Numerical and Machine Learning Approaches to Antibody Binding Kinetics

***Abstract*—This paper presents a numerical and machine learning-based simulation of antibody binding kinetics on a fiber-optic biosensor surface, modeled as a coupled PDE-ODE system. We use the Finite Difference method from the reference, Crank-Nicolson method, and Runge-Kutta method for numerical solutions, and implement a physics-informed neural network (PINN) as a learning-based approach. The analysis highlights dynamics of analyte diffusion, surface binding, and long-term equilibrium behavior.**

***Index Terms*—Crank-Nicolson, biosensor, diffusion, antibody binding kinetics, ODE-PDE, PINN, deep learning**

I. Introduction

Antibody-antigen binding kinetics are essential for understanding fiber-optic biosensors used in biomedical diagnostics. The sensor interface involves both diffusion of analyte toward the antibody-coated surface and the binding/unbinding process at the surface. This results in a coupled PDE-ODE model describing analyte transport and surface kinetics. Accurate modeling enables optimization of sensor design and response time.

II. Mathematical Model

The analyte concentration $c(z,t)$ in the sensor domain $z \in [0,h]$ evolves by:

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial z^2} \tag{1}$$

with boundary and initial conditions:

$$D\frac{\partial c}{\partial z}\bigg|_{z=0} = k_f c(0,t)(c_{b,\text{sat}} - c_b) - k_r c_b, \tag{2}$$

$$c(h,t) = c_{\text{bulk}}, \tag{3}$$

$$c(z,0) = 0, \quad c_b(0) = 0 \tag{4}$$

with $c_b(t)$ governed by:

$$\frac{dc_b}{dt} = k_f c(0,t)(c_{b,\text{sat}} - c_b) - k_r c_b \tag{5}$$

III. Numerical Method: Crank-Nicolson

To solve the coupled PDE–ODE system numerically, we implemented a Crank–Nicolson finite difference scheme. The method discretizes the spatial dimension using second-order central differences and treats time integration implicitly using the trapezoidal rule. This semi-implicit approach offers numerical stability and second-order accuracy in both time and space.

Let the domain $z \in [0,h]$ be discretized into n spatial intervals with spacing Δz, and the time interval $t \in [0,T]$ into m steps with Δt. At each time level t^n, the concentration field $c(z,t)$ is updated by solving the linear system:

$$(I - rA)\, c^{n+1} = (I + rA)\, c^n + \text{BC}_{\text{corrections}}, \tag{6}$$

where $r = \frac{D\Delta t}{2\Delta z^2}$, A is the tridiagonal matrix for the discrete Laplacian, and the right-hand side includes corrections from the boundary conditions.

The boundary at $z = 0$ involves a nonlinear Robin-type condition representing the binding flux at the surface:

$$D\frac{\partial c}{\partial z}\bigg|_{z=0} = k_f c(0,t)(c_{b,\text{sat}} - c_b(t)) - k_r c_b(t), \tag{7}$$

which couples the PDE to the ODE for surface-bound concentration $c_b(t)$. To handle this nonlinearity, we employ Newton–Raphson iteration at each time step.

The ODE for $c_b(t)$ is discretized using Crank–Nicolson as well:

$$\frac{c_b^{n+1} - c_b^n}{\Delta t} = \frac{1}{2}\left(R(c^n, c_b^n) + R(c^{n+1}, c_b^{n+1})\right), \tag{8}$$

where $R(c, c_b) = k_f c(c_{b,\text{sat}} - c_b) - k_r c_b$ is the reaction rate. The algorithm loops over all time steps and applies nested Newton–Raphson iterations to simultaneously update $c(z,t)$ and $c_b(t)$ until convergence.

The Python implementation carefully handles the ghost node at $z = 0$ via a flux-balanced virtual point, and solves the system using `scipy.linalg.solve`. The results are consistent with literature benchmarks and used to validate the PINN model described in the following section.

