
Algorithms, Implementation and Applications of pFFT++: Direct matrix, pre-correction and grid selection

Zhenhai Zhu

RLE Computational prototyping group, MIT

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

Outline

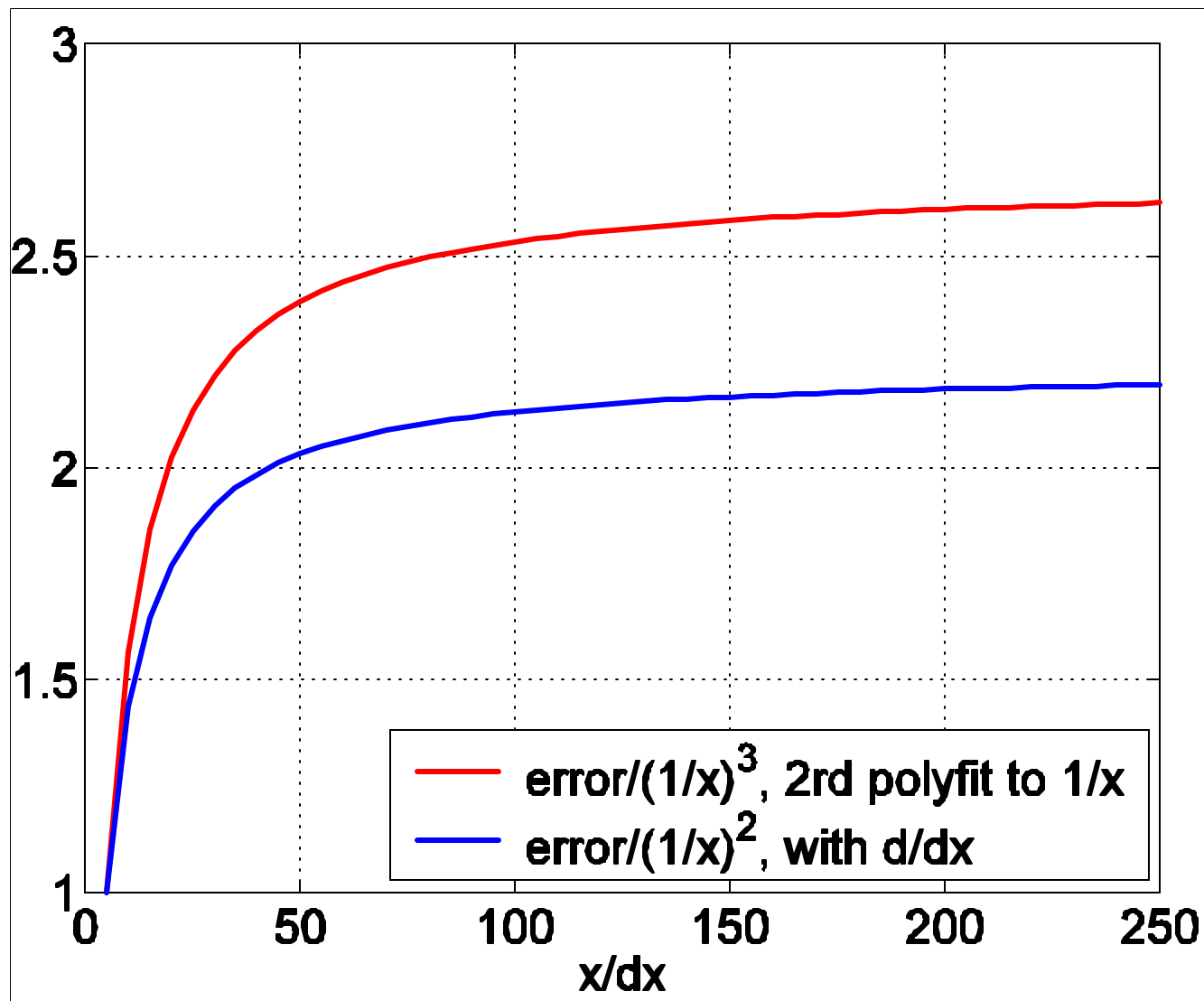
- **Derivation of interpolation error bound**
- **How to find nearby neighbors**
- **Pre-correction**
- **Grid selection**
- **Implementation details**

