An Introduction to

Data Assimilation

Presenter: Yuxi Zhang

Introduction

Assume:

A free-falling object with initial velocity of

$$v_0 = 1 \pm 0.3 \text{ m/s}$$

At $t = 1$ s, what is the speed of the object?

It is simple to calculate with the acceleration formula:

$$v_{t=1} = v_0 + gt = 1\ (\pm 0.3) + 9.8 \times 1 = 10.8\ (\pm 0.3)\ \text{m/s}.$$

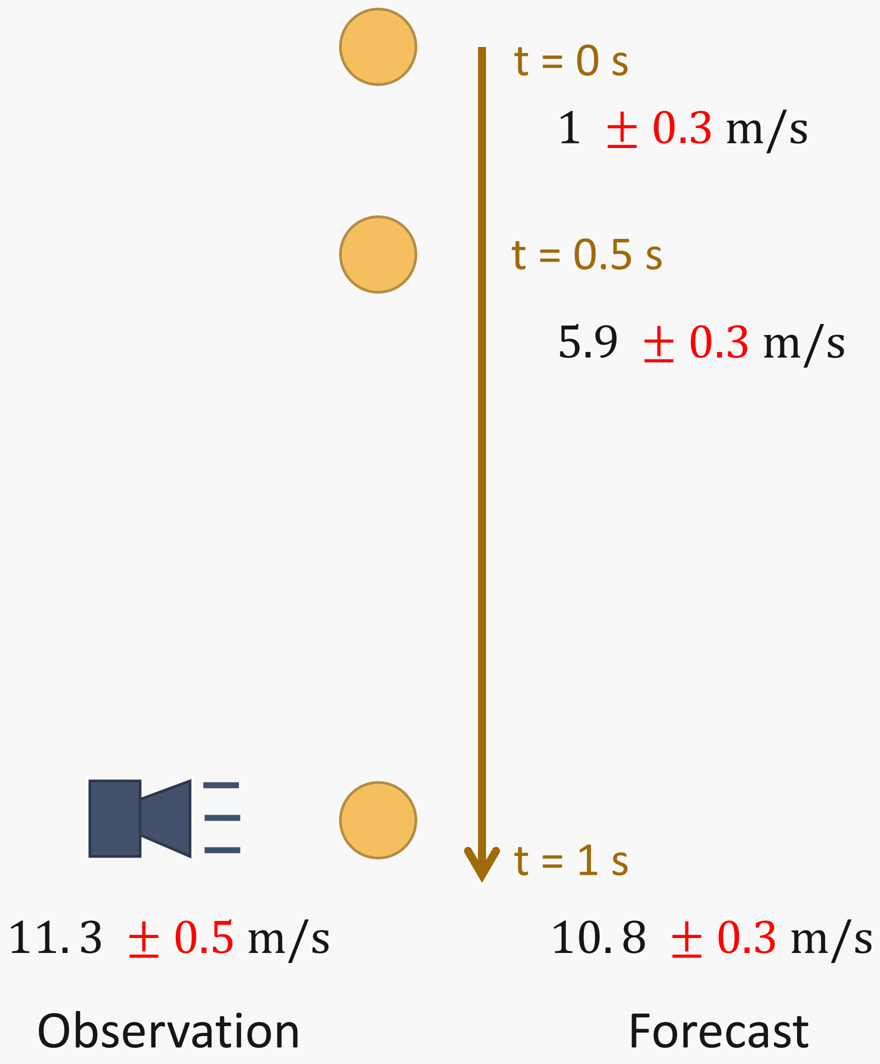
Now assume we have an additional information:

At $t = 1$ s, a sensor measured the speed of the object as 11.3 ± 0.5 m/s.

In this case, how do you estimate the true velocity of the object at $t = 1$ s?

Which data do you trust?

The purpose of data assimilation is to find the best estimate of the 'true' state.



