TTK4250

Week 4

From tuning to nonlinear filtering

Edmund Førland Brekke

12. September 2024

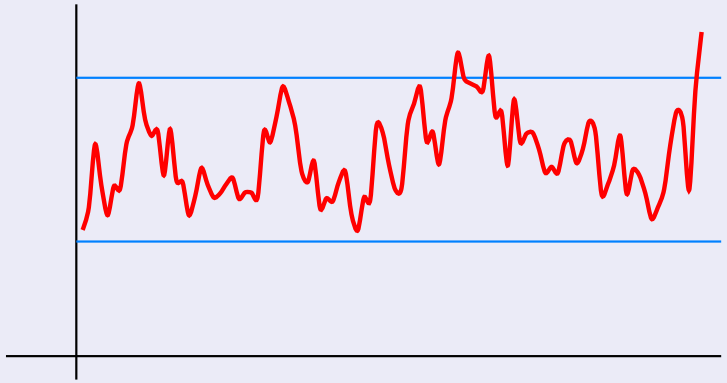
Recap from last week

Stochastic processes

- The Wiener process
- (Continuous time) White noise
- The Gauss-Markov process

We describe stochastic processes in terms of their ACF.

Tuning of the Kalman filter



What is white noise?

- The time-derivative of the Wiener process.
- A stationary stochastic process with ACF $R(\tau) = \delta(\tau)$.

Discretization

- Use techniques from TTK4105 or TTK4130 to relate $\mathbf{x}_k$ to $\mathbf{x}_{k-1}$.
- $\mathbf{Q} = \int_0^T e^{(T-\tau)\mathbf{A}} \mathbf{G}\mathbf{D}\mathbf{G}^\mathsf{T} e^{(T-\tau)\mathbf{A}^\mathsf{T}} \mathrm{d}\tau$.
- Consider Van Loan's formula or series expansions to evaluate $\mathbf{Q}$ efficiently.

The concept of filter consistency

A filter is said to be consistent if

1. The state errors are acceptable as zero mean.
2. The state errors have magnitude commensurate with the state covariance yielded by the filter.
3. The innovations are acceptable as zero mean.
4. The innovations have magnitude commensurate with the innovation covariance yielded by the filter.
5. The innovations are acceptable as white.

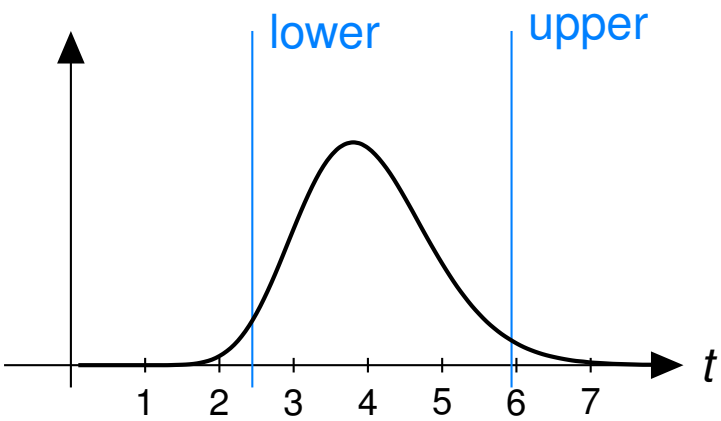
Criteria 2 and 4 are most important, and are tested by means of the normalized estimation error squared (NEES) and the normalized innovations squared (NIS):

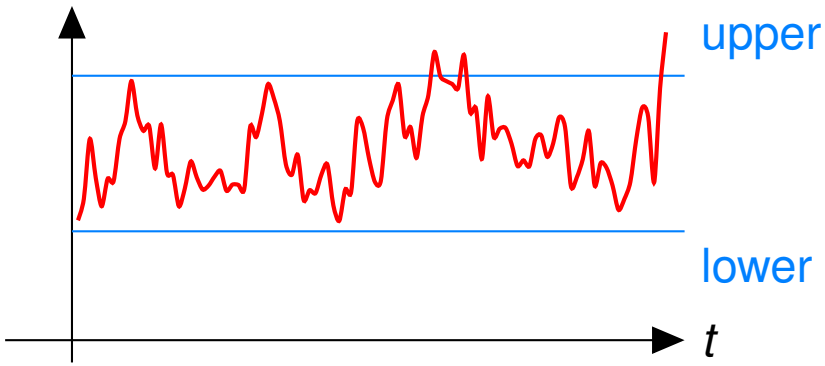
$$\begin{aligned}\epsilon_k &= (\hat{\mathbf{x}}_k - \mathbf{x}_k)^\mathsf{T} \mathbf{P}_k^{-1} (\hat{\mathbf{x}}_k - \mathbf{x}_k) \\ \epsilon_k^\nu &= \boldsymbol{\nu}_k^\mathsf{T} \mathbf{S}_k^{-1} \boldsymbol{\nu}_k = (\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1})^\mathsf{T} \mathbf{S}_k^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}).\end{aligned}$$

χ^2-test for filter consistency (NEES)

- Suppose that $\mathbf{x} \in \mathbb{R}^d$.
- We perform N Monte-Carlo simulations of our Kalman filter.
- Then a 95% confidence interval for the average (ANEES) value of ϵ_k is given by

$$\texttt{lower} = \texttt{chi2inv}(0.025, Nd)/N \quad \text{and} \quad \texttt{upper} = \texttt{chi2inv}(0.975, Nd)/N$$





- If ANEES lies below the χ^2 limits we can expect the filter to be overly conservative.
- If ANEES lies above the χ^2 limits we can expect the filter to be overconfident and put too little emphasis on the measurements.

Use NIS in a similar manner if working on real data without ground truth.

Example of consistency-based tuning

Finding a suitable value for σ_a in the CV model originally used in the radar-based Autosea tracker.[1]

Table 2 Process noise evaluation via AIS filter consistency. The (r_1, r_2) interval is the two-sided 95% probability concentration region for the χ^2 distribution related to the corresponding NIS. This varies with according to the AIS data record length N. The NIS values that are closest to being covariance-consistent, i.e. closest to the 95% probability region, are emphasised in bold

Name	$\sigma_a = 0.05$		$\sigma_a = 0.5$		(r_1, r_2)	N
	NIS	AI	NIS	AI		
GLUTRA	**4.67**	−0.02	0.90	0.01	(3.47, 4.55)	109
SULA	**3.61**	−0.22	0.51	−0.10	(3.49, 4.56)	106
KORSFJORD	71.8	−1.33	**4.31**	−0.44	(3.52, 4.51)	127
TR.FJORD II	11.3	−0.62	**3.24**	−0.16	(3.76, 4.24)	533
TELEMETRON	371	−0.04	**4.45**	−0.01	(3.77, 4.23)	579

[1] Wilthil et al. (2017): "A target tracking system for ASV collision avoidance based on the PDAF", Springer.

Testing the whiteness of the innovations

Whiteness test in Monte-Carlo simulations

Let ν_k be one of the innovation states in the vector $\boldsymbol{\nu}_k$. Let N be the number of Monte-Carlo simulations. Then the distribution of the **sample autocorrelation**

$$\rho_{kj} = \frac{\sum_{i=1}^{N} \nu_k^{(i)} \nu_j^{(i)}}{\sqrt{\sum_{i=1}^{N} (\nu_k^{(i)})^2 \sum_{i=1}^{N} (\nu_j^{(i)})^2}}$$

should tend to $\mathcal{N}(0, 1/N)$ for all $k \neq j$ when N is large.

Single-run whiteness test

The variance of the **time-average autocorrelation** should tend towards $1/K$:

$$\bar{\rho}_j = \frac{\sum_{k=1}^{K} \nu_k \nu_{k+j}}{\sqrt{\sum_{k=1}^{K} \nu_k^2 \sum_{k=1}^{K} \nu_{k+j}^2}}$$

Tuning the measurement noise covariance

For exteroceptive sensors, the appropriate values in $\mathbf{R}$ depend on

- The sensor resolution.
- The extent of targets or landmarks.

Example: Point targets with pixellated sensor

- 2-dimensional sensor with square cells of fixed resolution Δx.
- $\mathbf{x} = [x, y, v_x, v_y]^\mathsf{T}$.
- Measurement matrix $\mathbf{H} = [\mathbf{I}_2, \mathbf{0}]$

$$p(\mathbf{z} - \mathbf{H}\mathbf{x}|\mathbf{x}) = \begin{cases} 1/\Delta x^2 & \text{if } \|\mathbf{z} - \mathbf{H}\mathbf{x}\|_\infty < \Delta x/2 \\ 0 & \text{otherwise.} \end{cases}$$

This distribution can be approximated by a Gaussian

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}\,;\, \mathbf{H}\mathbf{x}, \mathbf{R})$$

with the same covariance:

$$\mathbf{R} = \begin{bmatrix} \frac{\Delta x^2}{12} & 0 \\ 0 & \frac{\Delta x^2}{12} \end{bmatrix}.$$

More about the measurement model

Mild nonlinearities in the measurement model can sometimes be removed by converting the measurements. We must then also convert $\mathbf{R}$ accordingly.

Nonlinear model.

The model is of the form $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k$, $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ where

$$\mathbf{h}(\mathbf{x}_k) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \text{atan2}(y, x) \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}$$

Converted measurements.

The model is of the form $\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k$, $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_c)$ where

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R}_c = \mathbf{J}\mathbf{R}\mathbf{J}^\mathsf{T}, \quad \mathbf{J} = \frac{\partial}{\partial \mathbf{z}_k} \mathbf{h}^{-1}(\mathbf{z}_k)$$

For more sophisticated conversion techniques see Lerro & Bar-Shalom (1993): “Tracking with debiased consistent converted measurements versus EKF”, IEEE-TAES.

Measurement conversion is generally preferable to EKF techniques because the linearization in an EKF can lead to filter instability.

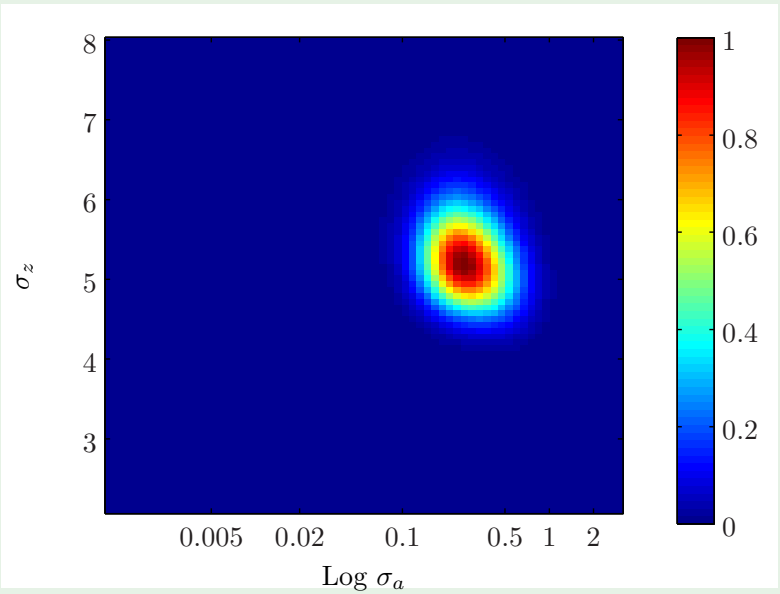
Maximum likelihood estimation of system parameters

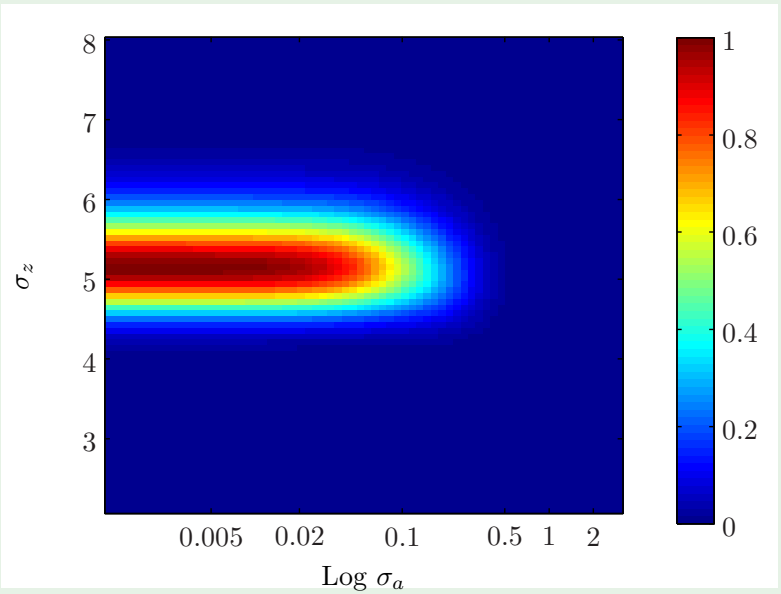
Theorem for maximum likelihood estimation

Assume that both **Q** and **R** depend on an unknown parameter vector $\boldsymbol{q}$. The maximum likelihood estimate of $\boldsymbol{q}$, if it exists, can then be found as

$$\boldsymbol{q}_{\mathrm{ML}} = \arg\max_{\boldsymbol{q}} \sum_{k} \log \mathcal{N}(\mathbf{z}_k \,;\, \mathbf{H}\hat{\mathbf{x}}_{k|k-1}, \mathbf{S}_k)$$

Example: Estimating σ_a and σ_z for CV model





The likelihood function, normalized relative to its maximum, for two realizations of the CV model.

Nonlinear systems and linearization

We are given a system of the form

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{v}_k, \quad , \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{Q}), \quad \mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{R}).$$

What are our options?

Linearization by means of Jacobians

- **Linearize around most recent predictions / estimates.**
- Linearize around a fixed operating point / external reference (e.g., observer).
- Repeated linearization to improve the linearization (Gauss-Newton iteration).

"Statistical linearization"

- The Unscented Kalman filter.
- Basically variations of LMMSE estimation where $\mathbf{P}_{xx}$, $\mathbf{P}_{xz}$ and $\mathbf{P}_{zz}$ are found propagating a strategic set of samples through $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$.

Sample-based methods

- Point-mass filter.
- **Particle filter.**

Linearization in the Extended Kalman Filter (EKF)

We introduce the error variables

$$\Delta \mathbf{x}_{k-1} = \mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}$$
$$\Delta \mathbf{x}_{k} = \mathbf{x}_{k} - \hat{\mathbf{x}}_{k|k-1}$$

and then make Taylor expansions in terms of the error variables:

$$\mathbf{f}(\mathbf{x}_{k-1}) \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1}) + \mathbf{F}(\hat{\mathbf{x}}_{k-1})\,\Delta \mathbf{x}_{k-1}$$
$$\mathbf{h}(\mathbf{x}_{k}) \approx \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{H}(\hat{\mathbf{x}}_{k|k-1})\,\Delta \mathbf{x}_{k}.$$

The matrices **F** and **H** are found as Jacobians of **f** and **h** with respect to $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$:

$$\mathbf{F}(\hat{\mathbf{x}}_{k-1}) = \left.\frac{\partial}{\partial \mathbf{x}_{k-1}} \mathbf{f}(\mathbf{x}_{k-1})\right|_{\mathbf{x}_{k-1}=\hat{\mathbf{x}}_{k-1}}$$
$$\mathbf{H}(\hat{\mathbf{x}}_{k|k-1}) = \left.\frac{\partial}{\partial \mathbf{x}_{k}} \mathbf{h}(\mathbf{x}_{k})\right|_{\mathbf{x}_{k}=\hat{\mathbf{x}}_{k|k-1}}$$

We use these matrices in the matrix products of the Kalman gain and posterior covariance in the same manner as in the regular KF.

Derivation of the EKF

(Only a sketch, but the full derivation is not much more complicated.)

Prediction in the EKF

The Taylor expansion $\mathbf{f}(\mathbf{x}_{k-1}) \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1}) + \mathbf{F}\Delta\mathbf{x}_{k-1}$ yields

$$
\begin{aligned}
p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}) &\approx \int \mathcal{N}(\mathbf{x}_k - \mathbf{f}(\hat{\mathbf{x}}_{k-1}) \,;\, \mathbf{F}\Delta\mathbf{x}_{k-1}, \mathbf{Q})\mathcal{N}(\Delta\mathbf{x}_{k-1} \,;\, \mathbf{0}, \mathbf{P}_{k-1})\mathrm{d}\Delta\mathbf{x}_{k-1} \\
&= \mathcal{N}(\mathbf{x}_k \,;\, \mathbf{f}(\hat{\mathbf{x}}_{k-1}), \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^{\mathsf{T}} + \mathbf{Q}).
\end{aligned}
$$

Measurement update in the EKF

The Taylor expansion $\mathbf{h}(\mathbf{x}_k) \approx \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{H}\mathbf{x}_k$ yields

$$
\begin{aligned}
p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) &\approx \mathcal{N}(\mathbf{z}_k - \mathbf{h}(\bar{\mathbf{x}}_k) \,;\, \mathbf{H}\Delta\mathbf{x}_k, \mathbf{R})\mathcal{N}(\Delta\mathbf{x}_k \,;\, \mathbf{0}, \mathbf{P}_{k|k-1}) \\
&\propto \mathcal{N}(\mathbf{x}_k \,;\, \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})), (\mathbf{I} - \mathbf{W}\mathbf{H})\mathbf{P}_{k|k-1}).
\end{aligned}
$$

The EKF algorithm

Algorithm 2 The extended Kalman filter

procedure EKF($\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1}, \mathbf{z}_k$)

$\hat{\mathbf{x}}_{k|k-1} \leftarrow \boldsymbol{f}(\hat{\mathbf{x}}_{k-1})$ ▷ The predicted state estimate

$\mathbf{F} \leftarrow \frac{\partial}{\partial \mathbf{x}_{k-1}} \boldsymbol{f}(\mathbf{x}_{k-1})$ ▷ The prediction Jacobian

$\mathbf{P}_{k|k-1} \leftarrow \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^{\mathsf{T}} + \mathbf{Q}$ ▷ The predicted covariance

$\hat{\mathbf{z}}_{k|k-1} \leftarrow \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})$ ▷ The predicted measurement

$\boldsymbol{\nu}_k \leftarrow \mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}$ ▷ The innovation

$\mathbf{H} \leftarrow \frac{\partial}{\partial \mathbf{x}_k} \mathbf{h}(\mathbf{x}_k)$ ▷ The measurement Jacobian

$\mathbf{S}_k \leftarrow \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^{\mathsf{T}} + \mathbf{R}$ ▷ The innovation covariance

$\mathbf{W}_k \leftarrow \mathbf{P}_{k|k-1}\mathbf{H}^{\mathsf{T}}\mathbf{S}_k^{-1}$ ▷ The Kalman gain

$\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}_k\boldsymbol{\nu}_k$ ▷ The posterior state estimate

$\mathbf{P}_k \leftarrow (\mathbf{I} - \mathbf{W}_k\mathbf{H})\mathbf{P}_{k|k-1}$ ▷ The posterior covariance

return $\hat{\mathbf{x}}_k, \mathbf{P}_k, \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}, \mathbf{S}_k$

end procedure

Example: The Coordinate Turn (CT) model

- We want to track a target that is turning with a nearly constant turn rate. We have the following relationships

$$\ddot{x} = -\omega \dot{y} \quad \text{and} \quad \ddot{y} = \omega \dot{x}.$$

- With the state vector $\mathbf{x} = [x, y, \dot{x}, \dot{y}, \omega]^{\mathsf{T}}$ this leads to a state-space model of the form

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\omega & 0 \\ 0 & 0 & \omega & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{n} \tag{1}$$

  where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{D})$ and $\mathbf{D} = \mathrm{diag}([\sigma_a^2, \sigma_a^2, \sigma_\omega^2])$.
- The model can be discretized by assuming that the turnrate ω is slowly varying. See (5.13)-(5.17) in the book.

Simulation setup: CT versus CV

- We use a standard model for $2D$ position measurements

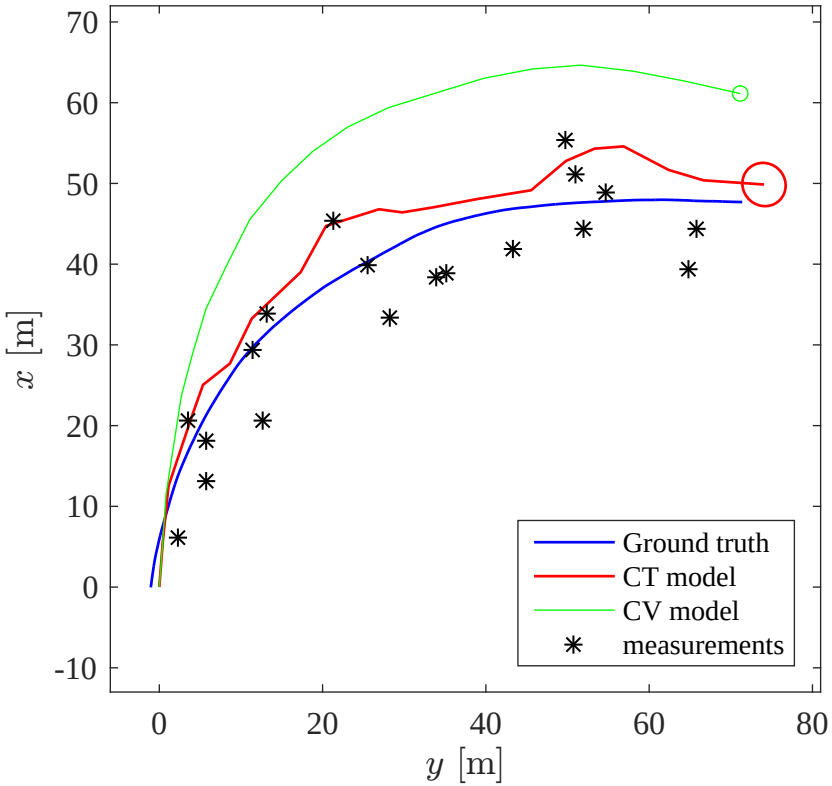
$$\mathbf{z}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}_k + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \sigma_z^2 \mathbf{I}_2).$$

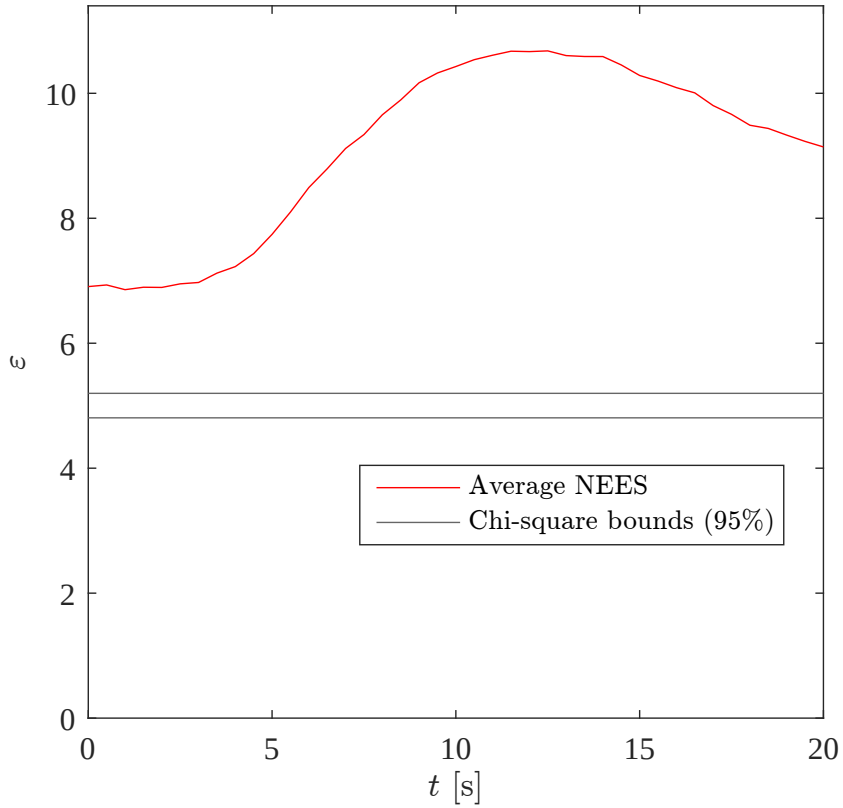
- We draw the initial state vector in the simulation according to $\mathcal{N}(\hat{\mathbf{x}}_1, \mathbf{P}_1)$ where $\hat{\mathbf{x}}_1 = [0, 0, 5, 0, 0.05]^\mathsf{T}$ and $\mathbf{P}_1 = \mathrm{diag}([25, 25, 0.25, 0.25, 0.0025])$.
- The following model parameters are used:

Symbol	Description	Value
σ_a	Acceleration plant noise standard deviation	$0.02\ \mathrm{m/s^2}$
σ_ω	Turn rate plant noise standard deviation	$0.005\ \mathrm{/s}$
σ_z	Measurement noise standard deviation	$5\ \mathrm{m}$
T	Discretization time interval	$0.5\ \mathrm{s}$

- The CV model is implemented with the same acceleration process noise σ_a as the CT model.

Simulation results: CT versus CV





Filtering results for a single simulation of the CT scenario and consistency analysis from 1000 Monte-Carlo simulations.

Another example: Bearings-only tracking

A challenge that has defeated solution attempts since the late 70s.

Problem statement

- Own-ship state: $\mathbf{x}^o = [x^o, y^o, \dot{x}^o, \dot{y}^o]^\mathsf{T}$. Both state vector and orientation known.
- Target state: $\mathbf{x}^\tau = [x^\tau, y^\tau, \dot{x}^\tau, \dot{y}^\tau]^\mathsf{T}$.
- Kinematics model (to be discretized using standard linear techniques):

$$\dot{\mathbf{x}}^\tau = \begin{bmatrix} \mathbf{0}_{2\times 2} & \mathbf{I}_2 \\ \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} \end{bmatrix} \mathbf{x}^\tau + \mathbf{v}(t)$$

- Measurement model:

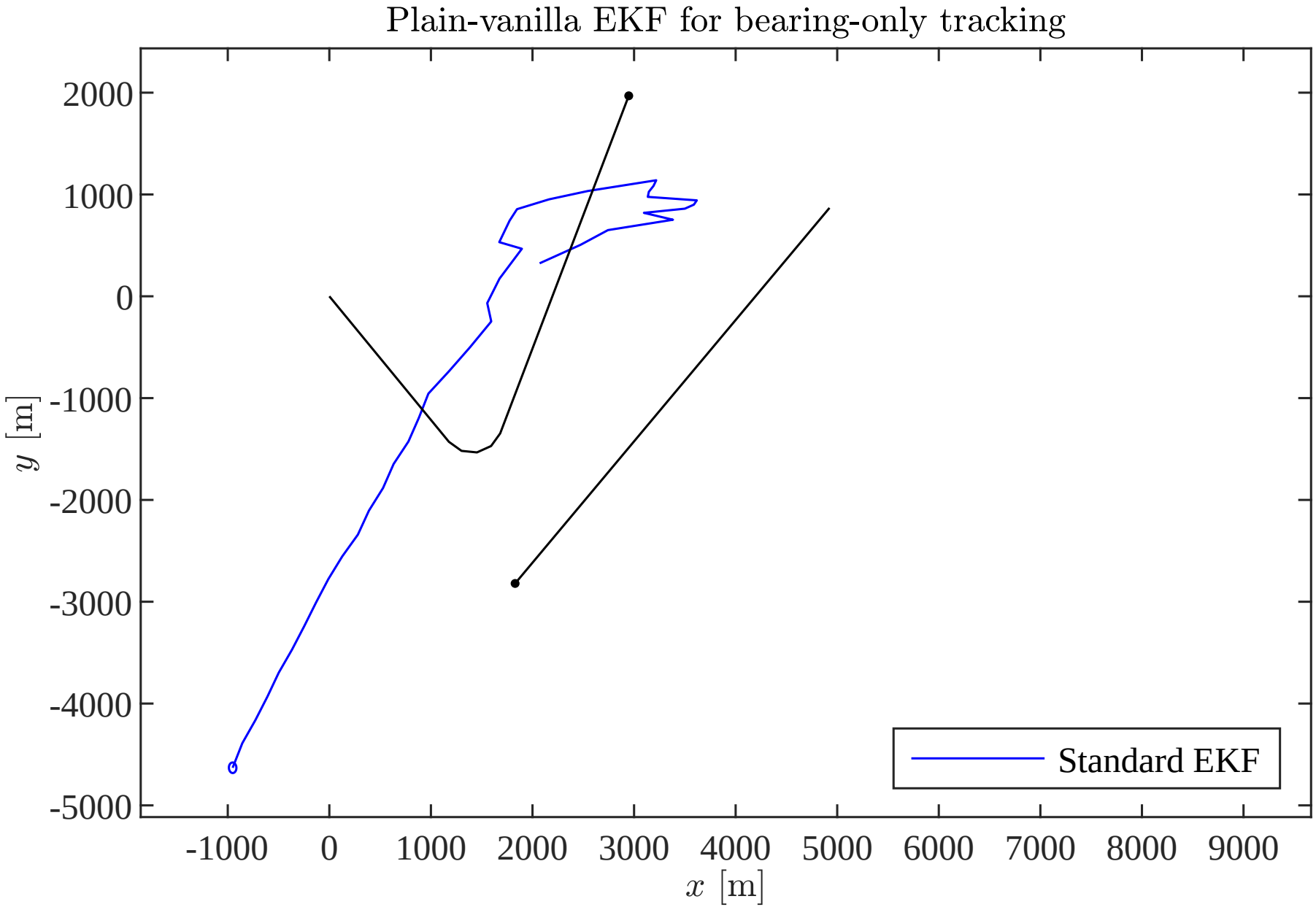
$$\mathbf{z}_k = \operatorname{atan2}(y_k^\tau - y_k^o, x_k^\tau - x_k^o) + \mathbf{w}_k$$

- Noise processes assumed zero-mean white mutually independent Gaussian.

Challenges

- The measurement model is nonlinear.
- The problem is not observable unless own-ship executes manoeuvres.

Let's attempt a standard EKF



EKF with modified polar coordinates (MPC)

The relative range and bearing are given by

$$r = \sqrt{(x^\tau - x^o)^2 + (y^\tau - y^o)^2} \qquad \beta = \text{atan2}(y^\tau - y^o, x^\tau - x^o)$$

We define an alternative state vector $\mathbf{y}$ in modified polar coordinates (MPC):

$$\mathbf{y} = \begin{bmatrix} \dot{\beta} \\ \dot{r}/r \\ \beta \\ 1/r \end{bmatrix}$$

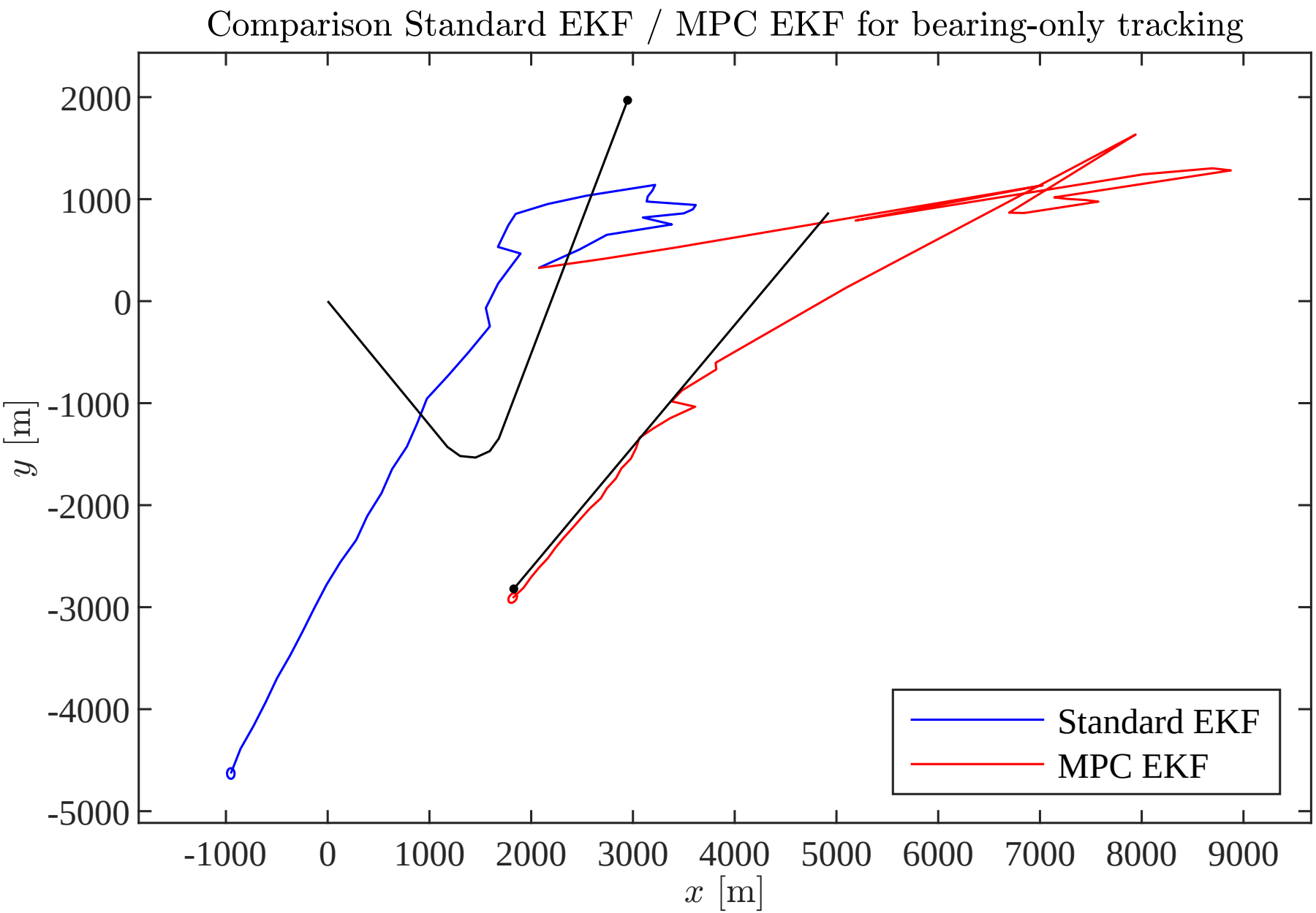
The measurement model then becomes linear:

$$\mathbf{z}_= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{y}_k + \mathbf{w}_k$$

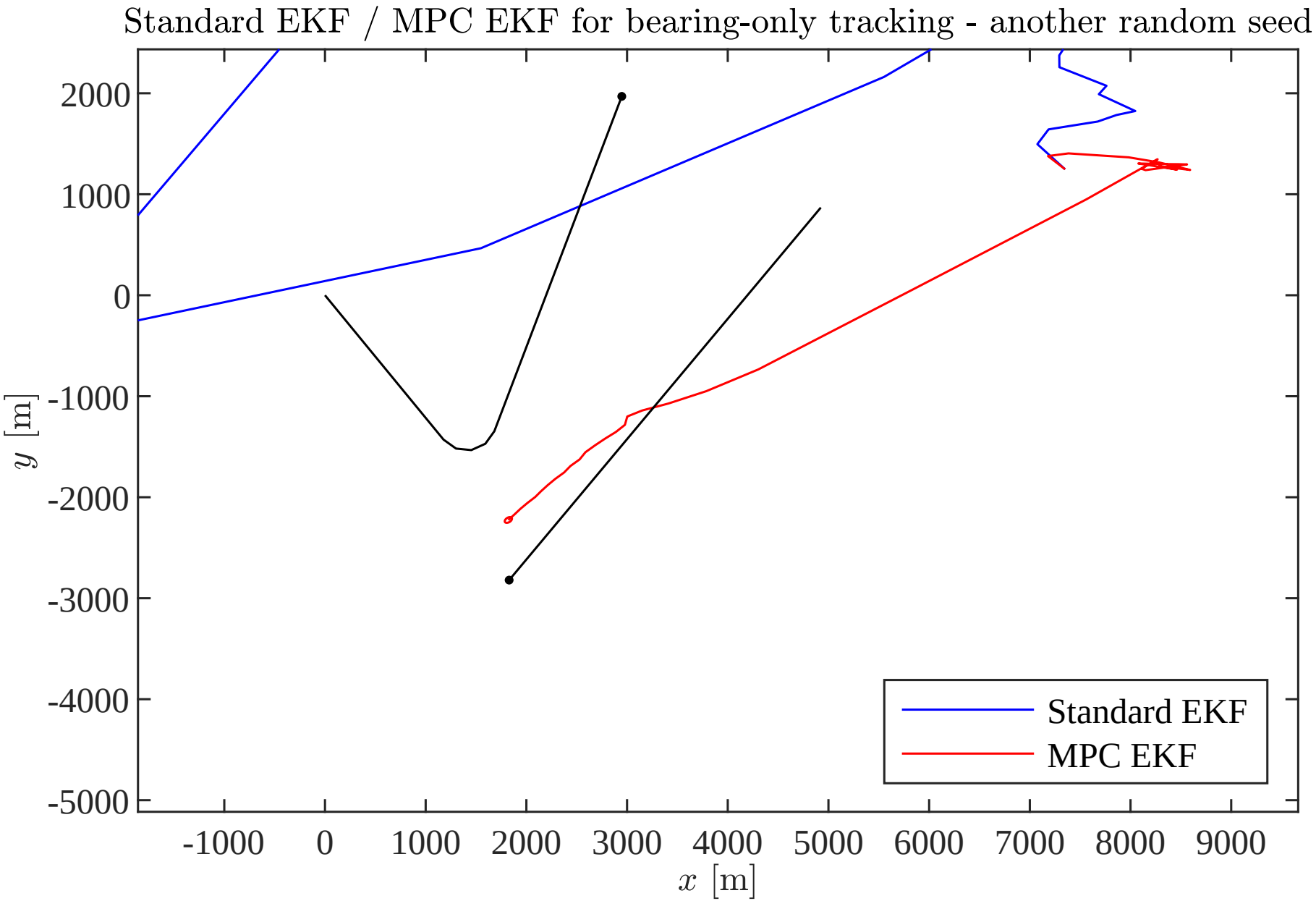
The process model becomes nonlinear and somewhat complicated. Nevertheless, its deterministic parts can be solved exactly[2] so that we get a discrete-time process model of the form $\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{v}_k$.

