DARC/NCEO Data Assimilation Training course
Data Assimilation Beyond Gaussianity

Jochen Bröcker

DARC/NCEO 2023

How does Gaussianity come into DA

Is non–Gaussianity relevant in DA

Three ways to deal with non–Gaussianity: Parametric approaches

Three ways to deal with non–Gaussianity: Particle filters

Three ways to deal with non–Gaussianity: EnKF

How does Gaussianity come into DA

Is non–Gaussianity relevant in DA

Three ways to deal with non–Gaussianity: Parametric approaches

Three ways to deal with non–Gaussianity: Particle filters

Three ways to deal with non–Gaussianity: EnKF

Setup

Consider *signal process* $\{X_0, X_1, X_2, \ldots\}$ satisfying

$$X_{n+1} = \mathcal{M}(X_n) + R_{n+1}, \qquad n = 0, 1, \ldots,$$

on some *state space* E and with model $\mathcal{M}$. The *observation process* $\{Y_1, Y_2, \ldots\}$ is given by

$$Y_n = \mathcal{H}(X) + S_n, \qquad n = 1, 2, \ldots.$$

Further

- the RV's $\{X_0, R_1, R_2, \ldots, S_1, S_2, \ldots\}$ are all independent,
- $\{R_1, R_2, \ldots\}$ have identical distribution r,
- $\{S_1, S_2, \ldots\}$ have identical distribution s.

Setup

Consider *signal process* $\{X_0, X_1, X_2, \ldots\}$ satisfying

$$X_{n+1} = \mathcal{M}(X_n) + R_{n+1}, \qquad n = 0, 1, \ldots,$$

on some *state space* E and with model $\mathcal{M}$. The *observation process* $\{Y_1, Y_2, \ldots\}$ is given by

$$Y_n = \mathcal{H}(X) + S_n, \qquad n = 1, 2, \ldots.$$

Further

- the RV's $\{X_0, R_1, R_2, \ldots, S_1, S_2, \ldots\}$ are all independent,
- $\{R_1, R_2, \ldots\}$ have identical distribution r,
- $\{S_1, S_2, \ldots\}$ have identical distribution s.

Setup

Consider *signal process* $\{X_0, X_1, X_2, \ldots\}$ satisfying

$$X_{n+1} = \mathcal{M}(X_n) + R_{n+1}, \qquad n = 0, 1, \ldots,$$

on some *state space* E and with model $\mathcal{M}$. The *observation process* $\{Y_1, Y_2, \ldots\}$ is given by

$$Y_n = \mathcal{H}(X) + S_n, \qquad n = 1, 2, \ldots.$$

Further

- the RV's $\{X_0, R_1, R_2, \ldots, S_1, S_2, \ldots\}$ are all independent,
- $\{R_1, R_2, \ldots\}$ have identical distribution r,
- $\{S_1, S_2, \ldots\}$ have identical distribution s.

Goal of data assimilation

Find the conditional distribution

$$\pi_n(x) = \mathbb{P}(X_n = x | Y_1 = y_1, \ldots, Y_n = y_n), \qquad n = 1, 2, \ldots,$$

(which is also a function of $y_1, \ldots, y_n$).

Can be "computed" using two–step recursion:

$$\pi_n^+(x) := \int_E r(x - \mathcal{M}(z)) \cdot \pi_n(z)\, \mathrm{d}z \qquad \textit{prediction step,} \tag{1}$$

$$\pi_{n+1}(x) := c_n \cdot s(y_{n+1} - \mathcal{H}(x)) \cdot \pi_n^+(x) \qquad \textit{update (or Bayesian) step.} \tag{2}$$

Here, c_n is a normalisation factor, and r, s are the densities of model error and obs error, resp.

Goal of data assimilation

Find the conditional distribution

$$\pi_n(x) = \mathbb{P}(X_n = x | Y_1 = y_1, \ldots, Y_n = y_n), \qquad n = 1, 2, \ldots,$$

(which is also a function of $y_1, \ldots, y_n$).
Can be "computed" using two–step recursion:

$$\pi_n^+(x) := \int_E r(x - \mathcal{M}(z)) \cdot \pi_n(z)\, \mathrm{d}z \qquad \textit{prediction step}, \tag{1}$$

$$\pi_{n+1}(x) := c_n \cdot s(y_{n+1} - \mathcal{H}(x)) \cdot \pi_n^+(x) \qquad \textit{update (or Bayesian) step}. \tag{2}$$

Here, c_n is a normalisation factor, and r, s are the densities of model error and obs error, resp.

