Adjoint Monte Carlo Methods for Kinetic Equation Constrained Optimization

Yunan Yang

April 25, 2024

Department of Mathematics, Cornell University

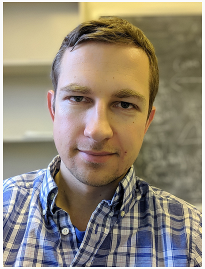
- Caflisch, R., Silantyev, D. and Y., 2021. Adjoint DSMC for nonlinear Boltzmann equation constrained optimization. *Journal of Computational Physics.*
- Y., Silantyev, D. and Caflisch, R., 2023. Adjoint DSMC for nonlinear spatially-homogeneous Boltzmann equation with a general collision model. *Journal of Computational Physics.*
- Li, Q., Wang, L. and Y., 2023. Monte Carlo Gradient in Optimization Constrained by Radiative Transport Equation. *SIAM Journal on Numerical Analysis.*

International Conference on Multiscale Modeling based on Physics and Data, IPAM

Collaborators

Russel Caflisch (NYU)

Denis Silantyev (Univ. of Colorado)



Qin Li (UW Madison)

Li Wang (UMN Twin Cities)

Happy Birthday to Russ!

Examples of Kinetic Equations

Nonlinear Boltzmann Equation for $f(t,x,v)$

$$\partial_t f + v \cdot \nabla_x f = \int_{\mathbb{R}^3} \int_{\mathcal{S}^2} q(v - v_1, \sigma) \left(f(v_1')f(v') - f(v_1)f(v)\right) d\sigma dv_1$$

Radiative Transfer/Transport Equation (RTE) for $f(t,x,v)$

$$\partial_t f + v \cdot \nabla_x f = \sigma(x) \left(\frac{1}{|\Omega|} \int_\Omega f(x,v)dv - f \right)$$

Vlasov–Poisson Equation for $f(t,x,v)$

$$\begin{cases} \partial_t f + v \cdot \nabla_x f - (H(x) + \nabla_x V_f) \cdot \nabla_v f = 0 \\ \Delta V_f = 1 - \int f \, dv \end{cases}$$

Linear Fokker–Planck Equation (FPE) for $f(t,x)$

$$\partial_t f + \nabla \cdot (V(x) f) = \nabla \cdot (D(x) \nabla f)$$

Example: Boltzmann Binary Equation

$$\begin{cases} \dfrac{\partial f}{\partial t} + v \cdot \nabla_x f = Q(f,f) \\ f(0,x,v) = f_0 \text{ on } \Omega \\ f = f^{eq}(\rho_b, \boldsymbol{u}_b, T_b) \text{ on } \partial\Omega \end{cases}$$

where

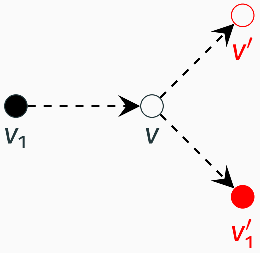
$$Q(f,f) = \int_{\mathbb{R}^3} \int_{\mathcal{S}^2} q(v - v_1, \sigma) \left(f(v_1')f(v') - f(v_1)f(v) \right) d\sigma dv_1,$$

and

$$v' = 1/2(v + v_1) + 1/2|v - v_1|\sigma,$$
$$v_1' = 1/2(v + v_1) - 1/2|v - v_1|\sigma.$$

This is the “Eulerian” version of the forward problem.

Binary Collisions



● v_1 ○ v pre-collision velocities

● v_1' ○ v' post-collision velocities

Elastic collision preserves mass, momentum and energy. Assume particles have equal mass, then

$$\begin{aligned} v_1 + v &= v_1' + v' \\ |v_1|^2 + |v|^2 &= |v_1'|^2 + |v'|^2 \\ |v - v_1| &= |v' - v_1'| \end{aligned}$$

Direct Simulation Monte Carlo (DSMC) Method

Due to splitting, the collision part is based on the spatially homogeneous setting.

$$\begin{cases} \dfrac{\partial f}{\partial t} = Q\left(f, f\right), \\ f(0, v) = f_0. \end{cases}$$

