Dynamic Electoral Model Using Biological Principles

Alan Royce Gabriel[1] **and Shivan Ajay Iyer**[2]

[1] *BS22B001*
[2] *BE22B048*
[*] *https://github.com/shivan-21/Electoral-Modelling*

Compiled June 3, 2025

Elections are intricate and dynamic systems shaped by the complex interactions among voters, candidates, and the media. Voting behaviour and electoral outcomes analyses have traditionally relied on agent-based simulations, opinion dynamics, and econometric approaches. In this study, we take an unconventional yet enlightening approach by applying biological modelling frameworks—specifically, the SIR (Susceptible-Infected-Recovered) model and Lotka-Volterra predator-prey dynamics—to examine electoral processes. We conceptualise voter behaviour as akin to the spread of disease, competition, or ecological balance, mapping entities such as undecided voters, political parties, and media influence onto biological counterparts. Through simulations and comparative analyses, we illustrate how these biological models can effectively capture emergent behaviours, including polarisation, opinion cascades, and candidate dominance

1. INTRODUCTION

Elections are a cornerstone of modern democratic systems, characterised by the dynamic interplay of individuals, institutions, and information. Modelling electoral dynamics is a key challenge in computational social science, aiming to understand voter behaviour, forecast outcomes, and analyse the impact of strategies such as campaigning, debates, and media influence. Traditional approaches include agent-based simulations, opinion dynamics models (such as the DeGroot and voter models), and statistical or econometric models that rely on historical data and behavioural assumptions. While these approaches have yielded valuable insights, they often involve high-dimensional parameter spaces and limited analytical tractability.

In contrast, biological systems—particularly those studied in computational systems biology—offer elegant frameworks for modelling complex, nonlinear interactions among populations. Models such as the SIR (Susceptible-Infected-Recovered) epidemiological model and the Lotka-Volterra predator-prey equations have proven influential in understanding disease spread, resource competition, and ecological stability. These models are characterised by low-dimensional, interpretable dynamics that yield rich emergent behaviours from simple rules.

This paper explores the application of biological modelling paradigms to election dynamics. We draw parallels between the spread of opinions and disease transmission, framing political campaigning as a contagious process. Similarly, we reinterpret predator-prey models to represent strategic interactions among competing candidates and voter groups. By treating undecided voters as "susceptible," party influence as "infection," and decided voters as "recovered," we simulate how electoral support spreads and stabilises. We also analyse how competition between candidates, analogous to species interactions, affects vote share dynamics.

Through simulations and comparative analysis, we assess the effectiveness of these biologically-inspired models against standard approaches in political modelling. We aim to highlight the utility of systems biology frameworks for understanding elections, offering a novel perspective that blends insights from biology, mathematics, and social science.

2. METHODOLOGY

A. Agent-Based Modeling [1]

To simulate individual behaviour and systemic electoral outcomes, an **Agent-Based Modelling (ABM)** is employed. ABM allows us to represent each voter and candidate as autonomous agents with distinct attributes and behavioural rules. This bottom-up approach captures emergent phenomena that arise from local interactions within a population, making it particularly suitable for studying nonlinear and dynamic electoral processes. In our model, voters are characterised by ideological leaning, policy preferences, susceptibility to peer influence, social connectivity, and likelihood of turnout. Candidates are represented by static traits (e.g., geographical reach, ideological stance, youth appeal) and dynamic strategies, including campaign outreach and media engagement.

The agents interact within a simulated environment, which may be spatially continuous or structured as a network (e.g., small-world or scale-free), to reflect real-world social or geographic connectivity. Voter opinions evolve through localised interactions governed by established models of opinion dynamics, such as the bounded confidence model or voter model. External campaign effects and media events introduce broader shifts in voter sentiment, simulating real-world political advertising, debates, or breaking news.

The simulation unfolds over discrete time steps, during which voters iteratively update their preferences, respond to candidate strategies, and influence their neighbours. We incorporate behavioural complexities such as opinion inertia, issue-based salience, and stubbornness among specific voter subgroups. The final decision rule determines that, on election day, each voter chooses a candidate—or abstains—based on their most updated preference and personal turnout probability.

