Coding as a Service: Librería pypsdier, aprendizajes y metodología.

Sebastián Flores

Pycon Argentina 2020

Coding as a Service: librería pypsdier, aprendizajes y metodología.

Observaciones:

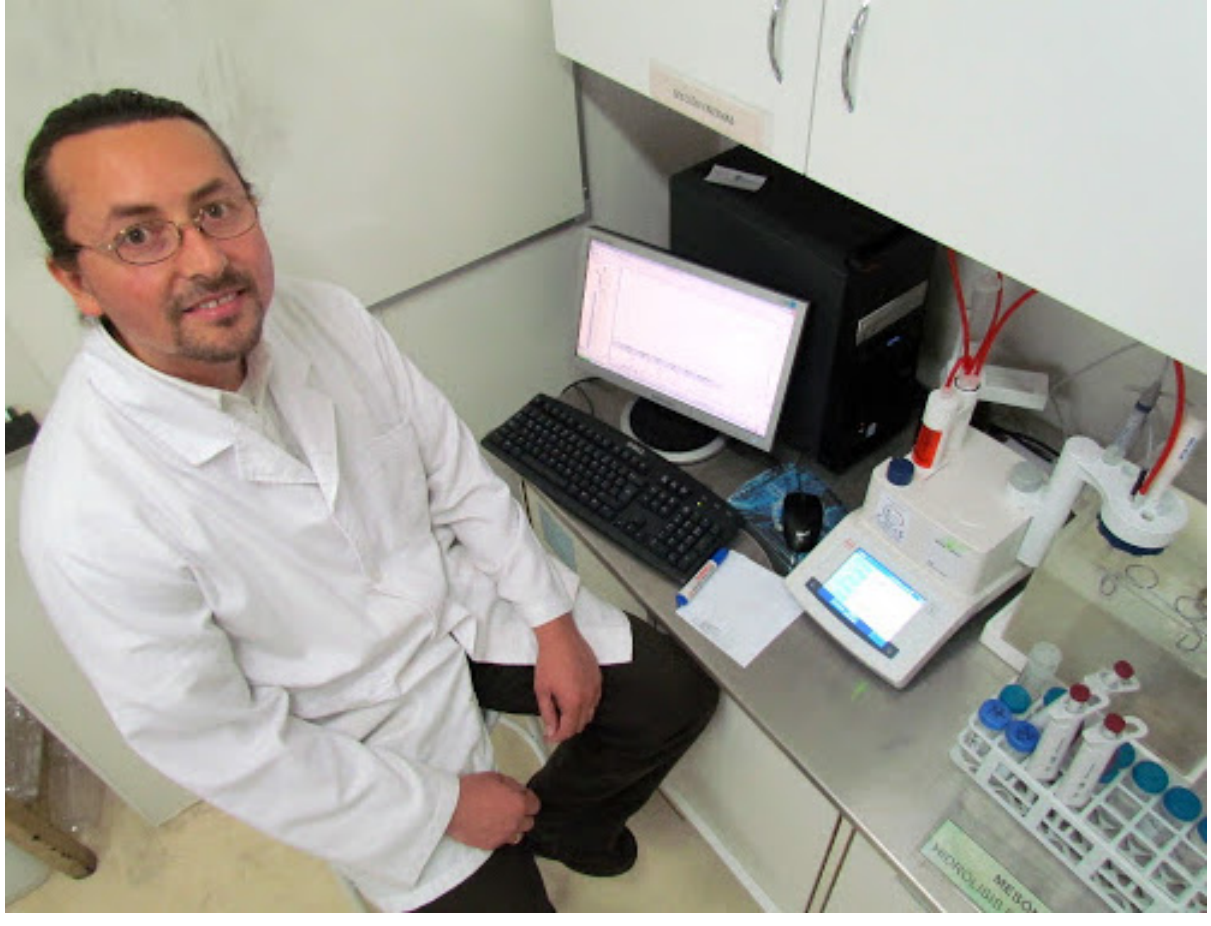
- @sebastiandres en twitter y github.
- Opiniones y críticas a título personal.
- No se necesita conocer de química, matemática o python avanzado.
- Presentación disponible en https://www.github.com/sebastiandres/charlas y realizada con jupyter+RISE.

Coding as a Service: librería pypsdier, aprendizajes y metodología.

Autores

Sebastián Flores

M.Sc. Mathematical and Computational Engineering
M.Sc. Mécanique Numérique
Ingeniero Civil Matemático
u-planner - Chile



Pedro Valencia

Ph.D Ingeniería Bioquímica

M.Sc. Ingeniería Bioquímica

Bioquímico

UTFSM - Chile

Coding as a Service: librería pypsdier, aprendizajes y metodología

Historia

- Origen:
    - Proyecto de colaboración en curso de matemáticas aplicadas.

Coding as a Service: librería pypsdier, aprendizajes y metodología

Historia

- Origen:
    - Proyecto de colaboración en curso de matemáticas aplicadas.
- Primeras implementaciones en MATLAB:
    - Problema: Cada nueva reacción química requiere una nueva implementación numérica.
    - Repetición de código y mantención tediosa.

Coding as a Service: librería pypsdier, aprendizajes y metodología

Historia

- Origen:
    - Proyecto de colaboración en curso de matemáticas aplicadas.
- Primeras implementaciones en MATLAB:
    - Problema: Cada nueva reacción química requiere una nueva implementación numérica.
    - Repetición de código y mantención tediosa.
- Desarrollo en Python:
    - Código genérico: librería pypsdier.
    - Interface simple e instalación por pypi.
    - Reacciones químicas definidas como funciones.
    - Número arbitrario de sustancias (sustratos o productos).
    - Tamaño de partículas definidas por una lista de radios y frecuencias.

