Parareal method and data assimilation for PDEs with Feel++

Presentation

AYDOGDU Melissa, LECOURTIER Frédérique

Strasbourg University

August 25, 2022

Centre de modélisation et de simulation de Strasbourg

Context of the internship

Cemosis

- Developing competences and projects in the energy sector of buildings.
- Accelerate simulations: parallel computing
- Complete models that may be coarse : Data assimilation

→ created in January 2013

→ hosted by the Institute of Advanced Mathematical Research (IRMA)

→ same thematics as CSMI

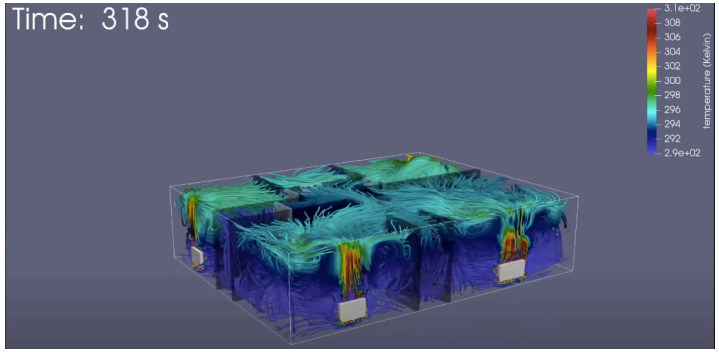


Figure 1: Visualization of the simulation made with the Feel++ toolboxes .

Objectives

- to read the following article about the Heat equation (Chapter 11) :
  Numerical Approximation of Partial Differential Equations.
  Alfio Quarteroni and Alberto Valli. (see)

- to set up a project on Github :
  - Organisation of the repository (Python library, cmake in C++ ...)
  - Create issues to see the progress of the project
  - Set up the CI : build,test,documentation

Antora

SPHINX Python Documentation Generator

Doxygen

Github repository : `https://github.com/master-csmi/2022-m1-lorenz`

cemosis
Centre de modélisation et de simulation de Strasbourg

Lorenz system

The system :

$$\begin{cases} \dfrac{dx(t)}{dt} = \sigma(y(t) - x(t)) \\ \dfrac{dy(t)}{dt} = x(t)(r - z(t)) - y(t) \\ \dfrac{dz(t)}{dt} = x(t)y(t) - bz(t) \end{cases}$$

where

- $\sigma > 0$ relates to the Prandtl number
- $r > 0$ relates to the Rayleigh number
- $b > 0$ is a geometric factor

to investigate atmospheric convection

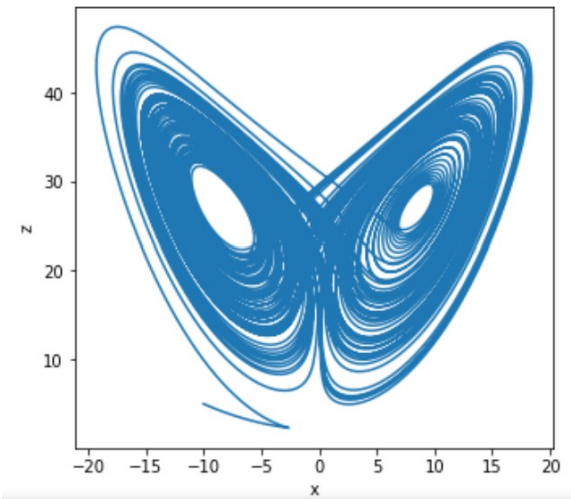


Figure 2: Butterfly effect

Harmonic oscillator

Equation :

$$\frac{d^2x}{dt^2} + \omega_0^2 x = 0$$

with

- ω_0 a pulsation
- x_0 : the amplitude of the oscillations
- ϕ_0 : the phase at the origin

We have the period $\quad Pe = \frac{2\pi}{\omega_0}$

Exact solution :

$$x(t) = x_0 \cos(\omega_0 t + \phi_0).$$

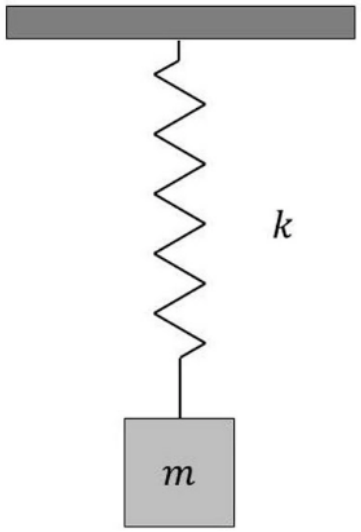


Figure 3: Spring/mass system

Heat equation

The heat equation with convective effects can be written as:

$$\rho C_p((\frac{\partial T}{\partial t}) + u \cdot \nabla T) - \nabla \cdot (k \nabla T) = Q$$

and it must be completed with boundary conditions and initial conditions.

Notation	Quantity	Unit
ρ	density	$Kg.m^{-3}$
C_p	Specific heat	$J/KgC = J/KgK$
k	Conductivity	$W/mC = W/mK$
u	Fluid velocity	$m.s^{-1}$
T	Temperature	K or C

Table 1: Parameters for the heat equation

Application of the parareal method to PDE simulation in Feel++

What is the parareal method ?

- parallel-in-time integration method, introduced in 2001 by Lions, Maday and Turinici [13]
- computes the numerical solution for multiple time steps in parallel
- categorized as a parallel-across-the-steps method

Objectives for parareal method

- Implement the parareal method in C++ and :
  - Test the method (with oscillator)
  - Check convergence and stability results
  - Check speed-up and efficiency
- Implement the resolution of the heat equation in C++ with Feel++
  $\Rightarrow$ Implement the resolution of the Laplace equation in C++ with Feel++
- Use the previous implementation of the heat equation with the parareal method
- Check the convergences/accuracies of the method

Theory

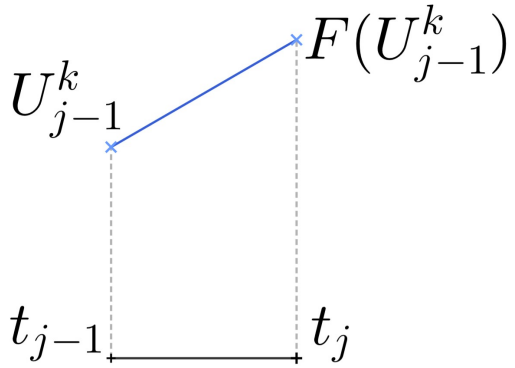
Generalities of the parareal method

Time decomposition :

- $[t_0, T] = [t_0, t_1] \cup \cdots \cup [t_{P-1}, t_P]$ with $t_P = T$ and $P =$ number of processes

Notations :

- F : high accuracy integrator, Δt_F : time step
  G : low accuracy integrator, Δt_G : coarse time step
- $U_j^k, j \in \{0, \ldots, P\}$: initial point at time t_j and at iteration k.
- $F(U_{j-1}^k), j \in \{1, \ldots, P\}$: value of the fine integrator at t_j starting by U_{j-1}^k (resp. $G(U_{j-1}^k)$)
  F and G are Runge Kutta of order 4



Explanation of the parareal method I

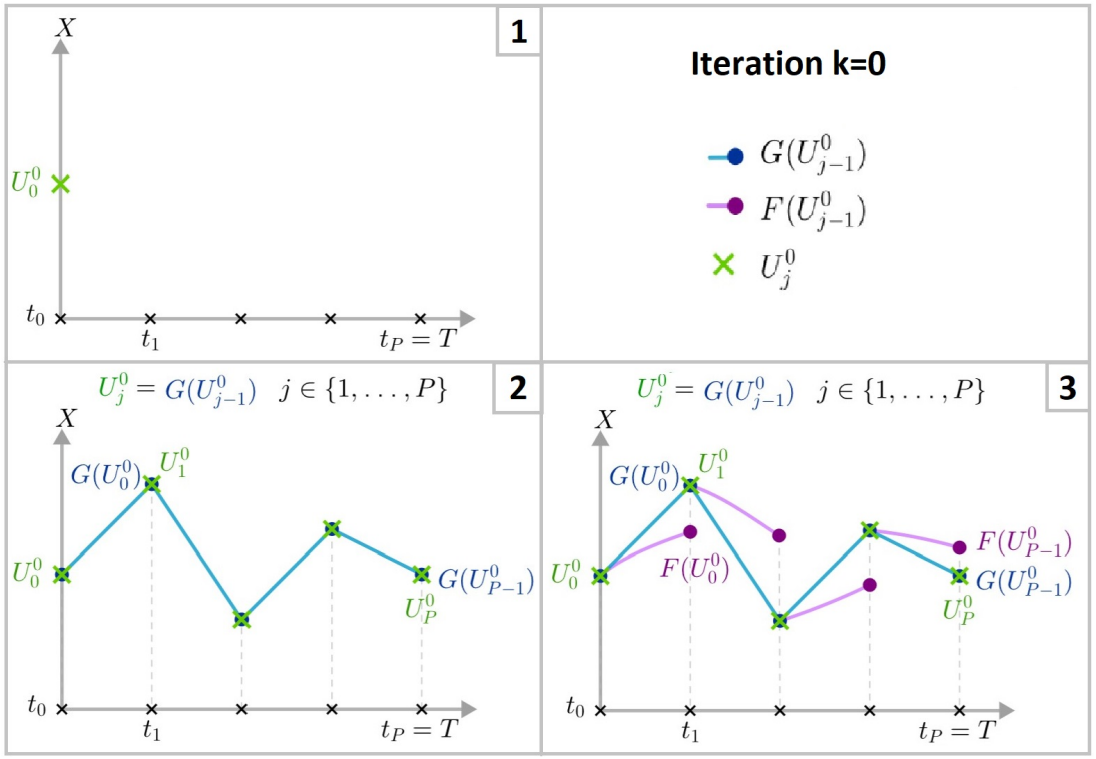


Figure 4: Explanation of the parareal method at iteration k $= 0$ (step 1 to 3)

