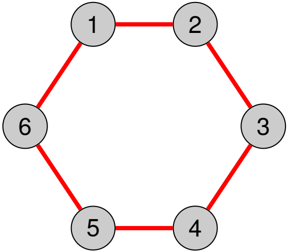
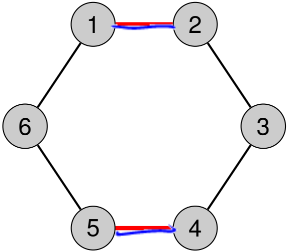
Outline

Communication Models: Diffusion vs. Matching

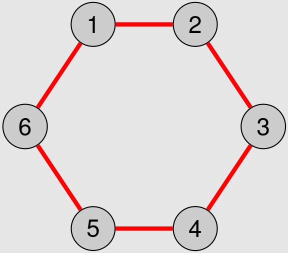




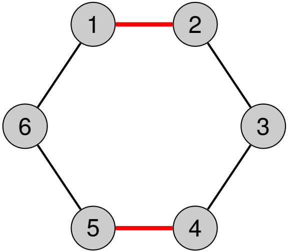
$$\mathbf{M} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

$$\mathbf{M}^{(t)} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Communication Models: Diffusion vs. Matching



$$\mathbf{M} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$



$$\mathbf{M}^{(t)} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Smoothness of the Load Distribution

- let $x \in \mathbb{R}^n$ be a load vector
- $\overline{x}$ denotes the average load

Smoothness of the Load Distribution

- let $x^t \in \mathbb{R}^n$ be a load vector at round t
- $\overline{x}$ denotes the average load

Smoothness of the Load Distribution

- let $x^t \in \mathbb{R}^n$ be a load vector at round t
- $\overline{x}$ denotes the average load

Metrics

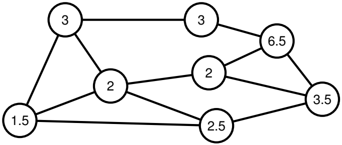
- ℓ_2-norm: $\Phi^t = \sqrt{\sum_{i=1}^n (x_i^t - \overline{x})^2}$
- makespan: $\max_{i=1}^n x_i^t$
- discrepancy: $\max_{i=1}^n x_i^t - \min_{i=1}^n x_i^t$. $\leq 2\Phi^t$

Smoothness of the Load Distribution

- let $x^t \in \mathbb{R}^n$ be a load vector at round t
- $\overline{x}$ denotes the average load

Metrics

- ℓ_2-norm: $\Phi^t = \sqrt{\sum_{i=1}^n (x_i^t - \overline{x})^2}$
- makespan: $\max_{i=1}^n x_i^t$
- discrepancy: $\max_{i=1}^n x_i^t - \min_{i=1}^n x_i$.

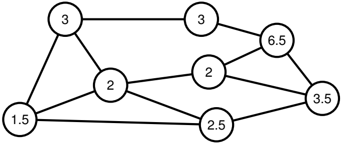


Smoothness of the Load Distribution

- let $x^t \in \mathbb{R}^n$ be a load vector at round t
- $\overline{x}$ denotes the average load

Metrics

- ℓ_2-norm: $\Phi^t = \sqrt{\sum_{i=1}^n (x_i^t - \overline{x})^2}$
- makespan: $\max_{i=1}^n x_i^t$
- discrepancy: $\max_{i=1}^n x_i^t - \min_{i=1}^n x_i$.



For this example:

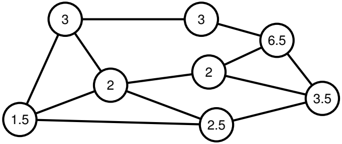
- $\Phi^t = \sqrt{0^2 + 0^2 + 3.5^2 + 0.5^2 + 1^2 + 1^2 + 1.5^2 + 0.5^2} = \sqrt{17}$
- $\max_{i=1}^n x_i^t = 6.5$
- $\max_{i=1}^n x_i^t - \min_{i=1}^n x_i^t = 5$

Smoothness of the Load Distribution

- let $x^t \in \mathbb{R}^n$ be a load vector at round t
- $\overline{x}$ denotes the average load

Metrics

- ℓ_2-norm: $\Phi^t = \sqrt{\sum_{i=1}^n (x_i^t - \overline{x})^2}$
- makespan: $\max_{i=1}^n x_i^t$
- discrepancy: $\max_{i=1}^n x_i^t - \min_{i=1}^n x_i$.



For this example:

- $\Phi^t = \sqrt{0^2 + 0^2 + 3.5^2 + 0.5^2 + 1^2 + 1^2 + 1.5^2 + 0.5^2} = \sqrt{17}$
- $\max_{i=1}^n x_i^t = 6.5$
- $\max_{i=1}^n x_i^t - \min_{i=1}^n x_i^t = 5$

Diffusion Matrix

Diffusion Matrix

Given an undirected, connected graph $G = (V, E)$ and a diffusion parameter $\alpha > 0$, the diffusion matrix M is defined as follows:

$$M_{ij} = \begin{cases} \alpha & \text{if } (i,j) \in E, \\ 1 - \alpha \deg(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Diffusion Matrix

How to choose α for a d-regular graph?

