

kinBPC - Ordinary Differential Equations (ODE) & Examples in Ligand Binding and Enzyme Kinetics

José Maurício Schneedorf Ferreira da Silva

2026-02-16

A broad range of scientific problems involves rates of change of one or more quantities with respect to one or more independent variables—situations naturally described by *differential equations*. When a single independent variable is involved (typically time), these derivatives are referred to as *ordinary differential equations (ODEs)*. Solving an *initial value problem* for an ODE is commonly done through numerical approximation schemes, among which the classical fourth-order *Runge–Kutta–Fehlberg* method is widely known. The program *kinBPC* (*Biophysical-chemistry kinetics*) uses the calculator’s internal ODE solver (RKF) to model the kinetics of molecular interaction.

1 Files

1. Program
2. Source code
3. This tutorial

2 Example 1 - Ligand-Binding

The interaction considered here is a bimolecular ligand–receptor reaction:



Where:

- R = free receptor (protein, nucleic acid, etc.);
- LR = complex formed;
- k_1 = association rate constant;
- k_2 = dissociation rate constant.

The *association* rates are described by:

$$\frac{d[LR]}{dt} = k_1[L][R]$$

$$\frac{d[L]}{dt} = -k_1[L][R]$$

$$\frac{d[R]}{dt} = -k_1[L][R]$$

Likewise, the *dissociation* rates are given by:

$$\frac{d[LR]}{dt} = -k_2[LR]$$

$$\frac{d[L]}{dt} = k_2[LR]$$

$$\frac{d[R]}{dt} = k_2[LR]$$

At equilibrium (and, more generally, along the full time course), the *net* reaction rates are the sum of association and dissociation contributions. Therefore, the system to be solved is:

$$\frac{d[L]}{dt} = -k_1[L][R] + k_2[LR]$$

$$\frac{d[R]}{dt} = -k_1[L][R] + k_2[LR]$$

$$\frac{d[LR]}{dt} = k_1[L][R] - k_2[LR]$$

Numerically, the evolution of ([L]), ([R]), and ([LR]) over small time steps (t) can be computed from a system of differential equations in vector form:

$$\Delta x = f(x), \Delta t$$

3 Program kinBPC

The program uses the HP50G RKF routine (Runge–Kutta–Fehlberg) to solve the ODE system numerically. The algorithm computes ([L]), ([R]), and ([LR]) along time and plots the resulting kinetic curves. Data entry requires a subroutine containing the ODE system written in *postfix* (RPN) syntax (figure).

Example inputs and outputs are:

```
# Inputs
k1 = 0.02;
k2 = 0.001;
tol = 1e-5 (tolerance);
x = 0 (initial time);
y = [0.8 1 0] (initial vector for L, R, and LR);
F (RPN subroutine for the 3 equations)

Run "kinBPC"

# Outputs
Plot of t vs. L, R, and LR;
Matrix of computed values on the stack;
```

The program is relatively slow on the physical calculator (~30 s), although it runs essentially instantaneously in virtual versions or scientific suites. Nearly identical results were obtained in ~1 s using the approach described by Prinz (2011, ch. 6) with [Octave](#) (function *lsode*).

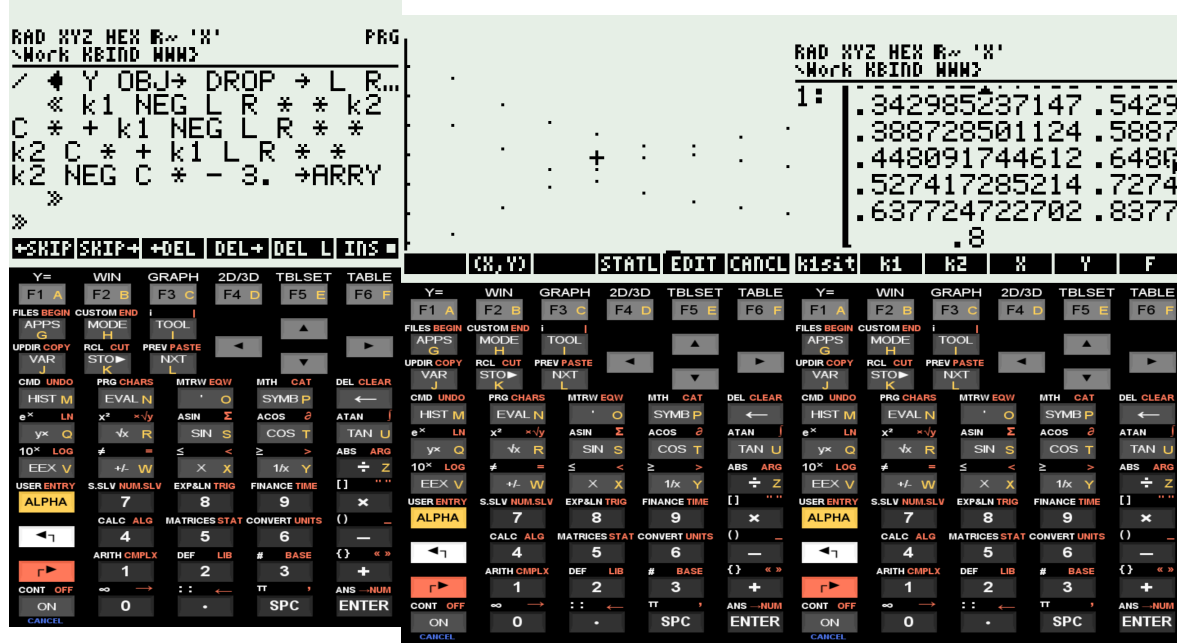
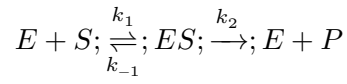


Figure 1: *kinBPC* running on the Android HP50G emulator ([Go49gp](#)), illustrating ligand–receptor kinetics for a one-site model with the provided example.

