Numerical Methods for PDEs

Boundary Element Methods, Lecture 6
Fast Algorithms for Integral Equations

L. Proctor, S. De, K. Nabors, J. Phillips, B. Buchmann, J.White

December 12, 2016

Outline

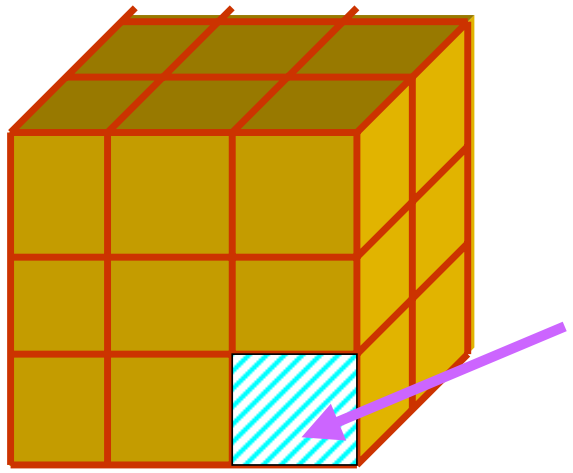
Background

Discretize Surface Into Panels

Piecewise Constant Basis

Integral Equation : $\Psi(x) = \int_{surface} \frac{\sigma(x')}{\|x-x'\|} dS'$

Discretize Surface into Panels



$$\text{Represent } \sigma(x) \approx \sum_{i=1}^{n} \alpha_i \underbrace{\varphi_i(x)}_{\text{Basis Functions}}$$

$$\varphi_j(x) = 1 \quad \text{if } x \text{ is on panel j}$$
$$\varphi_j(x) = 0 \quad \text{otherwise}$$

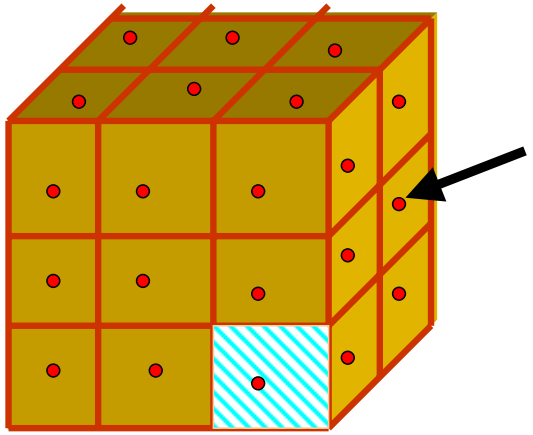
Background

Discretize Surface Into Panels

Centroid Collocation

Put collocation points at panel centroids

$$\Psi\left(x_{c_i}\right) = \sum_{j=1}^{n} \alpha_j \underbrace{\int_{panel\ j} \frac{1}{\left\| x_{c_i} - x' \right\|} dS'}_{A_{i,j}}$$



x_{c_i} Collocation point

$$\begin{bmatrix} A_{1,1} & \cdots & \cdots & A_{1,n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ A_{n,1} & \cdots & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \Psi\left(x_{c_1}\right) \\ \vdots \\ \vdots \\ \Psi\left(x_{c_n}\right) \end{bmatrix}$$

Background

Dense Matrix

Resultant Dense Matrix

Matrix Entries Are Never Zero

$$A_{i,j} = \int_{panel_j} \frac{1}{\|x_{c_i} - x'\|} dS'$$

Distant Elements Decay Slowly

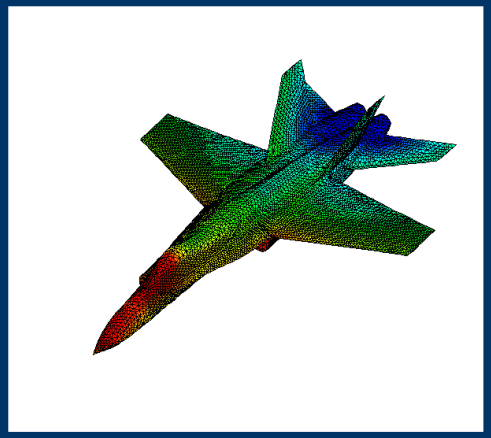
$$\propto \frac{1}{\|x_{c_i} - x_{panel_j\ center}\|}$$

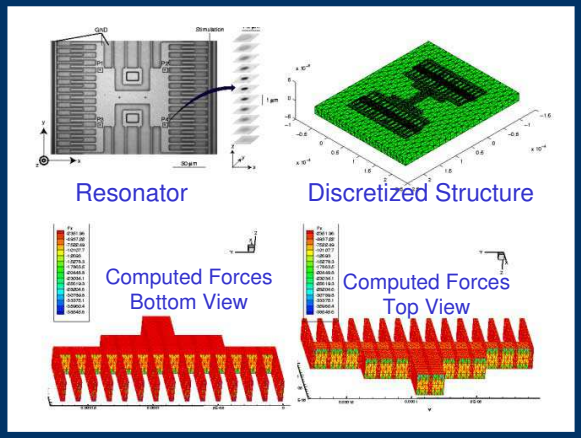
Too Slow To Ignore.

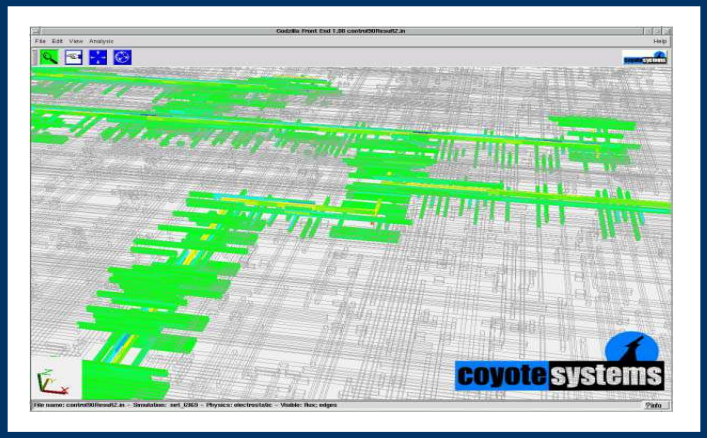
Background

Dense Matrix

Complicated Examples







Need More than 100,000 unknowns!!

Need 100 Gigabytes to Store Matrix.

Background

Dense Matrix

Gaussian Elimination

```
For i = 1 to n-1 {          "For each Row"
   For j = i+1 to n  {      "For each Row below pivot"
      For k = i+1 to n  { "For each element beyond Pivot"
```

$$A_{jk} \Leftarrow A_{jk} - \frac{A_{ji}}{A_{ii}} A_{ik}$$

Multiplier

Pivot

```
      }
   }
}
```

Form n-1 reciprocals (pivots)

Form $\sum_{i=1}^{n-1}(n-i) = \frac{n^2}{2}$ multipliers

Perform $\sum_{i=1}^{n-1}(n-i)^2 \approx \frac{2}{3}n^3$ Multiply-adds

n^3 – Too Expensive!

Electrostatics Application

General Iterative "Algorithm"

0 : Guess at panel charges $\vec{\alpha}$

1 : Compute the centroid potentials from the charges

$\mathbf{A}\vec{\alpha}$

2 : Compare the computed to known potentials

$\mathbf{R} = \mathbf{\Psi} - \mathbf{A}\vec{\alpha}$

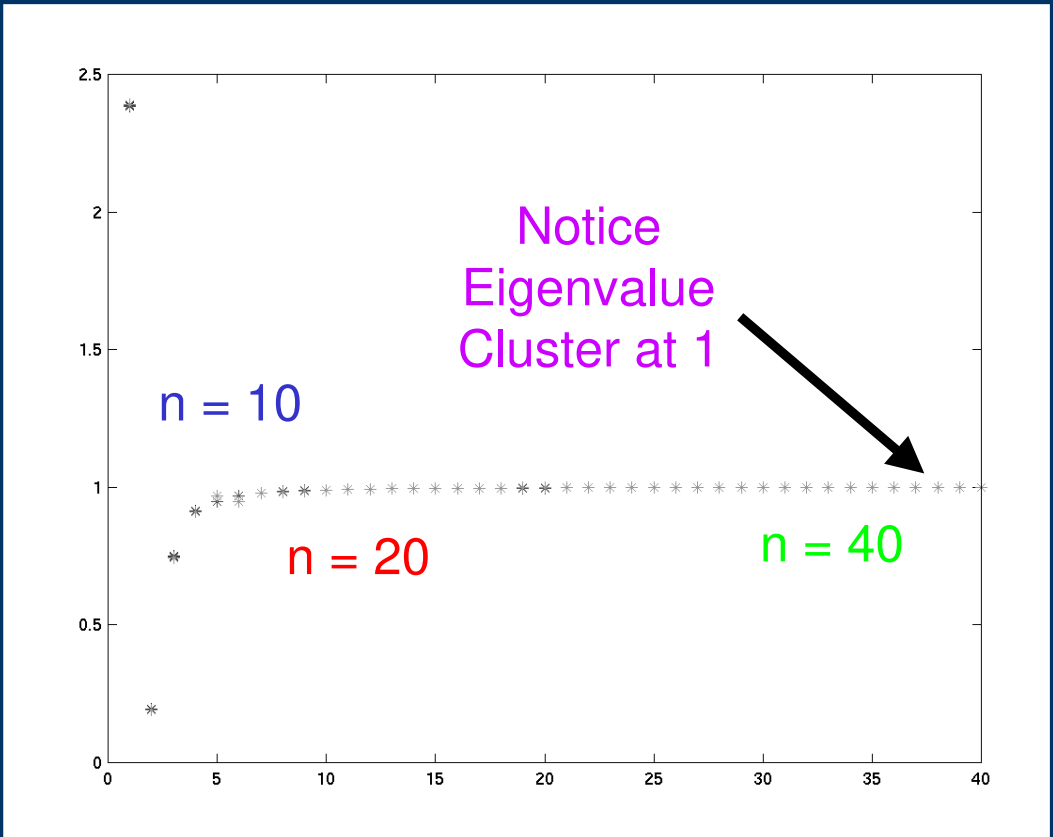
3 : Fix the panel charges, go to Step 1.

$\vec{\alpha}$

Conjugate Gradient (CG)

CG for 2nd Kind

Eigenvalues for 2^{nd} Kind Integral Equation



Conjugate-Gradient convergence rate

$$\|r^k\| \leq 2\left(\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}-1 \Big/ \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}+1\right)^k \|r^0\|$$

For discretized Second Kind equations

$$\frac{\lambda_{max}}{\lambda_{min}} \; is\ bounded\ independent\ of\ n$$

Number of CG iterations independent of n!!