- $\alpha = \frac{1}{d}$ may lead to oscillation (if graph is bipartite)
- $\alpha = \frac{1}{d+1}$ ensures convergence
- $\alpha = \frac{1}{2d}$ ensures convergence (and all eigenvalues of M are non-negative)

Diffusion Matrix

Given an undirected, connected graph $G = (V, E)$ and a diffusion parameter $\alpha > 0$, the diffusion matrix M is defined as follows:

$$M_{ij} = \begin{cases} \alpha & \text{if } (i,j) \in E, \\ 1 - \alpha \deg(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Diffusion Matrix

Diffusion Matrix

Given an undirected, connected graph $G = (V, E)$ and a diffusion parameter $\alpha > 0$, the diffusion matrix M is defined as follows:

$$M_{ij} = \begin{cases} \alpha & \text{if } (i,j) \in E, \\ 1 - \alpha \underbrace{\deg(i)}_{\text{\# neighbors of } i} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Diffusion Matrix

Diffusion Matrix

Given an undirected, connected graph $G = (V, E)$ and a diffusion parameter $\alpha > 0$, the diffusion matrix M is defined as follows:

$$M_{ij} = \begin{cases} \alpha & \text{if } (i,j) \in E, \\ 1 - \alpha \deg(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Further let $\gamma(M) := \boxed{\max_{\mu_i \neq 1} |\mu_i|}$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M.

Diffusion Matrix

Diffusion Matrix

Given an undirected, connected graph $G = (V, E)$ and a diffusion parameter $\alpha > 0$, the diffusion matrix M is defined as follows:

$$M_{ij} = \begin{cases} \alpha & \text{if } (i,j) \in E, \\ 1 - \alpha \deg(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Further let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M.

First-Order Diffusion: Load vector x^t satisfies

$$x^t = M \cdot x^{t-1}.$$

Diffusion Matrix

Diffusion Matrix

Given an undirected, connected graph $G = (V, E)$ and a diffusion parameter $\alpha > 0$, the diffusion matrix M is defined as follows:

$$M_{ij} = \begin{cases} \alpha & \text{if } (i,j) \in E, \\ 1 - \alpha \deg(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Further let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M.

This can be also seen as a random walk on G!

First-Order Diffusion: Load vector x^t satisfies

$$x^t = M \cdot x^{t-1}.$$

$\overline{x} = M \cdot \overline{x}$

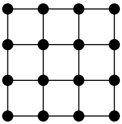
1D grid

$$\gamma(M) \approx 1 - \frac{1}{n^2}$$

1D grid

$$\gamma(M) \approx 1 - \frac{1}{n^2}$$

2D grid

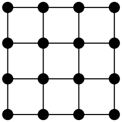


$$\gamma(M) \approx 1 - \frac{1}{n}$$

1D grid

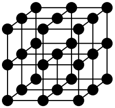
$\gamma(M) \approx 1 - \frac{1}{n^2}$

2D grid



$\gamma(M) \approx 1 - \frac{1}{n}$

3D grid

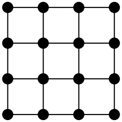


$\gamma(M) \approx 1 - \frac{1}{n^{2/3}}$

1D grid

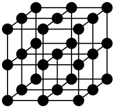
$\gamma(M) \approx 1 - \frac{1}{n^2}$

2D grid



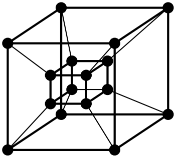
$\gamma(M) \approx 1 - \frac{1}{n}$

3D grid



$\gamma(M) \approx 1 - \frac{1}{n^{2/3}}$

Hypercube

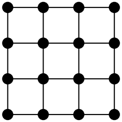


$\gamma(M) \approx 1 - \frac{1}{\log n}$

1D grid

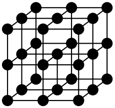
$$\gamma(M) \approx 1 - \frac{1}{n^2}$$

2D grid



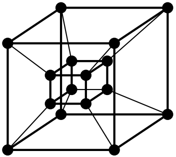
$$\gamma(M) \approx 1 - \frac{1}{n}$$

3D grid



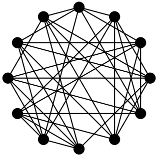
$$\gamma(M) \approx 1 - \frac{1}{n^{2/3}}$$

Hypercube



$$\gamma(M) \approx 1 - \frac{1}{\log n}$$

Random Graph

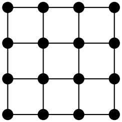


$$\gamma(M) < 1$$

1D grid

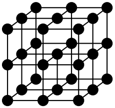
$\gamma(M) \approx 1 - \frac{1}{n^2}$

2D grid



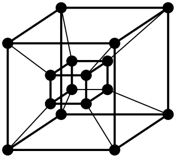
$\gamma(M) \approx 1 - \frac{1}{n}$

3D grid



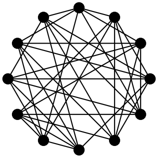
$\gamma(M) \approx 1 - \frac{1}{n^{2/3}}$

