Weekly Progress Report

Machine Learning for Delay Differential Equation Model Identification

Report Period: April 28 – May 4, 2025

Author: Kamil Hlubek

May 5, 2025

Week 4 Progress Summary

This week corresponds to Week 4 of the project plan, focusing on the task "Understanding the Predator-Prey Model". In line with this, the predator-prey model with a time delay (τ) described by Frank et al. [1] was studied. A Julia implementation of this model was also developed. Beyond this specific task, significant effort was dedicated to the implementation, testing, and theoretical understanding of the Manifold-constrained Gaussian Process Inference (MAGI) and DDE-Find algorithms for parameter inference in dynamical systems, particularly DDEs.

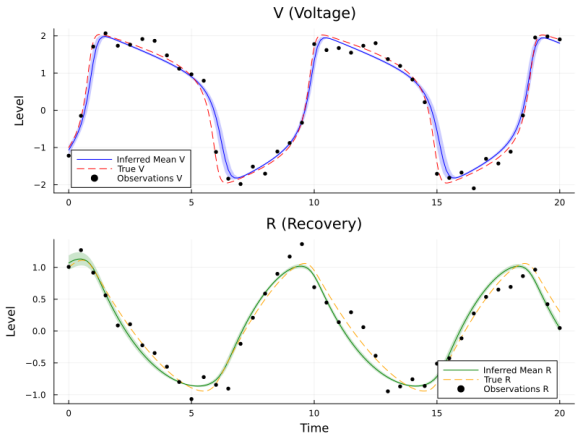
MAGI Algorithm Development & Analysis:

- The implementation of MAGI [2, 3] in Julia [4] is now largely complete regarding the core algorithm. Current focus is shifting towards developing convenience methods and improving the user interface. An existing MAGI implementation powered by TensorFlow Probability [5] serves as a useful reference. The Gaussian Process priors currently use the Matérn 5/2 kernel.
- Documented the algorithm for the Julia implementation of MAGI (see internal documentation [6]).
- Successfully tested the Julia MAGI implementation end-to-end, confirming it produces parameter estimations for ODE systems. This implementation can handle cases with both known and unknown observation variance (σ^2), although the complexity increase associated with unknown σ^2 requires more thorough investigation.
- Conducted simulations using the FitzHugh-Nagumo model with the MAGI implementation. Initial results (see Figure 1) show that parameter estimations track true values reasonably well. Future work requires conducting systematic parameter studies for MAGI.

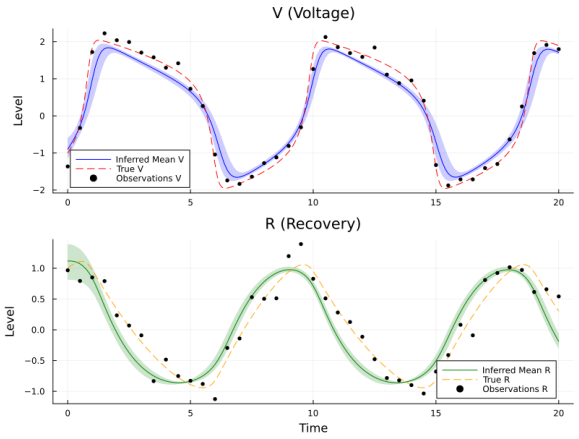
```
function fn_ode!(du, u, p, t)
    V = u[1]
    R = u[2]
    a, b, c = p # Or p[1], p[2], p[3] if p is a Vector/Tuple

    du[1] = c * (V - (V^3) / 3.0 + R)
    du[2] = -1.0/c * (V - a + b * R)
    return nothing
end