Coding as a Service: librería pypsdier, aprendizajes y metodología

Historia

- Origen:
    - Proyecto de colaboración en curso de matemáticas aplicadas.
- Primeras implementaciones en MATLAB:
    - Problema: Cada nueva reacción química requiere una nueva implementación numérica.
    - Repetición de código y mantención tediosa.
- Desarrollo en Python:
    - Código genérico: librería pypsdier.
    - Interface simple e instalación por pypi.
    - Reacciones químicas definidas como funciones.
    - Número arbitrario de sustancias (sustratos o productos).
    - Tamaño de partículas definidas por una lista de radios y frecuencias.
- Impacto:
    - 9 publicaciones.
    - Código abierto y en constante evolución.

Coding as a Service: librería pypsdier, aprendizajes y metodología

Problema

- Pedro no conocía Python. Actualmente, nivel de usuario básico.
- Sebastián tiene (aún) conocimientos nulos en bioquímica.

Coding as a Service: librería pypsdier, aprendizajes y metodología

Problema

- Pedro no conocía Python. Actualmente, nivel de usuario básico.
- Sebastián tiene (aún) conocimientos nulos en bioquímica.

¿Cómo colaborar de manera eficiente?

Coding as a Service: librería pypsdier, aprendizajes y metodología

Problema

- Pedro no conocía Python. Actualmente, nivel de usuario básico.
- Sebastián tiene (aún) conocimientos nulos en bioquímica.

¿Cómo colaborar de manera eficiente?

El problema más general:

¿Cuál es la manera más sencilla de entregar "Coding As A Service"?

Ejemplo

¿Cuál es la manera más sencilla de entregar "Coding As A Service"?

Ejemplo

¿Cuál es la manera más sencilla de entregar "Coding As A Service"?

Tiene que ser tan fácil que puedas explicárselo a tu abuela.

Ejemplo

¿Cuál es la manera más sencilla de entregar "Coding As A Service"?

Tiene que ser tan fácil que puedas explicárselo a tu abuela.


MIJO ¿CUÁNTO FALTA
PARA EL PRÓXIMO CUMPLEAÑOS?
TENGO QUE TEJER

Coding as a Service: librería pypsdier, aprendizajes y metodología

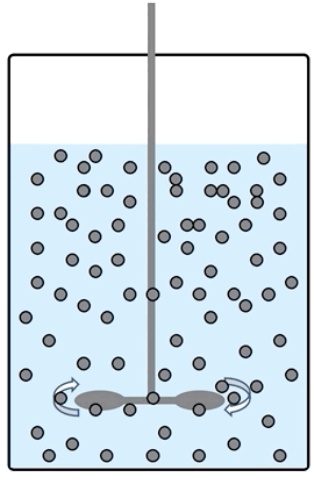
Solución para la abuela

Abue, vaya a este link y aprete play:

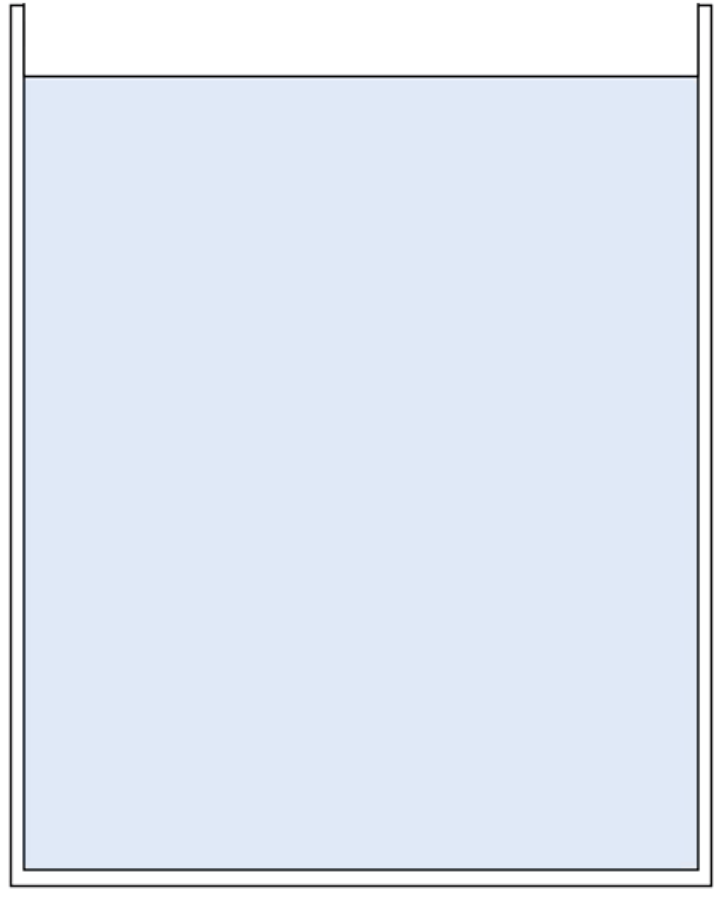
https://bit.ly/2IetMRn

Coding as a Service: librería pypsdier, aprendizajes y metodología.

Nuestro problema a resolver



Nuestro problema a resolver



Nuestro problema a resolver

El problema es obtener $S_b(t)$ y $S(t, r, R_i)$, para $t > 0$, $0 < r < R_i$.

