Particle Predictive Control

J.P. de Villiers[†], S.J. Godsill[‡], S.S. Singh[*]

[†] *Department of Electrical, Electronic and Computer Engineering, University of Pretoria, 0002, South Africa*
[‡], [*] *Department of Engineering, University of Cambridge, Trumpington Street, CB2 1PZ, United Kingdom*
email: [†]*pieter.devilliers@up.ac.za,* [‡]*sjg@eng.cam.ac.uk,* [*]*sss40@eng.cam.ac.uk*

Abstract

This work explores the use of sequential and batch Monte Carlo techniques to solve the Nonlinear Model Predictive Control (NMPC) problem with stochastic system dynamics and noisy state observations. This is done by treating the state inference and control optimisation problems jointly as a single artificial inference problem on an augmented state-control space. The methodology is demonstrated on the benchmark car-up-the-hill problem as well as an advanced F-16 aircraft terrain following problem.

Keywords: Model predictive control, Moving horizon control, Stochastic control, Markov Chain Monte Carlo, Particle filter, SAME algorithm

1. Introduction

In conventional NMPC control one computes the optimal control sequence over a finite future time interval (or horizon) consisting of several discrete time steps based on the current state estimate. The control is then executed for only a single discrete time step, after which the time variable is incremented, the new state is either partially or fully observed, the state estimate is updated, and the whole process is repeated (Findeisen and Allgöwer (2002)). In the control computation step, a point estimate of the system's initial state is obtained whereafter the system's future state trajectory given the estimated initial state is optimised. This two-step approach may be questionable since it does not take into consideration the uncertainty of the initial and predicted future states.

The approach followed here differs from the conventional approach in that state uncertainty is properly accounted for. The control optimisation step itself is treated as

an inference problem by augmenting the initial state distribution with an artificial distribution over the set of potential controls. The Maximum A-posteriori (MAP) control estimate of this augmented model coincides with the optimal control sequence. A simple Markov Chain Monte Carlo (MCMC) algorithm is devised to try to approximate the MAP control estimate, similar to the procedure proposed by Müller (1999) and Doucet et al. (2002). The aim of this work is to demonstrate that computational techniques for Bayesian inference are also effective tools for solving this class of control problems. This is important because it provides a principled alternative method for solving more general control problems lacking the smoothness required for continuous optimisation techniques to be applied. Additionally, the presented Bayesian techniques are parallelisable, which means they can be in some cases significantly sped-up with dedicated computational hardware (Lee et al. (2010).) This paper presents a numerical study of two examples. One is a benchmark problem in the Machine Learning literature known as the car-up-the-hill problem. The objective here is to drive an underpowered car starting at rest in a valley up a hill towards a goal position. The second is a challenging terrain-following task where an F-16 aircraft is to be navigated and kept as close as possible to a reference trajectory.

The remainder of this paper is organised as follows. Section 2 presents the control problem, which is comprised of a controlled state-space model and a performance criterion. The augmented statistical model whose MAP estimate coincides with the optimal controller is presented in Section 3, together with an MCMC algorithm to approximate this MAP estimate. A brief mention of constraints and alternative implementations is presented in Section 4. Numerical examples are presented in Section 5 and the subsequent section contains the concluding remarks.

2. The statistical model

Consider the discrete time state-space model of a controllable or affectable system

$$x_k = \phi(x_{k-1}, u_{k-1}, v_k), \tag{1}$$

$$y_k = \psi(x_k, w_k), \tag{2}$$

where k denotes the discrete-time index, ϕ characterises the state transition, v_k is the random disturbance in the dynamics, u_k denotes the control variable, ψ characterises the observation process and w_k denotes the random variable characterising the observation uncertainty. The state, observation and control could be real scalar or vector valued. Note that the system of equations (1) and (2) can also be used to model a discretised continuous dynamic system, where the function ϕ denotes the numerical integration function of the underlying continuous system. This approach is used for the continuous time examples in this paper.

The disturbance v_k together with the function ϕ determine the transition probability density

$$p(x_k|x_{k-1}, u_{k-1}), \tag{3}$$

where the notation $p(x_k|x_{k-1}, u_{k-1})$ denotes the probability density of the random variable x_k given the variables x_{k-1} and u_{k-1}. The hidden state model is completed by a prior distribution $p(x_0)$ over initial states which encapsulates the observer's a-priori knowledge regarding the initial state. As a consequence of the Markov property, the joint conditional state density up to time k is given by

$$p(x_{0:k}|u_{0:k-1}) = p(x_0) \prod_{j=1}^{k} p(x_j|x_{j-1}, u_{j-1}) \tag{4}$$

where $x_{0:k}$ denotes $x_0, \ldots, x_k$. Note that the density in (4) is parameterised by the sequence of controls applied at the previous decision epochs.

The observations are corrupted by the random variables w_k, which along with the function ψ determine the likelihood:

$$p(y_k|x_k). \tag{5}$$

$p(y_k|x_k)$ is the probability density of the observation at time k given the hidden state is x_k. It is assumed that conditioned on $x_{0:k}$, the observations $y_{0:k}$ are independent in the

following sense

$$p(y_{0:k}|x_{0:k}) = \prod_{j=0}^{k} p(y_j|x_j). \tag{6}$$

Using Bayes' theorem, the joint posterior over the whole state sequence up to decision epoch k, conditional on the sequence of controls and observations, is given by

$$p\left(x_{0:k}|y_{0:k}, u_{0:k-1}\right) = \frac{p\left(x_{0:k}|u_{0:k-1}\right) p\left(y_{0:k}|x_{0:k}\right)}{p\left(y_{0:k}|u_{0:k-1}\right)}.$$

The *filter* at time k is the marginal $p(x_k|y_{0:k}, u_{0:k-1})$ of the above posterior density. In the NMPC framework, only the filter is required at each control epoch. Also, in the NMPC and other sequential decision settings, it is of interest to update the filter recursively as new observations become available. As a consequence of the Markov assumption, and the independence of observations, the filter can be updated recursively using a two step procedure. The prediction step is given by

$$p(x_k|y_{0:k-1}, u_{0:k-1}) = \int p(x_k|x_{k-1}, u_{k-1}) p(x_{k-1}|y_{0:k-1}, u_{0:k-2}) dx_{k-1}, \tag{7}$$

and the observation update is

$$p(x_k|y_{0:k}, u_{0:k-1}) = \frac{p(y_k|x_k) p(x_k|y_{0:k-1}, u_{0:k-1})}{p(y_k|y_{0:k-1}, u_{0:k-1})}. \tag{8}$$

If random sampling methods are used, it is sufficient to calculate the filtering density up to a constant scaling, i.e. ignoring $p(y_k|y_{0:k-1}, u_{0:k-1})$.

A sequential Monte Carlo procedure, also known as a particle filter, is an importance sampling algorithm for generating samples from the filter $\{p(x_k|y_{0:k}, u_{0:k-1})\}_{k\geq 1}$ specified in (8). The basic algorithm (Gordon et al. (1993)) is shown below and is comprised of three main steps. The first is simulating samples, termed particles, forward in time according to some chosen transition probability densities $\{f_k(x_k|x_{k-1})\}$. (See Step 0 and Step 2.) For this step a trivial choice is $f_k(x_k|x_{k-1}) = p(x_k|x_{k-1}, u_{k-1})$. One could also choose f_k so that the variance of the importance weights is minimised (Doucet (1998)). Doing so would require $f_k(x_k|x_{k-1})$ to incorporate knowledge of the most recent observation y_k. The second step is to correct for the discrepancy between the target and the proposal via

importance sampling. (See the end of step 0 and step 2.) The third step is resampling the particles. (See Step 1.)

- Step 0. Set $k = 0$. For $i = 1, \ldots, I$ sample $\widetilde{x}_0^{[i]} \sim f_0(x_0)$ and compute $w_0^{[i]} \propto p(y_0|\widetilde{x}_0^{[i]})p(\widetilde{x}_0^{[i]})/f(\widetilde{x}_0^{[i]})$, $\sum_{i=1}^{I} w_0^{[i]} = 1$.

- Step 1. For $i = 1, \ldots, I$ resample particle i from the set $\{\widetilde{x}_k^{[j]} : j = 1, \ldots, I\}$ where particle j is selected with probability $w_k^{[j]}$. Call the resampled particle i $x_k^{[i]}$ and reset its weight to $w_k^{[i]} = I^{-1}$. Set $k = k + 1$.