Introduction

Forecast: $x_f = 10.8, \sigma_f = 0.3$

Observation: $x_o = 11.3, \sigma_o = 0.5$

Best Linear Unbiased Estimator (BLUE)

Analysis:

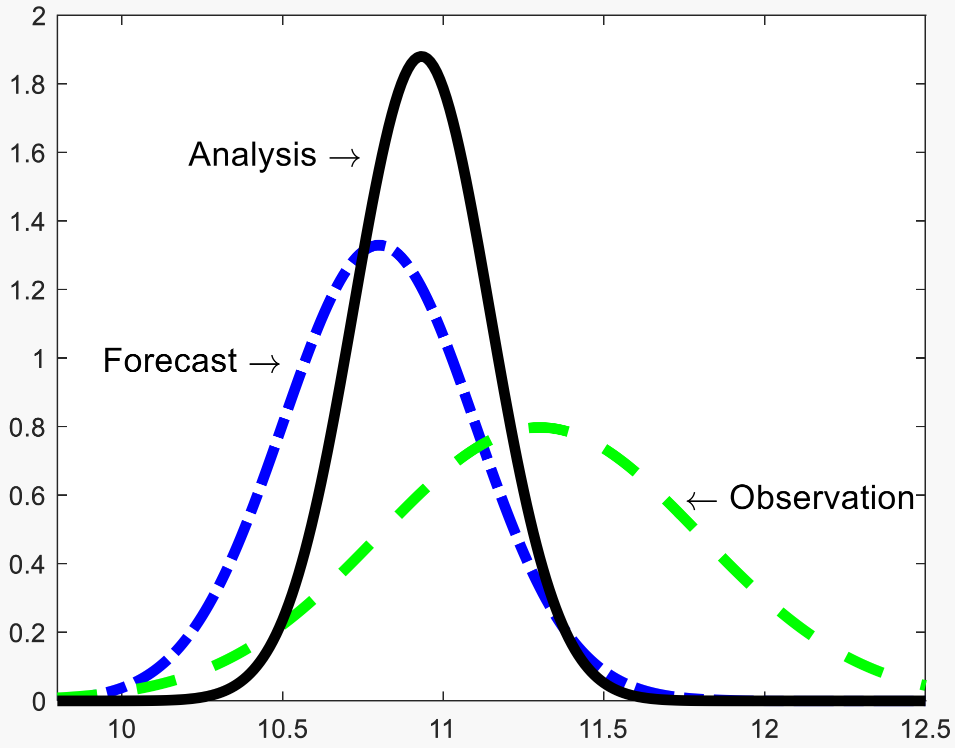
$$x_a = x_f + \frac{\sigma_f^2}{\sigma_f^2 + \sigma_o^2}(x_o - x_f)$$

