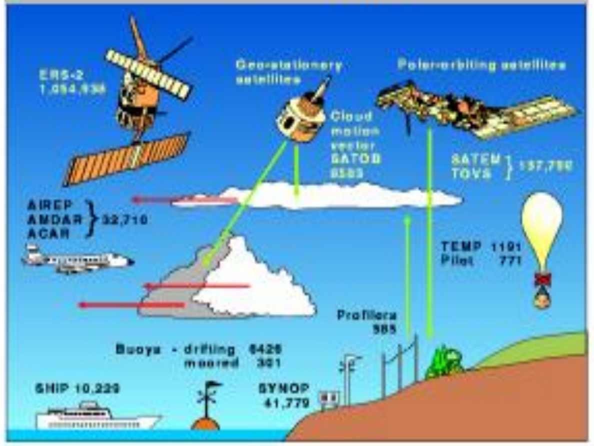
Data Assimilation with Kalman Filters

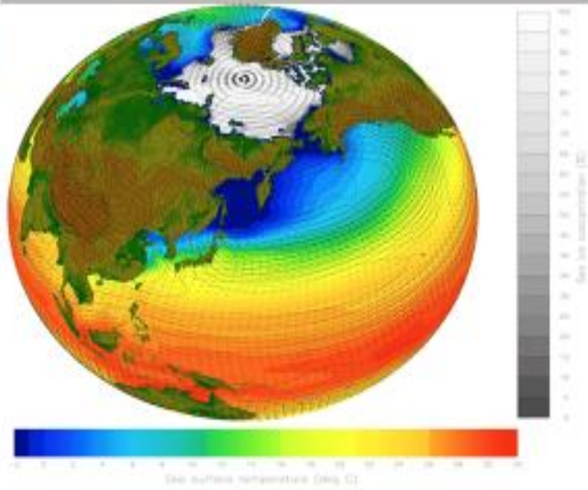
Machine Learning for Earth System Sciences

Carrassi et al

Observations



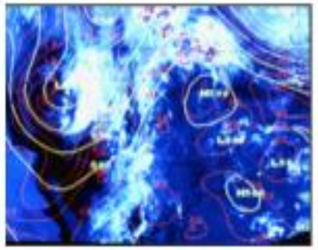
Model Simulations



Data Assimilation best combines model and observations and brings synergy

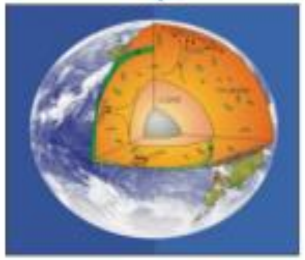
An on-going rapid expansion from **Weather Science (NWP)** into **Climate Science/Geophysics** in general:

- Oceanography
- Climate Prediction
- Climate Assessment
- Hydrology
- Geology
- Climatology
- Detection & Attribution
- ... and many more beyond geosciences ...