This model was implemented using Python's Mesa ABM framework, enabling flexible design, agent visualisation, and scalable experimentation. We conducted simulations across network topologies and campaign strategies to study emergent electoral patterns. Key outputs analysed include vote share trajectories, voter turnout, polarisation levels, opinion clustering, and the spatial influence of candidates. These insights illuminate how individual-level behaviours, social structure, and strategic actions collectively shape election outcomes.

Mathematical Formulation

To capture the dynamics of opinion formation, voter turnout, and candidate influence within the agent-based framework, we employed the following models:

1. Opinion Update Rule (Deffuant Model) Agents update their opinions only when they are within a predefined tolerance ϵ of each other:

$$\text{If } |o_i - o_j| < \epsilon, \text{ then}$$

$$o_i^{t+1} = o_i^t + \mu(o_j^t - o_i^t), \quad o_j^{t+1} = o_j^t + \mu(o_i^t - o_j^t)$$

Here, o_i and o_j are the opinions of agents i and j, $\mu \in (0, 0.5]$ is the convergence parameter.

2. Voter Turnout Probability The likelihood of an agent voting is modelled using a logistic function:

$$P_{\text{turnout},i} = \frac{1}{1 + e^{-(\alpha E_i + \beta S_i - \gamma C_i)}}$$

where E_i is the voter's enthusiasm, S_i is social influence (peer pressure), C_i is the perceived effort cost, and α, β, γ are sensitivity weights.

3. Candidate Influence (Spatial Decay Model) The influence of candidate c on voter i, depending on spatial or ideological distance d_{ic}, is given by:

$$I_{ic} = \frac{A_c}{1 + d_{ic}^k}$$

Where A_c is the candidate's campaign strength, and k controls the decay rate of influence over distance.

4. Voting Decision Rule (Softmax Utility) Voters choose candidates probabilistically based on perceived utility:

$$P_{ij} = \frac{e^{U_{ij}}}{\sum_k e^{U_{ik}}}$$

Here, U_{ij} is the utility voter i assigns to candidate j, incorporating ideological proximity, campaign exposure, and trust.

5. Polarization Metric To quantify opinion divergence over time, we calculate the variance in the opinion distribution:

$$\text{Polarization} = \frac{1}{N} \sum_{i=1}^{N} (o_i - \bar{o})^2$$

Where $\bar{o}$ is the mean opinion across all agents.

B. ARIMA [2]

The ARIMA model is denoted as ARIMA(p, d, q), where p is the order of the autoregressive (AR) component, d is the degree of differencing needed to make the time series stationary, and q is the order of the moving average (MA) component. The general form of an ARIMA model is given by:

$$\phi(B)(1 - B)^d y_t = \theta(B)\varepsilon_t$$

Here, y_t represents the vote share at time t, B is the backshift operator such that $By_t = y_{t-1}$, $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$ is the autoregressive polynomial, $\theta(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$ is the moving average polynomial, and ε_t is white noise. In the context of elections, y_t could correspond to the percentage of support for a particular candidate or party over time, measured via polling data. By differencing the data d times to remove trends and applying the AR and MA components, ARIMA can capture both short-term fluctuations and long-term trends in voter behaviour. This makes it especially useful in identifying momentum shifts, sudden swings in public opinion, and the impact of key events or policies in the lead-up to elections.

C. VAR [3]

Vector AutoRegression (VAR) is a multivariate time series approach that captures the linear interdependencies among multiple political variables over time—such as vote shares of different parties, approval ratings, or economic indicators. Unlike ARIMA, which models a single variable, VAR models a system of equations where each variable is a linear function of past values of itself and of all other variables in the system. This makes VAR particularly useful in capturing mutual influence among political entities or how external factors like inflation or unemployment affect vote shares.

A standard VAR model of order p, denoted as VAR(p), is written as:

$$\mathbf{Y}_t = \mathbf{A}_1 \mathbf{Y}_{t-1} + \mathbf{A}_2 \mathbf{Y}_{t-2} + \cdots + \mathbf{A}_p \mathbf{Y}_{t-p} + \mathbf{u}_t$$

Here, $\mathbf{Y}_t$ is a $k \times 1$ vector of endogenous variables (e.g., vote shares of k different parties), $\mathbf{A}_i$ are $k \times k$ coefficient matrices representing the effect of lag i, and $\mathbf{u}_t$ is a vector of white noise error terms. Each variable in $\mathbf{Y}_t$ is regressed on its own lags and the lags of every other variable in the system, making it possible to explore how past values of one candidate's or party's support might influence others.