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

Condición de borde en el centro de las partículas:

$$\frac{\partial S}{\partial r}(t, 0, R_i) = 0 \text{ para } t > 0 \tag{1}$$

Condiciones de borde en la superficie de la partícula:

$$S_b(t) = S(t, R_i, R_i) \text{ para } t > 0 \tag{2}$$

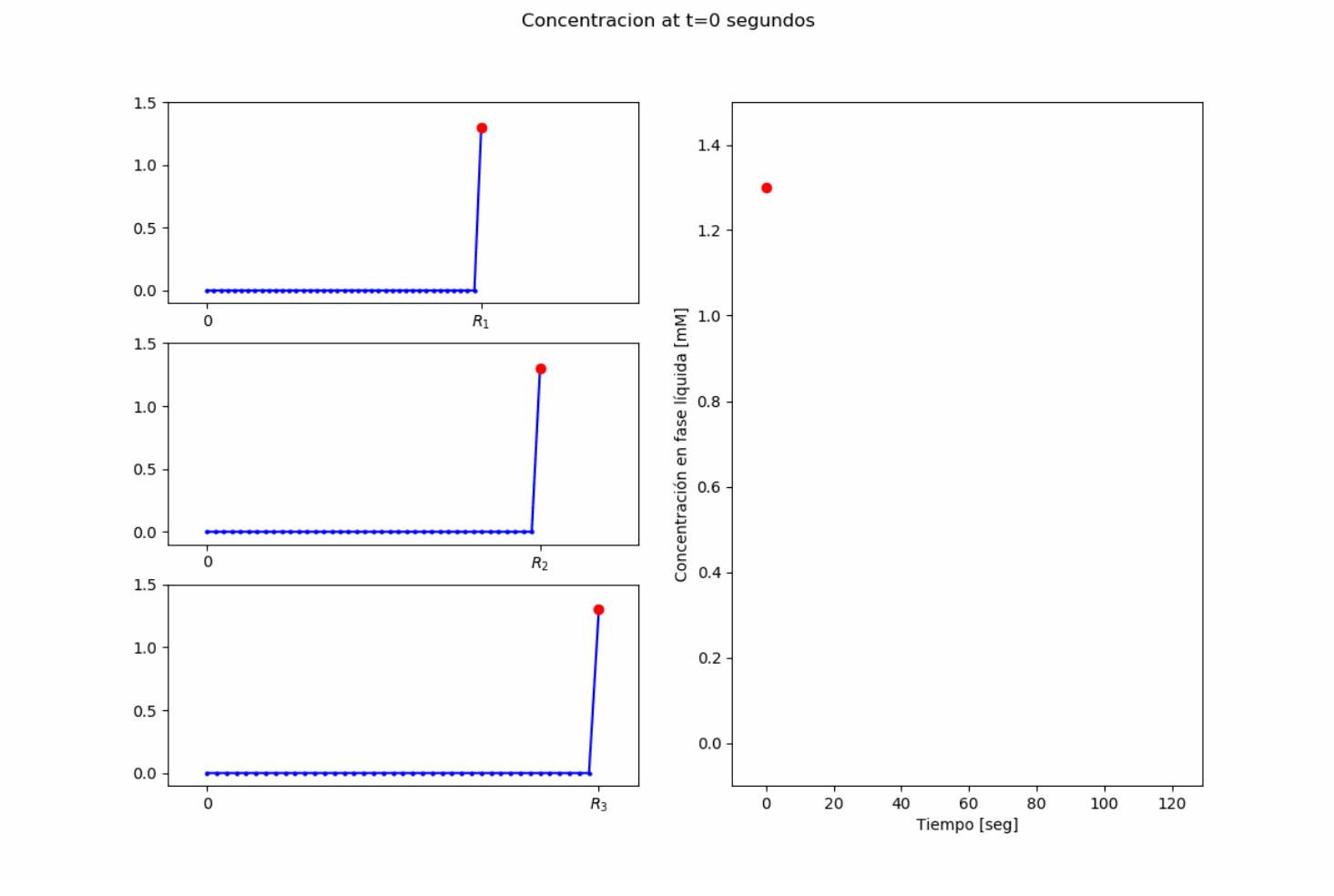
$$\frac{dS_b}{dt}(t) = -3 D_S \frac{V_c}{V_R E\left[R^3\right]} E\left[R^2 \frac{\partial S}{\partial r}\bigg|_{r=R} \right] \text{ para } t > 0 \tag{3}$$

Con la condición inicial

$$S_b(0) = S_0 \tag{4}$$

$$S(0, r, R_i) = 0 \text{ para } 0 \leq r < R_i \tag{5}$$

Nuestro problema a resolver



Coding as a Service: librería pypsdier, aprendizajes y metodología.

Los Desafíos

Desafíos técnicos:

- ¿Cómo discretizar las ecuaciones?
- ¿Cómo realizar la implementación numérica?
- ¿Cómo resolver para reacciones químicas distintas y un número arbitrario de sustancias y tamaños de partículas?

Coding as a Service: librería pypsdier, aprendizajes y metodología.

Los Desafíos

Desafíos técnicos:

- ¿Cómo discretizar las ecuaciones?
- ¿Cómo realizar la implementación numérica?
- ¿Cómo resolver para reacciones químicas distintas y un número arbitrario de sustancias y tamaños de partículas?

Esta es la parte tiene solución directa y conocida:

- Discretizar ecuaciones utilizando diferencias finitas para el tiempo (Δt) y espacio (ΔR_i).
- Asegurar estabilidad numérica dependiendo de los inputs.
- Implementación flexible con python + numpy + matplotlib + otros.

Coding as a Service: librería pypsdier, aprendizajes y metodología.

Los Desafíos

Desafíos organizacionales:

- ¿Cómo colaborar?
- ¿Cómo distribuir el código?
- ¿Cómo hacer fácil el uso?
- ¿Cómo hacer reproducible las simulaciones?
- ¿Cómo documentar el código?
- ¿Cómo almacenar los resultados de las simulaciones?

Coding as a Service: librería pypsdier, aprendizajes y metodología.

Los Desafíos

Desafíos organizacionales:

- ¿Cómo colaborar?
- ¿Cómo distribuir el código?
- ¿Cómo hacer fácil el uso?
- ¿Cómo hacer reproducible las simulaciones?
- ¿Cómo documentar el código?
- ¿Cómo almacenar los resultados de las simulaciones?

Esta parte no tiene una solución directa:

- No había mucha información al respecto.
- Cada equipo lo resuelve a su manera.
- Existen muchas herramientas que nos simplifican la vida.

Coding as a Service: librería pypsdier, aprendizajes y metodología

Framework: ¿Qué necesitábamos?

(1) Propocionar **instalación** y **versionamiento** de python, jupyter y librerías necesarias.

