FMM CMSC 878R/AMSC 698R

Lecture 2

Complexity

- The most common complexities are
  - O(1) - not proportional to any variable number, i.e. a fixed/constant amount of time
  - O(N) - proportional to the size of N (this includes a loop to N and loops to constant multiples of N such as 0.5N, 2N, 2000N - no matter what that is, if you double N you expect (on average) the program to take twice as long)
  - O(N^2) - proportional to N squared (you double N, you expect it to take four times longer - usually two nested loops both dependent on N).
  - O(log N) - this is tricker to show - usually the result of binary splitting.
  - O(N log N) this is usually caused by doing log N splits but also doing N amount of work at each "layer" of splitting.

Theta

Same Order of Growth:

- $f(n) = \Theta(g(n))$

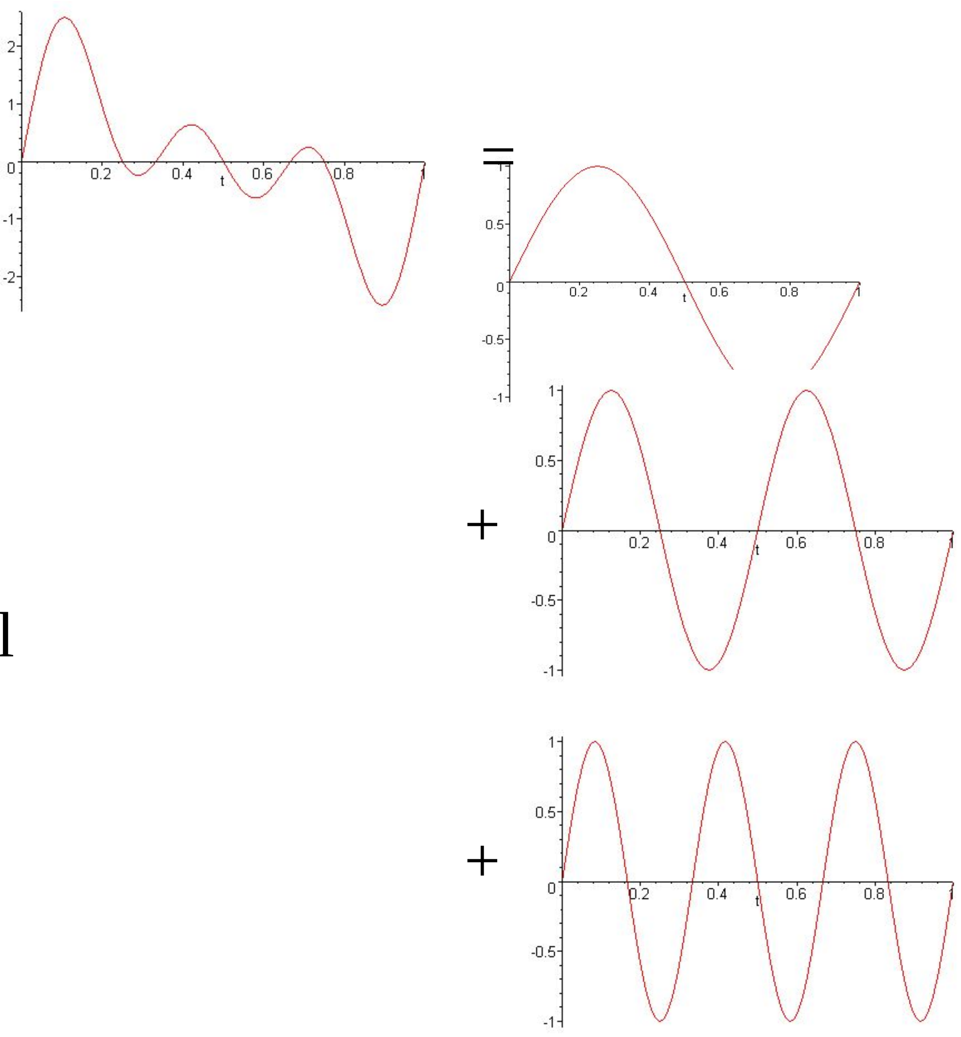
$f(n) = O(g(n))$ and $g(n) = O(f(n))$

Log complexity

- If you half data at each stage then number of stages until you have a single item is given (roughly) by $\log_2$ N. => binary search takes $\log_2$ N time to find an item.
- All logs grow a constant amount apart (homework)
  - So we normally just say log N not $\log_2$ N.
- Log N grows very slowly