- Step 2. For $i = 1, \ldots, I$ sample $\widetilde{x}_k^{[i]} \sim f_k(x_k|x_{k-1}^{[i]})$ and compute

$$w_k^{[i]} \propto \frac{p(y_k|\widetilde{x}_k^{[i]})p(\widetilde{x}_k^{[i]}|x_{k-1}^{[i]}, u_{k-1})}{f_k(\widetilde{x}_k^{[i]}|x_{k-1}^{[i]})},$$

  $\sum_{i=1}^{I} w_k^{[i]} = 1$ and return to Step 1.

It can then be shown that for any integrable function $x_k \to \varphi(x_k) \in \mathbb{R}$, the following integral

$$\int \varphi(x_k)p(x_k|y_{0:k}, u_{0:k-1})dx_k$$

can be estimated using the particle filter:

$$\sum_{i=1}^{I} \varphi(\widetilde{x}_k^{[i]})w_k^{[i]}. \tag{9}$$

This estimate converges almost surely under minimal technical conditions (Del Moral (2004)). It is even possible to use the equally weighted particle set available after the resampling step. From this point onwards it is assumed that it is possible to sample from the transition probability density (3). The particle approximation of (8) is denoted by $\{x_k^{[i]}\}_{i=1}^{I}$ after resampling and $\{\widetilde{x}_k^{[i]}, w_k^{[i]}\}_{i=1}^{I}$ before resampling.

3. The augmented model, particle predictive controller and computation

The future (predicted) state trajectory under the control trajectory $\mathbf{u} = (u_k, u_{k+1}, \ldots, u_{k+N-1})$ is denoted by $\mathbf{x} = (x_k, x_{k+1}, \ldots, x_{k+N})$, where N denotes the control horizon length. The

objective of the model predictive controller is to optimise $\mathbf{u}$ according to some criterion which penalises the deviation of $\mathbf{x}$ from "desired trajectories." In order to achieve this, a joint probability distribution is introduced for $\mathbf{x}$ and $\mathbf{u}$ and the optimal $\mathbf{u}$ will coincide with the marginal MAP estimate. Owing to strong prior dependence between $\mathbf{x}$ and $\mathbf{u}$, a slightly different parameterisation is adopted. Note that the realisation of the random variable x_k together with the transition disturbance trajectory random variable $\mathbf{v} = v_{k+1:k+N}$ uniquely determines $\mathbf{x}$; see (1). Let

$$\mathbf{c} = (x_k, \mathbf{v}).$$

A similar re-parameterisation was also used in a related context in Hoffman et al. (2009). Similar to the approach of Müller (1999), consider an artificially augmented probability model

$$\tilde{p}(\mathbf{c}, \mathbf{u}|y_{0:k}, u_{0:k-1}) \propto r(\mathbf{c}, \mathbf{u}) p(y_k|x_k) p(x_k|y_{0:k-1}, u_{0:k-1}) p(\mathbf{v}), \tag{10}$$

where

$$p(\mathbf{v}) = \prod_{i=k+1}^{k+N} p(v_i) \tag{11}$$

$p(x_k|y_{0:k-1}, u_{0:k-1})$ is the predicted filter defined in (7) and $r(\mathbf{c}, \mathbf{u})$ is the reward function which is chosen to be both non-negative and bounded. Since the filter is not available in closed form, the particle approximation may be used for $p(x_k|y_{0:k-1}, u_{0:k-1})$:

$$I^{-1} \sum_{i=1}^{I} p(x_k|\widetilde{x}_{k-1}^{[i]}, u_{k-1}) w_{k-1}^{[i]} \tag{12}$$

The reward must also ensure (10) is integrable. In the predictive control setting $r(\mathbf{c}, \mathbf{u})$ is the horizon reward, i.e. the reward of the planning horizon which starts at time index k and continues onwards for N discrete time steps. The appropriate r is application specific and discussed in more detail in Section 5. The tilde in (10) indicates that this distribution is an artificial distribution introduced to compute the optimal control. In this and the next section the value of k in (10) is fixed and for the sake of brevity, $(y_{0:k}, u_{0:k-1})$ will from now on be denoted by $\mathbf{h}$.

The objective in nonlinear model predictive control is to optimize the future control

trajectory according to the criterion which is the expected value of the horizon reward function:

$$\mathbf{u}^* = \arg\max_{\mathbf{u}} = \int_{\mathbf{c}} r(\mathbf{c}, \mathbf{u}) p(x_k|\mathbf{h}) p(\mathbf{v}) d\mathbf{c}, \tag{13}$$

Note that from (10) we also have

$$\mathbf{u}^* = \arg\max_{\mathbf{u}} \tilde{p}(\mathbf{u}|\mathbf{h}). \tag{14}$$

Hence solving the control problem is equivalent to a marginal MAP estimation problem. This idea of casting an optimisation problem as an inference problem was first put forward by Müller (1999) in the context of experimental design and later extended by Doucet et al. (2002) to solve the marginal MAP estimation problem. Lecchini et al. (2006) also applied this idea to an air traffic separation problem (optimisation of waypoint positions for specified aircraft separation).

3.1. Monte Carlo computation

The maximization and marginalisation of (14) are achieved using a two-step Gibbs approach known as the state augmented marginal estimation (SAME) algorithm (Doucet et al. (2002).) Consider the *further* augmentation of the joint model $\tilde{p}(\mathbf{c}, \mathbf{u}|\mathbf{h})$ with ρ replications $\mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(\rho)$ of the random variable $\mathbf{c}$. Considering each of these replications as random variables in their own right, the following density can be defined

$$\tilde{p}(\mathbf{u}, \mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(\rho)|\mathbf{h}) \propto \prod_{j=1}^{\rho} \tilde{p}(\mathbf{c}(j), \mathbf{u}|\mathbf{h}). \tag{15}$$

The ρth power of the required marginal can be obtained by integrating over all the replications of $\mathbf{c}$:

$$\tilde{p}^{\rho}(\mathbf{u}|\mathbf{h}) = \int \cdots \int \prod_{j=1}^{\rho} \tilde{p}(\mathbf{c}(j), \mathbf{u}|\mathbf{h}) d\mathbf{c}(1) \ldots d\mathbf{c}(\rho). \tag{16}$$

For an MCMC algorithm with invariant density $\tilde{p}(\mathbf{u}, \mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(\rho)|h)$, the MCMC samples $\{\mathbf{u}^{[l]}\}_l$ taken on their own will be from the required marginal by construction. If an MCMC sampler can be constructed for the joint conditional density $\tilde{p}(\mathbf{c}, \mathbf{u}|\mathbf{h})$, then

it is usually straightforward to construct a sampler for the ρ replica counterpart density $\tilde{p}(\mathbf{u}, \mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(\rho)|\mathbf{h})$, since the replications of $\mathbf{c}$ are statistically independent conditional upon $\mathbf{u}$.

Samples can be generated from the density (15) with a two-step Gibbs sampler. At sampling step l, a sample $\mathbf{c}^{[l]}(j)$ can be generated for each of the replications $j = 1, \ldots, \rho$ using a chain with a target density $\tilde{p}(\mathbf{c}|\mathbf{u}^{[l-1]}, \mathbf{h})$, whereas $\mathbf{u}^{[l]}$ can then be sampled from $\tilde{p}(\mathbf{u}|\mathbf{c}^{[l]}(1), \mathbf{c}^{[l]}(2), \ldots, \mathbf{c}^{[l]}(\rho), \mathbf{h})$; this is the SAME method of Doucet et al. (2002). If $\tilde{p}(\mathbf{u}|\mathbf{c}(j), \mathbf{h})$ is a member of the exponential family, then sampling from $\tilde{p}(\mathbf{u}|\mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(\rho), \mathbf{h})$ is straightforward since the product of conditionals $\prod_{j=1}^{\rho} \tilde{p}(\mathbf{u}|\mathbf{c}(j), \mathbf{h})$ will also be a member of the exponential family. In more general settings, samples can be generated using MCMC methods.

