Solving elliptic PDEs with Feynman-Kac formula

Giovanni Conforti, Berlin Mathematical School

Introduction

We implement in SCILAB a probabilistic method to solve elliptic PDE's based on the Feynman-Kac representation formula.

- This method is **not** implemented in most of the commercial math softwares (i.e. Matlab), though is widely used in many applications.
- It is very easy to implement within SCILAB, showing that Scilab is an ideal framework to designe efficient, cheap and innovative algorithms.

Scilab for PDEs and Stochastics

- **Xcos** has a block devoted to PDEs. It deals with 2-dimensional PDEs using classical finite differences or finite elements methods, parameters can be tuned by the user.

- An advanced course on Monte Carlo methods and simulation in Scilab is available at

  http://www.math.u-bordeaux1.fr/ pdelmora/lectures.html

Heat Equation: initial value problem

$$\begin{cases} \partial_t u - \Delta u = 0, & (t,x) \in D \\ u(x,0) = f(x) \end{cases}$$

describes the evolution of the temperature of a body without external energy exchange.

- $f(x,0)$ is the inital temperature at the point x
- $u(x,t)$ is the temperature at the point x at time t

The equation can be derived from the basic principles of thermodynamics:

- First law of thermodynamics
- Conservation of energy
- Fourier's law

Classical methods

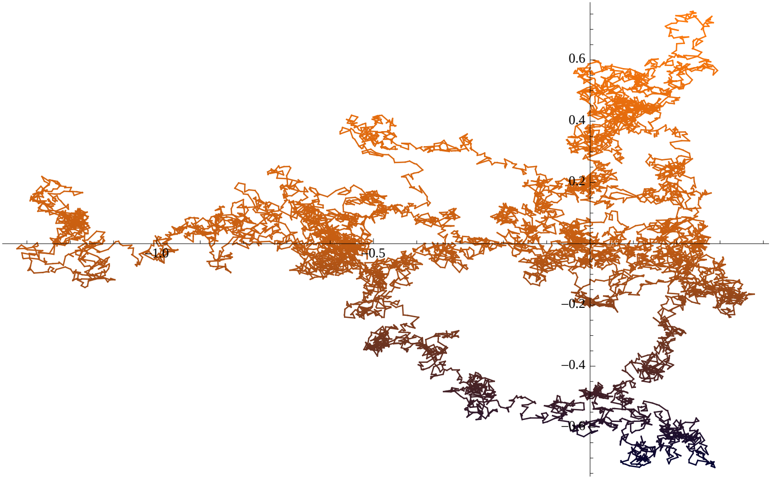
- **Finite differences:** iteratively solve a finite differences equation that approximates the PDE
- **Finite elements:** refined design of the approximating mesh.
- **Fourier methods:** are based on the study of the eigenfunctions of the Laplacian operator Δ.

Brownian motion

Should be thought as a **random curve** $t \mapsto B_t$ with the largest possible amount of randomness. It is the prototype of stochastic process. It is characterized by few simple properties :

- Continuous trajectories
- Independent increments $B_t - B_s \perp B_{[0,s]}$
- Stationary increments $B_{t+h} - B_t \overset{d}{=} B_{s+h} - B_s \quad \forall h.$

Brownian motion



Feynman-Kac formula

A surprising connection beetween stochastic processes and PDE's. First pointed out by Feynman to solve the Schrödinger equation

Feynman-Kac formula

Let u be a solution of:

$$\begin{cases} \partial_t u + b\partial_x u + \sigma^2 \partial^2_{xx} u - vu + f = 0, \\ u(x,0) = f(x) \end{cases}$$

Then u has the following representation:

$$u(t,x) = \mathbb{E}^x \left(f(X_t) \right)$$

where X_t is the solution of the **diffusion**:

$$dX_t = b(t,X_t)dt + \sigma(t,X_t)dB_t$$

- A special case of this formula is the **Black and Scholes** formula, the cornerstone of financial mathematics.

Feynman-Kac formula

Feynman-Kac formula (Brownian motion)

Let u be a solution of:

$$\begin{cases} \partial_t u - \frac{1}{2}\partial^2_{xx} u = 0, \\ u(x,0) = f(x) \end{cases}$$

Then u has the following representation:

$$u(t,x) = \mathbb{E}\left(f(x + B_t)\right)$$

where B_t is the **Brownian Motion** and $\mathbb{E}$ is the expectation operator

SOLVE HEAT EQUATION $\Leftrightarrow$ COMPUTE EXPECTATIONS OF BM

- **Problem**: How to approximate expectations?

Monte Carlo methods

Very robust tool first introduced in Physics.
They are based on the **Law of large numbers**:

LLN

Let $\left\{B_t^i\right\}$ be independent, identically distributed as B_t Then:

$$\lim_{N\to+\infty}\frac{1}{N}\sum_{i=1}^{N} f(B_t^i) = \mathbb{E}\left(f(B_t)\right)$$

- Scilab offers the Toolbox **labostat** to implement the Monte Carlo method

Numerical issues for Monte Carlo

A priori estimates

- The **Central Limit Theorem (CLT)** gives us an estimate on the rate of convergence
- In many cases **Large Deviations** techniques guarantee that the probability of falling out of a fixed tolerance interval decay exponentially fast.

