A review of Data Assimilation methods

Remi Barillec   Dan Cornford

Neural Computing Research Group (NCRG)

Managing Uncertainty in Complex Models

MUCM Theme Day on Data Assimilation
Aston University, 11 March 2009

Outline

Outline

The basics of data assimilation

AIM: Keep track of the evolution of the state of some physical system (the atmosphere, the ocean, a moving object...) in order to estimate and predict.

Ingredients

- The state of the system: $\mathbf{x}_t$
- A forward (observation) model: $\mathbf{x}_t = m_t(\mathbf{x}_{t-1})$
- Observations of the true process: $\mathbf{y}_t = h(\mathbf{x}_t)$

- The state of the system is a vector of all variables necessary to fully describe the system
- Example: $\mathbf{x} = \{Temperature, Salinity, \ldots\}$

The basics of data assimilation

AIM: Keep track of the evolution of the state of some physical system (the atmosphere, the ocean, a moving object...) in order to estimate and predict.

Ingredients

- The state of the system: $\mathbf{x}_t$
- A forward (observation) model: $\mathbf{x}_t = m_t(\mathbf{x}_{t-1})$
- Observations of the true process: $\mathbf{y}_t = h(\mathbf{x}_t)$

- The system model allows us to propagate the state forward in time $\Rightarrow$ prediction / propagation step
- Most of the time, m_t is non-linear

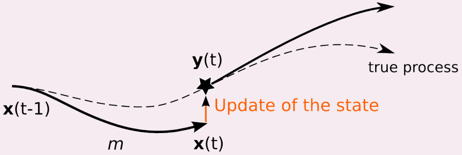
The basics of data assimilation

AIM: Keep track of the evolution of the state of some physical system (the atmosphere, the ocean, a moving object...) in order to estimate and predict.

Ingredients

- The state of the system: $\mathbf{x}_t$
- A forward (observation) model: $\mathbf{x}_t = m_t(\mathbf{x}_{t-1})$
- Observations of the true process: $\mathbf{y}_t = h(\mathbf{x}_t)$

- The model is not a perfect representation of the process and will eventually diverge



- Use observations to correct the propagated state $\Rightarrow$ update step

The basics of data assimilation

AIM: Keep track of the evolution of the state of some physical system (the atmosphere, the ocean, a moving object...) in order to estimate and predict.

Ingredients

- The state of the system: $\mathbf{x}_t$
- A forward (observation) model: $\mathbf{x}_t = m_t(\mathbf{x}_{t-1}) + \eta_t$
- Observations of the true process: $\mathbf{y}_t = h(\mathbf{x}_t) + \varepsilon_t$

- Both the model and the observations are inaccurate
- Use a probabilistic framework to account for uncertainties: $p(\mathbf{x}_t|\mathbf{y}_{1:t})$

The basics of data assimilation

- Bayes' rule allows us to update the state's posterior distribution given the observation and the prior:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1})}{p(\mathbf{y}_t)}$$

This is a filtering update.

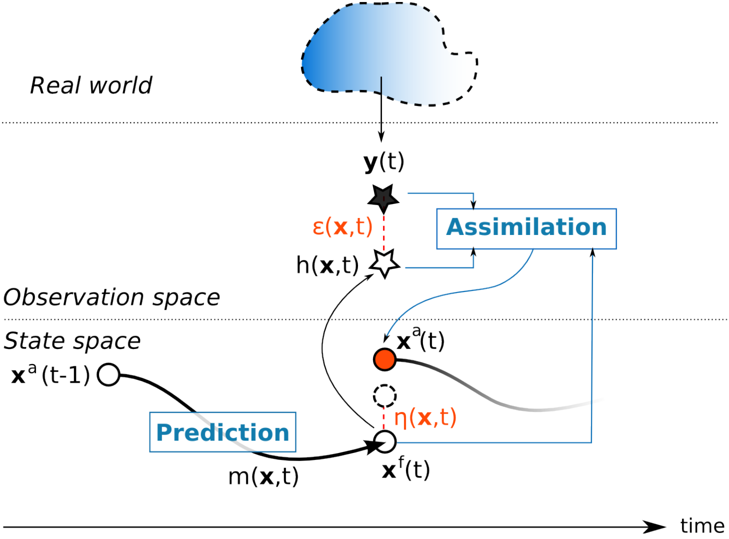
Data assimilation methods

Data assimilation methods can be categorised into:

	Filtering	Smoothing
Probabilistic	Kalman filter (KF)	Kalman smoother (KS)
	Extended KF	Extended KS
	Ensemble KF	Ensemble KS
	Particle filter	Particle smoother
Variational	3D VAR	4D VAR (strong/weak constraint)

Outline

The filtering problem



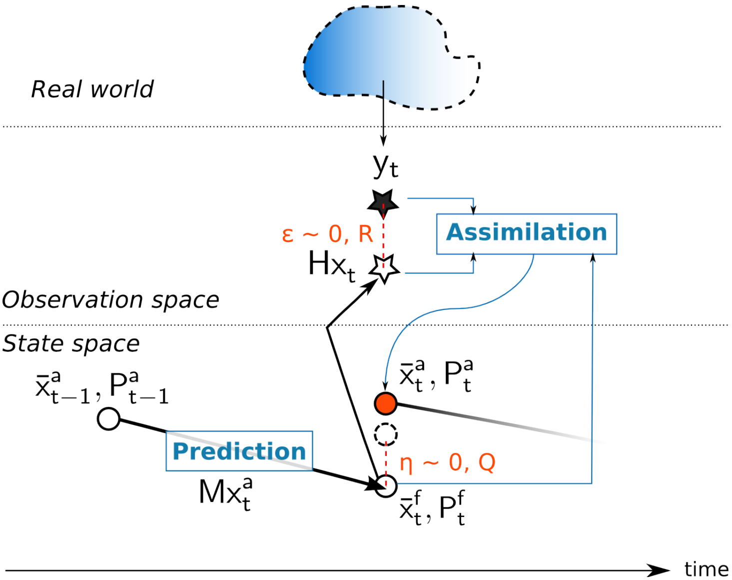
The Kalman Filter

Introduced by R. Kalman (and others) in the early 60's

Assumptions

- The state has a Gaussian distribution: $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\bar{\mathbf{x}}_{t-1}, \mathbf{P}_{t-1})$
- The system model is linear: $m = \mathbf{M}$
- The model error is Gaussian white noise: $\eta \sim \mathcal{N}(0, \mathbf{Q})$
- The observation operator is linear: $h = \mathbf{H}$
- The observation has a Gaussian likelihood: $p(\mathbf{y}_t|\mathbf{x}_t^f) = \mathcal{N}(\mathbf{H}\mathbf{x}_t^f, \mathbf{R})$

The Kalman Filter



The Kalman Filter

Prediction step

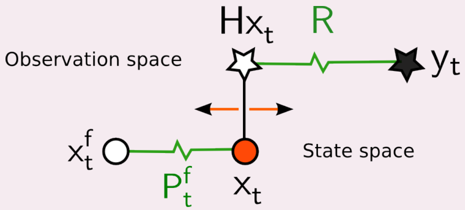
The predicted state $\mathbf{x}_t^f$ has a Gaussian distribution $\mathcal{N}(\bar{\mathbf{x}}_t^f, \mathbf{P}_t^f)$:

$$\bar{\mathbf{x}}_t^f = \mathbf{M}\bar{\mathbf{x}}_{t-1}$$

$$\mathbf{P}_t^f = \mathbf{M}\mathbf{P}_{t-1}\mathbf{M}^T + \mathbf{Q}$$

Update step

- A Best Linear Unbiased Estimated (BLUE) is sought



The Kalman Filter

Prediction step

The predicted state $\mathbf{x}_t^f$ has a Gaussian distribution $\mathcal{N}(\bar{\mathbf{x}}_t^f, \mathbf{P}_t^f)$:

$$\bar{\mathbf{x}}_t^f = \mathbf{M}\bar{\mathbf{x}}_{t-1}$$
$$\mathbf{P}_t^f = \mathbf{M}\mathbf{P}_{t-1}\mathbf{M}^T + \mathbf{Q}$$