How to find nearby neighbors: derivation of error bound

- **1D interpolation**
- **3D interpolation (optional)**

All shown with chalk and board

How to find nearby neighbors: error bound for double-layer (1D)



How to find nearby neighbors: interpolation error bound

If a panel is within the region covered by the stencil, leading term in relative error is

$$e \approx \left(\frac{h}{r_0} \right)^{n+1}$$

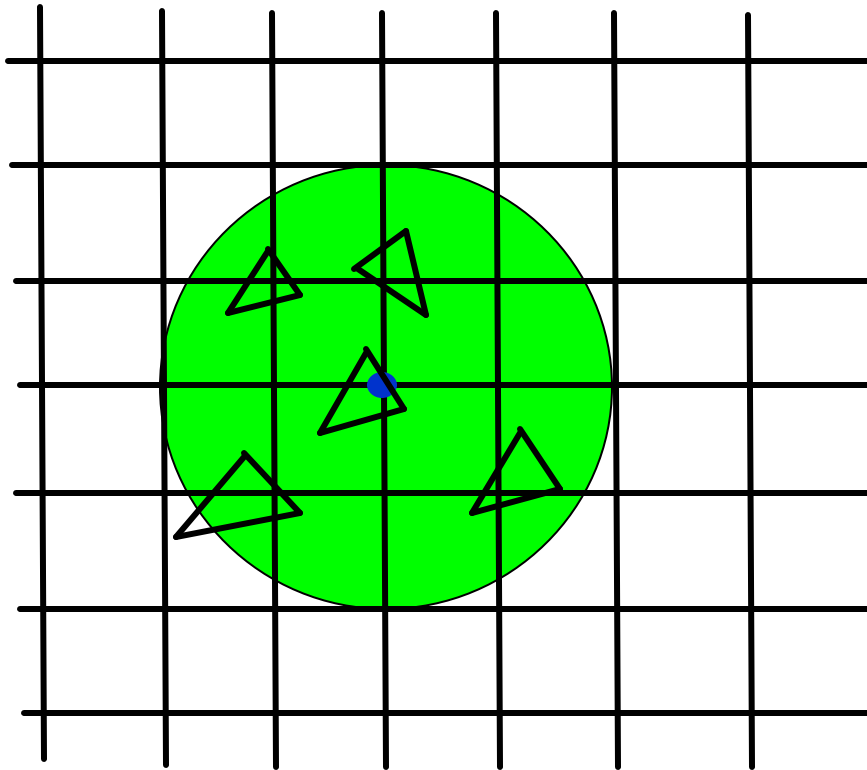
If some part of a panel hang out of the region covered by the stencil, leading term in relative error is

