SciML 2: advanced methods for PDEs in physics

E. Franck

Master CSMI, Strasbourg University

01/09/2025

1. Advanced time methods, nonlinear approximation

Standard Methods

- We consider the following space

$$V_n = \left\{ u_{\boldsymbol{\theta}}(t, \mathbf{x}), \quad \text{such that } u_{\boldsymbol{\theta}}(t, \mathbf{x}) = \sum_{i=1}^{n} \theta_i(t) \varphi_i(\mathbf{x}) \right\}$$

- We are given a time grid $(t_0, ..., t_n)$ and we assume that we know $u_{\boldsymbol{\theta}}(t^n, \mathbf{x})$ and therefore $\theta(t^n)$.
- How to calculate $u_{\boldsymbol{\theta}}(t^{n+1}, \mathbf{x})$. We incorporate the model into the PDE and use an Euler approximation which gives:

$$u_{\boldsymbol{\theta}}(t^{n+1}, \mathbf{x}) \approx u_{\boldsymbol{\theta}}(t^n, \mathbf{x}) + \Delta t (\Delta u_{\boldsymbol{\theta}}(t^n, \mathbf{x}) + f(t^n, \mathbf{x}))$$

- We must project onto V_n our estimate of time t^{n+1} given by the Euler scheme to obtain $u_{\boldsymbol{\theta}}(t^{n+1}, \mathbf{x})$.
- We therefore have:

$$\theta(t^{n+1}) = \min_{\theta} \sum_{i}^{N} \int_{\Omega} |(\langle \boldsymbol{\theta}, \boldsymbol{\Phi} \rangle - R(\mathbf{x}))\psi_i(\mathbf{x}) \ |^2 \, d\mathbf{x}$$

with $R(\mathbf{x}) = (u_{\boldsymbol{\theta}}(t^n, \mathbf{x}) + \Delta t (\Delta u_{\boldsymbol{\theta}}(t^n, \mathbf{x}) + f(t^n, \mathbf{x})))$

- This gives:

$$M\theta(t^{n+1}) = M\theta(t^n) + \Delta t \mathbf{b}$$

Discrete PINNs

Remark: The idea of discrete PINNs is exactly the same. At each step we will make a projection onto V_n of the estimate of time t^{n+1} given by the numerical scheme.

- Discrete PINNs + Expilicit Euler:

$$\theta(t^{n+1}) = \min_{\theta} \sum_{i}^{N} \int_{\Omega} \left| \left(nn_{\theta}(\mathbf{x}) - R\left(\mathbf{x}; u_{\boldsymbol{\theta}_n}(\mathbf{x})\right) \right) \psi_i(\mathbf{x}) \right|^2 \, d\mathbf{x}$$

- Discrete PINNs + Implicit Euler:

$$\theta(t^{n+1}) = \min_{\theta} \sum_{i}^{N} \int_{\Omega} \left| (nn_{\theta}(\mathbf{x}) - R(\mathbf{x}; nn_{\theta}(\mathbf{x})))\psi_i(\mathbf{x}) \right|^2 \, d\mathbf{x}$$

- In the two case: $u_{\theta}(t^{n+1}, \mathbf{x}) = nn_{\theta(t^{n+1})}(\mathbf{x})$
- Each substep of a time scheme requires a projection and therefore training. Since the parameters evolving little between two steps each training is short.

Remark (Initial condition): We must also project the initial condition into space using training (often longer).

Neural Galerkin

- Let's take the discrete PINNs + explicit Euler?

$$\theta(t^{n+1}) = \min_{\theta} \sum_{i}^{N} \int_{\Omega} |(nn_{\theta}(\mathbf{x}) - u_{\boldsymbol{\theta}}(t^n, \mathbf{x}) - \Delta t \Delta u_{\boldsymbol{\theta}}(t^n, \mathbf{x}))\psi_i(\mathbf{x})\ |^2\ dx$$

- Since the θ sought will be close to $\theta(t^n)$ we can linearize the network around $\theta(t^n)$ and reduce to a simple problem:

$$nn_{\theta}(\mathbf{x}) = u_{\boldsymbol{\theta}}(t^n, \mathbf{x}) + (\nabla_{\theta} u_{\boldsymbol{\theta}}(t^n, \mathbf{x}))(\theta - \theta(t^n))$$

- we get a quadratic problem

$$\theta(t^{n+1}) = \min_{\theta} \sum_{i}^{N} \int_{\Omega} |((\nabla_{\theta} u_{\boldsymbol{\theta}}(t^n, \mathbf{x}))(\theta - \theta(t^n)) - \Delta t \Delta u_{\boldsymbol{\theta}}(t^n, \mathbf{x}))\psi_i(\mathbf{x})\ |^2\ dx$$