In order to converge to the global maximum in (14), a simulated annealing (Metropolis et al. (1953); Kirkpatrick et al. (1983)) based approach may be adopted. Starting with an initial value for ρ, say $\rho(1) = 1$, ρ is incrementally increased such that $\lim_{l \to \infty} \rho(l) = +\infty$. Specific schedules are presented in Section 5. This idea leads to the non-homogeneous SAME method Doucet et al. (2002) which is summarised in Algorithm 1. In practice, Algorithm 1 should be run until the sequence of $\mathbf{u}^{[l]}$ "stabilises." If this happens after L iterations, the approximation of the optimal control trajectory is given by $\mathbf{u}^* = \mathbf{u}^{[L]}$.

Algorithm 1 SAME for NMPC

1: Initialise $\mathbf{u}^{[0]}$, $l = 1$
2: **for** $l = 1, \ldots, L$ **do**
3: **for** $j = 1, \ldots, \rho(l)$ **do**
4: Sample $\mathbf{c}^{[l]}(j) \sim \tilde{p}(\mathbf{c}|\mathbf{u}^{[l-1]}, h)$
5: **end for**
6: Sample $\mathbf{u}^{[l]} \sim \tilde{p}(\mathbf{u}|\mathbf{c}^{[l]}(1), \ldots, \mathbf{c}^{[l]}(\rho(l)), h)$
7: **end for**
8: Let $\mathbf{u}^* = \mathbf{u}^{[L]}$

Algorithm 1 provides a way of approximating the MAP estimate (14). For the sake of completeness, details of the MCMC algorithms for sampling from the two conditional densities $\tilde{p}(\mathbf{c}|\mathbf{u}^{[l-1]}, \mathbf{h})$ and $\tilde{p}(\mathbf{u}|\mathbf{c}^{[l]}(1), \ldots, \mathbf{c}^{[l]}(\rho(l)), \mathbf{h})$ are now presented.

Each SAME iteration requires the generation of $\rho(l)$ samples from the density $\tilde{p}(\mathbf{c}|\mathbf{u}^{[l-1]}, \mathbf{h})$ and the pseudo-code of the proposed sampler is presented in Algorithm 2. Every time ρ

is incremented, i.e. if at step l $\rho(l) > \rho(l-1)$, $\mathbf{c}(j)$ needs to be initialised for $j > \rho(l-1)$. This is done by sampling from the prior. The prior in this case (see (10)) is $p(\mathbf{v})$ in (11) and the particle approximation to $p(x_k|y_{0:k}, u_{0:k-1})$ available after step 2 in the particle filter algorithm of Section 2. The x_k component of the new jth MCMC chain can be initialised by resampling the current (time index k) output $\{\widetilde{x}_k^{[i]}, w_k^{[i]}\}_{i=1}^I$ of the state tracking particle filter of Section 2. The $\mathbf{v}$ component of the jth MCMC chain can be initialised by sampling from the prior (11). Of course, any arbitrary initialisation of x_k and $\mathbf{v}$ could be adopted as well, but this method was found to be more effective. For existing chains, sampling $\mathbf{c}^{[l]}(j) = (x_k^{[l]}(j), \mathbf{v}^{[l]}(j))$ from the jth chain is achieved using an independent proposal Metropolis-Hastings sampler detailed in Algorithm 2. The proposal is comprised of the priors (11) and (12). The proposed sample $\mathbf{c}(j) = (x_k(j), \mathbf{v}(j))$ is either accepted or rejected according to the acceptance ratio in (17). The sample required in line 4 of Algorithm 1 is $\mathbf{c}^{[l]}(j) = (x_k(j), \mathbf{v}(j))$ if the move is accepted, otherwise $\mathbf{c}^{[l]}(j) = \mathbf{c}^{[l-1]}(j)$. If a proposal other than the prior (12) is used, then this mixture density will enter into the evaluation of the acceptance probability in (17), which will be expensive. This problem can be avoided by using an auxiliary particle (Pitt and Shephard (1999)) type implementation.

Algorithm 2 Independent Metropolis-Hastings State Sampler

1: If first iteration of the jth chain, initialise $\mathbf{c}^{[l-1]}(j) = (x_k(j), \mathbf{v}(j))$ by sampling x_k from $\{\tilde{x}_k^{[i]}, w_k^{[i]}\}_{i=1}^I$ (see Step 2 of particle filter algorithm) and $\mathbf{v}(j)$ from (11). Otherwise $\mathbf{c}^{[l-1]}(j) = (x_k^{[l-1]}(j), \mathbf{v}^{[l-1]}(j))$ is the previous output of the algorithm.

2: Sample x_k from (12), $\mathbf{v}(j)$ from (11) and accept the proposal with probability

$$\min\left\{1, \frac{r(x_k(j), \mathbf{v}(j), \mathbf{u})p(y_k|x_k(j))}{r(x_k^{[l-1]}(j), \mathbf{v}^{[l-1]}(j), \mathbf{u})p(y_k|x_k^{[l-1]}(j))}\right\} \tag{17}$$

3: If accepted, $\mathbf{c}^{[l]}(j) = \{x_k(j), \mathbf{v}(j)\}$, otherwise $\mathbf{c}^{[l]}(j) = \mathbf{c}^{[l-1]}(j)$.

Sampling the control trajectory from the conditional density $\tilde{p}(\mathbf{u}|\mathbf{c}^{[l]}(1), \ldots, \mathbf{c}^{[l]}(\rho(l)), \mathbf{h})$ as required in line 6 of Algorithm 1 can be accomplished using a random walk Metropolis-Hastings sampler as detailed in Algorithm 3. A candidate sample $\mathbf{u}$ is produced from the conditional density $q(\mathbf{u}|\mathbf{u}^{[l-1]})$, which is then accepted with the probability given in (18).

Algorithm 3 Metropolis-Hastings Control Sampler

1: Let $\mathbf{u}^{[l-1]}$ be the previous output of the algorithm. Sample $\mathbf{u} \sim q(\mathbf{u}|\mathbf{u}^{[l-1]})$ and accept the proposal with probability

$$\min \left\{ 1, \frac{q(\mathbf{u}^{[l-1]}|\mathbf{u}) \prod_{j=1}^{\rho(l)} r(\mathbf{c}^{[l]}(j), \mathbf{u})}{q(\mathbf{u}|\mathbf{u}^{[l-1]}) \prod_{j=1}^{\rho(l)} r(\mathbf{c}^{[l]}(j), \mathbf{u}^{[l-1]})} \right\} \tag{18}$$

2: If accepted, $\mathbf{u}^{[l]} = \mathbf{u}$, otherwise $\mathbf{u}^{[l]} = \mathbf{u}^{[l-1]}$.

4. Extensions

Continuous time systems can also be controlled using the methods detailed in the previous section. In fact, the examples in Section 5 are discretised continuous time problems where the numerical integration time step is much smaller than the control time step. The control was parameterised by a piecewise constant (sample and hold) function.

Constraints can be quite straightforwardly incorporated within the particle predictive control framework. Control trajectory constraints may be easily enforced if the control trajectory parameterisation is piecewise constant (i.e. sample and hold), since in some cases it will merely involve checking for constraint violations at a small number of control points. Constraints can be enforced by simply rejecting trajectory proposals which violate the control constraints. This requires that the continuous control trajectory should be sampled sufficiently fine and should include a sampling error margin when checking for constraint violation. The drawback of this simple approach is that it may greatly reduce the acceptance rates in the MCMC algorithm, thereby reducing its efficiency in exploring the control space. Similarly, state constraints can be handled by rejection of state trajectories which violate constraints. Constraints may result in a target distribution with "disconnected support" and the resulting MCMC kernel may become reducible. A possible solution to this is to allow ventures into regions where constraints are violated but discourage it using a penalty function which sharply reduces the probability density when constraint boundaries are crossed. This is the approach used in the F-16 example of section 5.2.

