Fast Computation of Multilinear Operators

Haizhao Yang
Department of Mathematics
Texas Consortium for Computational Seismology
University of Texas at Austin

Advisor: Lexing Ying

Outline

- Multilinear operator in harmonic analysis
- Direct computation of multilinear operator
- Fast algorithm for multilinear operator
  - Fast Fourier transform and non uniform fast Fourier transform
  - Matrix decomposition
  - Matrix approximation

Multilinear Operators

- Definition: For k fixed, we consider the *k*-linear operator

$$(f_1,\dots,f_k) \mapsto M(f_1,\dots,f_k)$$

initially defined for $f_k \in S$ as follows. We write

$$\overline{\xi} = (\xi^{(1)},\dots,\xi^{(k)}) \in R^{nk}, \qquad \xi^{(j)} \in R^n,$$

and set $\sigma(\overline{\xi}) = \xi^{(1)} + \dots + \xi^{(k)}$. If $\overline{\xi} \mapsto m(\overline{\xi})$ is a bounded function on R^{nk}, we define multilinear operator *M* by

$$M(f_1,\dots,f_k)(x) = \int_{R^{nk}} e^{2\pi i x\cdot\sigma(\overline{\xi})} m(\overline{\xi})\, \hat{f}_1(\xi^{(1)}) \cdots\cdots \hat{f}_k(\xi^{(k)})\, d\overline{\xi}.$$

- Here we just talk about the cases when *n*=1 and *k*=2. So the multilinear operator is

$$M(f_1,f_2)(x) = \int_{R^2} e^{2\pi i x\cdot(\eta+\lambda)} m(\eta,\lambda)\, \hat{f}_1(\eta)\, \hat{f}_2(\lambda)\, d\eta\, d\lambda.$$

We can get the results for higher n and k similarly.

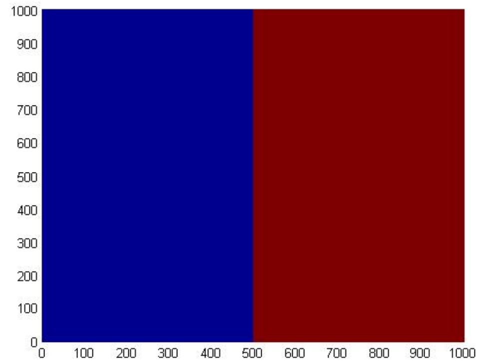
Multilinear Operators

- Some examples of *M:*

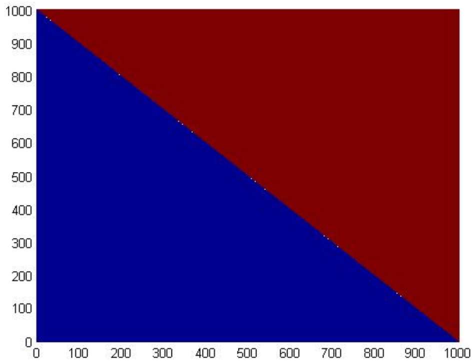
(1) $M(f_1, f_2)(x) = f_1(x) f_2(x)$, when $m(\eta, \lambda) \equiv 1.$

(2) $M(f_1, f_2) = f_1 H(f_2) - H(f_1 f_2)$, where *H* is the Hilbert transform, and

$$m(\eta, \lambda) = \frac{1}{i}[sign(\lambda) - sign(\eta + \lambda)].$$



Matrix m corresponding to $sign(\lambda)$



Matrix m corresponding to $sign(\eta + \lambda)$

Direct Computation

- Upon discretization of the multilinear operator, we get

$$M(f_1, f_2)(x) = \sum_{\eta} \sum_{\lambda} e^{2\pi i x \cdot (\eta + \lambda)} m(\eta, \lambda) \hat{f}_1(\eta) \hat{f}_2(\lambda),$$

where $\eta_s = s - 1 - N/2,\ \lambda_t = t - 1 - N/2,\ s, t = 1, \ldots, N,$
and $x_j = \dfrac{j-1}{2N-1},\ j = 1, \ldots, 2N-1.$

- We want to compute $M(f_1, f_2)(x_j)$ for all *j*. The time complexity of direct computation is $O(N^3)$.
- Note that we suppose f_1 and f_2 have support in [0,1] and discretize [0,1] with N points $\{0, \frac{1}{N}, \ldots, \frac{N-1}{N}\}$. So $\hat{f}_1$ and $\hat{f}_2$ take value at the points $\{-\frac{N}{2}, \ldots, \frac{N}{2} - 1\}$.