Explanation of the parareal method II

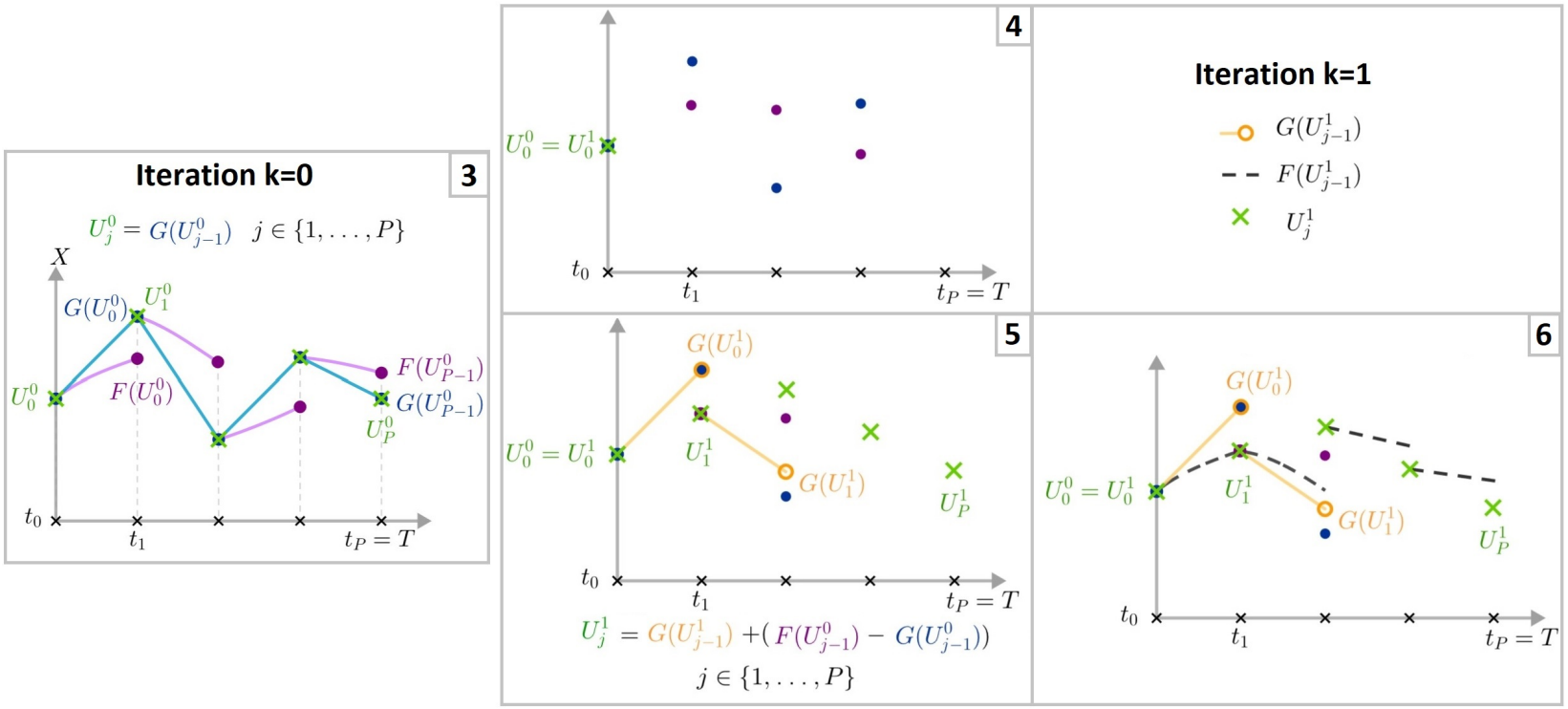


Figure 5: Explanation of the parareal method at iteration k $= 1$ (step 4 to 6)

We have at iteration k : $\quad U_j^k = G(U_{j-1}^k) + (F(U_{j-1}^{k-1}) - G(U_{j-1}^{k-1}))$

Algorithm I

input : t_0 : initial time ; T : final time ; U_0^0 : initial point ;
F : fine integrator ; G : coarse integrator ;
Δt_F : fine time step ; Δt_G : coarse time step ;
P : number of processes
output: sol : solution with parareal method

```
k←0;
init_pts←[U_0^0];
coarse←[];
fine←[];
for j ← 1 to P do
    coarse[j-1]←G(init_points[j-1])[-1];
    init_pts[j]←coarse[j];
end
U_j^k ← send init_pts[j] to each proc j;
do in parallel
    tab_F← F(U_j^k) // fine solution on each sub-interval
end
fine[j]← recv tab_F[-1] from each proc j;
```

Algorithm 1: initialization

```
initialization();
while not converge do
    for j ← 1 to P do
        val_G←G(init_points[j-1])[-1];
        init_pts[j]←val_G+fin[j-1]-coarse[j-1];
        coarse[j-1]←val_G;
    end
    U_j^k ← send init_pts[j] to each proc j;
    do in parallel
        tab_F← F(U_j^k) // fine solution on each sub-interval
    end
    fine[j]← recv tab_F[-1] from each proc j;
end
// Get the solution between y_0 and T
// sol(t_0, T) = [sol(t_0, t_1), ..., sol(t_{P-1}, t_P)]
sol ← send the final point from the last process;
return sol
```

Algorithm 2: parareal

Figure 6: Algorithm for the parareal method

Order of the parareal method

The parareal method is of order k if there is a constant c_k such that :

$$\forall j \in \{0, \ldots, P-1\} \qquad \mathcal{E}(j,k) \leq c_k (\Delta t_G)^k$$

with

$$\mathcal{E}(j,k) = |U_j^k - U_{ex}(t_j)| + \max_{t \in [t_j, t_{j+1}]} |U_k(t) - U_{ex}(t)|$$

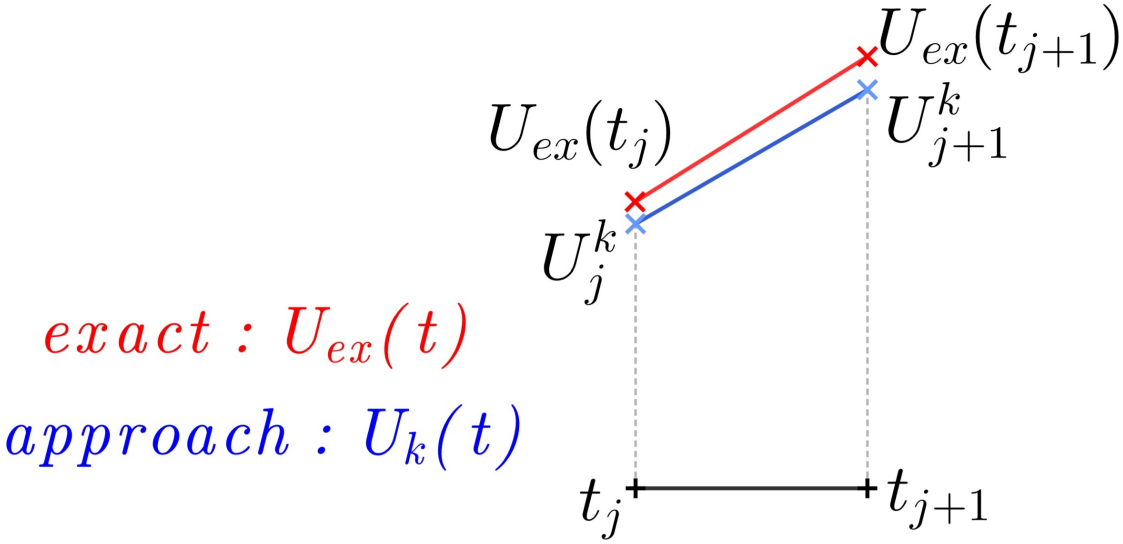


Figure 7: Explanation of the order property

Application in Python and C++

Example with the Lorenz system

$$\sigma = 10, \quad \mathsf{b} = \frac{8}{3}, \quad \mathsf{r} = 28, \quad \mathsf{X}_0 = (5,5,5), \quad \mathsf{t}_0 = 0, \quad \mathsf{T} = 20$$
$$\mathsf{P} = 4, \quad \Delta \mathsf{t}_\mathsf{G} = 0.1, \quad \Delta \mathsf{t}_\mathsf{F} = 0.01$$

