Greedy algorithm for Parametric Vlasov-Fokker-Planck System

Yuhua Zhu

University of Wisconsin-Madison

Enrique Zuazua

University of Deusto

January 22, 2018

1 Numerical experiments

Consider the one dimensional linear Vlasov-Fokker-Planck (VPFP) as following.

$$
\begin{cases}
\partial_t f + \sigma_1 v \partial_x f - \frac{\sigma_2}{\epsilon} \partial_x \phi \partial_v f = \frac{\sigma_3}{\epsilon} \partial_v \left(v f + \partial_v f \right), & t > 0, x \in [0, 2\pi], v \in \mathbb{R}, \\
f(0, x, v, z) = f_0(x, v, z), & z \in [a, b].
\end{cases}
\tag{1.1}
$$

In the numerical experiments, **we compare the results from the greedy algorithm with the results from the polynomial interpolation of the quadrature points.** We set $\epsilon = 1, T = 0.5$ for all examples. More specifically, we set **error** $= \sum_{i=1}^{50} \int \left(\hat{f}(T, x, v, z_i) - f(T, x, v, z_i) \right)^2 dx dv$, where $\hat{f}(T, x, v, z_i)$ is the approximation solution obtained from either method, $f(T, x, v, z_i)$ is the exact solution. z_i is uniformly chosen from $[a, b]$. **Dimension** refers to the number of snap shots.

Observe Figure 1-6, we found that the Greedy algorithm converges to the solution faster than the other method. Especially, when there is discontinuity, the polynomial interpolation doesn't converge as dimension increases, however, the Greedy algorithm still converges and almost keeps the same convergent rate.

Algorithm 1.1. The algorithm for the polynomial interpolation:
For **Dimension** $= n$, $\{\alpha_i\}_{i=1}^n$ *is the n roots of n-th order Legendre polynomial, and let*

$$
z_i^* = \frac{b-a}{2} \alpha_i + \frac{a+b}{2}, \quad 1 \le i \le n.
\tag{1.2}
$$

$$
f_i = f(T, x, v, z_i^*) \text{ be the exact solution at time T.}
\tag{1.3}
$$

Then by Lagrange interpolation,

$$
\hat{f}(T, x, v, z) = \sum_{i=1}^{n} l_i(z) f_i, \quad l_i = \prod_{j \ne i} \frac{z - z_j^*}{z_i^* - z_j^*}
\tag{1.4}
$$

1

1.1 Linear VFP with parametric initial data

In the first experiment, we consider the case where the parameter $z \in [1, 5]$ only involved in the initial data,

$$\sigma_1 = \sigma_2 = \sigma_3 = 1, \quad \partial_x \phi = \cos(x), \tag{1.5}$$

$$\text{Figure 1:} \quad f_0(x, v, z) = \frac{1}{\sqrt{2\pi}} \left(2 + \frac{\sin(x)}{z} \right) \exp \left(-\frac{\left| v + \cos(x) + \frac{z}{10} \right|^2}{2} \right). \tag{1.6}$$

$$\text{Figure 2:} \quad f_0(x, v, z) = \begin{cases} \frac{1}{\sqrt{2\pi}} \left(2 + \frac{\sin(x)}{z} \right) \exp \left(-\frac{\left| v + \cos(x) + \frac{z}{10} \right|^2}{2} \right), & z < 3 \\ \frac{1}{\sqrt{2\pi}} \left(2 + \frac{\sin(x)}{z} \right) \exp \left(-\frac{\left| v + \cos(x) + \frac{z+1}{10} \right|^2}{2} \right), & z \geq 3. \end{cases} \tag{1.7}$$

Algorithm 1.2. The algorithm for PDE with only parameter in initial data:
Offline:

Step 1: Set $\mathcal{N} = \{z_i : z_i = i\delta_z, \delta_z = \frac{b-a}{N_z}, i = 0, 1, \ldots, N_z\}$, we set $N_z = 100$ in this case.

$$z_1^* = \operatorname*{argmax}_{z \in \mathcal{N}} \| f_0(x, v, z) \|^2, \tag{1.8}$$

where $\| f \|^2 = \int f^2 \, dx dv$.

