Variational inference for nonlinear ordinary differential equations: Supplementary Materials

Sanmitra Ghosh[1] **Paul J. Birrell**[2,1] **Daniela De Angelis**[1,2]
{sanmitra.ghosh,paul.birrell,daniela.deangelis}@mrc-bsu.cam.ac.uk
[1]MRC Biostatistics Unit, University of Cambridge, [2]Public Health England, Colindale

9 Appendix A

9.1 Chemical Langevin equation

Over an infinitesimal interval $(t, t+dt]$ the reaction hazards will remain constant almost surely (Golightly and Gillespie, 2013). Thus, the occurrence of a reaction event can be considered as the occurrence of an event of a Poisson process with independent realisations for each type of reactions. Therefore we can write as dR_t the v-dimensional vector of the number reaction events of each type within the infinitesimal time interval. The i-th element of this vector is simply a Poisson$(h_i(\mathcal{X}_t, c_i)dt$ random variable, where $i = 1, \ldots, v$. Hence the corresponding mean and variance of dR_t are given by

$$\mathbb{E}[dR_t] = h(\mathcal{X}_t, \boldsymbol{c})dt, \quad \text{Var}[dR_t] = \text{diag}(h(\mathcal{X}_t, \boldsymbol{c})), \tag{27}$$

and thus we can write

$$dR_t = h(\mathcal{X}_t, \boldsymbol{c})dt + \text{diag}\left(\sqrt{h(\mathcal{X}_t, \boldsymbol{c})}\right)dW_t, \tag{28}$$

as an Ito stochastic differential equation, where W_t is a v-dimensional vector of standard Brownian motion, that has the same mean and variance as the true MJP describing the stochastic kinetic system. Since we know that $d\mathcal{X}_t = SdR_t$, therefore the increment in the number of molecules is immediately given by

$$d\mathcal{X}_t = Sh(\mathcal{X}_t, \boldsymbol{c})dt + S\,\text{diag}\left(\sqrt{h(\mathcal{X}_t, \boldsymbol{c})}\right)dW_t. \tag{29}$$

Noting that $\text{Var}[d\mathcal{X}_t] = S\,\text{diag}(h(\mathcal{X}_t, \boldsymbol{c}))S^T$, we can write the above SDE in an alternate form given by

$$d\mathcal{X}_t = Sh(\mathcal{X}_t, \boldsymbol{c})dt + \sqrt{S\,\text{diag}(h(\mathcal{X}_t, \boldsymbol{c}))S^T}dW_t, \tag{30}$$

such that both $\mathcal{X}_t$ and W_t are now u-dimensional vectors. This stochastic differential equation is known as the chemical Langevin equation, and represents the diffusion process which most closely matches the dynamics of the associated true MJP. This completes our heuristic derivation of the CLE. For a more formal derivation CLE we refer the reader to Gillespie (2000).

9.2 Further details of LNA

We obtain the LNA by expanding the solution $\mathcal{X}_t$ of the CLE into a deterministic term plus a residual noise as follows:

$$\mathcal{X}_t = \Omega \mathcal{Z}_t + \Omega \mathcal{M}_t. \tag{31}$$

Substituting this into the rescaled CLE given by Equation (13) we obtain

$$d\mathcal{Z}_t + \frac{1}{\sqrt{\Omega}} d\mathcal{M}_t = Sf(\mathcal{Z}_t + \mathcal{M}_t/\sqrt{\Omega}, \boldsymbol{c})dt + \sqrt{S \operatorname{diag}(f(\mathcal{Z}_t + \mathcal{M}_t/\sqrt{\Omega}, \boldsymbol{c}))S^T} dW_t. \tag{32}$$

We then Taylor expand the rate function around $\mathcal{Z}_t$ to obtain

$$f(\mathcal{Z}_t + \mathcal{M}_t/\sqrt{\Omega}, \boldsymbol{c}) = f(\mathcal{Z}_t, \boldsymbol{c}) + \frac{1}{\sqrt{\Omega}} F_t \mathcal{M}_t + \mathcal{O}(\Omega^{-1}). \tag{33}$$

Substituting Equation (33) into Equation (32) and collecting the $\mathcal{O}(1)$ terms gives us the ODE for $\mathcal{Z}_t$ given by

$$\frac{d\mathcal{Z}_t}{dt} = Sf(\mathcal{Z}_t, \boldsymbol{c}), \tag{34}$$

and collecting terms of $\mathcal{O}(1/\sqrt{\Omega})$ gives us the SDE satisfied by the residual process $\mathcal{M}_t$:

$$d\mathcal{M}_t = SF_t \mathcal{M}_t dt + \sqrt{S \operatorname{diag}(f(\mathcal{Z}_t, \boldsymbol{c}))S^T} dW_t. \tag{35}$$

10 Appendix B

10.1 Choice of integration technique for Adjoint sensitivity

Implementing the adjoint sensitivity method requires the integration of three systems. These are the original ODE system $\boldsymbol{f}\big(X(t), \boldsymbol{\theta}\big)$, the vector-matrix products between the adjoint state $\boldsymbol{a}(t)$ and i) the Jacobian of the ODE velocity function: $\frac{\partial \boldsymbol{f}}{\partial X}$ with respect to the system state $X(t)$, and ii) the Jacobian of the velocity: $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}$ with respect to the parameter. Note that integrating the system ii) requires a continuous solution of $X(t)$ for all t and consequently integrating iii) requires continuous solutions of both $X(t)$ and $\boldsymbol{a}(t)$. Clearly, when the original system is large then this becomes infeasible due to high memory costs.