(2) Facilitar **desarrollo incremental** considerando devs y users.

(3) Permitir una buena **documentación** para desarrolladores y usuarios.

(4) Exponer una **interface simple** al usuario final, permitiendo ocultar una implementación numérica compleja y con una curva de aprendizaje apropiado.

(5) Permitir **reproducir**, **almacenar** y **compartir** resultados de simulación.

(6) Permitir el uso de recursos computacionales en la **nube**.

Coding as a Service: librería pypsdier, aprendizajes y metodología

Necesidades

(1) Propocionar **instalación** y **versionamiento** de python, jupyter y librerías necesarias.

(2) Facilitar **desarrollo incremental** considerando desarrolladores y usuarios.

Coding as a Service: librería pypsdier, aprendizajes y metodología

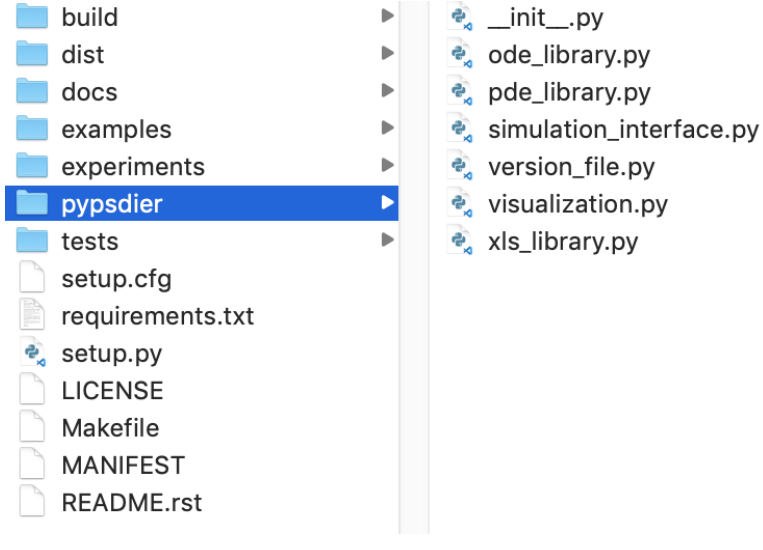
Necesidades

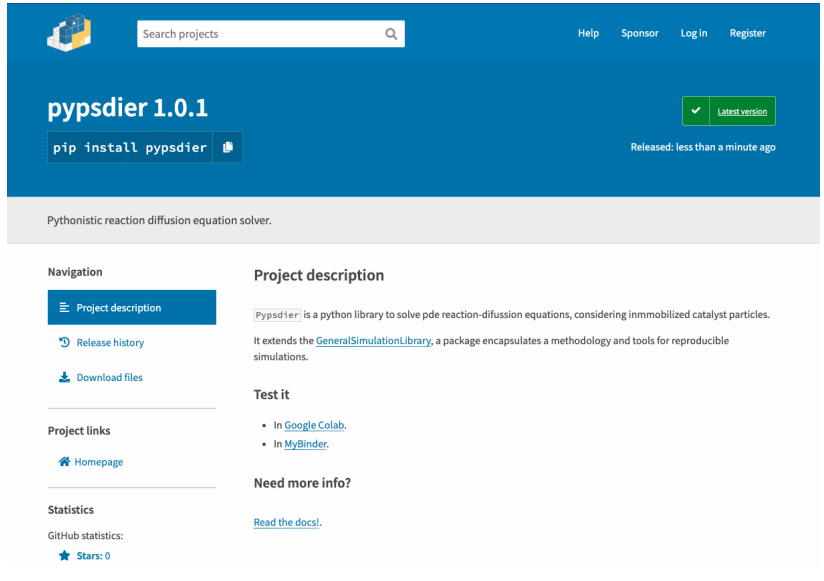
(1) Propocionar **instalación** y **versionamiento** de python, jupyter y librerías necesarias.

(2) Facilitar **desarrollo incremental** considerando desarrolladores y usuarios.

Solución

Estructura de proyecto de python, para instalar con pypi o desde github.





Coding as a Service: librería pypsdier, aprendizajes y metodología

Necesidades

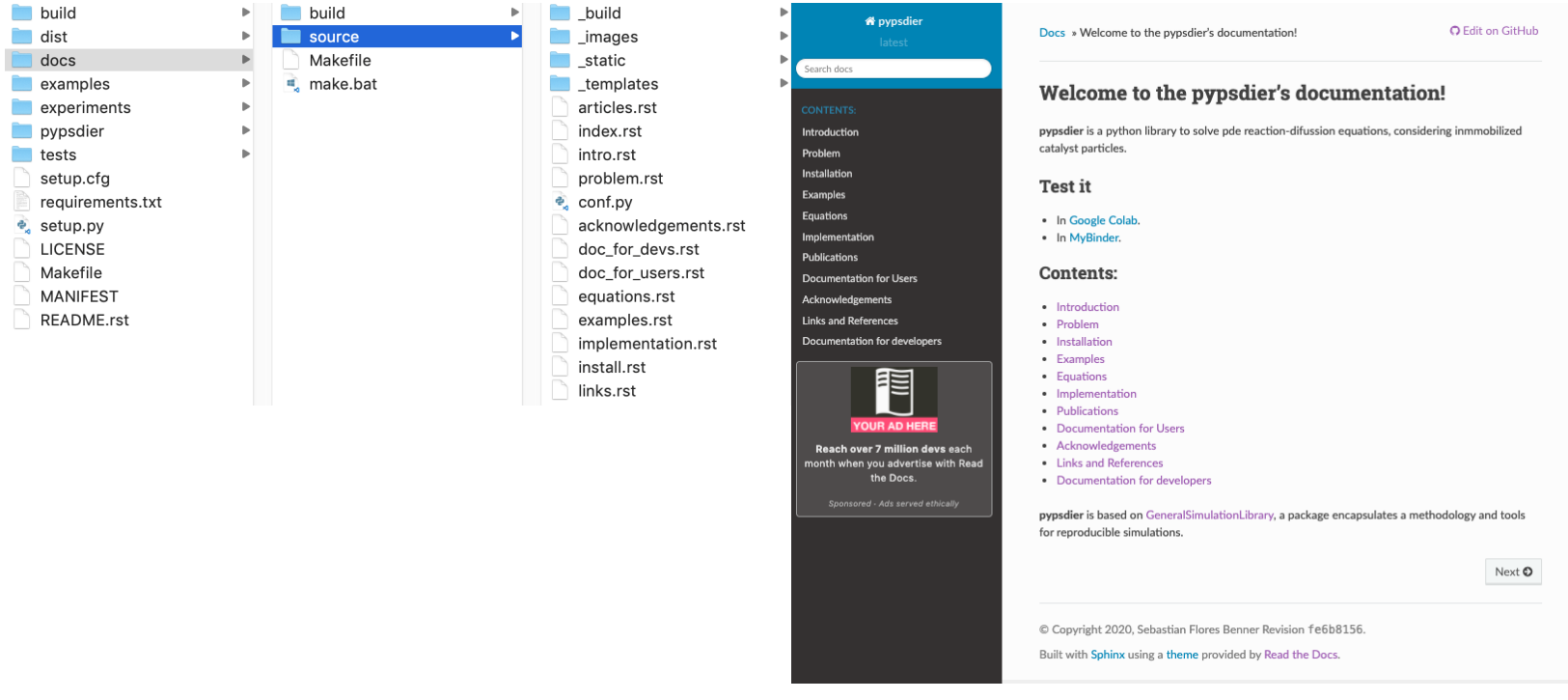
(3) Permitir una buena **documentación** para desarrolladores y usuarios.