4 Example 2 - Enzyme Kinetics

The program *kinBPC* solves the time evolution of *S* (substrate), *E* (enzyme), *ES* (enzyme–substrate complex), and *P* (product) according to the classical catalytic scheme:



Where:

- S = substrate
- E = free enzyme
- ES = enzyme–substrate complex
- P = product
- k_1 = association rate constant
- k_{-1} = dissociation rate constant
- k_2 = catalytic rate constant

The kinetic behavior of the system is described by the following system of ordinary differential equations:

$$\frac{d[S]}{dt} = -k_1[E][S] + k_{-1}[ES]$$

$$\frac{d[ES]}{dt} = k_1[E][S] - (k_{-1} + k_2)[ES]$$

$$\frac{d[P]}{dt} = k_2[ES]$$

Since enzyme is conserved during catalysis:

$$[E]_0 = [E] + [ES]$$

The numerical solution is obtained by integrating the system:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x})$$

or, in discrete approximation,

$$\Delta\mathbf{x} = f(\mathbf{x})\Delta t$$

The program uses the RKF function of the HP50G to compute the Runge–Kutta–Fehlberg numerical solution. The algorithm calculates the time evolution of S, ES, and P, and plots the kinetic profile showing substrate depletion and product formation.

Data input requires a subroutine written in postfix (RPN) syntax defining the system of differential equations (see figure).

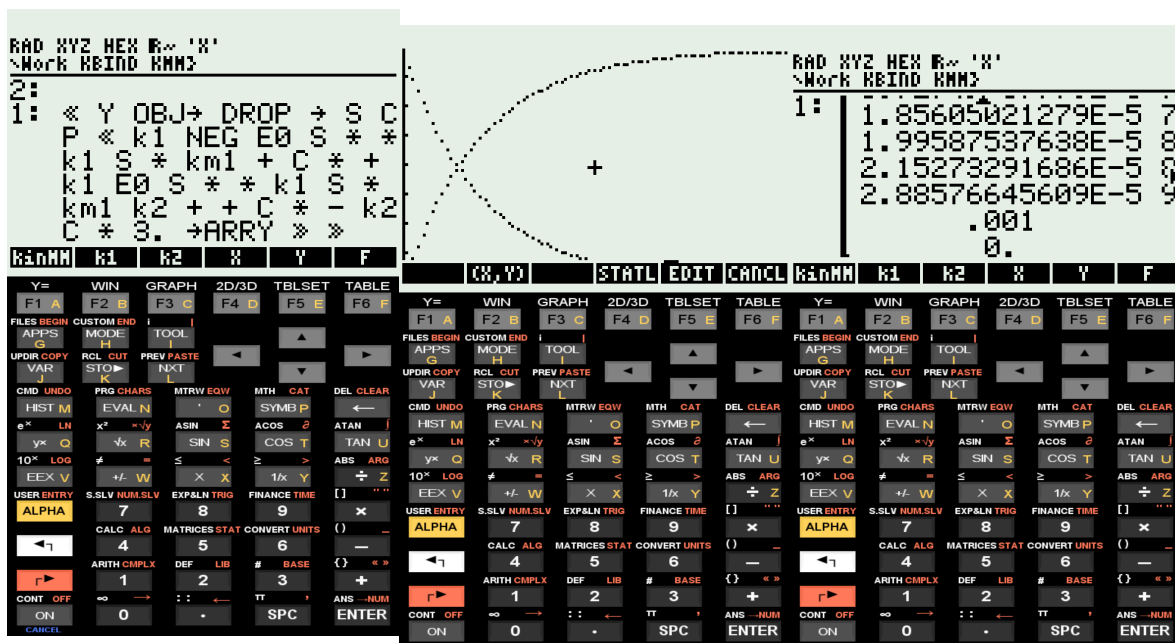
Example of input and output:

```
# Inputs
k1 = 1000;
km1 = 0.1;
k2 = 0.05;
E0 = 0.005;
tol = 1e-5 (tolerance);
x = 0 (initial time);
y = [0.001 0 0] (initial vector for S, ES and P)
F - RPN subroutine defining the 3 equations

Run "kinBPC"

# Outputs:
Plot of t vs S and P;
Matrix containing S, ES, P and t values on the stack;
```

The execution on the physical calculator is relatively slow (~20 s for 9 time points), whereas it is nearly instantaneous in virtual HP50G versions or scientific computing environments (e.g., Octave using custom scripts).



(a) Subroutine defining the three differential equations for Michaelis-Menten kinetics. (b) Kinetic plot showing substrate decay and product formation. (c) Matrix containing S, ES, P, and time values.

Figure 2: Program *kinBPC* running on the Android HP50G emulator ([Go49gp](https://www.go49gp.com/)), illustrating enzymatic kinetics simulation.

References

1. Prinz, Heino. *Numerical Methods for the Life Scientist: Binding and Enzyme Kinetics Calculated with GNU Octave and MATLAB*. Springer Science & Business Media, 2011.
2. Murray, J. D. *Mathematical Biology II: Spatial Models and Biomedical Applications*. Springer, 2003.

Author

José Maurício Schneedorf Ferreira da Silva
Dept. Biochemistry, Institute of Biomedical Sciences
Federal University of Alfenas - UNIFAL-MG, Brazil
Email: jose.dasilva@unifal-mg.edu.br

Site: <https://bioquanti.netlify.app>
Lattes: <http://lattes.cnpq.br/0436922594542722>
ORCID: 0000-0002-2031-6315