Modeling, Identification and Control, Vol. 44, No. 1, 2023, pp. 17–29, ISSN 1890–1328

HMC Techniques for Reducing the Uncertainty of Gas-Lifted Oil Field Model

Kushila Jayamanne, [1] Bernt Lie [1]

[1] *Department of Electrical Engineering, IT and Cybernetics, University of South-Eastern Norway, 3918 Porsgrunn, Norway. E-mail:* {Kushila.R.Jayamanne, Bernt.Lie}@usn.no

Abstract

Parametric model uncertainties could have a high impact on the predictive capabilities of a model. When process measurements become available, these uncertainties may be reduced using parameter estimation techniques. Estimation techniques founded on the Bayesian framework in particular are powerful: they produce a probability density function (PDF) of the estimated parameter rather than a single point estimate.

In this paper, we consider a gas lifted oil field model whose predictions are highly sensitive to uncertainty in its parameters. We apply Markov Chain Monte Carlo (MCMC) methods, which follow the Bayesian paradigm, to estimate these parameters, and thereby reduce the uncertainty in the model predictions; two different algorithms, Hamiltonian Monte Carlo (HMC) and No-U-Turn Sampler (NUTS), are used. The probabilistic programming language (PPL), Turing in Julia is used for implementation. Monte Carlo simulations and/or data retrodiction is performed prior to and post parameter estimation, to evaluate the uncertainty in model predictions; the outcomes are compared to determine the efficacy of parameter estimation. Results show that the computed posterior distributions yield model predictions that are in close agreement with the observations, and that model uncertainty is effectively reduced.

Keywords: Parameter estimation, Markov Chain Monte Carlo, Model uncertainty, Hamiltonian Monte Carlo, No-U-Turn Sampler

1 Introduction

Affordable and clean energy for all is one of United Nations' 17 goals for sustainable development[1]. This goal implies phasing out fossil fuel sources, and instead use solar and wind sources, etc. A growing global population with expectations of increased standard of living dictates that a diversity of energy sources must be used, and that fossil fuels will be phased out gradually. Ongoing wars and constraints on available energy have reinforced the need for use of an energy mixture, and may also lead to a speed-up in the transition towards distributed and sustainable energy sources.

Meanwhile, it is important to produce fossil fuel as cleanly as possible, while developing new ideas and technology that is vital for both current and future energy sources. Petroleum production is important for Norways national economy, with petroleum production income of approximately one third of her gross domestic product in 2021. This gives Norway a special incitement for developing technology to meet future restrictions on energy production.

In an on-going project DigiWell[2], the focus is on model uncertainty of oil production from reservoir to separator, and how improved sensor technology, optimization, and control can reduce the climate footprint as well as uncertainty in energy consumption and profit.

[1] https://sdgs.un.org/goals

[2] See Acknowledgements

doi:10.4173/mic.2023.1.2

A fundamental part of control system design is determining the sensor-actuator configuration, i.e., the control architecture (Goodwin et al., 2001). To have complete control over a system, we need sensors to monitor every state, and actuators to directly alter every state; physical and economic restrictions make this infeasible. Thus, successful control requires strategic placement of only a few sensors and actuators. Several studies have attempted to solve the optimal control architecture problem (Dhingra et al., 2014; Manohar et al., 2021; Muske and Georgakis, 2003; Sakha and Shaker, 2017; Zare et al., 2020). To our knowledge, none of these addresses how to handle uncertain systems whose dynamics vary over time, such as an oil field. Ignoring system uncertainties would affect the reliability of the found optimal design; it might fall short of the performance requirements. Once in place, if this indeed turns out to be the case, the architecture will need to be modified to improve performance. Such design flaws may be unacceptable, especially in oil and gas production where modification costs may be excessively high. Therefore, it is crucial to minimize uncertainties whenever possible and factor in any remaining uncertainty in control architecture design.

When process data is available, parameter estimation techniques may be used to reduce parametric model uncertainties. Estimation techniques founded on the Bayesian framework, in particular, are powerful: instead of point estimates, they produce posterior probability density functions (PDFs) of the parameters. With regard to control architecture design, posterior PDFs are especially advantageous: they enable description of remaining uncertainties associated with the updated model, which can subsequently be incorporated into the design process.

In Bayesian inference, the likelihood of the observed data is combined with prior PDFs of parameters to obtain posterior PDFs of the parameters. The exact solution of higher dimensional inference problems requires extensive computations that are often intractable. In such cases, the posterior PDFs are commonly approximated using numerical techniques, with Markov Chain Monte Carlo (MCMC) being the most common technique. The basic idea in MCMC sampling is to simulate draws from the posterior distributions and use these to compute various posterior statistics such as means, variances, and quantiles.

Bayesian inference is often implemented using probabilistic programming languages (PPLs) such as the widely popular Stan (Carpenter et al., 2017), the Python-based PyMC (Salvatier et al., 2016), and the relatively new Turing (Ge et al., 2018) written in Julia (Bezanson et al., 2017). These frameworks aim to make the typically laborious inference procedures as automated and efficient as possible, enabling users to concentrate on the model and associated questions rather than the underlying MCMC algorithm mechanics. As such, they serve as the backbone of modern Bayesian analysis.

Simple classical MCMC approaches, like random-walk Metropolis (Metropolis et al., 1953) and Gibbs sampling (Geman and Geman, 1984), use inefficient random walks to explore the parameter space. As a result, it may take too long for these methods to converge to high-dimensional posterior PDFs. Modern PPLs all feature a family of MCMC algorithms called Hamiltonian Monte Carlo (HMC) (Duane et al., 1987) that is widely applied in the field of Bayesian inference. This class of samplers promise better efficiency and faster inference than the simple approaches, which makes them suitable for estimating complex high-dimensional posterior PDFs. The HMC algorithm does away with this random walk behavior. Instead, it uses first-order gradient information from the likelihood to inform its every step; this enables HMC to estimate high-dimensional posterior PDFs considerably faster than the simple methods.

The original HMC algorithm requires expert, hands-on tuning to be efficient. Hoffman and Gelman (2014) proposed the No-U-Turn Sampler (NUTS) to overcome this hurdle. NUTS is an extension of the original HMC sampler and automatically tunes two key parameters — the so-called leapfrog step size and the integration time (also called trajectory length) — which otherwise have to be tuned by hand through many costly preliminary runs.

