pypsdier

Release 1.1.1

Nov 20, 2020

Documenation:

pypsdier is a python library to solve pde reaction-difussion equations, considering inmmobilized catalyst particles.

CHAPTER 1

Try it out!

- In Google Colab.
- In MyBinder.

1.1 Introduction

1.1.1 Objective

Around 2008, we started the development of a numerical implementation of generic reaction diffusion equations for reactors using enzyme immobilization on small porous particles. The objective was (and still is) to provide a simple interface to solve reaction diffusion equations.

The immobilization of enzymes is a requisite for the re-use of these catalysts in repeated cycles in batch configuration or in continuous reactors. The recovery and/or retention of the enzyme catalyst is technical and economically feasible when micrometric or milimetric particles are used. The covalent attachment of the enzyme molecule to a porous solid support have shown high stabilization with different enzymes. Therefore, immobilization is neccessary and convenient for the efficient utilization of enzymes in technological processes and industrial settings.

Simulation of the concentrations of substrates and products is, nevertheless, harder than expected due to the complexity of the reaction and diffusion processes.

The reaction takes places then in a heterogeneous system composed by the solid catalysts particles and the bulk liquid. The catalysis process is carried out inside the particle porous instead of the bulk liquid solvent. The immediate consequence of this fact is that, along with the reaction, mass transfer occurs inside the particle and between the particle and the bulk medium. The modeling of this heterogeneous process must considers reaction and diffusion components in the reactor performance equation.

1.2 Installation

The repository for the code is hosted at https://github.com/sebastiandres/pypsdier.

The current implementation has been developed in Python 3. To use as a simulation engine, the libraries numpy, xlwt, scipy and matplotlib are needed.

1.2.1 Install from pypi

You can install the library from pypi. This is the safe way. Don't stray from this path.

```
pip install pypsdier
```

1.2.2 Install from repository

You can install the library directly from the latest available version on github. This is good for testing the library, but might encounter some bugs, in which case you should let us know!

```
pip install git+https://github.com/sebastiandres/pypsdier.git
```

1.3 Examples

1.3.1 Example in Google colab

Here is an executable example using Google Colab. Requires a google account (but it's worth it :).

1.3.2 Example in mybinder

Here is an executable example using MyBinder. Does not requires any account, but it will not store results.

1.3.3 Code example

To run all the next lines you need to install the library. We hope you'll appreciate that all you need is to define the inputs and plot options, and run the simulation. Libraries and outputs are silently handled. Saving, plotting or exporting the results is trivially easy for the user.

We'll define the simplest experiment possible.

The first thing is to setup the inputs and plotting options. This requires to define dictionaries with specific keys.

```
def MichaelisMenten(S, E0, k, K):
"""Definition for Michaelis Menten reaction with inputs E0 [mM], k [1/s] and K [mM]"""
return (-k*E0*S[0]/(K+S[0]), )

inputs = {}
inputs["SimulationTime"] = 120. # [s]
inputs["SavingTimeStep"] = 1. # [s]
inputs["CatalystVolume"] = 0.5 # [mL]
inputs["BulkVolume"]  = 100.0  # [mL]
inputs["Names"] = ('Substrat',)  # legend for the xls, reports and plots
inputs["InitialConcentrations"] = (1.3,)    # [mM]
inputs["EffectiveDiffusionCoefficients"] = (5.3E-10,)  # [m2/s]
inputs["CatalystParticleRadius"] = [40.0E-6, 60.0E-6, 80.0E-6] # [m]
inputs["CatalystParticleRadiusFrequency"] = [0.3, 0.5, 0.2] # []
```

(continues on next page)

(continued from previous page)

```
inputs["ReactionFunction"] = MichaelisMenten # function
inputs["ReactionParameters"] = (41 , 0.13)    # [1/s], [mM/s], parameters
inputs["CatalystEnzymeConcentration"] = 0.35 # [mM]

plot_options = {}
plot_options["title"] = "Michaelis Menten Reaction"
plot_options["label_x"] = "Reaction time [s]"
plot_options["label_y"] = "Concentration [mM]"
plot_options["ode_kwargs"] = {'label':'ode', 'color':'black', 'marker':'', 'markersize
↪':6, 'linestyle':'dashed', 'linewidth':2}
plot_options["pde_kwargs"] = {'label':'pde', 'color':'black', 'marker':'', 'markersize
↪':6, 'linestyle':'solid', 'linewidth':2}
plot_options["data_kwargs"] = {'label':'exp', 'color':'red', 'marker':'s', 'markersize
↪':6, 'linestyle':'none', 'linewidth':2}
plot_options["data_x"] = [0.0, 30, 60, 90, 120]
plot_options["data_y"] = [1.3, 0.65, 0.25, 0.10, 0.0]
```

