New perspectives on computational mathematics: designing probabilistic-based algorithms suited for massively parallel computers

Ángel Rodríguez-Rozas

Center for Mathematics and its applications (CEMAT), Instituto Superior Técnico, Lisbon, Portugal

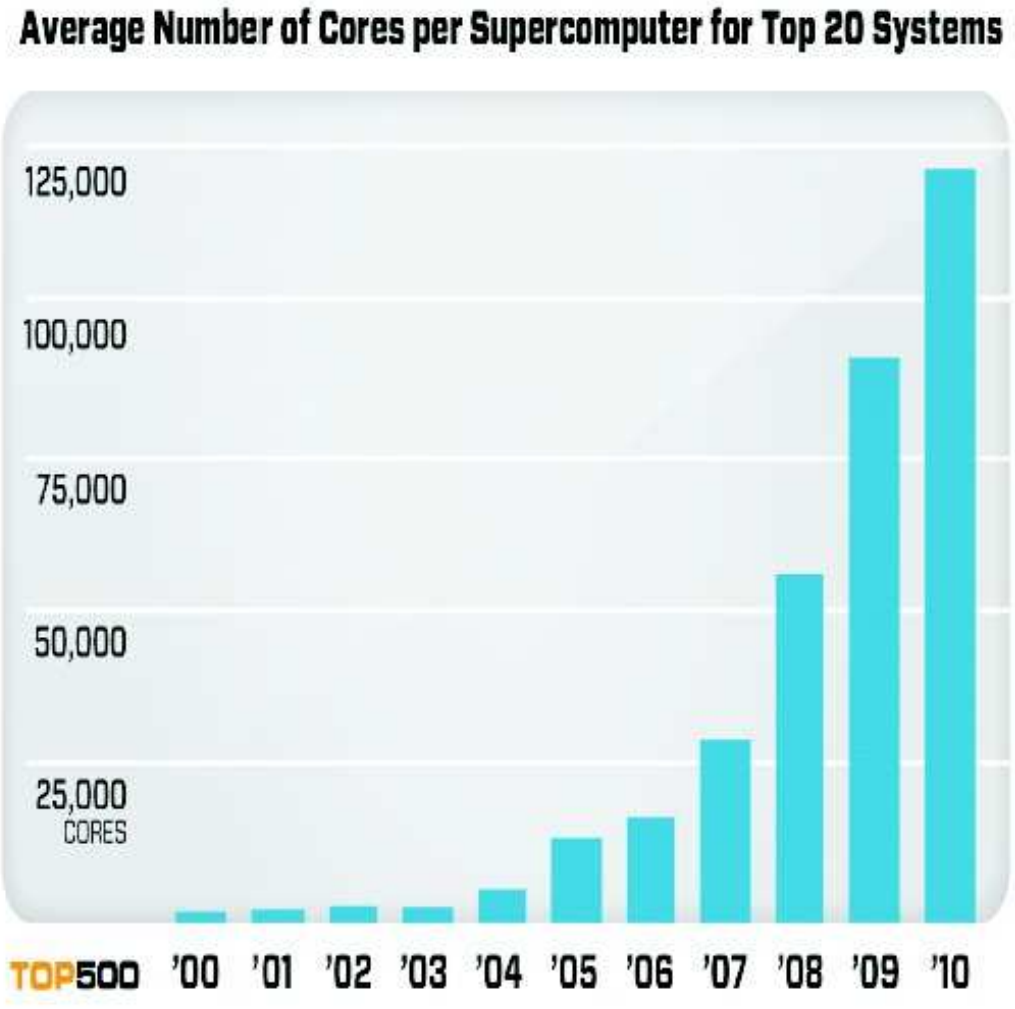
angel.rodriguez@ist.utl.pt

Collaborators:

Juan A. Acebrón (CEMAT,Lisbon), Renato Spigler (University of Roma “Tre”, Italy)

Motivation: HPC

- Increasing computing power today requires increasing the number of processors (or cores) at work.
- The algorithms capable of running efficiently on supercomputers should possess a high degree of parallelism. Parallel scientific computing is mandatory.



Motivation: Computational methods

- It has been observed two source of problems:
  1. A strong communication among the large number of processors reduces the effective performance.
  2. The chance to get a failure in one or several processors during the computation time increases with the number of processors involved.
- To exploit efficiently the increasingly available power, the numerical methods are required to be
  1. **SCALABLE**.
     - *Strong scalability*: More processors at work, less computational time;
     - *Weak scalability*: Larger problem size spending the same time.
  2. **FAULT-TOLERANT**.

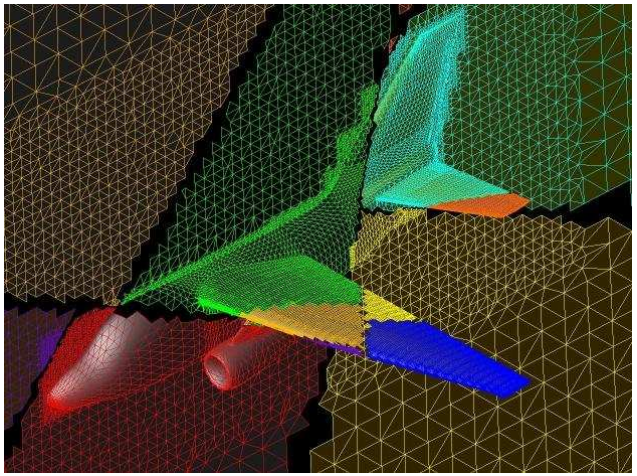
Motivation: Computational methods

- It has been observed in large-scale simulations a wasted of resources, making in some cases impossible to exploit fully the computational resources at hand.

Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak
1	RIKEN Advanced Inst for Comp Sci	K Computer Fujitsu SPARC64 VIIIfx + custom	Japan	548,352	8.16	93
2	Nat. SuperComputer Center in Tianjin	Tianhe-1A, NUDT Intel + Nvidia GPU + custom	China	186,368	2.57	55
3	DOE / OS Oak Ridge Nat Lab	Jaguar, Cray AMD + custom	USA	224,162	1.76	75
4	Nat. Supercomputer Center in Shenzhen	Nebulea, Dawning Intel + Nvidia GPU + IB	China	120,640	1.27	43
5	GSIC Center, Tokyo Institute of Technology	Tusbame 2.0, HP Intel + Nvidia GPU + IB	Japan	73,278	1.19	52
6	DOE / NNSA LANL & SNL	Cielo, Cray AMD + custom	USA	142,272	1.11	81
7	NASA Ames Research Center/NAS	Plelades SGI Altix ICE 8200EX/8400EX + IB	USA	111,104	1.09	83
8	DOE / OS Lawrence Berkeley Nat Lab	Hopper, Cray AMD + custom	USA	153,408	1.054	82
9	Commissariat a l'Energie Atomique (CEA)	Tera-10, Bull Intel + IB	France	138,368	1.050	84
10	DOE / NNSA Los Alamos Nat Lab	Roadrunner, IBM AMD + Cell GPU + IB	USA	122,400	1.04	76

Motivation: Computational methods

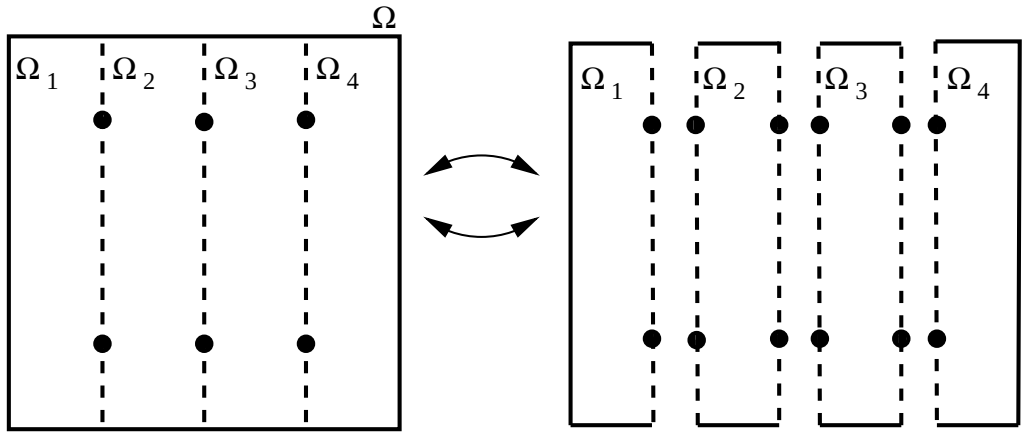
- Source of problems for solving numerically PDEs:
  1. "The tyrany of the computational mesh" (FD,FEM).



  2. "Linear algebra is everywhere", $A\mathbf{x} = \mathbf{b}$.
- Investigating about new numerical methods, and designing new powerful algorithms capable to fully exploit the best available machines, is of paramount importance.

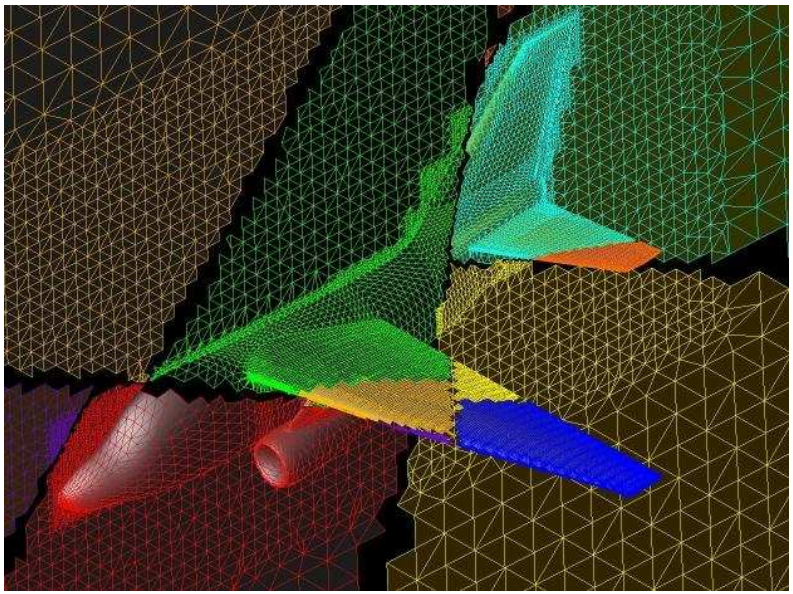
Background: Domain decomposition

- Solving PDEs using *Domain decomposition methods* has been considered one of the most natural ways to take advantage of parallel computer architectures, allowing for high-performance scientific computing of large-scale problems.
- IDEA: A physical domain is partitioned into several subdomains and the global solution is constructed from the “non independent” subproblems associated with the subdomains.



Background: Domain decomposition

- Two important issues arises:
  1. A major problem is represented by the need of having the solution on the interfaces which divide the domain into subdomains (Strong intercommunication overhead)
  2. Globally a computational grid should be deployed to solve numerically the problem.(Large memory resources)



- Is the DD method suited for a large number of processors? *Maybe!*, but for sure not in a “classical” way.