Hypercube



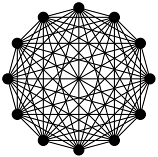
$\gamma(M) \approx 1 - \frac{1}{\log n}$

Random Graph



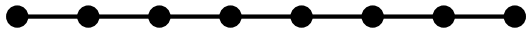
$\gamma(M) < 1$

Complete Graph



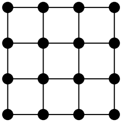
$\gamma(M) \approx 0$

1D grid



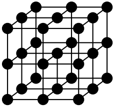
$\gamma(M) \approx 1 - \frac{1}{n^2}$

2D grid



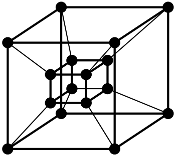
$\gamma(M) \approx 1 - \frac{1}{n}$

3D grid



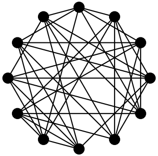
$\gamma(M) \approx 1 - \frac{1}{n^{2/3}}$

Hypercube



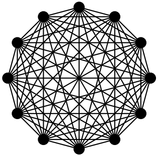
$\gamma(M) \approx 1 - \frac{1}{\log n}$

Random Graph



$\gamma(M) < 1$

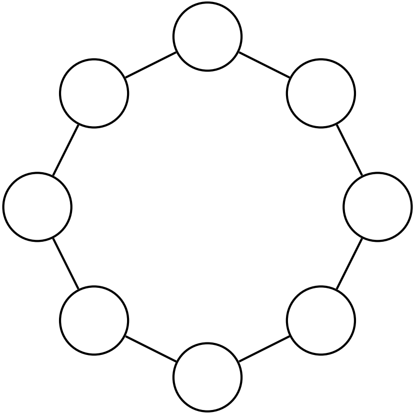
Complete Graph



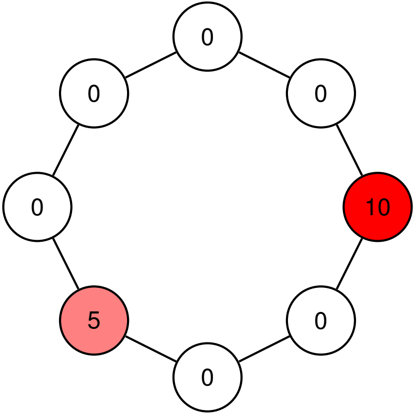
$\gamma(M) \approx 0$

$\gamma(M) \in (0,1]$ measures connectivity of G

Diffusion on a Ring

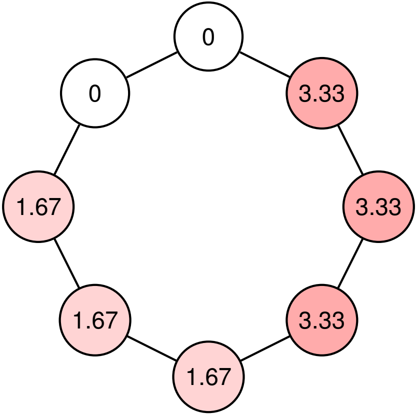


Diffusion on a Ring



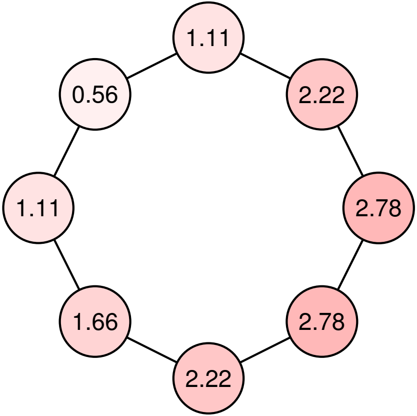
Diffusion on a Ring

after iteration 1:



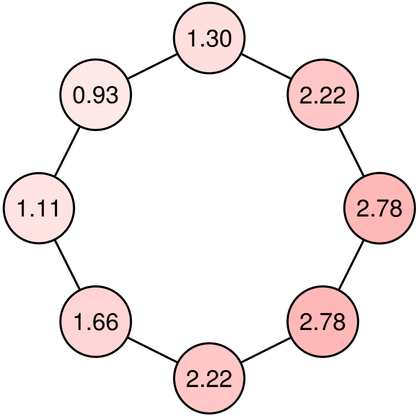
Diffusion on a Ring

after iteration 2:



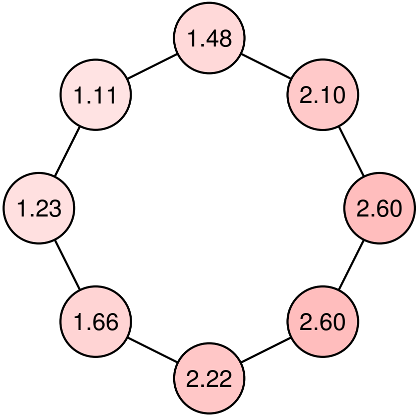
Diffusion on a Ring

after iteration 3:



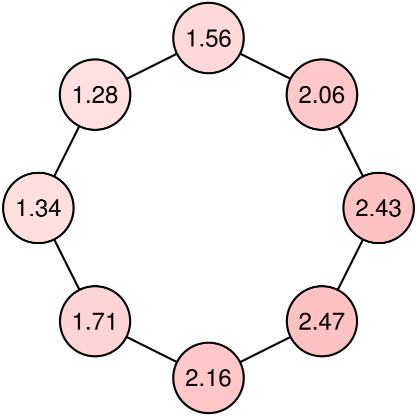
Diffusion on a Ring

after iteration 4:



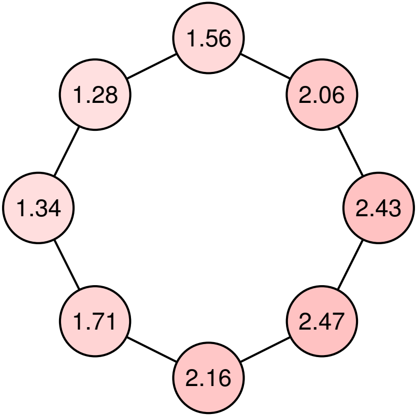
Diffusion on a Ring

after iteration 5:



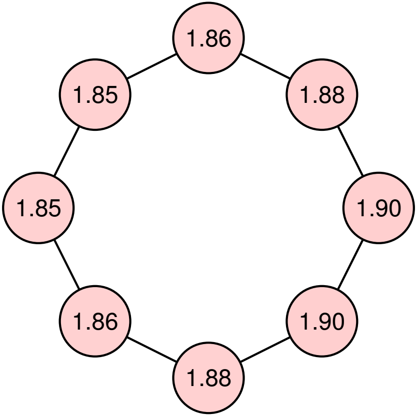
Diffusion on a Ring

after iteration 20:



Diffusion on a Ring

after iteration 20:



Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n$$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n = \boxed{\sum_{i=2}^{n} \alpha_i \cdot v_i}.$$

$\alpha_1 = 0$ ... $\langle e^t, v_1 \rangle = 0$

e^t is orthogonal to v_1

$v_1 = \overline{x}$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n = \sum_{i=2}^{n} \alpha_i \cdot v_i.$$

e^t is orthogonal to v_1

- For the diffusion scheme,

$$e^{t+1} = Me^t$$

$$e^{t+1} = x^{t+1} - \overline{x} = M \cdot x^t - M \cdot \overline{x}$$
$$= M \cdot (x^t - \overline{x}) = M \cdot e^t$$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n = \sum_{i=2}^{n} \alpha_i \cdot v_i.$$

e^t is orthogonal to v_1

- For the diffusion scheme,

$$e^{t+1} = Me^t = M \cdot \left(\sum_{i=2}^{n} \alpha_i v_i \right)$$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n = \sum_{i=2}^{n} \alpha_i \cdot v_i.$$

e^t is orthogonal to v_1

- For the diffusion scheme,

$$e^{t+1} = Me^t = M \cdot \left(\sum_{i=2}^{n} \alpha_i v_i \right) = \sum_{i=2}^{n} \alpha_i \mu_i v_i.$$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n = \sum_{i=2}^{n} \alpha_i \cdot v_i.$$

e^t is orthogonal to v_1

- For the diffusion scheme,

$$e^{t+1} = Me^t = M \cdot \left(\sum_{i=2}^{n} \alpha_i v_i \right) = \sum_{i=2}^{n} \alpha_i \mu_i v_i.$$

- Taking norms and using that the v_i's are orthogonal,

$$\|e^{t+1}\|_2 = \|Me^t\|_2$$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n = \sum_{i=2}^{n} \alpha_i \cdot v_i.$$

e^t is orthogonal to v_1

- For the diffusion scheme,

$$e^{t+1} = Me^t = M \cdot \left(\sum_{i=2}^{n} \alpha_i v_i \right) = \sum_{i=2}^{n} \alpha_i \mu_i v_i.$$

- Taking norms and using that the v_i's are orthogonal,

$$\|e^{t+1}\|_2 = \|Me^t\|_2 = \sum_{i=2}^{n} \alpha_i{}^2 \mu_i^2 \|v_i\|_2$$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n = \sum_{i=2}^{n} \alpha_i \cdot v_i.$$

e^t is orthogonal to v_1

- For the diffusion scheme,

$$e^{t+1} = Me^t = M \cdot \left(\sum_{i=2}^{n} \alpha_i v_i \right) = \sum_{i=2}^{n} \alpha_i \mu_i v_i.$$

- Taking norms and using that the v_i's are orthogonal,

$$\|e^{t+1}\|_2 = \|Me^t\|_2 = \sum_{i=2}^{n} {\alpha_i}^2 \mu_i^2 \|v_i\|_2 \leq \gamma^2 \sum_{i=2}^{n} {\alpha_i}^2 \|v_i\|_2$$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\boxed{\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.}$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n = \sum_{i=2}^{n} \alpha_i \cdot v_i.$$

e^t is orthogonal to v_1

- For the diffusion scheme,

$$e^{t+1} = Me^t = M \cdot \left(\sum_{i=2}^{n} \alpha_i v_i \right) = \sum_{i=2}^{n} \alpha_i \mu_i v_i.$$