It would be beneficial to be able to sample the marginal in (14) directly instead of

having to revert to a Gibbs approach as in Algorithm 1 because of the potential strong correlation between the variables which are sampled alternately. If the N horizon reward can be expressed in a product form as follows

$$r(\mathbf{x}, \mathbf{u}) = \prod_{i=k+1}^{k+N} r_i(x_i, u_{i-1})$$

where $\mathbf{x} = \{x_{k+1}, \ldots, x_{k+N}\}$, and r_i are positive, e.g. see (21), then it is possible to use a particle filter to form an unbiased estimate of the acceptance probability of an MCMC algorithm with proposal function $q(\mathbf{u}|\mathbf{u}^{[l-1]})$ and target density $\tilde{p}(\mathbf{u}|\mathbf{h})$ (Andrieu et al. (2010)), i.e.

$$\min\left\{1, \frac{\tilde{p}(\mathbf{u}|\mathbf{h})}{\tilde{p}(\mathbf{u}^{[l-1]}|\mathbf{h})} \times \frac{q(\mathbf{u}^{[l-1]}|\mathbf{u})}{q(\mathbf{u}|\mathbf{u}^{[l-1]})}\right\}.$$

The computational cost per iteration l is $I \times N$ and, in the context of Bayesian inference, the algorithm has been demonstrated to quite effective in a number of scenarios; see Andrieu et al. (2010) for precise details on the unbiasedness property and numerical performance.

A further extension which may be beneficial is to use parallel methods, along the lines of Higdon et al. (2002), in order to generate samples from multi-resolution control trajectories where fast mixing coarse scale control trajectory chains may aid explorations in fine scale control trajectory chains. See also Higdon (1999) for a different approach using auxiliary variable methods to improve the computational efficiency of MCMC.

In NMPC there has been much research effort focused on proving stability of automatic controller algorithms. Conditions for stability have been derived for deterministic non-linear NMPC, e.g. see Mayne et al. (2000), Findeisen and Allgöwer (2002) and Scokaert et al. (1999) for details. Similar results for the stochastic NMPC problem state that given a long enough finite horizon length, it is possible to guarantee stability (Jadbabaie and Hauser (2005)). However, this is largely an area of ongoing research.

5. Applications

5.1. Car up the Hill

The particle predictive controller was tested on the well known car-up-the-hill example problem of Moore and Atkeson (1995) (see also Sutton (1996).) The objective is to drive an underpowered car starting at rest in a valley up a hill towards a goal position (Figure 1). The original car-hill problem is adapted to the NMPC framework, in that once the goal state is reached the car is to be kept there indefinitely without rolling down the hill. A further modification is to introduce stochasticity in the dynamics by introducing a Gaussian distributed surface dynamic friction coefficient. The schematic representation of the problem with all the relevant forces acting on the car is depicted in Figure 1. The car-up-the-hill problem is an interesting example because the car cannot generate enough force to drive straight up the hill towards the goal position. The path to the goal state requires an initial push in the opposite direction in order to gain sufficient momentum and only then proceed straight towards the goal state. However, optimal control algorithms which act in a greedy manner (for example gradient based algorithms) will head towards a local maximum of performance criterion. In this example, such an approach may not be successful since a local maximum control could attempt to drive the car directly right towards the goal state even though it will not have enough momentum to proceed all the way up the hill. Whether such maxima exist depends on how the reward function is defined.

The state of the system is represented by a two dimensional column vector of position and velocity in the horizontal direction, i.e. $x = [z \quad \dot{z}]^\top$. Assuming the car has a mass of $m = 1$kg, the problem is constrained by the car being able to produce a maximum thrust u of 5N in either horizontal direction, i.e. $-5 \leq u \leq 5$. (This amounts to 1N more thrust being available than in the original problem of Moore and Atkeson (1995); Sutton (1996) in order to compensate for the introduction of friction.) Further state constraints on the horizontal position and velocity of the car are defined by $-1 \leq z \leq 1$ and $-2 \leq \dot{z} \leq 2$ respectively. The height is a function of the horizontal position z, and is given by

$$h_c(z) = \begin{cases} z^2 + z & \text{if } z < 0, \\ z/\sqrt{1+5z^2} & \text{if } z \geq 0. \end{cases} \tag{19}$$

The continuous time dynamics of the car, by Newtonian mechanics, is given by

$$\vec{a} = \ddot{z} = \frac{u}{m_c\sqrt{1+(h_c')^2}} - \frac{g(h_c' + v\text{sign}(z))}{1+(h_c')^2}, \tag{20}$$

where v denotes the dynamic friction coefficient, sign$(\cdot)$ denotes the signum function and h_c' denotes the derivative of the height function with respect to the horizontal position z. Stochastic dynamics are introduced by assuming a slightly uneven rough surface. The dynamic friction coefficient v_k at discrete-time k is assumed to be an independent Gaussian distributed random variable with $v_k \sim \mathcal{N}(0.05, (0.02)^2)$. Here dynamic friction is characterised as a function of time and not position. This approximation is made to simplify the example and is reasonable as long as the car is in motion.

If the continuous-time model is integrated using Euler's method the difference equation $x_k = \bar{F}x_{k-1} + \bar{G}\vec{a}_k$ is produced, where x_k denotes the state vector x at discrete-time instance k,

$$\bar{F} = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix}, \; \bar{G} = \begin{bmatrix} \frac{\Delta^2}{2} \\ \Delta \end{bmatrix},$$

where Δ represents the numerical integration interval and

$$\vec{a}_k = \frac{u_k}{m\sqrt{1+(h_c')^2}} - \frac{g(h_c' + v_k\text{sign}(z_k))}{1+(h_c')^2}.$$

Finally, the original problem is extended to include noisy observations, where the observation function is given by $y_k = x_k + w_k$, where in this case $w_k \sim \mathcal{N}(\mathbf{m}, \Sigma)$, $\mathbf{m} = [0\,0]^\top$ and

$$\Sigma = \begin{bmatrix} (0.01)^2 & 0 \\ 0 & (0.001)^2 \end{bmatrix}.$$

The exponential of the negative of the distance to the goal state for each of the

discrete-time state trajectory points was found to be an effective performance criterion. The reward over the whole control horizon is

$$r(\mathbf{c}, \mathbf{u}) = \prod_{j=k+1}^{k+N} \exp(-\mathbf{e}_j^\top \mathbf{Q} \mathbf{e}_j), \tag{21}$$

where $\mathbf{e}_j$ is the stage error between the car state and the goal state, N is the number of stages of the planning horizon and $\mathbf{Q}$ is an appropriate 2×2 weighting matrix, e.g. the identity matrix.