$$e \approx \left(\frac{a}{r_0} \right)^{n+1}$$

where a is the radius of the panel

How to find nearby neighbors: basic strategy

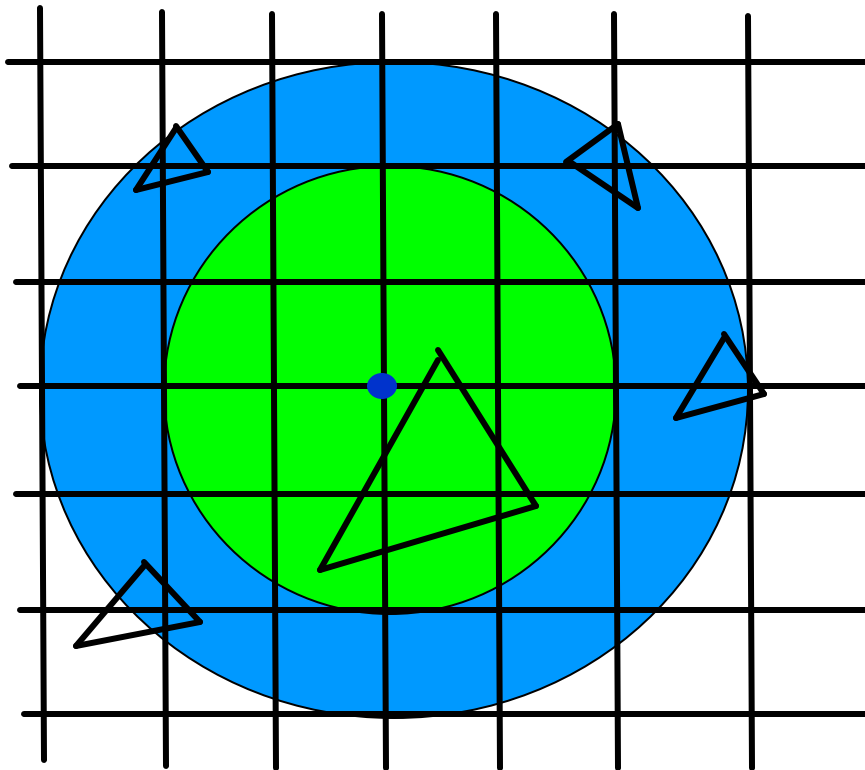
- For regular panels
 - Use direct stencil (sphere shape) as metric to find nearby neighbors



- Go through each point in the direct stencil and find the panels mapped to the point
- Add these panels to the regular neighbor list

How to find nearby neighbors: basic strategy

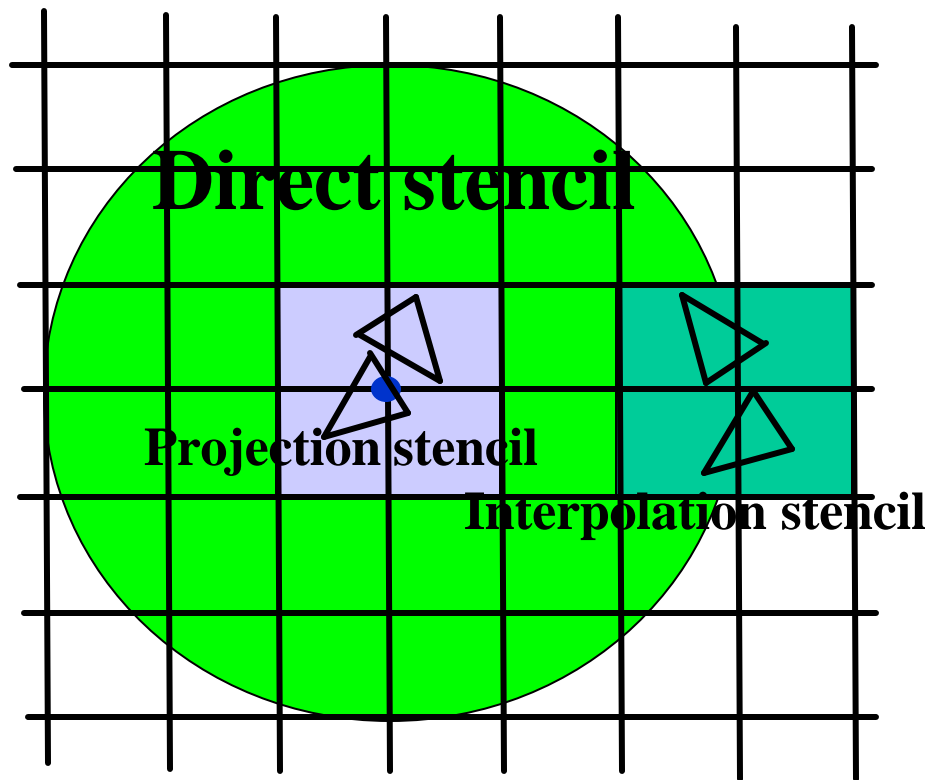
- **For large panels**
 - **Measure physical distance against the radius of the panels**