Fourier Analysis

- Def.: mathematical techniques for breaking u a signal into its components (sinusoids)
- Jean Baptiste Joseph Fourier (1768-1830)
- can represent any continuous periodic signal as a sum of sinusoidal waves



Applications

- Digital Signal Processing: analyzing and manipulating real-world signals using a computer
  - Toys and consumer electronics
  - Speech recognition
  - Audio/video compression
  - Medical imaging
  - Communications
  - Radar

Fourier Transform

- Fourier transform of a function *h(t)* is given by *H(f)* where *f* is the frequency

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$
$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df$$

- Discrete Fourier Transform: if the function is sampled at discrete

$$h_k \equiv h(t_k), \qquad t_k \equiv k\Delta, \qquad k = 0, 1, 2, \ldots, N-1$$

$$H(f_n) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f_n t} dt \approx \sum_{k=0}^{N-1} h_k \, e^{2\pi i f_n t_k} \Delta = \Delta \sum_{k=0}^{N-1} h_k \, e^{2\pi i k n / N}$$

Discrete Fourier Transform

- All notion of time has disappeared
- Multiplication of sampled data by a matrix

$$H_n \equiv \sum_{k=0}^{N-1} h_k \, e^{2\pi i k n/N}$$

- This matrix is called the Fourier Matrix
- As discussed earlier it is a structured matrix

A Fourier matrix of order n is defined as the following

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{n-1} \\ 1 & W^2 & W^4 & \cdots & W^{2(n-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & W^{n-1} & W^{2(n-1)} & \cdots & W^{(n-1)(n-1)} \end{bmatrix},$$

where

$$W = e^{\frac{2\pi i}{n}},$$

is an nth root of unity.

Fourier Transform Algorithm

```
For k=0 to N/2 {
  For i=0 to N-1 {
    Real_X[k] += x[i] * cos(2π * k * i / N)
    Imag_X[k] -= x[i] * sin(2π * k * i / N)
  }
}
```

Fast Fourier Transform

- Presented by Cooley and Tukey in 1965, but invented several times, including by Gauss (1809) and Danielson & Lanczos (1948)
- Danielson Lanczos lemma

$$\begin{aligned}
F_k &= \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j \\
&= \sum_{j=0}^{N/2-1} e^{2\pi ik(2j)/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi ik(2j+1)/N} f_{2j+1} \\
&= \sum_{j=0}^{N/2-1} e^{2\pi ikj/(N/2)} f_{2j} + W^k \sum_{j=0}^{N/2-1} e^{2\pi ikj/(N/2)} f_{2j+1} \\
&= F_k^e + W^k\ F_k^o
\end{aligned}$$

FFT

- So DFT of order *N* can be expressed as sum of two DFTs of order *N/2*
- Does this improve the complexity?
- Yes $(N/2)^2+(N/2)^2 = N^2/2 < N^2$

$$F_k^e = F_k^{ee} + W^k F_k^{eo}, \quad F_k^o = F_k^{oe} + W^k F_k^{oo},$$

- But we are not done ….
- Can apply the lemma recursively

$$F_k^{eoeeoeo\cdots oee} = f_n$$

- Finally we have a set of one point transforms

Fast Fourier Transform Algorithm

- J.W. Cooley and J.W. Tukey, 1965
- Karl Friedrich Gauss (1777-1855)
- 1: Break N-point signal into N 1-point signals
- 2: Calculate N frequency spectra
- 3: Combine the spectra into one spectrum

Illustration

- Step 1: 16-point signal ➔ 16 1-point signals

[x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}]

[x_0, x_2, x_4, x_6, x_8, x_{10}, x_{12}, x_{14}][x_1, x_3, x_5, x_7, x_9, x_{11}, x_{13}, x_{15}]

[x_0, x_4, x_8, x_{12}][x_2, x_6, x_{10}, x_{14}][x_1, x_5, x_9, x_{13}][x_3, x_7, x_{11}, x_{15}]

