České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra matematiky
Obor: Inženýrská Informatika
Zaměření: Softwarové inženýrství a matematická informatika

Sekvenční Monte Carlo metody pro asimilaci disperzního atmosférického modelu

Sequential Monte Carlo Methods for Atmospheric Dispersion Model Assimilation

VÝZKUMNÝ ÚKOL

Vypracoval: Matěj Laitl
Vedoucí práce: Ing. Václav Šmídl, Ph.D.
Rok: 2012

Před svázáním místo téhle stránky vložíte zadání práce s podpisem děkana!!!!

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Praze dne .....

.....
Matěj Laitl

Poděkování

Děkuji Ing. Václavu Šmídlovi, Ph.D. za vedení mé práce, četné konzultace a trpělivé vysvětlování matematických struktur problému, které vedlo k lepšímu návrhu softwarové implementace. Dále děkuji Ing. Radku Hofmanovi, Ph.D. za konzultace a ukázku implementace puff modelu, která dovolila ověřit správnost vlastního programu.

Matěj Laitl

Název práce:

Sekvenční Monte Carlo metody pro asimilaci disperzního atmosférického modelu

Autor: Matěj Laitl

Obor: Inženýrská Informatika

Druh práce: Výzkumný úkol

Vedoucí práce: Ing. Václav Šmídl, Ph.D.
Oddělení adaptivních systémů
Ústav teorie informace a automatizace
Akademie věd České republiky

Konzultant: Ing. Radek Hofman, Ph.D.
Oddělení adaptivních systémů
Ústav teorie informace a automatizace
Akademie věd České republiky

Abstrakt: Práce pojednává o asimilaci atmosférického radioaktivního úniku pomocí metod sekvenčního Monte Carlo vzorkování, což je relativně neprozkoumaná aplikace známé metodiky. Nejprve jsou prezentovány zmíněné metody a jejich aplikace na daný problém, následně se práce zaměřuje na volbu tzv. "proposal" funkce MC metody. Naivní a tzv. sdružená proposal funkce jsou následně porovnány, z čehož technika sdruženého proposalu vychází jako jasný vítěz, i když se uvažuje složitost, kterou do algoritmu vnáší. Práce je zakončena rozebráním několika implementačních detailů, z nichž nejzajímavější je 184-násobné urychlení Python kódu nástrojem Cython.

Klíčová slova: asimilace, Bayesovská filtrace, sekvenční Monte Carlo, radioaktivní únik

Title:

Sequential Monte Carlo Methods for Atmospheric Dispersion Model Assimilation

Author: Matěj Laitl

Abstract: The text deals with assimilation of an atmospheric radioactive release using the sequential Monte Carlo technique, which is a rather unexplored application of the well-known framework. The methods and their application to the given problem is presented and then the stress it put on the choice of the proposal density of the particle filter. Naive and conjugate proposals are then compared side-by-side, with conjugate proposal being shown as superior even accounting for its relative complexity. Implementation details are discussed and 184 × speed-up of the Python code is achieved using Cython.

Key words: assimilation, Bayesian estimation, sequential Monte Carlo, radioactive release

Contents

Notation

Following notation is used throughout this text:

$\mathbb{N}$	set of natural numbers excluding zero
$\mathbb{N}_0$	set of natural numbers including zero
$\mathbb{R}$	set of real numbers
t	discrete time moment; $t \in \mathbb{N}_0$
a_t	value of quantity a at time t ; $a_t \in \mathbb{R}^n, n \in \mathbb{N}$
$a_{t:t'}$	sequence of quantities $(a_t, a_{t+1}, \ldots, a_{t'-1}, a_{t'})$
$p(a_t)$	probability density function of quantity a at time t; for the purpose of this text, probability density function p is multivariate non-negative function $\mathbb{R}^n \to \mathbb{R}$; $\int_{\operatorname{supp} p} p(x_1, x_2, \ldots, x_n)\, \mathrm{d}x_1 \mathrm{d}x_2 \cdots \mathrm{d}x_n = 1$
$p(a_t\|b_{t'})$	conditional probability density function of quantity a at time t given value of quantity b at time t'
$\delta(a)$	Dirac delta function; used exclusively in context of probability density functions to denote discrete distribution within framework of continuous distributions, so that $\int_{-\infty}^{\infty} f(x)\delta(x-\mu)\, \mathrm{d}x = f(\mu)$ and more complex expressions can be derived using integral linearity and Fubini's theorem
$\mathcal{N}(\mu, \Sigma)$	multivariate normal (Gaussian) probability density function with mean vector μ and covariance matrix Σ
$\mathcal{G}(k, \theta)$	gamma probability density function with shape parameter k and scale parameter θ
$i\mathcal{G}(\alpha, \beta)$	inverse gamma probability density function with shape parameter α and scale parameter β

Introduction

Recursive Bayesian estimation of dynamic systems is a vital field that is commonly associated with robotics, object-tracking and the like as demonstrated for example by [5, 15]. Sequential Monte Carlo methods (also referred to as the SIR[1] filter or the particle filter) is a broad Bayesian estimation subfield attractive mainly when the posed problem doesn't offer analytical Bayesian solution. Among other publications, [6, 7] have shown that SMC techniques can be successfully applied to tracking of the radioactive plume in the early phase[2] of an atmospheric radioactive pollutant release accident. Should the research result in a self-contained and reliable tool, it would, in our opinion, significantly help personnel involved with the accident in making informed decisions, as SMC can effectively combine various noisy data sources into one coherent estimation of the situation.[3] This text aims to be a one step among many towards this goal. Specifically, we're concerned with computational efficiency of variations of the SMC technique in the first place, and secondly we've set up a goal of implementing basics of the desired tool.

This text is organized as follows:

1. In the first chapter the theoretical foundations of the Bayesian estimation are reviewed and algorithm of the particle filter is described and briefly discussed.

2. Second chapter presents physical background of the pollutant release and model we've used.

3. Chapter 3 combines previous chapters to form complete assimilation technique adapted to radioactive release. Proposal distribution possibilities are discussed and 2 are chosen to be benchmarked.

4. Finally, results from the comparison are presented. Furthermore our approach implementing the assimilation software is described and Cython is used to dramatically speed up Python code.

[1] Sequential Importance Resampling
[2] hours within the start of the release
[3] which can be further extended to give future prediction

Chapter 1

Sequential Monte Carlo Methods

This chapter presents the Sequential Monte Carlo method[4] (which is also known as the particle filter or the SIR[1] filter) and its variants within the framework of Bayesian statistics. The problem of the recursive Bayesian estimation is first stated, its general solution is then briefly derived. Finally, the particle filter is then introduced as an approximation of the general solution and its properties and alternate forms are discussed.

1.1 Recursive Bayesian Estimation

Suppose a hidden state vector sequence $x_{1:t}$ has to be estimated in discretized time steps $1:t$ using a sequence of measurements $y_{1:t}$ with knowledge of the state model (which describes how state evolves in time) and the observation model (which describes how a given state x_t is observed in y_t). Suppose also that both state model and observation are encumbered by random noise, whose statistical parameters are known. Such situation can then be mathematically formalized as

$$x_t = f_t(x_{t-1}, v_{t-1}) \tag{1.1}$$
$$y_t = h_t(x_t, w_t), \tag{1.2}$$

where f_t is the state model at time t, v_t is the random noise of the state model at time t, h_t is the observation model and w_t is the observation noise at time t. Both state and observation models often don't depend on time and are referred to simply as f and h respectively. To make our reasoning bellow correct, one must further assume that both state and observation posses the Markov property, i.e. x_t depends only on

General solution (in Bayesian sense) of the problem is known and portrayed here. Both x_t $(\forall t)$ and y_r $(\forall t)$ are treated as (multivariate) random variables from now