- **Find all grid points a certain distance from stencil center**
- **Exclude all direct stencil points and find the panels mapped to the remaining points**
- **Add these panels to the irregular neighbor list**

Pre-correction: regular neighbors

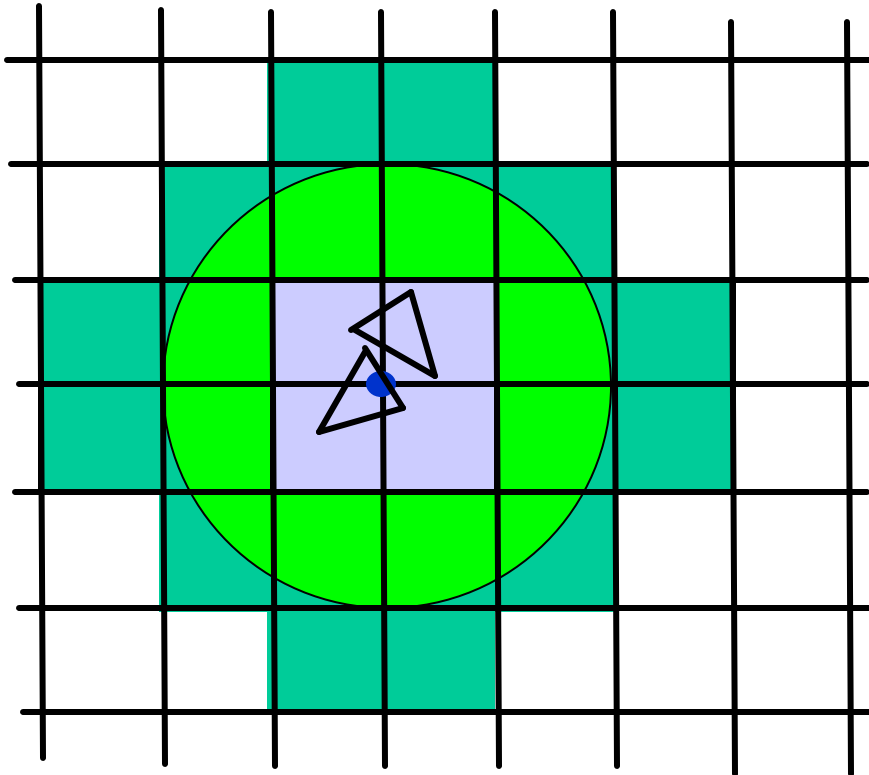
$$D_{i,j} = D_{i,j} - [I_i]_{1 \times 9} [H]_{9 \times 9} [P_j]_{9 \times 1}$$



- Find an interpolation stencil center around each direct stencil point
- Fill in a small convolution matrix $[H]$
- Pick appropriate row and column of $[I]$ and $[P]$, calculate the pre-correction
- Pre-correct the appropriate entry in $[D]$

Pre-correction: improvement on regular neighbors

$$D_{i,j} = D_{i,j} - [I_i]_{1 \times 9} [H]_{37 \times 9} [P_j]_{9 \times 1}$$



- Find the union of the interpolation stencil center around each direct stencil point (37 points in this figure)
- Fill in a long skinny convolution matrix $[H]$
- Pick appropriate column of $[P]$, calculate $[g] = [H][P]$
- Pick appropriate row in $[I]$ and $[g]$, calculate $[I][g]$
- Pre-correct the appropriate entry in $[D]$