[2] Aidala & Hammel (1983) "Utilization of Modified Polar Coordinates for Bearings-Only Tracking", IEEE-TAC

This seems to work much better



But with another random seed . . .



What have we learned?

- The standard EKF begins to diverge immediately.
- The MPC-EKF has a very good bearing estimate.
- The range estimate of MPC-EKF is sort of OK, but far from covariance-consistent.

What is missing?

An ability of understanding the actual shape of $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ and how this shape is being affected by Chapman-Kolmogorov and new measurements.

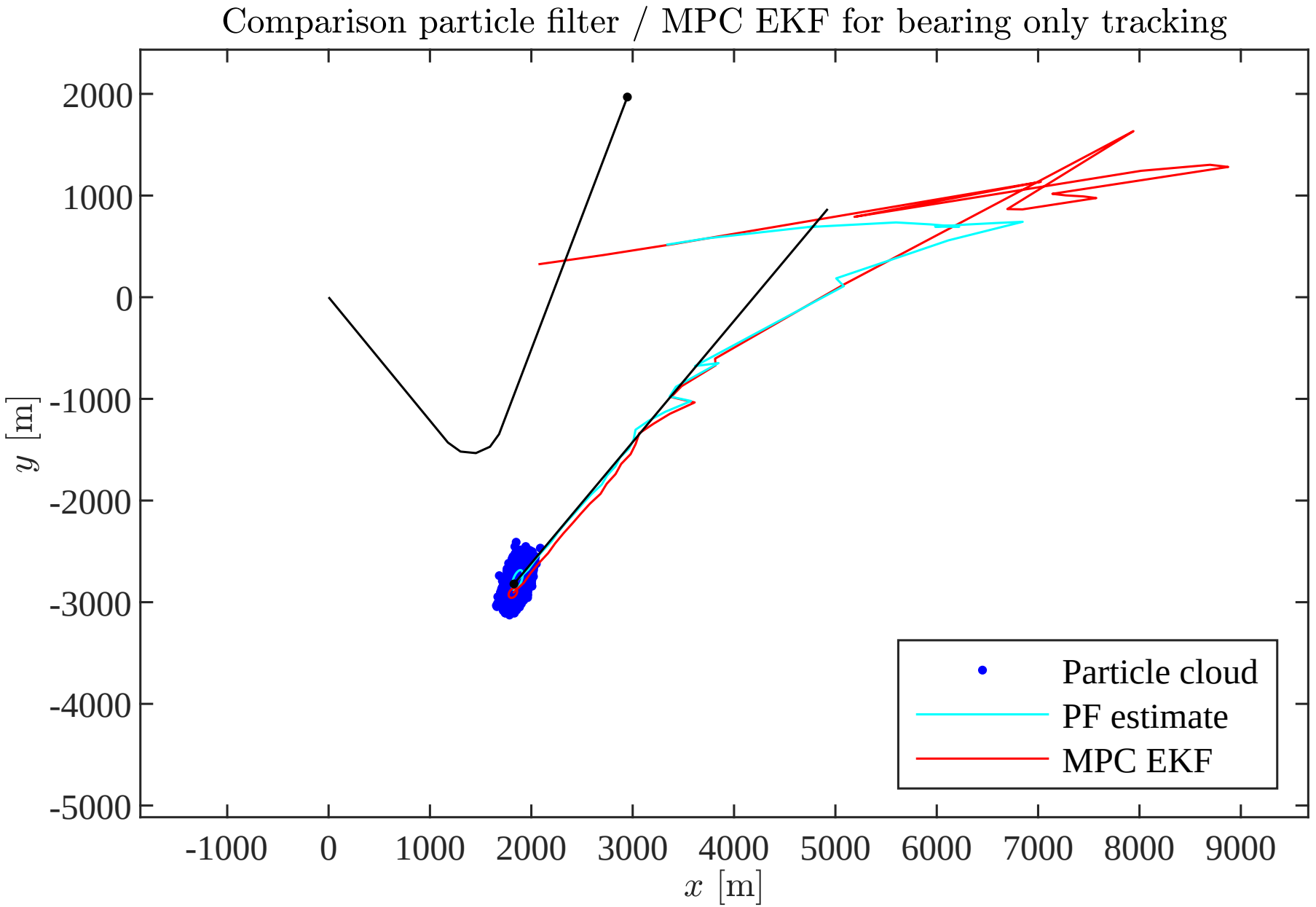
- The standard EKF makes gross misrepresentations of the shape.
- The MPC-EKF has a better understanding of the shape because it operates in a coordinate system where the posterior looks more like a Gaussian.

How can this be achieved?

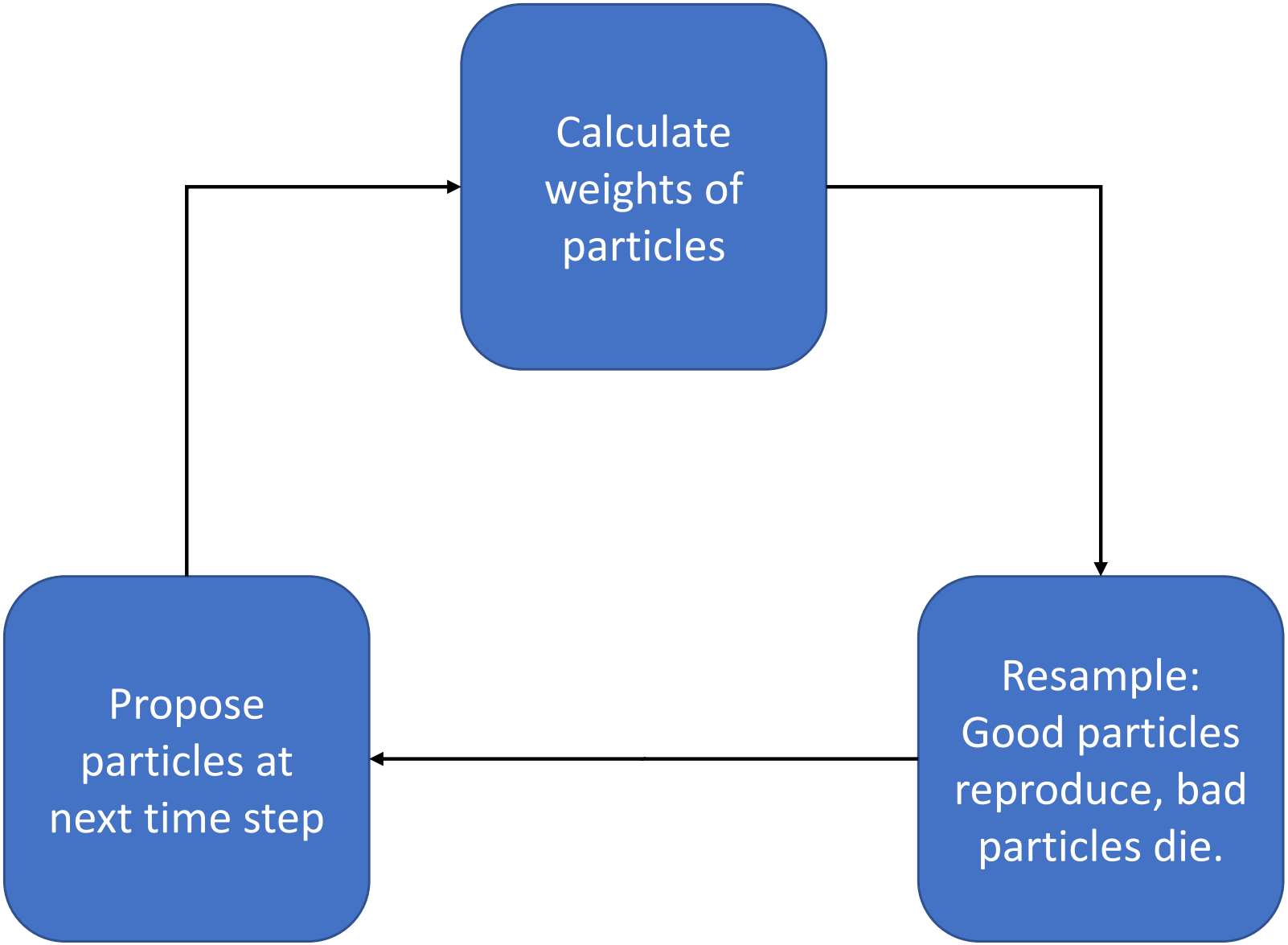
Evaluate $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ over samples of the state vector.

- A pre-defined grid would have to contain very many samples.
- Instead, we draw samples in strategic areas of the sample space by utilizing our knowledge about $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. This leads to the concept of a **Particle filter.**

Example of a particle filter for bearing-only



Bare bone structure of a particle filter



Key ideas behind particle filters

A particle filter provides an approximate solution of the Bayes recursion by means of random sampling.

- \- Computationally intensive, depending on how many particles are needed.
- \- Convergence proofs only valid for ∞ many of particles.
- \+ Very flexible: Can handle arbitrary nonlinearities.
- \+ Easy to implement.
- \+ Avoids the inversion of large covariance matrices.

Monte-Carlo integration.

Approximate evaluation of arbitrary integrals by random sampling.

Sampling of trajectories.

- Each particle represents a hypothesis for the entire trajectory $\mathbf{x}_{1:k}$.
- In other words, the particle filter doesn't really sample $\mathbf{x}_k$ but $\mathbf{x}_{1:k}$.

Resampling.

To combat degeneracy caused by the sampling-of-trajectories property.

First idea: Monte-Carlo integration and importance sampling

We want a more efficient alternative to numerical (fixed-grid, a.k.a. quadrature) integration.

Monte-Carlo integration.

- We want to evaluate an integral on the form $I = \int \mathbf{g}(\mathbf{x})\mathrm{d}\mathbf{x} = \int \mathbf{f}(\mathbf{x})\pi(\mathbf{x})\mathrm{d}\mathbf{x}$ where $\pi(\mathbf{x})$ is a pdf.
- Draw N samples $\mathbf{x}^i$ from $\pi(\mathbf{x})$ and let $I_N = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}^i)$.

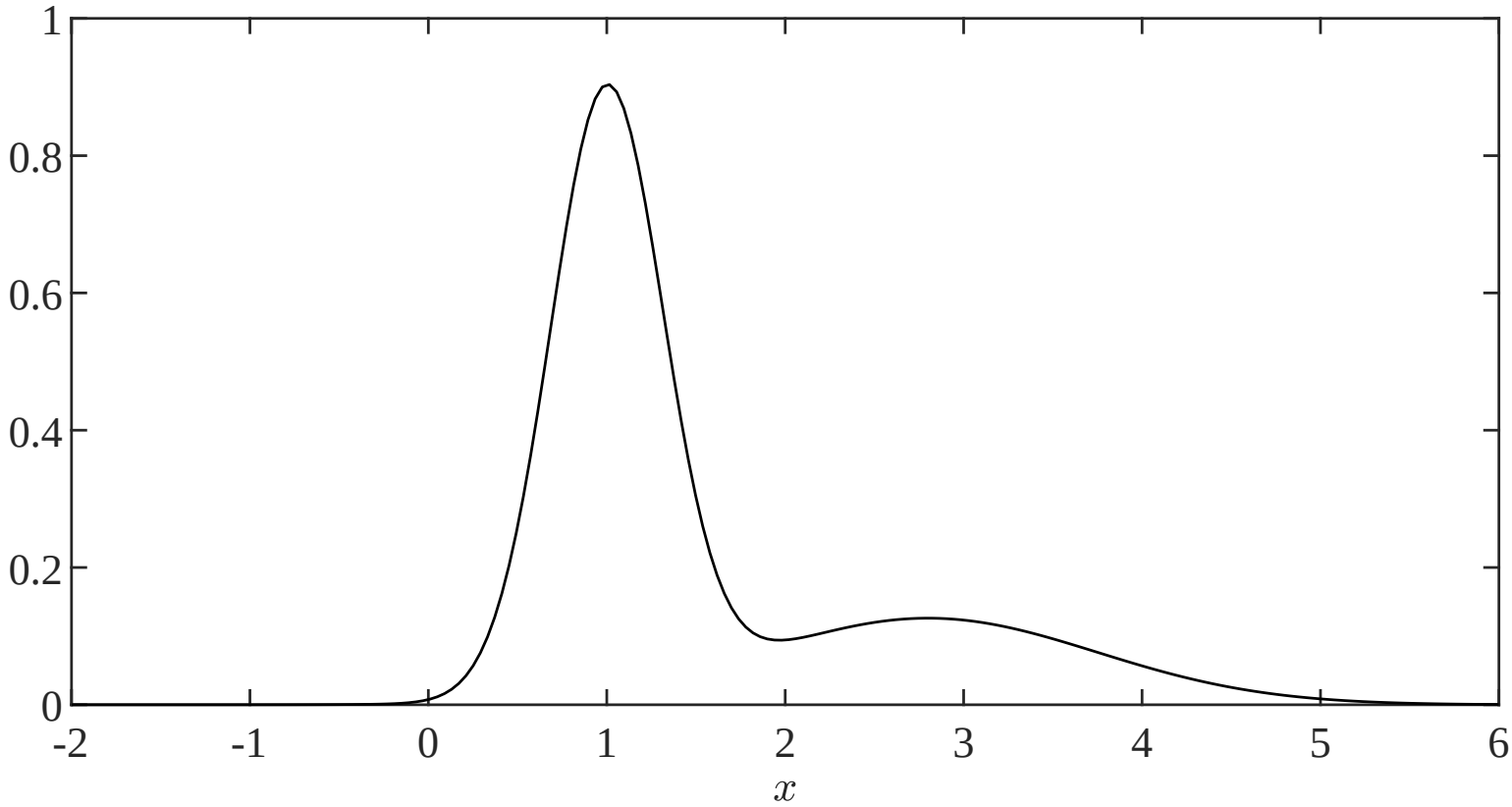
For now we refer to $\pi(\mathbf{x})$ as an **importance density**.

Importance sampling and the curse of dimensionality.

- The Monte-Carlo estimate I_N converges to the true integral value I with convergence rate $O(N^{-1/2})$ as $N \to \infty$, **irrespectively on the dimension of x**.
- However, there may be huge complexity constants hidden in this.
- $\mathbf{g}(\mathbf{x})$ should be factorized so that $\pi(\mathbf{x})$ is large where $\mathbf{g}(\mathbf{x})$ is large.

Example 1: A multi-modal integrand

Use importance sampling to evaluate the integral of a function $g(x)$ looking like this:



Which importance density do you think will work best?

$\pi_1(x) = \mathcal{N}(x\,;\,1, 0.1)$

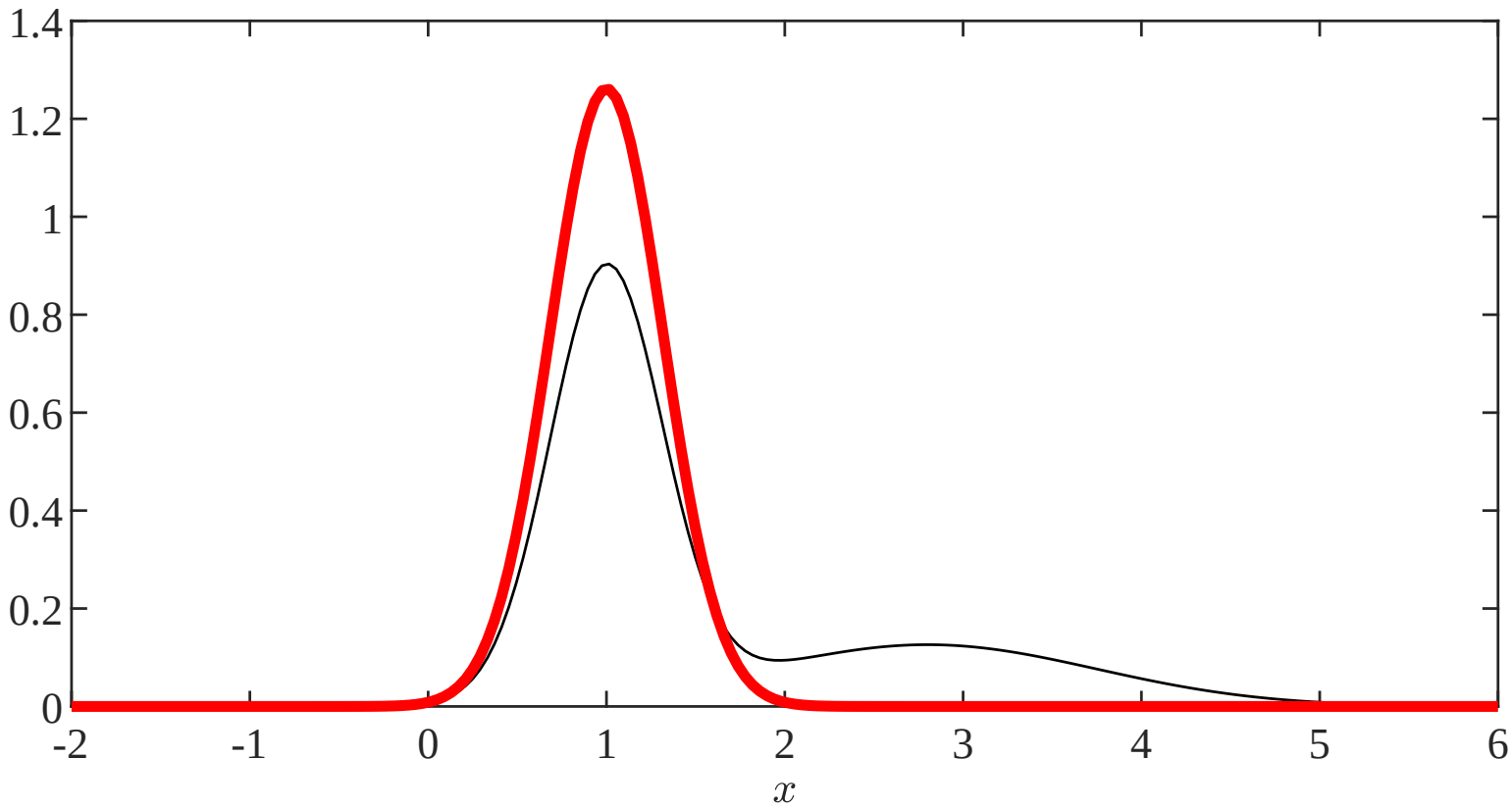
$\pi_2(x) = \mathcal{N}(x\,;\,2.8, 0.9)$

$\pi_3(x) = \mathcal{N}(x\,;\,1.54, 0.34)$

$\pi_4(x) = g(x).$

Example 1: A multi-modal integrand

Use importance sampling to evaluate the integral of a function $g(x)$ looking like this:



Which importance density do you think will work best?

$\pi_1(x) = \mathcal{N}(x\,;\,1, 0.1)$

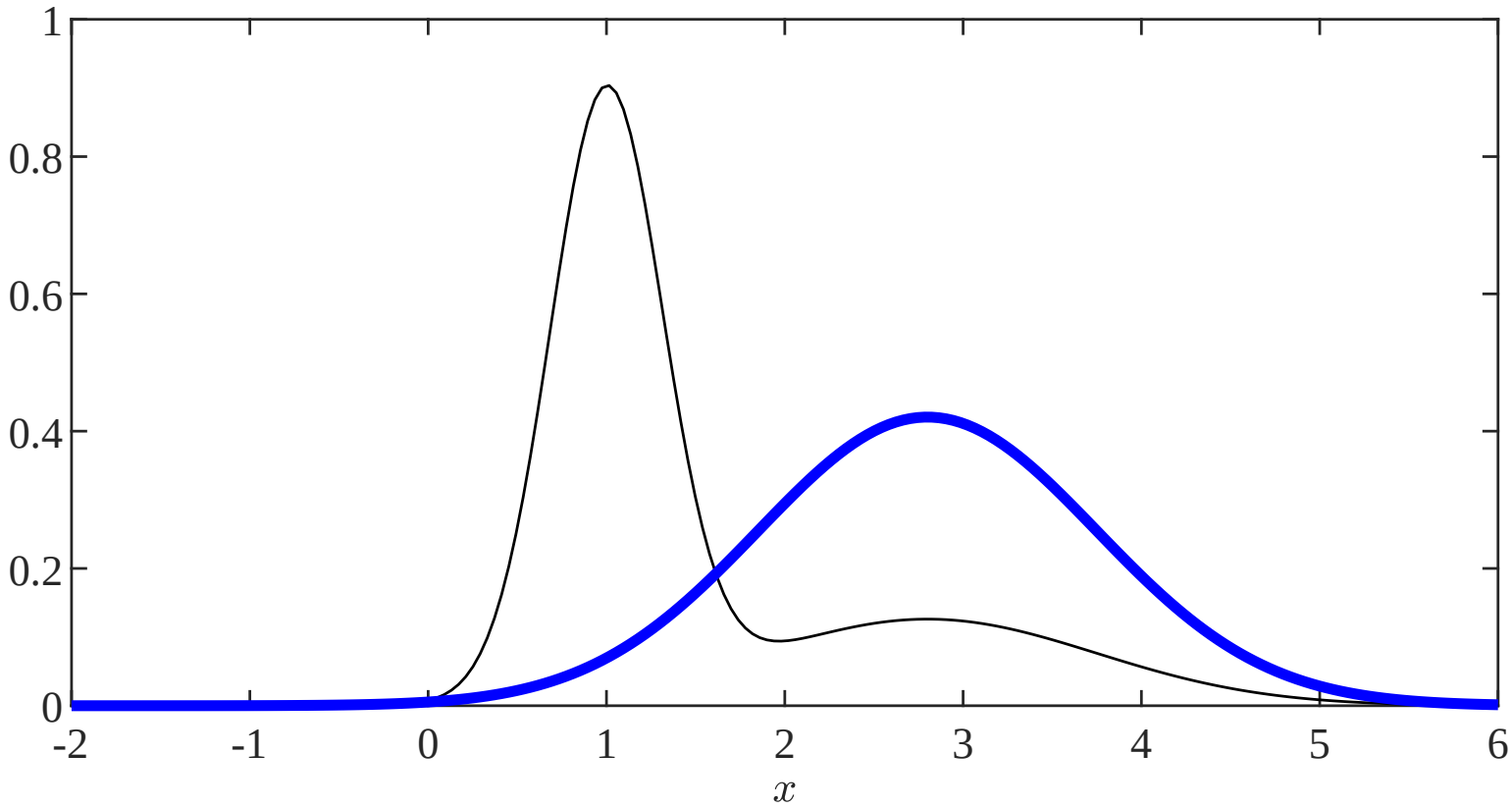
$\pi_2(x) = \mathcal{N}(x\,;\,2.8, 0.9)$

$\pi_3(x) = \mathcal{N}(x\,;\,1.54, 0.34)$

$\pi_4(x) = g(x).$

Example 1: A multi-modal integrand

Use importance sampling to evaluate the integral of a function $g(x)$ looking like this:



Which importance density do you think will work best?

$\pi_1(x) = \mathcal{N}(x\,;\,1, 0.1)$

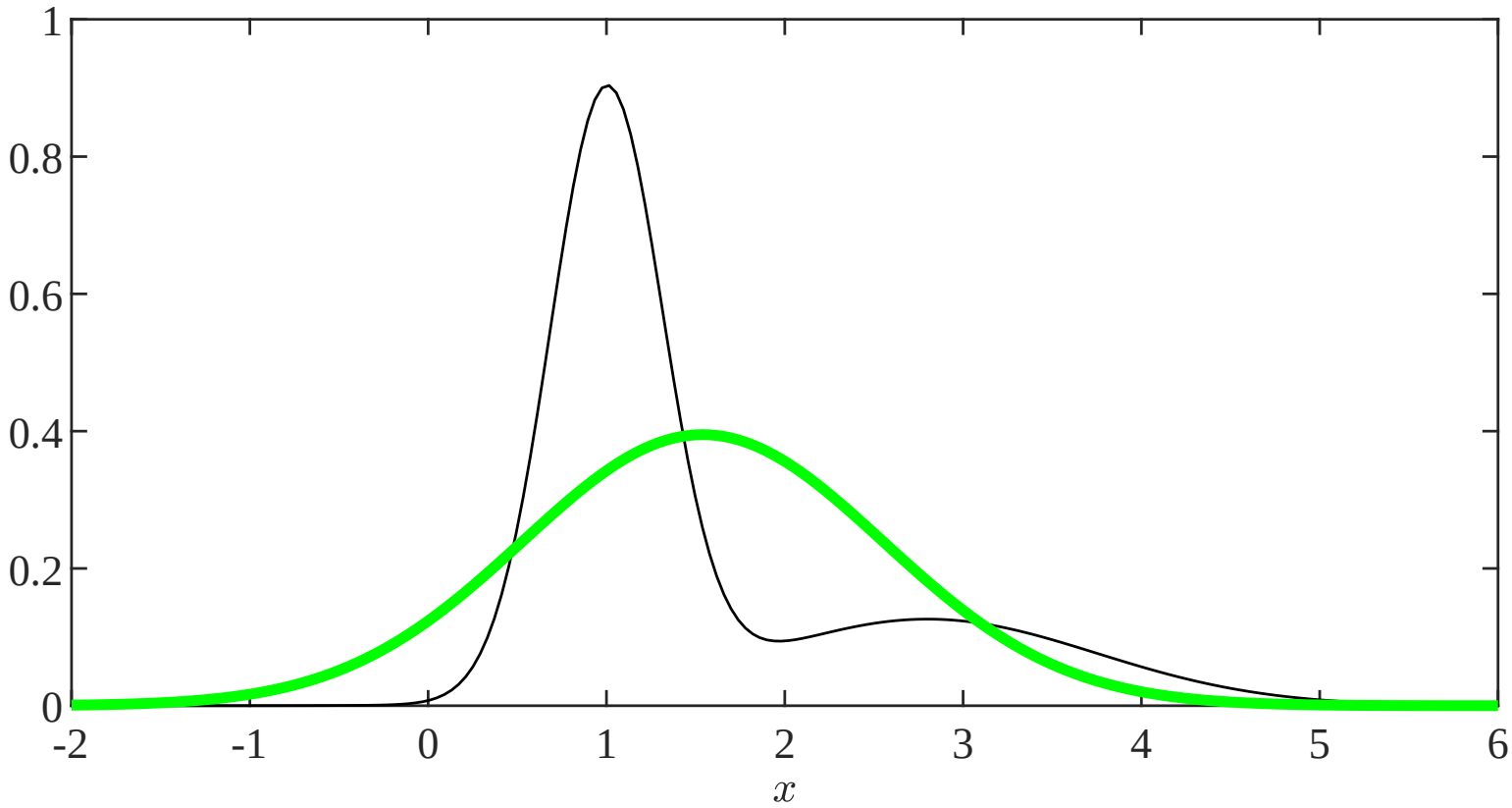
$\pi_2(x) = \mathcal{N}(x\,;\,2.8, 0.9)$

$\pi_3(x) = \mathcal{N}(x\,;\,1.54, 0.34)$

$\pi_4(x) = g(x).$

Example 1: A multi-modal integrand

Use importance sampling to evaluate the integral of a function $g(x)$ looking like this:



Which importance density do you think will work best?

$\pi_1(x) = \mathcal{N}(x\,;\,1, 0.1)$

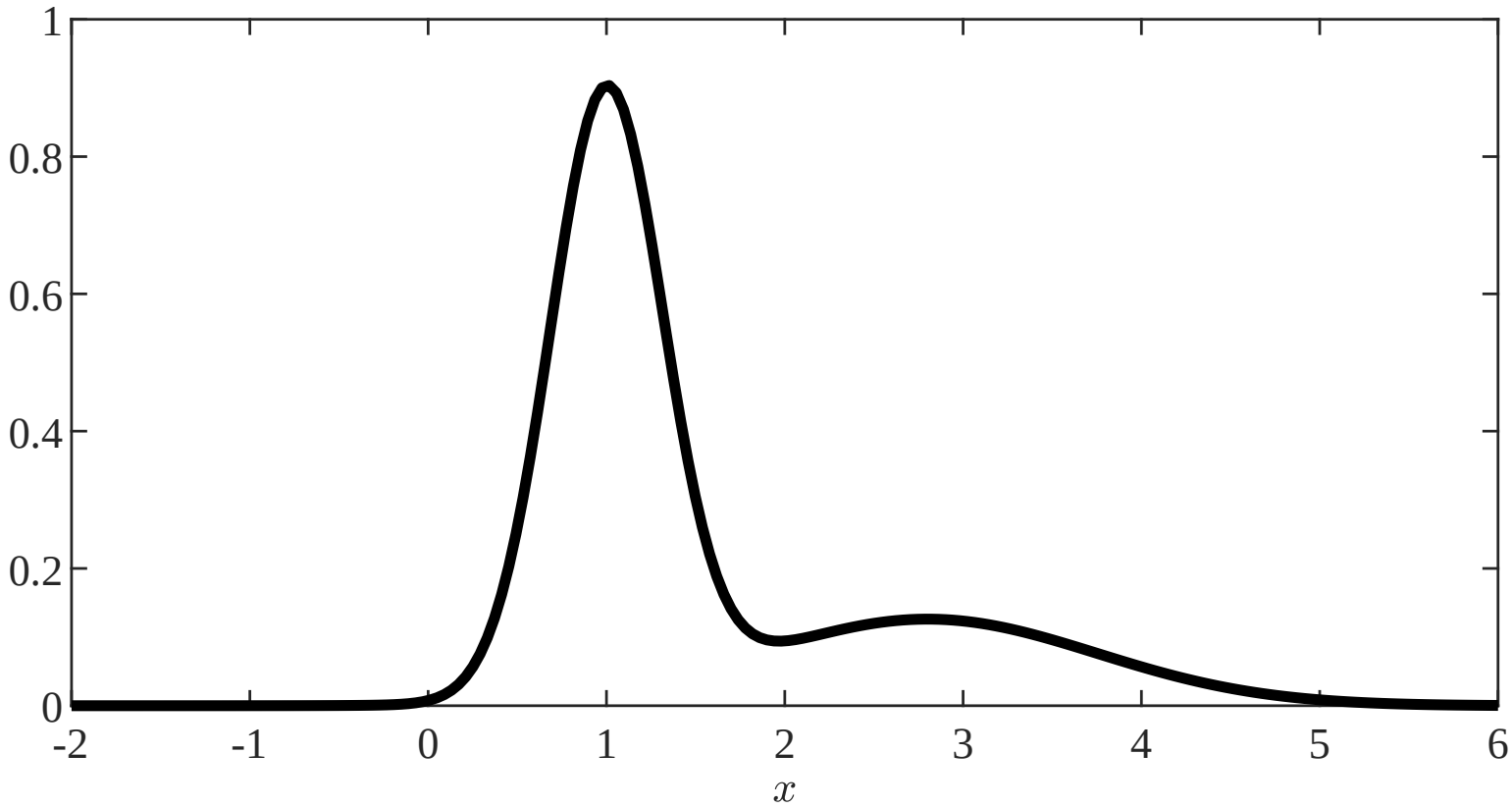
$\pi_2(x) = \mathcal{N}(x\,;\,2.8, 0.9)$

$\pi_3(x) = \mathcal{N}(x\,;\,1.54, 0.34)$

$\pi_4(x) = g(x).$

Example 1: A multi-modal integrand

Use importance sampling to evaluate the integral of a function $g(x)$ looking like this:



Which importance density do you think will work best?

$\pi_1(x) = \mathcal{N}(x\,;\,1, 0.1)$

$\pi_2(x) = \mathcal{N}(x\,;\,2.8, 0.9)$

$\pi_3(x) = \mathcal{N}(x\,;\,1.54, 0.34)$

$\pi_4(x) = g(x).$

Example 2: Collision probabilities

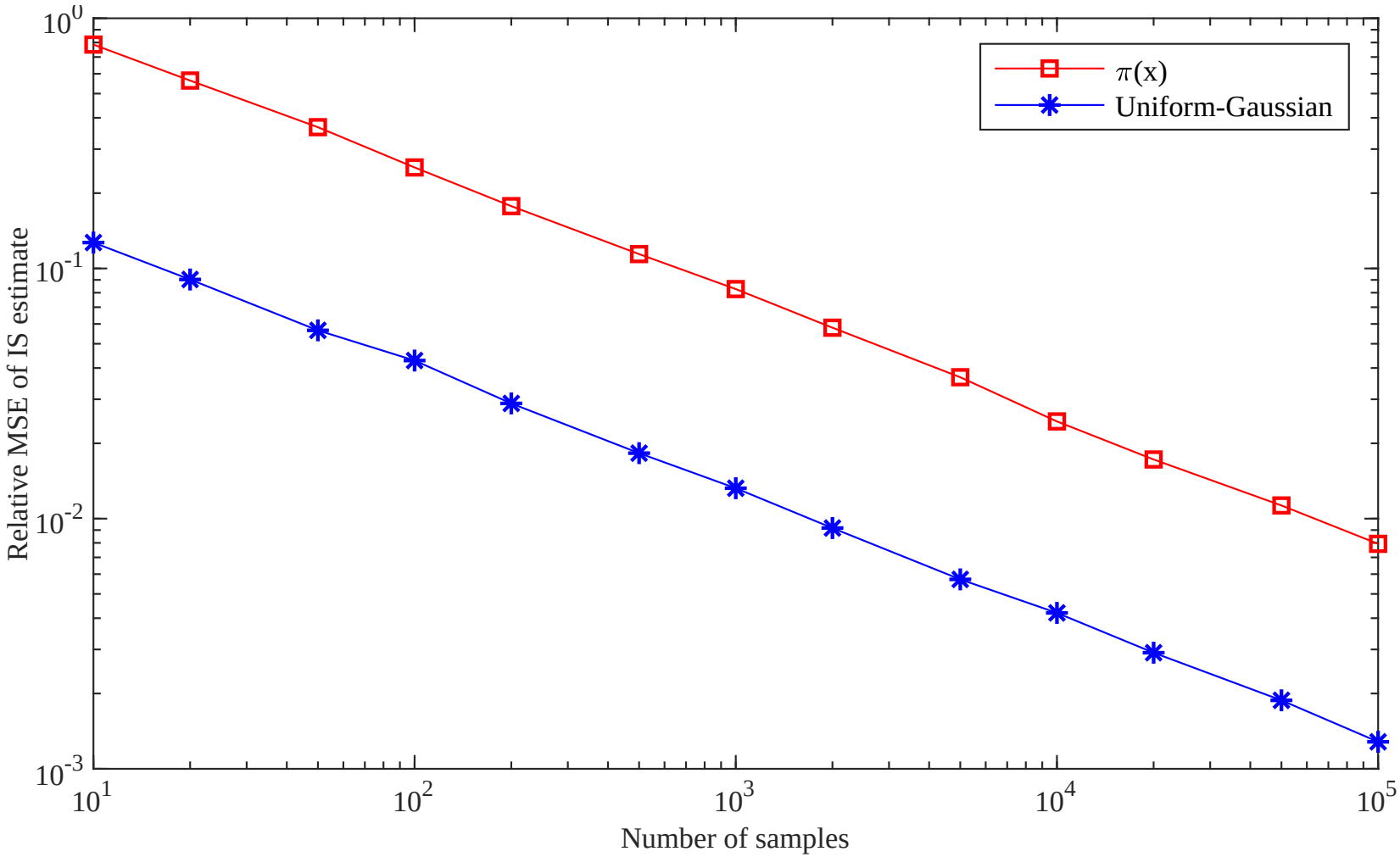
Evaluate the integral $\int_D \mathcal{N}(\mathbf{x}\,;\,\mathbf{a},\mathbf{A})\mathrm{d}\mathbf{x}$ over $D = (-\infty,\infty)\times[0,1]$ where

$$\mathbf{a} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Consider two possible importance densities:

$$\pi_1(\mathbf{x}) = \mathcal{N}(\mathbf{x}\,;\,\mathbf{a},\mathbf{A})$$
$$\pi_2(\mathbf{x}) = \mathcal{N}(x_1\,;\,0,1)\mathrm{Uniform}(x_2\,;\,[0,1])$$

Example 2: Collision probabilities



The role of importance sampling in particle filters

Recall the integral $I = \int \mathbf{f}(\mathbf{x})\pi(\mathbf{x})\mathrm{d}\mathbf{x}$.