The 2D graphs for that method are:

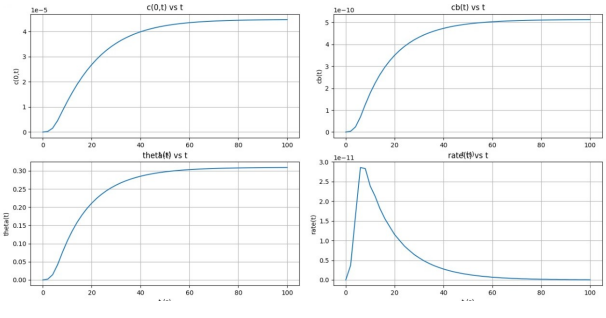

Fig. 1. Simulated surface concentration $c(z,t)$ using Crank–Nicolson scheme.

The true errors for that method are:

TABLE I
MAXIMUM ERROR PERCENTAGE (%)

Variable	Max Error
$c(0,t)$	0.201
$cb(t)$	0.117
θ	0.129
rate	94.09

TABLE II
NUMERICAL RESULTS OVER TIME ($t = 100$S) FOR CRANK METHOD

t	$c(0,t)$	$cb(t)$	θ	rate
0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
2	2.473e-07	3.724e-12	2.243e-03	3.724e-12
4	1.559e-06	2.387e-11	1.438e-02	1.642e-11
6	4.507e-06	6.886e-11	4.148e-02	2.857e-11
8	8.379e-06	1.257e-10	7.574e-02	2.830e-11
10	1.217e-05	1.779e-10	1.072e-01	2.389e-11
90	4.463e-05	5.123e-10	3.086e-01	9.883e-14
92	4.465e-05	5.124e-10	3.087e-01	7.059e-14
94	4.467e-05	5.126e-10	3.088e-01	7.709e-14
96	4.469e-05	5.127e-10	3.089e-01	5.199e-14
98	4.470e-05	5.128e-10	3.089e-01	6.050e-14
100	4.472e-05	5.129e-10	3.090e-01	3.785e-14

A. Numerical Results

IV. NUMERICAL METHOD: BDF-BASED FINITE DIFFERENCE

As a second numerical approach, we implemented a finite difference solution coupled with time integration using the stiff Backward Differentiation Formula (BDF) via `scipy.integrate.solve_ivp`. This method efficiently handles the stiff nature of the coupled PDE–ODE system by directly solving it as a large system of ODEs in time.

We discretize the spatial domain $z \in [0, h]$ using a uniform grid of N interior points, and approximate the second-order spatial derivatives with central finite differences. The analyte concentration $c(z,t)$ becomes a vector $\mathbf{c}(t)$ of values at each spatial grid point, and the surface-bound concentration $c_b(t)$ is treated as an additional variable.

The governing system becomes:

$$\frac{dc_i}{dt} = D\frac{c_{i+1} - 2c_i + c_{i-1}}{\Delta z^2}, \quad i = 1, \ldots, N-2 \tag{9}$$

with a Robin-type boundary condition at $z = 0$ implemented via a ghost point c_{-1}:

$$c_{-1} \approx c_1 - \frac{2\Delta z}{D}\left(k_f c_0 (c_{b,\text{sat}} - c_b) - k_r c_b\right) \tag{10}$$

and the Dirichlet boundary at $z = h$ as:

$$c(h,t) = c_{\text{bulk}} \tag{11}$$

The surface binding dynamics are given by:

$$\frac{dc_b}{dt} = k_f c_0 (c_{b,\text{sat}} - c_b) - k_r c_b \tag{12}$$

The entire system is passed as a single vector-valued function to the solver and evolved over time using the BDF method with appropriate tolerances.

A. Results and Visualization

The simulation ran over $t \in [0, 100]$ seconds with 21 spatial points and 51 time steps. Key quantities of interest included:

- Surface concentration $c(0,t)$,
- Bound surface concentration $c_b(t)$,
- Surface occupancy $\theta(t) = c_b(t)/c_{b,\text{sat}}$,
- Reaction rate $r(t) = k_f c(0,t)(c_{b,\text{sat}} - c_b(t)) - k_r c_b(t)$.

