Computational Physics

(4)

(partial differential equations)

→ Hydrodynamics ⇝ Solution of PDEs

→ Machine learning ⇝ optimization (3)

→ simulate the mechanical behavior of a solid

→ phase transition
calculation of equilibrium properties

Newton's equations

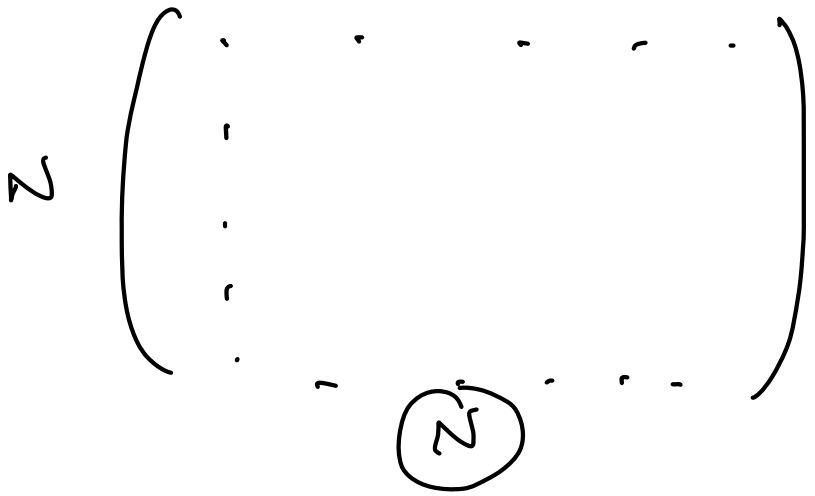
Monte Carlo method (2)

→ linear algebra, eigenvalues problems

large eigenvalue problems (1)

General feature of numerical problems: SCALING

How the computational time to solution grows with (1) the size of the problem



signalizing a matrix:

size = N

(2) (if uncertainty ε on the result)
with the uncertainty ε

Scaling: $T = f(N, \varepsilon)$

T: time to solution

Two categories of scaling

$\alpha > 0$

"BAD" SCALING: $T \sim \exp(A N^{\alpha})$

$$T \sim \exp\left(\frac{A}{\varepsilon^\alpha}\right)$$

"GOOD" SCALING: $T \sim \text{poly}_\alpha(N)$

$$O(N^\alpha)$$

$$T \propto \text{poly}_\alpha\left(\frac{1}{\varepsilon}\right)$$

Linear Algebra: eigenvalue problem

A $N \times N$ matrix

Hermitian

$$A^\dagger = A \qquad A_{ij} \in \mathbb{C}$$

$$(A^\dagger)_{ij} = A^*_{ji}$$

$$A \vec{x}_k = \lambda_k \vec{x}_k \qquad \lambda_k \in \mathbb{R}$$

$\vec{x}_k$: eigenvectors

λ_k: eigenvalue

$k = 1, \ldots, N$

$$U = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \vec{x}_1 & \vec{x}_2 & \cdots & \vec{x}_N \\ \vdots & \vdots & & \vdots \end{pmatrix}$$

$$U^+ U = \mathbb{1} \qquad \text{unitary}$$

$$U^+ A U = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$$

1) Quantum mechanics

$$\hat{O} |\psi_n\rangle = o_n |\psi_n\rangle$$

$\hat{O}$: observable

$o_n \in \mathbb{R}$

$|\psi_n\rangle$: eigenvectors $\in H$ Hilbert space

$$\hat{O} = \hat{O}^+$$

$\dim(H) = D$ finite-dimensional space

$\{|\phi_n\rangle\}$ basis of H

$n = 1, \ldots, D$

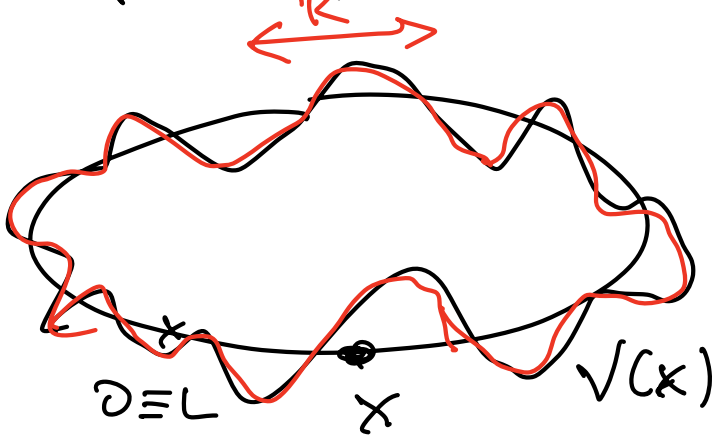
$$\sum_{n=1}^{D} |\phi_n\rangle\langle\phi_n| = \underline{\underline{1}}_{D\times D}$$