Conjugate Gradient (CG)

Steps of CG

The k^{th} step of the Conjugate Gradient Algorithm

compute Ap_k

For discretized Integral equations, A is dense

$$\alpha_k = \frac{\left(r^k\right)^T \left(Ap_k\right)}{\left(Ap_k\right)^T \left(Ap_k\right)}$$

Determine optimal step size in kth search direction

$$x^{k+1} = x^k + \alpha_k p_k$$

$$r^{k+1} = r^k - \alpha_k Ap_k$$

Update the solution and the residual

$$p_{k+1} = r^{k+1} - \frac{\left(Ar^{k+1}\right)^T \left(Ap_k\right)}{\left(Ap_k\right)^T \left(Ap_k\right)} p_k$$

Compute the new orthogonalized search direction

Conjugate Gradient (CG)

Cost of CG

Complexity of the Conjugate Gradient Method

compute Ap_k

Dense Matrix-vector product costs O(n²)

$$\alpha_k = \frac{\left(r^k\right)^T \left(Ap_k\right)}{\left(Ap_k\right)^T \left(Ap_k\right)}$$

Vector inner products, O(n)

$$x^{k+1} = x^k + \alpha_k p_k$$
$$r^{k+1} = r^k - \alpha_k Ap_k$$

Vector Adds, O(n)

$$p_{k+1} = r^{k+1} - \frac{\left(Ar^{k+1}\right)^T \left(Ap_k\right)}{\left(Ap_k\right)^T \left(Ap_k\right)} p_k$$

Inner products, total cost O(n)

Algorithm is $O(n^2)$ for integral equations even though # of iterations, k, is small!

Accelerate the Conjugate Gradient Method

Exactly compute $\mathbf{A}p_k$

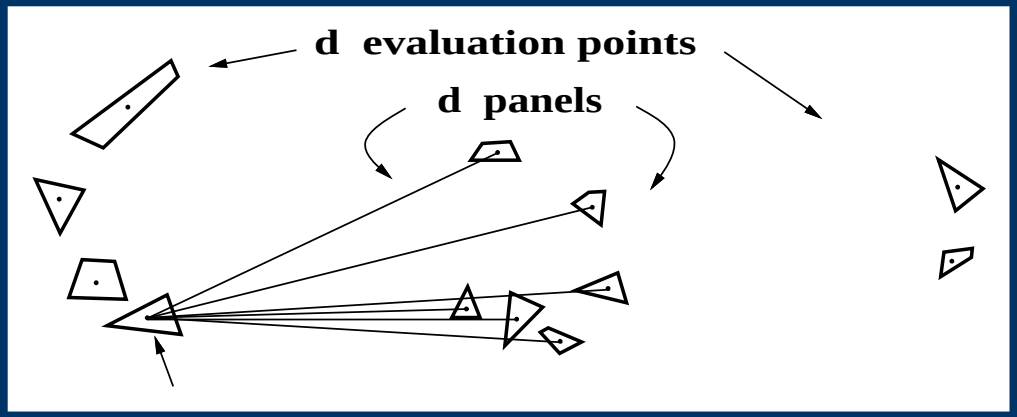
Dense matrix-vector (M-V) product costs $O(n^2)$

Approximately compute $\mathbf{A}p_k$

Reduces M-V product costs to $O(n)$ or $O(n\,log\,n)$

Need a fast approximation for matrix-vector products

Direct Computation



- Physical interpretation:

  $\mathbf{A}p = N$ "potentials" due to N charges.

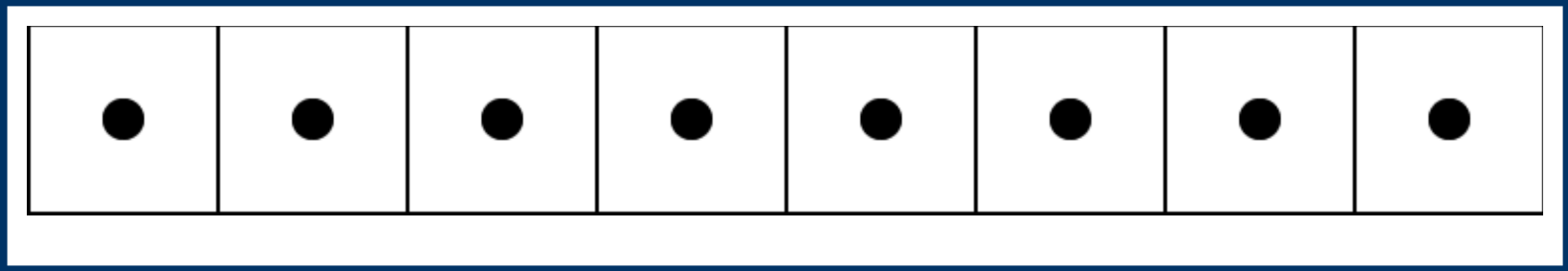
- $O(N^2)$ if done naively

Fast Solvers

1D Strip of Charge in 3D Space

Simplification of the $\mathbf{A}$ Matrix

1-D Strip of Charge in 3-D Space



$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{18} \\ A_{21} & A_{22} & \cdots & A_{28} \\ \vdots & \vdots & \ddots & \vdots \\ A_{81} & A_{82} & \cdots & A_{88} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_8 \end{bmatrix} = \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_8 \end{bmatrix}$$

What can one say about the $\mathbf{A}$ matrix?

The A matrix is:

- Symmetric

  Potential at j due to unit charge i = Potential at i due to unit charge j

- All the Diagonal Values are the Same

$$A_{ii} = \int_{\text{panel}_i} \frac{1}{\|\vec{x}' - \vec{x}_{c_i}\|}$$

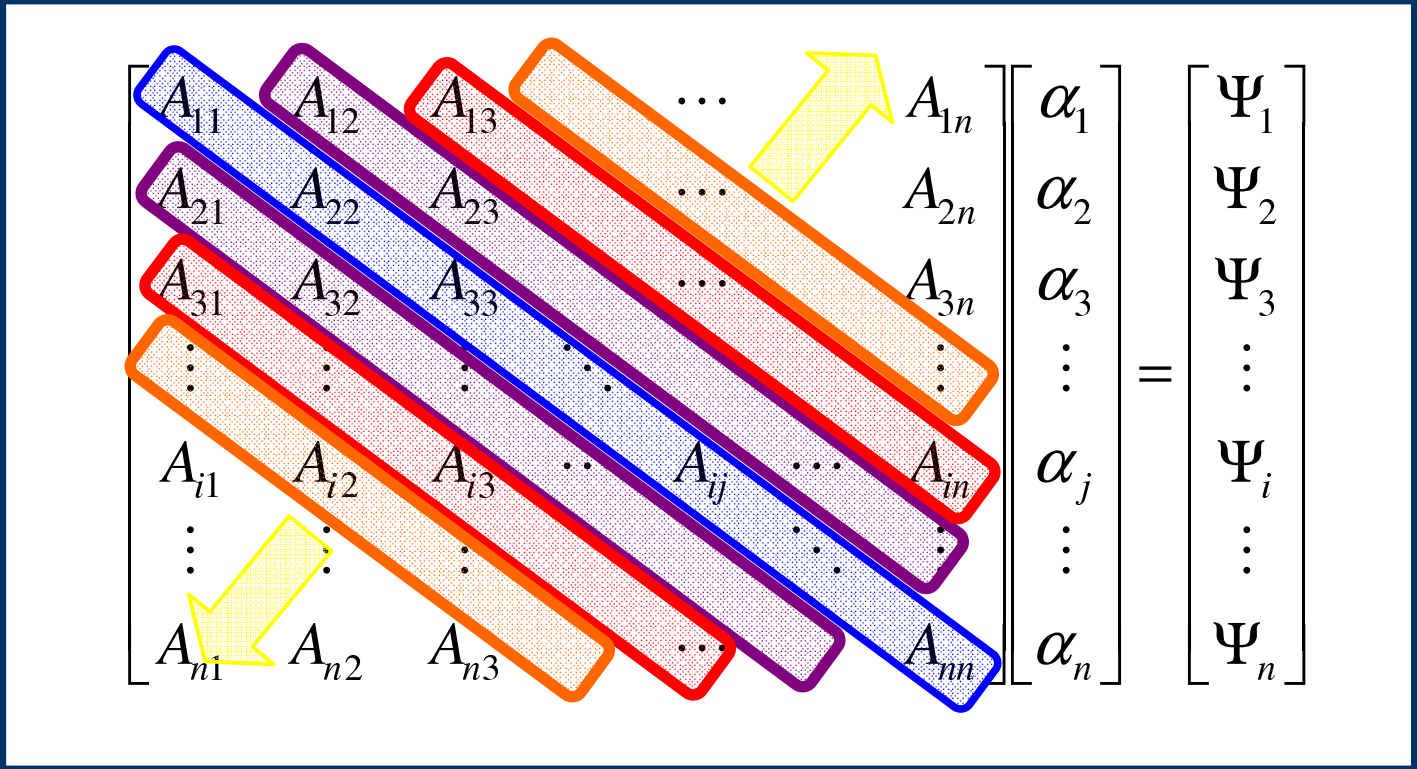
- Each Superdiagonal & Subdiagonal Element is Equal along Its Own Diagonal as Well

Fast Solvers

1D Strip of Charge in 3D Space

More Properties of $\mathbf{A}$.

