Numerical Issues In Fast Maxwell's Equation Solvers for Integrated Circuit Interconnect

***Zhenhai Zhu**, Ben Song*

and Jacob White

RLE Computational prototyping group, MIT

rleweb.mit.edu/vlsi

Outline

- <u>Background</u>
- Pre-corrected FFT algorithm
- Unit testing results
- Surface integral formulation
- Numerical Results

Mathematical Preliminaries

A simple integral equation:

$$\int_S dS' K(\vec{r},\vec{r}')\, r(\vec{r}') = f(\vec{r}), \quad \vec{r} \in S$$

$$K(\vec{r},\vec{r}') = \frac{1}{|\vec{r}-\vec{r}'|}, \qquad \frac{e^{ik|\vec{r}-\vec{r}'|}}{|\vec{r}-\vec{r}'|}$$

Project the solution on a functional space:

$$r_n(\vec{r}') = \sum_{j=1}^{n} a_j b_j(\vec{r}'), \quad B_n = \mathrm{span}\langle b_j(\vec{r}')\rangle$$

Mathematical Preliminaries

Residual:

$$\mathrm{e_n}(\vec{r}) = \int_S dS' K(\vec{r},\vec{r}\,')\, r_n(\vec{r}\,') - f(\vec{r})$$

Enforce the residual to be orthogonal to another functional space:

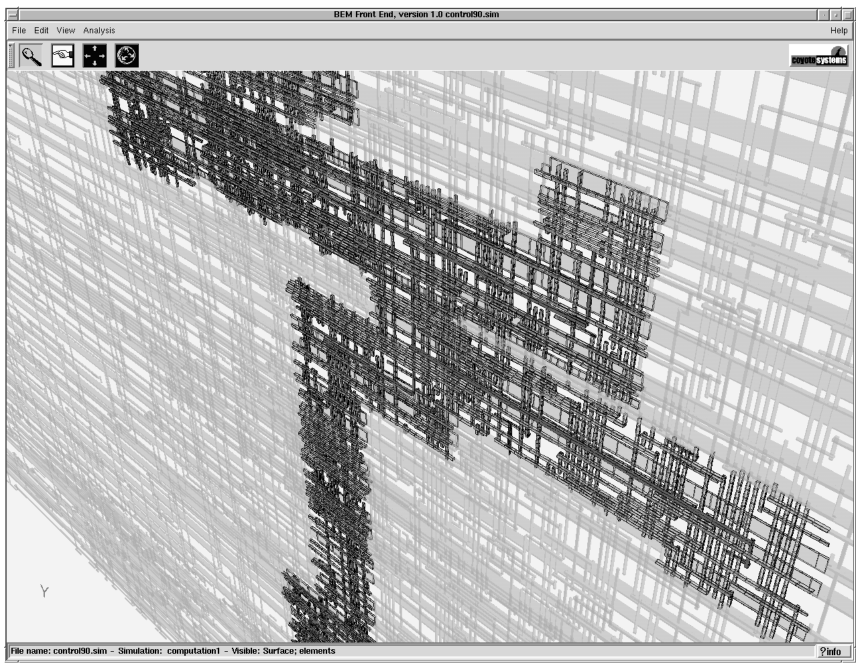
$$\langle t_i(\vec{r}), \mathrm{e_n}(\vec{r}) \rangle = 0,\ T_n = \mathrm{span}\langle t_i(\vec{r}) \rangle$$

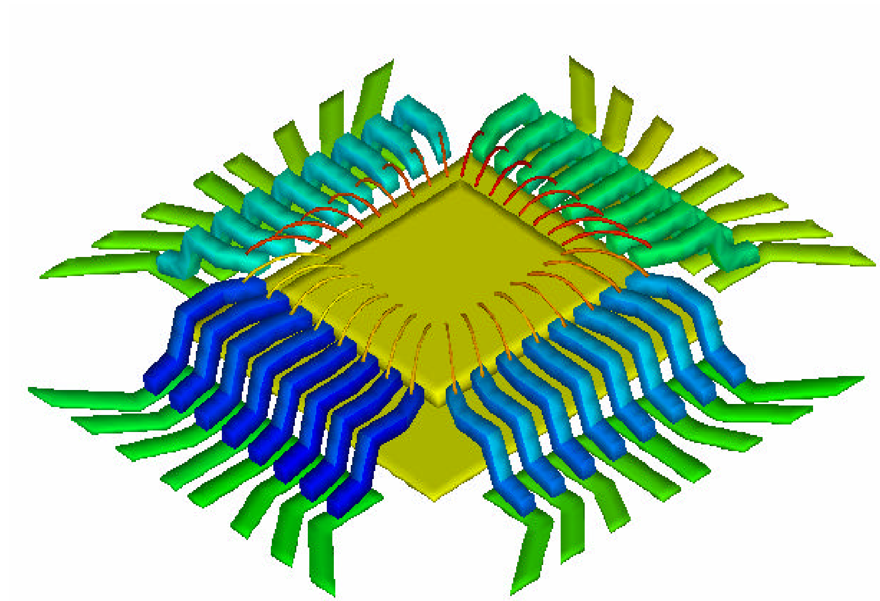
A dense linear system: $A\bar{a} = \bar{f}$

Some very useful applications

Electrostatic analysis to compute the capacitance

Magneto-quasi-static analysis to compute impedance

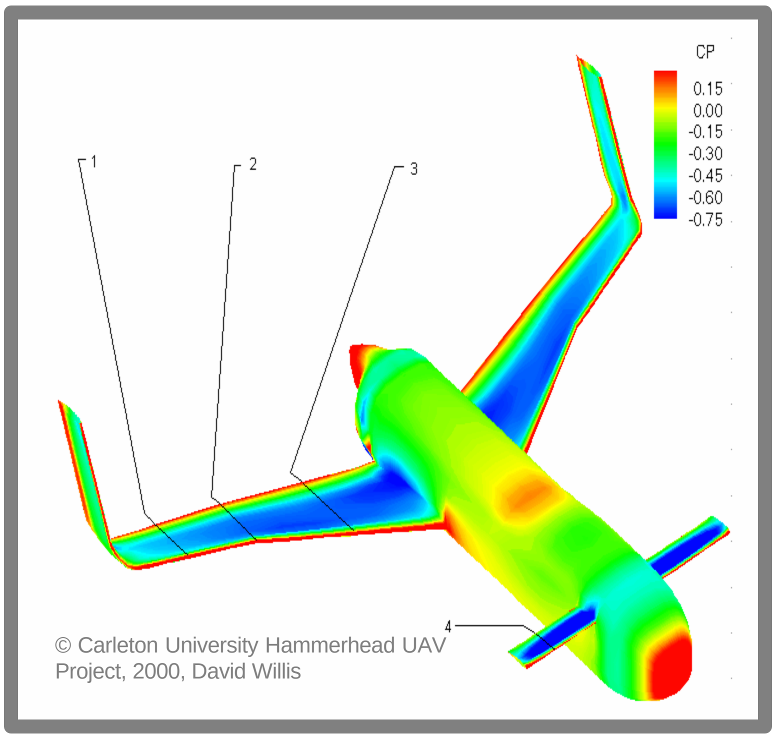




Figures thank to Coventor

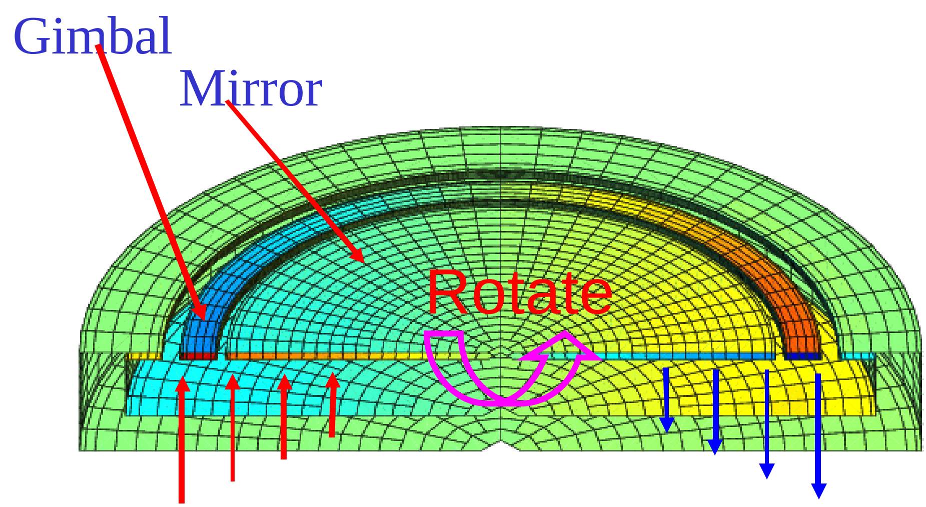
Some very useful applications

Computational Aerodynamics



Picture thanks to David Joe Willis

Stokes Flow Solver Viscous drag



Picture thanks to Xin Wang

Introduction to Iterative Solvers

A simple iterative solver:

Solve $Ax = b$

step 1: guess an initial solution x_0, let $k = 0$

step 2: compute the residual $r = b - Ax_k$

step 3: find an update $\triangle x$ from r

step 4: update the solution $x_{k+1} = x_k + \triangle x$

step 5: $k = k + 1$, go to step 2

Fast Matrix-Vector Product

The most expensive step:

$$Ax$$

Goal:

$$O(N^2) \quad \Rightarrow \quad O(N) \text{ or } O(N\log(N))$$

Well-known Fast Algorithms

- Fast Multiple Method
- Hierarchical SVD
- Panel Clustering Method

Key idea:

interaction matrix is low rank

Kernel “Independent” Technique

Basic requirements:

Reciprocity: $G(\vec{r},\vec{r}') = G(\vec{r}',\vec{r})$

Shift invariance: $G(\vec{r}+\triangle\vec{r},\vec{r}'+\triangle\vec{r}) = G(\vec{r},\vec{r}')$

Commonly used Green’s function all satisfy these requirements

$$\frac{1}{|\vec{r}-\vec{r}'|},\quad \frac{e^{ik|\vec{r}-\vec{r}'|}}{|\vec{r}-\vec{r}'|},\quad \frac{\partial}{\partial \mathrm{n}}\left(\frac{1}{|\vec{r}-\vec{r}'|}\right),\quad \frac{\partial}{\partial \mathrm{n}}\left(\frac{e^{ik|\vec{r}-\vec{r}'|}}{|\vec{r}-\vec{r}'|}\right)$$

Outline

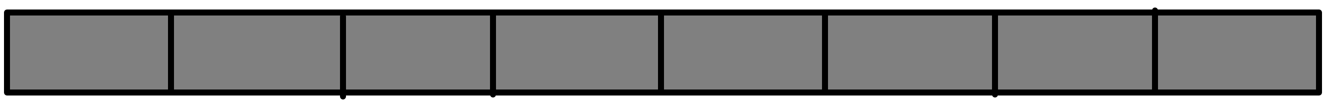
- Background
- <u>Pre-corrected FFT Algorithm</u>
- Unit testing results
- Surface Integral Formulation
- Numerical Results

FFT-based Method

Key idea: kernel is shift-invariant

$$G(\vec{r},\vec{r}') = G(\vec{r}-\vec{r}',0) = \tilde{G}(\vec{r}-\vec{r}')$$

A simple example:



$$\int_S dS' G(\vec{r},\vec{r}')\, r(\vec{r}') = f(\vec{r}), \quad \vec{r} \in S$$

$$H\bar{a} = \bar{f}$$

FFT-based Method

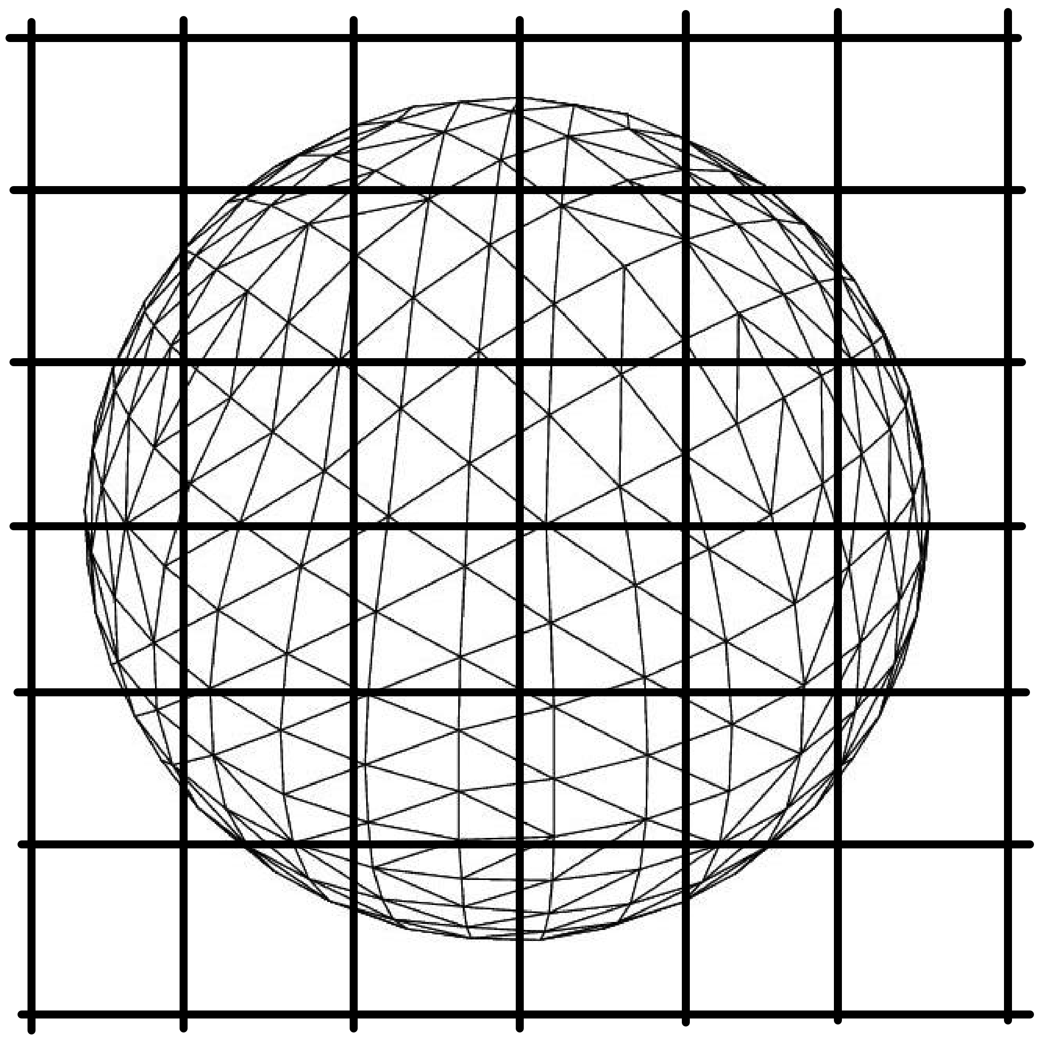
If collocation method with constant basis is used

