstructed from the Lanczos vectors computed during 4D-Var.

For example, dual 4D-Var (#if defined W4DPSAS & defined RPCG)

$$\mathbf{K} = \mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{T}_{\mathrm{m}}^{-1}\mathbf{V}_{\mathrm{m}}^{\mathrm{T}}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}$$

In which case:

$$\Delta I = \mathbf{d}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{T}_{\mathrm{m}}^{-1}\mathbf{V}_{\mathrm{m}}^{\mathrm{T}}\mathbf{G}\mathbf{B}\,\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b}$$



Observation Impact Analysis

$$\begin{aligned}\Delta I &= \mathbf{d}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{T}_{\mathrm{m}}^{-1}\mathbf{V}_{\mathrm{m}}^{\mathrm{T}}\mathbf{G}\mathbf{B}\,\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b}\\ &= \left(\mathbf{y} - G(\mathbf{x}_b)\right)^{\mathrm{T}}\mathbf{g}\end{aligned}$$

$$\mathbf{g} = \mathbf{R}^{-1}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{T}_{\mathrm{m}}^{-1}\mathbf{V}_{\mathrm{m}}^{\mathrm{T}}\mathbf{G}\mathbf{B}\,\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b}$$

$$\Delta I = \left(\mathbf{y} - G(\mathbf{x}_b)\right)^{\mathrm{T}}\mathbf{g} = \sum_{i=1}^{N_{obs}} \underbrace{\left(y_i - G_i(\mathbf{x}_b)\right)g_i}_{\text{The contribution of each obs to } \Delta I \text{ can be uniquely determined}}$$

Observation Impact Analysis

$$\Delta I = (\mathbf{y} - G(\mathbf{x}_b))^{\mathrm{T}}\mathbf{g} = \sum_{i=1}^{N_{obs}} (y_i - G_i(\mathbf{x}_b))g_i$$

OBSname MODname Covariance model

$$\mathbf{g} = \mathbf{R}^{-1}\underbrace{\mathbf{GBG}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{T}_{\mathrm{m}}^{-1}\mathbf{V}_{\mathrm{m}}^{\mathrm{T}}}_{\text{MODname input}}\mathbf{GB}\,\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b}$$

OBSname
input

TLROMS

ADSname
input

Observation Impact Analysis

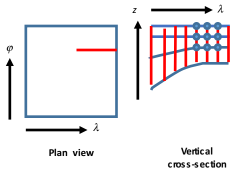
$$\mathbf{g} = \underbrace{\mathbf{R}^{-1}\mathbf{GBG}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{T}_{\mathrm{m}}^{-1}\mathbf{V}_{\mathrm{m}}^{\mathrm{T}}}_{\text{All precomputed during 4D-Var}}\mathbf{GB}\,\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b}$$

$$\mathbf{g} = \mathbf{AGB}\,\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b}$$

Example Functionals

Consider the *linear* functional: $I(\mathbf{x}) = \mathbf{h}^{\mathrm{T}}\mathbf{x}$
(e.g. transport)

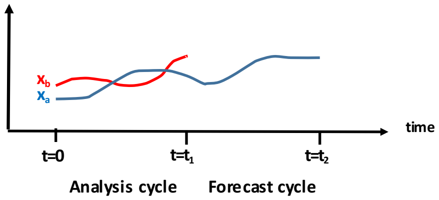
$$\mathbf{x} = \begin{pmatrix} T \\ S \\ u \\ v \\ \varsigma \end{pmatrix}$$



$$\mathbf{h}= \begin{pmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ d\lambda_n dz_n \\ d\lambda_{n+1} dz_{n+1} \\ d\lambda_{n+2} dz_{n+2} \\ \cdot \\ \cdot \\ d\lambda_m dz_m \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{pmatrix}$$

Example Functionals

Consider a typical analysis-forecast cycle:



Observation Impacts of the Analysis Cycle

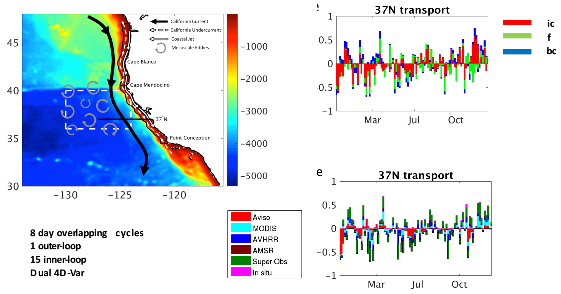
Consider the ***linear*** functional at a specific time: $I(\mathbf{x}(0)) = \mathbf{h}^{\mathrm{T}}\mathbf{x}(0)$
(e.g. transport at initial time)

$$\Delta I = \mathbf{h}^{\mathrm{T}}\mathbf{x}_a(0) - \mathbf{h}^{\mathrm{T}}\mathbf{x}_b(0) = \mathbf{h}^{\mathrm{T}}\delta\mathbf{x}(0)$$

$$\partial I / \partial \mathbf{x}(0) = \mathbf{h}$$

$$\Delta I = (\mathbf{y} - G(\mathbf{x}_b))^{\mathrm{T}}\mathbf{g} = \sum_{i=1}^{N_{obs}} (y_i - G_i(\mathbf{x}_b))g_i$$

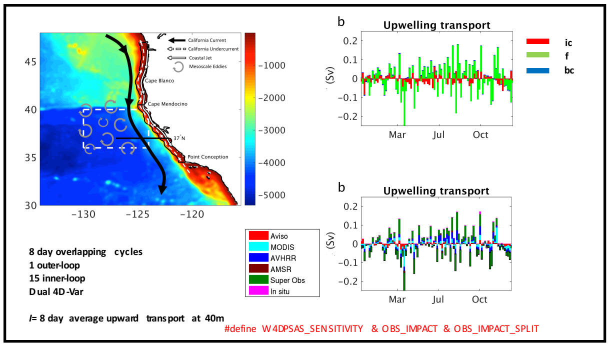
$$\boxed{\mathbf{g} = \mathbf{AGBh}}$$



8 day overlapping cycles
1 outer-loop
15 inner-loop
Dual 4D-Var

I= 8 day average 37N transport

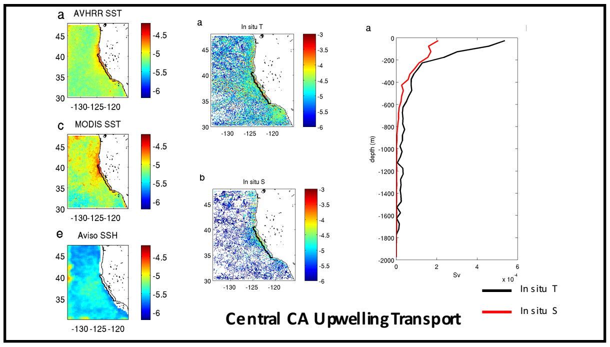
#define W4DPSAS_SENSITIVITY & OBS_IMPACT & OBS_IMPACT_SPLIT



