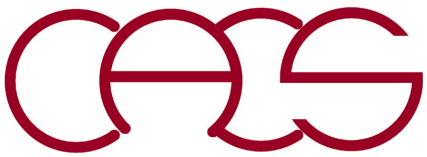
Quantum Dynamics

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California

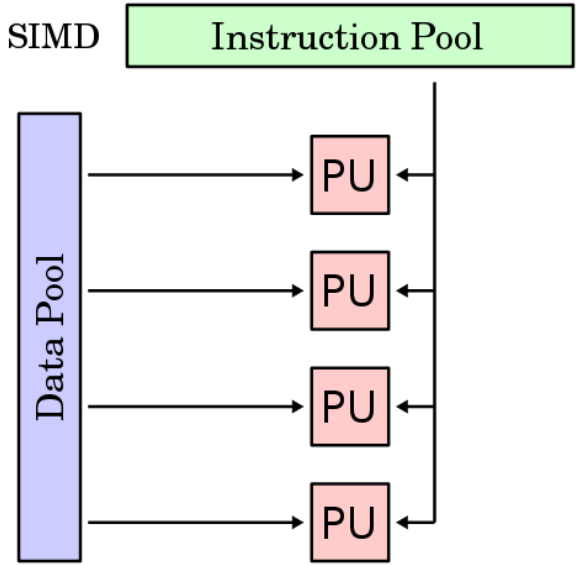
Email: anakano@usc.edu

Objective: Learn prototypical continuum simulation



What to Learn with QD?

- **An archetype of single-instruction multiple-data (SIMD) parallel applications: Continuum simulations based on partial differential equation (PDE)**
- **Alternative parallelization methods with distinct scalability**
- **Performance optimization on emerging accelerators**



SIMD: A single instruction operates on multiple different data streams

Complementary to molecular dynamics (MD) = multiple-instruction multiple-data (MIMD), discrete, ordinary differential equation (ODE)

Wave Equation

- **Complex wave function**

$$\psi(x,t) = \mathrm{Re}\psi(x,t) + i\,\mathrm{Im}\psi(x,t) \in \mathbf{C} \quad \left(i = \sqrt{-1}\right)$$

- **Normalization**

$$\int dx \left|\psi(x,t)\right|^2 = 1$$

- **Schrödinger equation (in atomic unit)**

$$i\frac{\partial}{\partial t}\psi(x,t) = H\psi(x,t)$$

- **Hamiltonian operator**

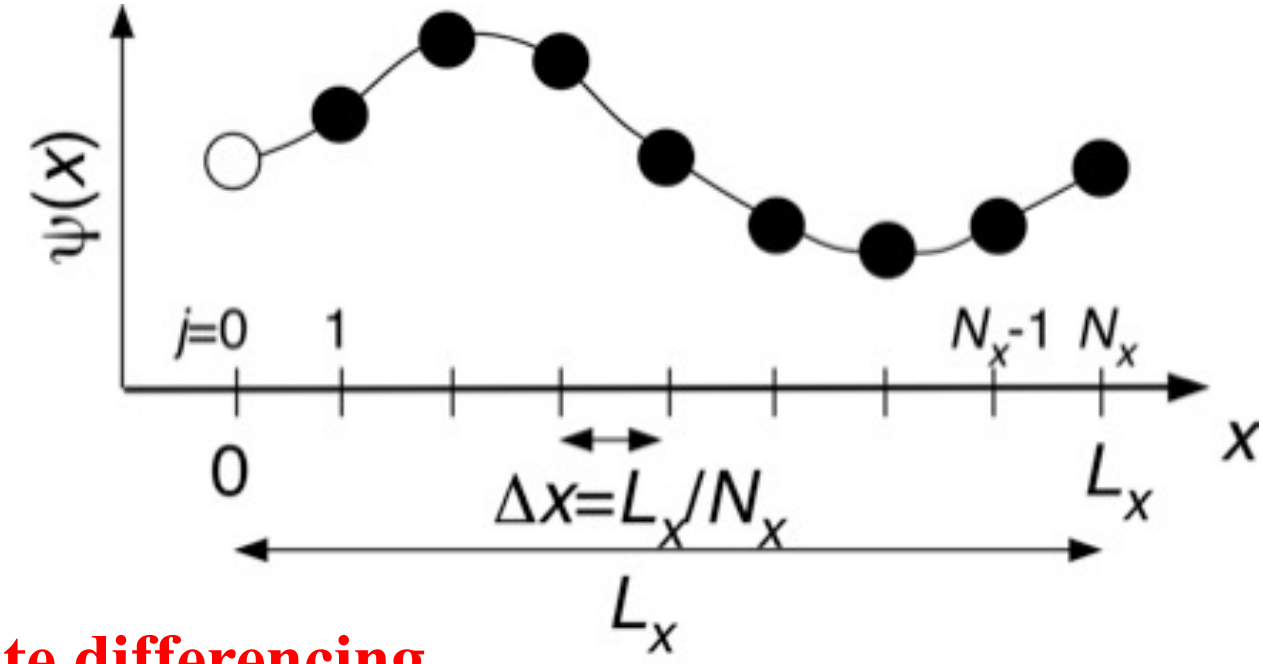
$$H = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + V(x) = T_x + V$$

- **Periodic boundary condition**

$$\psi(x + L_x) = \psi(x)$$

Spatial Discretization

- **Regular 1D mesh:** $\psi_j = \psi(j\Delta x)$ $(\Delta x = L_x/N_x)$



- **Finite differencing**

$$\begin{cases} (T_x\psi)_j = -\dfrac{1}{2}\dfrac{\psi_{j-1} - 2\psi_j + \psi_{j+1}}{(\Delta x)^2} \\ (V\psi)_j = V_j\psi_j \end{cases}$$

$$\frac{\partial^2}{\partial x^2}\psi(x,t)$$
$$= \frac{\partial\psi(x+\Delta/2,t)/\partial x - \partial\psi(x-\Delta/2,t)/\partial x}{\Delta}$$
$$= \frac{\frac{\psi(x+\Delta,t)-\psi(x,t)}{\Delta} - \frac{\psi(x,t)-\psi(x-\Delta,t)}{\Delta}}{\Delta}$$
$$= \frac{\psi(x+\Delta,t)-2\psi(x,t)+\psi(x-\Delta,t)}{\Delta^2}$$

Temporal Propagation

- **Formal solution to the Schrödinger equation:** $\frac{\partial}{\partial t}\psi(t) = -iH\psi(t)$

$$\psi(t+\Delta t) = \exp(-iH\Delta t)\psi(t)$$

$$\exp(-\mathrm{i}\hat{H}t) = \sum_{n=0}^{\infty}\frac{(-it)^n}{n!}\hat{H}^n$$

- **Split-operator method: unitary!**

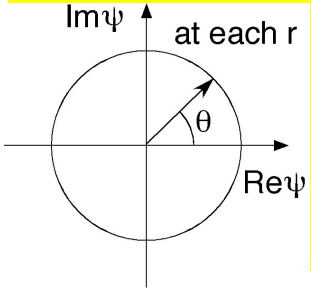
$$\begin{aligned}\psi(t+\Delta t) &= \exp(-i(T_x+V)\Delta t)\psi(t)\\ &= \exp(-iV\Delta t/2)\exp(-iT_x\Delta t)\exp(-iV\Delta t/2)\psi(t) + O([\Delta t]^3)\end{aligned}$$

Split in a way each operator is easily exponentiated

- **Potential propagator (mesh point-by-point complex-number multiplications)**

$$(u_0+iu_1)(\psi_0+i\psi_1) = (u_0\psi_0 - u_1\psi_1) + i(u_1\psi_0 + u_0\psi_1)$$

$$\begin{aligned}(\exp(-iV\Delta t/2)\psi)_j &= \exp(-iV_j\Delta t/2)\psi_j\\ &= [\cos(V_j\Delta t/2) - i\sin(V_j\Delta t/2)][\mathrm{Re}\psi_j + i\,\mathrm{Im}\psi_j]\\ &= [\cos(V_j\Delta t/2)\mathrm{Re}\psi_j + \sin(V_j\Delta t/2)\mathrm{Im}\psi_j]\\ &\quad + i[-\sin(V_j\Delta t/2)\mathrm{Re}\psi_j + \cos(V_j\Delta t/2)\mathrm{Im}\psi_j]\end{aligned}$$



Rotation

$$\exp(ia) = \sum_{n=0}^{\infty}\frac{(ia)^n}{n!} = \left(1 - \frac{a^2}{2!} + \frac{a^4}{4!} - \cdots\right) + i\left(a - \frac{a^3}{3!} + \frac{a^5}{5!} - \cdots\right)$$
$$= \cos(a) + i\sin(a)$$

$$\begin{bmatrix}\cos(V_{jk}\Delta t/2) & \sin(V_{jk}\Delta t/2)\\ -\sin(V_{jk}\Delta t/2) & \cos(V_{jk}\Delta t/2)\end{bmatrix}\begin{bmatrix}\mathrm{Re}\psi_{jk}\\ \mathrm{Im}\psi_{jk}\end{bmatrix}$$

Kinetic Propagator: It's a Matrix!

- **Mesh-point coupling**

$$T_x\psi_j = b\psi_{j-1} + 2a\psi_j + b\psi_{j+1}$$

- **Tridiagonal matrix representation**

$$T_x = \begin{bmatrix} 2a & b & & & & & b \\ b & 2a & b & & & & \\ & b & 2a & b & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & b & 2a & b & \\ & & & & b & 2a & b \\ b & & & & & b & 2a \end{bmatrix}$$

Note the periodic boundary condition

$$\begin{cases} a = 1/2(\Delta x)^2 \\ b = -1/2(\Delta x)^2 \end{cases}$$

Space Splitting Method (SSM)

- **2×2 block-diagonal decomposition & split-operator exponentiation**

$$T_x = \begin{bmatrix} 2a & b & & & & & b \\ b & 2a & b & & & & \\ & b & 2a & b & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & b & 2a & b & \\ & & & & b & 2a & b \\ b & & & & & b & 2a \end{bmatrix}$$

Block-by-block exponentiation

$$\exp\begin{bmatrix} \blacksquare & & \\ & \blacksquare & \\ & & \blacksquare \end{bmatrix} = \sum_{n=0}^{\infty} \frac{1}{n!} \begin{bmatrix} \blacksquare & & \\ & \blacksquare & \\ & & \blacksquare \end{bmatrix}^n$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} \begin{bmatrix} \blacksquare^n & & \\ & \blacksquare^n & \\ & & \blacksquare^n \end{bmatrix} = \begin{bmatrix} e^{\blacksquare} & & \\ & e^{\blacksquare} & \\ & & e^{\blacksquare} \end{bmatrix}$$