Coding as a Service: librería pypsdier, aprendizajes y metodología

Necesidades

(3) Permitir una buena **documentación** para desarrolladores y usuarios.

Solución



Coding as a Service: librería pypsdier, aprendizajes y metodología

Necesidades

(4) Exponer una **interface simple** al usuario final, permitiendo ocultar una implementación numérica compleja y con una curva de aprendizaje apropiado.

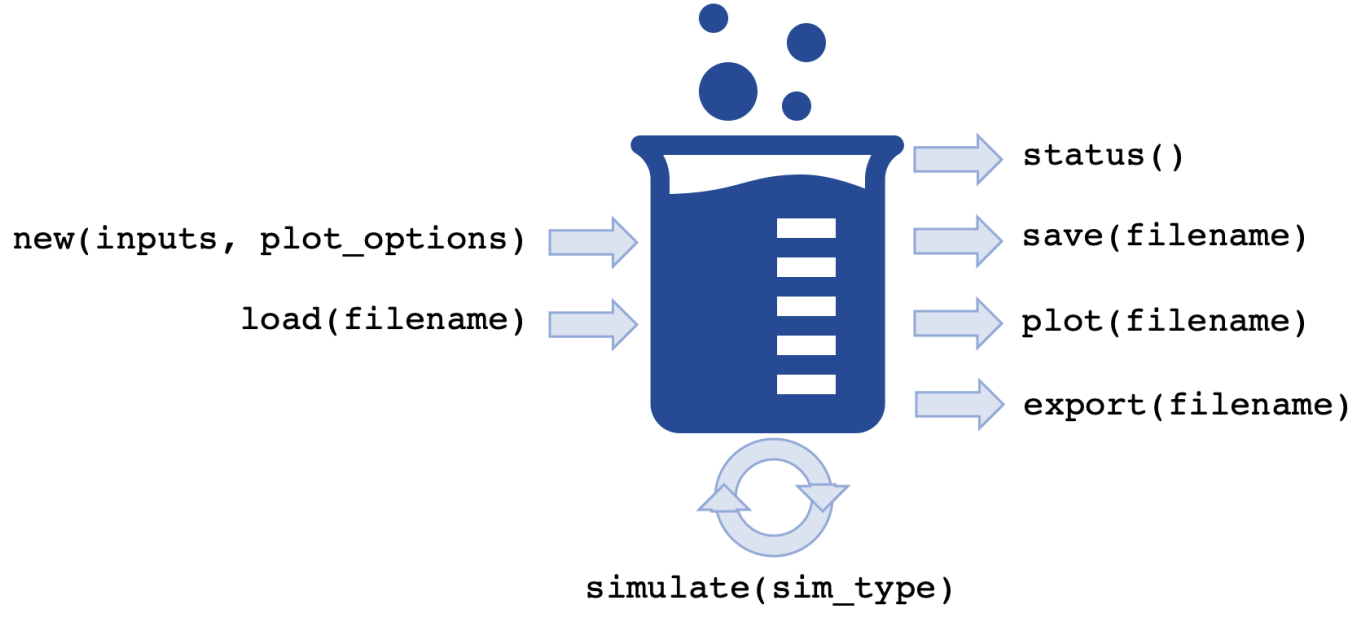
(5) Permitir **reproducir**, **almacenar** y **compartir** resultados de simulación.

Necesidades

(4) Exponer una **interface simple** al usuario final, permitiendo ocultar una implementación numérica compleja y con una curva de aprendizaje apropiado.

(5) Permitir **reproducir**, **almacenar** y **compartir** resultados de simulación.

Solución



Solución

(5) Permitir **reproducir**, **almacenar** y **compartir** resultados de simulación.

(6) Permitir el uso de recursos computacionales en la **nube**.

Solución

(5) Permitir **reproducir**, **almacenar** y **compartir** resultados de simulación.

(6) Permitir el uso de recursos computacionales en la **nube**.