Result: The scheme is given by

$$M(\theta(t^n))\theta(t^{n+1}) = M(\theta(t^n))\theta(t^n) + \mathbf{b}(t^n)$$

with $N(\theta(t^n)) = \boldsymbol{\Psi}^T(\mathbf{x})(\nabla_{\theta} u_{\boldsymbol{\theta}}(t^n, \mathbf{x}))$, $\quad M = \int_{\Omega} NN^t dx$ and $\mathbf{b} = \int_{\theta} N\Psi^t(\mathbf{x})\Delta u_{\boldsymbol{\theta}}(t^n, \mathbf{x}))dx$

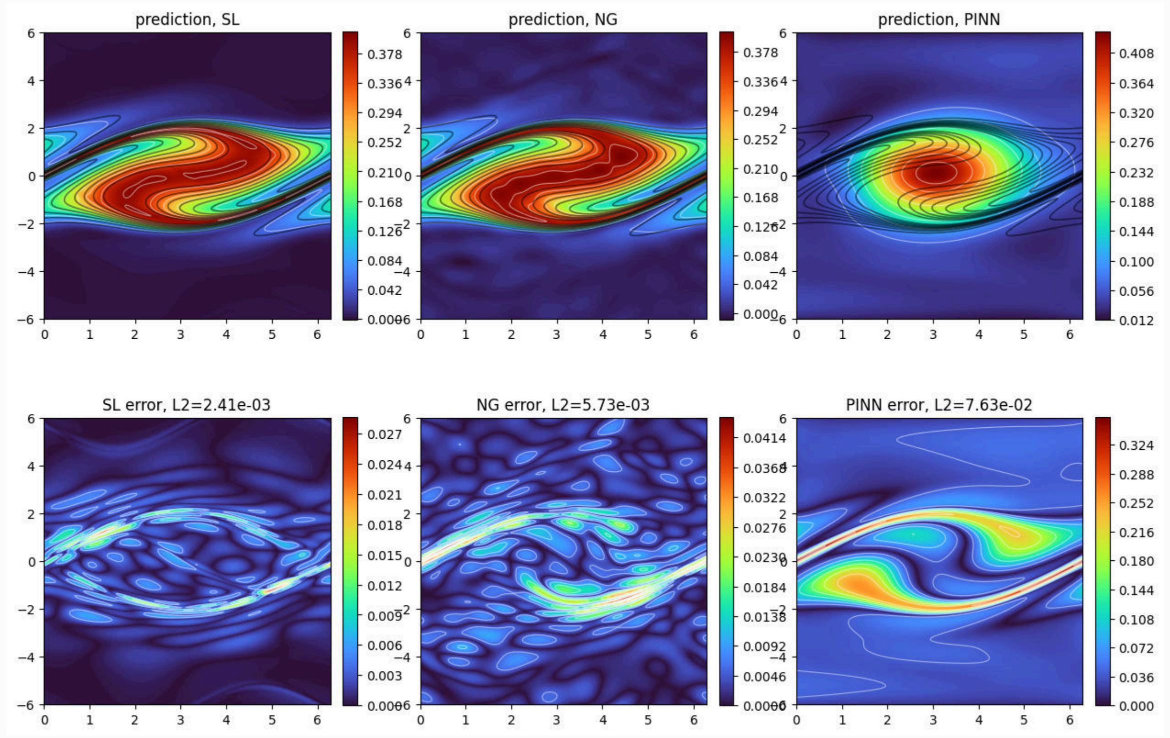
Comparison

- Example:

$$\partial_t f(t, x, v) + v\partial_x f + \sin(x)\partial_v f = 0$$

and $f(t = 0, x, v)$ a Gaussian in v.

- We compare a space-time PINNs, a Neural Galerkin method, a specially adapted discrete PINNs:



1.2. High-order Eulerian methods for nonlinear approximation

High-order Explicit scheme for neural Galerkin

Remark: At the first order, with a good optimiser the time error will dominate the space error. To obtain a better accuracy we must therefore use higher order schemes in time.

- We begin by **Neural Galerkin** method.

- RK4 for a ode $u'(t) = F(u(t), t)$ writes:

$$u_{n+1} = u_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

with

$$\begin{aligned} k_1 &= F(u_n, t_n), \\ k_2 &= F\left(u_n + \frac{\Delta t}{2}k_1, t_n + \frac{\Delta t}{2}\right), \\ k_3 &= F\left(u_n + \frac{\Delta t}{2}k_2, t_n + \frac{\Delta t}{2}\right), \\ k_4 &= F(u_n + \Delta t k_3, t_n + \Delta t) \end{aligned}$$