Creating a new simulation requires to use a new simulation interface.

```
import pypsdier
SIM = pypsdier.SimulationInterface()
SIM.new(inputs, plot_options)
```

To simulate you need to the corresponding method:

```
SIM.simulate("pde")
SIM.simulate("ode")
```

At any point of the code you can use the *status* method to know if the required libraries are installed, what are the inputs, plot options and simulation statuses.

```
SIM.status()
```

You can plot the results with the *plot* method. If needed, you can update the plot_options dictionary. Use *plot?* to know available plotting arguments.

```
SIM.plot()
```

You can generate and download a compressed simulation file, so you can late load your results

```
SIM.save("SIM.rde")
```

Or you can generate an excel file to explore the results to use a more familiar program.

```
SIM.export_xls("SIM.xls")
```

1.4 Problem

1.4.1 Subtitle

Some text

1.4.2 Subtitle

Some other text

1.5 Equations

1.5.1 Variables and parameters

Let's consider a substance S. Let's call:

- $S_b(t)$: Concentration of the substance on the bulk (liquid) phase, outside all particles. The substance could be a substract or a product of the reaction. It is usually measured in mols per liter.
- V_b: Total volume of the bulk (liquid) phase. Usually measured in liters.
- $S(t, r, R_i)$: Concentration at time t and radial position r, inside a particle of radius R_i. Measured in the same units of $S_b(t)$.
- $f(R)$: Particle size distribution. Typically, this is a discrete approximation of the real (measurable but ultimately unknown) particle size distribution. For practical purposes we will consider a finite discrete distribution with N_R different particle sizes, where the probability p_i for a particle having radius R_i for $i \in \{1, 2, \cdots, N_R\}$ with $\sum_{i=1}^{N_R} p_i = 1$.
- V_R: total volume of particles, experimentally obtained with the total weight and density of the catalyst particles. Measured in the same units as V_b.
- D_S: Effective diffusion coefficient of substance S inside the (porous) particle. It has the units meters squared / second.
- v_e: Effective reaction rate at which the amount of substance S changes without considering diffusional restrictions. If $v_e > 0$ it is ussually called a product, while $v_e < 0$ is called a substract. This is usually measured in the units of S_b per second.

1.5.2 Impact of a particle distribution

We define N_R the number of different particle radii. A discrete particle size distribution has probability p_i for a particle having radius R_i, for $i \in \{1, 2, \cdots, N_R\}$ with $\sum_{i=1}^{N_R} p_i = 1$. The probability p_i is interpreted in a frequentist approach: it is simply the fraction of particles of the size R_i, given by $p_i = n_i/n$ with $n = \sum_{i=1}^{N_R} n_i$ being the total number of particles.

We can then work out explicitely the total number of particles from the total volume of the particles:

$$\begin{aligned} V_R &= \sum_{i=1}^{N_R} n_i \frac{4\pi}{3} R_i^3 = \sum_{i=1}^{N_R} p_i \, n \, \frac{4\pi}{3} R_i^3 \\ &= n \, \frac{4\pi}{3} \sum_{i=1}^{N_R} p_i R_i^3 \end{aligned}$$

That is, the total number of particles is given by the total volume and the expected volume of a single particle.

$$n_i = p_i n = p_i \frac{V_R}{\frac{4\pi}{3} E\,[R^3]}$$

Let's consider a numerical example. Let's imagine we have a total volume of $V_R = 10\ [ml] = 1.0\ E-8\ [m^3]$ Let's consider the following distribution $p_1 = 0.4$ and $p_2 = 0.6$, for $R_1 = 0.9R_0$ and $R_2 = 1.1R_0$, where $R_0 = 6.5E-9\ [m]$ respectively.