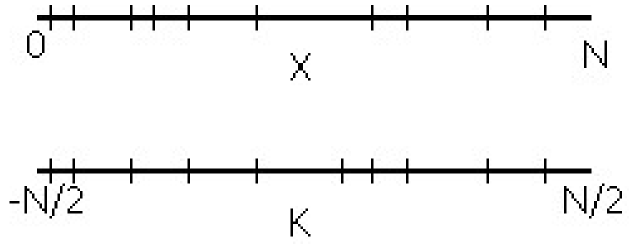
Fast Fourier transform

- Uniform $x \in X = \{0, 1, \ldots, N-1\}$
- Uniform $k \in K = \{-N/2, -N/2+1, \ldots, N/2-1\}$
- Fourier coeff $\{f(k), k \in \mathrm{K}\}$
- Fourier kernel $G(x,k) = e^{-2\pi i \frac{xk}{N}}$
- Compute *u(x)* for each *x* in X

$$u(x) = \sum_{k \in K} e^{-2\pi i \frac{xk}{N}} f(k)$$

- Order of FFT is $O(N \log N)$

Non-uniform fast Fourier transform



- Non-uniform $X \subset [0, N]$, and $|X| = O(N)$
- Non-uniform $K \subset [-N/2, N/2]$ and $|K| = O(N)$.
- Fourier coeff $\{ f(k), k \in K \}$
- Fourier kernel

$$G(x,k) = e^{-2\pi i \frac{xk}{N}}$$

- Compute *u(x)* for each *x* in X

$$u(x) = \sum_{k \in K} e^{-2\pi i \frac{xk}{N}} f(k)$$

- The idea of Non-uniform FFT: use equispaced data to approximate the non-uniform data.
- Order $O(N \log N + pN)$

$$u(x) \; = \; e^{-2\pi i \frac{xk}{N}} \; f(k)$$

We use nearest p equispaced data of x and k to get the approximation, where p is depending on error ε

Fast Algorithm for Multilinear Operator

(a) Special case for constant multiplication function

When the multiplier function $m(\eta, \lambda)$ is a constant, then

$$M(f_1, f_2)(x) = \sum_{\eta} \sum_{\lambda} e^{2\pi i x \cdot (\eta + \lambda)} \hat{f}_1(\eta) \hat{f}_2(\lambda)$$

$$= \sum_{\xi} \sum_{\eta} e^{2\pi i x \xi} \hat{f}_1(\eta) \hat{f}_2(\xi - \eta)$$

$$= \sum_{\xi} e^{2\pi i x \xi} \sum_{\eta} \hat{f}_1(\eta) \hat{f}_2(\xi - \eta)$$

$$= \sum_{\xi} e^{2\pi i x \xi} T(\xi)$$

where we let $\xi = \eta + \lambda$, and $T(\xi) = \sum_{\eta} \hat{f}_1(\eta) \hat{f}_2(\xi - \eta)$

Fast Algorithm for Multilinear Operator

- Observation:

  (1) $T(\xi)$ is the convolution of $\hat{f}_1(\eta)$ and $\hat{f}_2(\lambda)$.

  So $T = \hat{f}_1 * \hat{f}_2 = ((2\pi)^{d/2} \hat{\hat{f}}_1 \hat{\hat{f}}_2)^{\vee}$.

  (2) $M(f_1, f_2)(x)$ is the inverse Fourier transform of $T(\xi)$.

- Ideas: Use FFT to compute the Fourier transform above to reduce the time complexity, supposing that the data is uniformly distributed.

- Time complexity for T: $O(N\log N) + O(N\log N) + O(N) + O(N\log N) = O(N\log N)$

  Time complexity for $M(f_1, f_2)(x) = \sum_{\xi} e^{2\pi i x\xi} T(\xi)$ is $O(N\log N)$.