- For neural Galerkin:

$$M(\boldsymbol{\theta}_n)\boldsymbol{\theta}_{n+1} = M(\boldsymbol{\theta}_n)\boldsymbol{\theta}_n + \frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

with

$$\begin{aligned} M(\boldsymbol{\theta}_n)\mathbf{k}_{\varphi_1} &= \mathbf{b}(\boldsymbol{\theta}_n, t_n), \\ M\left(\boldsymbol{\theta}_n + \frac{\Delta t}{2}\mathbf{k}_{\varphi_1}\right)\mathbf{k}_{\varphi_2} &= \mathbf{b}\left(\boldsymbol{\theta}_n + \frac{\Delta t}{2}\mathbf{k}_{\varphi_1}, t_n + \frac{\Delta t}{2}\right), \\ M\left(\boldsymbol{\theta}_n + \frac{\Delta t}{2}\mathbf{k}_{\varphi_2}\right)\mathbf{k}_{\varphi_3} &= \mathbf{b}\left(\boldsymbol{\theta}_n + \frac{\Delta t}{2}\mathbf{k}_{\varphi_2}, t_n + \frac{\Delta t}{2}\right), \\ M\left(\boldsymbol{\theta}_n + \Delta t\mathbf{k}_{\varphi_3}\right)\mathbf{k}_{\varphi_4} &= \mathbf{b}\left(\boldsymbol{\theta}_n + \Delta t\mathbf{k}_{\varphi_3}, t_n + \Delta t\right) \end{aligned}$$

HO Explicit scheme for Discrete PINNs

- We now consider discrete PINNs with an explicit RK4 scheme.
- In PInns discret we apply the time scheme to the function $u_{\boldsymbol{\theta}}$ and after we optimise.
  - For example

$$u_{\boldsymbol{\theta}_{n+1}}(\mathbf{x}) \approx \hat{u}_{n+1}(\mathbf{x}) = u_{\boldsymbol{\theta}_n}(\mathbf{x}) + \frac{\Delta t}{6} F\left(u_{\boldsymbol{\theta}_n}(\mathbf{x}), t_n + \Delta t\right)$$

- We obtain a estimation of the solution at time t_{n+1} at some points of the domain. To find the new parameter we need a projection step:

$$\boldsymbol{\theta}_{n+1} = \arg\min_{\boldsymbol{\theta}} \sum_i^N \int_\Omega |(u_{\boldsymbol{\theta}}(\mathbf{x}) - \hat{u}_{n+1}(\mathbf{x}))\psi_i(\mathbf{x})\ |^2 \ d\mathbf{x}$$

Remark : When we want a equation between the ponctual values of two functions we need to make a projection and consequently a optimization.

- Base on RK4 for collocation mehod:

$$\boldsymbol{\theta}_{n+1} = \arg\min_{\boldsymbol{\theta}} \sum_i^N \left(u_{\boldsymbol{\theta}}(\mathbf{x}) - u_{\boldsymbol{\theta}_n}(\mathbf{x}) - \frac{\Delta t}{6}\left(u_{\theta_{k_1}} + \ldots\right)\right)^2$$

with

$$\boldsymbol{\theta}_{k_1} = \arg\min_{\boldsymbol{\theta}} \sum_i^N \left(u_{\boldsymbol{\theta}}(\mathbf{x}) - F\left(u_{\boldsymbol{\theta}_n}, \Delta t\right)\right)^2$$

....

HO Implicit scheme for Discrete PINNs

- The principe for implicit schemes is the same that for exmplicit.
- We write the relation between $u_{\boldsymbol{\theta}_{n+1}}(\mathbf{x})$ and $u_{\boldsymbol{\theta}_n}(\mathbf{x})$ given by the implicit scheme and you put it in the optimization.

Remark (): It is case we obtain only the implicit part

- **Example**: Crank Nicholson

$$\theta(t^{n+1}) = \min_{\theta} \sum_{i}^{N} \int_{\Omega} \left| \left(nn_{\theta}(\mathbf{x}) - \frac{1}{2} R\left(\mathbf{x}; u_{\boldsymbol{\theta}_n}(\mathbf{x})\right) - \frac{1}{2} R(\mathbf{x}; nn_{\boldsymbol{\theta}}(\mathbf{x})) \right) \psi_i(\mathbf{x}) \right|^2 d\mathbf{x}$$

- For implicit Runge-Kutta there this substep and one optimisation problem by substep.

1.3. Splitting schemes

Splitting for ODE

Definition (Splitting methods): Splitting methods are numerical methods to solve complex problems by decomposing them into simpler subproblems. Each subproblem is solved sequentially, and their solutions are combined to approximate the solution of the original problem.

- Consider an ODE of the form:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t)$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

which we want to solve on the interval $[t_0, t_f]$.

- Supposing that we can write $A = A_1 + A_2$ and solving the ODE gives the flow:

$$\phi^{t_0}(t, \mathbf{x}_0) = e^{A(t-t_0)}\mathbf{x}_0 \approx e^{A_2(t-t_0)}e^{A_1(t-t_0)}\mathbf{x}_0$$

Proposition (Splitting error): Let consider A, A_1, A_2 such that $A = A_1 + A_2$ The error between e^{At} and $e^{A_1 t}e^{A_2 t}$ is of the form:

$$e^{At} - e^{A_1 t}e^{A_2 t} = -\frac{t^2}{2}[A_1, A_2] + O(t^3)$$

with $[A_1, A_2] = A_1A_2 - A_2A_1$.