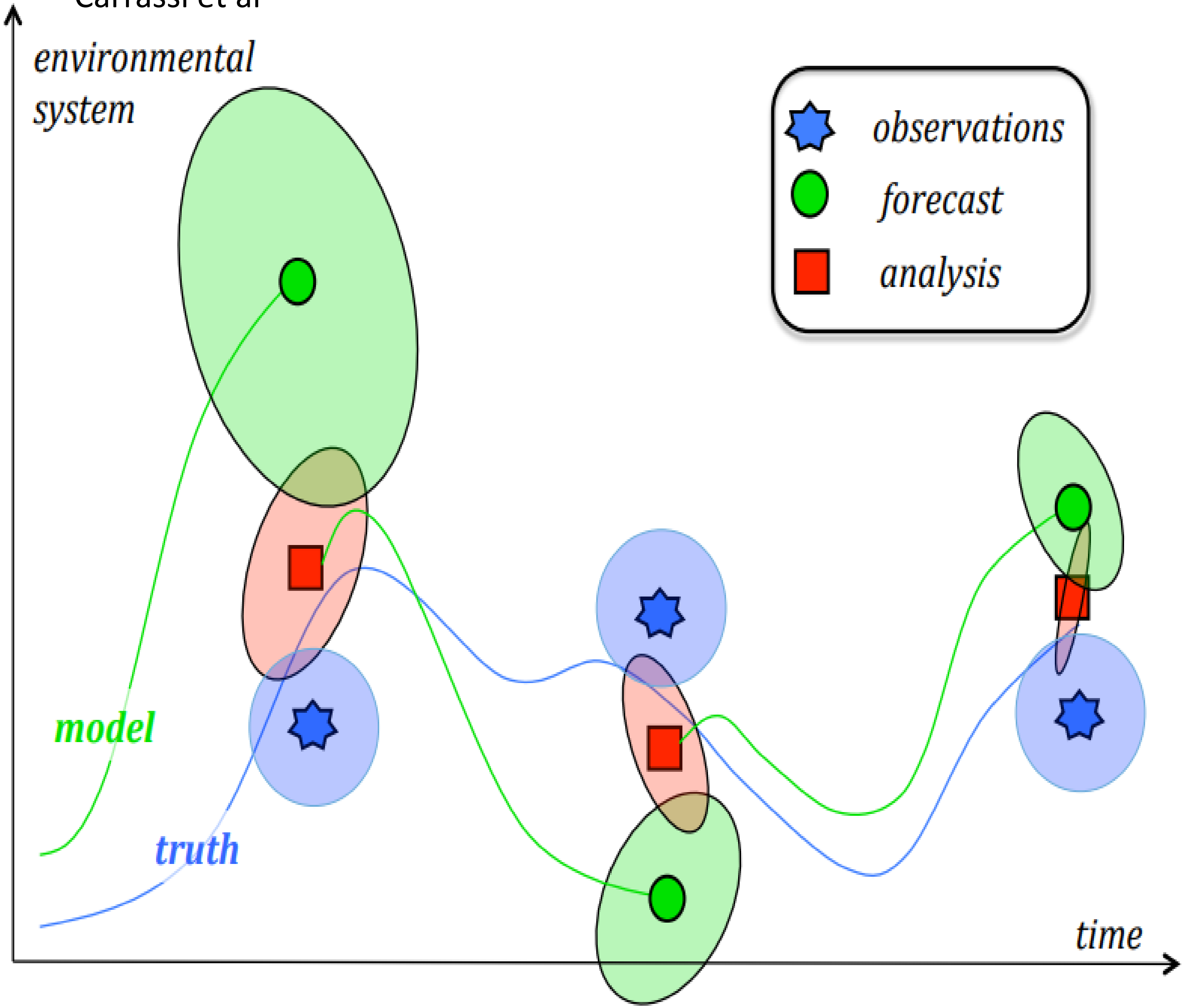


NWP

Geophysics



Carrassi et al



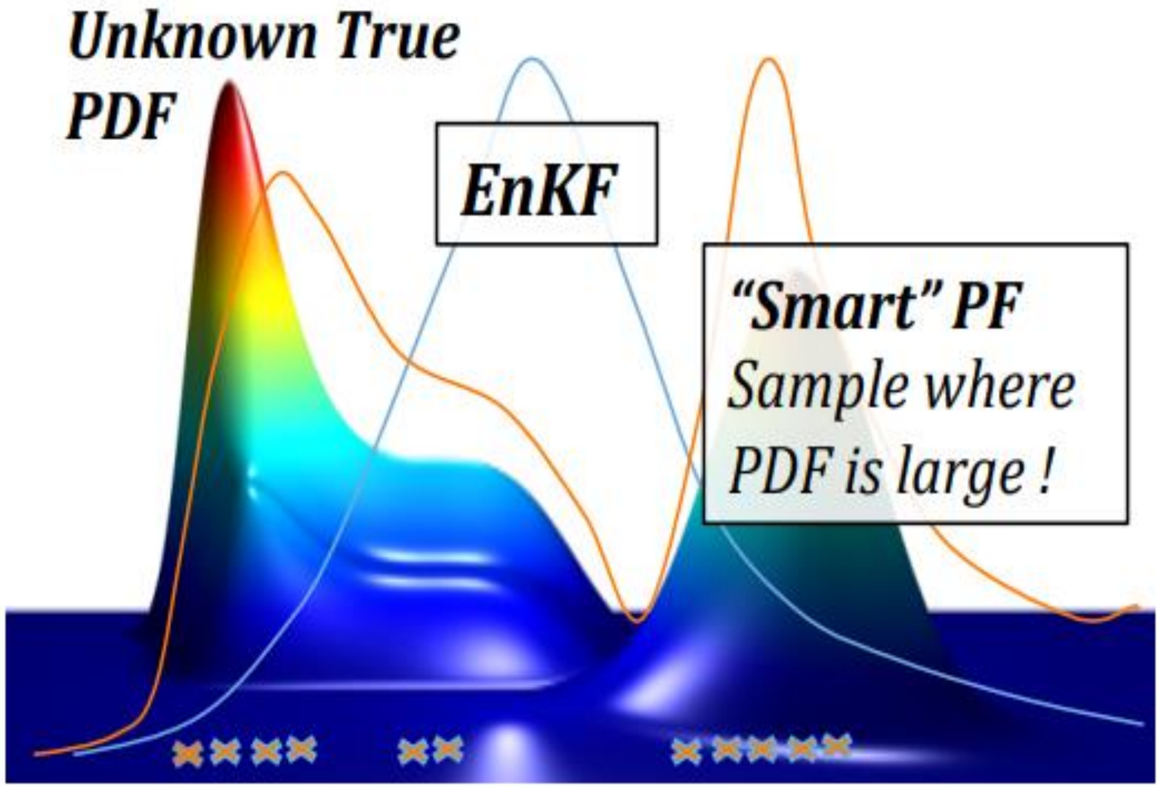
- The problem is in principle solved using a **probabilistic framework**
- The quantity of interest are the **probability density functions**, PDF
- The **PDFs are evolved in time** and updated at analysis times using **Bayes's rules**