[x_0, x_8][x_4, x_{12}][x_2, x_{10}][x_6, x_{14}][x_1, x_9][x_5, x_{13}][x_3, x_{11}][x_7, x_{15}]

[x_0][x_8][x_4][x_{12}][x_2][x_{10}][x_6][x_{14}][x_1][x_9][x_5][x_{13}][x_3][x_{11}][x_7][x_{15}]

- Step 2: Spectrum of 1-point signal = signal
- Step 3: Combine spectra in bottom-up fashion

Comparison

- Discrete Fourier Transform
  - 2 nested loops, N points each
  - $T_{DFT}(N) = \Theta(N^2)$. Execution time = $k_{DFT}N^2$.
- Fast Fourier Transform
  - Lg N stages, N/2 butterfly computations each
  - $T_{FFT}(N) = \Theta(N \lg N)$. Execution time = $k_{FFT}N \lg N$.
- E.g. 1024-point FT on 100MHz Pentium
  - DFT: k_{DFT}=25 microseconds; E.T.=25 seconds.
  - FFT: k_{FFT}=10 microseconds; E.T.=70 milliseconds

FFT

- So DFT of order *N* can be expressed as sum of two DFTs of order *N/2*
- Does this improve the complexity?
- Yes $(N/2)^2+(N/2)^2 = N^2/2 < N^2$

$$F_k^e = F_k^{ee} + W^k F_k^{eo}, \quad F_k^o = F_k^{oe} + W^k F_k^{oo},$$

- But we are not done ….
- Can apply the lemma recursively

$$F_k^{eoeeoeo\cdots oee} = f_n$$

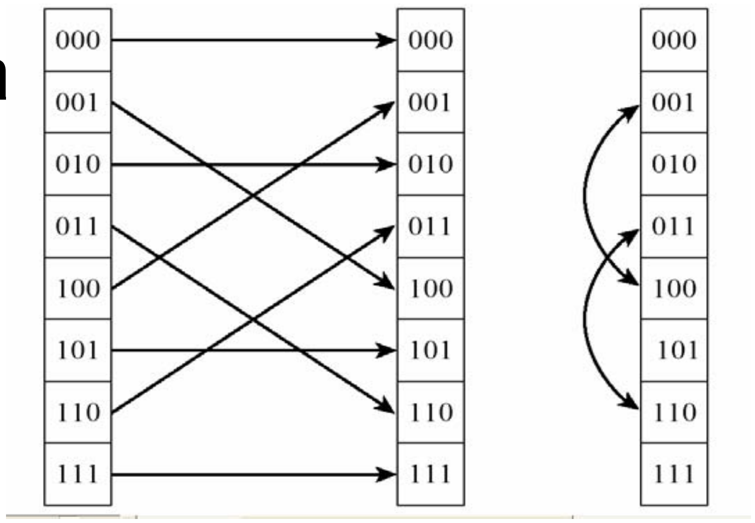
- Finally we have a set of one point transforms

Complexity

- Each F_k is a sum of $\log_2$ N transforms and (factors)
- There are N F_k s
- So the algorithm is $O(N \log_2 N)$

FFT and bit-shifts

- Set *o* to 1 and *e* to 0
- Then the sequence *ooeoeo…* can be interpereted as a binary number
- Reversing the pattern the binary value of *n*



Conclusion

- FFT is a divide-and-conquer algorithm
  - Divide: bit reversal sort
  - Conquer: calculate frequency spectra
  - Combine: recombine frequency spectra using butterfly computation
- N lg N time complexity
- Also implemented in hardware
- Makes many DSP applications practical

Outline

- Factorization – One of key parts of the FMM.
  - Extensions of our trick for fast summation
- Fields (Potentials)
  - Singular and Regular Fields
  - Far Field and Near Field
- Local Expansions (R-expansions)
  - Local Expansions of Regular and Singular Potentials
  - Power Series
  - Taylor Series