[1] Sequential Importance Resampling

on. First, let us observe that the state probability density function $p(x_t|x_{t-1})$ can be derived from (1.1) and the observation probability density function $p(y_t|x_t)$ can be derived from (1.2) likewise. So-called *prior* probability density function $p(x_t|y_{1:t-1})$ can be expressed using only known densities by applying reverse marginalization over x_{t-1}, applying chain rule for probability density functions and finally taking into account that x_t doesn't depend on y_{t-1}

$$\begin{aligned} p(x_t|y_{1:t-1}) &= \int p(x_t, x_{t-1}|y_{1:t-1})\ \mathrm{d}x_{t-1} \\ p(x_t|y_{1:t-1}) &= \int p(x_t|x_{t-1}, y_{1:t-1}) p(x_{t-1}|y_{1:t-1})\ \mathrm{d}x_{t-1} \\ p(x_t|y_{1:t-1}) &= \int p(x_t|x_{t-1}) p(x_{t-1}|y_{1:t-1})\ \mathrm{d}x_{t-1}. \end{aligned} \tag{1.3}$$

Next straightforward step in the solution is to derive *posterior* probability density function $p(x_t|y_{1:t})$, which is done by applying the Bayes' theorem and taking the fact that observation y_t depends only on the newest state x_t so that $p(y_t|x_t, y_{1:t-1}) = p(y_t|x_t)$

$$\begin{aligned} p(x_t|y_{1:t}) &= \frac{p(y_t|x_t, y_{1:t-1}) p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \\ p(x_t|y_{1:t}) &= \frac{p(y_t|x_t) p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \end{aligned} \tag{1.4}$$

where the denominator of (1.4) can be computed using marginalization over x_t or not at all, because it doesn't depend on x_t and serves only the purpose of the normalization constant. Formulas (1.3) and (1.4) together form the solution of the problem posed above.

On the other hand, this solution in its general form has only limited application, because the integral (1.3), unless tractable analytically, constitutes a major computational challenge. The situation when the analytical solution is known is represented mainly by the famous Kalman filter[9] and its variants, which applies when state and observation models are linear and noises are normally distributed. If these models are highly non-linear (which is our case in radioactive release assimilation), one of the viable approaches is to resort to some kind of approximation of the solution. One such approximation is the particle filter described below.

1.2 The Particle Filter

The particle filter (or sequential Monte Carlo technique, SIR filter) is based on approximation of the posterior probability density function using the weighted em-

Algorithm 1.1 Recursive update of the particle filter

1. for each i in $\{1, \ldots, N\}$ do:
   - draw $x_t^{(i)}$ from the proposal density $q(x_t|x_{t-1}, y_t)$, where x_{t-1} is substituted by $x_{t-1}^{(i)}$ and y_t is the observation.
   - update weight using (1.6) where x_{t-1} is substituted by $x_{t-1}^{(i)}$, x_t is substituted by $x_t^{(i)}$ and y_t is the observation,

$$\omega_t^{(i)} := \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q(x_t|x_{t-1}, y_t)}\omega_{t-1}^{(i)}. \tag{1.6}$$

2. for each i normalize weights according to $\omega_t^{(i)} := \frac{\omega_t^{(i)}}{\sum_{j=1}^{N}\omega_t^{(j)}}$.

3. resample $x_t^{(1:N)}$ from posterior probability density function 1.5 eliminating low-weight particles, but maintaining posterior statistics.

pirical density

$$p(x_t|y_{1:t}) \approx \sum_{i=1}^{N} \omega_t^{(i)}\delta(x_t - x_t^{(i)}), \tag{1.5}$$

where N is the total number of particles (an important parameter of the filter), $\omega_t^{(i)}$ is the weight of the i-th particle[2] and finally $x_t^{(i)}$ is the value of the i-th particle, a hypothetical state the model could be in. The algorithm begins by i.i.d.[3] sampling initial particles $x_0^{(1:N)}$ from the initial probability density function $p(x_0)$, which is assumed to be known. When new observation y_t is available, recursive update is performed as described in Algorithm 1.1. [3] The algorithm makes use of a so-called *proposal* density $q(x_t|x_{t-1}, y_t)$, which is an instrument to optimize the algorithm to a specific problem and which will be used extensively in chapter 3 to improve its performance. Simplest choice of the proposal density (and the one originally described in [4]) is directly the transition density $p(x_t|x_{t-1})$ (sometimes called *naive* proposal) that ignores the observation, which simplifies the weight calculation in (1.6) to $\omega_t^{(i)} := p(y_t|x_t)\omega_{t-1}^{(i)}$. Such algorithm is known as the *bootstrap filter*.

The biggest advantage of the particle filter comes from the fact that only sampling from the proposal density and evaluation of it, the transition and observation density is needed, thus it can be applied to virtually any problem of recursive

[2] constraint $\sum_{i=1}^{N}\omega_t^{(i)} = 1 \quad \forall t$ must hold
[3] independently identically distributed

Bayesian estimation, regardless of analytical tractability of the integrals therein. However, this comes with a cost. First and foremost, the particle filter is an approximate and stochastic method. While convergence to the optimal Bayesian solution (1.4) as N approaches infinity is proven under a set of assumptions[2], real-world applications must find a trade-off between accuracy (which improves as N grows) and performance (where computation time linearly depends on N).

Another phenomena worth mentioning is the *sample impoverishment* problem, which is described as a tendency of the weights of all-but-one particle to drop to zero as time progresses (unless counter-measured). Untreated, this can have a devastating effect on the accuracy of the filter as its posterior density degenerated to a single Dirac delta function. The major approach to eliminate this is the resampling described in step 3. of the Algorithm 1.1. *Number of effective particles* N_{eff} is commonly used as a measure of the sample impoverishment and can be estimated using (1.7).[3]

$$N_{\text{eff}}(t) \approx \left(\sum_{i=1}^{N} \left(\omega_t^{(i)} \right)^2 \right)^{-1}, \qquad 1 \leq N_{\text{eff}} \leq N. \tag{1.7}$$

N_{eff} close to N denotes near-uniform weight distribution while N_{eff} approaching 1 is a sign of the degenerate case. The resampling step mitigates some effects of the sample impoverishment, but if $N_{\text{eff}} \ll N$, available resources are used ineffectively. This is most pronounced when the proposal density doesn't match the resulting posterior density sufficiently (often being too broad) a condition often observed in the bootstrap filter. Careful choice of the proposal density can yield significantly better results with little cost as shown in chapter 3.

Chapter 2

Atmospheric Radioactive Release Model

In order to test various sequential Monte Carlo techniques on a real-world and useful example, the assimilation[1] of atmospheric radioactive release was chosen as a sufficiently demanding benchmark given its highly non-linear observation model. Chis chapter describes the release scenario and mathematical models used to simulate it. We've used a slightly simplified version of the one presented in [16].

2.1 Release Scenario

We are concerned with an early phase[2] of an atmospheric radioactive pollutant release from a nuclear power plant or a similar site. Our goal is to estimate absorbed ionizing radiation doses at points of interest using available environmental measurements and to predict future development of the radioactive plume and related irradiation. True power plant[3] and existing infrastructure was used as a basis for our simulation. As Figure 2.1 shows, the power plant is equipped with a network of on-line absorbed ionizing dose sensors (blue circles in the figure) that are laid out in a ring around the plant plus a few of them are scattered in nearby villages. As the plume containing the pollutant is subject to advection and dispersion in the atmosphere, accurate assimilation of atmospheric conditions is crucial. Following real-life possibilities, we assume that wind field predictions from the numerical model are available, though on a sparse grid both spatially (tile size 9 km) and temporally (1

[1] process of adapting numeric simulation to actual measured quantities so that the simulation is in accordance with them

[2] first hours after the release, when available information is scarce and good estimate of the current situation and expected progress is critical for decision-making authorities

[3] Temelín

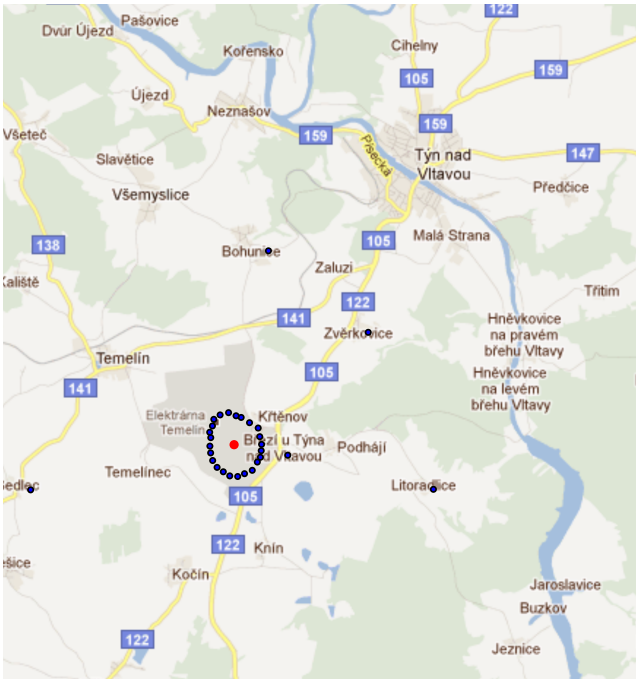

Figure 2.1: Temelín power plant and adjacent monitoring network

hour time step). Weather station with anemometer[4] is located in the power plant area and can be used to measure local wind conditions.