Linear models with Gaussian errors

Suppose that

- X_0, R_k, and S_k are Gaussian (in particular r, s are Gaussian densities),
- $\mathcal{M}, \mathcal{H}$ are linear mappings (more generally affine),

then π_n is a Gaussian density with some mean and covariance μ_n, Γ_n which are functions of μ_{n-1}, Γ_{n-1} and y_n (*Kalman recursion*).

Linear models with Gaussian errors

Suppose that

- X_0, R_k, and S_k are Gaussian (in particular r, s are Gaussian densities),
- $\mathcal{M}, \mathcal{H}$ are linear mappings (more generally affine),

then π_n is a Gaussian density with some mean and covariance μ_n, Γ_n which are functions of μ_{n-1}, Γ_{n-1} and y_n (*Kalman recursion*).

Deeper reason why linear models with Gaussian errors give Kalman Filter

For Gaussian distributions, the following two facts hold:

1. if X is Gaussian on some space E and A is an affine mapping from E to F, then AX is also Gaussian.
2. if X, Y are jointly Gaussian, then the conditional distribution $Y|X$ is also Gaussian.

Deeper reason why linear models with Gaussian errors give Kalman Filter

For Gaussian distributions, the following two facts hold:

1. if X is Gaussian on some space E and A is an affine mapping from E to F, then AX is also Gaussian.
2. if X, Y are jointly Gaussian, then the conditional distribution $Y|X$ is also Gaussian.

How does Gaussianity come into DA

Is non–Gaussianity relevant in DA

Three ways to deal with non–Gaussianity: Parametric approaches

Three ways to deal with non–Gaussianity: Particle filters

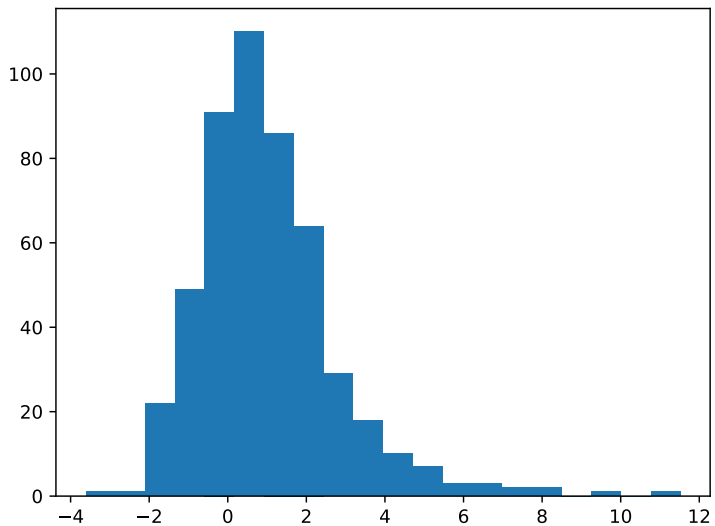
Three ways to deal with non–Gaussianity: EnKF

Reasons for non–Gaussianity: Nonlinear models

If X_n is Gaussian, then $X_{n+1} = \mathcal{M}(X_n) + R_{n+1}$ is generally not Gaussian as soon as $\mathcal{M}$ is nonlinear (whether R is Gaussian or not).