How many unique entry values are there in $\mathbf{A}$*?*

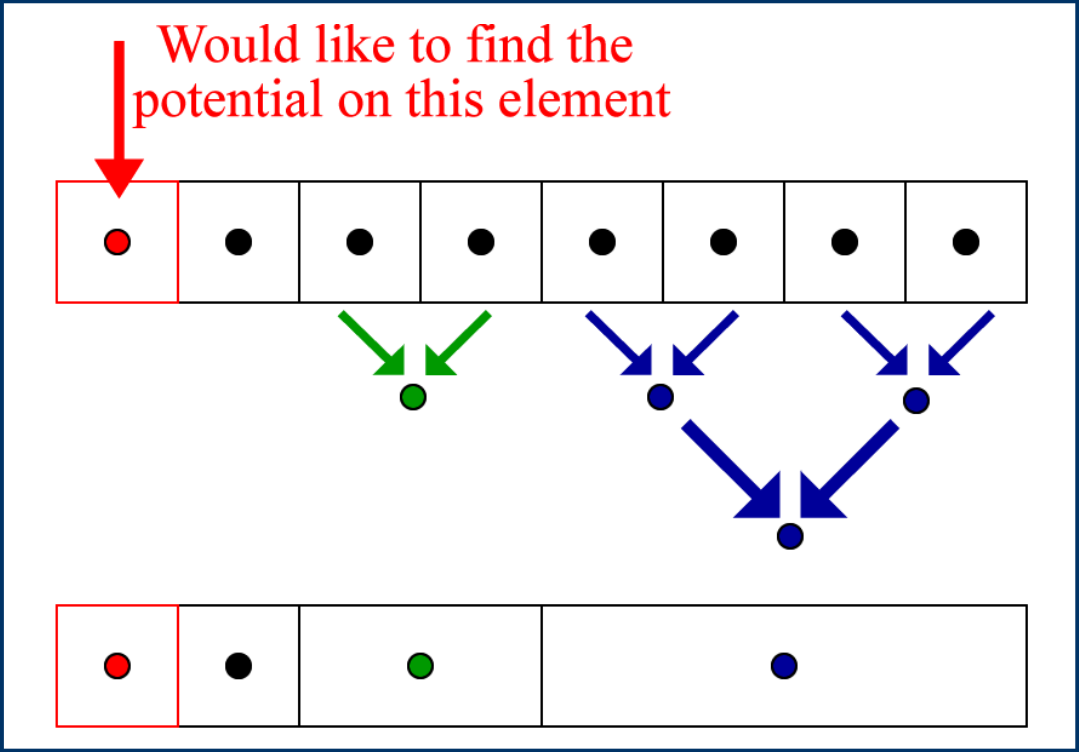


Fast Solvers

1D Strip of Charge in 3D Space

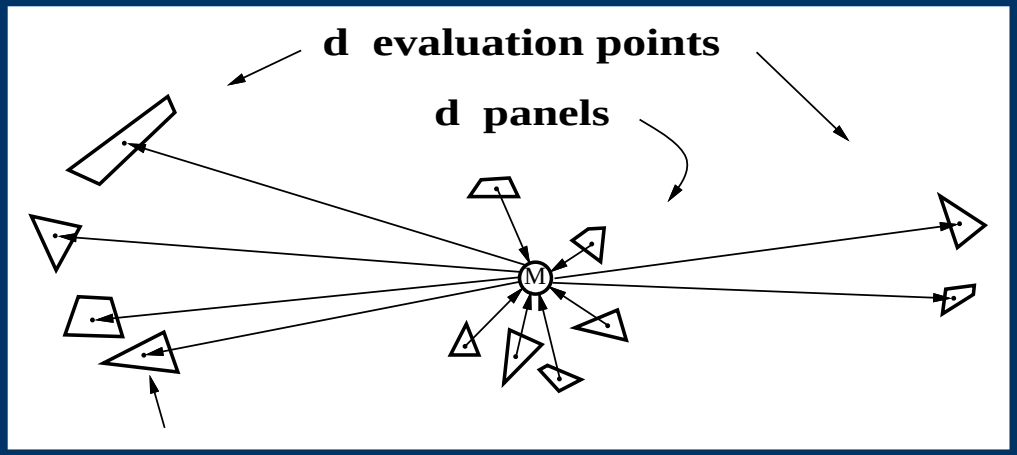
Geometric Simplification

Approximate (by grouping) the elements that are a “reasonable distance” away from the element which you are evaluating



Fast Solvers

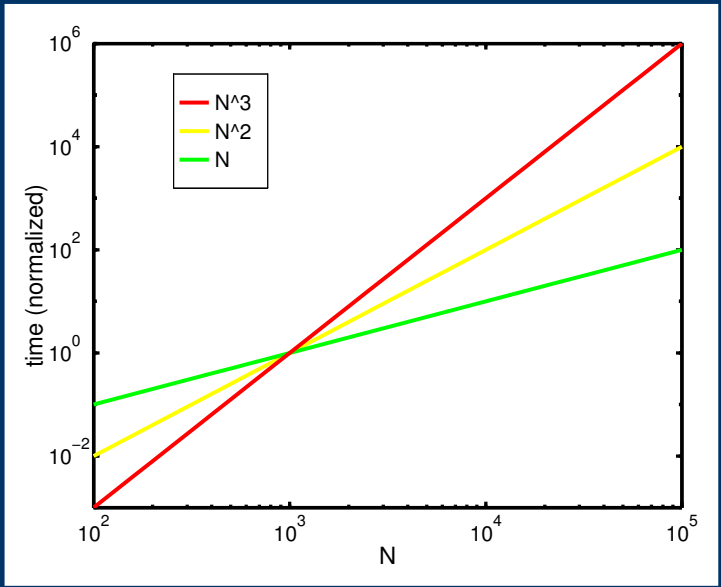
Fast Potential Concept



- Decompose potential into short- and long- range.
- Approximate long-range part of potential.
- Sum short-range part in normal manner
- Multilevel decomposition for "$O(N)$" algorithm

Fast Solvers

Computational Costs

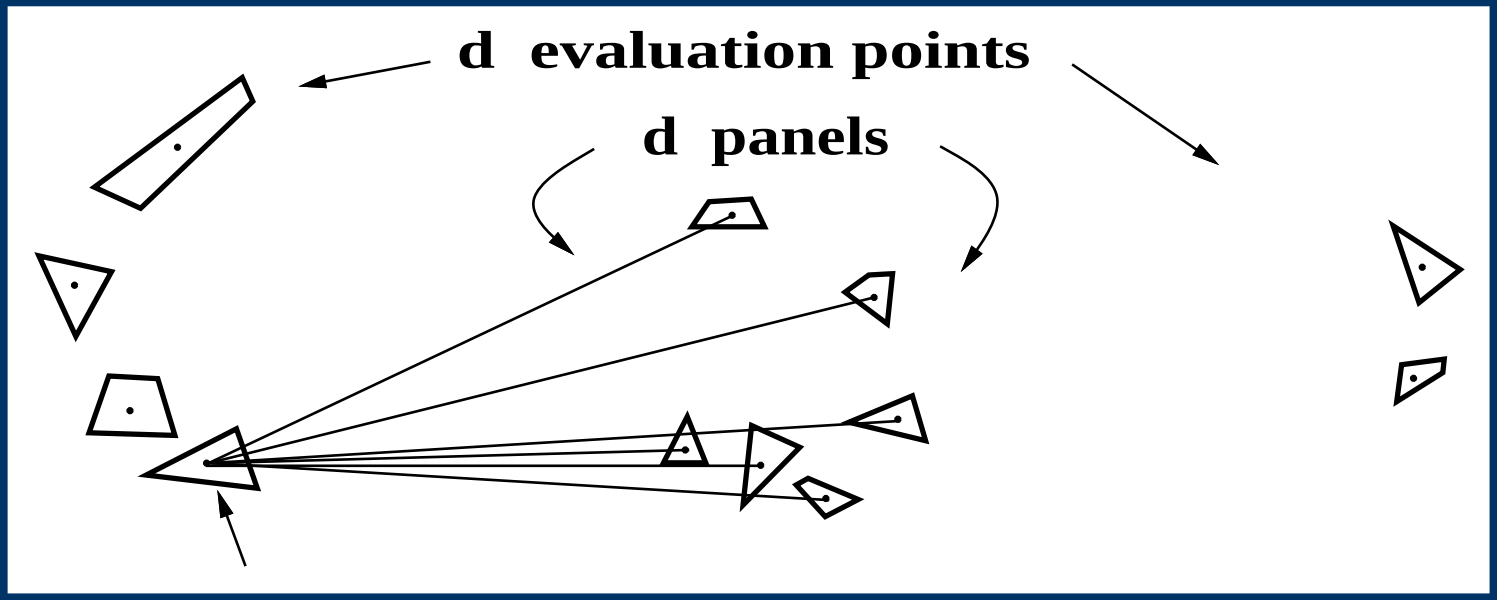


DEC 21164-333		
N	Gaussian Elim	"Fast" $O(N)$
	300 MFLOPS	30 MFLOPS
5e4	3 days,20GB	80sec, 130M
1e5	25 days,80GB	2.5min, 300M
5e5	8.8yrs,2TB	15min, 1.5GB

- Gaussian Elimination: $O(n^3)$ time, $O(n^2)$ memory
- Iterative with direct M-V: $O(n^2)$ time, $O(n^2)$ memory
- Fast Methods: $O(n)$ time, $O(n)$ memory

Multipole Algorithms

Direct Potential Evaluation



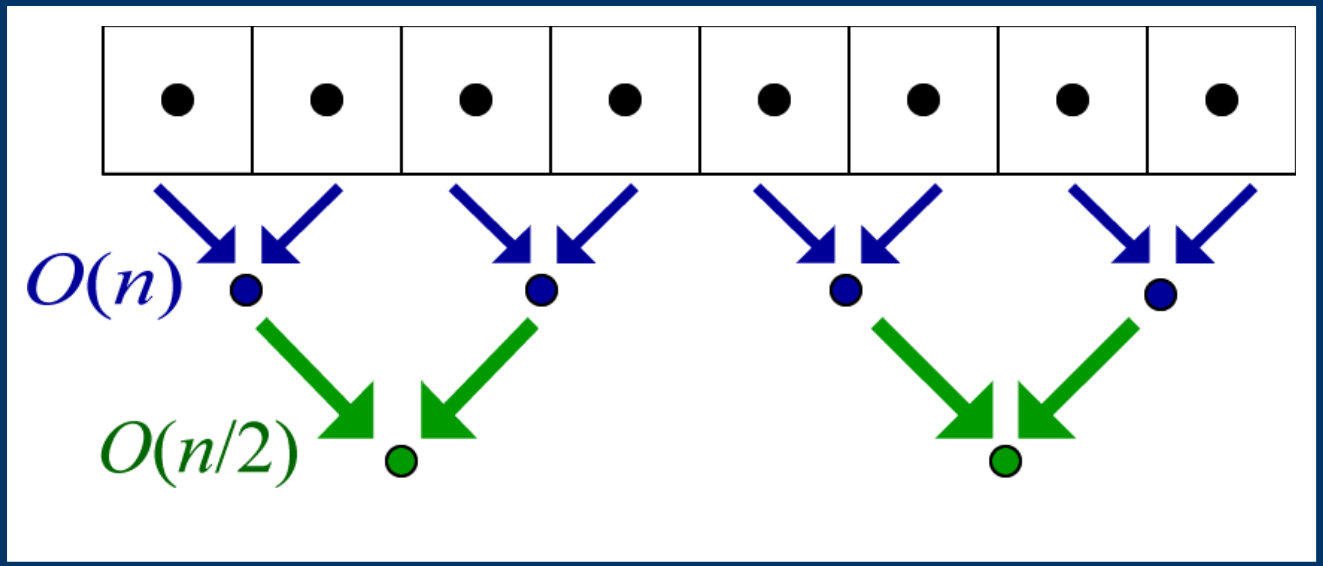
- Potential at point i: $v_i(r_i, \phi_i, \theta_i) = \sum_{j=1}^{d} q_j P_{ij}$.
- Complete evaluation at d points costs d^2 operations.

Multipole Algorithms

Multipole Representation

1D Strip in 3D Space...

How many operations are needed to form the clusters?



The cost of forming clusters is, in general,

$$O(n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \ldots) \approx O(n)$$

Multipole Algorithms

Multipole Representation

...1D Strip in 3D Space

What is the cost of estimating the evaluation point potential?