We can compute the following values:

- The expected radius is $E[R] = p_1 R_1 + p_2 R_2 = XxX$
- The volume for a expected radius is $\frac{4}{3}\pi(E[R])^3 = XxX$
- The number of particles is $V_R/\frac{4}{3}\pi(E[R])^3 = XxX$.
- The number of particles of size R_1 and R_1 are $n_1 = p_1 n = XxX$ y $n_1 = p_1 n = XxX$, respectively.
- The total surface is

1.5.3 How to model the effective reaction rate

The effective reaction rate v_e is a function of several terms, and occurs only inside the particles, where the catalist is attached to the surface of the porous structure. In a particle of radius R_i, it would be:

$$\begin{aligned} v_e(t, r, R_i) &= v_e\left(S(t, r, R_i), E(t, r, R_i), \text{other relevant parameters}\right) \\ &\approx v\left(S(t, r, R_i), E_{max}, \text{other relevant parameters}\right) \times I(t) \times Z(r, R_i) \end{aligned}$$

Where:

- $v(S, E, \text{other relevant parameters})$: the reaction rate, measured in the units of S_b per second.
- $I(t)$: Enzime Inactivation. It only has time dependance, being bounded between 0 and 1 and decreasing: $0 \leq I(t) \leq 1$. It has no units. It models tha catalyst inhibition growing over time.
- $Z(r, R_i)$: Enzime radial distribution, that only has space dependance and being non-negative, $0 < Z(r, R_i)$ and such that the total enzime applied to all particles is a known value E_0:

$$\begin{aligned} E_0 &= \sum_{i=1}^{N_R} n_i \int_0^{R_i} E_{max} Z(r, R_i) 4\pi r^2 dr \\ &= n \sum_{i=1}^{N_R} \frac{n_i}{n} \int_0^{R_i} E_{max} Z(r, R_i) 4\pi r^2 dr \\ &= n E_{max} 4\pi \sum_{i=1}^{N_R} p_i \int_0^{R_i} Z(r, R_i) r^2 dr \end{aligned}$$

Here we have used n_i the number of particles of size R_i, and n the total number of particles, and the relationship between volume and particle size distibution:

$$n = \frac{V_R}{\frac{4\pi}{3} E\left[R^3\right]} = \frac{V_R}{\sum_{i=1}^{N_R} p_i \frac{4}{3}\pi R_i^3}$$

1.5.4 The equations

The equations, boundary conditions and initial conditions are given for $S_b(t)$ and $S(t, r, R_i)$.

The reaction diffusion equation, for $t > 0$ and $0 < r < R_i$:

$$\frac{\partial S}{\partial t}(t, r, R_i) = D_S \left(\frac{\partial^2 S}{\partial r^2}(t, r, R_i) + \frac{2}{r} \frac{\partial S}{\partial r}(t, r, R_i) \right) - v_e\left(S(t, r, R_i)\right) I(t) Z(r, R_i)$$

The boundary condition at the center of the particle for $t > 0$:

$$\frac{\partial S}{\partial r}(t, 0, R_i) = 0$$

The boundary conditions at the surface of the particles are

$$S_b(t) = S(t, R_i, R_i)$$

and

$$\begin{aligned}\frac{dS}{dt}(t, R_i, R_i) &= -3D_S \frac{V_c}{V_R E\left[R^3\right]} E\left[R^2 \left.\frac{\partial S}{\partial r}\right|_{r=R}\right] \\ &= -3D_S \frac{V_c}{V_R \sum_{i=1}^{N_R} R_i^3} \sum_{i=1}^{N_R} R_i^2 \frac{\partial S(t, R_i, R_i)}{\partial R}\end{aligned}$$

The initial conditions are

$$\begin{aligned}S_b(0) &= S_0 \\ S(0, r, R_i) &= 0 \text{ for } 0 \leq r < R_i \text{ and } i \in \{1, 2, \cdots, N_R\}\end{aligned}$$