Matrix-Vector Multiplication

Compute matrix vector product

$$\mathbf{v} = \mathbf{\Phi u}$$

or

$$v_j = \sum_{i=1}^{N} \Phi_{ji} u_i, \quad j = 1,\ldots,M,$$

where

$$\Phi_{ji} = \Phi(\mathbf{y}_j, \mathbf{x}_i), \quad j = 1,\ldots,M, \quad i = 1,\ldots,N,$$

or

$$\mathbf{\Phi} = \begin{pmatrix} \Phi_{11} & \Phi_{12} & \ldots & \Phi_{1N} \\ \Phi_{21} & \Phi_{22} & \ldots & \Phi_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ \Phi_{M1} & \Phi_{M2} & \ldots & \Phi_{MN} \end{pmatrix} = \begin{pmatrix} \Phi(\mathbf{y}_1,\mathbf{x}_1) & \Phi(\mathbf{y}_1,\mathbf{x}_2) & \ldots & \Phi(\mathbf{y}_1,\mathbf{x}_N) \\ \Phi(\mathbf{y}_2,\mathbf{x}_1) & \Phi(\mathbf{y}_2,\mathbf{x}_2) & \ldots & \Phi(\mathbf{y}_2,\mathbf{x}_N) \\ \ldots & \ldots & \ldots & \ldots \\ \Phi(\mathbf{y}_M,\mathbf{x}_1) & \Phi(\mathbf{y}_M,\mathbf{x}_2) & \ldots & \Phi(\mathbf{y}_M,\mathbf{x}_N) \end{pmatrix}.$$

Generally we have two sets of points in d-dimensions:

$$\text{Sources}: \mathbb{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}, \quad \mathbf{x}_i \in \mathbb{R}^d, \quad i = 1,\ldots,N,$$
$$\text{Receivers}: \mathbb{Y} = \{\mathbf{y}_1,\ldots,\mathbf{y}_M\}, \quad \mathbf{y}_j \in \mathbb{R}^d, \quad j = 1,\ldots,M,$$

The receivers also can be called “targets” or “evaluation points”.

Why $\mathbf{R}^d$?

- d = 1
  - Scalar functions, interpolation, etc.
- d = 2,3
  - Physical problems in 2 and 3 dimensional space
- d = 4
  - 3D Space + time, 3D grayscale images
- d = 5
  - Color 2D images, Motion of 3D grayscale images
- d = 6
  - Color 3D images
- d = 7
  - Motion of 3D color images
- d = arbitrary
  - d-parametric spaces, statistics, database search procedures

Fields (Potentials)

Field (Potential) of a single (*i*th) unit source

$$v(\mathbf{y}) = \sum_{i=1}^{N} u_i \Phi(\mathbf{y}, \mathbf{x}_i), \quad \mathbf{y} \in \mathbb{R}^d,$$

$$v_j = v(\mathbf{y}_j), \quad j = 1, \ldots, M.$$

Field (Potential) of the set of sources of intensities $\{u_i\}$

Fields are continuous!
(Almost everywhere)

Examples of Fields

- There can be vector or scalar fields (we focus mostly on scalar fields)
- Fields can be *regular* or *singular*

Scalar Fields:

● Gravity

(singular at $\mathbf{y} = \mathbf{x}_i$)

$$\Phi(\mathbf{y}, \mathbf{x}_i) = \frac{1}{|\mathbf{y} - \mathbf{x}_i|}$$

● Monochromatic Wave (k is the wavenumber)

(singular at $\mathbf{y} = \mathbf{x}_i$)

$$\Phi(\mathbf{y}, \mathbf{x}_i) = \frac{\exp\{ik|\mathbf{y} - \mathbf{x}_i|\}}{|\mathbf{y} - \mathbf{x}_i|}$$

● Gaussian

(regular everywhere)

$$\Phi(\mathbf{y}, \mathbf{x}_i) = \exp\{-|\mathbf{y} - \mathbf{x}_i|^2/\sigma\}$$

Vector Field:

● 3D Velocity field:

(singular at $\mathbf{y} = \mathbf{x}_i$)

$$\Phi(\mathbf{y}, \mathbf{x}_i) = \nabla_y \frac{1}{|\mathbf{y} - \mathbf{x}_i|} = \mathbf{i}_1 \frac{\partial}{\partial y_1} \frac{1}{|\mathbf{y} - \mathbf{x}_i|} + \mathbf{i}_2 \frac{\partial}{\partial y_2} \frac{1}{|\mathbf{y} - \mathbf{x}_i|} + \mathbf{i}_3 \frac{\partial}{\partial y_3} \frac{1}{|\mathbf{y} - \mathbf{x}_i|},$$

$$\mathbf{y} = (y_1, y_2, y_3) \in \mathbb{R}^3.$$

Straightforward Computational Complexity:

$O(MN)$ Error: 0 (“machine” precision)

The Fast Multipole Methods look for computation of the same problem with complexity $o(MN)$ and error < prescribed error.