Example

$$X_{n+1} = X_n^2 + R_{n+1}, \qquad X_n \text{ and } R_{n+1} \text{ standard Gaussian.}$$

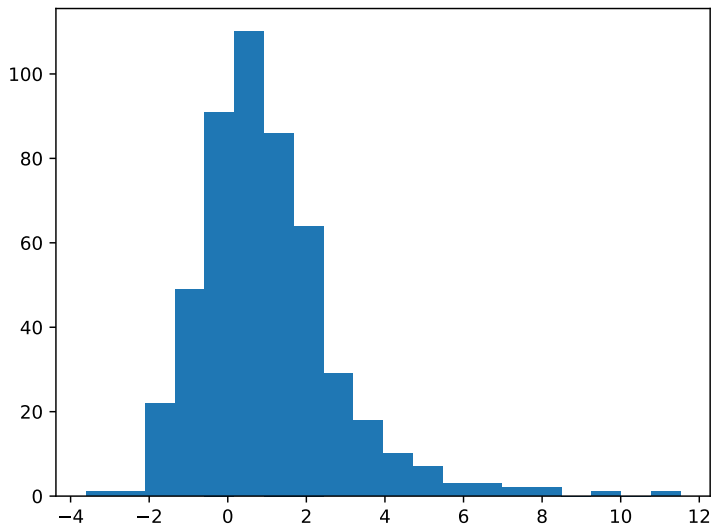


Reasons for non–Gaussianity: Nonlinear models

If X_n is Gaussian, then $X_{n+1} = \mathcal{M}(X_n) + R_{n+1}$ is generally not Gaussian as soon as $\mathcal{M}$ is nonlinear (whether R is Gaussian or not).

Example

$$X_{n+1} = X_n^2 + R_{n+1}, \qquad X_n \text{ and } R_{n+1} \text{ standard Gaussian.}$$

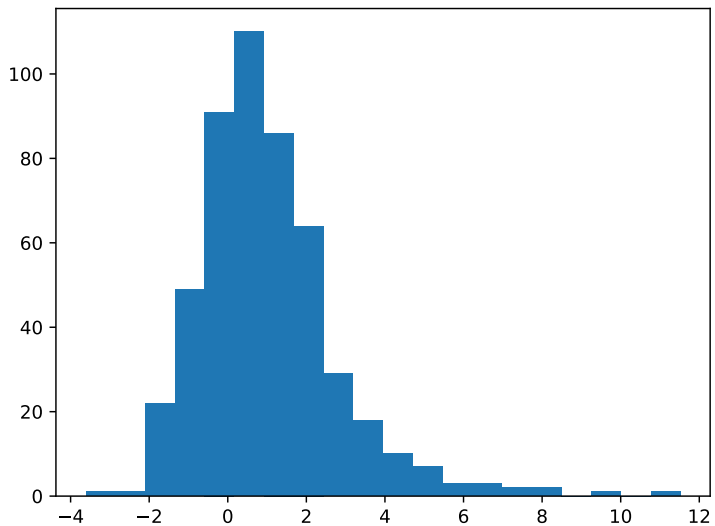


Reasons for non–Gaussianity: Nonlinear models

If X_n is Gaussian, then $X_{n+1} = \mathcal{M}(X_n) + R_{n+1}$ is generally not Gaussian as soon as $\mathcal{M}$ is nonlinear (whether R is Gaussian or not).

Example

$$X_{n+1} = X_n^2 + R_{n+1}, \qquad X_n \text{ and } R_{n+1} \text{ standard Gaussian.}$$



Reasons for non–Gaussianity: Nonlinear observations

If X_n is Gaussian and $Y_n = \mathcal{H}(X_n) + S_n$, then $X_n|Y_n$ is generally not Gaussian as soon as $\mathcal{H}$ is nonlinear (whether S is Gaussian or not).

Example

$$Y_n = X_n^2 + S_n, \qquad X_n \text{ and } S_n \text{ standard normal.}$$

Then $X_n|Y_n \sim \exp(q(x))$ with q a fourth–order polynomial.

Reasons for non–Gaussianity: Nonlinear observations

If X_n is Gaussian and $Y_n = \mathcal{H}(X_n) + S_n$, then $X_n|Y_n$ is generally not Gaussian as soon as $\mathcal{H}$ is nonlinear (whether S is Gaussian or not).

Example

$$Y_n = X_n^2 + S_n, \qquad X_n \text{ and } S_n \text{ standard normal.}$$

Then $X_n|Y_n \sim \exp(q(x))$ with q a fourth–order polynomial.

Reasons for non–Gaussianity: Non–Gaussianity errors

Clearly we also leave Gaussianity if R_n and/or S_n have non–Gaussian distributions.

