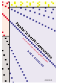
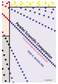
Laplacian Matrices
(PSC §4.8)



Physical domain

- In many applications, a physical domain exists that can be distributed naturally by assigning a contiguous subdomain to every processor.
- Communication is only needed for exchanging information across the subdomain boundaries.
- Often, the domain is structured as a multidimensional rectangular grid, where grid points interact only with a set of immediate neighbours.
- In the 2D case, these could be the neighbours to the north, east, south, and west.
- An example is the heat equation, where the value at a grid point represents the temperature at the corresponding location.



2D Laplacian operator for $k \times k$ grid

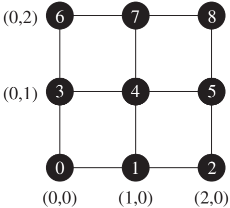
Compute

$$\Delta_{i,j} = x_{i-1,j} + x_{i+1,j} + x_{i,j+1} + x_{i,j-1} - 4x_{i,j}, \quad \text{for } 0 \leq i, j < k,$$

where $x_{i,j}$ denotes the temperature at grid point (i, j). By convention, $x_{i,j} = 0$ outside the grid.

- $x_{i+1,j} - x_{i,j}$ approximates the derivative of the temperature in the i-direction.
- $(x_{i+1,j} - x_{i,j}) - (x_{i,j} - x_{i-1,j}) = x_{i-1,j} + x_{i+1,j} - 2x_{i,j}$ approximates the second derivative.

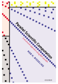
Relation grid–vector



- A 3×3 grid, which corresponds to a vector of length 9. For each grid point (i, j), the index $i + 3j$ of the corresponding vector component is shown.
- More in general,

$$v_{i+jk} \equiv x_{i,j}, \quad u_{i+jk} \equiv \Delta_{i,j},$$

for $0 \leq i, j < k$.

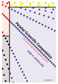


Relation operator–matrix

$$A = \begin{bmatrix} -4 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & -4 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & -4 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & -4 & 1 & \cdot & 1 & \cdot & \cdot \\ \cdot & 1 & \cdot & 1 & -4 & 1 & \cdot & 1 & \cdot \\ \cdot & \cdot & 1 & \cdot & 1 & -4 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & -4 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & -4 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & -4 \end{bmatrix}$$

$$\mathbf{u} = A\mathbf{v} \Longleftrightarrow$$

$$\Delta_{i,j} = x_{i-1,j} + x_{i+1,j} + x_{i,j+1} + x_{i,j-1} - 4x_{i,j}, \quad \text{for } 0 \leq i, j < k.$$



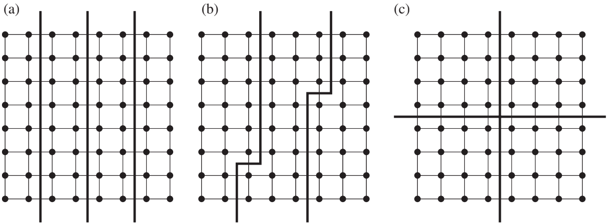
Domain view vs. matrix view

- In general, it is best to view the Laplacian as an operator on the physical domain.
- This domain view has the advantage that it naturally leads to the use of a regular data structure.
- Occasionally, however, it may be beneficial to view the Laplacian as a matrix, so that we can apply our knowledge about sparse matrix–vector multiplication.

Find a domain distribution

- Here, we adopt the domain view, so that we must assign each grid point to a processor.
- We assign the values $x_{i,j}$ and $\Delta_{i,j}$ to the owner of grid point (i,j), which translates into $\text{distr}(\mathbf{u}) = \text{distr}(\mathbf{v})$.
- We use a row distribution for the matrix and assign row $i + jk$ to the same processor as vector component u_{i+jk} and hence grid point (i,j).
- The resulting sparse matrix–vector multiplication algorithm has two supersteps, the fanout and the local matrix–vector multiplication.
- The computation time for an interior point is 5 flops; for a border point 4 flops; for a corner point 3 flops.