Despite the potential of HMC and its availability via modern PPLs, adoption has been slow in the oil and gas industry — likely because the original HMC algorithm requires time-consuming, expert hands-on tuning to be efficient, and because the tuning-free NUTS is relatively new. Most studies are restricted to the use of custom implementations of basic, quick-to-implement MCMC techniques, such as the random-walk Metropolis method (Ban et al., 2022; Ruiz Maraggi et al., 2022; Pan et al., 2021; Lødøen and Tjelmeland, 2007). There are only a handful of studies that apply HMC algorithms for parameter estimation, e.g., Taghavi and Ghaderi (2022) use HMC via Stan for a dimensionless model of a rate controlled production valve; Sandl et al. (2021) use HMC via Stan for logistic regression models that predict the occurrence of gas migration in oil wells; and Moen et al. (2022) use DynamicHMC (Papp, 2021) — an implementation of a variant of the NUTS algorithm proposed by Betancourt (2017) — via Turing for a model that estimates the inflow profiles in oil wells.

In this work, we look at a gas-lifted oil field model,

similar to the one used by Ban et al. (2022). The primary aim is to use HMC techniques to reduce prior model uncertainty by use of data, into a posterior uncertainty given by posterior PDFs. We illustrate how Turing can be used for parameter estimation, without requiring expert knowledge of the underlying HMC mechanisms.

The paper is organized as follows. In Section 2, the system under study is detailed, together with model structure and information about uncertainty; in Section 3, experiments are discussed together with MCMC tool options; in Section 4, results on parameter estimation and model uncertainty are given; in Section 5, some conclusions are drawn.

2 Gas Lifted Oil Field

A model of a gas lifted oil field with five oil wells, which has been appropriately validated against data from a real oil field, is proposed by Sharma et al. (2011). An adaptation of the said model, presented in (Jayamanne, 2021), is used for the purposes of this paper; a schematic is given in Fig. 1.

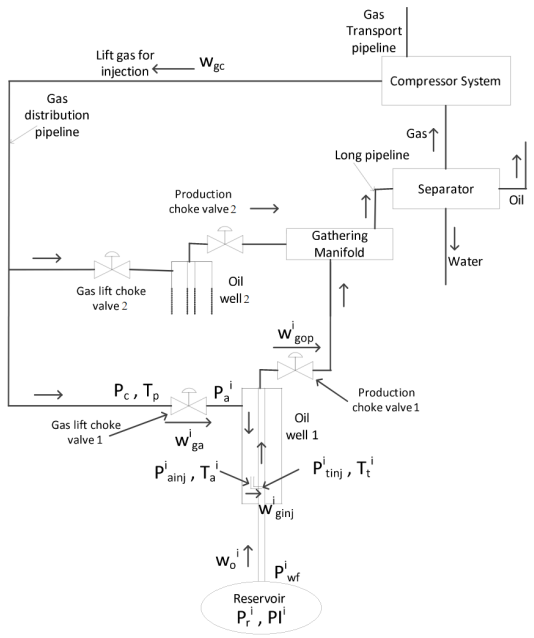


Figure 1: Schematic of the oil field adapted from Sharma et al. (2011).

Two oil wells sharing the same source of lift-gas make up the system. The fluid produced by both wells — a combination of lift gas, water, and oil — is delivered to a separator, which separates the fluid into its components; the lift-gas is recycled.

2.1 Model Form

The system is modelled by a set of Differential-Algebraic Equations (DAE). Essentially, the model consists of mass balances for the states, and algebraic equations for valve characteristics, pressures, average densities, etc. There are three states for each of the two wells i: mass of lift-gas in the annulus m^i_{ga}; mass of lift-gas in the tubing above the injection point m^i_{gt}; and mass of oil in the tubing above the injection point m^i_{ot}. See Jayamanne (2021) for the complete description of the mathematical model, and Sharma et al. (2011) for details about the development of the model.

There are two control inputs to the process: the mass flow rates of gas through the gas-lift choke valves w^i_{ga}. There are six measured variables in this study, the pressure downstream of the gas-lift choke valves P^i_{a}, the mass flow rate of the mixture of gas and oil through the production choke valve w^i_{gop}, and the pressure upstream of the production choke valve P^i_{wh}, one for each well.

The main application of the oil field model is to determine the optimal operating conditions that maximize the profit obtained from the oil field, which is dependent on the total amount of oil extracted from the field. Thus, this is the model output that is of highest interest.

2.2 Sources of Uncertainty

The primary sources of uncertainty that affect a system may be determined by performing a basic sensitivity analysis, e.g., using a tool such as the GlobalSensitivity package (Dixit and Rackauckas, 2022) in Julia.

Here, we omit the details of identifying the main sources of uncertainty in the oil field model, but it is possible to reduce them to three parameter sets: the initial states of the system $m^i_{\mathrm{ga}}(t=0)$, $m^i_{\mathrm{gt}}(t=0)$, and $m^i_{\mathrm{ot}}(t=0)$; the productivity index PI^i—which expresses a wells capacity to produce fluid flow; and the reservoir pressure P_{r}. In addition, we consider the measurement variances $\mathrm{Var}(P_{\mathrm{a}})$, $\mathrm{Var}(w_{\mathrm{gop}})$, and $\mathrm{Var}(P_{\mathrm{wh}})$ to be unknown.

All of these uncertain parameters are assumed to be time-invariant. Everything else is assumed to be known with absolute certainty.

2.3 Nominal Model

For the purposes of this study, we use the nominal model defined by Jayamanne (2021) as a substitute for the real plant. Process sensors are simulated by adding white Gaussian noise to the outputs from this nominal model.

The actual values of the uncertain parameters that are to be estimated, are given in the second column of Table 2.

3 Methods and Tools

In this section, we provide a description of the core concepts underlying MCMC and its associated terminology. In addition, we briefly explain how a parameter estimation problem is formulated in Turing and define the functions, keyword arguments, and macros necessary for performing inference using HMC and NUTS.