D. Neural ODE [4]

Neural Ordinary Differential Equations (Neural ODEs) is a continuous-time deep learning framework ideal for capturing the nonlinear and dynamic nature of election vote share evolution. Unlike discrete models such as ARIMA or VAR, Neural ODEs define the derivative of the hidden state using a neural network, allowing smooth interpolation between polling data points and extrapolation of voter behavior dynamics. The evolution of a candidate's support over time is modeled as a solution to a parameterized ODE:

$$\frac{d\mathbf{h}(t)}{dt} = f_\theta(\mathbf{h}(t), t)$$

Here, $\mathbf{h}(t)$ is the latent state (e.g., encoded voter preferences or vote share distribution) at time t, and f_θ is a neural network parameterized by θ. The solution to this ODE over a time interval $[t_0, t_1]$ is obtained via an ODE solver:

$$\mathbf{h}(t_1) = \mathbf{h}(t_0) + \int_{t_0}^{t_1} f_\theta(\mathbf{h}(t), t)\, dt$$

In election modeling, Neural ODEs allow learning from irregularly spaced polling data while capturing complex temporal dependencies and external shocks (e.g., scandals or media events).

E. S-curve logistic model [5]

The S-curve logistic model describes the cumulative adoption or support growth of a candidate or political idea over time, especially when the dynamics follow a slow-starting, rapidly accelerating, and then saturating pattern. This model captures the natural limits in voter base expansion due to saturation, resistance, or finite population size. The logistic growth function is given by:

$$V(t) = \frac{K}{1 + e^{-r(t-t_0)}}$$

Here, $V(t)$ represents the vote share or cumulative support at time t, K is the carrying capacity or maximum attainable support (usually normalized to 1 or 100%), r is the growth rate, and t_0 is the inflection point at which the support growth is the fastest. The S-curve effectively models how initial campaigns may have limited impact, but with growing awareness or momentum, the support increases rapidly before tapering off due to market or ideological saturation.

In electoral forecasting, the logistic model is especially useful for modeling the rise of insurgent candidates, viral political movements, or the diffusion of campaign messages across a population. It provides a parsimonious and interpretable structure to describe bounded growth and can be fitted to time-series polling data to predict eventual vote share ceilings.

F. PINNS [6]

Physics-Informed Neural Networks (PINNs) provide an innovative approach to election modeling by incorporating physical or domain-specific knowledge directly into the learning process. Unlike traditional deep learning models, PINNs enforce constraints derived from governing laws (e.g., voter behavior dynamics, economic models) during training. In the context of election modeling, PINNs can be employed to predict vote shares or electoral outcomes while respecting underlying principles such as continuity of voter behavior or conservation laws (e.g., total voter turnout).

The basic formulation of a PINN involves minimizing the residuals of a differential equation along with the loss from the data (e.g., polling data). For an election modeling scenario where $y(t)$ represents the vote share or support at time t, the governing equation might take the form of a differential equation (e.g., the rate of change in support could be a function of past trends, media influence, and other dynamic factors). The governing equation can be written as:

$$\frac{dy(t)}{dt} = f(t, y(t)) + \epsilon(t)$$

Here, $y(t)$ is the vote share at time t, $f(t, y(t))$ represents the dynamic function governing the evolution of support (which could include external forces, media events, or campaign strategies), and $\epsilon(t)$ is the error term or residual capturing the discrepancy between the data and the model.

In a PINN approach, the loss function L_{PINN} consists of two components:

1. **Data Loss**: This term ensures that the model fits the observed data points (e.g., polling data) by minimizing the difference between predicted and actual values:

$$L_{\text{data}} = \sum_i (y(t_i) - \hat{y}(t_i))^2$$

2. **Physics Loss**: This term ensures that the predictions satisfy the governing differential equation, effectively encoding the prior knowledge of how vote share should evolve over time:

$$L_{\text{physics}} = \sum_j \left(\frac{dy(t_j)}{dt} - f(t_j, y(t_j)) \right)^2$$

Thus, the total loss function for a PINN is the combination of both:

$$L_{\text{PINN}} = L_{\text{data}} + \lambda L_{\text{physics}}$$

Here, λ is a weight parameter that balances the importance of fitting the data versus respecting the underlying physics or dynamics. The PINN framework can be trained using gradient-based optimization techniques to learn the model parameters while ensuring that both the data and the physics are satisfied.