Splitting for ODE II

Making this approximation is equivalent to solving:

$$\dot{\mathbf{x}}_1(t) = A_1 \mathbf{x}_1(t) \qquad \dot{\mathbf{x}}_2(t) = A_2 \mathbf{x}_2(t)$$
$$\mathbf{x}_1(t_0) = \mathbf{x}_0 \qquad \mathbf{x}_2(t_0) = \mathbf{x}_1(t)$$

Now we will generalize the splitting method to the case where A can be written as a sum of several terms.

Definition (Lie Trotter Splitting scheme): Let consider an ODE of the form $\dot{\mathbf{x}}_1(t) = \mathbf{f}(\mathbf{x}(t))$ with

$$\mathbf{f}(t, \mathbf{x}) = \mathbf{f}(\mathbf{x}) = \mathbf{f}_1(\mathbf{x}) + \ldots + \mathbf{f}_{q(\mathbf{x})}$$

The numerical flow associated with the Lie splitting scheme is of the form

$$\Phi_{\mathbf{f},\Delta t}(\mathbf{x}_n) = \mathbf{\Phi}_{\mathbf{f}_q,\Delta t} \circ \ldots \circ \mathbf{\Phi}_{\mathbf{f}_1,\Delta t}$$

with $\mathbf{\Phi}_{\mathbf{f}_i,\Delta t}$ a discrete flow associated to the ODe with $\mathbf{f}_i$. If the discrete flow of each equation converge at order one, the local consistency error being in $O(t^2)$ we thus have a scheme of order 1.

Splitting for PDE

- For PDE the idea is the same. We consider a PDE of the form

$$\partial_t u(t, x) = L(u(t, x)) = L_1(u(t, x)) + L_2(u(t, x))$$

- We solve successively the two problems. At each time step, if we use an explicit Euler scheme we have:

$$u_1^{n+\frac{1}{2}}(\mathbf{x}) = u_1^n(\mathbf{x}) + \Delta t L_1(u_1^n(\mathbf{x})) \qquad u_2^{n+\frac{1}{2}}(\mathbf{x}) = u_1^{n+\frac{1}{2}}(\mathbf{x}) + \Delta t L_2\left(u_1^{n+\frac{1}{2}}(\mathbf{x})\right)$$

Definition (Lie Trotter Splitting for Neural Galerkin): The continuous ODE is given

$$M(\boldsymbol{\theta}(t))\dot{\boldsymbol{\theta}}(t) = \mathbf{b}(\boldsymbol{\theta}(t))$$

We split $\mathbf{b}(\boldsymbol{\theta}(t))$ as you want (splitting the PDE term form example) and solve each system with your favorite scheme.

Splitting for PDE

- For PDE the idea is the same. We consider a PDE of the form

$$\partial_t u(t,x) = L(u(t,x)) = L_1(u(t,x)) + L_2(u(t,x))$$

Definition (Lie Trotter Splitting for PINNs dscret): We split the PDE term and a each step we solve the discret PINNs for each subpde:

$$\boldsymbol{\theta}\left(t^{n+\frac{1}{2}}\right) = \min_{\theta} \sum_{i}^{N} \int_{\Omega} \left| \left(\langle \boldsymbol{\theta}, \boldsymbol{\Phi} \rangle - R_1\left(\mathbf{x}, u_{\boldsymbol{\theta}_n}\right)\right) \psi_i(\mathbf{x}) \ \right|^2 \, d\mathbf{x}$$

and

$$\boldsymbol{\theta}(t^{n+1}) = \min_{\theta} \sum_{i}^{N} \int_{\Omega} \left| \left(\langle \boldsymbol{\theta}, \boldsymbol{\Phi} \rangle - R_2\left(\mathbf{x}, u_{\boldsymbol{\theta}_{n+\frac{1}{2}}}\right)\right) \psi_i(\mathbf{x}) \ \right|^2 \, d\mathbf{x}$$

Remark (how used spliting):

- We can use very chip scheme for each substep (less true in neural networks case)
- the conditioning of the full problem is bad and it is better to solve several well conditioned problems

1.4. Semi Lagrangian methods

Lagrangian method for transport I

- We consider here linear PDEs for particle transport.

Definition (Eulerian/Lagrangian description): Eulerian methods fix a domain (and generally a grid) and follow the evolution of quantities in this domain. Lagrangian methods directly follow the evolution of particles or fluid elements.