Step 2: After obtaining $z_i^*, f_i = f_0(x, v, z_i^*), i = 1, \ldots, n - 1$,

$$z_n^* = \operatorname*{argmax}_{z \in \mathcal{N} \backslash \{z_i^*\}_{i=1}^{n-1}} \left\| f_0(x, v, z) - P^{n-1} f_0(x, v, z) \right\|^2, \tag{1.9}$$

where $P^{n-1} f_0$ is the projection onto the subspace spanned by $f_1, \ldots, f_{n-1}$

Step 3: Stops when

$$\max_{z \in \mathcal{N} \backslash \{z_i^*\}_{i=1}^{n-1}} \left\| f_0(x, v, z) - P^{n-1} f_0(x, v, z) \right\|^2 \leq \epsilon \tag{1.10}$$

Online: *After we obtained the snap shots $\{z_i^*\}_{i=1}^n$,*

Step 1: Orthogonalize $\{f_i\}_{i=1}^n$, and calculate the corresponding $\{f_i^T = f(T, x, v, z_i)\}_{i=1}^n$.

Step 2: for any given z, one can approximate $f(T, x, v, z)$ by

$$a_i = \langle f_0(x, v, z), f_i \rangle, \quad \hat{f}(x, v, z) = \sum_{i=1}^n a_i f_i^T. \tag{1.11}$$

1.2 Linear parametric VFP with deterministic initial data

In the second experiment, we consider the case where the parameter $z \in [1, 5]$ only involved in the PDE,

$$f_0(x, v, z) = \frac{1}{\sqrt{2\pi}} (2 + \sin(x)) \exp \left(-\frac{|v + \cos(x)|^2}{2} \right), \tag{1.12}$$

$$\text{Figure 3:} \quad \sigma_1 = 1 + \frac{1.5}{z}, \quad \sigma_2 = \sigma_3 = 1 + 5e^{\frac{1}{z}}, \quad \partial_x \phi = \cos(x) + \frac{z}{10}; \tag{1.13}$$