  Total time complexity to compute *M* is $O(N\log N)$.

- If the data is non-uniform, then the time complexity is $O(N\log N + pN)$ using NFFT.

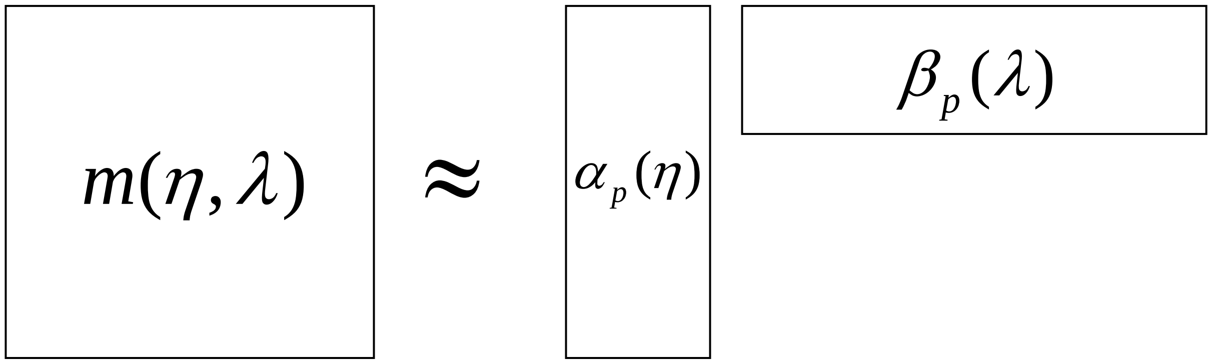
Fast Algorithm for Multilinear Operator

(b) Smooth multiplicative function

- Suppose that $m(\eta,\lambda)$ is smooth enough, it has a *h-term* ε*-expansion about* η *and* λ, which means there exists functions $\{\alpha_p(\eta)\}_{1\le p\le h}$ and $\{\beta_p(\lambda)\}_{1\le p\le h}$ such that

$$|\, m(\eta,\lambda) - \sum_{p=1}^{h} \alpha_p(\eta)\beta_p(\lambda)\,| \le \varepsilon.$$

- We can use the random method provided by V. Rokhlin, B. Engquist and L.Ying to get the *h-term* ε*-expansion of it with time complexity* $O(hN)$ *for a* $N \times N$ *matrix.*



Fast Algorithm for Multilinear Operator

- Then

$$M(f_1, f_2)(x) = \sum_{\eta}\sum_{\lambda} e^{2\pi i x\cdot(\eta+\lambda)} m(\eta,\lambda)\,\hat{f}_1(\eta)\,\hat{f}_2(\lambda)$$

$$= \sum_{\eta}\sum_{\lambda} e^{2\pi i x\cdot(\eta+\lambda)} \sum_{p=1}^{h} \alpha_p(\eta)\beta_p(\lambda)\hat{f}_1(\eta)\,\hat{f}_2(\lambda)$$

$$= \sum_{\eta}\sum_{\lambda} e^{2\pi i x\cdot(\eta+\lambda)} \sum_{p=1}^{h} [\alpha_p(\eta)\,\hat{f}_1(\eta)][\beta_p(\lambda)\hat{f}_2(\lambda)]$$

$$= \sum_{\xi} e^{2\pi i x\xi} \sum_{p=1}^{h}\sum_{\eta} [\alpha_p(\eta)\,\hat{f}_1(\eta)][\beta_p(\lambda)\,\hat{f}_2(\lambda)]$$

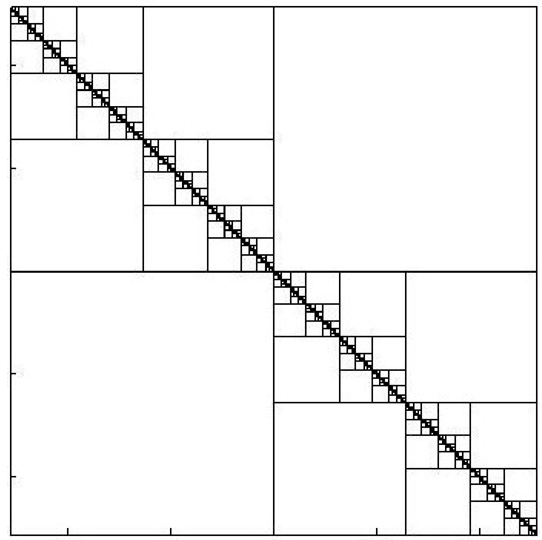
$$= \sum_{\xi} e^{2\pi i x\xi} \sum_{p=1}^{h} T_p(\xi)$$