Many nonlinear observation functions essentially require non–Gaussian error distributions, or in fact non–additive error models. Examples where Gaussian distribution (or additive error model) is not adequate:

- Ratios, rates, percentages and angles are inherently bounded,
- Wind *speed* and precipitation is nonnegative
- Measurement error of variable A depends on A itself or other variable B.

Reasons for non–Gaussianity: Non–Gaussianity errors

Clearly we also leave Gaussianity if R_n and/or S_n have non–Gaussian distributions.
Many nonlinear observation functions essentially require non–Gaussian error distributions, or in fact non–additive error models. Examples where Gaussian distribution (or additive error model) is not adequate:

- Ratios, rates, percentages and angles are inherently bounded,
- Wind *speed* and precipitation is nonnegative
- Measurement error of variable A depends on A itself or other variable B.

Reasons for non–Gaussianity: Non–Gaussianity errors

Clearly we also leave Gaussianity if R_n and/or S_n have non–Gaussian distributions.
Many nonlinear observation functions essentially require non–Gaussian error distributions, or in fact non–additive error models. Examples where Gaussian distribution (or additive error model) is not adequate:

- Ratios, rates, percentages and angles are inherently bounded,
- Wind *speed* and precipitation is nonnegative
- Measurement error of variable A depends on A itself or other variable B.

Reasons for non–Gaussianity: Non–Gaussianity errors

Clearly we also leave Gaussianity if R_n and/or S_n have non–Gaussian distributions.
Many nonlinear observation functions essentially require non–Gaussian error distributions, or in fact non–additive error models. Examples where Gaussian distribution (or additive error model) is not adequate:

- Ratios, rates, percentages and angles are inherently bounded,
- Wind *speed* and precipitation is nonnegative
- Measurement error of variable A depends on A itself or other variable B.

Effects of distributional errors in DA

in fact, any type of error

DA is an *iterative* procedure, hence errors made at some point in time will affect performance at *all* later times and can accumulate, rendering results invalid due to unphysical effects or "numerical fireworks".

More specifically

- the algorithm places probability in regions that are physically implausible or numerically unstable . . .
- while important regimes are deemed unlikely,
- extreme (large magnitude) events might not be represented well,
- computational resources will be allocated poorly.

Effects of distributional errors in DA

in fact, any type of error

DA is an *iterative* procedure, hence errors made at some point in time will affect performance at *all* later times and can accumulate, rendering results invalid due to unphysical effects or "numerical fireworks".
More specifically

- the algorithm places probability in regions that are physically implausible or numerically unstable . . .
- while important regimes are deemed unlikely,
- extreme (large magnitude) events might not be represented well,
- computational resources will be allocated poorly.

Effects of distributional errors in DA

in fact, any type of error

DA is an *iterative* procedure, hence errors made at some point in time will affect performance at *all* later times and can accumulate, rendering results invalid due to unphysical effects or "numerical fireworks".
More specifically

- the algorithm places probability in regions that are physically implausible or numerically unstable . . .
- while important regimes are deemed unlikely,
- extreme (large magnitude) events might not be represented well,
- computational resources will be allocated poorly.

Effects of distributional errors in DA

in fact, any type of error

DA is an *iterative* procedure, hence errors made at some point in time will affect performance at *all* later times and can accumulate, rendering results invalid due to unphysical effects or "numerical fireworks".
More specifically

- the algorithm places probability in regions that are physically implausible or numerically unstable . . .
- while important regimes are deemed unlikely,
- extreme (large magnitude) events might not be represented well,
- computational resources will be allocated poorly.

Effects of distributional errors in DA

in fact, any type of error

DA is an *iterative* procedure, hence errors made at some point in time will affect performance at *all* later times and can accumulate, rendering results invalid due to unphysical effects or "numerical fireworks".
More specifically

- the algorithm places probability in regions that are physically implausible or numerically unstable . . .
- while important regimes are deemed unlikely,
- extreme (large magnitude) events might not be represented well,
- computational resources will be allocated poorly.

The poor man's remedy

DA attempts to "optimally" combine information from present observations with information from past observations, propagated forward.