We remark that one of the primary motivations for using a PPL in this study is to implement MCMC with minimal domain knowledge. Therefore, only the most essential information required for implementation is presented here.

3.1 MCMC

MCMC combines two concepts: *Markov Chains* and *Monte Carlo* simulations.

A Markov Chain is a mathematical process involving random transitions from one state to another in a chain. One of its defining characteristics is that the next state X_{n+1} depends solely on the current state X_n — not on those that came before it: $X_{n-1}, ..., X_0$. This is referred to as the *Markov property*. In mathematical terms, we have

$$P\left(X_{n+1} = x_{n+1} | X_n = x_n, \cdots, X_0 = x_0\right) = P\left(X_{n+1} = x_{n+1} | X_n = x_n\right);\ n \in \mathbb{Z}_0^+.$$

A Markov Chain also has what is known as a *stationary distribution* $p(x)$. After a time of jumping from one state to another, called the *burn-in period*, the chain will *converge* to its stationary distribution no matter what state it started in. When it does, it will stay at this distribution for all subsequent samples. A simple illustration of a Markov Chain is provided in Figure 2.

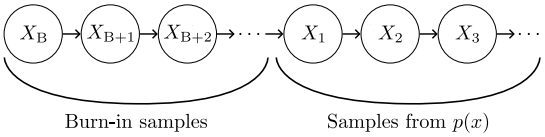


Figure 2: Markov Chain.

In MCMC, we engineer a Markov Chain whose stationary distribution is the posterior PDF that we want to sample from. This means that after a period of burn-in, the Markov Chain is going to simulate draws from the posterior PDF. When it does, we use the Monte Carlo method to approximate the posterior PDF. Essentially, we record many samples from the converged Markov Chain and take their distribution to be equivalent to the posterior PDF.

The general steps of an MCMC algorithm are as follows.

1. Initialization:
   - Choose the total number of *iterations* N.
   - Create an N-vector x for storing the state (parameters).
   - Set the iteration count to $i = 1$.
   - Draw an initial state (parameters) x_i randomly from the assumed prior distribution and set $x\left(i\right) = x_i$.
2. Propose a next state $\hat{x}_{i+1}$ based on the current state x_i — different MCMC algorithms do this differently (Metropolis-Hastings, HMC, etc.).
3. Assess whether the proposed state $\hat{x}_{i+1}$ should be accepted (i.e., explain the data better than the current state x_i).
   a) Calculate the *acceptance* probability p_a based on the likelihoods of the current state x_i and the proposed state $\hat{x}_{i+1}$. This calculation involves, among other things, simulating the system outputs using the current and proposed states.
   b) Draw a random number p_u from the uniform distribution $\mathbb{U}_{[0,1)}$.
   c) Decide on the proposed state:
   if $p_\mathrm{a} \geq p_\mathrm{u}$
   - accept the proposed state, i.e., $x\left(i+1\right) = \hat{x}_{i+1}$
   - set $p_\mathrm{c} = p_\mathrm{a}$

   else
   - reject the proposed state, i.e., $x\left(i+1\right) = x_i$
   - maintain p_c as before

   end
4. Increment i by 1 and if $i < N$, set $x_i = x\left(i+1\right)$, and go to step 2.

Here, it is important to choose N sufficiently large so that the state converges to its stationary distribution. The initial burn-in iterates are then removed from collection x, leaving a smaller *chain* of N_po iterates x_po in the stationary, posterior distribution. From the law of large numbers: the more iterations N_po that are in

the stationary distribution chain, the more accurate the approximate posterior distribution is.

For some samplers, Turing automatically removes the burn-ins, e.g., NUTS (but not for HMC).

It is always possible that the MCMC algorithm converges to an incorrect chain of iterates. Because of this, it is usually necessary to compute multiple chains and check whether the chains converge to the same distribution. Typically, 3 or more chains are needed for such an assessment.

3.2 Turing

To perform parameter estimation using Turing, the statistical model that should be used for generating samples must be specified using the `@model` macro. The general syntax of a Turing model is as follows.

```
@model function prediction_model(y_d)
    # Specify prior distribution
    θ ~ P_r(θ)

    # Simulate process output
    y = dynamic_model(θ)

    # Output: dynamic model + noise
    for i in 1:length(y_d)
        y_d[i] ~ 𝒩(y[i], σ_y^2)
    end
end
```

Once we have defined the statistical model, MCMC sampling can be performed using the `sample` function, which has the form

```
sample(prediction_model, sampler, parallel, N, nchains).
```

This generates `nchains` number of independent chains, each contaning `N` samples, using the statistical `model` and specified MCMC `sampler`. The sampling is performed in parallel using multiple cores if a `parallel` algorithm is specified.

Turing offers several different MCMC samplers. Among them are two classes of HMC: AdvancedHMC (Xu et al., 2020) and DynamicHMC (Papp, 2021). In this work, we use implementations from AdvancedHMC: the HMC sampler (which is different from the HMC class) and NUTS. Section 3.2.1 and Section 3.2.2 below detail their use.

3.2.1 Hamiltonian Monte Carlo (HMC)

The HMC sampler function from AdvancedHMC has the form

```
HMC(ϵ, L),
```

where

`ϵ` is the leapfrog step size, and

`L` is the number of leapfrog steps.

The appropriate values for ϵ and `L`, which determine trajectory length ϵ`L`, are typically selected by monitoring the acceptance rate; Brooks et al. (2011) found that the optimal balance between the two parameters occurs when the acceptance rate is around 65%. In addition to the acceptance rate, it is also helpful to examine the trace plots of MCMC chains: slow-moving chains often indicate a too-short trajectory length ϵ`L`.

While they may be tuned simultaneously, selecting ϵ first — while keeping `L` fixed — and then fine-tuning `L` is preferable. A small ϵ will result in a high acceptance rate. But to ensure that the trajectory length ϵ`L` is sufficiently long to move to a distant point in the parameter space, a small ϵ would have to be coupled with a large `L`; this is computationally more expensive. Ideally, we would want the largest possible value of ϵ that yields a reasonable acceptance rate.

Our choices of parameters are discussed in Section 4.4.

3.2.2 No-U-Turn Sampler (NUTS)

The NUTS sampler function from AdvancedHMC has the form

```
NUTS(n_adapts, δ),
```

where

`n_adapts` is the number of adaptation samples,

δ is the target acceptance rate for dual averaging.