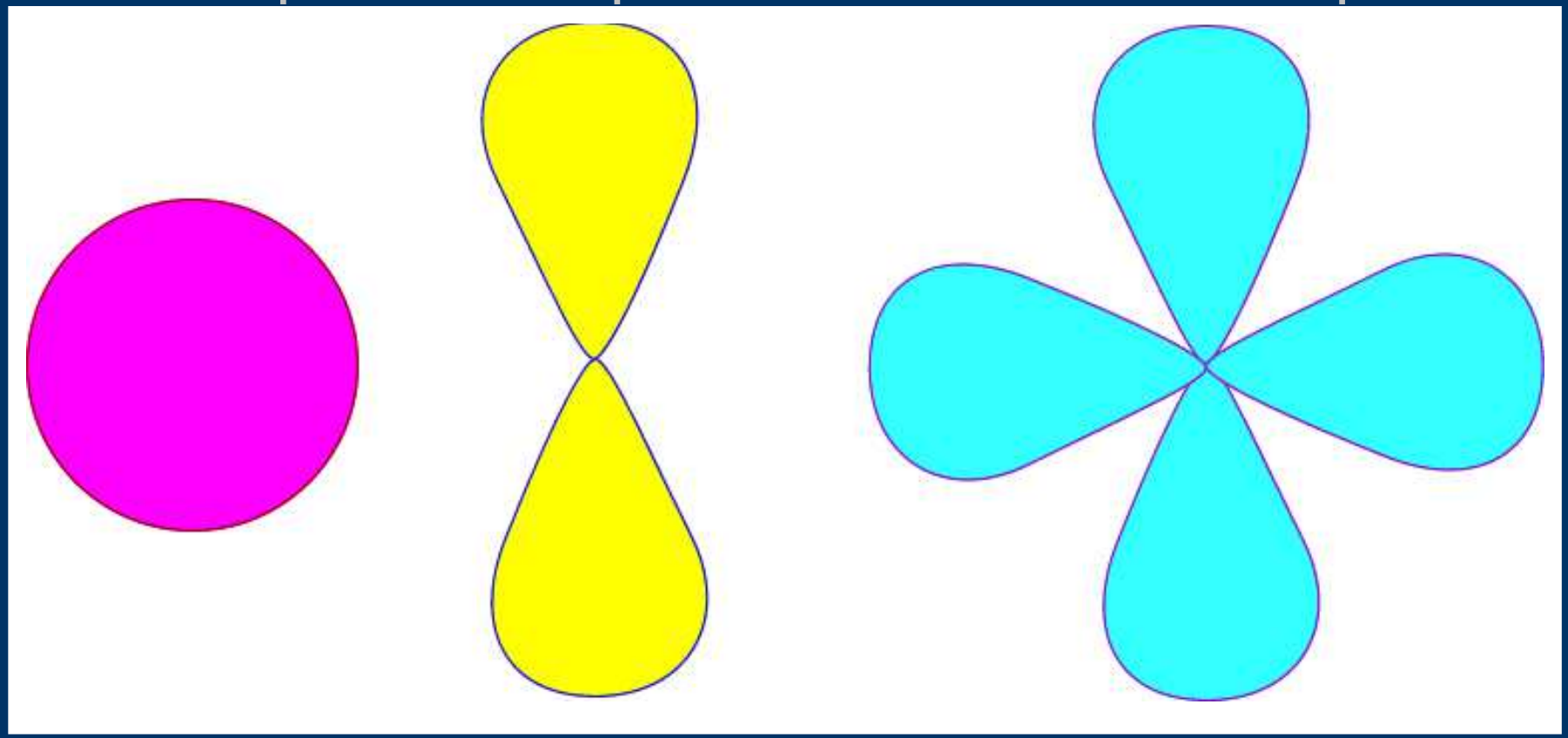
The cost of gathering clusters is $O(n \log n)$

Multipole Algorithms

Multipole Representation

Computational Example

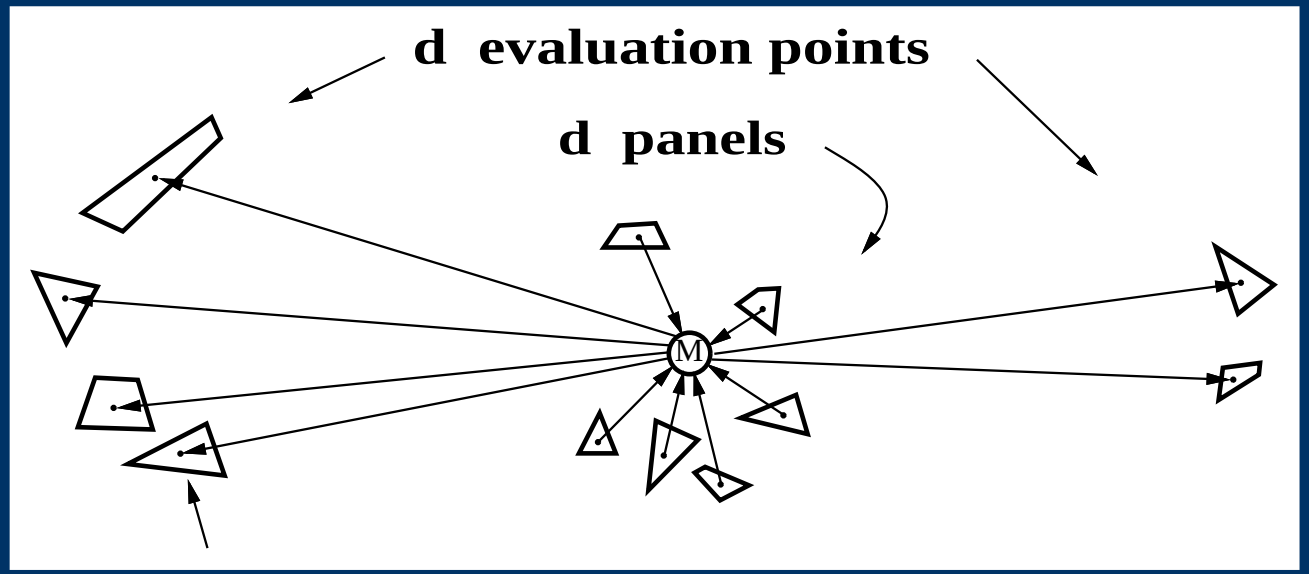
Monopole Dipole Quadrapole



Multipole Algorithms

Multipole Representation

General Case...



- Approximate potential at point i:

$$v_i(r_i, \phi_i, \theta_i) \approx \sum_{j=0}^{order} \sum_{k=-j}^{j} \frac{M_j^k}{r_i^{j+1}} Y_j^k(\phi_i, \theta_i)$$

Multipole Algorithms

Multipole Representation

...General Case

- Multipole coefficients function of panel charges:

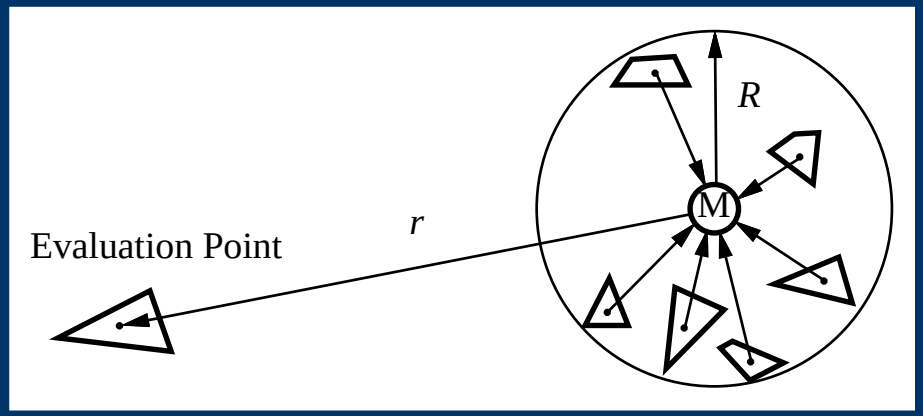
$$M_j^k \triangleq \sum_{i=1}^{d} \frac{q_i}{A_i} \int_{\text{panel } i} \rho^j Y_j^{-k}(\alpha, \beta) dA.$$

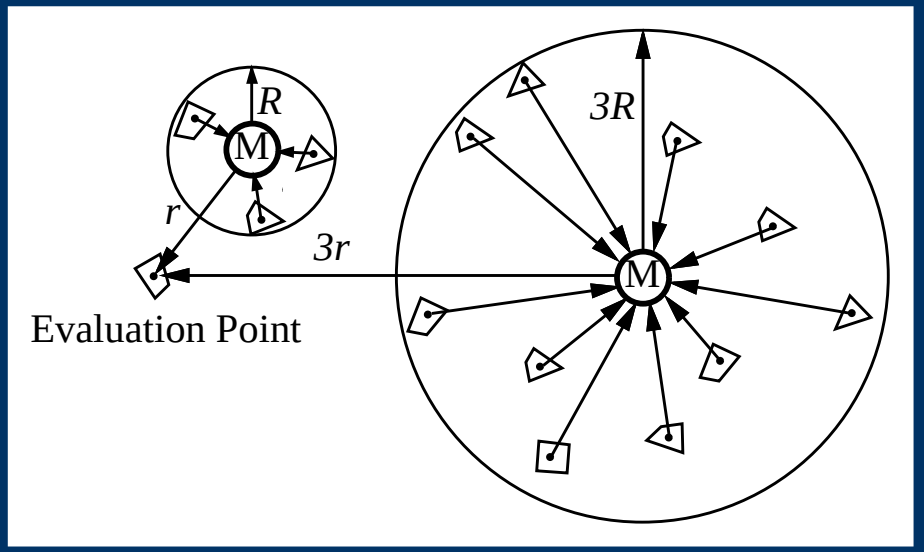
- Computing Multipole expansions costs order d operations.
- Each approximate potential evaluation costs order 1 operations.

d potential evaluation due to d panels in order d operations

Multipole Algorithms

Error Scale Invariance



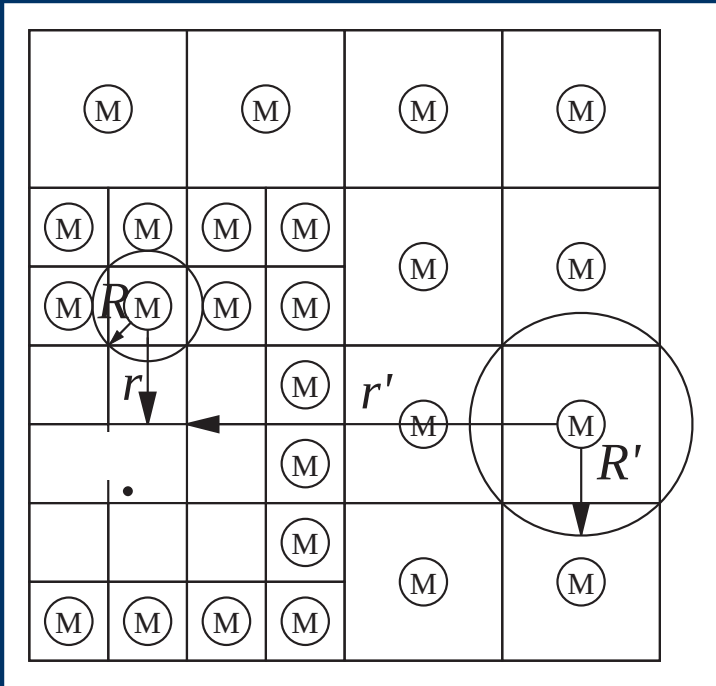


$$\text{Error} \leq \boldsymbol{K} \left(\frac{\boldsymbol{R}}{\boldsymbol{r}}\right)^{\boldsymbol{order+1}}$$

$$\text{Error} \leq \boldsymbol{K} \left(\frac{\boldsymbol{3R}}{\boldsymbol{3r}}\right)^{\boldsymbol{order+1}}$$

Multipole Algorithms

Multipole Algorithm Hierarchy



Hierarchy guarantees:

- Bounded error:

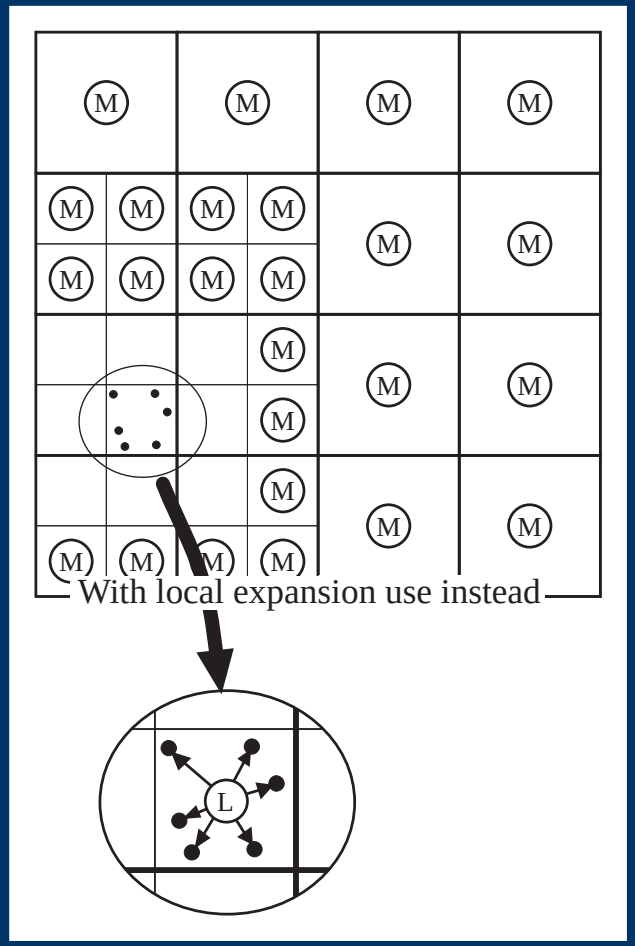
  $$\text{Error} \leq K \left(\frac{R}{r}\right)^{order+1} \leq K \left(\frac{1}{2}\right)^{order+1}$$

  $order = 2$ yields one percent accuracy.
- Order n ops for n potentials.