Update step

- The updated state $\mathbf{x}_t^a$ has a Gaussian distribution $\mathcal{N}(\bar{\mathbf{x}}_t, \mathbf{P}_t^a)$:

$$\bar{\mathbf{x}}_t^a = \bar{\mathbf{x}}_t^f + \mathbf{K}(\mathbf{y}_t - \mathbf{H}\bar{\mathbf{x}}_t^f)$$
$$\mathbf{P}_t^a = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_t^f$$
$$\mathbf{K} = \mathbf{P}_t^f\mathbf{H}^T(\mathbf{H}\mathbf{P}_t^f\mathbf{H}^T + \mathbf{R})^{-1}$$

Extensions to non-Linear/non-Gaussian case

The Kalman Filter's assumptions are very restrictive:

- What if the model m is non-linear?
- What if the observation operator h is non-linear?
- What if the various distributions (especially the errors) are non-Gaussian?

Several approximations can be introduced:

- Linearise the system and observation operator $\Rightarrow$ Extended KF
- Use a discrete approximation to the state's Gaussian distribution $\Rightarrow$ Ensemble KF, Unscented KF
- Use a Monte-Carlo approximation to the (non-Gaussian) state's distribution $\Rightarrow$ Particle Filter

The Extended Kalman Filter

Assumptions

- Same as the original Kalman Filter, but
- Use the tangent linear model $\hat{\mathbf{M}} = \frac{\partial m}{\partial \mathbf{x}_t}(\mathbf{x}^a_{t-1})$ in the propagation of the covariance

Prediction step

$$\bar{\mathbf{x}}^f_t = m(\bar{\mathbf{x}}^a_{t-1})$$
$$\mathbf{P}^f_t = \hat{\mathbf{M}}\mathbf{P}^a_{t-1}\mathbf{M}^T + \mathbf{Q}$$

The Extended Kalman Filter

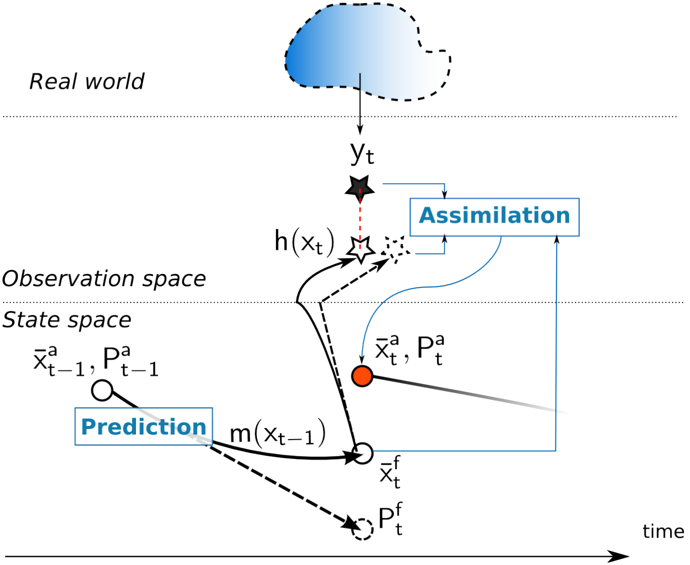
Assumptions

- Same as the original Kalman Filter, but
- Use the tangent linear observation operator $\hat{\mathbf{H}} = \frac{\partial h}{\partial \mathbf{x}_t}(\mathbf{x}_t^f)$ in the update equations

Update step

$$\bar{\mathbf{x}}_t^a = \bar{\mathbf{x}}_t^f + \mathbf{K}(\mathbf{y}_t - h(\bar{\mathbf{x}}_t^f))$$
$$\mathbf{P}_t^a = (\mathbf{I} - \mathbf{K}\hat{\mathbf{H}})\mathbf{P}_t^f$$
$$\mathbf{K} = \mathbf{P}_t^f\hat{\mathbf{H}}^{\mathrm{T}}(\hat{\mathbf{H}}\mathbf{P}_t^f\hat{\mathbf{H}}^{\mathrm{T}} + \mathbf{R})^{-1}$$