$$H_{i,j} = \int_{panel_j} dS' \tilde{G}(\vec{r}_i - \vec{r}_j')$$

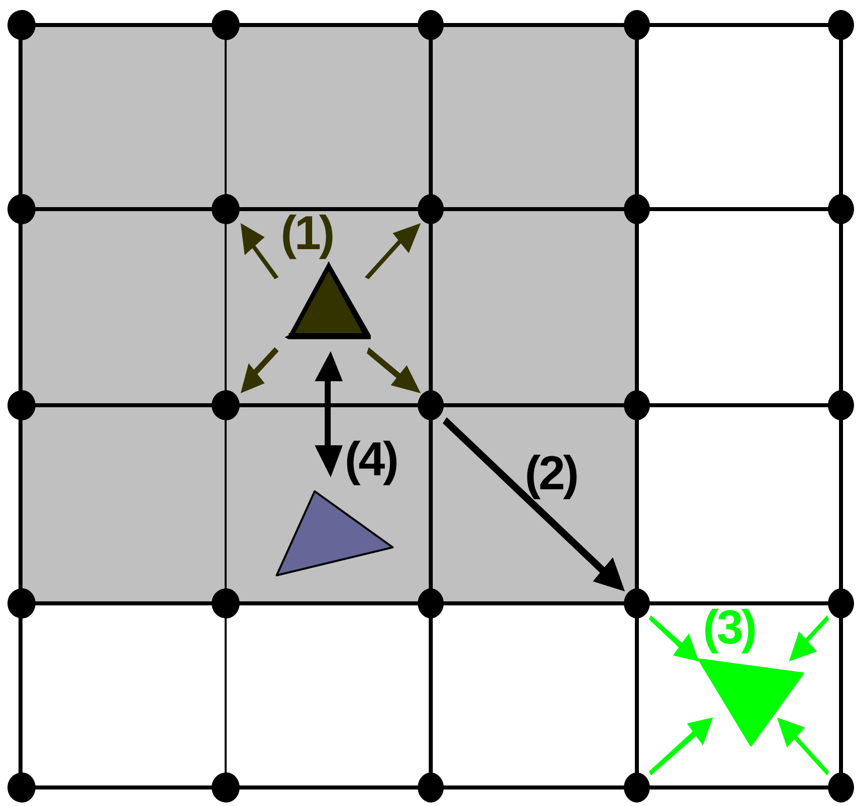
Only $H_{1,j}$ $(j = 1, 2, ..., N)$ are unique. H is a Toeplitz matrix. Matrix vector product could be computed using FFT in O(Nlog(N)) time.

Operations: O(Nlog(N))   Memory: O(N)

Separation of Regular Grid From Discretization Panels



pFFT Algorithm: Basic steps



(1) Project: $\overline{Q}_g = [P]\overline{a}$

(2) Convolve: $\overline{f}_g = [H]\overline{Q}_g$

(3) Interpolate: $\overline{\Psi}_g = [I]\overline{f}_g$

(4) Direct: $\overline{\Psi}_d = [D]\overline{a}$

$$\Psi = \Psi_g + \Psi_d = ([D] + [I][H][P])\overline{a}$$

pFFT Algorithm: Basic Idea

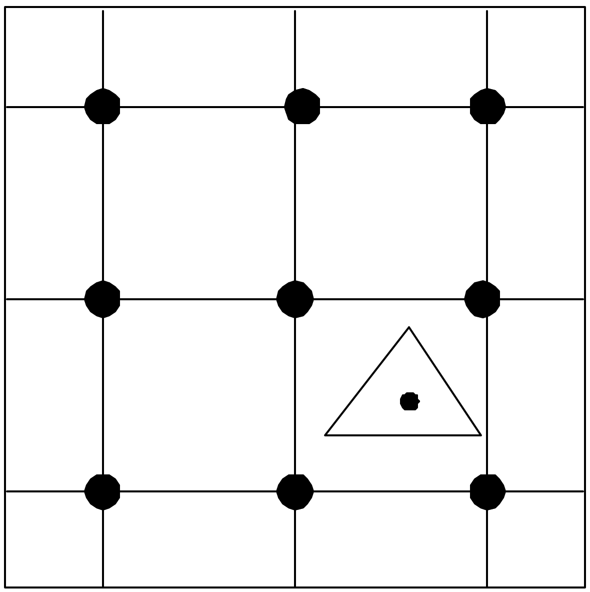
A sparse representation of the system matrix

$$[A]_{N_b \times N_b} = [D]_{N_b \times N_b} + [I]_{N_b \times N_g} [H]_{N_g \times N_g} [P]_{N_g \times N_b}$$

$$O(N_b^{\ 2}) \quad O(N_b) \quad O(N_b) \quad O(N_g \log(N_g)) \quad O(N_b)$$

$$O(N_b^{\ 2}) \quad O(N_b) \quad O(N_b) \quad O(N_g) \quad O(N_b)$$

pFFT Algorithm: Interpolation Matrix



Given $\overline{f}_g$

Compute $f(x, y)$

pFFT Algorithm: Interpolation Matrix

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

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

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

pFFT Algorithm: Interpolation Matrix

$$f(x,y) = \begin{bmatrix} f_1(x,y) & f_2(x,y) & \cdots & f_9(x,y) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_9 \end{bmatrix} = \bar{f}^{\,t}(x,y)\bar{c}$$

$$\bar{f}_g = \begin{bmatrix} f_{g,1} \\ f_{g,2} \\ \vdots \\ f_{g,9} \end{bmatrix} = \begin{bmatrix} f_1(x_1,y_1) & f_2(x_1,y_1) & \cdots & f_9(x_1,y_1) \\ f_1(x_2,y_2) & f_2(x_2,y_2) & \cdots & f_9(x_2,y_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_9,y_9) & f_2(x_9,y_9) & \cdots & f_9(x_9,y_9) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_9 \end{bmatrix} = [F]\bar{c}$$

$$f(x,y) = \bar{f}^{\,t}(x,y)[F]^{-1}\bar{f}_g$$

pFFT Algorithm: Interpolation Matrix

$$\Psi_i = \left\langle t_i(\vec{r}), f(\vec{r}) \right\rangle = \int_{\Delta_i^t} dS\, t_i(\vec{r})\, \bar{f}^{\,t}(\vec{r}) [F]^{-1} \bar{f}_g = \overline{W}^{\,t} \bar{f}_g$$

$$\overline{\Psi} = \begin{bmatrix} \vdots \\ \Psi_i \\ \vdots \end{bmatrix} = \begin{bmatrix} \cdots & W_1 & \cdots & W_9 & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ f_{g,1} \\ \vdots \\ f_{g,9} \\ \vdots \end{bmatrix} = [I] \bar{f}_g$$

Operations: $9N_b$ Memory: $9N_b$

pFFT Algorithm: Outer Differential Operator

If the kernel has a differential operator outside:

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

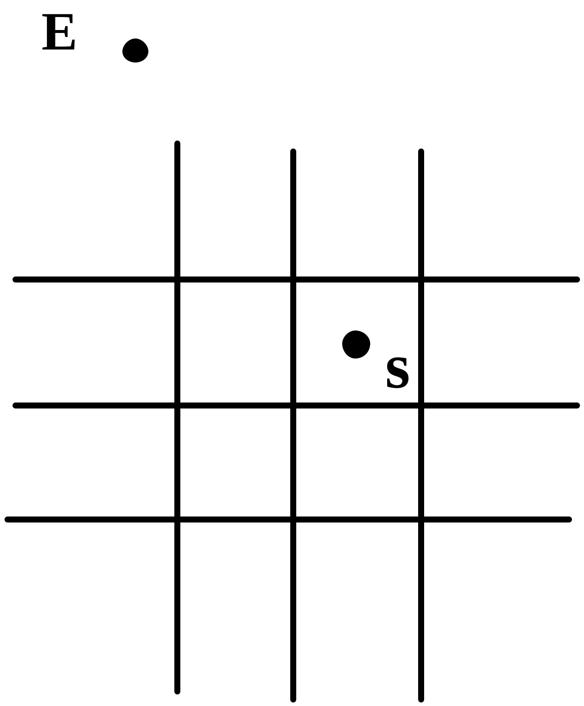
The operator works on the interpolation

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

$$\bar{W}_n^t = \int_{\Delta_i^t} dS\, t_i(\vec{r}) \frac{\partial}{\partial n(\vec{r})} \bar{f}^{\,t}(\vec{r}) \left[F\right]^{-1}$$