$$\sigma_a^{-2} = \sigma_f^{-2} + \sigma_o^{-2}$$

Thus,

$$x_a = 10.8 + \frac{0.3^2}{0.3^3 + 0.5^2}(11.3 - 10.8) = 10.93$$

$$\sigma_a = \frac{1}{1/0.3^2 + 1/0.5^2} = 0.21$$



10.93 ± 0.21 m/s

Analysis

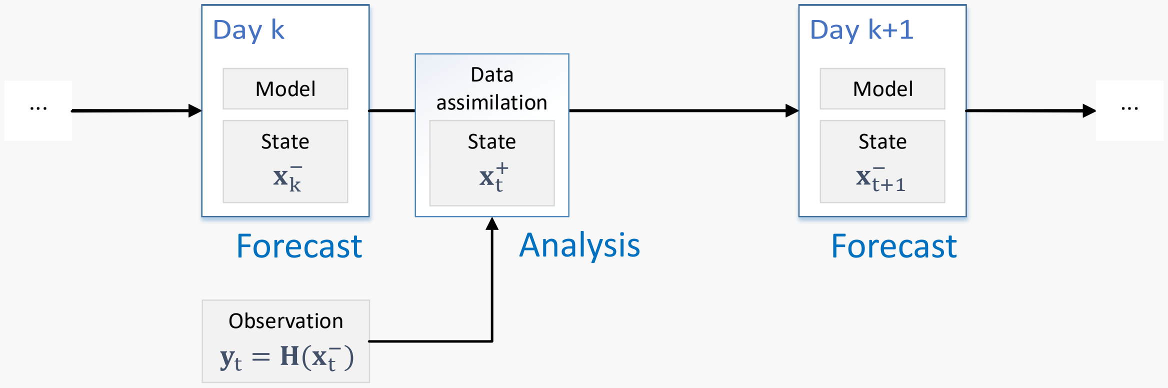
11.3 ± 0.5 m/s

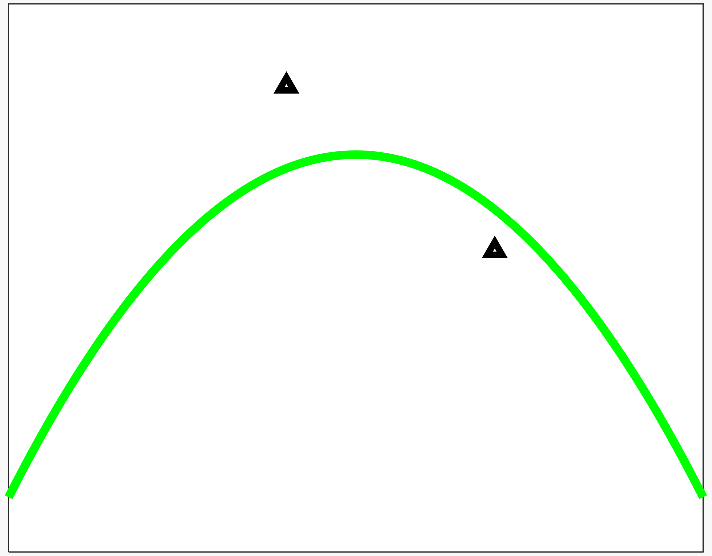
Observation

10.8 ± 0.3 m/s

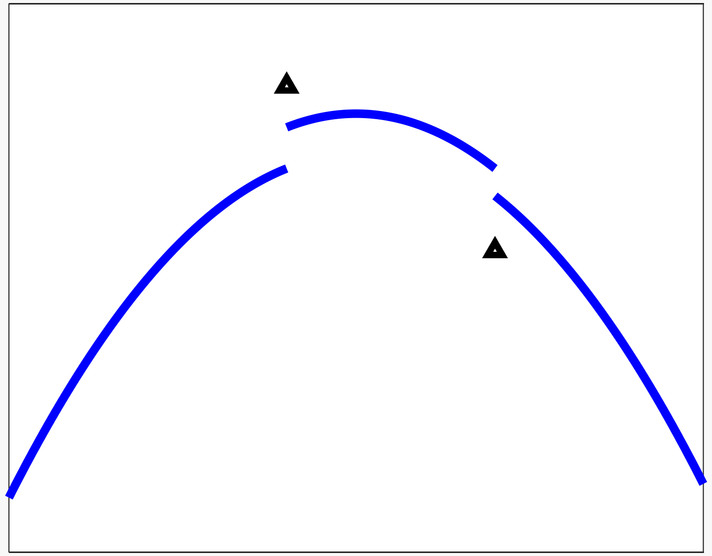
Forecast

Concepts





Open-loop



Closed-loop (data assimilation)

- State: $\mathbf{x}$
- Observation: $\mathbf{y}$
- State transfer function: $\mathbf{x}_t = M(\mathbf{x}_{t-1})$
- Error covariance: $\mathbf{P}$
- Prior/background/forecast: $\mathbf{x}^-$, $\mathbf{P}^-$
- Posterior/analysis: $\mathbf{x}^+$, $\mathbf{P}^+$

Kalman filter

Hypothesis

Linear system with uncertainty:

$$\begin{aligned}\mathbf{x}_t &= M(\mathbf{x}_{t-1}) + \mathbf{w}_t \\ &= \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{f}_t + \mathbf{C}_t + \mathbf{w}_t\end{aligned}$$

Observations:

$$\begin{aligned}\mathbf{y}_t &= H(\mathbf{x}_t) + \mathbf{v}_t \\ &= \mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t\end{aligned}$$

where $\mathbf{w}_t$ and $\mathbf{v}_t$ are independent uncertainties, following Gaussian distribution with a mean of zero and variance of $\mathbf{Q}$ and $\mathbf{R}$, respectively.

Mathematically,

$$\mathbf{w} \sim \mathrm{N}(0, \mathbf{Q})$$
$$\mathbf{v} \sim \mathrm{N}(0, \mathbf{R})$$

$\mathbf{Q}$ and $\mathbf{R}$ are the variance of model and observational errors, respectively.

Kalman filter

Forecast:

$$\mathbf{x}_t^- = M_t(\mathbf{x}_{t-1}^+)$$

The **prior** error:

$$\begin{aligned}
\mathbf{e}_t^- &= \mathbf{x}_t^- - \mathbf{x}_t \\
&= M_t(\mathbf{x}_{t-1}^+) - M_t(\mathbf{x}_{t-1}) - \mathbf{w}_t \\
&= \mathbf{A}_t(\mathbf{x}_{t-1}^+ - \mathbf{x}_{t-1}) - \mathbf{w}_t \\
&= \mathbf{A}_t\mathbf{e}_{t-1}^+ - \mathbf{w}_t
\end{aligned}$$

The **prior** error covariance

$$\begin{aligned}
\mathbf{P}_t^- &= \mathrm{cov}(\mathbf{e}_t^-, \mathbf{e}_t^-) \\
&= \mathrm{E}\left(\mathbf{e}_t^- \mathbf{e}_t^{-\mathrm{T}}\right) \\
&= \mathbf{A}_t\mathbf{P}_{t-1}^+\mathbf{A}_t^\mathrm{T} + \mathbf{Q}
\end{aligned}$$

This equation links the prior error covariance to the posterior error covariance of the previous step.