Think of $\pi(\mathbf{x})$ as the posterior, and I as some description of the posterior, such as for example the expectation $E[\mathbf{x}]$. (In that case, $\mathbf{f}(\mathbf{x}) = \mathbf{x}$.)

- In a particle filter we only know $\pi(\mathbf{x})$ up to proportionality, and we are not able to sample directly from $\pi(\mathbf{x})$.
- Therefore we define another importance density $q(\mathbf{x})$ so that

$$I = \int \mathbf{f}(\mathbf{x}) \frac{\pi(\mathbf{x})}{q(\mathbf{x})} q(\mathbf{x}) \, \mathrm{d}\mathbf{x}$$

$$w(\mathbf{x}^i) \propto \frac{\pi(\mathbf{x}^i)}{q(\mathbf{x}^i)}$$

$$I_N = \frac{1}{\sum_{i=1}^N w(\mathbf{x}^i)} \sum_{i=1}^N \mathbf{f}(\mathbf{x}^i) w(\mathbf{x}^i) \longrightarrow I \text{ as } N \to \infty.$$

- Must choose $q(\mathbf{x})$ so that $\pi(\mathbf{x})/q(\mathbf{x})$ is upper bounded.

The main task in designing a particle filter is to decide what $q(\mathbf{x})$ should be, and correspondingly calculate the weights $w(\mathbf{x}^i)$ correctly.

Second idea: Bayesian filtering in terms of trajectories

We make the standard Markov assumptions

- $p(\mathbf{x}_k \mid \mathbf{x}_{1:k-1}, \mathbf{z}_{1:k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$
- $p(\mathbf{z}_k \mid \mathbf{x}_{1:k}, \mathbf{z}_{1:k-1}) = p(\mathbf{z}_k \mid \mathbf{x}_k)$

Conventional Bayes filtering (repetition)

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1}) \mathrm{d}\mathbf{x}_{k-1} \tag{2}$$

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k \mid \mathbf{z}_{1:k-1})} \tag{3}$$

Bayes filtering in terms of trajectories

$$p(\mathbf{x}_{1:k} \mid \mathbf{z}_{1:k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{1:k-1} \mid \mathbf{z}_{1:k-1}) \tag{4}$$

$$p(\mathbf{x}_{1:k} \mid \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_{1:k} \mid \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k \mid \mathbf{z}_{1:k-1})} \tag{5}$$

The old posterior distribution

If the particles provide a representation of $p(\mathbf{x}_{1:k-1}|\mathbf{z}_{1:k-1})$, how can we turn them into a representation of $p(\mathbf{x}_{1:k}|\mathbf{z}_{1:k})$?

The particle cloud at time $k-1$ ...

At time $k-1$ we have a set of N particles

$$\mathbf{x}_{1:k-1}^i = \begin{bmatrix} \mathbf{x}_1^i & \mathbf{x}_2^i & \dots & \mathbf{x}_{k-2}^i & \mathbf{x}_{k-1}^i \end{bmatrix}, \quad i = 1, \dots, N.$$

- Assume that these are drawn according to $q(\mathbf{x}_{1:k-1} \mid \mathbf{z}_{1:k-1})$.
- The particles have normalized weights $w_{k-1}^i \propto \dfrac{p(\mathbf{x}_{1:k-1}^i \mid \mathbf{z}_{1:k-1})}{q(\mathbf{x}_{1:k-1}^i \mid \mathbf{z}_{1:k-1})}$.

... can be used in importance sampling for $\int \mathbf{f}(\mathbf{x}_{1:k-1}) p(\mathbf{x}_{1:k-1}|\mathbf{z}_{1:k-1}) \mathrm{d}\mathbf{x}_{1:k-1}$.

The particle cloud then provides a representation of $p(\mathbf{x}_{1:k-1}^i \mid \mathbf{z}_{1:k-1})$:

- $p(\mathbf{x}_{1:k-1}^i|\mathbf{z}_{1:k-1}) \approx \sum_{i=1}^N w_{k-1}^i \delta(\mathbf{x}_{1:k-1} - \mathbf{x}_{1:k-1}^i)$.
- $E[\mathbf{x}_{1:k-1}|\mathbf{z}_{1:k-1}] = \int \mathbf{x}_{1:k-1}\, p(\mathbf{x}_{1:k-1} \mid \mathbf{z}_{1:k-1})\, \mathrm{d}\mathbf{x}_{1:k-1} \approx \sum_{i=1}^N w_{k-1}^i \mathbf{x}_{1:k-1}^i$.

Augmenting the particles with samples at time k

- We have already sampled $\mathbf{x}_{1:k-1}^i$ from $q(\mathbf{x}_{1:k-1} \mid \mathbf{z}_{1:k-1})$.
- Now we want to sample $\mathbf{x}_{1:k}^i$ from $q(\mathbf{x}_{1:k} \mid \mathbf{z}_{1:k})$.

$\Rightarrow$ Can we define $q(\mathbf{x}_{1:k} \mid \mathbf{z}_{1:k})$ so that the sampling already done can be re-used?

Proposing new samples.

If the importance density is chosen to factorize such that

$$q(\mathbf{x}_{1:k} \mid \mathbf{z}_{1:k}) = q(\mathbf{x}_k \mid \mathbf{x}_{1:k-1}, \mathbf{z}_{1:k}) q(\mathbf{x}_{1:k-1} \mid \mathbf{z}_{1:k-1}) \tag{6}$$

then we can obtain the new samples $\mathbf{x}_{1:k}^i$ by augmenting each of the existing samples with new state vectors drawn from $q(\mathbf{x}_k | \mathbf{x}_{1:k-1}^i, \mathbf{z}_{1:k})$.

The augmenting operation means that we sample conditional on $\mathbf{x}_{1:k-1}^i$. That is, we sample trajectories.

But why not just sample $\mathbf{x}_k^i$ without this trajectory stuff?

- We could sample $\mathbf{x}_k^i$ from any suitable proposal density $q(\mathbf{x}_k | \mathbf{z}_{1:k})$.
- To calculate the weights we would then need $p(\mathbf{x}_k^i | \mathbf{z}_{1:k})$. **But we we don't have any expression for this pdf.**

The new posterior distribution

- From the Bayesian filtering equations for trajectories (5), it follows that the posterior density of $\mathbf{x}_{1:k}$ is proportional to

$$p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{1:k-1} \mid \mathbf{z}_{1:k-1})$$

- Repeating (6), our proposal density $q(\mathbf{x}_{1:k} \mid \mathbf{z}_{1:k})$ is of the form

$$q(\mathbf{x}_k \mid \mathbf{x}_{1:k-1}, \mathbf{z}_{1:k}) q(\mathbf{x}_{1:k-1} \mid \mathbf{z}_{1:k-1})$$

$\Rightarrow$ By inserting this into the fundamental weight formula we obtain

$$\begin{aligned} w_k^i &\propto \frac{p(\mathbf{x}_{1:k}^i \mid \mathbf{z}_{1:k})}{q(\mathbf{x}_{1:k}^i \mid \mathbf{z}_{1:k})} \\ &\propto \frac{p(\mathbf{z}_k \mid \mathbf{x}_k^i) p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)\, p(\mathbf{x}_{1:k-1}^i \mid \mathbf{z}_{1:k-1})}{q(\mathbf{x}_k^i \mid \mathbf{x}_{1:k-1}, \mathbf{z}_{1:k})\, q(\mathbf{x}_{1:k-1}^i \mid \mathbf{z}_{1:k-1})} \\ &\propto \frac{p(\mathbf{z}_k \mid \mathbf{x}_k^i) p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i \mid \mathbf{x}_{1:k-1}, \mathbf{z}_{1:k})}\, w_{k-1}^i \end{aligned}$$

Because the formula is recursive we can obtain the particle weights without ever evaluating $p(\mathbf{x}_{1:k}^i \mid \mathbf{z}_{1:k})$ directly.

From trajectories to filtering

But we are not really interested in dealing with trajectories $\mathbf{x}_{1:k}$. We only want the posterior of the current state $\mathbf{x}_k$!

Filtering and Markovian importance density

- In filtering (as opposed to, e.g., smoothing) we are interested in the marginal posterior distribution

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) = \int p(\mathbf{x}_{1:k} \mid \mathbf{z}_{1:k}) \, \mathrm{d}\mathbf{x}_{1:k-1}$$

  We simply "forget" the previous estimates $\mathbf{x}_{1:k-1}$ from each particle.
- It is therefore not desirable to invoke ancient states in the importance density. The importance density must then be of the form $q(\mathbf{x}_k \mid \mathbf{x}^i_{1:k-1}, \mathbf{z}_{1:k}) = q(\mathbf{x}_k \mid \mathbf{x}^i_{k-1}, \mathbf{z}_k)$.
- The importance weights are then given by

$$w^i_k \propto w^i_{k-1} \frac{p(\mathbf{z}_k \mid \mathbf{x}^i_k) \, p(\mathbf{x}^i_k \mid \mathbf{x}^i_{k-1})}{q(\mathbf{x}^i_k \mid \mathbf{x}^i_{k-1}, \mathbf{z}_k)}$$

Nevertheless, we are in reality sampling from $q(\mathbf{x}_{1:k}|\mathbf{z}_{1:k})$. This means that the sampling space is growing indefinitely.

Degeneracy

Unfortunately, discarding the previous states $\mathbf{x}_{1:k-1}$ doesn't change the fact that we are sampling $\mathbf{x}_{1:k}$ and not just $\mathbf{x}_k$.

Degeneracy

- In early experiments with particle filters, it was observed that after a short time one particle had all the probability mass.
- This degeneracy problem can be monitored by means of the effective sample size

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2}$$

- $\hat{N}_{\text{eff}} \approx N$ is good, while $\hat{N}_{\text{eff}} \to 1$ is a sign of degeneracy.

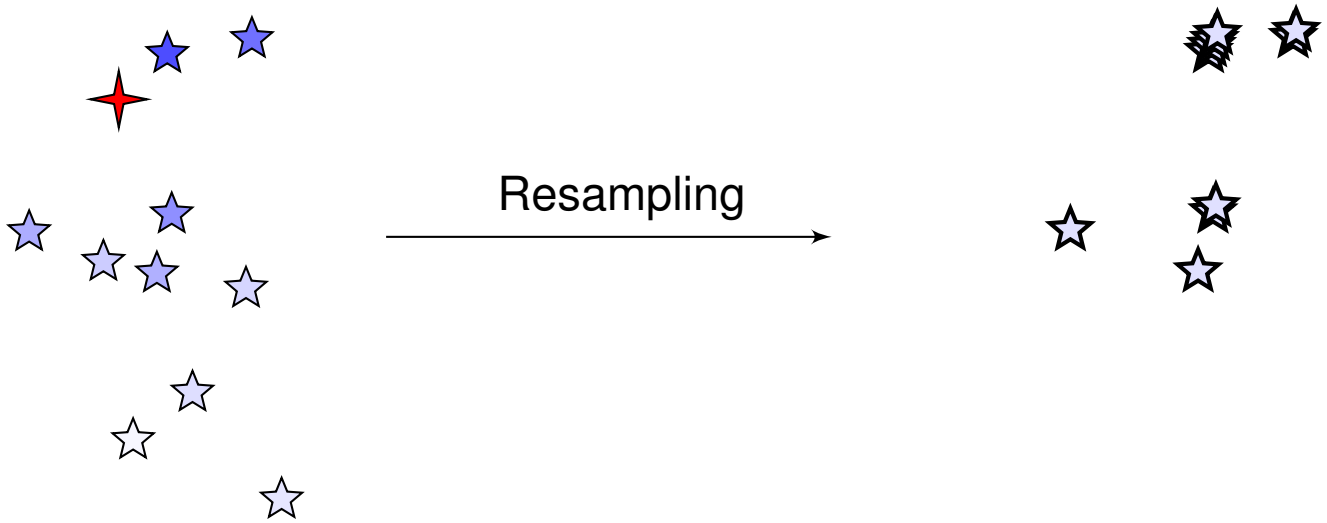
Causes of degeneracy

- **Increasing dimension of trajectory:** It will be more and more difficult for the particle filter to place the particles where $p(\mathbf{x}_{1:k}|\mathbf{z}_{1:k})$ is large.
- **The weight as a betting process:** More and more particles will eventually have made a bad bet. The particle that has made the least bad bets gets all the probability mass.

Resampling: A solution to degeneracy

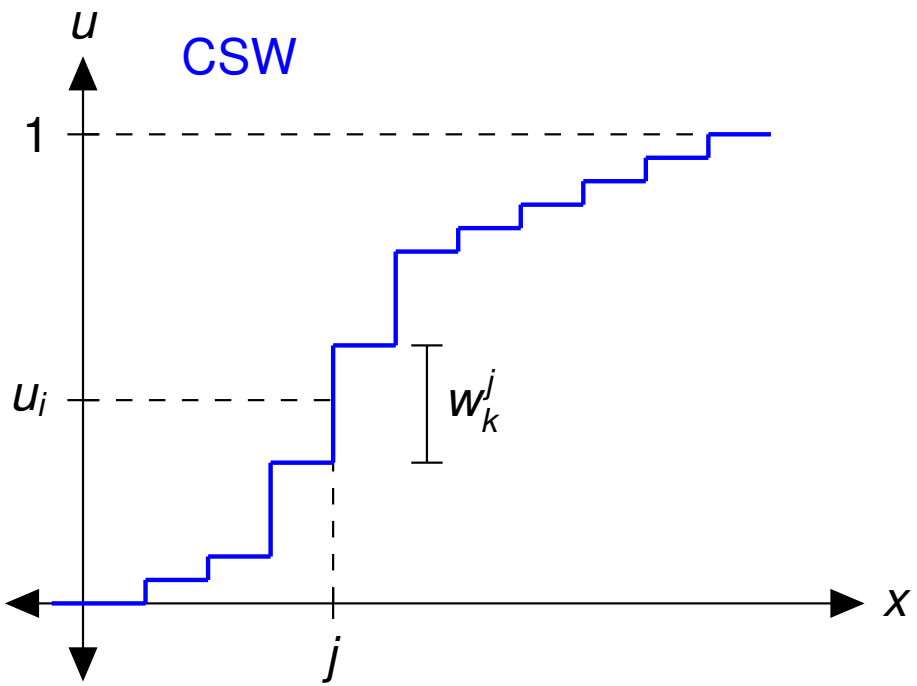
Idea 3: Resampling.

- When $\hat{N}_{\text{eff}}$ goes below some threshold we resample.
- **Replicate the existing particles into a resampled set where each old particle is equal to any of the old particles with probability equal to the old weights.**
- The resampled particles have uniform weights.
- Main drawback: Resampling kills particles $\longrightarrow$ particle depletion.



Resampling method 1 - CDF inversion

- Calculate the cumulative distribution function (CDF) of the particle weights.
- Draw a uniform number u_i for each new particle.
- Let the new particle be equal to the old particle for which CDF crosses u_i.



- For this scheme to work as efficiently as possible, we must draw N numbers from $\mathrm{unif}(u_i\,;\,[0,1])$ and sort these numbers.
- Thus, straightforward CDF inversion has complexity $O(N \log N)$.

Resampling method 2 - Systematic resampling

- Systematic resampling is an alternative method with complexity $O(N)$.
- In systematic resampling we draw one random number and then move along the CDF until all the new particles have been assigned an old particle.

```
function indicesout = rouletteSystematic(weights,k)

cumweights = cumsum(weights);
m = length(cumweights);

indicesout = zeros(k,1);
noise = rand(1,1)/k;

i = 1;
for j=1:k
    uj = noise + (j-1)/k;
    while uj > cumweights(i)
        i = i + 1;
    end
    indicesout(j) = i;
end
indicesout = indicesout(randperm(size(indicesout,1)));
```

How to make particle filters work

Use a good importance density

- We can propose new particles simply by means of the process model.
- It is possible to do better by also including knowledge about $\mathbf{z}_k$.

Regularization

- Similar to inflating the process noise covariance.
- But slightly less dirty. Often hard to avoid.

We will discuss regularization in the next session.

Rao-Blackwellization

- Only sample the nonlinear states.
- Use a KF or EKF for the linear states.

We will postpone Rao-Blackwellization to our treatment of SLAM later in the course.

Selection of the importance density

The Sampling Importance Resampling (SIR) filter.

The simplest and most popular choice is to use the process model as importance density:

$$q(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i, \mathbf{z}_k) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)$$

For this choice the weight update becomes

$$\tilde{w}_k^i = w_{k-1}^i p(\mathbf{z}_k \mid \mathbf{x}_k^i).$$

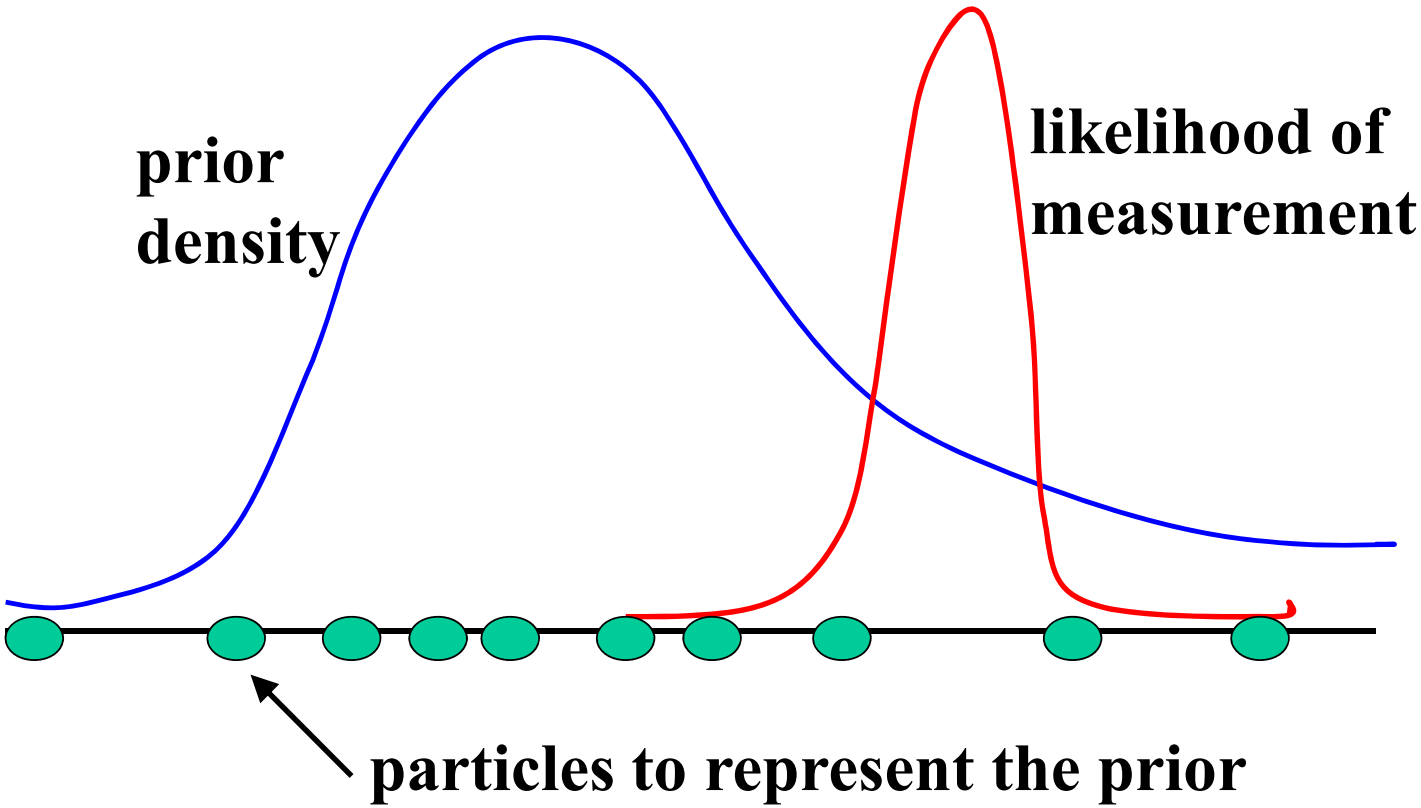
Algorithm 1 One iteration of the SIR filter

1: **for** $i = 1$ to N **do**
2: Draw $\mathbf{x}_k^i \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)$
3: Calculate $\tilde{w}_k^i = w_{k-1}^i p(\mathbf{z}_k \mid \mathbf{x}_{k-1}^i)$
4: **end for**
5: Normalize the weights: $w_k = \tilde{w}_k / \sum_i \tilde{w}_k^i$.
6: Resample using e.g. systematic resampling.

- Does not exploit information available in the current measurement.
- Runs into serious difficulties for dimensions > 4.

A problem with the SIR filter

- Poor placement of the particle if the overlap between prior and likelihood is poor.[3]
- The better the measurements, the worse the problem.



[3] Illustration from Daum & Huang (2011): "Particle degeneracy: root cause and solution", Proc. SPIE.

Selection of the proposal density

To do better than the SIR filter we can use information from the latest measurement.

The optimal importance density

For particle i at time step k, the best we can do is to sample from the posterior density of $\mathbf{x}_k$ conditional on $\mathbf{x}_{k-1}^i$ and $\mathbf{z}_k$:

$$q(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i, \mathbf{z}_k)_{\text{opt}} = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i, \mathbf{z}_k) = \frac{p(\mathbf{z}_k \mid \mathbf{x}_{k-1}^i, \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)}{p(\mathbf{z}_k \mid \mathbf{x}_{k-1}^i)}$$

- The weight update is now given by

$$w_k^i \propto w_{k-1}^i p(\mathbf{z}_k \mid \mathbf{x}_{k-1}^i) = w_{k-1}^i \int p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i)\, \mathrm{d}\mathbf{x}_k$$

  Evaluation of this integral may be very difficult.
- It may also be very difficult to sample from $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i, \mathbf{z}_k)$.

Selection of the proposal density

Optimal density for nonlinear dynamics and Gaussian noise

Let the state space model be $\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}) + \mathbf{v}_{k-1}$, $\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{w}_k$, and let $\mathbf{v}_{k-1}$ and $\mathbf{w}_k$ be mutually independent zero-mean white Gaussian sequences with covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$.

In this case

$$q(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i, \mathbf{z}_k)_{\text{opt}} = \mathcal{N}(\mathbf{x}_k\,;\, \mathbf{a}_k, \Sigma_k)$$
$$p(\mathbf{z}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{z}_k\,;\, \mathbf{b}_k, \mathbf{S}_k)$$

where

$$\mathbf{a}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}) + \Sigma_k\mathbf{H}_k^{\mathsf{T}}\mathbf{R}_k^{-1}(\mathbf{z}_k - \mathbf{b}_k)$$
$$\Sigma_k = \mathbf{Q}_k - \mathbf{Q}_k\mathbf{H}_k^{\mathsf{T}}\mathbf{S}_k^{-1}\mathbf{H}_k\mathbf{Q}_k$$
$$\mathbf{S}_k = \mathbf{H}_k\mathbf{Q}_k\mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k$$
$$\mathbf{b}_k = \mathbf{H}_k\mathbf{f}(\mathbf{x}_{k-1}).$$

Sampling from a multivariate Gaussian is easy:

```
xCloudNew = a(z,xCloud) + chol(Sigma)'*randn(n,m)
```

Selection of the proposal density

Local linearization

- Approximate $q(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i, \mathbf{z}_k)_{\text{opt}}$ by means of the output from an EKF/UKF:

$$q(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i, \mathbf{z}_k) = \mathcal{N}(\mathbf{x}_k^i \,;\, \hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i)$$

- For each particle i we run an EKF/UKF: $(\mathbf{x}_{k-1}^i, \hat{\mathbf{P}}_{k-1}^i) \longrightarrow (\bar{\mathbf{x}}_k^i, \bar{\mathbf{P}}_k^i) \longrightarrow (\hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i)$.
- Then we sample the new particles,
- and calculate their weights according to

$$\tilde{w}_k^i = w_{k-1}^i \frac{p(\mathbf{z}_k \mid \mathbf{x}_k^i) p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)}{\mathcal{N}(\mathbf{x}_k^i \,;\, \hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i)}$$

- and finally we normalize and resample. The resampled particles inherit the covariances of their parents.

Improving the sample diversity

Resampling can cause particle depletion: Only replicates of a few good particles take over the entire particle cloud. Subsequent steps of importance sampling may not contain sufficient noise to counteract the collapse.

Solution: The regularized particle filter (RPF).

- Before resampling, the RPF calculates the empirical covariance $\mathbf{S}_k$ of the particle cloud, and its Cholesky factorization $\mathbf{D}_k$.
- After resampling, the RPF regularizes the particle cloud according to

$$\mathbf{x}_k^{i*} = \mathbf{x}_k^i + h_{\text{opt}} \mathbf{D}_k \epsilon^i$$

where
  - ϵ^i is drawn from a scaled kernel density $K_h(\mathbf{x}) = \frac{1}{h^{n_x}} K\left(\frac{\mathbf{x}}{h}\right)$ with bandwidth h.
  - h_{opt} is the optimal bandwidth value.

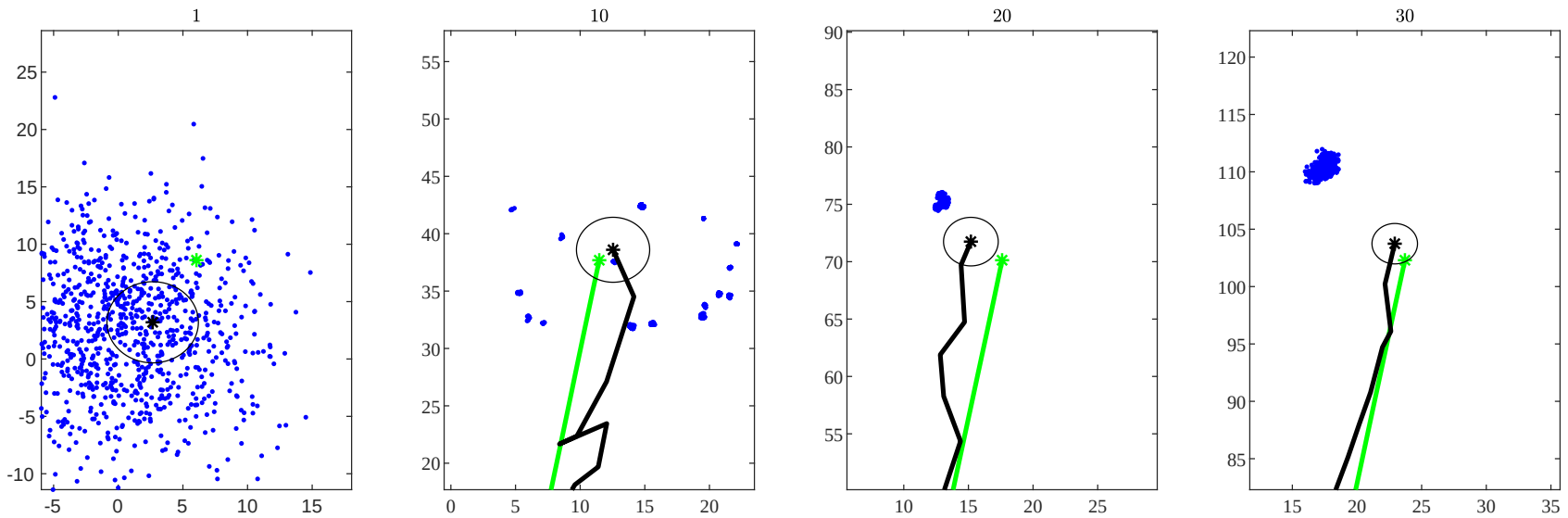
See the textbook Section 5.2.6 for further details.

To ensure convergence towards true posterior as $N \to \infty$ the regularization step $\mathbf{x}_k^{i*} = \mathbf{x}_k^i + h_{\text{opt}} \mathbf{D}_k \epsilon^i$ can be accepted or rejected according to the Metropolis-Hastings algorithm.[4]

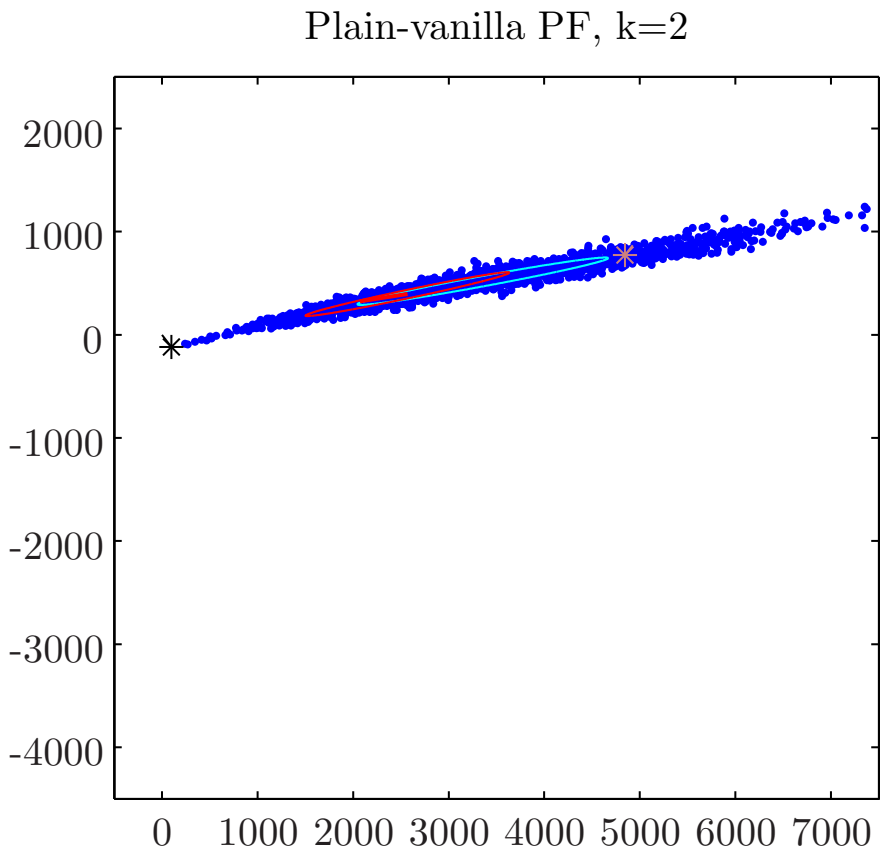
[4] Ristic et al. (2004) “Beyond the Kalman Filter: Particle Filters for Tracking Applications”, Artech House.

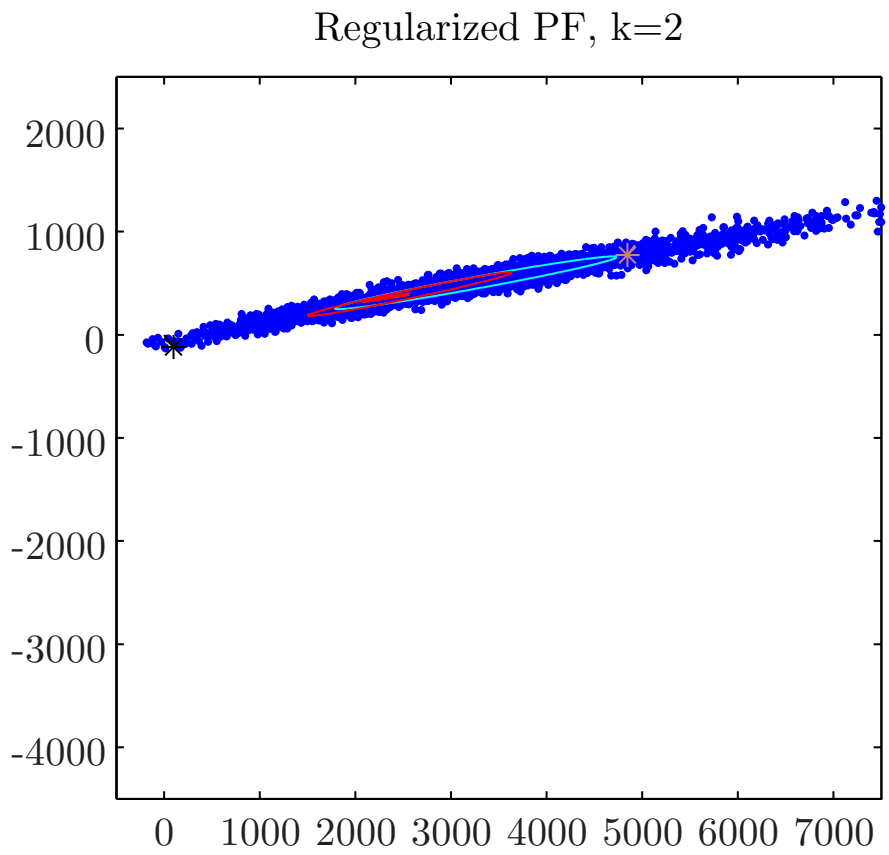
Example of particle depletion

- We simulate a simple scenario with the CV model and position measurements.
- We use a fairly low process noise $\sigma_a = 0.05$ and typical GPS-resolution measurement noise $\sigma_z = 5$.
- Particle depletion causes the particles to form compact clusters, and this leads to divergence.