Multipole Algorithms

Local Representations

Cost Reduction

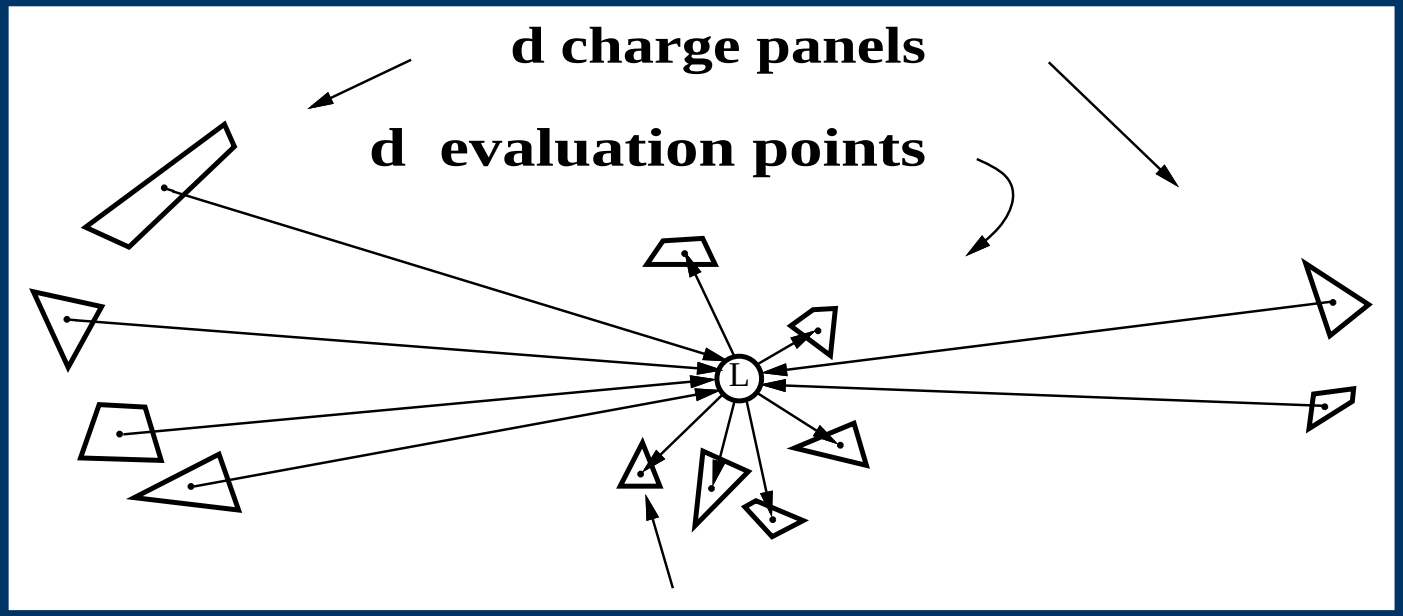


- Construct a local expansion to represent distant charge potentials.
- Evaluate a single local expansion, rather than many multipole expansions, at each evaluation point.

Multipole Algorithms

Local Representations

Clustered Evaluations



- Local expansion summarizes the influence of distant charge for clusters of evaluation points.

Multipole Algorithms

Local Representations

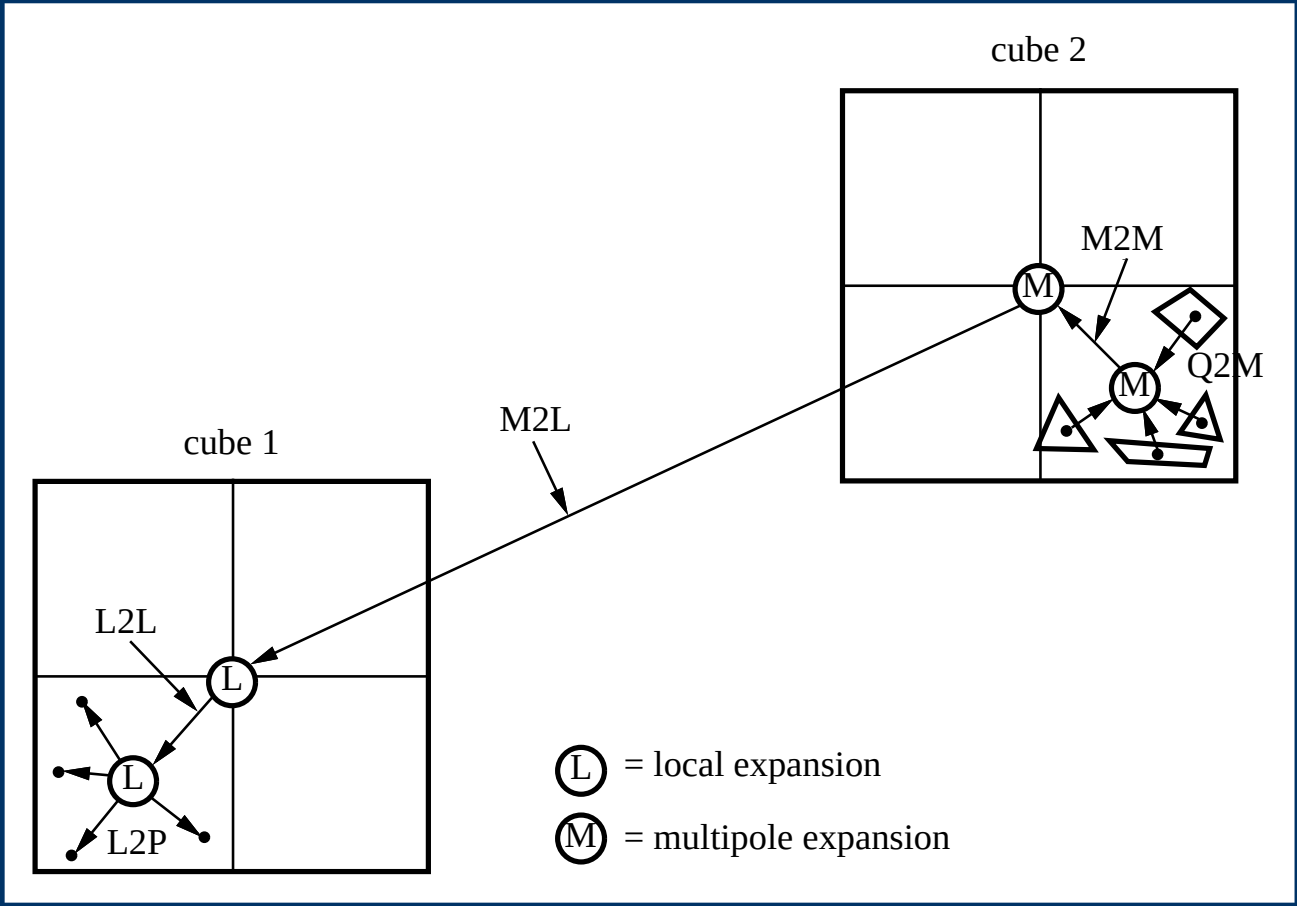
Clustered Evaluations

- Gives $O(n)$ potential evaluation when combined with coalescing of charge done by multipole expansions.
- Approximate potential at point i:

$$v_i(r_i, \phi_i, \theta_i) \approx \sum_{j=0}^{order} \sum_{k=-j}^{j} L_j^k Y_j^k(\phi_i, \theta_i) r_i^j.$$

Local Representations

Summary of Operations



Local Representations

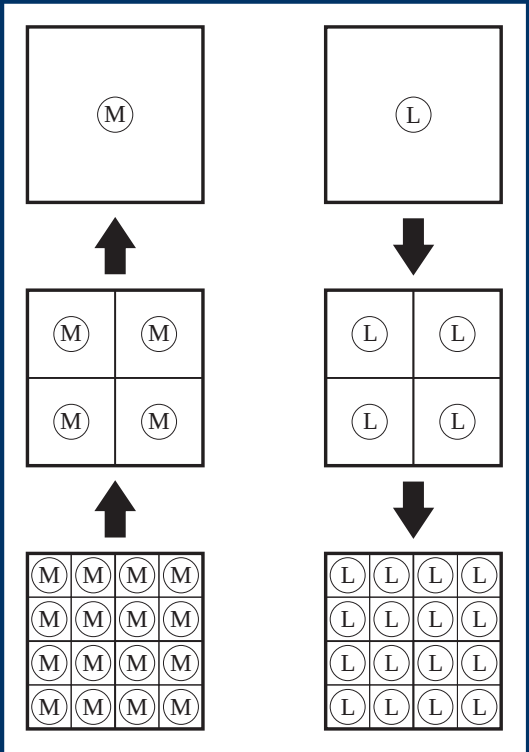
Summary of Operations

- Multipole and local expansions are built using complementary hierarchies.
- Complete calculation consists of:
  1. Build multipoles (Upward Pass).
  2. Build locals (Downward Pass).
  3. Evaluate local expansions and nearby charge potential (Evaluation Pass).

Multipole Algorithms

Local Representations

Hierarchy Construction

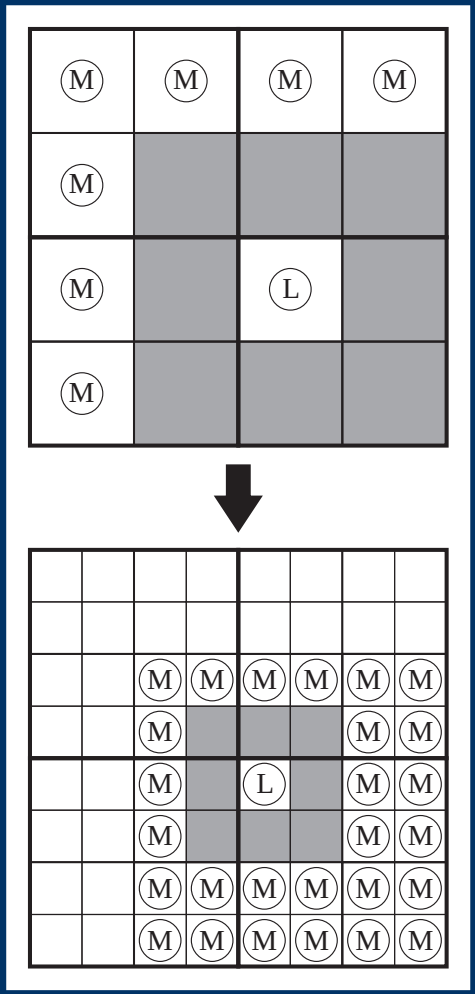


- First build the multipole expansions moving upward from child to parent.
- Then build the local expansions by moving downward from parent to child.
- Computation has a tree structure.