$$\sum_m \langle\phi_n|\hat{O}|\phi_m\rangle\langle\phi_m|\psi_\alpha\rangle = \langle\phi_n|O_\alpha|\psi_\alpha\rangle$$

$$\sum_m O_{nm}\, \psi_m^{(\alpha)} = O_\alpha\, \psi_n^{(\alpha)}$$

$$\leftrightarrow \quad \hat{O}\,\vec{\psi}_\alpha = O_\alpha\,\vec{\psi}_\alpha$$

Example: finding the energy spectrum of a particle in a ring



L: length of the ring

$R \gg \lambda_{min}$

$$\hat{H} = \frac{\hat{p}^2}{2M} + V(\hat{x})$$

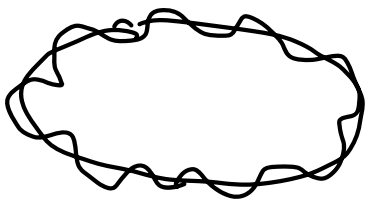
Basis for H of a particle on a ring

(1) momentum eigenstates

$$\hat{p}\,|q_n\rangle = \hbar q_n |q_n\rangle \quad , \quad \left(\hat{x}|x\rangle = x|x\rangle\right)$$

$$\langle x|q_n\rangle = \frac{1}{\sqrt{L}} e^{iq_n x}$$

↑ position eigenstate



$$q_n = \frac{2\pi}{L} n \qquad n \in \mathbb{Z}$$

Truncation of Hilbert space

$$|q_n| \leq q_{max} = \frac{2\pi}{L} n_0$$

$$|n| \leq n_0 \qquad n_0 > 0$$

$$D = 2n_0 + 1$$

$$\frac{\hat{p}^2}{2M}|q_n\rangle = \frac{\hbar^2 q_n^2}{2M}|q_n\rangle$$

$$\lambda_{min} : \quad q_{max} = \frac{2\pi}{\lambda_{min}}$$

$$\lambda_{min} = \frac{2\pi}{q_{max}} = \frac{L}{n_0}$$

$$\langle q_n | \hat{H} | q_m \rangle = H_{nm}$$

$$= \frac{\hbar^2 q_n^2}{2M} \delta_{nm} + \langle q_n | V(\hat{x}) | q_m \rangle$$

$$\Downarrow$$

$$\frac{1}{L} \int dx \; e^{-i(q_n - q_m)x} \; V(x)$$

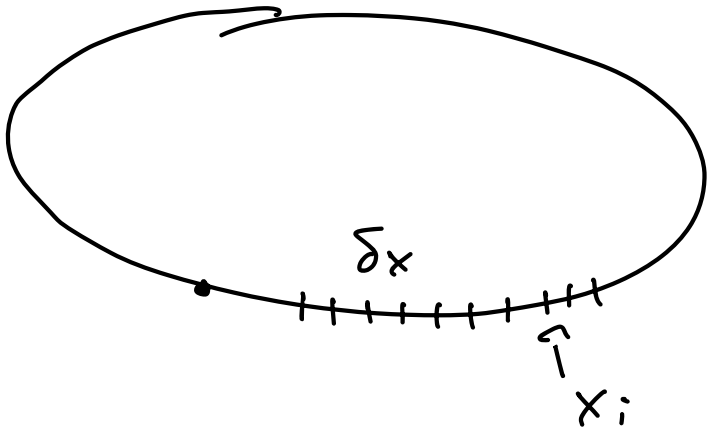
ft of V : $\tilde{V}(q_n - q_m)$

(2) position eigenbasis $|x\rangle$

$$\hat{H} |\psi\rangle = E |\psi\rangle$$

$$\Downarrow$$

$$\left[-\frac{\hbar^2}{2M} \frac{d^2}{dx^2} + V(x) \right] \psi(x) = E \psi(x)$$



