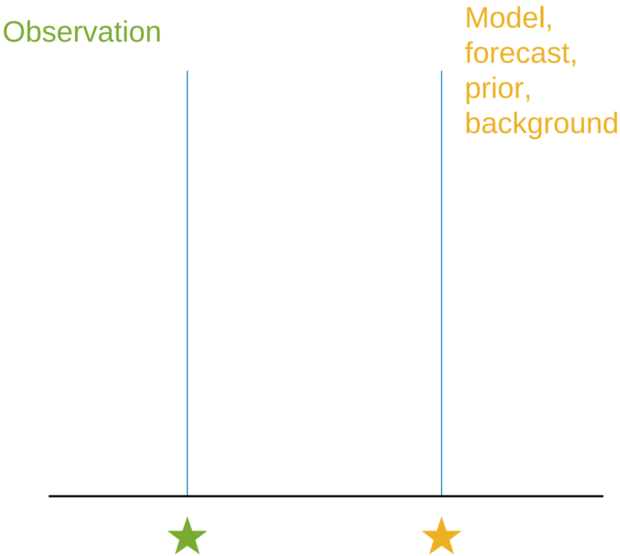
Data Assimilation: Part 1
Overview and Particle Filters

Elaine Spiller

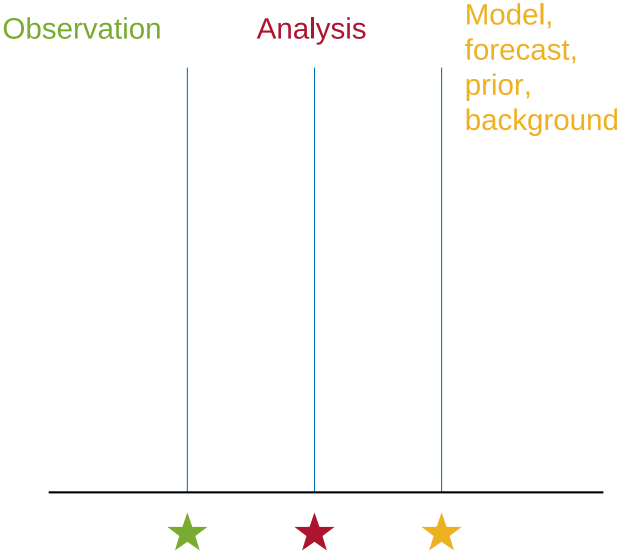
Marquette University

May 21, 2017

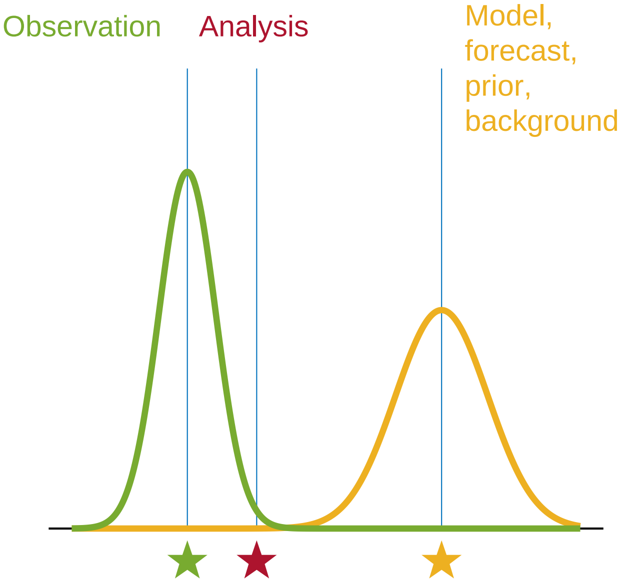
Data assimilation – scalar example



Data assimilation – scalar example



Data assimilation – scalar example



Kalman Filter

Optimal way to combine data and model-based predictions if...

- model noise/uncertainty & obs error are both Gaussian

$$x^a = x^f + \frac{\sigma_f^2}{\sigma_f^2 + \sigma_o^2}(y - x^f)$$

x^f forecasted state estimate (model, prior, background)

y observation of state (data)

x^a analysis, best (combined) state estimate

σ_f^2 forecast variance (uncertainty)

σ_o^2 data model variance (uncertainty)

Kalman Filter

Optimal way to combine data and model-based predictions if...

- model noise/uncertainty & obs error are both Gaussian

$$x^a = x^f + \frac{\sigma_f^2}{\sigma_f^2 + \sigma_o^2}(y - x^f)$$

If $\sigma_o^2 >> \sigma_f^2$

Kalman Filter

Optimal way to combine data and model-based predictions if...

- model noise/uncertainty & obs error are both Gaussian

$$x^a = x^f + \frac{\sigma_f^2}{\sigma_f^2 + \sigma_o^2}(y - x^f)$$

If $\sigma_o^2 >> \sigma_f^2$

$$x^a \approx x^f$$

Kalman Filter

Optimal way to combine data and model-based predictions if...

- model noise/uncertainty & obs error are both Gaussian

$$x^a = x^f + \frac{\sigma_f^2}{\sigma_f^2 + \sigma_o^2}(y - x^f)$$

If $\sigma_f^2 >> \sigma_o^2$

Kalman Filter

Optimal way to combine data and model-based predictions if...

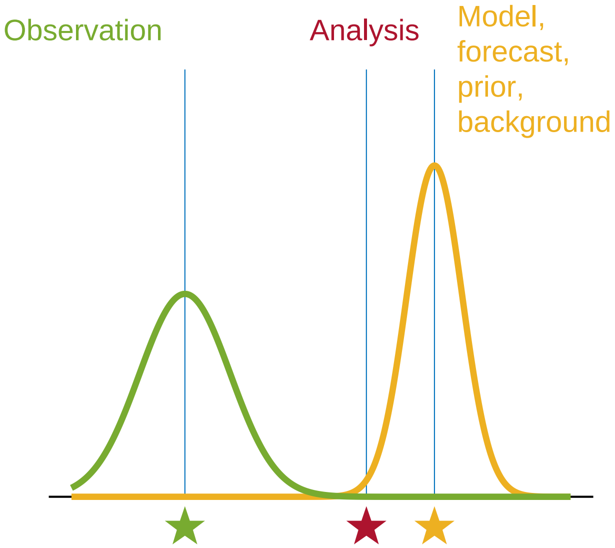
- model noise/uncertainty & obs error are both Gaussian

$$x^a = x^f + \frac{\sigma_f^2}{\sigma_f^2 + \sigma_o^2}(y - x^f)$$

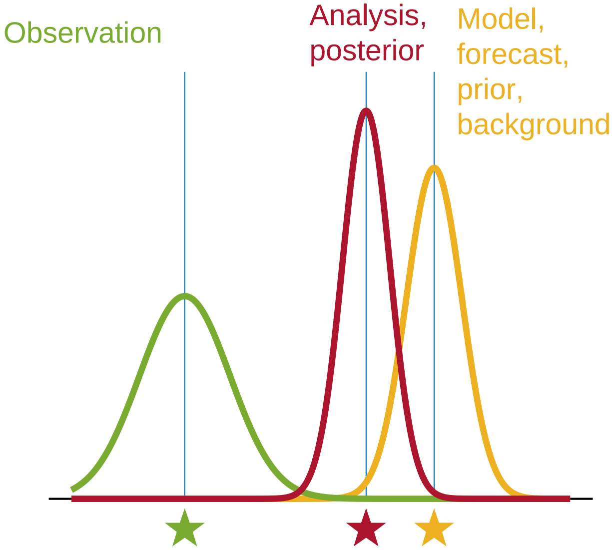
If $\sigma_f^2 >> \sigma_o^2$

$$x^a \approx y$$

Data assimilation – scalar example



Data assimilation – scalar example



Kalman Filter

Optimal way to combine data and model-based predictions if...