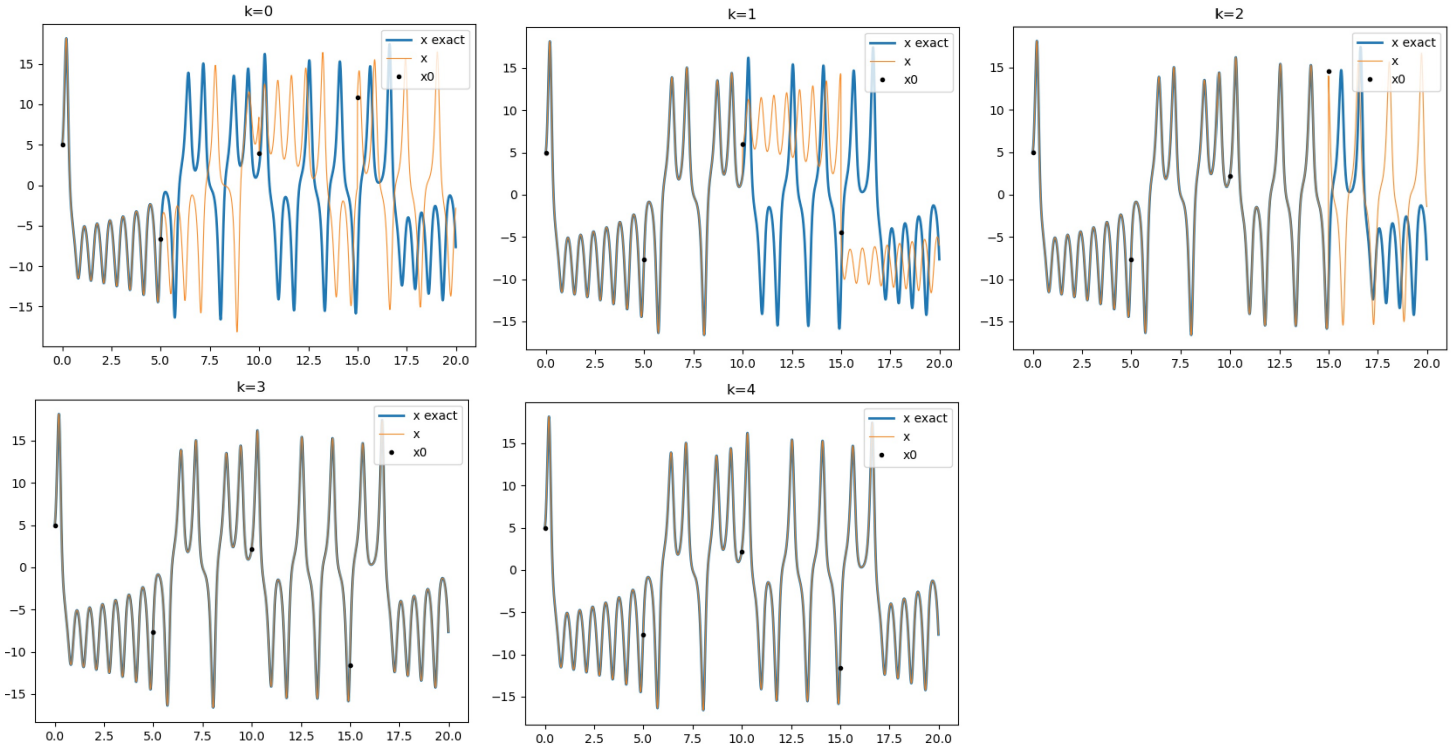


Figure 8: Solution of the Lorenz system with C++ at each iteration

Speed-up for the Lorenz system

Python

$t_0 = 0, \ \Delta t_G = 0.1, \ \Delta t_F = 0.01$

Δt	Seq (RK4)	1 proc	2 proc	3 proc	4 proc
F : 0.00125 G : 0.0125	1m19	1m42	39s	32s	29s

Table 2: Executuion time with Python (T=200)

C++

$t_0 = 0, \ \Delta t_G = 0.01, \ \Delta t_F = 0.001$

Python : T = 200
1m24s

C++ : T = 200000
1m15s

Table 3: Difference of the execution times between Python and C++ with P = 4

$\Rightarrow$

Parameters : $T, \Delta t_F, pb$

Lorenz $\Rightarrow$ d * 3 equations

·	Seq	1 proc	2 proc	3 proc	4 proc
T : 200 d : 10000	53s	1m46s	1m29s	1m25s	1m27s

Table 4: Execution time with C++

cemosis
Centre de modélisation et de simulation de Strasbourg

Incorrect results for the Lorenz System

$$\sigma = 10,\ \mathsf{b} = \frac{8}{3},\ \mathsf{r} = 28,\ \mathsf{X}_0 = (5,5,5),\ \mathsf{t}_0 = 0,\ \Delta\mathsf{t}_G = 0.1,\ \Delta\mathsf{t}_F = 0.01$$

·	Seq	1 proc	2 proc	3 proc	4 proc
T : 500 d : 1000	13s	27s	22s	23s	22s

Table 5: Exemple of incorrect results with the Lorenz system.

✓ 1-2 procs : $(x(T), y(T), z(T)) = (-3.33169, -4.35437, 18.3289)$

✗ 3 procs : $(x(T), y(T), z(T)) = (-0.935968, 0.240612, 21.186)$

4 procs : $(x(T), y(T), z(T)) = (8.18609, 1.32531, 33.8032)$

cemosis
Centre de modélisation et de simulation de Strasbourg

Example for the Harmonic oscillator

$$\mathsf{P} = 3, \quad \mathsf{x}(0) = 0, \quad \mathsf{v}(0) = 1, \quad \omega_0 = 5, \quad \mathsf{x}_0 = \frac{-1}{5}, \quad \phi_0 = \frac{\pi}{2}$$

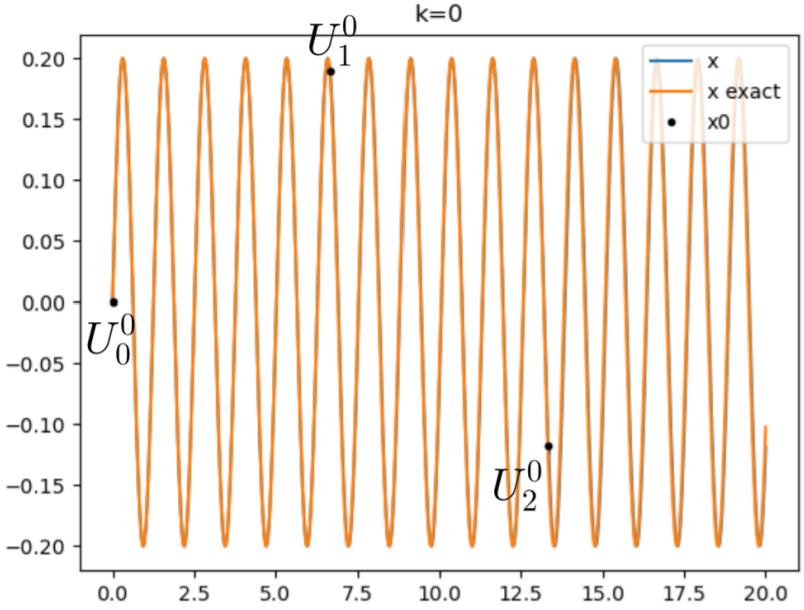


Figure 9: Parareal method applied to the harmonic oscillator (with C++)

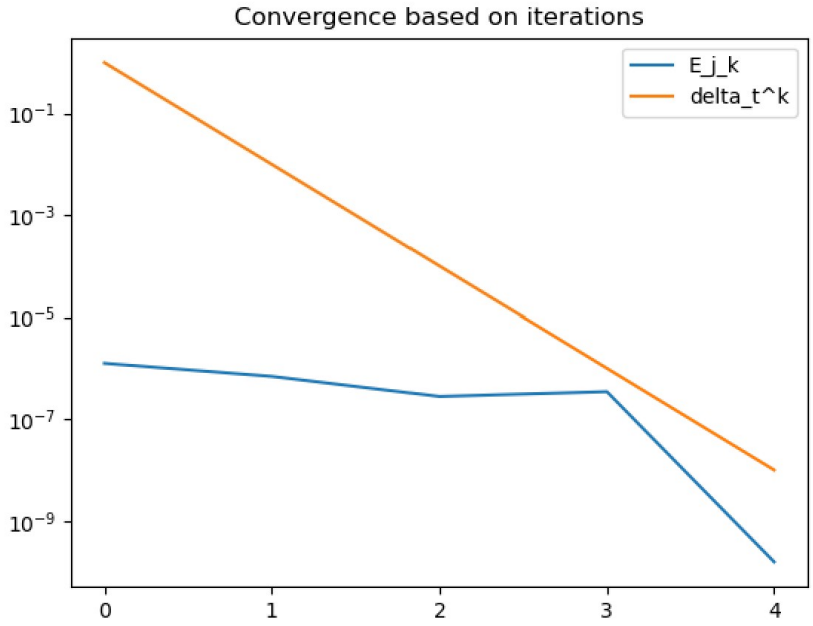


Figure 10: Convergence property of parareal method with the harmonic oscillator (with Python)

cemosis
Centre de modélisation et de simulation de Strasbourg

Efficiency for the Harmonic oscillator

Efficiency :

Final time : T Number of processes : P	~	Final time : 2*T Number of processes : 2*P

Figure 11: Explanation of efficiency.

	First case	Second case
Example 1 $T = 10000$	Seq : 17s 1 proc : 36s	2 proc : 30s
Example 2 $T = 20000$	2 proc : 58s	4 proc : 1m

Table 6 : Two tests of the efficiency of the parareal method in C++.