Split-operator (Trotter expansion) again

$$= \frac{1}{2} \begin{bmatrix} a & b & & & & & \\ b & a & & & & & \\ & & a & b & & & \\ & & b & a & & & \\ & & & & \ddots & & \\ & & & & & a & b \\ & & & & & b & a \end{bmatrix} + \begin{bmatrix} a & & & & & & b \\ & a & b & & & & \\ & b & a & & & & \\ & & & \ddots & & & \\ & & & & a & b & \\ & & & & b & a & \\ b & & & & & & a \end{bmatrix} + \frac{1}{2} \begin{bmatrix} a & b & & & & & \\ b & a & & & & & \\ & & a & b & & & \\ & & b & a & & & \\ & & & & \ddots & & \\ & & & & & a & b \\ & & & & & b & a \end{bmatrix}$$

$$\exp(-i\Delta t T_x) = U_x^{(\text{half})} U_x^{(\text{full})} U_x^{(\text{half})} + \boldsymbol{O}([\Delta t]^3)$$

$$= \exp\left(-\frac{i\Delta t}{2} \begin{bmatrix} a & b & & & & & \\ b & a & & & & & \\ & & a & b & & & \\ & & b & a & & & \\ & & & & \ddots & & \\ & & & & & a & b \\ & & & & & b & a \end{bmatrix} \right) \exp\left(-i\Delta t \begin{bmatrix} a & & & & & & b \\ & a & b & & & & \\ & b & a & & & & \\ & & & \ddots & & & \\ & & & & a & b & \\ & & & & b & a & \\ b & & & & & & a \end{bmatrix} \right) \exp\left(-\frac{i\Delta t}{2} \begin{bmatrix} a & b & & & & & \\ b & a & & & & & \\ & & a & b & & & \\ & & b & a & & & \\ & & & & \ddots & & \\ & & & & & a & b \\ & & & & & b & a \end{bmatrix} \right)$$

How? Block diagonal → block-by-block exponentiation

Space Splitting Method (SSM)

$$\exp(-i\Delta t T_x) = U_x^{(\text{half})} U_x^{(\text{full})} U_x^{(\text{half})} + O([\Delta t]^3)$$

$$= \exp\left(-\frac{i\Delta t}{2}\begin{bmatrix} a & b & & & & & \\ b & a & & & & & \\ & & a & b & & & \\ & & b & a & & & \\ & & & & \ddots & & \\ & & & & & a & b \\ & & & & & b & a \end{bmatrix}\right) \exp\left(-i\Delta t \begin{bmatrix} a & & & & & & b \\ & a & b & & & & \\ & b & a & & & & \\ & & & \ddots & & & \\ & & & & a & b & \\ & & & & b & a & \\ b & & & & & & a \end{bmatrix}\right) \exp\left(-\frac{i\Delta t}{2}\begin{bmatrix} a & b & & & & & \\ b & a & & & & & \\ & & a & b & & & \\ & & b & a & & & \\ & & & & \ddots & & \\ & & & & & a & b \\ & & & & & b & a \end{bmatrix}\right)$$

$$= \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- & & & & & \\ \varepsilon_2^- & \varepsilon_2^+ & & & & & \\ & & \varepsilon_2^+ & \varepsilon_2^- & & & \\ & & \varepsilon_2^- & \varepsilon_2^+ & & & \\ & & & & \ddots & & \\ & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix} \begin{bmatrix} \varepsilon_1^+ & & & & & & \varepsilon_1^- \\ & \varepsilon_1^+ & \varepsilon_1^- & & & & \\ & \varepsilon_1^- & \varepsilon_1^+ & & & & \\ & & & \ddots & & & \\ & & & & \varepsilon_1^+ & \varepsilon_1^- & \\ & & & & \varepsilon_1^- & \varepsilon_1^+ & \\ \varepsilon_1^- & & & & & & \varepsilon_1^+ \end{bmatrix} \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- & & & & & \\ \varepsilon_2^- & \varepsilon_2^+ & & & & & \\ & & \varepsilon_2^+ & \varepsilon_2^- & & & \\ & & \varepsilon_2^- & \varepsilon_2^+ & & & \\ & & & & \ddots & & \\ & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix}$$

$$\begin{cases} \varepsilon_n^+ = \frac{1}{2}\left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) + \exp\left(-\frac{i\Delta t}{n}(a-b)\right)\right] \\ \varepsilon_n^- = \frac{1}{2}\left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) - \exp\left(-\frac{i\Delta t}{n}(a-b)\right)\right] \end{cases}$$

Just need 2×2 exponentiation

$$\exp\left(-\frac{i\Delta}{(2)}\begin{bmatrix} a & b \\ b & a \end{bmatrix}\right)$$

Use eigen-decomposition & telescoping

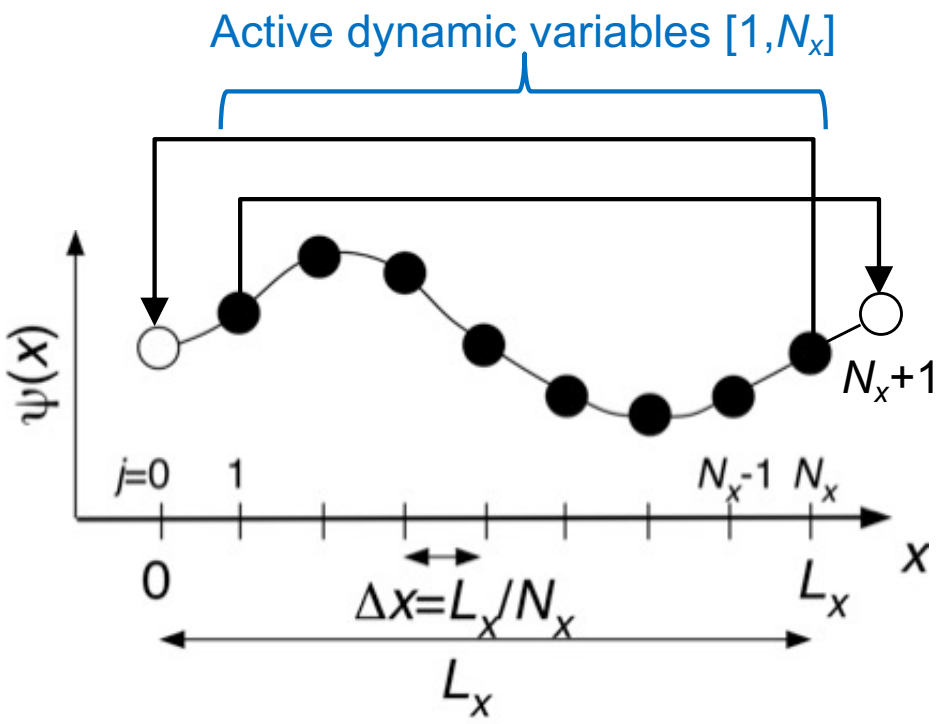
$$(UDU^{-1})^n = \overbrace{UDU^{-1}\,UDU^{-1}\cdots UDU^{-1}}^{n} = UD^nU^{-1}$$

$$D = \begin{bmatrix} \varepsilon_+ & 0 \\ 0 & \varepsilon_- \end{bmatrix} \qquad D^n = \begin{bmatrix} \varepsilon_+^n & 0 \\ 0 & \varepsilon_-^n \end{bmatrix}$$

Data Structures in Program qd1.c

- **Wave function: psi[NX+2][2]** $psi[j][0|1] = (\mathrm{Re}|\mathrm{Im})\psi_{j\Delta x}$
- **Periodic boundary condition by auxiliary elements**

```
for (s=0; s<=1; s++) {
  psi[0][s] = psi[NX][s];
  psi[NX+1][s] = psi[1][s];
}
```



https://aiichironakano.github.io/cs653/src/QD/

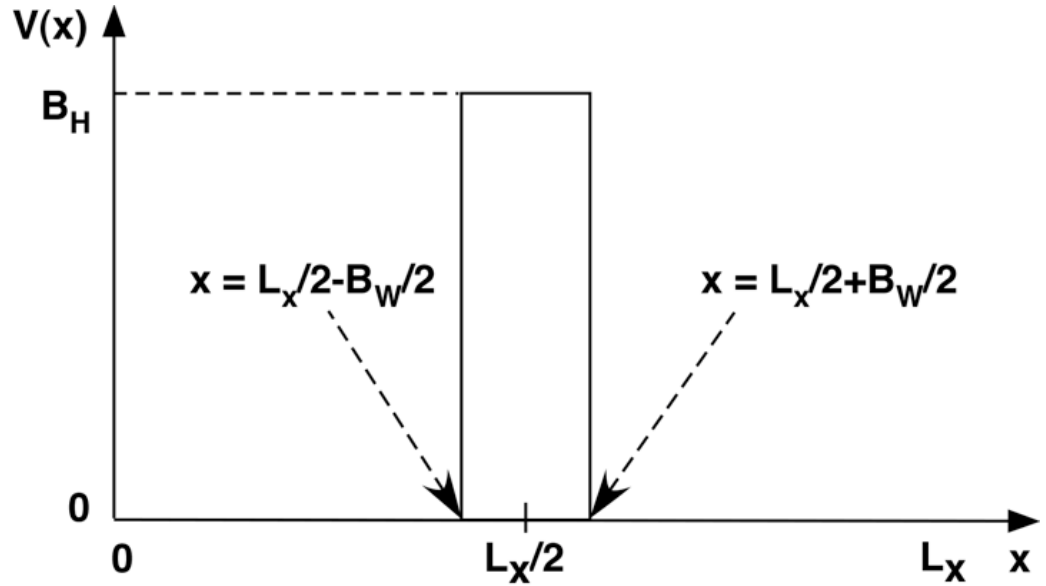
Potential Propagator in qd1.c

- **Potential barrier: v[NX+2]**
- **Potential propagator: exp(-*iV*Δ*t*/2), u[NX+2][2]**
- **Potential propagation: $\psi \leftarrow \exp(-iV\Delta t/2)\psi$**

```
for (sx=1; sx<=NX; sx++)
  wr=u[sx][0]*psi[sx][0]-u[sx][1]*psi[sx][1];
  wi=u[sx][0]*psi[sx][1]+u[sx][1]*psi[sx][0];
  psi[sx][0]=wr;
  psi[sx][1]=wi;
}
```

$$\begin{cases} u[j][0] = \cos\left(-\frac{\Delta}{2}V_j\right) \\ u[j][1] = \sin\left(-\frac{\Delta}{2}V_j\right) \end{cases}$$

$$\exp\left(-\frac{iV_j\Delta}{2}\right)\psi_j \equiv u\psi = (u_0 + iu_1)(\psi_0 + i\psi_1) = \overbrace{(u_0\psi_0 - u_1\psi_1)}^{\text{new } \psi_0} + i\overbrace{(u_0\psi_1 + u_1\psi_0)}^{\text{new } \psi_1}$$



Kinetic Propagator in qd1.c

$$\left(U_x^{(\mathrm{half})}\psi\right)_i = \varepsilon_2^- \delta_{\mathrm{mod}(i,2),0}\psi_{i-1} + \varepsilon_2^+\psi_i + \varepsilon_2^- \delta_{\mathrm{mod}(i,2),1}\psi_{i+1}$$