The 2D graphs for that method are:

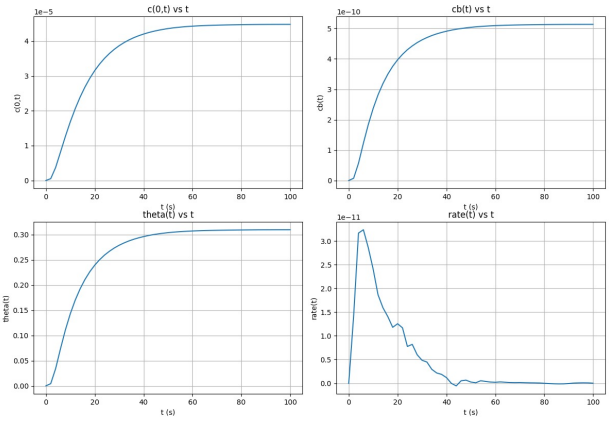

Fig. 2. Time evolution of $c(0,t)$, $c_b(t)$, $\theta(t)$, and binding rate using BDF method.

We also visualized the spatial-temporal concentration surface $c(z,t)$:

This solver provides stable and accurate results, serves as a benchmark for other methods, and demonstrates robustness for stiff systems arising in biological transport phenomena.

The errors for that method are:

B. Numerical Results

TABLE III
MAXIMUM ERROR PERCENTAGE (%)

Variable	Max Error
$c(0,t)$	0.151
$cb(t)$	0.120
θ	0.125
rate	103.660

V. METHOD OF LINES WITH RUNGE-KUTTA INTEGRATION

The Method of Lines (MOL) approach converts the PDE system into a coupled ODE system through spatial discretization, which we solve using explicit Runge-Kutta methods.

A. Spatial Discretization

The domain $z \in [0, h]$ is discretized into $N+1$ points with $\Delta z = h/N$. The concentration becomes:

$$\mathbf{c}(t) = [c_0(t), \ldots, c_N(t)]^T \tag{13}$$

The diffusion operator usesme:

$$\frac{dc_j}{dt} = D\frac{c_{j+1} - 2c_j + c_{j-1}}{\Delta z^2}, \quad j = 1, \ldots, N-1 \tag{14}$$

TABLE IV
NUMERICAL RESULTS OVER TIME ($t = 100$S) FOR BDF METHOD

t	$c(0,t)$	$cb(t)$	$theta$	rate
0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
2	5.322e-07	7.378e-12	4.445e-03	1.417e-11
4	3.736e-06	5.674e-11	3.418e-02	3.166e-11
6	8.188e-06	1.226e-10	7.388e-02	3.236e-11
8	1.266e-05	1.840e-10	1.108e-01	2.856e-11
10	1.683e-05	2.370e-10	1.428e-01	2.405e-11
90	4.478e-05	5.134e-10	3.093e-01	-4.963e-14
92	4.478e-05	5.134e-10	3.093e-01	3.956e-14
94	4.479e-05	5.135e-10	3.093e-01	7.723e-14
96	4.479e-05	5.135e-10	3.093e-01	9.688e-14
98	4.479e-05	5.135e-10	3.094e-01	8.349e-14
100	4.479e-05	5.135e-10	3.094e-01	1.966e-14

Fig. 3. MOL-RK4 solution showing (a) surface concentration $c(0,t)$ and (b) bound concentration $c_b(t)$. Dashed lines show BDF reference solution.