Recently Chen et al. (2018) used the adjoint sensitivity method to extend automatic differentiation to an ODE. Since, the motivating systems in that work were all black-box ODEs with velocity fields described by neural networks, having large state and parameter dimensions, thus continuous solutions could not be used. As a result Chen et al. (2018) simultaneously solved the three systems, mentioned above, backward in time with an adaptive numerical ODE solver. However, this clever reversible formulation assumes that the ODE integrator is reversible. This is not the case for first order adaptive ODE solvers.

One simple technique to avoid reverse solving the system ODEs is the idea of checkpointing. That is we avoid a continuous solution of the original system and simply store $X(t_i)$ for each of the i-th measurement time points. We can then solve the systems ii) and iii) simultaneously and backward in time by first employing a continuous forward solution of the original system starting from the nearest checkpoint and then using this continuous solution to solve in turn ii) and iii). We have provided code for both a full continuous as well as a checkpointed solution of the original system.

We recommend using a checkpointed solution when both K, the size of the original system sate, and D, the parameter dimension, are large and $D > K$. A continuous solution would suffice when K is not large but $D > K$. The need to solve the original system backward in time arises from the extreme constraints posed by a neural network, but for most mechanistic models such constraints may not be relevant.

10.2 Choosing between forward and adjoint sensitivity for evaluating the VJP

Forward sensitivity requires solving $K \times D$ ODEs once. In comparison the adjoint method requires solving $2K + D$ ODEs between each pair of time points. This is since the adjoint ODE needs to be perturbed by the

gradient of the cost function with respect to the ODE solution at each time point. Since the perturbed system is slightly different for each pair of time points the solver might cumulatively incur significantly higher steps (and evaluations of the right hand side) when compared to solving between the first and last time points only once, as in the forward sensitivity method.

For large systems where $K \times D >> 2K + D$ adjoint method should be the default choice. But for smaller systems where $K \times D \approx 2K + D$ adjoint method should be chosen only when the number of calls to the ODE solver is the same if forward method had been used. This was the case for the linear noise approximation example. However, for similar small systems, that is where $K \times D \approx 2K + D$, the forward sensitivity would benefit if significantly more calls to the solver is required for the adjoint method due to handling many measurement time points. Thus, the choice between these methods cannot be trivially reduced to using adjoint method, by default, whenever we have a system where $D > K$.

Our above recommendations are reflect the timing results for our benchmarking examples and are consistent with the findings in Rackauckas et al. (2018). For this reason we have evaluated the VJP using both method. Using our holistic approach one can try a few iterations of variational inference using each of this sensitivity techniques and choose the most suitable one.

10.3 Calculating the state and parameter Jacobians

The state and parameter Jacobians: $\frac{\partial \boldsymbol{f}}{\partial X}$, $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}$ required in both the sensitivity methods can be formed explicitly. This can be done using manual derivation or symbolic differentiation. Alternatively, using automatic differentiation the vector-matrix products (see main text) can be directly evaluated without evaluating the matrices. Since, we have used `SciPy`'s ODE solver thus using the `SymPy` symbolic system for calculating these matrices explicitly was significantly faster than using `PyTorch`'s VJP routine. However, for models with a large state or parameter dimensions avoiding explicit calculation of these matrices may appear beneficial.

We like to point out that writing a solver from scratch and the associated sensitivity methods in `PyTorch`, or any other AD package that supports just-in-time compilation, can be an altogether different approach than the ones we have taken, and mentioned above. Our approach on the other hand can be easily integrated with domain specific modelling software, that use legacy solvers, with just the additional effort of implementing a suitable Python wrapper.

11 Appendix C

11.1 SIR model marginal density plots

Figure 4 compares the marginal distributions, shown as kernel density plots, for the SIR model between **VI** and **NUTS** using forward (figure 4(a)) and adjoint (figure 4(b)) based VJP. Note that these marginal distribution were summarised in Table 1.

11.2 SIR model pairwise joint density plots

Figure 5 shows the pairwise joint density plots for results obtained from running **VI** and **NUTS**. Notice how well the strong correlation is captured by **VI** in comparison to **NUTS**. These plots are based on running **VI** and **NUTS** with forward sensitivity, since this method for VJP was found to be faster than adjoint. However, we get similar density plots (not shown here for brevity) corresponding to adjoint sensitivity based VJP.

11.3 Protein Transduction model marginal density plots

Figure 6 compares the marginal distributions, shown as kernel density plots, for the protein transduction model between **VI** and **NUTS**. Note that these marginal distribution were summarised in Table 1.