- We can see the previous approaches as Eulerian approaches. Here we propose an intermediate approach.
- We consider

$$\begin{aligned} &\partial_t u(t,x) + a\partial_x u(t,x) = 0 \\ &u(t=0,x) = u_0(x) \end{aligned}$$

Proposition: The solution is given by the method of characteristics:

$$u(t,x) = u_0(x - at)$$

Lagrangian method for transport II

- This gives a natural estimate for the next time:

$$u^{n+1}(x) = u^n(x - a\Delta t)$$

- If we are given an approximation space with parameter $\boldsymbol{\theta}$ linear or nonlinear we obtain

$$\boldsymbol{\theta}_{n+1} = \arg\min_{\boldsymbol{\theta}} \sum_i^N \int_\Omega |(\langle \boldsymbol{\theta}, \boldsymbol{\Phi} \rangle - u^n(x - a\Delta t))\psi_i(x)\ |^2\ dx$$

Definition (Characetistic curve): We consider a equation of the form

$$\partial_t u(t,x) + a(t,x)\partial_x u(t,x) = 0$$

The characteristic curves are given by the ODE:

$$\dot{X}(t) = a(t, X(t))$$
$$X(t=0) = x$$

The solution is given by

$$u(t,x) = u_0(X(t;x))$$

Semi Lagrangian variable transport

Algorithm (Semi lagrangian for variable velocity):

- Given $u^n(x)$ and $a(t, x)$
- For each point of the domain x_i:
  1. Solve the characteristic curve $\dot{X}(t) = a(t, X(t))$ with $X(t^{n+1}) = x_i$ to get $X(t^n; x_i)$
  2. Set $\hat{u}^{n+1}(x_i) = u_{\boldsymbol{\theta}_n}(X(t^n; x_i))$
- end for
- Find $\boldsymbol{\theta}_{n+1} = \arg\min_{\boldsymbol{\theta}} \sum_i^N |(\langle \boldsymbol{\theta}, \boldsymbol{\Phi} \rangle - \hat{u}^{n+1}(x))\ |^2 \ dx$

- We can see the part where we solve the characteristics as a Lagrangian part where we evolve particles.
- Then the part where we update the model to interpolate between particles as an Eulerian phase.

Remark: We can use as a parametric model a linear model which will come down to inverting the mass matrix or a nonlinear model (neural networks) which will come down to an optimization.

Semi Lagrangian and splitting

- We consider a kinetic equation of the form:

$$\partial_t f(t, \mathbf{x}, \mathbf{v}) + \mathbf{v} \cdot \nabla_{\mathbf{x}} f + F(\mathbf{x}) \cdot \nabla_{\mathbf{v}} f = 0$$

Remark : Taking the limit corresponding to an infinity of particles in Newton's laws

$$\begin{cases} \frac{d}{dt}\mathbf{x}_i(t) = \mathbf{v}_i(t) \\ \frac{d}{dt}\mathbf{v}_i(t) = F(\mathbf{x}_i(t)) \end{cases}$$

we obtain this equation. The force field $F(\mathbf{x})$ can be obtained by a PDE dependent on f and therefore on all the particles.

- The characteristics are given by this ODE for the set of particles.
- If we split

$$\begin{cases} \frac{d}{dt}\mathbf{x}_i(t) = 0 \\ \frac{d}{dt}\mathbf{v}_i(t) = F(\mathbf{x}_i(t)) \end{cases} \qquad \begin{cases} \frac{d}{dt}\mathbf{x}_i(t) = \mathbf{v}_i(t) \\ \frac{d}{dt}\mathbf{v}_i(t) = 0 \end{cases}$$

- On each time step one of the variables being frozen we can calculate the exact flow and obtain the characteristics:

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t \mathbf{v}_i^n, \qquad \mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \Delta t F(\mathbf{x}_i^n)$$

Semi Lagrangian for advection-reaction

- We consider the following equation:

$$\partial_t u(t,x) + a(t,x)\partial_x u(t,x) = -\sigma(t,x)u(t,x) + S(t,x)$$

- We define the characteristics:

$$\dot{X}(t) = a(t, X(t))$$
$$X(t=0) = x$$

- Let's inject $u(t, X(t))$ into the equation:

$$\begin{aligned}\frac{d}{dt}u(t,X(t)) &= \partial_t u(t,X(t)) + \partial_x u(t,X(t)) \cdot X'(t) \\ &= \partial_t u(t,X(t)) + \partial_x u(t,X(t)) \cdot a(t,X(t)) \\ &= -\sigma(t,X(t))u(t,X(t)) + S(t,X(t))\end{aligned}$$

- We must therefore solve an ODE of the form:

$v'(t) + \beta(t)v(t) = \gamma(t)$ with $v(t) = u(t, X(t))$

Remark: We can solve this equation with the method of variation of the constant then return to the PDE

Semi Lagrangian for advection-reaction

Theorem: The solution of the reaction-transport equation is given by:

$$u(t,x) = u_0(X(t;x)) \exp\left(-\int_0^t \sigma(s, X(s;x))ds\right) + \int_0^t \exp\left(-\int_s^t \sigma(\tau, X(\tau;x))d\tau\right) S(s, X(s;x))ds$$

with $X(t;x)$ the characteristic issued from x.

- By solving the characteristics and using quadratures in the interval [t_n, t_n + Delta t] we obtain an estimate of $u(t^{n+1}, x)$ as a function of $u(t^n, x)$ given for example at first order by:

$$\hat{u}^{n+1}(x) = u^n(X(t^n;x)) \exp(-\Delta t \sigma(t^n, X(t^n;x))) + \Delta t \exp(-\Delta t \sigma(t^n, X(t^n;x))) S(t^n, X(t^n;x))$$

Remark: We then apply an SL scheme as before.