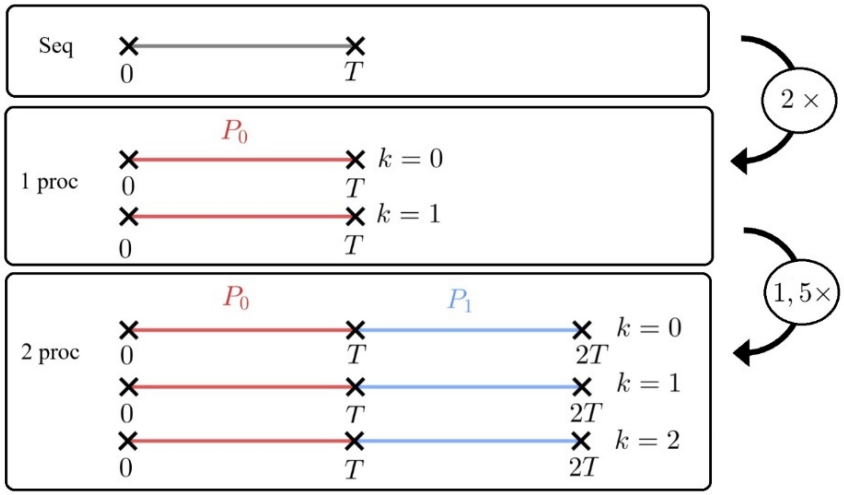


Figure11: Explanation - Sequential vs Parareal with 2 processes

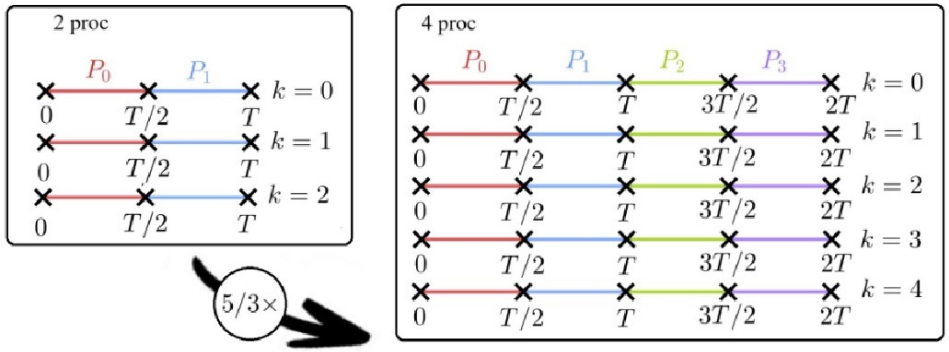


Figure12: Explanation - 2 proc vs 4 proc

+ sequential part + operations + communications

Solving PDEs with Feel++

Heat conduction equation

Goal : apply the parareal method to the resolution of the heat equation using Feel++ (solver developed by Cemosis that uses the finite element method)

The problem :
We study the heat equation without the convection term and with $\rho = \mathsf{C_p} = \mathsf{k} = 1$

$$\begin{cases} \dfrac{\partial \mathsf{u}}{\partial \mathsf{t}} - \Delta \mathsf{u} = \mathsf{f} & (\mathsf{t}_0, \mathsf{T}) \times \Omega \\ \mathsf{u} = 0 & (\mathsf{t}_0, \mathsf{T}) \times \partial\Omega \\ \mathsf{u} = \mathsf{u}_0 & \{0\} \times \Omega \end{cases}$$

It describes the time evolution of the temperature u of a medium contained in Ω under an external heat source f; u_0 is the initial temperature.

Laplacian equation

The problem :

$$\begin{cases} -\Delta u = f & \Omega \\ u = g & \Gamma_D \\ \dfrac{\partial u}{\partial n} = h & \Gamma_N \\ \dfrac{\partial u}{\partial n} + u = l & \Gamma_R \end{cases}$$

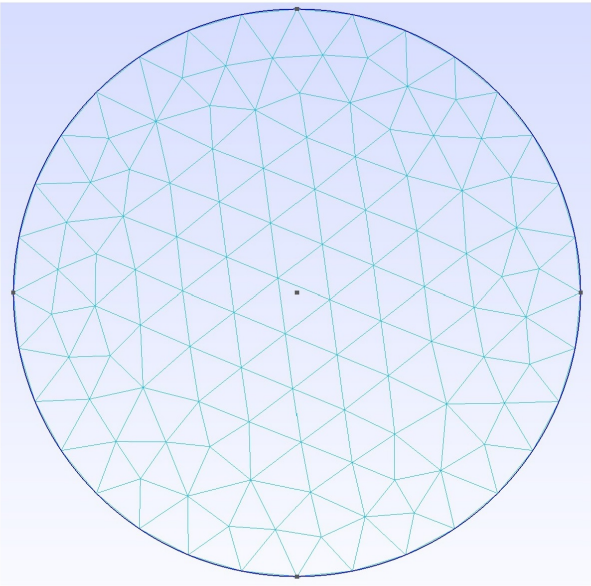


Figure 13 : Mesh of the geometry (with Dirichlet BC)

Weak formulation :

Find $u : \Omega \mapsto \mathbb{R}$ such that $u \in H^1_{g,\Gamma_D}(\Omega)$ and

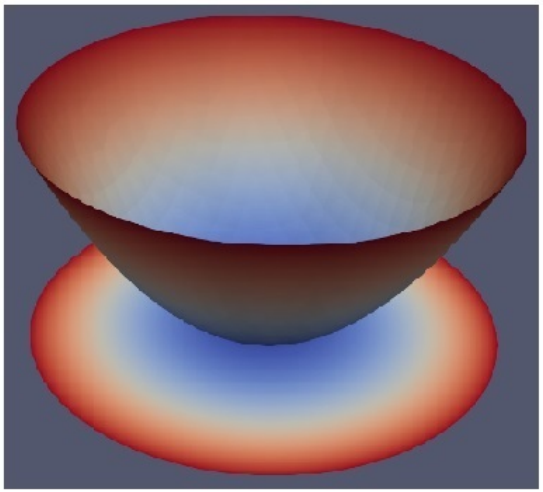
$$\int_\Omega \nabla u \cdot \nabla v + \int_{\Gamma_R} uv = \int_\Omega fv + \int_{\Gamma_N} hv + \int_{\Gamma_R} lv \quad \forall v \in H^1_{0,\Gamma_D}(\Omega)$$

cemosis
Centre de modélisation et de simulation de Strasbourg

Convergence with Laplacian I

$$\begin{cases} -\Delta u = f & \Omega \\ u = g & \Gamma_D \end{cases} \quad \text{with} \quad u_{exact} = g = x^2 + y^2, \quad f = -\Delta u_{exact} = -4$$

$$||u - u_h||_{L^2} \sim h^2 \qquad ||u - u_h||_{H^1} \sim h^1$$



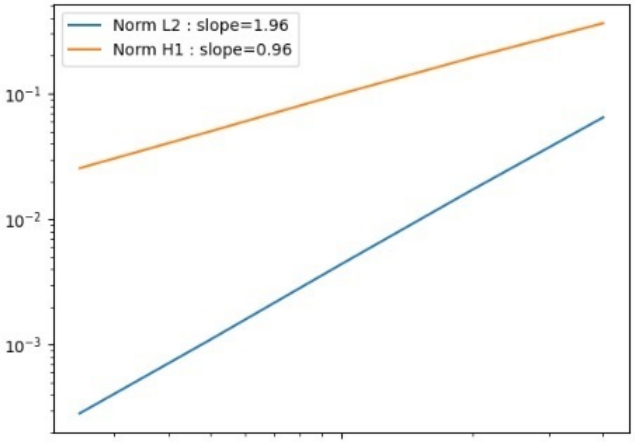
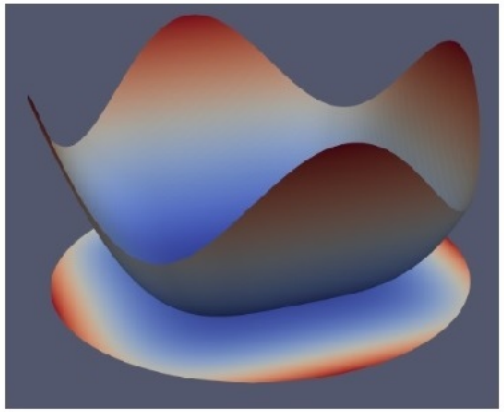


Figure 14 : Example 1 with $P^1_{c,h}$

Convergence with Laplacian II

$$\begin{cases} -\Delta u = f & \Omega \\ u = g & \Gamma_D \end{cases} \quad \text{with} \quad u_{exact} = g = x^4 + y^4, \quad f = -\Delta u_{exact} = -(12x^2 + 12y^2)$$

$$||u - u_h||_{L^2} \sim h^4 \qquad ||u - u_h||_{H^1} \sim h^3$$



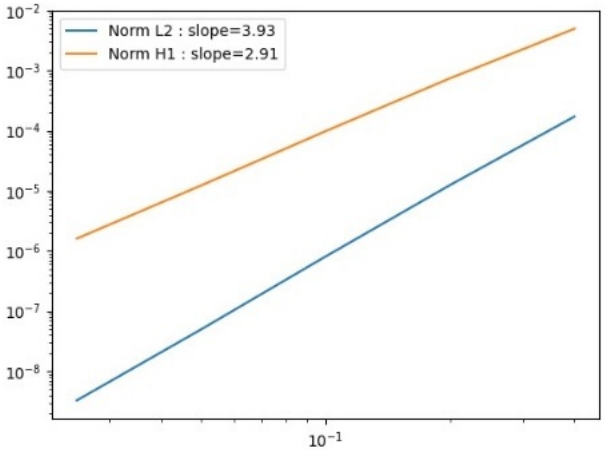


Figure 15 : Example 2 with $P^3_{c,h}$

Back to the heat equation

Weak formulation :

Find $u : (t_0, T) \times \Omega \mapsto \mathbb{R}$ such that $u(t, \cdot) \in H^1_{g,\Gamma_D}(\Omega)$ and

$$\boxed{\int_\Omega \frac{\partial u}{\partial t}(t,x)v(x)dx} + \int_\Omega \nabla u(t,x) \cdot \nabla v(x)dx = \int_\Omega f(t,x)v(x)dx \quad \forall v \in H^1_{0,\Gamma_D}(\Omega)$$

for almost every $t \in (0, T)$.

Temporal discretization :

$$\int_\Omega \frac{\partial u}{\partial t}(t,x)v(x)dx \simeq \int_\Omega \frac{u^{n+1}(x) - u^n(x)}{\Delta t} v(x)dx$$

We then obtain :

$$\frac{1}{\Delta t}\int_\Omega u^{n+1}(x)v(x)dx + \int_\Omega \nabla u^{n+1}(t,x) \cdot \nabla v(x)dx$$
$$= \int_\Omega f(t,x)v(x)dx + \frac{1}{\Delta t}\int_\Omega u^n(x)v(x)dx$$

cemosis
Centre de modélisation et de simulation de Strasbourg

Parareal method with the Heat equation I

Provided : a class **Heat** allowing the solution of the heat equation with Feel++ (constructor define : the bilinear form a + the linear form l + boundary conditions).