Multipole Algorithms

Local Representations

Construction Details



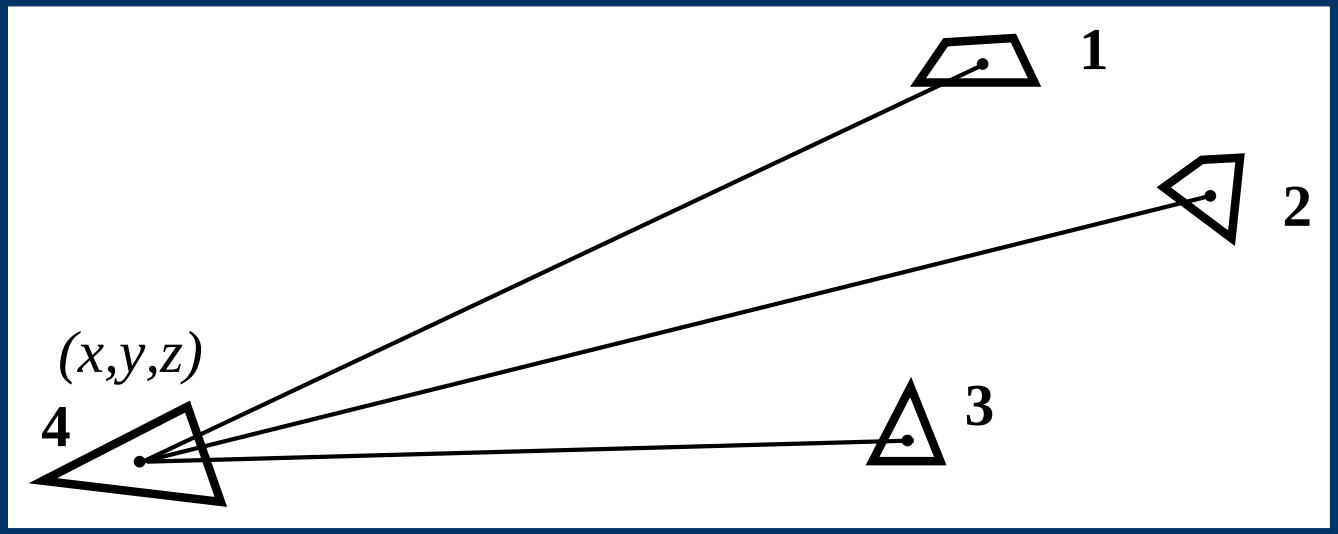
- Conversion of multipole expansions to local expansions.
- A child's local expansion is its parents local expansion plus conversions of multipole expansions in child's interaction range.

Multipole Algorithms

Adaptive Algorithm

Multipole Inefficiency...

Direct Evaluation



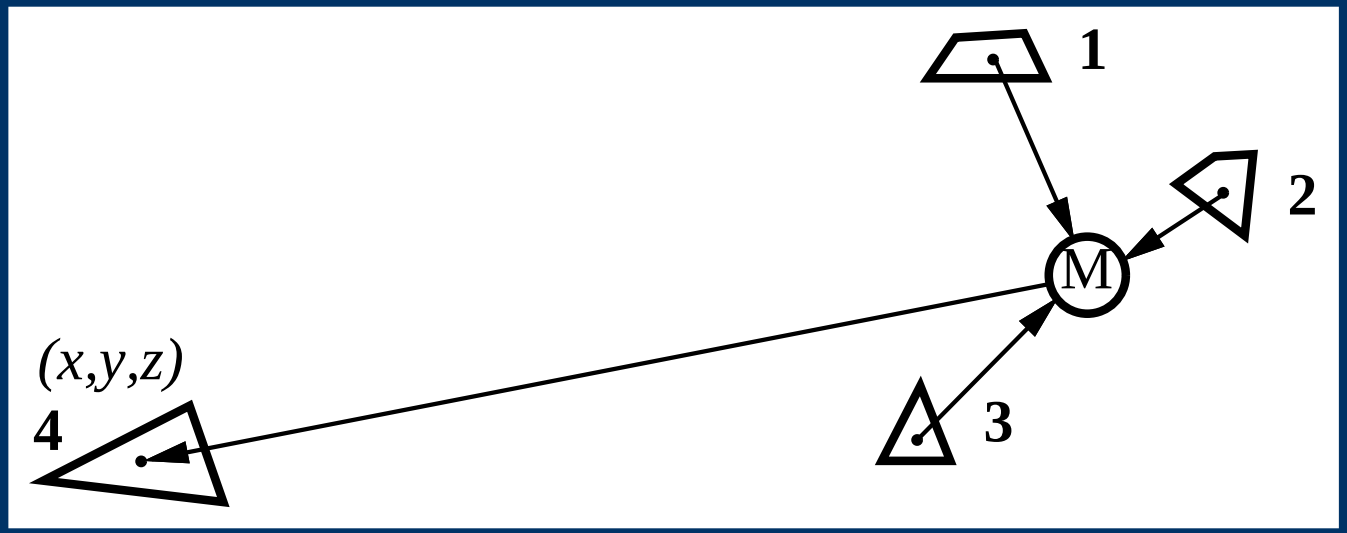
$$v_4(x,y,z) = q_1 P_{41} + q_2 P_{42} + q_3 P_{43}$$

Multipole Algorithms

Adaptive Algorithm

...Multipole Inefficiency

Multipole Evaluation



$$v_4(x,y,z) \approx \bar{M}_0^0 \frac{1}{r} + \bar{M}_1^0 \frac{z}{r^3} - \bar{M}_1^1 \frac{x}{2r^3} - \tilde{M}_1^1 \frac{y}{2r^3}$$

Using Multipole MORE expensive than Direct.

Multipole Algorithms

Adaptive Algorithm

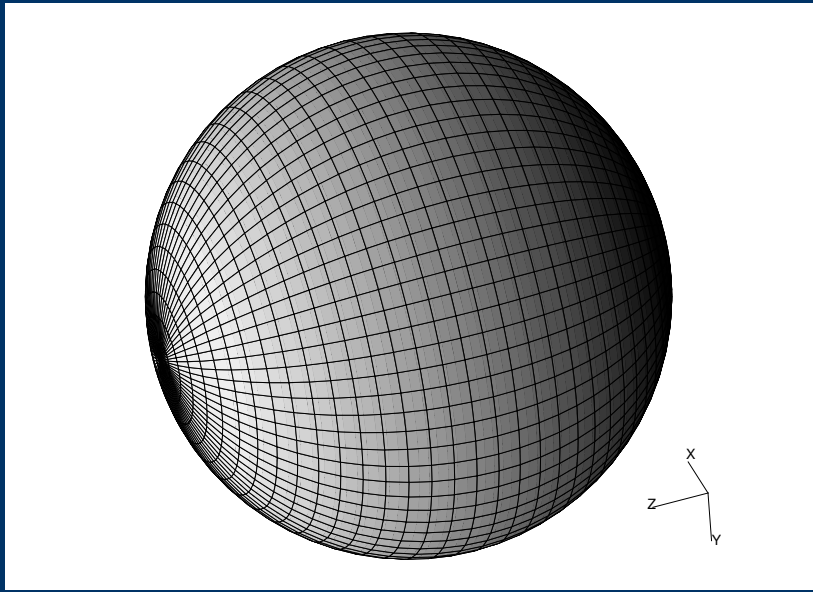
Simple Adaptive Scheme

If there are fewer panels than multipole coefficients, calculate the panels' influence directly.

- Similarly, local expansions are not used if there are fewer evaluation points than local expansion coefficients.
- Retains $O(mn)$ complexity for nonuniform panel distributions.

Computational Examples

Sphere Potential Distribution

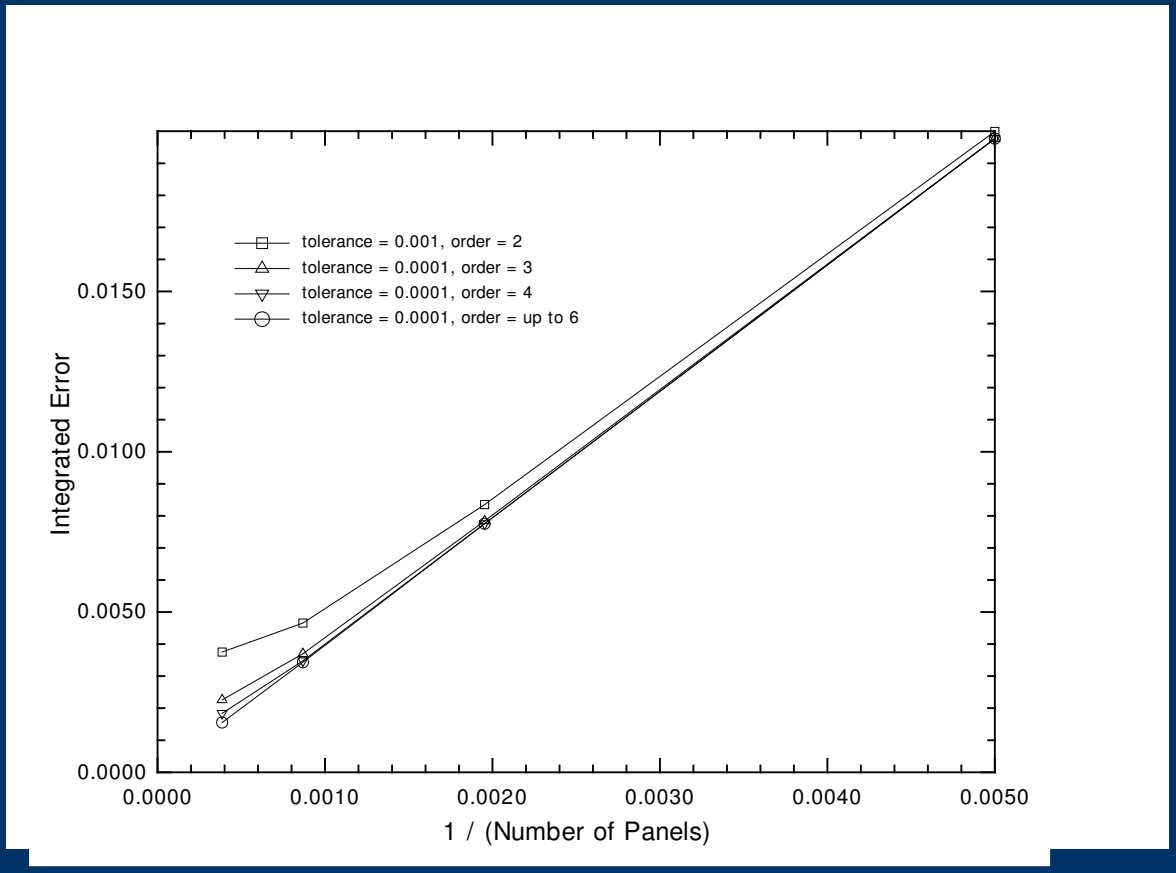


- Potential given by $\psi(x) = -\frac{x_3}{2\|x\|^3}$.
- Charge given by $\sigma(x) = \frac{-3}{8\pi} x_3$.