11.4 Protein Transduction pairwise joint density plots

In the following figures we show the pairwise joint density plots for results obtained from running **VI** with forward (figure 7) sensitivity.

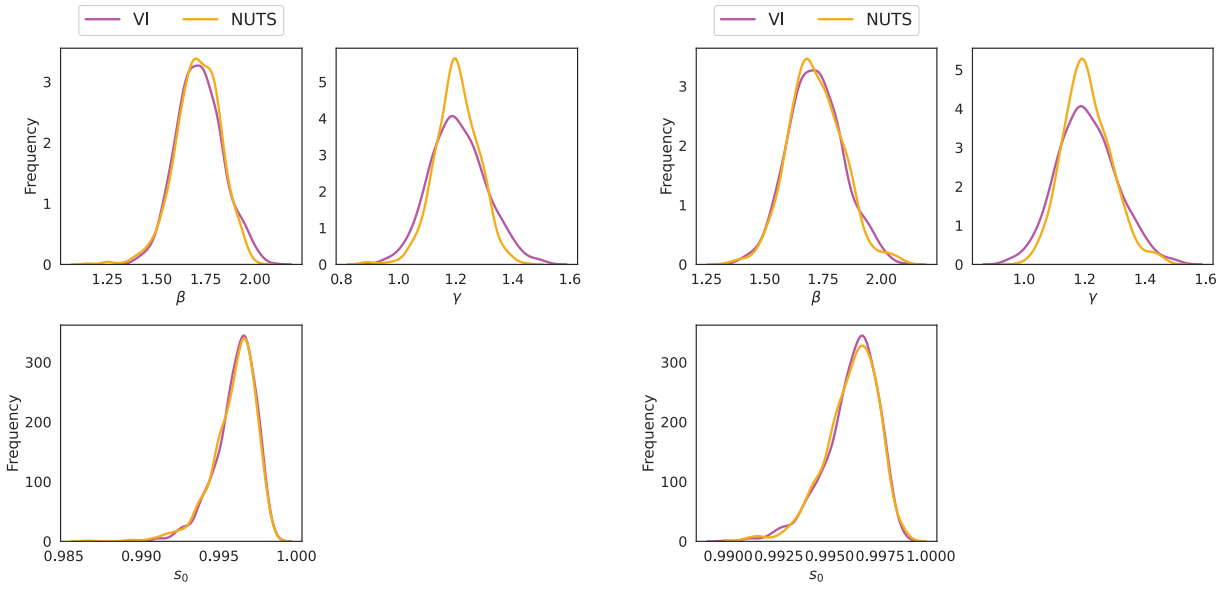


Figure 4: **SIR model** marginal density plots: (a) using **Forward** and (b) using **Adjoint** sensitivity based VJP.

The pairwise plots for the **NUTS** results are shown in figure 8, again using forward sensitivity based VJP. Similar to the SIR model results we notice that **VI** can capture the strong correlations in the posteriors well. See for example the joint densities between parameters p_2, p_3 and p_5, p_6. Such good quality estimates as delivered by our proposed method is crucial in scientific data analysis applications such as uncertainty quantification in complex systems and inverse problems.

11.5 Protein Transduction model fit plots

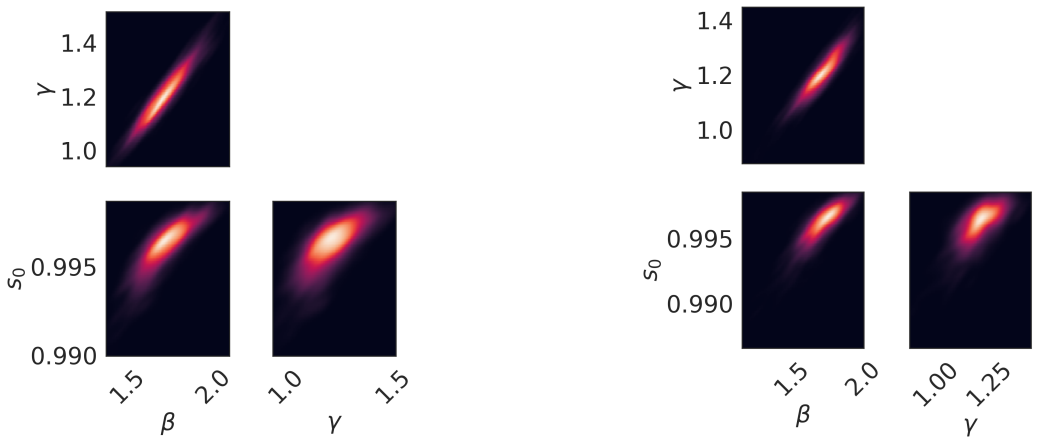
In figure 9 we show the model fit to the simulated data. The model fit is shown by drawing the mean and 95% credible intervals from the posterior predictive distributions. This model fit is based on the results obtained after running **VI** and **NUTS** with **forward** sensitivity based VJP. Again for brevity we have not shown here similar plots corresponding to adjoint sensitivity based VJP, which produces similar results.