- Time complexity is $O(hN \log N)$ for uniform data and $O(hN \log N + hpN)$ for non-uniform data for a single matrix of size N.

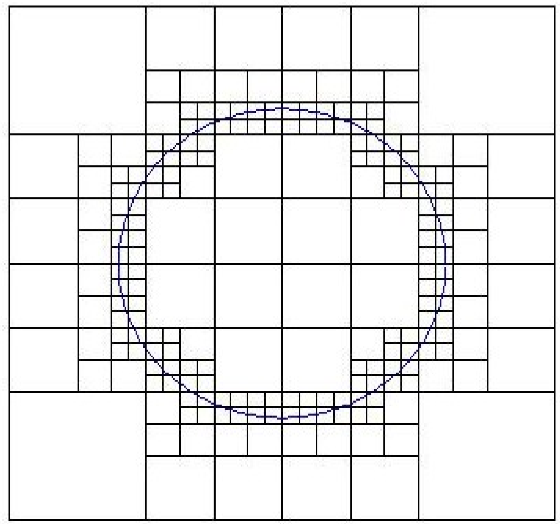
Fast Algorithm for Multilinear Operator

(c) P.W. smooth multiplicative function

- If $m(\eta, \lambda)$ is discontinuous along a curve, we can use matrix division to get the small square domains in which the function is smooth.



- Number of squares：
  - *O(N) squares of size 1 by 1*
  - *O(N/2) squares of size 2* by 2
  - …
  - *O(1)* squares of size *N* by *N*
  - Total number of squares is *O(N).*



- Time complexity of low rank approximation:

$$O(h \cdot 1 \cdot N) + O(h \cdot 2 \cdot \frac{N}{2}) + \ldots + O(h \cdot N \cdot 1)$$
$$= O(hN \log N)$$

Fast Algorithm for Multilinear Operator

- *Time complexity for uniform data:*

$$O(Nh \cdot 1 \log 1 + Nh) + O(\frac{N}{2} h \cdot 2 \log 2 + \frac{N}{2} h \cdot 2) + \ldots$$