Test case :

- Spatial : 2 sub-domains.
- Temporal : 2 sub-domains.

$\Rightarrow$ 2*2 processes : fine integrator + 2 processes : coarse integrator $\Rightarrow$ 6 processes.

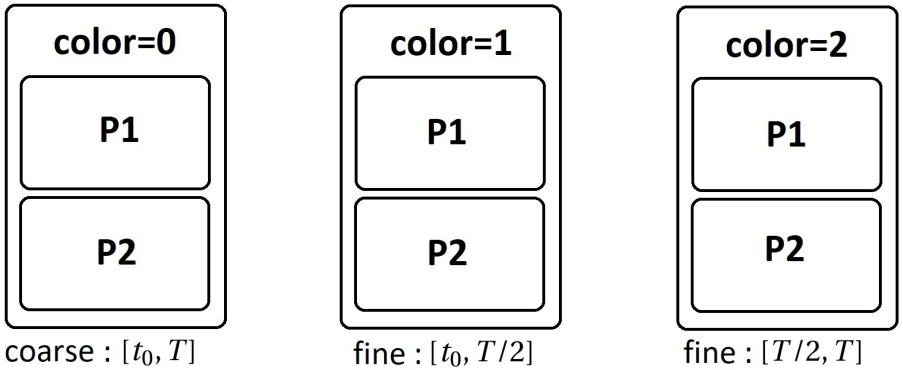


Figure 16 : Explanation of the test case.

Parareal method with the Heat equation II

Need to communicate :

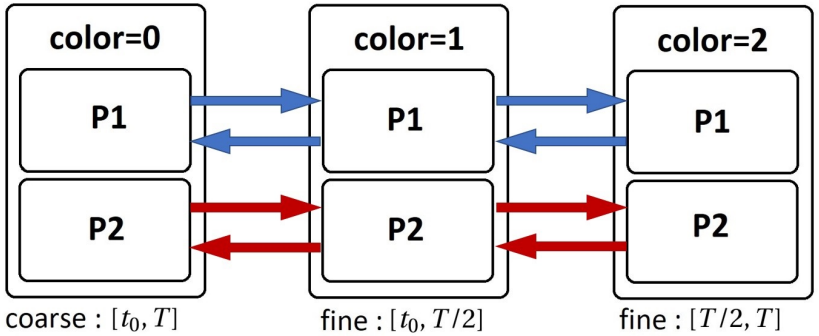


Figure 17 : Communications.

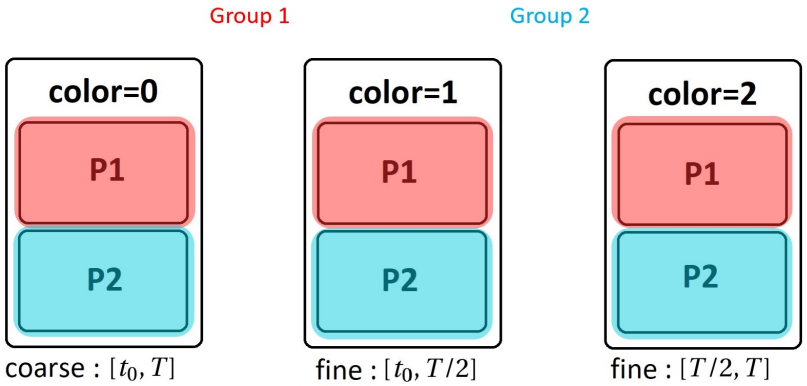


Figure 18 : Groups.

Parareal method with the Heat equation III

Results : Let $t_0 = 0$, $T = 1$ and a piece of finned heat exchanger.

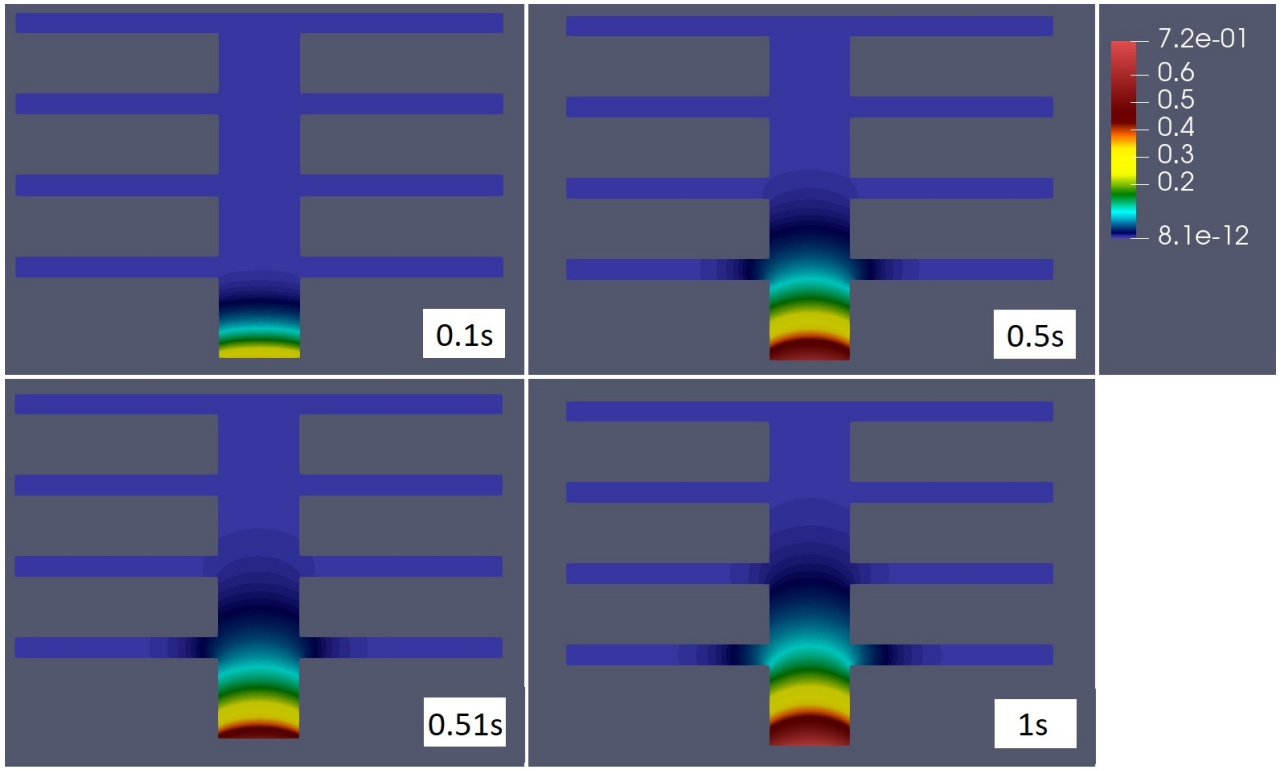


Figure 19 : Parareal method apply to the heat equation.

Application of the data assimilation method with PDE models in Feel++

Objectives

- Write a class for the EnKF in C++, inspired by the EnKF written in Python (FilterPy); Test the algorithm.
- To read the following article :
  - Fundamentals Of Building Performance Simulation By Beausoleil-Morrison
- Understand the heat equation and what are the phenomena involved in the modification of the temperature of a building (conduction, convection, radiation).
- Write the mathematical problem to be solved if we want to simulate an office then realize the simulation using Feel++ toolboxes.
- Introduce data assimilation using the sensor and correct the simulation.

Introduction to data assimilation

Data assimilation is widely used in:

- weather forecasting
- ocean simulation

The main idea of computational data assimilation is to combine:

- a model
- some observations

The best estimation:

$$x^a = Lx^b + Ky^0$$

with x^a the analyzed state, x^b the state of the model and y^0 the observations.

Theory

Kalman filter in multi-dimensional case

Now that we have explained the method for finding x^a let's try to generalize our formula in a **multi-dimensional case**.

$$\begin{cases} x^a = (I - KH)x^b + Ky^0 = x^b + K(y^0 - H(x^b)) \\ K = BH^T(HBH^T + R)^{-1} \end{cases}$$

With K the gain or weight matrix.