1.6 Implementation

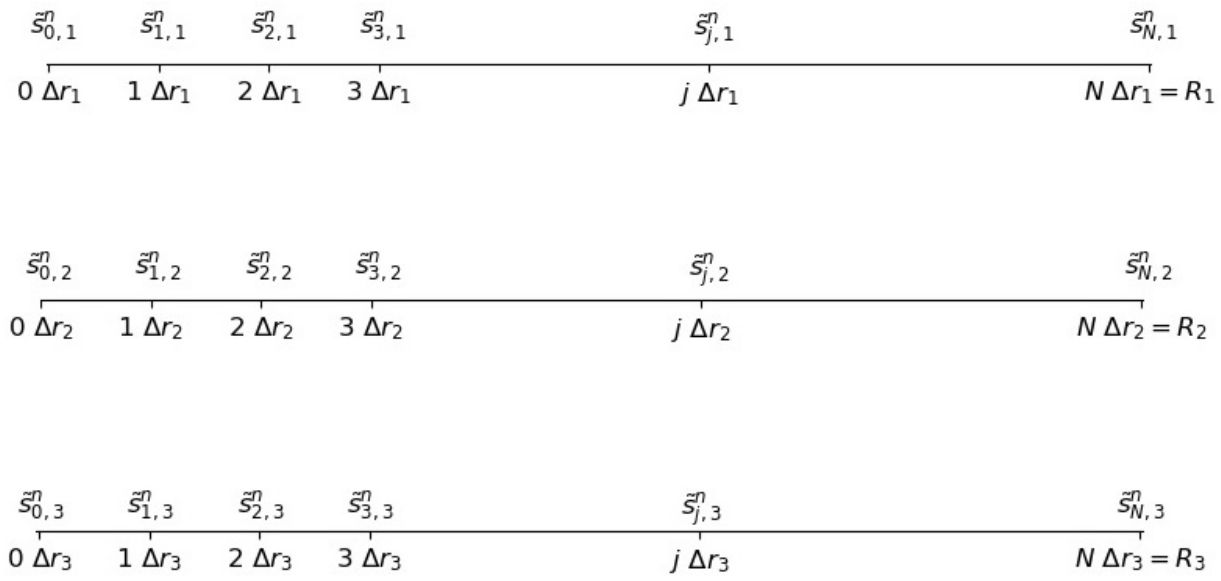
1.6.1 The Numerical Discretization

Consider the following discretization, with N_x intervals, thus having $\Delta r_i = \frac{R_i}{N_x}$ and timestep Δt.

Let's define the approximations:

$$\begin{aligned}\tilde{s}_b^n &= S_b(n\Delta t) \\ \tilde{s}_{j,i}^n &= S(n\Delta t, j\Delta r_i, R_i)\end{aligned}$$

Consider the figure:



For each timestep Δt, we have $N_x + 1$ unknowns in each particle size, and one unknown in the bulk phase, so the total number of unknown are $(N_x + 1)N_R + 1$.

The time partial derivative can be discretized as:

$$\frac{\partial S}{\partial t}(n\Delta t, j\Delta r_i, R_i) \approx \frac{\tilde{s}_{j,i}^{n+1} - \tilde{s}_{j,i}^n}{\Delta t}$$

The central implicit discretization for the first and second partial space derivatives are:

$$\frac{\partial S}{\partial r}(n\Delta t, j\Delta r_i, R_i) \approx \frac{1}{2}\frac{\tilde{s}^{n+1}_{j+1,i} - \tilde{s}^{n+1}_{j-1,i}}{2\Delta r_i} + \frac{1}{2}\frac{\tilde{s}^{n}_{j+1,i} - \tilde{s}^{n}_{j-1,i}}{2\Delta r_i}$$
$$\frac{\partial^2 S}{\partial r^2}(n\Delta t, j\Delta r_i, R_i) \approx \frac{1}{2}\frac{\tilde{s}^{n+1}_{j+1,i} - 2\,\tilde{s}^{n+1}_{j,i} + \tilde{s}^{n+1}_{j-1,i}}{(\Delta r_i)^2} + \frac{1}{2}\frac{\tilde{s}^{n}_{j+1,i} - 2\,\tilde{s}^{n}_{j,i} + \tilde{s}^{n}_{j-1,i}}{(\Delta r_i)^2}$$

The one-sided discretization for the first space derivatives are:

$$\frac{\partial S}{\partial r}(n\Delta t, 0, R_i) \approx \frac{-3\,\tilde{s}^{n}_{0,i} + 4\,\tilde{s}^{n}_{1,i} - \tilde{s}^{n}_{2,i}}{2\Delta r}$$
$$\frac{\partial S}{\partial r}(n\Delta t, N_x\Delta r_i, R_i) \approx \frac{\tilde{s}^{n}_{N_x-2,i} - 4\,\tilde{s}^{n}_{N_x-1,i} + 3\tilde{s}^{n}_{N_x,i}}{2\Delta r}$$