Practical tricks

- Make sure you have a good pseudo-random number generator
- Change of measure, Variance reduction (Polynomial chaos)

Method sketch

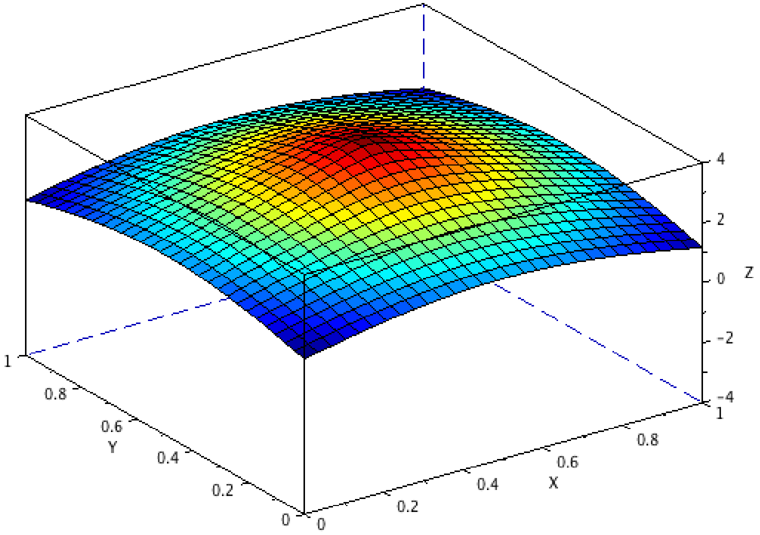
1. Simulate a N trajectories of the Brownian motion $(B^i_s)_{s\in[0,t], i\leq N}$
2. Compute the average of f shifted by x on the sample:

$$\tilde{u}^N(t,x) = \frac{1}{N}\sum_{k=1}^{N} f(x + B^i_t)$$

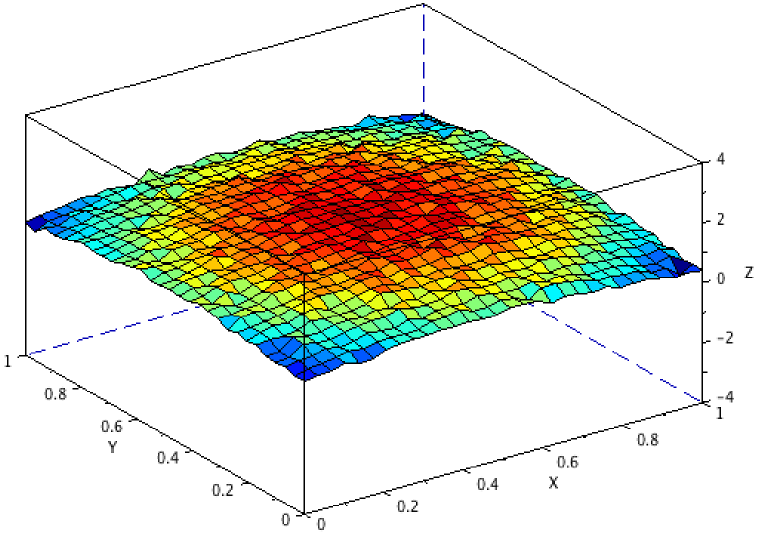
Just a few properties of the method:

- It does not suffer of the curse of dimensionality. Indeed the only approximation is isample size.
- It is not an iterative method, i.e. to compute the value of $\tilde{u}^N$ at a specific point (t,x) you do not need to know anything about the values of $\tilde{u}^N$ at other sites (cfr. finite differences)
- It is very **simple** to implement, it costs 0.00000 Euro.
- It is **competitive**, (ask your Finance division or Physics Dept)