A common practice is to adjust any tunable MCMC parameters during the burn-in phase and to lock them in place thereafter. With this practice, `n_adapts` is the same as the burn-in period. (Note: Turing discards the adaptation samples of NUTS by default. I.e., the resulting chains provided by Turing do not include the adaptation samples.)

Hoffman and Gelman (2014) found that NUTS's optimal performance occurs around $\delta = 0.6$, but depends little on δ within the range $\delta \in [0.45, 0.65]$.

Our choices of parameters are discussed in Section 4.4.

4 Results and Discussion

4.1 Parameter Priors

Extreme values of uncertainty ranges are less likely to occur than intermediate values. Hence, we use Gaussian distributions with specified means and standard deviations as priors. The mean values are chosen based on best physical knowledge of the system

Table 1: Prior uncertainty description.

Statistics	Uncertain model parameters								
	m^1_{ga} [kg]	m^2_{ga} [kg]	m^1_{gt} [kg]	m^2_{gt} [kg]	m^1_{ot} [kg]	m^2_{ot} [kg]	PI^1 [kg/hr/bar]	PI^2 [kg/hr/bar]	P_r [bar]
μ	19,000	20,000	1,050	1,350	23,000	21,000	26,000	15,000	151
σ	1,900	2,000	105	135	2,300	2,100	6,500	3,750	7.55
min	15,200	16,000	840	1,080	18,400	16,800	13,000	7,500	135.9
max	22,800	24,000	1,260	1,620	27,600	25,200	39,000	22,500	166.1

— here, somewhat arbitrarily because the actual values are known. The standard deviations for the initial states, productivity indices, and reservoir pressure are taken to be 10%, 25%, and 5% of their chosen mean values, respectively. We truncate these distributions at two standard deviations from the mean value on either end.

It is standard to use the inverse gamma distribution as prior for measurement noise variances.

Table 1 provides a summary of the prior uncertainty description of the parameters to be estimated.

For the measurement noise variances, the chosen priors are $\text{Var}(P^i_{\text{a}}) \sim \Gamma^{-1}(1, 15)$, $\text{Var}(w^i_{\text{gop}}) \sim \Gamma^{-1}(1, 1)$, $\text{Var}(P^i_{\text{wh}}) \sim \Gamma^{-1}(1, 0.05)$.

4.2 Synthetic Data Generation

Generation of data for parameter estimation is done using the following step input signal.

$$w^i_{\text{ga}}(t) = a - b\mathbb{H}_{t_1} + c\mathbb{H}_{t_2}$$

where $\mathbb{H}_t$ is the Heaviside function, and

$$\begin{aligned} a &= \frac{20\,000 \times 0.83}{60 \times 60} \text{ kg/s} \\ b &= \frac{2\,000 \times 0.83}{60 \times 60} \text{ kg/s} \\ c &= \frac{1\,000 \times 0.83}{60 \times 60} \text{ kg/s} \\ t_1 &= 2\,000 \text{ min} \\ t_2 &= 4\,000 \text{ min} \end{aligned}$$

We simulate the nominal oil field model using the step input for 100 hours. The outputs are sampled every half-hour, yielding a total of 201 data points. We then introduce white Gaussian noise to the gathered samples, see column two of Table 2. Figure 7 shows the generated data as well as the nominal model outputs — to which the Gaussian noise was added.

4.3 Prior Model Uncertainty Analysis

It is useful to generate potential scenarios using random samples from the prior distributions and compare them with the actual case; the spread of the results would indicate whether the initial parameter space leads to a realistic representation of the uncertainty in model predictions. Here, we solve the model for 10 000 prior samples selected at random, over a period of 100 hours using the step input signals described in Section 4.2.

In Fig. 3, the oil production rate from the 10,000 ensemble members is plotted in grey, and the actual oil production rate—based on actual parameters—is plotted in blue; the upper plot shows the total production of the field, while the two lower plots show the individual production of the two wells. Clearly, the uncertainties in the initial parameter space lead to a large degree of uncertainty in the model outputs, but they also capture the magnitude of the actual output.

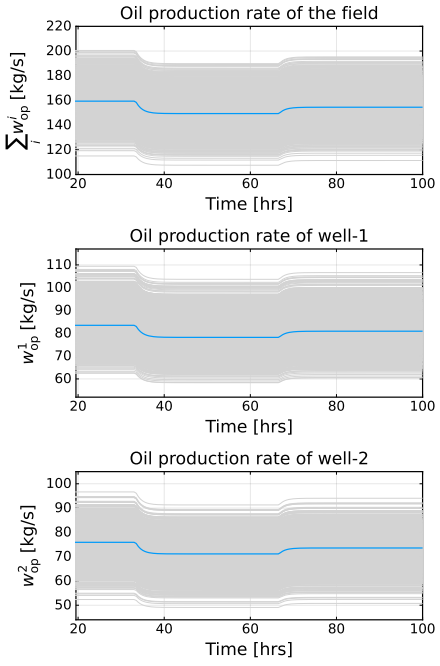


Figure 3: Prior model outputs.

It is also useful to apply statistical methods to analyse the prior predictions of *accumulated* field produc-

tion at the end of 100 hours–omitting the first 20 hours during which the system has not yet reached steady state. Figure 4 shows the histogram of the field production along with some statistics; the red vertical line shows the actual accumulated field production of 4.44×10^7 kg, while the blue line shows the mean of the prior set.

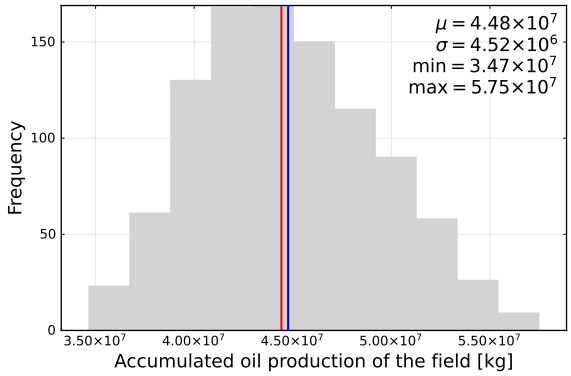


Figure 4: Prior model uncertainty. Red line is the actual production.