The Extended Kalman Filter



The Extended Kalman Filter

- The EKF is a straightforward adaptation of the KF for non-linear models
- However, EKF will fail if the model is strongly non-linear
- Still limited by the Gaussian assumption on all distributions

The Ensemble Kalman Filter

- Rather than linearise the model, approximate the state's distribution
- Use a sample of realisations from which the mean and covariance can be estimated

$$\{\mathbf{x}_i\}_{i=1:N} \sim \mathcal{N}(\bar{\mathbf{x}}, \mathbf{P})$$

$$\bar{\mathbf{x}} \approx \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

$$\mathbf{P} \approx \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathsf{T}}$$

- Other assumptions are the same as in KF

The Ensemble Kalman Filter

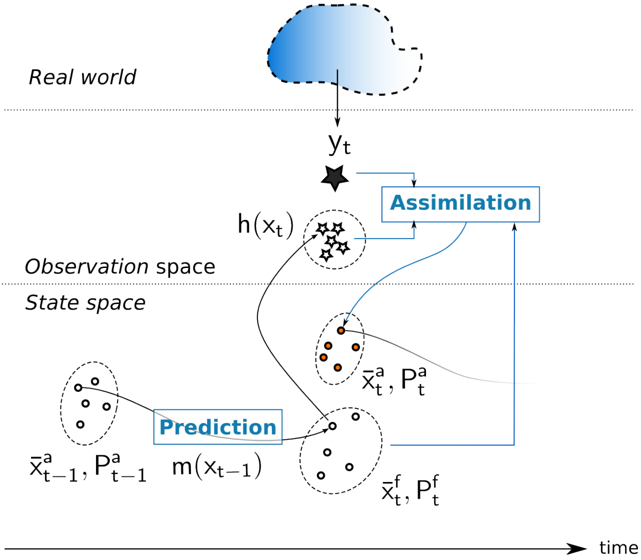
Prediction step

- Each ensemble member is propagated using the full non-linear model
- Mean and covariance are estimated from the propagated ensemble

Update step

- An ensemble of observed realisations is generated by perturbing the observation
- Each ensemble member is updated as in the KF
- This method is known to underestimate the updated covariance
  $\Rightarrow$ Square-root formulations of the filter address this issue

The Ensemble Kalman Filter



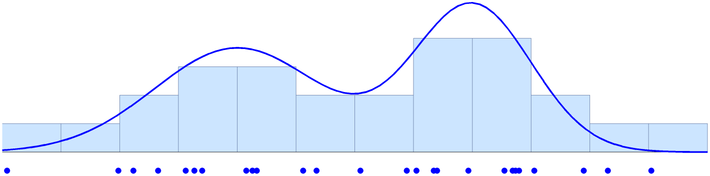
The Ensemble Kalman Filter

- The EnKF provides a robust approach to non-linear filtering
- It is used operationally in weather centres (Canada)
- Still relies on the assumption that all distributions are Gaussian
- Note in operational formulations the covariance matrices are never explicitly computed

The Particle Filter

- Discards the Gaussian assumption Kalman filters are based on
- Instead, represents $\mathrm{p}(\mathbf{x}|\mathbf{y})$ using a Monte-Carlo approximation:
  - a set of particles $\{\mathbf{x}^i\}_{i=1:N}$
  - a set of associated weights $\{w^i\}_{i=1:N}$
- Then

$$\mathrm{p}(\mathbf{x}|\mathbf{y}) \approx \sum_{i=1}^{N} w^i \, \delta(\mathbf{x}^i \in \mathrm{d}\mathbf{x})$$