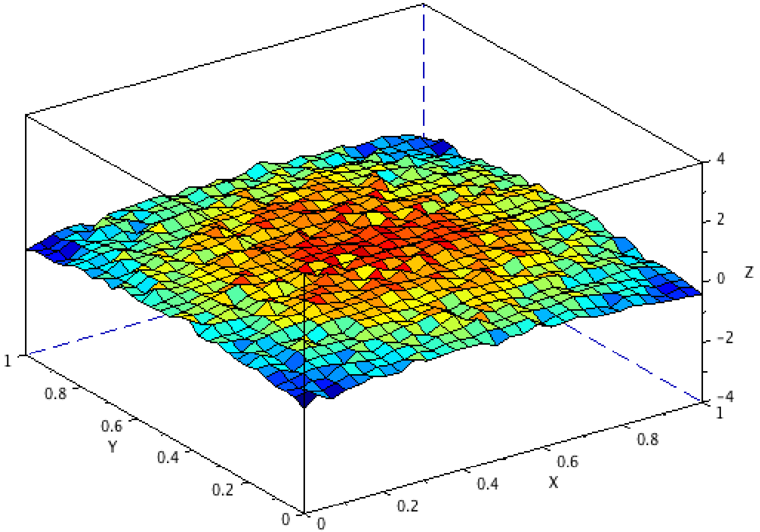
Heat spread through a metal plate



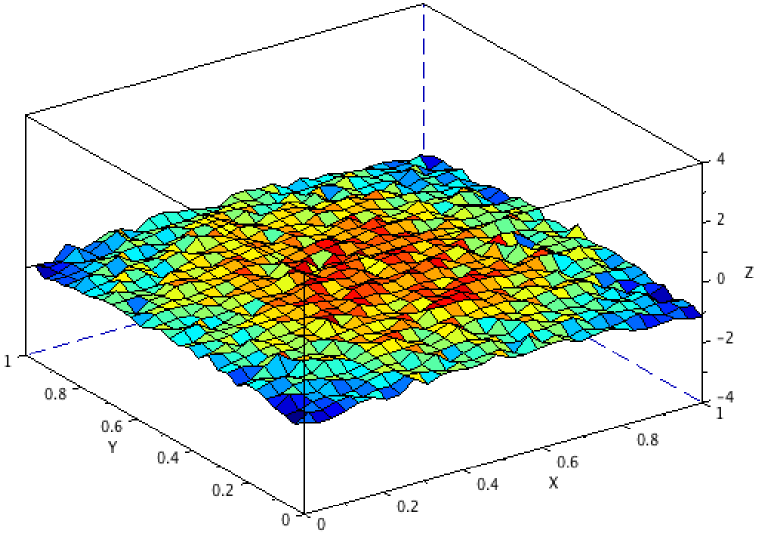
Heat spread through a metal plate



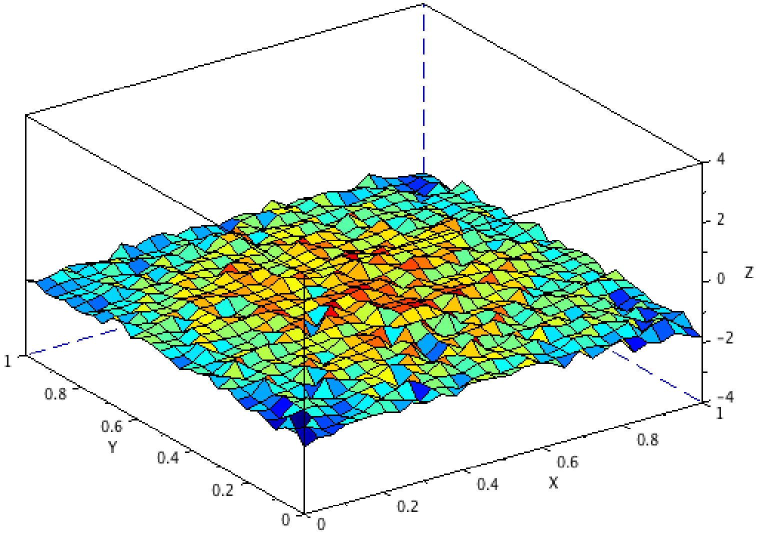
Heat spread through a metal plate



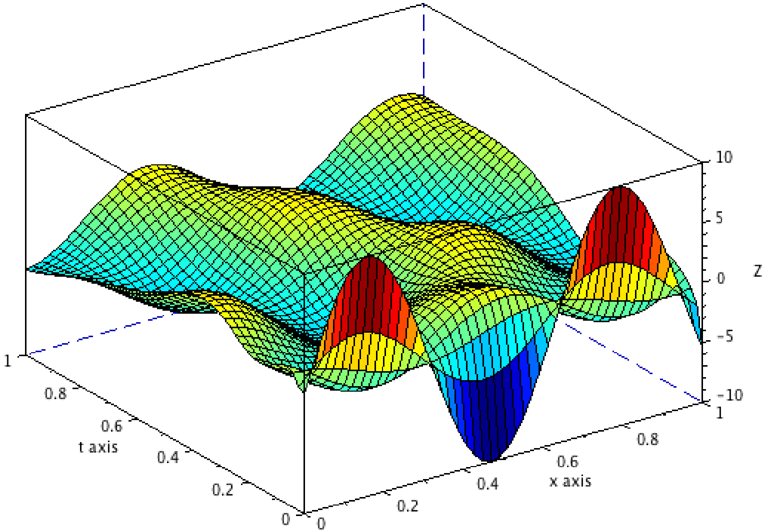
Heat spread through a metal plate



Heat spread through a metal plate



1-Dimensional heat equation, time-space plot



(Multiscale) Conclusions

- **Microscopic:** We have seen an example of a classical domain of applications where mixing a bit of technical knowledge and **high quality open source software** gives good results, with minmal costs.

- **Macroscopic: Don't like PDE's? Quite possible!**
  Howhever, the **methodology** we adopted has very little to do with the specific problem considered.

  1. Search for "state of the art" theoretical methods.
  2. Implementation of the methods **or** use existing code **certified by a community of experts**

Thank you for the attention