Some maths, and thoughts about PDEs

- The heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \text{ in } \mathbf{R} \quad u(x,0) = f(x),$$

- How can we solve numerically?
  1. A finite difference approach:

$$t_i = i\Delta t, x_j = j\Delta x, \quad i = 0, \ldots, N, j = -M, \ldots, M,$$
$$u_j^i = u(t_i, x_j).$$

$$\frac{u_j^{i+1} - u_j^i}{\Delta t} = \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{\Delta x^2}, \quad u_j^0 = f(x_j)$$

  2. An integral approach,

$$u(x,t) = \int_{\mathbf{R}} dy f(y) G(x,y,t,0),$$

$$\frac{\partial G}{\partial t} = \frac{\partial^2 G}{\partial x^2} \text{ in } \mathbf{R} \quad G(x,y,0,0) = \delta(x-y)$$

Some maths, and thoughts about PDEs

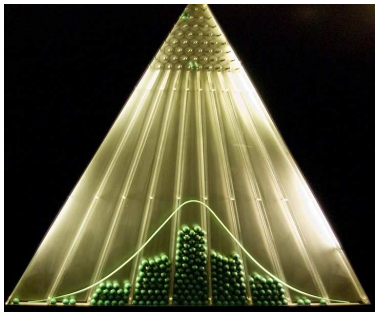
- Important facts: If $f(x) = 1$, then $u(x,t) = 1$, therefore $\int_{\mathbf{R}} dy G(x,y,t,0) = 1$.
  G is a density probability, so can we compute $u(x,t)$ "tossing a coin"?

$$u(x,t) = E[f(\mathbf{y})],$$

  where $\mathbf{y}$ is distributed according to $G(x,y,t,0)$.
- For the heat equation $G(x,y,t,0) = e^{-(x-y)^2/4t}/\sqrt{4\pi t}$

Some maths, and thoughts about PDEs

- A “probabilistic” algorithm for solving the heat equation at a single point (x, t):
  1. Generate a random number y picked up from the density probability $G(x, y, t, 0)$. Note that the “pseudo”random numbers obtained with a computer often are distributed uniformly between $(0, 1)$. Therefore, some transformations are required (Box-Muller method,probability integral transform,etc)



  2. Compute $f(y)$
  3. Repeat N times and take the average

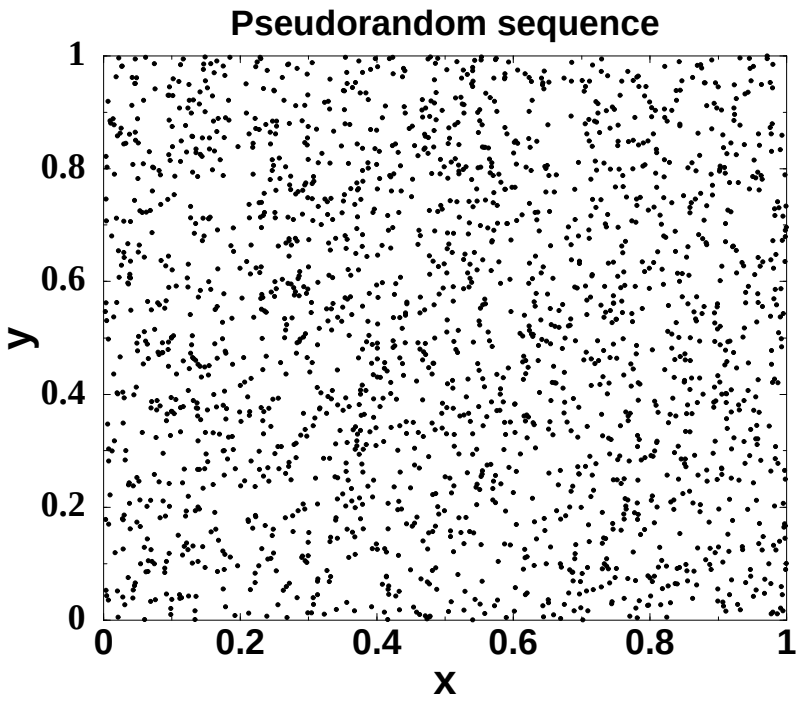
$$u(x,t) \approx u^N(x,t) = \frac{1}{N}\sum_{i=1}^{N} f(y_i)$$

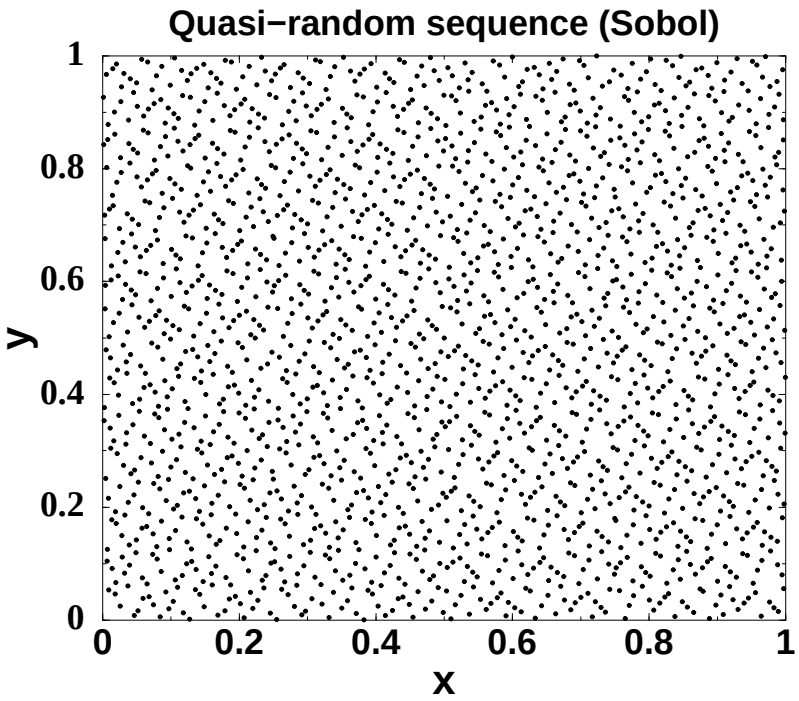
- The error $|u - u^N|$ is purely statistical and of order of $O(N^{-1/2})$.

Monte Carlo simulations: Random numbers

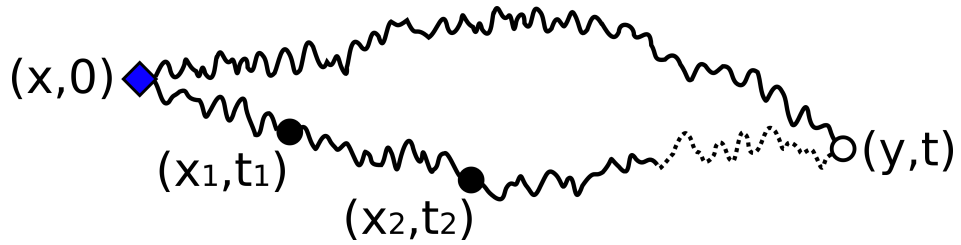
- A reliable source of uniform random numbers is an essential building block for any sort of stochastic modeling or Monte Carlo computer work
- (Pseudo)- random numbers. Linear congruential generators.
- Quasi-random (low discrepancy) sequences are deterministic alternative to random sequences based on pseudorandom numbers.
- The error in uniformity for a sequence of N points in the s-dimensional unit cube is measured by its discrepancy
- The discrepancy is of size $(logN)^s/N$ for large N, as opposed to discrepancy $N^{-1/2}$ for a pseudorandom sequence.

Monte Carlo simulations: Random numbers





Linear parabolic PDEs



- Let suppose the path between $(x,0)$ and (y,t) is divided in several pieces of length Δt. What is the density probability?

$$G(x_2,x,2\Delta t,0) = \int_{\mathbf{R}} dx_1 G(x_2,x_1,2\Delta t,\Delta t)G(x_1,x,\Delta t,0),$$

$$G(x_3,x,3\Delta t,0) = \int_{\mathbf{R}} dx_2 G(x_3,x_2,3\Delta t,2\Delta t,0)G(x_2,x,2\Delta t,\Delta t,0) =$$

$$\int_{\mathbf{R}} dx_2 \int_{\mathbf{R}} dx_1 G(x_3,x_2,3\Delta t,2\Delta t,0)G(x_2,x_1,2\Delta t,\Delta t)G(x_1,x,\Delta t,0),$$

and in general

$$\begin{aligned} G(y,x,n\Delta t,0) \quad = \quad & \int_{\mathbf{R}} dx_{n-1} \int_{\mathbf{R}} dx_{n-2} \ldots \int_{\mathbf{R}} dx_1 G(y,x_{n-1},t,(n-1)\Delta t) \\ & \times G(x_{n-1},x_{n-2},(n-1)\Delta t,(n-2)\Delta t) \cdots G(x_1,x,\Delta t,0), \end{aligned}$$

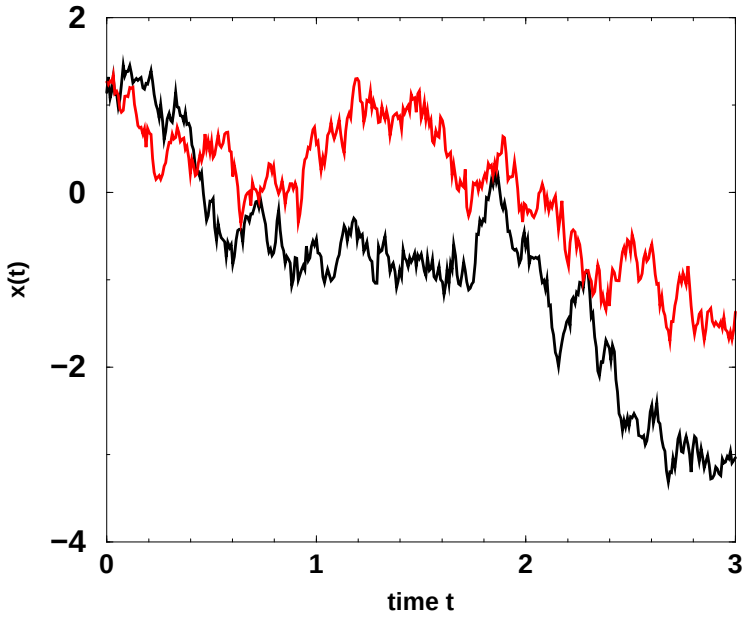
Linear parabolic PDEs

- Mathematically, it is equivalent to the following SDE (Stochastic Differential Equation) equation in discrete form,

$$x_{n+1} = x_n + W_n - W_{n-1}, \quad x_0 = x$$

where W_i is the so-called Brownian motion, which satisfies W_0=0, and $W_n - W_{n-1}$ has independent increments with probability distribution $e^{-y^2/4\Delta t}/\sqrt{4\pi\Delta t}$

- In continuous form is written formally as $dx(t) = dW(t)$.



Linear parabolic PDEs

- Let the BV problem for a linear parabolic PDE,

$$\frac{\partial u}{\partial t} = Lu - c(x,t)u, \quad u(x,0) = f(x), \quad u(x,t)|_{x\in\partial\Omega} = g(x,t)$$

L is a linear elliptic operator $L = a(x,t)\partial_{xx}/2 + b(x,t)\partial_x$, with continuous bounded coefficients, $c(x,t) \geq 0$ and continuous bounded, continuous initial condition, f.

- The probabilistic representation of the solution u through the Feynman-Kac formula is

$$u(x,t) = E\left[f(\beta(t))e^{-\int_0^t c(\beta(s),t-s)}ds\mathbf{1}_{[\tau_{\partial\Omega}>t]}\right]$$
$$+E\left[g(\beta(\tau_{\partial\Omega}),t-\tau_{\partial\Omega})e^{-\int_0^{\tau_{\partial\Omega}} c(\beta(s),t-s)}ds\mathbf{1}_{[\tau_{\partial\Omega}<t]}\right].$$

$$d\beta = b(\beta,t)\,dt + \sigma(\beta,t)\,dW(t).$$

$\beta(\cdot)$ is the stochastic process starting at $(x,0)$, associated to L; $\tau_{\partial\Omega}$ denotes the first exit time, and the expected values are taken with respect to the corresponding measure.