$N \times N$

$N = \frac{L}{\delta x}$

$|x\rangle \rightarrow |x_i\rangle$

$\psi(x) \rightarrow \psi(x_i) = \psi_i$

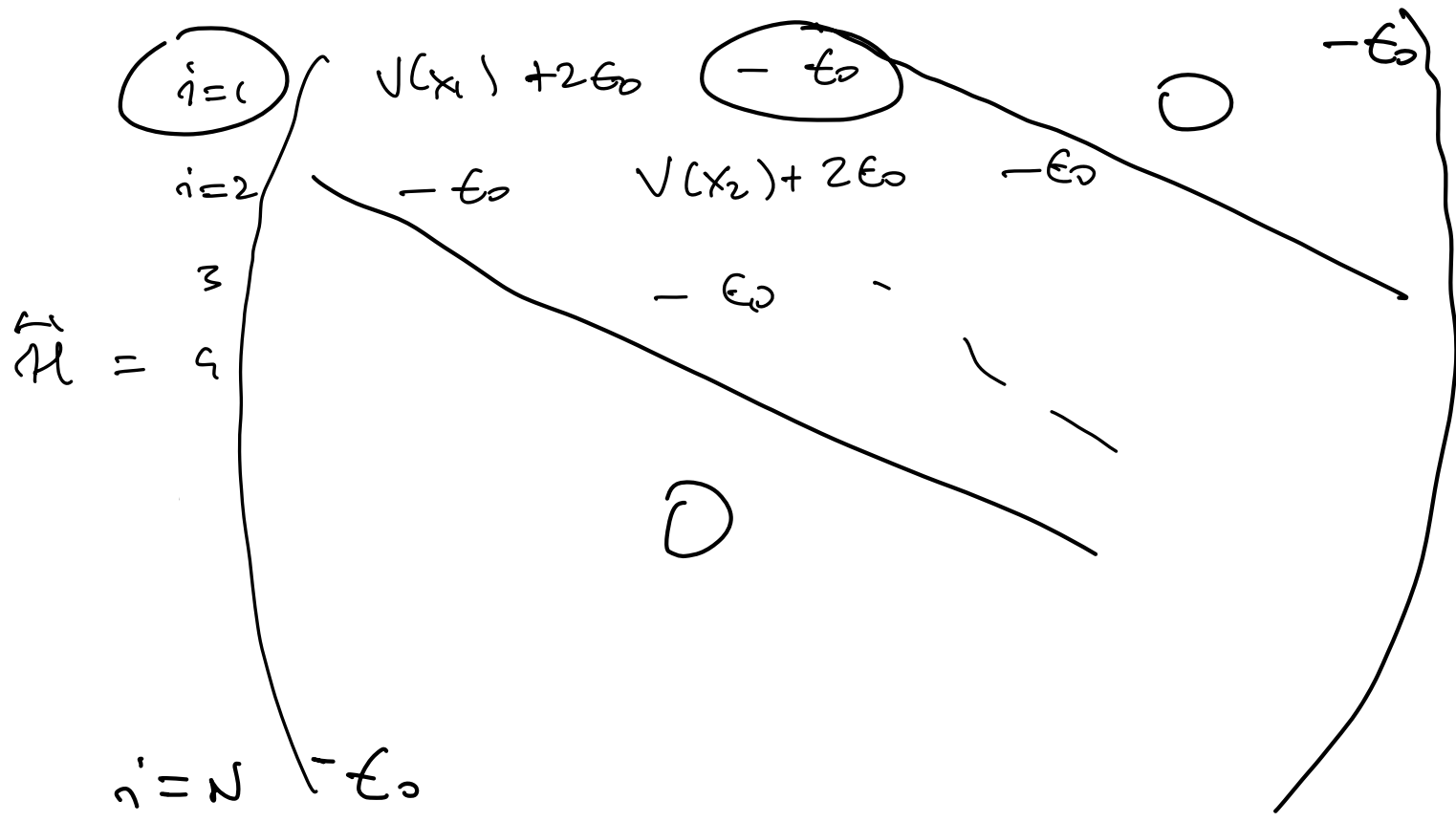
$$\left.\frac{d}{dx}\psi(x)\right|_{x_i} \simeq \frac{\psi(x_{i+1}) - \psi(x_i)}{\delta x}$$

$$\left.\frac{d^2}{dx^2}\psi(x)\right|_{x_i} \simeq \frac{\psi(x_{i+1}) + \psi(x_{i-1}) - 2\psi(x_i)}{(\delta x)^2}$$