2.2 State-Space Model

We've chosen the state-space model described in [8] with some extensions found in [16] for its simplicity and suitability for sequential Monte Carlo assimilation. Its brief summary continues. The so-called *puff model* approximates the pollutant plume using a sequence of puffs released periodically from the source location (marked by red circle in Figure 2.1), each with certain amount of pollutant[5]. Pollutant concentration is modeled using 3D Gaussian distribution with σ_x, σ_y, σ_z dispersion coefficients in respective dimensions that describe how the puff is spread (dispersed) spatially. More precisely, a concentration of the pollutant is given by

$$C(\mathbf{s},\,\tau) = \frac{Q_\tau}{(2\pi)^{\frac{3}{2}}\sigma_x\sigma_y\sigma_z}\exp\left[-\frac{(s_x - l_{x,\tau})^2}{2\sigma_x^2} - \frac{(s_y - l_{y,\tau})^2}{2\sigma_y^2} - \frac{(s_z - l_{z,\tau})^2}{2\sigma_z^2}\right], \qquad (2.1)$$

[4] a device measuring wind speed and direction

[5] described by total radioactive activity in our case

where $\mathbf{s} = (s_x, s_y, s_z)$ denotes spatial coordinates, $\mathbf{l}_\tau = (l_{x,\tau}, l_{y,\tau}, l_{z,\tau})$ coordinates of the puff center at time τ, Q_τ [Bq] total puff activity at given time. Puffs are carried away by wind (direction and speed of which fully determines puff trajectory) in discrete time steps and are subject to dispersion which manifests as growth of the σ parameters. The rate of growth depends on total distance flown by the puff and on current Pasquill's stability category.[6] In case of radioactive release the total puff activity decreases over time by radioactive decay.

Once concentration of the pollutant at given location and time (2.1) is known, total dose rate at given receptor location $\mathbf{s}_r$ that comes from a single puff can be computed using

$$D(\mathbf{s}_r, t_1, t_2) = K_c \int_{t_1}^{t_2} \Phi(\mathbf{s}_r, \tau, E)\, \mathrm{d}\tau, \tag{2.2}$$

where Q is the puff activity, K_c is a physical constant, E the energy produced by gamma decay of released radionuclide (assuming only one radionuclide contributes to absorbed dose) and $\Phi(\mathbf{s}_{R_j}, \tau, E)$ fluency rate that can be computed using

$$\Phi(\mathbf{s}_r, \tau, E) = \int_\Omega \frac{C(\mathbf{s},\, \tau)\,(1 + k\,\mu r)\exp(-\mu r)}{4\pi r^2}\, \mathrm{d}\mathbf{s}. \tag{2.3}$$

Here, k and μ are physical constants, $r = ||\mathbf{s}_r - \mathbf{s}||$ distance from receptor and Ω is the volume where the puff has not-negligible concentration. Dose from a series of puffs is additive. It should be noted that analytical solution of (2.3) is not known and must be evaluated numerically and presents the most resource-demanding part of the model even if techniques to reduce problem size. Integral in (2.2) is computed as a simple addition of the element of integration at chosen time sub-steps.

Using a model described above, simulation was performed to test assimilation techniques against. Series of 6 puffs with activities $[1, 5, 4, 3, 2, 1] \times 10^{16}\, Bq$ were released at 10-minute intervals. Simulated wind field had constant speed of 2 m/s, but slightly varying wind direction; Pasquill's stability category was D (neutral). Figure 2.2 shows total absorbed doses at ground in Sieverts (contour) and puff centre trajectories (color lines).

[6] a meteorologic measure of atmospheric turbulence, an important factor affecting dispersion

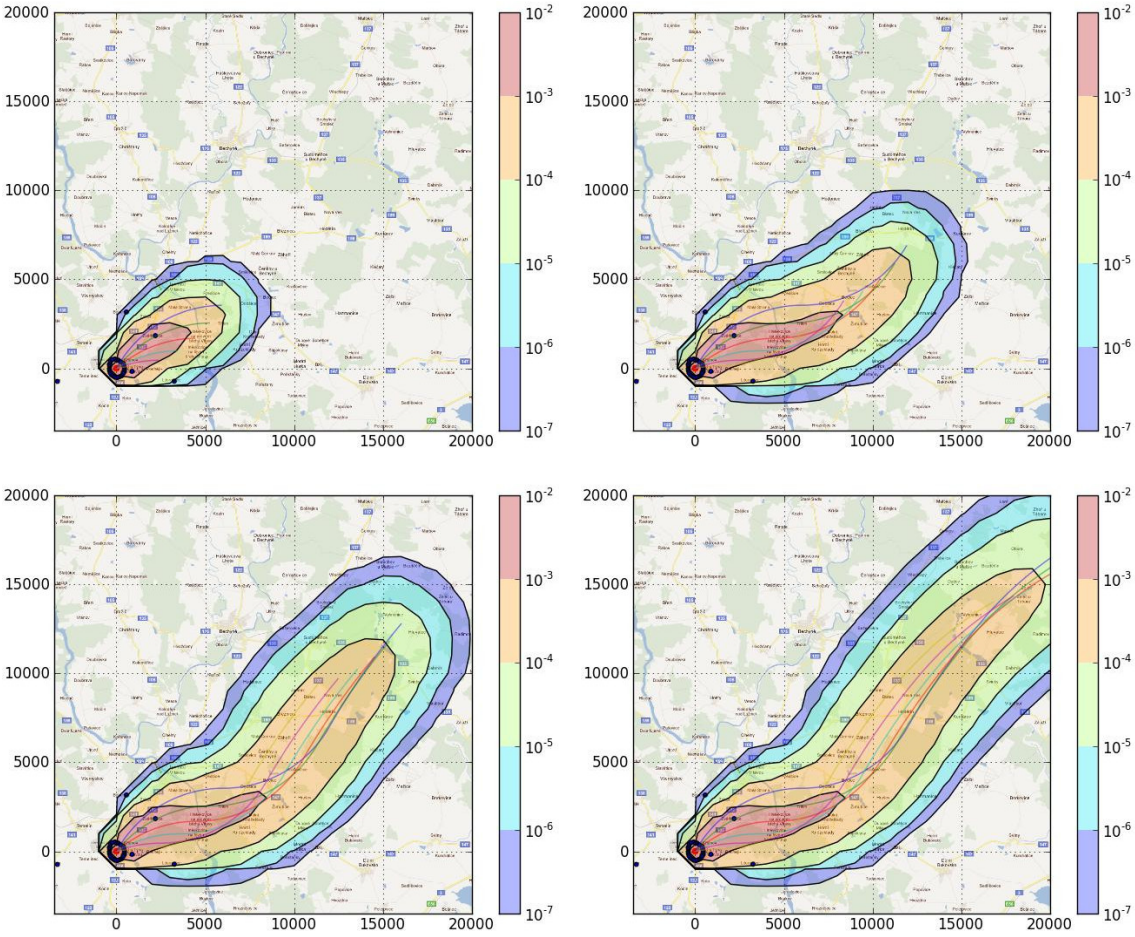

Figure 2.2: Simulation of radioactive release; 1, 2, 3, 4 hours after start

Chapter 3

Assimilation

In this chapter we finally apply theoretical solution from chapter 1 to assimilation problem described in chapter 2. First, we show our choice of measurement (observation) and transition (state) model, the rest is dedicated to discussing the choice of the proposal density. Similar models were already proposed in [16].