11.6 Protein Transduction model additional results

For the protein transduction model, in order to generate the simulated data for benchmarking inference we have used artificial noise corruption. For this reason we repeated the inference processes using two additional realisations of noise corrupted simulated data. We found similar estimates to what we reported in the main text and above while carrying out this repeat of the inference steps using both **VI** and **NUTS**. We did this to ensure that our findings are consistent and not an artefact of artificial noise corruption. In figure 10 we plot the marginal distributions obtained from running **VI** and **NUTS** with one of the additional datasets. The run-time corresponding to **VI-FOR**, **VI-ADJ**, **NUTS-FOR**, **NUTS-ADJ** were found to be 456, 1050, 2420 and 6101 seconds respectively.

11.7 Lotka-Volterra model marginal density plots

Figure 11 compares the marginal distributions, shown as kernel density plots, for the stochastic Lotka-Volterra model between **VI** and **ABC-SMC** using forward and adjoint based VJP.



(a) (b)

Figure 5: **SIR model** pairwise joint density plots: (a) for **VI** and (b) for **NUTS**.

11.8 Lotka-Volterra pairwise joint density plots

Figure 12 shows the pairwise joint density plots for results obtained from running **VI** and **ABC-SMC**. These plots are based on running **VI** with adjoint sensitivity.

11.9 Lotka-Volterra model fit plots

In figure 13 we show the model fit to the simulated data. This model fit is based on the results obtained after running **VI** and with **adjoint** sensitivity based VJP.

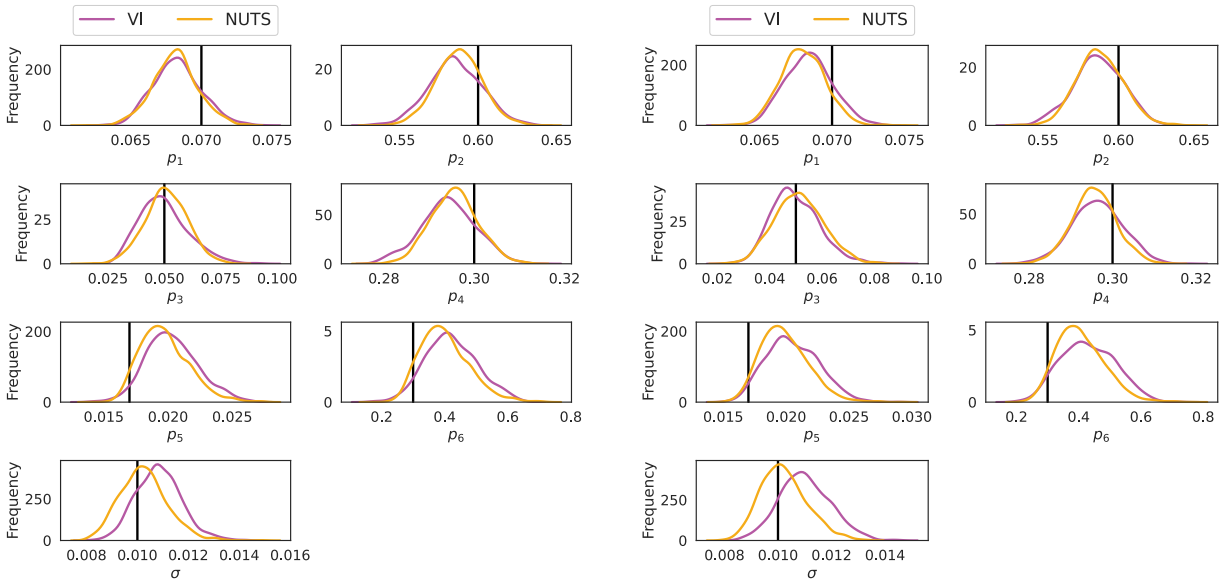


Figure 6: **Protein transduction model** marginal density plots: (a) using **Forward** and (b) using **Adjoint** sensitivity based VJP. The black line denotes the true generative parameter values.