$\beta_{\mathrm{half}}^{(\mathrm{low})}$ α_{half} $\beta_{\mathrm{half}}^{(\mathrm{up})}$

$$\left(U_x^{(\mathrm{full})}\psi\right)_i = \varepsilon_1^- \delta_{\mathrm{mod}(i,2),1}\psi_{i-1} + \varepsilon_1^+\psi_i + \varepsilon_1^- \delta_{\mathrm{mod}(i,2),0}\psi_{i+1}$$

$\beta_{\mathrm{full}}^{(\mathrm{low})}$ α_{full} $\beta_{\mathrm{full}}^{(\mathrm{up})}$

```
for (sx=1; sx<=NX; sx++) {  // wrk[][]|psi[][] holds new|old wave function
  wr=al[t][0]*psi[sx][0]-al[t][1]*psi[sx][1];  // al[0|1][]: α half|full
  wi=al[t][0]*psi[sx][1]+al[t][1]*psi[sx][0];
  wr+=(bl[t][sx][0]*psi[sx-1][0]-bl[t][sx][1]*psi[sx-1][1]);  // bl[0|1][][]: β(low) half|full
  wi+=(bl[t][sx][0]*psi[sx-1][1]+bl[t][sx][1]*psi[sx-1][0]);
  wr+=(bu[t][sx][0]*psi[sx+1][0]-bu[t][sx][1]*psi[sx+1][1]);  // bu[0|1][][]: β(high) half|full
  wi+=(bu[t][sx][0]*psi[sx+1][1]+bu[t][sx][1]*psi[sx+1][0]);}
  wrk[sx][0]=wr;
  wrk[sx][1]=wi;}
```

$$\psi_j \leftarrow \beta_l\,\psi_{j-1} + \alpha\psi_j + \beta_u\,\psi_{j+1}$$

```
for (sx=1; sx<=NX; sx++)  // Copy new wave function back to psi
  for (s=0; s<=1; s++)
    psi[sx][s]=wrk[sx][s];
```

$$\begin{cases} \varepsilon_n^+ = \frac{1}{2}\left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) + \exp\left(-\frac{i\Delta t}{n}(a-b)\right)\right] \\ \varepsilon_n^- = \frac{1}{2}\left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) - \exp\left(-\frac{i\Delta t}{n}(a-b)\right)\right] \end{cases}$$

$$\exp(-i\Delta t T_x) = U_x^{(\mathrm{half})} U_x^{(\mathrm{full})} U_x^{(\mathrm{half})} + O([\Delta t]^3)$$

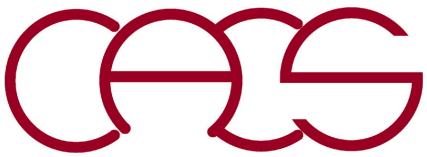
$$= \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- & & & & & \\ \varepsilon_2^- & \varepsilon_2^+ & & & & & \\ & & \varepsilon_2^+ & \varepsilon_2^- & & & \\ & & \varepsilon_2^- & \varepsilon_2^+ & & & \\ & & & & \ddots & & \\ & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix} \begin{bmatrix} \varepsilon_1^+ & & & & & & \varepsilon_1^- \\ & \varepsilon_1^+ & \varepsilon_1^- & & & & \\ & \varepsilon_1^- & \varepsilon_1^+ & & & & \\ & & & \ddots & & & \\ & & & & \varepsilon_1^+ & \varepsilon_1^- & \\ & & & & \varepsilon_1^- & \varepsilon_1^+ & \\ \varepsilon_1^- & & & & & & \varepsilon_1^+ \end{bmatrix} \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- & & & & & \\ \varepsilon_2^- & \varepsilon_2^+ & & & & & \\ & & \varepsilon_2^+ & \varepsilon_2^- & & & \\ & & \varepsilon_2^- & \varepsilon_2^+ & & & \\ & & & & \ddots & & \\ & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix}$$

Quantum Dynamics—II
Spectral Method

Aiichiro Nakano

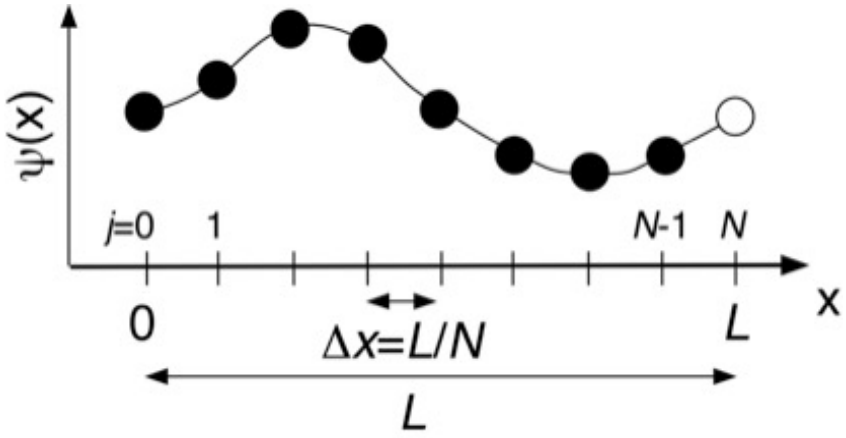
Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California

Email: anakano@usc.edu

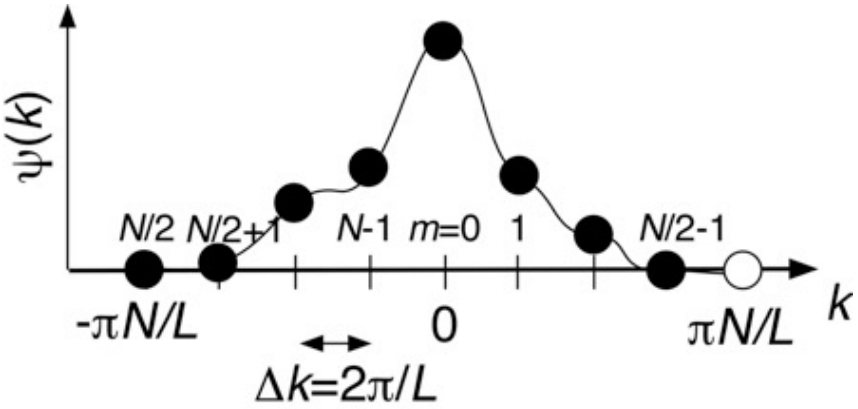


Discrete Fourier Transform

- **Discretize $\psi(x) \in C$ ($x \in [0, L]$) on N mesh points, $x_j = j\Delta x$ ($j = 0, \ldots, N\text{-}1$), with equal mesh spacing, $\Delta x = L/N$**
- **Periodic boundary condition: $\psi(x + L) = \psi(x)$**



First Brillouin zone

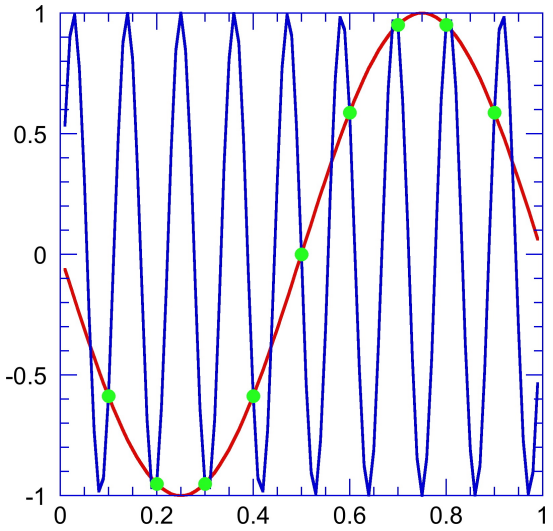


- **Discrete Fourier transform: Represents $\psi(x)$ as a linear combination of $\exp(ikx) = \cos(kx) + i \sin(kx)$, with different wave numbers, k**

$$\psi_j = \sum_{m=0}^{N-1} \tilde{\psi}_m \exp(ik_m x_j)$$

$$k_m = \begin{cases} 2\pi m/L & (m = 0,1, \ldots, N/2 - 1) \\ 2\pi\,(m - N)/L & (m = N/2\,, N/2 + 1, \ldots, N - 1) \end{cases}$$

$$\tilde{\psi}_m = \frac{1}{N} \sum_{j=0}^{N-1} \psi_j \exp(-ik_m x_j)$$



Wave Numbers

- **Periodic boundary condition, $\psi(x + L) = \psi(x)$, is guaranteed by choosing $k_m = 2\pi m/L$**

$$e^{i\left(\frac{2\pi m}{L}(x+L)\right)} = e^{i\left(\frac{2\pi m}{L}x+2\pi m\right)} = e^{i\frac{2\pi m}{L}x}\overbrace{e^{i2\pi m}}^{(e^{i2\pi})^m=1^m=1} = e^{i\frac{2\pi m}{L}x}$$

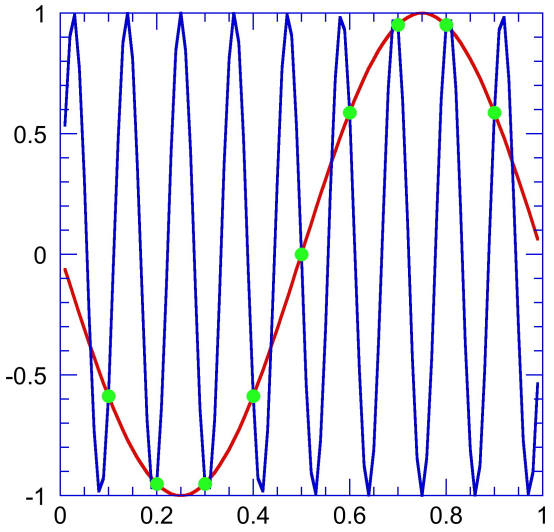
- **Folding back the latter half of wave numbers by $\frac{2\pi N}{L} = 2\pi/\Delta x$** ($\Delta x = \frac{L}{N}$)

  (*cf.* first Brillouin zone):

  From $\left[0, \frac{2\pi N}{L}\right] = [0, \frac{2\pi}{\Delta x}]$ **to** $\left[-\frac{2\pi}{L}\cdot\frac{N}{2}, \frac{2\pi}{L}\cdot\frac{N}{2}\right] = [-\frac{\pi}{\Delta x}, \frac{\pi}{\Delta x}]$

The shift won't change wave-function value on any grid point

$$e^{i\left(k_m-\frac{2\pi N}{L}\right)x_j}$$
$$= e^{ik_m x_j}e^{-i\frac{2\pi N}{L}\bullet\frac{L}{N}j}$$
$$= e^{ik_m x_j}\overbrace{e^{-i2\pi j}}^{1}$$
$$= e^{ik_m x_j}$$