In the case when the error of the FMM does not exceed the machine precision error (for given number of bits) there is no difference between the “exact” and “approximate” solution.

Factorization
“Middleman Method”

Global Factorization

Expansion center

Truncation number

$\forall \mathbf{x}_i, \mathbf{y}_j \in \Omega \subset \mathbb{R}^d$:

$$\Phi\left(\mathbf{y}_j, \mathbf{x}_i\right) = \sum_{m=0}^{\infty} a_m(\mathbf{x}_i - \mathbf{x}_*) f_m\left(\mathbf{y}_j - \mathbf{x}_*\right) = \sum_{m=0}^{p-1} a_m(\mathbf{x}_i - \mathbf{x}_*) f_m\left(\mathbf{y}_j - \mathbf{x}_*\right) + Error(p, \mathbf{x}_i, \mathbf{y}_j)$$

Expansion coefficients

Basis functions

Factorization Trick

$$\begin{aligned}
v_j &= \sum_{i=1}^{N} \Phi\left(\mathbf{y}_j, \mathbf{x}_i\right) u_i \\
&= \sum_{i=1}^{N} \left[\sum_{m=0}^{p-1} a_m(\mathbf{x}_i - \mathbf{x}_*) f_m\left(\mathbf{y}_j - \mathbf{x}_*\right) + Error(p; \mathbf{x}_i, \mathbf{y}_j) \right] u_i \\
&= \sum_{m=0}^{p-1} f_m\left(\mathbf{y}_j - \mathbf{x}_*\right) \sum_{i=1}^{N} a_m(\mathbf{x}_i - \mathbf{x}_*) u_i + \sum_{i=1}^{N} Error(p; \mathbf{x}_i, \mathbf{y}_j) u_i \\
&= \sum_{m=0}^{p-1} c_m f_m\left(\mathbf{y}_j - \mathbf{x}_*\right) + Error(N, p),
\end{aligned}$$

where

$$c_m = \sum_{i=1}^{N} a_m(\mathbf{x}_i - \mathbf{x}_*) u_i.$$

Reduction of Complexity

Straightforward (nested loops):

```
for $j = 1,\dots,M$
    $v_j = 0;$
    for $i = 1,\dots,N$
        $v_j = v_j + \Phi\left(\mathbf{y}_j, \mathbf{x}_i\right) u_i;$
    end;
end;
```

Complexity: *O*(*MN*)

Factroized:

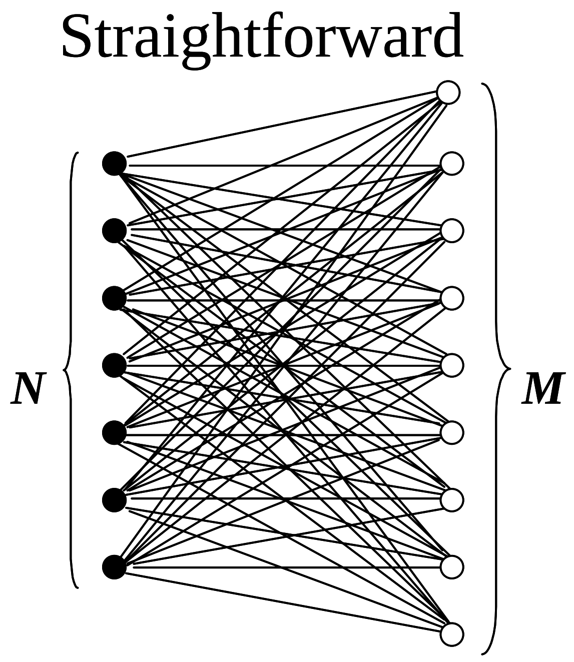
```
for $m = 0,\dots,p-1$
    $c_m = 0;$
    for $i = 1,\dots,N$
        $c_m = c_m + a_m(\mathbf{x}_i - \mathbf{x}_*) u_i;$
    end;
end;
```