Carrassi et al

Efficient Bayesian Data Assimilation

- EnKF/4DVar fails in highly nonlinear/non-Gaussian situations
- Nonlinear Bayesian ***Particle Filter*** are required.

Key Scientific Issue
- **Curse of Dimensionality**
- **Big Data Problem** (model/obs $10^9/10^7$)
- Not **computational power alone**



- NERSC DA group is actively studying advanced formulations of Particle Filter and EnKF to deal with nonlinearity
- The main idea is to study PF which incorporates model dynamic's features in its design (see Raanes and Grudzien poster)

State Update Model: Kalman Filter

- State representation: X(1), X(2), ……., X(t), …. [latent variables]
- Observations: Z(1), Z(2), ……, Z(t), ………
- Input: U(1), U(2), ……. , U(t), …….

- Observation model: Z(t) = f(X(t), u(t)) [f: linear/non-linear]
- State updation model: X(t+1) = g(X(t), u(t))

- **Filtering problem**: Estimate X(t) based on Z(1), Z(2), ….. , Z(t)
- **Smoothing problem**: Estimate X(t) based on Z(1), Z(2), …., Z(t), Z(t+1), …. , Z(T)
- **Prediction problem**: Estimate X(t) based on Z(1), Z(2), ….., Z(t-1)

Kalmanfilter.net

$$\hat{x}_{n+1,n} = F\hat{x}_{n,n} + Gu_n + w_n$$

Where:

$\hat{x}_{n+1,n}$ is a predicted system state vector at time step $n+1$

$\hat{x}_{n,n}$ is an estimated system state vector at time step n

u_n is a **control variable** or **input variable** - a measurable (deterministic) input to the system

w_n is a **process noise** or disturbance - an unmeasurable input that affects the state

F is a **state transition matrix**

G is a **control matrix** or **input transition matrix** (mapping control to state variables)

$$z_n = Hx_n + v_n$$

Where:

z_n is a measurement vector

x_n is a true system state (hidden state)

v_n is a random noise vector

H is an **observation matrix**

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - H\hat{x}_{n,n-1})$$

where:

$\hat{x}_{n,n}$ is a estimated system state vector at time step n

$\hat{x}_{n,n-1}$ is a predicted system state vector at time step $n-1$

K_n is a Kalman Gain

z_n is a measurement

H is an observation matrix

Consider a free-falling object. The state vector includes the altitude h and the object's velocity $\dot{h}$:

$$\hat{\boldsymbol{x}}_n = \begin{bmatrix} \hat{h}_n \\ \hat{\dot{h}}_n \end{bmatrix}$$

The state transition matrix $\boldsymbol{F}$ is:

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

The control matrix $\boldsymbol{G}$ is:

$$\boldsymbol{G} = \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix}$$

The input variable $\boldsymbol{u}_n$ is:

$$\boldsymbol{u}_n = [\, g \,]$$

where g is the gravitational acceleration.