pFFT Algorithm: Projection Matrix

Assume a unit charge at point S



$$f_E^{(1)} = G(\vec{r}_s, \vec{r}_E)$$

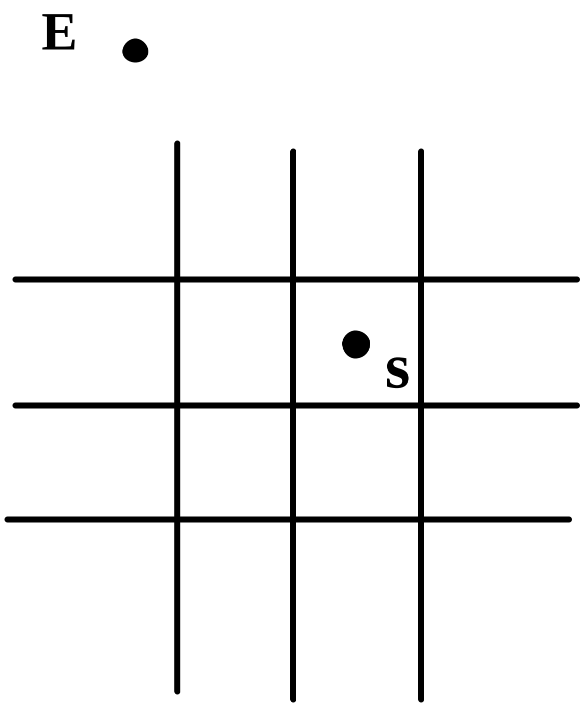
find grid charge $\bar{r}_g$

$$f_E^{(2)} = \sum_i r_{g,i} G(\vec{r}_i, \vec{r}_E) = (\bar{r}_g)^t \bar{f}_g$$

such that $f_E^{(1)} = f_E^{(2)}$

UNC Seminar

pFFT Algorithm: Projection Matrix



Expand the Green's function

$$G(\vec{r},\vec{r}_E) = \sum_k f_k(\vec{r})c_k$$

match both sides at grid point $\vec{r}_i$

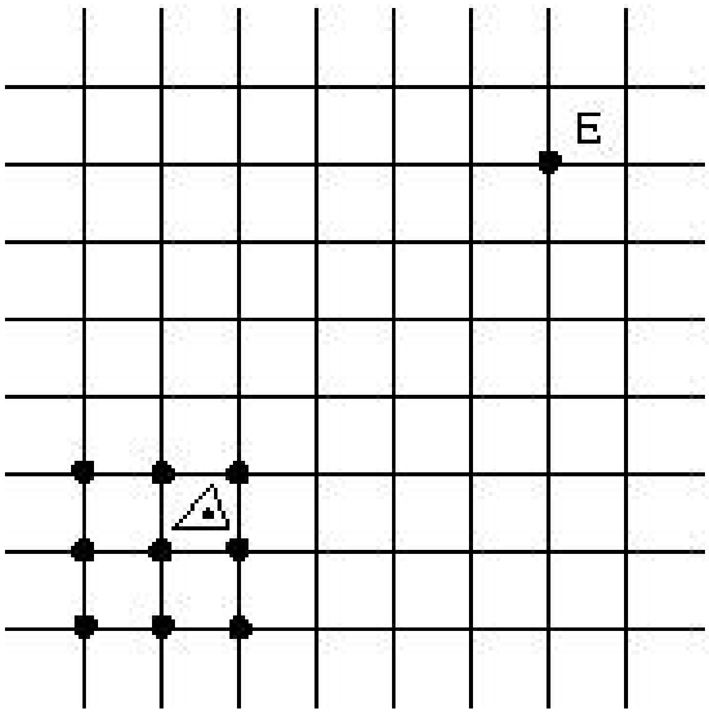
$$\bar{c} = F^{-1}\bar{f}_g$$

$$f_E^{(1)} = G(\vec{r}_s,\vec{r}_E) = \bar{f}^{\,t}(\vec{r}_s)F^{-1}\bar{f}_g$$

$$f_E^{(2)} = \sum_i r_{g,i}G(\vec{r}_i,\vec{r}_E) = (\bar{r}_g)^t\bar{f}_g$$

$$\boxed{(\bar{r}_g)^t = \bar{f}^{\,t}(\vec{r}_s)[F]^{-1}}$$

pFFT Algorithm: Projection Matrix



For unit point charge

$$(\bar{r}_g)^t = \bar{f}^t(\vec{r}_s)[F]^{-1}$$

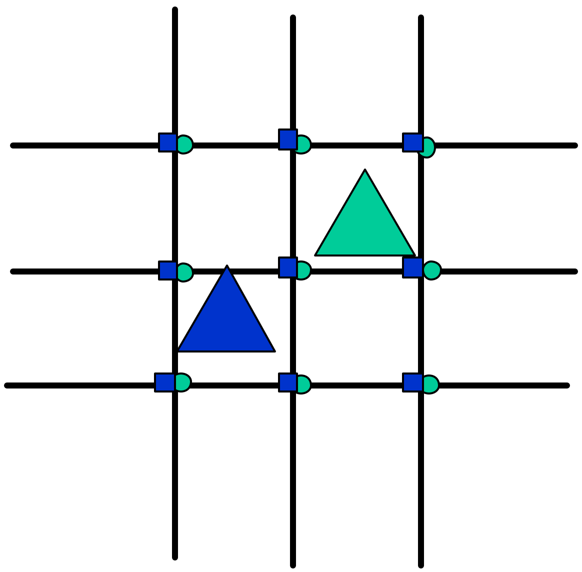
If the charge is a distribution $b_j(\vec{r})$

$$(\bar{r}_g^{(j)})^t = \int_{\Delta_j^b} dS\, b_j(\vec{r})\, \bar{f}^t(\vec{r})[F]^{-1}$$

pFFT Algorithm: Projection Matrix

For multiple panels:

$$r(\vec{r}) = \sum_j a_j b_j(\vec{r})$$



$$(\overline{r}_g^{(j)})^t = \int_{\Delta_j^b} dS\, b_j(\vec{r})\, \overline{f}^{\,t}(\vec{r})\, [F]^{-1}$$

$$\overline{Q}_g = \sum_{j=1}^{N_b} a_j (\overline{r}_g^{(j)})^t$$

pFFT Algorithm: Projection Matrix

$$\overline{Q}_g = \sum_{j=1}^{N_b} a_j \left(\overline{r}_g^{(j)}\right)^t$$

$$\overline{Q}_g = \begin{bmatrix} \vdots \\ Q_{g,1} \\ \vdots \\ Q_{g,9} \\ \vdots \end{bmatrix} = \begin{bmatrix} \ddots & 0 & \vdots & 0 & \vdots \\ \vdots & r_{g,1}^{(j)} & \vdots & r_{g,1}^{(k)} & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & r_{g,9}^{(j)} & \vdots & r_{g,9}^{(k)} & \vdots \\ \vdots & 0 & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ a_j \\ \vdots \\ a_k \\ \vdots \end{bmatrix} = [P]\overline{a}$$

Operations: $9N_b$ Memory: $9N_b$

pFFT Algorithm: Inner Differential Operator

If the kernel has a differential operator inside:

$$\int_S dS' \frac{d}{dn(\vec{r}')} G(\vec{r},\vec{r}')\, r(\vec{r}')$$

The operator works on the projection

$$\frac{d}{dn(\vec{r})} f(\vec{r}_s) = \frac{d}{dn(\vec{r})}\, \bar{f}^{\,t}(\vec{r}_s) F^{-1} \overline{f}_g$$

$$\overline{r}_g^{\,t} = \int_{\Delta_j^b} dS\, b_j(\vec{r}) \frac{d}{dn(\vec{r})}\, \bar{f}^{\,t}(\vec{r}) \left[F\right]^{-1}$$

pFFT Algorithm: Duality of [*I*] and [*P*]

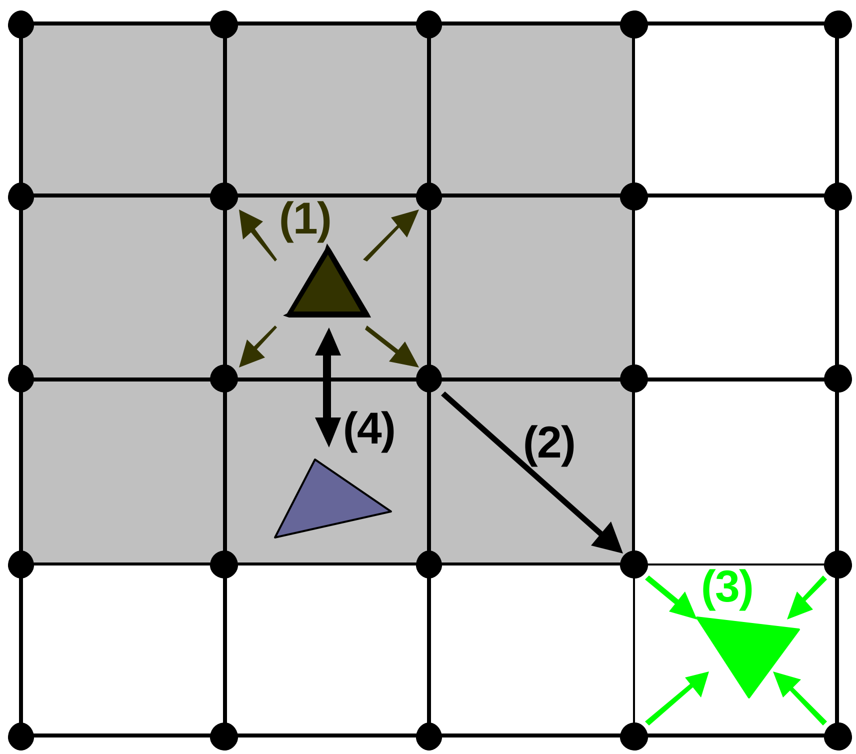
***i*th row of [*I*]:** $\int_{\Delta_i^t} dS t_i(\vec{r})\, \bar{f}^{\,t}(\vec{r}) [F]^{-1}$

***j*th column of [*P*]:** $\int_{\Delta_j^b} dS b_j(\vec{r})\, \bar{f}^{\,t}(\vec{r}) [F]^{-1}$

If $t_i(\vec{r}) = b_j(\vec{r})$, or $T_n = B_n$, then

$$P = I^t$$

pFFT Algorithm: Convolution Matrix



$$f_{g,j} = \sum_i G(\vec{r}_i', \vec{r}_j) Q_{g,i}$$

$$\overline{f}_g = [H]\overline{Q}_g$$

pFFT Algorithm: Convolution Matrix

Regular grid and position invariance

$$f_{g,j} = \sum_i \tilde{G}(\vec{r}_i' - \vec{r}_j) Q_{g,i}$$

Fast convolution by FFT in O(*N*log(*N*)) time

$$H_{i,j} = \tilde{G}(\vec{r}_i' - \vec{r}_j)$$

Only $H_{1,j}$ $(j = 1, 2, \ldots, N_g)$ are unique

Operations: O($N_g$log(N_g))   Memory: O(N_g)

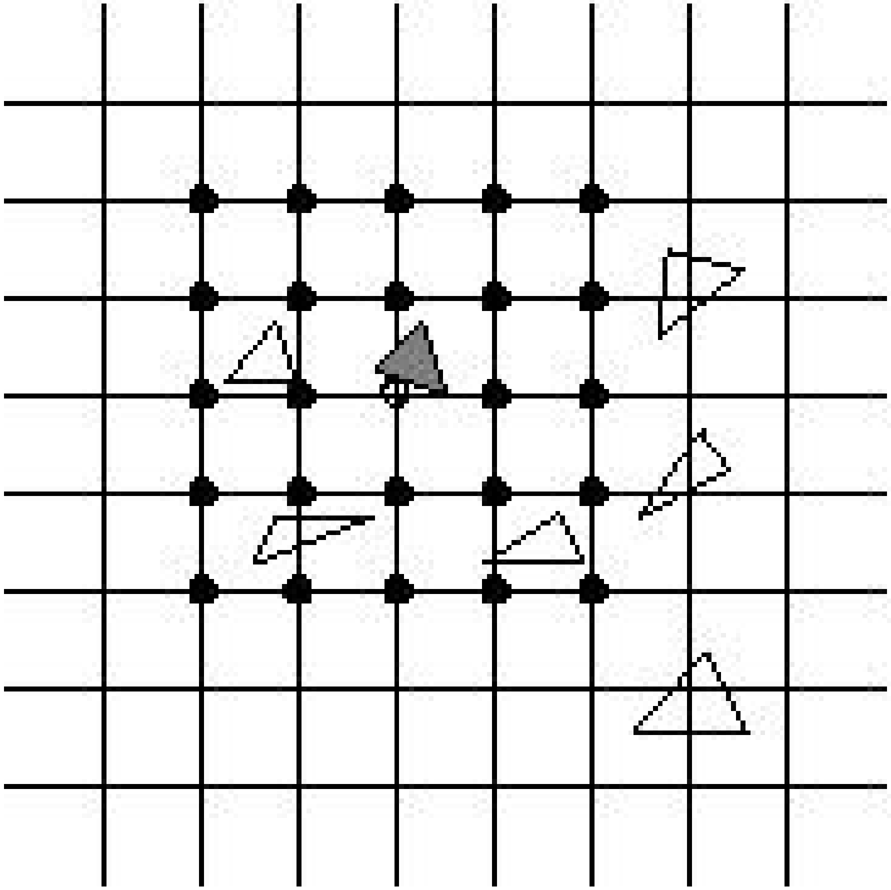
UNC Seminar

pFFT Algorithm: Direct Matrix

Summary of the first three steps:

$$\begin{cases} \overline{Q}_g = [P]\overline{a} \\ \overline{f}_g = [H]\overline{Q}_g \\ \overline{\Psi} = [I]\overline{f}_g \end{cases} \Rightarrow \overline{\Psi} = [I][H][P]\overline{a}$$

pFFT Algorithm: Direct Matrix



$$D_{i,j} = A_{i,j} - (\overline{W}^{(i)})^t \left[H^{(i,j)}\right] \overline{r}_g^{(j)}$$

$$j \in \mathrm{N}^{(i)}$$

Operations: O(N_b) Memory: O(N_b)

pFFT Algorithm: Four sparse matrices

$$\overline{\Psi} = [A]\overline{a} \approx ([D] + [I][H][P])\overline{a}$$

- **Projection:**

  Operations: O(N_b)   Memory: O(N_b)

- **Convolution:**

  Operations: O($N_g$log(N_g)) Memory: O(N_g)

- **Interpolation:**

  Operations: O(N_b)   Memory: O(N_b)

- **Nearby interaction:**

  Operations: O(N_b)   Memory: O(N_b)

Outline

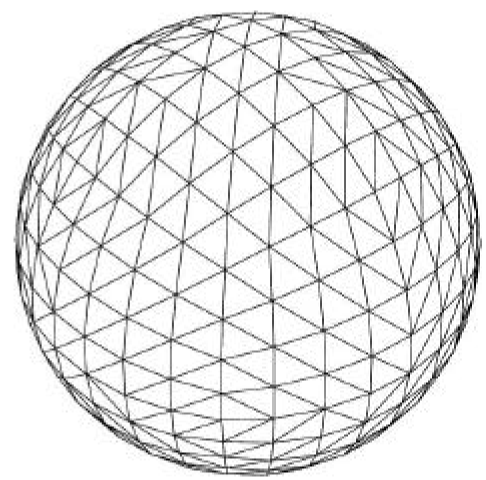
- **Background**
- **Pre-corrected FFT Algorithm**
- **<u>Unit testing results</u>**
- **Surface Integral Formulation**
- **Numerical Results**