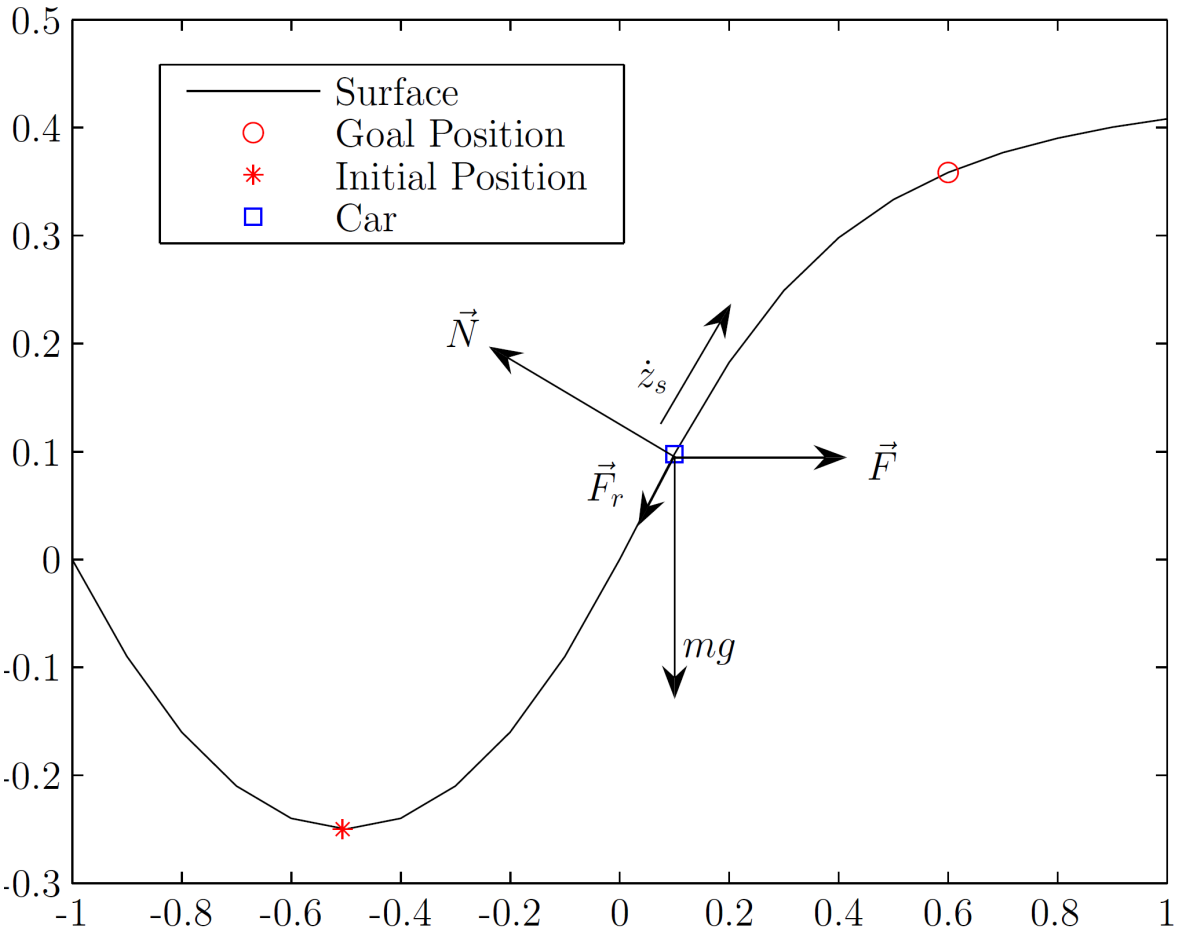

Figure 1: Schematic description of the car-hill problem. The car is traveling at a surface velocity $\dot{z}_s$ with its centre of gravity acted upon by gravity, a horizontal propulsion force $\vec{F}$, the normal force $\vec{N}$ as a result of Newton's law and friction $\vec{F}_r$

.

5.1.1. Simulation Setup

Simulations were run with a dynamic model time discretisation interval of 0.02s and a control interval of 0.1s. The future control function (trajectory) was parameterised with piecewise constant pulses between control instances. State observations were timed to coincide with control change point time instances and a particle filter with 20 particles was used to track the state. Most of the computation effort was spent on the first control horizon (50000 MCMC iterations), after which subsequent horizons were run for 4000 iterations each[1]. In these subsequent horizons, the control function solution of the previous optimisation run was shifted according to the control time step and used as initialisation of the planning stage of the next horizon – the end of the horizon's control function was simply appended with zeros. The annealing scheme was set up as follows. The first 10000 iterations of the first planning horizon were run with $\rho = 1$, after which ρ was incremented every 1000 iterations. For the subsequent planning horizons, the first 2000 iterations were run with $\rho = 1$, after which ρ was incremented every 500 iterations. The time horizon length over which dynamic optimisation was effected at each decision instance was $N = 20$ which corresponds to 2 seconds.

5.1.2. Results

Since NMPC involves repeatedly solving the optimal control problem over a shifting horizon, it is of interest to see how the particle predictive controller performs during the *first* horizon (see Figure 2). The control for the first horizon will set the car in motion and will have a considerable influence on the rest of the car's trajectory. It is therefore quite important that the first horizon's control can generate a path for the car that could eventually end up in the goal state. It is for this reason that much more computational effort was put into the first dynamic optimisation run than into subsequent runs.

To gauge the performance of Algorithm 1, the natural logarithm of the product of rewards, $\hat{r}(l) = \sum_{j=1}^{\rho(l)} \ln(r(\tilde{\mathbf{c}}^{[l]}(j), \mathbf{u}^{[l]}))$, after dividing by $\rho(l)$, where $\tilde{\mathbf{c}}^{[l]}(j)$ are the proposed samples in Algorithm 2, is depicted in Figure 2. Ideally, one would have liked to plot

[1]Once the initial planning stage was complete, the controller simply had to follow the plan while correcting for disturbances.

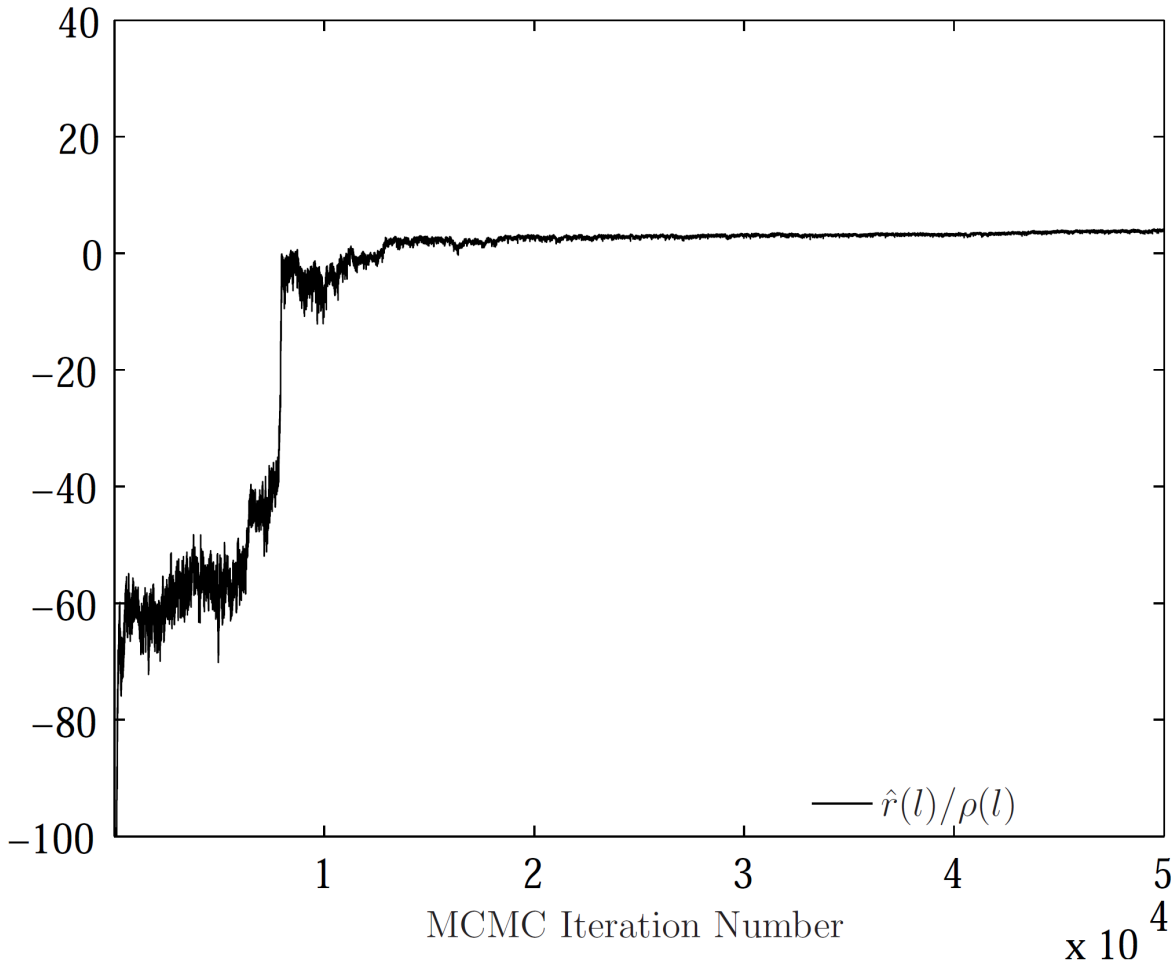

Figure 2: Plot of $\rho(l)^{-1}\sum_{j=1}^{\rho(l)}\ln(r(\tilde{\mathbf{c}}^{[l]}(j),\mathbf{u}^{[l]}))$ versus MCMC iteration number l for the first planning horizon of the car-up-the-hill problem.