Analyse:

$$\begin{aligned}
\mathbf{y}_t &= \mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t \\
\mathbf{x}_t^+ &= \mathbf{x}_t^- + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t^-)
\end{aligned}$$

The **posterior** error:

$$\begin{aligned}
\mathbf{e}_t^+ &= \mathbf{x}_t^+ - \mathbf{x}_t \\
&= \mathbf{x}_t^- + \mathbf{K}_t(\mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t - \mathbf{H}_t\mathbf{x}_t^-) - \mathbf{x}_t \\
&= \mathbf{K}_t\mathbf{v}_t + (\mathbf{1} - \mathbf{K}_t\mathbf{H}_t)(\mathbf{x}_t^- - \mathbf{x}_t) \\
&= \mathbf{K}_t\mathbf{v}_t + (\mathbf{1} - \mathbf{K}_t\mathbf{H}_t)\mathbf{e}_t^-
\end{aligned}$$

The **posterior** error covariance

$$\begin{aligned}
\mathbf{P}_t^+ &= \mathrm{cov}(\mathbf{e}_t^+, \mathbf{e}_t^+) \\
&= \mathrm{E}\left(\mathbf{e}_t^+ \mathbf{e}_t^{+\mathrm{T}}\right) \\
&= (\mathbf{1} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_t^-(\mathbf{1} - \mathbf{K}_t\mathbf{H}_t)^\mathrm{T} + \mathbf{K}_t\mathbf{R}\mathbf{K}_t^\mathrm{T}
\end{aligned}$$

This equation links posterior error covariance to the prior error covariance of the same step.

Kalman filter

In KF, the Kalman Gain ($\mathbf{K}_t$) is solved by minimising the squared difference between posterior estimates and the 'true' state, that is, the posterior error covariance ($\mathbf{P}_t^+$).

By minimising $\mathbf{P}_t^+$, $\mathbf{K}_t$ is solved as

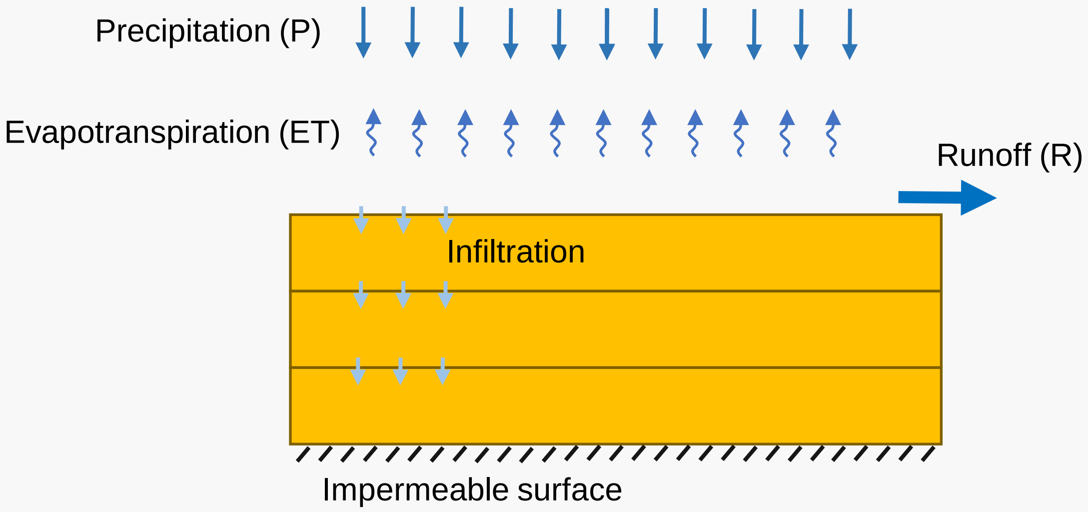
$$\mathbf{K}_t^* = \mathbf{P}_t^- \mathbf{H}_t^T \left(\mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R}_t\right)^{-1}$$

Once $\mathbf{K}_t$ is calculated, update the state and error covariance.