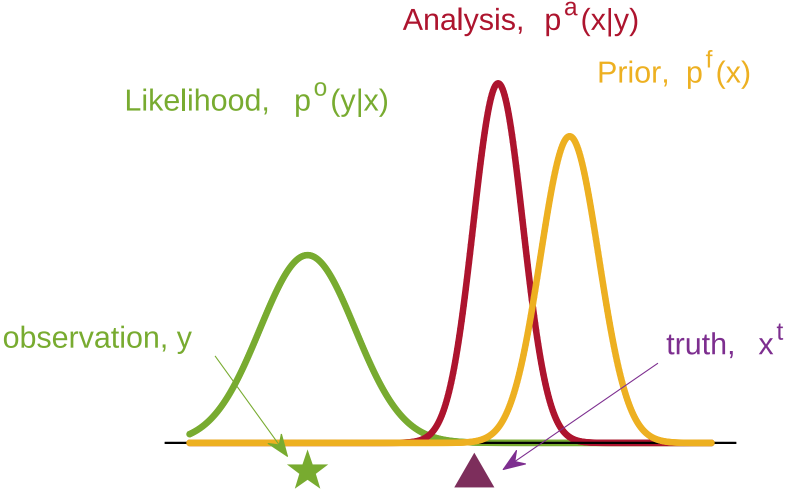
- model noise/uncertainty & obs error are both Gaussian

$$x^a = x^f + k(y - x^f) \qquad \text{analysis mean}$$

$$\sigma_a^2 = (1-k)^2\sigma_f^2 + k^2\sigma_o^2 \qquad \text{analysis variance}$$

$$k = \frac{\sigma_f^2}{\sigma_f^2 + \sigma_o^2} \qquad \text{Kalman gain}$$

Bayes



Analysis, $p^a(x|y)$
Prior, $p^f(x)$
Likelihood, $p^o(y|x)$
observation, y
truth, x^t

Bayes

$$p(\text{state} \mid \text{data}) \propto p(\text{data} \mid \text{state})p(\text{state})$$

Bayes

$$p(\text{state} \mid \text{data}) \propto p(\text{data} \mid \text{state})p(\text{state})$$

or

$$p^a(x \mid y) \propto p^o(y \mid x)p^f(x)$$

Bayes

$$p(\text{state} \mid \text{data}) \propto p(\text{data} \mid \text{state}) p(\text{state})$$

or

$$p^a(x \mid y) \propto p^o(y \mid x) p^f(x)$$

or

$$\text{posterior} \propto \text{likelihood} \cdot \text{prior}$$

Kalman Filter via Least Squares

Since both posterior and prior are Gaussian,

$$p^f(x) = \frac{1}{\sqrt{2\pi\sigma_f^2}} \exp\left(\frac{-(x - x^f)^2}{2\sigma_f^2}\right)$$

$$p^o(y \mid x) = \frac{1}{\sqrt{2\pi\sigma_o^2}} \exp\left(\frac{-(x - y)^2}{2\sigma_o^2}\right)$$

Kalman Filter via Least Squares

Since both posterior and prior are Gaussian,

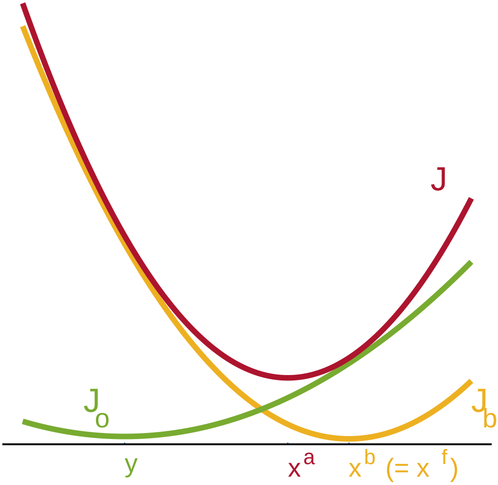
$$p^f(x) = \frac{1}{\sqrt{2\pi\sigma_f^2}} \exp\left(\frac{-(x - x^f)^2}{2\sigma_f^2}\right)$$
$$p^o(y \mid x) = \frac{1}{\sqrt{2\pi\sigma_o^2}} \exp\left(\frac{-(x - y)^2}{2\sigma_o^2}\right)$$

maximizing the posterior is equivalent to minimizing a sum of squares.

$$J(x) = J_o(x) + J_b(x)$$
$$J(x) = \frac{(x - y)^2}{2\sigma_o^2} + \frac{(x - x^b)^2}{2\sigma_b^2}$$

(Note, change in notation — $f = b$, forecast = background)

Least Squares Cost Function



Beyond scalars — $\mathbf{x} \in \mathbb{R}^n$ *and* $\mathbf{y} \in \mathbb{R}^p$

$\mathbf{x}^t$ true model state (dim n)

$\mathbf{x}^b$ background model state (dim n, also mean of prior $\mathbf{x}^b = \mathbf{x}^f$)

$\mathbf{x}^a$ analysis model sate (dim n, also mean of posterior)

$\mathbf{y}$ vector of observations (dim p)

Beyond scalars — $\mathbf{x} \in \mathbb{R}^n$ *and* $\mathbf{y} \in \mathbb{R}^p$

- $\mathbf{x}^t$ true model state (dim n)
- $\mathbf{x}^b$ background model state (dim n, also mean of prior $\mathbf{x}^b = \mathbf{x}^f$)
- $\mathbf{x}^a$ analysis model sate (dim n, also mean of posterior)
- $\mathbf{y}$ vector of observations (dim p)
- H observation operator (from dim n to p)
- $\mathbf{H}$ assuming $H(\mathbf{x}) = \mathbf{Hx}$ (dim $p \times n$)
- $\mathbf{B}$ covariance of background errors ($\mathbf{x}^b - \mathbf{x}^t$) (dim $n \times n$, $\mathbf{B} = \mathbf{P}^f$)
- $\mathbf{R}$ covariance matrix of observation errors ($\mathbf{x}^b - H(\mathbf{x}^t)$) (dim $p \times p$)
- $\mathbf{A}$ covariance matrix of analysis errors ($\mathbf{x}^a - \mathbf{x}^t$) (dim $n \times n$, $\mathbf{A} = \mathbf{P}^a$)

