Algorithms, Implementation and Applications of pFFT++:

More on projection and interpolation

Zhenhai Zhu

RLE Computational prototyping group, MIT

www.mit.edu/people/zhzhu/pfft.html

Outline

- Error bound
- Loss of accuracy in double-layer potential
- Consistent polynomials
- Compact stencil
- Selection of three stencil sizes
- Implementation details

1D polynomial fit

Suppose the function is

$$f(x) = \frac{1}{x}$$

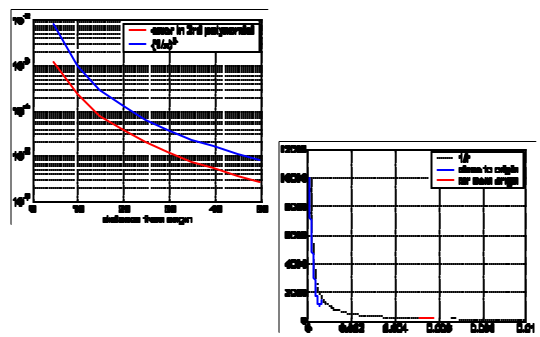
Using a second-order polynomial to interpolate around x_0, the <u>relative</u> error is

$$e \approx \left(\frac{h}{x_0}\right)^3$$

where h is the uniform grid spacing.

See "Introduction to numerical analysis" by Stoer and Bulirsch

1D polynomial fit



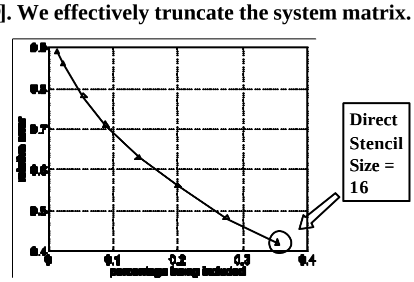
Error bound in pfft++:

Two factors in determining the error:

1. **Direct stencil size**
   Larger size means more interactions are calculated directly and larger distance to non-neighbor elements.
2. **Interpolation and projection size**
   Higher order means lower interpolation error.

Error bound in pfft++: Truncation error

Suppose we only keep the direct interaction and ignore far field completely, i.e. let
$[A] = [D]$. We effectively truncate the system matrix.



Error bound in pfft++: 3D interpolation error

Given (x_l, y_l, z_l) and $f_l = f(x_l, y_l, z_l)$, use n-th order polynomial to approximate $f(x,y,z)$

$$P_n(x,y,z) = \sum_{m=0}^{n}\sum_{i=0}^{m}\sum_{j=0}^{m-i} C_{ij}\, x^i y^j z^{m-i-j}$$

Let

$$F(x,y,z) = f(x,y,z) - P_n(x,y,z) - w_{n+1}(x,y,z)$$

where

$$w_{n+1}(x,y,z) = \sum_{i=0}^{n+1}\sum_{j=0}^{n+1-i} k_{ij}(x-x_l)^i (y-y_l)^j (z-z_l)^{n+1-i-j}$$

is the leading error term

Error bound in pfft++: 3D interpolation error

For 1/r kernel, the leading term in error is

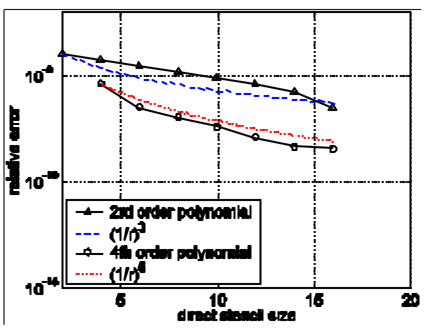
$$\frac{1}{r}\left(\frac{h}{r}\right)^{n+1}\sum_{i=0}^{n+1}\sum_{j=0}^{n+1-i}\left(\frac{\partial r}{\partial x}\right)^i\left(\frac{\partial r}{\partial y}\right)^j\left(\frac{\partial r}{\partial z}\right)^{n+1-i-j}$$

So the relative error is $e \approx \left(\frac{h}{r_0}\right)^{n+1}$

where h is the uniform grid spacing and r_0 is the distance between the source and the evaluation point.