The reaction-diffusion equation is:

$$\tilde{s}^{n+1}_j - \frac{\Delta t D_S}{2(\Delta r)^2}\left[\left(1-\frac{2}{j}\right)\tilde{s}^{n+1}_{j-1} - 2\,\tilde{s}^{n+1}_j + \left(1+\frac{2}{j}\right)\tilde{s}^{n+1}_{j+1}\right] =$$
$$\tilde{s}^{n}_j + \frac{\Delta t D_S}{2(\Delta r)^2}\left[\left(1-\frac{2}{j}\right)\tilde{s}^{n}_{j-1} - 2\,\tilde{s}^{n}_j + \left(1+\frac{2}{j}\right)\tilde{s}^{n}_{j+1}\right] - \Delta t\; v_e(\tilde{s}^{n+1}_{j,i})\; I(n\Delta t)\; Z(j\Delta r_i, R_i)$$

The boundary condition at $r = 0$ gets discretized as

$$-3\tilde{s}^{n+1}_0 + 2\tilde{s}^{n+1}_1 + \tilde{s}^{n+1}_2 = 0$$

The continuity conditions at $r = R_i$ are:

$$\tilde{s}^{n}_b = \tilde{s}^{n}_{N_x,i} \text{ for } i \in \{1, 2, \cdots, N_R\}$$
$$\tilde{s}^{n+1}_b + \sum_{i=1}^{N_c}\gamma_i(\tilde{s}^{n+1}_{N_x-2,i} + 2\,\tilde{s}^{n+1}_{N_x-1,i} - 3\,\tilde{s}^{n+1}_{N_x,i}) = \tilde{s}^{n}_b - \sum_{i=1}^{N_c}\gamma_i(\tilde{s}^{n}_{N_x-2,i} + 2\,\tilde{s}^{n}_{N_x-1,i} - 3\,\tilde{s}^{n}_{N_x,i})$$

where $\gamma_i = \frac{3D_s V_c}{V_R E[R^3]} N_x^2 \Delta r_i$

The initial condition at the surface of the particles are

$$\tilde{s}_b(0) = S_0$$
$$\tilde{s}^{0}_{j,i} = 0 \text{ for } 0 \le j < N_x$$

1.6.2 The Numerical Implementation

We stack the vectors and explicitely replace $\tilde{s}^n_b = \tilde{s}^n_{N_x,i}$, so the vector has size $NxN_R + 1$. We will use the notation $s^n_{j+iN_x} = \tilde{s}^n_{j,i}$ and $s^n_{N_xN_R+1} = s^n_b$.

For each time step, we must solve:

$$(I + A)\vec{s}^{n+1} = (I - A)\vec{s}^{n} + \Delta t\vec{v}^{n}$$

As the matrices are fixed (do not depend on the time variable), they can be computed and stored. A PLU factorization (Permutation Lower Upper) is computed for efficiently solve the equation in each time step.

The vectors and matrices are defined as:

$$\vec{s}^n = \begin{pmatrix} s_0^n \\ s_1^n \\ s_2^n \\ \vdots \\ s_{N_x-3}^n \\ s_{N_x-2}^n \\ s_{N_x-1}^n \\ \hline \vdots \\ \hline s_{(N_c-1)N_x}^n \\ s_{(N_c-1)N_x+1}^n \\ s_{(N_c-1)N_x+2}^n \\ \vdots \\ s_{(N_c-1)N_x+N_x-3}^n \\ s_{(N_c-1)N_x+N_x-2}^n \\ s_{(N_c-1)N_x+N_x-1}^n \\ \hline s_{N_R N_x+1}^n \end{pmatrix} = \begin{pmatrix} s_{0,1}^n \\ s_{1,1}^n \\ s_{2,1}^n \\ \vdots \\ s_{N_x-3,1}^n \\ s_{N_x-2,1}^n \\ s_{N_x-1,1}^n \\ \hline \vdots \\ \hline s_{0,N_c}^n \\ s_{1,N_c}^n \\ s_{2,N_c}^n \\ \vdots \\ s_{N_x-3,N_c}^n \\ s_{N_x-2,N_c}^n \\ s_{N_x-1,N_c}^n \\ \hline s_b^n \end{pmatrix}$$