Kalman update

$$J(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^b)^T \mathbf{B}^{-1} (\mathbf{x} - \mathbf{x}^b) + (\mathbf{y} - H(\mathbf{x}))^T \mathbf{R}^{-1} (\mathbf{y} - H(\mathbf{x}))$$

or

$$p^a(\mathbf{x} \,|\, \mathbf{y}) \propto \exp\left(-(\mathbf{x} - \mathbf{x}^f)^T (\mathbf{P}^f)^{-1} (\mathbf{x} - \mathbf{x}^f)\right) \exp\left(-(\mathbf{y} - H(\mathbf{x}))^T \mathbf{R}^{-1} (\mathbf{y} - H(\mathbf{x}))\right)$$

Kalman update

$$J(\mathbf{x}) = (\mathbf{x}-\mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b) + (\mathbf{y}-H(\mathbf{x}))^T\mathbf{R}^{-1}(\mathbf{y}-H(\mathbf{x}))$$

or

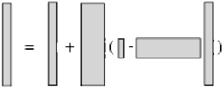
$$p^a(\mathbf{x}\,|\,\mathbf{y}) \propto \exp\left(-(\mathbf{x}-\mathbf{x}^f)^T(\mathbf{P}^f)^{-1}(\mathbf{x}-\mathbf{x}^f)\right)\exp\left(-(\mathbf{y}-H(\mathbf{x}))^T\mathbf{R}^{-1}(\mathbf{y}-H(\mathbf{x}))\right)$$

$$\mathbf{x}^a = \mathbf{x}^b + \mathbf{K}(\mathbf{y}-H(\mathbf{x}^b)) \quad \text{analysis mean}$$

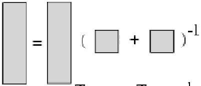
$$\mathbf{K} = \mathbf{B}\mathbf{H}^T\left(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R}\right)^{-1} \quad \text{Kalman gain}$$

$$\mathbf{A} = \mathbf{P}^a = (\mathbf{I}-\mathbf{K}\mathbf{H})\mathbf{B} \quad \text{analysis covariance}$$

Kalman update[1]

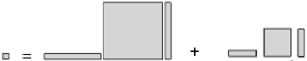


$$x_a = x_b + K(y - Hx_b)$$



$$K = BH^T(HBH^T + R)^{-1}$$

$$H\ B\ H^T$$



$$J(x) = (x-x_b)^T B^{-1}(x-x_b) + (y-Hx)^T R^{-1}(y-Hx)$$

[1] borrowed from tutorial by F. Bouttier and P. Courtier, Data assimilation concepts and methods March 1999

Challenges (not an exhaustive list)

- What if n and/or p are large?

Challenges (not an exhaustive list)

- What if n and/or p are large?
- What if prior and/or likelihood non-Gaussian?

Challenges (not an exhaustive list)

- What if n and/or p are large?
- What if prior and/or likelihood non-Gaussian?
- What if we have a time series of data?

  $$\mathbf{y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} = \mathbf{y}_{1:N}$$

  And a dynamic model $\dot{\mathbf{x}} = f(\mathbf{x})$?

Challenges (not an exhaustive list)

- What if *n* and/or *p* are large?
- What if prior and/or likelihood non-Gaussian?
- What if we have a time series of data?

$$\mathbf{y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} = \mathbf{y}_{1:N}$$

  And a dynamic model $\dot{\mathbf{x}} = f(\mathbf{x})$?
- What if dynamics are nonlinear?

Dynamic data assimilation[2]

$t = t_0$

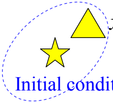
$x_1^{\mathrm{t}}(t_0), x_2^{\mathrm{t}}(t_0)$

[2] borrowed from random talk of Chris Jones

Dynamic data assimilation

$t = t_0$

$t = t_1$

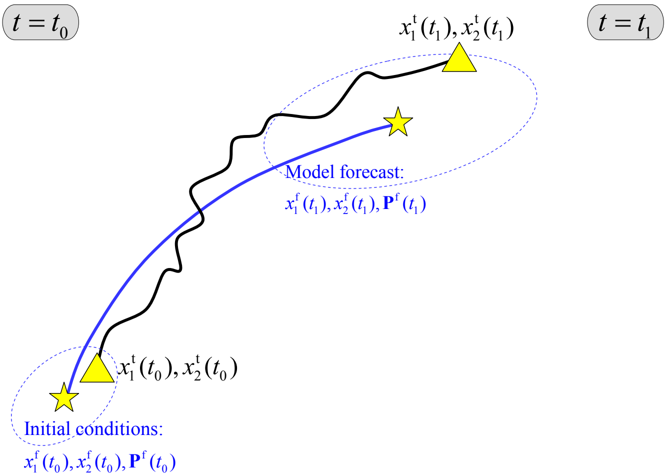


$x_1^{\mathrm{t}}(t_0), x_2^{\mathrm{t}}(t_0)$

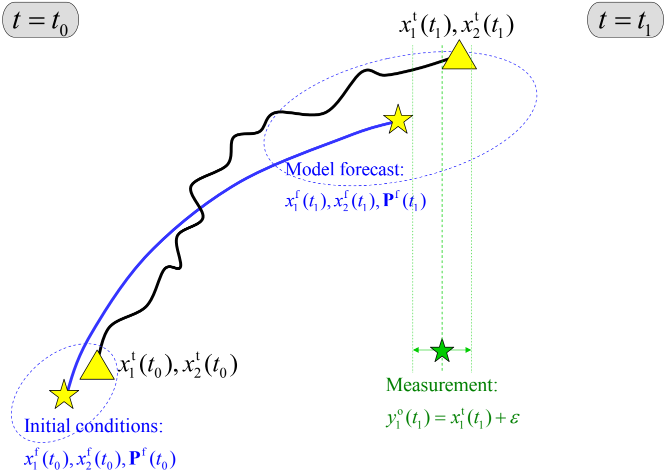
Initial conditions:

$x_1^{\mathrm{f}}(t_0), x_2^{\mathrm{f}}(t_0), \mathbf{P}^{\mathrm{f}}(t_0)$

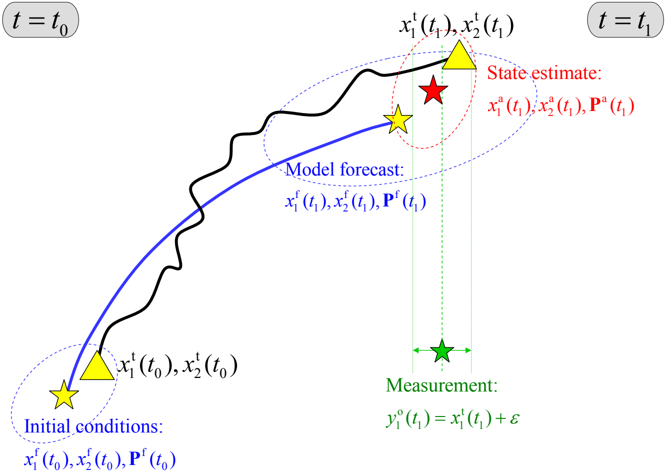
Dynamic data assimilation



Dynamic data assimilation



Dynamic data assimilation



Data assimilation is a state estimation problem[3]

- Dynamical model for the state vector $\mathbf{x} \in X$

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}) \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_0$$

- The solution denoted by $\mathbf{x}(t) = \Phi(\mathbf{x}_0, t)$

[3] borrowed from random talk of Amit Apte

Data assimilation is a state estimation problem[3]

- Dynamical model for the state vector $\mathbf{x} \in X$

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}) \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_0$$

- The solution denoted by $\mathbf{x}(t) = \Phi(\mathbf{x}_0, t)$
- Given some noisy observations of the system at times $0 < t_1 < t_2 < \cdots < t_N$, we can consider three problems:

Smoothing: Obtain a state estimate $\mathbf{x}(t)$ for $t < t_N$;
In particular, determine $\mathbf{x}(0)$.