Unit testing

On the surface of a sphere with radius *R*

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

Let *x* be a random vector

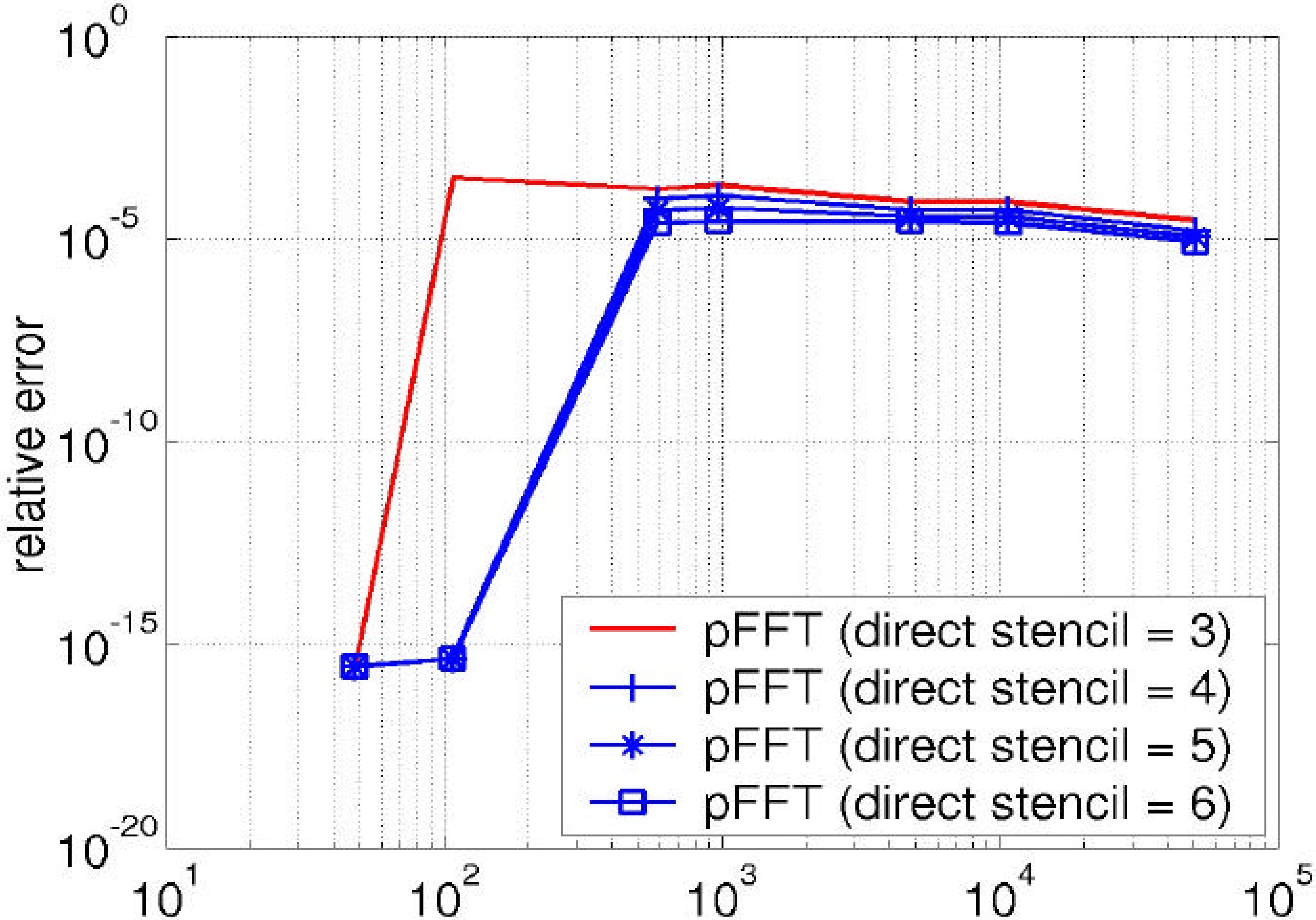


$$y_1 = Ax$$

$$y_2 = pfft(x)$$

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

Single-Layer Kernel Accuracy (4-5 digits)

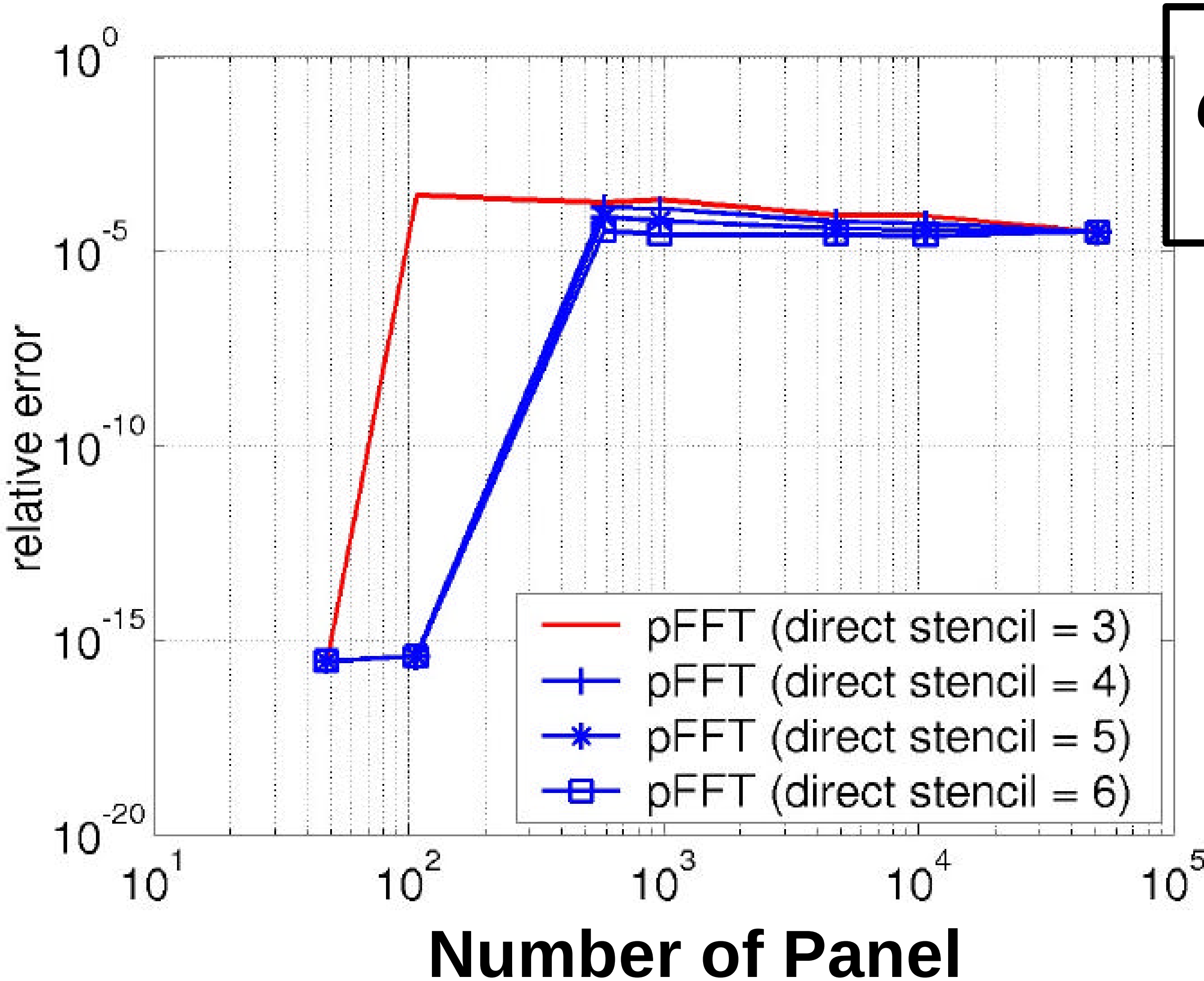


Number of Panel

$$\frac{1}{r}$$

Single-Layer Kernel Accuracy (4-5 digits), R/l = 1e-6

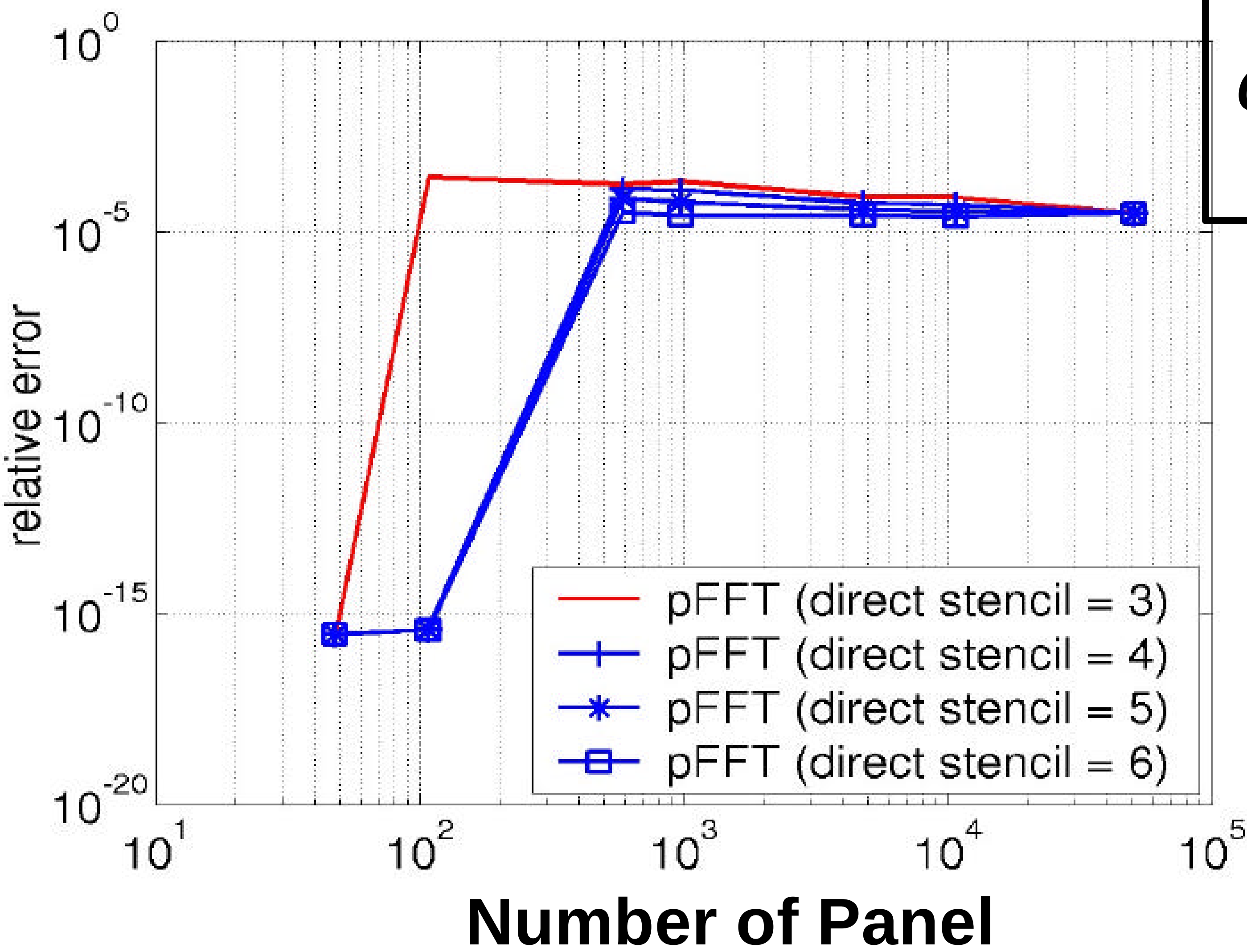
$$e^{ikr}/r$$



UNC Seminar

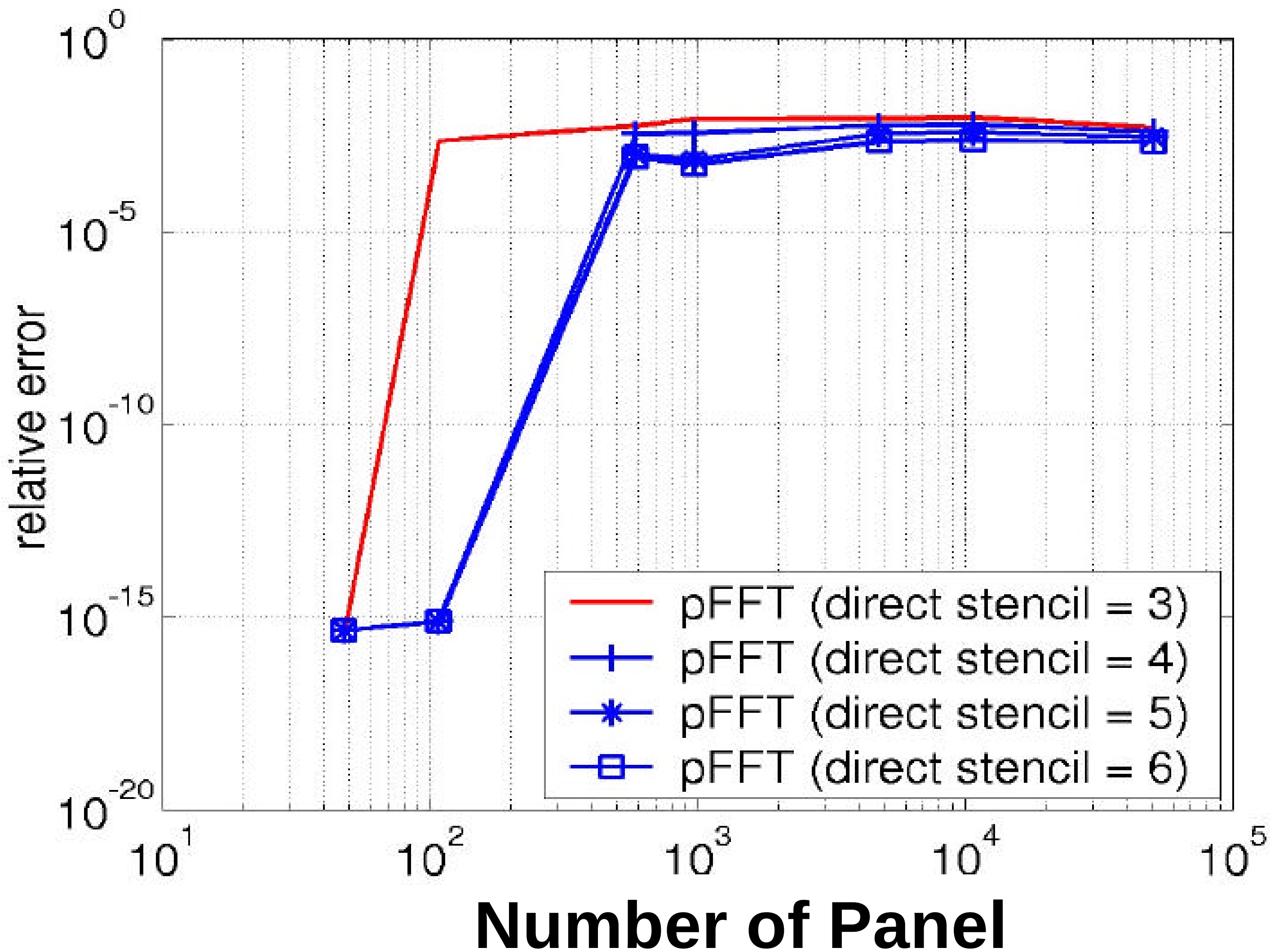
Single-Layer Kernel Accuracy (4-5 digits), *R*/*l* = 1e2

$$e^{ikr} / r$$



UNC Seminar

Double-Layer Kernel Accuracy (2-3 digits)