4.4 Parameter Estimation

We set up the parameter estimation problem in Turing using the default automatic differentiation backend, ForwardDiff (Revels et al., 2016). The DifferentialEquations package (Rackauckas and Nie, 2017) is used for implementing the oil field model.

We run HMC with $L = 10$ leapfrog steps and a step size of $\epsilon = 0.004$ — the best tuning found through several preliminary runs — and NUTS with a target acceptance rate of $\delta = 0.6$ — the default value recommended by Hoffman and Gelman (2014). In both cases, we use the MCMCThreads algorithm to sample multiple chains in parallel; 3 chains for NUTS, and 10 chains for HMC — additional chains are required for HMC since some of them fail to converge within the specified number of iterations (note: only the first 3 chains that reach convergence are presented here). The HMC chains are run for 5000 iterations, where we chose to discard the first 1000 as burn-in. The NUTS chains are run for 750 iterations, with the first 250 automatically discarded as burn-in.

For HMC, the call to the sample function is thus

```
sample(prediction_model, HMC(0.004, 10), MCMCThreads(),
                        5000, 10).
```

For NUTS, the call to the sample function is thus

```
sample(prediction_model, NUTS(250, 0.60), MCMCThreads()
                        , 750, 3).
```

Sampling is performed on a 2.40 GHz laptop workstation with 32 GB memory.

Figure 5a shows the trace plots (evolution of iterations) and posterior PDFs produced by HMC, and Fig. 5b shows the same produced by NUTS. The plots show that the chains are similar, which suggests successful convergence to target posterior distributions. A visual overview of how well the algorithms estimated the parameters is provided in Fig. 6a for HMC and Fig. 6b for NUTS; they illustrate how the calculated posterior PDFs compare to the actual parameter values and the prior PDFs. Table 2 gives the posterior statistics.

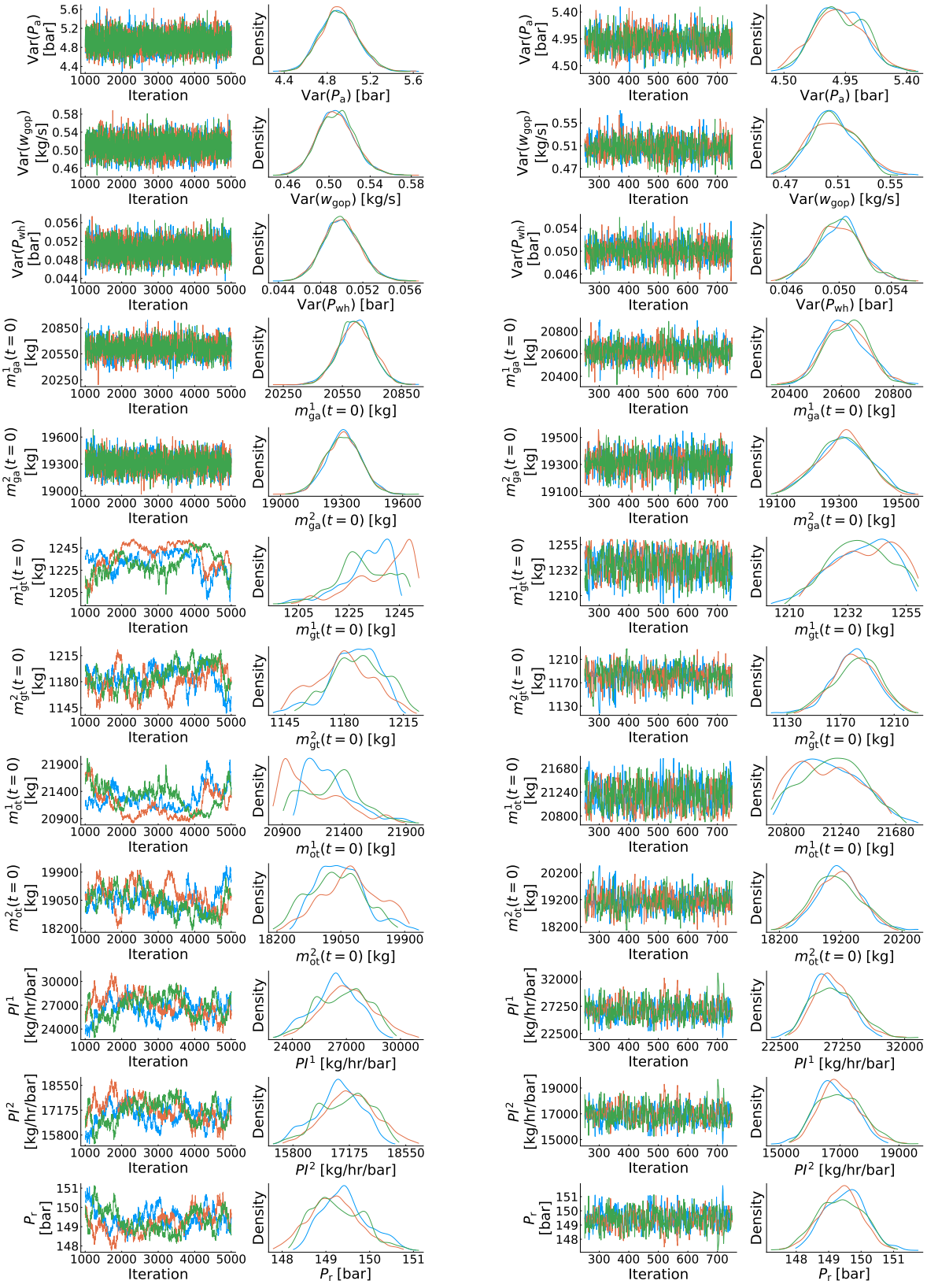
The spread of each posterior is observed to be much smaller than that of the corresponding prior. Moreover, all posterior modes either coincide with or are in close proximity to the actual parameter values. These findings suggest that we are able to retrieve the actual parameter values with substantially less uncertainty than in the prior, using either strategy.