Filtering: Obtain a state estimate $\mathbf{x}(t_i)$.

Prediction: Obtain a state estimate $\mathbf{x}(t)$ for $t > t_N$
(the time horizon of prediction is important).

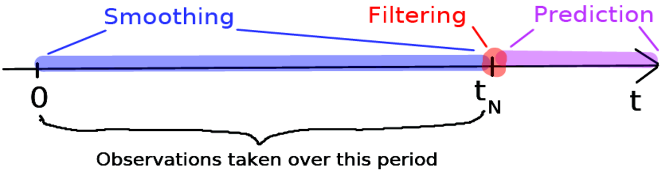
[3] borrowed from random talk of Amit Apte

Data assimilation is a state estimation problem[3]

- Dynamical model for the state vector $\mathbf{x} \in X$

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}) \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_0$$

- The solution denoted by $\mathbf{x}(t) = \Phi(\mathbf{x}_0, t)$
- Given some noisy observations of the system at times $0 < t_1 < t_2 < \cdots < t_N$, we can consider three problems:



[3] borrowed from random talk of Amit Apte

Or data assimilation ≡ determination of posterior distribution

Observations $\mathbf{y}_i \in Y$ at time t_i depend on the state at that time.

$$y_i = H\big(\mathbf{x}^t(t_i)\big) + \eta_i, \quad i = 1, \ldots, N$$

η_i is observational noise which is usually finite dimensional.

Probabilistic statement of data assimilation problem: find the posterior distribution of the state conditioned on the observations

$$\begin{aligned} \textit{Smoothing:} \quad & p(\mathbf{x}(t) \mid \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N) \quad \text{for } t < t_N \\ \textit{Filtering:} \quad & p(\mathbf{x}(t_N) \mid \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N) \\ \textit{Prediction:} \quad & p(\mathbf{x}(t) \mid \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N) \quad \text{for } t > t_N \end{aligned}$$

Or data assimilation ≡ determination of posterior distribution

Observations $\mathbf{y}_i \in Y$ at time t_i depend on the state at that time.

$$y_i = H\big(\mathbf{x}^t(t_i)\big) + \eta_i, \quad i = 1, \ldots, N$$

η_i is observational noise which is usually finite dimensional.

Probabilistic statement of data assimilation problem: find the posterior distribution of the state conditioned on the observations

$$\textit{Filtering:}\ \ p\big(\mathbf{x}(t_N) \mid \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\big)$$

