Fast Calculate for Evolution Operators

Fu Bin
<fubin1991@outlook.com>

June 21, 2016

0.1 Introduction

Suppose we want to calculate an evolution operator multiply a vector, namely,

$$g(\epsilon) = \sum_{n,m} \exp^{i\varepsilon_n t} M_{nm} \exp^{-i\epsilon_m^* t} \tag{1}$$

and matrix M can be viewed as a vector. We mark the matrix M as $M = (c_1, c_2, ..., c_n) = \{c\}$.
If we calculate this expression at different time and combine them in a vector, we can get this matrix:

$$\begin{aligned}
\mathbf{G}^<(\mathbf{t_l}, \mathbf{t_l}) &= \sum_{n,m} \exp^{i\varepsilon_n t} M_{nm} \exp^{-i\epsilon_m^* t} \\
&= \begin{pmatrix} e^{i\varepsilon_1} M_{11} e^{-i\epsilon_1^*} + e^{i\varepsilon_2} M_{21} e^{-i\epsilon_1^*} + \cdots + e^{i\varepsilon_1} M_{12} e^{-i\epsilon_2^*} + \cdots + e^{i\varepsilon_n} M_{nm} e^{-i\epsilon_m^*} \\ e^{2\times i\varepsilon_1} M_{11} e^{-2\times i\epsilon_1^*} + e^{2\times i\varepsilon_2} M_{21} e^{-2\times i\epsilon_1^*} + \cdots + e^{2\times i\varepsilon_1} M_{12} e^{-2\times i\epsilon_2^*} + \cdots + e^{2\times i\varepsilon_n} M_{nm} e^{-2\times i\epsilon_m^*} \\ \vdots \\ \vdots \\ e^{N_T\times i\varepsilon_1} M_{11} e^{-N_T\times i\epsilon_1^*} + e^{N_T\times i\varepsilon_2} M_{21} e^{-N_T\times i\epsilon_1^*} + \cdots + e^{N_T\times i\varepsilon_1} M_{12} e^{-N_T\times i\epsilon_2^*} + \cdots + e^{N_T\times i\varepsilon_n} M_{nm} e^{-N_T\times i\epsilon_m^*} \end{pmatrix}
\end{aligned}$$

If we define A as

$$A = \begin{pmatrix} e^{i\varepsilon_1} e^{-i\epsilon_1^*} & e^{i\varepsilon_1} e^{-i\epsilon_2^*} & \cdots & e^{i\varepsilon_1} e^{-i\epsilon_m^*} \\ e^{i\varepsilon_2} e^{-i\epsilon_1^*} & e^{i\varepsilon_2} e^{-i\epsilon_2^*} & \cdots & e^{i\varepsilon_2} e^{-i\epsilon_m^*} \\ \vdots & \vdots & \ddots & \vdots \\ e^{i\varepsilon_n} e^{-i\epsilon_1^*} & e^{i\varepsilon_n} e^{-i\epsilon_2^*} & \cdots & e^{i\varepsilon_n} e^{-i\epsilon_m^*} \end{pmatrix} \tag{4}$$

$$\equiv \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} \tag{5}$$

And finally we can express this equation:

$$\mathbf{G}^<(\mathbf{t_l}, \mathbf{t_l}) = \tag{6}$$

$$\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 & \cdots & 1 \\ A_{11} & A_{21} & \cdots & A_{n1} & A_{12} & A_{22} & \cdots & A_{n2} & \cdots & A_{nm} \\ A_{11}^2 & A_{21}^2 & \cdots & A_{n1}^2 & A_{12}^2 & A_{22}^2 & \cdots & A_{n2}^2 & \cdots & A_{nm}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ A_{11}^{N_T} & A_{21}^{N_T} & \cdots & A_{n1}^{N_T} & A_{12}^{N_T} & A_{22}^{N_T} & \cdots & A_{n2}^{N_T} & \cdots & A_{nm}^{N_T} \end{pmatrix} \begin{pmatrix} M_{11} \\ M_{21} \\ \vdots \\ M_{n1} \\ M_{12} \\ M_{22} \\ \vdots \\ M_{n2} \\ \vdots \\ M_{nm} \end{pmatrix} \tag{7}$$