$\tilde{p}(\mathbf{u}^{[l[}|\mathbf{h})$, but this quantity cannot be evaluated directly, so any statistic to help diagnose convergence will do. Note that $\rho(l) = 1$ for the first 10000 iterations and then $\rho(l)$ was increased by one every 1000 iterations after that. The final ρ value was 40. Figure 2 indicates that the algorithm quickly converged to a local maximum within the first few hundred iterations. Acceptance rates decreased from 60% to 15% as ρ was increased from 1 to 40. The acceptance rate for Algorithm 2 was approximately 20% for all values of ρ. The sudden log-reward increase at about 8000 indicates escape from the local maximum. As with most global optimisation methods, convergence to the global maximum cannot be guaranteed, although with this setup the success rate was about 88%; the global optimum trajectory of the non-stochastic version of this problem can be found by

discretised dynamic programming and serves as a reference. This rate could perhaps be further improved by the use of population MCMC methods (see Jasra et al. (2007)) to sample control trajectories.

Figure 3 depicts how the car is guided through the state-space towards the goal area (the box on the right hand side) using the particle predictive controller. The crosses indicate the instances when control decisions were made and the planning horizon was shifted. This also corresponds to points where observations were taken and incorporated into the particle filter.

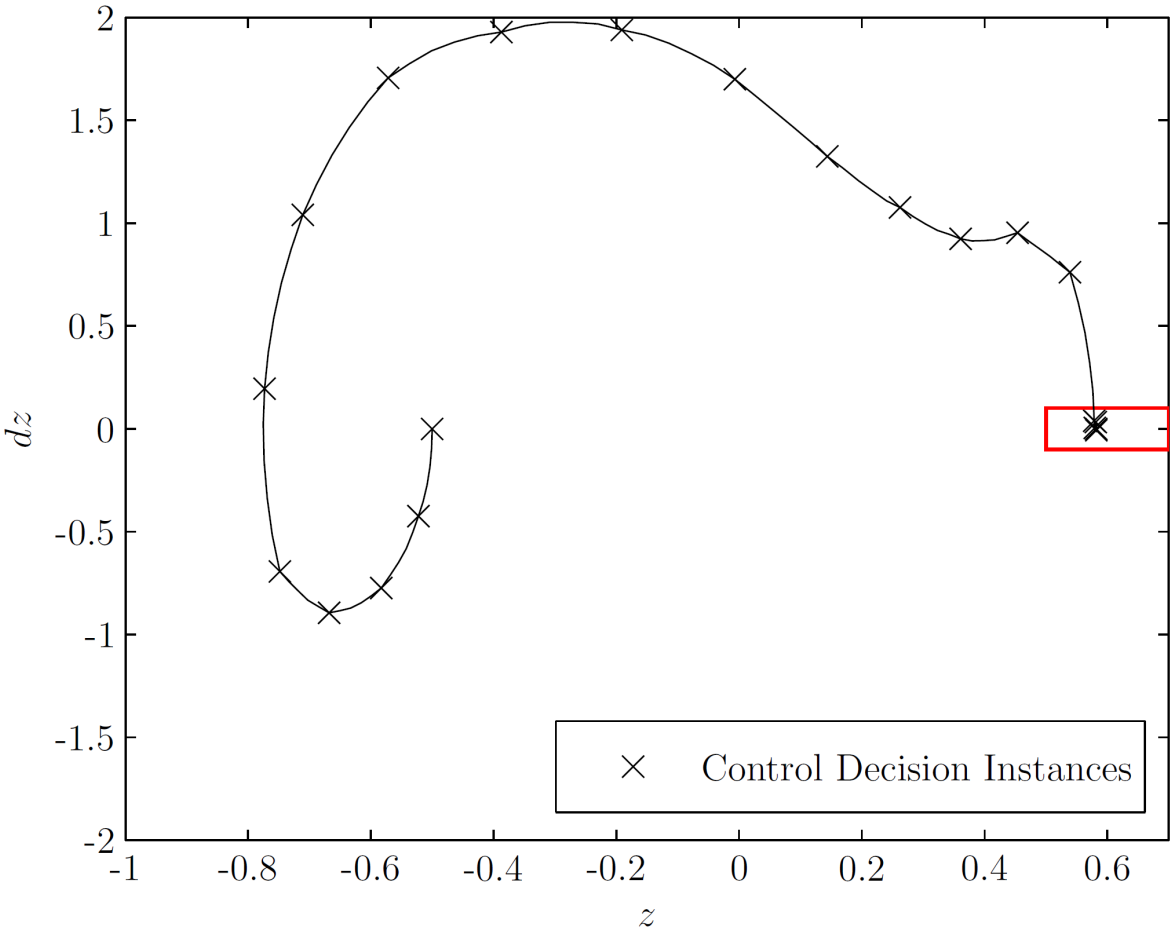

Figure 3: The actual trajectory traveled by the car for 21 control steps computed with particle predictive control.

The controller executed 20 controls in total, spaced at 0.1s intervals, resulting in a total

elapsed control time of 2 seconds. This controller simulation was completed in about four and a half minutes of simulation time on a 2.0GHz dual core Intel processor. This is more than two orders of magnitude slower than real time control. Graphical Processing Units (GPUs) and dedicated digital hardware (FPGA and VLSI), combined with parallel implementation of parallel running Markov Chains for the different sample replicates, may greatly accelerate the operation of the relevant algorithms to be acceptable for real time implementation. Emphasis must be placed on the fact that the above methods allow a generic unified approach to solving NMPC problems.

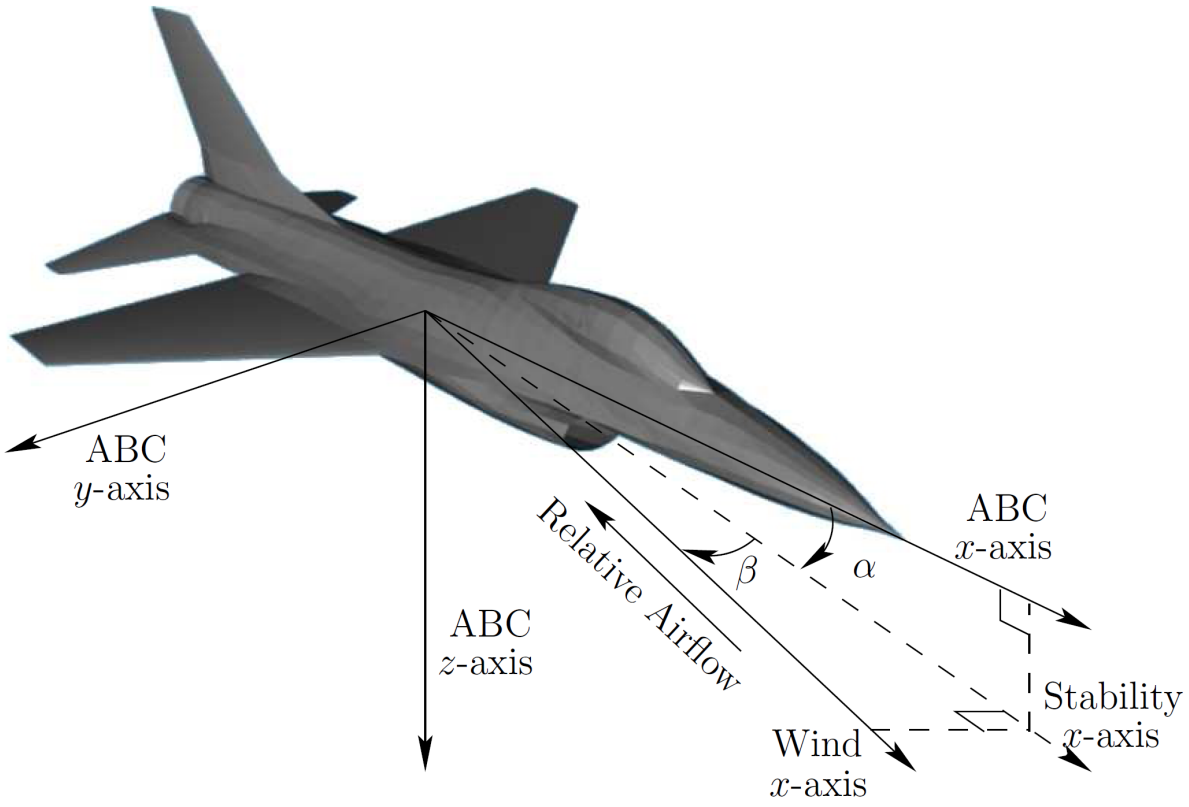

Figure 4: Axes systems for the 6-DOF aircraft model

5.2. F-16 Trajectory Following

5.2.1. Simulation Setup

A more challenging problem for testing the particle predictive controller is terrain following by an aircraft. A six-degree-of-freedom nonlinear aircraft model of Stevens and