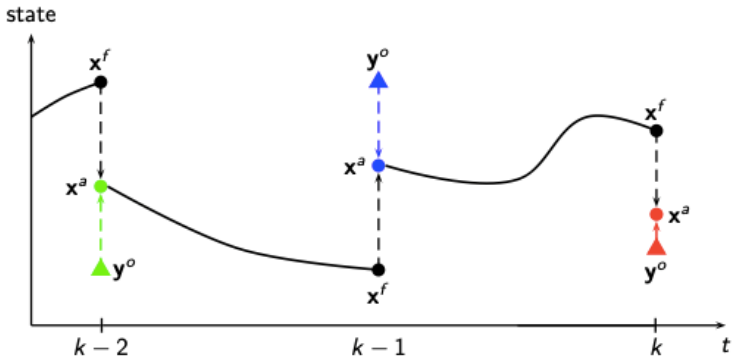


Figure 11: Kalman Filter

Ensemble Kalman Filter I

The ENKF method consists in using the Kalman filter method in high dimension and compare P by a set of states $x_1, x_2, .., x_m$. So we can approximate the moments of the error by the moments of the sample. For all the samples, we have:

$$x_i^a = x_i^f + K[y - h(x_i^f)]$$

with $h(x_i^f)$ the observation operator.
To begin with we can estimate the forecast error covariance matrix as:

$$P^f = \frac{1}{m-1}\sum_{i=1}^{m}(x_i^f - \bar{x}^f)(x_i^f - \bar{x}^f)^T \text{ with } \bar{x}^f = \frac{1}{m}\sum_{i=1}^{m} x_i^f.$$

We can factorized the forecast error covariance matrix by:

$$P^f = X_f X_f^T$$

Ensemble Kalman Filter II

where X_f is an $n \times m$ matrix whose columns are the normalized anomalies or normalized perturbations,

$$[X_f]_i = \frac{x_i^f - \bar{x}^f}{\sqrt{m-1}}$$

We can also define the Kalman gains:

$$K = P^f H^T (HP^f H^T + R)^{-1}$$

In addition, we have:

$$\bar{x}^a = \frac{1}{m} \sum_{i=1}^{m} x_i^a \ , \quad [X_a]_i = \frac{x_i^a - \bar{x}^a}{\sqrt{m-1}}.$$

Ensemble Kalman Filter III

We can write the covariance matrix of the analysis errors as:

$$P^a = (I_n - KH)P^f(I_n - KH)^T + KRK^T = (I_n - KH)P^f.$$

Add a perturbation to the observation: $y \to y_i + \bar{u}_i$ where u_i is drawn from the Gaussian distribution $u_i \sim \mathcal{N}(0, R)$
the innovation perturbations as :

$$[Y_f]_i = \frac{Hx_i^f - u_i - H\bar{x}^f + \bar{u}}{\sqrt{m-1}}.$$

Finally we can modify the posterior anomalies:

$$X_i^a = X_i^f - KY_i^f = (I_n - KH)X_i^f + \frac{K(u_i - \bar{u})}{\sqrt{m-1}}.$$

Ensemble Kalman Filter Algorithm

input : For k=0,...,K: the observation error covariance matrices R_k, the observation models H_k, the forward models M_k.

Initialize the ensemble $\left\{x^f_{i,0}\right\}_{=1,\dots,m}$;

for k=0,...,K **do**

  Draw a statistically consistent observation set:

  for i=1,...,m: $y_{i,k} = y_k + u_i$, with $u_i \sim \mathcal{N}(0, R_k)$;

  Compute the ensemble means

  $\bar{x}^f_k = \frac{1}{m}\sum_{i=1}^m x^f_{i,k}$, $\bar{u} = \frac{1}{m}\sum_{i=1}^m u_i$, $\bar{y}^f_k = \frac{1}{m}\sum_{i=1}^m H_k(x^f_{i,k})$

  and the normalized anomalies

  $[X_f]_{i,k} = \frac{x^f_{i,k} - \bar{x}^f_k}{\sqrt{m-1}}$, $[Y_f]_{i,k} = \frac{H_k(x^f_{i,k}) - u_i - \bar{y}^f_k + \bar{u}}{\sqrt{m-1}}$;

  Compute the gain: $K_k = X^f_k (Y^f_k)^T \left\{Y^f_k (Y^f_k)^T\right\}^{-1}$;

  Update of the ensemble:

  for i=1,...,m: $x^a_{i,k} = x^f_{i,k} + K_k(y_{i,k} - H_k(x^f_{i,k}))$;

  Compute the ensemble forecast:

  for i=1,...,m: $x^f_{i,k+1} = M_{k+1}(x^a_{i,k})$

end

Algorithm 4: Ensemble Kalman Filter

cemosis
Centre de modélisation et de simulation de Strasbourg

Comparing Cpp results with Python I

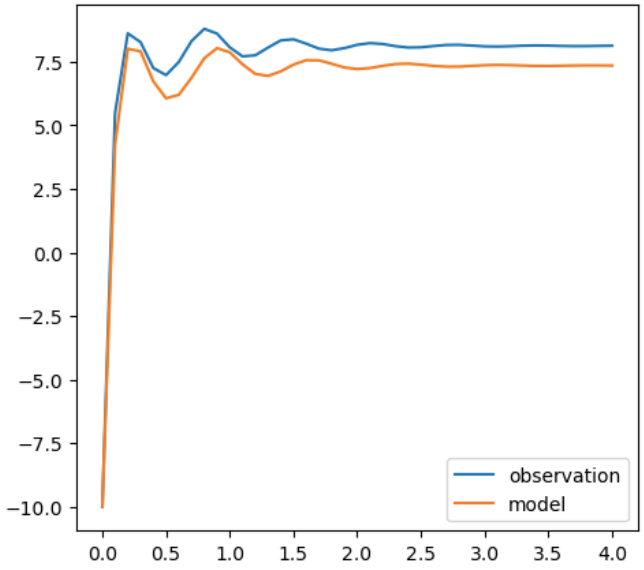


Figure 12: Visualization of the model and observations (Initial condition: $X_0 = (-10., -10., 25.)$)

- $(\sigma, \mathrm{r}, \mathrm{b}) = (12., 6., 12.)$ for observation
- $(\sigma, \mathrm{r}, \mathrm{b}) = (10., 6., 10.)$ for the model

$$\mathrm{P} = \begin{pmatrix} 0.1 & 0. & 0. \\ 0. & 0.1 & 0. \\ 0. & 0. & 0.1 \end{pmatrix} \mathrm{Q} = \begin{pmatrix} 0.1 & 0. & 0. \\ 0. & 0.1 & 0. \\ 0. & 0. & 0.1 \end{pmatrix}$$

$$\mathrm{R} = \begin{pmatrix} 0.01 & 0. & 0. \\ 0. & 0.01 & 0. \\ 0. & 0. & 0.01 \end{pmatrix}.$$