G. Least Square fitting

SIR Model [7]

When applied to the SIR (Susceptible-Infected-Recovered) model, the goal is to estimate the parameters of the model (e.g., infection rate, recovery rate) by minimising the discrepancy between the predicted and observed values of the susceptible, infected, and recovered populations over time.

The SIR model is typically described by the following system of ordinary differential equations (ODES):

$$\frac{dS(t)}{dt} = -\beta S(t) I(t)$$

$$\frac{dI(t)}{dt} = \beta S(t) I(t) - \gamma I(t)$$

$$\frac{dR(t)}{dt} = \gamma I(t)$$

Where: $S(t)$ is the susceptible population at time t, $I(t)$ is the infected population at time t, $R(t)$ is the recovered population at time t, β is the transmission rate (probability of infection per contact), γ is the recovery rate (rate at which individuals recover).

Given time-series data for $S(t)$, $I(t)$, and $R(t)$, the task is to fit the model by adjusting β and γ so that the predicted values of $S(t)$, $I(t)$, and $R(t)$ match the observed values as closely as possible.

Predator Prey model [7]

The Lotka-Volterra equations describe the continuous-time interactions between the prey population $x(t)$ and predator population $y(t)$ as follows:

$$\frac{dx(t)}{dt} = \alpha x(t) - \beta x(t) y(t)$$

$$\frac{dy(t)}{dt} = \delta x(t) y(t) - \gamma y(t)$$

Where: $x(t)$ is the prey population at time t, $y(t)$ is the predator population at time t, α is the growth rate of the prey population, β is the rate at which predators capture and consume prey, δ is the rate at which predators grow as a result of consuming prey, γ is the death rate of the predator population.

Given observed data for the prey and predator populations over time, the task is to estimate the parameters α, β, δ, and γ by minimizing the least squares error between the observed and predicted population values.

Formulation

The objective of the Least Squares Method is to minimize the sum of squared errors between the observed data and the predicted values from the model. Let $S_{\text{obs}}(t)$, $I_{\text{obs}}(t)$, and $R_{\text{obs}}(t)$ denote the observed data at time t, and $S_{\text{model}}(t)$, $I_{\text{model}}(t)$, and $R_{\text{model}}(t)$ represent the model's predicted values at the same time. The least squares objective function L is defined as:

$$L(\beta,\gamma) = \sum_t \Big[(S_{\text{obs}}(t) - S_{\text{model}}(t))^2 + (I_{\text{obs}}(t) - I_{\text{model}}(t))^2$$

$$+ (R_{\text{obs}}(t) - R_{\text{model}}(t))^2 \Big]$$

The model's predicted values, $S_{\text{model}}(t)$, $I_{\text{model}}(t)$, and $R_{\text{model}}(t)$, are obtained by solving the system of ODEs using numerical methods (e.g., Euler's method, Runge-Kutta methods) for given initial conditions $S(0)$, $I(0)$, and $R(0)$. The difference between the observed and predicted values is computed once the model is solved for each parameter set β and γ.

Optimization Process

The parameters β and γ are estimated by minimizing the least squares objective function. This is typically done using optimization algorithms such as gradient descent or nonlinear least squares optimization. The goal is to find the values of β and γ that minimize $L(\beta,\gamma)$, thereby providing the best fit of the SIR model to the observed data.

$$(\hat{\beta},\hat{\gamma}) = \arg\min_{\beta,\gamma} L(\beta,\gamma)$$

where $\hat{\beta}$ and $\hat{\gamma}$ are the estimated parameters.

H. Evolutionary Algorithm [8]

Evolutionary Algorithms (EAs) are a class of optimization algorithms inspired by the process of natural selection. These algorithms are particularly useful for solving complex optimization problems where traditional methods may struggle, especially when the problem space is large, nonlinear, or poorly understood. EAs simulate the process of evolution through the selection, crossover, and mutation of candidate solutions to iteratively improve the solution over time.

The general workflow of an Evolutionary Algorithm can be broken down into several steps:

1. **Initialization**: A population of candidate solutions (individuals) is created. Each individual represents a potential solution to the problem, typically encoded as a string (often binary, real-valued, or other representations). The population size is usually denoted by N, where each individual x_i has a fitness value $f(x_i)$ that quantifies how well it solves the problem.