Properties and remarks

Proposition (Properties of SL schemes):

- No CFL-type stability condition
- Just a mass matrix to invert
- Potentially high order especially in the case of transport

- For domain management, point sampling, integral calculation in the Galerkin case it works like usual methods
- Boundary conditions can be difficult to manage especially since we potentially go quite far to look for information.

1.5. Bonus: Lagrangian methods

1.6. Lagrangian methods for kinetic methods

PIC methods for transport

Definition (Lagrangian method): We really follow the particles during all the time evolution.

- We consider the following kinetic equation:

$$\partial_t f(t,x,v) + v\partial_x f(t,x,v) + F(x)\partial_v f(t,x,v) = 0$$

Definition (Approximation space of the PIC method): PIC method approximate the solution by a sum of Dirac masses (or regularized version) in phase space:

$$f(t,x,v) \approx \sum_{k=1}^{N} \delta\left(x - X_{k(t)}\right)\delta\left(v - V_{k(t)}\right)$$

with $\left(X_{k(t)}, V_{k(t)}\right)$ the position and velocity of each particle at time t

- The particles evolve according to the Newton laws:

$$\dot{X}_{k(t)} = V_{k(t)}$$
$$\dot{V}_{k(t)} = F\left(X_{k(t)}\right)$$

with initial condition $\left(X_{k(0)}, V_{k(0)}\right)$ given by an sampling of $f_0(x,v)$.

PIC as a Petrov-Galerkin method

We consider a nonlinear approximation space of the form

$$\mathcal{M}_n = \left\{ u_{\boldsymbol{\theta}}(x) = \sum_{k=1}^{n} \delta(\mathbf{x} - \mathbf{X}_k(t))\delta(\mathbf{v} - \mathbf{V}_k(t)), \boldsymbol{\theta} = (\mathbf{X}_1, \mathbf{V}_1, \ldots, \mathbf{X}_n, \mathbf{V}_n) \in R^{6n} \right\}$$

- We consider a Petrov-Galerkin method with test space W_n with $\dim W_n = 6n$.

Idea : We impose the orthgonality of the residual to the test functions:

$$\int_{\mathbb{R}^d} \int_{\Omega} (\partial_t u_{\boldsymbol{\theta}}(t,x,v) + v\partial_x u_{\boldsymbol{\theta}}(t,x,v) + F(x)\partial_v u_{\boldsymbol{\theta}}(t,x,v))\varphi_i(x,v)dxdv = 0$$

- We must make the computation term by term:

$$\int_{\mathbb{R}^d} \int_{\Omega} (\partial_t u_{\boldsymbol{\theta}}(t,\mathbf{x},\mathbf{v})\varphi_i(\mathbf{x},\mathbf{v})d\mathbf{x}\mathbf{v} = \sum_{k=1}^{N} \frac{d}{dt}\varphi_i(\mathbf{X}_k(t), \mathbf{V}_k(t))$$

$$\int_{\mathbb{R}^d} \int_{\Omega} (\mathbf{v} \cdot \partial_{\mathbf{x}} u_{\boldsymbol{\theta}}(t,\mathbf{x},\mathbf{v})\varphi_i(\mathbf{x},\mathbf{v})d\mathbf{x}\mathbf{v} = -\sum_{k=1}^{N} \mathbf{V}_k(t) \cdot \nabla_{\mathbf{x}}\varphi_i(\mathbf{X}_k(t), \mathbf{V}_k(t))$$

PIC as a Petrov-Galerkin method II

$$\int_{\mathbb{R}^d}\int_{\Omega}(\mathbf{F}(\mathbf{x})\cdot\partial_{\mathbf{v}}u_{\boldsymbol{\theta}}(t,\mathbf{x},\mathbf{v})\varphi_i(\mathbf{x},\mathbf{v})d\mathbf{x}\mathbf{v} = -\sum_{k=1}^{N}F(\mathbf{X}_k(t))\cdot\nabla_{\mathbf{v}}\varphi_i(\mathbf{X}_k(t),\mathbf{V}_k(t))$$

- We obtain

$$0=\sum_{k=1}^{N}\frac{d}{dt}\varphi_i(\mathbf{X}_k(t),\mathbf{V}_k(t))-\mathbf{V}_k(t)\cdot\nabla_{\mathbf{x}}\varphi_i(\mathbf{X}_k(t),\mathbf{V}_k(t))-F(\mathbf{X}_k(t))\cdot\nabla_{\mathbf{v}}\varphi_i(\mathbf{X}_k(t),\mathbf{V}_k(t))$$

- **First**: we choose $\varphi_i(\mathbf{x},\mathbf{v})=\delta(\mathbf{x}-\mathbf{X}_k)$ we obtain:

$$\frac{d}{dt}\mathbf{X}_k(t)=\mathbf{V}_k(t)$$

- **Then**: we choose $\varphi_i(\mathbf{x},\mathbf{v})=\delta(\mathbf{v}-\mathbf{V}_k)$ we obtain:

$$\frac{d}{dt}\mathbf{V}_k(t)=F(\mathbf{X}_k(t))$$

Remark: " We can use regularized Dirac masses (e.g. Gaussian) instead of Dirac masses". In this case we can probably have an equivalence with a optimisation problem

1.6.1. BDSE methods

Backward PDEs

Remark (Backward PDE): This equation is well-posed if $b(t, x)$ is a velocity field going from T to 0.