Strategy to avoid error accumulation

Do not trust your own computations!

More precisely, stronger weight is placed on current rather than past information, e.g. by inflating the model error. Problems with this strategy:

- not optimal as potentially useful information from the past is ignored,
- often still leads to rather unphysical effects,
- each observation represents a big update (rather than a small perturbation) which is a burden on computational resources.

The poor man's remedy

DA attempts to "optimally" combine information from present observations with information from past observations, propagated forward.

Strategy to avoid error accumulation

Do not trust your own computations!

More precisely, stronger weight is placed on current rather than past information, e.g. by inflating the model error. Problems with this strategy:

- not optimal as potentially useful information from the past is ignored,
- often still leads to rather unphysical effects,
- each observation represents a big update (rather than a small perturbation) which is a burden on computational resources.

The poor man's remedy

DA attempts to "optimally" combine information from present observations with information from past observations, propagated forward.

Strategy to avoid error accumulation

Do not trust your own computations!

More precisely, stronger weight is placed on current rather than past information, e.g. by inflating the model error. Problems with this strategy:

- not optimal as potentially useful information from the past is ignored,
- often still leads to rather unphysical effects,
- each observation represents a big update (rather than a small perturbation) which is a burden on computational resources.

The poor man's remedy

DA attempts to "optimally" combine information from present observations with information from past observations, propagated forward.

Strategy to avoid error accumulation

Do not trust your own computations!

More precisely, stronger weight is placed on current rather than past information, e.g. by inflating the model error. Problems with this strategy:

- not optimal as potentially useful information from the past is ignored,
- often still leads to rather unphysical effects,
- each observation represents a big update (rather than a small perturbation) which is a burden on computational resources.

The poor man's remedy

DA attempts to "optimally" combine information from present observations with information from past observations, propagated forward.

Strategy to avoid error accumulation

Do not trust your own computations!

More precisely, stronger weight is placed on current rather than past information, e.g. by inflating the model error. Problems with this strategy:

- not optimal as potentially useful information from the past is ignored,
- often still leads to rather unphysical effects,
- each observation represents a big update (rather than a small perturbation) which is a burden on computational resources.

How does Gaussianity come into DA

Is non–Gaussianity relevant in DA

Three ways to deal with non–Gaussianity: Parametric approaches

Three ways to deal with non–Gaussianity: Particle filters

Three ways to deal with non–Gaussianity: EnKF

Parametric families

A *parametric family* is a set of probability densities $\{p(x, \theta), \theta \in \Theta\}$ where Θ is some parameter space.

Idea

Find parametric family so that filtering process is given by $\pi_n(x) = p(x, \theta_n)$, where $\theta_1, \theta_2, \ldots$ satisfy the finite–dimensional system

$$\theta_{n+1} = F(\theta_n, y_{n+1}), \qquad n = 0, 1, 2, \ldots \qquad (\text{“Meta–Model”}).$$

The Kalman Filter is an example, with the Gaussian densities as parametric family.

Unfortunately, . . .

apart from the Kalman Filter, there are very few examples where this works (of hardly any practical relevance).

Parametric families

A *parametric family* is a set of probability densities $\{p(x, \theta), \theta \in \Theta\}$ where Θ is some parameter space.

Idea

Find parametric family so that filtering process is given by $\pi_n(x) = p(x, \theta_n)$, where $\theta_1, \theta_2, \ldots$ satisfy the finite–dimensional system

$$\theta_{n+1} = F(\theta_n, y_{n+1}), \qquad n = 0, 1, 2, \ldots \qquad (\text{“Meta–Model”}).$$

The Kalman Filter is an example, with the Gaussian densities as parametric family.

Unfortunately, ...

apart from the Kalman Filter, there are very few examples where this works (of hardly any practical relevance).

Sufficient statistics

. . . but useful as an approximation. Given parametric family $\{p(x, \theta), \theta \in \Theta\}$, suppose there is a function Φ so that the equation

$$m = \int_E \Phi(x) p(x, \theta) \, \mathrm{d}x \tag{3}$$

can be solved for θ.

Example is the Gaussian family with $\Phi(x) = (x, x^2)$ (i.e. first and second moment).