Lewis (2003), with the flat earth assumption was used to model the F-16 aircraft (Figure 4). The nonlinear aircraft model is too extensive to present here, but consists of 13 state variables expressed in terms of three force equations, three kinematic equations, three moment equations, three navigation equations and a propulsion equation. Several moment and force coefficients were provided by the interpolation of experimentally determined lookup tables, which further contribute to the nonlinearity of the model. In order to simplify the particle filter which tracks the state, the following simplifying assumptions were made. Apart from the positional (navigational) state variables, all other state variables (body frame velocity, Euler angles and body frame angular rates) were assumed to be perfectly observed and to have deterministic dynamics. Where the navigational equations were concerned, the dynamic disturbance variable modeled atmospheric turbulence and was assumed to be zero-mean Gaussian with a variance of 1 foot in each spacial dimension. The observations took the form of noisy GPS measurements with a variance of 6 feet in each dimension and the corrupting noise was assumed to be zero-mean Gaussian as well. For this case, which corresponds to a state-space model with nonlinear dynamics and a linear observation equation with Gaussian noise, the optimal importance density (see Doucet et al. (2000)) was used for the particle filter, i.e. f_k in the particle filter algorithm of Section 2, which tracks the state.

A reference trajectory 50m above ground level was generated such that several criteria were adhered to. The reference trajectory was constrained to lie within maximum deviation from a straight line between two locations (the dashed lines in Figure 6) while minimising the altitude of the trajectory. In this way, the aircraft trajectory favours valleys which in turn facilitates terrain masking. Also, adherence to the aircraft's maximum turning rate was specified as a constraint to the reference trajectory generation.

Simulations were run with a dynamic model time integration interval of 0.02s and a control interval of 0.32s, and the model was integrated between control instances using a Runga-Kutta-4 numerical integration scheme. The state transition disturbance v_k and observation disturbance w_k (see (1) and (2)) were inserted in the relevant navigational

state variables at control change point instances. As with the previous example, the future control trajectory was parameterised using sample-and-hold between control instances, and state observations were timed to coincide with control change point time instances. The particle filter with 50 particles was used to track the state. In Algorithm 1, $L = 8000$ MCMC iterations for the first control horizon, and 4000 iterations for the subsequent shifted horizons. In the optimisation of these subsequent horizons, $\mathbf{u}^{[0]}$ is initialised in step 1 of Algorithm 1 by using the following procedure. The control trajectory solution $\mathbf{u}^*$ of the previous horizon is taken, the first (already executed) control vector entry is discarded and the last entry is appended with another entry of the same value. This primes the algorithm with the solution of the previous optimisation run where the two horizons overlap in time. The annealing scheme for the F-16 trajectory following example was set up as follows. In the first planning horizon ρ was incremented every 1000 iterations and for the subsequent planning horizons ρ was incremented every 500 iterations. $N = 16$ which corresponds to 5.12 seconds. In this example the reward function took a complicated form, which included a quadratic reference trajectory deviation penalty term, as well as several constraint penalty functions. Constraints included a limitation on the height allowed below the reference trajectory, a G-force constraint, aircraft attitude constraints as well as a ground clearance constraint.

Figures 5 and 6 depict the trajectory following results of the particle predictive controller. In both cases the F-16's actual trajectory deviated from the reference by not more than 40m horizontally and 20m vertically over the whole path traveling at 600km/h.

6. Conclusions

Particle predictive control is a simple and unified approach to deal with NMPC problems with stochastic dynamics. The reward function need not be smooth while constraints can in principle be handled with the same framework. This work shows that, in principle, these methods can be applied to the NMPC problem, but the relevant methods could be further improved through more efficient sampling procedures as discussed in Section 4. The applications considered in this paper operate a fair way short of real time implementation, and questions arise as to the practicality of the application of particle predictive control to high sampling rate contexts such as trajectory following. Several considerations

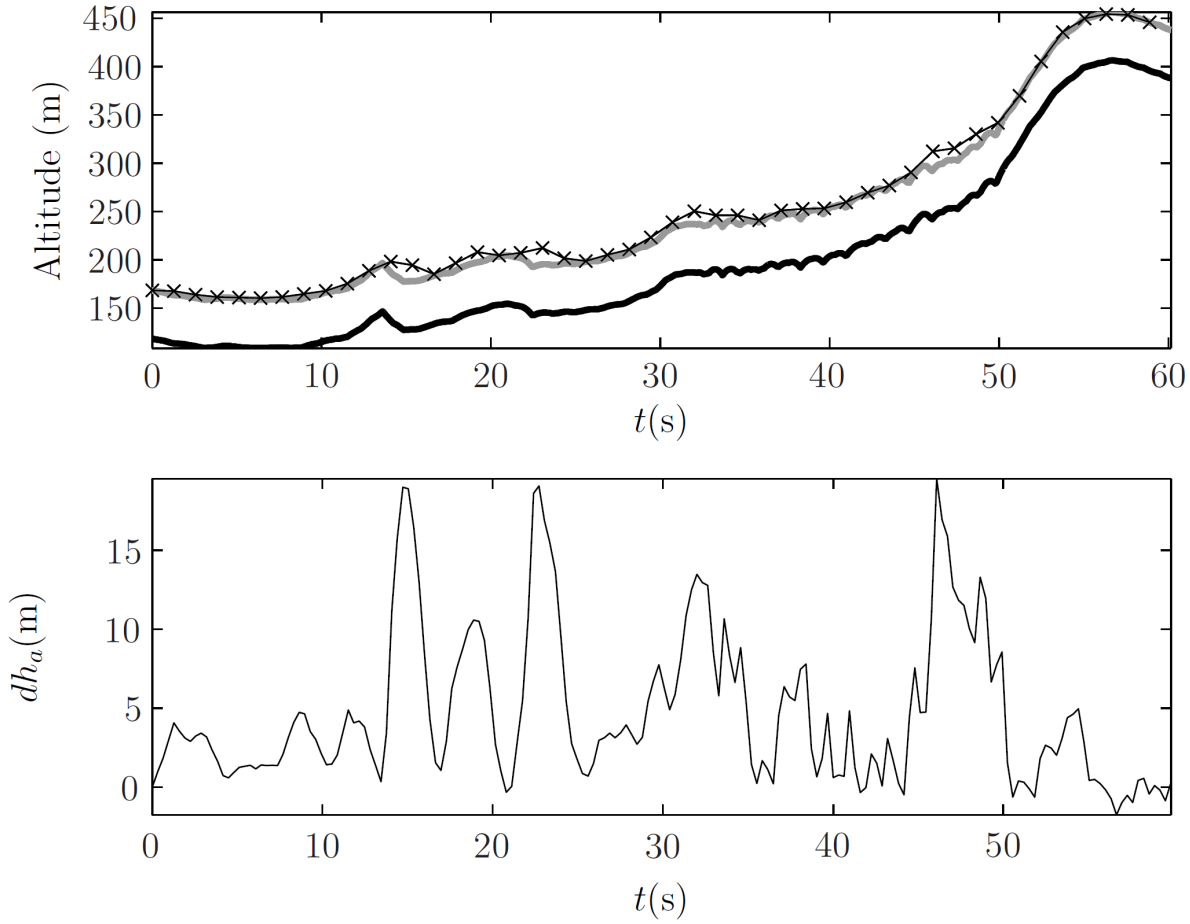

Figure 5: Top: Elevation profile of terrain under the reference trajectory (gray) and actual aircraft path (black-x) as produced by the baseline particle predictive controller. Bottom: Difference between the reference trajectory height and the actual height of the aircraft.

need mentioning in this regard. Firstly, owing to the replicated sample based nature of the SAME algorithm, it was envisaged that the SAME algorithm could be implemented using parallel processing units. Secondly, the DSP based FPGAs are available which can churn out hardware multiplications at MHz rates. State-of-the-art Graphical Processing Units (GPUs) on the other hand can handle a large number of parallel threads. This allows for very quick calculation of the nonlinear system dynamics used for the predictive part of the optimisation as well parallelising the MCMC algorithms (Sohn (1995).) Considering these points, using a combination of parallel processing and high speed reconfigurable logic of GPU technology, practical implementation may be possible currently or in the near future. Another consideration is the elapsed time from the moment the new observa-