8 day overlapping cycles
1 outer-loop
15 inner-loop
Dual 4D-Var

I= 8 day average upward transport at 40m

#define W4DPSAS_SENSITIVITY & OBS_IMPACT & OBS_IMPACT_SPLIT

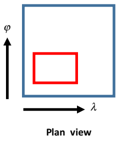


Central CA Upwelling Transport

Quadratic Functionals

Consider the *quadtratic* functional: $I(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{E}\mathbf{x}$
(e.g. kinetic energy)

$$\mathbf{x} = \begin{pmatrix} T \\ S \\ u \\ v \\ \varsigma \end{pmatrix}$$



Plan view

$$\mathbf{E}= \begin{pmatrix} 0 & & & & & & & & & & & \\ & \cdot & & & & & & & & & & \\ & & \cdot & & & & & & & & & \\ & & & 0 & & & & & & & & \\ & & & & 0.5\rho d\lambda_n d\varphi_n & & & & & & & \\ & & & & & 0.5\rho d\lambda_{n+1} d\varphi_{n+1} & & & & & & \\ & & & & & & 0.5\rho d\lambda_{n+2} d\varphi_{n+2} & & & & & \\ & & & & & & & \cdot & & & & \\ & & & & & & & & \cdot & & & \\ & & & & & & & & & 0.5\rho d\lambda_m d\varphi_m & & \\ & & & & & & & & & & 0 & \\ & & & & & & & & & & & \cdot \\ & & & & & & & & & & & & \cdot \\ & & & & & & & & & & & & & 0 \end{pmatrix}$$

Observation Impacts of the Analysis Cycle

Consider the *quadratic* functional at a specific time: $I\left(\mathbf{x}(0)\right)=\mathbf{x}(0)^{\mathrm{T}}\mathbf{E}\mathbf{x}(0)$
(e.g. KE at initial time)

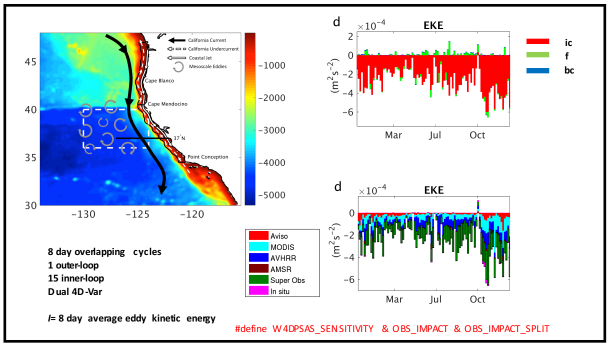
$$\Delta I=\mathbf{x}_a^{\mathrm{T}}(0)\mathbf{E}\mathbf{x}_a(0)-\mathbf{x}_b^{\mathrm{T}}(0)\mathbf{E}\mathbf{x}_b(0)$$

$$\simeq\delta\mathbf{x}^{\mathrm{T}}(0)\mathbf{E}\mathbf{x}_b(0)+\mathbf{x}_b^{\mathrm{T}}(0)\mathbf{E}\delta\mathbf{x}(0)$$

$$\partial I/\partial\mathbf{x}(0)=2\mathbf{E}^{\mathrm{T}}\mathbf{x}_b(0)\quad\text{if }\mathbf{E}\text{ is symmetric}$$

$$\Delta I=\left(\mathbf{y}-G(\mathbf{x}_b)\right)^{\mathrm{T}}\mathbf{g}=\sum_{i=1}^{N_{obs}}\left(y_i-G_i(\mathbf{x}_b)\right)g_i$$

$$\boxed{\mathbf{g}=2\mathbf{AGBE}^{\mathrm{T}}\mathbf{x}_b(0)}$$



8 day overlapping cycles
1 outer-loop
15 inner-loop
Dual 4D-Var

***I*= 8 day average eddy kinetic energy**

#define W4DPSAS_SENSITIVITY & OBS_IMPACT & OBS_IMPACT_SPLIT

Observation Impacts during the Analysis Cycle

Consider the *linear* functional at a some other time during the analysis cycle, such as t_1: $I\left(\mathbf{x}(t_1)\right)$

$$\Delta I=I\left(\mathbf{x}_a(t_1)\right)-I\left(\mathbf{x}_b(t_1)\right)$$

$$=I\left(M(\mathbf{x}_a(0))\right)-I\left(M(\mathbf{x}_b(0))\right)$$

$$=I\left(M(\mathbf{x}_b(0)+\mathbf{Kd})\right)-I\left(M(\mathbf{x}_b(0))\right)$$

$$\simeq\mathbf{d}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b}$$

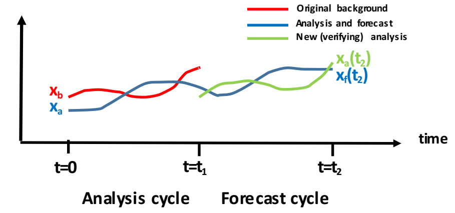
$$\Delta I=\left(\mathbf{y}-G(\mathbf{x}_b)\right)^{\mathrm{T}}\mathbf{g}=\sum_{i=1}^{N_{obs}}\left(y_i-G_i(\mathbf{x}_b)\right)g_i$$

$$\boxed{\mathbf{g}=\mathbf{AGBM}^{\mathrm{T}}\partial I/\partial\mathbf{x}\big|_{\mathbf{x}_b}}$$

$\mathbf{M}^{\mathrm{T}}$ is the ADROMS

Observation Impacts for the Forecast Cycle

Consider a typical analysis-forecast cycle:



Observation Impacts for the Forecast Cycle

In this case we considera functional that is a measure of the difference between the forecastand the verifying analysis:

$$\Delta I = I\left(\mathbf{x}_f(t_2)\right) - I\left(\mathbf{x}_a(t_2)\right)$$

$$= I\left(M(\mathbf{x}_a(0))\right) - I\left(\mathbf{x}_a(t_2)\right)$$

Computing the obs impacts requires two steps:

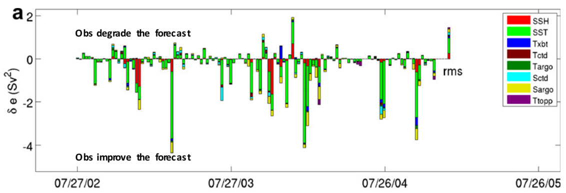
1. An adjoint sensitivity calculation over the forecast cyde from t_2 to t_1

$$\mathbf{x}^{\dagger}(t_1) = \mathbf{M}_f^{\mathrm{T}} \partial I / \partial \mathbf{x}\big|_{\mathbf{x}_f}$$

This step can be refined for 2nd order accuracy (Errico, 2007; Gelaro et al, 2007; Moore et al, 2011c)

2. A regular obs impact calculation over the analysis cycle from t_1 to t=0:

$$\mathbf{g} = \mathbf{AGBM}^{\mathrm{T}}\mathbf{x}^{\dagger}(t_1)$$



37N transport 7 day forecast error

Practical Matters: How to do it yourself

First, write a matlab script to compute the functional *I* of interest.

Important considerations:

- Abandon fancy matlab programming – keep it simple!
- Avoid intrinsic matlab functions, structures and cell arrays (at least until you know what you are doing!)
- Use "for-loops" for transparency

Writing Adjoint Operators

Recall that what we need to run the adjoint model is $\partial I/\partial \mathbf{x}$

So we need is a method for differentiating a matlab script.

A useful result is that if **y=Ax**, then d**y**/d**x**=**A**^{T}

A fool-proof recipe for differentiating code (Giering and Kaminski, 1998):

Matlab code to compute y=Ax

```
x(1:N)=input;
y(1:M)=zeros(M,1);
for i=1:M
  for j=1:N
    y(i)=y(i)+a(i,j)*x(j);
  end
end
y(1:M)=>output
```

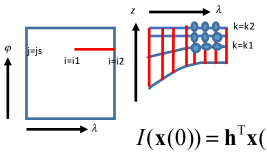
Matlab code to compute $x^\dagger$=A^{T}y†

```
ad_y(1:M)=input;
ad_x=zeros(N,1);
for i=1:M
  for j=1:N
    ad_x(j)=ad_x(j)+a(i,j)*ad_y(i);
  end
End
ad_y(1:M)=zeros(M,1);
ad_x(1:N)=>output
```

This represents the derivative of "$y_i=y_i+a_{ij}x_j$" wrt to x_j.
Step 1: $d/dx_j(y_i+a_{ij}x_j)=a_{ij}$.
Step 2: Multiply this derivative by the adjoint of the variable on the rhs, ad_y_j in this case, a_{ij}*ad_y_j.
Step 3: Keep a running sum in case x_j is used elsewhere later, ad_x_j=ad_x_j+a_{ij}*ad_y_j

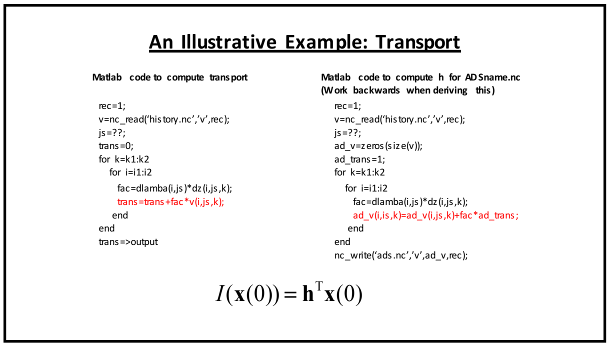
An Illustrative Example: Transport

```
rec=1;
v=nc_read('history.nc','v',rec);
js=??;
trans=0;
for k=k1:k2
   for i=i1:i2
      fac=dlamba(i,js)*dz(i,js,k);
      trans=trans+fac*v(i,js,k);
    end
end
trans=>output
```



$$I(\mathbf{x}(0)) = \mathbf{h}^{\mathrm{T}}\mathbf{x}(0)$$

The next step is to create appropriate forcing fields for the adjoint model. This means we need the derivative of our matlab script.



An Illustrative Example: Transport

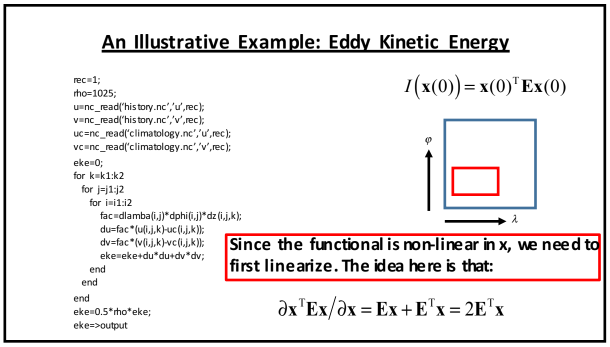
Matlab code to compute transport

```
rec=1;
v=nc_read('history.nc','v',rec);
js=??;
trans=0;
for k=k1:k2
   for i=i1:i2
      fac=dlamba(i,js)*dz(i,js,k);
      trans=trans+fac*v(i,js,k);
    end
end
trans=>output
```

Matlab code to compute h for ADSname.nc (Work backwards when deriving this)

```
rec=1;
v=nc_read('history.nc','v',rec);
js=??;
ad_v=zeros(size(v));
ad_trans=1;
for k=k1:k2
   for i=i1:i2
      fac=dlamba(i,js)*dz(i,js,k);
      ad_v(i,is,k)=ad_v(i,js,k)+fac*ad_trans;
    end
end
nc_write('ads.nc','v',ad_v,rec);
```

$$I(\mathbf{x}(0)) = \mathbf{h}^{\mathrm{T}}\mathbf{x}(0)$$



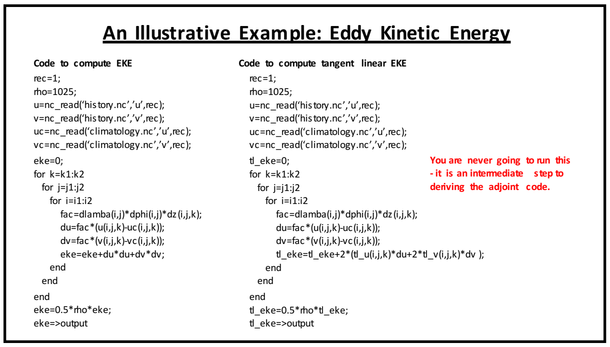
An Illustrative Example: Eddy Kinetic Energy

```
rec=1;
rho=1025;
u=nc_read('history.nc','u',rec);
v=nc_read('history.nc','v',rec);
uc=nc_read('climatology.nc','u',rec);
vc=nc_read('climatology.nc','v',rec);
eke=0;
for k=k1:k2
  for j=j1:j2
    for i=i1:i2
       fac=dlamba(i,j)*dphi(i,j)*dz(i,j,k);
       du=fac*(u(i,j,k)-uc(i,j,k));
       dv=fac*(v(i,j,k)-vc(i,j,k));
       eke=eke+du*du+dv*dv;
    end
  end
end
eke=0.5*rho*eke;
eke=>output
```