Pre-correction: improvement on regular neighbors

- **Before:** $[H]_{9 \times 9} [P_j]_{9 \times 1}$

repeat n time, n being number of points in direct stencil

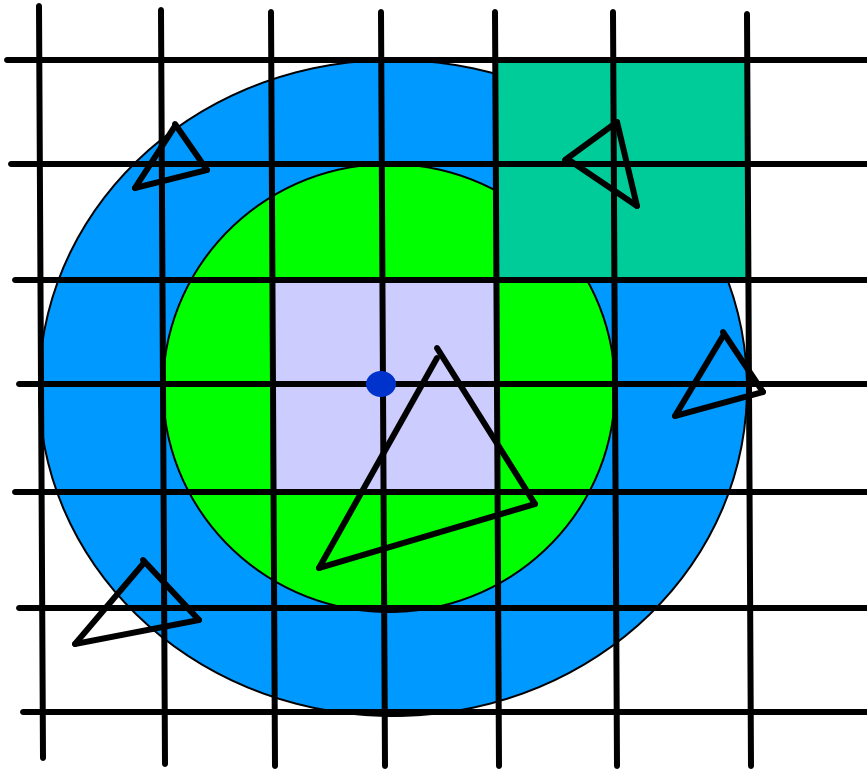
- **After:** $[H]_{37 \times 9} [P_j]_{9 \times 1}$

Do it just once. Keeping track of index may add some overhead cost.

- **Key: interpolation stencils share grid points**

Pre-correction: irregular neighbors

$$D_{i,j} = D_{i,j} - [I_i]_{1 \times 9} [H]_{9 \times 9} [P_j]_{9 \times 1}$$



- Find the interpolation stencil for each irregular neighbor
- Fill in a small convolution matrix $[H]$
- Pick appropriate row and column of $[I]$ and $[P]$, calculate the pre-correction
- Pre-correct the appropriate entry in $[D]$

Grid selection: minimize memory usage

- **Once the grid size satisfies certain constraints with respect to the element size, the accuracy is decided by the error bound.**
- **Larger number of grid points leads to larger memory usage by $[H]$, but $[D]$, $[I]$ and $[P]$ would be sparser.**
- **There is an optimal grid size that balance these two factors.**