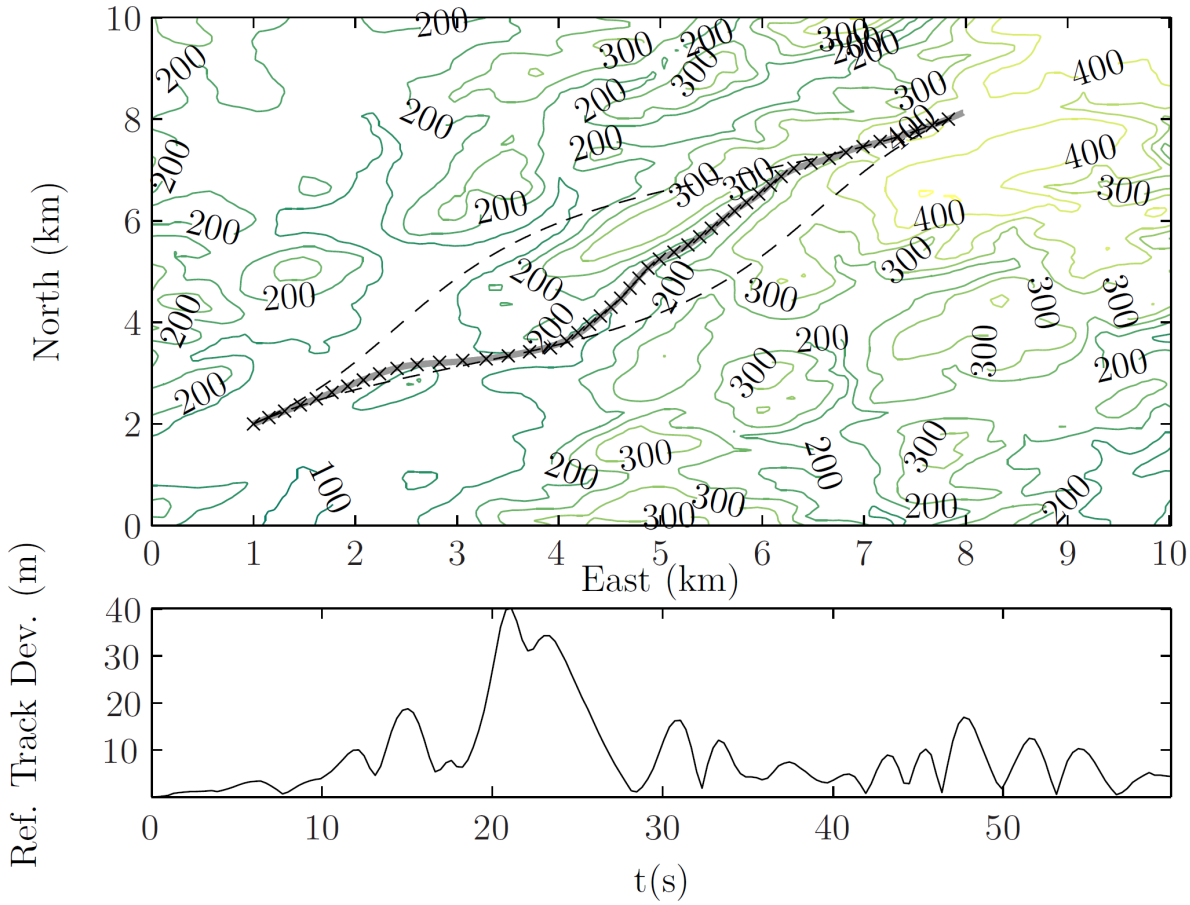

Figure 6: Top: Terrain contour plot with waypoint reference trajectory (gray) and actual aircraft path (black-x) produced by the particle predictive controller. Bottom: NE-plane Euclidean distance between aircraft position and reference trajectory point for the same simulation run. The dashed lines indicate the maximum allowed deviation from a straight line during trajectory generation.

tion is made to the moment the optimal control for the current horizon is computed. The effect of this delay for real applications should be accounted for since the system state would have evolved during this period. Finally, the trajectory following application was used to show the utility of particle predictive control for a nonlinear system. Emphasis was placed on the demonstration of the relevant methods and concepts. Nonlinear applications which operate at much lower sampling rates, such as industrial and chemical process control (Qin and Badgwell (2000)), green energy control (Silva et al. (2003)), as well as the control of financial and econometric systems may be more viable applications for the particle predictive controller and its variants.

C. Andrieu, A. Doucet, and R. Holstein. Particle Markov Chain Monte Carlo methods

(with discussion). *Journal Royal Statistical Society B*, 72:269–342, 2010. 11

P. Del Moral. *Feynman-Kac formulae: Genealogical and Interacting Particle Systems with Applications*. Springer, New York, 2004. 5

A. Doucet. On sequential simulation-based methods for Bayesian filtering. Technical Report CUED-F-ENG-TR310, University of Cambridge, Department of Engineering, 1998. URL `www.stats.bris.uk/MCMC/`. 4

A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10:197–208, 2000. 19

A. Doucet, S.J. Godsill, and C.P. Robert. Marginal maximum a posteriori estimation using MCMC. *Statistics and Computing*, 12:77–84, 2002. 2, 7, 8

R. Findeisen and F. Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, pages 1–23, Veldhoven, 2002. 1, 11

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to non-linear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140:107–113, 1993. 4

D. Higdon. Auxiliary variable methods for Markov Chain Monte Carlo with applications. *Journal of the American Statistical Association*, 93(442):585–595, June 1999. 11

D. Higdon, H. Lee, and Z. Bi. A Bayesian approach to characterizing uncertainty in inverse problems using coarse and fine scale information. *IEEE Transactions on Signal Processing*, 50(2):389–399, February 2002. 11

Matthias Hoffman, Hendrik Kueck, Nando de Freitas, and Arnaud Doucet. New inference strategies for solving markov decision processes using reversible jump mcmc. In *Proceedings of the Proceedings of the Twenty-Fifth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-09)*, pages 223–231, Corvallis, Oregon, 2009. AUAI Press.

A. Jadbabaie and J. Hauser. On the stability of receding horizon control with a general terminal cost. *IEEE Transactions on Automatic Control*, 50(5):674–678, 2005. 11

A. Jasra, D.A. Stephens, and C. Holmes. On population based simulation for static inference. *Statistics and Computing*, 17(3):263–279, 2007. 17

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983. 8

A. Lecchini, W. Glover, J. Lygeros, and J. Maciejowski. Predictive control of complex stochastic systems using Markov chain Monte Carlo with application to air traffic control. In *Nonlinear statistical signal processing workshop*, September 2006. 7

A. Lee, C. Yau, C. Holmes, M. Giles, and A. Doucet. On the utility of Graphics cards to perform massively parallel implementation of advanced Monte Carlo methods. *J. Comp. Graph. Statist. (in press)*, 2010. URL `http://arxiv.org/abs/0905.2441`. 2

D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 26(6):789–814, 2000. 11

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21: 1087–1092, 1953. 8

A. W. Moore and C. G. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state spaces. *Machine Learning*, 21(3):199–233, 1995. 12

P. Müller. Simulation based optimal design. In J.M. Bernado, J.M. Berger, A.P. David, and A.F.M. Smith, editors, *Bayesian Statistics 6*, pages 459–474. Oxford University Press, 1999. 2, 6, 7

M.K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):pp. 590–599, 1999. 9

S. Joe Qin and Thomas A. Badgwell. An overview of nonlinear model predictive control applications. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control*, pages 369–392. Birkhäuser, 2000. 22

P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999. 11

R. N. Silva, J. M. Lemos, and L. M. Rato. Variable sampling adaptive control of a distributed collector solar field. *IEEE Trans. Control Systems Technology*, 11(5):765–772, 2003. 22

A. Sohn. Parallel N-ary speculative computation of simulated annealing. *IEEE Trans. Parallel Distrib. Syst.*, 6(10):pp. 997–1005, 1995. 21

B. L. Stevens and F. L. Lewis. *Aircraft Control and Simulation*. Wiley, N.J., 3rd edition, 2003. 18

R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems*, 8:1038–1044, 1996. 12