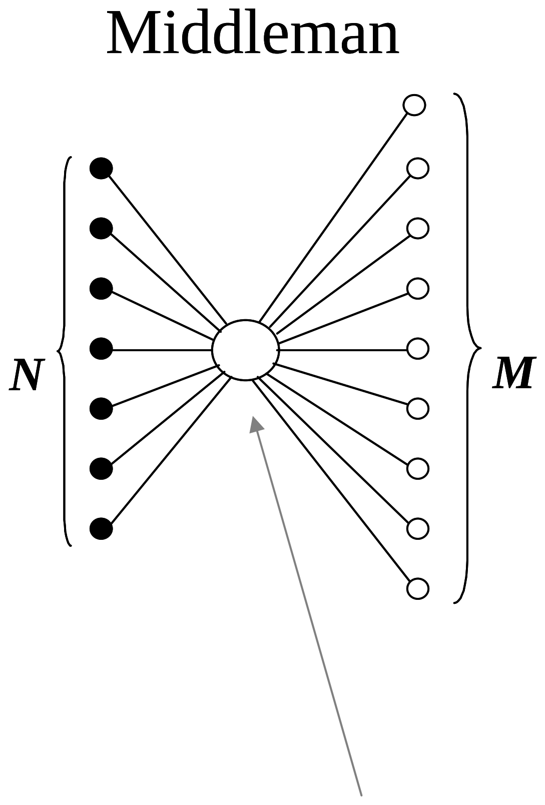
```
for $j = 1,\dots,M$
    $v_j = 0;$
    for $m = 0,\dots,p-1$
        $v_j = v_j + c_m f_m\left(\mathbf{y}_j - \mathbf{x}_*\right);$
    end;
end;
```

Complexity: *O*(*pN*+*pM*)

If *p* << min*(M,N)* then complexity reduces!

Middleman Scheme





Complexity: *O*(*pN*+*pM*)

Set of coefficients $\{c_m\}$

Far Field and Near Field

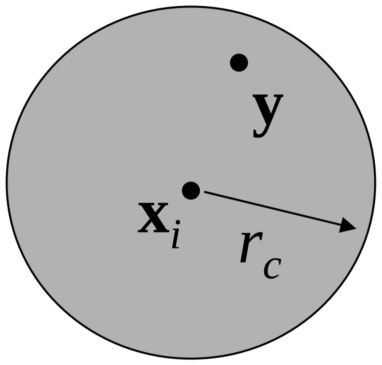
- Near Field of the *i*th source:

$$\boxed{|\mathbf{y} - \mathbf{x}_i| < r_c.}$$

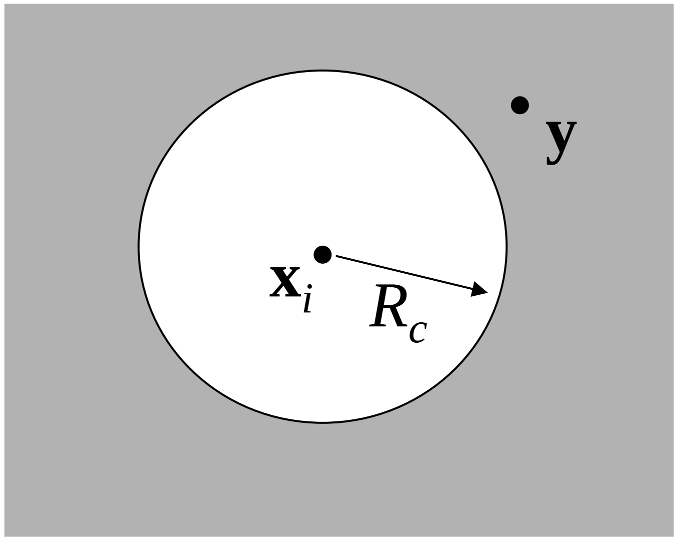
- Far Field of the *i*th source:

$$\boxed{|\mathbf{y} - \mathbf{x}_i| > R_c.}$$

Near Field



Far Field



What are these r_c and R_c ?
depends on the potential + some conventions for the terminology

Local (Regular) Expansion

Do not confuse with the Near Field!