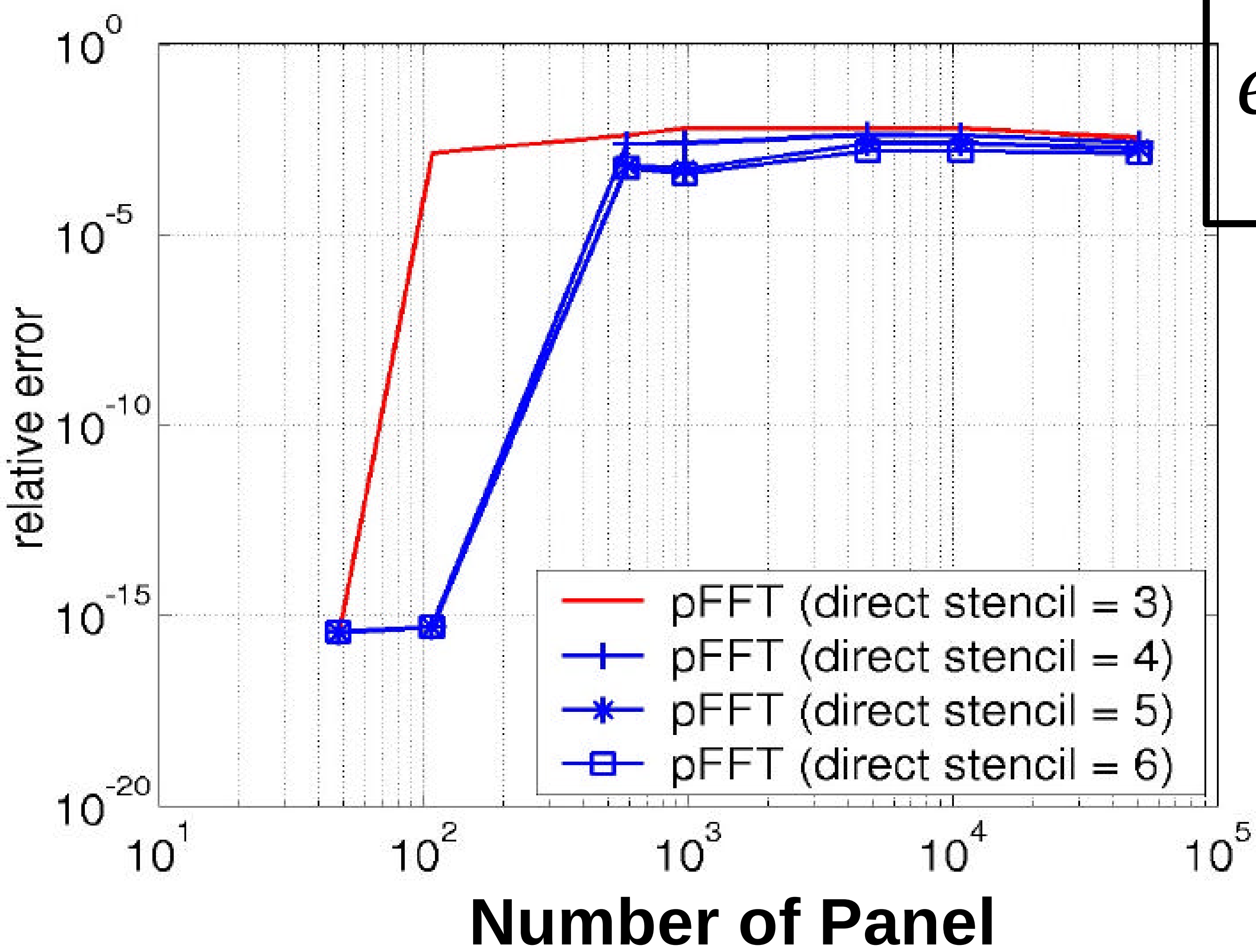


$$\frac{1}{r}$$

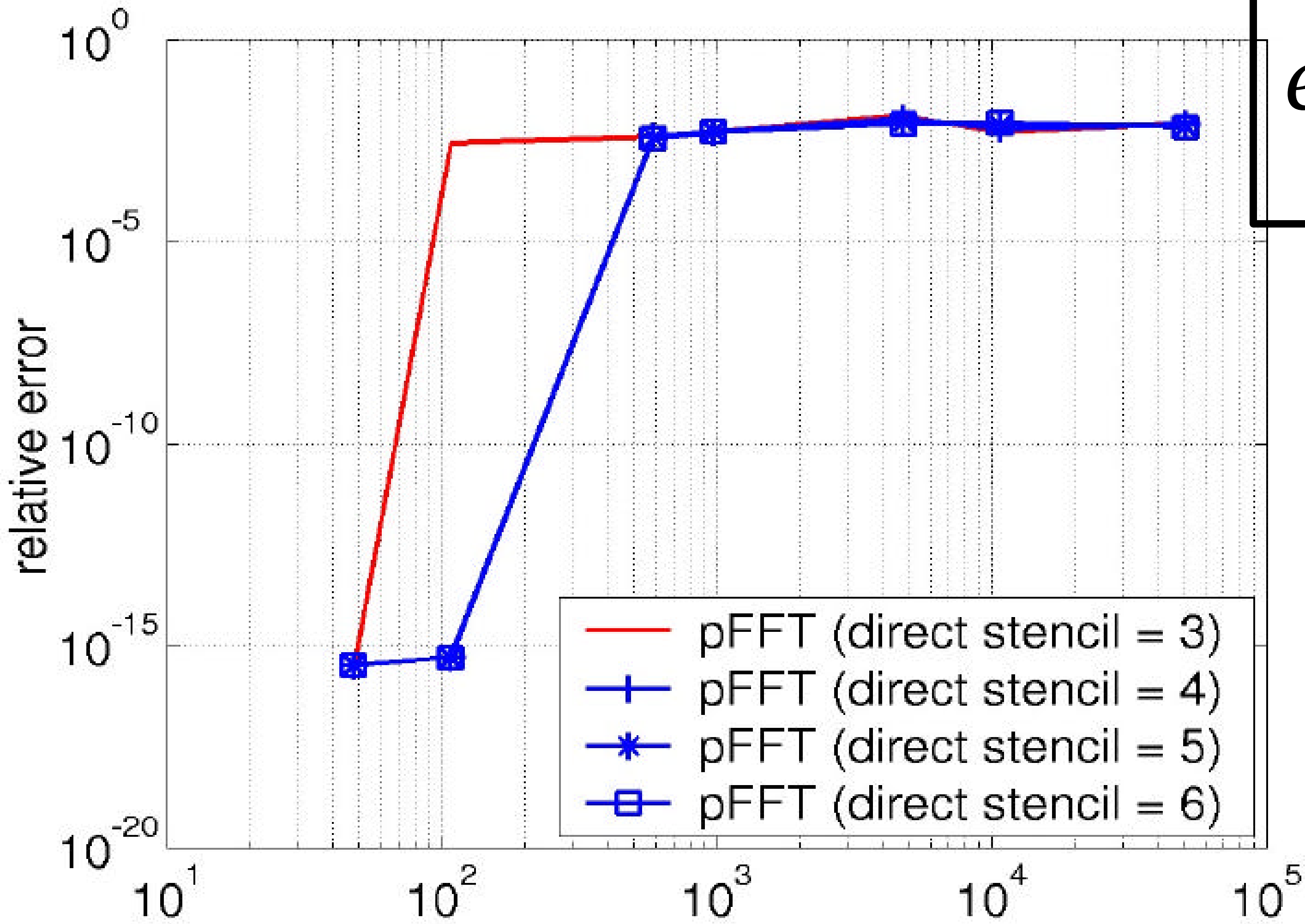
UNC Seminar

Double-Layer Kernel Accuracy (2-3 digits), *R*/l = 1e-6

$$e^{ikr}/r$$



Double-Layer Kernel Accuracy (2-3 digits), R/l = 1e2

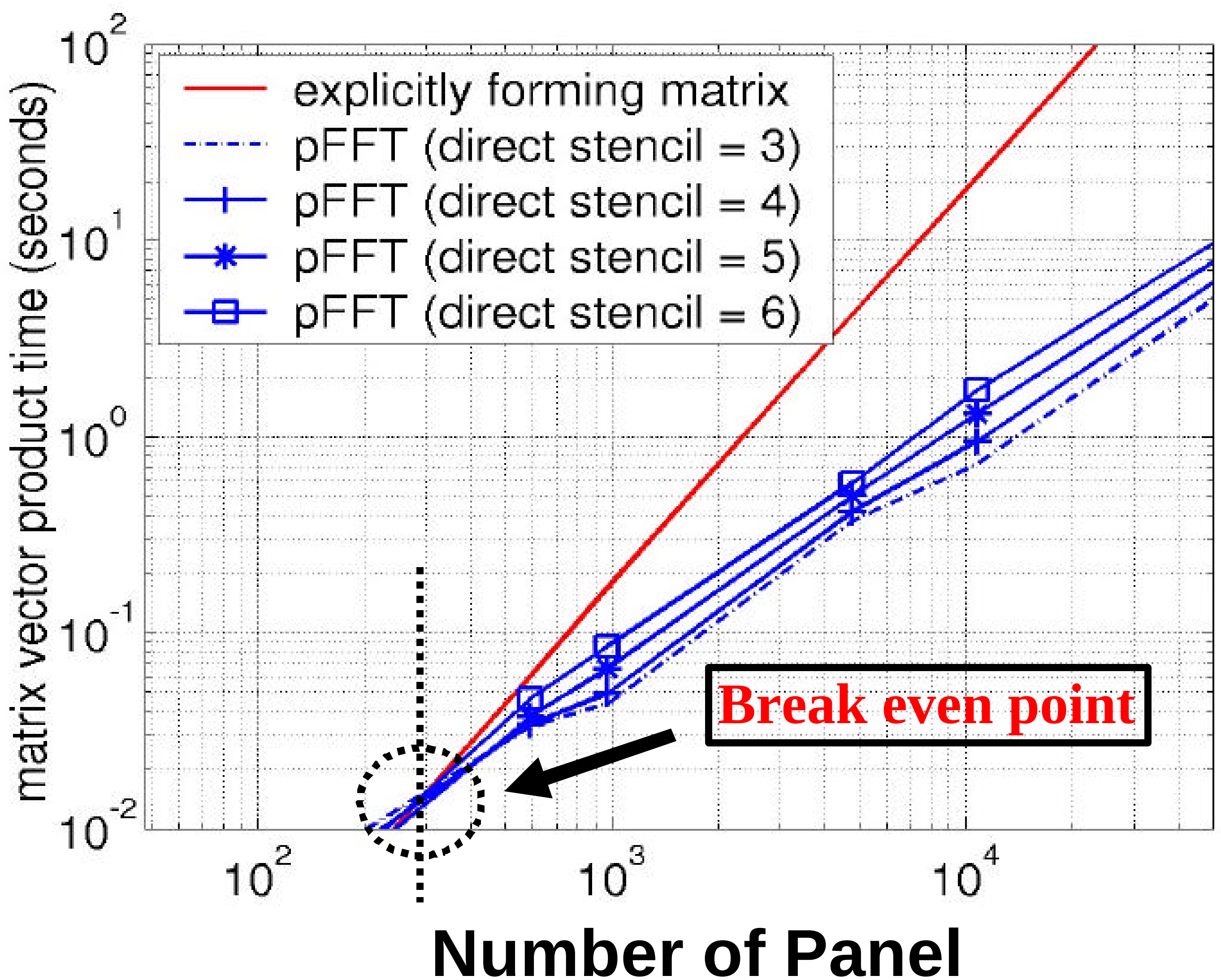


$$e^{ikr}/r$$

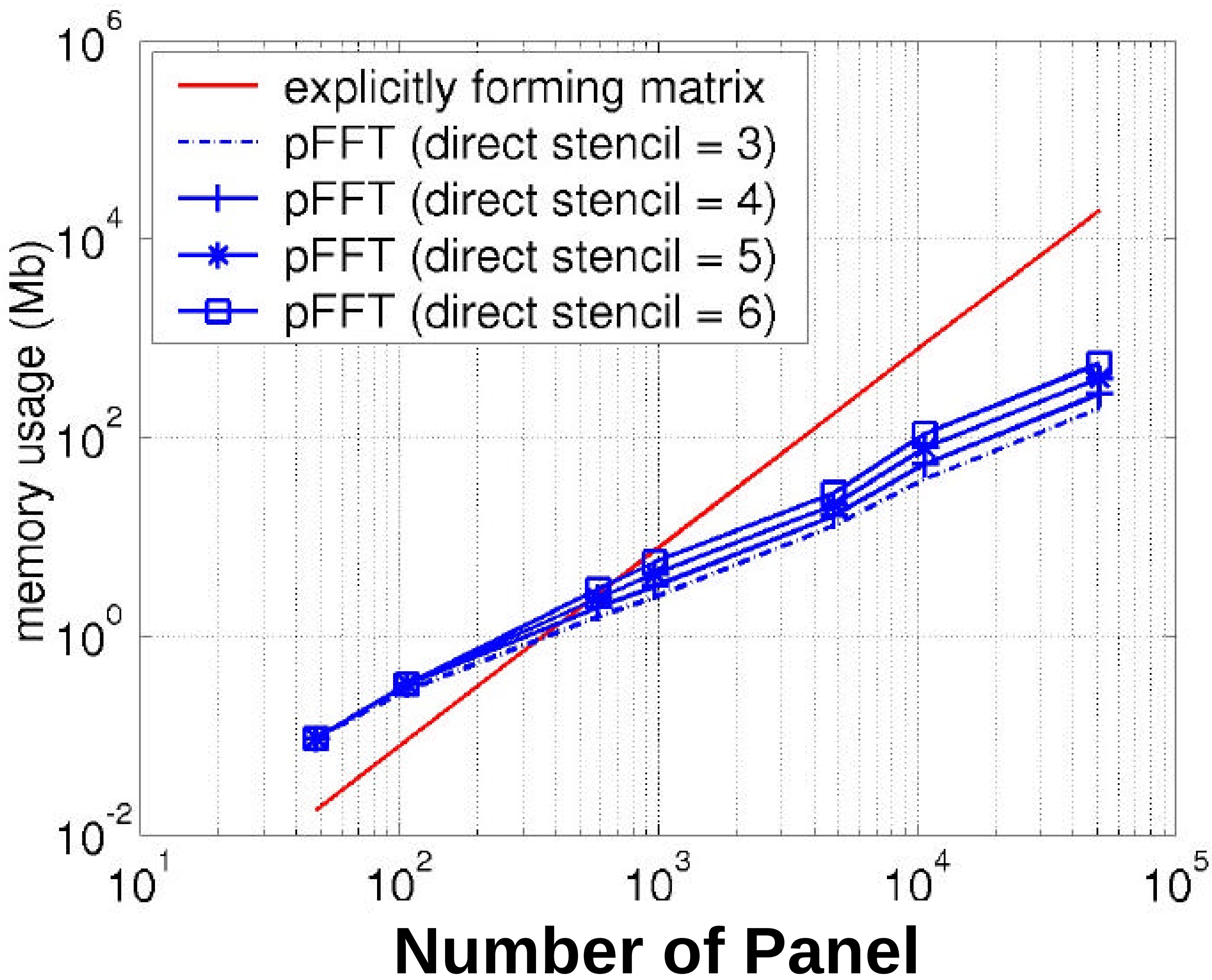
Number of Panel

UNC Seminar

Matrix vector product time O(*N*)



Memory O(*N*)


explicitly forming matrix
pFFT (direct stencil = 3)
pFFT (direct stencil = 4)
pFFT (direct stencil = 5)
pFFT (direct stencil = 6)
memory usage (Mb)
10^{-2}
10^{0}
10^{2}
10^{4}
10^{6}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
Number of Panel

Outline

- Background
- Pre-corrected FFT Algorithm
- Unit testing results
- Surface Integral Formulation
- Numerical Results

Summary of the Surface Integral Formulation

$$\frac{1}{2}\vec{E}(\vec{r}) = \int_{S_i} dS'\left(G_1(\vec{r},\vec{r}')\frac{\partial \vec{E}(\vec{r}')}{\partial n(\vec{r}')} - \frac{\partial G_1(\vec{r},\vec{r}')}{\partial n(\vec{r}')}E(\vec{r}')\right) \quad \vec{r} \in S_i$$

$$-\frac{1}{2}\vec{E}(\vec{r}) = \int_{S} dS'\left(G_0(\vec{r},\vec{r}')\frac{\partial \vec{E}(\vec{r}')}{\partial n(\vec{r}')} - \frac{\partial G_0(\vec{r},\vec{r}')}{\partial n(\vec{r}')}E(\vec{r}')\right) + \nabla f(\vec{r}) \quad \vec{r} \in S$$

$$\int_{S} d\vec{r}' G_0(\vec{r}',\vec{r}) r_s(\vec{r}') = e f(\vec{r}) \quad \vec{r} \in S$$