$$I\big(\mathbf{x}(0)\big) = \mathbf{x}(0)^{\mathrm{T}}\mathbf{E}\mathbf{x}(0)$$

Since the functional is non-linear in x, we need to first linearize. The idea here is that:

$$\partial\mathbf{x}^{\mathrm{T}}\mathbf{E}\mathbf{x}/\partial\mathbf{x} = \mathbf{E}\mathbf{x} + \mathbf{E}^{\mathrm{T}}\mathbf{x} = 2\mathbf{E}^{\mathrm{T}}\mathbf{x}$$



An Illustrative Example: Eddy Kinetic Energy

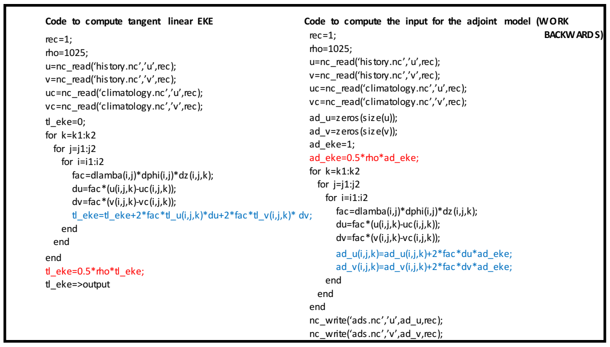
Code to compute EKE

```
rec=1;
rho=1025;
u=nc_read('history.nc','u',rec);
v=nc_read('history.nc','v',rec);
uc=nc_read('climatology.nc','u',rec);
vc=nc_read('climatology.nc','v',rec);
eke=0;
for k=k1:k2
  for j=j1:j2
    for i=i1:i2
       fac=dlamba(i,j)*dphi(i,j)*dz(i,j,k);
       du=fac*(u(i,j,k)-uc(i,j,k));
       dv=fac*(v(i,j,k)-vc(i,j,k));
       eke=eke+du*du+dv*dv;
    end
  end
end
eke=0.5*rho*eke;
eke=>output
```

Code to compute tangent linear EKE

```
rec=1;
rho=1025;
u=nc_read('history.nc','u',rec);
v=nc_read('history.nc','v',rec);
uc=nc_read('climatology.nc','u',rec);
vc=nc_read('climatology.nc','v',rec);
tl_eke=0;
for k=k1:k2
  for j=j1:j2
    for i=i1:i2
       fac=dlamba(i,j)*dphi(i,j)*dz(i,j,k);
       du=fac*(u(i,j,k)-uc(i,j,k));
       dv=fac*(v(i,j,k)-vc(i,j,k));
       tl_eke=tl_eke+2*(tl_u(i,j,k)*du+2*tl_v(i,j,k)*dv );
    end
  end
end
tl_eke=0.5*rho*tl_eke;
tl_eke=>output
```

You are never going to run this - it is an intermediate step to deriving the adjoint code.

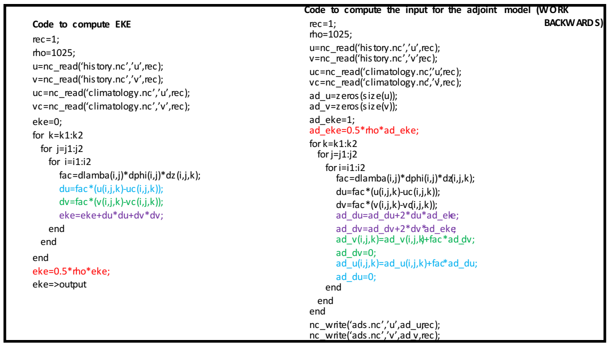


Code to compute tangent linear EKE

```
rec=1;
rho=1025;
u=nc_read('history.nc','u',rec);
v=nc_read('history.nc','v',rec);
uc=nc_read('climatology.nc','u',rec);
vc=nc_read('climatology.nc','v',rec);
tl_eke=0;
for k=k1:k2
  for j=j1:j2
    for i=i1:i2
       fac=dlamba(i,j)*dphi(i,j)*dz(i,j,k);
       du=fac*(u(i,j,k)-uc(i,j,k));
       dv=fac*(v(i,j,k)-vc(i,j,k));
       tl_eke=tl_eke+2*fac*tl_u(i,j,k)*du+2*fac*tl_v(i,j,k)* dv;
    end
  end
end
tl_eke=0.5*rho*tl_eke;
tl_eke=>output
```

Code to compute the input for the adjoint model (WORK BACKWARDS)

```
rec=1;
rho=1025;
u=nc_read('history.nc','u',rec);
v=nc_read('history.nc','v',rec);
uc=nc_read('climatology.nc','u',rec);
vc=nc_read('climatology.nc','v',rec);
ad_u=zeros(size(u));
ad_v=zeros(size(v));
ad_eke=1;
ad_eke=0.5*rho*ad_eke;
for k=k1:k2
  for j=j1:j2
    for i=i1:i2
       fac=dlamba(i,j)*dphi(i,j)*dz(i,j,k);
       du=fac*(u(i,j,k)-uc(i,j,k));
       dv=fac*(v(i,j,k)-vc(i,j,k));
       ad_u(i,j,k)=ad_u(i,j,k)+2*fac*du*ad_eke;
       ad_v(i,j,k)=ad_v(i,j,k)+2*fac*dv*ad_eke;
    end
  end
end
nc_write('ads.nc','u',ad_u,rec);
nc_write('ads.nc','v',ad_v,rec);
```