Assume $q(v - v_1, \sigma) \leq \Sigma$ [Pareschi-Russo, 1999]

$$\begin{aligned} \frac{\partial f}{\partial t} &= \iint f' f_1' q \, d\sigma dv_1 - \iint f f_1 q \, d\sigma dv_1 \\ &= \underbrace{\iint f' f_1' q \, d\sigma dv_1}_{\text{real collision}} + \underbrace{f \iint f_1 (\Sigma - q) d\sigma dv_1}_{\text{virtual but non-real collision}} - \underbrace{\iint \Sigma f f_1 d\sigma dv_1}_{\text{non-virtual collsion, } = \mu f,\ \mu = 4\pi\Sigma\rho} . \end{aligned}$$

If collision kernel q has a complicated form, we use **rejection sampling** to sample q, while the rejected samples are effective samples of $\Sigma - q$.

Direct Simulation Monte Carlo (DSMC) Method for General Collision Kernel

```
1: Compute the initial velocity particles based on the initial condition, $\mathcal{V}^0 = \{v_1^0, \ldots, v_N^0\}$.
2: for $k = 0$ to $M - 1$ do
3:     Given $\mathcal{V}^k$, choose $N_c = \lceil \Delta t \mu N/2 \rceil$ velocity pairs $(i_\ell, i_{\ell_1})$ uniformly without replacement. The remaining $N - 2N_c$ particles do not have a virtual (or real) collision and set $v_i^{k+1} = v_i^k$.
```

Direct Simulation Monte Carlo (DSMC) Method for General Collision Kernel

```
1: Compute the initial velocity particles based on the initial condition, $\mathcal{V}^0 = \{v_1^0, \dots, v_N^0\}$.
2: for $k = 0$ to $M - 1$ do
3:     Given $\mathcal{V}^k$, choose $N_c = \lceil \Delta t \mu N/2 \rceil$ velocity pairs $(i_\ell, i_{\ell_1})$ uniformly without replacement. The
       remaining $N - 2N_c$ particles do not have a virtual (or real) collision and set $v_i^{k+1} = v_i^k$.
4:     for $\ell = 1$ to $N_c$ do
5:         Sample $\sigma_\ell^k$ uniformly over $\mathbb{S}^2$ and compute $\theta_\ell^k = \arccos(\sigma_\ell^k \cdot \alpha_\ell^k)$ and $q_\ell^k = q(|v_{i_\ell}^k - v_{i_{\ell_1}}^k|, \theta_\ell^k)$.
6:         Draw a random number $\xi_\ell^k$ from the uniform distribution $\mathcal{U}([0, 1])$.
```