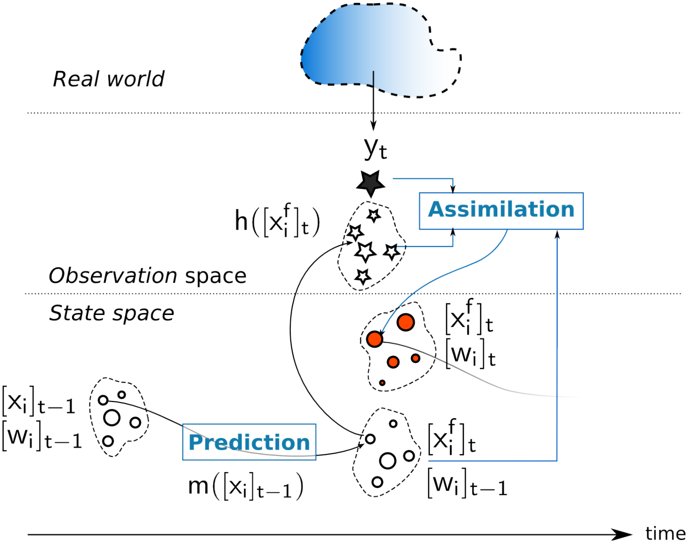
The Particle Filter

- Initialisation:
  - Sample N particles $\mathbf{x}_0^i$ from $p(\mathbf{x}_0)$
  - Set all weights w_0^i to $1/N$
- Prediction:
  - Propagate each particle through the full non-linear model
  - Leave weights unchanged
- Update:
  - Reweight (and normalise) particles according to the observation's likelihood:

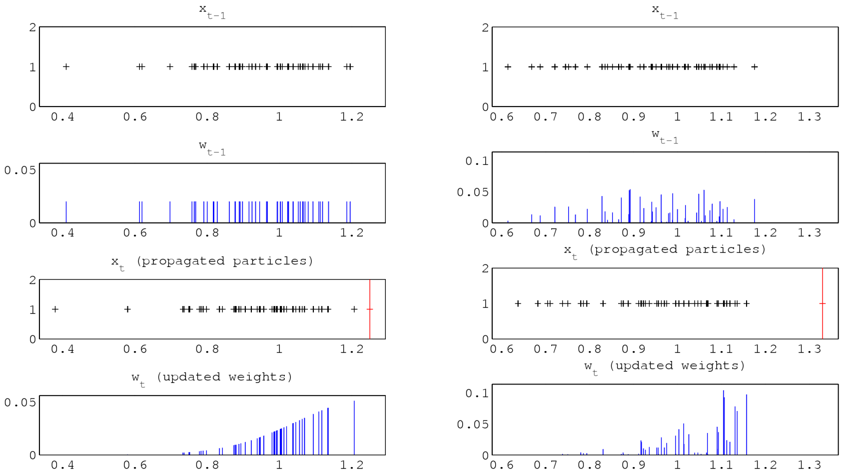
$$w_{\mathrm{t}+1}^i \propto w_{\mathrm{t}}^i \times \mathrm{p}(\mathbf{y}_{\mathrm{t}+1} | \mathbf{x}_{\mathrm{t}+1}^i)$$

  - Leave particles unchanged

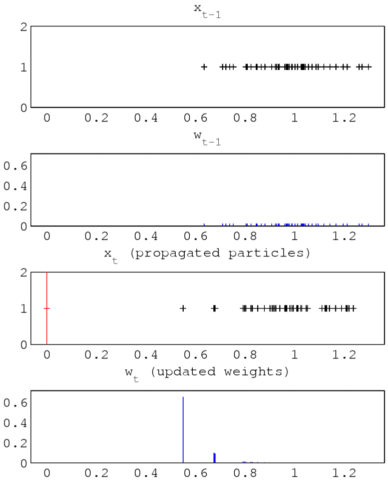
The Particle Filter



The Particle Filter



The Particle Filter



The Particle Filter

- The PF suffers from a degeneracy problem
  $\Rightarrow$ all but a few particles see their weight collapse to zero

- **Solution:** Introduce a resampling step after the update, whereby:
  - particles with high weights are duplicated,
  - particles with no weight are discarded,
  - all weights are reset to $1/N$

- Each particle is replicated M times, where $M \approx w^i \times N$

- Several resampling algorithms available: systematic importance resampling, multinomial resampling, stratified resampling ...