<u>**Derivation shown with chalk and board**</u>

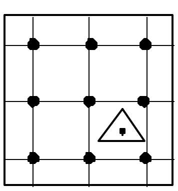
Error bound in pfft++



Loss of Accuracy in double-layer potential

	$P = 3$	$P = 5$	$P = 7$
$\frac{1}{r}$	8.4e-5	1.3e-6	4.3e-9
$\frac{\partial}{\partial n}\frac{1}{r}$	8.5e-3	1.1e-4	8.4e-7
$\frac{e^{ikr}}{r}$ $kR = 1.1e\text{-}9$	8.3e-5	1.3e-6	1.7e-9
$\frac{\partial}{\partial n}\frac{e^{ikr}}{r}$ $kR = 1.1e\text{-}9$	6.0e-3	7.5e-5	5.9e-7
$\frac{e^{ikr}}{r}$ $kR = 11.1$	4.9e-4	1.1e-5	4.0e-7
$\frac{\partial}{\partial n}\frac{e^{ikr}}{r}$ $kR = 11.1$	1.4e-2	2.8e-4	6.5e-6

Reminder of Interpolation Algorithm



Given $\bar{f}_g$

Compute $f(x,y)$

Reminder of Interpolation Algorithm

$$f(x,y) = \sum_k c_k f_k(x,y) = \bar{f}^{\,t}(x,y)\bar{c}$$

An example of $f_k(x,y)$:

$$1, x, x^2, y, xy, x^2y, y^2, xy^2, x^2y^2$$

Reminder of Interpolation Algorithm

If the kernel has a differential operator outside:

$$\frac{\partial}{\partial n(\vec r)}\int_S dS'\, G(\vec r,\vec r\,')\,\sigma(\vec r\,')$$

The operator works on the interpolation

$$\frac{\partial}{\partial n(\vec r)} f(\vec r) = \frac{\partial}{\partial n(\vec r)}\, \overline{f}^{\,t}(\vec r) F^{-1}\overline{f}_g$$

Loss of Interpolation Order

$$\frac{\partial \overline{f}^{\,t}}{\partial n} = n_x \frac{\partial \overline{f}^{\,t}}{\partial x} + n_y \frac{\partial \overline{f}^{\,t}}{\partial y}$$

$$\overline{f}^{\,t} = 1, x, x^2, y, xy, x^2y, y^2, y^2x, y^2x^2$$

$$\frac{\partial \overline{f}^{\,t}}{\partial x} = 0, 1, 2x, 0, y, 2xy, 0, y^2, 2y^2x$$

$$\frac{\partial \overline{f}^{\,t}}{\partial y} = 0, 0, 0, 1, x, x^2, 2y, 2yx, 2yx^2$$

**We automatically lose interpolation order.
This is why double-layer is less accurate.**

Loss of Interpolation Order

- **Double-layer still has reasonable accuracy because of smoothness of far field.**
- **We can simply increase interpolation order to compensate the loss of degree in derivative**
- **But stencil size grows exponentially**

order	2	4	6
# monomials	27	125	343
# stencil grid points	27	125	343

Consistent Polynomials

Suppose the highest order is *n*, using consistent polynomial, we have

$$f(x,y,z) = \sum_{m=0}^{n}\sum_{i=0}^{m}\sum_{j=0}^{m-i} c_{ijk}\, x^i y^j z^{m-i-j}$$

$$\text{total number of terms} = (n+1)(n+2)(n+3)/6$$

n	2	4	6
# monomials	10	35	84

Compact Stencil

Number of interpolation terms is much fewer than number of regular interpolation or projection stencil points. We have a least square problem

$$[F]^{-1}_{n_s \times n_p}$$

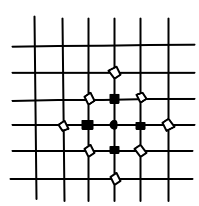
where n_s is number of stencil points and n_p is number of monomials. We could pick points from regular stencil points. Many options are possible.