$$I = \left[\begin{array}{ccccccc|c|ccccccc|c} 0 & & & & & & & \cdots & & & & & & & & \\ & 1 & & & & & & \cdots & & & & & & & & \\ & & 1 & & & & & \cdots & & & & & & & & \\ & & & \ddots & & & & \cdots & & & & & & & & \\ & & & & 1 & & & \cdots & & & & & & & & \\ & & & & & 1 & & \cdots & & & & & & & & \\ & & & & & & 1 & \cdots & & & & & & & & \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \hline & & & & & & & \cdots & 0 & & & & & & & \\ & & & & & & & \cdots & & 1 & & & & & & \\ & & & & & & & \cdots & & & 1 & & & & & \\ & & & & & & & \cdots & & & & \ddots & & & & \\ & & & & & & & \cdots & & & & & 1 & & & \\ & & & & & & & \cdots & & & & & & 1 & & \\ & & & & & & & \ddots & & & & & & & 1 & \\ \hline & & & & & & & \ddots & & & & & & & & 1 \end{array}\right]$$

$$
A = \left[\begin{array}{ccccccc|c|ccccccc}
-3 & 2 & 1 & & & & & \cdots & & & & & & & \\
a_{2,1} & b_{2,1} & c_{2,1} & & & & & \cdots & & & & & & & \\
 & a_{3,1} & b_{3,1} & c_{3,1} & & & & \cdots & & & & & & & \\
 & & & \ddots & & & & \cdots & & & & & & & \\
 & & & a_{N_x-3,1} & b_{N_x-3,1} & c_{N_x-3,1} & & \cdots & & & & & & & \\
 & & & & a_{N_x-2,1} & b_{N_x-2,1} & c_{N_x-2,1} & \cdots & & & & & & & \\
 & & & & & a_{N_x-1,1} & b_{N_x-1,1} & \cdots & & & & & & & \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
 & & & & & & & \cdots & -3 & 2 & 1 & & & & \\
 & & & & & & & \cdots & a_{2,N_R} & b_{2,N_R} & c_{2,N_R} & & & & \\
 & & & & & & & \cdots & & a_{3,N_R} & b_{3,N_R} & c_{3,N_R} & & & \\
 & & & & & & & & & & & \ddots & & & \\
 & & & & & & & \cdots & & & & a_{N_x-3,N_R} & b_{N_x-3,N_R} & c_{N_x-3,N} & \\
 & & & & & & & \cdots & & & & & a_{N_x-2,N_R} & b_{N_x-2,N} & \\
 & & & & & & & \cdots & & & & & & a_{N_x-1,N} & \\
\hline
 & & & & & -\gamma_1 & -2\gamma_1 & \cdots & & & & & & & -\gamma_1
\end{array}\right]
$$

$$
\vec{v}^n = \left(\begin{array}{c}
0 \\
V(s^n_{1,1})\ I(n\Delta t)\ Z(1\Delta r_1, R_1) \\
V(s^n_{2,1})\ I(n\Delta t)\ Z(2\Delta r_1, R_1) \\
\vdots \\
V(s^n_{N_x-3,1})\ \ I(n\Delta t)\ Z((N-x-3)\Delta r_1, R_1) \\
V(s^n_{N_x-2,1}) I(n\Delta t)\ Z((N-x-2)\Delta r_1, R_1) \\
V(s^n_{N_x-1,1}) I(n\Delta t)\ Z((N-x-1)\Delta r_1, R_1) \\
\hline
\vdots \\
\hline
0 \\
V(s^n_{1,N_c})\ I(n\Delta t)\ Z(1\Delta r_{N_c}, R_{N_c}) \\
V(s^n_{2,N_c})\ I(n\Delta t)\ Z(2\Delta r_{N_c}, R_{N_c}) \\
\vdots \\
V(s^n_{N_x-3,N_c} I(n\Delta t)\ Z((N_x-3)\Delta r_{N_c}, R_{N_c}) \\
V(s^n_{N_x-2,N_c} I(n\Delta t)\ Z((N_x-2)\Delta r_{N_c}, R_{N_c}) \\
V(s^n_{N_x-1,N_c} I(n\Delta t)\ Z((N_x-1)\Delta r_{N_c}, R_{N_c}) \\
\hline
0
\end{array}\right)
$$

1.7 Publications

1.7.1 Articles

List of articles related to pypsdier, from most recent to oldest.