The Particle Filter

- Non-Gaussian, non-linear $\Rightarrow$ ideal in theory!
- However, the number of particles needed grows exponentially with the dimension of the state vector...
  $\Rightarrow$ Totally impractical in high-dimension

Outline

3D VAR

- Variational methods replace the update step with an optimisation problem
- Typically non-probabilistic (i.e. provide a single “best” estimate of the state)

Update step

- Given a first guess of the state (background) $\mathbf{x}^b$ with covariance matrix $\mathbf{B}$ (prior),
- And an observation likelihood with mean $h(\mathbf{x})$ and covariance matrix $\mathbf{R}$,
- The updated state minimises the departure to both the background and the observation:

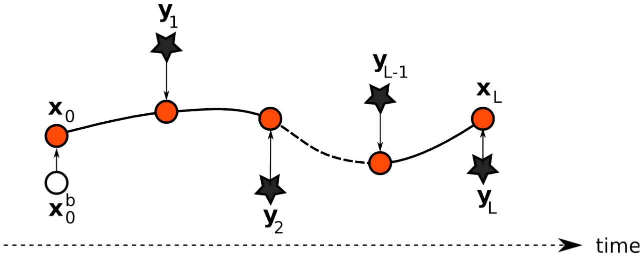
$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}_\mathrm{t} - \mathbf{x}_\mathrm{t}^b)^\mathsf{T}\mathbf{B}^{-1}(\mathbf{x}_\mathrm{t} - \mathbf{x}_\mathrm{t}^b) + \frac{1}{2}(\mathbf{y}_\mathrm{t} - h(\mathbf{x}_\mathrm{t}))\mathbf{R}^{-1}(\mathbf{y}_\mathrm{t} - h(\mathbf{x}_\mathrm{t}))$$

4D VAR - Strong constraint

- 4D VAR adds in the temporal dimension
- The optimal trajectory minimises the departure to both the background and a series of observations:

$$J(\mathbf{x}_{0:L}) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_0^b)^\top \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_0^b) + \frac{1}{2}\sum_{t=1}^{L}(\mathbf{y}_\mathrm{t} - h(\mathbf{x}_\mathrm{t}))\mathbf{R}^{-1}(\mathbf{y}_\mathrm{t} - h(\mathbf{x}_\mathrm{t}))$$

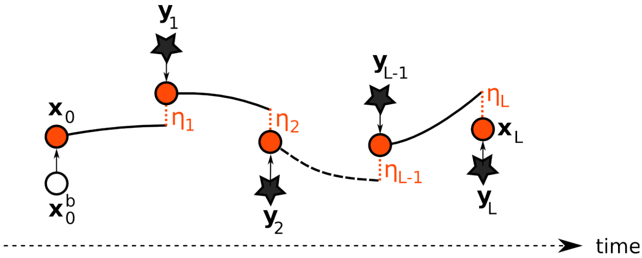
- There is no model error: the model acts as a strong constraint



4D VAR - Weak constraint

- Same as 4D VAR strong constraint, but allows for model error
- The cost function becomes:

$$J(\mathbf{x}_{0:L}) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_0^b)^{\mathrm{T}}\mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_0^b) + \frac{1}{2}\sum_{t=1}^{L}(\mathbf{y}_{\mathrm{t}} - h(\mathbf{x}_{\mathrm{t}}))\mathbf{R}^{-1}(\mathbf{y}_{\mathrm{t}} - h(\mathbf{x}_{\mathrm{t}}))$$
$$+ \frac{1}{2}\sum_{t=1}^{L}\eta_{\mathrm{t}}^{\mathrm{T}}\mathbf{Q}^{-1}\eta_{\mathrm{t}}$$



Variational methods

- 3D VAR and 4D VAR are used operationally in several weather centres (UK, USA, Australia)
- Minimisation of the 4D VAR cost function relies on linearisation of the forward model
- Weak constraint 4D VAR is still very much at development stage

Summary

- Filtering methods provide a probabilistic means to estimate the state in real-time
- Most variational methods are non-probabilistic and replace the update step with an optimisation problem
- Smoothing methods extend filtering to the problem of estimating the trajectory of the state rather than its last value (similar to 4D VAR vs. 3D VAR)