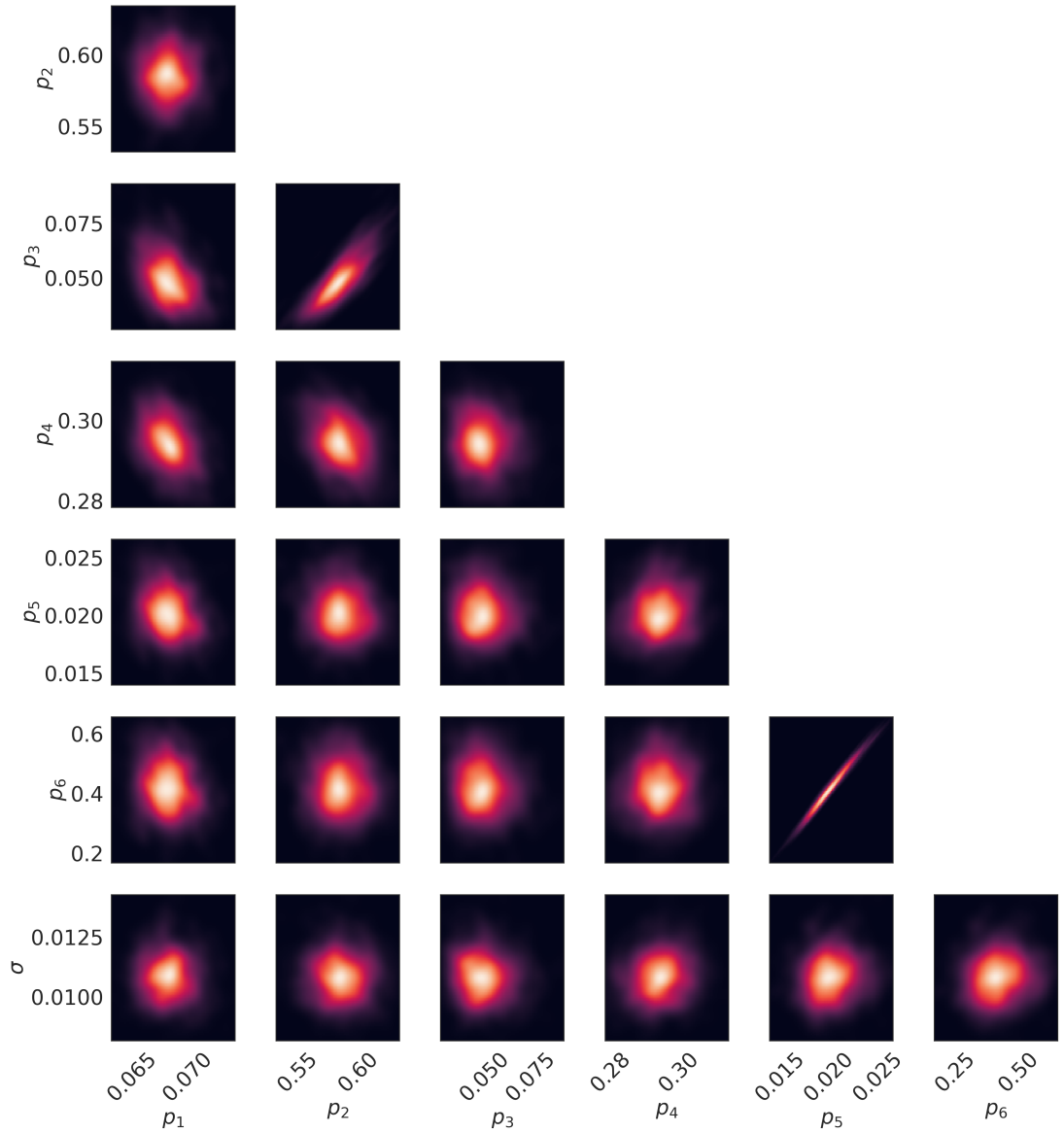


Figure 7: **Protein transduction** pairwise joint density plots obtained from running **VI** with **forward sensitivity** based VJP.