$$+O(1 \cdot h \cdot N \log N + 1 \cdot h \cdot N) \le O(hN(\log N)^2)$$

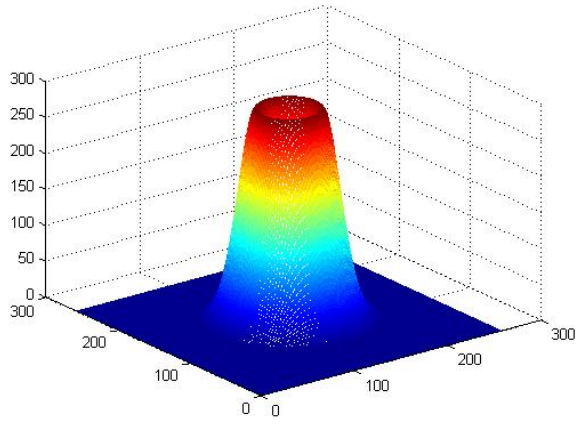
- *For non-uniform data, it's*

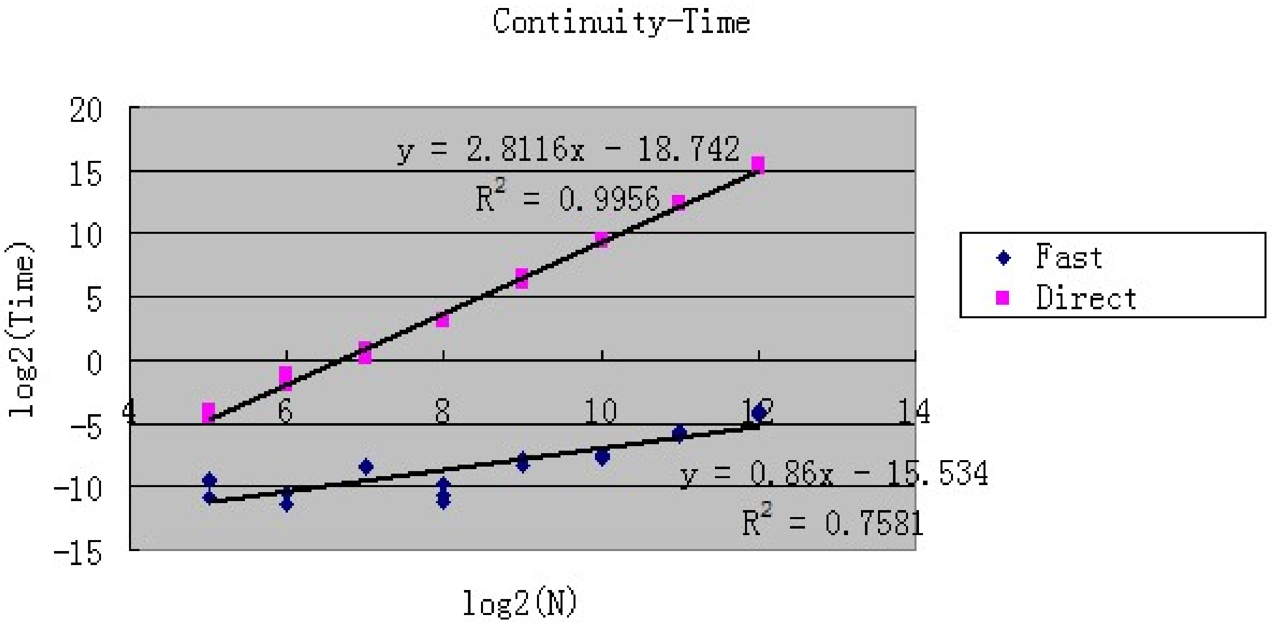
$$O(hN(\log N)^2 + hpN \log N)$$

Summary of ideas

- Division of matrices according to the discontinuous curve, so that the function $m(\eta, \lambda)$ is smooth enough.
- Use the random method to get the low rank approximation of $m(\eta, \lambda)$.
- Use the low rank approximation to get the form of separated variables
$$m(\eta, \lambda) \approx \sum_{p=1}^{h} \alpha_p(\eta)\beta_p(\lambda),$$
such that we can reduce the general cases to the simplest case in which $m(\eta, \lambda)$ is constant.
- Use FFT or DFFT to compute the multilinear operator in the simplest case.