cemosis
Centre de modélisation et de simulation de Strasbourg

Comparing Cpp results with Python II

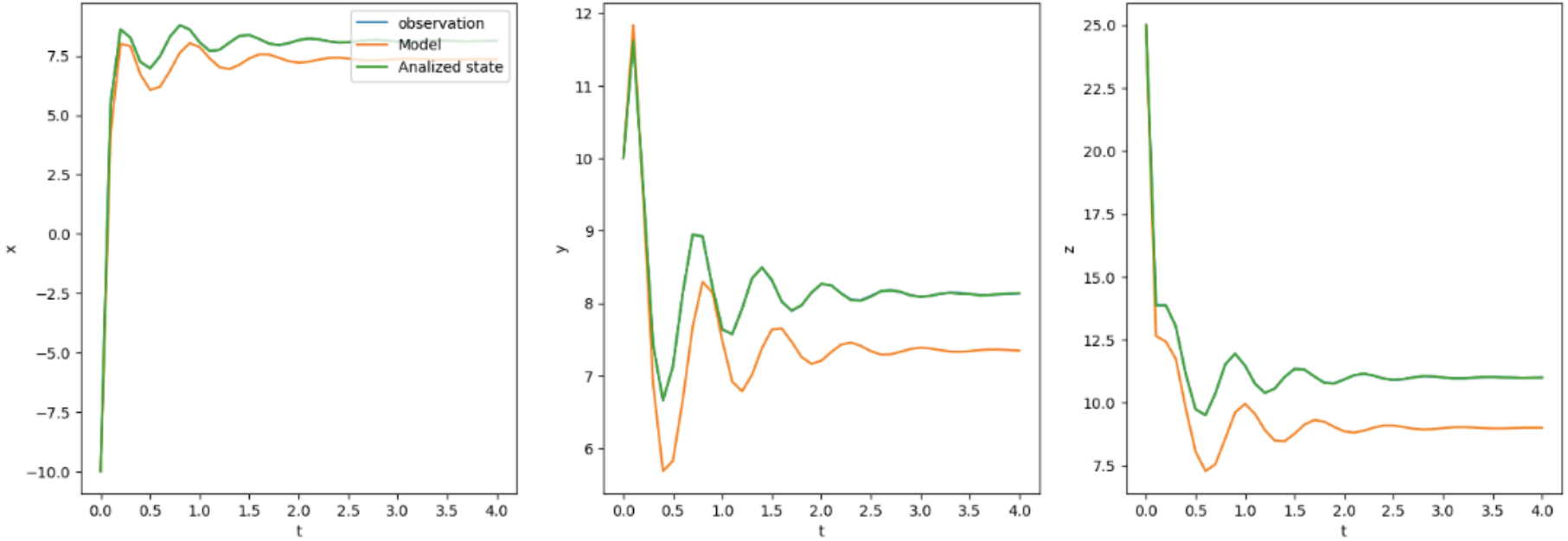


Figure 13: Result with Filterpy

Comparing Cpp results with Python III

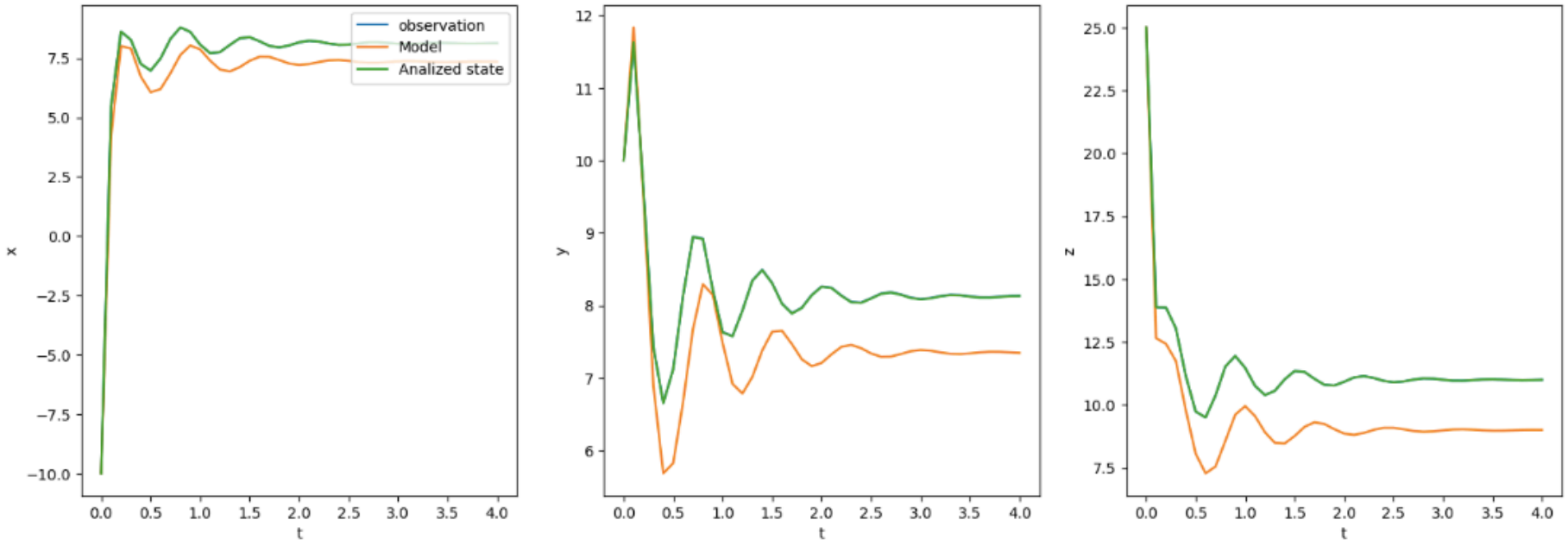


Figure 14: Result with Cpp

Integrate data assimilation to Feel++

The context I

- Our goal is to make the thermal simulation of an office in the university of Strasbourg in which we have 10 sensors to measure the temperature.
- We want to apply data assimilation and use our sensors to correct the temperature of the room.

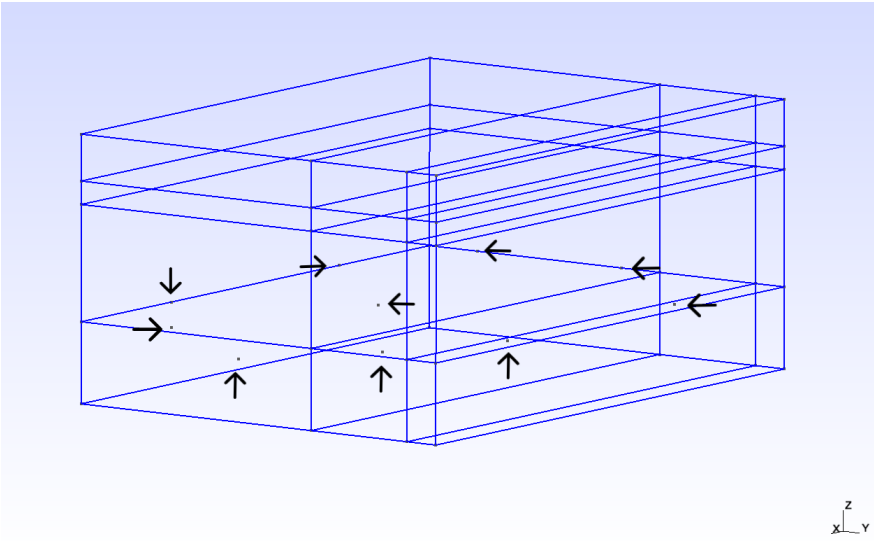


Figure 15: Geometry of the office that we will use for the simulation.

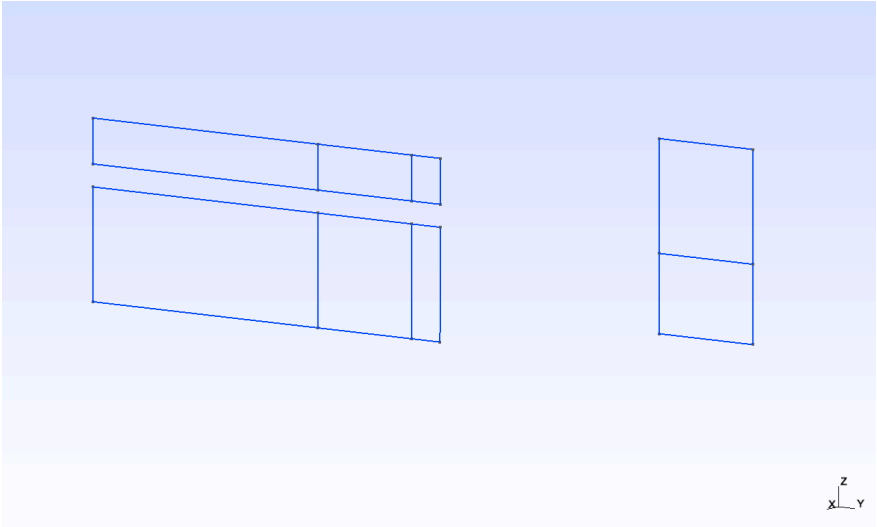


Figure 16: Visualization of windows and door in the geometry.

The context II

- Heat equation with convective effects

$$\rho C_p((\frac{\partial T}{\partial t}) + u \cdot \nabla T) - \nabla \cdot (k \nabla T) = Q$$

Which is completed with boundary conditions and initial value.

ρ	Air density	$Kg.m^{-3}$	1.125
C_p	Specific heat	J/KgK	1004
k	Conductivity	W/mK	0.025
u	Fluid velocity	$m.s^{-1}$	unknown
T	Temperature	K or C	unknown

cemosis
Centre de modélisation et de simulation de Strasbourg

The context III

- Equation of the air motion (Navier-Stokes).

$$\begin{cases} \rho(\frac{\partial \mathrm{u}}{\partial \mathrm{t}} + \mathrm{u} \cdot \nabla \mathrm{u}) - \nabla \cdot (\mu \nabla \mathrm{u}) + \nabla \mathrm{P} = -\rho_0 \beta (\mathrm{T} - \mathrm{T}_{\mathrm{ref}}) \mathrm{g} \\ \nabla \cdot \mathrm{u} = 0 \end{cases}$$

ρ	fluid density	Kg.m^{-3}	1.125
u	fluid velocity	m.s^{-1}	unknown
β	coefficient of thermal expansion	K^{-1}	0.0034
μ	dynamic viscosity	Pa.s	$1.81\mathrm{e}^{-5}$
g	gravitational acceleration	m.s^{-2}	9.8
ρ_0	fluid density of air	kg.m^{-3}	1.225

cemosis
Centre de modélisation et de simulation de Strasbourg

The context IV

- Heat equation without convective effects

$$\rho C_p\left(\frac{\partial T}{\partial t}\right) - \nabla \cdot (k\nabla T) = Q$$

Which is completed with boundary conditions and initial value.

ρ	Air density	$Kg.m^{-3}$	1.125
C_p	Specific heat	J/KgK	1004
k	Conductivity	W/mK	0.025
T	Temperature	K or C	unknown

The context V

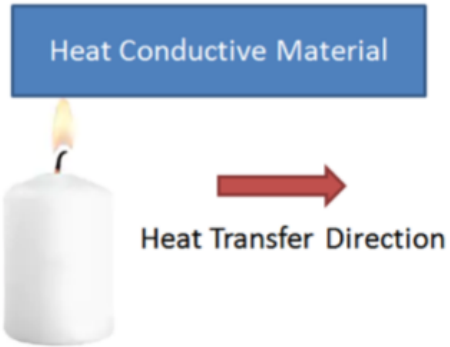


Figure 17: Conduction

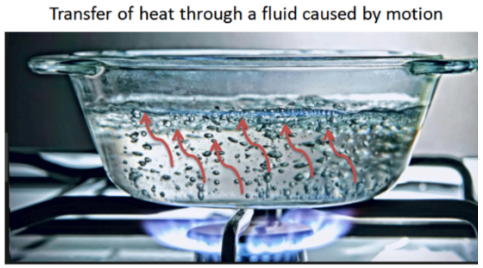


Figure 18: Convection

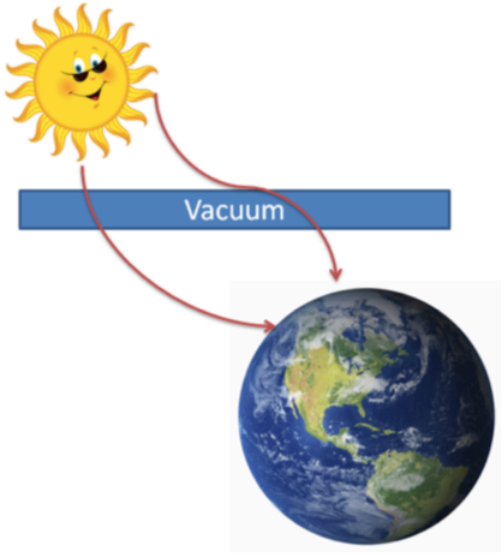


Figure 19: Radiation

Simulation I

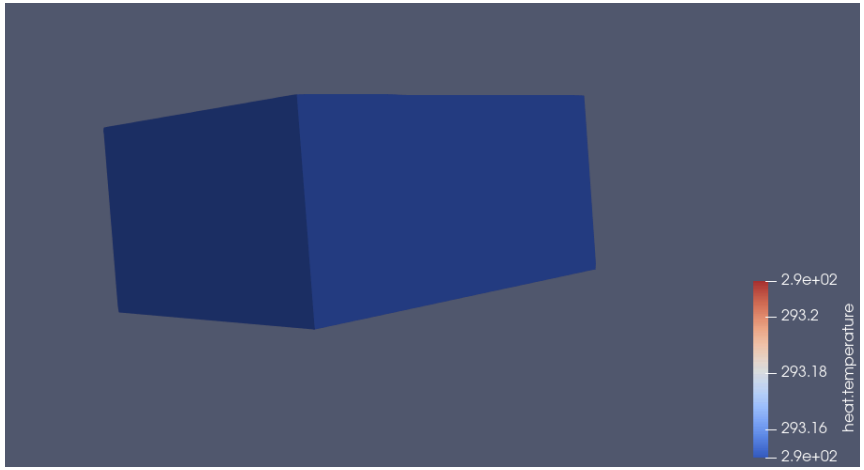


Figure 20: Visualization of the simulation made with the toolbox heat (Initial condition) .