Grid selection: An optimization problem

**Minimize memory usage by $[I]$, $[P]$, $[D]$ and $[H]$
Subject to**

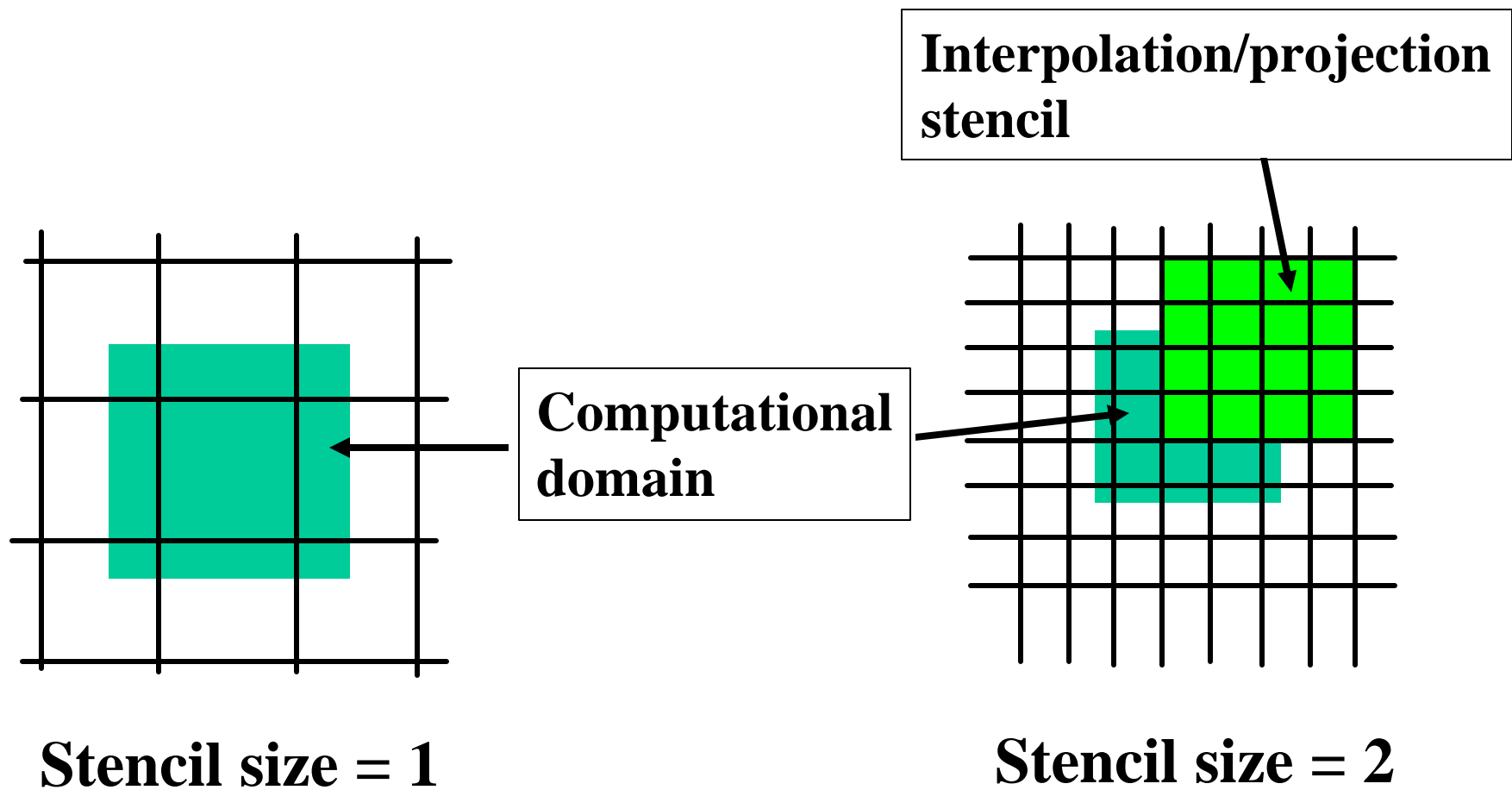
- 1. reasonable ratio between grid size and element size, no extrapolation.**
- 2. grid size is smaller than a tenth of a wavelength if the kernel is oscillatory.**
- 3. not too many elements associated with one grid point.**
- 4. Minimize size of the region occupied by grid.**
- 5. Number of grids is 2^n**