Orthonormal Basis Set

- ***N*-dimensional vector space:** $|\psi\rangle = (\psi_0, \psi_1, \dots, \psi_{N-1})$
- **Plane-wave basis set:** $\left\{ |m\rangle = b_m(x_j) = \frac{1}{\sqrt{N}} \exp(ik_m x_j) \middle| m = 0,1,\dots,N-1 \right\}$
- **Orthonormality:** $\langle m|n\rangle = \sum_{j=0}^{N-1} b_m^*(x_j) b_n(x_j) = \delta_{m,n} = \begin{cases} 1 & (m=n) \\ 0 & (m \neq n) \end{cases}$

$$\therefore \langle m|n\rangle = \frac{1}{N}\sum_{j=0}^{N-1} \exp(i(k_n - k_m)x_j) = \frac{1}{N}\sum_{j=0}^{N-1} \exp\left(i\frac{2\pi}{N}(n-m)j\right)$$

$$= \begin{cases} \frac{1}{N}\frac{\exp(i2\pi(n-m))-1}{\exp\left(i\frac{2\pi}{N}(n-m)\right)-1} = 0 & (m \neq n) \\ \frac{1}{N}\cdot N = 1 & (m = n) \end{cases}$$

$(k_n - k_m)x_j = \frac{2\pi(n-m)}{L}\cdot\frac{L}{N}j$

Geometric series

$S = \sum_{j=0}^{N-1}\left(e^{i\frac{2\pi}{N}(n-m)}\right)^j = 1 + \cdots + r^{N-1}$

$rS = r + \cdots + r^N$

$\therefore (r-1)S = (r^N - 1)$

$|\psi\rangle = \sum_m c_m |m\rangle$

$\langle n| \times \Downarrow$

$\langle n|\psi\rangle = \sum_m c_m \overbrace{\langle n|m\rangle}^{\delta_{nm}} = c_n$

- **Completeness:** $|\psi\rangle = \sum_{m=0}^{N-1} |m\rangle\langle m|\psi\rangle$ **or** $1 = \sum_{m=0}^{N-1} |m\rangle\langle m|$
- **Fourier transform:** $\psi_j = \sum_{m=0}^{N-1} \exp(ik_m x_j) \boxed{\frac{1}{N}\sum_{l=0}^{N-1} \exp(-ik_m x_l)\psi_l}$ $\tilde{\psi}_m$

Spectral Method

- **Kinetic-energy operator is diagonal in the momentum space:** $\tilde{\psi}_m \xrightarrow{T} \frac{k_m^2}{2}\tilde{\psi}_m$

$$-\frac{1}{2}\frac{\partial^2}{\partial x^2}\sum_{m=0}^{N-1}\tilde{\psi}_m \exp(ik_m x) = \sum_{m=0}^{N-1}\frac{k_m^2}{2}\tilde{\psi}_m \exp(ik_m x)$$

- **Potential-energy operator is diagonal in the real space:** $\psi_j \xrightarrow{V} V_j\,\psi_j$
- **Split-operator technique & spectral method**

$$\psi(t+\Delta t) = \exp\left(-\frac{iV\Delta t}{2}\right) \overset{F}{\exp(-iT\Delta t)} \overset{F^{-1}}{\exp\left(-\frac{iV\Delta t}{2}\right)}\psi(t) + O\left((\Delta t)^3\right)$$

1. $\psi_j \xrightarrow[\exp(-iV\Delta t/2)]{} \exp\left(-iV_j\Delta t/2\right)\psi_j$ **O(N)**
2. $\psi_j \xrightarrow[F^{-1}]{} F^{-1}\psi_j = \tilde{\psi}_m = \frac{1}{N}\sum_{j=1}^{N}\psi_j \exp\left(-ik_m x_j\right)$ **O(N²)?**
3. $\tilde{\psi}_m \xrightarrow[\exp(-iT\Delta t)]{} \exp\left(-ik_m^2\Delta t/2\right)\tilde{\psi}_m$ **O(N)**
4. $\tilde{\psi}_m \xrightarrow[F]{} F\tilde{\psi}_m = \psi_j = \sum_{m=1}^{N}\tilde{\psi}_m \exp\left(ik_m x_j\right)$ **O(N²)?**
5. $\psi_j \xrightarrow[\exp(-iV\Delta t/2)]{} \exp\left(-iV_j\Delta t/2\right)\psi_j$ **O(N)**

Exact exponentiation!

$$\exp\left(\frac{i\Delta t}{2}\frac{\partial^2}{\partial x^2}\right)\sum_m \tilde{\psi}_m e^{-ik_m x}$$
$$= \sum_m \exp\left(-\frac{i\Delta t k_m^2}{2}\right)\tilde{\psi}_m e^{-ik_m x}$$

Solution: Fast Fourier Transform

In putting together this issue of *Computing in Science & Engineering*, we knew three things: it would be difficult to list just 10 algorithms; it would be fun to assemble the authors and read their papers; and, whatever we came up with in the end, it would be controversial. We tried to assemble the 10 algorithms with the greatest influence on the development and practice of science and engineering in the 20th century. Following is our list (here, the list is in chronological order; however, the articles appear in no particular order):

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

IEEE Comput. Sci. Eng. **2(1)**, 22 ('00)

Fast Fourier Transform

- **Danielson-Lanczos algorithm:**

$$\psi_j = \sum_{m=0}^{N-1} \tilde{\psi}_m \exp\left(ik_m x_j\right) = \sum_{m=0}^{N-1} \tilde{\psi}_m \exp\left(i2\pi mj/N\right) \quad O(N^2)!$$

$$= \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m} \exp\left(i2\pi(2m)j/N\right) + \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m+1} \exp\left(i2\pi(2m+1)j/N\right)$$

even terms odd terms

$$= \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m} \exp\left(i2\pi mj/(N/2)\right) + \exp\left(i2\pi j/N\right) \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m+1} \exp\left(i2\pi mj/(N/2)\right)$$

$$\psi_j = \psi_j^0 + W_N^j \psi_j^1$$

$$\begin{cases} \psi_j^0 = \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m} \exp\left(i2\pi mj/(N/2)\right) \\ \psi_j^1 = \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m+1} \exp\left(i2\pi mj/(N/2)\right) \\ W_N = \exp\left(i2\pi/N\right) \end{cases}$$

even & odd subarray Fourier decompositions

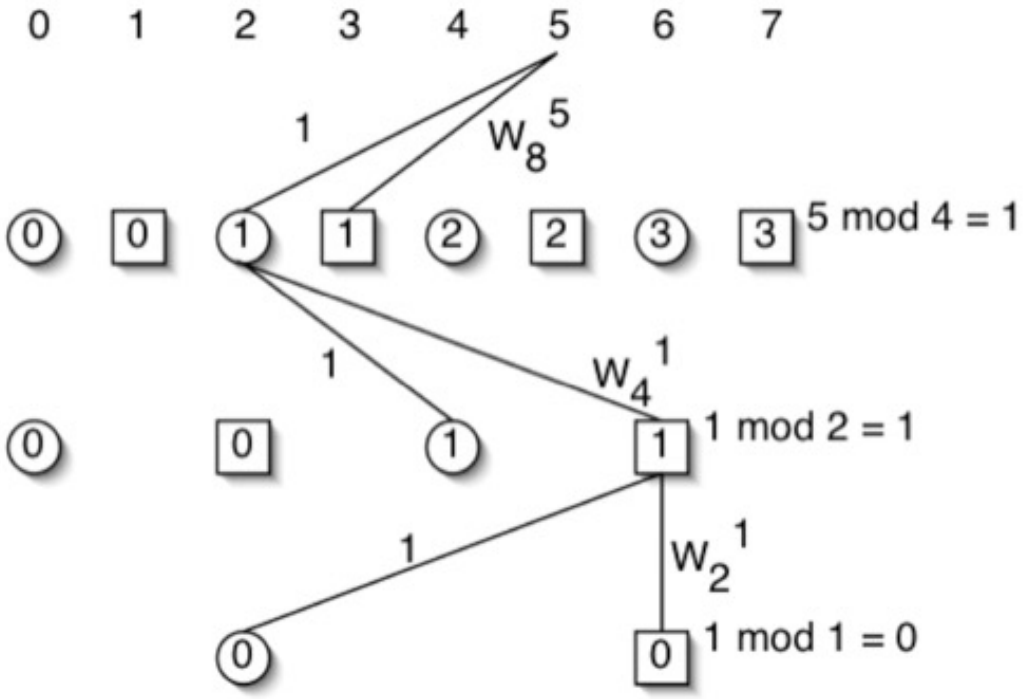
j read as *j* mod *N*/2

Divide-and-conquer

Fast Fourier Transform

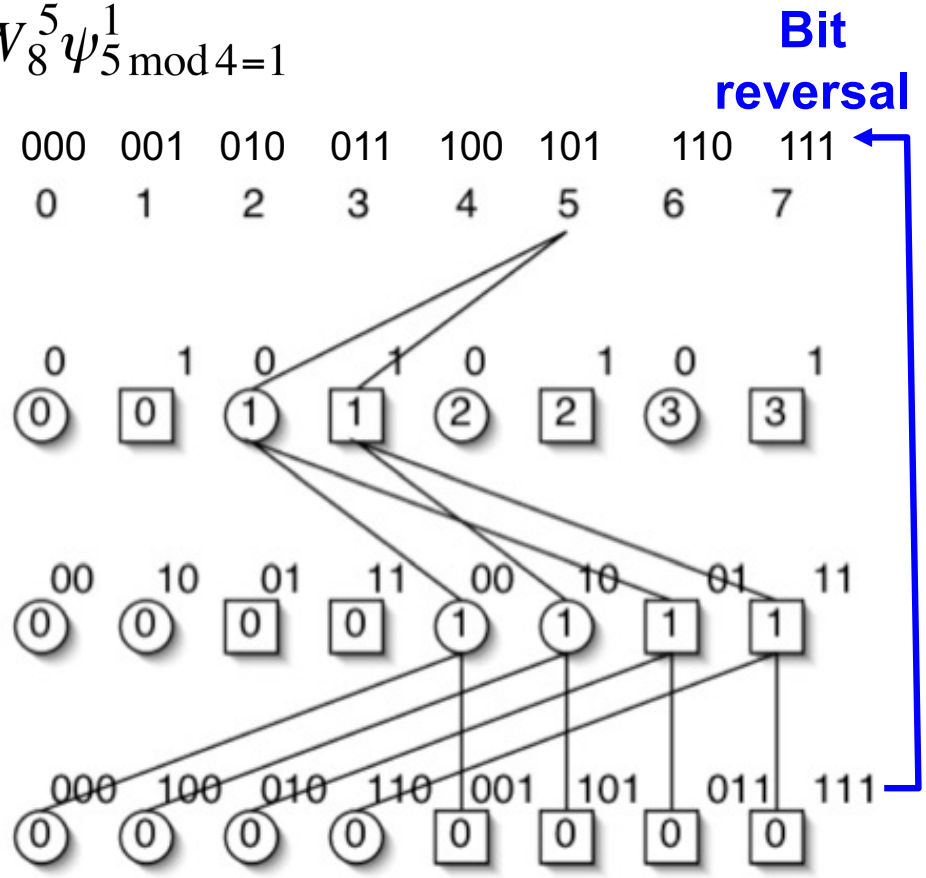
- **Recursive sub-Fourier transforms:** $\psi_j = \psi_j^0 + W_N^j \psi_j^1$