Numerical analysis

- Referring to the numerical simulation of interfacial values, internal to Ω, with N MC or quasi-MC samples), there are three sources of numerical errors:

1. Statistical error.

Numerical analysis

- Referring to the numerical simulation of interfacial values, internal to Ω, with N MC or quasi-MC samples), there are three sources of numerical errors:

1. Statistical error.
2. Truncation error of the SDE.

Numerical analysis

- Referring to the numerical simulation of interfacial values, internal to Ω, with N MC or quasi-MC samples), there are three sources of numerical errors:

1. Statistical error.
2. Truncation error of the SDE.
3. For boundary-value problems: Precise evaluation of the first exit point and time.

Nonlinear PDEs

It has been proposed so far several probabilistic methods for dealing with nonlinear PDEs (eliptic,parabolic, and hyperbolic):

- *Linearization*.
- *Branching diffusion processes*. They are specially suited for solving a class of semilinear PDEs, being the nonlinearity a polynomial function of the solution.They provide higher order numerical methods.
- *Forward-backward stochastic differential equations (FBSDEs)*.They are well suited for any nonlinear PDE, even fully nonlinear. Only first-order numerical methods are known.

Semilinear parabolic PDEs

- A probabilistic representation does exist also for *semilinear* parabolic equations. Such a representation is based on generating *branching* diffusion processes, associated with the elliptic operator, and governed by an exponential random time, S, with density probability $p(S) = c\ \exp(-cS)$.
- H.P. McKean ('75) derived the representation

$$u(x,t) = E[\prod_{i=1}^{k(\omega)} f(x_i(\omega,t))]$$

for the KPP equation

$$u_t = u_{xx} + u(u-1), \quad -\infty < x < +\infty,\ t > 0,$$

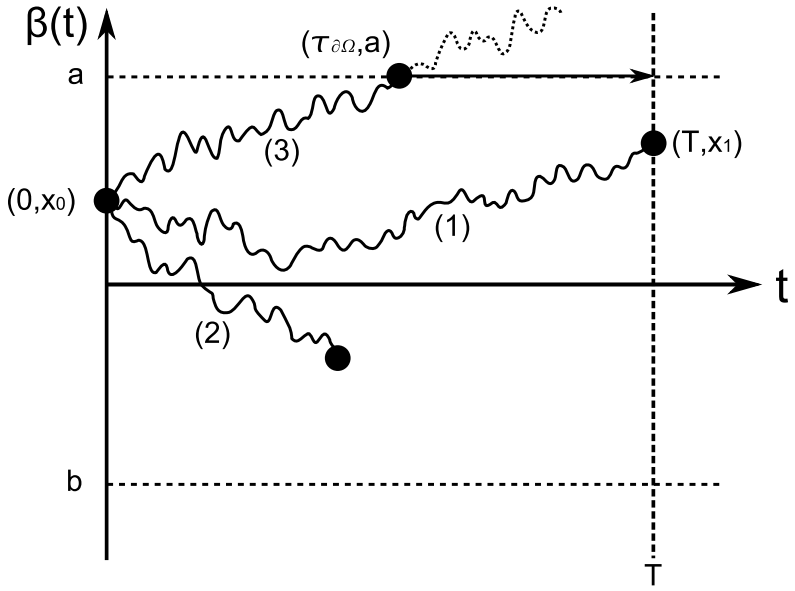
subject to the initial value $u(x,0) = f(x)$, for $-\infty < x < +\infty$. Here the point $x_i(\omega,t)$ is the position of the ith stochastic process surviving at time t, and $k(\omega)$ is the random number of branches at time t.

Solving PDEs with “random trees”

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - cu, \quad a < x < b,\ t > 0$$

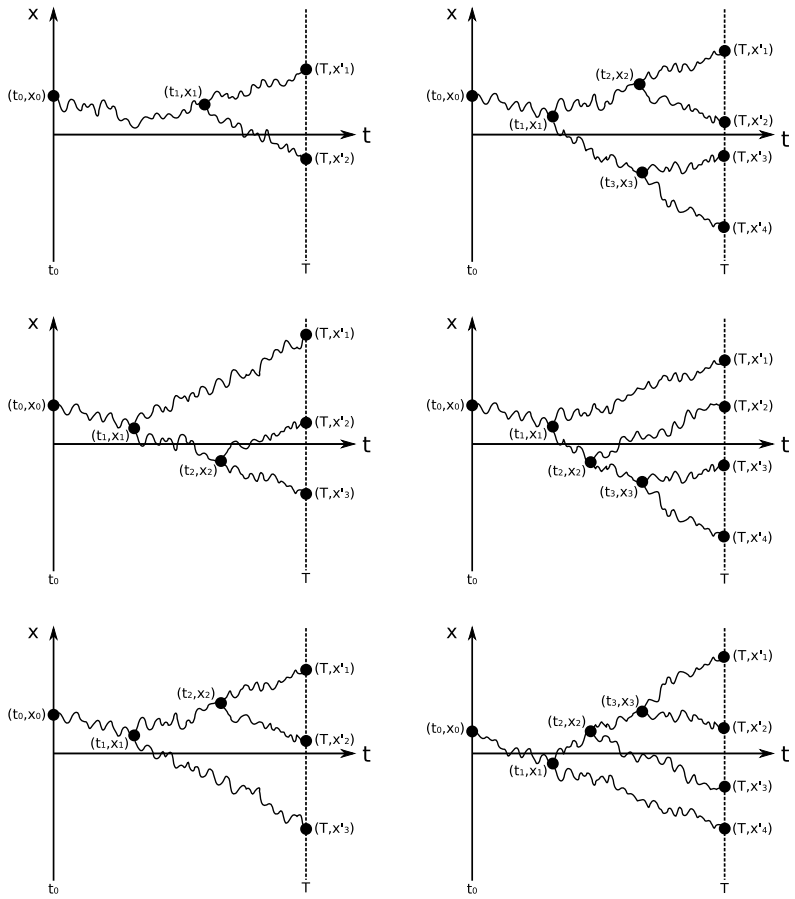
$$u(a,t) = 0,\ u(b,t) = 0$$

$$u(x,0) = f(x).$$



$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - cu + u^2, \quad -\infty < x < +\infty,\ t > 0$$

$$u(x,0) = f(x).$$



General semilinear parabolic PDEs

$$\frac{\partial u}{\partial t} = Lu + \sum_{j=2}^{m} c_j(x,t)u^j, \ x \in \mathbf{R}^n, \ t > 0$$

$$u(x,0) = g(x),$$

- Generalizes the previous representation, since it accounts for the following aspects:
  1. A constant potential term such as $-cu$ is not required anymore;
  2. The coefficients multiplying the nonlinear terms, $c_j(x,t)$, can be now chosen arbitrarily
  3. The initial data $g(x)$ may now be chosen negative, or greater than 1
- The set of all branches of a given “tree” play the role of the single path (or realization) of the stochastic processes used in the linear case. An average is now taken over all trees whose “root” is the space point x.

New Strategies: FBSDEs

- The forward-backward differential stochastic differential equations and its connection with PDEs was recently proposed by Pardoux ('90) as a kind of generalization of the Feynman-Kac formula for nonlinear parabolic PDEs.
- Assume the function u is sufficiently regular and solves,

$$\begin{cases} u_t(t,x) + Lu(t,x) + f\big(t,x,u(t,x),(\sigma\nabla u)(t,x)\big) &= 0 \\ u(T,x) &= g(x), \end{cases} \tag{1}$$

the solution can be obtained through the following forward-backward stochastic differential equation for $0 \le t \le s \le T$

$$\begin{cases} X_s^{t,x} &= x + \int_t^s b(r, X_r^{t,x})dr + \int_t^s \sigma(r, X_r^{t,x})dW_r \\ Y_s^{t,x} &= g(X_T^{t,x}) - \int_s^T Z_r^{t,x} dW_r + \int_s^T f(r, X_r^{t,x}, Y_r^{t,x}, Z_r^{t,x})dr. \end{cases} \tag{2}$$

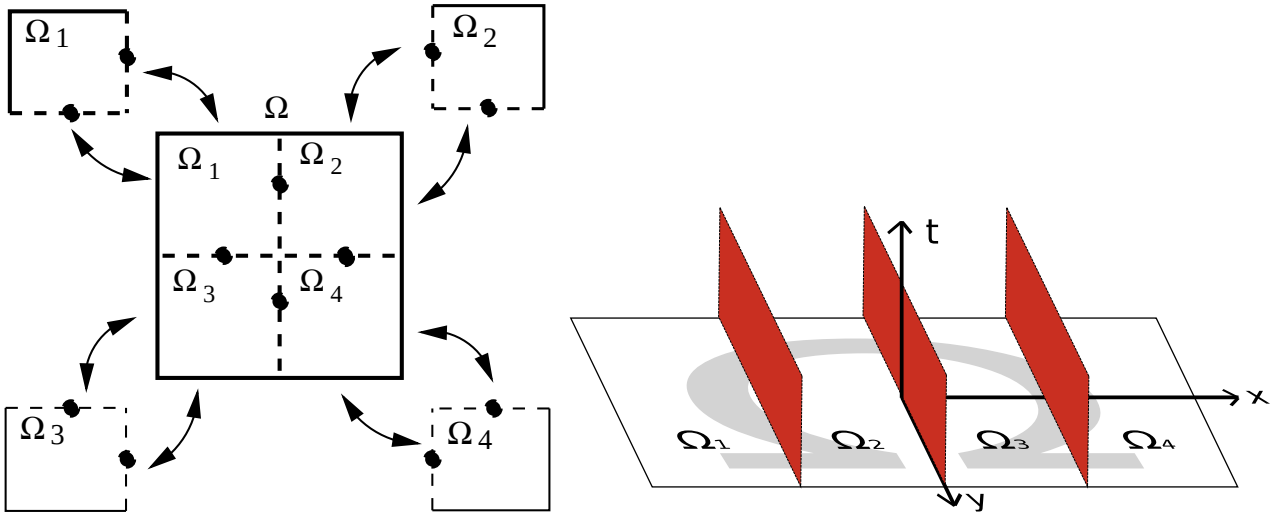
in the following way:

$$u(t,x) = Y_t^{t,x}, \quad t \in [0,T],\ x \in \mathbf{R}^d. \tag{3}$$

Numerical solution of PDEs probabilistically

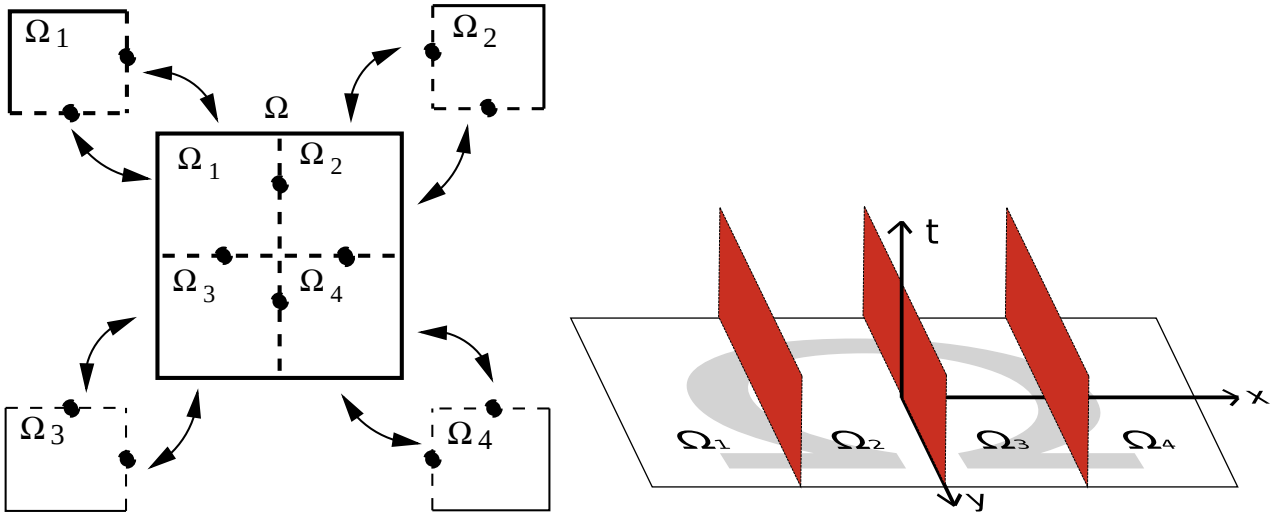
- **Advantages:**
  - gridless method;
  - solution obtained through an average of independent calculations;
  - suited for Parallel computing
  - naturally fault-tolerant
- **Disadvantages:** The idea of representing and even computing solutions to PDEs via a probabilistic method is old. The latter has been considered rather inefficient (Monte Carlo), at least in low dimension.
- **Key idea**: Implement a “**Probabilistically [or quasi-prob] Domain Decomposition method** (PDD)".

The algorithm PDD



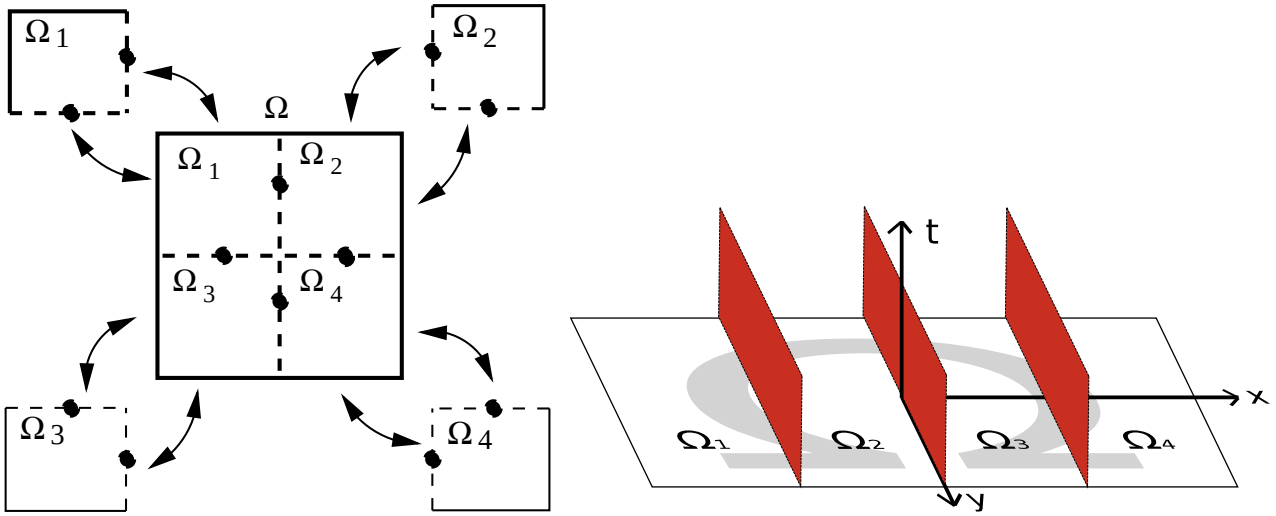
- Compute only *few* interfacial values by standard Monte Carlo or quasi–Monte Carlo

The algorithm PDD



- Compute only *few* interfacial values by standard Monte Carlo or quasi–Monte Carlo
- Interpolate on the corresponding nodes to obtain BVs for the subdomains

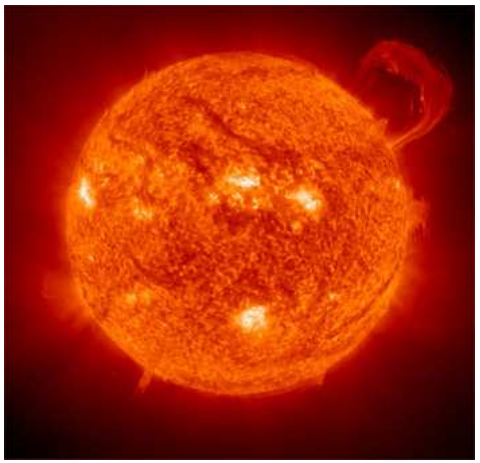
The algorithm PDD

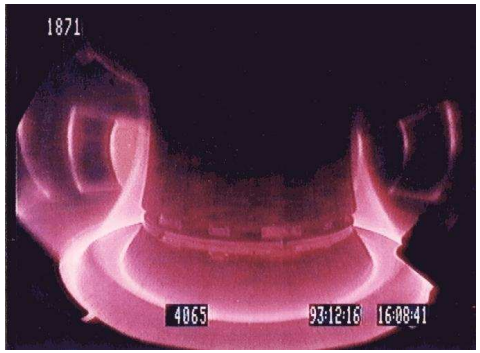


- Compute only *few* interfacial values by standard Monte Carlo or quasi–Monte Carlo
- Interpolate on the corresponding nodes to obtain BVs for the subdomains
- Compute the solution to the original problem in each subdomain, by standard methods (finite differences, or finite elements)

Plasma dynamics

- A plasma is a gas in which an important fraction of the atoms is ionized (the electrons and ions are separately free), because the temperature is hot enough.





- The dynamics of a thermonuclear plasma can be correctly described by the Vlasov equation. This is a transport equation describing time evolution of the distribution function of plasma consisting of charged particles with long-range (for example, Coulomb) interaction, and where the collisions have been neglected.
- Ignoring the magnetic field, the Vlasov equation should be coupled with the Poisson equation.

$$\frac{\partial f}{\partial t} + \bar{v} \cdot \nabla_{\bar{x}} f - \nabla \phi \cdot \nabla_{\bar{v}} f = 0, \quad \triangle_{\bar{x}} \phi = - \left[\int f d\bar{v} - 1 \right],$$

Plasma dynamics: Numerical methods

- Nowadays, there are basically two type of methods for solving the Vlasov-Poisson equation
  1. *Particle-in-Cell (PIC) simulations*. The simulation particles can be regarded as Lagrangian markers embedded randomly in the Vlasov fluid moving with it through phase space, and are simulated using a Monte-Carlo method.

     Advantages: It uses only 3D spatial grids for charge calculation, which is convenient for massively parallel computation.

     Disadvantages: To avoid large statistical error and be able to "see" the physics behind the phenomenon, many particles may be required.
  2. *Vlasov continum simulation*. Direct numerical integration of the system equations.

     Advantages: It has no statistical error.

     Disadvantages: It uses 3D spatial grids + 3D velocity space grids, which is harder for massively parallel computation. Since the dynamics of the plasma generates "ripples" in the phase-space for long times, very fine grids are required.

Plasma dynamics: Computational cost

- ITER is a large-scale scientific experiment intended to prove the viability of fusion as an energy source, and to collect the data necessary for the design and subsequent operation of the first electricity-producing fusion power plant. Currently, it is being built in Cadarache (France).

- To simulate the dynamics of ITER for a typical experimental over scales of interest with the most commonly used algorithmic technologies would require approximately 10^{24} floating-point operations (10^8 times higher than the most powerful supercomputer today)

- With the best forecast in performance, we should wait until 50 years (very optimistic estimation) to be able to simulate globally ITER (It should be operational in 10 years).

Computational complexity-probabilistic part

- The branching stochastic process associated to the nonlinear term u^j, requires creating j branches every time a splitting event occurs.
- The computational time spent to generate any given branch, is a function of the final time, t, time step, Δt, chosen to solve numerically the associated SDE, and the random times Δt_s responsible for branching.
- It is measured, typically, in terms of the number of iterations in time, required to fully generate a random tree with k branches up to the final time, t. Defining t_c as the time spent per iteration, such computational time can be estimated as $kt_c\, t/\overline{\Delta t_s}$
- In case of N random trees, it holds that

$$t_b = N \sum_{k=1}^{\infty} kt_c \frac{t}{\overline{\Delta t_s}} P(k,m),$$

Computational complexity-probabilistic part

- Known the probability funcion $P(k,m)$, it follows that

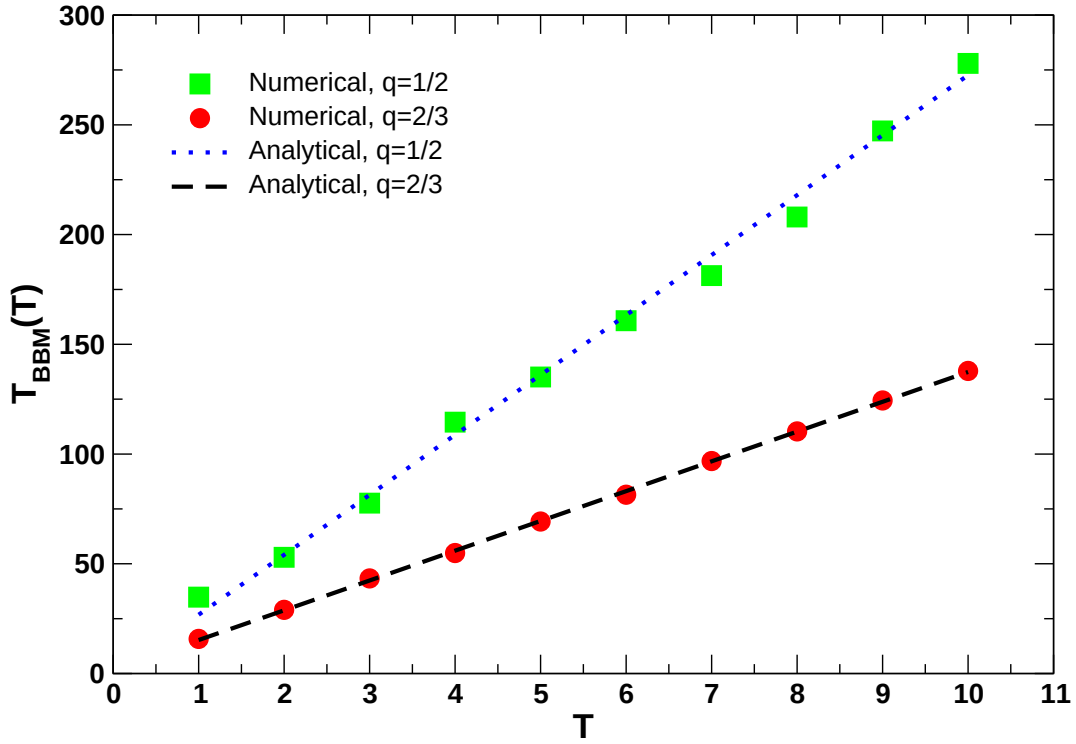
$$t_b \leq N t_c \frac{t}{\Delta t_s} \langle k \rangle_{P(k,m)} ,$$

where $\langle k \rangle_{P(k,m)}$ denotes the mean number of leaves, that is $\sum_{k=1}^{\infty} kP(k,m)$.