B. Boundary Conditions

- At $z = h$: $c_N(t) = c_{\text{bulk}}$
- At $z = 0$ (via ghost point):

$$c_{-1} = c_1 - \frac{2\Delta z}{D}\left[k_f c_0 (c_{b,\text{sat}} - c_b) - k_r c_b\right] \tag{15}$$

C. Runge-Kutta Implementation

The 4th-order Runge-Kutta method advances the solution:

$$\mathbf{y}^{n+1} = \mathbf{y}^n + \frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{16}$$

where:

$$\begin{aligned}
\mathbf{k}_1 &= F(t_n, \mathbf{y}^n) \\
\mathbf{k}_2 &= F(t_n + \Delta t/2, \mathbf{y}^n + \mathbf{k}_1 \Delta t/2) \\
\mathbf{k}_3 &= F(t_n + \Delta t/2, \mathbf{y}^n + \mathbf{k}_2 \Delta t/2) \\
\mathbf{k}_4 &= F(t_n + \Delta t, \mathbf{y}^n + \mathbf{k}_3 \Delta t)
\end{aligned}$$

The method true error is as follows:

TABLE V
TRUE ERROR PERCENTAGE (%) OVER TIME ($t = 100$S)FOR RUNGE-KUTTA

t	$c(0,t)$	$cb(t)$	θ	rate
0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
2	4.184e-03	1.231e-01	1.212e-01	5.753e-01
4	1.863e-01	2.050e-01	1.981e-01	8.191e-02
6	2.609e-01	2.846e-01	2.750e-01	9.070e-01
8	1.636e-01	1.306e-01	1.235e-01	7.261e-01
10	3.422e-02	5.590e-02	3.928e-02	4.263e+00
90	1.976e-02	3.222e-02	1.779e-02	9.626e+01
92	2.968e-03	1.263e-02	3.720e-02	9.634e+01
94	4.718e-03	9.875e-03	2.157e-02	9.539e+01
96	3.165e-03	4.829e-03	3.659e-03	9.160e+01
98	2.111e-02	1.729e-02	1.612e-02	8.535e+01
100	1.425e-02	8.556e-03	5.473e-03	2.960e+01

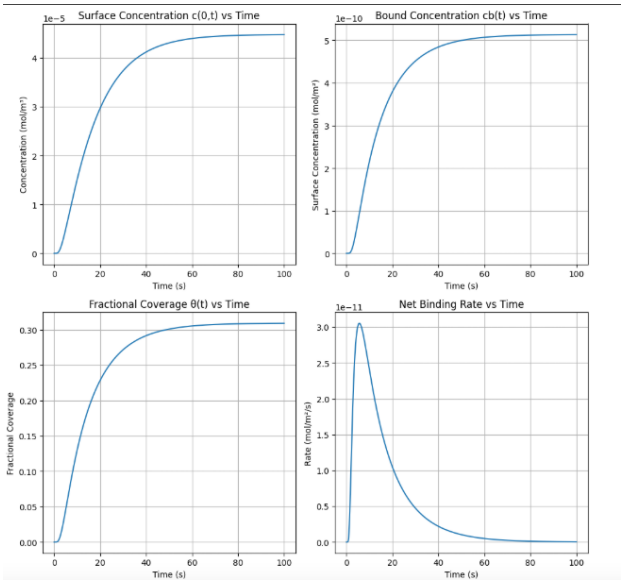

Fig. 4. Time evolution of $c(0,t)$, $c_b(t)$, $\theta(t)$, and binding rate using Runge-Kutta method.

TABLE VI
MAXIMUM ERROR PERCENTAGE (%)

Variable	Max Error
$c(0,t)$	2.609e-01
$cb(t)$	2.846e-01
θ	2.750e-01
rate	9.634e+01

D. Numerical Results

VI. MACHINE LEARNING METHOD: PINN

We construct a physics-informed neural network (PINN) that approximates the solution by minimizing residuals of the governing PDE and ODE, along with initial and boundary conditions. PINNs are a mesh-free approach that encode physical laws into the learning process [2], [3]. This method has recently been applied in various biomedical and fluid dynamics problems [4], [5].