2. **Fitness Evaluation**: Each individual in the population is evaluated using a fitness function $f(x)$ that measures the quality of the solution represented by x. This fitness function could represent the objective function of the problem being optimized.

$$f(x): \mathbb{R}^n \to \mathbb{R}, \quad f(x) = \text{Objective function}$$

$$\text{fitness function}(\beta,\gamma) = \sum_t (I_{\text{model}}(t) - I_{\text{obs}}(t))^2$$

3. **Selection**: Selection is the process by which individuals are chosen for reproduction based on their fitness. The fitter individuals are more likely to be selected. Common selection methods include Roulette Wheel, Tournament, and Rank Selection.

4. **Crossover (Recombination)**: Crossover (or recombination) involves combining parts of two or more parent solutions to create offspring. This mimics biological recombination. A typical crossover operator exchanges portions of two parent individuals to create two new offspring. For example, a one-point crossover on binary strings can be written as:

$$\text{Offspring}_1 = \text{Parent}_1^{1\ldots k} \oplus \text{Parent}_2^{k+1\ldots n}$$

$$\text{Offspring}_2 = \text{Parent}_2^{1\ldots k} \oplus \text{Parent}_1^{k+1\ldots n}$$

where k is a randomly selected crossover point.

5. **Mutation**: Mutation introduces small random changes to an individual's genetic code, helping to maintain genetic diversity within the population and avoid premature convergence to local optima. For example, a mutation operator on a binary string might flip a bit at a randomly chosen position:

$$\text{Mutation}(x) = x_{\text{flip}} \quad \text{where} \quad x_{\text{flip}} = x_{\text{original}} \oplus \text{bit}$$

6. **Replacement**: After crossover and mutation, the population is updated. This can be done in several ways, such as replacing the entire population with the offspring (generational replacement) or keeping the best individuals (elitism) while replacing others.

7. Termination: The algorithm terminates when a stopping criterion is met. This could be a maximum number of generations, a target fitness value, or a time limit.

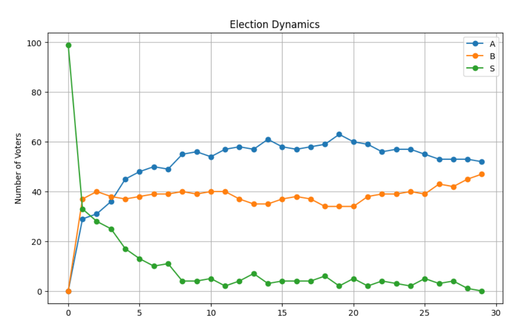
Objective Function and Fitness Function

In most optimisation problems, the objective function $f(x)$ defines what is being optimised. The fitness function $f(x)$ is typically the same as the objective function, though sometimes it can be transformed (e.g., by taking the inverse of the objective in the case of minimization problems).

$$f(x) = \text{Objective Function}, \quad x \in \mathbb{R}^n$$

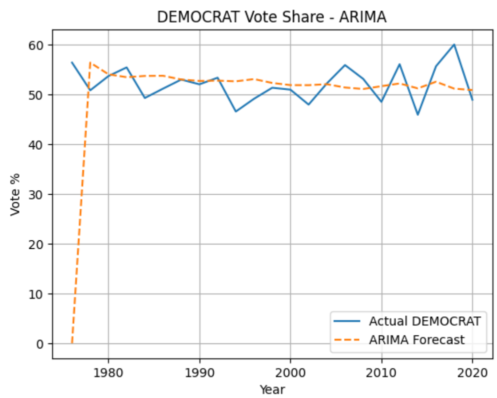
3. RESULTS

A. ABM



This figure shows the effect of a targeted smear campaign against candidate A. A and B represent the votes for candidates A and B, while S shows the number of undecided voters.