$$\text{Figure 4:} \quad \sigma_1 = 1 + \frac{1.5}{z}, \quad \sigma_2 = \sigma_3 = 1 + 5e^{\frac{1}{z}}, \quad \partial_x \phi = \begin{cases} \cos(x) + \frac{z}{10}, & z < 3 \\ \cos(x) + \frac{z+1}{10}, & z \geq 3 \end{cases} \tag{1.14}$$
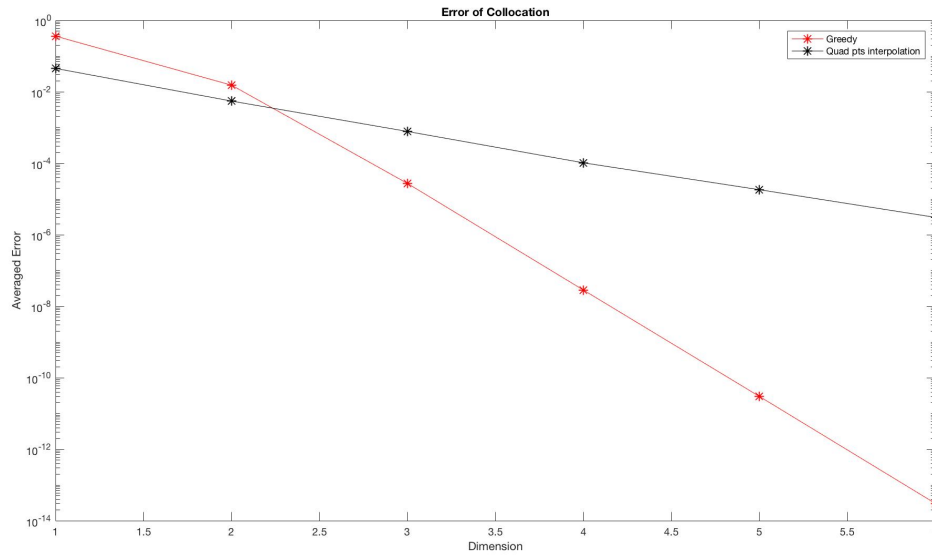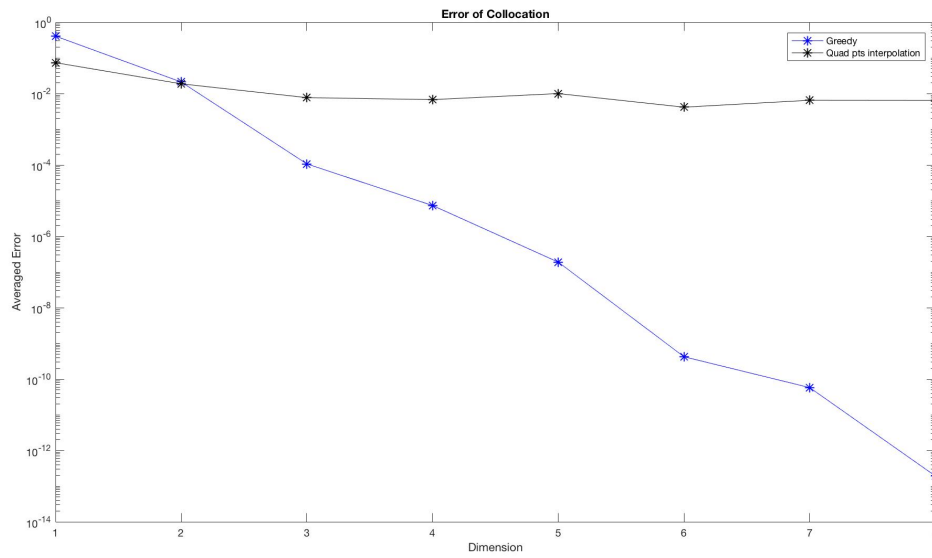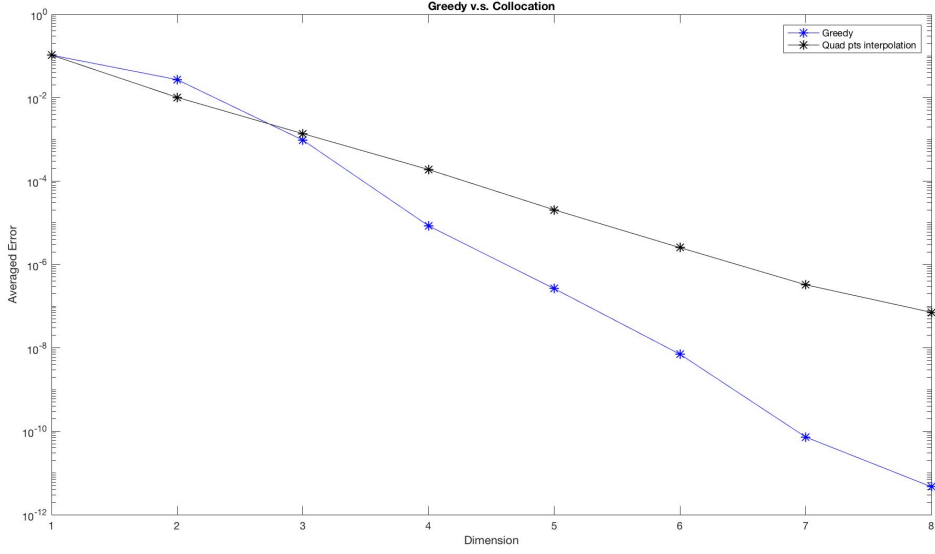
2

Figure 1: Continuous initial data

Figure 2: Discontinuous initial data

Figure 3: Continuous parameter in PDE

Algorithm 1.3. The algorithm for PDE with only parameters in PDE*:*

Offline: *Set $\mathcal{N} = \{z_i : z_i = i\delta_z, \delta_z = \frac{b-a}{N_z}, i = 0, 1, \ldots, N_z\}$, we set $N_z = 100$ in this case.*

Step 1: choose $z_1^* = \text{argmax}_{z \in \mathcal{N}} \|(\sigma_1, \sigma_2, \sigma_3)\|^2$.