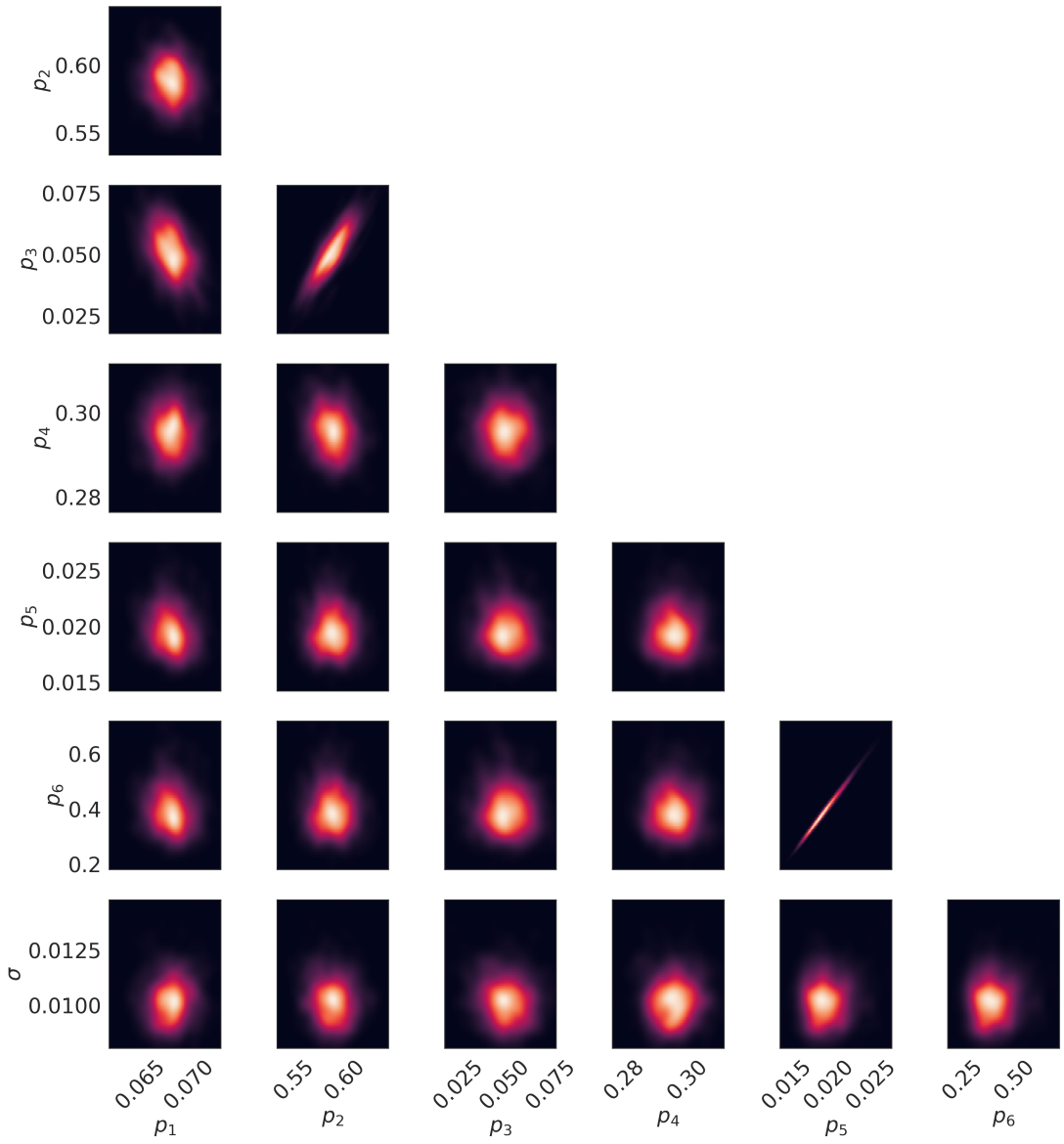


Figure 8: **Protein transduction** pairwise joint density plots obtained from running **NUTS** with **forward sensitivity** based VJP.