ϵ_0

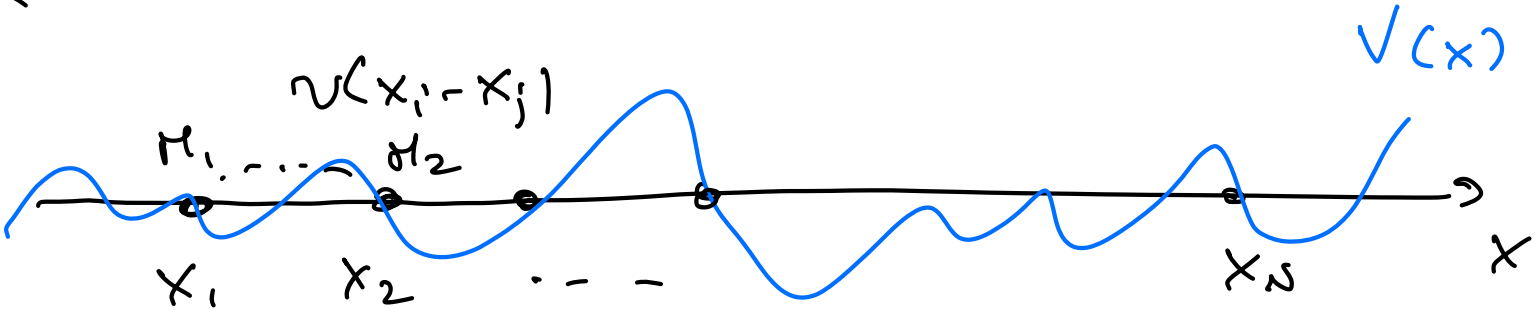
$$-\underbrace{\frac{\hbar^2}{2m(\delta x)^2}}_{=}\left[\psi(x_{i+1}) + \psi(x_{i-1}) - 2\psi(x_i)\right] + V(x_i)\,\psi(x_i) = E\,\psi(x_i)$$

$$\overleftrightarrow{H}\,\vec{\psi} = E\,\vec{\psi}$$



Classical mechanics

$\vec{x} = (x_1, x_2, \ldots x_N)$



$$U(\{x_i\}) = \sum_{i=1}^{N} V(x_i) + \frac{1}{2} \sum_{i \neq j} v(x_i - x_j)$$

$$\vec{\nabla}_{\vec{x}} U \Big|_{\vec{x} = \vec{x}^{col}} = 0$$

$$M_i \ddot{x}_i = -\frac{\partial U}{\partial x_i}$$

$$x_i = x_i^{(0)} + y_i$$

(equilibrium)

$$\ddot{y}_i = -\frac{\partial U}{\partial x_i}\Big|_{(\vec{x}=\vec{x}^{(0)})} - \sum_j \frac{\partial^2 U}{\partial x_i \partial x_j}\Big|_{\vec{x}=\vec{x}^{(0)}} (x_j - x_j^{(0)}) + \alpha y^2)$$

$\frac{\partial U}{\partial x_i} = 0$

H_{ij} Hessian matrix

y_i small

$$M_i \ddot{y}_i = -\sum_j H_{ij} y_j + O(y^2)$$

$$\rightarrow \ddot{y}_i = -\sum_j A_{ij} y_j \qquad (\ddot{y} = -\omega^2 y)$$

$$y_i = C_i^{(n)} e^{i\omega_n t}$$

$$-\omega_n^2 y_i = -\sum_j A_{ij} y_j \leftarrow$$

$$\vec{A}\vec{y} = \omega_n^2 \vec{y} \leftarrow$$

$\omega_n^2 \geq 0$ A semi-positive definite

$\Rightarrow$ stable minimum

it could happen that $\omega_n^2 < 0$ $\omega_n = \pm i \lambda_n$

$$\psi_i = c_i^{(n)} e^{i\omega_n t} = c_i^{(n)} e^{\pm \lambda_n t}$$

instability

FULL DIAGONALIZATION OF A MATRIX

LAPACK $\rightarrow$ Matlab
Mathematica
Python
...