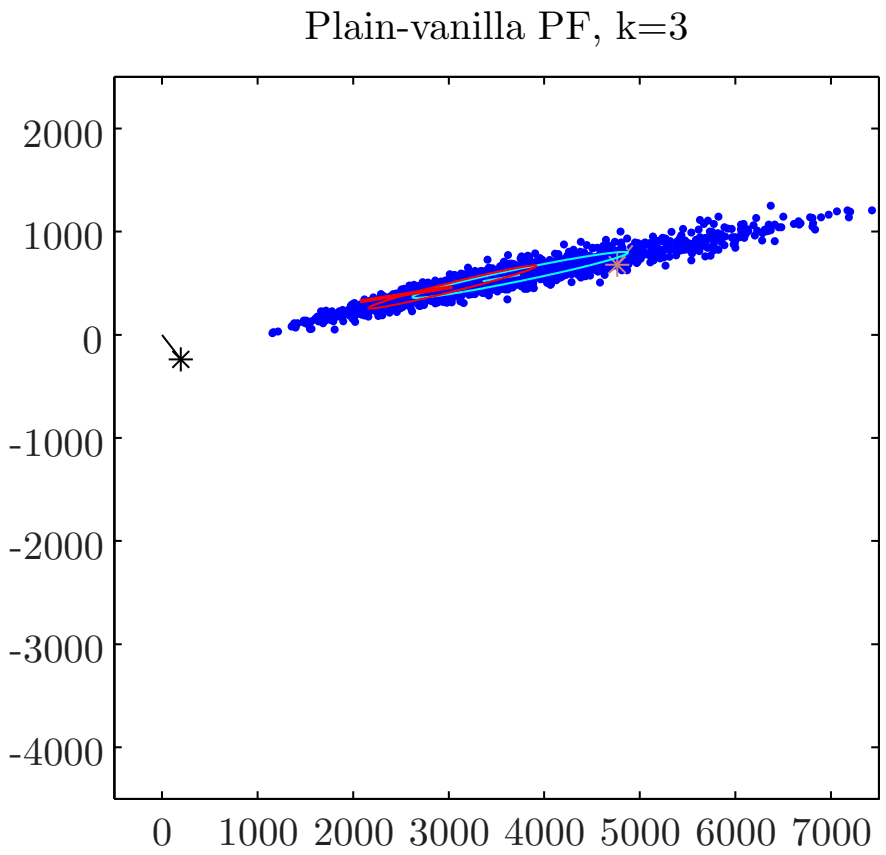
Plain-vanilla vs regularized PF for bearing-only

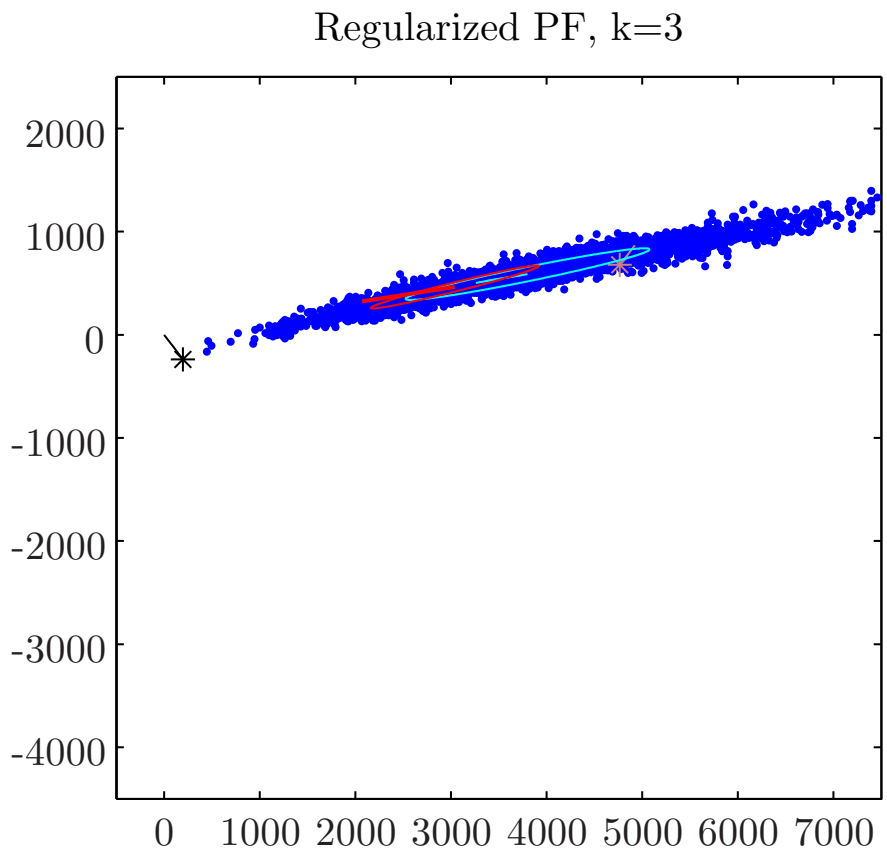




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

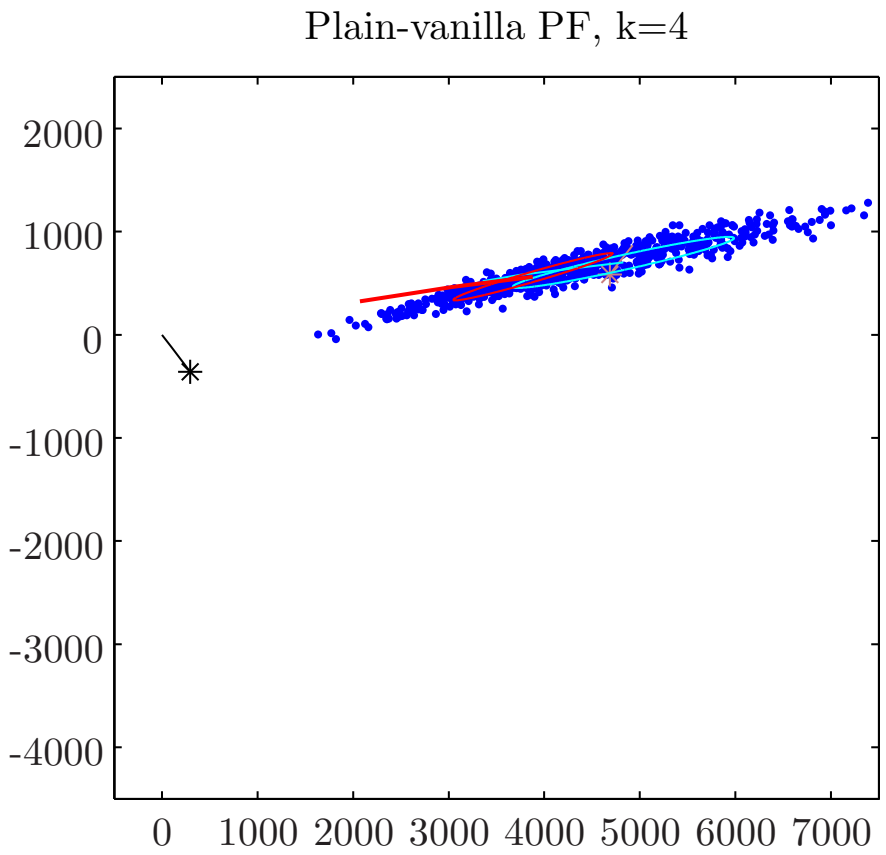
Plain-vanilla vs regularized PF for bearing-only

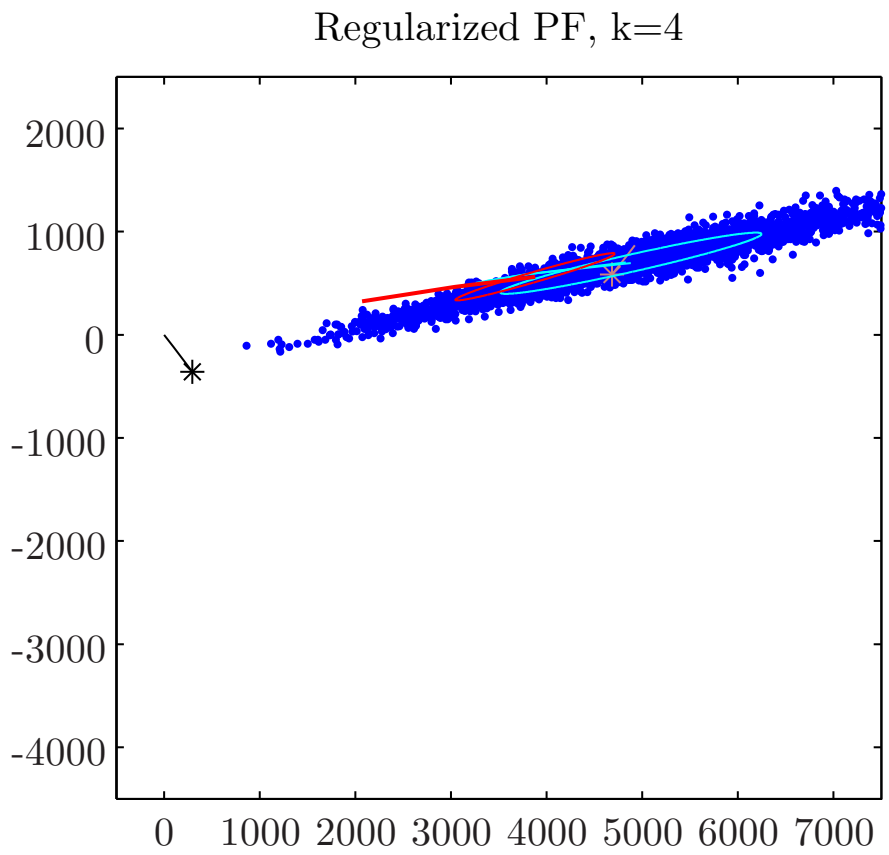




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

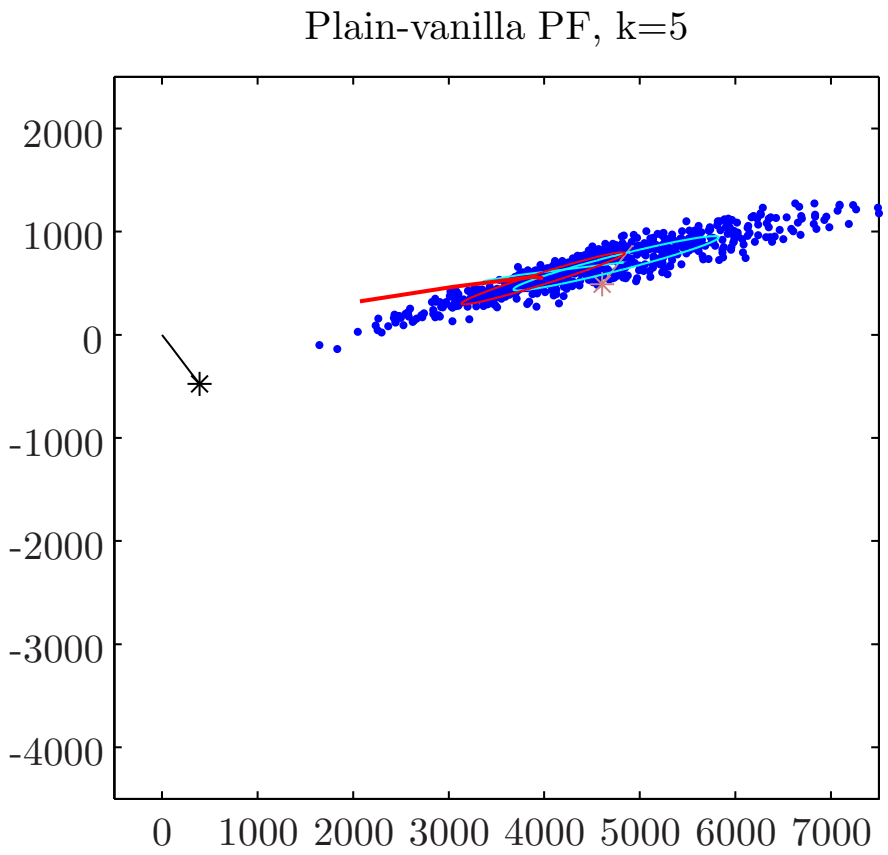
Plain-vanilla vs regularized PF for bearing-only

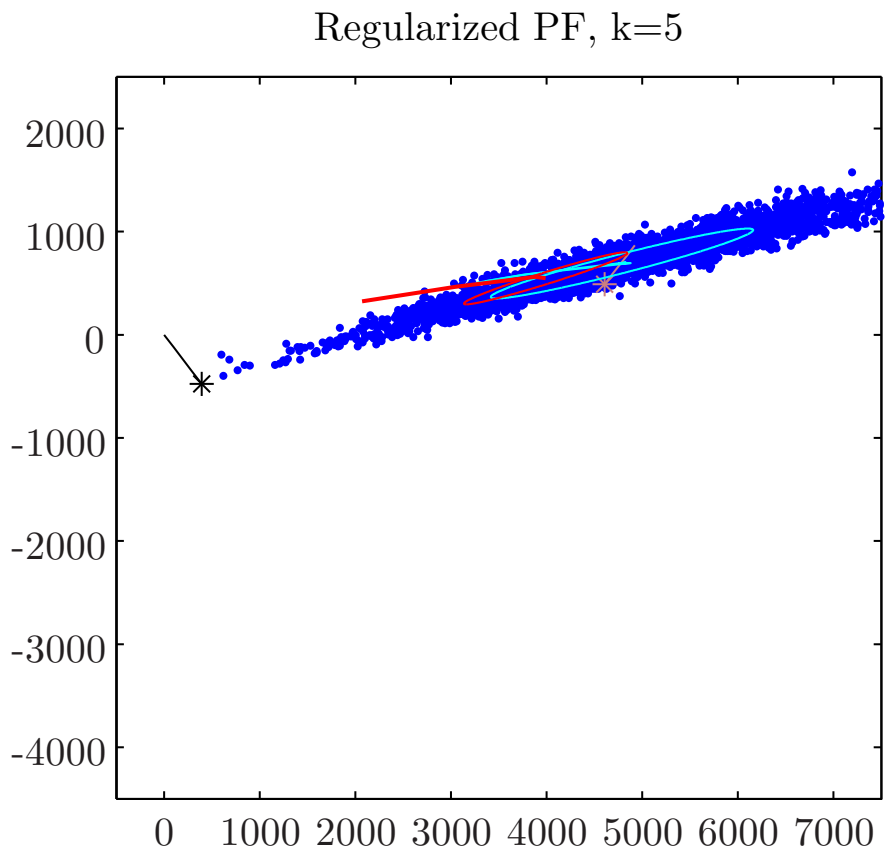




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

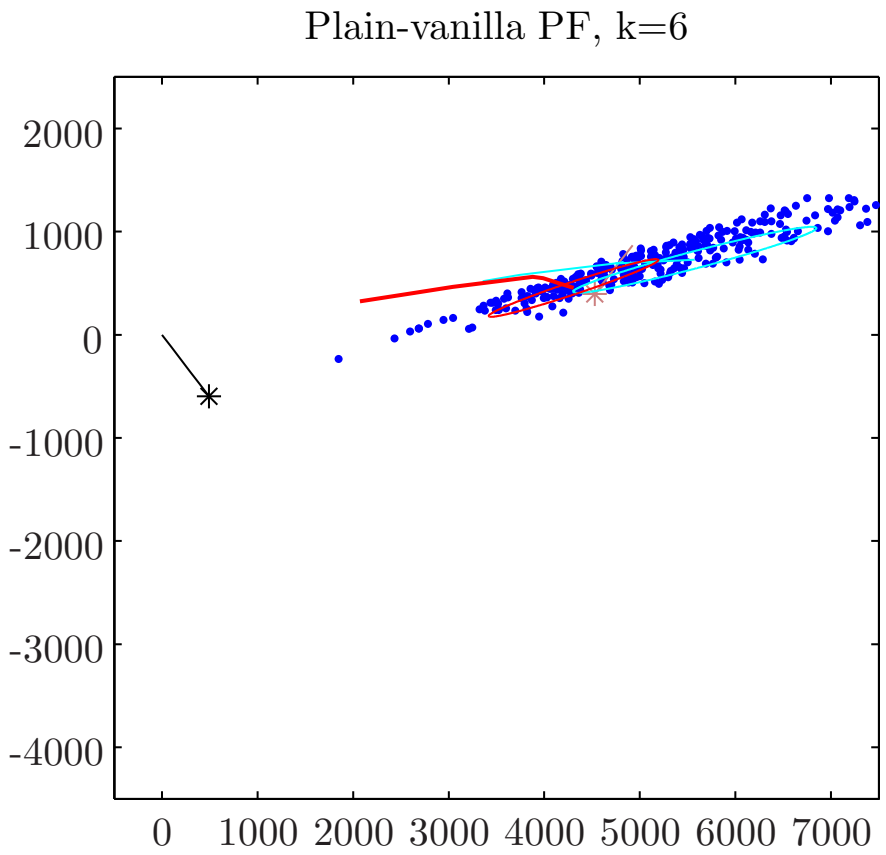
Plain-vanilla vs regularized PF for bearing-only

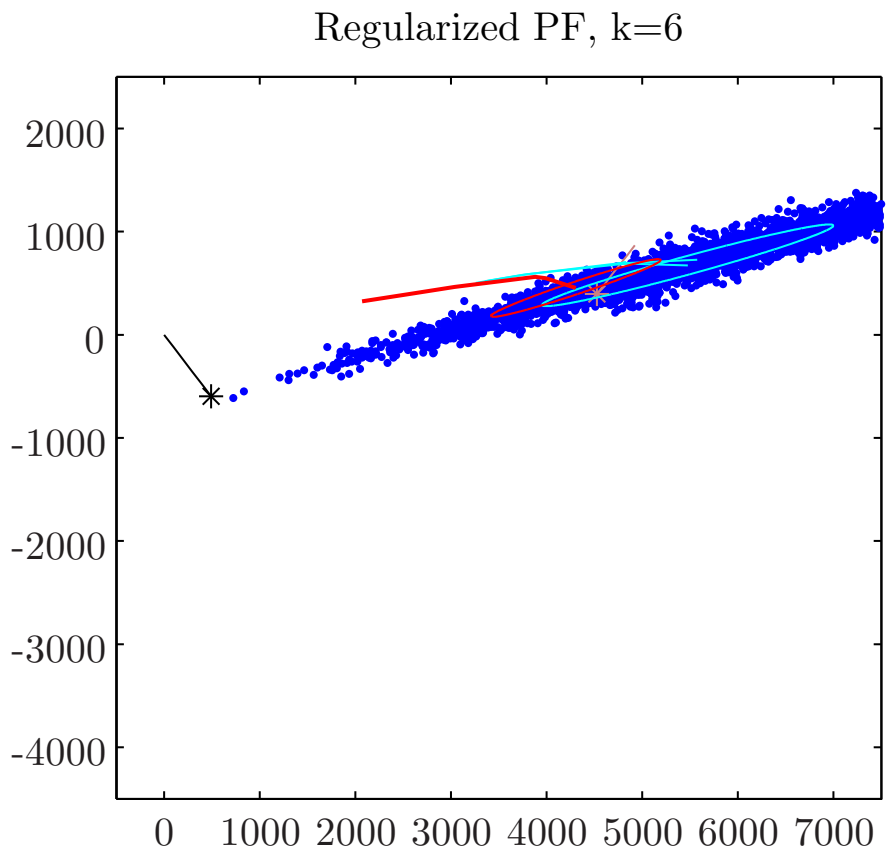




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

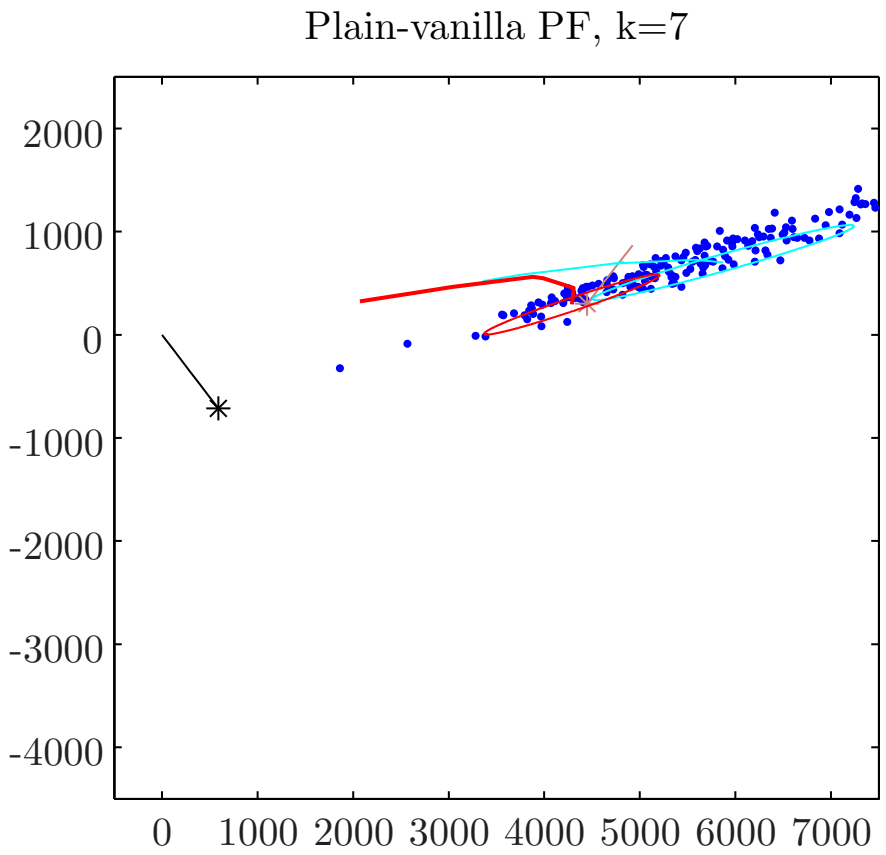
Plain-vanilla vs regularized PF for bearing-only

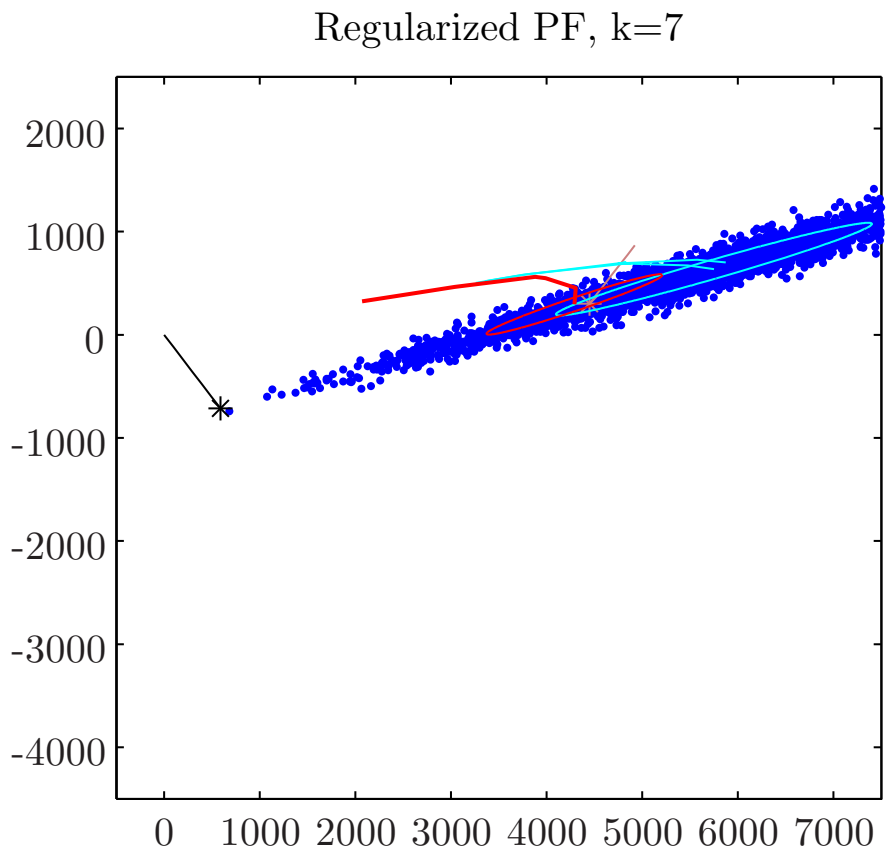




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

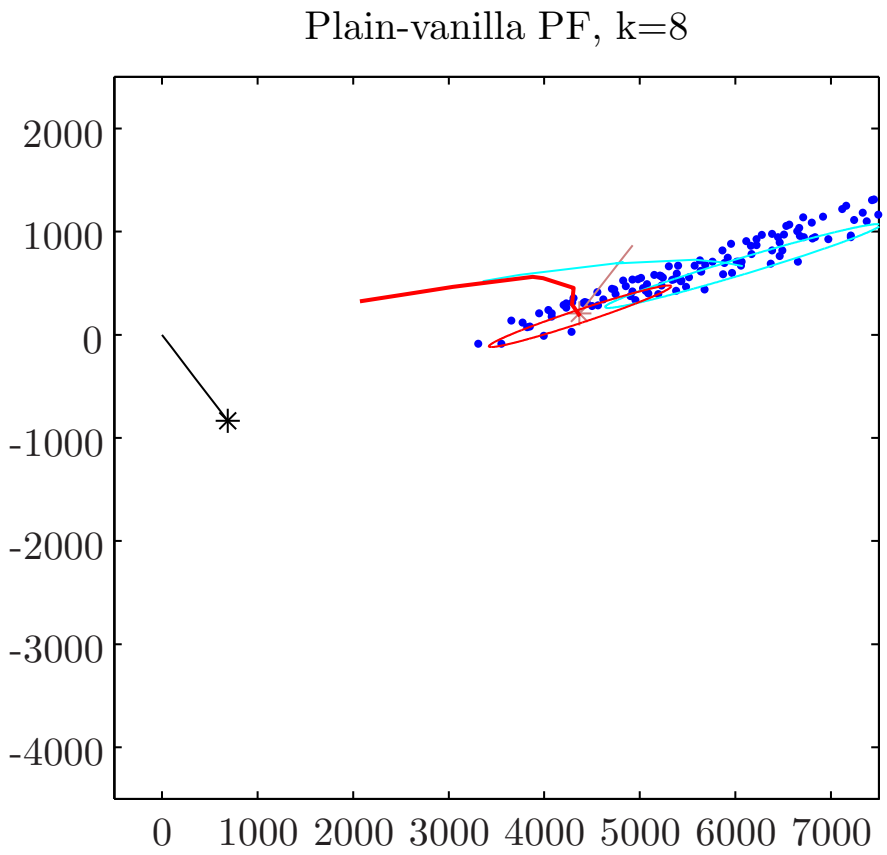
Plain-vanilla vs regularized PF for bearing-only

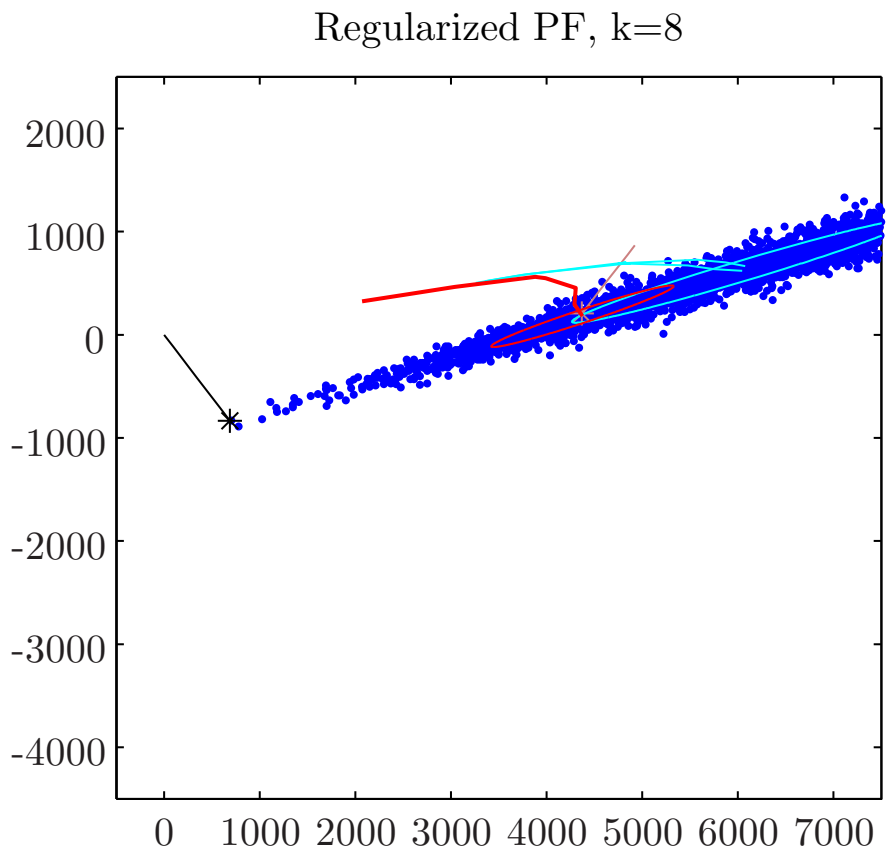




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

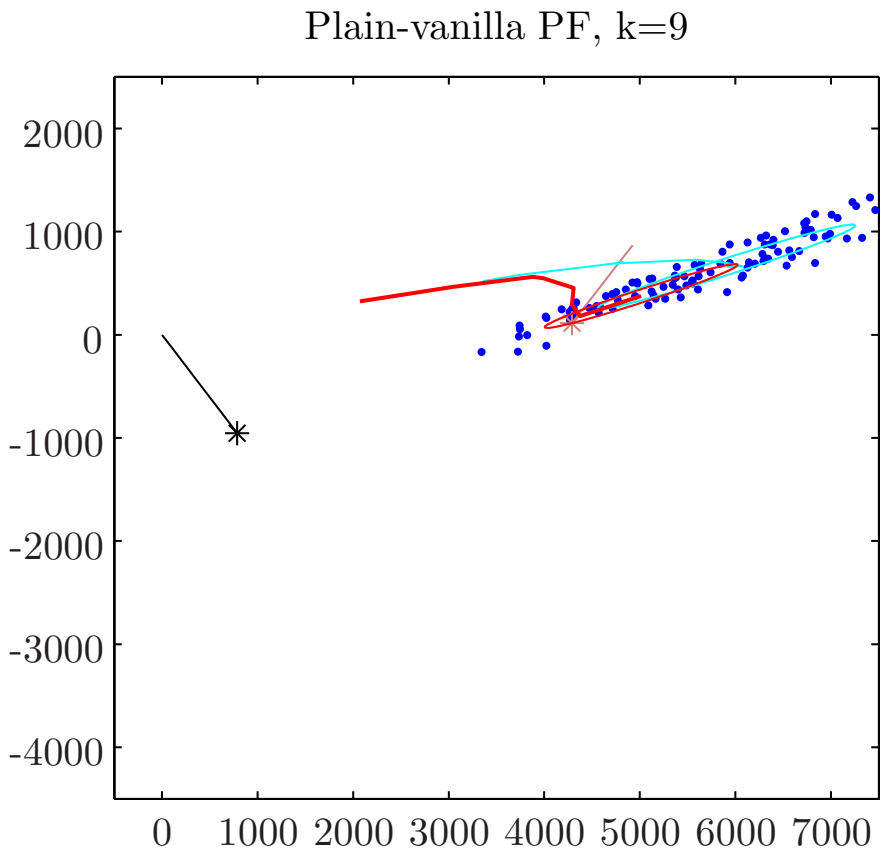
Plain-vanilla vs regularized PF for bearing-only

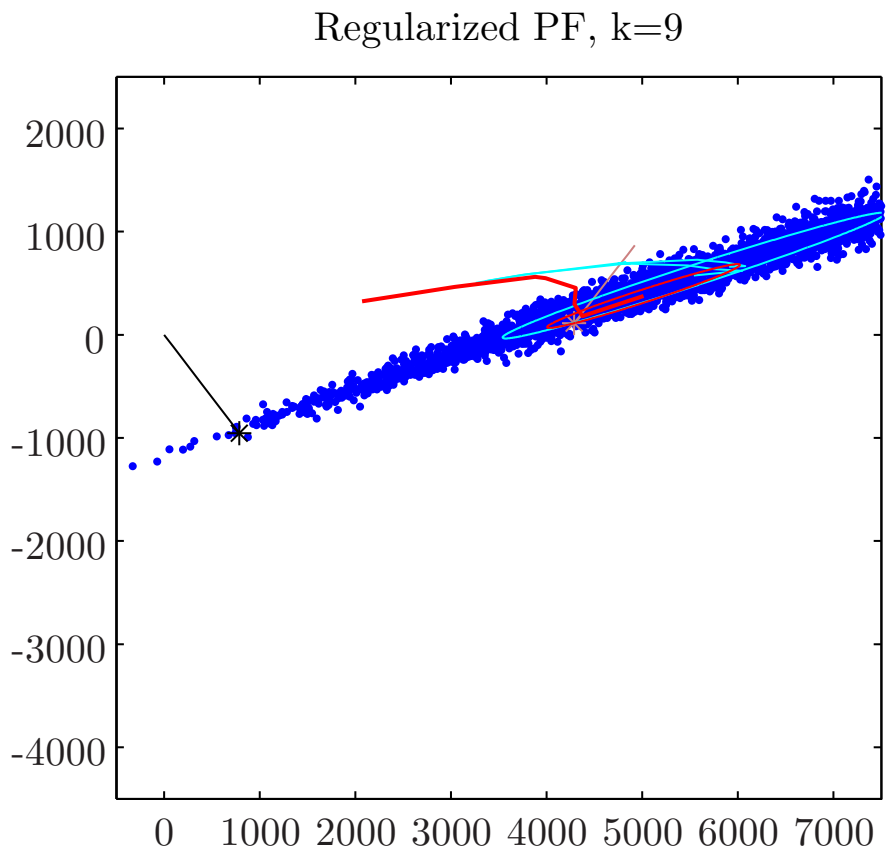




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

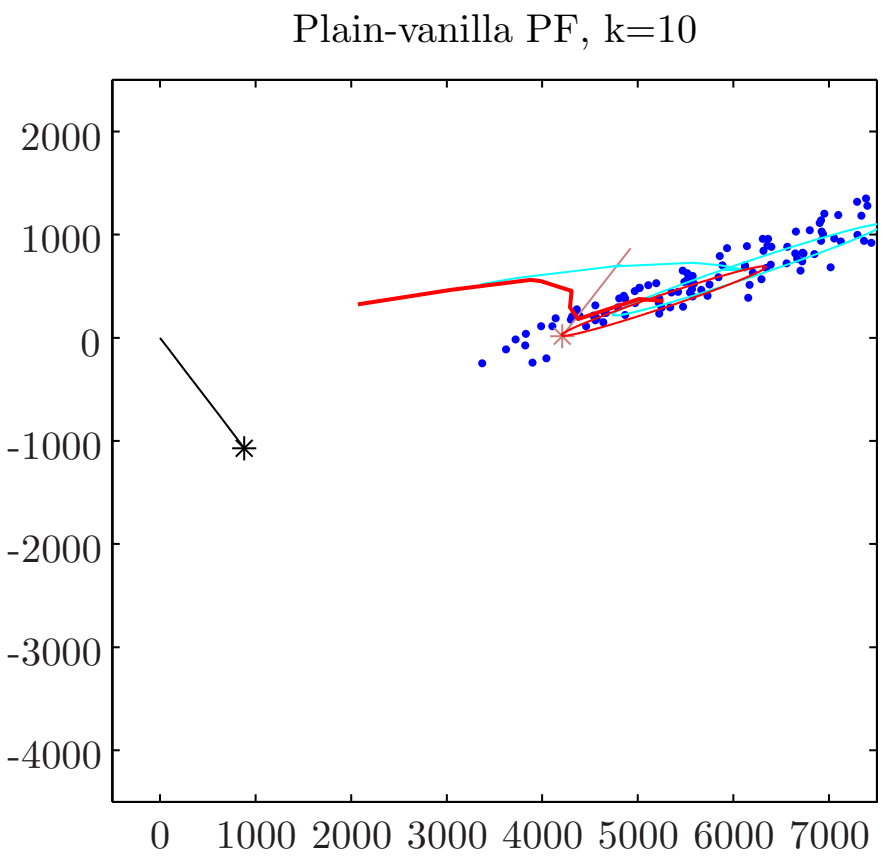
Plain-vanilla vs regularized PF for bearing-only

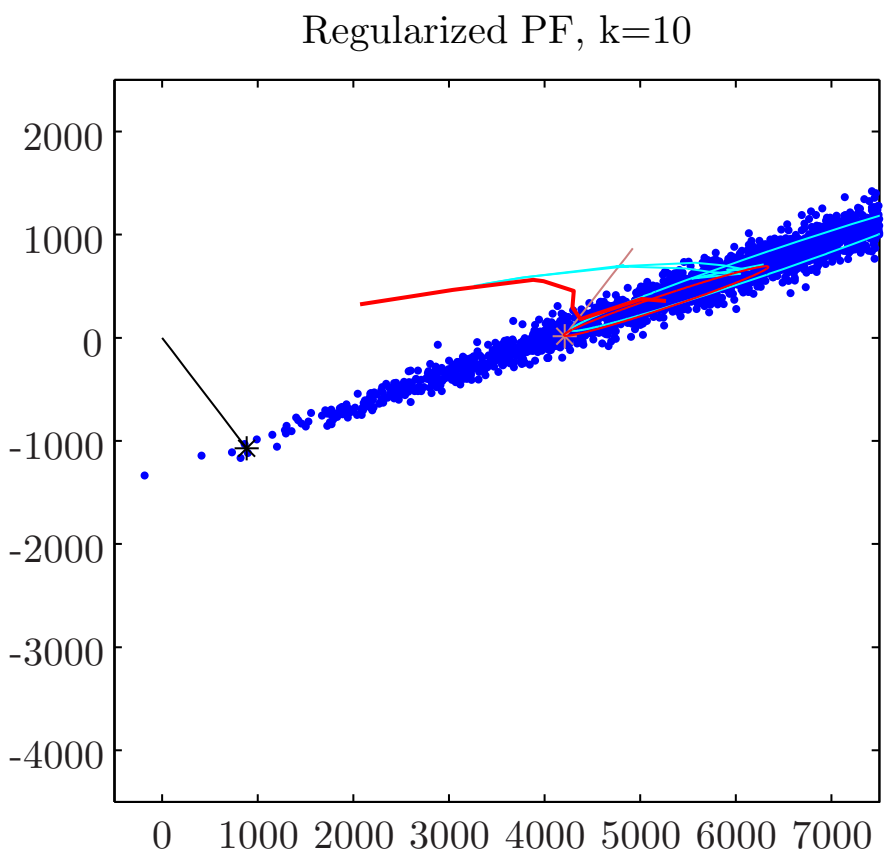




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

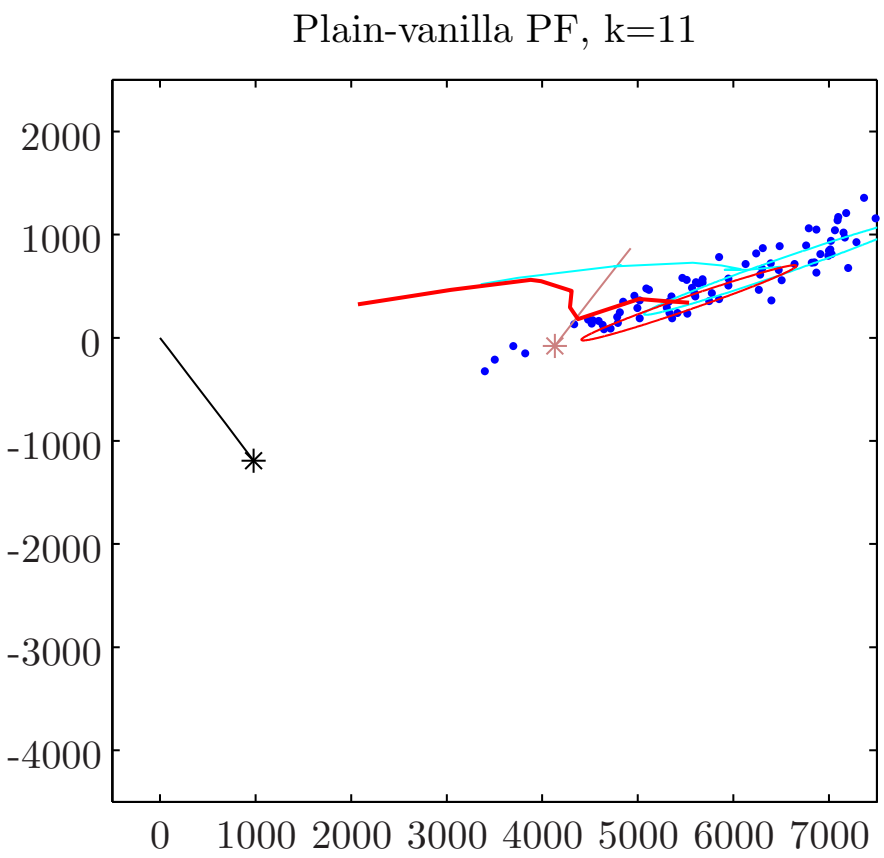
Plain-vanilla vs regularized PF for bearing-only