B. ARIMA

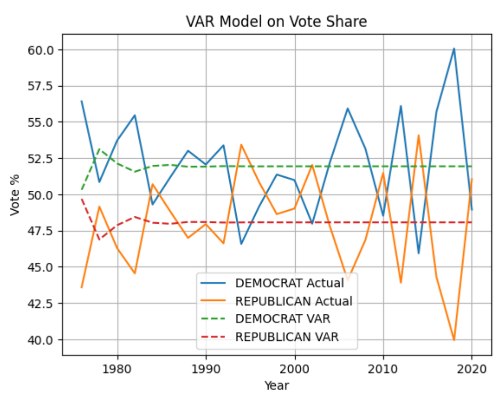


Fitting the ARIMA model gave the following parameters

$$y(t) = -0.1582 + (-0.1582) \cdot y(t-1) + (-0.9997) \cdot e(t-1)$$

we used this to model the history-dependent aggregate behaviour

C. VAR



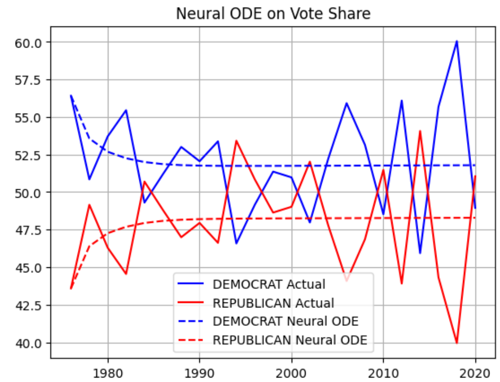
Fitting the VAR model, the following parameter were found The VAR paradigm has the advantage of modelling multiple time series simultaneously. It treats them as an interconnected system.

$$\begin{aligned}\text{DEMOCRAT\_pct}(t) = {} & 0.0053 + 0.1569 \cdot \text{DEMOCRAT\_pct}(t-1) \\ & + 0.3430 \cdot \text{REPUBLICAN\_pct}(t-1) \\ & + 0.1250 \cdot \text{DEMOCRAT\_pct}(t-2) \\ & + 0.3749 \cdot \text{REPUBLICAN\_pct}(t-2)\end{aligned}$$

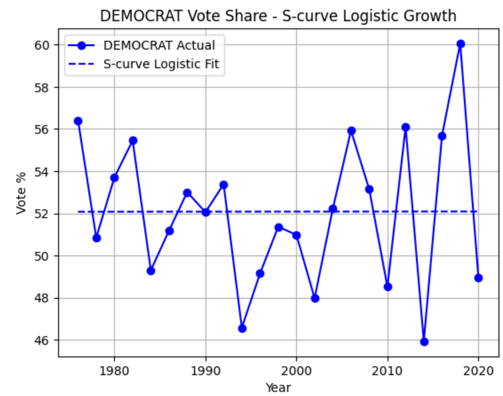
$$\begin{aligned}\text{REPUBLICAN\_pct}(t) = {} & 0.0047 + 0.3719 \cdot \text{DEMOCRAT\_pct}(t-1) \\ & + 0.1281 \cdot \text{REPUBLICAN\_pct}(t-1) \\ & + 0.4038 \cdot \text{DEMOCRAT\_pct}(t-2) \\ & + 0.0962 \cdot \text{REPUBLICAN\_pct}(t-2)\end{aligned}$$

D. Neural ODE

Neural ODEs model the system's (which is taken to be interconnected) evolution as a continuous process, defined by a differential equation. Thus, this model is suitable for non linear systems. This model learns a generating process for the ODE, while ARIMA/VAR fits statistical patterns. The Mean Squared Error comes as 19.3133.



E. S-curve logistic model



The S-curve paradigm assumes that the underlying process (change in national party support) follows a pattern of growth, diffusion, or transition phenomena. Despite its simplicity, it has the advantage of being highly interpretable, unlike the others, which can behave like black boxes.

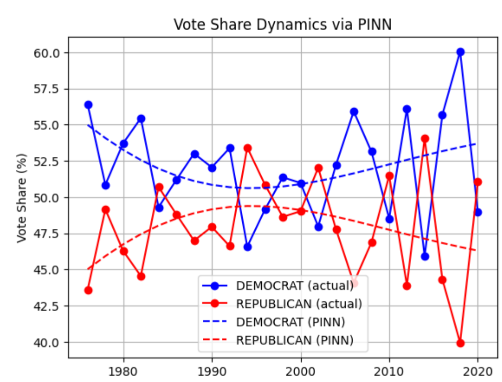
Fitted Parameters:

$$L = 104.27896614118241$$

$$k = 9.935007064820528 \times 10^{-6}$$

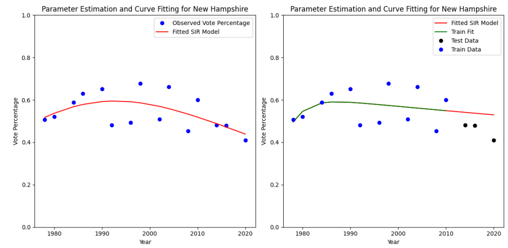
$$t_0 = 240.70514014056084$$

F. PINNS



PINNS learn y(t) and use the Pytorch function auto-diff in the loss to enforce dy/dt - f(y,t) = 0. The physics equation was chosen similarly to the Lotka-Volterra models. However, balancing the data physics loss terms (often requiring weighting), choosing network architectures, and tuning optimisers are crucial and can be non-trivial. The MSE comes as 51.6002.

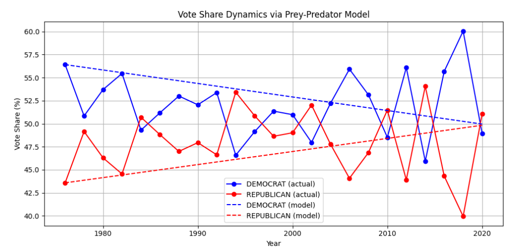
G. SIR



Leverages the well-understood SIR framework to model opinion dynamics as a social contagion process. Support for a party "spreads" through the susceptible population.
Fitted Parameters:

$$\beta = 0.466667261949707 87$$
$$\gamma = 0.0190994051822268 72$$

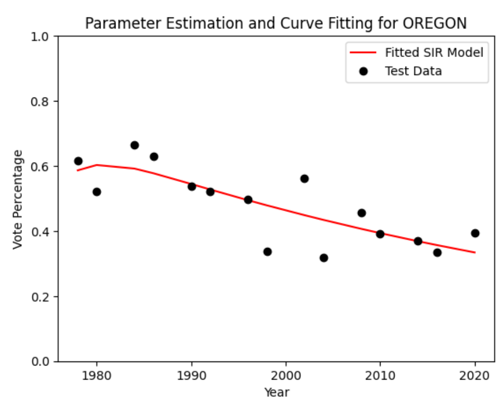
H. Predator Prey model



The SIR models spread/recovery within one population (party), while the Lotka-Volterra models the interaction between two populations (parties). It's quite similar to the PINN model but learns weights using the Least squares method instead of backpropagation. **Fitted Parameters:**

$$\alpha = 0.0000$$
$$\beta = 0.0001$$
$$\delta = 0.0001$$
$$\gamma = 0.0000$$

I. Evolutionary Algorithm



This plot shows how the SIR model best fits the data using an evolutionary algorithm to estimate the beta and gamma parameters. This highlights how, due to its simplicity, the parameters can be estimated accurately using less mathematically involved methods. **Best Parameters**

$$\beta = 0.6786447370381923$$
$$\gamma = 0.691741016475063$$

4. DISCUSSION

A. Key Findings

Our study demonstrates that biological modelling frameworks, particularly the SIR and Lotka-Volterra paradigms, offer novel insights into electoral dynamics. The SIR model successfully captured opinion spread as a contagion process, with fitted parameters

$$\beta = 0.47$$

$$\gamma = 0.019$$

indicating rapid transmission and slow recovery, consistent with polarisation trends observed in modern democracies.

The Lotka-Volterra model revealed oscillatory dynamics between competing parties, though low fitted interaction rates

$$\beta = 0.0001$$

$$\delta = 0.0001$$

Suggest weaker candidate competition than predator-prey systems.

The Agent-Based Modelling (ABM) simulations highlighted how targeted campaigns disproportionately sway undecided voters, aligning with studies on misinformation propagation.

B. Consistency and Divergence with Prior Work

Our SIR-based results align with prior applications of epidemiological models to social dynamics but contrast qualitatively with agent-based studies emphasising network structure. The Neural ODE's higher MSE compared to ARIMA underscores the trade-off between continuous-time flexibility and overfitting risks.

C. Interpretability and Transparency of Models

The S-curve logistic and SIR models offer high interpretability, allowing researchers and policymakers to understand the underlying mechanisms of opinion spread and saturation. In contrast, models like Neural ODES and PINNS, while powerful, act more as "black boxes" and may be less transparent to non-expert stakeholders. This difference is crucial for real-world adoption and trust in model recommendations.