QR algorithm for any square matrix NXN

$A = QR$

Q unitary

R is upper triangular

$O(N^3)$ operations

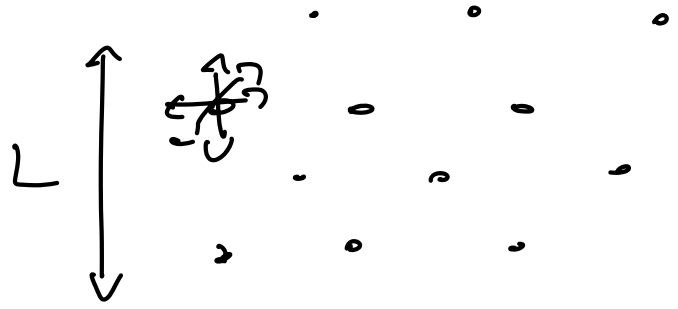
Memory storage : RAM

n Gbytes

to encode a number 8 bytes

$$N^2 \leq \frac{n \times 10^9}{8} \qquad n = 8$$

$$N \leq 10^{9/2} \sim 32000$$



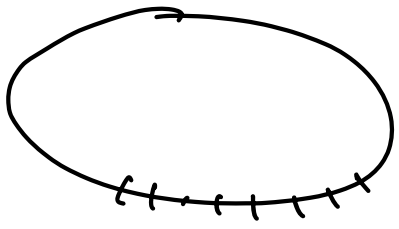
$$N = L^3$$

$$3N \qquad N \leq \frac{10^{9/2}}{3}$$

$$L = N^{1/3} \leq \sqrt[3]{\frac{10^{9/2}}{3}} \simeq 22$$

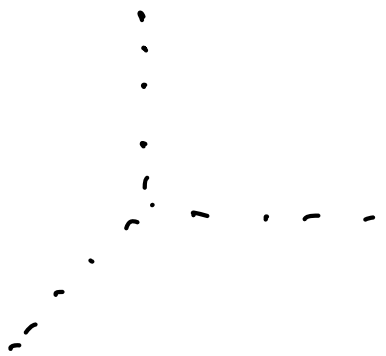
One quantum particle in $d = 1$

$$N \leq 32000$$



$$N = L \delta x$$

$d = 3$ L in each direction



$\dim(H) = L^3$

$N = L^3 \times L^3 \leq 32000$

$L \leq (32000)^{\frac{1}{6}} \approx 5.6$

2 particles

$N = (L^3 \times L^3)^2$

TRICKS :

1) Sometimes matrices we want to store are SPARSE : they contain a lot of zero elements

Ex. $N \times N$ matrix with $O(N)$ non-zero entries

2) Use symmetries

A : $N \times N$ matrix

V unitary $\quad V^\dagger V = \mathbb{1}$

Assumption 1

$V V^\dagger A V = V A$ $\qquad A V = V A$

$\mathbb{1}$ $\qquad [A, V] = 0$

A and V have the same eigenvectors

Assumption 2

V is simple to diagonalize

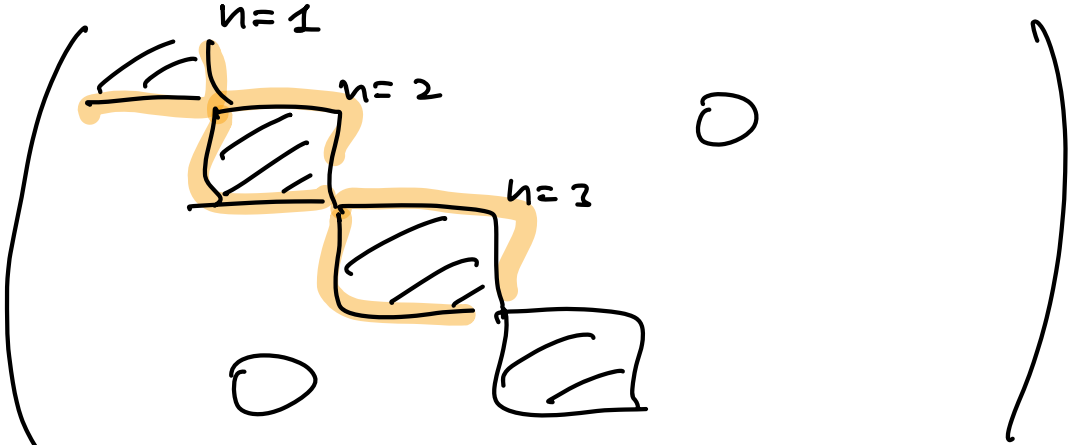
eigenvectors of V : $\vec{v}_k^{(n)}$ → $V^{(n)}$

eigenvector eigenvalue

degeneracy $\qquad k = 1, \dots, d_n$

$$A \Rightarrow \left(\vec{v}_k^{(n)}\right)^\dagger A \left(\vec{v}_{k'}^{(n')}\right) \sim \delta_{nn'}$$