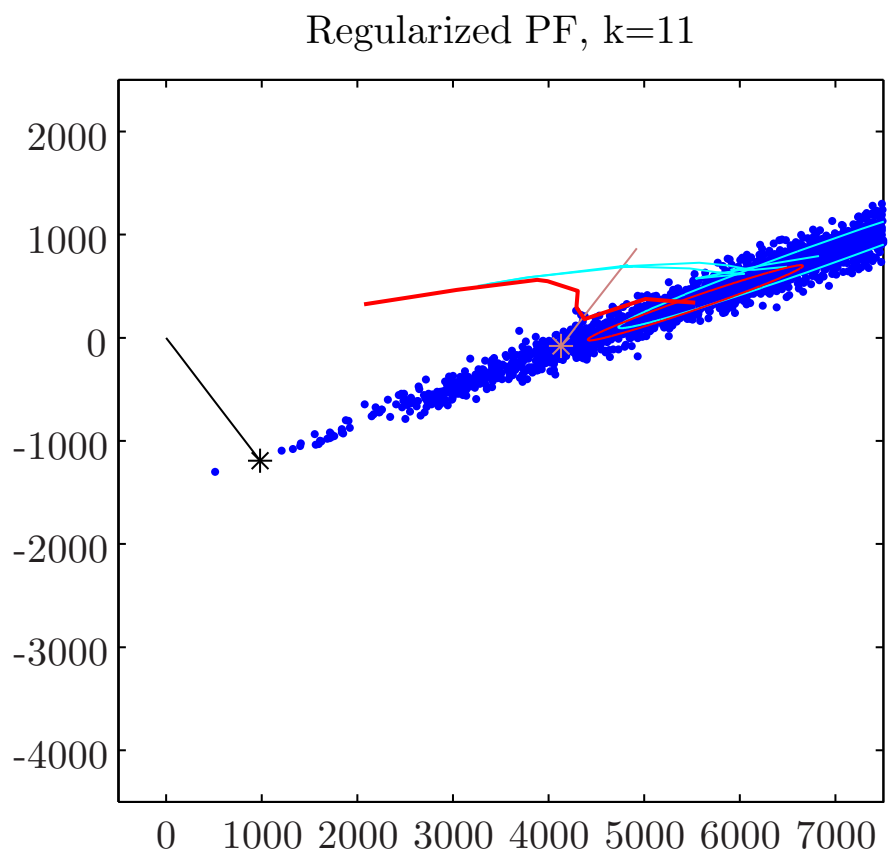




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

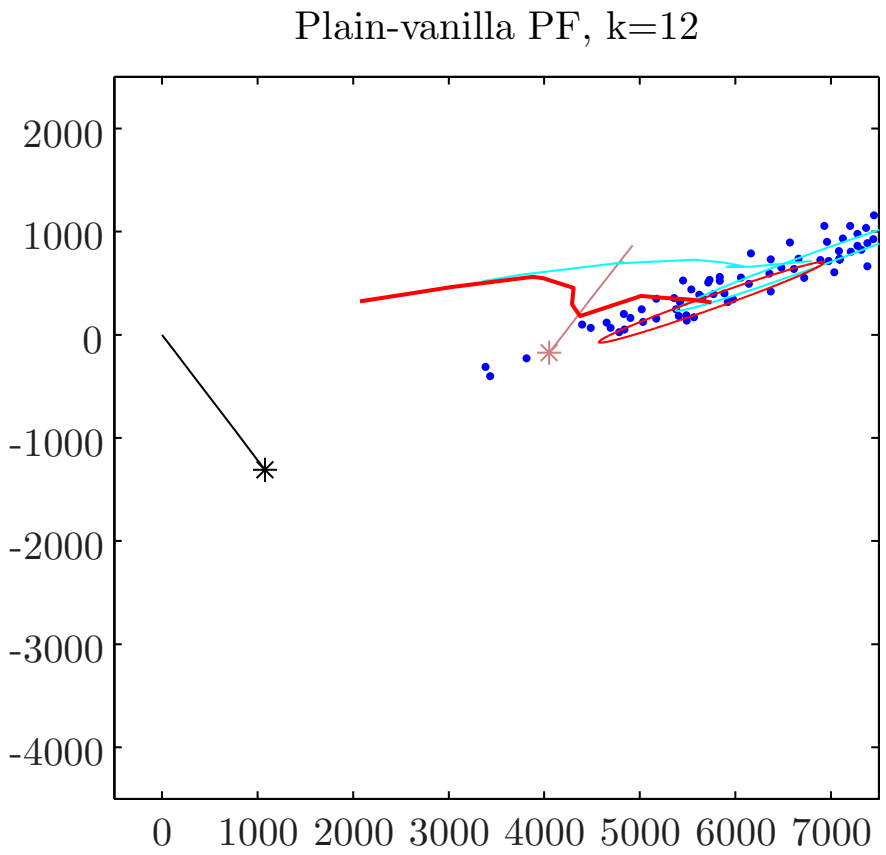
Plain-vanilla vs regularized PF for bearing-only

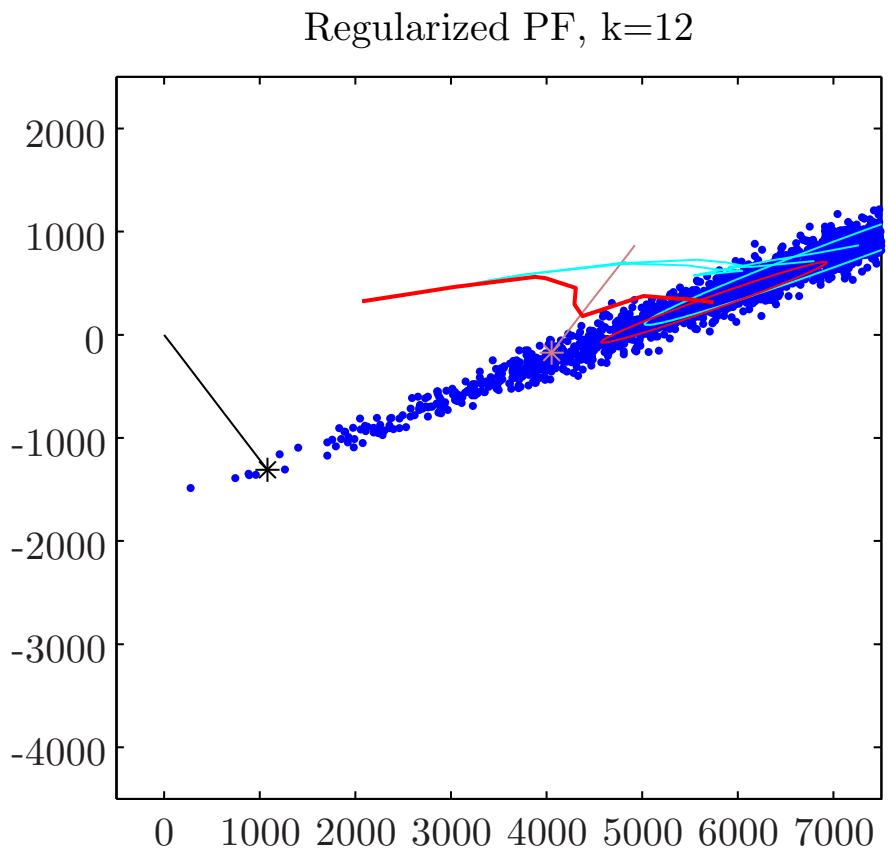




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

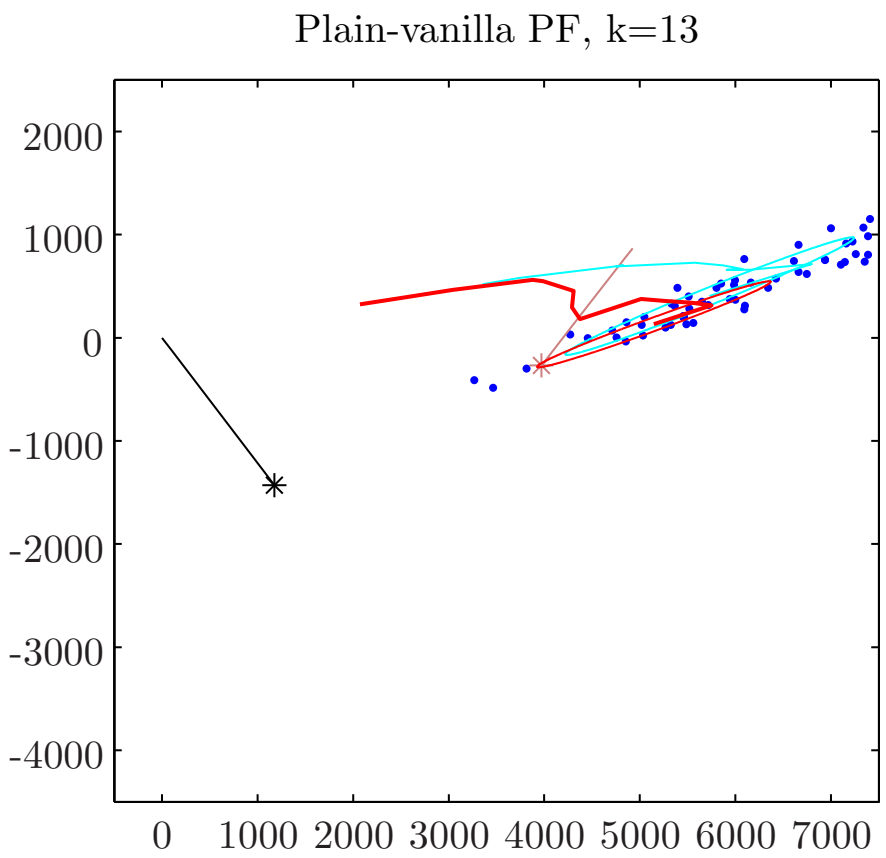
Plain-vanilla vs regularized PF for bearing-only

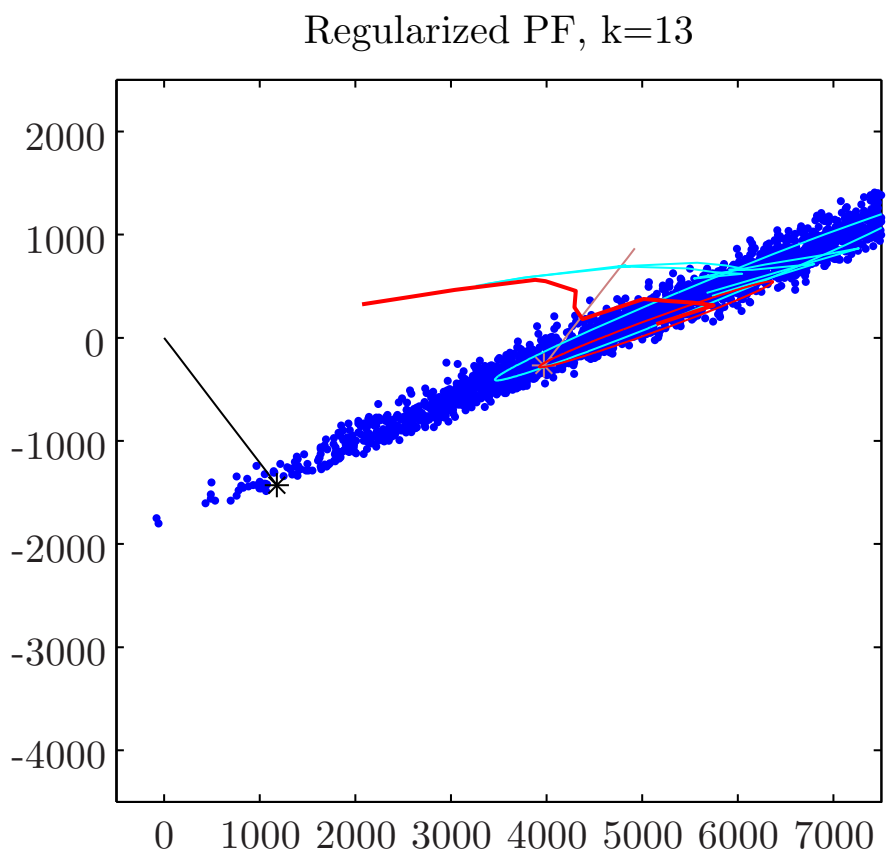




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

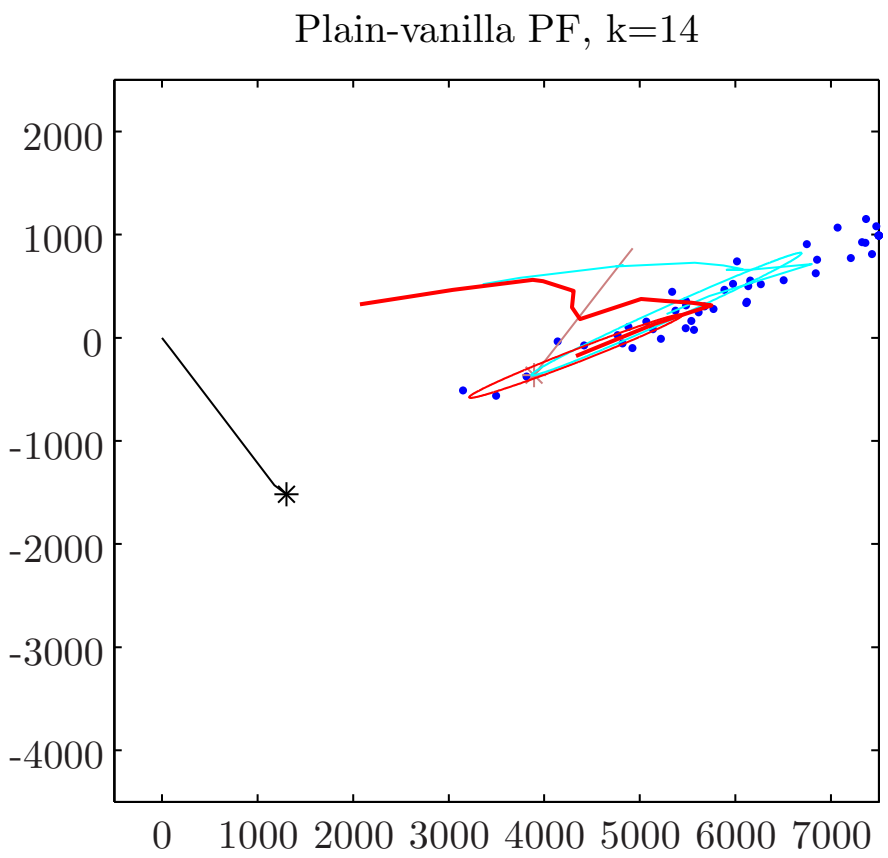
Plain-vanilla vs regularized PF for bearing-only

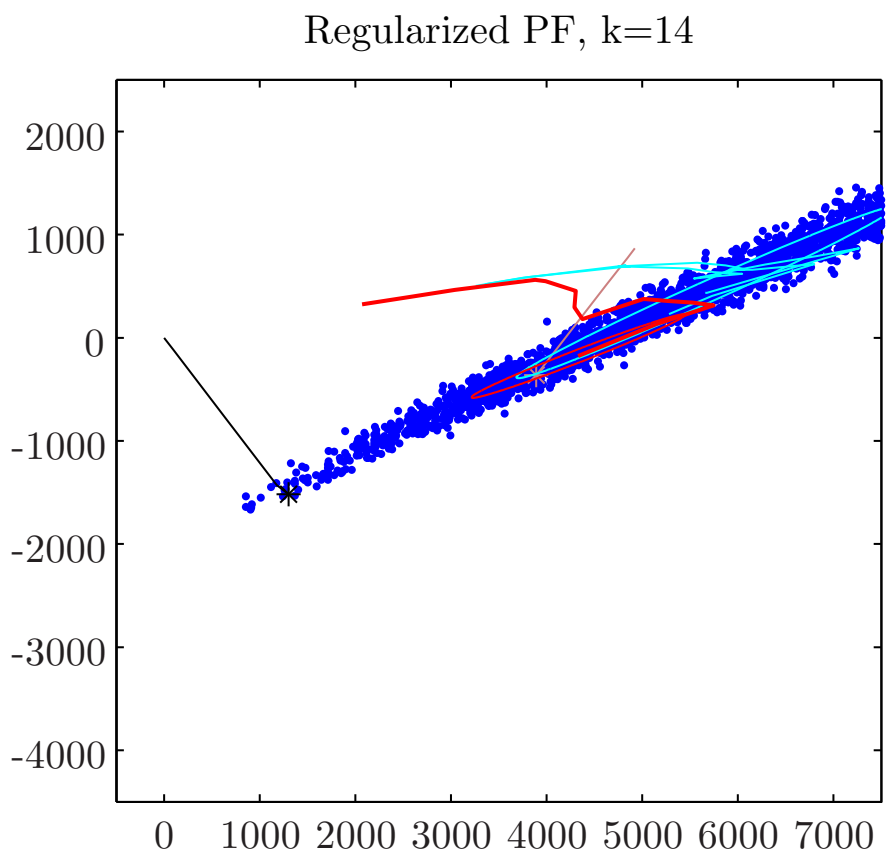




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

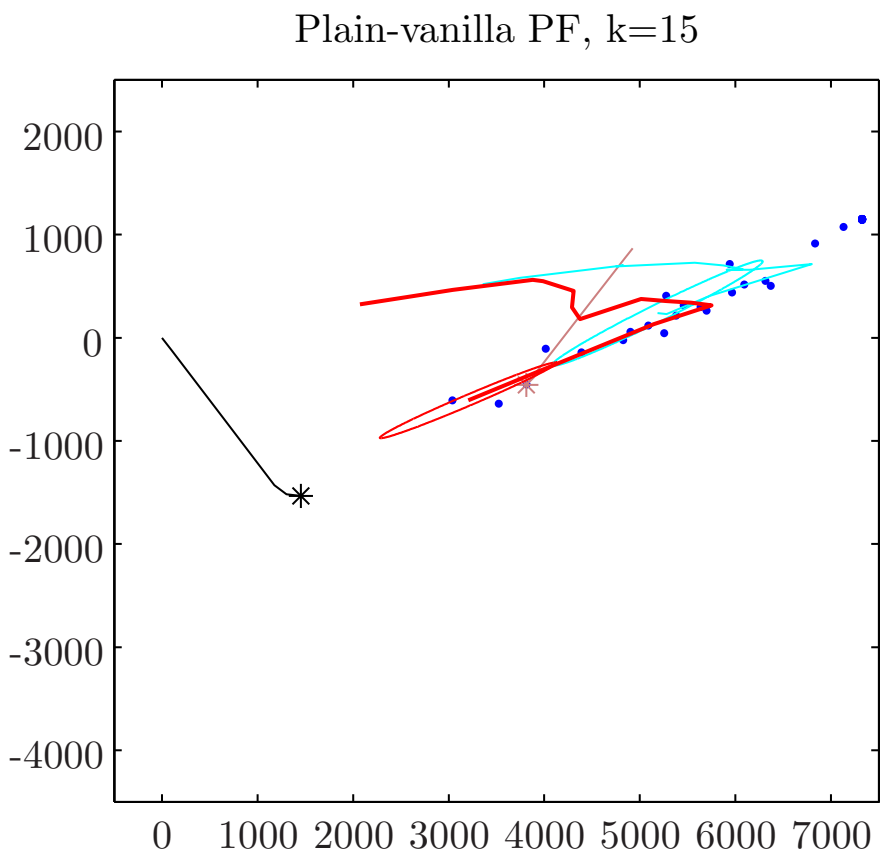
Plain-vanilla vs regularized PF for bearing-only

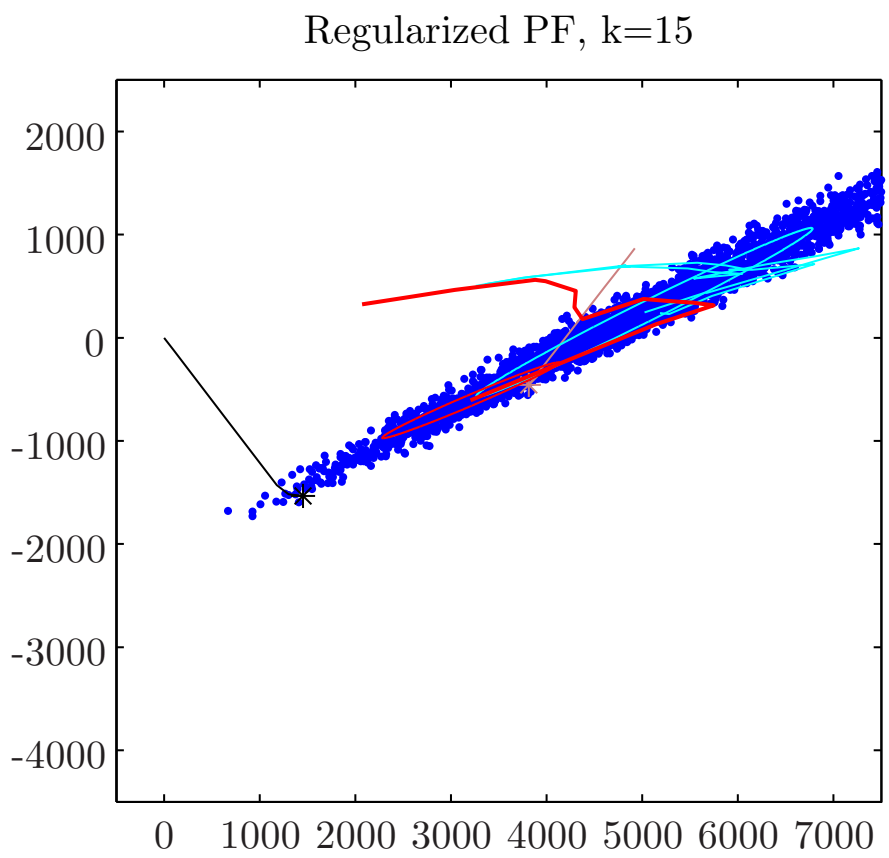




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

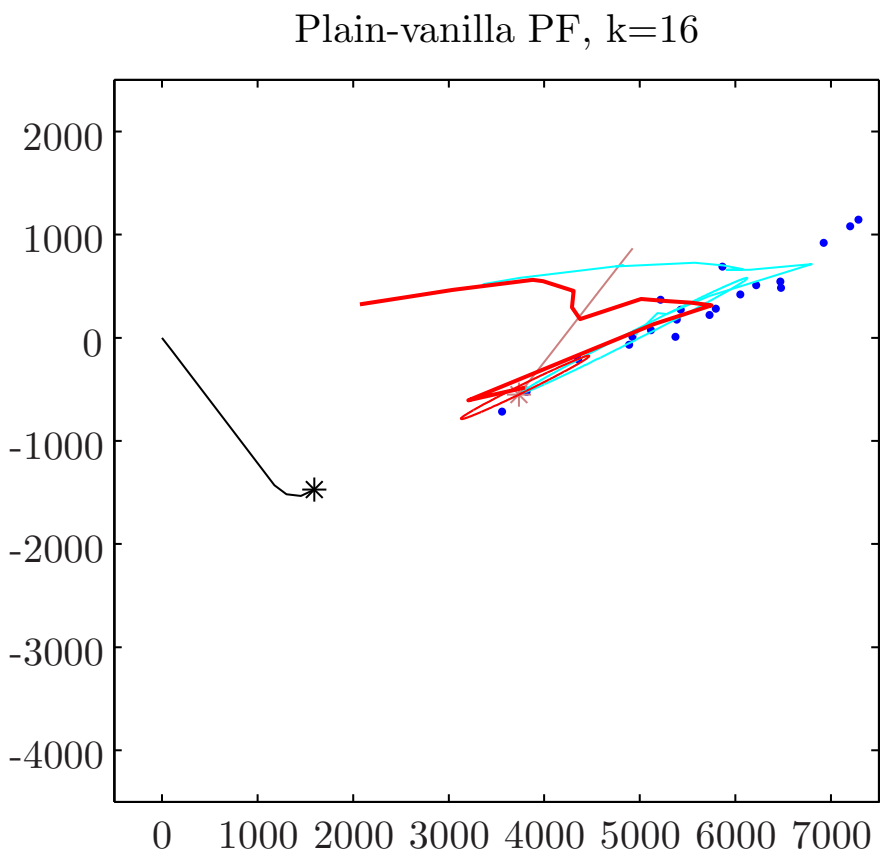
Plain-vanilla vs regularized PF for bearing-only

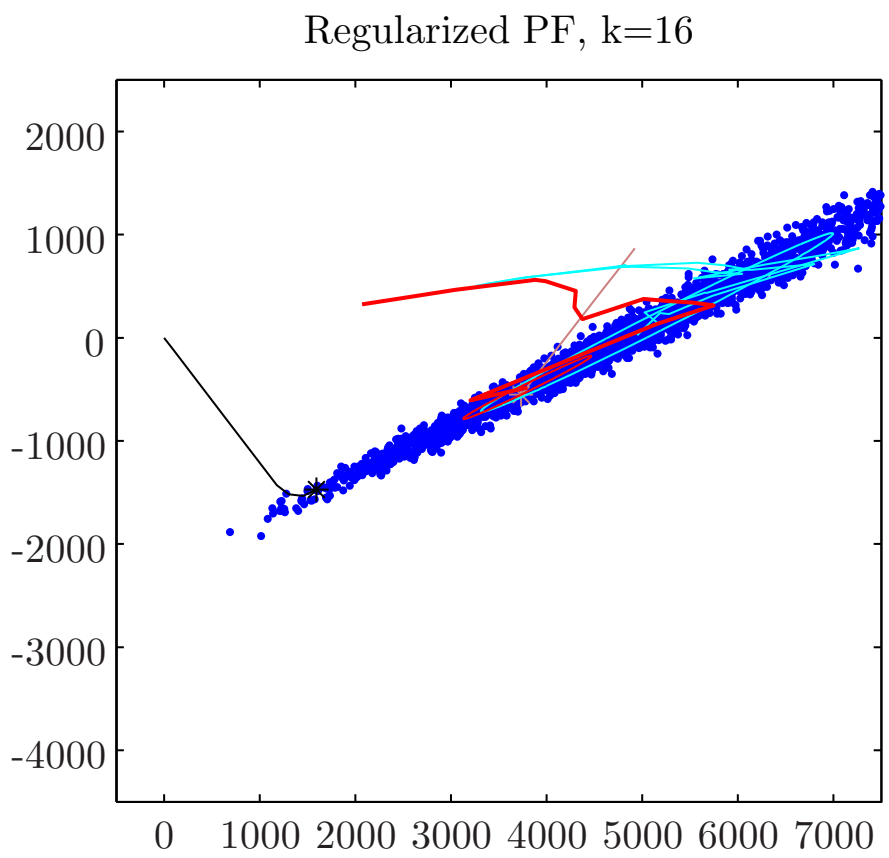




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

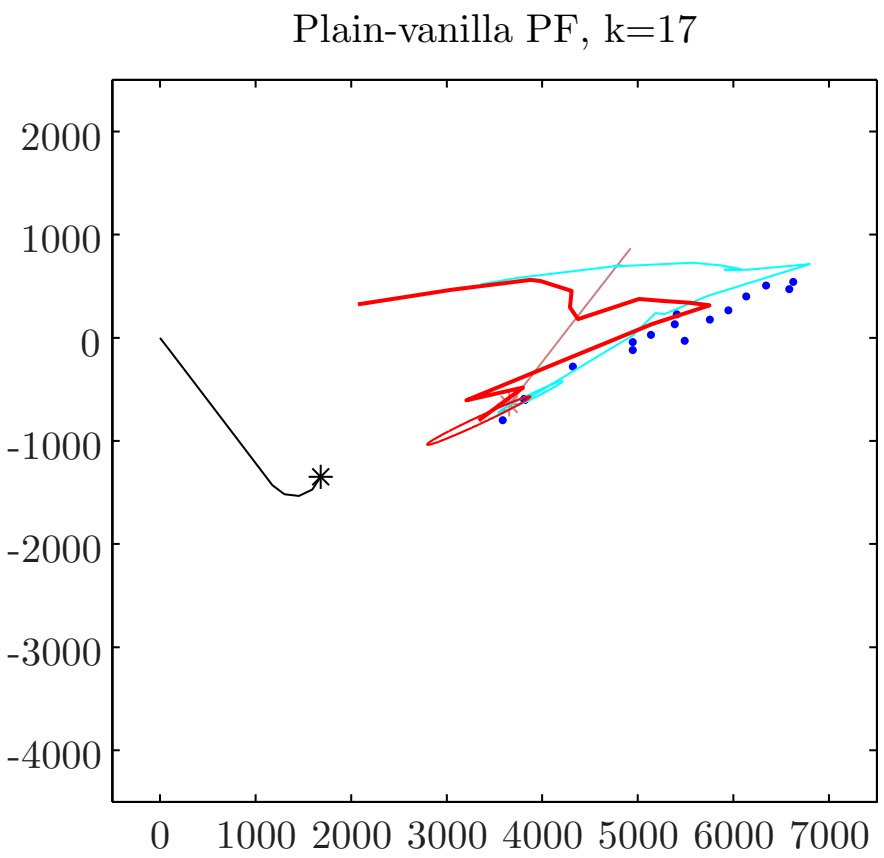
Plain-vanilla vs regularized PF for bearing-only

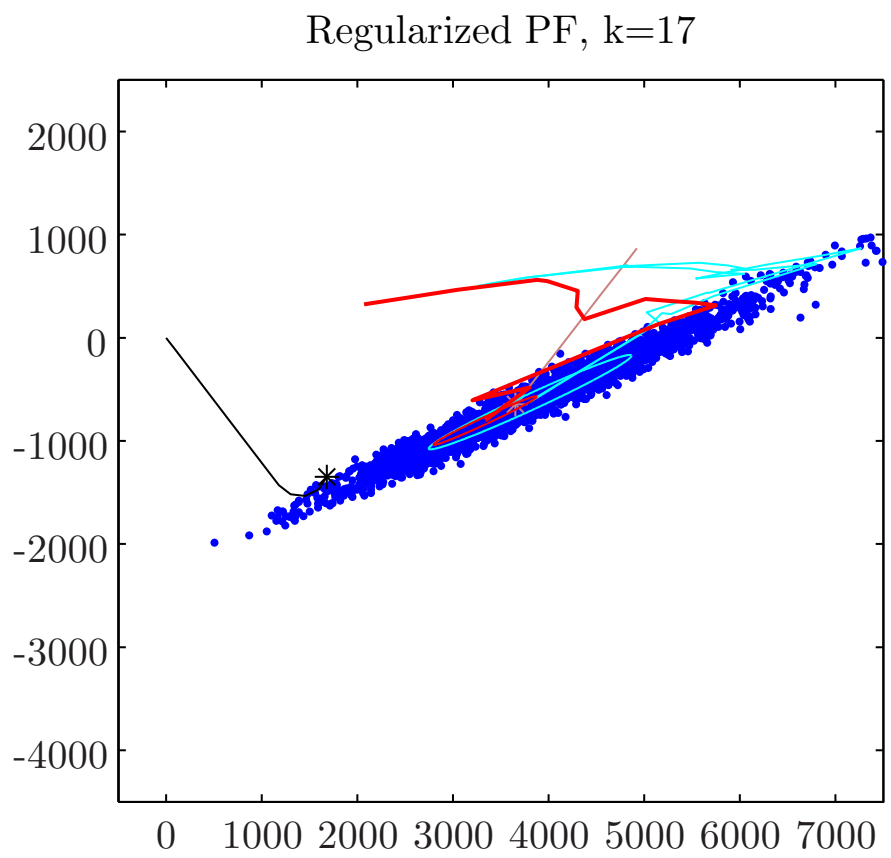




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

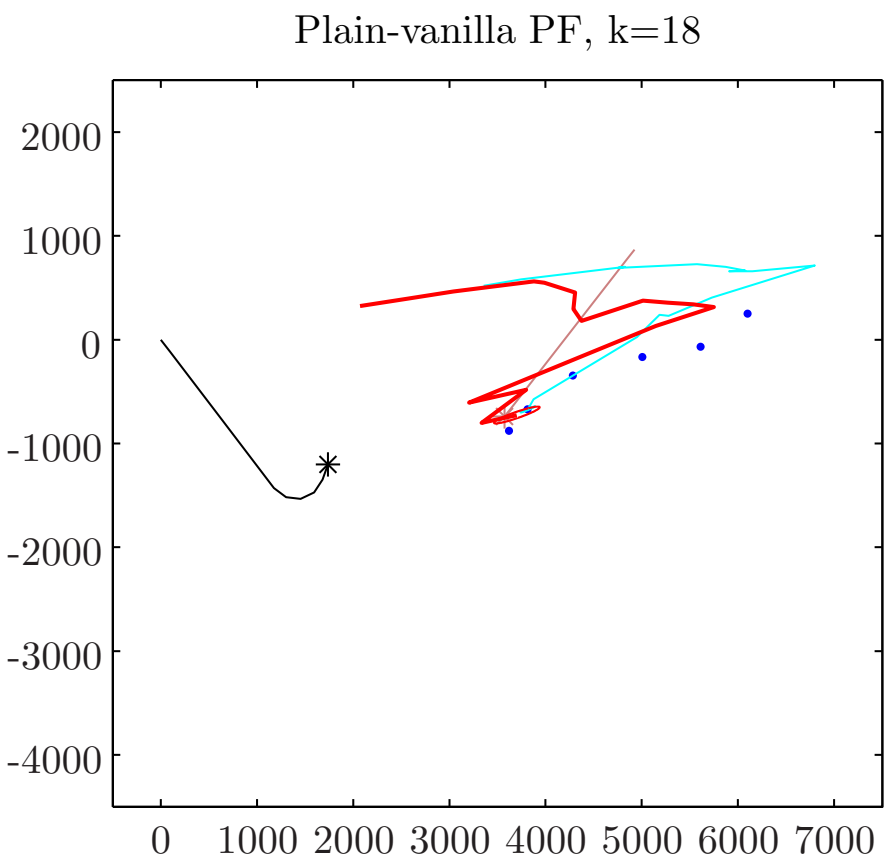
Plain-vanilla vs regularized PF for bearing-only