- Taking norms and using that the v_i's are orthogonal,

$$\|e^{t+1}\|_2 = \|Me^t\|_2 = \sum_{i=2}^{n} \alpha_i{}^2 \mu_i^2 \|v_i\|_2 \leq \gamma^2 \sum_{i=2}^{n} \alpha_i{}^2 \|v_i\|_2 = \underline{\gamma^2 \cdot \|e^t\|_2}$$

Convergence of the Quadratic Error (Upper Bound)

Lemma

Let $\gamma(M) := \max_{\mu_i \neq 1} |\mu_i|$, where $\mu_1 = 1 > \mu_2 \geq \cdots \geq \mu_n \geq -1$ are the eigenvalues of M. Then for any iteration t,

$$\Phi^t \leq \gamma(M)^{2t} \cdot \Phi^0.$$

Proof:

- Let $e^t = x^t - \overline{x}$, where $\overline{x}$ is the column vector with all entries set to $\overline{x}$
- Express e^t through the orthogonal basis given by the eigenvectors of M:

$$e^t = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \cdots + \alpha_n \cdot v_n = \sum_{i=2}^{n} \alpha_i \cdot v_i.$$

e^t is orthogonal to v_1

- For the diffusion scheme,

$$e^{t+1} = Me^t = M \cdot \left(\sum_{i=2}^{n} \alpha_i v_i \right) = \sum_{i=2}^{n} \alpha_i \mu_i v_i.$$

- Taking norms and using that the v_i's are orthogonal,

$$\|e^{t+1}\|_2 = \|Me^t\|_2 = \sum_{i=2}^{n} \alpha_i{}^2 \mu_i^2 \|v_i\|_2 \leq \gamma^2 \sum_{i=2}^{n} \alpha_i{}^2 \|v_i\|_2 = \gamma^2 \cdot \|e^t\|_2 \qquad \square$$

Convergence of the Quadratic Error (Lower Bound)

(skipped)

Lemma

For any eigenvalue μ_i, $1 \leq i \leq n$, there is an initial load vector x^0 so that

$$\Phi^t = \mu_i^{2t} \cdot \Phi^0.$$

Convergence of the Quadratic Error (Lower Bound)

Lemma

For any eigenvalue μ_i, $1 \leq i \leq n$, there is an initial load vector x^0 so that

$$\Phi^t = \mu_i^{2t} \cdot \Phi^0.$$

Proof:

Convergence of the Quadratic Error (Lower Bound)

Lemma

For any eigenvalue μ_i, $1 \leq i \leq n$, there is an initial load vector x^0 so that

$$\Phi^t = \mu_i^{2t} \cdot \Phi^0.$$

Proof:

- Let $x^0 = \overline{x} + v_i$, where v_i is the eigenvector corresponding to μ_i

Convergence of the Quadratic Error (Lower Bound)

Lemma

For any eigenvalue μ_i, $1 \leq i \leq n$, there is an initial load vector x^0 so that

$$\Phi^t = \mu_i^{2t} \cdot \Phi^0.$$

Proof:

- Let $x^0 = \overline{x} + v_i$, where v_i is the eigenvector corresponding to μ_i
- Then

$$e^t = Me^{t-1} = M^t e^0 = M^t v_i = \mu_i^t v_i,$$

Convergence of the Quadratic Error (Lower Bound)

Lemma

For any eigenvalue μ_i, $1 \leq i \leq n$, there is an initial load vector x^0 so that

$$\Phi^t = \mu_i^{2t} \cdot \Phi^0.$$

Proof:

- Let $x^0 = \overline{x} + v_i$, where v_i is the eigenvector corresponding to μ_i
- Then

$$e^t = Me^{t-1} = M^t e^0 = M^t v_i = \mu_i^t v_i,$$

and

$$\Phi^t = \|e^t\|_2 = \mu_i^{2t} \|v_i\|_2 = \mu_i^{2t} \Phi^0.$$

Convergence of the Quadratic Error (Lower Bound)

Lemma

For any eigenvalue μ_i, $1 \leq i \leq n$, there is an initial load vector x^0 so that

$$\Phi^t = \mu_i^{2t} \cdot \Phi^0.$$

Proof:

- Let $x^0 = \overline{x} + v_i$, where v_i is the eigenvector corresponding to μ_i
- Then

$$e^t = Me^{t-1} = M^t e^0 = M^t v_i = \mu_i^t v_i,$$

and

$$\Phi^t = \|e^t\|_2 = \mu_i^{2t} \|v_i\|_2 = \mu_i^{2t} \Phi^0. \qquad \square$$

Outlook: Idealised versus Discrete Case

Idealised Case

Outlook: Idealised versus Discrete Case

Idealised Case

$$
\begin{aligned}
x^t &= M \cdot x^{t-1} \\
&= M^t \cdot x^0
\end{aligned}
$$

Outlook: Idealised versus Discrete Case

Idealised Case

$$\begin{aligned} x^t &= M \cdot x^{t-1} \\ &= M^t \cdot x^0 \end{aligned}$$

Linear System

- corresponds to Markov chain
- well-understood

Outlook: Idealised versus Discrete Case

Idealised Case

$$x^t = M \cdot x^{t-1} \\ = M^t \cdot x^0$$

Linear System

- corresponds to Markov chain
- well-understood

Given any load vector x^0, the number of iterations until x^t satisfies $\Phi^t \leq \epsilon$ is at most $\frac{\log(\Phi^0/\epsilon)}{1-\gamma(M)}$.