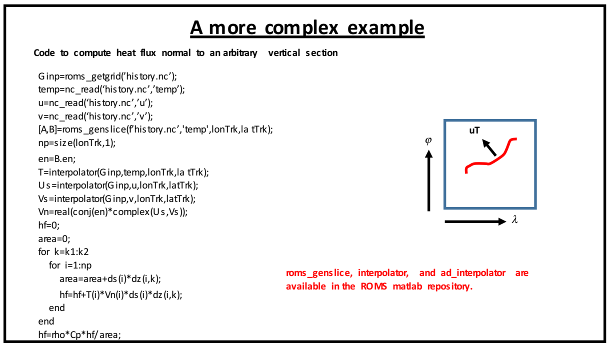


Code to compute EKE

```
rec=1;
rho=1025;
u=nc_read('history.nc','u',rec);
v=nc_read('history.nc','v',rec);
uc=nc_read('climatology.nc','u',rec);
vc=nc_read('climatology.nc','v',rec);
eke=0;
for k=k1:k2
  for j=j1:j2
    for i=i1:i2
       fac=dlamba(i,j)*dphi(i,j)*dz(i,j,k);
       du=fac*(u(i,j,k)-uc(i,j,k));
       dv=fac*(v(i,j,k)-vc(i,j,k));
       eke=eke+du*du+dv*dv;
    end
  end
end
eke=0.5*rho*eke;
eke=>output
```

Code to compute the input for the adjoint model (WORK BACKWARDS)

```
rec=1;
rho=1025;
u=nc_read('history.nc','u',rec);
v=nc_read('history.nc','v'rec);
uc=nc_read('climatology.nc','u',rec);
vc=nc_read('climatology.nc','v',rec);
ad_u=zeros(size(u));
ad_v=zeros(size(v));
ad_eke=1;
ad_eke=0.5*rho*ad_eke;
for k=k1:k2
  for j=j1:j2
    for i=i1:i2
       fac=dlamba(i,j)*dphi(i,j)*dz(i,j,k);
       du=fac*(u(i,j,k)-uc(i,j,k));
       dv=fac*(v(i,j,k)-vc(i,j,k));
       ad_du=ad_du+2*du*ad_eke;
       ad_dv=ad_dv+2*dv*ad_eke;
       ad_v(i,j,k)=ad_v(i,j,k)+fac*ad_dv;
       ad_dv=0;
       ad_u(i,j,k)=ad_u(i,j,k)+fac*ad_du;
       ad_du=0;
    end
  end
end
nc_write('ads.nc','u',ad_u,rec);
nc_write('ads.nc','v',ad_v,rec);
```



A more complex example

Code to compute heat flux normal to an arbitrary vertical section

```
Ginp=roms_getgrid('history.nc');
temp=nc_read('history.nc','temp');
u=nc_read('history.nc','u');
v=nc_read('history.nc','v');
[A,B]=roms_genslice(f'history.nc','temp',lonTrk,la tTrk);
np=size(lonTrk,1);
en=B.en;
T=interpolator(Ginp,temp,lonTrk,la tTrk);
Us=interpolator(Ginp,u,lonTrk,latTrk);
Vs=interpolator(Ginp,v,lonTrk,latTrk);
Vn=real(conj(en)*complex(Us,Vs));
hf=0;
area=0;
for k=k1:k2
  for i=1:np
     area=area+ds(i)*dz(i,k);
     hf=hf+T(i)*Vn(i)*ds(i)*dz(i,k);
  end
end
hf=rho*Cp*hf/ area;
```

roms_genslice, interpolator, and ad_interpolator are available in the ROMS matlab repository.

Code to compute heat flux normal to an arbitrary vertical section

```
Ginp=roms_getgrid('history.nc');
temp=nc_read('history.nc','temp');
u=nc_read('history.nc','u');
v=nc_read('history.nc','v');
[A,B]=roms_genslice('history.nc','temp',lonTrk,latTrk);
np=size(lonTrk,1);
en=B.en;
T=interpolator(Ginp,temp,lonTrk,latTrk);
Us=interpolator(Ginp,u,lonTrk,latTrk);
Vs=interpolator(Ginp,v,lonTrk,latTrk);
Vn=real(conj(en)*complex(Us,Vs));
hf=0;
area=0;
for k=k1:k2
   for i=1:np
      area=area+ds(i)*dz(i,k);
      hf=hf+T(i)*Vn(i)*ds(i)*dz(i,k);
   end
end
hf=rho*Cp*hf/area;
hf=>output
```

Code to compute the input for the adjoint model

```
SAME PREAMBLE AS LEFT
ad_temp=zeros(size(temp));
ad_u=zeros(size(u));
ad_v=zeros(size(v));
ad_T=zeros(size(T));
ad_Vn=zeros(size(Vn));
ad_hf=rho*Cp/area;
for k=k1:k2
   for i=1:np
      ad_T(i)=ad_T(i)+   Vn(i)*ds(i)*dz(i,k)*ad_hf;
      ad_Vn(i)=ad_Vn(i)+T(i)*ds(i)*dz(i,k)*ad_ hf;
   end
end
ad_Us=real(conj(en))*ad_Vn;
ad_Vs=imag(conj(en))*ad_Vn;
ad_u=ad_interpolator(Ginp,u,lonTrk,latTrk,ad_Us);
ad_v=ad_interpolator(Ginp,v,lonTrk,latTrk,ad _Vs);
ad_temp=ad_interpolator(Ginp, temp,l onTrk,latTrk,ad_T );
nc_write('ads.nc','u','ad_u);
nc_write('ads.nc','uv,'ad_v);
nc_write('ads.nc','temp','ad_temp);
```

cpp options and input parameters

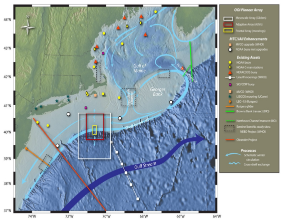
```
#define W4DPSAS_SENSITIVITY
#define OBS_IMPACT
#define OBS_IMPACT_SPLIT
#define AD_IMPULSE
```

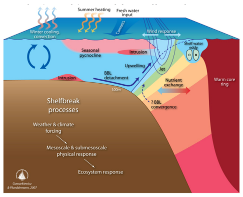
ocean.in:

```
DstrSb=0;
DendSb=0;
KstrSb=1;
KendSb=# levels;

Lstate(isFsur) == T
Lstate(isUbar) == T
Lstate(isVbar) == T
Lstate(isUvel) == T
Lstate(isVvel) == T
Lstate(isTvar) == T T
```