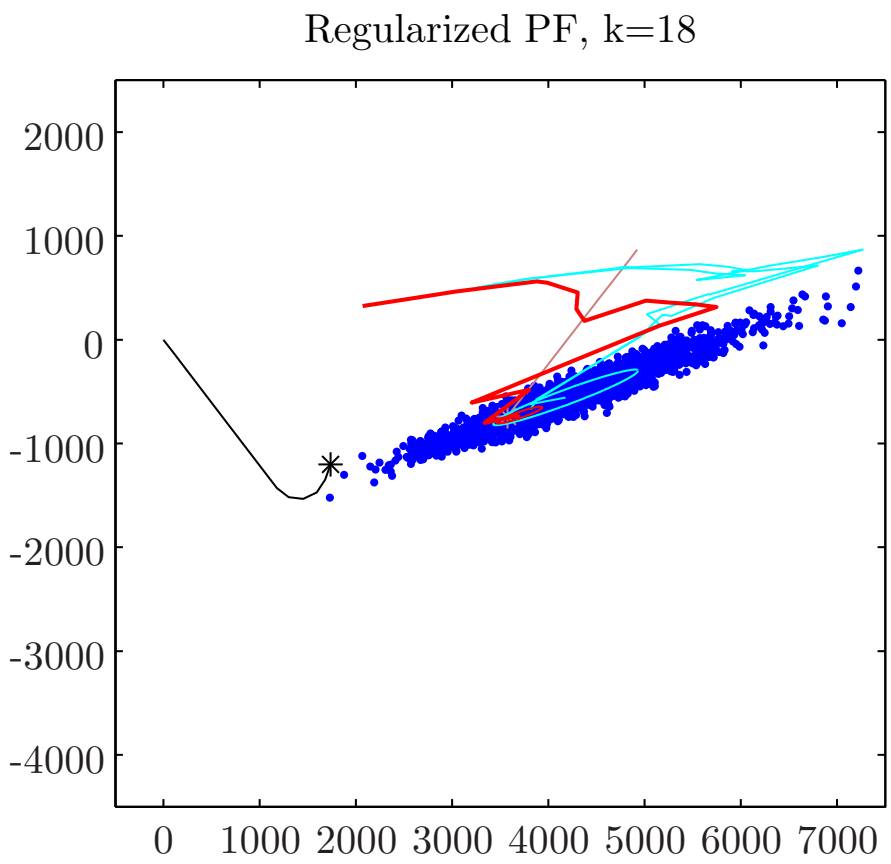




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

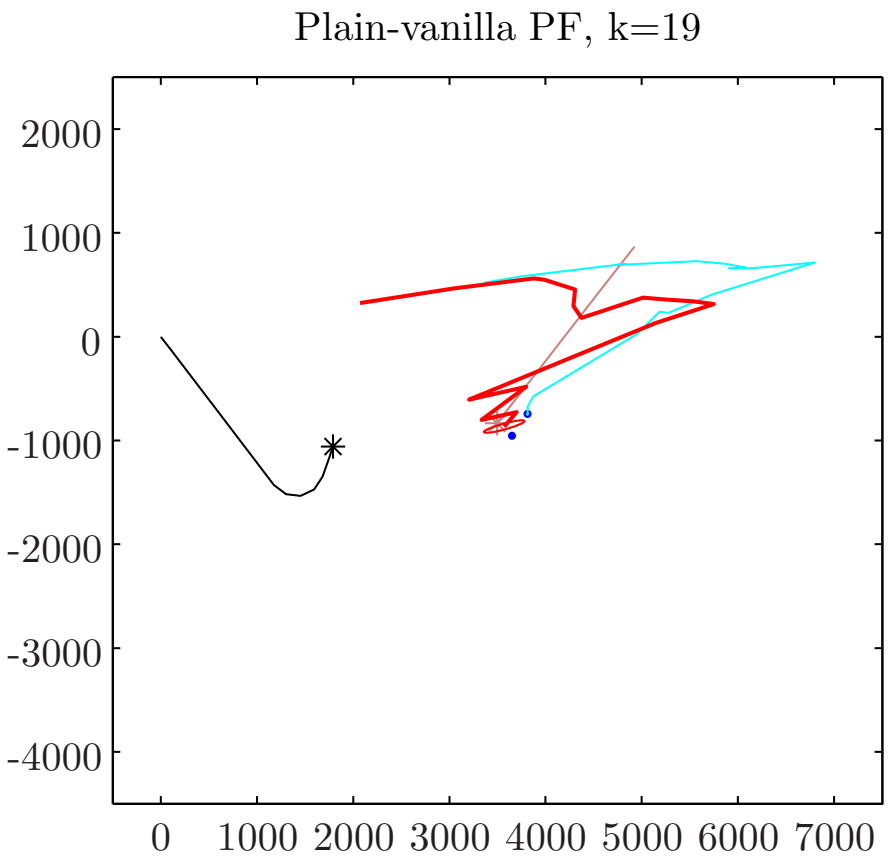
Plain-vanilla vs regularized PF for bearing-only

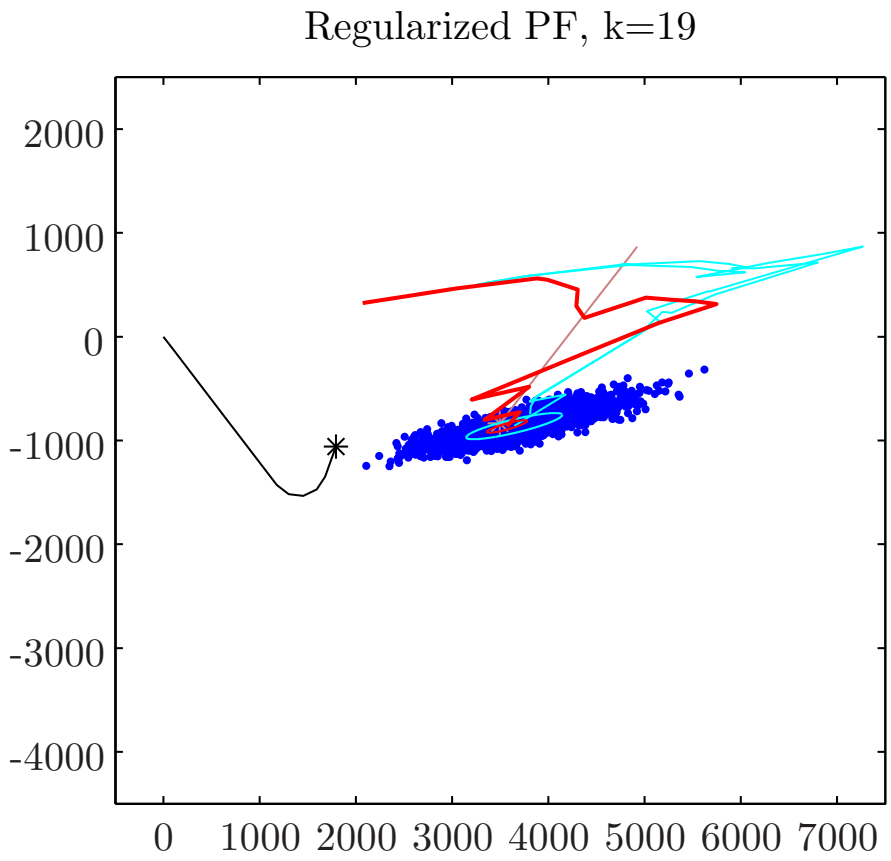




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

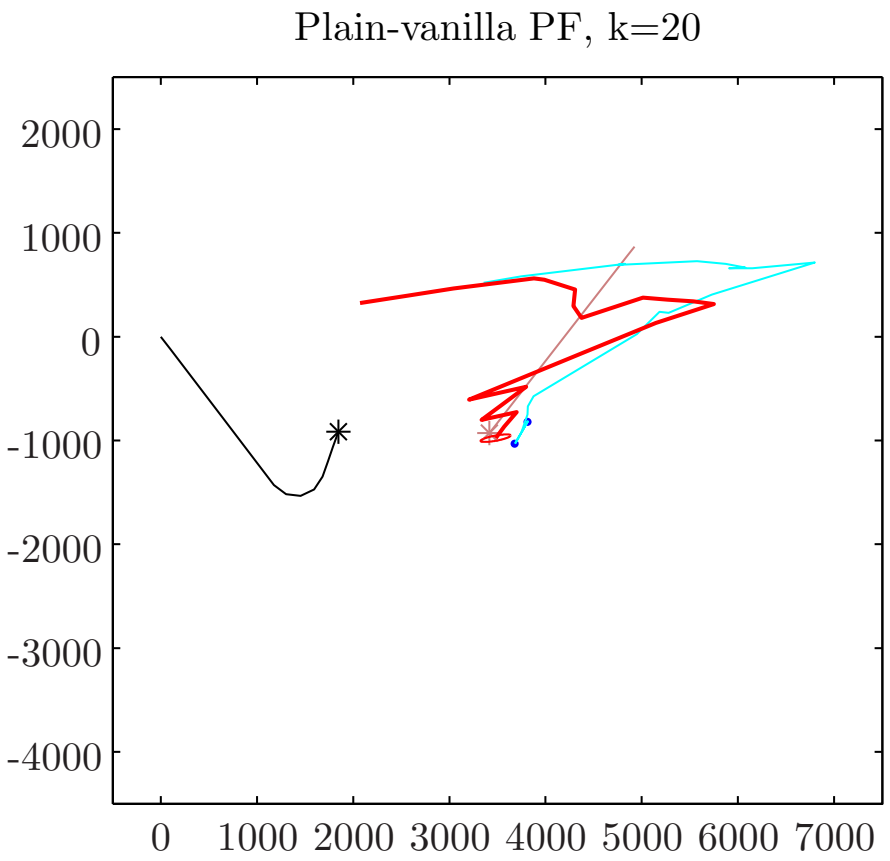
Plain-vanilla vs regularized PF for bearing-only

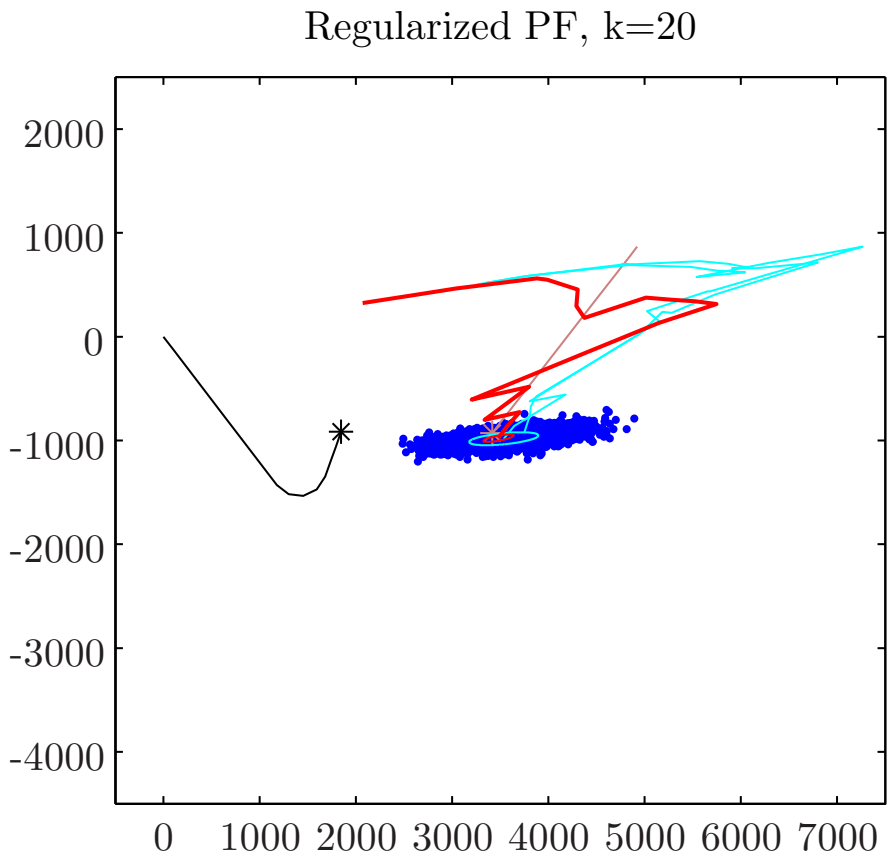




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

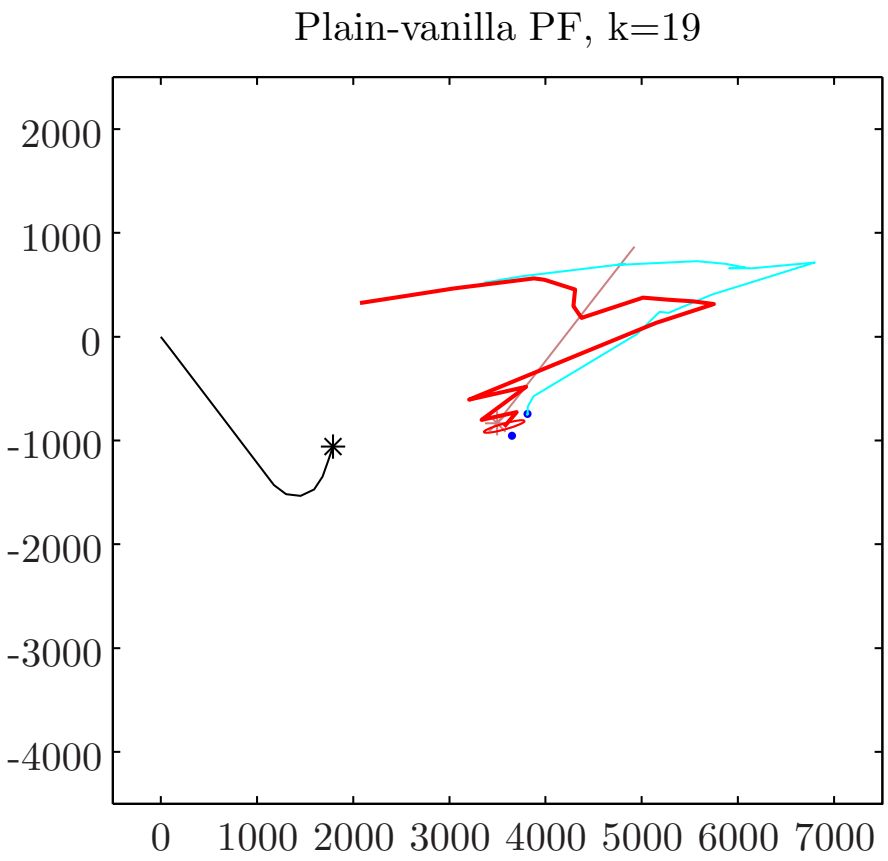
Plain-vanilla vs regularized PF for bearing-only

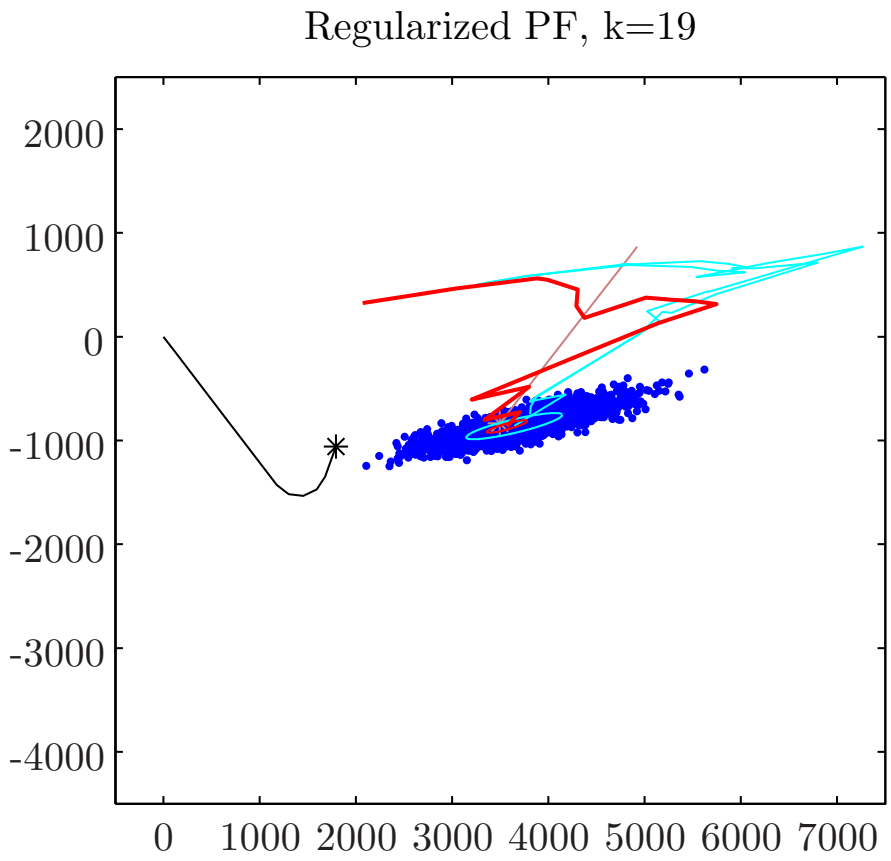




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

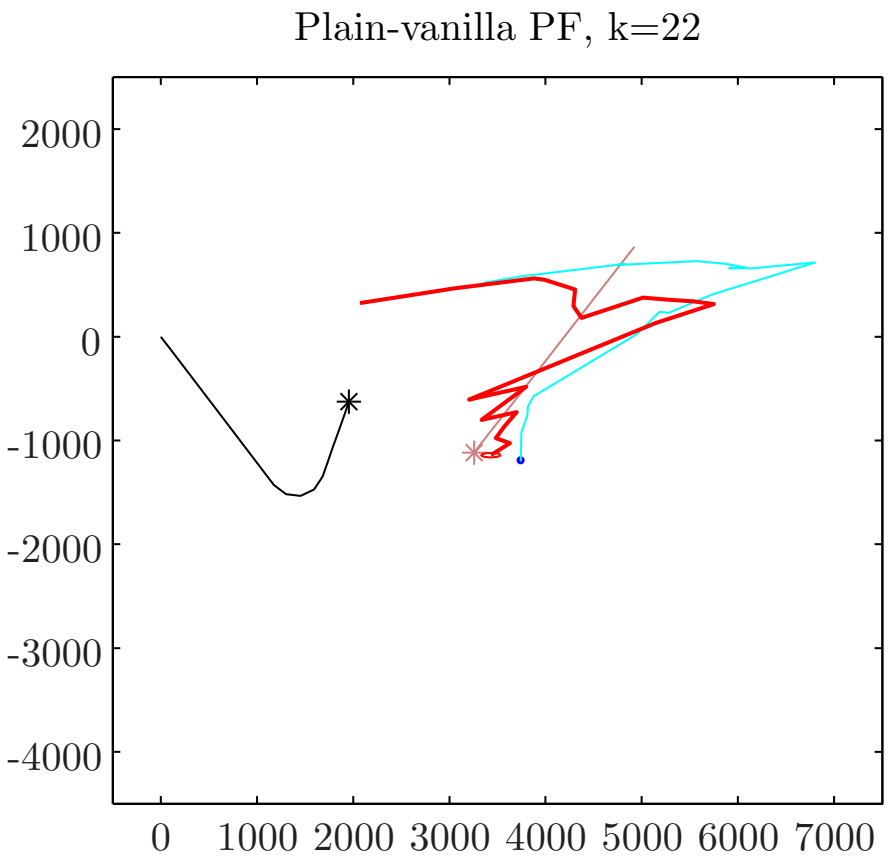
Plain-vanilla vs regularized PF for bearing-only

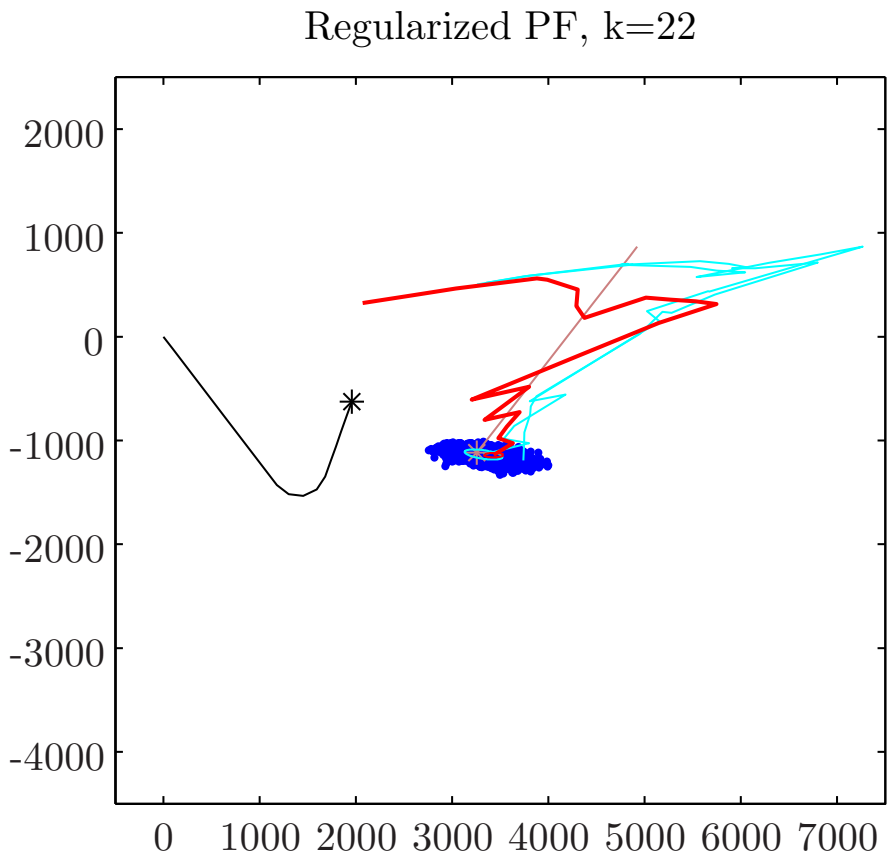




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

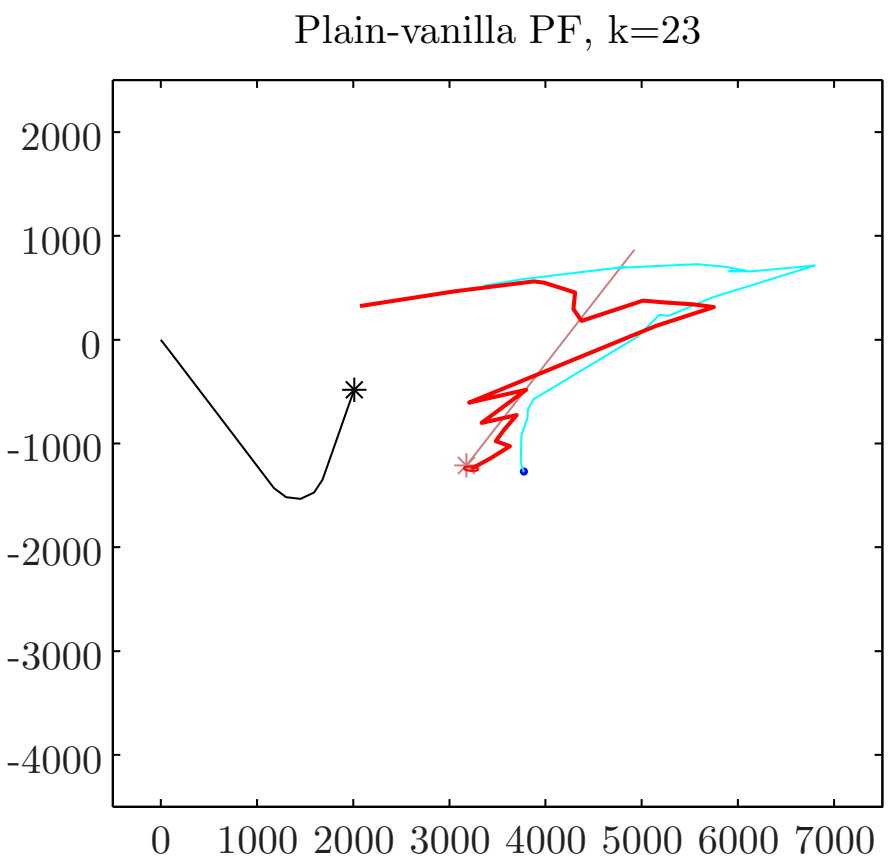
Plain-vanilla vs regularized PF for bearing-only

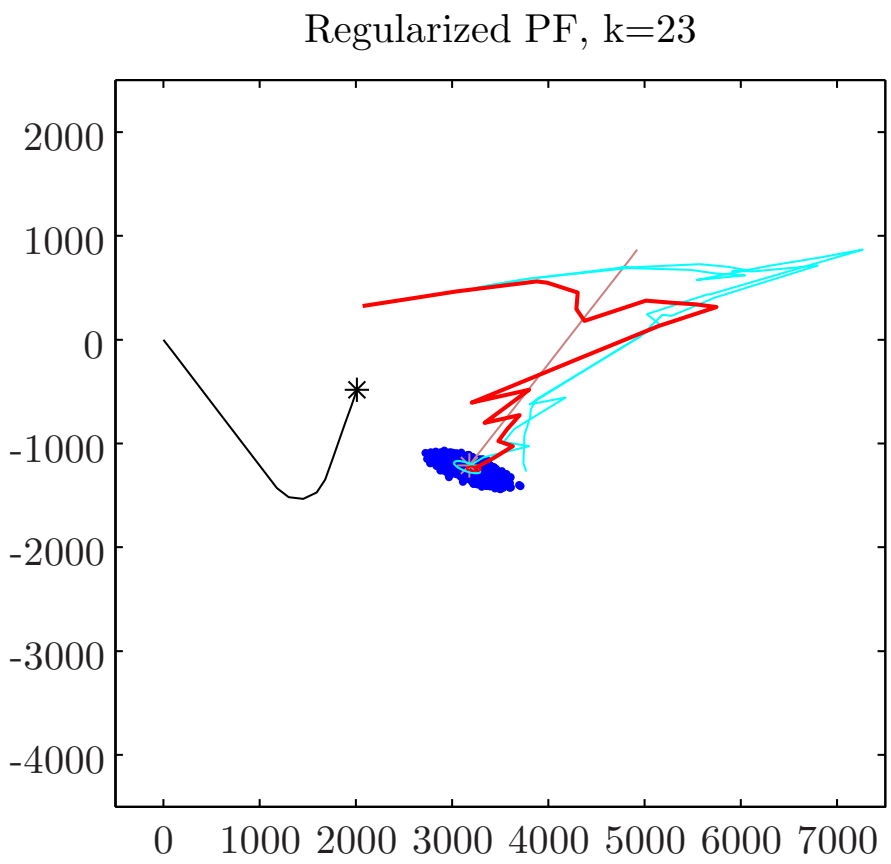




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

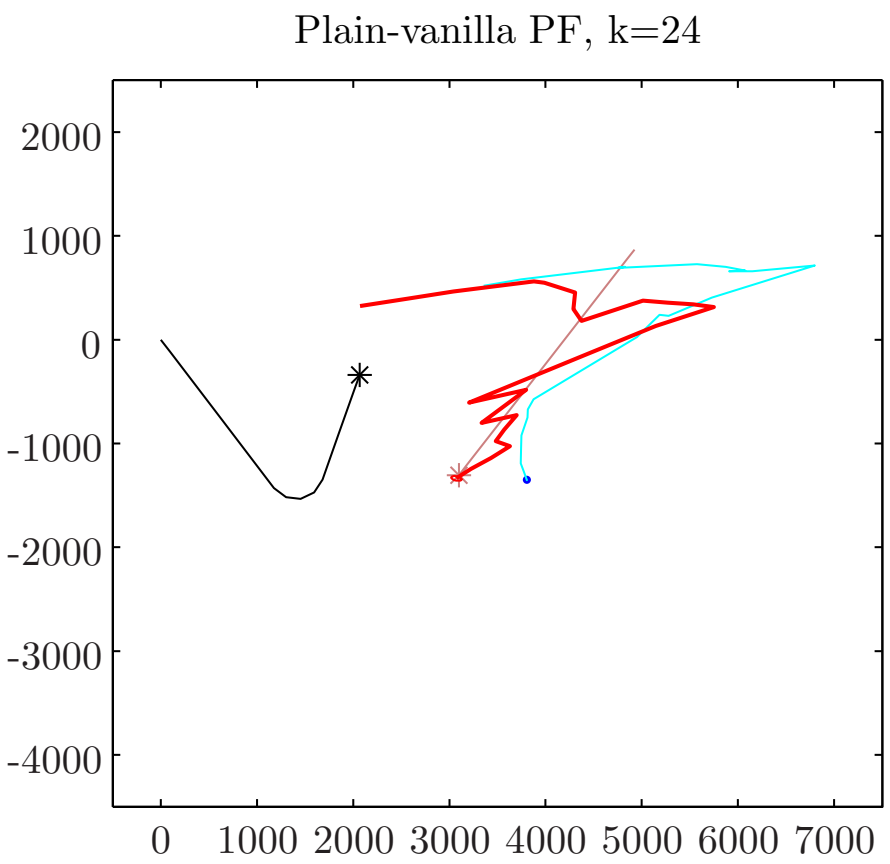
Plain-vanilla vs regularized PF for bearing-only

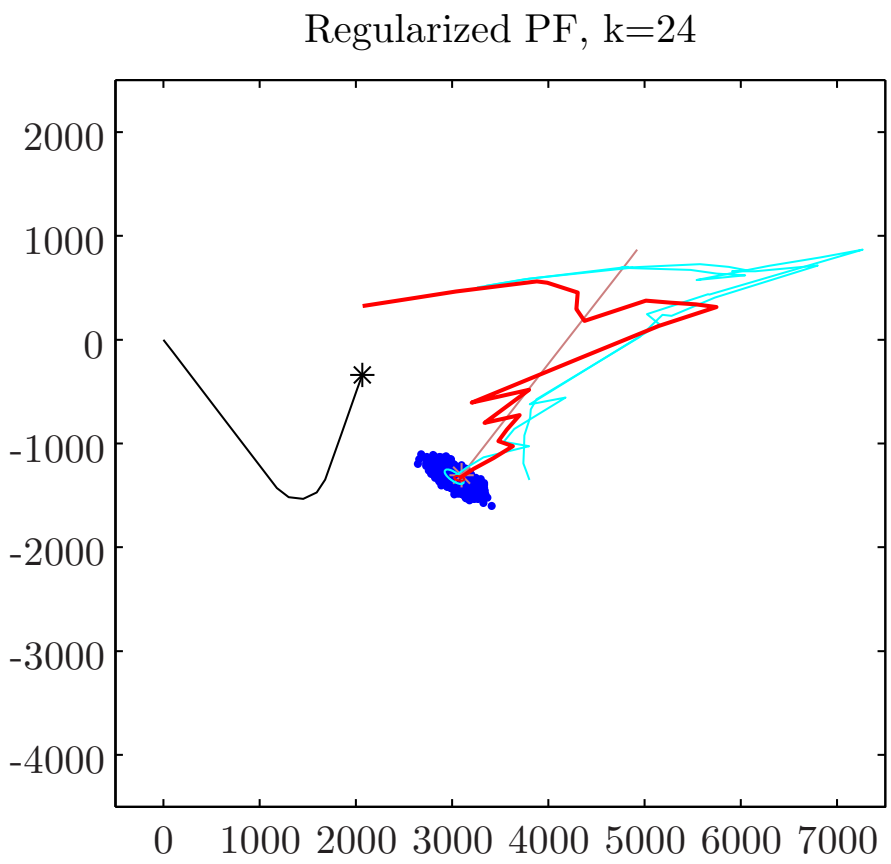




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

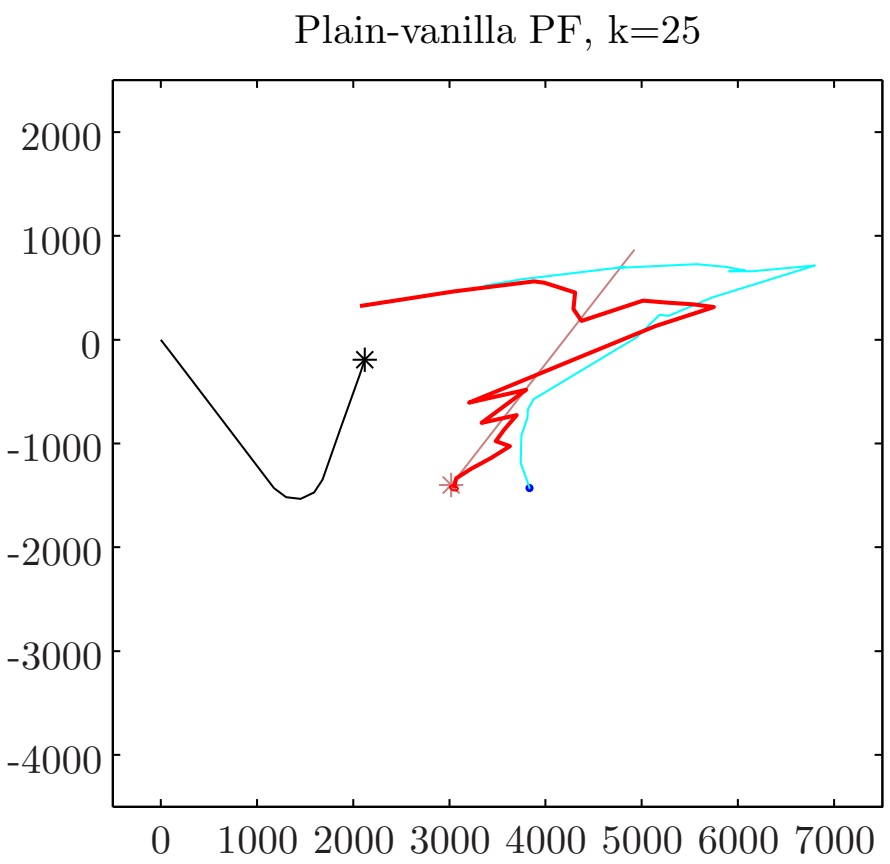
Plain-vanilla vs regularized PF for bearing-only

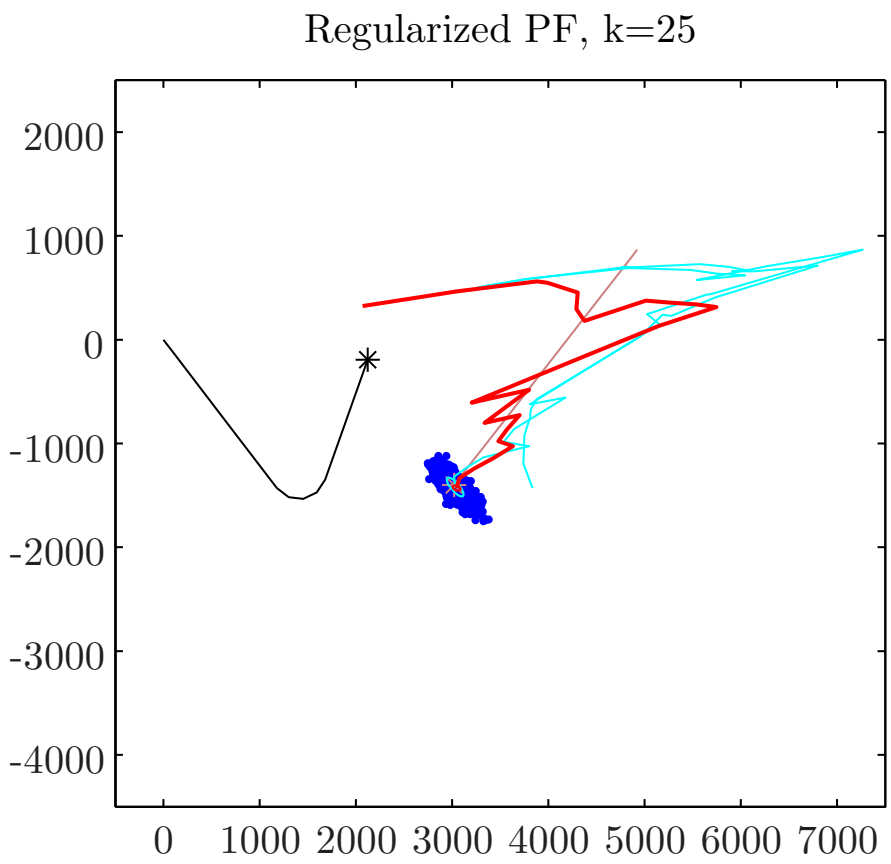




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

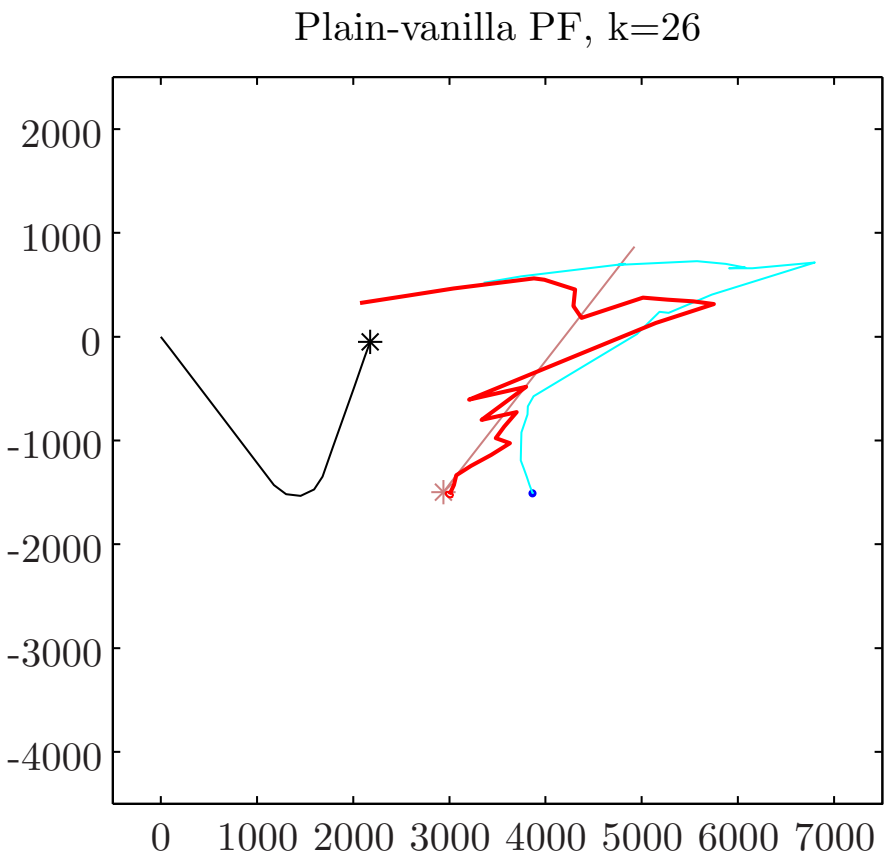
Plain-vanilla vs regularized PF for bearing-only