$$\psi_5 = \psi_{5 \bmod 4=1}^0 + W_8^5 \psi_{5 \bmod 4=1}^1$$



010→010→2 011→110→6

Path bit reversal = element

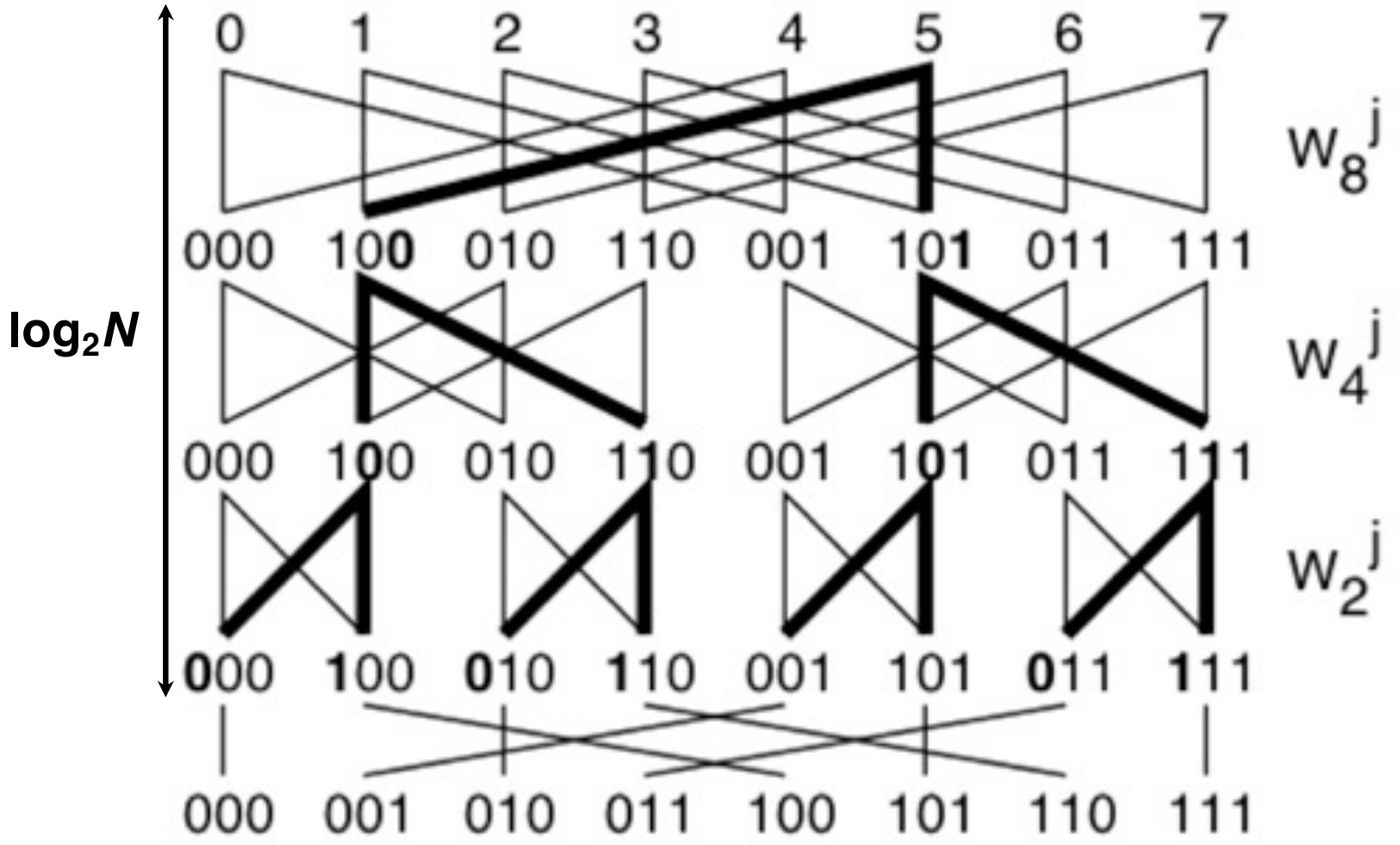


Recursion tree

***O*(*N*) operations per element**

Fast Fourier Transform Algorithm

- Many computations are shared among the recursion trees
- Butterfly (hypercube) data exchange after bit-reversal:
- $2N\log_2 N$ complex arithmetic operations

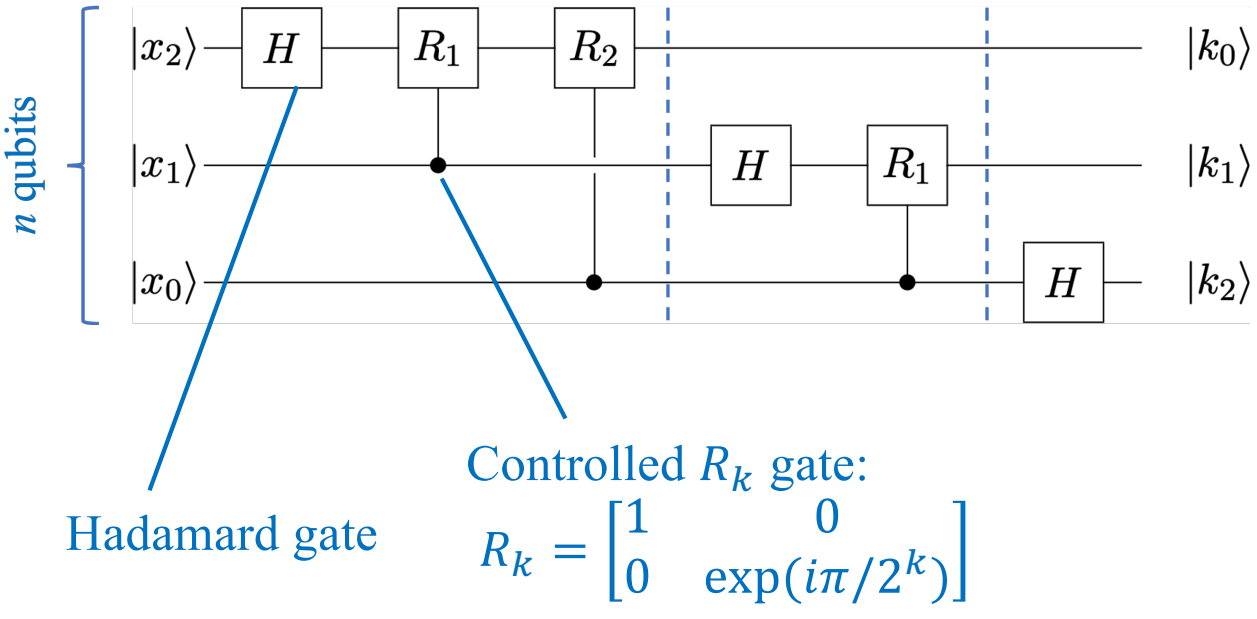


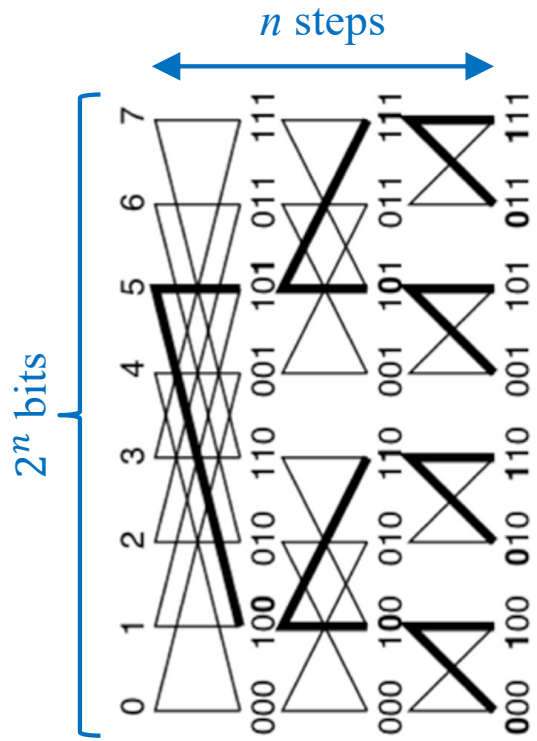
Quantum Fourier Transform (QFT)

- **On a quantum computer, quantum parallelism allows Fourier transform to be performed using only n qubits & $O(n^2)$ gates for $N = 2^n$**

$$\text{QFT: } |x\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i\frac{2\pi kx}{N}} |k\rangle$$

$n + (n-1) + \cdots + 2 + 1 = \frac{n(n+1)}{2}$ gates (depths)





- **Compare QFT with $2N\log_2 N = O(2^n n)$ arithmetic operations in classical fast Fourier transform (FFT):**

Exponential operation reduction: $O(2^n n/n^2) = O(2^n/n)$

Preskill, *arXiv*:2106.10522 ('21)