We don't have a sensor that measures acceleration, but we know that for a falling object, the acceleration equals g.

The state extrapolation equation looks like:

$$\begin{bmatrix} \hat{h}_{n+1,n} \\ \hat{\dot{h}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{h}_{n,n} \\ \hat{\dot{h}}_{n,n} \end{bmatrix} + \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix} [\, g \,]$$

Kalmanfilter.net

$$P_{n+1,n} = FP_{n,n}F^T + Q$$

Where:

$P_{n,n}$	is the uncertainty of an estimate - covariance matrix of the current state
$P_{n+1,n}$	is the uncertainty of a prediction - covariance matrix for the next state
F	is the state transition matrix that we derived in the "Modeling linear dynamic systems" section
Q	is the process noise matrix

$$P_{n,n} = (I - K_nH)\,P_{n,n-1}(I - K_nH)^T + K_nR_nK_n^T$$

where:

$P_{n,n}$	is the estimate uncertainty (covariance) matrix of the current state
$P_{n,n-1}$	is the prior estimate uncertainty (covariance) matrix of the current state (predicted at the previous state)
K_n	is the Kalman Gain
H	is the observation matrix
R_n	is the Measurement Uncertainty (measurement noise covariance matrix)

$$K_n = P_{n,n-1}H^T\left(HP_{n,n-1}H^T + R_n\right)^{-1}$$

where:

K_n	is the Kalman Gain
$P_{n,n-1}$	is the prior estimate uncertainty (covariance) matrix of the current state (predicted at the previous step)
H	is the observation matrix
R_n	is the Measurement Uncertainty (measurement noise covariance matrix)

Kalmanfilter.net

Time Update ("Predict")

1. Extrapolate the state

$$\hat{x}_{n+1,n} = F\hat{x}_{n,n} + Gu_n$$

2. Extrapolate uncertainty

$$P_{n+1,n} = FP_{n,n}F^T + Q$$

Measurement Update ("Correct")

1. Compute the Kalman Gain

$$K_n = P_{n,n-1}H^T\left(HP_{n,n-1}H^T + R_n\right)^{-1}$$

2. Update estimate with measurement

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n\left(z_n - H\hat{x}_{n,n-1}\right)$$

3. Update the estimate uncertainty

$$P_{n,n} = (I - K_nH)P_{n,n-1}(I - K_nH)^T + K_nR_nK_n^T$$

Initial Estimate: $\hat{x}_{0,0}$, $P_{0,0}$

	Equation	Equation Name	Alternative names
Predict	$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$	State Extrapolation	Predictor Equation Transition Equation Prediction Equation Dynamic Model State Space Model
	$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$	Covariance Extrapolation	Predictor Covariance Equation
Update (correction)	$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$	State Update	Filtering Equation
	$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$	Covariance Update	Corrector Equation
	$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$	Kalman Gain	Weight Equation
Auxiliary	$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n$	Measurement Equation	
	$\boldsymbol{R}_n = E\left(\boldsymbol{v}_n\boldsymbol{v}_n^T\right)$	Measurement Uncertainty	Measurement Error
	$\boldsymbol{Q}_n = E\left(\boldsymbol{w}_n\boldsymbol{w}_n^T\right)$	Process Noise Uncertainty	Process Noise Error
	$\boldsymbol{P}_{n,n} = E\left(\boldsymbol{e}_n\boldsymbol{e}_n^T\right) = E\left((\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n})^T\right)$	Estimation Uncertainty	Estimation Error

Bayesian Kalman Filtering

- Express the above equations as probabilistic models to account for the uncertainty
- Prediction Model: $P(X(t) \mid X(t-1))$
- Observation Model: $P(Y(t) \mid X(t))$
- Observation sequence: $P(Y(1:T) \mid X(1:T)) = \prod_t P(Y(t) \mid X(t))$
- State sequence: $P(X(1:T)) = P(X(1)) * \prod_t P(X(t) \mid X(t-1))$

- Combining: $P(X(1:T+1) \mid Y(1:T)) = P(X(1)) * \prod_t P(Y(t) \mid X(t)) * \prod_t P(X(t+1) \mid X(t))$