Let $\boxed{\mathbf{x}_* \in \mathbb{R}^d.}$

Basis Functions

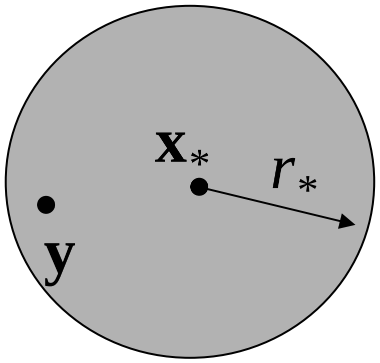
We call expansion

$$\boxed{\Phi(\mathbf{y},\mathbf{x}_i) = \sum_{m=0}^{\infty} a_m(\mathbf{x}_i,\mathbf{x}_*) R_m(\mathbf{y}-\mathbf{x}_*)}$$

local (regular) inside a sphere $\boxed{|\mathbf{y}-\mathbf{x}_*| < r_*,}$

Expansion Coefficients

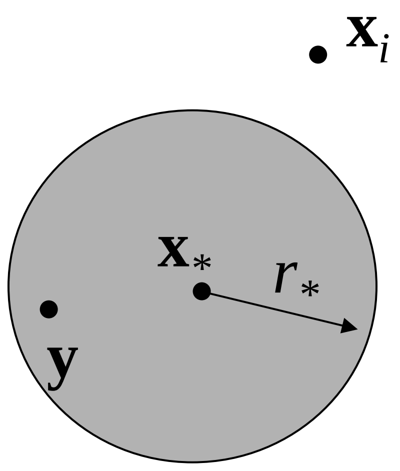
if the series converges for $\forall \mathbf{y},\ |\mathbf{y}-\mathbf{x}_*| < r_*.$



We also call this R-expansion,
since basis functions R_m should be *regular*

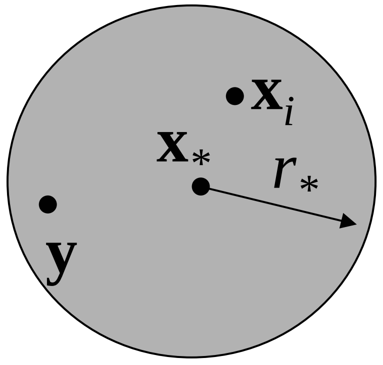
Local Expansion of a Regular Potential

Can be like this:



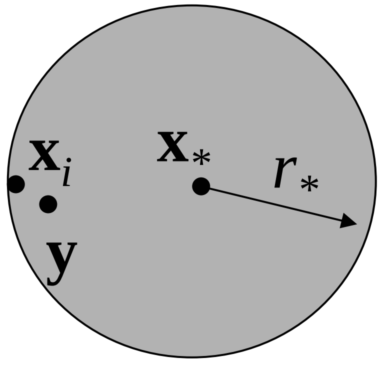
$|\mathbf{y} - \mathbf{x}_*| < r_* < |\mathbf{x}_i - \mathbf{x}_*|$

…or like this:



$r_* > |\mathbf{y} - \mathbf{x}_*| > |\mathbf{x}_i - \mathbf{x}_*|$

…or like this:



$r_* > |\mathbf{x}_i - \mathbf{x}_*| > |\mathbf{y} - \mathbf{x}_*|$

Local Expansion of a Regular Potential (Example)

Valid for any $r_* < \infty$, and x_i.

$$\boxed{x, y \in \mathbb{R}^1.}$$

$$\boxed{\Phi(y, x_i) = e^{-(y-x_i)^2}.}$$

Looking for factorization:

$$\boxed{\Phi(y, x_i) = \sum_{m=0}^{\infty} a_m(x_i - x_*) R_m(y - x_*).}$$

We have

$$e^{-(y-x_i)^2} = e^{-[y-x_*-(x_i-x_*)]^2} = e^{-(y-x_*)^2} e^{-(x_i-x_*)^2} e^{2(x_i-x_*)(y-x_*)}$$

$$= e^{-(y-x_*)^2} e^{-(x_i-x_*)^2} \sum_{m=0}^{\infty} \frac{2^m (x_i - x_*)^m (y - x_*)^m}{m!}.$$