Grid selection: An optimization problem

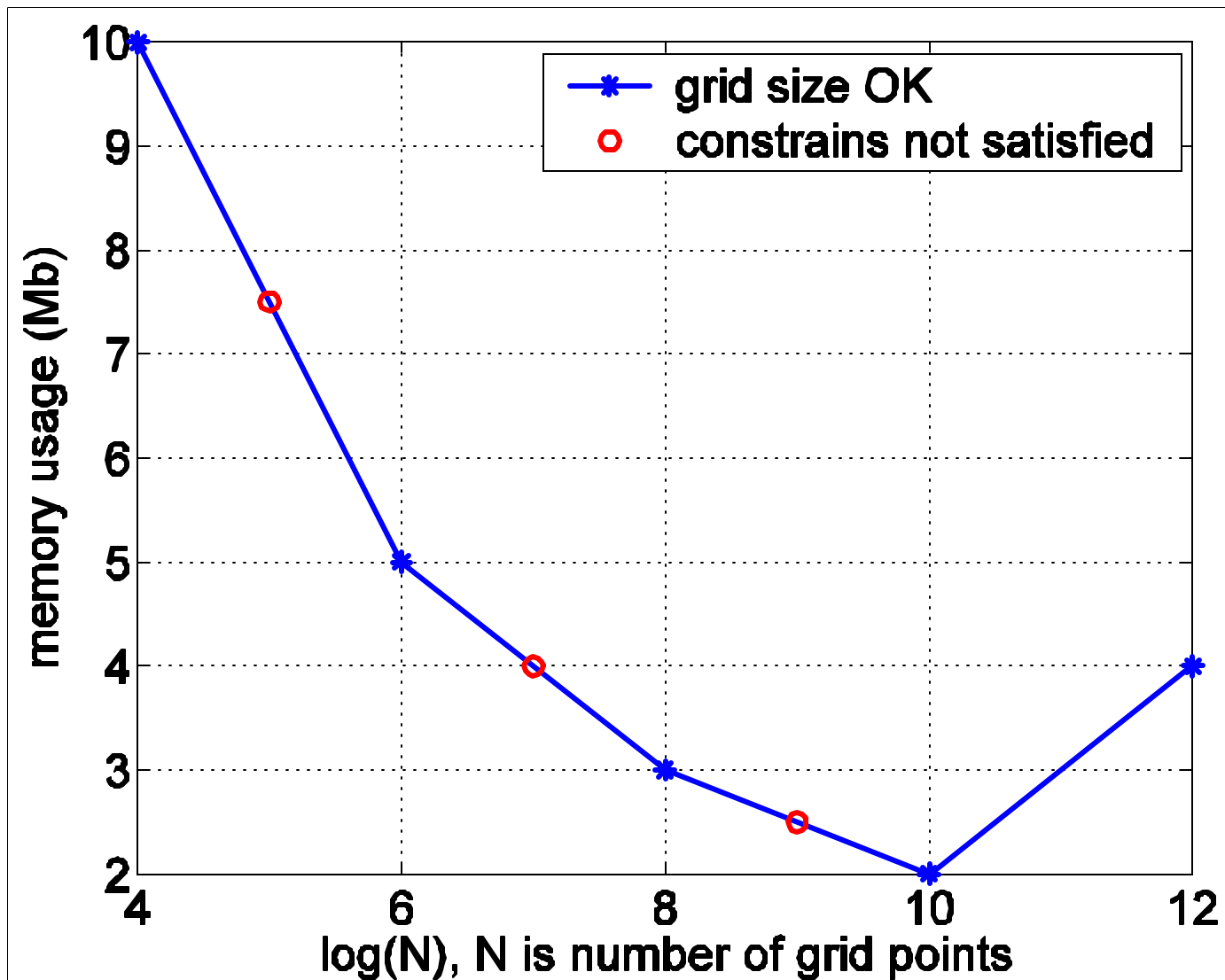
- **Ultimate test is the memory usage**
- **Constraints are used to weed out a grid option cheaply**
- **To gauge memory usage, grid and element map is to be set up.**
- **Density and size of $[I]$, $[P]$, $[D]$ and $[H]$ can be easily derived from grid element map. Hence memory estimation itself is cheap.**
- **There is not guarantee that a legitimate grid can be found, particularly for long and thin panels.**

Grid selection: starting point

Need extra layers to ensure that the interpolation and the projection stencil are cube



Grid selection: typical search pattern



Implementation: Source codes

- **See `gridElement.cc` for how to find the nearby neighbors**
- **See `directMat.h` and `g2gUnion.h` for pre-correction details**
- **See `grid.cc` for grid selection**

Next Lecture

- **High-order element in pfft**
- **Application of pfft++ in computational aerodynamics**