$$\nabla \bullet \vec{E}(\vec{r}) = 0$$

← Current Conservation

$$f(\vec{r}) = c, \qquad \vec{r} \in \text{contact}$$

System Matrix Structure

$$
\begin{bmatrix}
P_1 & & & D_1 & & & & \\
 & P_1 & & & D_1 & & & \\
 & & P_1 & & & D_1 & & \\
t_{1x}P_0 & t_{1y}P_0 & t_{1z}P_0 & t_{1x}D_0 & t_{1y}D_0 & t_{1x}D_0 & \hat{t}_1 \bullet \nabla & \\
t_{2x}P_0 & t_{2y}P_0 & t_{2z}P_0 & t_{2x}D_0 & t_{2y}D_0 & t_{2z}D_0 & \hat{t}_2 \bullet \nabla & \\
n_x & n_y & n_z & & & & & {}^{jw}\!/\!{}_{s} \\
a & a & a & c_x & c_y & c_z & & \\
 & & & & & & A & P_0
\end{bmatrix}
\begin{bmatrix}
dE_x/dn \\ dE_y/dn \\ dE_z/dn \\ E_x \\ E_y \\ E_z \\ f \\ r
\end{bmatrix}
$$

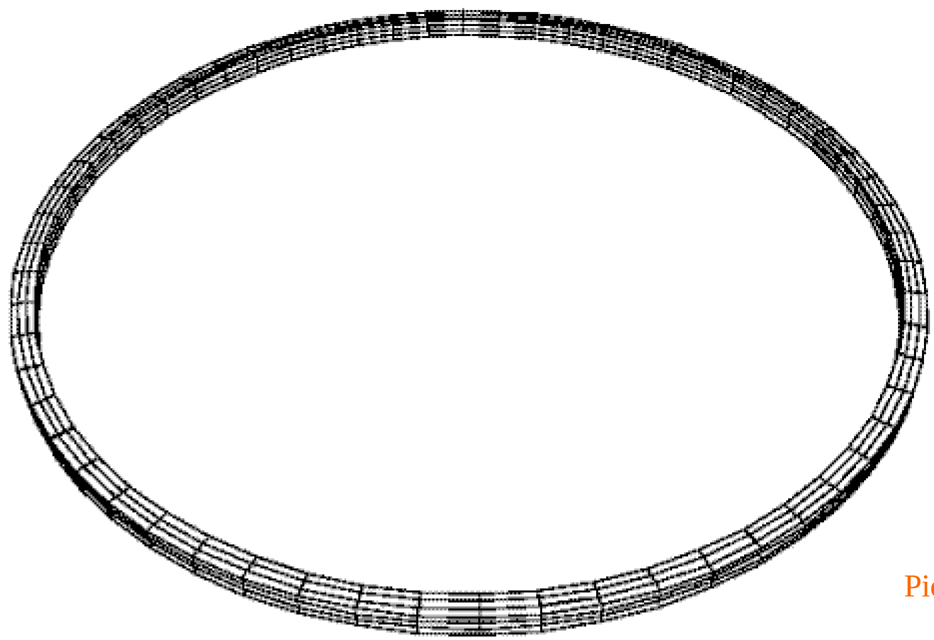
Pre-conditioner

$$
\begin{bmatrix}
P_1^d & & & -2p\,I & & & & \\
& P_1^d & & & -2p\,I & & & \\
& & P_1^d & & & -2p\,I & & \\
t_{1x}P_0^d & t_{1y}P_0^d & t_{1z}P_0^d & t_{1x}2p\,I & t_{1y}2p\,I & t_{1x}2p\,I & \hat{t}_1\bullet\nabla & \\
t_{2x}P_0^d & t_{2y}P_0^d & t_{2z}P_0^d & t_{2x}2p\,I & t_{2y}2p\,I & t_{2z}2p\,I & \hat{t}_2\bullet\nabla & \\
n_x & n_y & n_z & & & & & jw/s \\
a & a & a & c_x & c_y & c_z & & \\
& & & & & & A & P_0^d
\end{bmatrix}
\begin{bmatrix}
dE_x/dn \\ dE_y/dn \\ dE_z/dn \\ E_x \\ E_y \\ E_z \\ f \\ r
\end{bmatrix}
$$

Outline

- **Background**
- **Pre-corrected FFT Algorithm**
- **Unit testing results**
- **Surface Integral Formulation**
- **<u>Numerical Results</u>**