Multipole Algorithms

Computational Examples

Sphere Potential Distribution



- Error should decay like $\frac{1}{n}$.

Multipole Algorithms

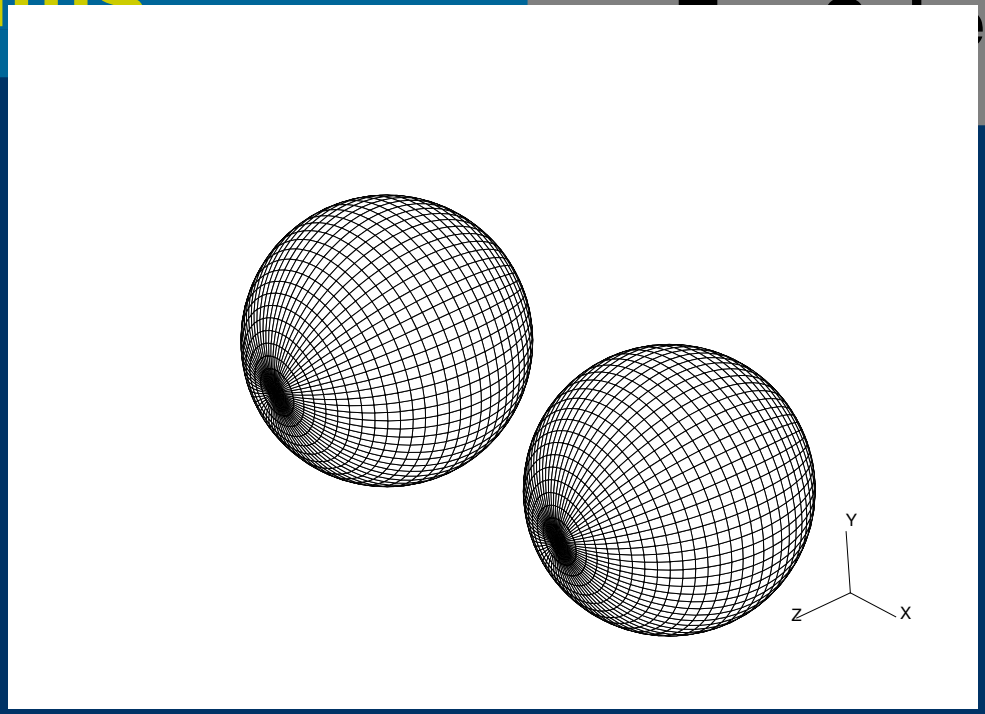
Computational Examples

Sphere Potential Distribution

- Multipole approximations eventually interfere.
- Higher-order multipole expansions needed for higher accuracy.

Computational Examples

ere Example

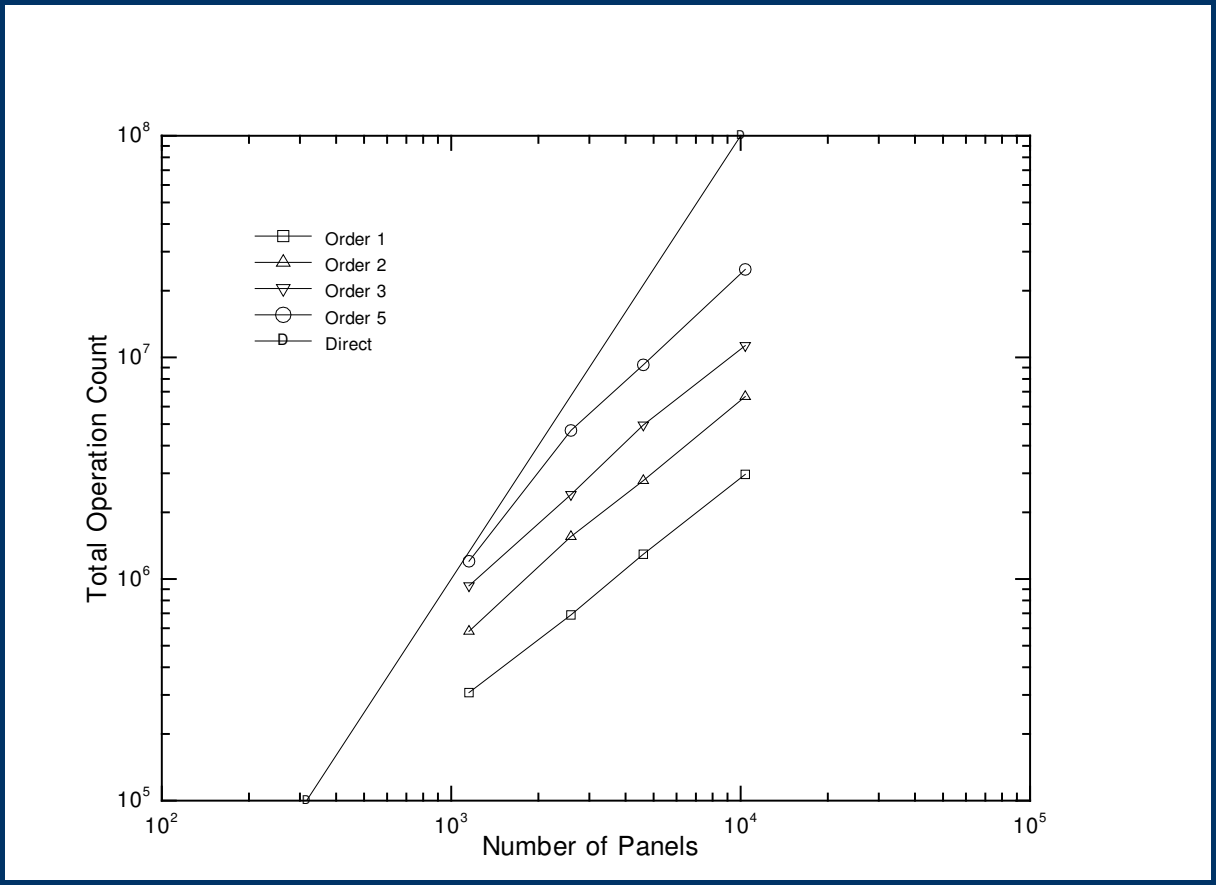


- Potential on each sphere: $\psi(x) = -\frac{x_3}{2\|x\|^3}$.
- Does not correspond to a simple physical problem.

Multipole Algorithms

Computational Examples

Two Sphere Example



- Direct matrix-vector product cost increases like n^2.

Multipole Algorithms

Computational Examples

Two Sphere Example

- Multipole matrix-vector product cost increases like n.
- The slope for the multipole algorithm depends on accuracy.
- For order 2 expansions, breakpoint is about $n = 400$.

Multipole Algorithms

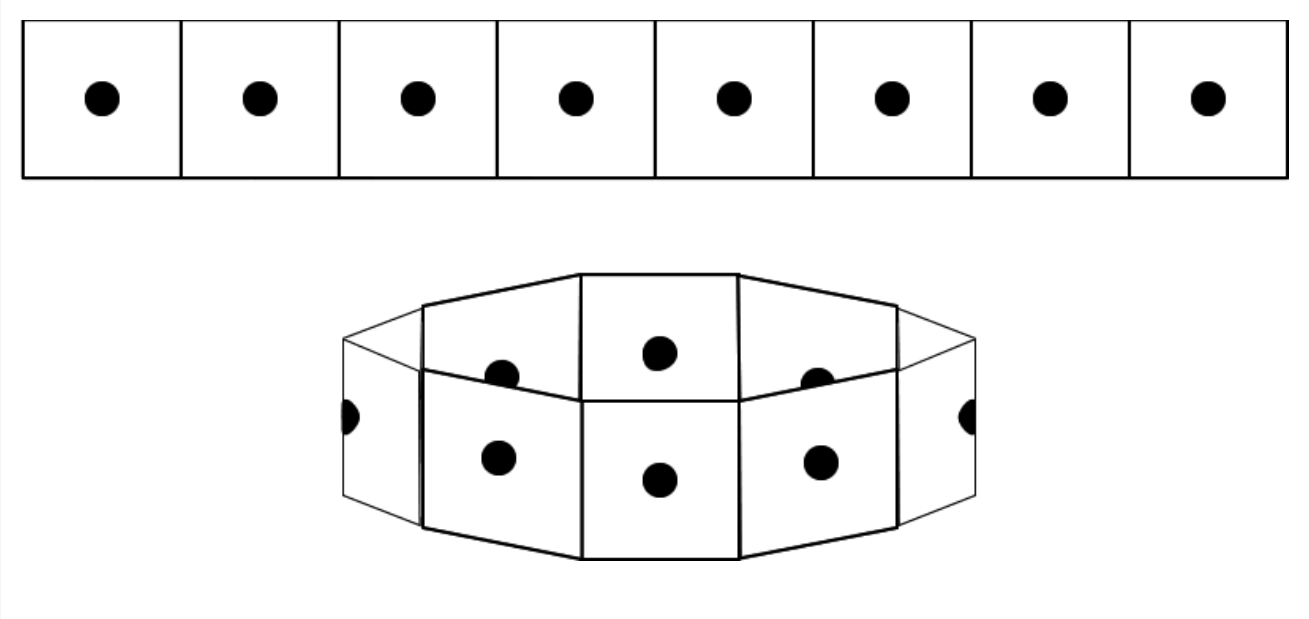
Complexity Summary

For an integral equation discretized with n panels:

- Gaussian elimination: $O(n^3)$.
- Iterative Matrix Solution, direct M-V $O(n^2)$.
- Multipole accelerated Iterative method $O(n)$.

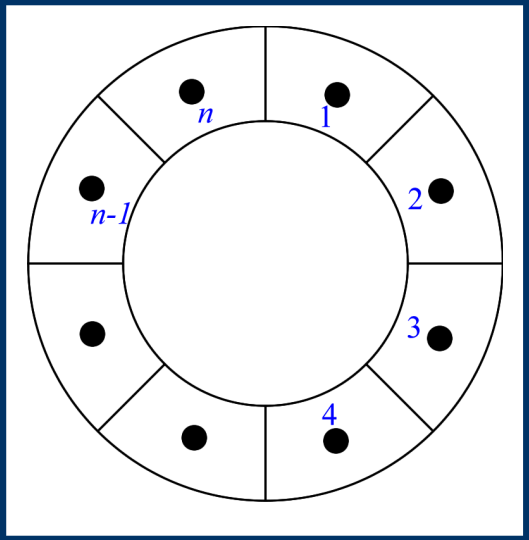
Strip of Charge in Space

Bring the ends of the strip of charge together to form a ring.