$$t_b \leq N t_c \frac{t}{\Delta t_s} \frac{q}{1-m(1-q)}.$$

Computational complexity-probabilistic part

- Note that the computational time exhibits a linear growth on t,



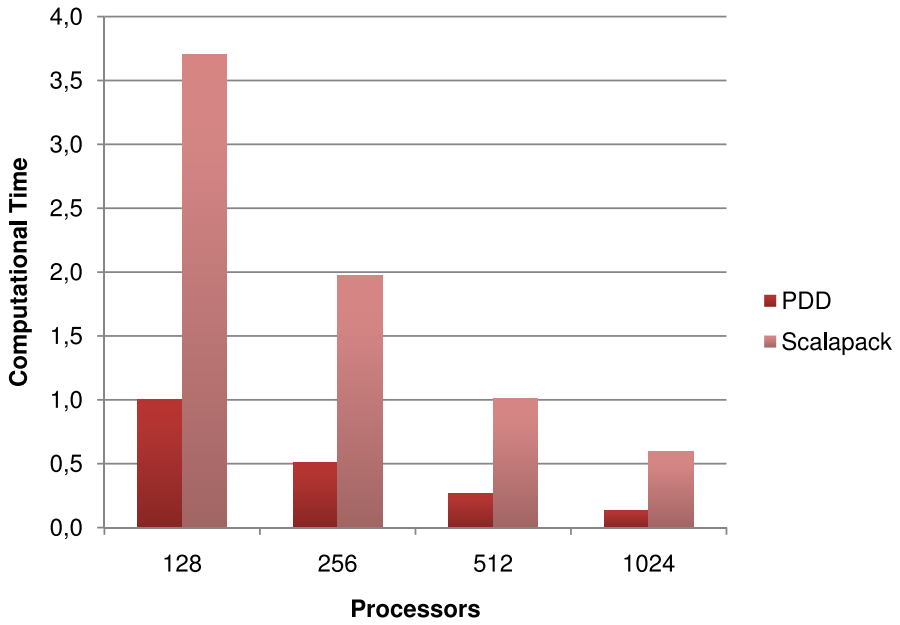
Numerical examples

- Some examples were runned on the MareNostrum supercomputer located at the Barcelona Supercomputing Center (BSC).
- Simple domains were used, and a finite difference scheme adopted.
- ScaLAPACK were used for comparison, which is considered reasonably efficient for the parallel solution of banded linear systems.
- The PDD algorithm was implemented in MPI environment. Local solver for each subdomain was a LU decomposition based on LAPACK.

Examples in 1D

$$u_t = u_{xx} - u + u^2, \quad u(x,0) = 1 - \frac{1}{\left(1 + \exp \frac{x}{\sqrt{6}}\right)^2}$$

on the line. This problem is known to possess the traveling wave solution $u(x,t) = 1 - \frac{1}{\left(1+\exp \frac{x - \frac{5}{\sqrt{6}} t}{\sqrt{6}}\right)^2}$.



Comparison of the computational times (measured in units of t_0) for both methods, PDD and SCALAPACK.

Examples in 2D

$$u_t = u_{xx} + u_{yy} - (1+a)u^2 - u^3, \quad (x,y) \in \mathbf{R}^2,$$
$$u(x,y,0) = -2\cos^2(\frac{\pi\, x}{2A_x})\cos^2(\frac{\pi\, y}{2A_y}).$$

Procs.	T_{MC}	T_{INT}	*Memory*	$\mathbf{T_{PDD}}$	$\mathbf{T_{ScaLAPACK}}$
128	902”	<1”	0.86 GBs	5572”	29881”
256	998”	<1”	0.19 GBs	2086”	23953”
512	1018”	<1”	0.06 GBs	1327”	23334”

Examples in 2D

$$\frac{\partial u}{\partial t} = (1+x^2)\frac{\partial^2 u}{\partial x^2} + (1+y^3)\frac{\partial^2 u}{\partial y^2}$$
$$+(\sin x\, e^y + 2)\frac{\partial u}{\partial x} + (\sin x\, \cos y + 2)\frac{\partial u}{\partial y} - u + \frac{1}{2}u^2 + \frac{1}{2}u^3,$$
$$\text{in } \Omega = [-L, L] \times [-L, L],\ t > 0,$$

with $L = 1$ and BV conditions $u(x, y, t)|_{\partial\Omega} = 0$, $u(x, y, 0) = \cos^2\left(\frac{\pi x}{2L}\right)\cos^2\left(\frac{\pi y}{2L}\right)$.

Number of processors	PDD		
	T_{MC}	T_{INTERP}	$\mathbf{T_{TOTAL}}$
128	1' 05"	<1"	209' 44"
256	1' 05"	<1"	53' 10"
512	1' 06"	<1"	12' 13"
1024	1' 08"	<1"	3' 59"

Solving Vlasov-Poisson with “random trees”

- Consider the adimensional Vlasov-Poisson equation in $\mathbf{R}^d$,

$$\frac{\partial f}{\partial t} + \bar{v} \cdot \nabla_{\bar{x}} f - \nabla \phi \cdot \nabla_{\bar{v}} f = 0, \quad \triangle_{\bar{x}} \phi = -\left[\int f d\bar{v} - 1 \right],$$

with initial data, $f(\bar{x}, \bar{v}, 0)$, and periodic boundary conditions in $\bar{x}$. A stochastic representation for the Fourier-transformed equation is given by

$$\chi(\bar{\xi}_1, \bar{\xi}_2, \tau) = e^{-\lambda\tau} \chi(\bar{\xi}_1, \bar{\xi}_2 - \tau \frac{\bar{\xi}_1}{\gamma(|\bar{\xi}_2|)}, 0) + \frac{|\bar{\xi}_1'|^{-1} h * h(\bar{\xi}_1)}{\lambda h(\bar{\xi}_1)}$$

$$\times \int_0^\tau ds \lambda e^{-\lambda s} \sum_j d\bar{\xi}_{1(j)}' p(\bar{\xi}_1, \bar{\xi}_{1(j)}') \frac{\bar{\xi}_{1(j)}' \cdot (\bar{\xi}_2 - s \frac{\bar{\xi}_1}{\gamma(|\bar{\xi}_2|)})}{\gamma(|\bar{\xi}_2 - s \frac{\bar{\xi}_1}{\gamma(|\bar{\xi}_2|)}|)|\bar{\xi}_{1(j)}'|}$$

$$\times [(2\pi)^{d/2} e^{\lambda(\tau - s)} \chi(\bar{\xi}_{1(j)}', \mathbf{0}, \tau - s) - \frac{\hat{\rho}_B(\bar{\xi}_{1(j)}')}{h(\bar{\xi}_{1(j)}')}] \chi(\bar{\xi}_1 - \bar{\xi}_{1(j)}', \bar{\xi}_2 - s \frac{\bar{\xi}_1}{\gamma(|\bar{\xi}_2|)}, \tau - s),$$

with $\chi = F\, e^{-\lambda\tau} / h(\bar{\xi}_1)$, $|\bar{\xi}_1'|^{-1} h * h(\bar{\xi}_1) = \int d\bar{\xi}_1' |\bar{\xi}_1'|^{-1} h(\bar{\xi}_1 - \bar{\xi}_1') h(\bar{\xi}_1')$, and

$$p(\bar{\xi}_1, \bar{\xi}_1') = \frac{|\bar{\xi}_1'|^{-1} h(\bar{\xi}_1 - \bar{\xi}_1') h(\bar{\xi}_1')}{|\bar{\xi}_1'|^{-1} h * h(\bar{\xi}_1)}.$$

Vlasov-Poisson equation

- There exists a probabilistic interpretation of the eq. above as an exponential random process along with a branching process governed by $p(\bar{\xi_1}, \bar{\xi_1}')$.
- $\chi(\bar{\xi_1}, \bar{\xi_2}, \tau)$ can be computed as the expectation value of a multiplicative functional associated to the processes.
- Convergence of the multiplicative functional requires:

$$\text{(A)} \quad \left| \frac{F(\bar{\xi_1}, \bar{\xi_2}, 0)}{h(\bar{\xi_1})} \right| \leq 1$$

$$\text{(B)} \quad \left| \frac{\rho(\bar{\xi_1})}{(2\pi)^{d/2} h(\bar{\xi_1})} \right| \leq 1$$

$$\text{(C)} \quad |\bar{\xi_1}'|^{-1} h * h(\bar{\xi_1}) \leq h(\bar{\xi_1})$$

- The condition (C) for $d = 2, 3$ can be satisfied choosing as kernel

$$h(\bar{\xi_1}) = \frac{c}{(1 + |\bar{\xi_1}|^2)^2}$$

Vlasov-Poisson equations

Let consider the 1-dimensional Vlasov-Poisson system of equations

$$\partial_t f + v \cdot \partial_x f + E(x,t) \cdot \partial_v f = 0,$$
$$\partial_x E(x,t) = -\partial_{xx}\phi(x,t) = \int_{\mathbf{R}} f \, dv - 1 \tag{4}$$

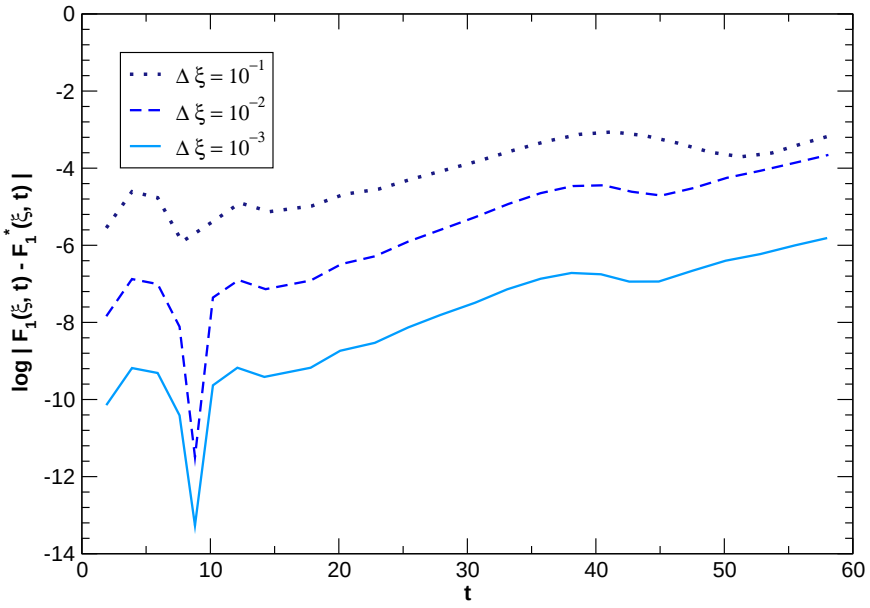
Let assume periodic boundary conditions in the space dimension, and $f \to 0$ as $|v| \to \infty$.