```

Listing 1: FitzHugh-Nagumo ODE definition for MAGI experiment

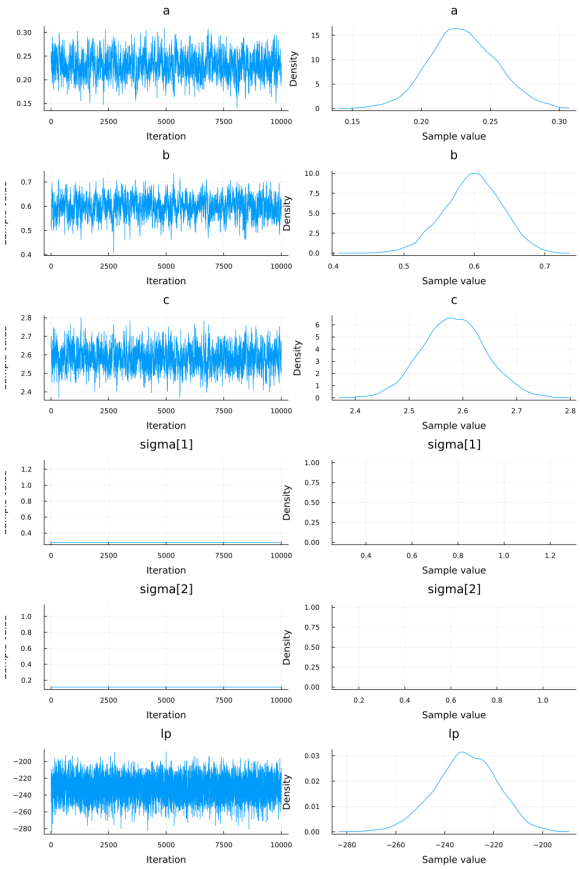
- Studied the MAGI DDE paper by Zhao and Wong [7]. Key findings include the imposition of a prior on the delay τ and a linear approximation of the delayed state's contribution to the RHS using its two nearest measured observations. This simplifies the adaptation of MAGI to MAGIDDE, primarily affecting the calculation of the Jacobian of the log posterior distribution gradient, crucial for the HMC sampler.



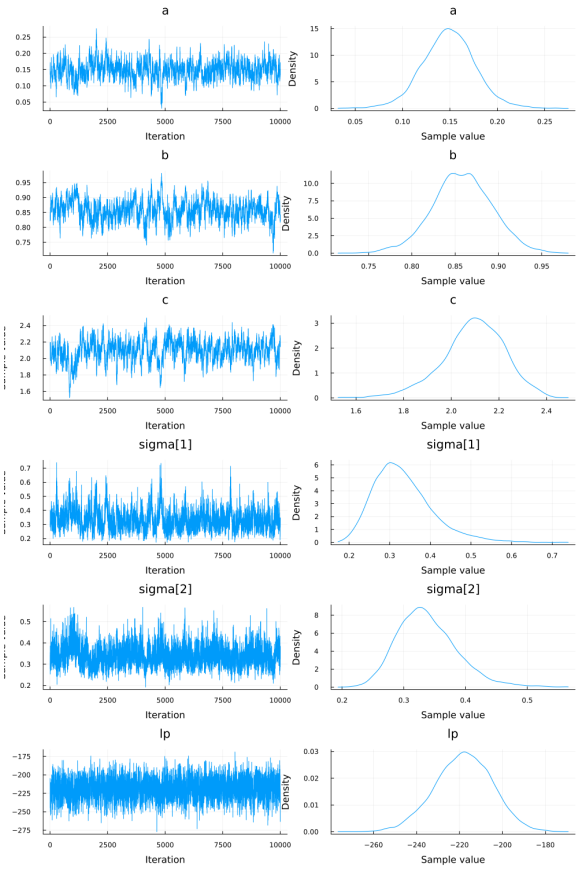
(a) Trajectory (Known σ^2)



(b) Trajectory (Unknown σ^2)



(c) Parameter Traces (Known σ^2)



(d) Parameter Traces (Unknown σ^2)

Figure 1: MAGI results for the FitzHugh-Nagumo model simulation. Top row: Trajectory reconstructions for known (a) and unknown (b) σ^2. Bottom row: Corresponding parameter trace plots for known (c) and unknown (d) σ^2.

- Created documentation detailing the planned adaptation of the MAGI algorithm for DDEs (MAGIDDE) in Julia, including explicit analytical calculations of the required Jacobians (see internal documentation [8]).
- Briefly explored using Automatic Differentiation (AD) for the MAGI Jacobian calculations but decided against it for now. The explicit analytical form of the Jacobian is known, and using AD might introduce computational overhead. The current focus is on assessing the approximative power of the MAGI approach; the explicit implementation is deemed easier to work with initially. AD will be reconsidered later for potentially improved generalizability.