Precorrected-FFT

Introduction

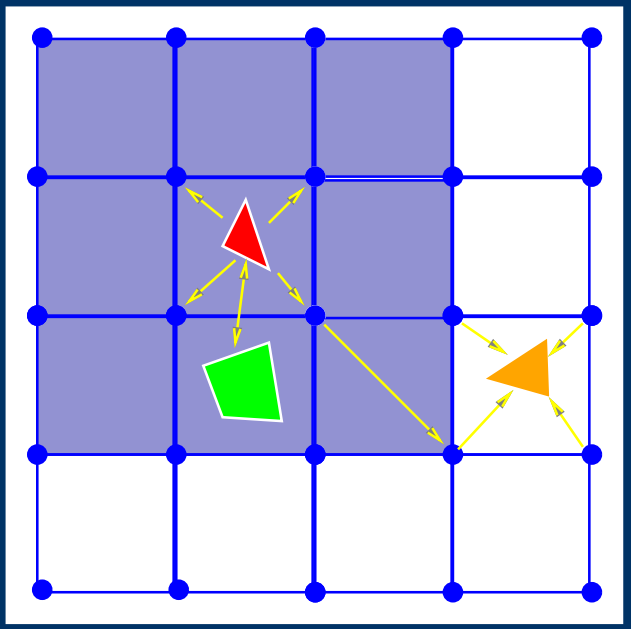


Produces a "**Circulant Matrix**"

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2n} \\ A_{31} & A_{32} & A_{33} & \cdots & A_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{i1} & A_{i2} & A_{i3} & \cdots & A_{in} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_j \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \\ \vdots \\ \Psi_i \\ \vdots \\ \Psi_n \end{bmatrix}$$

Precorrected-FFT

Algorithm Outline

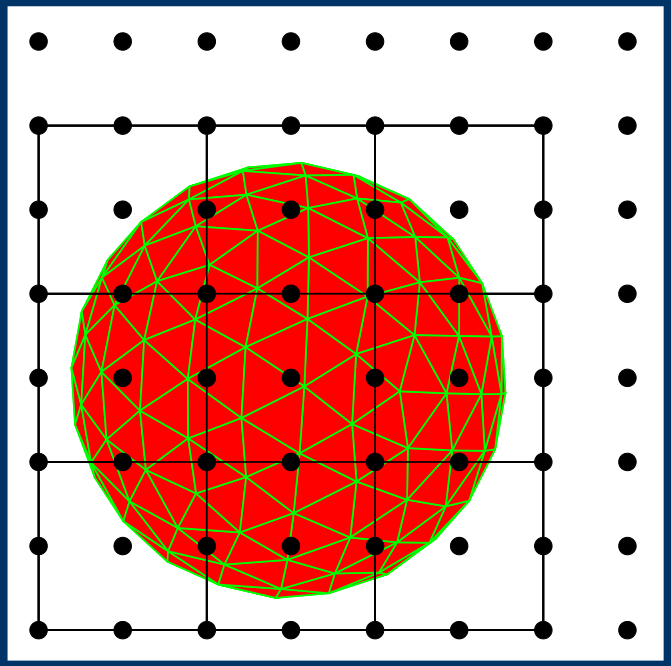


1. Project panel charges on grid
2. Calculate grid-charge potentials on grid
3. Interpolate grid potentials onto panels
4. Local corrections [compute nearby interactions directly]

Precorrected-FFT

Algorithm Outline

PFFT Grid Balances Costs

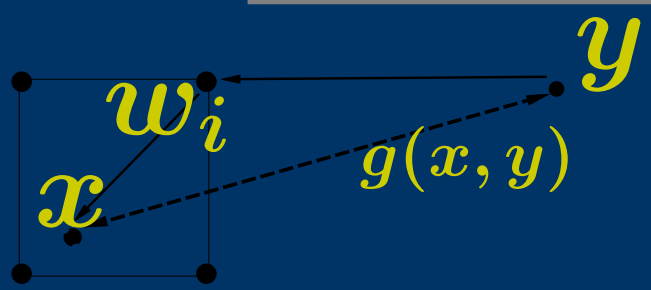


- Grid Selected So Direct Cost equals FFT Cost.
- Finer Discretizations Usually Yield Finer Grids.

Precorrected-FFT

Algorithm Analysis

Interpolation and Projection...



Approximate potential Ψ at x due to charge at y by *interpolating* potential using points and weights x_i, w_i

Precorrected-FFT

Algorithm Analysis

Interpolation and Projection...

Interpolate: potential at x due to unit charge at y

$$\Psi(x|y) \simeq \hat{\Psi}(x|y) = \sum w_i g(x_i, y)$$

Anterpolate: potential at y due to unit charge at x

$$\Psi(y|x) \simeq \hat{\Psi}(y|x) = \sum w_i g(y, x_i)$$

So

$$\hat{\Psi}(y|x) = \hat{\Psi}(x|y)$$

Same as representing charge at x with w_i and evaluating at y

Precorrected-FFT

Algorithm Analysis

...Interpolation and Projection

Equivalent conditions:

- Approx Potential in cell due to charge at large R.
- Approx Potential at large R due to charge in cell.
- Cost is $O(N)$

Precorrected-FFT

Algorithm Analysis

Grid Potentials

- Let H be grid charge-potential mapping

$$H : q_g \rightarrow \Psi_g$$

- H is Toeplitz
- Embed H in circulant matrix

$$\begin{bmatrix} \psi_g \\ x \end{bmatrix} = \begin{bmatrix} H & X \\ X & X \end{bmatrix} \begin{bmatrix} q_g \\ 0 \end{bmatrix}$$

- Use FFT for matrix multiply

Must Have Translation Invariance

Precorrected-FFT

Algorithm Analysis

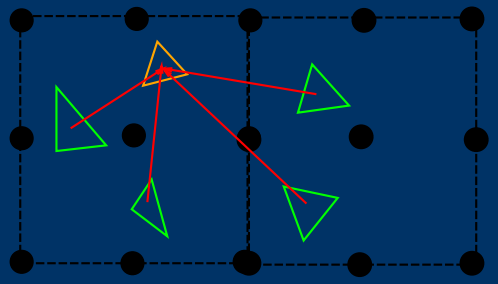
Grid Potentials

- Cost $O(M \log_2 M)$, $M =$ # cells

Precorrected-FFT

Algorithm Analysis

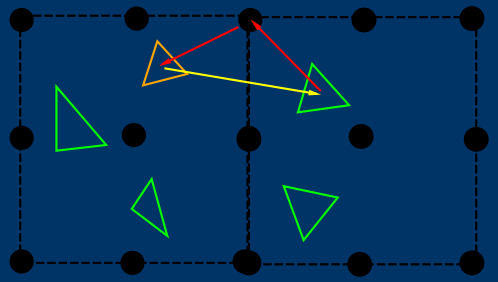
Nearby Interactions



Direct interactions

Cost $O(N\lceil n_c \rceil)$

$\lceil n_c \rceil$ = max # panels /cell



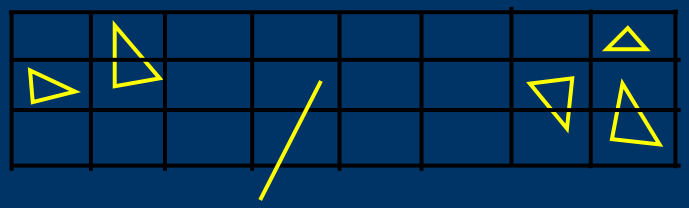
Local corrections

Cost $O(1) - O(N) - O(Nn_I^2)$

Precorrected-FFT

Algorithm Analysis

Inhomogeneity Problem

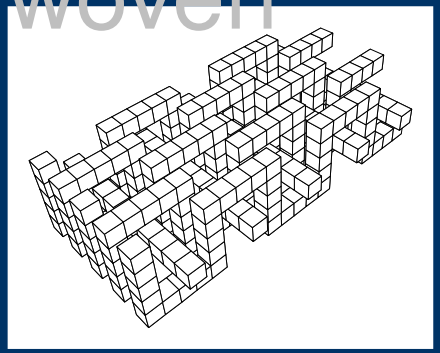


- Empty Grid due to FFT - Inefficient

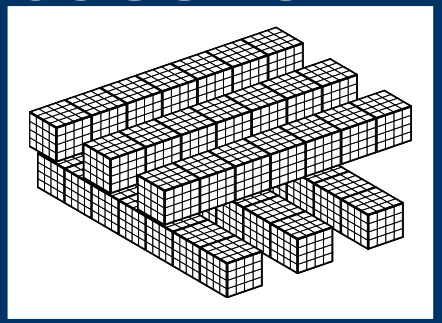
Precorrected-FFT

Examples

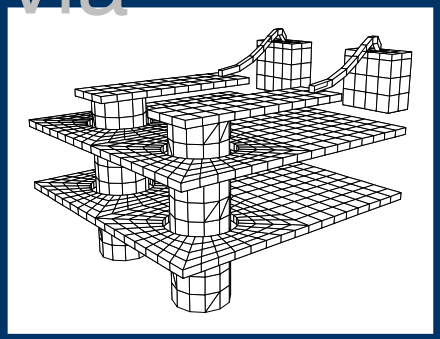
woven



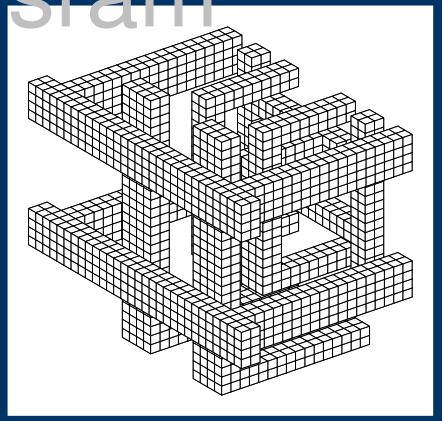
bus3x8



via



sram



Precorrected-FFT

PFFT vs. Multipole

- Comparisons: PFFT $p = 3$ to Multi $l = 2$

Example	CPU	Memory	Product	Error
via	0.61	0.37	0.23	0.18
woven5x5	0.45	0.48	0.22	0.09
cube	0.38	0.32	0.12	0.12
bus3x8	0.27	0.27	0.07	0.01
SRAM	0.39	0.43	0.17	0.07
mean	0.42	0.37	0.16	0.09

- Faster with $10\times$ better accuracy !

Precorrected-FFT

PFFT vs. direct

Memory

Example		Memory Usage	
Name	Panels[conductors]	P/FFT	Direct
via	6120[4]	21 Mb	(286 Mb)
woven5x5	9360[10]	50 Mb	(668 Mb)
woven15	82080[30]	246 Mb	(50.2 Gb)
cube	126150[1]	225 Mb	(119 Gb)

Precorrected-FFT

PFFT vs. direct

Time

Example	CPU Usage		
Name	P/FFT	Dir. Iter. Gauss. Elim.	
via	1.1 min	(5.6 min)	(1.9 hrs)
woven5x5	5.2 min	(42 min)	(6.9 hrs)
woven15	1.7 hrs	(11.5 days)	(194 days)
cube	3.3 min	(8.4hrs)	(2.7 yrs)

Summary

Reasons for Fast Solvers

- Collocation System Reminder

Fast Solver General Approach

- Using Iterative methods
- Fast matrix-vector products

Two Fast Methods

- Fast Multipole - Multiresolution
- Precorrected-FFT - Translation Invariance