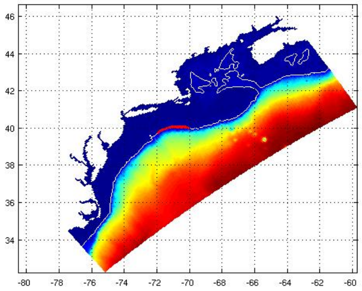
Example: Mid Atlantic Bight

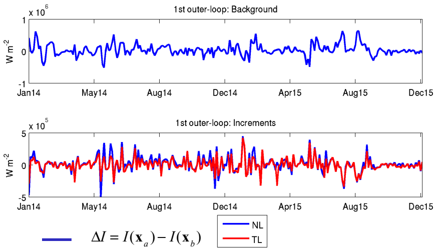




Shelfbreak Front

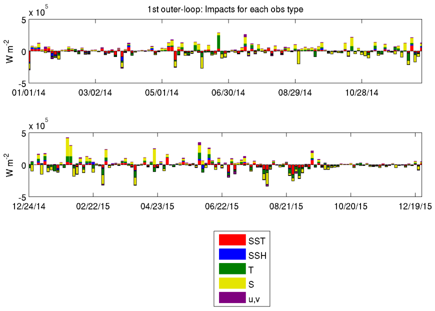
I = heat flux across a section of the 200m isobath following Chen and He (2010, *JPO*, **40**, 949-964) in the vicinity of the shelfbreak front.

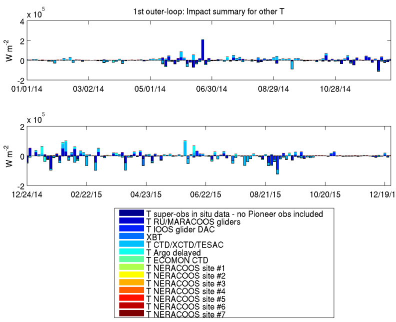


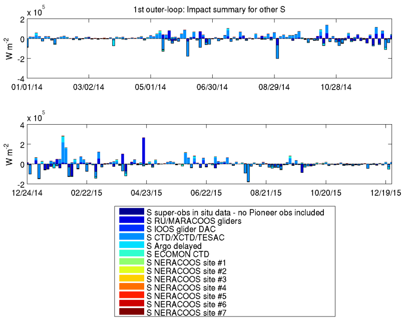


$$\Delta I = I(\mathbf{x}_a) - I(\mathbf{x}_b)$$

$$\Delta I \simeq \left(\mathbf{y} - H(\mathbf{x}_b)\right)^{\mathrm{T}} \mathbf{g}$$







Array Modes: Assessing the Efficacy of the Observing System

- We have explored how the observations impact different aspects of the 4D-var circulation estimates and ensuing forecasts.
- However, we have not yet established how effective the observing system is at "observing" the circulation given our prior hypotheses about the system.
- Recall the analysis equation:

$$\begin{aligned}\mathbf{x}_a &= \mathbf{x}_b + \mathbf{BG}^{\mathrm{T}}\underbrace{(\mathbf{GBG}^{\mathrm{T}} + \mathbf{R})^{-1}(\mathbf{y} - H(\mathbf{x}_b))}_{\mathbf{w}} \\ &= \mathbf{x}_b + \mathbf{Bw}\end{aligned}$$

- So the increment $\mathbf{x}_a$- $\mathbf{x}_b$ lies *entirely* in the space spanned by **B**.

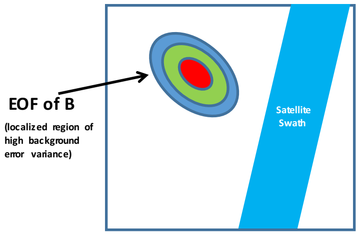
The Importance of the Background Error Covariance Matrix

$$\underset{\substack{\uparrow\\\text{analysis}}}{\mathbf{x}_a} = \underset{\substack{\uparrow\\\text{background}}}{\mathbf{x}_b} + \underbrace{\boxed{\mathbf{B}\underset{\substack{\uparrow\\\text{adjoint}\\\text{obs operator}}}{\mathbf{G}^{T}}\left(\mathbf{G}\underset{\substack{\uparrow\\\text{background}\\\text{error cov}\\\text{matrix}}}{\mathbf{B}}\mathbf{G}^{T} + \underset{\substack{\uparrow\\\text{obs error}\\\text{cov matrix}}}{\mathbf{R}}\right)^{-1}\left(\underset{\substack{\uparrow\\\text{obs}}}{\mathbf{y}} - \underset{\substack{\uparrow\\\text{obs operator}}}{H}(\mathbf{x}_b)\right)}}_{\text{Analysis increment}}$$

The analysis increment "lives" in the space spanned by B !!!

Therefore, to reduce errors in x_b, the observing system must effectively observe (directly via G or indirectly via G^T) the dominant EOFs of B.

An Illustrative Example

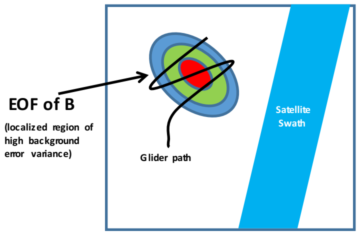


EOF of B

(localized region of high background error variance)

The satellite swath does not directly (G) or indirectly (G^T) observe the region of elevated background error variance associated with the EOF of B, so errors in this regions will not be corrected during data assimilation by the satellite observations.

An Illustrative Example



EOF of B

(localized region of high background error variance)

The glider path does directly observe the region of high error background error variance associated with the EOF of B, so errors in this regions will be corrected during data assimilation by the glider observations.

Eigenvectors

We will be concerned with two different sets of eigenvectors:

1. The EOFs of **B**: $\mathbf{B} = \mathbf{E}\Lambda\mathbf{E}^{\mathrm{T}}$ (More specifically the EOFs of $\mathbf{C} = \Phi\Pi\Phi^{\mathrm{T}}$ where $\mathbf{B} = \Sigma\mathbf{C}\Sigma^{\mathrm{T}}$)
   These tell us about the space in which the increments live.
2. The eigenvectors of the inverse stabilized representer matrix:

$$(\mathbf{GBG}^{\mathrm{T}} + \mathbf{R})^{-1}$$

If this is poorly conditioned, then the increment will be dominated by the eigenvectors of (**GBG**$^{\mathrm{T}}$**+R**) with the *smallest* eigenvalues.

In some sense, it is the juxtaposition of these two sets of eigenvectors that determines the efficacy of the observing system.