In order to speed up this calculation, we need use Vandermonde matrix. The Vandermonde matrix can be express as below

$$\mathbf{V} = \begin{pmatrix} 1 & a_1 & a_1^2 & \ldots & a_1^{n-1} \\ 1 & a_2 & a_2^2 & \ldots & a_2^{n-1} \\ 1 & a_3 & a_3^2 & \ldots & a_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_m & a_m^2 & \ldots & a_m^{n-1} \end{pmatrix} \tag{8}$$

By using Vandermonde matrix, we can express equation(1) in transposed Vandermonde matrix. We first defined

$$b \equiv V^t c \tag{9}$$

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ a_1 & a_2 & a_3 & \dots & a_m \\ a_1^2 & a_2^2 & a_3^2 & \dots & a_m^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1^{n-1} & a_2^{n-1} & a_3^{n-1} & \dots & a_m^{n-1} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{pmatrix} \tag{10}$$

$$= \begin{pmatrix} c_1 + c_2 + c_3 + \ldots + c_m \\ c_1 a_1 + c_2 a_2 + c_3 a_3 + \ldots + c_m a_m \\ c_1 a_1^2 + c_2 a_2^2 + c_3 a_3^2 + \ldots + c_m a_m^2 \\ \vdots \\ c_1 a_1^m + c_2 a_2^m + c_3 a_3^m + \ldots + c_m a_m^m \end{pmatrix} \tag{11}$$

where c is a vector in matrix M.
A direct computation shows that the entries of $b = V_a^t c$ are the first $m+1$ coefficients of the Taylor expansion of

$$S(x) = \sum_{j=0}^{m} \frac{c_j}{1 - a_j x} = \sum_{n}^{\infty} \sum_{j=0}^{m} c_j \left(a_j\right)^n x^n = \sum_{n} b_n \tag{12}$$

where $b_n = \sum_{j=0}^{m} c_j (a_j)^n x^n$ and we have used the Taylor expansion

$$\frac{1}{1 - a_j x} = \sum_{k=0}^{\infty} \left(a_j x\right)^k \tag{13}$$

If we use Fourier transform, where $x = \omega_{N_T}^l$ and $\omega_{N_T} = \exp^{i \frac{2\pi}{N_T}}$, we can get

$$\bar{S}(l) = \bar{S}(\omega_{N_T}^l) = \sum_{j=0}^{N} \sum_{n=0}^{N_T} c_j (a_j)^n \omega_{N_T}^{nl} \tag{14}$$

$$= \sum_{n=0}^{N_T} \left(\sum_{j=0}^{N} c_j (a_j)^n \right) \omega_{N_T}^{nl} \tag{15}$$

$$= \sum_{j=0}^{N} c_j \frac{1 - \left(a_j \omega_{N_T}^l\right)^{N_T+1}}{1 - a_j \omega_{N_T}^l} \tag{16}$$

$$= \sum_{j=0}^{N} \frac{c_j}{\left(\frac{1}{\omega_{N_T}}\right)^l - a_j} - \omega_{N_T}^{l(N_T+1)} \sum_{j=0}^{N} \frac{c_j a_j^{N_T+1}}{\left(\frac{1}{\omega_{N_T}}\right)^l - a_j} \tag{17}$$

$$= \omega_T^{-l} \sum_{j=0}^{N-1} \frac{c_j (1 - a_j^T)}{(1/\omega_T)^l - a_j} \tag{18}$$