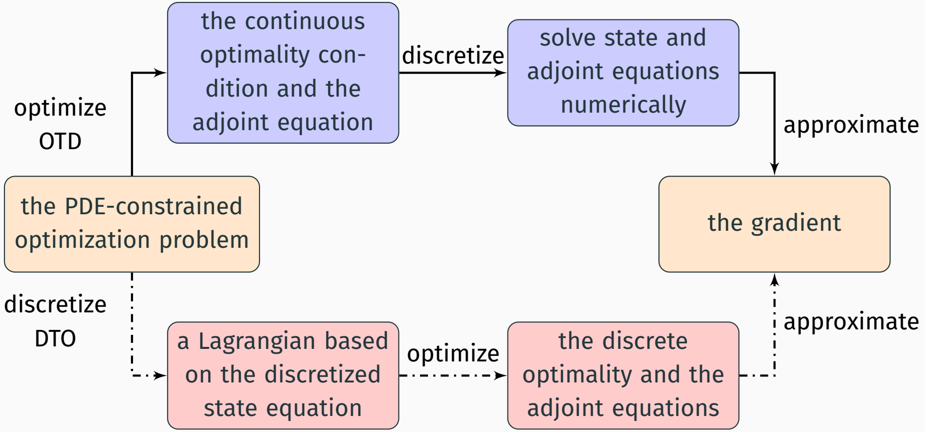
Direct Simulation Monte Carlo (DSMC) Method for General Collision Kernel

```
1: Compute the initial velocity particles based on the initial condition, $\mathcal{V}^0 = \{v_1^0, \ldots, v_N^0\}$.
2: for $k = 0$ to $M-1$ do
3:     Given $\mathcal{V}^k$, choose $N_c = \lceil \Delta t \mu N/2 \rceil$ velocity pairs $(i_\ell, i_{\ell_1})$ uniformly without replacement. The remaining $N - 2N_c$ particles do not have a virtual (or real) collision and set $v_i^{k+1} = v_i^k$.
4:     for $\ell = 1$ to $N_c$ do
5:         Sample $\sigma_\ell^k$ uniformly over $\mathbb{S}^2$ and compute $\theta_\ell^k = \arccos(\sigma_\ell^k \cdot \alpha_\ell^k)$ and $q_\ell^k = q(|v_{i_\ell}^k - v_{i_{\ell_1}}^k|, \theta_\ell^k)$.
6:         Draw a random number $\xi_\ell^k$ from the uniform distribution $\mathcal{U}([0,1])$.
7:         if $\xi_\ell^k \leq q_\ell^k/\Sigma$ then
8:             Perform real collision between $v_{i_\ell}^k$ and $v_{i_{\ell_1}}^k$ and obtain $(v_{i_\ell}^{k+1}, v_{i_{\ell_1}}^{k+1}) = (v_{i_\ell}^{k\,\prime}, v_{i_{\ell_1}}^{k\,\prime})$.
9:         else
10:            The virtual collision is not a real collision. Set $(v_{i_\ell}^{k+1}, v_{i_{\ell_1}}^{k+1}) = (v_{i_\ell}^k, v_{i_{\ell_1}}^k)$.
11:        end if
12:    end for
13: end for
```

This is the “Lagrangian” version of the forward problem.

Common Frameworks for PDE-Constrained Optimization Problems

Optimize-then-Discretize (OTD) and Discretize-then-Optimize (DTO) approaches


the continuous optimality condition and the adjoint equation
discretize
solve state and adjoint equations numerically
optimize OTD
approximate
the PDE-constrained optimization problem
the gradient
discretize DTO
approximate
a Lagrangian based on the discretized state equation
optimize
the discrete optimality and the adjoint equations

[Caflisch, R., Silantyev, D. and Y., 2021]

The DTO Approach

The constrained optimization becomes

$$\min_{m} J(f) \quad \Longrightarrow \quad \boxed{\min_{m} J\left(\{(v_i^k\}, \{x_i^k\}\right)}, \quad i = 1, \dots, N, \; k = 0, \dots, M$$

subject to

$$h(f, m) = 0 \quad \Longrightarrow \quad \boxed{h\left(\{(v_i^k\}, \{x_i^k\}, m\right) = 0} \quad \text{(the constraints for the particles)}$$

That is, all **random** particles inherit the dependence on the target parameter m.

The DTO Approach

The constrained optimization becomes

$$\min_{m} J(f) \quad \Longrightarrow \quad \boxed{\min_{m} J\left(\{(v_i^k\}, \{x_i^k\}\right)}, \quad i = 1, \ldots, N, \; k = 0, \ldots, M$$

subject to

$$h(f, m) = 0 \quad \Longrightarrow \quad \boxed{h\left(\{(v_i^k\}, \{x_i^k\}, m\right) = 0} \quad \text{(the constraints for the particles)}$$

That is, all **random** particles inherit the dependence on the target parameter m.

For gradient-based optimization (necessary for large-scale problems), we need to $\boxed{\text{differentiate all the } \textbf{random} \text{ particles with respect to } m}$!

Monte Carlo Gradient

Here are a few common techniques to handle gradients of Monte Carlo Samples

- **Pathwise Derivative Method/ the "the reparameterization trick"** (LOTUS)

$$\partial_\theta \int \phi(x) d\mu(x;\theta) = \partial_\theta \int \phi(F(x;\theta))\, d\pi(x) = \int \partial_\theta \phi(F(x;\theta))\, d\pi(x)$$

if $\underline{\mu = F(x;\theta)\sharp\pi}$ and ϕ is a test function.