Array Modes

Recall that the analysis equation is solved using the Lanczos vectors:

$$\mathbf{x}_a = \mathbf{x}_b + \mathbf{BG}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{T}_{\mathrm{m}}^{-1}\mathbf{V}_{\mathrm{m}}^{\mathrm{T}}\mathbf{GBG}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y} - H(\mathbf{x}_b))$$

This can be rewritten as:

$\mathbf{x}_a = \mathbf{x}_b + \sum_{i=1}^{m} \alpha_i \Psi_i$ where $\Psi_i = \mathbf{BG}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{u}_i$ are the “array modes” (Bennett, 1985)

$$\alpha_i = \lambda^{-1}\mathbf{u}_i^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}^{\mathrm{T}}\mathbf{GBG}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y} - H(\mathbf{x}_b))$$

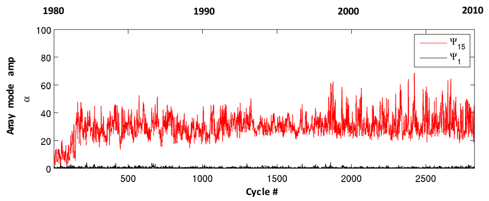
$(\lambda_i, \mathbf{u}_i)$ are the eigenpairs of **T**$_{\mathrm{m}}$

NOTE: The array modes depend *ONLY* on the obs locations, and *NOT* the obs values

Array Modes

- The array modes are a set of generally non-orthogonal basis functions that depend *only* on the obs locations.
- The contribution of each Ψ_i to the increment $\mathbf{x}_a$-$\mathbf{x}_b$ (i.e. the amplitude α_i) depends on the obs values.
- Each Ψ_i is associated with an eigenpair (λ_i,$\mathbf{u}_i$).
- The number of arrays modes equals the number of inner-loops
- Bennett (1985) refers to the array modes as “interpolation patterns.”
- The amplitude α_i depends on $(\lambda_i)^{-1}$, so Ψ_1 represents the most “stable” interpolation pattern wrt changes in the obs values.
- Ψ_m is the least stable, and may represent a significant source of unphysical noise.

Array Modes of the California Current System



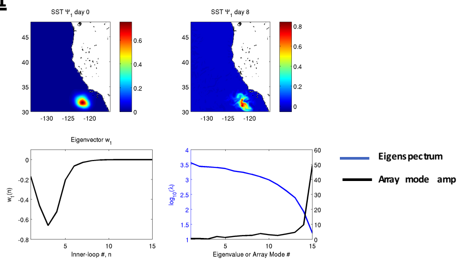
31 year sequence of 4D-Var analyses (1980-2010)
8 day overlapping windows
Obs assimilated: SST, SSH, in situ T and S
1 outer-loop, 15 inner-loops

Array Mode #1

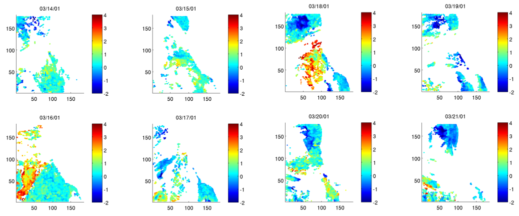
Time evolution:

$$\Psi_i(t) = \mathbf{M}\Psi_i$$

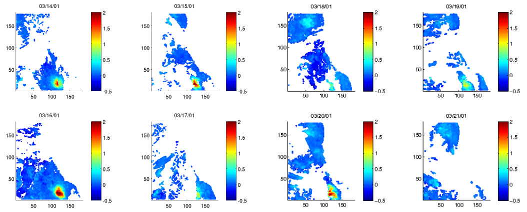
15-23 March, 2001



SST observations: 15-23 March, 2001



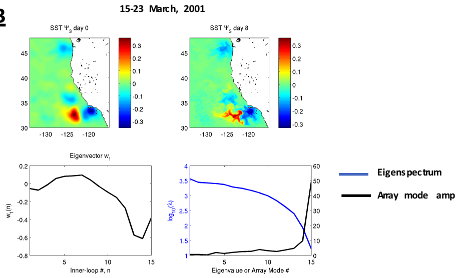
$$\Psi_1 = \mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{V}_m\mathbf{u}_1$$



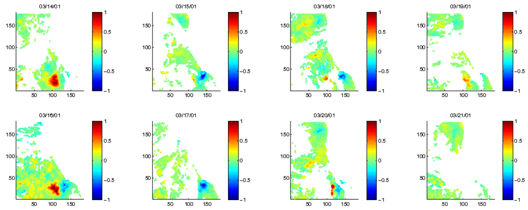
G^T maps these fields into state space

Array Mode #3

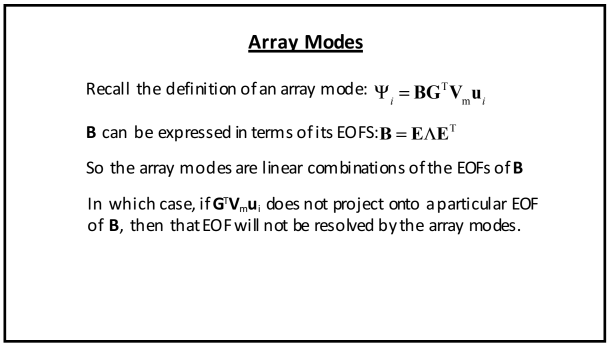
$$\Psi_i(t) = \mathbf{M}\Psi_i$$



$$\Psi_3 = \mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{V}_m\mathbf{u}_3$$



G^T maps these fields into state space



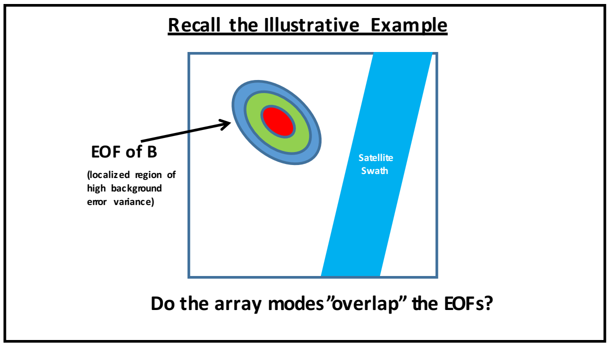
Array Modes

Recall the definition of an array mode: $\Psi_i = \mathbf{BG}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{u}_i$

B can be expressed in terms of its EOFS: $\mathbf{B} = \mathbf{E}\Lambda\mathbf{E}^{\mathrm{T}}$

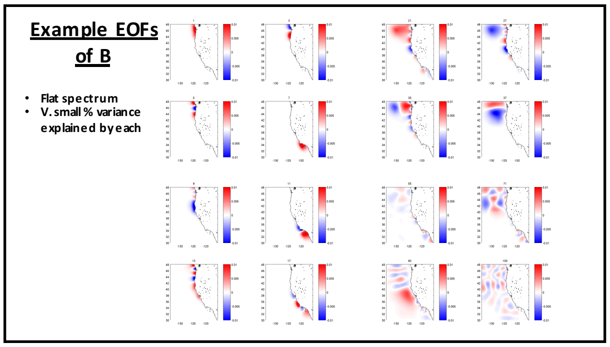
So the array modes are linear combinations of the EOFs of **B**

In which case, if $\mathbf{G}^{\mathrm{T}}\mathbf{V}_{\mathrm{m}}\mathbf{u}_{\mathrm{i}}$ does not project onto a particular EOF of **B**, then that EOF will not be resolved by the array modes.