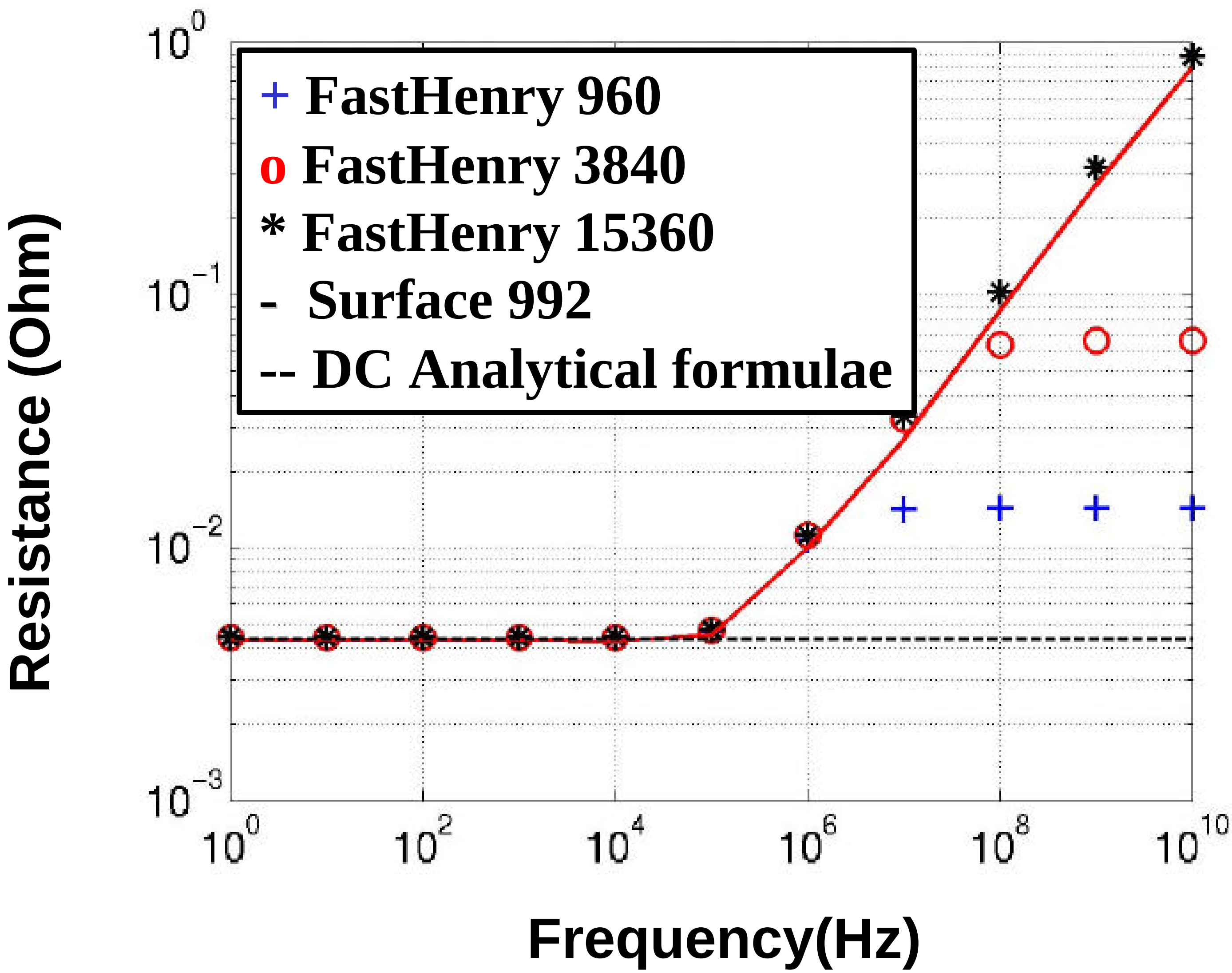
A Ring Example



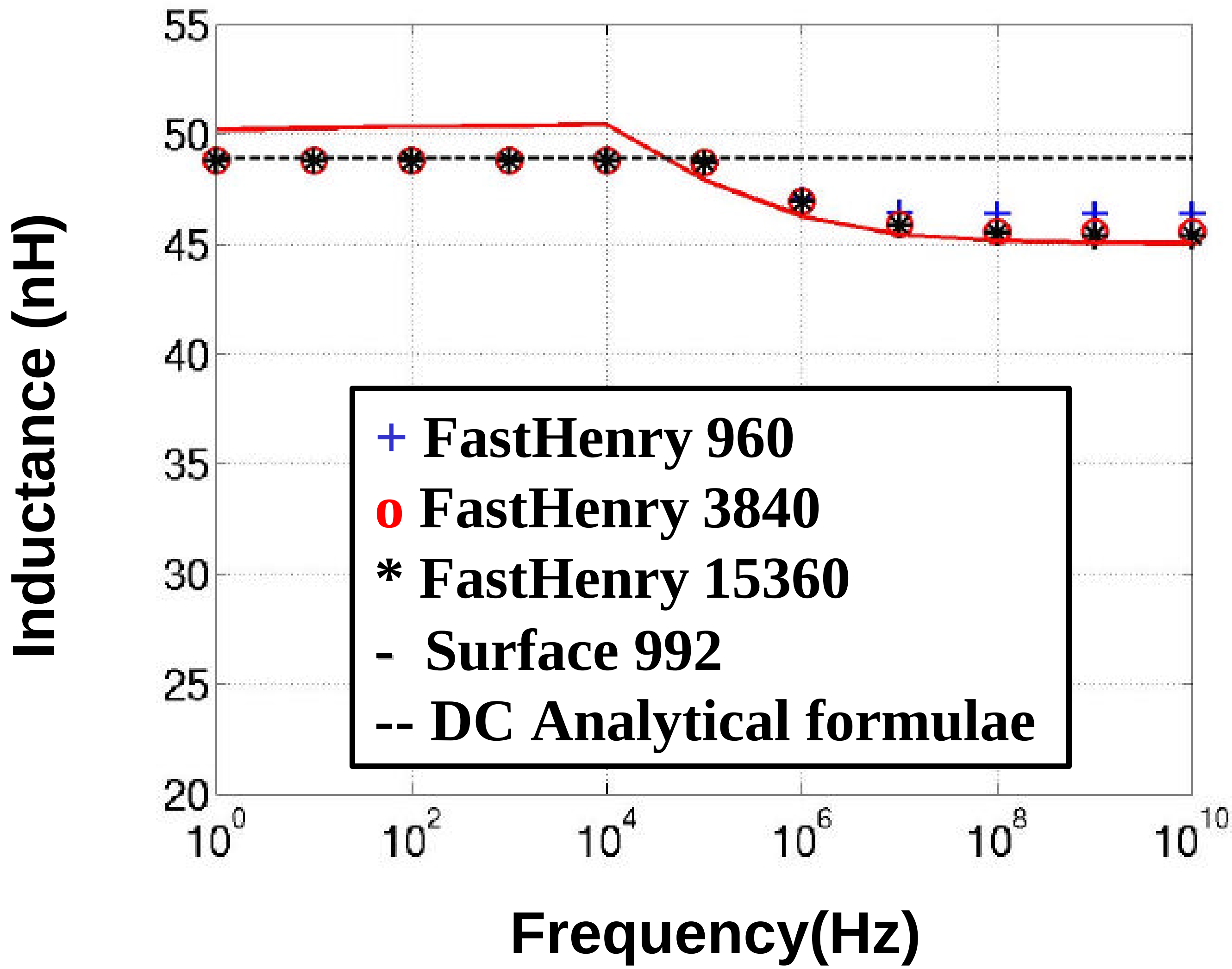
Picture thanks to Junfeng Wang

Cross-section: 0.5x0.5 mm^2
Radius: 10 mm

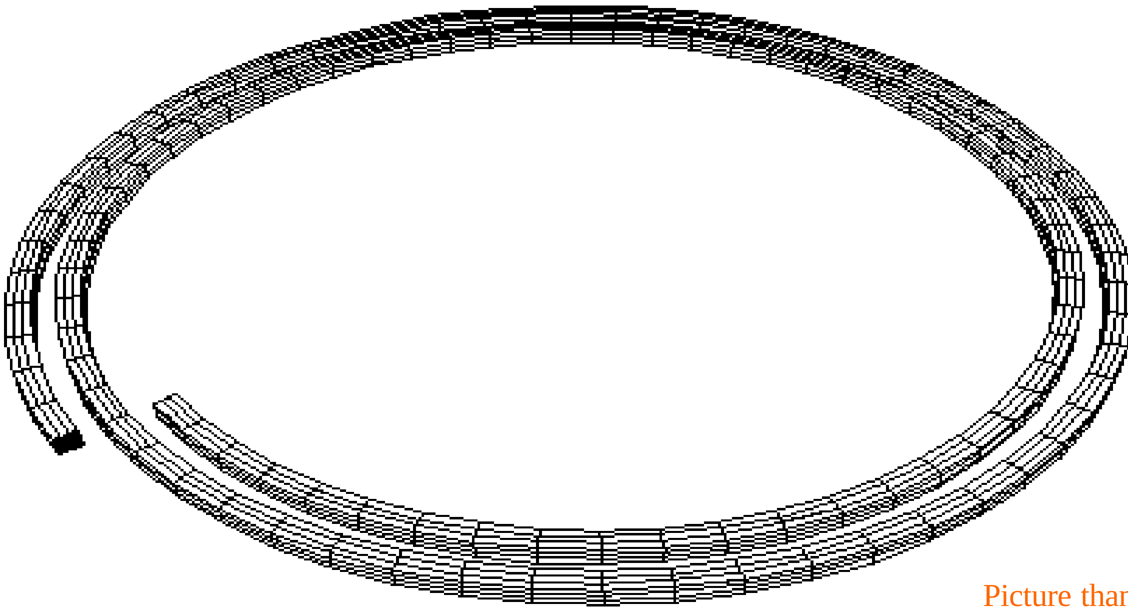
A Ring Example



A Ring Example



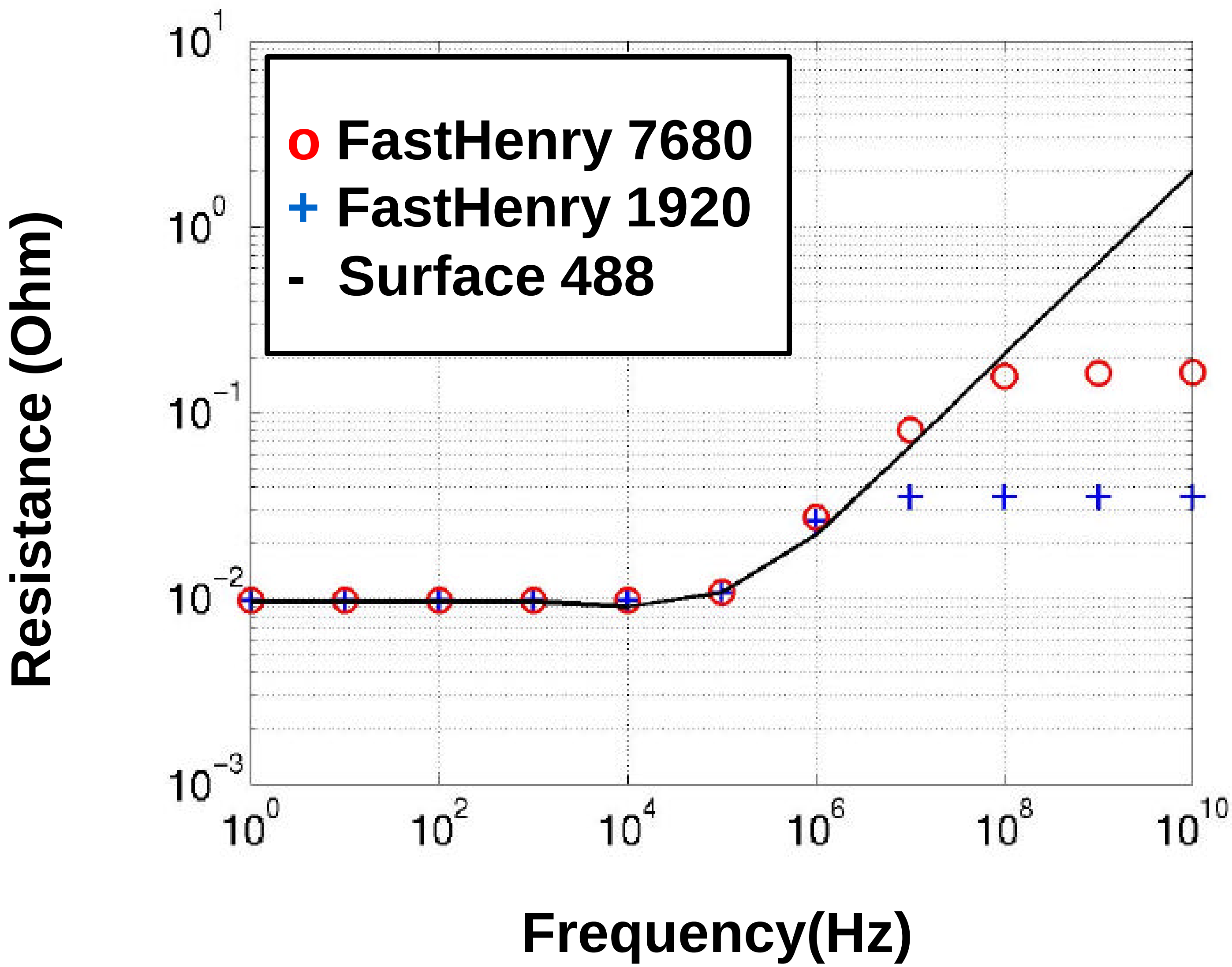
A Spiral Example



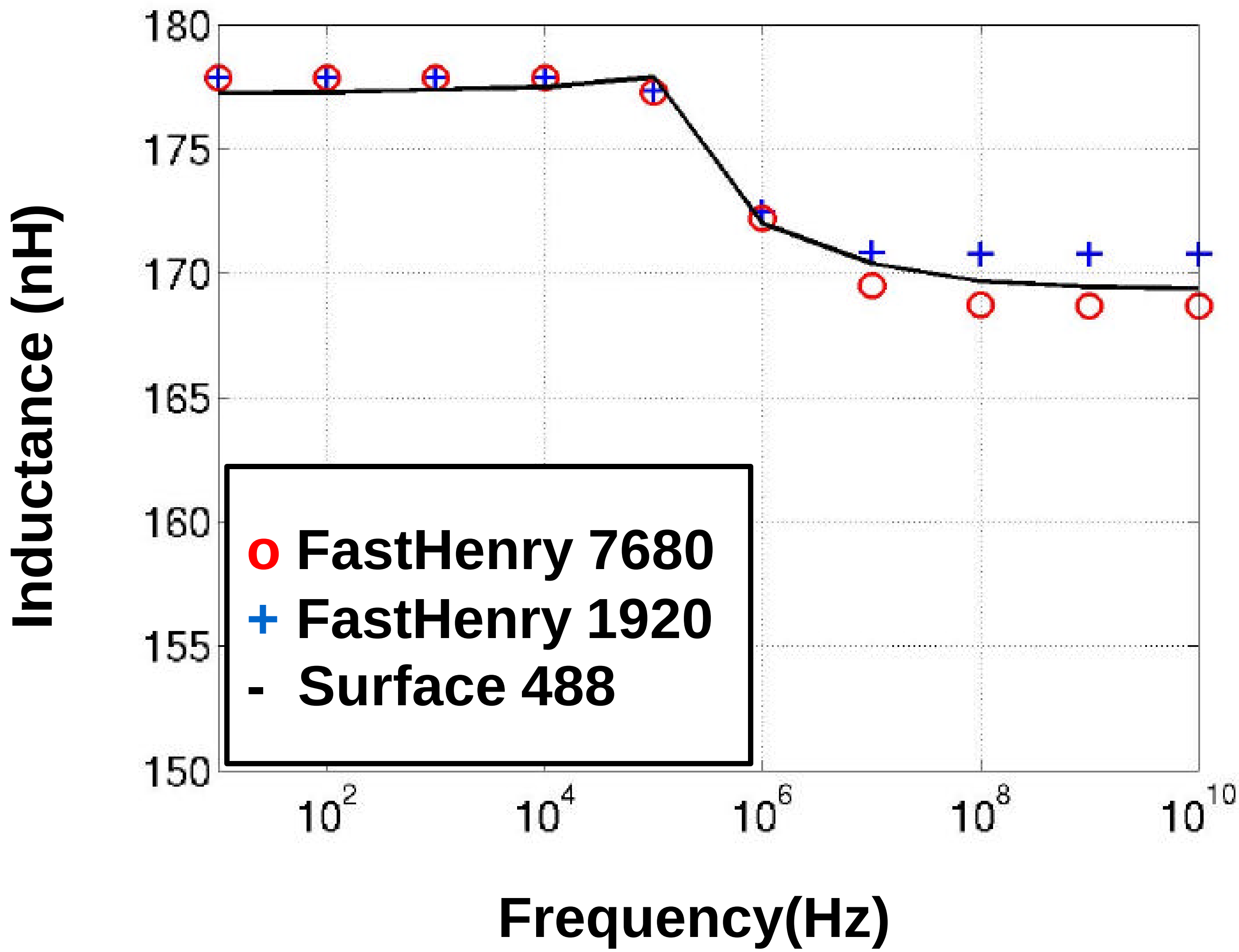
Picture thanks to Junfeng Wang

Cross-section: 0.5x0.5 mm^2
Inner Radius: 10 mm
Spacing: 0.5 mm
Number of turns: 2