Distribution into strips and blocks



- (a) Distribution into strips: long Norwegian borders,

$$T_{\text{comm, strips}} = 2kg.$$

- (b) Boundary corrections improve load balance.
- (c) Distribution into square blocks: shorter borders,

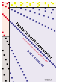
$$T_{\text{comm, squares}} = \frac{4k}{\sqrt{p}} g \qquad (\text{for } p > 4).$$

Surface-to-volume ratio

- The communication-to-computation ratio for square blocks is

$$\frac{T_{\text{comm, squares}}}{T_{\text{comp, squares}}} = \frac{4k/\sqrt{p}}{5k^2/p} g = \frac{4\sqrt{p}}{5k} g.$$

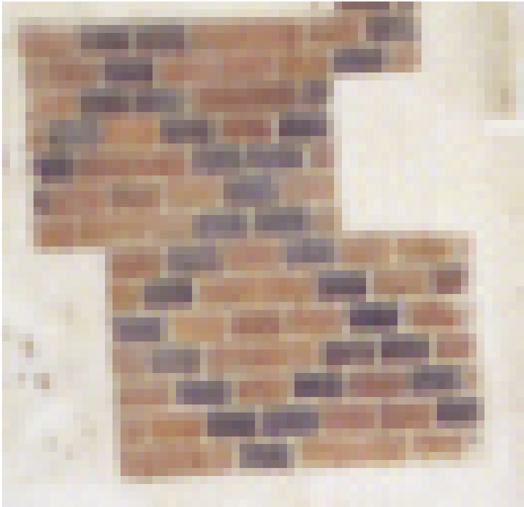
- This ratio is often called the surface-to-volume ratio, because in 3D the surface of a domain represents the communication with other processors and the volume represents the amount of computation of a processor.



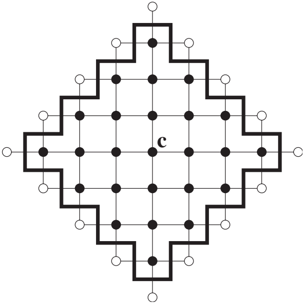
What do we do at scientific workshops?

Participants of HLPP 2001, International Workshop on High-Level Parallel Programming, Orléans, France, June 2001, studying Château de Blois.

The high-level object of our study



Blocks are nice, diamonds . . .



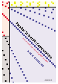
$r = 3$

- Digital diamond, or closed l_1-sphere, defined by

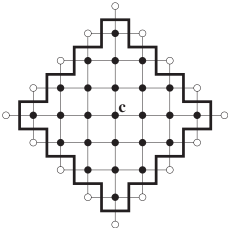
$$B_r(c_0, c_1) = \{(i, j) \in \mathbf{Z}^2 : \ |i - c_0| + |j - c_1| \le r\},$$

for integer radius $r \ge 0$ and centre $\mathbf{c} = (c_0, c_1) \in \mathbf{Z}^2$.

- $B_r(\mathbf{c})$ is the set of points with Manhattan distance $\le r$ to the central point $\mathbf{c}$.



Points of a diamond



$r = 3$

The number of points of $B_r(\mathbf{c})$ is

$$\begin{aligned} & 1+3+5+\cdots+(2r-1)+(2r+1)+(2r-1)+\cdots+1 \\ = \; & 2\sum_{k=0}^{r-1}(2k+1)+(2r+1) = 4\sum_{k=0}^{r-1} k + 4r + 1 \\ = \; & 2(r-1)r + 4r + 1 = 2r^2 + 2r + 1. \end{aligned}$$

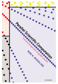
The number of neighbouring points is $4r+4$.

Diamonds are forever

- Assume that the diamond has its fair share $2r^2 + 2r + 1 = \frac{k^2}{p}$ of the grid points.
- Therefore, $2r^2 \approx \frac{k^2}{p}$ for large r, and hence $r \approx \frac{k}{\sqrt{2p}}$.
- Just on the basis of $4r + 4$ receives, we have

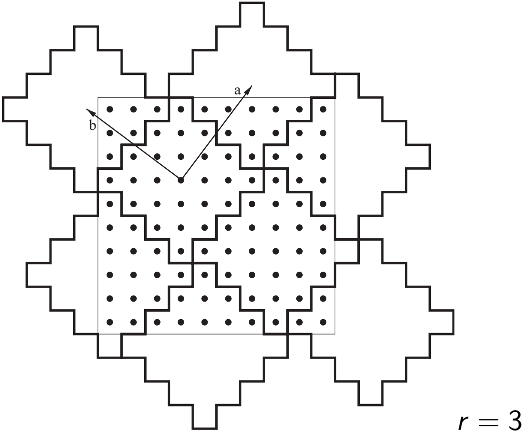
$$\frac{T_{\text{comm, diamonds}}}{T_{\text{comp, diamonds}}} = \frac{4r+4}{5(2r^2+2r+1)} g \approx \frac{2}{5r} g \approx \frac{2\sqrt{2p}}{5k} g.$$

- Compare with value $\frac{4\sqrt{p}}{5k} g$ for square blocks: factor $\sqrt{2}$ less.
- Gain caused by reuse of data: value at grid point is used twice but sent only once.