Solución


2020_11_XX_pycon_ar_full_sim_experiment_PGA650R72
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing
[3]
inputs["InitialConcentrations"] = (10.0, 0., 0.) # [mM], initial concentration of substrates and products
inputs["EffectiveDiffusionCoefficients"] = (5.30E-10, 7.33E-10, 5.89E-10) # [m2/s], effective diffusion coefficient
inputs["CatalystParticleRadius"] = (75.7E-6,) # [m], list of possible catalyst particle radiuses
inputs["CatalystParticleRadiusFrequency"] = (1.0,) # [], list of corresponding frequencies of catalyst particle radiu
inputs["ReactionFunction"] = PenG_HydrolysisReaction # function defining the reaction
inputs["ReactionParameters"] = 41., 0.13, 821., 1.82, 48. #[1/s] and [mM]*4 # [1/s], [mM/s], parameters to be used
inputs["CatalystEnzymeConcentration"] = 0.140 # [mM] can be a float, int or a function returning float or int.
plot_options = {}
plot_options["t_exp"] = [0, 2, 4, 6, 8, 10, 20, 30, 40, 50, 65, 80, 100, 120, 150] # Time in mins
plot_options["PenG_exp"] = [10.00, 9.68, 9.30, 8.93, 8.85, 8.70, 8.00, 6.50, 5.21, 3.59, 2.58, 1.65, 1.01, 0.49, 0.25
# Simulate only if file not found
SI = pypsdier.SimulationInterface()
SI.new(inputs, plot_options)
¿Qué versiones de todas las librerías necesarias estamos usando? ¿Qué datos serán utilizados en la simulación?
SI.status()
Simulemos y guardemos el resultado:
[ ]
# Define filename for storing the simulation and plots
filename = "PGA650R72.rde"
SI.simulate("ode")
SI.simulate("pde")
SI.save(filename)

Ejemplo de pypsdier

Primero que nada, instalaremos la librería:

In [63]:
```
#!pip install pypsdier --upgrade
!pip install git+https://github.com/sebastiandres/pypsdier.git --upgrade
```

```
Collecting git+https://github.com/sebastiandres/pypsdier.git
  Cloning https://github.com/sebastiandres/pypsdier.git to /private/var/fo
lders/68/v1ds0ld152q_wk4sbcns3ckw0000gn/T/pip-req-build-mm52_b8o
  Running command git clone -q https://github.com/sebastiandres/pypsdier.g
it /private/var/folders/68/v1ds0ld152q_wk4sbcns3ckw0000gn/T/pip-req-build-
mm52_b8o
Building wheels for collected packages: pypsdier
  Building wheel for pypsdier (setup.py) ... done
  Created wheel for pypsdier: filename=pypsdier-1.1.1-cp37-none-any.whl si
ze=14659 sha256=34556c2c7b1b3be6222355f403a45817e8b5a0154f44037f940a492371
1fe71d
  Stored in directory: /private/var/folders/68/v1ds0ld152q_wk4sbcns3ckw000
0gn/T/pip-ephem-wheel-cache-csxbayor/wheels/57/71/0f/2f6f43ab73e6371b622a1
e181130163a445c394d19b74a80d1
Successfully built pypsdier
Installing collected packages: pypsdier
  Found existing installation: pypsdier 1.1.1
    Uninstalling pypsdier-1.1.1:
      Successfully uninstalled pypsdier-1.1.1
Successfully installed pypsdier-1.1.1
```

Ejemplo

Consideremos un ejemplo de una reacción simple. Definamos los inputs de la simulación:

In [64]:

```
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"] = 30. # [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] # []
inputs["ReactionFunction"] = MichaelisMenten # function
inputs["ReactionParameters"] = (41 , 0.13)   # [1/s], [mM/s], parameters
inputs["CatalystEnzymeConcentration"] = 0.35 # [mM]
```

Ejemplo

Definamos ahora opciones de gráficos para la simulación:

In [65]:

```
plot_options = {}
plot_options["label_x"] = "Tiempo de reacción [s]"
plot_options["label_y"] = "Concentración [mM]"
plot_options["title"] = "Simulación de Michaelis Menten para la PyconAr"
plot_options["ode_kwargs"] = {'label':'Enzima Libre', 'color':'blue', 'marker':'', 'markersi
plot_options["pde_kwargs"] = {'label':'Enzima Inmobilizada', 'color':'blue', 'marker':'', 'm
plot_options["data_kwargs"] = {'label':'Experimento', 'color':'green', 'marker':'s', 'marker
plot_options["data_x"] = [0.0, 30, 60, 90, 120]
plot_options["data_y"] = [1.3, 0.65, 0.25, 0.10, 0.0]
```

Ejemplo pypsdier

Todo está listo para realizar una simulación (muy sencilla):

In [66]:
```
import pypsdier
SIM1 = pypsdier.SimulationInterface()
SIM1.new(inputs, plot_options)
```

In [67]:
```
SIM1.simulate("ode")
```

```
ODE: Solving without diffusional restriction (simplified problem). No poro
us particles are present. The substance and the enzyme are assumed to be d
iluted on the bulk phase
Simulation completed.
```

In [68]:
```
SIM1.simulate("pde")
```

```
PDE: Solving the complete reaction-diffusion problem, considering a single
substance and 3 particle radii.
Simulated 030 secs out of  30 secs (Remaining time < 1 mins)
```

In [69]:
```
SIM1.save("pycon_saving_example.rde")
```

```
Saving simulation into file at /Users/sebastiandres/Desktop/Personales/git
hub_repos/charlas/2020_11_XX_pycon_ar_pypsdier/pycon_saving_example.rde
```