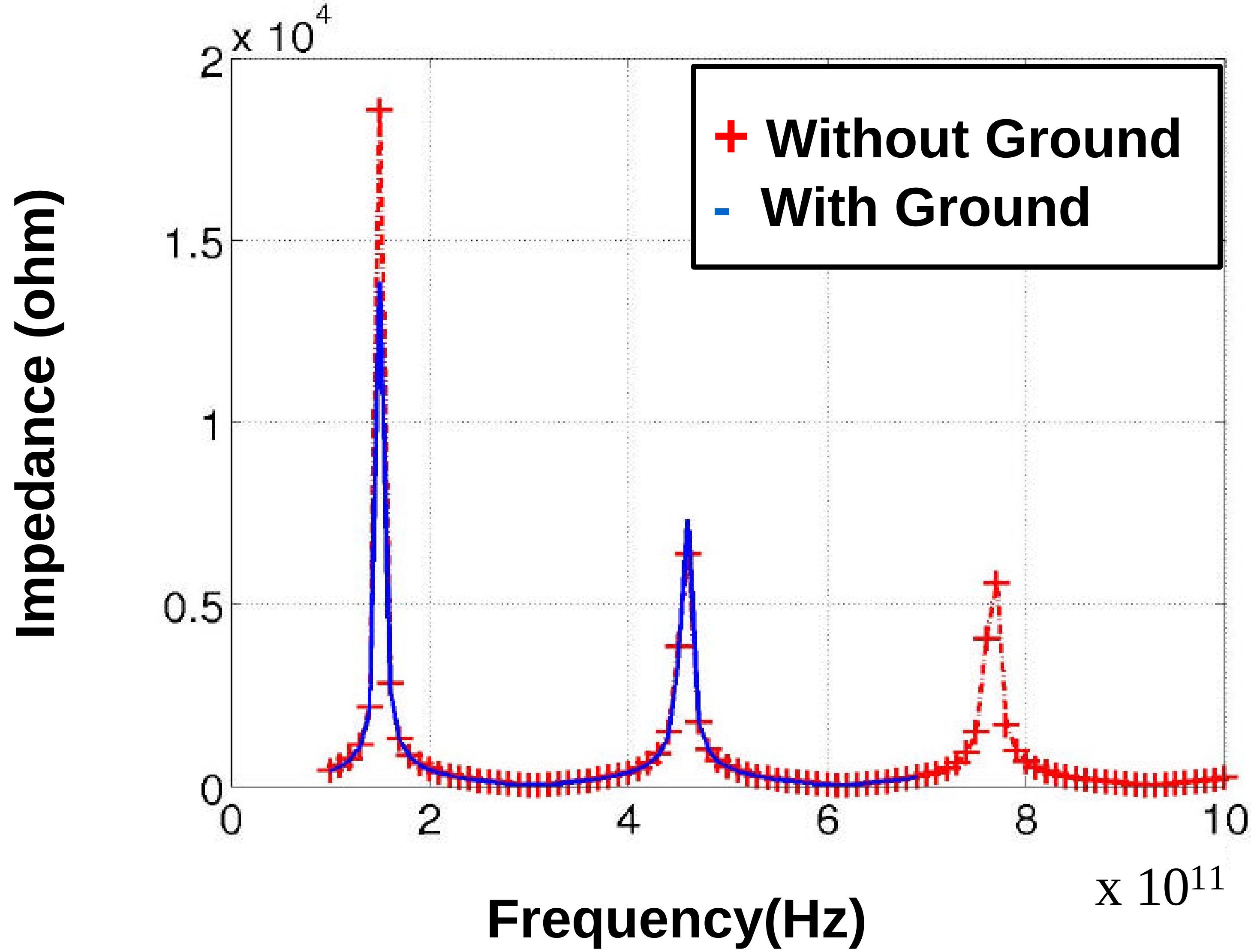
A Spiral Example



A Spiral Example

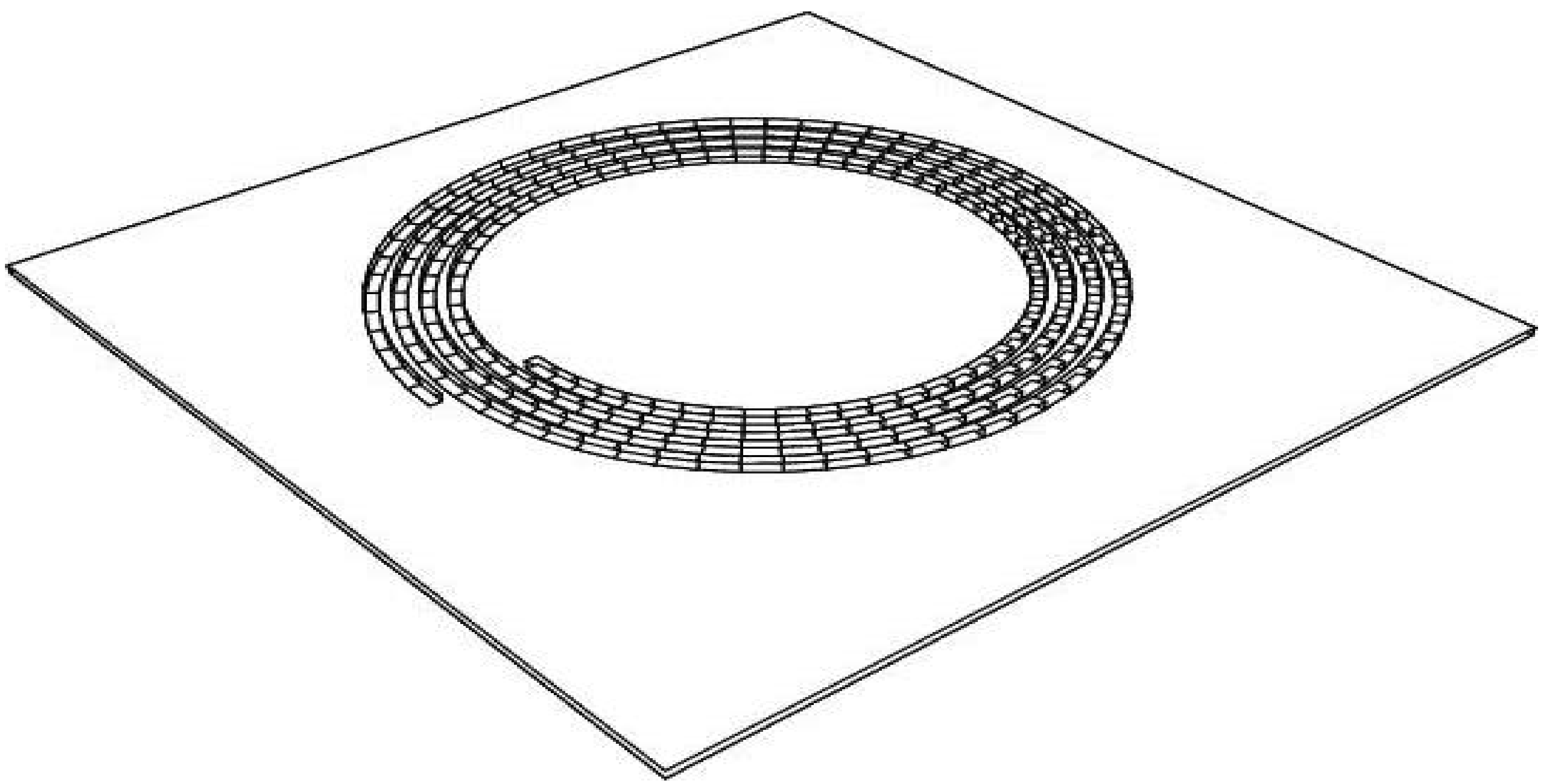


A Shorted Transmission Line



A Spiral Over Ground

4-turn spiral over substrate 15162 panels
MQS: 106k unknowns, 69 minutes, 348 Mb
EMQS: 121k unknowns, 93minutes, 379 Mb

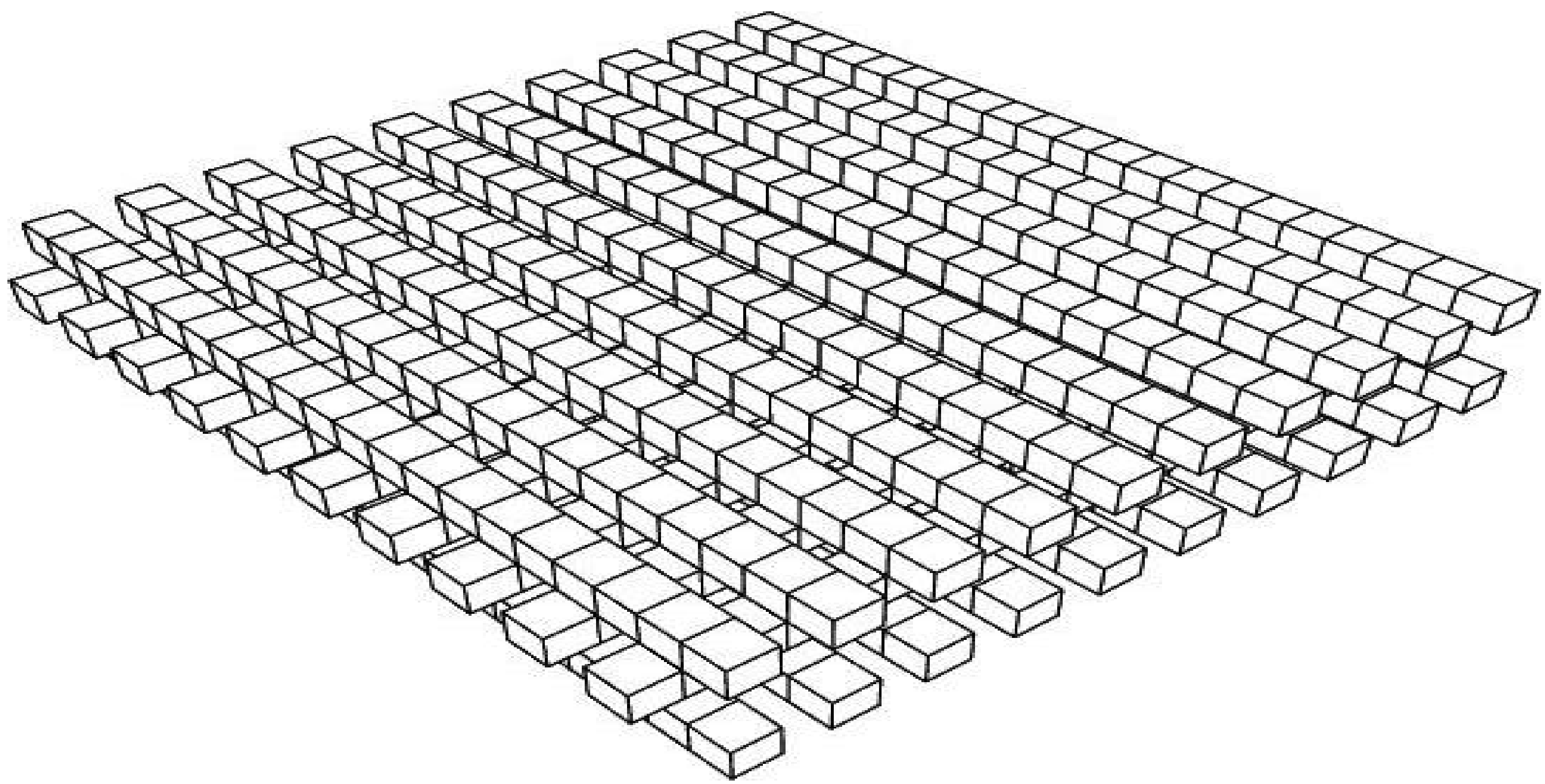


Multi-conductor bus

3 layer, 10 conductors each layer   12540 panels

MQS: 87.5k unknowns, 41 minutes, 165 Mb

EMQS: 100k unknowns, 61 minutes, 218 Mb



Conclusions

- Pre-corrected FFT algorithm
- Performance of the algorithm
- A surface integral formulation
- Numerical results