Step 2: After obtaining $z_i^*, f_i = f(T, x, v, z_i^*)$, $i = 1, \ldots, n-1$,

$$z_n^* = \underset{z \in \mathcal{N} \setminus \{z_i^*\}_{i=1}^{n-1}}{\text{argmax}} \left[\min_{\hat{f} = \sum a_i f_i, \sum a_i = 1} \left\| \partial_t \hat{f} + \underbrace{\sigma_1(z) v \partial_x \hat{f} - \frac{\sigma_2(z)}{\epsilon} \partial_x \phi \partial_v \hat{f} - \frac{\sigma_3(z)}{\epsilon} \partial_v \left(v\hat{f} + \partial_v \hat{f} \right)}_{A(z)\hat{f}} \right\|^2 \right], \tag{1.15}$$

Step 3: Stops when the above maximum residual smaller than ϵ.

Online: *After we obtained the snap shots $\{z_i^*\}_{i=1}^n$, and the corresponding $f_i = f(T, x, v, z_i)$,*

$$\hat{f} = \underset{\hat{f} = \sum a_i f_i, \sum a_i = 1}{\text{argmin}} \left\| \partial_t \hat{f} + \underbrace{\sigma_1(z) v \partial_x \hat{f} - \frac{\sigma_2(z)}{\epsilon} \partial_x \phi \partial_v \hat{f} - \frac{\sigma_3(z)}{\epsilon} \partial_v \left(v\hat{f} + \partial_v \hat{f} \right)}_{A(z)\hat{f}} \right\|^2 \tag{1.16}$$
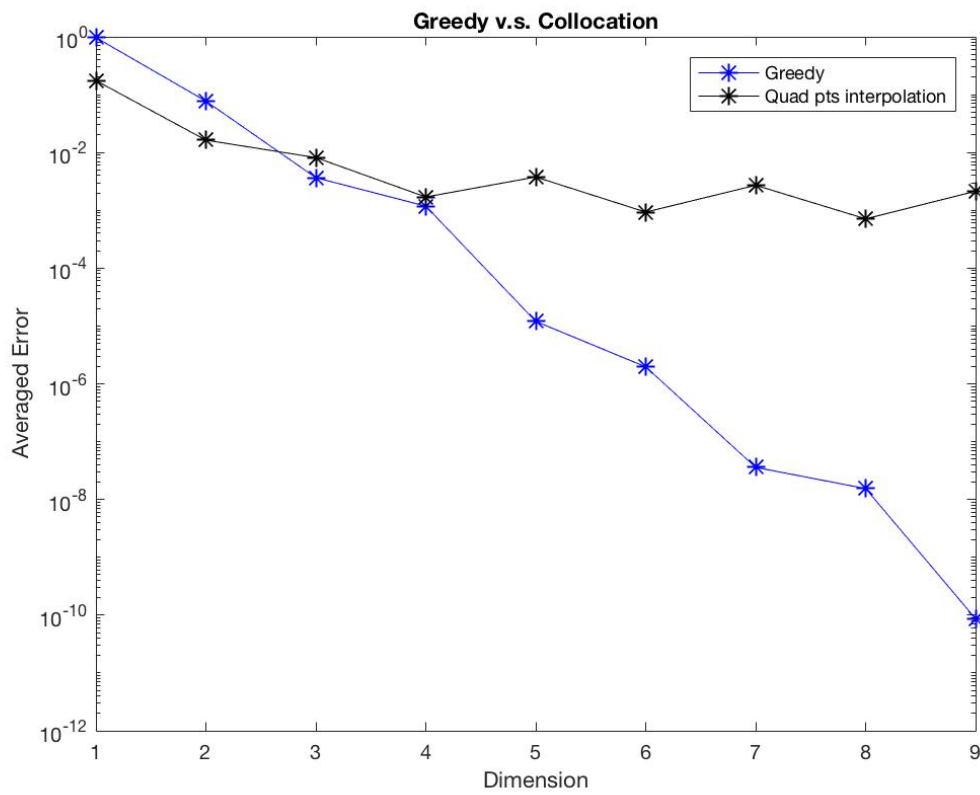
4