3.1 Wind Field Corrections

Observation of the wind field is available from 2 sources: *i)* forecast from the numerical model, here denoted as $\tilde{v}_{t,\mathbf{s}}$ for wind speed and $\tilde{\phi}_{t,\mathbf{s}}$ for wind direction at time t and spatial coordinates $\mathbf{s}$, *ii)* readings from on-site anemometer, whose wind speed measurement is denoted as v_t, wind direction measurement as ϕ_t. None of these quantities are perfect, $\tilde{v}_{t,\mathbf{s}}$ and $\tilde{\phi}_{t,\mathbf{s}}$ are computed on a sparse grid and must be interpolated in time and space, they are additionally not well adapted to local conditions such as terrain, while v_t and ϕ_t are burdened by inevitable measurement error (significant for anemometer) and are less relevant as the distance from the power plant increases. In order to combine both measurements for data assimilation, a popular scheme

$$\dot{v}_{t,\mathbf{s}} = a_t \tilde{v}_{t,\mathbf{s}} \tag{3.1}$$

$$\dot{\phi}_{t,\mathbf{s}} = b_t + \tilde{\phi}_{t,\mathbf{s}} \tag{3.2}$$

can be used, where $\dot{v}_{t,\mathbf{s}}$ and $\dot{\phi}_{t,\mathbf{s}}$ are assimilated (most likely true) wind speed and wind direction, a_t is wind speed correction coefficient estimated from v_t and b_t is wind direction correction coefficient estimated using ϕ_t. When we assume that both released radioactive activity in each time step and atmospheric Pasquill's stability category are known, a_t and b_t are the only quantities that need to be assimilated, because progress of the model (e.g. puff coordinates and other parameters) is fully

deterministic then. Because of that, our state variable x_t for sequential Monte Carlo technique is simply $[a_t, b_t]$.

3.2 Measurement Model

Power plant's radioactive monitoring network is our second (and more important) source of real-world observations. Read-outs of total absorbed doses per fixed time interval (10 minutes in our case) from each receptor are available. Putting it together with previous section, we may define observation vector as $[v_t, \phi_t, y_{t,1}, \ldots, y_{t,m}]$, where $y_{t,i}$ is total absorbed dose of i-th receptor per time slot t and m is total number of receptors. With state and observation quantities defined, probabilistic description of the observation model (1.2) can be expressed as (3.3) and factored to conditionally independent densities

$$p(v_t, \phi_t, y_{t,1}, \ldots, y_{t,m}|a_t, b_t) \quad = \quad p(v_t|a_t)p(\phi_t|b_t)p(y_{t,1}, \ldots, y_{t,m}|a_t, b_t). \tag{3.3}$$

Anemometer speed measurements usually have non-biased relative error, denoted as γ_v in this text, therefore it should be modeled using a distribution with mean value $\dot{v}_{t,\mathbf{s}_0}$ and variance $\gamma_v \dot{v}_{t,\mathbf{s}_0}$, where $\mathbf{s}_0$ are spatial coordinates of the anemometer. We've used the inverse gamma density $i\mathcal{G}\left(\gamma_v^{-2} + 2, (\gamma_v^{-2} + 1)\dot{v}_{t,\mathbf{s}_0}\right)$, that fulfills the requirements, for its favorable properties and positive support, hence

$$p(v_t|a_t) = i\mathcal{G}\left(\gamma_v^{-2} + 2, (\gamma_v^{-2} + 1)\dot{v}_{t,\mathbf{s}_0}\right) = i\mathcal{G}\left(\gamma_v^{-2} + 2, (\gamma_v^{-2} + 1)a_t\tilde{v}_{t,\mathbf{s}_0}\right). \tag{3.4}$$

Wind direction is more usually simulated as an unbiased measurement with constant standard deviation σ_ϕ, so the normal density $\mathcal{N}(\dot{\phi}_{t,\mathbf{s}_0}, \sigma_\phi^2)$ was a natural choice, therefore

$$p(\phi_t|b_t) = \mathcal{N}(\dot{\phi}_{t,\mathbf{s}_0}, \sigma_\phi^2) = \mathcal{N}(b_t + \tilde{\phi}_{t,\mathbf{s}_0}, \sigma_\phi^2). \tag{3.5}$$

To derive measurement model for radioactive doses, $p(y_{t,1}, \ldots, y_{t,m}|a_t, b_t)$ can be further factored to conditionally independent densities

$$p(y_{t,1}, \ldots, y_{t,m}|a_t, b_t) = \left(\prod_{i=1}^{m} p(y_{t,i}|d_{t,i})\right) p(d_{t,1}, \ldots, d_{t,m}|a_t, b_t), \tag{3.6}$$

where $d_{t,i}$ is expected dose measured on i-th receptor in time slot t (including expected natural background dose y_{nb}). $p(d_{t,1}, \ldots, d_{t,m}|a_t b_t)$ represents computation of the expected doses using formulas 2.1–2.3 (wind wind field substituted from (3.1), 3.2) and is therefore fully deterministic (i.e. Dirac delta distribution). Parameters of the $p(y_{t,i}|d_{t,i})$ correspond to possibilities of dose measuring devices, and according to [14] their error is proportional to the measured dose with constant of proportionality

γ_y usually between 7–20%. Dose measurements were again simulated using inverse gamma distribution,

$$p(y_{t,i}|d_{t,i}) = i\mathcal{G}\left(\gamma_y^{-2} + 2, (\gamma_y^{-2} + 1)d_{t,i}\right). \tag{3.7}$$

We conclude that equations 3.3–3.7 fully establish used observation model, where by far the most computationally expensive task is to perform 4D integration to compute expected doses in (2.2), (2.3).

3.3 Transition Model

Having only 2 variables in state vector, the transition model is rather simple. Factoring transition density $p(a_t, b_t|a_{t-1}b_{t-1})$ into conditionally independent components

$$\begin{aligned} p(a_t, b_t|a_{t-1}b_{t-1}) &= p(a_t|a_{t-1})p(b_t|b_{t-1}) && (3.8)\\ p(a_t|a_{t-1}) &= \mathcal{G}(\gamma_a^{-2}, \gamma_a^2 a_{t-1}) && (3.9)\\ p(b_t|b_{t-1}) &= \mathcal{N}(b_{t-1}, \sigma_b^2) && (3.10) \end{aligned}$$

we've selected gamma distribution for wind speed correction coefficient with a_{t-1} mean and $\gamma_a a_{t-1}$ variance (3.9), which simulates that the wind speed is distributed around the previous value with relative variance γ_a. Wind direction correction is also assumed to be distributed around the last value, but with constant standard deviation σ_b (3.10).

3.4 Techniques Countering Sample Impoverishment

While transition and observation models as presented in section 3.3 and section 3.2 are sufficient input for the bootstrap variant (the one with naive transition density as its proposal) of the particle filter, it may suffer for a problem where very low N_{eff} causes its degeneration. Various techniques to combat this exist, virtually all of which use proposal density $q(x_t|x_{t-1}, y_t)$ (or its approximation) to optimize sampling phase of the particle filter. This is possible because there's a relative freedom in proposal density choice, as only strict requirement posed on it is that is is not narrower than the actual posterior density.[1] Various techniques exist, to name a few:

1. Auxiliary sampling density[12, 13], which introduces first-stage weights $\lambda_{1:N}$ and modifies the sampling step so that $x_t^{(i)}$ is drawn from $\lambda_i p(x_t|x_{t-1}^{(i)})$, where λ_i should correspond to probability of future particle $x_t^{(i)}$. Optimal choice for

[1] i.e. $\text{supp}(q(x_t|x_{t-1}, y_t)) \supset \text{supp}(p(x_t|y_{1:t}))$

λ_i is said to be $\lambda_i = \int p(y_t|x_t)p(x_t|x_{t-1}^{(i)})$, which can be approximated using $\lambda_i \approx p(y_t|\mu_t^{(i)})$, where $\mu_t^{(i)}$ is mean, mode or other significant value of $p(x_t|x_{t-1}^{(i)})$.