Monte Carlo Gradient

Here are a few common techniques to handle gradients of Monte Carlo Samples

- **Pathwise Derivative Method/ the “the reparameterization trick”** (LOTUS)

$$\partial_\theta \int \phi(x) d\mu(x;\theta) = \partial_\theta \int \phi(F(x;\theta))\, d\pi(x) = \int \partial_\theta \phi(F(x;\theta))\, d\pi(x)$$

  if $\underline{\mu = F(x;\theta)\sharp\pi}$ and ϕ is a test function.

- **Likelihood Ratio Method (LRM) / Score Function Method**

$$\partial_\theta \int \phi(x)\rho(x;\theta) dx = \partial_\theta \int \phi(x) \log\rho(x;\theta)\ \rho(x;\theta) dx = \mathbb{E}_\rho \left[\phi(x)\, \partial_\theta \log\rho(x;\theta)\right].$$

Monte Carlo Gradient

Here are a few common techniques to handle gradients of Monte Carlo Samples

- **Pathwise Derivative Method/ the "the reparameterization trick"** (LOTUS)

$$\partial_\theta \int \phi(x) d\mu(x;\theta) = \partial_\theta \int \phi(F(x;\theta))\, d\pi(x) = \int \partial_\theta \phi(F(x;\theta))\, d\pi(x)$$

  if $\underline{\mu = F(x;\theta)\sharp\pi}$ and ϕ is a test function.
- **Likelihood Ratio Method (LRM) / Score Function Method**

$$\partial_\theta \int \phi(x)\rho(x;\theta) dx = \partial_\theta \int \phi(x) \log \rho(x;\theta)\ \rho(x;\theta) dx = \mathbb{E}_\rho \left[\phi(x)\, \partial_\theta \log \rho(x;\theta)\right].$$

- **Coupling Method:** run two correlated simulations and use their outcomes to estimate the gradient
  (e.g., fix the random seed, but not practical for large-scale optimization)

Example: Boltzmann Binary Collision

Recall the Boltzmann binary collision ($\alpha = \frac{v-v_1}{|v-v_1|}$, $\sigma = \frac{v'-v_1'}{|v'-v_1'|}$)

$$\begin{pmatrix} v' \\ v_1' \end{pmatrix} = A(\sigma, \alpha) \begin{pmatrix} v \\ v_1 \end{pmatrix}, \qquad A(\sigma, \alpha) = \frac{1}{2} \begin{pmatrix} I + \sigma\alpha^T & I - \sigma\alpha^T \\ I - \sigma\alpha^T & I + \sigma\alpha^T \end{pmatrix}.$$

However, (v, v_1, σ) is selected with **rejection sampling**. We **cannot** simply use

$$\frac{\partial(v', v_1')}{\partial(v, v_1)} = A(\sigma, \alpha)\,,$$

(except for Maxwellian gas). It does not take into consideration that the event (v, v_1) collide with a probability depending on parameter .

Example: Boltzmann Binary Collision

Recall the Boltzmann binary collision ($\alpha = \frac{v-v_1}{|v-v_1|}$, $\sigma = \frac{v'-v_1'}{|v'-v_1'|}$)

$$\begin{pmatrix} v' \\ v_1' \end{pmatrix} = A(\sigma,\alpha) \begin{pmatrix} v \\ v_1 \end{pmatrix}, \qquad A(\sigma,\alpha) = \frac{1}{2} \begin{pmatrix} I+\sigma\alpha^T & I-\sigma\alpha^T \\ I-\sigma\alpha^T & I+\sigma\alpha^T \end{pmatrix}.$$

However, (v, v_1, σ) is selected with **rejection sampling**. We **cannot** simply use

$$\frac{\partial(v', v_1')}{\partial(v, v_1)} = A(\sigma,\alpha)\,,$$

(except for Maxwellian gas). It does not take into consideration that the event (v, v_1) collide with a probability depending on parameter .

How do we differentiate with respect to “rejection sampling”?!