DDE-Find Algorithm Exploration & Testing:

- The entire premise of migrating from reference Python DDE-Find implementations (which often use manually implemented adjoints) to the Julia SciML ecosystem [9] is to replace manual adjoint calculations with the automatic, robust, and performant adjoint sensitivity analysis methods built into `DifferentialEquations.jl` and `SciMLSensitivity.jl`. Analysis of documentation related to the reference PyTorch implementation [10] was also conducted.
- Initial tests were performed using this DDE-Find framework [11] within Julia to estimate parameters for benchmark DDE problems. These tests leveraged Julia's optimization libraries (`Optimization.jl`) and sensitivity analysis tools.
- Specifically, the multi-dimensional logistic delay equation benchmark from [11] was tested, defined component-wise for $i = 1, \ldots, d$ as:

$$\frac{du_i}{dt} = \theta_{0,i} u_i(t)(1 - \theta_{1,i} u_i(t - \tau)) \tag{1}$$

with periodic history function:

$$h_i(p_{\text{hist}}, t) = A_i \sin(w_i t) + b_i \tag{2}$$

Results for a 1D case are shown in Table 1. They demonstrate promise for parameter recovery, although sensitivity to the initial parameter guesses and varying accuracy across parameters (especially history parameters A, w) were observed, indicating the need for robust initialization strategies or global optimization approaches.

- A key advantage identified is the ease of adapting numerical methods (solvers, AD/sensitivity algorithms) thanks to Julia's SciML ecosystem.

Table 1: Parameter estimation results for the 1D logistic DDE benchmark (1-2) using the DDE-Find framework in Julia [9]. Initial guess from [11] was used, with noise level 0.01 over $T = [0, 10]$. Final Loss: 0.006986.

Parameter	Estimated	True Value	Error (%)
θ_0	1.985589	2.000000	0.720573
θ_1	1.510411	1.500000	0.694098
τ	1.005068	1.000000	0.506771
A	-0.329578	-0.500000	34.084378
w	2.282584	3.000000	23.913852
b	2.068189	2.000000	3.409426

General Observations & Related Reading:

- Reflected on the common practice in related literature of benchmarking primarily on standard, distinct evaluations. An overall measure of applicability or robustness seems often missing, sometimes giving the impression of cherry-picked results. However, algorithmic complexity can generally be estimated from the papers.
- Read the MAGI-X paper [12], which uses a Neural Network for the ODE's RHS within the MAGI framework. The paper concludes this approach is more suited for forecasting than parameter estimation. This concept might be relevant for DDEs in the future but is not an immediate priority. Notably, the DDE-Find authors also mentioned using NNs for the RHS as future work, suggesting a current boundary in the state-of-the-art.

Next Steps & Future Directions: The following steps outline the planned work, with an emphasis on leveraging the Julia SciML ecosystem for robust and scalable DDE inference:

- **Implement MAGIDDE:** Complete the Julia implementation of MAGI adapted for DDEs based on the derived analytical Jacobians [8].
- **Refine Implementations:** Enhance both the MAGI [4] and DDE-Find [9] Julia implementations with better user interfaces, documentation, and testing protocols. Further investigate the performance and complexity trade-offs of MAGI with unknown σ^2.
- **Adjoint Method Deep Dive:** Gain a deeper understanding of the various adjoint sensitivity analysis methods available in `SciMLSensitivity.jl` and their applicability to DDE-Find.
- **Benchmarking & Scalability:** Evaluate the performance (accuracy, speed, robustness) of MAGIDDE and the Julia DDE-Find implementation on a wider range of DDE models, including potentially larger, multi-dimensional systems.
- **Strengths & Weaknesses Analysis:** Systematically compare the two approaches to identify their respective advantages and limitations under different data conditions (sparsity, noise) and model complexities.
- **Leverage Julia/SciML Ecosystem:** Actively explore existing tools within the SciML ecosystem (e.g., modeling tools, solvers, parameter estimation libraries) for inspiration and potential integration to enhance the current frameworks.
- **Hybrid Approach Exploration:** Investigate the potential of a meta-algorithm combining the strengths of both methods, for example, using MAGIDDE to obtain a good initial parameter guess for subsequent refinement by the DDE-Find optimization framework.