$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t \mathbf{x}_t^-)$$

$$\mathbf{P}_t^+ = (\mathbf{I} - \mathbf{K}_t^* \mathbf{H}_t)\, \mathbf{P}_t^-$$

A simple water balance model



Assume depth = 1 m, the water balance model can be written as

Layer 1: $\theta(1)_{t+1} = \theta(1)_t + P_t - ET_t - R_t - I(1)_t$

Layer 2: $\theta(2)_{t+1} = \theta(2)_t + I(1)_t - I(2)_t$

Layer 3: $\theta(3)_{t+1} = \theta(3)_t + I(2)_t$

Rewrite in matrix

$$\underbrace{\begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_{t+1}}_{\mathbf{x}_{t+1}} = \underbrace{\begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_{t}}_{\mathbf{x}_{t}} + \begin{bmatrix} P_t - ET_t - R_t \\ 0 \\ 0 \end{bmatrix}_t + \begin{bmatrix} I(1) \\ I(1) - I(2) \\ I(2) \end{bmatrix}_t$$

State transfer function

$$\begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_{t} + \begin{bmatrix} 1 & -1 & -1 \\ & & \\ & & \end{bmatrix} \begin{bmatrix} P_t \\ ET_t \\ R_t \end{bmatrix}_t + \begin{bmatrix} I(1) \\ I(1) - I(2) \\ I(2) \end{bmatrix}_t$$

$$\mathbf{x}_{t+1} = A_t \quad \mathbf{x}_t + B_t \quad \text{weather}_t + C_t$$

Kalman filter

Formula

1. Define the model that runs one step forward.

$$\begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_{t} + \begin{bmatrix} 1 & -1 & -1 \\ & & \\ & & \end{bmatrix} \begin{bmatrix} P_t \\ ET_t \\ R_t \end{bmatrix}_{t} + \begin{bmatrix} 1 & \\ 1 & -1 \\ & 1 \end{bmatrix} \begin{bmatrix} I(1) \\ I(2) \end{bmatrix}_{t}$$

$$\mathbf{x}_{t+1} = A_t \quad \mathbf{x}_t + B_t \quad \text{weather}_t + C_t \quad \text{parameter}_t$$

MATLAB code

HydroModel.m | run_KF.m | run_EnKF.m | run_OL_deterministic.m | run_OL_stochastic.m

```
function [state_out] = HydroModel(state_in, weather)

    % Define matrices in model
    A = [1 0 0; 0 1 0; 0 0 1];
    B = [1 -1 -1; 0 0 0; 0 0 0] ;
    C = [1 0; 1 -1; 0 1];
    parameter = [20 10]';
    state_out = A * state_in + B * weather * 0.001 + C * parameter  * 0.001;

    % Constrain range
    state_out(state_out<0.1) = 0.1;
    state_out(state_out>0.9) = 0.9;
```

Kalman filter

Formula

2. Define observational operator.

$$\begin{bmatrix} y(1) \\ y(2) \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}_t \begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_t$$

$$\mathbf{y}_t = \mathbf{H}_t \cdot \mathbf{x}_t$$

3. Set initial conditions

$$\begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_0 = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \end{bmatrix}$$

MATLAB code

HydroModel.m | run_KF.m | run_EnKF.m | run_OL_deterministic.m | run_OL_sto

```
%% Loading data
clear
close all
cd 'H:\My Drive\Slides\Introduction_DA\scripts'
weathers = load('data\ExampleData.csv');
observations = load('data\Obs.csv');

%% Model initial conditions
state= [0.3 0.3 0.3]';

%% KF matrices
% Define observational matrix H.
H = [1 0 0; 0, 1, 0];
% Set model and observational error variance by estimation.
Q = [0.1 0 0; 0 0.1 0; 0 0 0.1].^2;
R = [0.1 0; 0, 0.1].^2;
% Set  initial error covariance by guess.
% Note that P, Q, R are symmetric matrices.
P = [0.1 0 0; 0 0.1 0; 0 0 0.1].^2;
```

Kalman filter

Formula

4. Run the model one step forward.

If observation at the current timestep is available, run data assimilation.

$$\mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}_t^T \left(\mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R}_t\right)^{-1}$$

$$\mathbf{P}_t^+ = (\mathbf{I} - \mathbf{K}_t^* \mathbf{H}_t)\, \mathbf{P}_t^-$$

When $\mathbf{K}_t$ is calculated, the posterior state can be calculated.