Filtering is also know as Sequential data assimilation

Sequential data assimilation

Model + observations $\longmapsto$ prediction

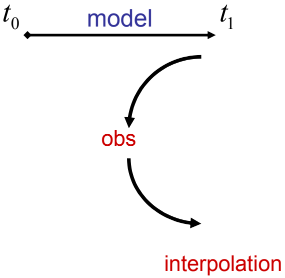
Sequential data assimilation

Model + observations $\longrightarrow$ prediction

$t_0 \xrightarrow{\text{model}} t_1$

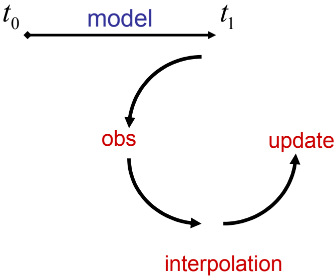
Sequential data assimilation

Model + observations $\longmapsto$ prediction



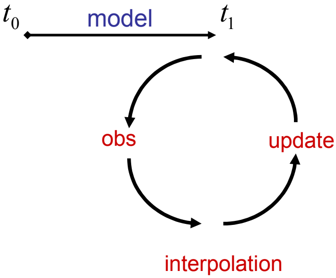
Sequential data assimilation

Model + observations $\longrightarrow$ prediction

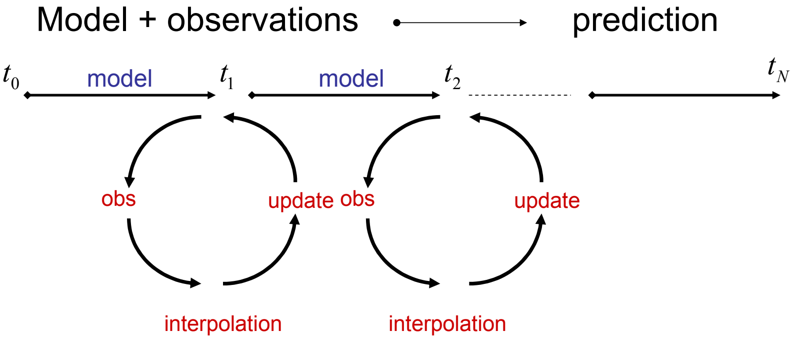


Sequential data assimilation

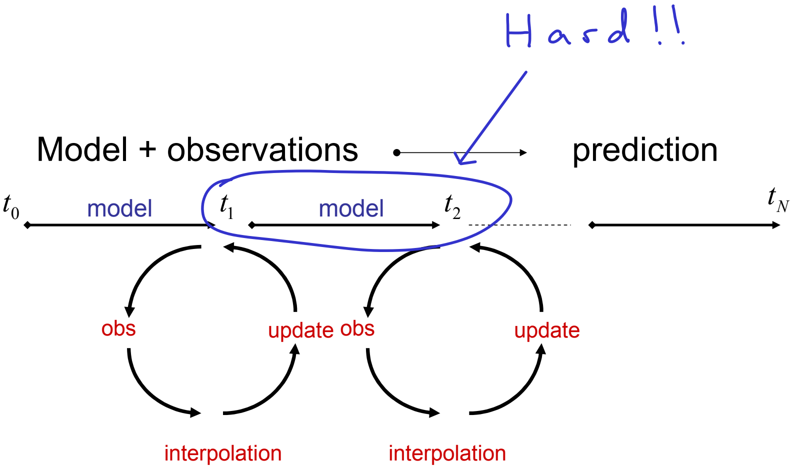
Model + observations ⟶ prediction



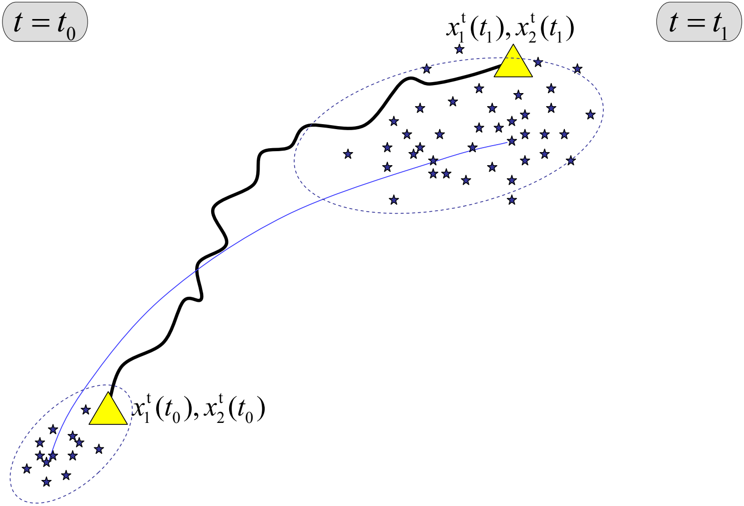
Sequential data assimilation



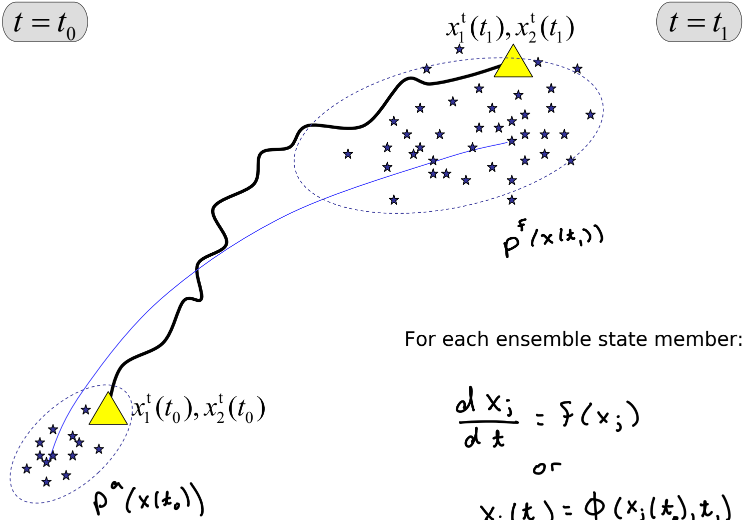
Sequential data assimilation



Sequential data assimilation — filtering with ensembles



Sequential data assimilation — filtering with ensembles



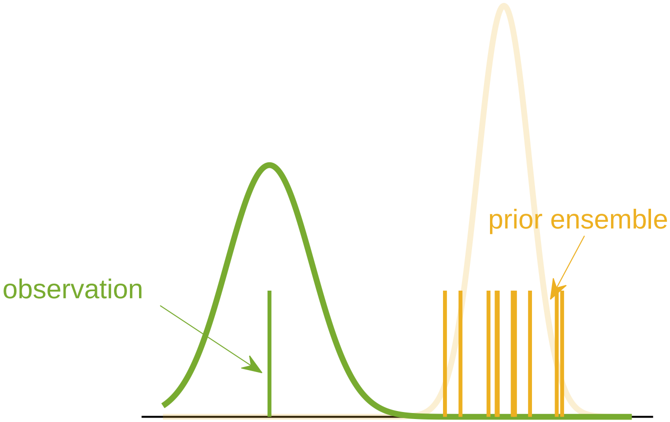
For each ensemble state member:

$$\frac{dx_j}{dt} = f(x_j)$$

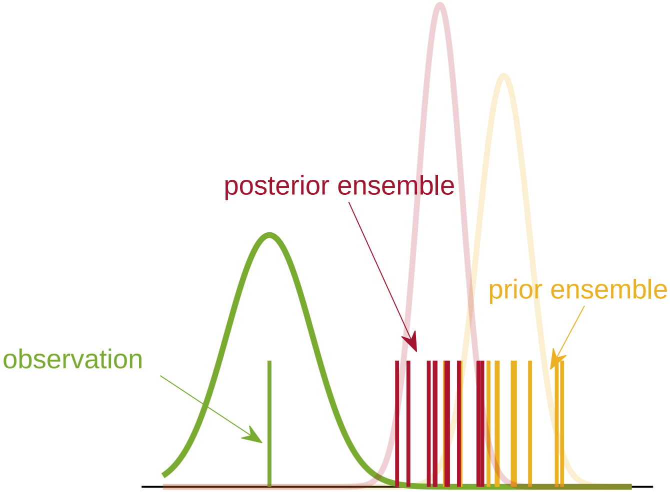
or

$$x_j(t_1) = \Phi(x_j(t_0), t_1)$$

Ensemble Kalman Filter – Forecast



Ensemble Kalman Filter – Analysis



Ensemble Kalman Filter

Start with an ensemble of state values $\{\mathbf{x}_i^{a,j}\}$ $j = 1, \ldots, M$ at time t_i.

Evenly weighted samples of $p^a(\mathbf{x}(t_i) \mid \mathbf{y}_{1:i})$

Ensemble Kalman Filter

Start with an ensemble of state values $\{\mathbf{x}_i^{a,j}\}$ $j = 1, \ldots, M$ at time t_i.

Evenly weighted samples of $p^a(\mathbf{x}(t_i) \mid \mathbf{y}_{1:i})$

1. Evolve each ensemble member according to system dynamics to t_{i+1}

$$\mathbf{x}_{i+1}^{f,j} = \phi(\mathbf{x}_i^{a,j}, t_{i+1})$$

Ensemble Kalman Filter

Start with an ensemble of state values $\{\mathbf{x}_i^{a,j}\}$ $j = 1, \ldots, M$ at time t_i.

Evenly weighted samples of $p^a(\mathbf{x}(t_i) \mid \mathbf{y}_{1:i})$

1. Evolve each ensemble member according to system dynamics to t_{i+1}

$$\mathbf{x}_{i+1}^{f,j} = \phi(\mathbf{x}_i^{a,j}, t_{i+1})$$

2. Compute sample covariance $\hat{\mathbf{P}}_{i+1}^f$, often just $\mathbf{H}\hat{\mathbf{P}}_{i+1}^f$

Ensemble Kalman Filter

Start with an ensemble of state values $\{\mathbf{x}_i^{a,j}\}$ $j = 1, \ldots, M$ at time t_i.

Evenly weighted samples of $p^a(\mathbf{x}(t_i) \mid \mathbf{y}_{1:i})$

1. Evolve each ensemble member according to system dynamics to t_{i+1}

$$\mathbf{x}_{i+1}^{f,j} = \phi(\mathbf{x}_i^{a,j}, t_{i+1})$$

2. Compute sample covariance $\hat{\mathbf{P}}_{i+1}^f$, often just $\mathbf{H}\hat{\mathbf{P}}_{i+1}^f$
3. Update states (below, so-called *perturbed observation EnKF*)

$$\mathbf{x}_{i+1}^{a,j} = \mathbf{x}_{i+1}^{f,j} + \hat{\mathbf{K}}\left(\mathbf{y}_{i+1} + \eta_i - \mathbf{H}\mathbf{x}_{i+1}^{p,j}\right)$$

   where $\eta_i \sim N(\mathbf{0}, \mathbf{R})$ and Kalman gain $\hat{\mathbf{K}}$ uses $\hat{\mathbf{P}}_{i+1}^f$.

   Another possibility — so-called *square root filter*.