Outlook: Idealised versus Discrete Case

Here load consists of integers that cannot be divided further.

Idealised Case

$$\begin{aligned} x^t &= M \cdot x^{t-1} \\ &= M^t \cdot x^0 \end{aligned}$$

Discrete Case

Linear System

- corresponds to Markov chain
- well-understood

Given any load vector x^0, the number of iterations until x^t satisfies $\Phi^t \leq \epsilon$ is at most $\frac{\log(\Phi^0/\epsilon)}{1-\gamma(M)}$.

Outlook: Idealised versus Discrete Case

Here load consists of integers that cannot be divided further.

Idealised Case

$$x^t = M \cdot x^{t-1}$$
$$= M^t \cdot x^0$$

Discrete Case

$$y^t = \underline{M \cdot y^{t-1}} + \boxed{\Delta^t}$$

Linear System

- corresponds to Markov chain
- well-understood

Given any load vector x^0, the number of iterations until x^t satisfies $\Phi^t \leq \epsilon$ is at most $\frac{\log(\Phi^0/\epsilon)}{1-\gamma(M)}$.

Outlook: Idealised versus Discrete Case

Here load consists of integers that cannot be divided further.

Idealised Case

$$x^t = M \cdot x^{t-1}$$
$$= M^t \cdot x^0$$

Discrete Case

Rounding Error

$$y^t = M \cdot y^{t-1} + \Delta^t$$

Linear System

- corresponds to Markov chain
- well-understood

Given any load vector x^0, the number of iterations until x^t satisfies $\Phi^t \leq \epsilon$ is at most $\frac{\log(\Phi^0/\epsilon)}{1-\gamma(M)}$.

Outlook: Idealised versus Discrete Case

Here load consists of integers that cannot be divided further.

Idealised Case

$$x^t = M \cdot x^{t-1} = M^t \cdot x^0$$

Discrete Case

Rounding Error

$$y^t = M \cdot y^{t-1} + \Delta^t = M^t \cdot y^0 + \sum_{s=1}^{t} M^{t-s} \cdot \Delta^s$$

Linear System

- corresponds to Markov chain
- well-understood

Given any load vector x^0, the number of iterations until x^t satisfies $\Phi^t \leq \epsilon$ is at most $\frac{\log(\Phi^0/\epsilon)}{1-\gamma(M)}$.

Outlook: Idealised versus Discrete Case

Here load consists of integers that cannot be divided further.

Idealised Case

$$x^t = M \cdot x^{t-1}$$
$$= M^t \cdot x^0$$

Linear System

- corresponds to Markov chain
- well-understood

Given any load vector x^0, the number of iterations until x^t satisfies $\Phi^t \leq \epsilon$ is at most $\frac{\log(\Phi^0/\epsilon)}{1-\gamma(M)}$.

Discrete Case

Rounding Error

$$y^t = M \cdot y^{t-1} + \Delta^t$$
$$= M^t \cdot y^0 + \sum_{s=1}^{t} M^{t-s} \cdot \Delta^s$$

Non-Linear System

- rounding of a Markov chain
- harder to analyze

Outlook: Idealised versus Discrete Case

Here load consists of integers that cannot be divided further.

Idealised Case

$$x^t = M \cdot x^{t-1}$$
$$= M^t \cdot x^0$$

Linear System

- corresponds to Markov chain
- well-understood

Given any load vector x^0, the number of iterations until x^t satisfies $\Phi^t \leq \epsilon$ is at most $\frac{\log(\Phi^0/\epsilon)}{1-\gamma(M)}$.

Discrete Case

Rounding Error

$$y^t = M \cdot y^{t-1} + \underline{\Delta^t}$$
$$= M^t \cdot y^0 + \sum_{s=1}^{t} M^{t-s} \cdot \Delta^s$$

Non-Linear System

- rounding of a Markov chain
- harder to analyze

How close can it be made to the idealised case?

II. Matrix Multiplication

Thomas Sauerwald

Easter 2015

UNIVERSITY OF CAMBRIDGE

Outline

Matrix Multiplication

Remember: If $A = (a_{ij})$ and $B = (b_{ij})$ are square $n \times n$ matrices, then the matrix product $C = A \cdot B$ is defined by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \qquad \forall i, j = 1, 2, \ldots, n.$$

Matrix Multiplication

Remember: If $A = (a_{ij})$ and $B = (b_{ij})$ are square $n \times n$ matrices, then the matrix product $C = A \cdot B$ is defined by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \qquad \forall i, j = 1, 2, \ldots, n.$$

```
SQUARE-MATRIX-MULTIPLY(A, B)
n = A.rows
let C be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

Matrix Multiplication

Remember: If $A = (a_{ij})$ and $B = (b_{ij})$ are square $n \times n$ matrices, then the matrix product $C = A \cdot B$ is defined by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \qquad \forall i, j = 1, 2, \ldots, n.$$

```
SQUARE-MATRIX-MULTIPLY(A, B)
n = A.rows
let C be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

SQUARE-MATRIX-MULTIPLY(A, B) takes time $\Theta(n^3)$.