How to Differentiate Samples from Rejection Sampling

Consider that $\xi \sim \mathcal{U}([0,1])$ and $v' = \begin{cases} C(v, v_1), & \text{if } \xi < q(v - v_1, \sigma). \\ v, & \text{otherwise.} \end{cases}$

How to Differentiate Samples from Rejection Sampling

Consider that $\xi \sim \mathcal{U}([0,1])$ and $v' = \begin{cases} C(v, v_1), & \text{if } \xi < q(v - v_1, \sigma). \\ v, & \text{otherwise.} \end{cases}$

Instead of enforcing the relationship **pointwisely**

$$v' = \mathbb{1}_{\xi < q(v - v_1, \sigma)}\, C(v, v_1) + \left(1 - \mathbb{1}_{\xi < q(v - v_1, \sigma)}\right) v\,,$$

How to Differentiate Samples from Rejection Sampling

Consider that $\xi \sim \mathcal{U}([0,1])$ and $v' = \begin{cases} C(v, v_1), & \text{if } \xi < q(v - v_1, \sigma). \\ v, & \text{otherwise.} \end{cases}$

Instead of enforcing the relationship **pointwisely**

$$v' = \mathbb{1}_{\xi < q(v - v_1, \sigma)}\, C(v, v_1) + \left(1 - \mathbb{1}_{\xi < q(v - v_1, \sigma)}\right) v\,,$$

we enforce the collision relation **weakly through expectation.** $\forall \phi$,

$$\mathbb{E}_{v'}[\phi] = \int \phi(v') dv' = q\, \phi(C(v, v_1)) + (1 - q)\, \phi(v)\,, \quad q = q(v - v_1, \sigma)\,.$$

How to Differentiate Samples from Rejection Sampling

Consider that $\xi \sim \mathcal{U}([0,1])$ and $v' = \begin{cases} C(v, v_1), & \text{if } \xi < q(v - v_1, \sigma). \\ v, & \text{otherwise.} \end{cases}$

Instead of enforcing the relationship **pointwisely**

$$v' = \mathbb{1}_{\xi < q(v - v_1, \sigma)}\, C(v, v_1) + \left(1 - \mathbb{1}_{\xi < q(v - v_1, \sigma)}\right) v\,,$$

we enforce the collision relation **weakly through expectation.** $\forall \phi$,

$$\mathbb{E}_{v'}[\phi] = \int \phi(v') dv' = q\, \phi(C(v, v_1)) + (1 - q)\, \phi(v)\,, \quad q = q(v - v_1, \sigma)\,.$$

The derivative of the above weak relation is

$$\begin{aligned} \partial_v \left(\mathbb{E}_{v'}[\phi]\right) &= \partial_v \left(q \phi(C(v, v_1))\right) + \partial_v \left((1 - q)\, \phi(v)\right) \\ &= q\, \partial_v \left(\log q\, \phi(C(v, v_1))\right) + (1 - q)\, \partial_v \left(\log (1 - q)\, \phi(v)\right) \\ &= \mathbb{E}_{v'}[(\partial_v \log p)\ \phi]\,, \qquad p = q\, (\text{if } \xi < q),\ p = 1 - q\, (\text{otherwise}). \end{aligned}$$

The Adjoint DSMC Algorithm for Gradient Calculation

```
1:  Given final-time velocities from the forward DSMC, set $\gamma_i^M = \frac{1}{N}\partial_v r(v_i^M)$ for all $i$.
2:  for $k = M-1$ to $0$ do
3:      Given $\{\gamma_1^{k+1}, \dots, \gamma_N^{k+1}\}$ and collision parameters from the forward DSMC.
4:      if $v_i^k \in \mathcal{V}_k$ did not virtually collide at $t_k$ then
5:          Set $\gamma_i^k = \gamma_i^{k+1}$.
6:      else if $v_i^k, v_{i_1}^k \in \mathcal{V}_k$ virtually collided at $t_k$ then
7:          Set $\begin{pmatrix} \gamma_i^k \\ \gamma_{i_1}^k \end{pmatrix} = D_i^k \begin{pmatrix} \gamma_i^{k+1} \\ \gamma_{i_1}^{k+1} \end{pmatrix} + \frac{1}{N}\left(r(v_i^M) + r(v_{i_1}^M)\right) \begin{pmatrix} \frac{\partial \log h_i^k}{\partial v_i^k} \\ \frac{\partial \log h_i^k}{\partial v_{i_1}^k} \end{pmatrix}$.
8:      end if
9:  end for
10: Compute the gradient $\nabla_m J = \sum_{i=1}^N \frac{\partial v_i^0}{\partial m} \cdot \gamma_i^0$.
```