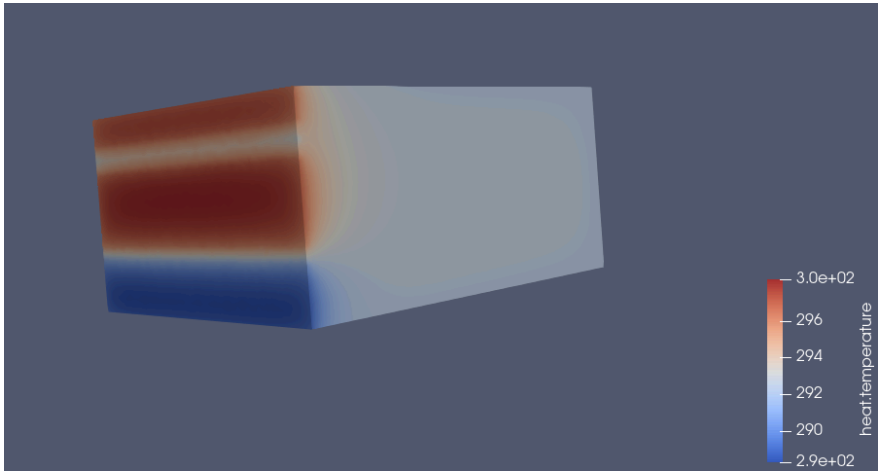


Figure 21: Visualization of the simulation made with the toolbox heat (1 hour) .

- For this simulation we do not consider the convection .

Including data assimilation to our simulation I

```
EnsembleKalmanFilter(double dim_x,double dim_z,MyMatrix x,MyMatrix P,double dt,
                     MyMatrix ( *hx)(MyMatrix  x),MyMatrix ( *fx)(double t,MyMatrix  x));
```

Ensemble Kalman Filter class:

- the dimension of our model;
- the dimension of the vector containing our observations;
- a vector X which will represent the initial state for our model;
- the matrix P ;
- our time step dt.
- a function hx;
- a function fx.

Including data assimilation to our simulation II

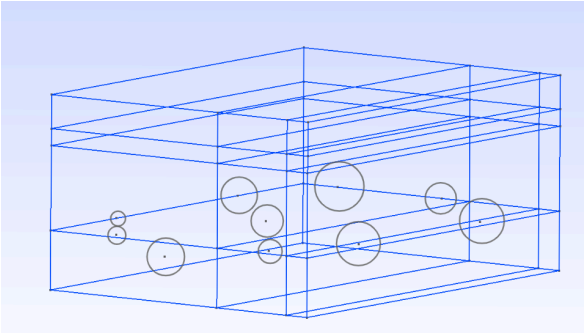


Figure 22: Geometry with radius on the sensors

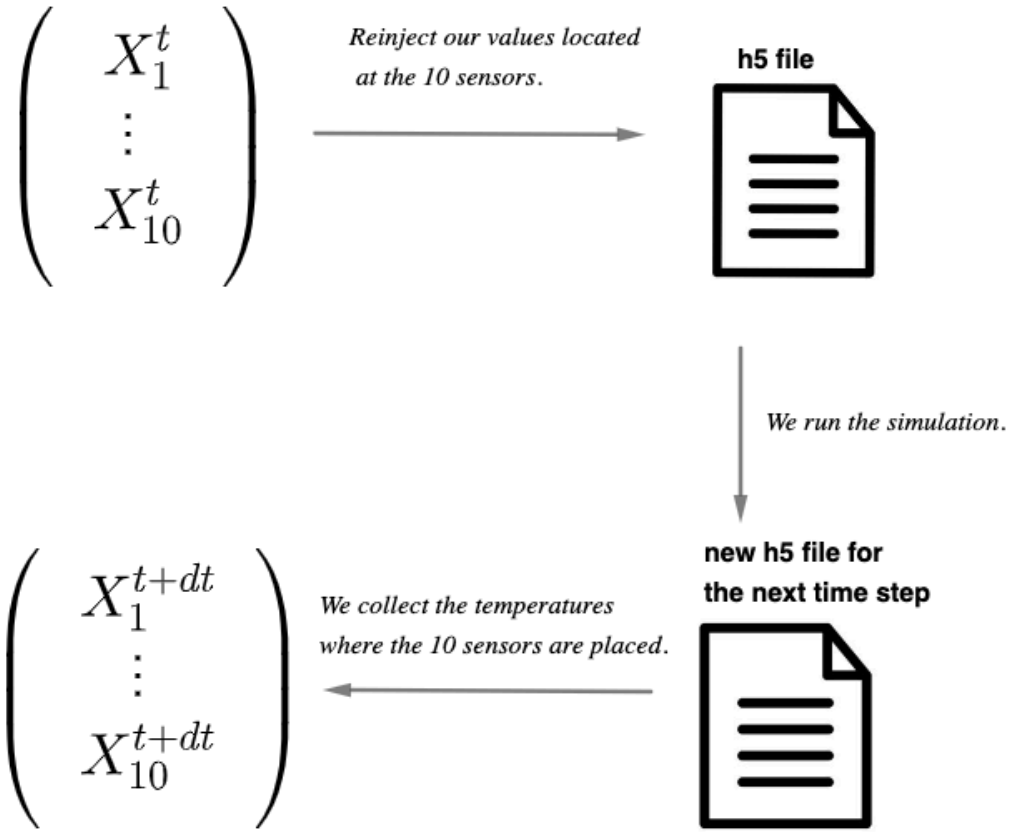


Figure 23: Steps for the fx function

A few comments about the internship

Bibliography I

[1] Antora Documentation : Antora Docs. URL: https://docs.antora.org/antora/latest/ (visited on 08/14/2022).

[2] University Corporation for Atmospheric Research. The Lorenz 63 model and its relevance to data assimilation. URL: https://docs.dart.ucar.edu/en/latest/guide/lorenz-63-model.html (visited on 04/03/2022).

[3] Ian Beausoleil-Morrison. "Fundamentals of Building Performance Simulation". In: (), p. 410.

[4] Eric Blayo. An introduction to data assimilation. en. URL: https://www.eccorev.fr/IMG/pdf/Assimilationdonnees_EBlayo.pdf.

[5] cmake-presets(7) — CMake 3.24.0 Documentation. URL: https://cmake.org/cmake/help/latest/manual/cmake-presets.7.html#test-preset (visited on 08/14/2022).

[6] Doxygen: Doxygen. URL: https://doxygen.nl/index.html (visited on 08/14/2022).

[7] Eigen: Getting started. URL: https://eigen.tuxfamily.org/dox/GettingStarted.html (visited on 08/17/2022).

[8] Feel++ documentation :: Feel++ Docs. URL: https://docs.feelpp.org/feelpp/0.109/index.html (visited on 08/17/2022).

[9] Laplacian : Feel++ Docs. URL: https://docs.feelpp.org/user/latest/cpp/laplacian.html (visited on 08/14/2022).

Bibliography II

[10] Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. Résolution d'EDP par un schéma en temps pararéel. en. Apr. 2001. DOI: 10.1016/S0764-4442(00)01793-6. URL: https://hal.archives-ouvertes.fr/hal-00798372/file/CRAS_01_lions_maday_turinici.pdf (visited on 05/22/2022).

[11] Y. Maday. The 'Parareal in Time' Algorithm. en. Ed. by F. Magoulès. Stirlingshire, UK, May 2010. DOI: 10.4203/csets.24.2. URL: https://www.ljll.math.upmc.fr/publications/2008/R08030.pdf (visited on 05/22/2022).

[12] Maëlle Nodet. Introduction to Data Assimilation. en. URL: https://team.inria.fr/airsea/files/2012/03/Nodet_Intro_DataAssimilation.pdf.

[13] Parareal. en. Page Version ID: 1047894968. Oct. 2021. URL: https://en.wikipedia.org/w/index.php?title=Parareal&oldid=1047894968 (visited on 05/22/2022).

[14] Alfio Quarteroni and Alberto Valli. Numerical Approximation of Partial Differential Equations. Red. by R. L. Graham, J. Stoer, and R. Varga. Vol. 23. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994. ISBN: 978-3-540-85267-4 978-3-540-85268-1. DOI: 10.1007/978-3-540-85268-1. URL: https://link.springer.com/content/pdf/10.1007/978-3-540-85268-1.pdf (visited on 08/17/2022).

[15] S Ricci. Data Assimilation training course @ CEMRACS Introduction and variational algorithms. en. URL: http://smai.emath.fr/cemracs/cemracs16/images/DA_Ricci_CEMRACS2016.pdf.

Bibliography III

[16] John Stockie and Ken Wong. Laboratory #6: A Simple Model of the Unpredictability of Weather: The Lorenz Equations. en. URL: https://ftp.emc.ncep.noaa.gov/mmb/sref/Doc/lorenz_fcst.pdf.

[17] THREE DIMENSIONAL SYSTEMS, Lecture 6 : The Lorenz Equations. URL: https://www2.physics.ox.ac.uk/sites/default/files/profiles/read/lect6-43147.pdf (visited on 04/03/2022).

[18] Welcome — Sphinx documentation. URL: https://www.sphinx-doc.org/en/master/# (visited on 08/14/2022).