2. Gradient-based methods, where $x_t^{(i)}$ is sampled from naive proposal, but before use shifted to a region with higher likelihood.[13]

3. Approximation of the optimal proposal density[3] (3.11) using Taylor series and its subsequent fitting to a known probability density function.[16]

4. Adaptive importance sampling, where the proposal density is parametrized as $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \theta)$ and the parameter θ is estimated using several smaller populations of particles.[11, 16] This approach can be called population Monte Carlo method.

5. Analytic derivation of the optimal proposal density (3.11). Often intractable, but it can happen that the transition density $p(x_t|x_{t-1})$ and the observation density $p(y_t|x_t)$ are so-called conjugate (under given parametrization), together giving a probability density function from a known family. In this case the denominator of 3.11 doesn't need to be computed, because the normalizing constant is known from theory. We call this method a *conjugate* proposal density.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(y_t|x_{t-1})} = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{\int p(y_t|x_t)p(x_t|x_{t-1})\,\mathrm{d}x_t} \tag{3.11}$$

Because our model exhibits favorable conjugate properties (under certain simplification), we've chosen the conjugate proposal density (which in theory gives the best results as other methods are just some kind of its approximation), the form of which is derived below. In chapter 4 we compare it to the naive proposal, whose undeniable advantage is algorithmic simplicity and straightforward ability to improve accuracy by increasing the number of particles N.

3.5 Conjugate Proposal

For our choice of the gamma transition density for a_t (3.9) coupled with its inverse gamma observation density (3.4), normal transition density for b_t (3.10) coupled with its normal observation density (3.5) and ignoring dose measurements $\mathbf{y}_t$ in the observation, the optimal sampling density can be approximated using formula

$$\begin{aligned} q(x_t|x_{t-1}, y_t) &= p(a_t, b_t|a_{t-1}, b_{t-1}, v_t, \phi_t, \mathbf{y}_t) \\ &\approx p(a_t|a_{t-1}, v_t)p(b_t|b_{t-1}, \phi_t). \end{aligned} \tag{3.12}$$

Proposal densities for wind speed correction coefficient and wind direction correction coefficient in 3.12 are conditionally independent, and are derived below.

3.5.1 Wind Speed Conjugate Density

In order to derive formula for $p(a_t|a_{t-1}, v_t)$, let us first show general (and well known) result for for inverse gamma density and gamma density with appropriate parametrization. Suppose that $p(y|a) = i\mathcal{G}\left(y; \alpha, (\alpha-1)\, a\right)$, $p(a) = \mathcal{G}(a; k, \theta)$ and that y is known. Then $p(a|y)$ is also a gamma distribution and can be obtained using the Bayes rule (3.13), yielding (3.15). Note that the transition from (3.14) to (3.15) makes use of the fact that the whole term in (3.14) is a probability density function, therefore it must integrate to 1; however, the non-constant parts (with regards to quantity a) have the form of the gamma distribution, therefore the whole term must necessarily be a gamma distribution itself

$$
\begin{aligned}
p(a|y) &= \frac{p(y|a)p(a)}{p(y)} && (3.13)\\
&= \frac{1}{p(y)} i\mathcal{G}\left(y; \alpha, (\alpha-1)\, a\right) \mathcal{G}(a; k, \theta)\\
&= \frac{1}{p(y)} \frac{\left((\alpha-1)\, a\right)^{\alpha}}{\Gamma(\alpha)} y^{-\alpha-1} \exp\left(-\frac{(\alpha-1)\, a}{y}\right) \frac{1}{\theta^k \Gamma(k)} a^{k-1} \exp\left(-\frac{a}{\theta}\right)\\
&= \frac{(\alpha-1)^{\alpha}\, y^{-\alpha-1}}{p(y)\Gamma(\alpha)\, \theta^k \Gamma(k)} a^{\alpha+k-1} \exp\left(-a\left((\alpha-1)\, y^{-1} + \theta^{-1}\right)\right)\\
&= C\left(y, \alpha, \theta, k\right) a^{\alpha+k-1} \exp\left(-a\left((\alpha-1)\, y^{-1} + \theta^{-1}\right)\right) && (3.14)\\
&= \mathcal{G}\left(a; \alpha+k, \left((\alpha-1)\, y^{-1} + \theta^{-1}\right)^{-1}\right). && (3.15)
\end{aligned}
$$

In our case

$$
\begin{aligned}
a &= a_t \tilde{v}_t\\
y &= v_t\\
p(y|a) &= p(v_t|a_t\tilde{v}_t)\\
&= i\mathcal{G}\left(v_t; \gamma_v^{-2} + 2, (\gamma_v^{-2}+1) a_t \tilde{v}_t\right)\\
p(a) &= p(a_t\tilde{v}_t|a_{t-1})\\
&= \mathcal{G}(a_t\tilde{v}_t; \gamma_a^{-2}, \gamma_a^2 a_{t-1}\tilde{v}_t)\\
p(a|y) &= p(a_t\tilde{v}_t|a_{t-1}, v_t)\\
&= \mathcal{G}\left(a_t\tilde{v}_t; \gamma_v^{-2} + \gamma_a^{-2} + 2, \left(\left(\gamma_v^{-2}+1\right) v_t^{-1} + \gamma_a^{-2} a_{t-1}^{-1} \tilde{v}_t^{-1}\right)^{-1}\right)\\
p(a_t|a_{t-1}, v_t) &= \mathcal{G}\left(a_t; \gamma_v^{-2} + \gamma_a^{-2} + 2, \left(\left(\gamma_v^{-2}+1\right) v_t^{-1}\tilde{v}_t + \gamma_a^{-2} a_{t-1}^{-1}\right)^{-1}\right)
\end{aligned}
$$

3.5.2 Wind Direction Conjugate Density

For the choice of wind direction transition density $p(b_t|b_{t-1}) = p(\phi_t|b_t, b_{t-1}) = \mathcal{N}(b_t; b_{t-1}, \sigma_b)$ and wind direction observation density $p(\phi_t|b_t) = \mathcal{N}(\phi_t; \tilde{\phi}_t + b_t, \sigma_\theta)$, the conjugate proposal can be derived as follows:

$$\begin{aligned}
p(b_t|b_{t-1}, \phi_t) &= \frac{p(\phi_t|b_t, b_{t-1})p(b_t|b_{t-1})}{p(\phi_t|b_{t-1})} \\
&= \frac{\mathcal{N}(\phi_t; \tilde{\phi}_t + b_t, \sigma_\theta)\mathcal{N}(b_t; b_{t-1}, \sigma_b)}{p(\phi_t|b_{t-1})} \\
&= \frac{1}{p(\phi_t|b_{t-1})} \frac{1}{2\pi} \frac{1}{\sigma_\theta} \exp\left(-\frac{1}{2}\left(\frac{\phi_t - \tilde{\phi}_t - b_t}{\sigma_\theta}\right)^2\right) \frac{1}{\sigma_b} \exp\left(-\frac{1}{2}\left(\frac{b_t - b_{t-1}}{\sigma_b}\right)^2\right) \\
&= \mathcal{N}\left(\left(\sigma_b^{-2} + \sigma_\phi^{-2}\right)^{-1}\left(\sigma_b^{-2} b_{t-1} + \sigma_\phi^{-2}\left(\phi_t - \tilde{\phi}_t\right)\right), \left(\sigma_b^{-2} + \sigma_\phi^{-2}\right)^{-1}\right).
\end{aligned}$$

Chapter 4

Results

This chapter starts by constructing so-called *twin experiment*, where the 2 selected Sequential Monte Carlo technique variants are benchmarked on a simulation. We are mainly concerned with computational demands needed to achieve given expectation of accuracy. After experiment results are shown are interpreted, some remarkable implementation details are mentioned. The chapter is ended by suggestions for future improvements, extensions and application.

4.1 Twin Experiment

With complete atmospheric dispersion model in place, we are able to perform the twin experiment. We've used following steps:

1. Choose time step, duration of the experiment T, $t \in [1, T]$, atmospheric parameters, namely numerical model wind field predictions $\tilde{v}_{1:T,\mathbf{s}}, \tilde{\phi}_{1:T,\mathbf{s}}$ and local wind field $\dot{v}_{1:T}, \dot{\phi}_{1:T}$ with slight variations from the global prediction, Pasquill's stability category and finally released radioactive pollutant activities per time slot $Q_{1:T}$.