Matrix Multiplication

Remember: If $A = (a_{ij})$ and $B = (b_{ij})$ are square $n \times n$ matrices, then the matrix product $C = A \cdot B$ is defined by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \qquad \forall i, j = 1, 2, \ldots, n.$$

This definition suggests that $n \cdot n^2 = n^3$ arithmetic operations are necessary.

SQUARE-MATRIX-MULTIPLY(A, B)

```
n = A.rows
let C be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

SQUARE-MATRIX-MULTIPLY(A, B) takes time $\Theta(n^3)$.

Outline

Divide & Conquer: First Approach

Assumption: n is always an exact power of 2.

Divide & Conquer: First Approach

> **Assumption:** n is always an exact power of 2.

Divide & Conquer:
Partition A, B, and C into four $n/2 \times n/2$ matrices:

Divide & Conquer: First Approach

Assumption: n is always an exact power of 2.

Divide & Conquer:
Partition A, B, and C into four $n/2 \times n/2$ matrices:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}.$$

Divide & Conquer: First Approach

Assumption: n is always an exact power of 2.

Divide & Conquer:
Partition A, B, and C into four $n/2 \times n/2$ matrices:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}.$$

Hence the equation $C = A \cdot B$ becomes:

Divide & Conquer: First Approach

Assumption: n is always an exact power of 2.

Divide & Conquer:
Partition A, B, and C into four $n/2 \times n/2$ matrices:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}.$$

Hence the equation $C = A \cdot B$ becomes:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Divide & Conquer: First Approach

Assumption: n is always an exact power of 2.

Divide & Conquer:
Partition A, B, and C into four $n/2 \times n/2$ matrices:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}.$$

Hence the equation $C = A \cdot B$ becomes:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

This corresponds to the four equations:

$$\begin{aligned} C_{11} &= A_{11} \cdot B_{11} + A_{12} \cdot B_{21} \\ C_{12} &= A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ C_{21} &= A_{21} \cdot B_{11} + A_{22} \cdot B_{21} \\ C_{22} &= A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{aligned}$$

Divide & Conquer: First Approach

Assumption: n is always an exact power of 2.

Divide & Conquer:
Partition A, B, and C into four $n/2 \times n/2$ matrices:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}.$$

Hence the equation $C = A \cdot B$ becomes:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

This corresponds to the four equations:

$$\begin{aligned} C_{11} &= A_{11} \cdot B_{11} + A_{12} \cdot B_{21} \\ C_{12} &= A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ C_{21} &= A_{21} \cdot B_{11} + A_{22} \cdot B_{21} \\ C_{22} &= A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{aligned}$$

Each equation specifies two multiplications of $n/2 \times n/2$ matrices and the addition of their products.

Divide & Conquer: First Approach (Pseudocode)

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{11} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Divide & Conquer: First Approach (Pseudocode)

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c11 = a11 · b11
else partition A, B, and C as in equations (4.9)
    C11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B21)
    C12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B22)
    C21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B21)
    C22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B22)
return C
```

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{11} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Divide & Conquer: First Approach (Pseudocode)

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
1   n = A.rows
2   let C be a new n × n matrix
3   if n == 1
4       c_11 = a_11 · b_11
5   else partition A, B, and C as in equations (4.9)
6       C_11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_11)
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_21)
7       C_12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_12)
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_22)
8       C_21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_11)
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_21)
9       C_22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_12)
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_22)
10  return C
```

Line 5: Handle submatrices implicitly through index calculations instead of creating them.

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{11} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Divide & Conquer: First Approach (Pseudocode)

$\textsc{Square-Matrix-Multiply-Recursive}(A, B)$

$n = A.rows$
let C be a new $n \times n$ matrix
if $n == 1$
  $c_{11} = a_{11} \cdot b_{11}$
else partition A, B, and C as in equations (4.9)
  $C_{11} = \textsc{Square-Matrix-Multiply-Recursive}(A_{11}, B_{11})$
    $+ \textsc{Square-Matrix-Multiply-Recursive}(A_{12}, B_{21})$
  $C_{12} = \textsc{Square-Matrix-Multiply-Recursive}(A_{11}, B_{12})$
    $+ \textsc{Square-Matrix-Multiply-Recursive}(A_{12}, B_{22})$
  $C_{21} = \textsc{Square-Matrix-Multiply-Recursive}(A_{21}, B_{11})$
    $+ \textsc{Square-Matrix-Multiply-Recursive}(A_{22}, B_{21})$
  $C_{22} = \textsc{Square-Matrix-Multiply-Recursive}(A_{21}, B_{12})$
    $+ \textsc{Square-Matrix-Multiply-Recursive}(A_{22}, B_{22})$
return C

Let $T(n)$ be the runtime of this procedure.

Divide & Conquer: First Approach (Pseudocode)

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c11 = a11 · b11
else partition A, B, and C as in equations (4.9)
    C11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B21)
    C12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B22)
    C21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B21)
    C22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B22)
return C
```