$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t \mathbf{x}_t^-)$$

Then the prior state will be replaced by the posterior state values.

MATLAB code

HydroModel.m | run_KF.m | run_EnKF.m | run_OL_deterministic.m | run_OL_stochastic

```
%% Run the model in a loop.
days = length(weathers);
% Create empty matrices to save data.
state_prior = nan(length(state),days);
state_posterior = nan(size(state_prior));
P_all = nan([size(P) days]);
tmp = size(H);
K_all = nan(length(state), tmp(1), days);

for i = 1 : days
    % get daily data
    weather = weathers (i, :)';
    obs = observations (i, :)';
    state = HydroModel(state, weather);
    state_prior(:, i) = state;

    % Check if observation is available.
    % If yes, run data assimilation to update the state vector.
    if ~isnan(obs)
        K = P * H' * (H * P * H' + R ) ^ (-1);
        P = (1 - K * H) * P;
        state = state + K * (obs - H * state);
        % Save or export the values of theta, K, P if necessary.
        P_all(:,:,i) = P;
        K_all(:,:,i) = K;
        disp('State is updated.')
    else
        disp('Observation is not available!')
    end
    state_posterior(:, i) = state;
end
```

Non-liner models

The KF is based on the assumption of linear system.

In a general system, state transfer function can be written as:

$$\mathbf{x}_{\mathrm{t}} = M(\mathbf{x}_{\mathrm{t}-1}) + \mathbf{w}_{\mathrm{t}}$$

Observation to the system:

$$\mathbf{y}_{\mathrm{t}} = H(\mathbf{x}_{\mathrm{t}}) + \mathbf{v}_{\mathrm{t}}$$

where $\mathbf{x}$ and $\mathbf{y}$ are column vectors, and the subscriptions t and t+1 are the time step.

Solutions for non-linear model

Extended Kalman filter, EKF
Ensemble Kalman filter, EnKF
Particle filter, PF

Extended Kalman filter

Hypothesis

Model can be linearised by a Jacobians matrix.

Ensemble Kalman filter

Hypothesis

The probability distribution of uncertainty can be represented by a finite number of simulations (ensemble).

The ensemble

$$\mathbf{X} = [\mathbf{x}^1 \quad \mathbf{x}^2 \quad \dots \quad \mathbf{x}^N]$$

$\mathbf{X}$ is a M x N matrix, where M is the number of state in the state vector $\mathbf{x}$, N is the ensemble size.

Forecast

$$\mathbf{x}_t = M(\mathbf{x}_{t-1}) + \mathbf{w}_t$$

$$\mathbf{P}_t^- = \frac{1}{N-1}\mathbf{D}_t\mathbf{D}_t^T$$

where

$$\mathbf{D}_t^T = \left[\mathbf{x}^1 - E[\mathbf{x}^1] \quad \mathbf{x}^2 - E[\mathbf{x}^2] \quad \dots \quad \mathbf{x}^N - E\left[\mathbf{x}^N\right]\right]$$

Analyse:

$$\mathbf{y}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t$$

$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t^-)$$

Ensemble Kalman filter

Formula

1. Define the model that runs one step forward.

$$\mathbf{x}_{\mathrm{t}} = M(\mathbf{x}_{\mathrm{t-1}})$$

MATLAB code

HydroModel.m | run_KF.m | run_EnKF.m | run_OL_deterministic.m | run_OL_stochastic.m

```
function [state_out] = HydroModel(state_in, weather)

    % Define matrices in model
    A = [1 0 0; 0 1 0; 0 0 1];
    B = [1 -1 -1; 0 0 0; 0 0 0] ;
    C = [1 0; 1 -1; 0 1];
    parameter = [20 10]';
    state_out = A * state_in + B * weather * 0.001 + C * parameter  * 0.001;

    % Constrain range
    state_out(state_out<0.1) = 0.1;
    state_out(state_out>0.9) = 0.9;
```

Ensemble Kalman filter

Formula

2. Set initial conditions

$$\begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_0 = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \end{bmatrix}$$