2. Perform simulation with chosen parameters, compute and save simulated absorbed doses per each receptor and time slot $\dot{y}_{t,i} \quad i \in \{1, \ldots, m\}$.

3. Add noise to measurements. Simulate anemometer error by drawing v_t from $p(v_t|\dot{v}_t)$, ϕ_t from $p(\phi_t|\dot{\phi}_t)$; same distributions and parameters as in (3.4) and (3.5) (measurement model) were used, i.e. v_t has relative variance of γ_v around actual value and ϕ_t has standard error σ_ϕ around its mean. Add natural background dose per time step y_{nb} to dose measurements and simulate receptor noise: sample $y_{t,i}$ from $p(y_{t,i}|\dot{y}_{t,i} + y_{nb})$ with same parameters as in (3.7) — unbiased measurement with γ_y relative variance.

4. Choose total number of particles N. Perform assimilation using selected variants of the particle filter. The filter has access to numerically-predicted wind field $\tilde{v}_{1:T,\mathbf{s}}, \tilde{\phi}_{1:T,\mathbf{s}}$, Pasquill's stability category and released activities $Q_{1:T}$, but actual weather and dose measurements are hidden by noisy quantities $v_t, \phi_t, y_{t,i}$.

Following properties of filtering techniques were evaluated:

- Convergence to simulated release course. While convergence as N approaches infinity is proven, for realistic N choices it needs to be checked.
- Number of effective particles for each step $N_{\text{eff}}(t)$ (1.7). If for example $N_{\text{eff}} = 0.05\,N$, it means that 95% of resources were wasted computing particles with negligible weight (that virtually don't affect resulting posterior density). Certain overhead is inevitable in the particle filter, though.
- Computational costs $C(t)$: approximate number of CPU seconds[1] needed to perform assimilation step from $t-1$ to t.

When particle filtering improvements are suggested, a valid question comes to one's mind: Wouldn't simply increasing the particle count N have similar effects (increasing N_{eff} and to some extent required resources) while keeping the simple algorithm? To answer this, we propose to measure *effective particles per second*, i.e. $\frac{N_{\text{eff}}(t)}{C(t)}$.

4.1.1 Assimilation Model Setup

For our purposes we've chosen assimilation time step of 10 minutes,[2] and total simulation length of 4 hours. North-East wind with minor time-varying deviation in direction and with steady $2\,\mathrm{m/s}$ wind speed was used, while numerical prediction with strictly North-East wind passing at $2.1\,\mathrm{m/s}$ was assumed. Radioactive release happened through first hour of the simulation with 10-minute activities of $[1, 5, 4, 3, 2, 1] \times 10^{16}\,Bq$, dispersing in atmosphere according to Pasquill's stability category D (neutral).

Observation model was tuned as follows: anemometer parameters were assumed to be $\gamma_v = 0.1$ (wind direction relative variance) and $\sigma_\phi = 5°$ standard deviation for wind direction. Relative error of radioactive receptors was chosen as $\gamma_y = 0.2$ in accordance with [14].

For transition model $\gamma_a = 0.2$ and $\sigma_b = 15°$ was employed. $N = 1000$ particles was fixed for all particle filter runs.

[1] a measure provided by the operating system that slightly differs from wall-clock time: it excludes time when given CPU was tasked with different OS processes

[2] because studied power plant radiation monitoring network measures dose in 10-minute intervals

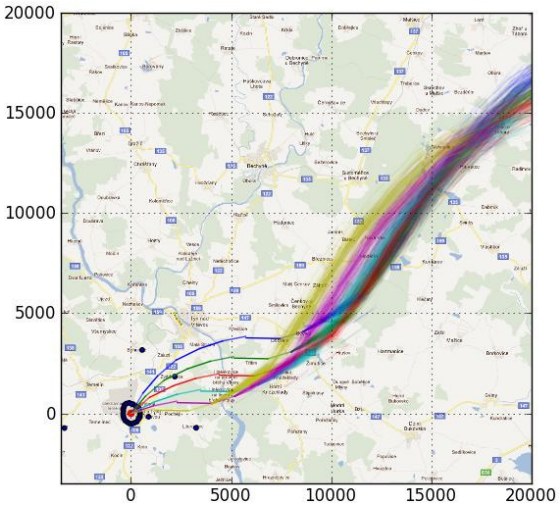

Figure 4.1: Assimilation, naive proposal

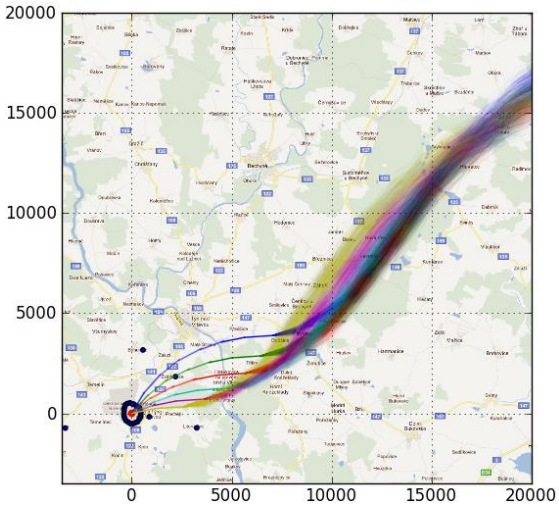

Figure 4.2: Assimilation, conjugate proposal

4.2 Naive vs. Conjugate Proposal

Figure 4.1 and Figure 4.2 shows assimilation results for particle filter with naive, respectively conjugate proposal. Only puff trajectories are shown, total ground doses would be straightforward to compute and are not considered; each color represents trajectory of one puff, each line represents one particle in the particle filter, its opacity is proportional to its weight. Thicker bunches of one color at given time therefore signify higher uncertainty (variance) in trajectory estimation of that puff. Comparing assimilated trajectories with simulation (Figure 2.2) that was data source for the twin experiment, we can safely state that both assimilation techniques succeeded in tracking the radioactive plume. Furthermore, trajectories obtained using the naive and conjugate proposal are visually indistinguishable.

One interesting effect can be seen on both figures: approximately the first 70 minutes the variance of trajectories is extremely low and both particle filters assign negligible weights to all-but-one one particle. This coincides with the phase when the puffs can be observed using dose measurements from radioactive monitoring network receptors (denoted by blue dots in the figures). Our explanation is that this is due to dose measurement model being extremely non-linear, where a small (tens of meters) shift in puff trajectory causes dramatic change in assigned likelihood. In other words, the dose part of the observation model is extremely tight-peaked compared to the transition model. To confirm this claim, we've run the assimilation again, but this time the dose measurements weren't supplied to the filters. Results of these filters are shown in Figure 4.3 and Figure 4.4. Indeed, trajectory variance

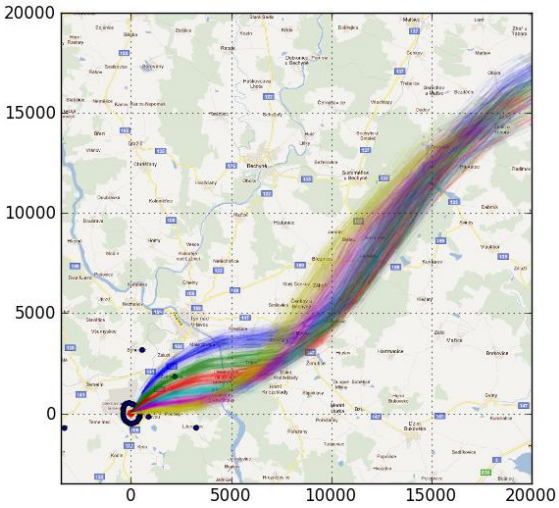

Figure 4.3: Assimilation, naive proposal, dose ignored

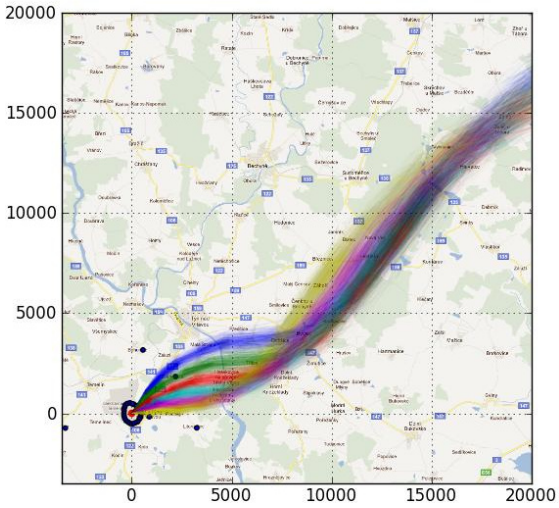

Figure 4.4: Assimilation, conjugate proposal, dose ignored

raised right from the beginning there and continued to be slightly greater through the rest of the experiment compared to dose-aware assimilation.