While it is quite certain that our tuning of HMC presented here can be optimized, we believe it is still worthwhile to compare the performance of HMC and NUTS. The comparison would answer the question of how a naively tuned HMC sampler stacks up against NUTS. Figure 5 – 6b, and Table 2 indicate that the performance of the two samplers is comparable in terms of posterior mean values and standard deviations. It is then interesting to see how they compare in terms of efficiency. The Effective Sample Size per unit time (ESS/t) is a standard method for assessing and comparing the efficiency of different MCMC algorithms; ESS/t is automatically reported when running the sample function. This efficiency metric roughly corresponds to the number of independent samples generated per unit of time. ESS/t values for HMC and NUTS are given in Table 2; although the figures for measurement variances are similar, the disparity in efficiency becomes evident when comparing the values for the other estimated parameters: NUTS is far more efficient than the naively tuned HMC sampler.

4.5 Data Retrodiction

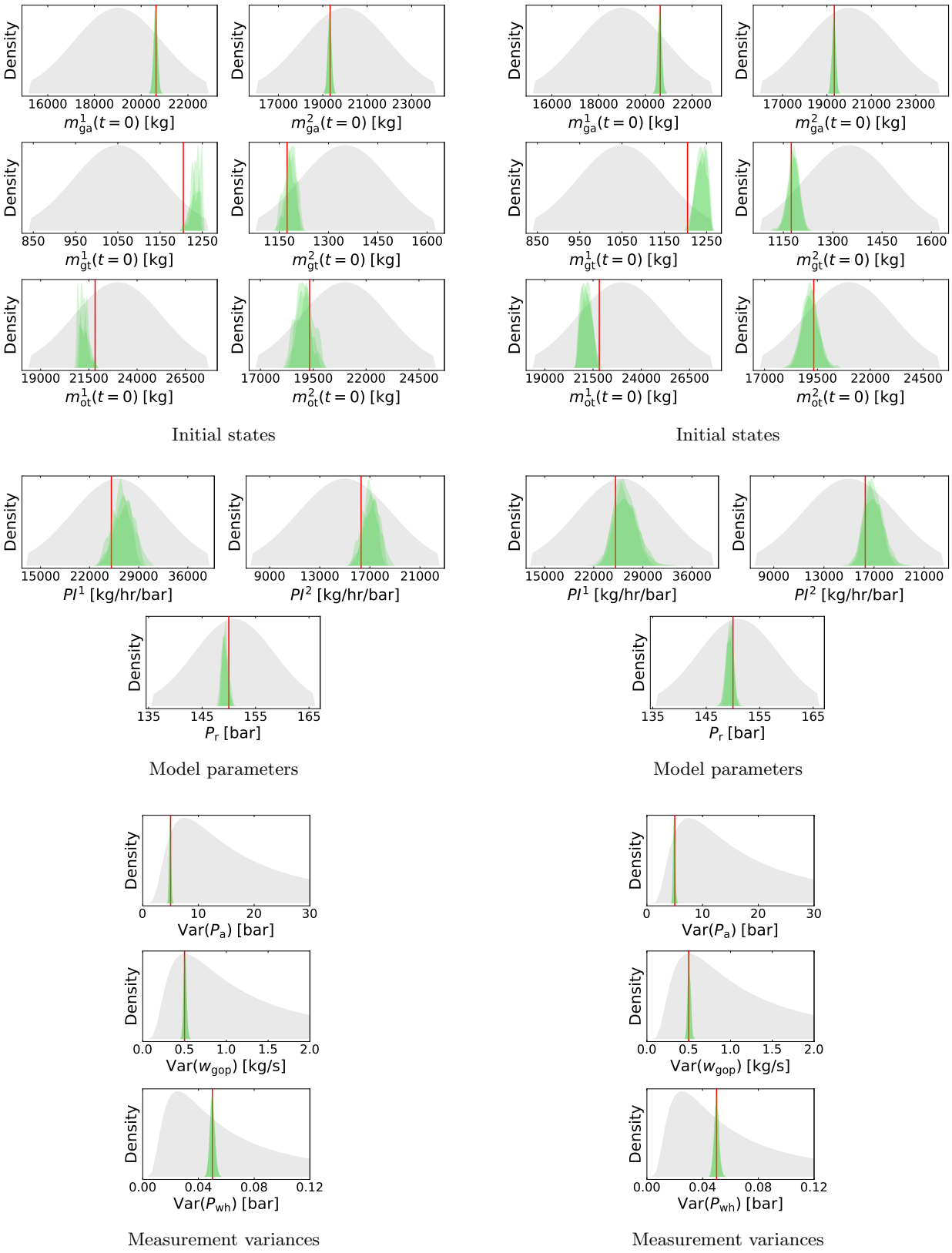
It is useful to perform retrodiction, i.e., simulate the model for random parameter values drawn from the posterior PDFs: the results, when compared with the original data or the actual model solution, would provide an indication of how good the model fits is.

Here, we retrodict the measured variables for 10,000 random samples from the posterior PDFs. In Fig. 7, the results are compared with the original noisy measurement data — which was used for parameter estimation — as well as the actual model solution — which was used to create the synthetic measurement data.



(a) HMC (b) NUTS

Figure 5: Trace plots and posterior PDFs. NOTE: The burn-in samples are not shown.



(a) HMC

(b) NUTS

Figure 6: Prior PDFs (grey) vs. posterior PDFs (green) vs. actual values (red).

Table 2: Comparison of NUTS and HMC, posterior statistics.