Parallel Quantum Dynamics

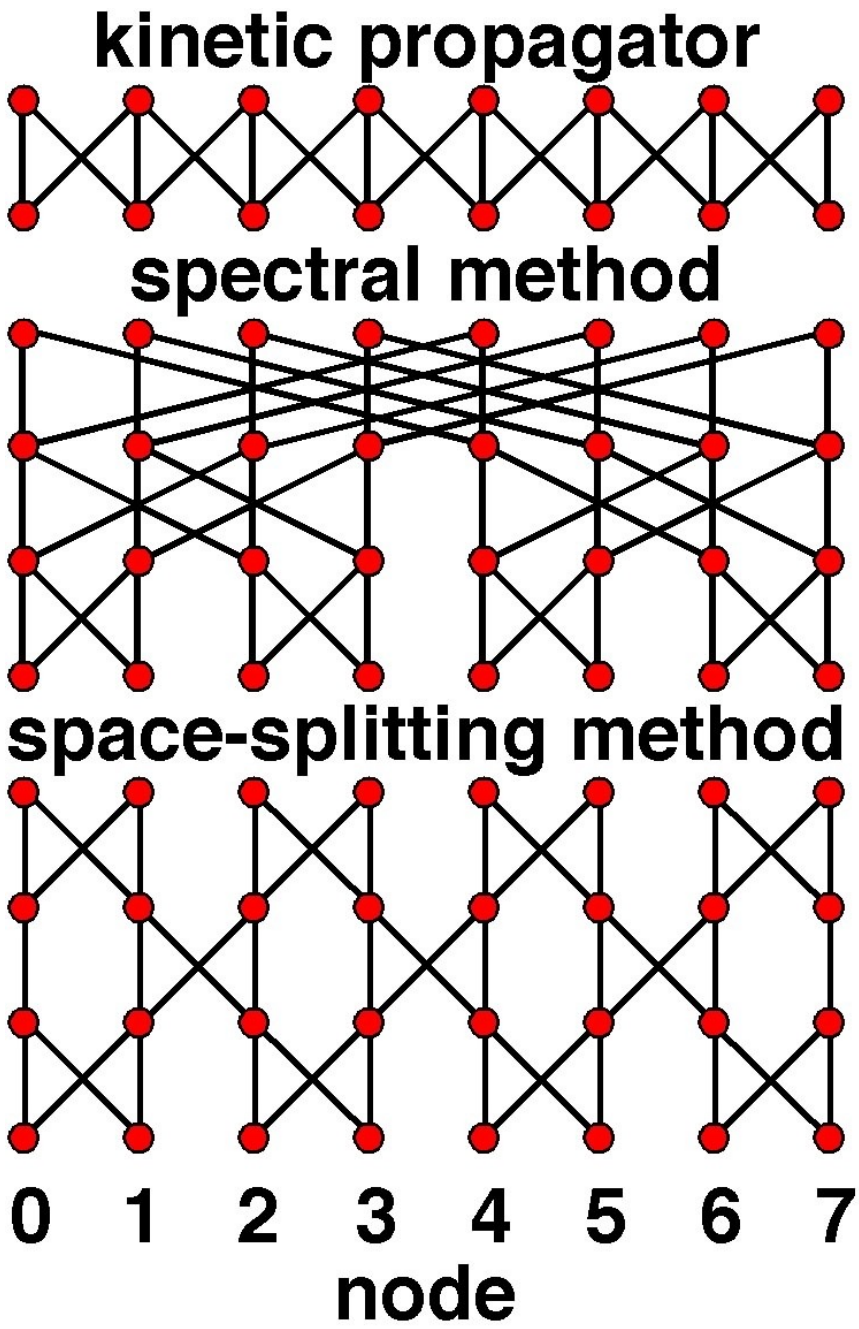
Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California

Email: anakano@usc.edu

Self-centric parallelization of a partial-differential-equation solver as a 'boundary condition'

Parallel QD Communications



$\psi \leftarrow K\psi$

$\psi \leftarrow \exp(-iK\Delta)\psi$

$O(M \log M)$

Large-stride communications; good up to 10^3 processors

$\psi \leftarrow \exp(-iK\Delta)\psi$

$O(M)$

Metascalable

Parallel QD Algorithms

- **Not all algorithms are scalable on parallel computers**
- **Implicit solvers (*e.g.* Crank-Nicholson method) are numerically stable but less scalable due to sequential dependence**

$$\psi(t+\Delta t) \leftarrow \exp\left(-\frac{i}{\hbar}\hat{H}\Delta t\right)\psi(t) \cong \frac{1-\frac{i}{2\hbar}\hat{H}\Delta t}{1+\frac{i}{2\hbar}\hat{H}\Delta t}\psi(t) + O\left((\Delta t)^3\right)$$

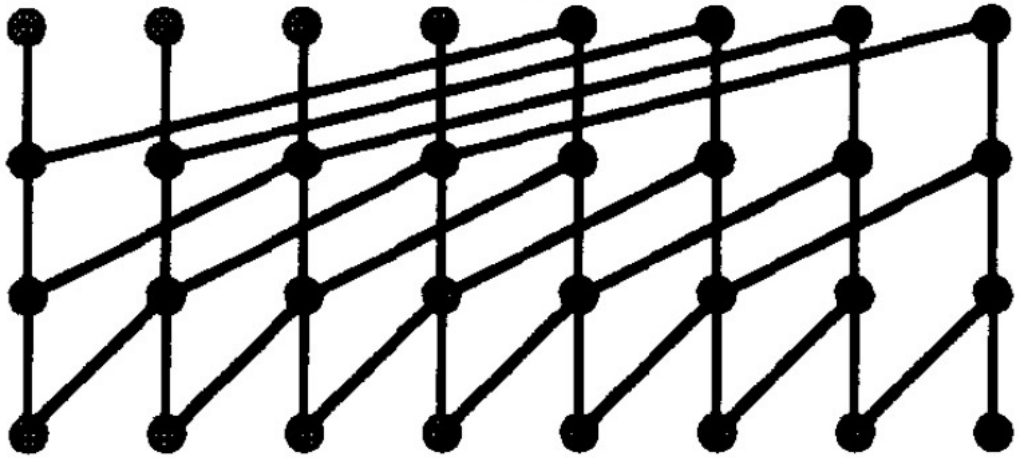
$$\overbrace{\left(1+\frac{i}{2\hbar}\hat{H}\Delta t\right)}^{A}\overbrace{\psi(t+\Delta t)}^{x} = \overbrace{\left(1-\frac{i}{2\hbar}\hat{H}\Delta t\right)\psi(t)}^{b}$$

$$\alpha x_{i-1} + \beta x_i + \alpha x_{i+1} = b_i$$

$$\Longrightarrow$$

$$x_{i+1} \leftarrow \frac{1}{\alpha}b_i - \frac{\beta}{\alpha}x_i - x_{i-1}$$

- **Sequential recursion needs be converted to divide-&-conquer (recursive doubling) for parallelization**



A. Nakano, *Comput. Phys. Commun.* **83**, 181 ('94)

Self-Centric (SC) Parallelization

- **SC is the easiest serial-to-parallel migration path *via* single-program multiple-data (SPMD) programming**
  1. **Take a serial code**
  2. **Each MPI rank only works on a spatial subsystem**
  3. **Boundary information obtained from neighbor ranks**
  4. **Long-range information (if needed) by divide-&-conquer, like real-space multigrids; scalability behavior similar to short-ranged**

F. Shimojo *et al*., *J. Chem. Phys.* 140, 18A529 ('14)
K. Nomura *et al*., *IEEE/ACM Supercomputing, SC14* ('14)
A. Nakano, *Comput. Phys. Commun*. **104**, 59 ('97)

Quantum Dynamics Program: `qd1.c`

```
for step = 1 to NSTEP
  pot_prop(): ψj←exp(-iVjΔt/2)ψj (j∈[1,NX])
  kin_prop(Δt/2)
  kin_prop(Δt)
  kin_prop(Δt/2)
  pot_prop(): ψj←exp(-iVjΔt/2)ψj (j∈[1,NX])
```

$$\psi(t+\Delta t) \leftarrow \exp(-iV\Delta t/2)\exp(-iT_x\Delta t)\exp(-iV\Delta t/2)\psi(t)$$
$$= e^{-iV\Delta t/2}U_x^{(\text{half})}U_x^{(\text{full})}U_x^{(\text{half})}e^{-iV\Delta t/2}\psi(t)$$

```
kin_prop(Δ)
  periodic_bc(): ψ0←ψNX; ψNX+1←ψ1
  for ∀j∈[1,NX]
    ψj←blx(Δ)jψj-1+al(Δ)jψj+bux(Δ)jψj+11
```

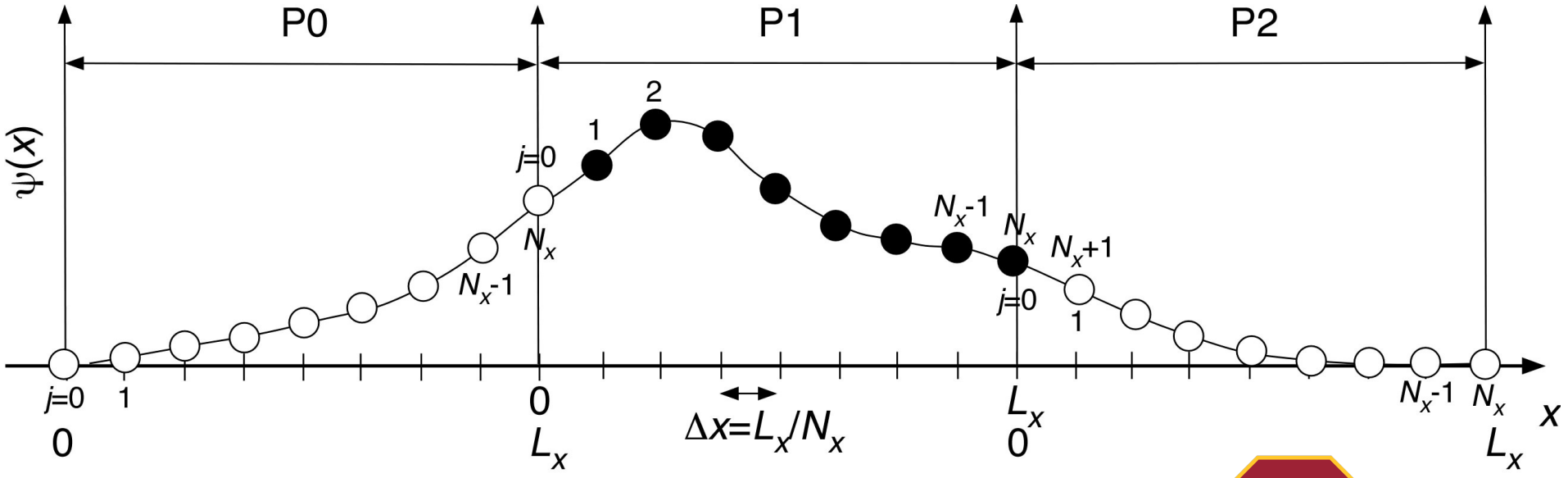
$$\begin{cases} \varepsilon_n^+ = \frac{1}{2}\left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right)+\exp\left(-\frac{i\Delta t}{n}(a-b)\right)\right] \\ \varepsilon_n^- = \frac{1}{2}\left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right)-\exp\left(-\frac{i\Delta t}{n}(a-b)\right)\right] \end{cases}$$

$$\exp(-i\Delta t T_x) \cong U_x^{(\text{half})}U_x^{(\text{full})}U_x^{(\text{half})} = \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- & & & & & \\ \varepsilon_2^- & \varepsilon_2^+ & & & & & \\ & & \varepsilon_2^+ & \varepsilon_2^- & & & \\ & & \varepsilon_2^- & \varepsilon_2^+ & & & \\ & & & & \ddots & & \\ & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix} \begin{bmatrix} \varepsilon_1^+ & & & & & & \\ & \varepsilon_1^+ & \varepsilon_1^- & & & & \\ & \varepsilon_1^- & \varepsilon_1^+ & & & & \\ & & & \ddots & & & \\ & & & & \varepsilon_1^+ & \varepsilon_1^- & \\ & & & & \varepsilon_1^- & \varepsilon_1^+ & \\ & & & & & & \varepsilon_1^+ \end{bmatrix} \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- & & & & & \\ \varepsilon_2^- & \varepsilon_2^+ & & & & & \\ & & \varepsilon_2^+ & \varepsilon_2^- & & & \\ & & \varepsilon_2^- & \varepsilon_2^+ & & & \\ & & & & \ddots & & \\ & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix}$$