Sufficient statistics

. . . but useful as an approximation. Given parametric family $\{p(x, \theta), \theta \in \Theta\}$, suppose there is a function Φ so that the equation

$$m = \int_E \Phi(x) p(x, \theta) \, \mathrm{d}x \tag{3}$$

can be solved for θ.
Example is the Gaussian family with $\Phi(x) = (x, x^2)$ (i.e. first and second moment).

Assumed density filters, moment matching filters, ...

Approximations ρ_n of π_n can be calculated iteratively as follows:

1. Suppose ρ_n is given. Calculate predicted moments

$$m_n = \int_{E \times E} \Phi(\mathcal{M}(x) + \xi) \rho_n(x) r(\xi) \, \mathrm{d}x \mathrm{d}\xi$$

2. Use m_n as lhs in Eq (3) and solve for θ_n.
3. Perform update step (exact Bayesian step)

$$\rho_{n+1}(x) = c_n \cdot s(y_{n+1} - \mathcal{H}(x)) \cdot p(x, \theta_n)$$

Applications: Fast but small scale such as target tracking, robotics, finance, ...

Assumed density filters, moment matching filters, ...

Approximations ρ_n of π_n can be calculated iteratively as follows:

1. Suppose ρ_n is given. Calculate predicted moments

$$m_n = \int_{E \times E} \Phi(\mathcal{M}(x) + \xi) \rho_n(x) r(\xi) \, \mathrm{d}x \mathrm{d}\xi$$

2. Use m_n as lhs in Eq (3) and solve for θ_n.
3. Perform update step (exact Bayesian step)

$$\rho_{n+1}(x) = c_n \cdot s(y_{n+1} - \mathcal{H}(x)) \cdot p(x, \theta_n)$$

Applications: Fast but small scale such as target tracking, robotics, finance, ...

Assumed density filters, moment matching filters, ...

Approximations ρ_n of π_n can be calculated iteratively as follows:

1. Suppose ρ_n is given. Calculate predicted moments

$$m_n = \int_{E \times E} \Phi(\mathcal{M}(x) + \xi) \rho_n(x) r(\xi) \, \mathrm{d}x \mathrm{d}\xi$$

2. Use m_n as lhs in Eq (3) and solve for θ_n.
3. Perform update step (exact Bayesian step)

$$\rho_{n+1}(x) = c_n \cdot s(y_{n+1} - \mathcal{H}(x)) \cdot p(x, \theta_n)$$

Applications: Fast but small scale such as target tracking, robotics, finance, ...

Assumed density filters, moment matching filters, ...

Approximations ρ_n of π_n can be calculated iteratively as follows:

1. Suppose ρ_n is given. Calculate predicted moments

$$m_n = \int_{E \times E} \Phi(\mathcal{M}(x) + \xi)\rho_n(x) r(\xi)\, \mathrm{d}x\mathrm{d}\xi$$

2. Use m_n as lhs in Eq (3) and solve for θ_n.
3. Perform update step (exact Bayesian step)

$$\rho_{n+1}(x) = c_n \cdot s(y_{n+1} - \mathcal{H}(x)) \cdot p(x, \theta_n)$$

Applications: Fast but small scale such as target tracking, robotics, finance, ...

How does Gaussianity come into DA

Is non–Gaussianity relevant in DA

Three ways to deal with non–Gaussianity: Parametric approaches

Three ways to deal with non–Gaussianity: Particle filters

Three ways to deal with non–Gaussianity: EnKF

Basic Monte Carlo

Basic idea of Monte Carlo

Let $X \sim p(x)$ (density) and ϕ some function, then

$$\mathbb{E}(\phi(X)) = \int \phi(x)p(x)\,\mathrm{d}x \cong \frac{1}{K}\sum_{k=1}^{K}\phi(X^{(k)})$$

where $X^{(1)}, X^{(2)}, \ldots$ are independent with distribution $p(x)$.

