# What to use when?

## With a brief recap

Amos Lawless

Data Assimilation Research Centre & NCEO, Univ. of Reading

With thanks to Ross Bannister

# Reminder: what is data assimilation?

- To blend information from models and observations.
  - State/parameter estimation (some kind of 'optimal' blending).
  - The posterior PDF or certain moments of it.

**Bayes' theorem**

$$p(x|y) = \frac{p(x) \times p(y|x)}{p(y)}$$

$$\text{posterior dist.} = \frac{\text{prior dist.} \times \text{likelihood}}{\text{normalizing constant}}$$

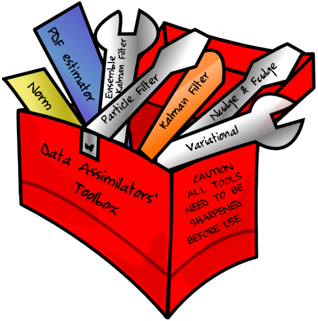
- Prior distribution: PDF of the state before observations are considered (e.g. PDF of model forecast).
- Likelihood: PDF of observations given that the state is x.
- Posterior distribution: PDF of the state given the observations.

# Confused? Overwhelmed?

In realistic practical applications we cannot represent the PDFs explicitly, so we need approximate DA methods

- Variational data assimilation
- Kalman filter (+ extended KF)
- Ensemble Kalman filters
- En-Var filters
- Hybrid methods
- Particle filters

Which method should you use for your application?



# Variational data assimilation

▷ Forecast is mean of the prior, analysis is mode of the posterior (minimises a cost fn); ▷ OK when $n$ is large;.

$$J[\mathrm{x}_0] = \frac{1}{2}(\mathrm{x}_0 - \mathrm{x}^b)^{\mathrm{T}}\mathrm{B}_0^{-1}(\mathrm{x}_0 - \mathrm{x}^b) + \frac{1}{2}\sum_{t=0}^{T}(\mathrm{y}_t - \mathscr{H}_t(\mathrm{x}))^{\mathrm{T}}\mathrm{R}_t^{-1}(\mathrm{y}_t - \mathscr{H}_t(\mathrm{x}))$$

$$\mathrm{x}_{t+1} = \mathscr{M}_t(\mathrm{x}_t)$$

**Flavours**:

- Strong-constraint 4DVar (as above)
- Weak-constraint 4DVar (allow for model errors)
- Incremental version
- 3D-FGAT (incremental version with $\mathrm{M}_t = \mathrm{I}$)
- 3DVar ( $\mathscr{M}_t(\mathrm{x}_t) = \mathrm{x}_t$).

# Variational data assimilation (cont)

Properties:

- Uses observations at correct time.
- Uses dynamical model as a constraint, so can fit changes in observations over a window.
- Weak-constraint formulation allows for model error (but need to specify $Q_t$).
- Assumes Gaussian prior and observations.
- $B_0$ is modelled/parametrised (e.g. need control variable transforms). Not properly flow-dependent and is too simple, but can include balances.
- Analysis is sub-optimal if $\mathscr{M}$ or $\mathscr{H}$ is non-linear; can end up in a local minimum.
- Need tangent linear of $\mathscr{M}_t$ and $\mathscr{H}_t$ and their adjoints (for gradient calculation) - Difficult to develop (time and expertise). But 3D-FGAT avoids this.
- Usually does not provide information on analysis uncertainty.
- Difficult to parallelize.

# The Kalman filter (and extended Kalman filter)

▷ Propagates the mean state and its error covariance sequentially; ▷ forecast/analysis is mean of the prior/posterior; ▷ the analysis is the state that has minimum variance; ▷ strong theoretical basis.

$$
\begin{aligned}
\text{forecast state: } \mathsf{x}_t^{\mathrm{f}} &= \mathscr{M}_t(\mathsf{x}_{t-1}^{\mathrm{a}}) \\
\text{forecast covariance: } \mathsf{P}_t^{\mathrm{f}} &= \mathsf{M}_t \mathsf{P}_{t-1}^{\mathrm{a}} \mathsf{M}_t^{\mathrm{T}} + \mathsf{Q}_t \\
\text{analysis state: } \mathsf{x}_t^{\mathrm{a}} &= \mathsf{x}_t^{\mathrm{f}} + \mathsf{K}_t \left(\mathsf{y}_t - \mathscr{H}_t(\mathsf{x}_t^{\mathrm{f}})\right) \\
\text{Kalman gain: } \mathsf{K}_t &= \mathsf{P}_t^{\mathrm{f}} \mathsf{H}_t^{\mathrm{T}} \left(\mathsf{H}_t \mathsf{P}_t^{\mathrm{f}} \mathsf{H}_t^{\mathrm{T}} + \mathsf{R}_t\right)^{-1} \\
\text{analysis covariances: } \mathsf{P}_t^{\mathrm{a}} &= (\mathsf{I} - \mathsf{K}_t \mathsf{H}_t) \mathsf{P}_t^{\mathrm{f}}
\end{aligned}
$$