$\psi_j(t+1) \leftarrow f(\psi_{j-1}(t), \psi_j(t), \psi_{j+1}(t))\ (j \in [1, NX])$

SC Parallelization

- **Self-centric spatial decomposition**



- **Local & global coordinates**

$$\begin{cases} x_j = j\Delta x \\ x_j^{(\text{global})} = j\Delta x + pL_x \end{cases}$$

off-set

- **Global coordinates only in `init_prop()` & `init_wavefn()`**

***O*(*N*) divide-&-conquer algorithms maximally expose data locality**
→ easy self-centric parallelization & metascalable

Boundary Wave Function Caching

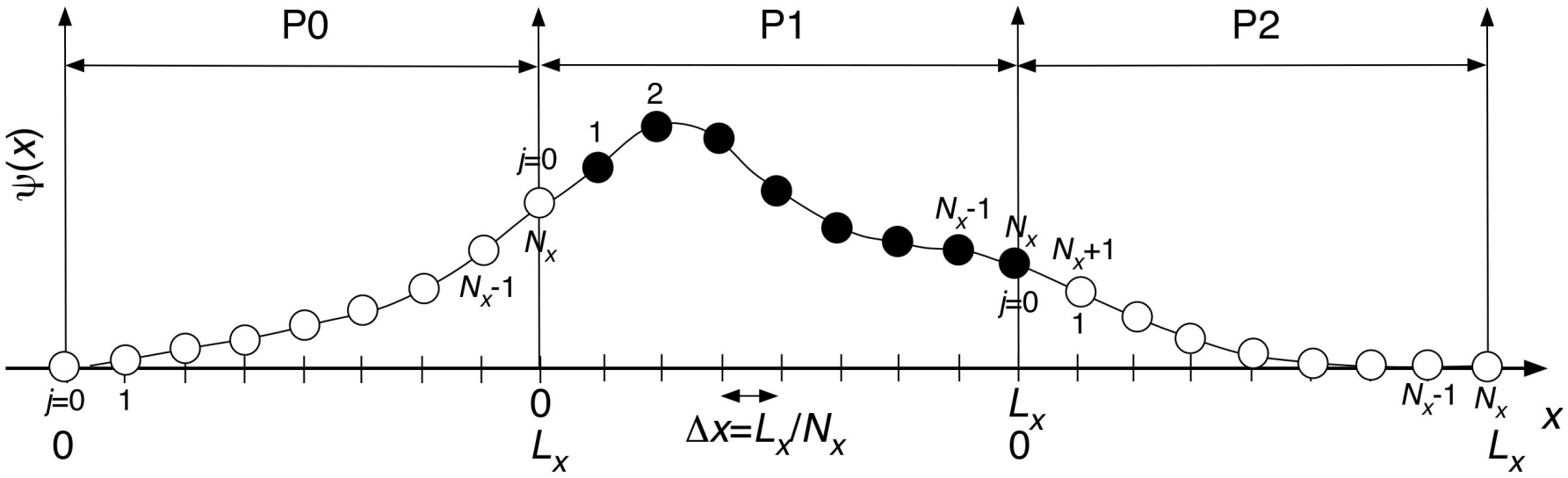
- **Parallelized `periodic_bc()`**

```
plw = (myid-1+nproc)%nproc; /* Lower partner process */
pup = (myid+1      )%nproc; /* Upper partner process */

/* Cache boundary wave function value at the lower end */
dbuf[0:1] ← psi[NX][0:1];
Send dbuf to pup;
Receive dbufr from plw;
psi[0][0:1] ← dbufr[0:1];

/* Cache boundary wave function value at the upper end */
dbuf[0:1] ← psi[1][0:1];
Send dbuf to plw;
Receive dbufr from pup;
psi[NX+1][0:1] ← dbufr[0:1];
```

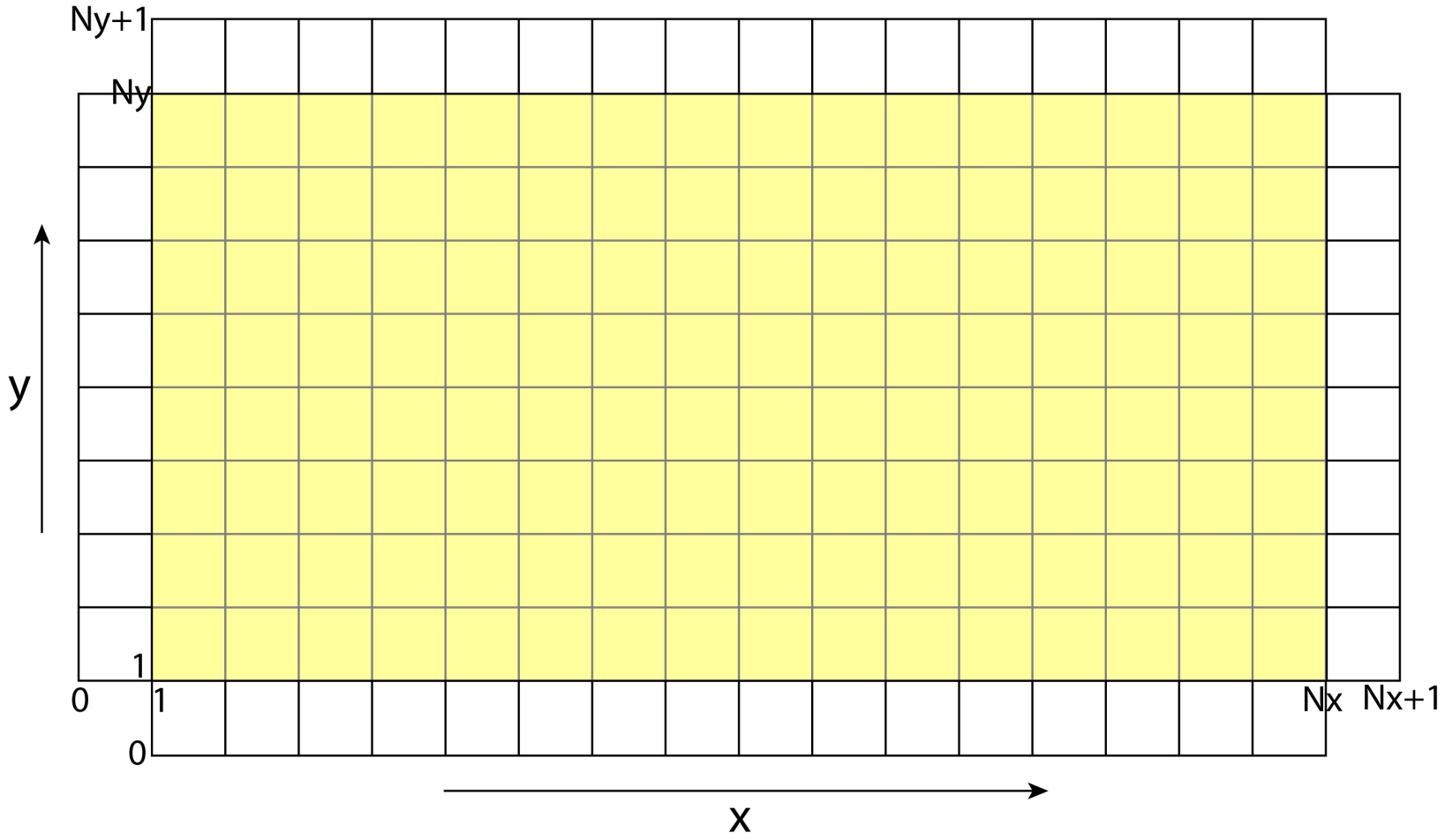
cf. `serial_periodic_bc()`:
$\psi_0 \leftarrow \psi_{NX}$; $\psi_{NX+1} \leftarrow \psi_1$



Multidimensional Parallelization

- **Parallelized `periodic_bc()`**

```
for ∀directions
  send front row psi(...,1 or Nα,...) to forward neighbor
  receive back appendage psi(...,Nα+1 or 0,...) from back neighbor
```



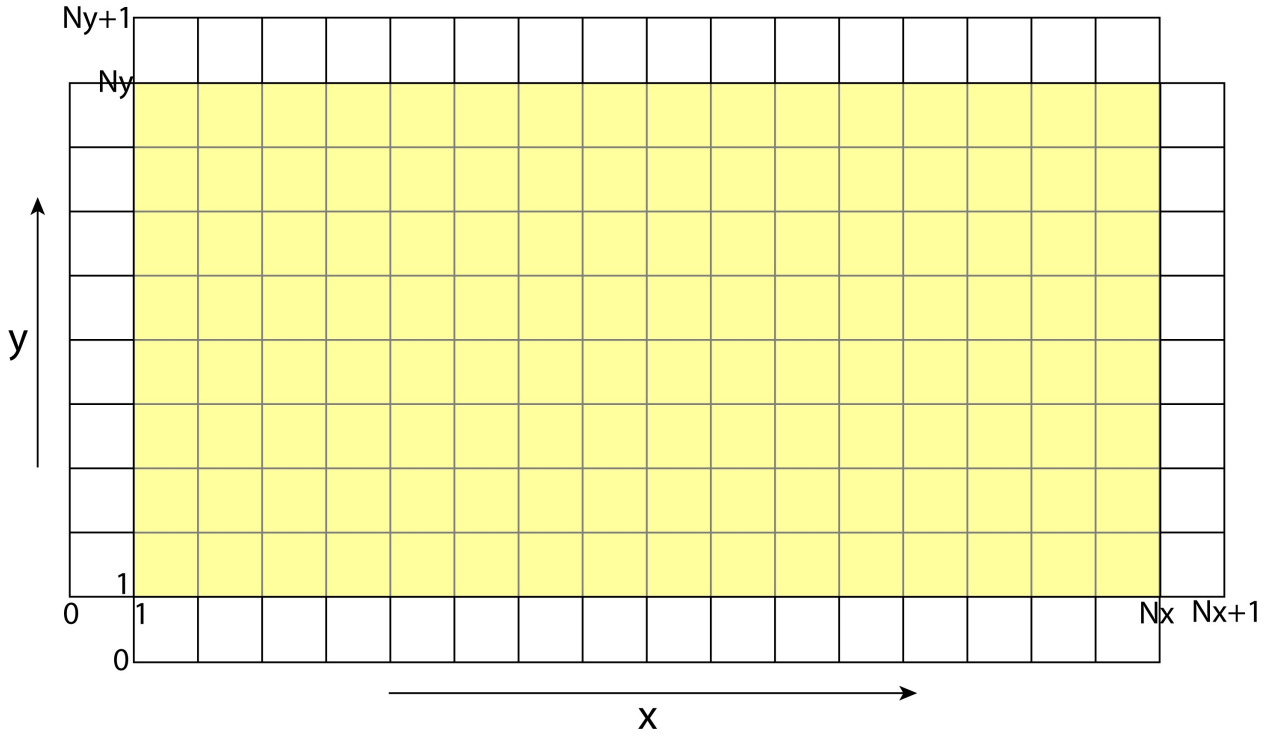
Multidimensional Parallelization

- **Parameterized message composition**

$$dbuf \leftarrow psi(i_b : i_e, j_b : j_e, k_b : k_e)$$
$$psi(i'_b : i'_e, j'_b : j'_e, k'_b : k'_e) \leftarrow dbufr$$