Definition (Backward advection diffusion PDE): We consider the following PDE:

$$\partial_t p(t, x) + b(t, x)\partial_x p(t, x) + \frac{1}{2}\partial_{xx}(\sigma^2(t, x)p(t, x)) = F(t, x, p(t, x))$$

with final condition $p(T, x) = g(x)$.

Applications:

- Control optimal/Hamilton-Jacobi-Bellman equations:

•

$$F(t, x, V(t, x)) = \min_u (L(t, x, u) + b(t, x, u)\partial_x V(t, x))$$

with $V(t, x)$ the value function.

- Finance/option pricing:

$$F(t, x, V(t, x)) = -rV(t, x)$$

with r the interest rate and $V(t, x)$ the option price.

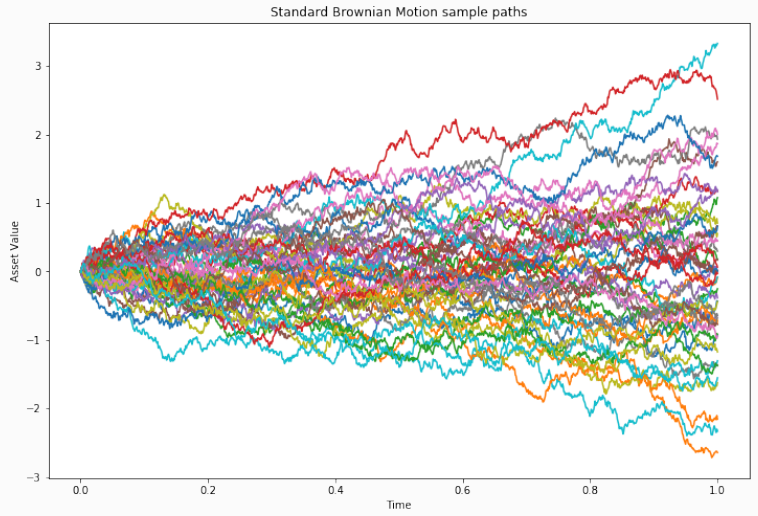
Brownian Dynamics

- A continuous stochastic process can be seen as a time-dependent function whose evolution between two instants is described by a probability law rather than by a deterministic relation.

Definition (Brownian motion): A Brownian motion W_t is a stochastic process such that:

- $W_0 = 0$ almost surely,
- W_t has independent increments,
- $W_t - W_s \sim \mathcal{N}(0, t - s)$ for $0 \leq s < t$.

- W_t is almost surely nowhere differentiable.
- $\mathbb{E}[dW_t] = 0$ and $\mathrm{Var}[dW_t] = dt$



EDS

- Classical ODE:

$$\frac{dX_t}{dt} = b(t, X_t)$$

with $X_t = X(t)$ a function depending of time. Fully deterministic modeling.
- Stochastic ODE (SDE):

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

with W_t a Brownian process. Fully stochastic modeling.
- The first term produces, at each instant, an infinitesimal variation proportional to the time increment dt
- The second term produces, at each instant, an infinitesimal variation proportional to dW_t which corresponds to a stochastic variation of order $\sqrt{dt}$.

Definition (Stochastic Euler scheme): Let Δt a time step, the stochastic Euler scheme for the SDE $dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t$ is given by:

$$X_{n+1} = X_n + b(t_n, X_n)\Delta t + \sigma(t_n, X_n)\sqrt{\Delta t}Y_n$$

where Y_n are independent random variables with law $\mathcal{N}(0, 1)$.

Ito Lemma

- For a deterministic function $f(t, x(t))$ the chain rules gives:

$$df(t, x(t)) = \partial_x f(t, x(t))dx(t) + \partial_t f(t, x(t))dt$$

Remark: This formula is no longer valid if $x(t)$ is a stochastic process since W_t is almost surely nowhere differentiable.

Lemma (Ito formula): Let X_t a stochastic process solution of the SDE $dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t$ and $p(t, x)$ a regular function, then

$$dp(t, X_t) = \partial_x p(t, X_t)dX_t + \partial_t p(t, X_t)dt + \frac{1}{2}\sigma^2(t, X_t)\partial_{xx}p(t, X_t)dt$$

Remark: The additional term $\frac{1}{2}\sigma^2(t, X_t)\partial_{xx}p(t, X_t)$ accounts for the stochastic nature of X_t.