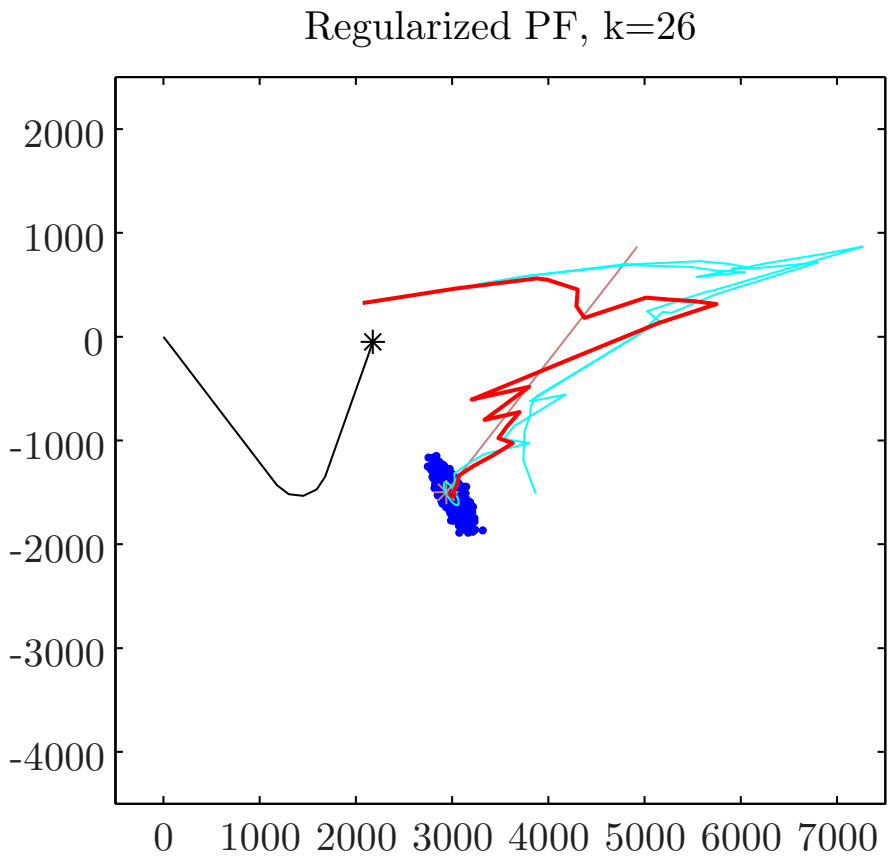




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

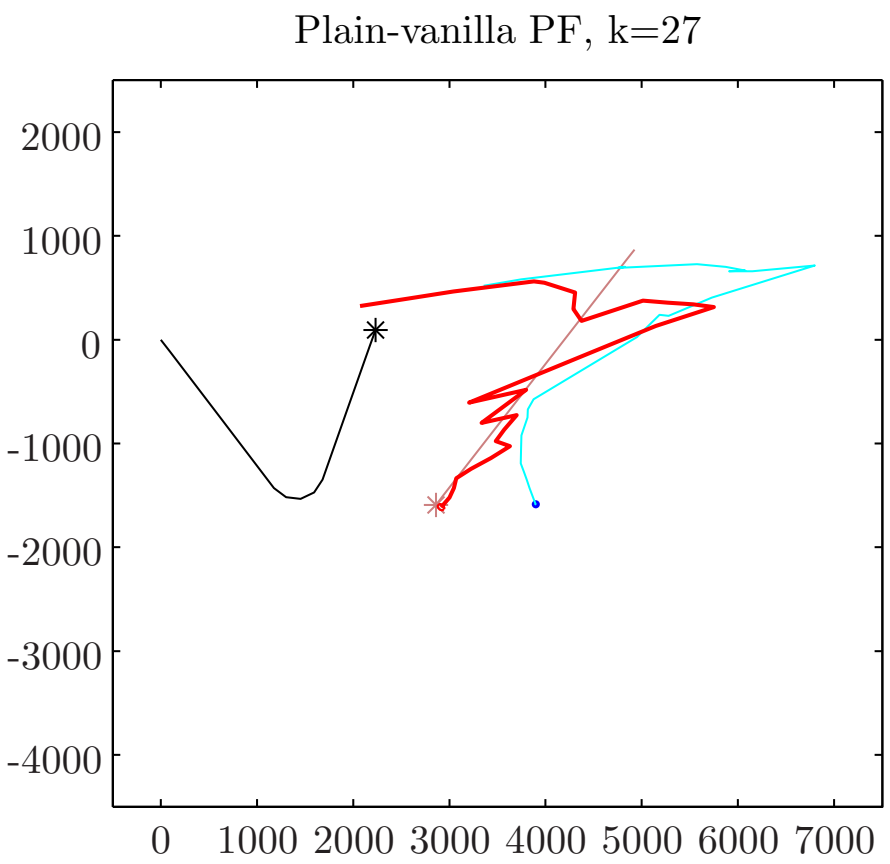
Plain-vanilla vs regularized PF for bearing-only

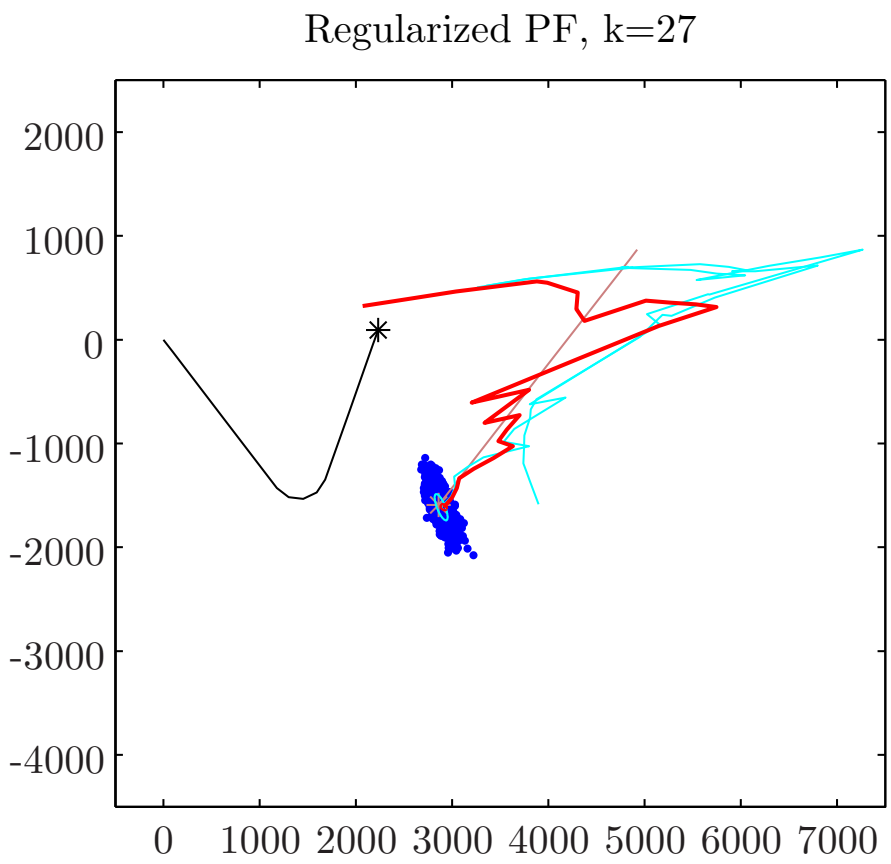




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

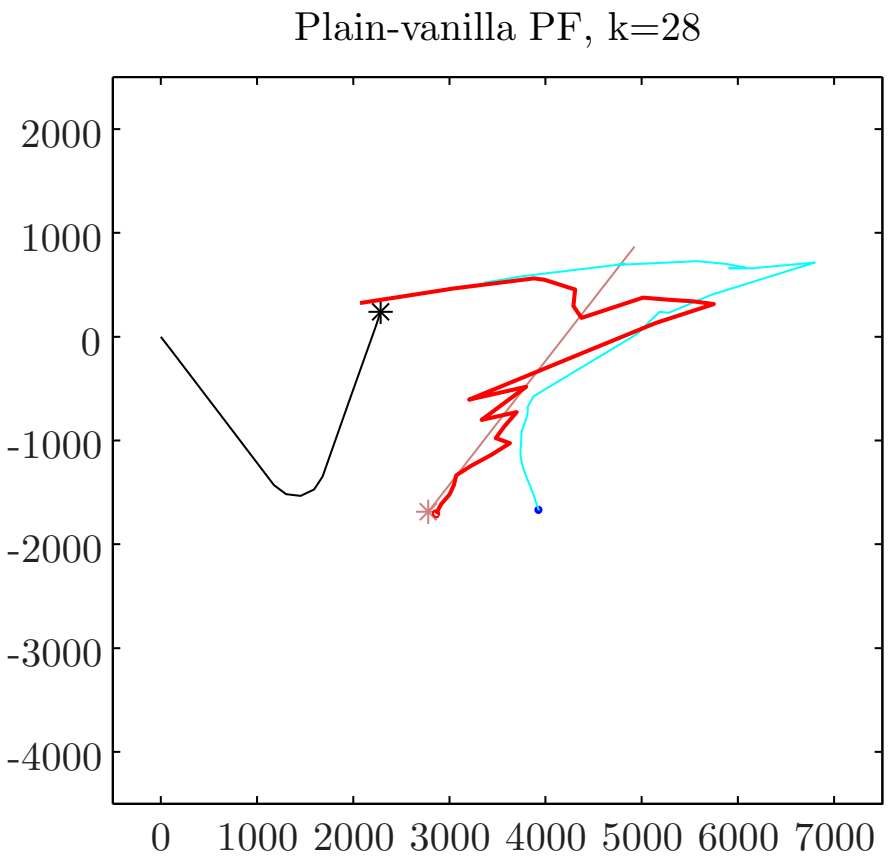
Plain-vanilla vs regularized PF for bearing-only

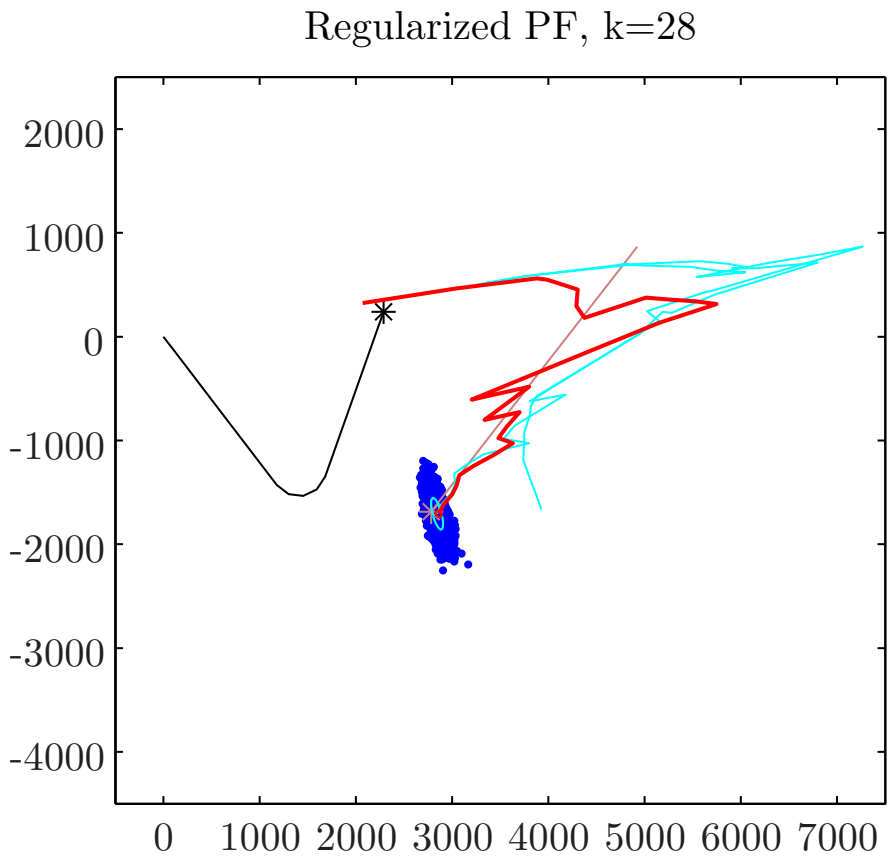




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

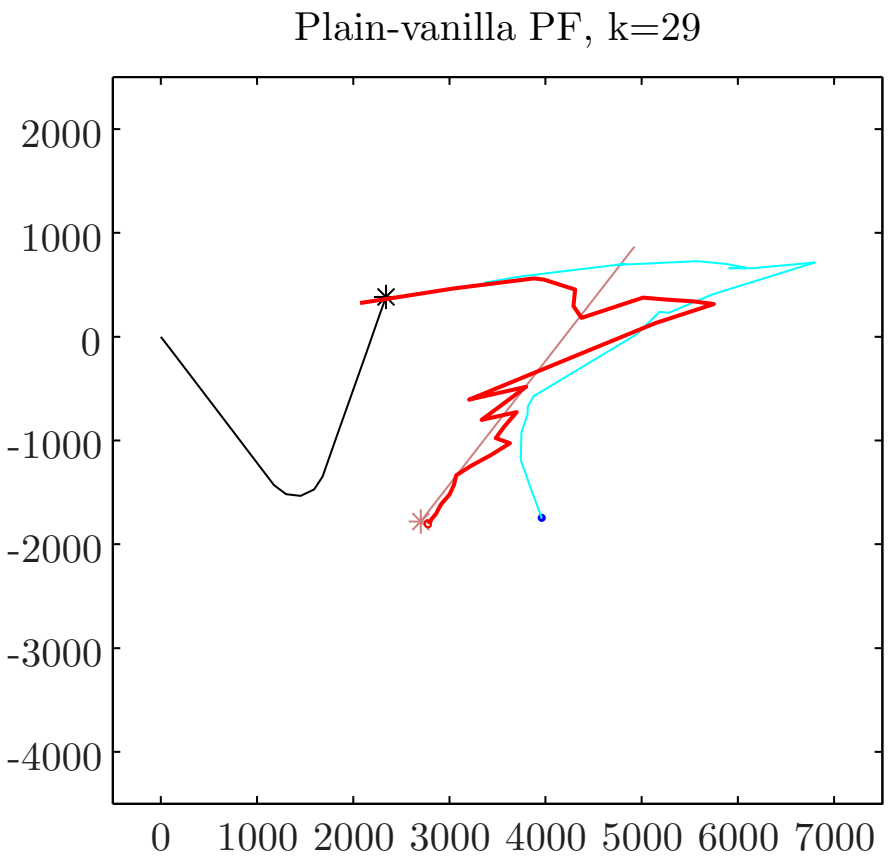
Plain-vanilla vs regularized PF for bearing-only

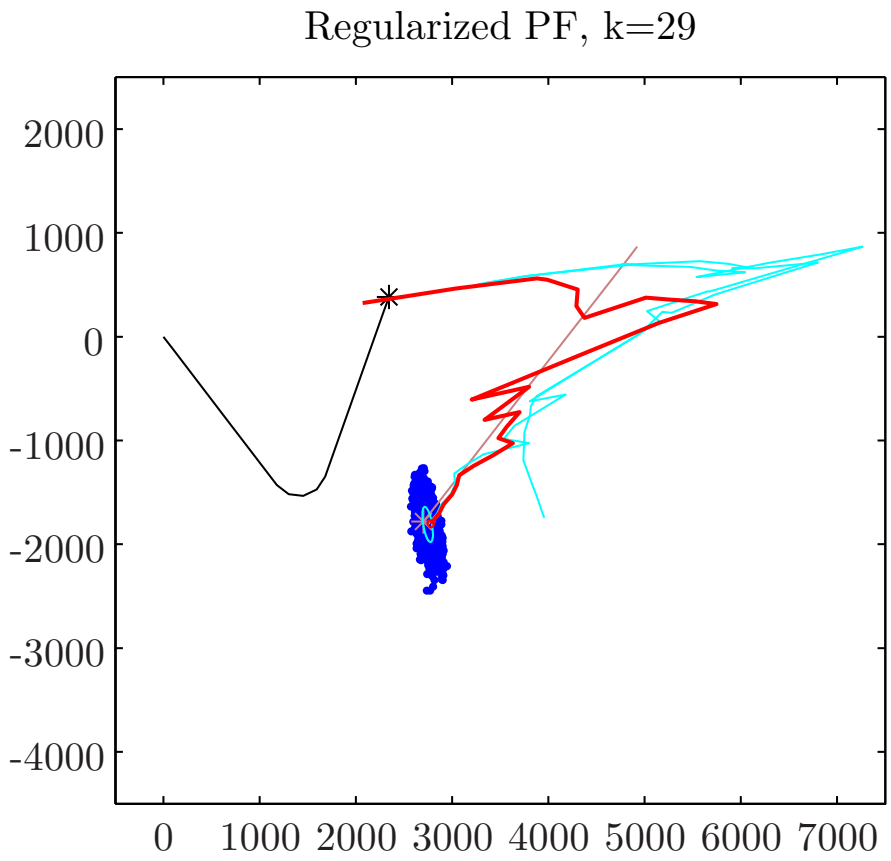




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

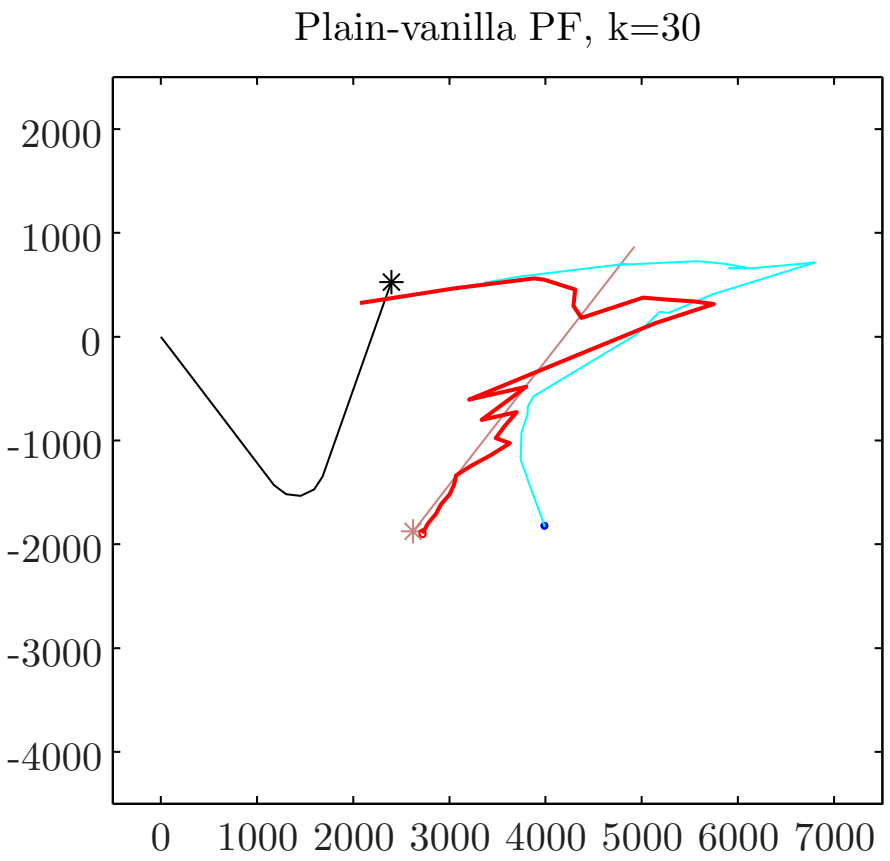
Plain-vanilla vs regularized PF for bearing-only

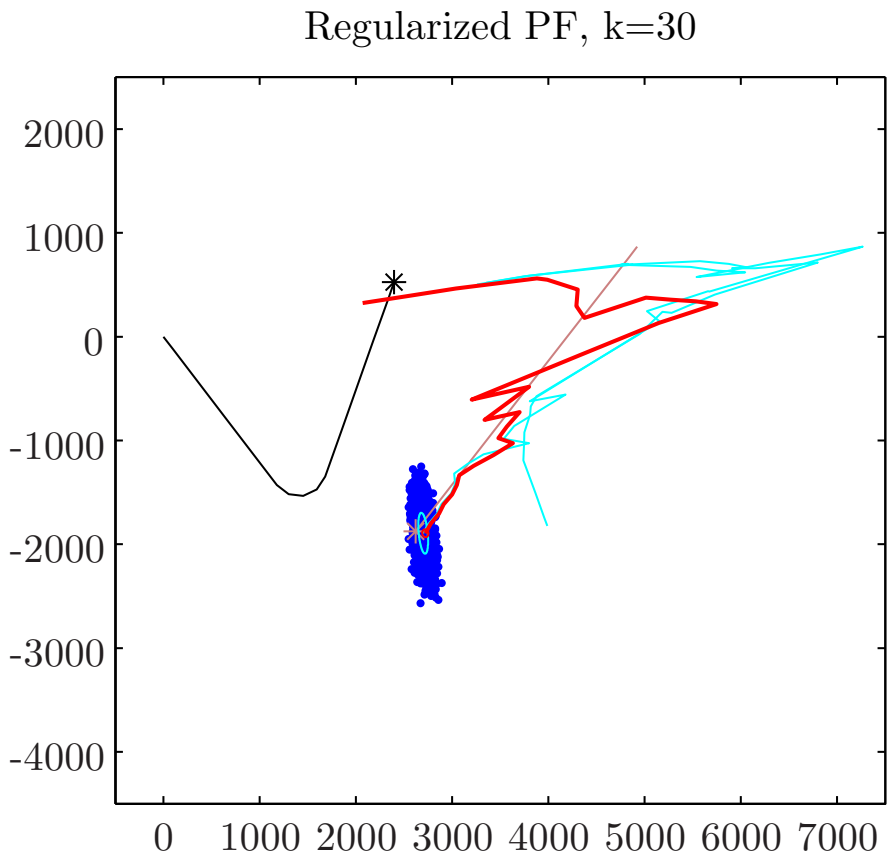




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

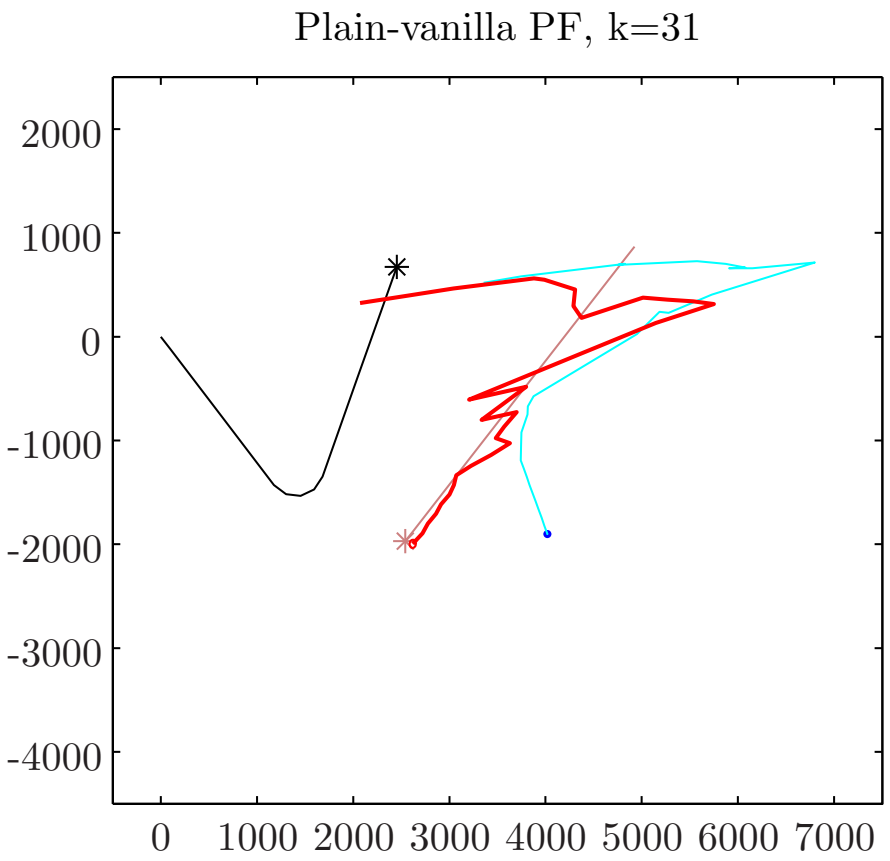
Plain-vanilla vs regularized PF for bearing-only

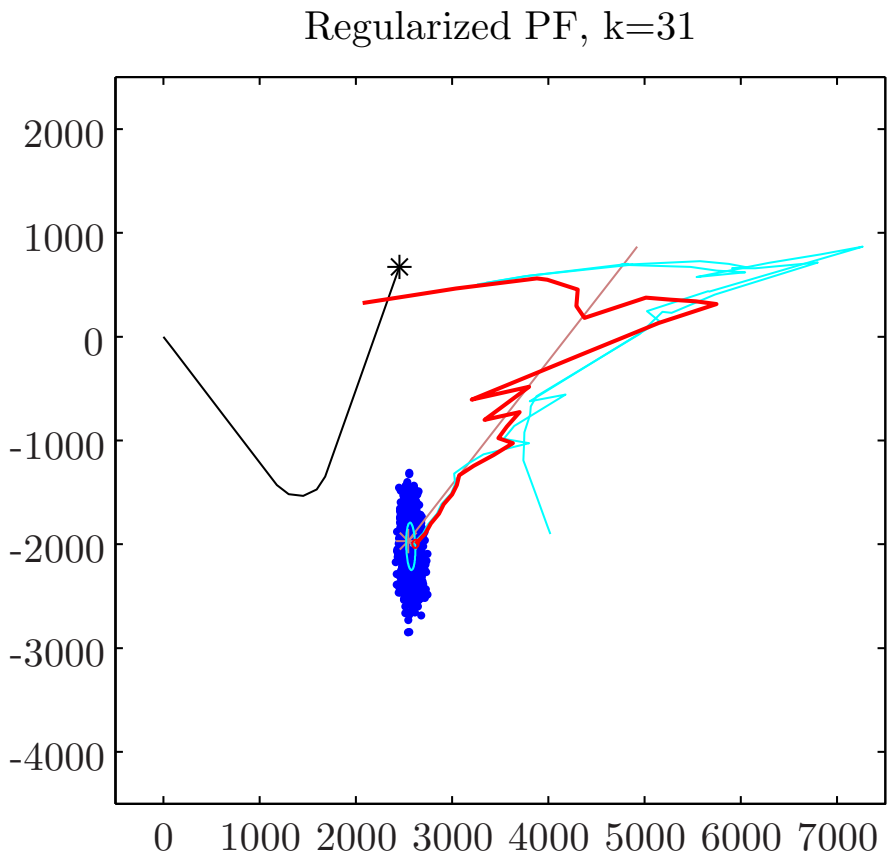




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

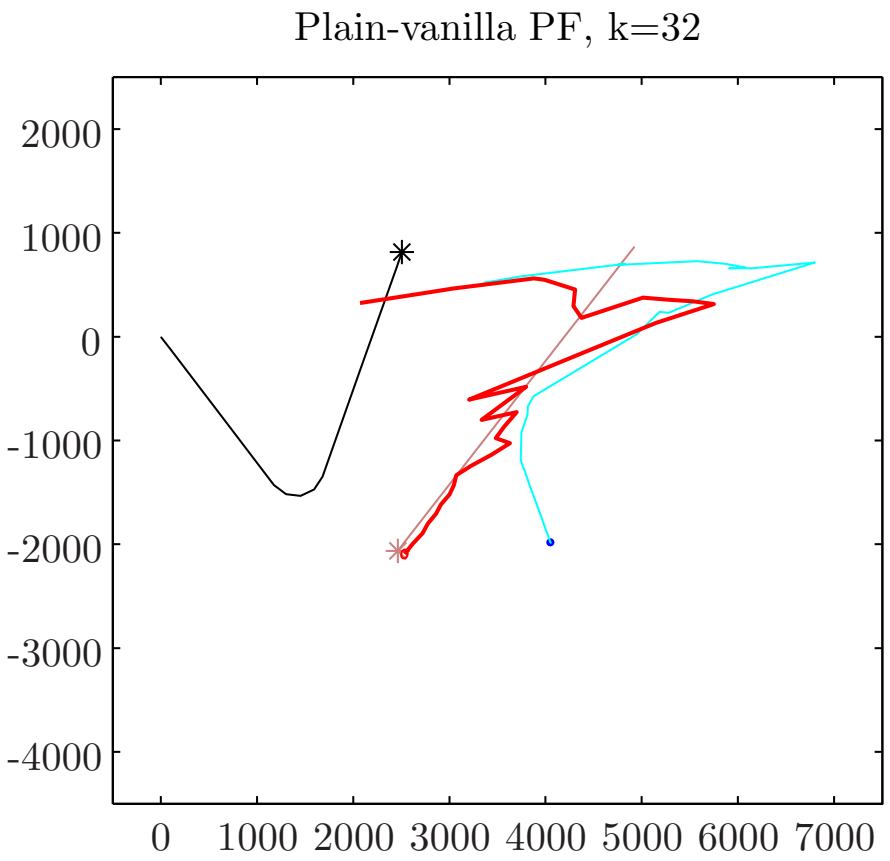
Plain-vanilla vs regularized PF for bearing-only

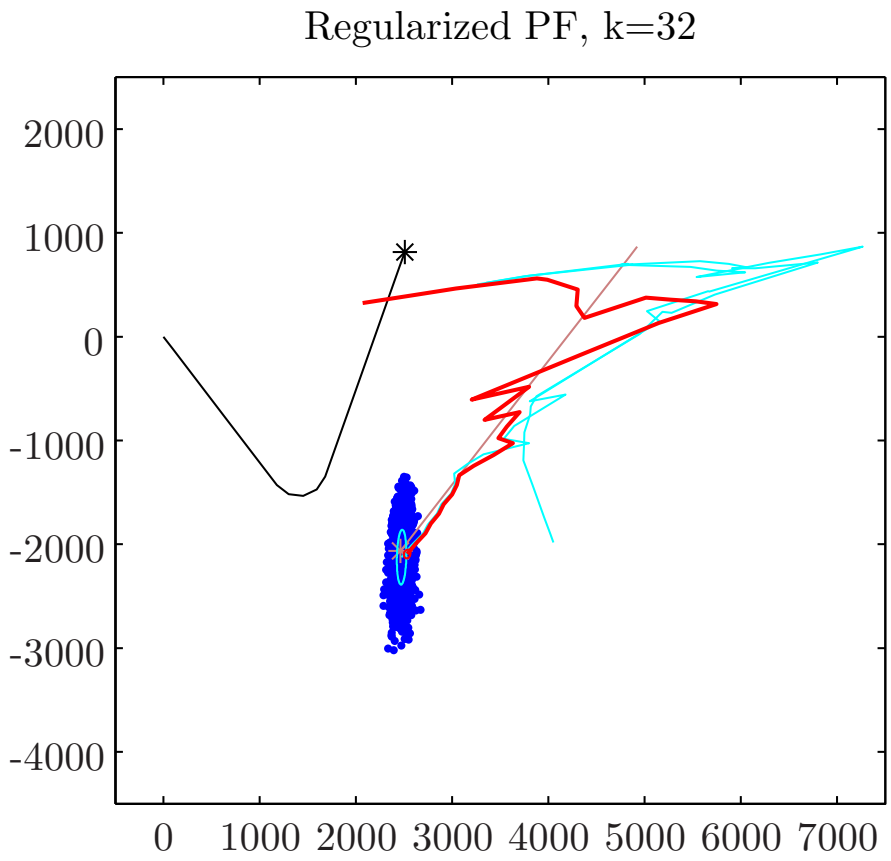




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

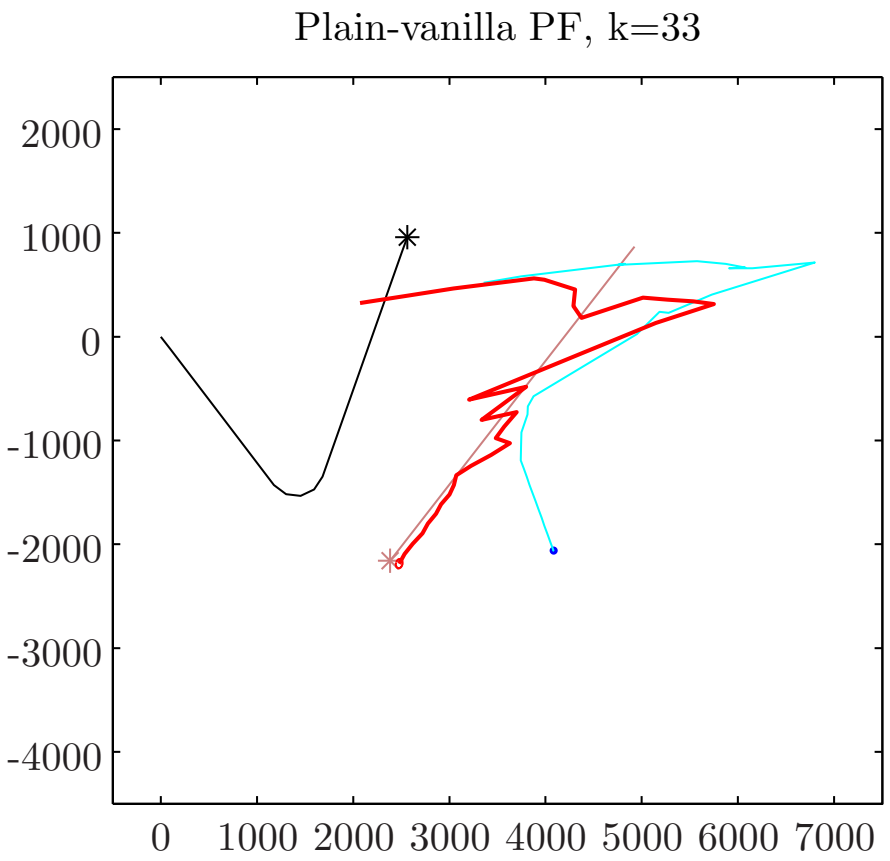
Plain-vanilla vs regularized PF for bearing-only