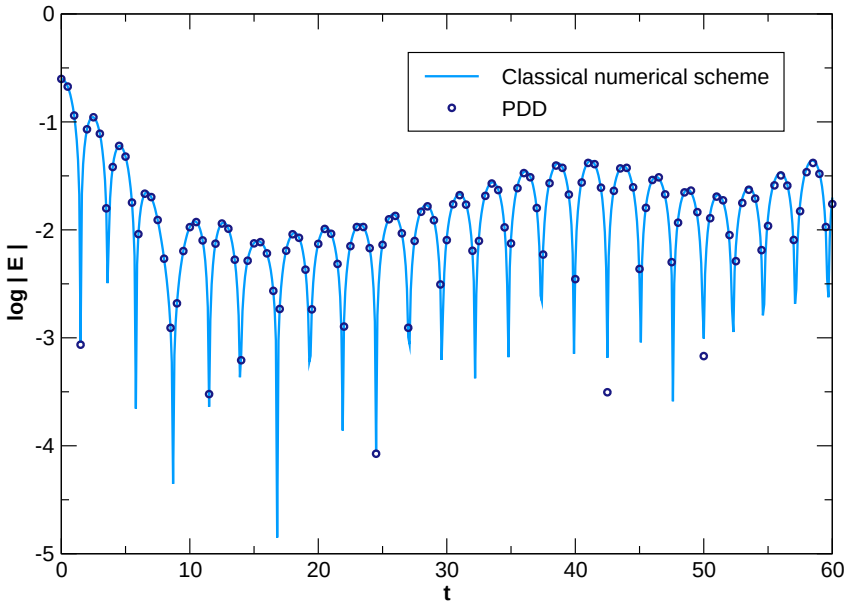
$$\begin{aligned}
\hat{f}(k,\xi,t) &= \hat{f}(k, \xi + t\frac{2\pi}{L}\, k, 0) \\
&- \int_0^t ds \sum_{\substack{k'=-\infty \\ k' \neq 0}}^{\infty} p(k') \frac{(\pi^2/3)\, k' \, (\xi + (t-s)\frac{2\pi}{L}\, k)}{2\pi/L} \\
&\times \hat{f}(k-k', \xi + (t-s)\frac{2\pi}{L}\, k, s) \left[\hat{f}(k', 0, t) - \delta(k') \right]
\end{aligned} \tag{5}$$

Vlasov-Poisson equations

Strong landau damping: Only one species is considered, boundary conditions are 2π-periodic in $\bar{x}$, and the initial condition is

$$f(\bar{x}, \bar{v}, 0) = (1/2\pi)^{d/2} \exp(-\bar{v}^2/2)[1 + A\cos(k_1 x)]$$

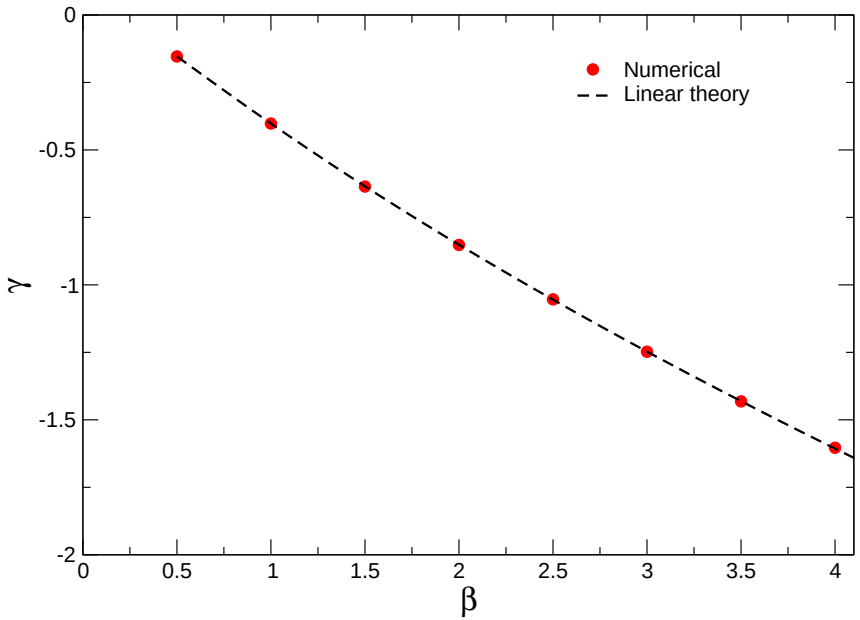


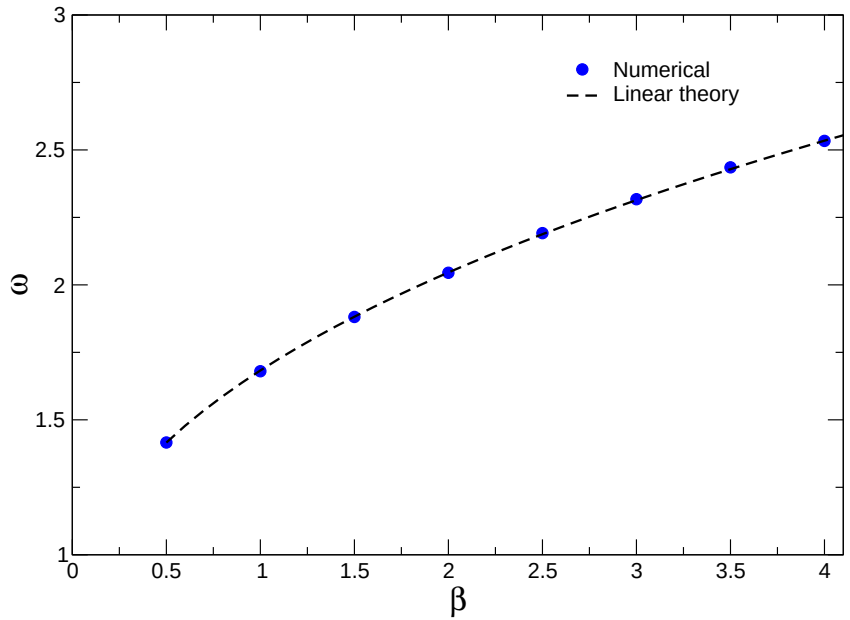


Numerical error and solution. Parameters: $A = 0.5$.

Vlasov-Poisson equations

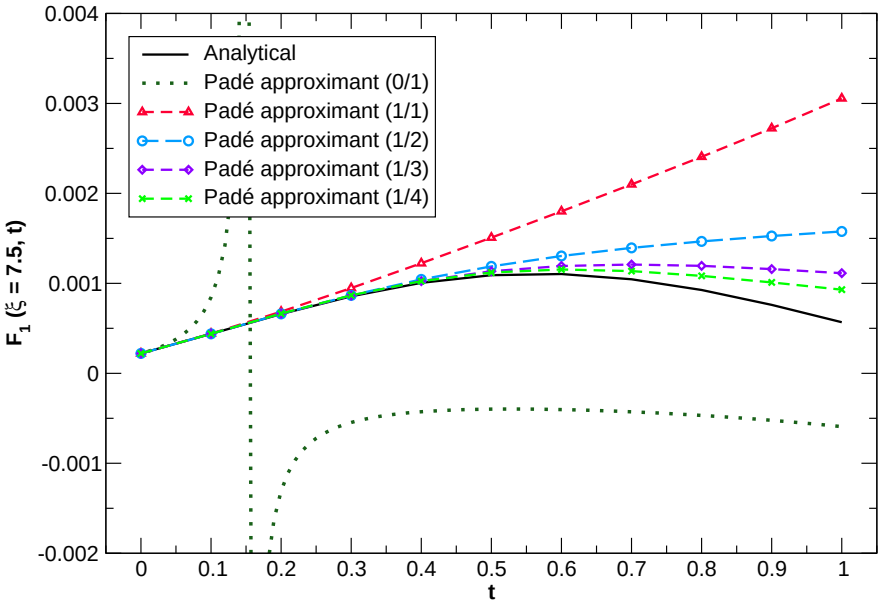
Linear landau damping ($A = 0.01$).

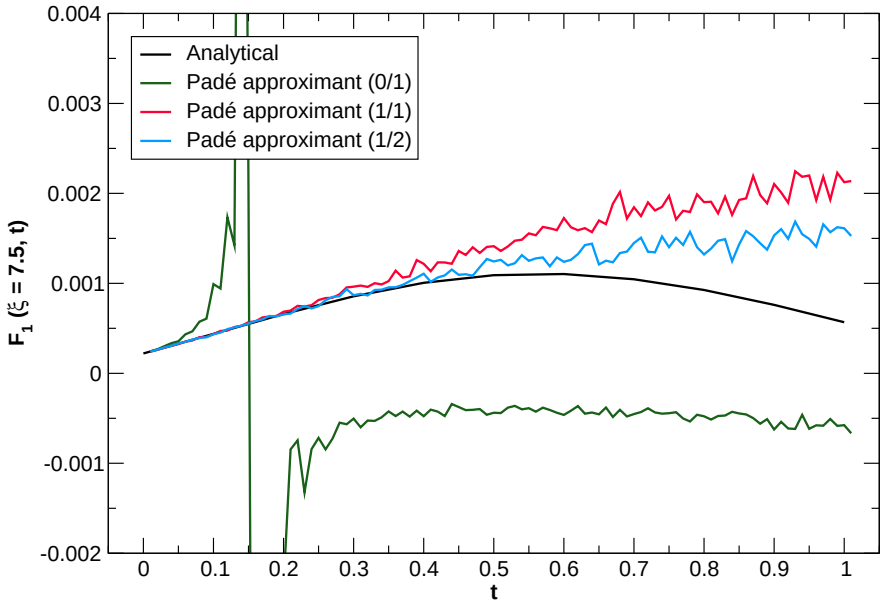


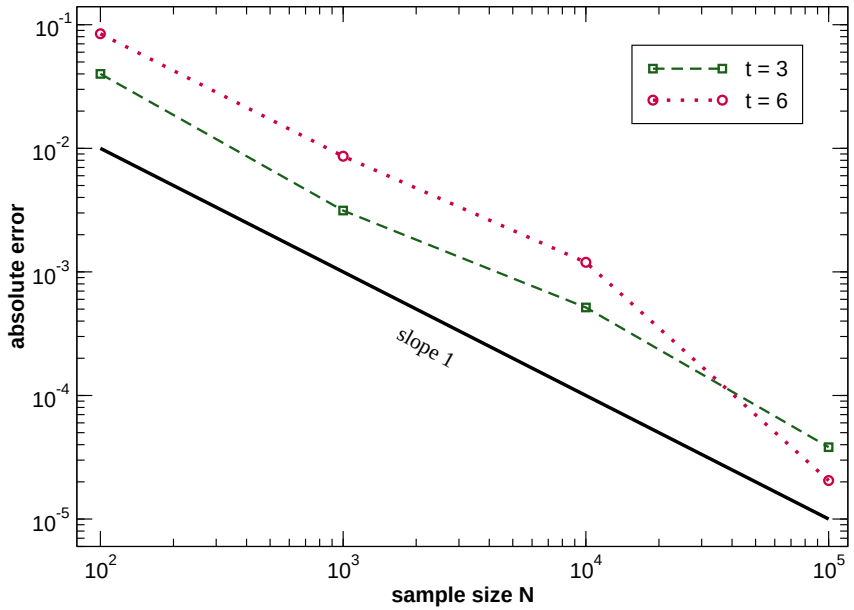


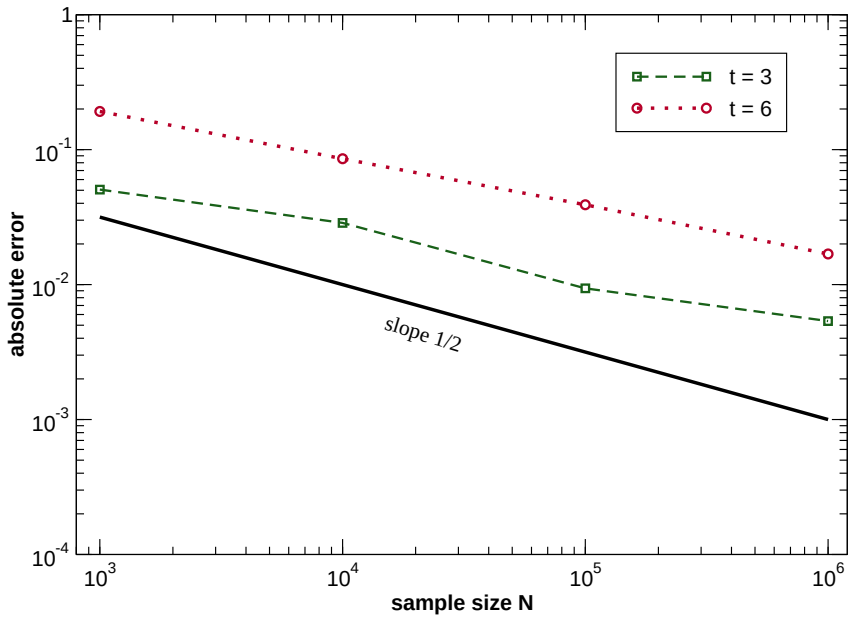
Comparison with linear theory, as a function of the temperature β.