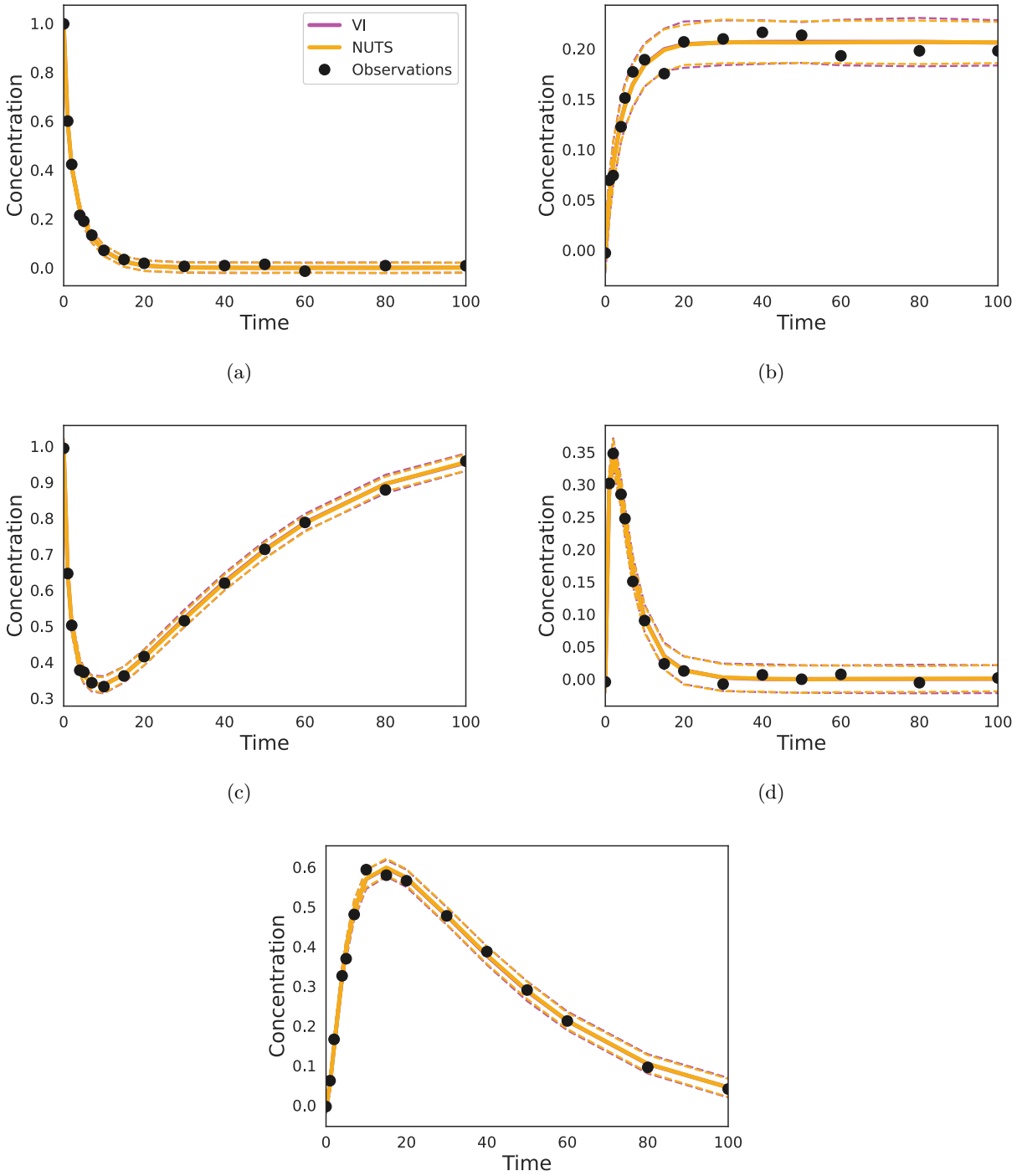
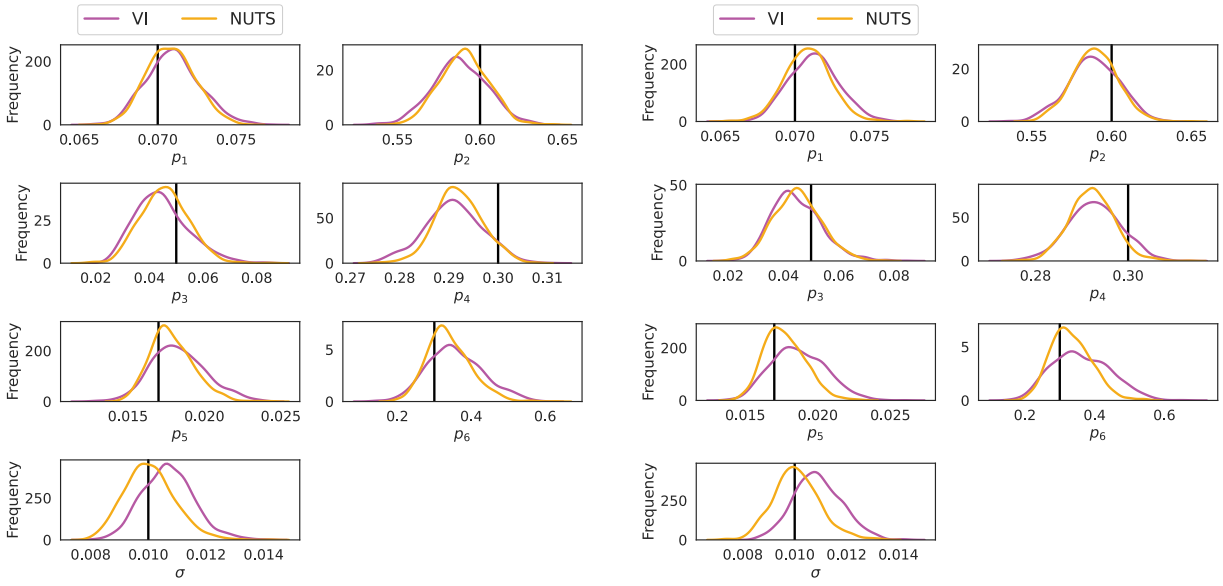
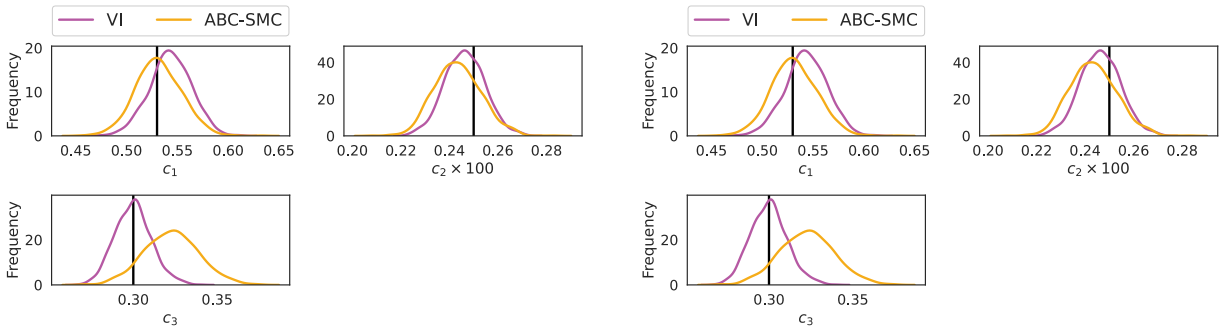


Figure 9: Model fit plots for **protein transduction model**. Mean (solid lines) and the 95% credible intervals (dashed lines) of the posterior predictive distributions are plotted. These plots are based on running **VI** and **NUTS** with **forward** sensitivity based VJP.



(a) (b)

Figure 10: **Protein transduction model** marginal density plots for **one of the additional artificial dataset**: (a) using **Forward** and (b) using **Adjoint** sensitivity based VJP. The black line denotes the true parameter values.



(a) (b)

Figure 11: **Stochastic Lotka-Volterra model** marginal density plots: (a) using **Forward** and (b) using **Adjoint** sensitivity based VJP. The black line denotes the true parameter values. The black line denotes the true parameter values.