The Adjoint DSMC Algorithm for Gradient Calculation

```
1: Given final-time velocities from the forward DSMC, set $\gamma_i^M = \frac{1}{N}\partial_v r(v_i^M)$ for all $i$.
2: for $k = M - 1$ to $0$ do
3:     Given $\{\gamma_1^{k+1}, \ldots, \gamma_N^{k+1}\}$ and collision parameters from the forward DSMC.
4:     if $v_i^k \in \mathcal{V}_k$ did not virtually collide at $t_k$ then
5:         Set $\gamma_i^k = \gamma_i^{k+1}$.
6:     else if $v_i^k, v_{i_1}^k \in \mathcal{V}_k$ virtually collided at $t_k$ then
7:         Set $\begin{pmatrix} \gamma_i^k \\ \gamma_{i_1}^k \end{pmatrix} = D_i^k \begin{pmatrix} \gamma_i^{k+1} \\ \gamma_{i_1}^{k+1} \end{pmatrix} + \frac{1}{N}\left(r(v_i^M) + r(v_{i_1}^M)\right) \begin{pmatrix} \frac{\partial \log h_i^k}{\partial v_i^k} \\ \frac{\partial \log h_i^k}{\partial v_{i_1}^k} \end{pmatrix}$.
8:     end if
9: end for
10: Compute the gradient $\nabla_m J = \sum_{i=1}^N \frac{\partial v_i^0}{\partial m} \cdot \gamma_i^0$.
```

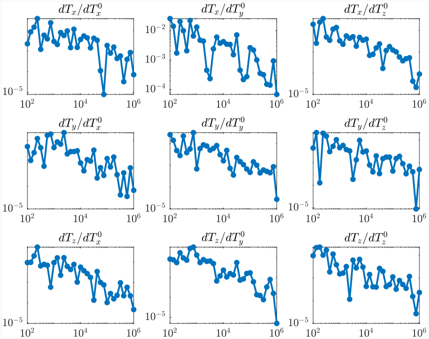
(Similar but simpler adjoint MC methods exists for RTE; see [Li-Wang-Y., 2023])

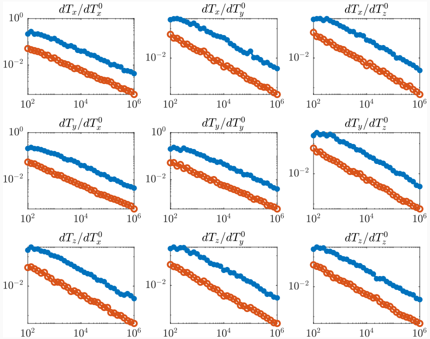
Numerical Examples

General Collision Kernel

We consider the general collision kernel (both velocity- and angle-dependent)

$$q(v - v_1, \sigma) \sim (1 + \cos\theta)^{\kappa} |v - v_1|^{\beta}, \quad \cos\theta = \sigma \cdot \frac{v - v_1}{|v - v_1|}.$$





$\kappa = 5$, $\beta = 1$ and $M = 20$. Left: error compared with central-difference gradient. Right: standard deviations for the central-difference gradient (blue) and the adjoint DSMC gradient (red).

Numerical Comparison (Memory, Error, Speed)

We compute the gradient numerically after the forward DSMC simulations solving the Boltzmann equation ($N = 10^6$)***:

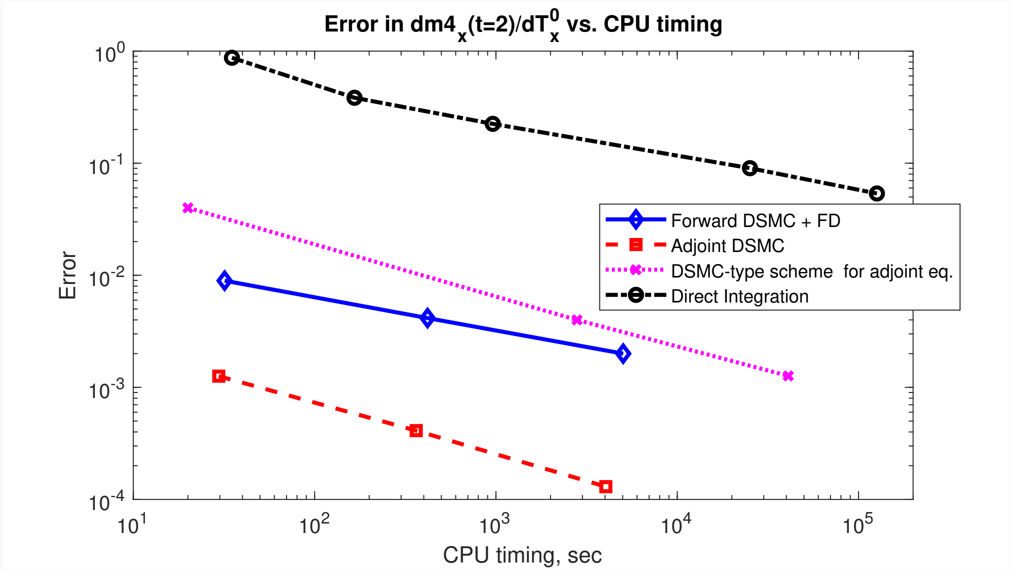
1. finite difference method; (0.38s *for one parameter**)
2. adjoint DSMC method; (0.22 s)
3. particle method for the continuous adjoint eqn; (280 s)
4. direct discretization of the continuous adjoint eqn. (overnight)**

*Computational costs for #2, #3 and #4 are independent of the size of the unknowns, but untrue for #1 — The beauty of the adjoint-state method.

**This is a result of backward Euler scheme in time and Riemann sum for the RHS integral.

*** More details in the paper [Caflisch, R., Silantyev, D. and Y., 2021]

Error vs. CPU Timing



Summary (Adjoint Monte Carlo Method)

- **The OTD approach:**

Summary (Adjoint Monte Carlo Method)

- **The OTD approach:**
    1. The forward PDE MC solve produces trajectories

Summary (Adjoint Monte Carlo Method)

- **The OTD approach:**
  1. The forward PDE MC solve produces trajectories
  2. Solving the linear adjoint equation needs to be careful and consistent

Summary (Adjoint Monte Carlo Method)

- **The OTD approach:**
  1. The forward PDE MC solve produces trajectories
  2. Solving the linear adjoint equation needs to be careful and consistent
- **The DTO approach:**

Summary (Adjoint Monte Carlo Method)

- **The OTD approach:**
  1. The forward PDE MC solve produces trajectories
  2. Solving the linear adjoint equation needs to be careful and consistent
- **The DTO approach:**
  1. The forward PDE MC solve produces trajectories

Summary (Adjoint Monte Carlo Method)

- **The OTD approach:**
  1. The forward PDE MC solve produces trajectories
  2. Solving the linear adjoint equation needs to be careful and consistent
- **The DTO approach:**
  1. The forward PDE MC solve produces trajectories
  2. Use the same randomness from the forward simulation and back-propagate adjoint Monte Carlo particles γ (**no more sampling!**)

Summary (Adjoint Monte Carlo Method)

- **The OTD approach:**
  1. The forward PDE MC solve produces trajectories
  2. Solving the linear adjoint equation needs to be careful and consistent
- **The DTO approach:**
  1. The forward PDE MC solve produces trajectories
  2. Use the same randomness from the forward simulation and back-propagate adjoint Monte Carlo particles γ (**no more sampling!**)

Summary (Adjoint Monte Carlo Method)

- **The OTD approach:**
  1. The forward PDE MC solve produces trajectories
  2. Solving the linear adjoint equation needs to be careful and consistent
- **The DTO approach:**
  1. The forward PDE MC solve produces trajectories
  2. Use the same randomness from the forward simulation and back-propagate adjoint Monte Carlo particles γ (**no more sampling!**)

Some Remarks

- Both approaches only requires MC forward solve with sampling
- The adjoint MC method is always consistent with the forward MC solves.
- We **can** differentiate random processes
- Main contribution of this sequence of works:

Generalize adjoint-state method to differentiate random Monte Carlo particles