$\Rightarrow$



We renounce to calculate the full spectrum, and focus on a subset of eigenvalues & eigenvectors

POWER METHOD

$\tilde{A}$: $\tilde{\lambda}_1, \tilde{\lambda}_2, \dots, \tilde{\lambda}_N$

$\tilde{\lambda}_1 < \tilde{\lambda}_2 < \dots < \tilde{\lambda}_N$

$\sigma > \tilde{\lambda}_N$

$\tilde{\lambda}_1 - \sigma < \tilde{\lambda}_2 - \sigma < \dots < \tilde{\lambda}_N - \sigma$

: $|\tilde{\lambda}_1 - \sigma| > |\tilde{\lambda}_2 - \sigma| > \dots > |\tilde{\lambda}_N - \sigma|$

$\lambda_i = \tilde{\lambda}_i - \sigma$ $\qquad$ $\lambda_1 < \lambda_2 < \dots < \lambda_N$

$(\tilde{A} - \sigma \mathbb{1})\vec{v}_i = \lambda_i \vec{v}_i$

A

random vector $\vec{u} = \sum_{n=1}^{N} c_n \vec{v}_n$

$\vec{u} \cdot \vec{v}_1 \neq 0$?

$$A^M \vec{u} = \sum_{n=1}^{N} c_n \lambda_n^M \vec{v}_n$$

$$= c_1 \lambda_1^M \vec{v}_1 + \sum_{n \geq 2} c_n \lambda_n^M \vec{v}_n$$

$$= c_1 \lambda_1^M \left[\vec{v}_1 + \sum_{n \geq 2} \frac{c_n}{c_1} \left(\frac{\lambda_n}{\lambda_1} \right)^M \vec{v}_n \right]$$

$\to 1$

$$\left(\frac{\lambda_n}{\lambda_1} \right)^M = \exp\left[- \left(\log \left| \frac{\lambda_1}{\lambda_n} \right| \right) M \right]$$

$$|\lambda_n| < |\lambda_1| \qquad \forall \; n \geq 2$$

$$A^M \vec{u} \longrightarrow \vec{v}_1$$

$$\| A^M \vec{u} \| \quad M \to \infty$$

$$=$$

$$\begin{pmatrix} & & \lambda_N \\ & & \\ \to \cdots\cdots & \epsilon & \left[(A - \epsilon)^2 - \epsilon^2 \mathbb{1} \right] \\ & & \\ & \lambda_1 & \end{pmatrix}$$

KRYLOV-SPACE METHODS: LANCZOS ALGORITHM

$$\{\vec{u},\ A\vec{u},\ A^2\vec{u},\ A^3\vec{u},\ \dots,\ A^{M-1}\vec{u}\}$$

M vectors

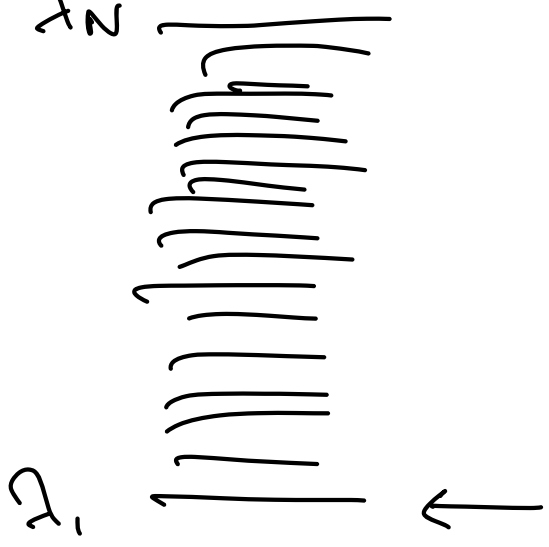
Krylov space

$$\mathcal{K}_M = \text{span}\{\vec{u},\ A\vec{u},\ \dots,\ A^{M-1}\vec{u}\}$$

M dimensional space

all linearly independent (unless $\vec{u}$ is an eigenvector of A)