Second examined property was number of effective particles (shown in Figure 4.5) out of 1000 total and number of computed effective particles per second shown in Figure 4.6. In the first phase (again lasting around 70 minutes, or 7 time steps) both filters struggle to sample significant number particles to the area where they would gain weight and their N_{eff} is close to 1. Even the filter with conjugate proposal, that takes anemometer measurements into account when drawing particles, suffers from the sample impoverishment problem. We argue that this is because even the improved proposal density has much greater variance than the observation model (dose part of it). The second phase sees sharp rise of the effective particles, which is especially pronounced for the case of the conjugate proposal. While this is expected, we consider the second figure much more important; indeed, it answers the main question we've posed: *conjugate proposal (when given sufficient information)*[3] *yields significantly higher* N_{eff} *even when additional complexity is accounted for.* As can be seen, number of effective particles computed per CPU second oscillates roughly around 200 p/s for the conjugate variant of the filter, and around 100 p/s for the naive variant.

[3]by sufficient information we mean the measurements most significant for the observation model; this was not the case for the first phase

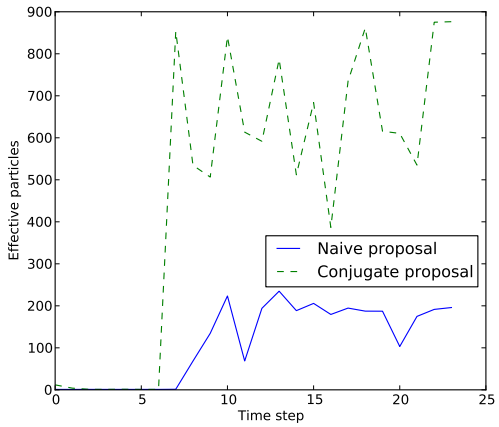

Figure 4.5: Number of effective particles

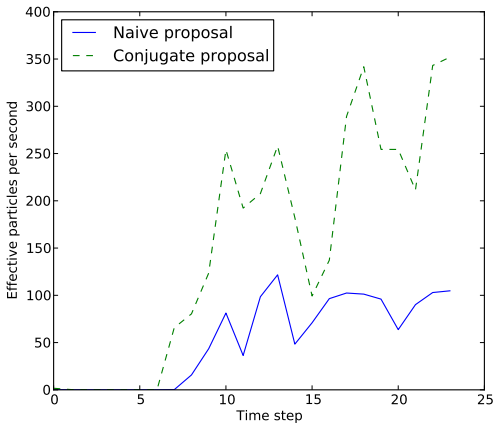

Figure 4.6: Effective particles per second

4.3 Implementation Decisions

One of the goals of this work was to establish a solid software framework for assimilation of the radioactive pollutant release, a basis for planned future research in this field and projects such as web-based assimilation lab accessible by interested public or ultimately a real-time continuous assimilation using actual real-world meteorologic forecast and measurements from power plant radioactive monitoring network and meteorologic station. Object-oriented approach using the Python Programming Language[4] was employed along with Cython[1] to optimize performance-critical code paths. For discussion surrounding this decision see [10].

The software project was named *Asim*, it's source tree and API documentation are available on the enclosed DVD-ROM, see Appendix A.

4.3.1 Dispersion Model

Task of the dispersion model is to simulate atmospheric release and compute quantities such as absorbed dose or pollutant concentration at given spatial coordinates; it needs to work with various sources of wind field data and pollutant release data. We propose following interface (simplified, only important methods shown):

`DispersionModel` stateful class tracking radioactive plume in the atmosphere.

> `propagate(meteo_model, source_model)` transition from $t-1$ time step to t according to meteorological situation passed through `meteo_model` and released activity passed through `source_model`.

[4] www.python.org

`vector concentration_at(loc)` compute per-nuclide concentration at given coordinates `loc`.

`vector dose_at(loc)` compute per-nuclide absorbed dose at coordinates `loc`.

`MeteoModel` stateless class describing meteorologic situation.

`real wind_speed_at(loc, time)` return wind speed at given location and time.

`real wind_direction_at(loc, time)` return wind speed at given location and time.

`real dispersion_xy(loc, time, total_distance)` return horizontal dispersion coefficients (σ_x, σ_y in (2.1)) for a plume that has already flown `total_distance`.

`real dispertion_z(loc, time, total_distance)` return vertical dispersion coefficient (σ_z in (2.1)) for a plume that has already flown `total_distance`.

`SourceModel` stateless class describing source term of the radioactive release.

`list[Nuclide] inventory()` return a list of radionuclides contained in the release.

`vector release_rate(time)` return release rate in Bq/s per each radionuclide in inventory at given `time`.

`Location location()` return coordinates of the release source.

`Nuclide` container for parameters of a radionuclide such as half-life.

Dispersion model was then implemented using the mathematical puff model in 2 classes: `Puff` (`DispersionModel` subclass) that describes one puff in the plume; `concentration_at()` is implemented using (2.1) and dose_at() using (2.2). `PuffModel` (also `DispersionModel` subclass) is then a collection of puffs with additive implementation of `concentration_at()` an `dose_at()` enriched with the logic to create new puffs when non-zero activity is released and destroy old puffs whose activity has fallen below a certain threshold. For the purpose of simulation, `MeteoModel` and `SourceModel` implementations that return precomputed values were made use of.

4.3.2 Assimilation

Data assimilation was implemented on top of the PyBayes library.[5][10] For this the PyBayes library was extended with implementation of the gamma distribution,

[5] freely available from https://github.com/strohel/PyBayes

inverse gamma distribution and common conditional variants thereof. Additionally, the particle filter from PyBayes was taught to make use of the proposal distribution in form of another Bayesian filter. Motivation for this is 2-sided: *a)* the proposal density has the form $q(x_t|x_{t-1}, y_t)$, which coincides with the posterior density of a Bayesian filter (1.4); it has sense to plug the posterior density of for example a Kalman filter as a proposal density approximation into the particle filter, *b)* another option would be to pass a conditional probability density function directly, but that would involve (potentially expensive) double evaluation of its condition under current PyBayes constraints.

The side of the Asim project consisted of creating custom conditional probability density functions for the observation model (3.3) and transition model (3.8) to be used by PyBayes' particle filter. This involved creating `AssimilationMeteoModel` (`MeteoModel` subclass), which performs transformations (3.1) and (3.2). `SourceModel` from simulation was reused, but once released activity Q is assimilated, it will be replaced by `AssimilationSourceModel`.

4.3.3 Optimization Using Cython

While programmer convenience, fast prototyping and availability of many supplemental 3rd party packages are appreciated when Python is used, it is well-known fact that Python itself is badly suited for expensive numerical computations. The Cython project allows one to compile Python code into C code which can subsequently be compiled into binary Python extension modules (technically dynamically-loaded libraries: .so files on UNIX and .dll files on Windows platforms) that behave very similarly to interpreted .py modules, but needn't be interpreted by the Python interpreter.[1] This provides moderate speedup of 0% to 300% because it eliminates the interpreter overhead. However, much more dramatic improvement can by achieved by statically typing[6] variables that are part of the critical code paths.