Key Takeaways for Week 4:

- **Predator-Prey Model:** Studied the Frank et al. [1] DDE model (Eqs. **??-??**) and developed a Julia implementation, fulfilling the primary task for Week 4.
- **MAGI Progress:** Julia MAGI implementation [4] using Matérn 5/2 kernel is largely complete; tested on FitzHugh-Nagumo (Figure 1), confirming capability for known/unknown σ^2. Focus shifts to usability and further investigation [6].
- **MAGIDDE Theoretical Foundation:** Analyzed the MAGI DDE paper [7], documented the analytical adaptation for Julia [8], prioritizing explicit Jacobians over AD initially.

- **DDE-Find Motivation & Testing:** Articulated the rationale for using Julia/SciML (automatic adjoints) over manual methods [9]. Conducted initial tests on benchmark problems like the Logistic DDE (Eqs. 1-2) [11], showing promise but also sensitivity to initial guesses (Table 1). Analyzed reference implementation documentation [10].
- **Algorithm Comparison & Context:** Gained insights into MAGI-X [12] (NN-based RHS for forecasting) and reflected on benchmarking practices.
- **Documentation:** Continued emphasis on documentation for MAGI [6], MAGIDDE [8], and DDE-Find analysis [10].
- **Future Strategy:** Outlined clear next steps focusing on implementation completion, leveraging Julia/SciML for scalability and advanced features, benchmarking, and exploring hybrid methods.

References

[1] A. Frank et al. *Population dynamic regulators in an empirical predator-prey system*. Accessed via analysis document. 2021. DOI: 10.1016/j.jtbi.2021.110814.

[2] Shihao Yang, Samuel WK Wong, and S. C. Kou. *Inference of dynamic systems from noisy and sparse data via manifold-constrained Gaussian processes*. Primary MAGI algorithm reference. 2021. DOI: 10.1073/pnas.2020397118.

[3] MAGI Development Team. *magi: A Package for Inference of Dynamic Systems from Noisy and Sparse Data via Manifold-Constrained Gaussian Processes*. Software package or documentation reference. Associated with [2]. Accessed 2025.

[4] Kamil Hlubek. *MAGI.jl: Manifold-Constrained Gaussian Process Inference in Julia*. GitHub Repository. Accessed 2025-05-05. 2025. URL: https://github.com/k1m9l/MAGI.

[5] Shihao Yang (skbwu). *Python-MAGI: MAGI Implementation in Python/TensorFlow*. GitHub Repository. Accessed 2025-05-05. 2020. URL: https://github.com/skbwu/Python-MAGI.

[6] Kamil Hlubek. *Documentation of MAGI Julia Implementation*. Internal Documentation. 2025. URL: https://drive.google.com/file/d/1WY_i9ln3Tgn4OVucFd1qgOiHaKXNZgOj/view?usp=share_link (visited on 05/05/2025).

[7] Yuxuan Zhao and Samuel W.K. Wong. *Inference for Delay Differential Equations Using Manifold-Constrained Gaussian Processes*. Unpublished manuscript/preprint. Dated June 24, 2024. Department of Statistics and Actuarial Science, University of Waterloo. June 2024.

[8] Kamil Hlubek. *Documentation for the MAGI DDE Julia Algorithm Adaptation*. Internal Documentation. 2025. URL: https://drive.google.com/file/d/10t9UX-7QVvUjdBqUJiGo9zA1GVq9PaK/view?usp=share_link (visited on 05/05/2025).

[9] Kamil Hlubek. *DDE-FindJl: Delay Differential Equation Finding in Julia*. GitHub Repository. Accessed 2025-05-05. 2025. URL: https://github.com/k1m9l/MagiDDEJl/tree/main/DDE-FindJl.

[10] Kamil Hlubek. *Analysis of DDE-Find PyTorch Implementation*. Internal Documentation/-Analysis. 2025. URL: https://drive.google.com/file/d/1JstWa5Gse8MEGcWz0WiYq9HzcdkEDCNi/view?usp=sharing (visited on 05/05/2025).

[11] Robert Stephany. *DDE-Find: Learning Delay Differential Equations from Noisy, Limited Data*. arXiv preprint arXiv:2405.02661. Accessed via code documentation. 2024.

[12] Chaofan Huang, Simin Ma, and Shihao Yang. *MAGI-X: Manifold-Constrained Gaussian Process Inference for Unknown System Dynamics*. arXiv preprint arXiv:2305.18801. Accessed 2025. 2023.