3. Define observational operator.

$$\begin{bmatrix} y(1) \\ y(2) \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}_t \begin{bmatrix} \theta(1) \\ \theta(2) \\ \theta(3) \end{bmatrix}_t$$

$$\mathbf{y}_t = \mathbf{H}_t \cdot \mathbf{x}_t$$

4. Set uncertainty values for weather, initial conditions and observations.

MATLAB code

HydroModel.m | run_KF.m | run_EnKF.m | run_OL_deterministic.m | run_OL_stochastic.m

```
%% Loading data
clear
close all
cd 'H:\My Drive\Slides\Introduction_DA\scripts'
weathers = load('data\ExampleData.csv');
observations = load('data\Obs.csv');

%% Model initial conditions
state= [0.3 0.3 0.3]';

%% EnKF settings.
% Define observational matrix H.
H = [1 0 0; 0, 1, 0];
% Set model and observational error variance by estimation.
R = [0.03 0; 0, 0.03].^2;
% Set uncertainty for weather, initial conditions and observations.
error_weather = [5 3 1]';
error_init = [0.1 0.1 0.1]';
error_obs = [0.03 0.03]';
```

Kalman filter

Formula

5. Generate ensembles. Construct a M x N state matrix, where M is the length of the state vector, and N is the ensemble size. Perturb initial conditions by adding Gaussian noise.

6. Run the model **stochastically** in a loop.

- Perturb weather data by adding Gaussian noise. The matrix f a matrix with size of the length of the weather vector by the ensemble size.
- Run stochastic model. The estimated state is the mean of ensemble.

MATLAB code

```
%% Generate ensemble.
ensemble_size = 5;
% Perturb initial conditions.
X = repmat(state, 1, ensemble_size);
for i = 1:length(state)
    X(i, :) =  X(i, :) + randn(1, ensemble_size) * error_init(i);
end
```

HydroModel.m | run_KF.m | run_EnKF.m | run_OL_deterministic.m | run_OL_stochastic.m

```
%% Run the model in a loop.
for i = 1 : days
    % Get daily data
    weather = weathers (i, :)';
    obs = observations (i, :)';
    % Perturb weather forces. f is a matrix of no_of_weather by
    % ensemble_size.
    f = repmat(weather, 1, ensemble_size);
    for j = 1:length(weather)
        f(j, :) = f(j, :) + randn(1, ensemble_size) * error_weather(j);
    end
    % Run stochastic model. Note that X and f are matrices in EnKF rather
    % than vectors in KF.
    X = HydroModel(X, f);
    X_prior_ensemble(:, :, i) = X;
    X_prior(:, i) = mean(X, 2);
```

Kalman filter

Formula

7. If observation at the current timestep is available, run data assimilation.

- Perturb observations data by adding Gaussian noise. The matrix y a matrix with size of the length of the observation vector by the ensemble size.

- Calculate

$$\mathbf{D}_t^T = \left[\mathbf{x}^1 - \mathrm{E}[\mathbf{x}^1] \quad \mathbf{x}^2 - \mathrm{E}[\mathbf{x}^2] \quad \dots \quad \mathbf{x}^N - \mathrm{E}[\mathbf{x}^N]\right]$$

$$\mathbf{P}_t^- = \frac{1}{N-1}\mathbf{D}_t\mathbf{D}_t^T$$

$$\mathbf{K}_t = \mathbf{P}_t^-\mathbf{H}_t^T\left(\mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t\right)^{-1}$$

Once $\mathbf{K}_t$ is calculated, update the state and error covariance.

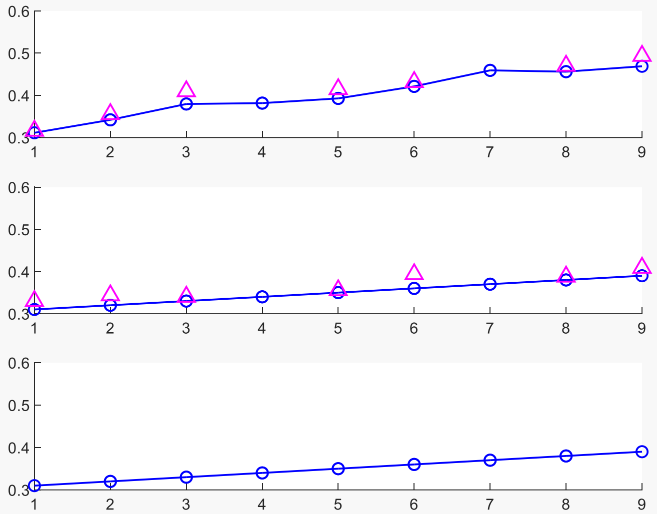
$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t^-)$$