Ensemble Kalman Filter

Start with an ensemble of state values $\{\mathbf{x}_i^{a,j}\}$ $j = 1, \ldots, M$ at time t_i.

Evenly weighted samples of $p^a(\mathbf{x}(t_i) \mid \mathbf{y}_{1:i})$

1. Evolve each ensemble member according to system dynamics to t_{i+1}

$$\mathbf{x}_{i+1}^{f,j} = \phi(\mathbf{x}_i^{a,j}, t_{i+1})$$

2. Compute sample covariance $\hat{\mathbf{P}}_{i+1}^f$, often just $\mathbf{H}\hat{\mathbf{P}}_{i+1}^f$
3. Update states (below, so-called *perturbed observation EnKF*)

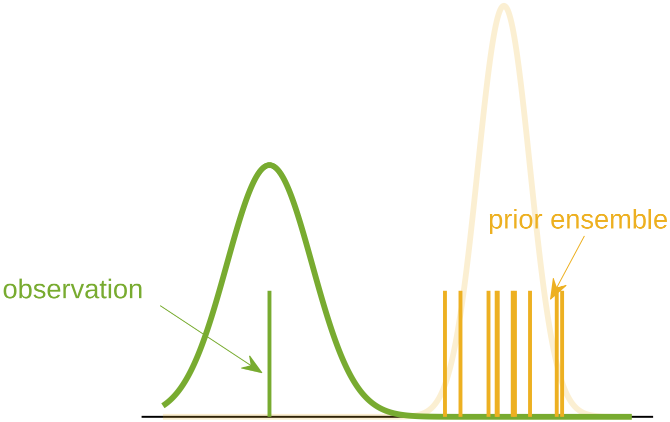
$$\mathbf{x}_{i+1}^{a,j} = \mathbf{x}_{i+1}^{f,j} + \hat{\mathbf{K}}\left(\mathbf{y}_{i+1} + \eta_i - \mathbf{H}\mathbf{x}_{i+1}^{p,j}\right)$$

   where $\eta_i \sim N(\mathbf{0}, \mathbf{R})$ and Kalman gain $\hat{\mathbf{K}}$ uses $\hat{\mathbf{P}}_{i+1}^f$.

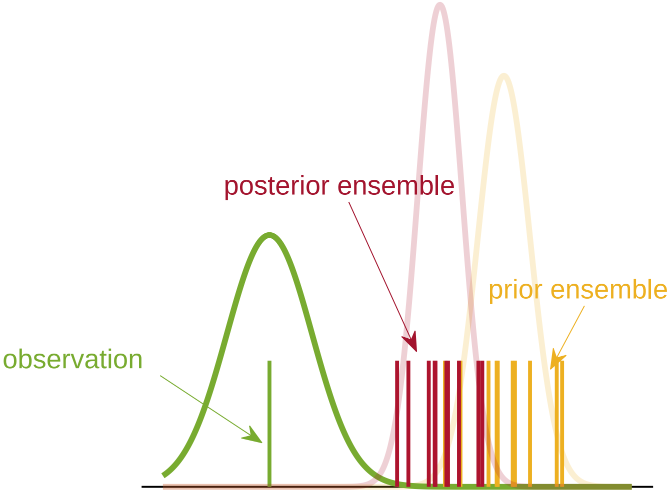
   Another possibility — so-called *square root filter*.

4. Repeat

Ensemble Kalman Filter – Forecast



Ensemble Kalman Filter – Analysis



Ensemble Kalman Filter (EnKF)

Pros:

- Works well for high dimensional state space
  - Why? Open research question
- Relatively easy to implement
- Relatively low computational overhead $M \sim O(10 - 10^2)$

Ensemble Kalman Filter (EnKF)

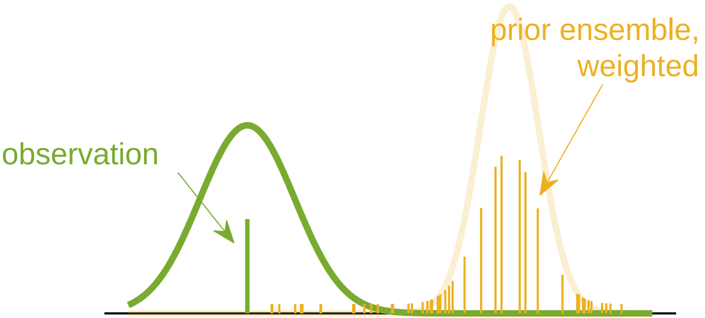
Pros:

- Works well for high dimensional state space
  - Why? Open research question
- Relatively easy to implement
- Relatively low computational overhead $M \sim O(10 - 10^2)$

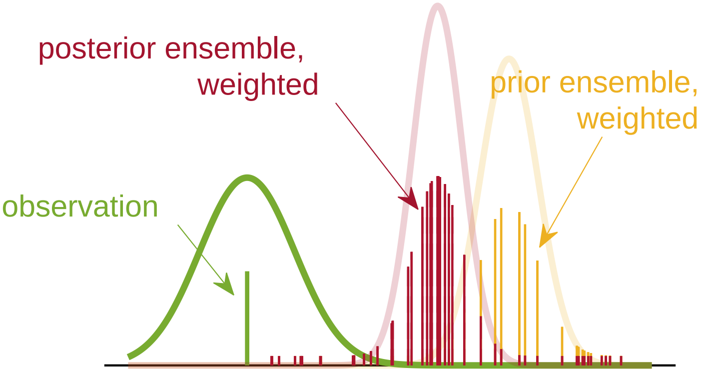
Cons:

- Struggles with nonlinearity
  - think saddles
- Struggles with non-Gaussian probability densities
  - bi-modal
  - skew or crescent shaped

Particle Filter – Forecast



Particle Filter – Analysis



Particle Filter

Start with an ensemble of state values and weights $\{\mathbf{x}_i^j, w_i^j\}$ $j = 1, \ldots, M$ at time t_i.

$$p^a(\mathbf{x}_i \mid \mathbf{y}_{1:i}) \approx \sum_{j=1}^{M} \delta(\mathbf{x} - \mathbf{x}_i^j) w_i^j \qquad \text{at} \quad t = t_i$$

Particle Filter

Start with an ensemble of state values and weights $\{\mathbf{x}_i^j, w_i^j\}$ $j = 1, \ldots, M$ at time t_i.

$$p^a(\mathbf{x}_i \mid \mathbf{y}_{1:i}) \approx \sum_{j=1}^{M} \delta(\mathbf{x} - \mathbf{x}_i^j) w_i^j \qquad \text{at} \quad t = t_i$$

Often, particle filters are called Sequential Monte Carlo, because if we're looking for expectations

$$\mathbb{E}[g(\mathbf{X}_i) \mid \mathbf{y}_{1:i}] = \int g(\mathbf{x}) p^a(\mathbf{x} \mid \mathbf{y}_{1:i}) d\mathbf{x} \approx \sum_{j=1}^{M} g(\mathbf{x}_i^j) w_i^j$$

Particle Filter

Start with an ensemble of state values and weights $\{\mathbf{x}_i^j, w_i^j\}$ $j = 1, \ldots, M$ at time t_i.