Vlasov-Poisson equations

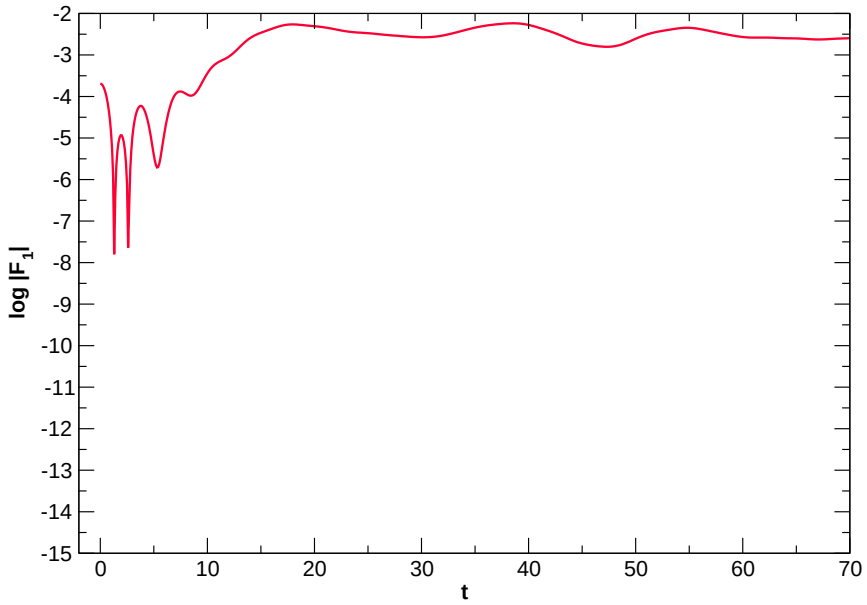


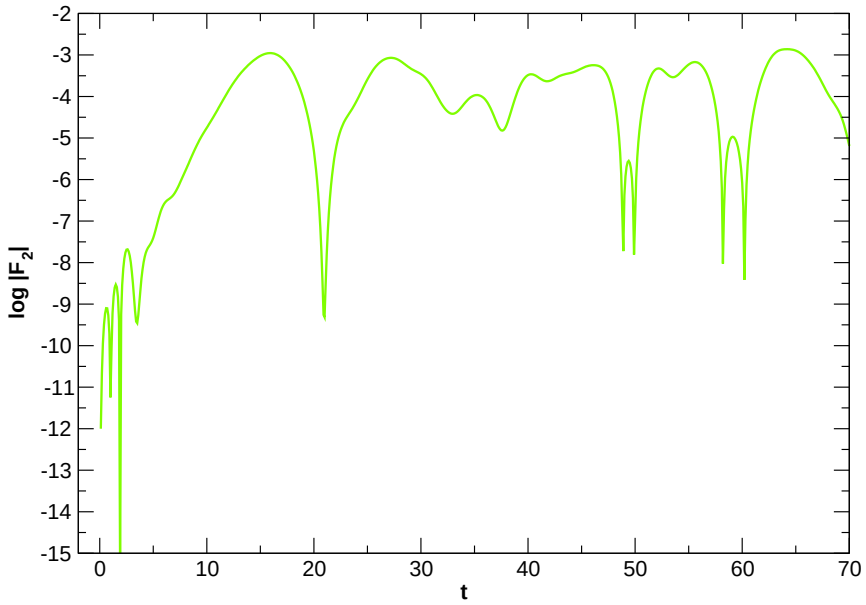


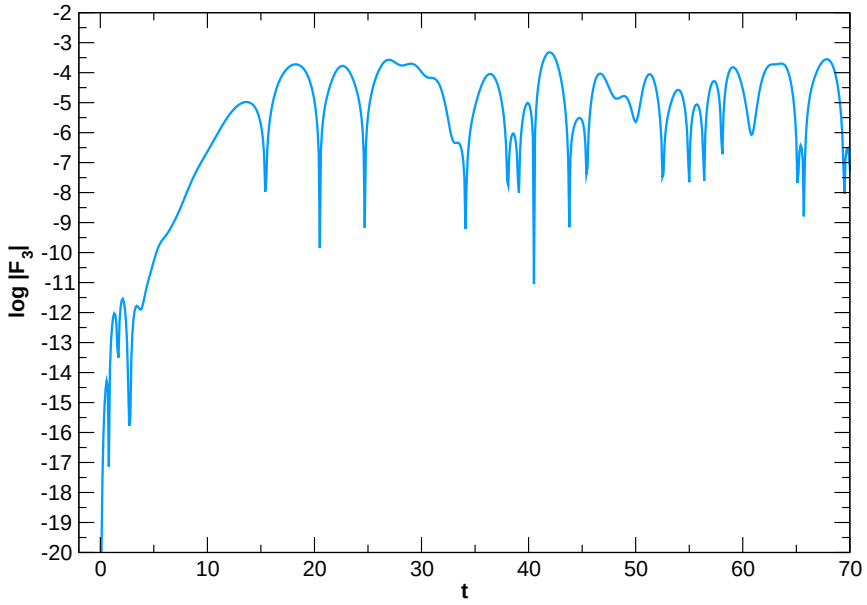


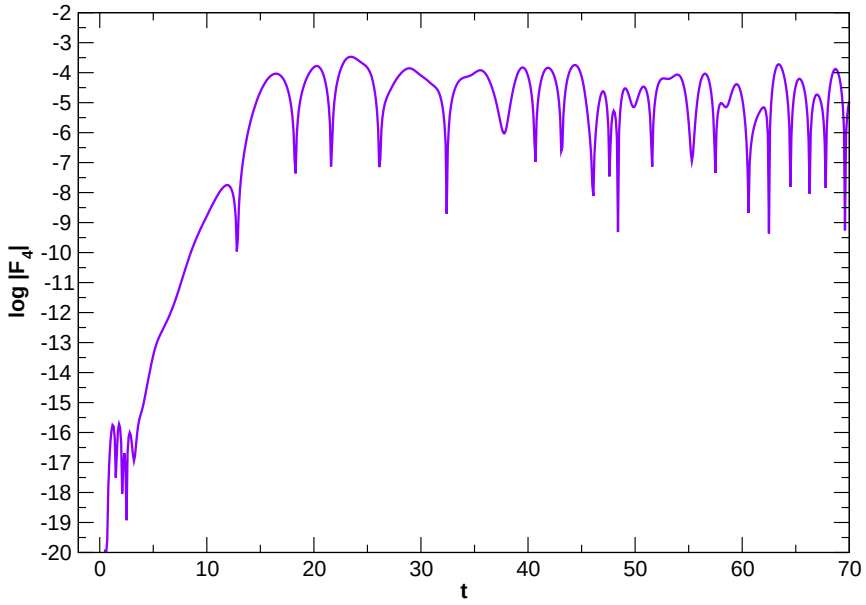


Vlasov-Poisson equations









Vlasov-Poisson equations

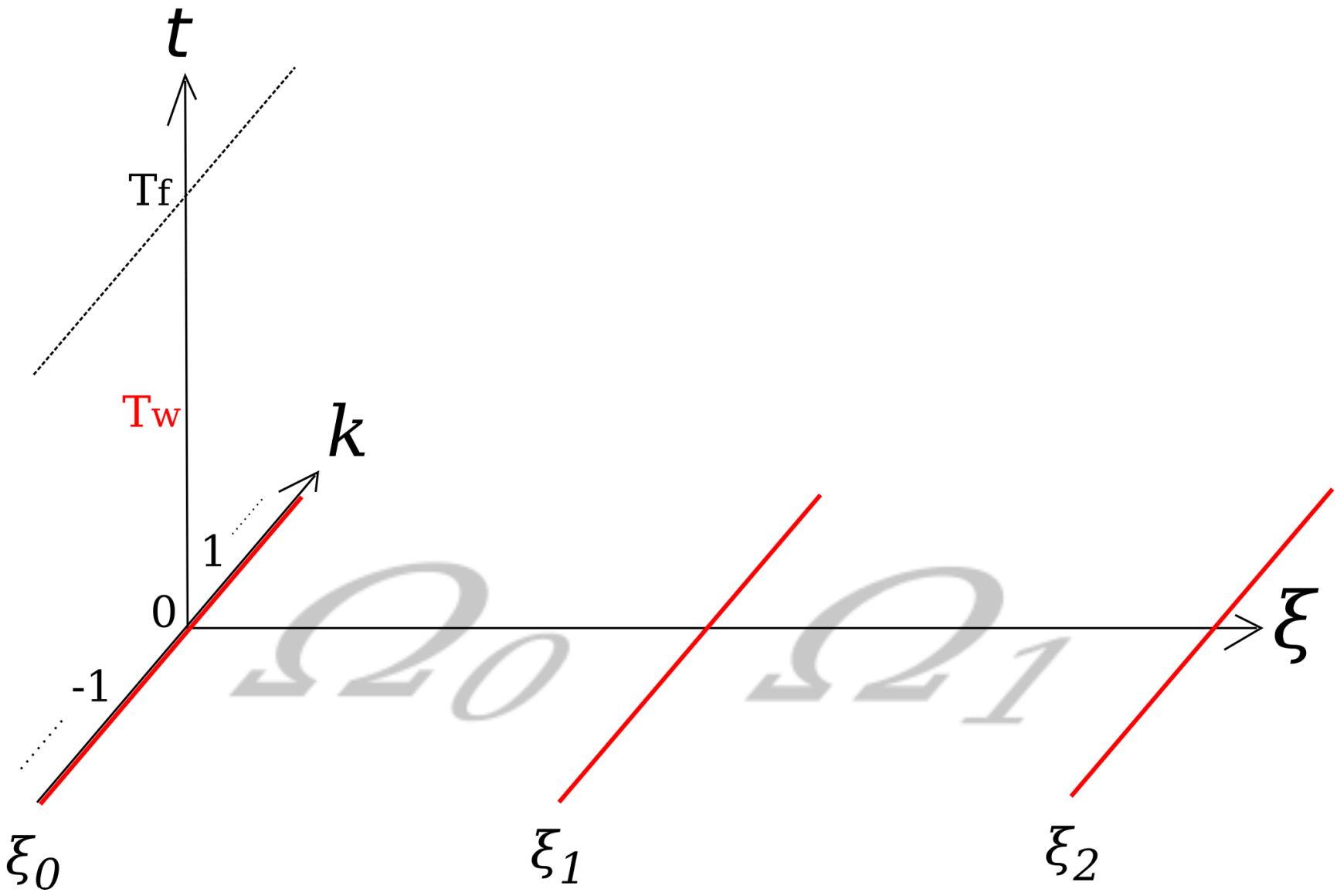


Figure 1: Computational domain

Vlasov-Poisson equations

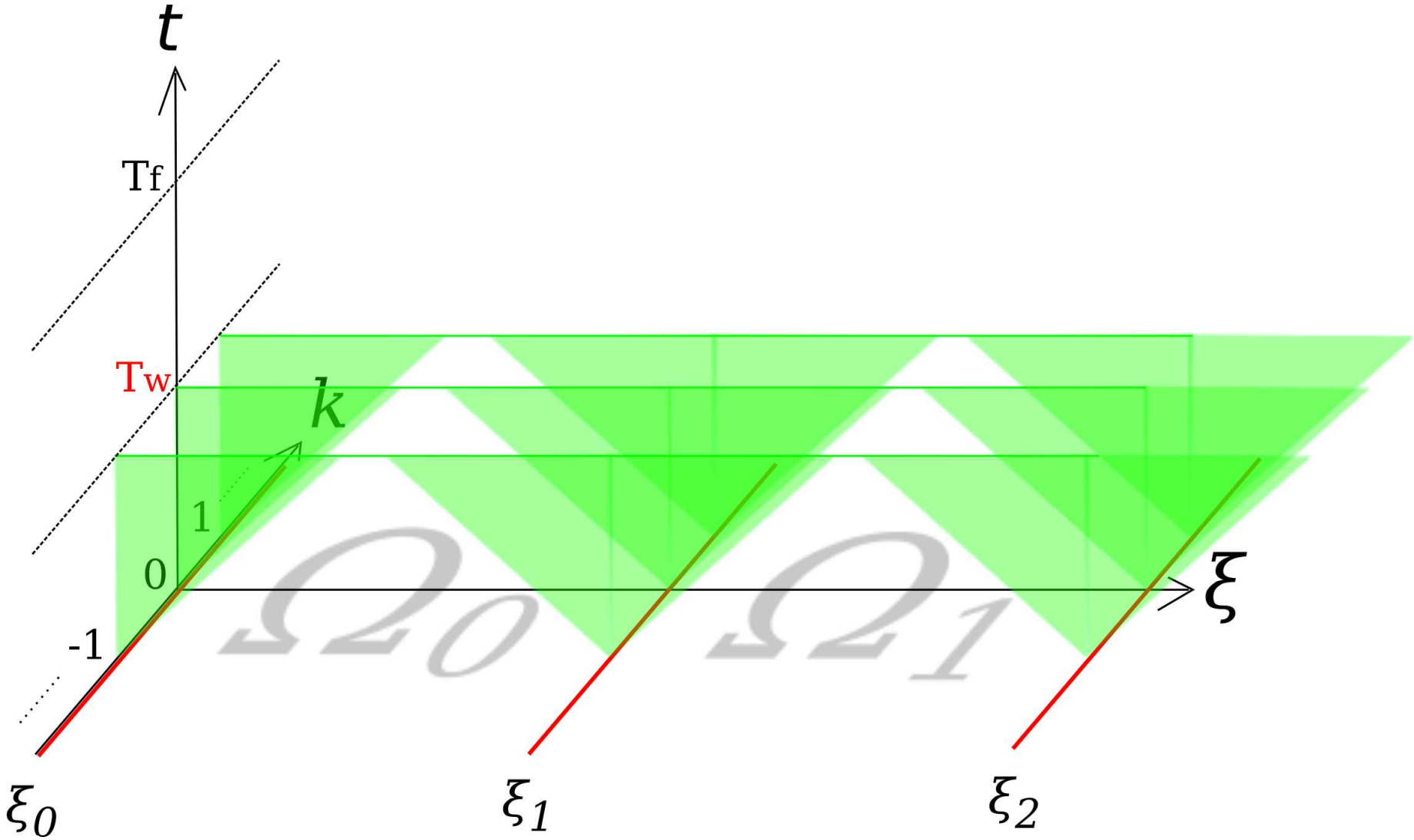


Figure 2: Computational domain

Vlasov-Poisson equations

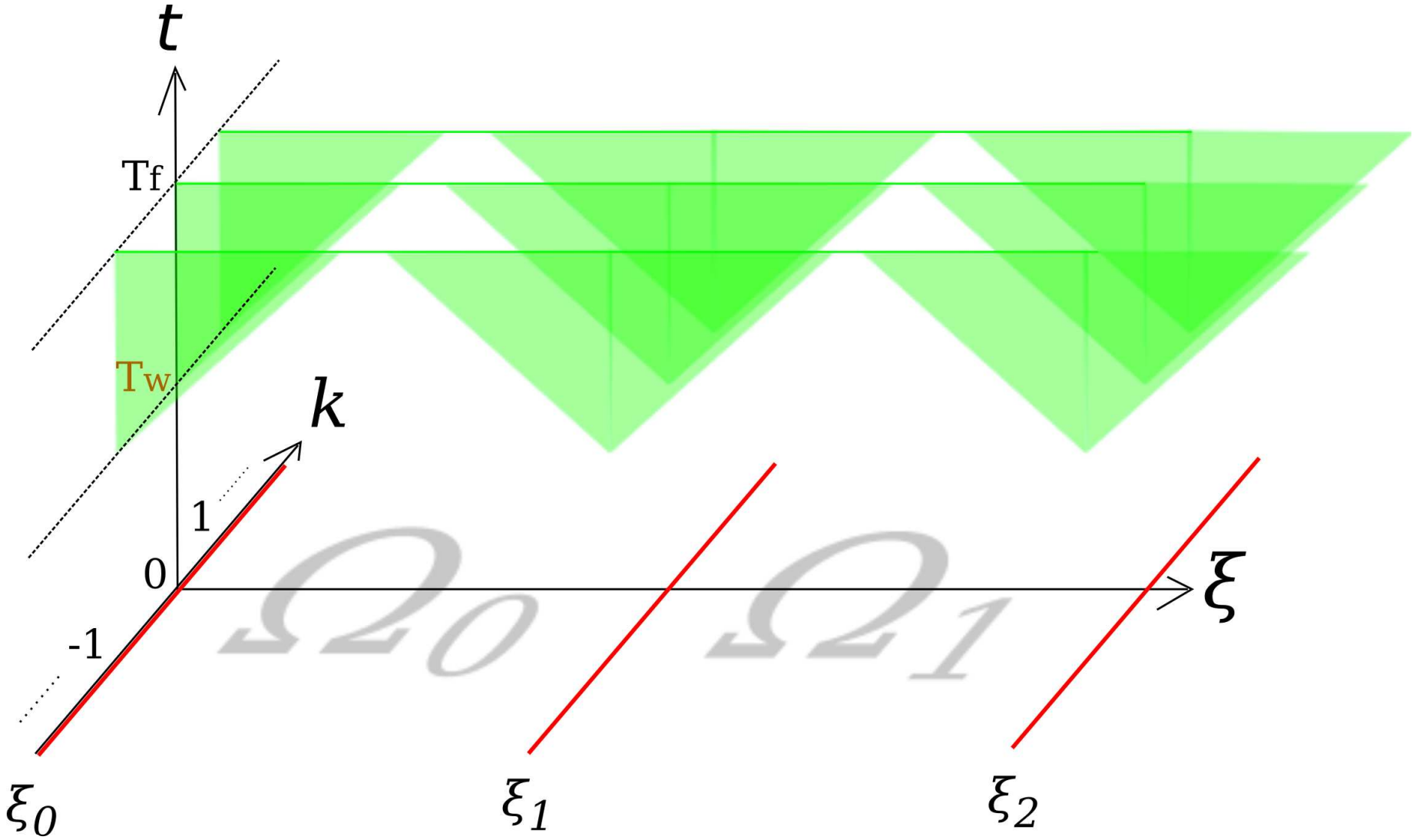


Figure 3: Computational domain

Numerical examples

- Numerical simulations for Vlasov-Poisson in 1D were runned on the Matrix supercomputer located at the Rome Supercomputing Center (CASPUR). The supercomputer is equipped with $2,048$ processors linked among them by an infiniband interconnection network.
- Comparison is done with a parallelized upwind numerical scheme.
- Only one species is considered, boundary conditions are 2π-periodic in $\bar{x}$, and the initial condition is

$$f(\bar{x},\bar{v},0) = (1/2\pi)^{d/2}\,\exp\left(-\bar{v}^2/2\right)[1 + A\,\cos(k_1 x)]$$

Vlasov-Poisson in 1D

Procs.	T_{PDD}
1	3115”
2	1581”
4	775”
8	368”
16	180”
32	92”
64	46”
128	24”
256	13”
512	7”

Conclusions

- Monte Carlo and Domain Decomposition allow for a “double” parallelization.

Conclusions

- Monte Carlo and Domain Decomposition allow for a “double” parallelization.
- Stochastic differential equations can be efficiently computed resorting to sequences of quasi-random numbers (low-discrepancy sequences).

Conclusions

- Monte Carlo and Domain Decomposition allow for a “double” parallelization.
- Stochastic differential equations can be efficiently computed resorting to sequences of quasi-random numbers (low-discrepancy sequences).
- The PDD method is fully scalable in contrast with the classical DD.

Conclusions

- Monte Carlo and Domain Decomposition allow for a “double” parallelization.