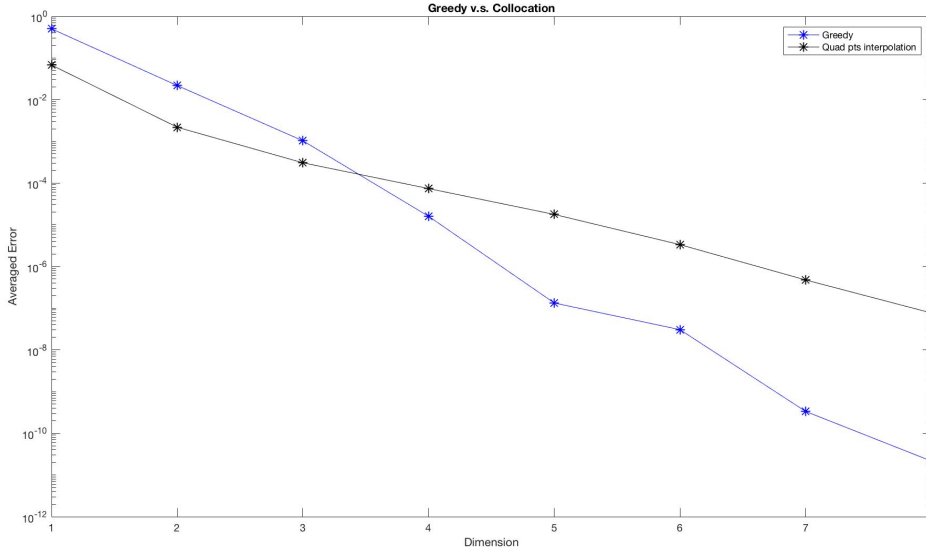**Figure 4:** Discontinuous parameter in PDE

Figure 5: Continuous parameter in PDE and initial data

1.3   Linear parametric VFP with parametric initial data

In the third experiment, we consider the case where the parameter $z \in [1,5]$ involved in both the initial data and the PDE,

$$\sigma_1 = 1 + \frac{1.5}{z}, \quad \sigma_2 = \sigma_3 = 1 + 5e^{\frac{1}{z}}, \quad f_0(x,v,z) = \frac{1}{\sqrt{2\pi}} \left(2 + \frac{\sin(x)}{z} \right) \exp\left(-\frac{|v + \partial_x \phi|^2}{2} \right).$$

Figure 5: $\partial_x \phi = \cos(x) + \dfrac{z}{10}, \quad \text{for } t \geq 0,$ \hfill (1.17)

Figure 6: $\partial_x \phi = \begin{cases} \cos(x) + \frac{z}{10}, & z < 3 \\ \cos(x) + \frac{z+1}{10}, & z \geq 3 \end{cases} \quad \text{for } t \geq 0.$ \hfill (1.18)

\hfill (1.19)

Algorithm 1.4. The algorithm for PDE with parameters in PDE and initial data:
Offline: *Set $\mathcal{N} = \{z_i : z_i = i\delta_z, \delta_z = \frac{b-a}{N_z}, i = 0, 1, \ldots, N_z\}$, we set $N_z = 10000$ in this case.*