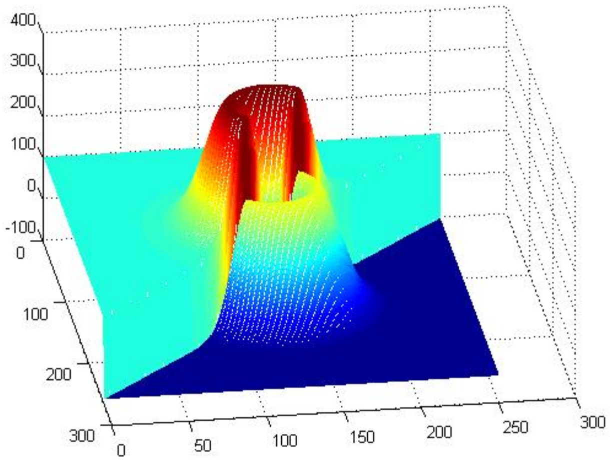
Numerical Examples

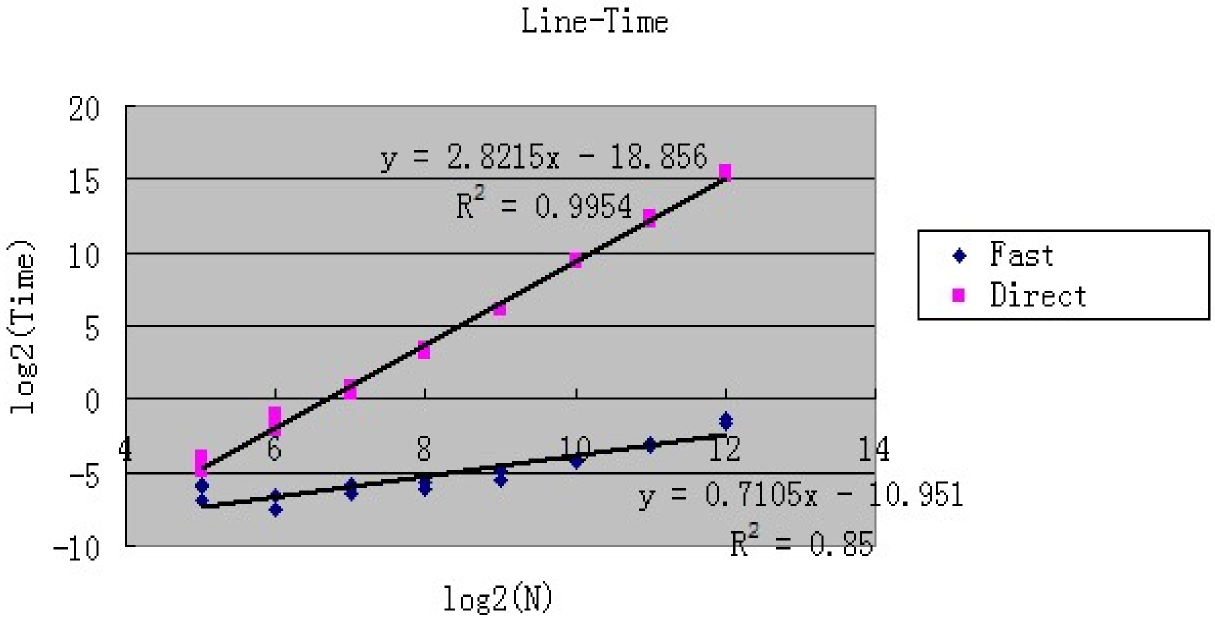




Let $m(\eta,\lambda)=(\eta^2+\lambda^2)e^{-(\eta^2+\lambda^2)/C}$ and increase the number N of data to compare the time complexity. Direct computation is approximately $O(N^3)$ and the fast algorithm has almost linear order.