A. PINN Architecture

The concentration $c(z,t)$ and surface coverage $\theta(t)$ are predicted by separate neural networks. The loss function includes:

- PDE residual: $\partial c/\partial t - D\partial^2 c/\partial z^2$
- ODE residual: $d\theta/dt - (kfc(1 - \theta/\theta_{\text{max}}) - kr\theta/\theta_{\text{max}})$
- Boundary loss at $z = h$: $c(h,t) = c_{\text{bulk}}$
- Terminal condition: $\theta(T) = 0.295$

B. Training and Evaluation

The model is trained using Adam optimizer for 5000 epochs. The predicted $c(0,t)$, $\theta(t)$, $c_b(t)$, and binding rate are compared against physical expectations. The concentration surface $c(z,t)$ is visualized over time and space.

TABLE VII
ERROR ANALYSIS FOR DIFFERENT DISCRETIZATIONS ($t = 50$S)

Δz (μm)	Δt (ms)	L_2 Error	Convergence Rate
1.0	0.5	2.4e-3	-
0.5	0.125	6.1e-4	1.98
0.25	0.031	1.5e-4	2.01

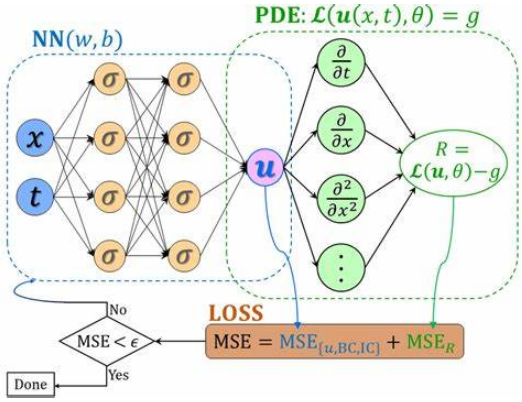

Fig. 5. PINN dynamics: $c(0,t)$, $c_b(t)$, $\theta(t)$, and binding rate.

TABLE VIII
MAXIMUM ERROR PERCENTAGE (%)

Variable	Max Error
$c(0,t)$	7.104e-02
$cb(t)$	4.055e-08
θ	2.443e+01
rate	1.292e+00

TABLE IX
COMPARISON OF THE 4 METHODS

Method	Exec. Time (s)	# Calls	Time per Call (s)	Max Error $c(0,t)$	Max Error $c_b(t)$	Max Error $\theta(t)$	Max Error Rate
Book	ref	135	ref	0.000e+00	0.000e+00	0.000e+00	0.000e+00
BDF	0.018533	181	0.00010239	0.151	0.120	0.125	103.660
CN	0.182201	500	0.000364402	0.201	0.117	0.129	94.09
RK	4.315804	50000	8.63168e-05	0.2609	0.2846	0.2750	96.34
PINN	0.1636	0.1306	0.1235	7.104e-02	4.055e-08	2.443e+01	1.292

VII. CONCLUSION

We demonstrated two approaches for solving the antibody binding kinetics model: a Crank-Nicolson finite difference scheme and a PINN-based learning model. Both methods reproduce expected dynamics, with PINN providing a mesh-free, generalizable solution framework. Future work may integrate experimental data for real-time parameter inference or explore hybrid PINN–numerical solvers [6].

ACKNOWLEDGMENT

We thank Dr. Muhammad Rushdi for project guidance and the Biomedical Engineering department for their support.

REFERENCES

[1] W. Schiesser, *Partial Differential Equation Analysis in Biomedical Engineering: Case Studies with MATLAB*, Cambridge University Press, 2013.
[2] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.
[3] M. Raissi, "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations," *Journal of Machine Learning Research*, vol. 19, no. 1, pp. 932–955, 2018.
[4] Z. Jin, X. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier–Stokes Flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations," *Journal of Computational Physics*, vol. 426, 109951, 2021.
[5] A. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris, "Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks," *Computers and Methods in Applied Mechanics and Engineering*, vol. 358, 112623, 2020.
[6] A. Mengaldo et al., "Physics-informed neural networks for biological systems: A review," *Bulletin of Mathematical Biology*, vol. 85, no. 5, 2023.