Catalysts 2019, 9(11), 930.

- Title: Estimation of the effectiveness factor for immobilized enzyme catalysts through simple conversion assay.
- Authors: Valencia, P., Ibañez, F.
- Link: https://doi.org/10.3390/catal9110930
- Reproducible Computation: MyBinder , Colab.

Mathematical Methods in the Applied Sciences 2019, 42:4170-4183.

- Title: An inverse problem for an immobilized enzyme model.
- Authors: Gajardo, D., Mercado, A., Valencia, P.
- Link: https://doi.org/10.1002/mma.5637
- Reproducible Computation: MyBinder , Colab.

New Biotechnology 29(2), 218-226, 2012.

- Title: Batch reactor performance for enzymatic synthesis of cephalexin: Influence of catalyst enzyme loading and particle size.
- Authors: Valencia, P., Flores, S., Wilson, L., Illanes,
- Link: http://dx.doi.org/10.1016/j.nbt.2011.09.002
- Reproducible Computation: MyBinder , Colab.

Applied Biochemistry and Biotechnology, September 2011, Volume 165, Issue 2, pp 426-441.

- Title: Effect of Internal Diffusional Restrictions on the Hydrolysis of Penicillin G: Reactor Performance and Specific Productivity of 6-APA with Immobilized Penicillin Acylase.
- Authors: Pedro Valencia, Sebastián Flores, Lorena Wilson, Andrés Illanes.
- Link: http://dx.doi.org/10.1007/s12010-011-9262-7
- Reproducible Computation: MyBinder , Colab.

Journal of Biotechnology, Volume 150, Supplement, November 2010, Pages 77–78

- Title: Batch reactor performance for enzymatic synthesis of cephalexin: influence of catalyst enzyme loading and particle size.
- Authors: Pedro Valencia, Sebastián Flores, Lorena Wilson, Andrés Illanes.
- Link: http://dx.doi.org/10.1016/j.jbiotec.2010.08.200
- Reproducible Computation: MyBinder , Colab.

Journal of Biotechnology, Volume 150, Supplement, November 2010, Pages 388.

- Title: Temperature effect on heterogeneous enzyme catalyzed reaction: Thièle modulus and effectiveness factor analysis.
- Authors: Pedro Valencia, Sebastián Flores, Alik Abakarov.
- Link: http://dx.doi.org/10.1016/j.jbiotec.2010.09.491
- Reproducible Computation: MyBinder , Colab.

Biochemical Engineering Journal, Volume 49, Issue 2, 15 April 2010, Pages 256–263.

- Title: Effect of particle size distribution on the simulation of immobilized enzyme reactor performance.
- Authors: Pedro Valencia, Sebastián Flores, Lorena Wilson, Andrés Illanes.
- Link: http://dx.doi.org/10.1016/j.bej.2010.01.002
- Reproducible Computation: MyBinder , Colab.

List of aditional articles related to immobilized enzyme catalysts and difusional restrictions.

Biochemical Engineering Journal, Volume 91, 129-139, 2014.

- Title: Theoretical analysis of intrinsic reaction kinetics and the behavior of immobilized enzymas system for steady-state conditions.
- Authors: Praveen, T., Valencia, P., Rajendran, L.
- Link: http://dx.doi.org/10.1016/j.bej.2014.08.001

Enzyme and Microbial Technology, Volume 47, Issue 6, pp 268-276, 2010

- Title: Evaluation of the incidence of diffusional restrictions on the enzymatic reactions of hydrolysis of penicillin G and synthesis of cephalexin.
- Authors: Pedro Valencia, Lorena Wilson, Carolina Aguirre, Andrés Illanes.
- Link: http://dx.doi.org/10.1016/j.enzmictec.2010.07.010

Electronic Journal of Biotechnology, Volume 13, Issue 1, 15 January 2010, Pages 256–263.