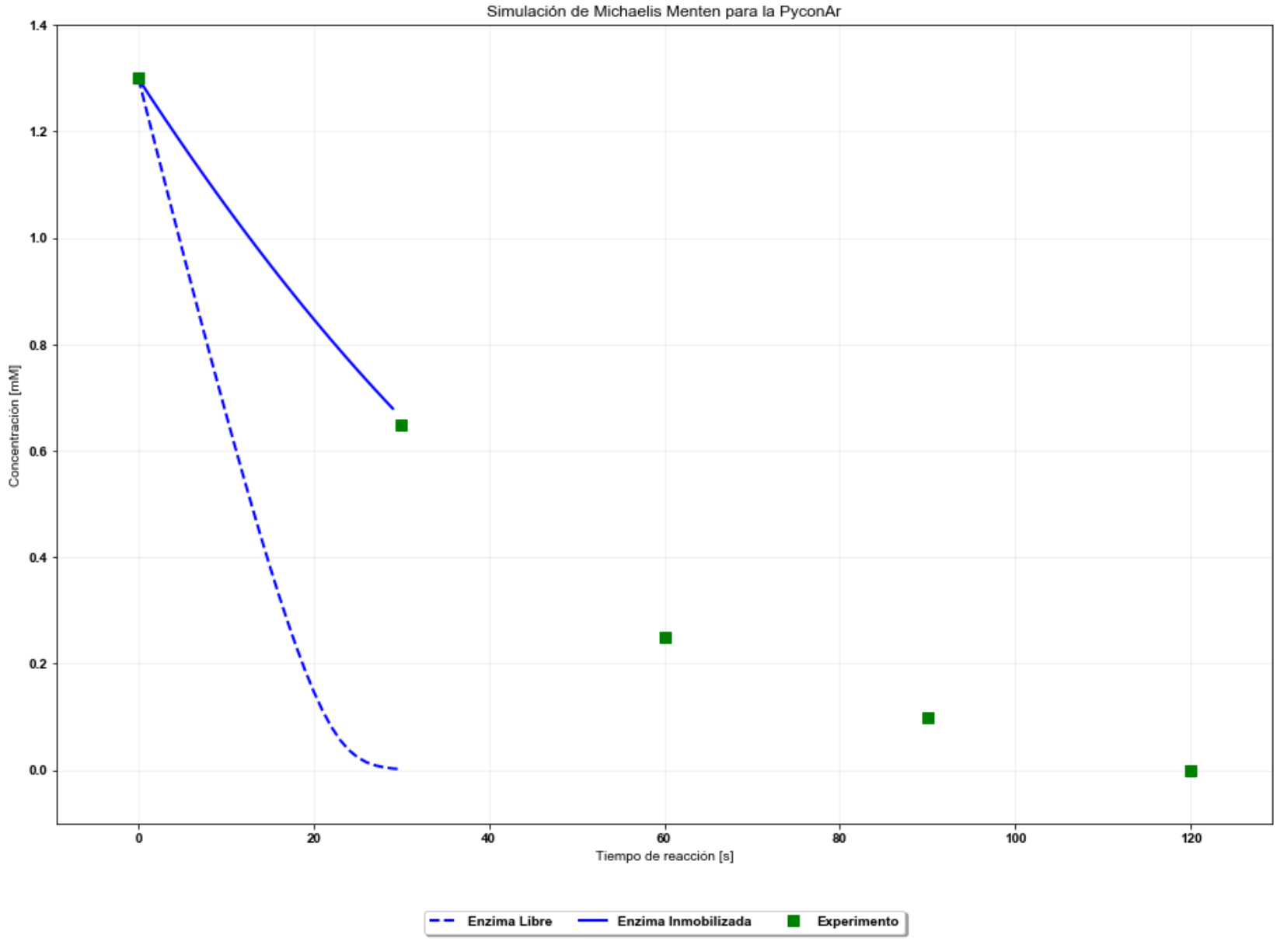
Ejemplo pypsdier

Podemos definir funciones específicas para exportar en formato excel o graficar

In [70]:
```
SIM1.export_xls("pycon_excel_example.xls")
```

```
Saving simulation as xls file at /Users/sebastiandres/Desktop/Personales/g
ithub_repos/charlas/2020_11_XX_pycon_ar_pypsdier/pycon_excel_example.xls
```

In [71]:
```
SIM1.plot(figsize=(16,12))
```


Simulación de Michaelis Menten para la PyconAr
Concentración [mM]
Tiempo de reacción [s]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
20
40
60
80
100
120
Enzima Libre
Enzima Inmobilizada
Experimento

Ejemplo pypsdier

¿Y la reproducibilidad?

Una simulación realizada almacena todas las características con las cuales se generó.

In [72]:
```
import pypsdier
SIM2 = pypsdier.SimulationInterface()
SIM2.load("pycon_colab_example.rde") # load instead of new
```

```
Loaded a simulation from /Users/sebastiandres/Desktop/Personales/github_re
pos/charlas/2020_11_XX_pycon_ar_pypsdier/pycon_colab_example.rde
```