Let $T(n)$ be the runtime of this procedure. Then:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ & \text{if } n > 1. \end{cases}$$

Divide & Conquer: First Approach (Pseudocode)

SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)

$n = A.rows$
let C be a new $n \times n$ matrix
if $n == 1$
  $c_{11} = a_{11} \cdot b_{11}$
else partition A, B, and C as in equations (4.9)
  $C_{11} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{11})
    $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{21})
  $C_{12} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{12})
    $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{22})
  $C_{21} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{11})
    $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{21})
  $C_{22} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{12})
    $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{22})
return C

Let $T(n)$ be the runtime of this procedure. Then:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ & \text{if } n > 1. \end{cases}$$

8 Multiplications

Divide & Conquer: First Approach (Pseudocode)

SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)

$n = A.rows$
let C be a new $n \times n$ matrix
if $n == 1$
    $c_{11} = a_{11} \cdot b_{11}$
else partition A, B, and C as in equations (4.9)
    C_{11} = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{11})
        + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{21})
    C_{12} = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{12})
        + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{22})
    C_{21} = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{11})
        + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{21})
    C_{22} = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{12})
        + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{22})
return C

Let $T(n)$ be the runtime of this procedure. Then:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 8 \cdot T(n/2) & \text{if } n > 1. \end{cases}$$

8 Multiplications

Divide & Conquer: First Approach (Pseudocode)

SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)

$n = A.rows$
let C be a new $n \times n$ matrix
if $n == 1$
    $c_{11} = a_{11} \cdot b_{11}$
else partition A, B, and C as in equations (4.9)
    $C_{11} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{11})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{21})
    $C_{12} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{12})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{22})
    $C_{21} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{11})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{21})
    $C_{22} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{12})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{22})
return C

Let $T(n)$ be the runtime of this procedure. Then:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 8 \cdot T(n/2) & \text{if } n > 1. \end{cases}$$

8 Multiplications

4 Additions and Partitioning

Divide & Conquer: First Approach (Pseudocode)

SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)

$n = A.rows$
let C be a new $n \times n$ matrix
if $n == 1$
    $c_{11} = a_{11} \cdot b_{11}$
else partition A, B, and C as in equations (4.9)
    $C_{11} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{11})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{21})
    $C_{12} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{12})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{12}, B_{22})
    $C_{21} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{11})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{21})
    $C_{22} =$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{21}, B_{12})
        $+$ SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{22}, B_{22})
return C

$$8^{\log_2 n} \cdot \Theta(1) = \Theta(n^3)$$

Let $T(n)$ be the runtime of this procedure. Then:

$$c \cdot n^2$$

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ \boxed{8} \cdot T(\boxed{n/2}) + \boxed{\Theta(n^2)} & \text{if } n > 1. \end{cases}$$

$$c \cdot n^2 + c \cdot 8 \cdot \left(\frac{n}{2}\right)^2 + c \cdot 8^2 \cdot \left(\frac{n}{4}\right)^2 + \ldots = \Theta(n^3)$$

8 Multiplications

4 Additions and Partitioning

Divide & Conquer: First Approach (Pseudocode)

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c_11 = a_11 · b_11
else partition A, B, and C as in equations (4.9)
    C_11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_11)
         + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_21)
    C_12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_12)
         + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_22)
    C_21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_11)
         + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_21)
    C_22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_12)
         + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_22)
return C
```

Let $T(n)$ be the runtime of this procedure. Then:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 8 \cdot T(n/2) + \Theta(n^2) & \text{if } n > 1. \end{cases}$$

Solution: $T(n) =$

Divide & Conquer: First Approach (Pseudocode)

SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)

```
n = A.rows
let C be a new n × n matrix
if n == 1
    c_11 = a_11 · b_11
else partition A, B, and C as in equations (4.9)
    C_11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_11)
             + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_21)
    C_12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_12)
             + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_22)
    C_21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_11)
             + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_21)
    C_22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_12)
             + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_22)
return C
```

Let $T(n)$ be the runtime of this procedure. Then:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 8 \cdot T(n/2) + \Theta(n^2) & \text{if } n > 1. \end{cases}$$

Solution: $T(n) = \Theta(8^{\log_2 n})$

Divide & Conquer: First Approach (Pseudocode)

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c11 = a11 · b11
else partition A, B, and C as in equations (4.9)
    C11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B21)
    C12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B22)
    C21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B21)
    C22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B22)
return C
```

Let $T(n)$ be the runtime of this procedure. Then:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 8 \cdot T(n/2) + \Theta(n^2) & \text{if } n > 1. \end{cases}$$

Solution: $T(n) = \Theta(8^{\log_2 n}) = \Theta(n^3)$

No improvement over the naive algorithm!