	Actual	μ		σ		ESS/t	
Parameter	Value	NUTS	HMC	NUTS	HMC	NUTS	HMC
$\mathrm{Var}(P_{\mathrm{a}})$	5	4.9043	4.9100	0.1801	0.1742	0.0505	0.0648
$\mathrm{Var}(w_{\mathrm{gop}})$	0.5	0.5066	0.5070	0.0182	0.0180	0.0650	0.0520
$\mathrm{Var}(P_{\mathrm{wh}})$	0.005	0.0500	0.0500	0.0017	0.0018	0.0739	0.0626
$m^1_{\mathrm{ga}}(t=0)$	20,636	20,617	20,617	85	87	0.0404	0.0253
$m^2_{\mathrm{ga}}(t=0)$	19,331	19,312	19,308	83	86	0.0780	0.0393
$m^1_{\mathrm{gt}}(t=0)$	1,205	1,238	1,234	12.4	10.9	0.0183	0.0007
$m^2_{\mathrm{gt}}(t=0)$	1,174	1,181	1,184	17.1	16.3	0.0233	0.0010
$m^1_{\mathrm{ot}}(t=0)$	21,824	21,153	21,244	260	229	0.0183	0.0007
$m^2_{\mathrm{ot}}(t=0)$	19,325	19,154	19,072	378	364	0.0233	0.0010
PI^1	25,100	26,517	26,761	1600	1373	0.0229	0.0009
PI^2	16,300	16,947	17,059	716	615	0.0231	0.0010
P_r	150	149.41	149.30	6.6	5.7	0.0228	0.0010

The figure reveals that, despite introducing a substantial amount of noise into the data, the computed posterior PDFs reproduce the actual model solutions quite well.

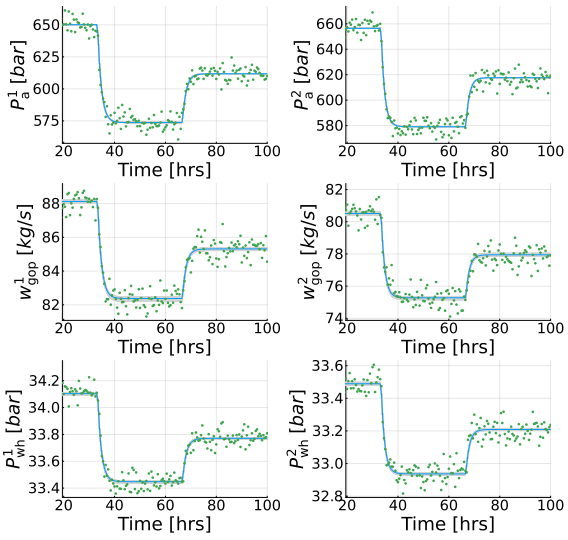


Figure 7: Data retrodiction of measured outputs (grey) compared to the original noisy data (green) and the nominal model solution that was used to generate the data (blue).

4.6 Posterior Model Uncertainty Analysis

Figure 3 indicates the predictive uncertainty when drawing parameters from the prior distribution. We now retrodict the model output for 10,000 random samples from the posterior distribution. Figure 8 shows predictive abilities when drawing from the posterior distributions, which should be compared with Fig. 3. We also calculate the accumulated field production at the end of 100 hours for each posterior ensemble member; Fig. 9 shows the resulting histogram. Compared to the prior outputs in Fig. 3 and 4, Fig. 8 and 9 show that the posterior outputs are markedly less dispersed about the actual output.

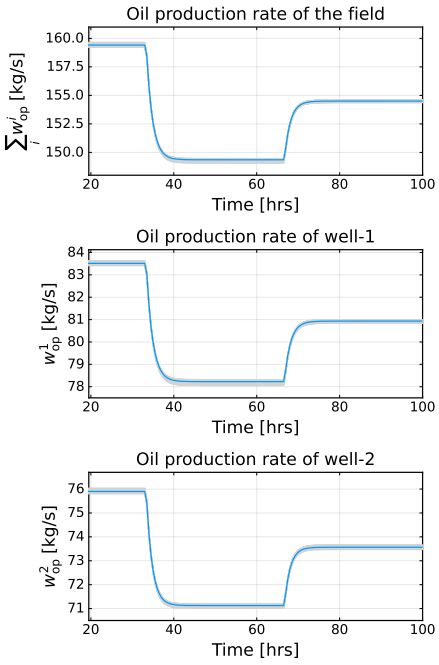


Figure 8: Data retrodiction of oil production (grey) vs. actual production (blue).

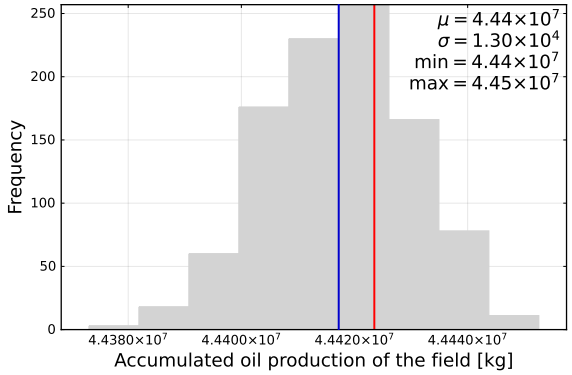


Figure 9: Posterior model uncertainty. NOTE the scale.

5 Conclusions

In this paper we considered the problem of reducing the uncertainty of a gas lifted oil-field model using synthetic process data and assumed priors. The data consists of time series of 2 inputs and 6 noisy outputs. The model has 6 states with poorly known initial values, and 3 unknown model parameters. In the Bayesian inference problem, we thus considered 12 unknown parameters: the 6 initial states, the 3 model parameters, and the 3 measurement variances, and postulated some prior distributions for these 12 parameters.

Ban et al. (2022) studied the same problem of uncertainty reduction with a somewhat different model. They implemented the random-walk Metropolis MCMC algorithm from scratch to accomplish parameter estimation. In this study, we opted for a more sophisticated family of Markov Chain Monte Carlo samplers via the probabilistic programming language Turing.

The original HMC sampler and its "tuning-free" variant, NUTS, were used to estimate the values of the 12 parameters. Results demonstrated that both samplers are capable of computing posterior distributions for the parameters that are significantly narrower than the assumed prior distributions. The generated posterior distributions were then utilized to show that the predictive capabilities of the model (retrodiction) had been improved considerably compared to using the prior distributions. This implies that both MCMC samplers result in accurate estimates of the uncertain parameters.

When using real data instead of synthetic data, poorer predictive capabilities can be expected based on the posterior distributions. However, the demonstrated strategy will still give a best possible estimate of the model uncertainty.

The work presented here may be viewed as a first step toward integrating system uncertainty into a comprehensive framework for optimum control architecture design.

Acknowledgments

We gratefully acknowledge the economic support from The Research Council of Norway and Equinor ASA through Research Council project 308817 - Digital wells for optimal production and drainage (DigiWell).