# The Kalman filter (and extended Kalman filter) cont.

Properties:

- Allows for correct propagation of forecast covariance matrix.
- Provides estimate of analysis error covariance.
- Allows for model error (but need to specify $Q_t$).
- Assumes Gaussian prior and observations.
- Assumes $\mathscr{M}$ and $\mathscr{H}$ are linear (weak non-linearity is allowed in the extended KF).
- Unfeasible when $n$ is large as matrices are treated explicitly.

# Ensemble Kalman filters

▷ Based on KF equations; ▷ propagates $N$-member ensemble of forecasts to estimate $\mathrm{P}_t^{\mathrm{f}}$.

$$
\begin{aligned}
& \mathrm{x}_t^{(i),\mathrm{f}} &=& \quad \mathscr{M}_t(\mathrm{x}_{t-1}^{(i),\mathrm{a}}) + \beta^{(i)} \\
n \times N: \quad & \mathrm{X}_t^{\prime\mathrm{f}} &=& \quad \begin{pmatrix} \mathrm{x}_t^{(1),\mathrm{f}} - \bar{\mathrm{x}}_t^{\mathrm{f}} & \cdots & \mathrm{x}_t^{(N),\mathrm{f}} - \bar{\mathrm{x}}_t^{\mathrm{f}} \end{pmatrix} \\
p \times N: \quad & \mathrm{Y}_t^{\prime} &=& \quad \begin{pmatrix} \mathscr{H}_t(\mathrm{x}_t^{(1),\mathrm{f}}) - \mathscr{H}(\bar{\mathrm{x}}_t^{\mathrm{f}}) & \cdots & \mathscr{H}_t(\mathrm{x}_t^{(N),\mathrm{f}}) - \mathscr{H}_t(\bar{\mathrm{x}}_t^{\mathrm{f}}) \end{pmatrix} \\
n \times N: \quad & \mathrm{X}_t^{\prime\mathrm{a}} &=& \quad \begin{pmatrix} \mathrm{x}_t^{(1),\mathrm{a}} - \bar{\mathrm{x}}_t^{\mathrm{a}} & \cdots & \mathrm{x}_t^{(N),\mathrm{a}} - \bar{\mathrm{x}}_t^{\mathrm{a}} \end{pmatrix}
\end{aligned}
$$

**Stochastic EnKF**

$$
\mathrm{x}_t^{(i)\mathrm{a}} = \mathrm{x}_t^{(i)\mathrm{f}} + \mathrm{K}_t \left( \mathrm{y}_t + \varepsilon_y^{(i)} - \mathscr{H}_t(\mathrm{x}_t^{(i)\mathrm{f}}) \right)
$$

$$
\mathrm{K}_t = \mathrm{X}_t^{\prime\mathrm{f}} \mathrm{Y}_t^{\prime\mathrm{T}} \left( \mathrm{Y}_t^{\prime} \mathrm{Y}_t^{\prime\,\mathrm{T}} + (N-1)\mathrm{R}_t \right)^{-1}
$$

**Ensemble Transform KF**

$$
\bar{\mathrm{x}}_t^{\mathrm{a}} = \bar{\mathrm{x}}_t^{\mathrm{f}} + \mathrm{K}_t \left( \mathrm{y}_t - \mathscr{H}_t(\bar{\mathrm{x}}_t^{\mathrm{f}}) \right)
$$

$$
\mathrm{X}_t^{\prime\mathrm{a}} = \mathrm{X}_t^{\prime\mathrm{f}} \mathrm{T}_t
$$

$$
\mathrm{K}_t = \mathrm{X}_t^{\prime\mathrm{f}} \mathrm{T}_t \mathrm{T}_t^{\mathrm{T}} \mathrm{Y}_t^{\prime\,\mathrm{T}} \mathrm{R}_t^{-1}
$$

$$
\mathrm{T}_t = \left( \mathrm{I} + \mathrm{Y}_t^{\prime\,\mathrm{T}} \mathrm{R}_t^{-1} \mathrm{Y}_t^{\prime} \right)^{-1/2}
$$

# Ensemble Kalman filters (cont)

## Flavours

- Stochastic EnKF
- Singular Evolutive Interpolated Kalman Filter (SEIK)
- Ensemble Transform Kalman Filter (ETKF)
- Ensemble Adjustment Kalman Filter (EAKF)
- Ensemble Square Root Filter (EnSRF)
- etc.