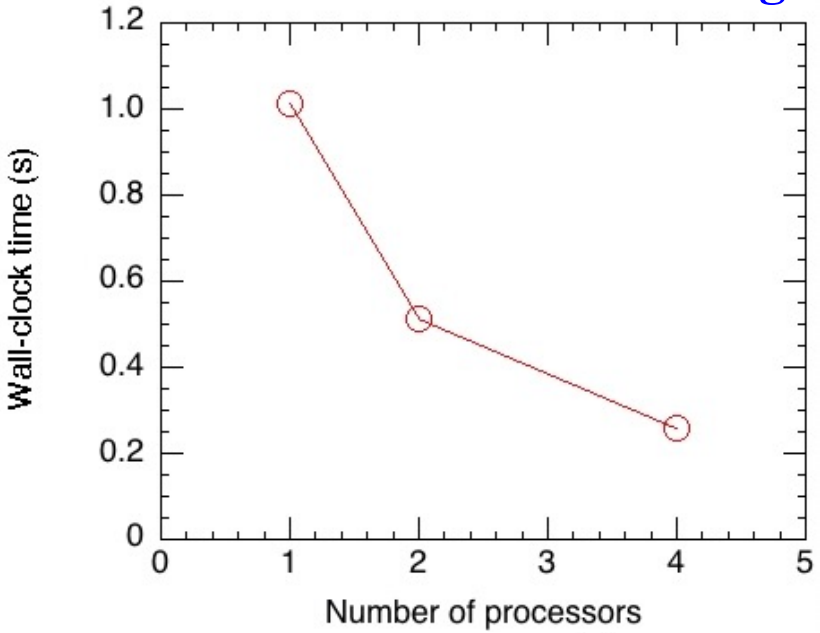
(Example) x-low direction

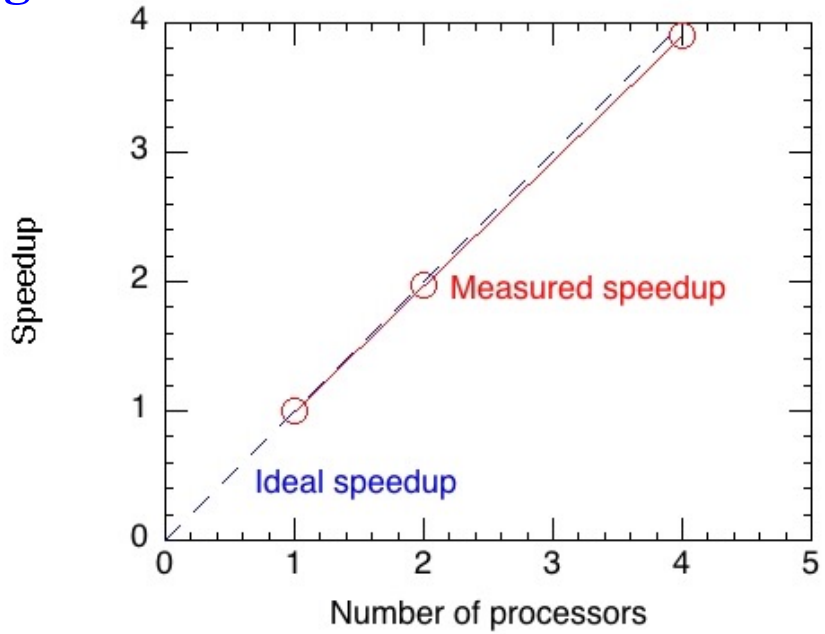
$$i_b = 1, i_e = 1, j_b = 1, j_e = N_y, k_b = 1, k_e = N_z$$
$$i'_b = N_x+1, i'_e = N_x+1, j'_b = 1, j'_e = N_y, k'_b = 1, k'_e = N_z$$

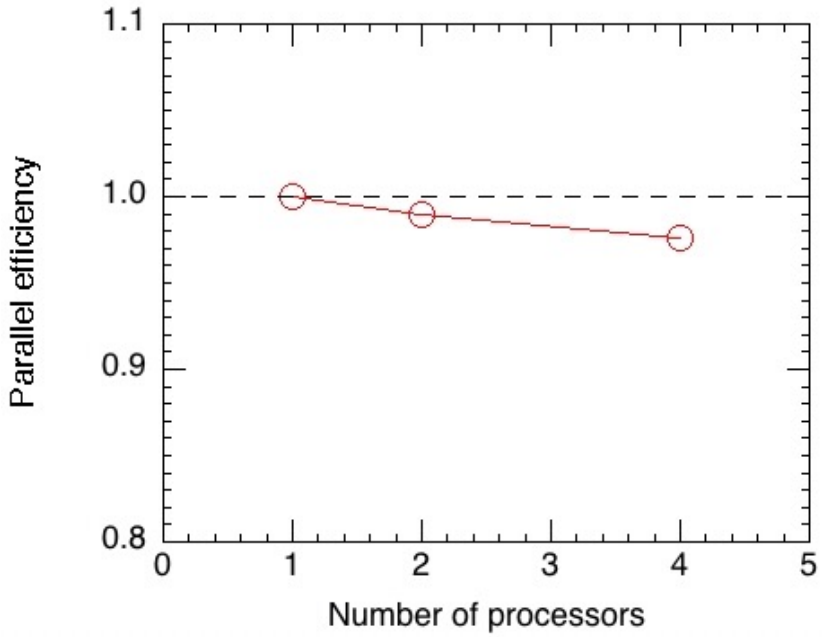


Parallel QD Results

Strong scaling results

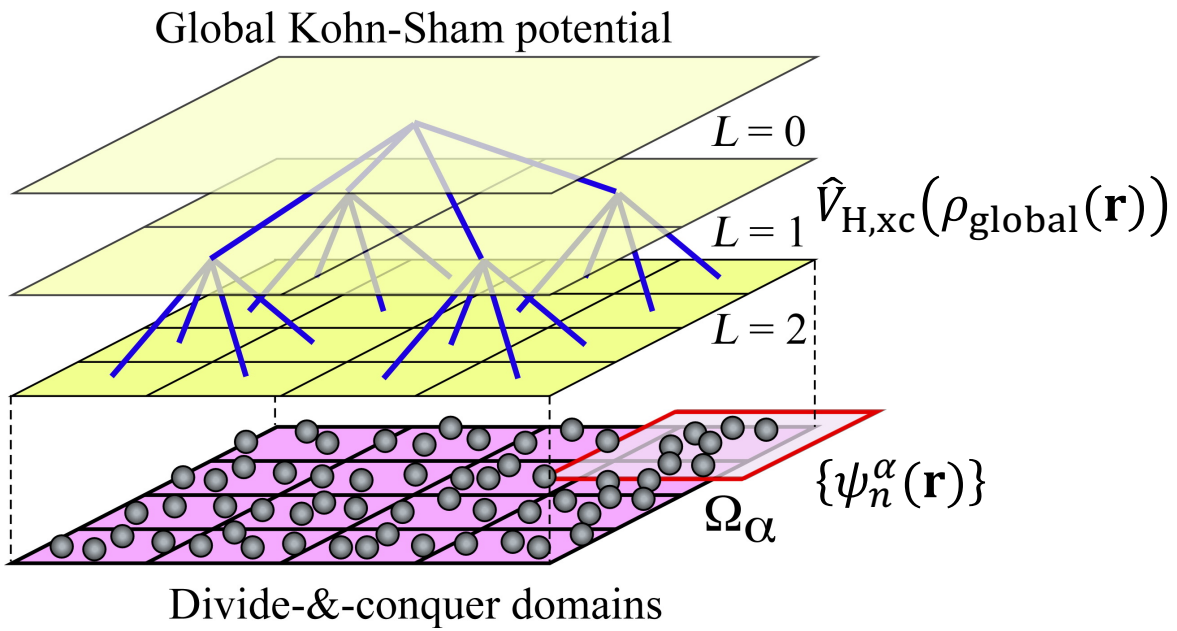




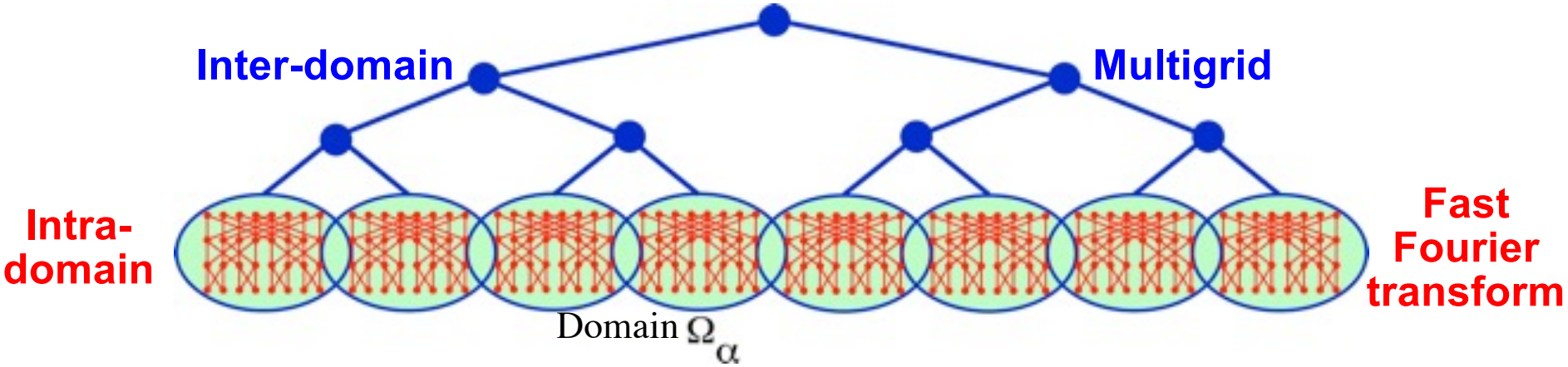


Hierarchical Parallel QD

- **Divide-&-conquer density functional theory (DFT): Global-local self-consistent field (SCF) iteration to determine local electronic wave functions & global electrostatic potential**



- **Globally scalable (real-space multigrid) + locally fast (plane wave) solver**



F. Shimojo *et al.*, *J. Chem. Phys.* **140**, 18A529 ('14)
K. Nomura *et al.*, *IEEE/ACM Supercomputing, SC14* ('14)