Numerical Examples

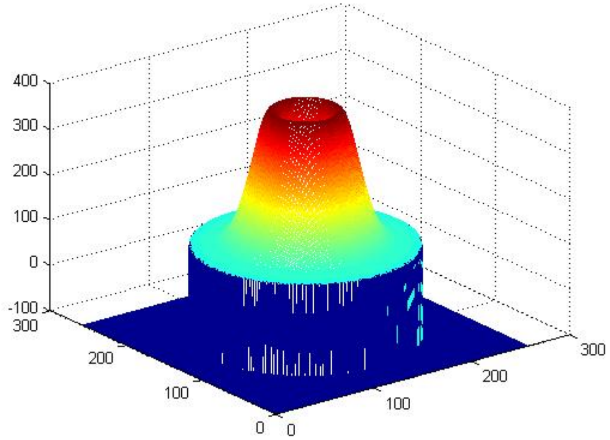


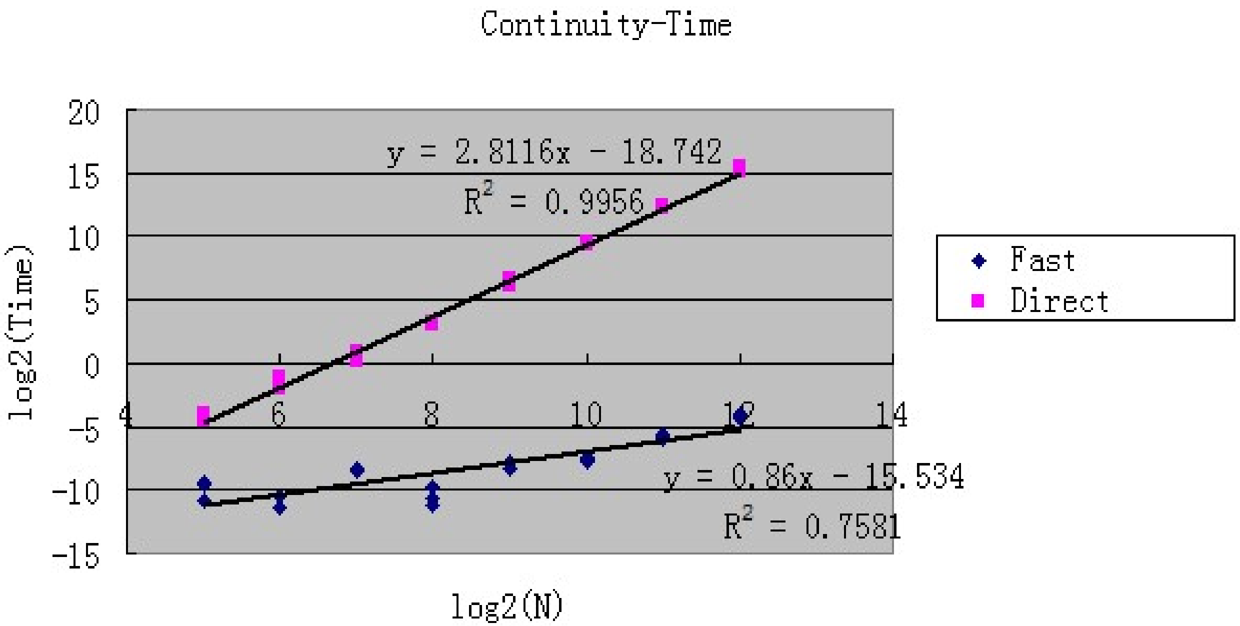


Increasing the number N of data to compare the time complexity. Direct computation is approximately $O(N^3)$ and the fast algorithm has almost linear order.

$$m(\eta,\lambda)=\begin{cases}(\eta^2+\lambda^2)e^{-(\eta^2+\lambda^2)/C}+100 & \textit{if} \quad \eta+\lambda\le 0,\\ (\eta^2+\lambda^2)e^{-(\eta^2+\lambda^2)/C}-100 & \textit{if} \quad \textit{otherwise.}\end{cases}$$