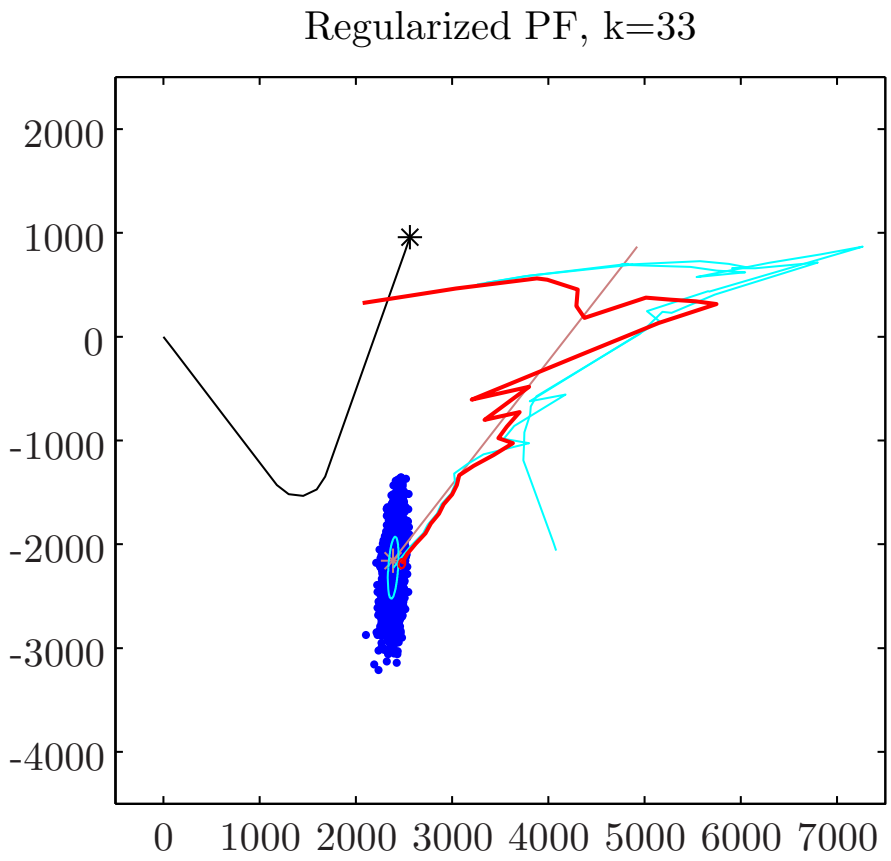




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

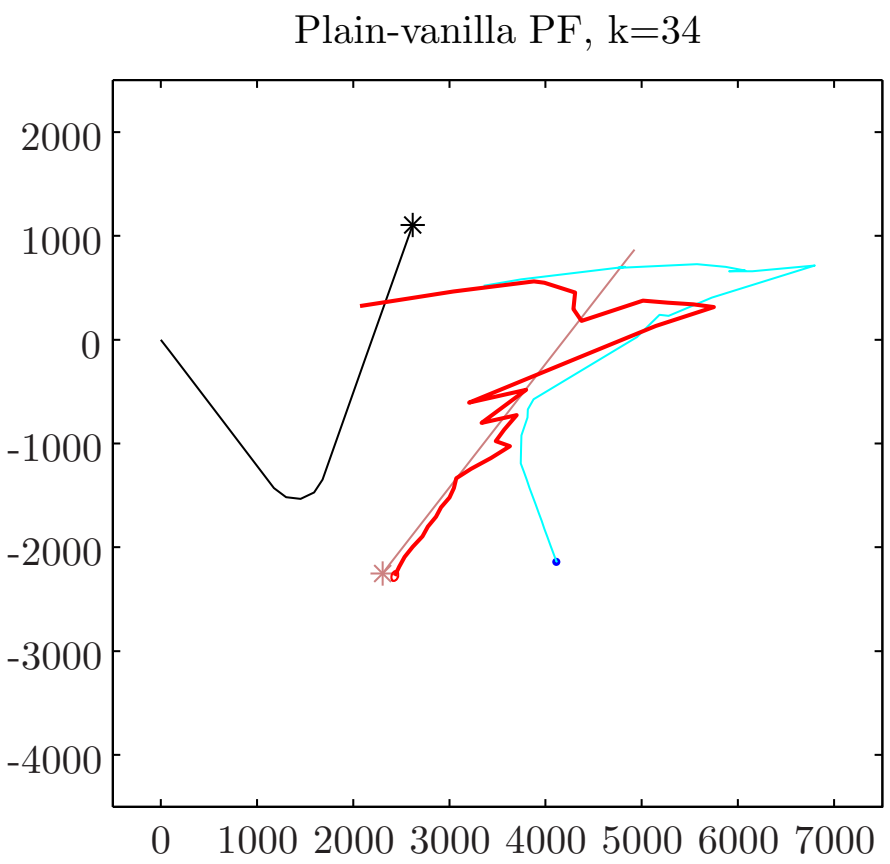
Plain-vanilla vs regularized PF for bearing-only

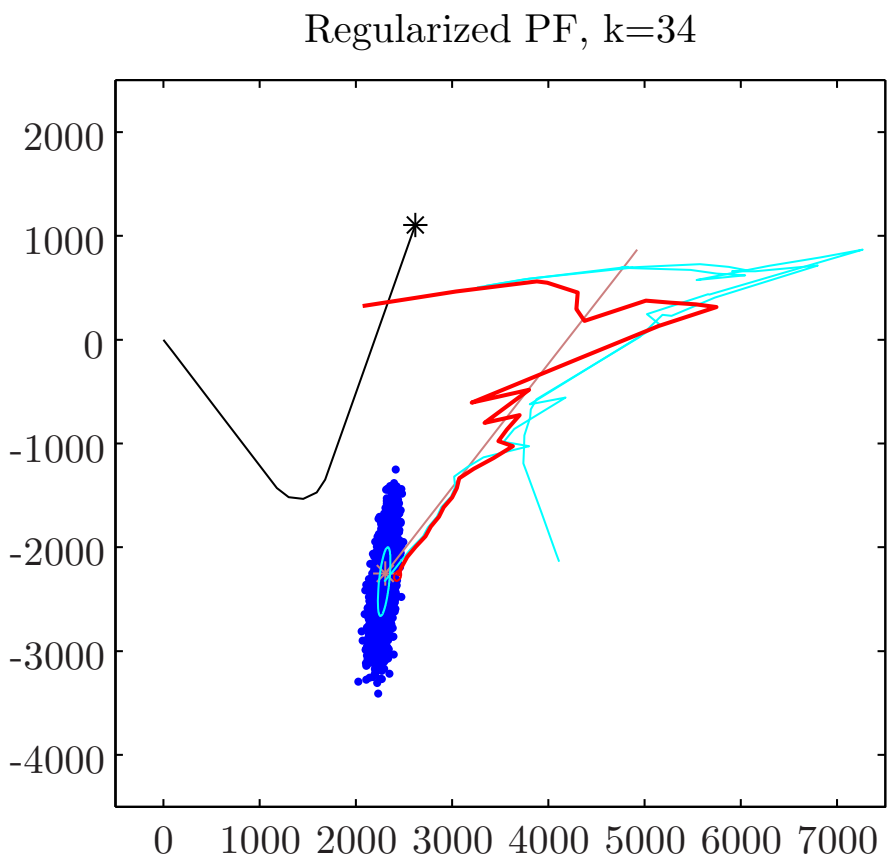




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

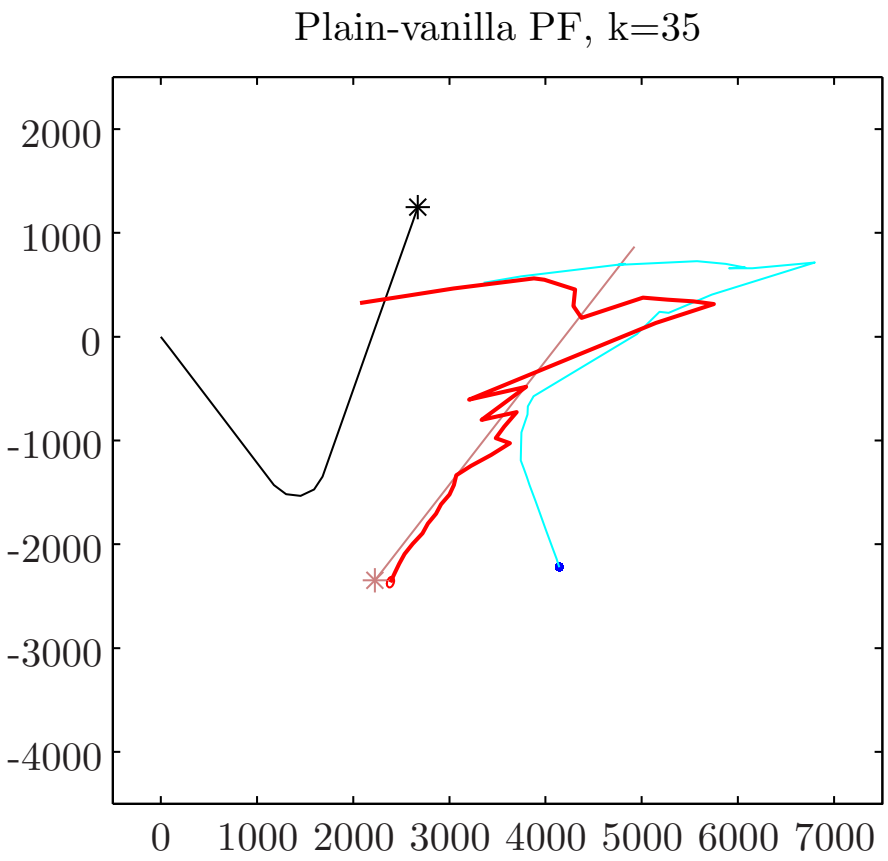
Plain-vanilla vs regularized PF for bearing-only

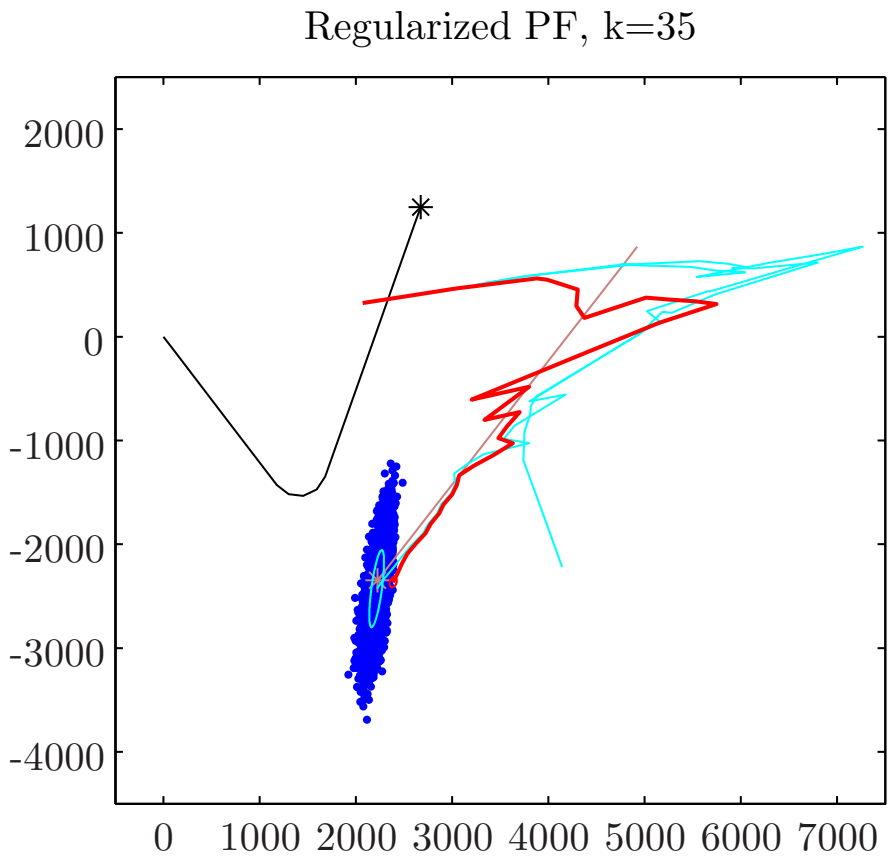




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

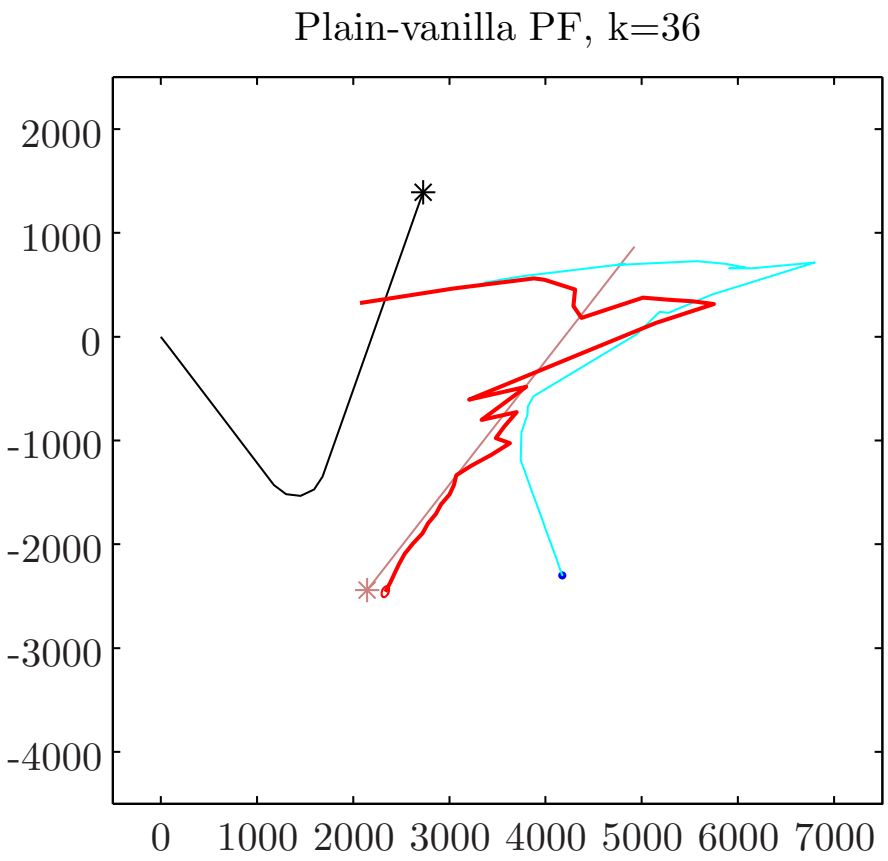
Plain-vanilla vs regularized PF for bearing-only

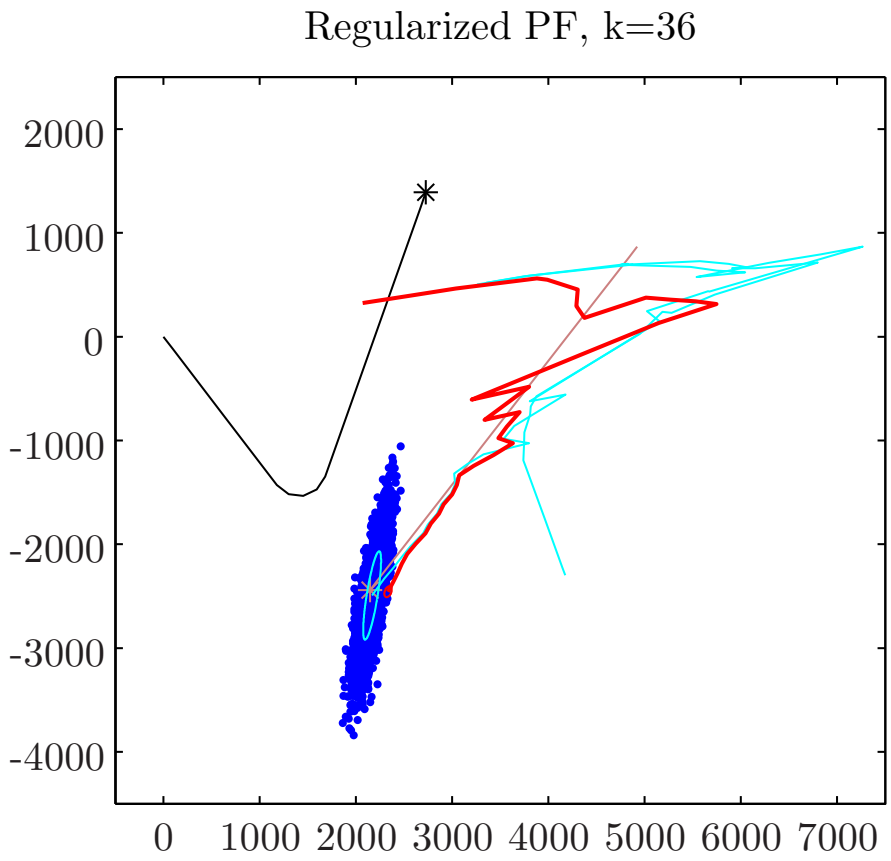




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

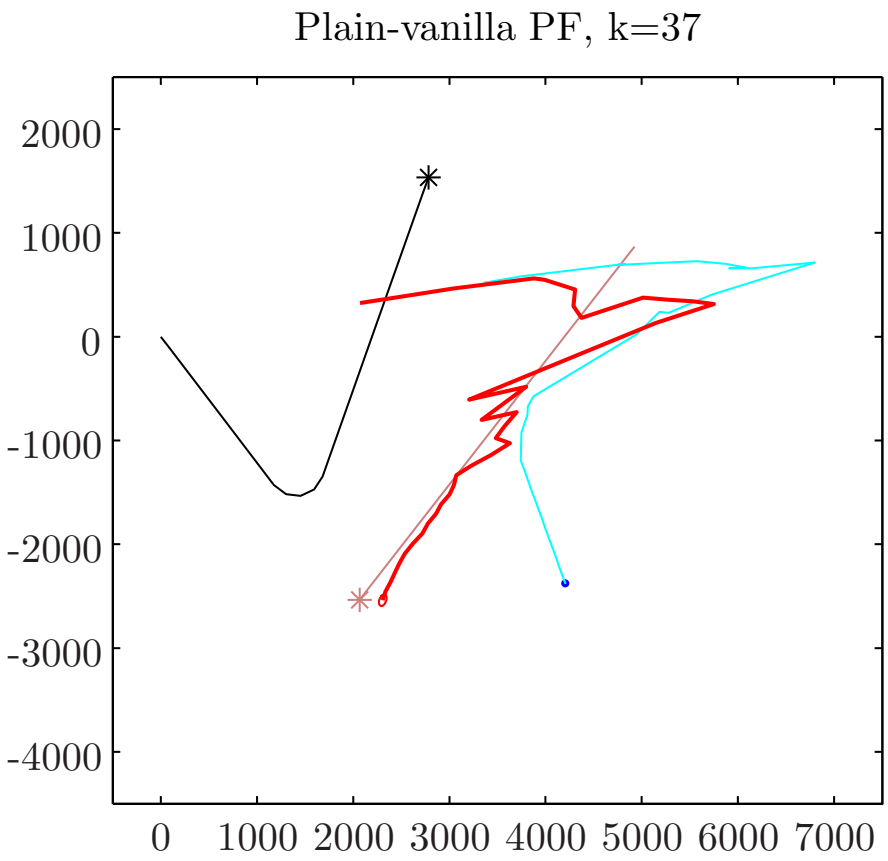
Plain-vanilla vs regularized PF for bearing-only

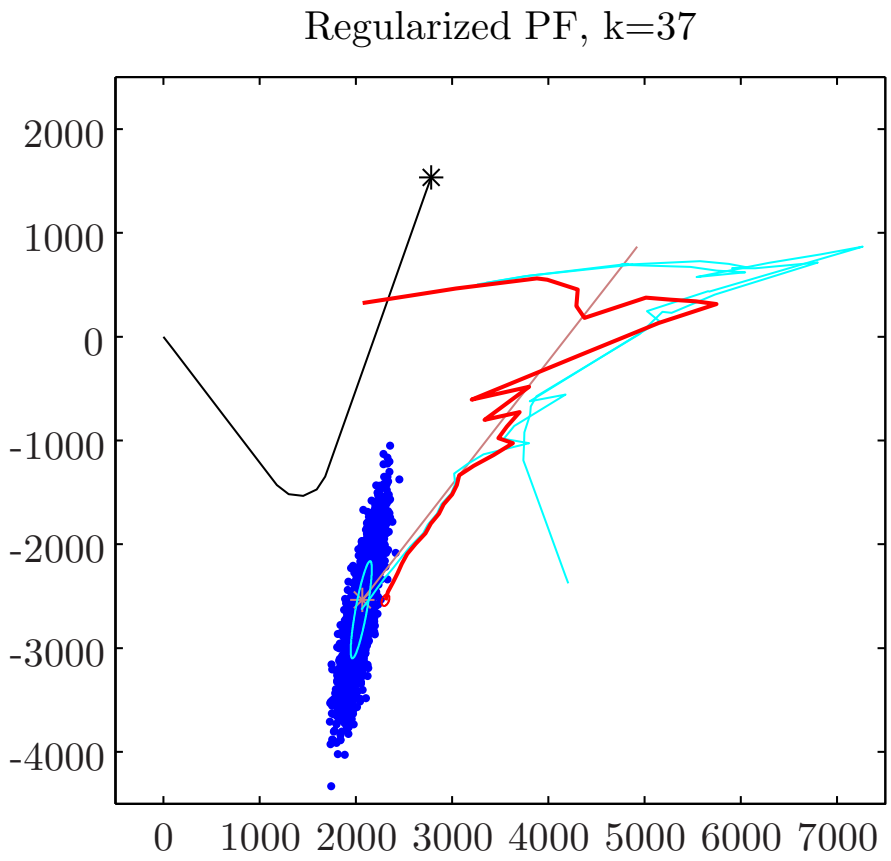




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

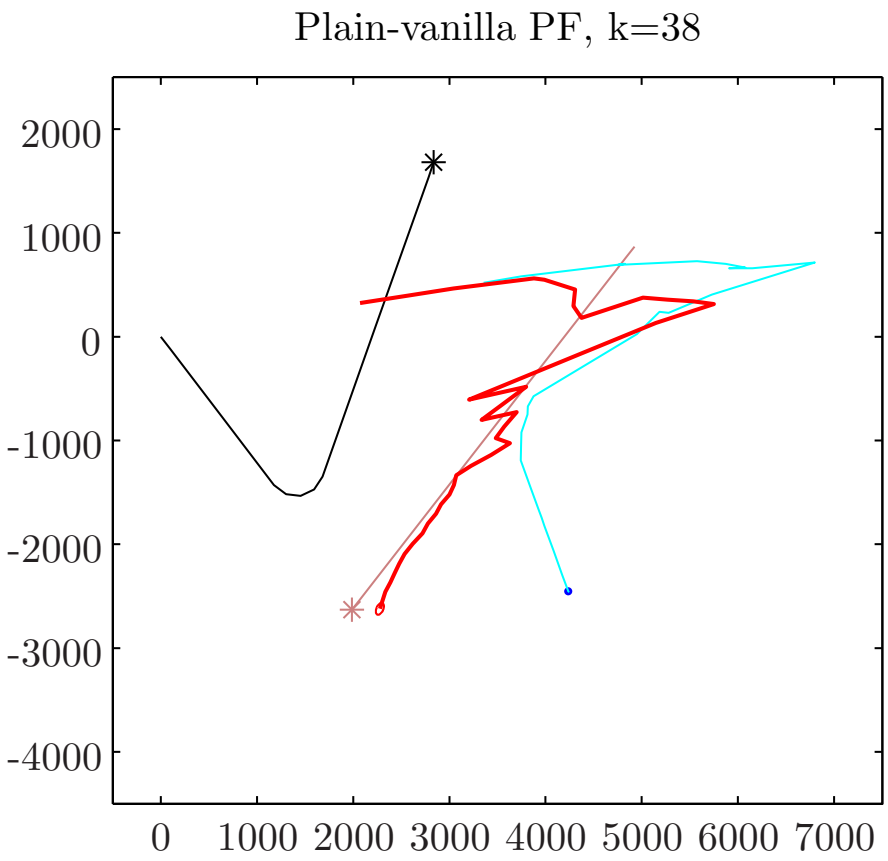
Plain-vanilla vs regularized PF for bearing-only

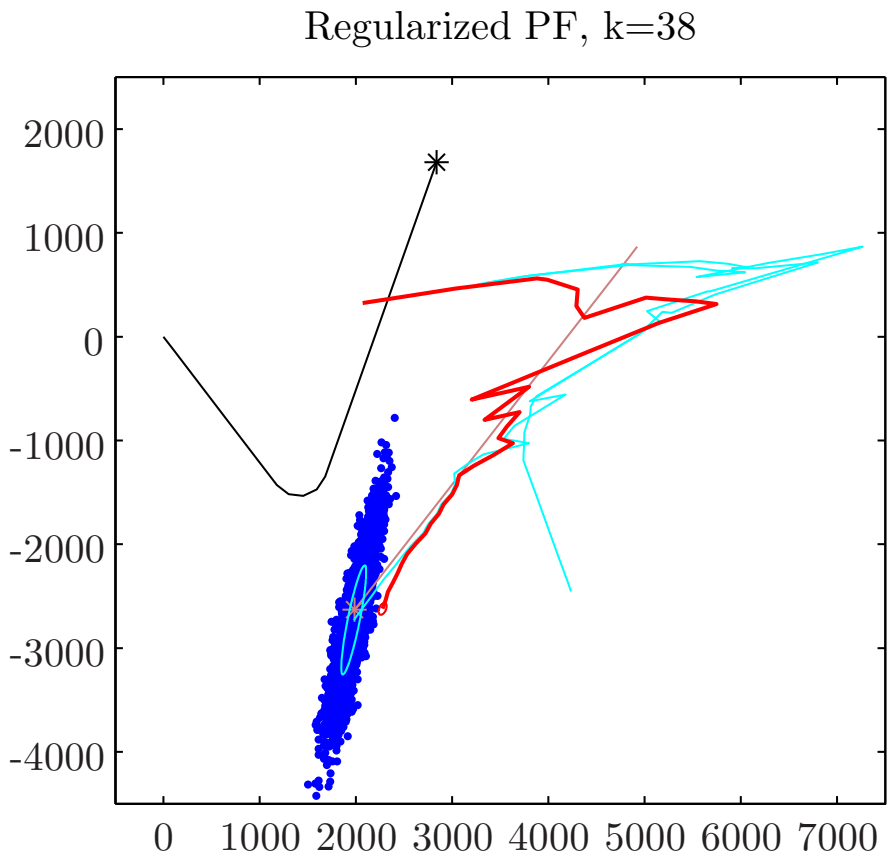




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

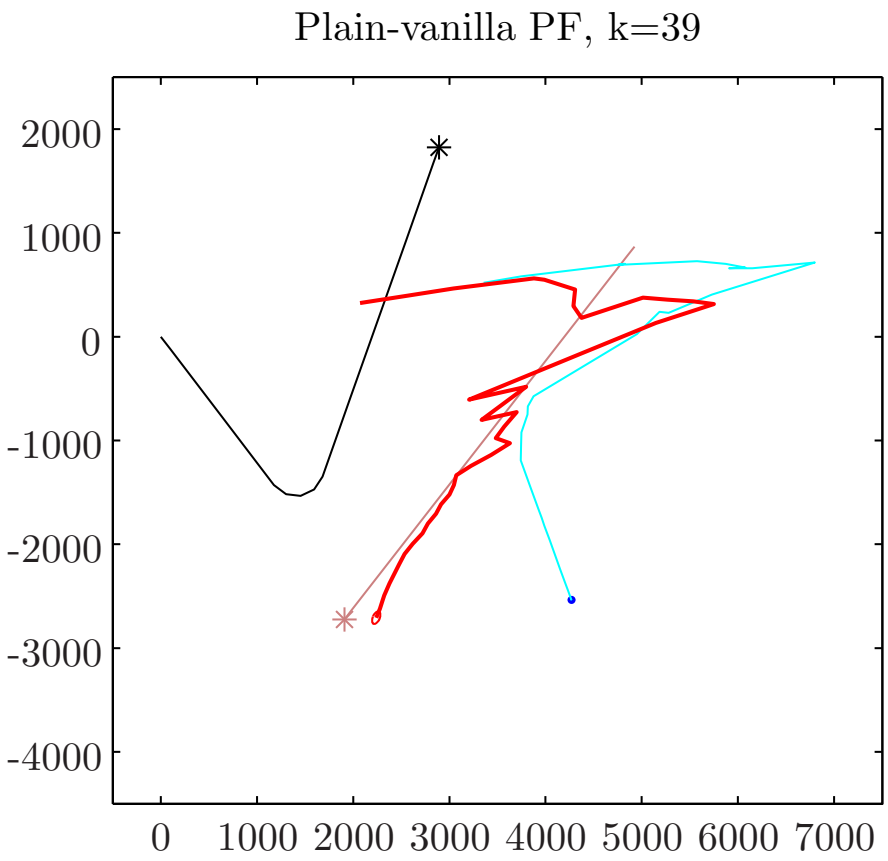
Plain-vanilla vs regularized PF for bearing-only

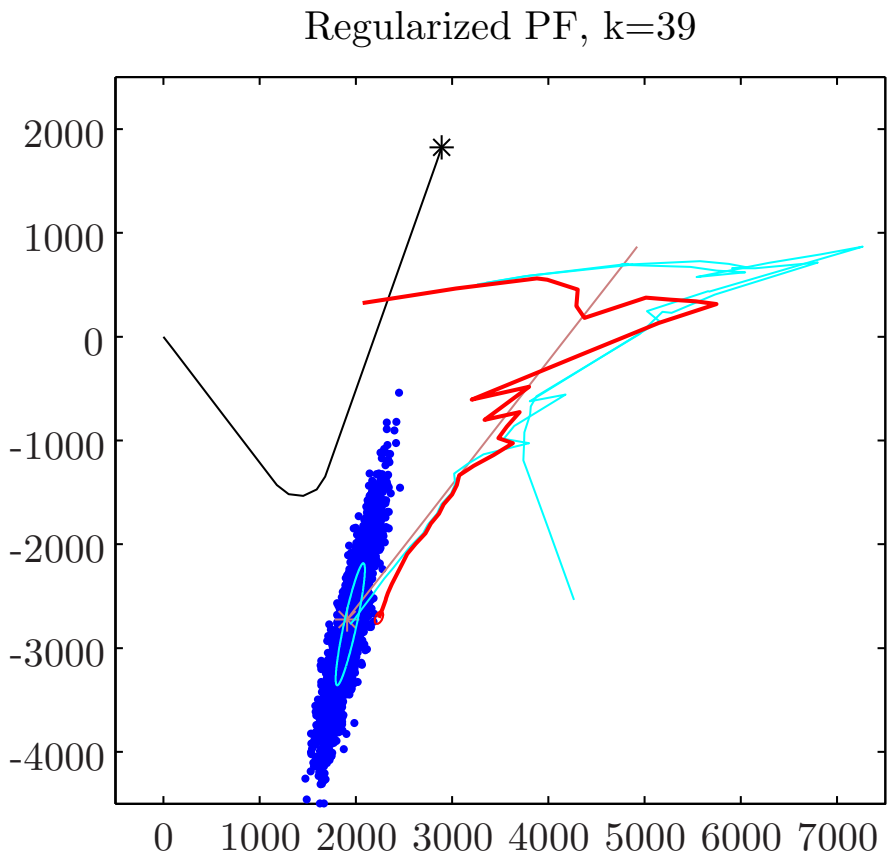




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

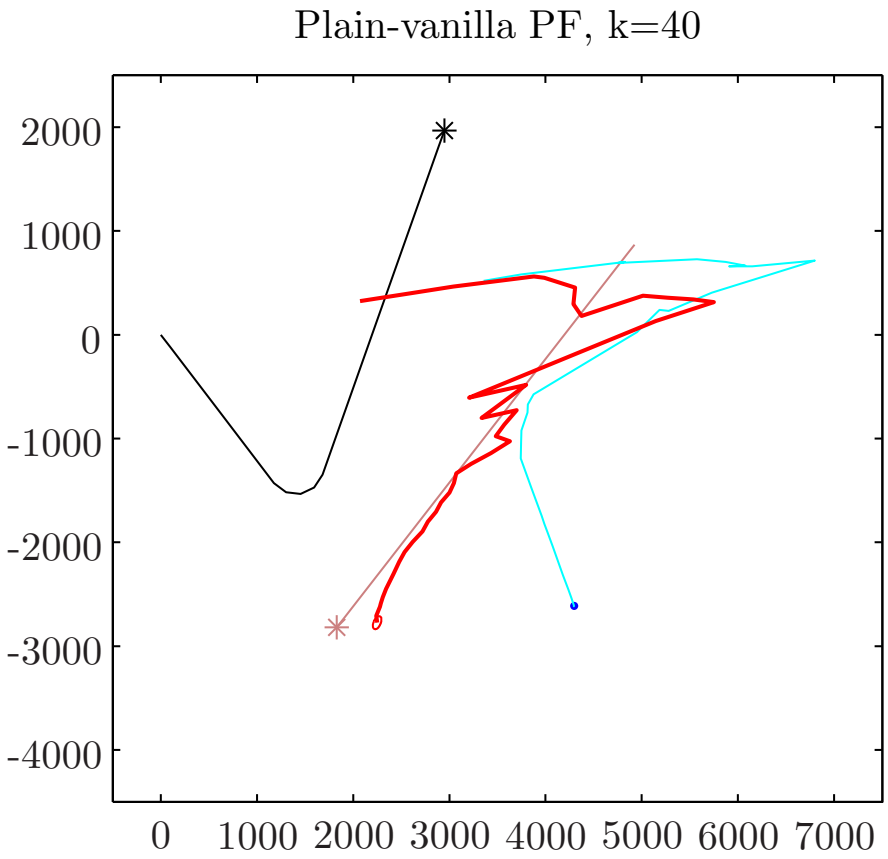
Plain-vanilla vs regularized PF for bearing-only

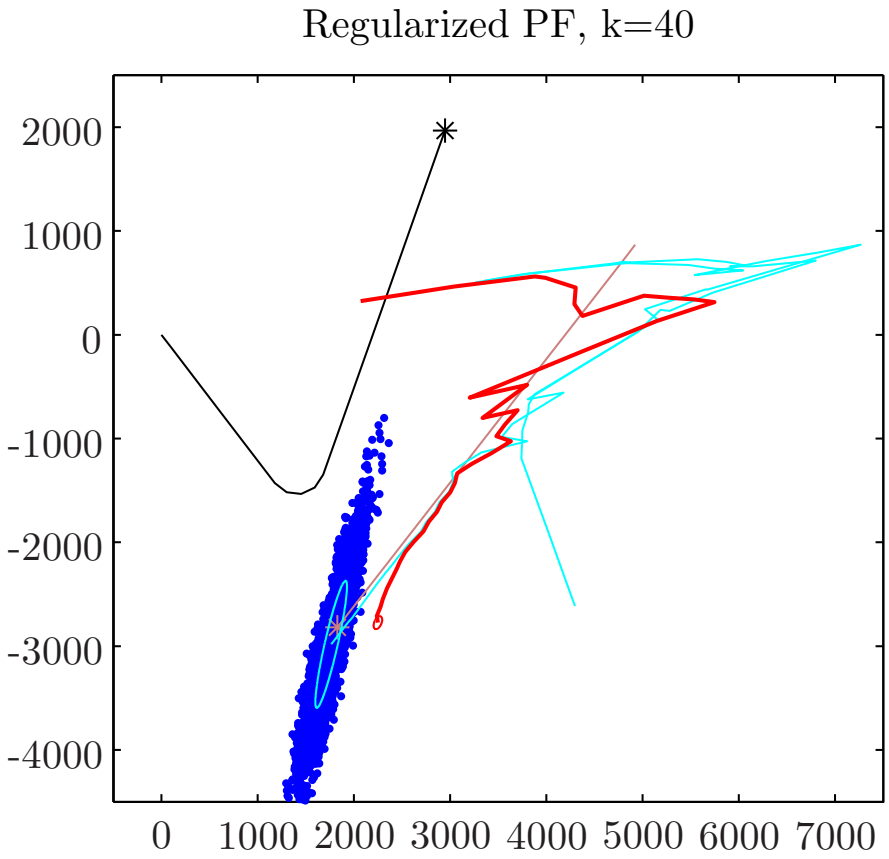




- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

Plain-vanilla vs regularized PF for bearing-only





- Expectation and covariance of EKF (in modified coordinates) displayed in red.
- Particles, expectation and covariance of particle filter displayed in blue colors.

Representing the output from a particle filter

Empirical moments.

$$E[\mathbf{x}_k|\mathbf{z}_{1:k}] \approx \sum_{i=1}^{N} w_k^i \mathbf{x}_k^i \qquad \mathrm{Cov}[\mathbf{x}_k|\mathbf{z}_{1:k}] \approx \sum_{i=1}^{N} w_k^i (\mathbf{x}_k^i - \boldsymbol{\mu}_k)(\mathbf{x}_k^i - \boldsymbol{\mu}_k)^{\mathsf{T}}$$

MAP estimate.

- A dubious approach is to pick the particle with the greatest weight.
- A more refined approximation of the MAP-estimator was proposed in Driessen & Boers (2008): “MAP Estimation in Particle Filter Tracking”.

Empirical PDF on grid.