In [73]:
```
SIM2.status()
```

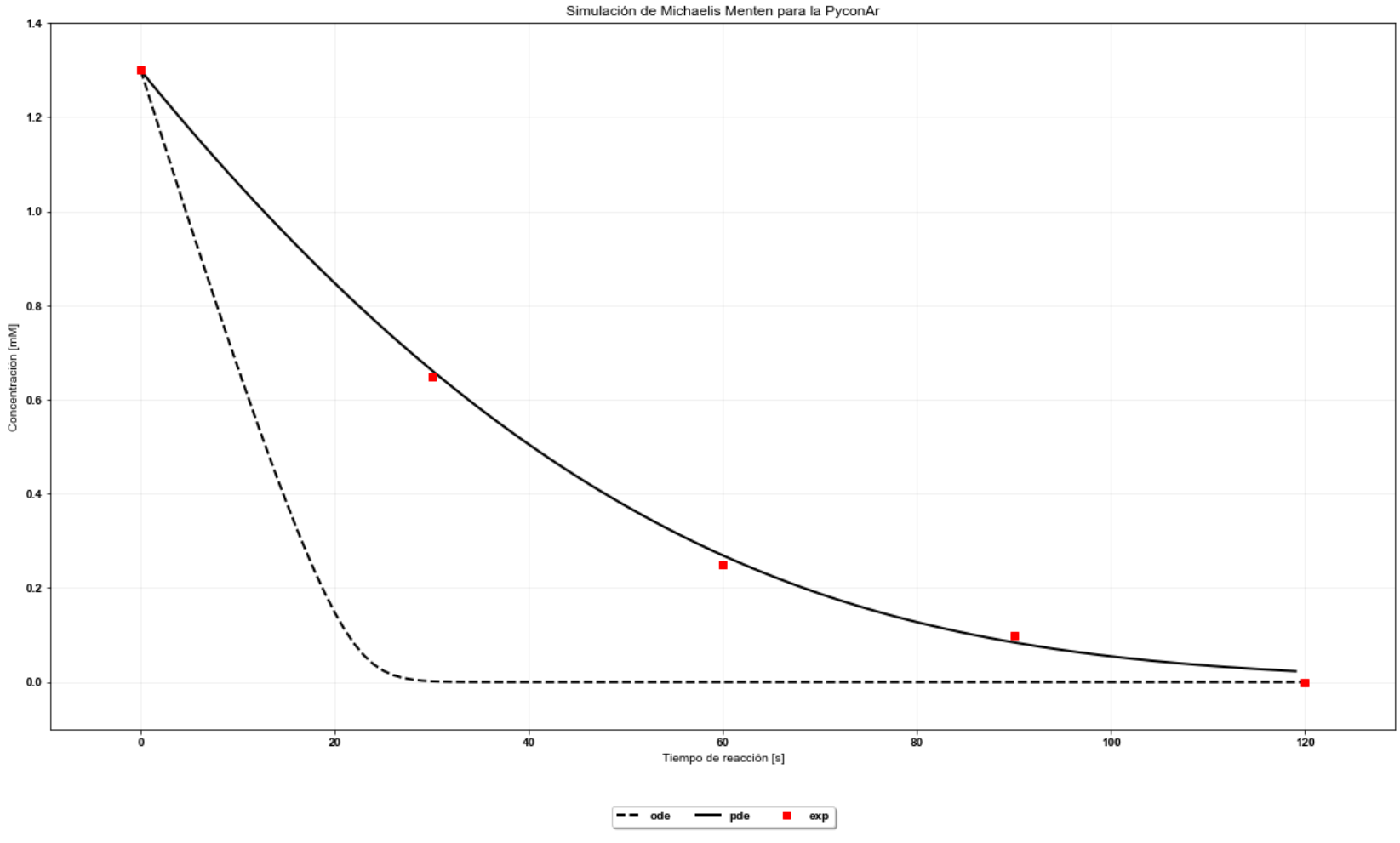
```
System configuration:
    environment: google_colab
    python version: 3.6.9
    GenericSimulationLibrary version: 1.1.0
    numpy version: 1.18.5
    scipy version: 1.4.1
    xlwt version: 1.3.0
    matplotlib version: 3.2.2
    dill version: 0.3.2
Inputs:
    SimulationTime: 120.0
    SavingTimeStep: 1.0
    CatalystVolume: 0.5
```

```
    BulkVolume: 100.0
    Names: ('Substrat',)
    InitialConcentrations: (1.3,)
    EffectiveDiffusionCoefficients: (5.3e-10,)
    CatalystParticleRadius: [4e-05, 6e-05, 8e-05]
    CatalystParticleRadiusFrequency: [0.3, 0.5, 0.2]
    ReactionFunction: <function MichaelisMenten at 0x134d50dd0>
    ReactionParameters: (41, 0.13)
    CatalystEnzymeConcentration: 0.35
Plot Options:
    title: Simulación de Michaelis Menten para la PyconAr
    label_x: Tiempo de reacción [s]
    label_y: Concentración [mM]
    ode_kwargs: {'label': 'ode', 'color': 'black', 'marker': '', 'markersi
ze': 6, 'linestyle': 'dashed', 'linewidth': 2}
    pde_kwargs: {'label': 'pde', 'color': 'black', 'marker': '', 'markersi
ze': 6, 'linestyle': 'solid', 'linewidth': 2}
    data_kwargs: {'label': 'exp', 'color': 'red', 'marker': 's', 'markersi
ze': 6, 'linestyle': 'none', 'linewidth': 2}
    data_x: [0.0, 30, 60, 90, 120]
    data_y: [1.3, 0.65, 0.25, 0.1, 0.0]
Simulations:
    ODE: yes
    PDE: yes
```

Ejemplo pypsdier

Podemos cargar una simulación realizada previamente y analizarla (sin necesidad de volver a realizar la simulación).

In [74]:
```
SIM2.plot(figsize=(20,12))
```


Simulación de Michaelis Menten para la PyconAr
Concentración [mM]
Tiempo de reacción [s]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
20
40
60
80
100
120
ode
pde
exp

Framework Propuesto

La librería pypsdier es específica al problema de reacción-difusión para reactores de enzima inmobilizada, con flexibilidad para diferentes modelos de reacción y condiciones de operación.

Sin embargo, se propone un framework de trabajo con las funcionalidades mínimas: **GenericSimulationLibrary**

Disponible en https://github.com/sebastiandres/GenericSimulationLibrary:

- En `simulation_interface.py` actualizar los métodos: new, load, simulate, save, plot, export.
- En `docs/source/*.rst` actualizar la documentación.

Conclusión

Todo lo anterior no es muy distinto a lo que se hace en una librería "formal".

Conclusión

Todo lo anterior no es muy distinto a lo que se hace en una librería "formal".

Aprendizaje:

Puedes usar las mismas herramientas e ideas, aunque tu equipo y proyecto sea más pequeño e informal, porque simplifican y mejoran la colaboración.

Conclusión

Todo lo anterior no es muy distinto a lo que se hace en una librería "formal".

Aprendizaje:

Puedes usar las mismas herramientas e ideas, aunque tu equipo y proyecto sea más pequeño e informal, porque simplifican y mejoran la colaboración.

- git + github/bitbucket.
- readthedocs
- pypi
- google colab/mybinder
- Interface OO

Coding as a Service: Librería pypsdier, aprendizajes y metodología.

Sebastián Flores

Pycon Argentina 2020

Encuesta:

https://bit.ly/354jhrQ