1. Evolve each ens member according to system dynamics to t_{i+1}

$$\mathbf{x}_{i+1}^j = \phi(\mathbf{x}_i^j, t_{i+1})$$

Particle Filter

Start with an ensemble of state values and weights $\{\mathbf{x}_i^j, w_i^j\}$ $j = 1, \ldots, M$ at time t_i.

1. Evolve each ens member according to system dynamics to t_{i+1}

$$\mathbf{x}_{i+1}^j = \phi(\mathbf{x}_i^j, t_{i+1})$$

2. Update weight of each ens member using obs $\mathbf{y}_{i+1}$ & likelihood

$$w_{i+1}^j = p^o(\mathbf{y}_{i+1} \mid \mathbf{x}_{i+1}^j) \cdot w_i^j$$

Particle Filter

Start with an ensemble of state values and weights $\{\mathbf{x}_i^j, w_i^j\}$ $j = 1, \ldots, M$ at time t_i.

1. Evolve each ens member according to system dynamics to t_{i+1}

$$\mathbf{x}_{i+1}^j = \phi(\mathbf{x}_i^j, t_{i+1})$$

2. Update weight of each ens member using obs $\mathbf{y}_{i+1}$ & likelihood

$$w_{i+1}^j = p^o(\mathbf{y}_{i+1} \mid \mathbf{x}_{i+1}^j) \cdot w_i^j$$

3. Note forecast and analysis at time t_{i+1} are respectively given by

$$p^f(\mathbf{x}_{i+1} \,|\, \mathbf{y}_{1:i}) \approx \sum_{j=1}^{M} \delta(\mathbf{x} - \mathbf{x}_{i+1}^j) w_i^j \ \ \& \ \ p^a(\mathbf{x}_{i+1} \,|\, \mathbf{y}_{1:i+1}) \approx \sum_{j=1}^{M} \delta(\mathbf{x} - \mathbf{x}_{i+1}^j) w_{i+1}^j$$

Particle Filter

Start with an ensemble of state values and weights $\{\mathbf{x}_i^j, w_i^j\}$ $j = 1, \ldots, M$ at time t_i.

1. Evolve each ens member according to system dynamics to t_{i+1}

$$\mathbf{x}_{i+1}^j = \phi(\mathbf{x}_i^j, t_{i+1})$$

2. Update weight of each ens member using obs $\mathbf{y}_{i+1}$ & likelihood

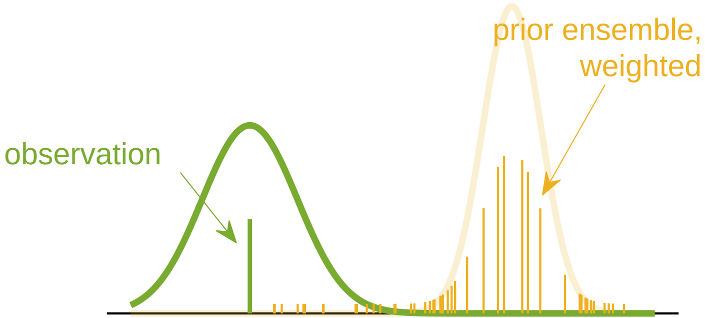
$$w_{i+1}^j = p^o(\mathbf{y}_{i+1} \mid \mathbf{x}_{i+1}^j) \cdot w_i^j$$

3. Note forecast and analysis at time t_{i+1} are respectively given by

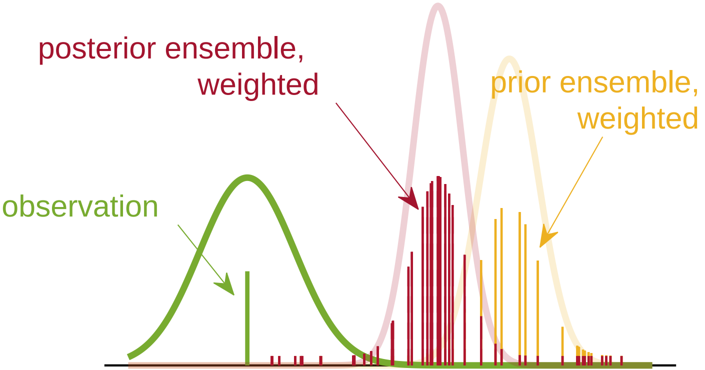
$$p^f(\mathbf{x}_{i+1} \,|\, \mathbf{y}_{1:i}) \approx \sum_{j=1}^{M} \delta(\mathbf{x} - \mathbf{x}_{i+1}^j) w_i^j \;\;\&\;\; p^a(\mathbf{x}_{i+1} \,|\, \mathbf{y}_{1:i+1}) \approx \sum_{j=1}^{M} \delta(\mathbf{x} - \mathbf{x}_{i+1}^j) w_{i+1}^j$$

4. Repeat

Particle Filter – Forecast



Particle Filter – Analysis



Particle Filter, Sequential Monte Carlo (SMC)

Pros:

- Nonlinearity not a problem
  - think saddles
- Works well for non-Gaussian, non-symmetric probability densities
  - bi-modal
  - skew or crescent shaped

Particle Filter, Sequential Monte Carlo (SMC)

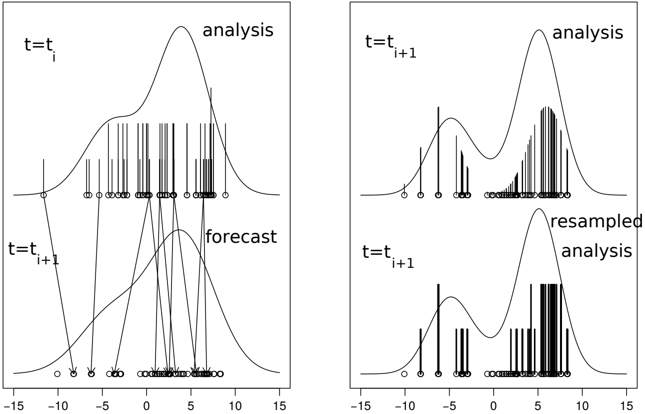
Pros:

- Nonlinearity not a problem
  - think saddles
- Works well for non-Gaussian, non-symmetric probability densities
  - bi-modal
  - skew or crescent shaped

Cons:

- Suffers from "curse of dimensionality"
  - SMC fails in high dimensional systems
- Relatively high computational overhead $M \sim O(10^3 - 10^6)$
- Weight collapses onto a few particles – need resampling

Resampling[4]



[4] borrowed from SIAM UQ16 mini-tutorial of H.R. Künsch

Dowd, *Environmetrics 2006*, **17**: *435-455*
An SMC approach for Marine Ecology

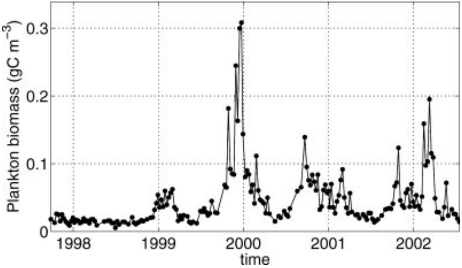


Figure 2. Satellite derived time series of phytoplankton biomass concentration from SeaWiFS in the eastern equatorial Pacific (12^circN 95°W). Dates here indicate the beginning of calendar year