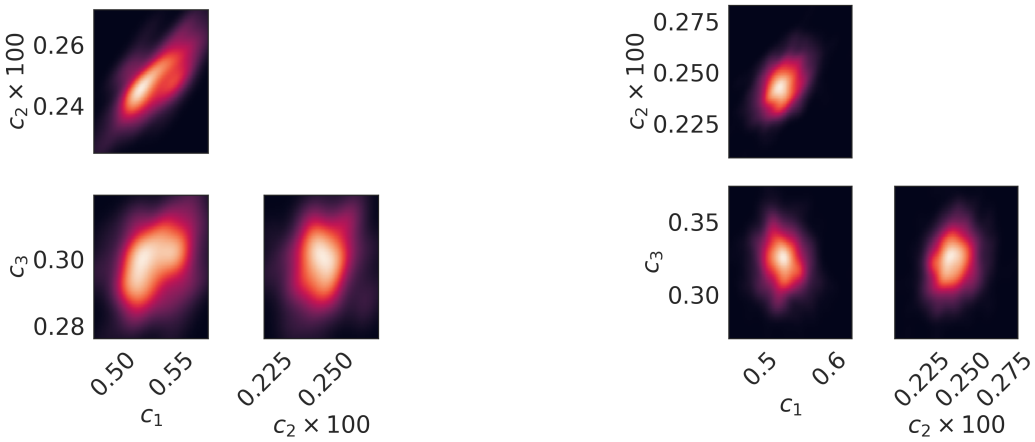


Figure 12: **Lotka-Volterra model** pairwise joint density plots: (a) for **VI** (with adjoint sensitivity) and (b) for **ABC-SMC**.

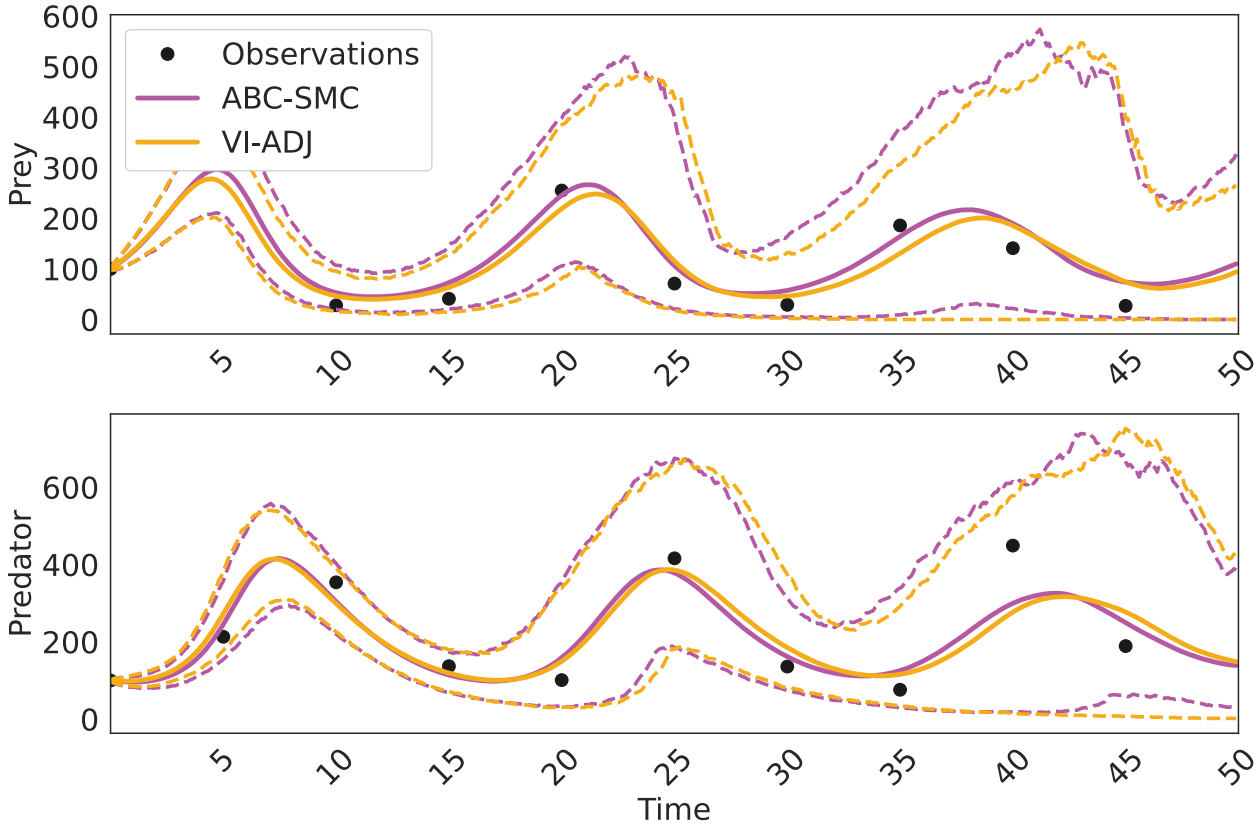


Figure 13: **Lotka-Volterra** model fit plots for inference using **VI-ADJ** and **ABC-SMC**. Mean (solid lines) and the 95% credible intervals (dashed lines) of the posterior predictive distributions are plotted.