$$M \leq N$$



orthonormalize the Krylov vectors

→ build a reduced $M \times M$ matrix

out of these vectors

Lanczos algorithm

Gram – Schmidt orthogonalization

$$\vec{u}_0 = \vec{u} \quad \text{(normalized)}$$

$$\beta_1 \vec{u}_1 = A\vec{u}_0 - (\vec{u}_0^{+} A\vec{u}_0)\vec{u}_0$$

β_1: normalization

$$\beta_2 \vec{u}_2 = A\vec{u}_1 - (\vec{u}_1^{+} A\vec{u}_1)\vec{u}_1 - (\vec{u}_0^{+} A\vec{u}_1)\vec{u}_0$$

$$\beta_3 \vec{u}_3 = A\vec{u}_2 - (\vec{u}_2^{+} A\vec{u}_2)\vec{u}_2 - (\vec{u}_1^{+} A\vec{u}_2)\vec{u}_1 - (\vec{u}_0^{+} A\vec{u}_2)\vec{u}_0$$

$$\vec{u}_2^{+} A\vec{u}_0 = \vec{u}_2^{+} \cdot (\beta_1 \vec{u}_1 + (\vec{u}_0^{+} A\vec{u}_0)\vec{u}_0)$$

...

$$\beta_{k+1}\vec{u}_{k+1} = A\vec{u}_k - (\vec{u}_k^{+} A\vec{u}_k)\vec{u}_k - (\vec{u}_{k-1}^{+} A\vec{u}_k)\vec{u}_{k-1}$$

M vectors → Lanczos basis

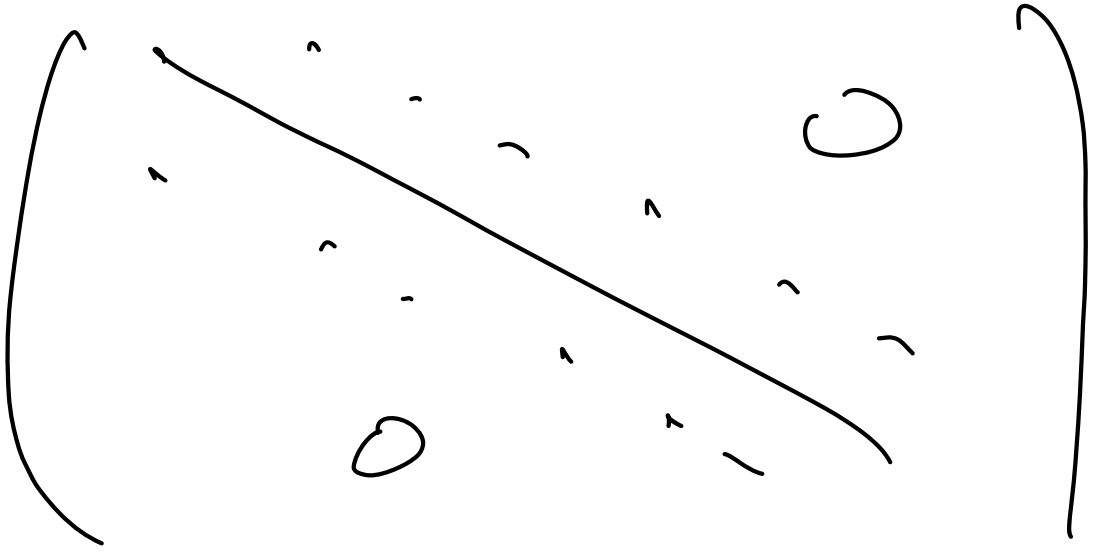
M×M reduced matrix

$$A_{ul} = \vec{u}_u^{+} A \vec{u}_l$$

$$= \vec{u}_u \cdot \left(\beta_{l+1} \vec{u}_{l+1} + (u_l^{+} A \vec{u}_l) \vec{u}_l + (\vec{u}_{l-1}^{+} A \vec{u}_l) u_{l-1} \right)$$

$$= \beta_{l+1} \delta_{u,l+1} + (u_l^{+} A u_l) \delta_{ul} + (\vec{u}_{l-1}^{+} A \vec{u}_l) \delta_{u,l-1}$$

→



tridiagonal matrix

algorithms with
$\rightarrow T \sim O(M^2)$
$O(M \log M)$

diagonalize the reduced matrix

$\lambda_1^{(M)}, \ldots, \lambda_M^{(M)}$

$\lambda_1^{(M)} \geq \lambda_1$

$\lambda_2^{(M)}, \lambda_3^{(M)}, \ldots$

$\forall \vec{v}: \dfrac{\vec{v}^+ A \vec{v}}{\|\vec{v}\|^2} \geq \lambda_1$