References

Ban, Z., Ghaderi, A., Janatian, N., and Pfeiffer, C. F. Parameter Estimation for a Gas Lifting Oil Well Model Using Bayes' Rule and the MetropolisHastings Algorithm. *Modeling, Identification and Control*, 2022. 43(2):39–53. doi:10.4173/mic.2022.2.1.

Betancourt, M. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017. doi:10.48550/arXiv.1701.02434.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 2017. 59(1):65–98. doi:10.1137/141000671.

Brooks, S., Gelman, A., Jones, G., and Meng, X. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, 2011.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*, 2017. 76(1):132. doi:10.18637/jss.v076.i01.

Dhingra, N. K., Jovanovi, M. R., and Luo, Z.-Q. An ADMM algorithm for optimal sensor and actuator selection. In *53rd IEEE Conference on Decision and Control*. pages 4039–4044, 2014. doi:10.1109/CDC.2014.7040017.

Dixit, V. K. and Rackauckas, C. GlobalSensitivity.jl: Performant and Parallel Global Sensitivity Analysis with Julia. *Journal of Open Source Software*, 2022. 7(76):4561. doi:10.21105/joss.04561.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. Hybrid Monte Carlo. *Physics Letters B*, 1987. 195(2):216–222. doi:10.1016/0370-2693(87)91197-X.

Ge, H., Xu, K., and Ghahramani, Z. Turing: a language for flexible probabilistic inference. In *International Conference on Artificial Intelligence and*

Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain. pages 1682–1690, 2018. URL http://proceedings.mlr.press/v84/ge18b.html.

Geman, S. and Geman, D. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1984. PAMI-6(6):721–741. doi:10.1109/TPAMI.1984.4767596.

Goodwin, G. C., Graebe, S. F., and Salgado, M. E. *Control system design*. Prentice Hall Upper Saddle River, 2001.

Hoffman, M. D. and Gelman, A. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 2014. 15(47):1593–1623. URL http://jmlr.org/papers/v15/hoffman14a.html.

Jayamanne, K. R. *Optimal Operation of Processes Under Uncertainty Using Robust Model Predictive Control*. Master's thesis, University of South-Eastern Norway, 2021. URL https://hdl.handle.net/11250/2765105.

Lødøen, O. P. and Tjelmeland, H. Bayesian calibration of reservoir models using a coarse-scale reservoir simulator in the prior specification. In *EAGE Conference on Petroleum Geostatistics*. 2007. doi:10.3997/2214-4609.201403057.

Manohar, K., Kutz, J. N., and Brunton, S. L. Optimal Sensor and Actuator Selection Using Balanced Model Reduction. *IEEE Transactions on Automatic Control*, 2021. 67(4):2108–2115. doi:10.1109/TAC.2021.3082502.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 1953. 21(6):1087–1092. doi:10.1063/1.1699114.

Moen, O. R., Mo, S., and Varagnolo, D. A Bayesian Approach for Improved Estimation of Inflow Profiles in Oil Wells. In *SPE Norway Subsurface Conference*. OnePetro, 2022. doi:10.2118/209573-MS.

Muske, K. R. and Georgakis, C. Optimal measurement system design for chemical processes. *AIChE Journal*, 2003. 49(6):1488–1494. doi:10.1002/aic.690490612.

Pan, Y., Li, G., Qin, J., Zhang, J., Deng, L., and Bi, R. A Novel Probabilistic Approach for GOR Forecast in Unconventional Oil Reservoirs. In *Unconventional Resources Technology Conference, 26–28 July 2021*. Unconventional Resources Technology Conference (URTeC), pages 1811–1830, 2021. doi:10.15530/urtec-2021-5068.

Papp, T. DynamicHMC.jl. https://www.tamaspapp.eu/DynamicHMC.jl/dev/, 2021. (Accessed: 27 October 2022).

Rackauckas, C. and Nie, Q. DifferentialEquations.jl A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *The Journal of Open Research Software*, 2017. 5(1):15. doi:10.5334/jors.151.

Revels, J., Lubin, M., and Papamarkou, T. Forward-Mode Automatic Differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016. URL https://arxiv.org/abs/1607.07892.

Ruiz Maraggi, L. M., Lake, L. W., and Walsh, M. P. A Bayesian Framework for Addressing the Uncertainty in Production Forecasts of Tight-Oil Reservoirs Using a Physics-Based Two-Phase Flow Model. *SPE Reservoir Evaluation & Engineering*, 2022. 25(03):486–508. doi:10.2118/209203-PA.

Sakha, M. S. and Shaker, H. R. Optimal sensors and actuators placement for large-scale unstable systems via restricted genetic algorithm. *Engineering Computations*, 2017. doi:10.1108/EC-04-2016-0138.

Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2016. 2:e55. doi:10.7717/peerj-cs.55.

Sandl, E., Cahill, A., Welch, L., and Beckie, R. Characterizing oil and gas wells with fugitive gas migration through Bayesian multilevel logistic regression. *Science of The Total Environment*, 2021. 769:144678. doi:10.1016/j.scitotenv.2020.144678.

Sharma, R., Fjalestad, K., and Glemmestad, B. Modeling and control of gas lifted oil field with five oil wells. In *52nd International Conference of Scandinavian Simulation Society, SIMS*. pages 29–30, 2011.

Taghavi, S. and Ghaderi, A. On Uncertainty Analysis of the Rate Controlled Production (RCP) Model. *Scandinavian Simulation Society*, 2022. pages 271–278. doi:10.3384/ecp21185271.

Xu, K., Ge, H., Tebbutt, W., Tarek, M., Trapp, M., and Ghahramani, Z. AdvancedHMC.jl: A robust, modular and efficient implementation of advanced HMC algorithms. In *Symposium on Advances in*

Approximate Bayesian Inference. PMLR, pages 1–10, 2020. URL https://proceedings.mlr.press/v118/xu20a.html.

Zare, A., Mohammadi, H., Dhingra, N. K., Georgiou, T. T., and Jovanovi, M. R. Proximal Algorithms for Large-Scale Statistical Modeling and Sensor/Actuator Selection. *IEEE Transactions on Automatic Control*, 2020. 65(8):3441–3456. doi:10.1109/TAC.2019.2948268.