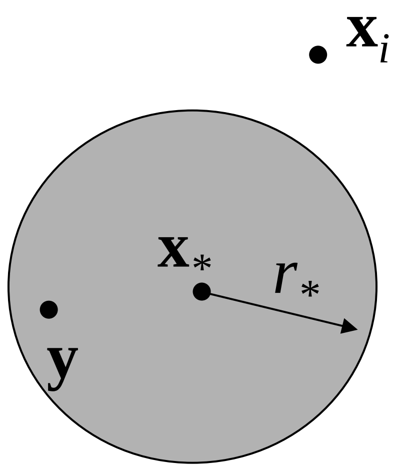
Choose

$$a_m(x_i - x_*) = e^{-(x_i-x_*)^2} \sqrt{\frac{2^m}{m!}} (x_i - x_*)^m, \quad m = 0, 1, \ldots,$$

$$R_m(y - x_*) = e^{-(y-x_*)^2} \sqrt{\frac{2^m}{m!}} (y - x_*)^m, \quad m = 0, 1, \ldots$$

Local Expansion of a Singular Potential

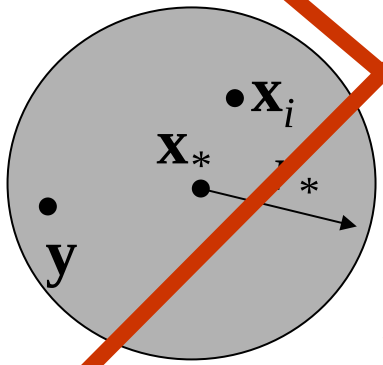
Can be like this:



$|\mathbf{y} - \mathbf{x}_*| < r_* \le |\mathbf{x}_i - \mathbf{x}_*|$

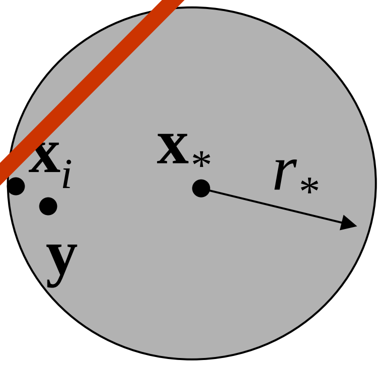
Like this only!

~~…or like this:~~



~~$r_* > |\mathbf{y} - \mathbf{x}_*| > |\mathbf{x}_i - \mathbf{x}_*|$~~

~~…or like this:~~



~~$r_* > |\mathbf{x}_i - \mathbf{x}_*| > |\mathbf{y} - \mathbf{x}_*|$~~

Never ever!

Because $\mathbf{x}_i$ is a singular point!

Local Expansion of a Singular Potential (Example)

Valid for any $|x_{i.}\text{-}x_*| > |y\text{-}x_*|$

$$\boxed{x, y \in \mathbb{R}^1.}$$

$$\boxed{\Phi(y, x_i) = \frac{1}{y - x_i}.}$$

Looking for factorization:

$$\boxed{\Phi(y, x_i) = \sum_{m=0}^{\infty} a_m(x_i - x_*) R_m(y - x_*).}$$

We have

$$\boxed{\frac{1}{y - x_i} = \frac{1}{y - x_* - (x_i - x_*)} = -\frac{1}{(x_i - x_*)[1 - \frac{y - x_*}{x_i - x_*}]} = -\frac{1}{(x_i - x_*)}\left[1 - \frac{y - x_*}{x_i - x_*}\right]^{-1}.}$$

Geometric progression:

$$\boxed{(1 - \alpha)^{-1} = 1 + \alpha + \alpha^2 + \ldots = \sum_{m=0}^{\infty} \alpha^m, \quad |\alpha| < 1.}$$

$$\boxed{\left[1 - \frac{y - x_*}{x_i - x_*}\right]^{-1} = \sum_{m=0}^{\infty} \frac{(y - x_*)^m}{(x_i - x_*)^m}, \quad |y - x_*| < |x_i - x_*|.}$$

Choose

$$\boxed{a_m(x_i - x_*) = -\frac{1}{(x_i - x_*)^{m+1}}, \quad m = 0, 1, \ldots,}$$

$$\boxed{R_m(y - x_*) = (y - x_*)^m, \quad m = 0, 1, \ldots}$$