Weighted Monte Carlo

Yet better idea: weighted Monte Carlo

Let $X \sim p(x)$ (density) and ϕ some function, then

$$\begin{aligned}\mathbb{E}(\phi(X)) &= \int \phi(x)p(x)\,\mathrm{d}x \\ &= \int \phi(x)\frac{p(x)}{q(x)}q(x)\,\mathrm{d}x \\ &\cong \frac{1}{K}\sum_{k=1}^{K}\phi(X^{(k)})\frac{p(X^{(k)})}{q(X^{(k)})}\end{aligned}$$

where $X^{(1)}, X^{(2)}, \ldots$ are independent with distribution $q(x)$.

Heuristically the samples $X^{(k)}$ with corresponding weights $w_k = \frac{p(X^{(k)})}{q(X^{(k)})}$ for $k = 1, 2, \ldots$ represent the distribution p.

Weighted Monte Carlo

Yet better idea: weighted Monte Carlo

Let $X \sim p(x)$ (density) and ϕ some function, then

$$\begin{aligned}\mathbb{E}(\phi(X)) &= \int \phi(x)p(x)\,\mathrm{d}x \\ &= \int \phi(x)\frac{p(x)}{q(x)}q(x)\,\mathrm{d}x \\ &\cong \frac{1}{K}\sum_{k=1}^{K}\phi(X^{(k)})\frac{p(X^{(k)})}{q(X^{(k)})}\end{aligned}$$

where $X^{(1)}, X^{(2)}, \ldots$ are independent with distribution $q(x)$.

Heuristically the samples $X^{(k)}$ with corresponding weights $w_k = \frac{p(X^{(k)})}{q(X^{(k)})}$ for $k = 1, 2, \ldots$ represent the distribution p.

A first particle filter

Does not quite work yet

Let $X_n^{(1)}, \ldots, X_n^{(K)}$ with weights $w_n^{(1)}, \ldots, w_n^{(K)}$ an approximation to π_n.

Algorithm I

1. Create new particles using model

$$X_{n+1}^{(k)} = \mathcal{M}(X_n^{(k)}) + R^{(k)}, \qquad k = 1, \ldots, K$$

(The new samples $X_{n+1}^{(k)}$ with weights $w_n^{(1)}, \ldots, w_n^{(K)}$ represent an approximation to π_n^+.)

2. Update weights according to update step

$$w_{n+1}^{(k)} = c_{n+1} \cdot w_n^{(k)} \cdot s(y_{n+1} - \mathcal{H}(X_{n+1}^{(k)})) \qquad k = 1, \ldots, K$$

(c_n is a normalisation constant so that $\sum_k w_{n+1}^{(k)} = 1$.)

This approach suffers from weight degeneration.

A first particle filter

Does not quite work yet

Let $X_n^{(1)}, \ldots, X_n^{(K)}$ with weights $w_n^{(1)}, \ldots, w_n^{(K)}$ an approximation to π_n.

Algorithm I

1. Create new particles using model

$$X_{n+1}^{(k)} = \mathcal{M}(X_n^{(k)}) + R^{(k)}, \qquad k = 1, \ldots, K$$

(The new samples $X_{n+1}^{(k)}$ with weights $w_n^{(1)}, \ldots, w_n^{(K)}$ represent an approximation to π_n^+.)

2. Update weights according to update step

$$w_{n+1}^{(k)} = c_{n+1} \cdot w_n^{(k)} \cdot s(y_{n+1} - \mathcal{H}(X_{n+1}^{(k)})) \qquad k = 1, \ldots, K$$

(c_n is a normalisation constant so that $\sum_k w_{n+1}^{(k)} = 1$.)

This approach suffers from weight degeneration.

A first particle filter

Does not quite work yet

Let $X_n^{(1)}, \ldots, X_n^{(K)}$ with weights $w_n^{(1)}, \ldots, w_n^{(K)}$ an approximation to π_n.

Algorithm I

1. Create new particles using model

$$X_{n+1}^{(k)} = \mathcal{M}(X_n^{(k)}) + R^{(k)}, \qquad k = 1, \ldots, K$$

(The new samples $X_{n+1}^{(k)}$ with weights $w_n^{(1)}, \ldots, w_n^{(K)}$ represent an approximation to π_n^+.)