D. Correlation within States

The Pearson Correlation of voter percentages of various states was very low, with an average value of 0.012. This suggests that all aggregate models should be fitted on States individually. The SIR and Lotka-Volterra frameworks are sensitive to initial parameter choices (e.g., initial vote shares, infection/recovery rates). In real-world applications, it is necessary to account for campaign shocks that can perturb initial conditions.

E. Parameter Estimation

The models are sensitive to parameters, and the parameter search is non-trivial, although similar optimum parameter sets can be found through different methods. We used evolutionary algorithms, the least squares method, sliding window average and the simple grid search. They all gave similar parameter values. Interestingly, the parameters in the SIR model were fairly consistent among states and the variation is primarily due to initial conditions. This suggests that the 'infection rates' and susceptible rates analogously used in our model might be due to social or physical constraints on the system.

5. CONCLUSION

The SIR model provides a straightforward framework for examining political behaviour through its parameters, beta and gamma. However, its application in political contexts is limited due to several shortcomings. The model oversimplifies complex dynamics by neglecting factors such as voter demographics, external events, campaign expenditures, media influence, and the presence of multiple parties. Moreover, the assumption of constant parameters over a prolonged period (1976-2020) is problematic, as political dynamics are unlikely to remain static. The linear progression from Susceptible to Infected to Recovered/Removed overlooks the possibility of voters switching allegiances or experiencing waning support. Furthermore, the model's implicit assumption of a homogeneous electorate undermines its applicability, presuming equal influence across individuals.

Similarly, the Lotka-Volterra model reduces intricate political interactions to a simplistic two-population framework. The assignment of "prey" and "predator" roles may be arbitrary and misleading. While Physics-Informed Neural Networks (PINNS) offer innovative modelling approaches, they are sensitive to hyperparameters and initialisation, which can lead to instability. The reliance on highly aggregated national data also risks overlooking regional variations and specific political events that significantly impact party dynamics. However, even in their current form, both the SIR and Lotka-Volterra models serve as useful conceptual tools with large scope to be modified for real-world forecasting.

In light of these limitations, we propose a framework for a Multi-Compartment and Multi-Group Susceptible-Infected-Recovered (SIR) model, which opens several promising avenues for further research into electoral dynamics. By stratifying the electorate based on demographic, ideological, and geographic factors, future studies can explore how distinct SIR tracks with varying parameters across groups might capture the nuanced behaviour of voter transitions, including potential return flows that account for party allegiance and disengagement shifts.

Additionally, investigating the incorporation of vaccination-style "immunity" compartments could provide insights into long-term party loyalty and its effects on re-infection through persuasion over time. Researchers may also examine the introduction of time-varying and externally forced parameters to assess how they adapt to the evolving landscape of electoral campaigns and the impacts of significant events, such as scandals and news cycles, on voter behaviour.

Embedding the SIR contagion within realistic social and contact networks could facilitate studies on clustering phenomena and echo chambers, offering valuable insights into how interpersonal ties influence the spread of political ideas. Moreover, adapting Lotka-Volterra equations to interpret inter-party interactions within a bipartite voter-party graph could deepen our understanding of heterogeneous influences and the complexities of non-symmetric competition among parties.

Lastly, exploring the hybridisation of SIR or Lotka-Volterra layers with agent-based models can enhance research by integrating individual-level behaviours, thresholds, and media biases. This multifaceted approach not only encourages the advancement of theoretical understanding but also holds practical implications for political strategists and researchers moving forward.

REFERENCES

1. J. M. Epstein, *Generative Social Science: Studies in Agent-Based Computational Modeling* (Princeton University Press, 2006).
2. G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control* (Wiley, 2008), 4th ed.
3. C. A. Sims, Econometrica **48**, 1 (1980).
4. R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, “Neural ordinary differential equations,” in *Advances in Neural Information Processing Systems (NeurIPS),* , vol. 31 (2018).
5. R. Pearl and L. J. Reed, Proc. National Acad. Sci. United States Am. **6**, 175 (1920).
6. M. Raissi, P. Perdikaris, and G. E. Karniadakis, J. Comput. Phys. **378**, 686 (2019).
7. W. O. Kermack and A. G. McKendrick, Proc. Royal Soc. A: Math. Phys. Eng. Sci. **115**, 700 (1927).
8. J. H. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan Press, 1975).