- Stochastic differential equations can be efficiently computed resorting to sequences of quasi-random numbers (low-discrepancy sequences).
- The PDD method is fully scalable in contrast with the classical DD.
- In addition, it appears that the algorithm is also naturally **fault tolerant**.

Conclusions

- Monte Carlo and Domain Decomposition allow for a “double” parallelization.
- Stochastic differential equations can be efficiently computed resorting to sequences of quasi-random numbers (low-discrepancy sequences).
- The PDD method is fully scalable in contrast with the classical DD.
- In addition, it appears that the algorithm is also naturally **fault tolerant**.
- Nowadays, it is becoming not only important to design and exploit parallel algorithms, but also to be able to handle possible failure of a certain number of processors. In some case, even the failure of a single processor might stop or ruine the entire computation.

Conclusions

- Monte Carlo and Domain Decomposition allow for a “double” parallelization.
- Stochastic differential equations can be efficiently computed resorting to sequences of quasi-random numbers (low-discrepancy sequences).
- The PDD method is fully scalable in contrast with the classical DD.
- In addition, it appears that the algorithm is also naturally **fault tolerant**.
- Nowadays, it is becoming not only important to design and exploit parallel algorithms, but also to be able to handle possible failure of a certain number of processors. In some case, even the failure of a single processor might stop or ruine the entire computation.
- This algorithm is fully exempt from such disease, and the failure of a percentage of the in-use processors only affects the error, but does neither imply any stop of the entire process, nor produce false results.

Future work

- Monte Carlo for large scale linear algebra problems?

Future work

- Monte Carlo for large scale linear algebra problems?
- Of course, but again, **hybrid ideas**!