# Ensemble Kalman filters (cont)

Properties:

- $\mathscr{M}$ and $\mathscr{H}$ can be non-linear.
- Works when $N \ll n$ (but caveats).
- Avoids linear/adjoint coding.
- Easy to code.
- Parallelization is scalable with $N$.
- Assumes Gaussian prior and observations.
- Need localization to deal with sampling noise.
- Localization can disturb physical properties of ensemble (e.g. balance).
- Needs inflation to avoid filter divergence (ensemble under-spread).

# EnVar (ensemble-variational)

▷ As variational DA, but where $\mathrm{B} \rightarrow \mathrm{X}_0'^{\mathrm{f}} \mathrm{X}_0'^{\mathrm{f}\,\mathrm{T}} / (N-1)$ from a parallel ensemble; ▷ analysis increment is a linear combination of forecast ensemble perturbations. E.g. En4DVar:

$$
\begin{aligned}
\mathrm{x}_0^{\mathrm{a}} &= \mathrm{x}_0^{\mathrm{f}} + \mathrm{X}'^{\mathrm{f}} \delta \mathrm{v}_{\mathrm{ens}} / \sqrt{N-1} \qquad \delta \mathrm{v}_{\mathrm{ens}} \text{ is an } N\text{-element vector} \\
J[\delta \mathrm{v}_{\mathrm{ens}}] &= \frac{1}{2} \delta \mathrm{v}_{\mathrm{ens}}^{\mathrm{T}} \delta \mathrm{v}_{\mathrm{ens}} + \frac{1}{2} \sum_{t=0}^{T} \left(\mathrm{y}_t - \mathscr{H}_t(\mathrm{x}_t)\right)^{\mathrm{T}} \mathrm{R}_t^{-1} (\bullet) \\
& \text{subject to } \delta \mathrm{x}_{t+1} = \mathrm{M}_t (\delta \mathrm{x}_t) \quad \text{and } \delta \mathrm{x}_0 = \mathrm{X}'^{\mathrm{f}} \delta \mathrm{v}_{\mathrm{ens}} / \sqrt{N-1} \\
& \qquad \mathrm{x}_{t+1}^{\mathrm{f}} = \mathscr{M}_t \left(\mathrm{x}_t^{\mathrm{f}}\right)
\end{aligned}
$$

# EnVar (ensemble-variational) cont.

Properties:

- Has the benefits of variational DA but with a flow-dependent B-matrix.
- Assumes Gaussian prior and observations.
- Needs localization and a separate parallel ensemble.
- En4DVar still needs the linear model and adjoint. 4DEnVar uses 4D ensembles and avoids these, but localization becomes very difficult.

# Hybrid methods

As variational DA, but where

$$\mathsf{B}_0 \rightarrow (1-\beta)\mathsf{B}_0 + \beta \mathsf{X}_0'^{\mathrm{f}} \mathsf{X}_0'^{\mathrm{f}^{\mathrm{T}}}/(N-1)$$

(new matrix is full rank and flow-dependent).

# Hybrid methods (cont)

Traditional 4DVar with control variable transform:

$$J[\delta \mathrm{v_B}] = \frac{1}{2}\delta \mathrm{v_B^T}\delta \mathrm{v_B} + \frac{1}{2}\sum_{t=0}^{T}\left(\mathrm{y}_t - \mathscr{H}_t(\mathrm{x}_t^{\mathrm{f}}) - \mathrm{H}_t\delta \mathrm{x}_t\right)^{\mathrm{T}}\mathrm{R}_t^{-1}(\bullet)$$

$$\text{subject to } \delta \mathrm{x}_{t+1} = \mathrm{M}_t(\delta \mathrm{x}_t), \quad \delta \mathrm{x}_0 = \mathrm{U}\delta \mathrm{v_B}$$

Hybrid-En4DVar:

$$J[\delta \mathrm{v_B}, \delta \mathrm{v_{ens}}] = \frac{1}{2}\delta \mathrm{v_B^T}\delta \mathrm{v_B} + \frac{1}{2}\delta \mathrm{v_{ens}^T}\delta \mathrm{v_{ens}} + \frac{1}{2}\sum_{t=0}^{T}\left(\mathrm{y}_t - \mathscr{H}_t(\mathrm{x}_t^{\mathrm{f}}) - \mathrm{H}_t\delta \mathrm{x}_t\right)^{\mathrm{T}}\mathrm{R}_t^{-1}(\bullet)$$

$$\text{subject to} \quad \delta \mathrm{x}_{t+1} = \mathrm{M}_t(\delta \mathrm{x}_t), \quad \delta \mathrm{x}_0 = \sqrt{1-\beta}\,\mathrm{U}\delta \mathrm{v_B} + \sqrt{\beta}\,\mathrm{X}'^{\mathrm{f}}\delta \mathrm{v_{ens}}/\sqrt{N-1}$$

# Hybrid methods (cont)

Properties:

- Has the benefits of variational DA but with a full-rank flow-dependent B-matrix.
- Assumes Gaussian prior and observations.
- Still needs localization and a separate parallel ensemble.
- Can get very complex to develop.