MATLAB code

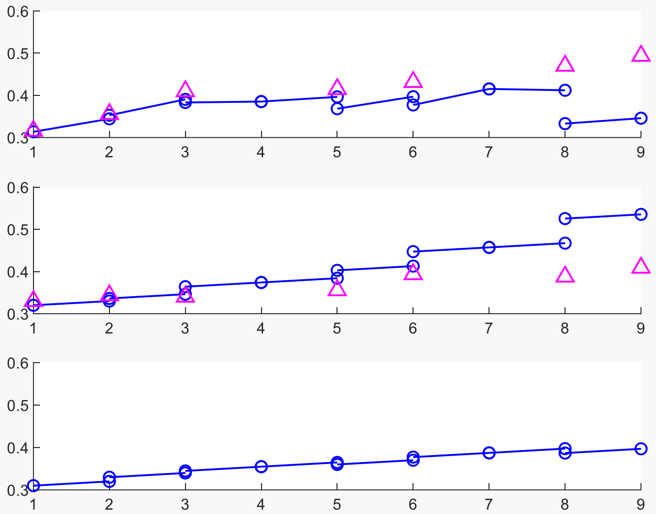
HydroModel.m | run_KF.m | run_EnKF.m | run_OL_deterministic.m | run_OL_stochastic.m

```
    % Check if observation is available.
    % If yes, run data assimilation to update the state vector.
    if ~isnan(obs)
        % Perturb observation (y).
        y = repmat(obs, 1, ensemble_size);
        for j = 1:length(obs)
            y(j, :) = y(j, :) + randn(1, ensemble_size) * error_obs(j);
        end
        % Run EnKF
        X_mean = mean(X, 2);
        D = X - repmat(X_mean, 1, ensemble_size);
        P = D * D' / (1 + ensemble_size);
        K = P * H' * (H * P * H' + R) ^ (-1);
        X = X + K * (y - H * X);
        % Save or export the values of theta, K, P if necessary.
        P_all(:,:,i) = P;
        K_all(:,:,i) = K;
        obs_ensemble(:, :, i) = y;
        disp('State is updated.')
    else
        disp('Observation is not available!')
    end
    X_posterior_ensemble(:, :, i) = X;
    X_posterior(:, i) = mean(X, 2);
end
```

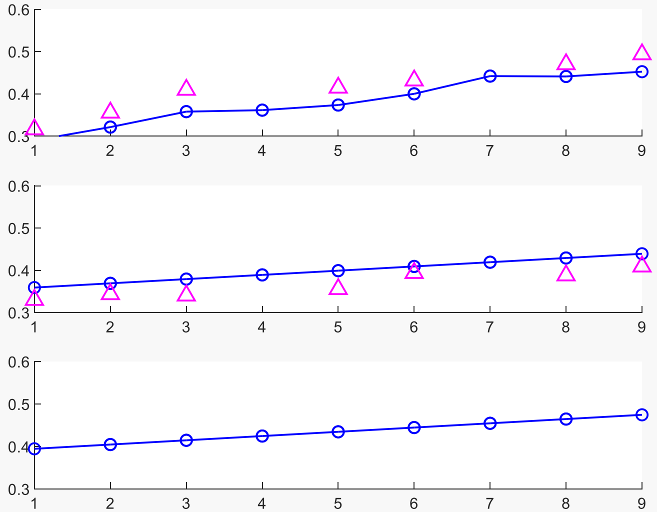
Demo output



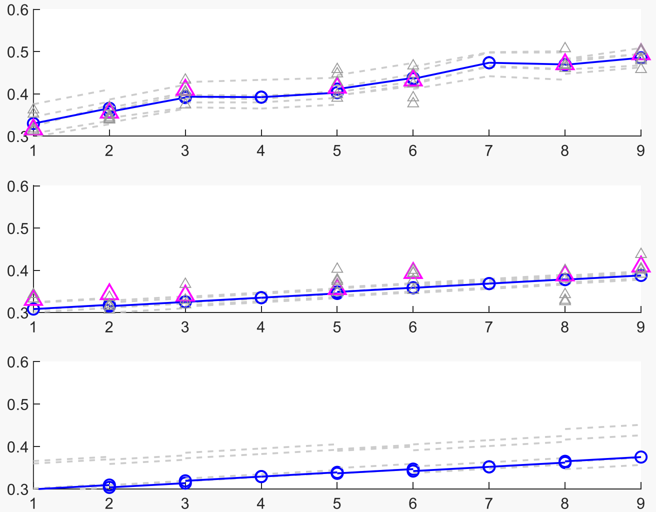
Open-loop, deterministic run



KF



Open-loop, stochastic run



EnKF