Now we estimate the computational complexity for $T \leq N$. For FMM we need $\kappa_1 max(T, N)$ operations where κ_1 is about $40 \log_2(1/\tau)$ with τ the tolerance. For FFT the computational complexity is at most $\kappa_2 N \log_2 N$ where κ_2 is a coefficient for FFT calculation. To compute $V^t M$ where M has N vectors, we have to calculate $V^t c$ N times. Hence the total computational complexity is $\kappa_1 N^2 + \kappa_2 N^2 \log_2 N$. For $T = N = 10^4$, numerical calculation using FMM and FFT shows that $\kappa_1 N^2$ dominates due to large κ_1 and the speed up factor is about 8 over TN^2 scaling discussed in the main text. In the calculation, FMM costs about 48 seconds and FFT costs about 11 seconds.
For very large T up to $T = N^2$ (if $N = 10^4$ we have $T = 10^8$), we will show that the computational complexity is $\kappa_1 N^2 + 2\kappa_2 N^2 \log_2 N$. In fact, it is easy to see that $I(t_j)$ is the first T coefficients of the Taylor expansion of

$$S(x) = \sum_{n,m=0}^{N-1} \frac{M_{nm}}{1 - a_n a_m^* x} \tag{19}$$

$$= \sum_{j}^{\infty} \sum_{n,m=0}^{N-1} M_{nm} (a_n a_m^*)^j x^j \tag{20}$$

where $a_n = \exp(-i\epsilon_n)$. Now we define two new vectors u and d which have N^2 components with $u^t = (c_0^t, c_1^t, ..., c_{N-1}^t)$ and $d^t = (a_0^* a^t, a_1^* a^t, ..., a_{N-1}^* a^t)$. With the new vectors defined, $S(x)$ is expressed as

$$S(x) = \sum_{j=0}^{N^2-1} \frac{u_j}{1 - d_j x} \tag{21}$$

In this new form, the computational complexity is $\kappa_1 N^2 + \kappa_2 N^2 \log_2 N^2$. In this case, for $N = 10^4$ and $T = 10^8$, if we use 10 levels, FMM will take 3116 seconds; if we use 11 levels, FMM will take 4203 seconds. The FFT will take 50 seconds.

0.2 Program Summary

0.3 Appendix

Recalling the discrete Fourier transform and inverse Fourier transform,

$$F(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n) e^{-2\pi i k n/N} \tag{22}$$

$$f(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) e^{2\pi i k n/N} \tag{23}$$

We can find in equation(14), the second equation is the Fourier transform, which is

$$f(n) = \sum_{j=0}^{N} c_j (a_j)^n \tag{24}$$

and

$$F(l) = \sum_{n=0}^{N_T} f(n) e^{-2\pi i n l/N_T} = \sum_{n=0}^{N_T} f(n) \omega_{N_T}^{-nl} \tag{25}$$

$$f(n) = \sum_{l=0}^{N_T} F(l) e^{2\pi i n l/N_T} = \sum_{l=0}^{N_T} F(l) \omega_{N_T}^{nl} \tag{26}$$

In the equation(14), we can first calculate $\bar{S}(l)$ by the forth equation and then using inverse Fourier transform(26) to get $f(n) = \sum_{j=0}^{N} c_j (a_j)^n$.
If we consider evolution operator equation(1),we can get

$$\begin{aligned} b_n &= \sum_m \exp^{i\epsilon_m t_n} c_m &(27)\\ &= \sum_m \exp^{i\epsilon_m \triangle \cdot n} c_m &(28)\\ &= \sum_m \left(\exp^{i\epsilon_m \triangle}\right)^n c_m &(29)\\ &= \sum_m a_m^n c_m &(30)\end{aligned}$$

So we can get

$$\mathbf{b} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ e^{i\epsilon_1 \triangle} & e^{i\epsilon_2 \triangle} & e^{i\epsilon_3 \triangle} & \dots & e^{i\epsilon_m \triangle} \\ e^{2i\epsilon_1 \triangle} & e^{2i\epsilon_2 \triangle} & e^{2i\epsilon_3 \triangle} & \dots & e^{2i\epsilon_m \triangle} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e^{ni\epsilon_1 \triangle} & e^{ni\epsilon_2 \triangle} & e^{ni\epsilon_3 \triangle} & \dots & e^{ni\epsilon_m \triangle} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix} \tag{31}$$

We have defined a_m and separated time variable $t_n = n \cdot \triangle$, where $n = 1, 2, ..., N_T$.