# Particle filters

▷ Non-Gaussian; ▷ approximates prior and posterior PDFs as summation of 'delta-functions'. Standard PF:

$$\text{prior PDF: } p(\mathrm{x}) = \sum_{i=1}^{N} w_i^{\text{prior}} \delta(\mathrm{x} - \mathrm{x}_i), \qquad \sum_{i=1}^{N} w_i^{\text{prior}} = 1/N$$

$$\text{posterior PDF: } p(\mathrm{x}|\mathrm{y}) = \sum_{i=1}^{N} w_i^{\text{post}} \delta(\mathrm{x} - \mathrm{x}_i), \qquad w_i^{\text{post}} = \frac{w_i^{\text{prior}} p(\mathrm{y}|\mathrm{x}_i)}{\sum_{i=1}^{N} w_i^{\text{prior}} p(\mathrm{y}|\mathrm{x}_i)}$$

# Particle filters (cont)

Properties:

- Fundamentally no need for covariance matrices - Sample from full pdf.
- No need to assume Gaussianity
- Standard PF is degenerate (weight tends to accumulate for one particle). But several approaches to try to overcome this.
- 'Resampling' still a problem for lots of obs.

# Which method is right for you?

| | Var | KF | EnKF | EnVar | Hybrid | PF |
|---|---|---|---|---|---|---|
| Non-Gaussian | X | X | X | X | X | ✓ |
| Large system | ✓ | X | ✓ | ✓ | ✓ | ✓ |
| Need info on analysis error | X | ✓ | ✓ | X | X | ✓ |
| TLM/ adjoint needed | ✓ | ✓ | X | (✓X) | (✓X) | X |
| Model expensive to run (no more than 50-100 runs) | ✓ | ✓ | ✓ | ✓ | ✓ | X |
| Easily parallelizable | X | X | ✓ | X | X | ✓ |

DA software:
PDAF=Parallel Data Assimilation Framework;

DART=Data Assimilation Research Testbed;

DAPPER=Data assimilation package in Python for experimental research;

JEDI=Joint Effort for Data assimilation Integration;

...

# Which method is right for you?

| | Var | KF | EnKF | EnVar | Hybrid | PF |
|---|---|---|---|---|---|---|
| Non-Gaussian | X | X | X | X | X | ✓ |
| Large system | ✓ | X | ✓ | ✓ | ✓ | ✓ |
| Need info on analysis error | X | ✓ | ✓ | X | X | ✓ |
| TLM/ adjoint needed | ✓ | ✓ | X | (✓X) | (✓X) | X |
| Model expensive to run (no more than 50-100 runs) | ✓ | ✓ | ✓ | ✓ | ✓ | X |
| Easily parallelizable | X | X | ✓ | X | X | ✓ |

DA software:
PDAF=Parallel Data Assimilation Framework;

DART=Data Assimilation Research Testbed;

DAPPER=Data assimilation package in Python for experimental research;

JEDI=Joint Effort for Data assimilation Integration;

...

# Which method is right for you?

| | Var | KF | EnKF | EnVar | Hybrid | PF |
|---|---|---|---|---|---|---|
| Non-Gaussian | X | X | X | X | X | ✓ |
| Large system | ✓ | X | ✓ | ✓ | ✓ | ✓ |
| Need info on analysis error | X | ✓ | ✓ | X | X | ✓ |
| TLM/ adjoint needed | ✓ | ✓ | X | (✓X) | (✓X) | X |
| Model expensive to run (no more than 50-100 runs) | ✓ | ✓ | ✓ | ✓ | ✓ | X |
| Easily parallelizable | X | X | ✓ | X | X | ✓ |

DA software:

PDAF=Parallel Data Assimilation Framework;

DART=Data Assimilation Research Testbed;

DAPPER=Data assimilation package in Python for experimental research;

JEDI=Joint Effort for Data assimilation Integration;

...