Ito Lemma and Feyman-Kac

- using the SDE $dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t$ in the Ito formula we get:

$$dp(t, X_t) = \left(\partial_t p(t, X_t) + b(t, X_t)\partial_x p(t, X_t) + \frac{1}{2}\sigma^2(t, X_t)\partial_{xx} p(t, X_t)\right)dt + \sigma(t, X_t)\partial_x p(t, X_t)dW_t$$

- We can now integrate between t and $t + \Delta t$:

$$p(T, X_T) - p(t, X_t) = \int_t^T \left(\partial_s p(s, X_s) + b(s, X_s)\partial_x p(s, X_s) + \frac{1}{2}\sigma^2(s, X_s)\partial_{xx} p(s, X_s)\right)ds + \int_t^T \sigma(s, X_s)\partial_x p(s, X_s)dW_s$$

Definition (Backward Fokker-Planck equation): We suppose that $f(t, x)$ is solution of the Fokker-Planck equation:

$$\partial_t p(t, x) + b(t, x)\partial_x p(t, x) + \frac{1}{2}\partial_{xx}(\sigma^2(t, x)p(t, x)) = F(t, x, p(t, x))$$

with final condition $p(T, x) = g(x)$.

We therefore obtain

$$p(T, X_T) - p(t, X_t) = -\int_t^T F(s, X_s, p(s, X_s))ds + \int_t^T \sigma(s, X_s)\partial_x p(s, X_s)dW_s$$

PDEs and Feyman-Kac formula

- We rearrange the previous equation to get:

$$Y_t = g(X_T) + \int_t^T F(s, X_s, y_s)ds - \int_t^T Z_s dW_s$$

with $Z_s = \sigma(s, X_s)\partial_x p(s, X_s)$ and $y_s = p(s, X_s)$.

Definition (Feyman-Kac formula): We consider X_t solution of the forward SDE

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

with $X_t = x$ and (Y_t, Z_t) solution of the backward SDE

$$Y_t = -F(t, X_t, Y_t)dt + Z_t dW_t$$

with $Z_t = \sigma(t, X_t)\partial_x p(t, X_t)$ and $Y_T = g(X_T)$ so that $p(t, x)$ the solution of the previous PDE is given by

$$p(t, x) = \mathbb{E}\left[g(T) + \int_t^T F(s, X_s, Y_s)ds \mid X_t = x\right]$$

We do not detail the proof obtained from the expectation.

-

Deep Feyman Kac method I

- We discretize the time interval (t, T) in N intervals of size Δt and we use an Euler scheme for the SDEs:

$$X_{n+1} = X_n + b(t_n, X_n)\Delta t + \sigma(t_n, X_n)\sqrt{\Delta t}Y_n$$

- We propose to discretize in time the expectation. We consider a time nterval $\Delta t_n = t_{n+1} - t_n$. We begin at the end and iterate.
- At each time step we have a terminal condition which is the solution at the following time step.

$$p_{t_n}(x) = \mathbb{E}\left[p_{t_{n+1}}\left(X_{t_{n+1}}\right) + \int_{t_n}^{t_{n+1}} F(s, X_s, p(s, X_s))ds \mid X_{t_{n+1}} = x\right]$$

- We approximate the integral with a right rectangle method:

$$p_{t_n}(x) = \mathbb{E}\left[p_{t_{n+1}}\left(X_{t_{n+1}}\right) + F\left(t_{n+1}, X_{t_{n+1}}, p\left(t_{n+1}, X_{t_{n+1}}\right)\right)\Delta t_n \mid X_{t_{n+1}} = x\right]$$

Idea (Deep Feyman Kac method): We will use a neural network to represent $p_{t_n}(x)$ at each time step“

Deep Feyman Kac methods II

- We simulate M trajectories of X_t with $X_0 = x_i$ some points of a mesh or cloud points. We obtain the trajectories $\left\{X_{t_n}^i\right\}_{1 \leq i \leq M}$
- We evaluate at this points the solution so

$$p_{t_n}\left(X_{t_n}^i\right) = p_{t_{n+1}}\left(X_{t_{n+1}}^i\right) + F\left(t_{n+1}, X_{t_{n+1}}^i, p\left(t_{n+1}, X_{t_{n+1}}^i\right)\right)\Delta t_n$$

So we known the excepted value of the solution at some points $\left\{X_{t_n}^i\right\}_{1 \leq i \leq M}$. Since we known the solution at some point we can just make **a function approximation using neural network** solving

$$p_{\theta_{t_n}}(x) = \min_{\theta} \int_{\Omega} \| p_{\theta}(x) - p_{t_n}(x) \|^2 dx \approx \sum_{i=1}^{M} \| p_{\theta}\left(X_{t_n}^i\right) - p_{t_n}\left(X_{t_n}^i\right) \|^2$$

where $p_{t_n}(.)$ use the network obtained at the time t_{n+1}

Remark: Other algorithm fit all the networks with one training