Step 1: Using Algorithm 1.4 to obtain 50 snap shots out of $\mathcal{N}$ to form a new set $\mathcal{N}^*$.
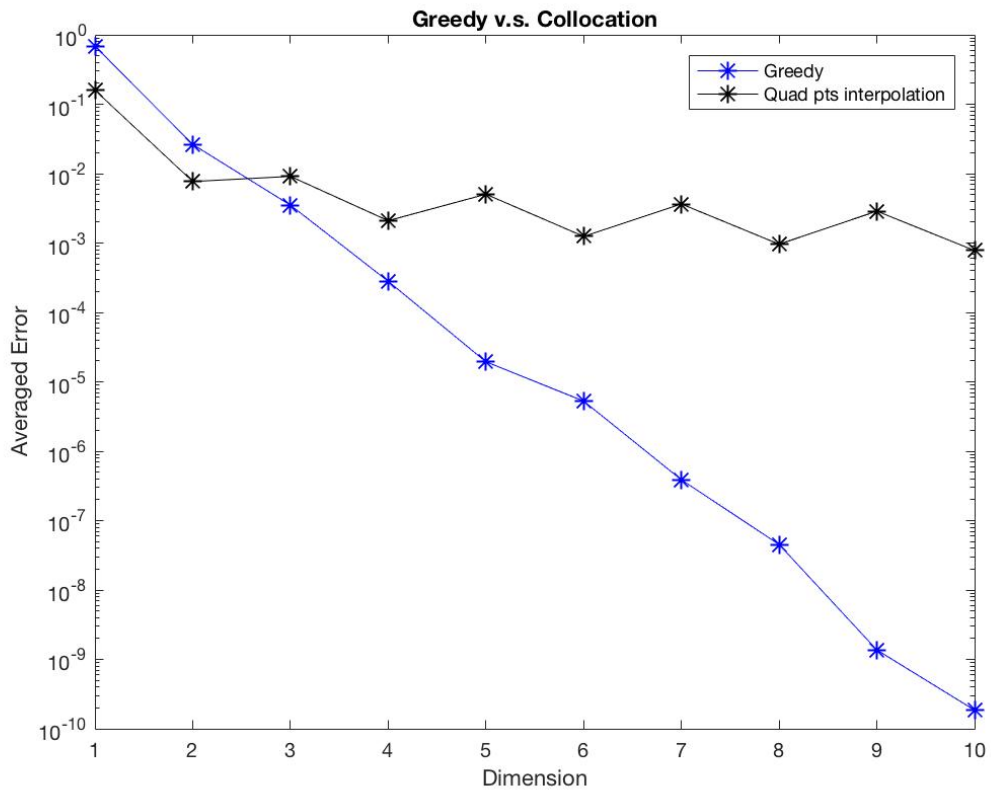
Step 2: Using Algorithm 1.3 to get snap shot $\{z_i^*\}_{i=1}^n$.

Online: *Same as Algorithm 1.3.*

Remark 1.5. *In the experiments, the exact solution $f(T, x, v, z)$ refers to the numerical solution to VFP, that is, $f(T, x, v, z)$ is a $N_x \times N_v$ dimensional vector obtained by the following scheme*

$$\frac{f_{i,j}^{n+1} - f_{i,j}^n}{\delta_t} + \sigma_1 \left(v \partial_x f \right)_{i,j}^n = \frac{\sigma_3}{\epsilon} P(f_{i,j}^{n+1}) \hfill (1.20)$$

where $P(f)$ is the discretization of $\partial_v \left(M \partial_v \left(\frac{f}{M} \right) \right)$, $M = \frac{1}{\sqrt{2\pi}} e^{-\frac{\left| v + \frac{\sigma_2}{\sigma_3} \partial_x \phi \right|^2}{2}}$.

6

Figure 6: Discontinuous parameter in PDE and initial data

Remark 1.6. *For the matlab coding, the algorithm for the there examples is in the "graph.m" file of the folder "eq1", "eq2", "eq3" respectively. You can change the coefficients in "sig_1", "sig_2", "sig_3", and the electric field in "fcn_phi_x", and initial data in "fcn_phi_x_0", "fcn_rho_0".*

  P.S. Remember to change the path of other functions at the beginning of "graph.m".