# Which method is right for you?

| | Var | KF | EnKF | EnVar | Hybrid | PF |
|---|---|---|---|---|---|---|
| Non-Gaussian | X | X | X | X | X | ✓ |
| Large system | ✓ | X | ✓ | ✓ | ✓ | ✓ |
| Need info on analysis error | X | ✓ | ✓ | X | X | ✓ |
| TLM/ adjoint needed | ✓ | ✓ | X | (✓X) | (✓X) | X |
| Model expensive to run (no more than 50-100 runs) | ✓ | ✓ | ✓ | ✓ | ✓ | X |
| Easily parallelizable | X | X | ✓ | X | X | ✓ |

DA software:
PDAF=Parallel Data Assimilation Framework;

DART=Data Assimilation Research Testbed;

DAPPER=Data assimilation package in Python for experimental research;

JEDI=Joint Effort for Data assimilation Integration;

...

# Which method is right for you?

| | Var | KF | EnKF | EnVar | Hybrid | PF |
|---|---|---|---|---|---|---|
| Non-Gaussian | X | X | X | X | X | ✓ |
| Large system | ✓ | X | ✓ | ✓ | ✓ | ✓ |
| Need info on analysis error | X | ✓ | ✓ | X | X | ✓ |
| TLM/ adjoint needed | ✓ | ✓ | X | (✓X) | (✓X) | X |
| Model expensive to run (no more than 50-100 runs) | ✓ | ✓ | ✓ | ✓ | ✓ | X |
| Easily parallelizable | X | X | ✓ | X | X | ✓ |

DA software:
PDAF=Parallel Data Assimilation Framework;

DART=Data Assimilation Research Testbed;

DAPPER=Data assimilation package in Python for experimental research;

JEDI=Joint Effort for Data assimilation Integration;

...

# Which method is right for you?

| | Var | KF | EnKF | EnVar | Hybrid | PF |
|---|---|---|---|---|---|---|
| Non-Gaussian | X | X | X | X | X | ✓ |
| Large system | ✓ | X | ✓ | ✓ | ✓ | ✓ |
| Need info on analysis error | X | ✓ | ✓ | X | X | ✓ |
| TLM/ adjoint needed | ✓ | ✓ | X | (✓X) | (✓X) | X |
| Model expensive to run (no more than 50-100 runs) | ✓ | ✓ | ✓ | ✓ | ✓ | X |
| Easily parallelizable | X | X | ✓ | X | X | ✓ |

DA software:
PDAF=Parallel Data Assimilation Framework;

DART=Data Assimilation Research Testbed;

DAPPER=Data assimilation package in Python for experimental research;

JEDI=Joint Effort for Data assimilation Integration;

...

# Which method is right for you?

| | Var | KF | EnKF | EnVar | Hybrid | PF |
|---|---|---|---|---|---|---|
| Non-Gaussian | X | X | X | X | X | ✓ |
| Large system | ✓ | X | ✓ | ✓ | ✓ | ✓ |
| Need info on analysis error | X | ✓ | ✓ | X | X | ✓ |
| TLM/ adjoint needed | ✓ | ✓ | X | (✓X) | (✓X) | X |
| Model expensive to run (no more than 50-100 runs) | ✓ | ✓ | ✓ | ✓ | ✓ | X |
| Easily parallelizable | X | X | ✓ | X | X | ✓ |

DA software:
PDAF=Parallel Data Assimilation Framework;

DART=Data Assimilation Research Testbed;

DAPPER=Data assimilation package in Python for experimental research;

JEDI=Joint Effort for Data assimilation Integration;

...

# Which method is right for you?

| | Var | KF | EnKF | EnVar | Hybrid | PF |
|---|---|---|---|---|---|---|
| Non-Gaussian | X | X | X | X | X | ✓ |
| Large system | ✓ | X | ✓ | ✓ | ✓ | ✓ |
| Need info on analysis error | X | ✓ | ✓ | X | X | ✓ |
| TLM/ adjoint needed | ✓ | ✓ | X | (✓X) | (✓X) | X |
| Model expensive to run (no more than 50-100 runs) | ✓ | ✓ | ✓ | ✓ | ✓ | X |
| Easily parallelizable | X | X | ✓ | X | X | ✓ |

DA software:
PDAF=Parallel Data Assimilation Framework;

DART=Data Assimilation Research Testbed;

DAPPER=Data assimilation package in Python for experimental research;

JEDI=Joint Effort for Data assimilation Integration;

...

# What about machine learning?

Can I combine ML with my DA scheme?

Many ideas currently being looked at

- Learn the model and apply DA to the learned model.
- Learn model corrections.
- Learn the observation operator.
- Learn the error covariances.
- Learn the DA scheme.
- ...

Plenty of room for research to come up with something new!