Compact Stencil: 2D Example

Union of points equal distance from the origin of the interpolation stencil

$$\text{Cube stencil} = S_0 \cup S_1 \cup \frac{1}{2} S_2$$

$$\text{Compact stencil} = S_0 \cup S_1 \cup S_2$$



S_0: 1 solid dot
S_1: 4 solid squares
S_2: 8 empty diamonds

New Interpolation Scheme

- Same order of accuracy but with much fewer monomials and stencil points.
- Particularly useful for high-order interpolation

order	2	4
# monomials	27	35 (consistent)
# stencil grid points	27	57

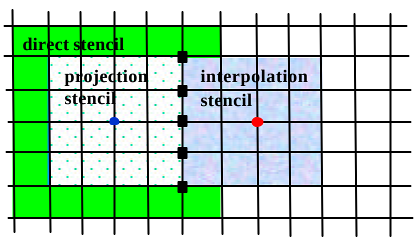
Compact stencil S_{012345} is used here.

Selection of Three Stencil Sizes

- Increase of direct stencil size is not cost effective
  - ❑ accuracy improves slowly
  - ❑ density of [*D*] increases relatively fast
- Increase of interpolation or projection stencil size improves accuracy exponentially.
  - ❑ Consistent polynomial makes the cost low
- There exists a constrain on these three stencil sizes.

Selection of Three Stencil Sizes

■ points shared by projection and interpolation stencil, corresponding entries in [*H*] are zero!



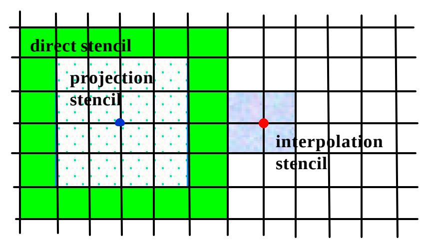
Center of direct and projection stencil

Center of interpolation stencil

Selection of Three Stencil Sizes

To make sure interpolation and projection Stencil don't touch, enforce

<u>directStencilSize >= projectStencilSize + interpStencilSize</u>



Selection of Three Stencil Sizes: rule of thumb

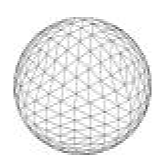
- Single-layer
  - ➢ 3-4 digit: I=P=1, D=2
  - ➢ 4-5 digit: I=P=1, D=3
  - ➢ 5-6 digit: I=P=2, D=4
- Double-layer
  - ➢ 2-3 digit: I=P=1, D=3
  - ➢ 4-5 digit: I=P=2, D=4
  - ➢ 5-6 digit: I=P=2, D=5 or 6
- pfft++ supports I=P=3, but try not to use it. It is too costly.

Numerical Experiments

On the surface of a sphere with radius *R*

$$\int_S dS' K(\bar{r},\bar{r}')\, r(\bar{r}') \Rightarrow Ax$$

Let *x* be a random vector



$$y_1 = Ax$$

$$y_2 = pfft(x)$$

$$error = \frac{\|y_1 - y_2\|_2}{\|y_1\|_2}$$

New Accuracy in double-layer potential

	l=*P* = 1, D=3	*l*=*P*=2, D=4 (Consistent poly)
$1/r$	8.4e-5	1.3e-6
$\frac{\partial}{\partial n} 1/r$	8.5e-3	3e-4
e^{ikr}/r *kR = 11.1*	4.9e-4	1.1e-5
$\frac{\partial}{\partial n} e^{ikr}/r$ *kR = 11.1*	1.4e-2	6.2e-4

Implementation: Source codes

- See stencil.cc
- See pfft.h

Next Lecture

- Direct matrix and pre-correction
- Grid selection