Dowd, *Environmetrics 2006*, **17**: *435-455*
An SMC approach for Marine Ecology

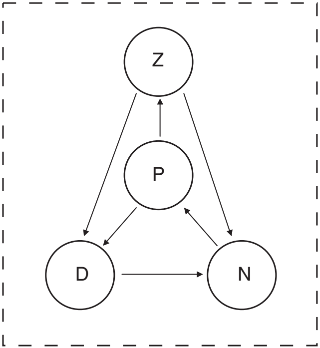


Figure 1. Conceptual diagram of the ecosystem box model. Prognostic ecosystem state variables are phytoplankton (P), zooplankton (Z), nutrients (N) and detritus (D). Arrows represent the direction of mass, or nutrient, fluxes between these populations

Dowd, *Environmetrics 2006*, **17**: *435-455*

An SMC approach for Marine Ecology

$$\frac{dP}{dt} = \gamma_p \frac{N}{\frac{1}{5}+N}P - \frac{1}{10}P - \frac{3}{5}\frac{P}{\frac{1}{10}+P}Z$$

$$\frac{dZ}{dt} = \frac{9}{50}\frac{P}{\frac{1}{10}+P}Z - \frac{1}{10}Z$$

$$\frac{dN}{dt} = \frac{1}{10}D + \frac{12}{50}\frac{P}{\frac{1}{10}+P}Z - \gamma_p \frac{N}{\frac{1}{5}+N}P + \frac{1}{20}Z$$

$$\frac{dD}{dt} = -\frac{1}{10}D + \frac{1}{10}P + \frac{9}{50}\frac{P}{\frac{1}{10}+P}Z + \frac{1}{20}Z$$

Dowd, *Environmetrics 2006*, **17**: *435-455*
An SMC approach for Marine Ecology

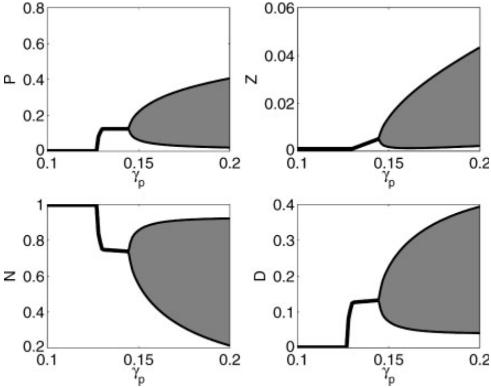
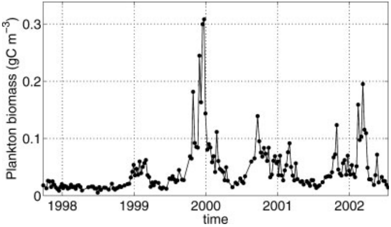
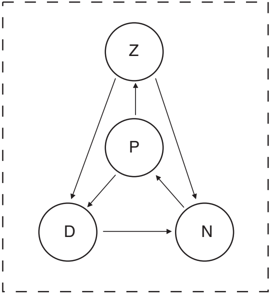


Figure 4. Hopf bifurcation for ecosystem state variables resulting from varying the parameter γ. For γ below bifurcation point, the solid line represents the equilibrium value for the ecosystem state variables (stable attractor). For γ above this value, it represents upper and lower limits of a periodic orbit, with the gray shaded area indicating the range

Dowd, *Environmetrics 2006*, **17**: *435-455*
An SMC approach for Marine Ecology



"+"



Dowd, *Environmetrics 2006*, **17**: *435-455*
An SMC approach for Marine Ecology

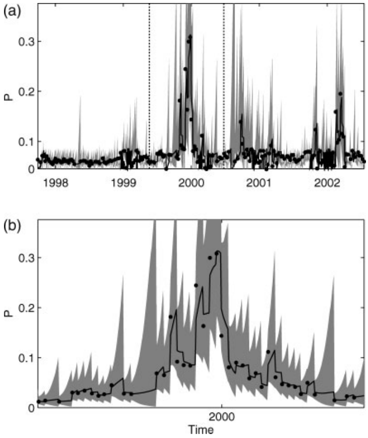


Figure 7. Filtering results for the observed state variable *P*. Shown are the filter estimate of the median (black line), the observations (dots), the approximate 95 per cent confidence intervals (gray shaded area). a: Results for the full analysis period from mid-1997 to mid-2002. b: Results for the central time period as designated in panel a by the vertical-dashed lines

Dowd, *Environmetrics 2006*, **17**: *435-455*
An SMC approach for Marine Ecology

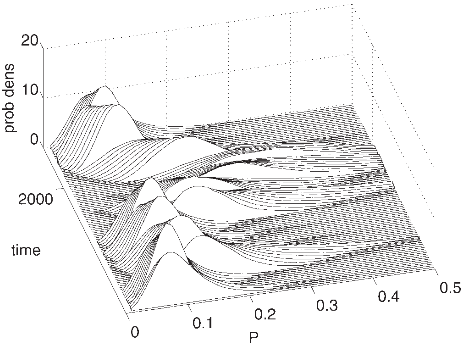


Figure 9. Time evolution of the pro pdf for *P* as estimated by the filter. The period covered is 100 days. The beginning of the 2000 calendar year is indicated

Dowd, *Environmetrics 2006*, **17**: *435-455*
An SMC approach for Marine Ecology

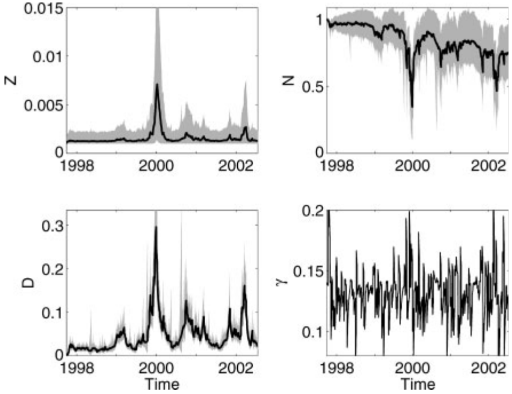


Figure 8. Filtering results for the unobserved state variables Z, N and D, as well as the dynamic parameter γ. The median (black line) is given for all state variables. For Z, N and D the approximate 95 per cent confidence intervals are also shown (gray-shaded area)

Take away

- Data assimilation is a broad framework to combine data and dynamical models
- Often provides both estimates of state and of uncertainty
- Large body of research over last 20 years

Some references

- A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In Handbook on Nonlinear Filtering, Oxford, 2011.
- G. Evensen. Data Assimilation: The Ensemble Kalman Filter, Springer, 2007
- H. R. Künsch, Particle filters. Bernoulli 19, 2013.
- K. Law, A. Stuart, K. Zygalakis. Data Assimilation: A Mathematical Introduction. Springer 2015.
- J.S. Liu. Sequential Monte Carlo methods for dynamic systems. Journal of the American Statistical Association. 93 (443), 1998.
- A. Majda, J. Harlim. Filtering Complex Turbulent Systems. Cambridge 2012.