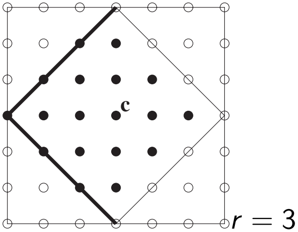
Alhambra: tile the whole space

Tile the whole sky with diamonds



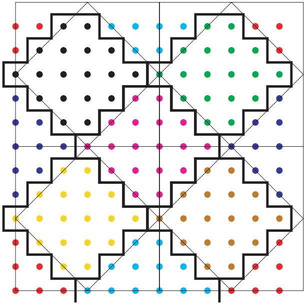
Diamond centres at $\mathbf{c} = \lambda\mathbf{a} + \mu\mathbf{b}$, $\lambda, \mu \in \mathbf{Z}$, where $\mathbf{a} = (r, r+1)$ and $\mathbf{b} = (-r-1, r)$. Good method for an infinite grid.

Practical method for finite grids



- Discard one layer of points from the north-eastern and south-eastern border of the diamond.
- For $r = 3$, the number of points decreases from 25 to 18.

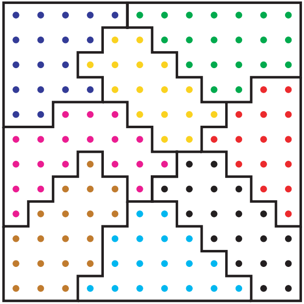
12×12 computational grid: periodic partitioning



8 processors

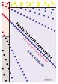
- Total computation: 672 flops. Avg 84. Max 90.
- Communication: 104 values. Avg 13. Max 14.
- Total time: $90 + 14g = 90 + 14 \cdot 10 = 230$ (ignoring $2l$).
- Rectangular 6×3 blocks: time would be $87 + 15 \cdot 10 = 237$. Worse!

12×12 computational grid: Mondriaan partitioning



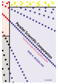
8 processors

- Total computation: 672 flops. Avg 84. Max 91. ($\epsilon = 10\%$.)
- Communication: 85 values. Avg 10.525. Max 16.
- Total time: $91 + 16g = 91 + 16 \cdot 10 = 251$.
- Challenge: better solution can be obtained manually, using ideas from both solutions shown. Current best known solution is 199 (Bas den Heijer 2006).

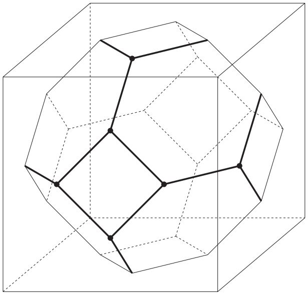


Three dimensions

- If a processor has a cubic block of $N = k^3/p$ points, about $\frac{6k^2}{p^{2/3}} = 6N^{2/3}$ are boundary points. In 2D, only $4N^{1/2}$.
- If a processor has a $10 \times 10 \times 10$ block, 488 points are on the boundary. About half!
- Thus, communication is important in 3D.
- Based on the surface-to-volume ratio of a 3D digital diamond, we can aim for a reduction by a factor $\sqrt{3} \approx 1.73$ in communication cost.
- The prime application of diamond-shaped distributions will most likely be in 3D.



Basic cell for 3D

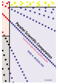


- Basic cell: grid points in a truncated octahedron.
- For load balancing, take care with the boundaries.
- What You See, Is What You Get (WYSIWYG):
  4 hexagons and 3 squares visible at the front are included.
  Also 12 edges, 6 vertices.
- Gain factor of 1.68 achieved for $p = 2q^3$.

Comparing 3 distribution methods in 2D and 3D

Grid	p	Rectangular	Diamond	Mondriaan
1024×1024	2	1024	2046	1024
	4	1024	2048	1240
	8	1280	1026	1378
	16	1024	1024	1044
	32	768	514	766
	64	512	512	548
	128	384	258	395
$64 \times 64 \times 64$	16	4096	2402	2836
	128	1024	626	829

Communication cost (in g) for a Laplacian operation on a grid. Mondriaan with $\epsilon = 10\%$.



Summary

- Communication can be reduced tremendously by using knowledge of the physical domain.
- To achieve a good distribution with a low surface-to-volume ratio, all dimensions must be cut. In 2D, this gives square blocks. In 3D, cubic subdomains.
- In 2D, an even better method is to use digital diamonds (with two edge layers removed). This basic cell can be used to tile a rectangular domain in a straightforward manner. Best performance is obtained for $p = 2q^2$.
- In 3D, the best method is to use truncated octahedra with WYSIWYG tie breaking at the boundaries. Best performance is obtained for $p = 2q^3$.
- In 3D, the performance of Mondriaan is between that of cubes and truncated octahedra.