Static typing is usually achieved by extending the Python language, which breaks compatibility. However, Cython offers so-called pure Python mode, where the implementation is written in strict Python language and *augmented* using .pxd files that contain static typing declarations. This allows the programmer to prototype quickly using Python (perhaps testing on a reduced-size problem) and once happy with the algorithm, use Cython to achieve required performance.

Profiling our code we've identified that over 97% of the time was spent computing the 4D integration (2.2). All involved variables were statically typed, and after compiling using Cython, we've noted execution time drop from 651 s to mere 3.53 s, a 184 × speedup. This is especially pronounced in the assimilation experiment,

[6] Python itself is a dynamically-typed language: a variable can arise with different incompatible data types throughout its lifetime

where the simulation is run N times with $N = 1000$ in our case; this is the reason why assimilation wasn't even tested under the pure Python mode.

It has been showed that Cython-optimized numerical code performance is comparable to equivalent C code.[10] The PyBayes project already comes with a Cython version.

4.4 Discussion

We've achieved moderate success showing favorable properties of the sequential Monte Carlo technique enhanced with conjugate proposal density. On the other hand, we were unable to attain improvement (with regards to number of effective particles) in the first phase of the experiment where the dose measurements dominate the observation model with our approximation of the optimal proposal density. In our opinion, this is mainly due to the fact that our variant cannot account for dose measurements. We propose following techniques to be studied to improve efficiency:

- incorporating dose observation into proposal density. This natural approach is not straightforward to achieve, as inverting dose observation model 3.6 is not analytically tractable in contrast to the wind field model. We propose extending the proposal density with gradient techniques for finding local maxima of the dose observation density.

- shortening the assimilation time step. In our view the 10-minute step is too rough and variance in wind direction causes poor results when simulated doses are compared against measured ones. This approach is unfortunately unachievable with current monitoring network infrastructure, and furthermore its gains questionable given that shorter time step could cause greater error in receptor read-outs.

It should be further noted that assimilation problem described here is a rather simplified version of the problem faced in real life. We assume that released activities, atmospheric stability category and distribution of radionuclides in the release is known. This is not the case in most situations. [16] already adds released activity Q to the list of assimilated quantities.

Another outcome of this text is the finding that the radioactive monitoring network provides the most valuable information for tracking the toxic plume. On the other hand, dose receptors are laid very scarcely in areas more distant from the nuclear site. If there were more sensors in the area, our uncertainty about the plume trajectory would be much lower.

Finally, we've implemented presented algorithms using clean object-oriented approach with future extensions in mind. Existing PyBayes library that was developed

earlier was used for algorithms of Bayesian estimation, and Cython Python-to-C compiler was employed with great success to increase performance of the Python code to levels comparable with low-level languages.

Conclusion

After starting with a brief review of some known results from the recursive Bayesian estimation with accent on the particle filter (sequential Monte Carlo method), we've presented hypothetical but realistic scenario of an atmospheric radioactive pollutant release. The puff model was chosen to simulate it for its relative simplicity and adequateness for particle filter estimation, and is briefly described.

In the next chapter, assimilation model is inferred using the release scenario and described in more detail. Continues a discussion about techniques to fight the sample impoverishment problem, a phenomenon that degrades quality of the estimation especially noticeable in situations where the transition model (probability density function thereof) is significantly broader than the observation model — the case we've faced. The problem is most usually solved using appropriate choice of the proposal density: the naive proposal was set to be benchmarked against a so-called conjugate proposal, whose form was subsequently derived for our choice of the wind field transition and observation model.

Then it was shown that both particle filter variants are able to successfully track the plume trajectory, however the emphasis was put on computational resources needed by the filter variants to achieve a certain level of accuracy expectation. For this purpose a measure of effective particles per second was suggested. The conjugate proposal technique was recorded to achieve roughly double the rate during the second phase of the assimilation, and declared well worth the complexity it adds. On the other hand, both variants suffered equally from the sample impoverishment problem within approximately first 70 minutes of the simulated release; discussion that followed pointed to ignorance of the dose measurements in the chosen proposal as the main cause.

Finally, some implementation details are revealed and interface of the core classes employed in an object-oriented software design is shown; these form the basis of the Asim software project whose implementation was started. A notable speedup of the code written in Python was achieved by engaging the Cython Python-to-C compiler.

If we were to identify 2 main contributions of this work, it would be according to us:

- Performed comparison of the naive and conjugate particle filter proposal den-

sities with regards to resources needed to compute one "effective particle" performed under a realistic and demanding dispersion model assimilation.

- The implementation of the radioactive release assimilation and simulation framework itself (coupled with enhancements of the PyBayes library). We believe that it can be extended painlessly in future to cope with larger-scale problems.

Speaking about future work, we suggest to extend the assimilation to more quantities to be more useful for actual emergency situations. This includes assimilation of released activities (already shown in [16]) and distribution of the radionuclides within the release. Furthermore the model of weather-related variables as presented here was rather simplistic.

While the performance of the assimilation seems near-optimal on a uniprocessor machine thanks to Cython, parallelization is still being worked on and was not yet employed in results shown in this text.

List of Figures

Bibliography

[1] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31–39, march-april 2011. 28, 30

[2] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *Signal Processing, IEEE Transactions on*, 50(3):736–746, 2002. 13

[3] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001. 12, 13, 21

[4] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, apr 1993. 10, 12

[5] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.J. Nordlund. Particle filters for positioning, navigation, and tracking. *Signal Processing, IEEE Transactions on*, 50(2):425–437, 2002. 9

[6] R. Hofman, V. Šmídl, and P. Pecha. Data assimilation in early phase of radiation accident using particle filter. In *The Fifth WMO International Symposium on Data Assimilation*, Melbourne, Australia, 2009. 9

[7] R. Hofman and Šmídl V. Assimilation of spatio-temporal distribution of radionuclides in early phase of radiation accident. *Bezpečnost jaderné energie*, 18:226–228, 2010. 9

[8] G. Johannesson, B. Hanley, and J. Nitao. Dynamic bayesian models via monte carlo–an introduction with examples. Technical report, Lawrence Livermore National Laboratory, 2004. 15

[9] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME — Journal of Basic Engineering*, 82(Series D):35–45, 1960. 11

[10] M. Laitl. Implementation environment for Bayesian filtering algorithms. Bachelor thesis, Faculty of Nuclear Sciences and Physical Engineering of the Czech Technical University in Prague, Czech Republic, 2011. 28, 29, 31

[11] M.S. Oh and J.O. Berger. Adaptive importance sampling in Monte Carlo integration. *Journal of Statistical Computation and Simulation*, 41(3):143–168, 1992. 21

[12] Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599. 20

[13] M. Simandl and O. Straka. Sampling densities of particle filter: A survey and comparison. In *American Control Conference, 2007. ACC '07*, pages 4437–4442, july 2007. 20, 21

[14] IMG Thompson, CE Andersen, L. Bøtter-Jensen, E. Funck, S. Neumaier, and JC Sáez-Vergara. An international intercomparison of national network systems used to provide early warning of a nuclear accident having transboundary implications. *Radiation protection dosimetry*, 92(1-3):89, 2000. 19, 25

[15] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. 2005. 9

[16] Šmídl V. and R. Hofman. Efficient sequential monte carlo sampling for continuous monitoring of radiation situation. *Technometrics*, submitted, 2012. 14, 15, 18, 21, 31, 34

Appendix A

Contents of the Enclosed DVD-ROM

asim/ source code tree of the Asim project

- **assimilaiton**/ folder with implementation of the assimilation model and related classes; pure Python version of this module was not tested because of substantial time requirements it would cosume
- **dispmodel**/ folder with dispersion model interfaces and puff model implementation
- **doc**/ documentation source code
- **simulation**/ implementation of the pollutant release simulation
- **support**/ folder with supportive code such as graph plotters and math routines
- **README.rst** text file with instructions how to build and run the project
- **setup.py** Python distutils script used to build/install the project

asim-doc.pdf API documentation for the Asim project in PDF format

PyBayes/ source code tree of the PyBayes project, bundled for convenience

PyBayes-doc.pdf PyBayes API documentation in PDF format, bundled for convenience

text.pdf this text itself in navigable form