- Title: Diffusional restrictions in glyoxyl-agarose immobilized penicillin G acylase of different particle size and protein loading.
- Authors: Andrés Illanes, José M. González, Juan M. Gómez, Pedro Valencia, Lorena Wilson.
- Link: http://dx.doi.org/10.2225/vol13-issue1-fulltext-12

1.7.2 Seminars

List of seminar publications, listed from most recent to oldest.

14th International Biotechnology Symposium, Rimini, Italy, September 2010.

- Title: Batch reactor performance for enzymatic synthesis of cephalexin: influence of catalyst enzyme loading and particle size.
- Authors: Pedro Valencia, Sebastián Flores, Lorena Wilson, Andrés Illanes.
- Link: http://www.ibs2010.org
- Reproducible Computation: MyBinder , Colab.

XXII Congreso Iberoamericano de Catálisis (CICAT), Chile, Viña del Mar, September 2010.

- Title: Efecto de las restricciones difusionales internas sobre la reacción de síntesis de cefalexina con penicilina acilasa inmovilizada.
- Authors: Pedro Valencia, Sebastián Flores, Lorena Wilson, Andrés Illanes.
- Link: http://www.cicat2010.cl/
- Reproducible Computation: MyBinder , Colab.

1.8 Documentation for Users

As a regular user of pypsdier, this is the only class and methods you should know and use.

class simulation_interface.**SimulationInterface**

Bases: object

GenericSimulationLibrary is a package encapsulates a methodology and tools for reproducible simulations. The main idea is to use python and/or jupyter notebooks to provide a lightweight and for-dummies easy "Simulation as a Service". The framework puts emphasis on simplicity: for the client to install and use, for the programmer to distribute and update, and for everyone to store and reproduce results. The framework can be personalized and extended for a specific simulation need. Link: https://pypsdier.readthedocs.io/

download(*filename*)

Utility to download file, using colab.

export_xls(*filename*)

Creates an excel file and saves the plot data and simulation data. It helps providing a file format that final users might be more familiar with.

Parameters **filename** (*string*) – Name for the file.

load(*filename*)

Loads a simulation from a simulation file generated with the *save* method to restore the simulation.

Parameters **filename** (*string*) – Name for the simulation file.

new(*inputs*, *plot_options=None*)

Associates inputs and plot options to the simulation.

Parameters

- **inputs** (*dict*) – The inputs that will be used in the simulation. This can be completely personalized.
- **plot_options** (*dict, optional*) – The plot options, defaults to None

plot(*figsize=(12, 8)*, *plot_type='all'*, *filename=''*, *display=True*)

Conditionally imports the matplotlib library, and if possible, plots the experimental data given in plot_options, and the simulation data.

Parameters

- **plot_type** – ?
- **figsize** (*tuple, optional*) – Size of the figure
- **filename** (*str, optional*) – Filename to save the graph. If not provided, figure is not saved. Defaults to ''.

- **display** (*bool, optional*) – Boolean to show (True) or not show (False) the graph. Defaults to False

save (*filename*)
Saves the current state of the simulation, with all the provided information. The created file can be used with the *load* method to restore the simulation.

Parameters **filename** (*string*) – Name for the simulation file.

simulate (*sim_type*)
Function that encapsulates the numerical simulation. Stores the simulation internally.

Parameters **sim_typle** – Type of simulation required. Only two options: ode or pde.

Returns Dictionary with the results of the simulation

Return type dict

status ()
Prints out the detected configuration: environment, python and library versions.

1.9 Acknowledgements

We would like to thank:

- The python ecosystem: python and a large list of libraries (numpy, scipy, matplotlib, among many others).
- The Jupyter Notebooks: the game changer in using python interactive and documentable.
- Colab and MyBinder: for making easy to share and run jupyter notebooks.
- Read the docs: for making very easy to make and share good documentation.
- This project is based on the framework proposed by GenericSimulationLibrary.

1.10 Links and References

1.10.1 Generic Simulation Library Repositories

This library extends the GenericSimulationLibrary framework.

1.11 Documentation for developers

As a developer of pypsdier, these are the other functions that you might want to understand, re-document, extend and/or improve.

pypsdier is based on the ideas of GeneralSimulationLibrary, a package encapsulates a methodology and tools for reproducible simulations.

Index