Recall the Illustrative Example

Do the array modes"overlap" the EOFs?

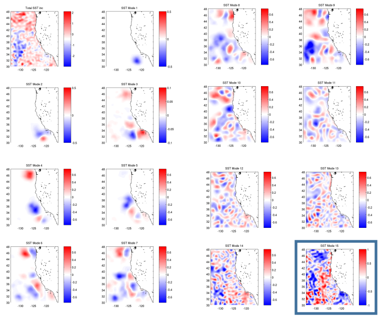


Example EOFs of B

- Flat spectrum
- V. small % variance explained by each

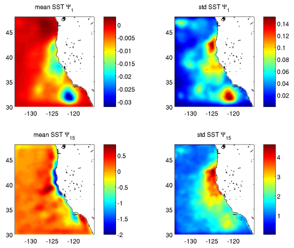
Contribution of each array mode to the increment on 15 March 2001.

The Bennett and McIntosh (1982) "1% rule": Discard array modes with eigenvalues 1% or less of the max value.

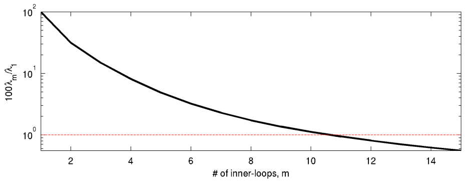


Reject based on 1% rule

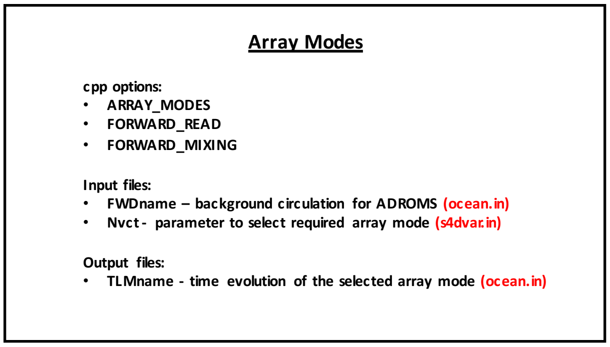
Mean and std of SST associated with array mode 1 and array mode 15 computed from ALL 4D-Var cycles, 1980-2010



The 1% Rule & Overfitting to the Observations



Average eigenvalue ratio for all cycles vs number of inner-loops employed: (suggests we should use no more than 10 inner-loops to prevent over-fitting of the observations)



Array Modes

cpp options:

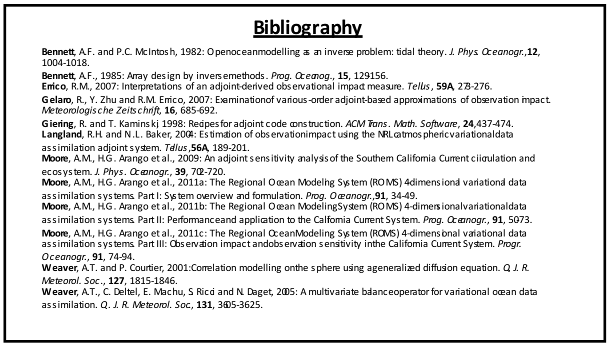
- ARRAY_MODES
- FORWARD_READ
- FORWARD_MIXING

Input files:

- FWDname – background circulation for ADROMS (ocean.in)
- Nvct - parameter to select required array mode (s4dvar.in)

Output files:

- TLMname - time evolution of the selected array mode (ocean.in)



Bibliography

Bennett, A.F. and P.C. McIntosh, 1982: Openoceanmodelling as an inverse problem: tidal theory. *J. Phys. Oceanogr.*,**12**, 1004-1018.

Bennett, A.F., 1985: Array design by inversemethods. *Prog. Oceanog.*, **15**, 129156.

Errico, R.M., 2007: Interpretations of an adjoint-derived observational impact measure. *Tellus*, **59A**, 273-276.

Gelaro, R., Y. Zhu and R.M. Errico, 2007: Examinationof various-order adjoint-based approximations of observation impact. *Meteorologische Zeitschrift*, **16**, 685-692.

Giering, R. and T. Kaminskj 1998: Recipes for adjoint code construction. *ACM Trans. Math. Software*, **24**,437-474.

Langland, R.H. and N.L. Baker, 2004: Estimation of observationimpact using the NRLatmosphericvariationaldata assimilation adjoint system. *Tellus* ,**56A**, 189-201.

Moore, A.M., H.G. Arango et al., 2009: An adjoint sensitivity analysis of the Southern California Current ciiculation and ecosystem. *J. Phys. Oceanogr.*, **39**, 702-720.

Moore, A.M., H.G. Arango et al., 2011a: The Regional Ocean Modeling System (ROMS) 4dimensional variational data assimilation systems. Part I: System overview and formulation. *Prog. Oceanogr.*,**91**, 34-49.

Moore, A.M., H.G. Arango et al., 2011b: The Regional Ocean ModelingSystem (ROMS) 4-dimensionalvariationaldata assimilation systems. Part II: Performanceand application to the Calfornia Current System. *Prog. Oceanogr.*, **91**, 5073.

Moore, A.M., H.G. Arango et al., 2011c: The Regional OceanModeling System (ROMS) 4-dimensional variational data assimilation systems. Part III: Observation impact andobservation sensitivity inthe California Current System. *Progr. Oceanogr.*, **91**, 74-94.

Weaver, A.T. and P. Courtier, 2001:Correlation modelling onthe sphere using ageneralized diffusion equation. *Q. J. R. Meteorol. Soc.*, **127**, 1815-1846.

Weaver, A.T., C. Deltel, E. Machu, S Ricci and N. Daget, 2005: A multivariate balanceoperator for variational ocean data assimilation. *Q. J. R. Meteorol. Soc*, **131**, 3605-3625.