2. Update weights according to update step

$$w_{n+1}^{(k)} = c_{n+1} \cdot w_n^{(k)} \cdot s(y_{n+1} - \mathcal{H}(X_{n+1}^{(k)})) \qquad k = 1, \ldots, K$$

(c_n is a normalisation constant so that $\sum_k w_{n+1}^{(k)} = 1$.)

This approach suffers from weight degeneration.

A resampling particle filter

which avoids degeneracy

Let $X_n^{(k)}$ with weights $w_n^{(k)}$, $k = 1, \ldots, K$ approximate π_n.

Algorithm II

1. Each particle $X_n^{(k)}$ generates *multiple* (or potentially no) offspring according to weight $w_n^{(k)}$

$$X_{n+1}^{(l)} = \mathcal{M}(X_n^{(k)}) + R^{(l)}, \qquad l = 1, \ldots, L_k$$

so that $L_k \cong K w_n^{(k)}$.

2. Repeat step 1 for each $k = 1, \ldots, K$. New samples $X_{n+1}^{(k)}$ with *equal* weights represent an approximation to π_n^+.

3. Compute new weights according to update step

$$w_{n+1}^{(k)} = c_{n+1} \cdot s(y_{n+1} - \mathcal{H}(X_{n+1}^{(k)})) \qquad k = 1, \ldots, K$$

(Again, c_n is a normalisation constant.)

A resampling particle filter

which avoids degeneracy

Let $X_n^{(k)}$ with weights $w_n^{(k)}$, $k = 1, \ldots, K$ approximate π_n.

Algorithm II

1. Each particle $X_n^{(k)}$ generates *multiple* (or potentially no) offspring according to weight $w_n^{(k)}$

$$X_{n+1}^{(l)} = \mathcal{M}(X_n^{(k)}) + R^{(l)}, \qquad l = 1, \ldots, L_k$$

   so that $L_k \cong K w_n^{(k)}$.

2. Repeat step 1 for each $k = 1, \ldots, K$. New samples $X_{n+1}^{(k)}$ with *equal* weights represent an approximation to π_n^+.

3. Compute new weights according to update step

$$w_{n+1}^{(k)} = c_{n+1} \cdot s(y_{n+1} - \mathcal{H}(X_{n+1}^{(k)})) \qquad k = 1, \ldots, K$$

   (Again, c_n is a normalisation constant.)

A resampling particle filter

which avoids degeneracy

Let $X_n^{(k)}$ with weights $w_n^{(k)}$, $k = 1, \ldots, K$ approximate π_n.

Algorithm II

1. Each particle $X_n^{(k)}$ generates *multiple* (or potentially no) offspring according to weight $w_n^{(k)}$

$$X_{n+1}^{(l)} = \mathcal{M}(X_n^{(k)}) + R^{(l)}, \qquad l = 1, \ldots, L_k$$

   so that $L_k \cong K w_n^{(k)}$.

2. Repeat step 1 for each $k = 1, \ldots, K$. New samples $X_{n+1}^{(k)}$ with *equal* weights represent an approximation to π_n^+.

3. Compute new weights according to update step

$$w_{n+1}^{(k)} = c_{n+1} \cdot s(y_{n+1} - \mathcal{H}(X_{n+1}^{(k)})) \qquad k = 1, \ldots, K$$

   (Again, c_n is a normalisation constant.)

How does Gaussianity come into DA

Is non–Gaussianity relevant in DA

Three ways to deal with non–Gaussianity: Parametric approaches

Three ways to deal with non–Gaussianity: Particle filters

Three ways to deal with non–Gaussianity: EnKF

Thank you!


Andrew H. Jazwinski.
Stochastic Processes and Filtering Theory, volume 64 of *Mathematics in Science and Engineering*.
Academic Press, 1970.
ISBN 9780123815507.


Alan Bain and Dan Crisan.
Fundamentals of Stochastic Filtering, volume 60 of *Stochastic Modelling and Applied Probability*.
Springer-Verlag, New York, first edition, 2010.


N. J. Gordon, D. J. Salmond, and A. F. M. Smith.
Novel approach to nonlinear/nongaussian bayesian state estimation.
IEE Proceedings, Part F, 140(2), 1993.