Numerical Examples

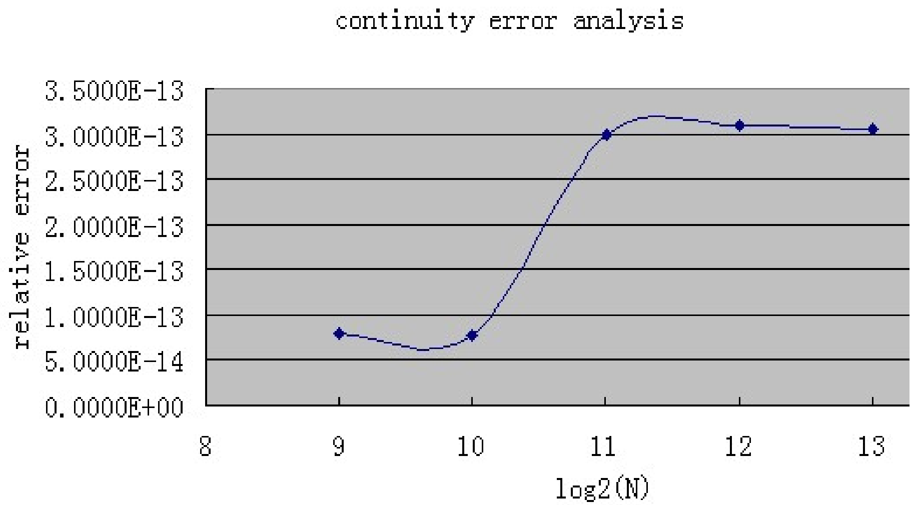


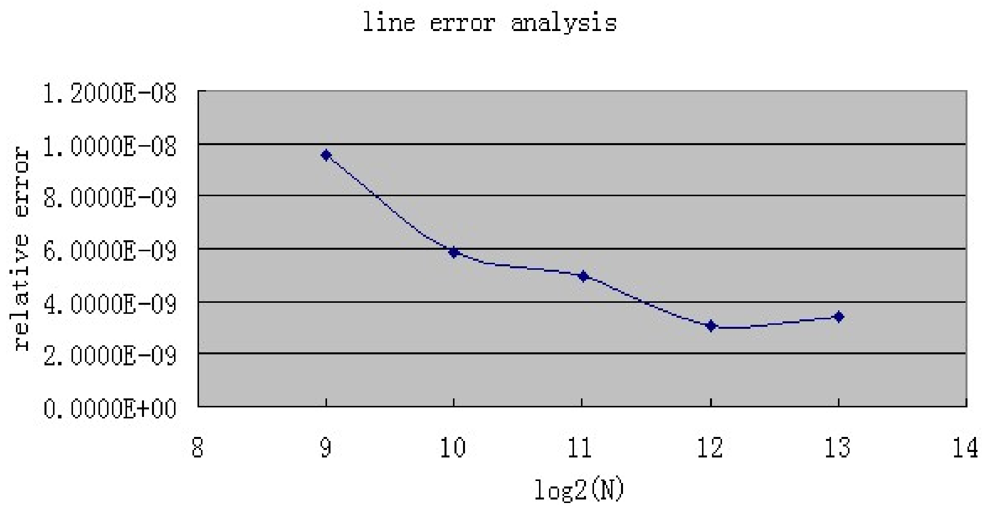


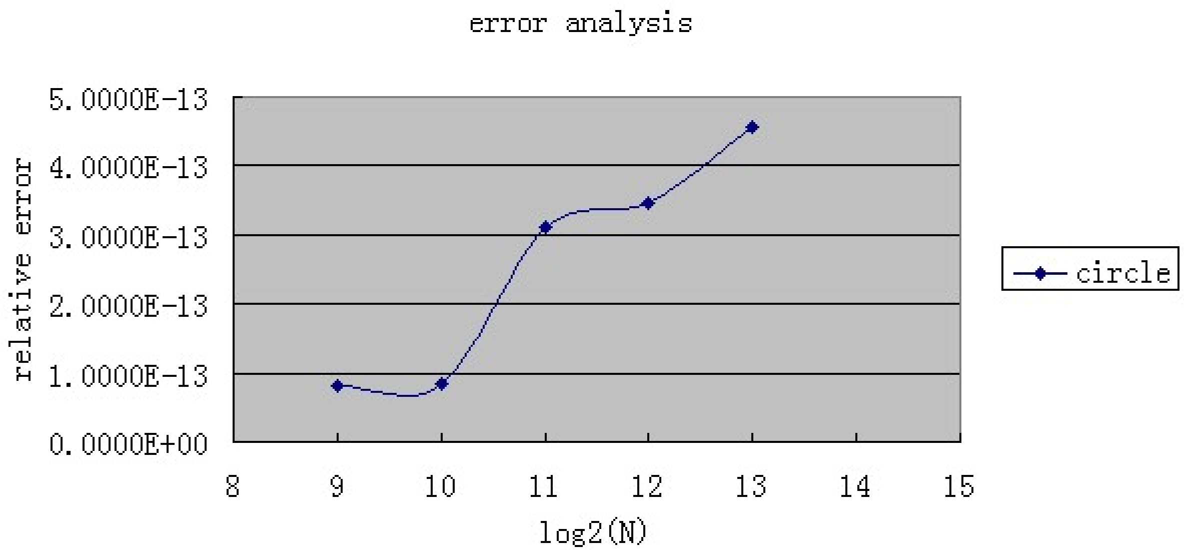
Increasing the number N of data to compare the time complexity. Direct computation is approximately $O(N^3)$ and the fast algorithm has almost linear order.

$$m(\eta,\lambda)=\begin{cases}(\eta^2+\lambda^2)e^{-(\eta^2+\lambda^2)/C}+100 & if \quad (\eta)^2+(\lambda)^2\le R^2, \\ (\eta^2+\lambda^2)e^{-(\eta^2+\lambda^2)/C}-100 & if \quad otherwise.\end{cases}$$

Numerical Examples







Thank you!