PHAS0102: Techniques of High-Performance Computing

Assignment 1 feedback: scope

Introduction to sparse solvers

$$Ax = b$$

(known) sparse matrix

(unknown) vector

(known) vector

- Sparse iterative solvers
- Sparse direct solvers

PyAMG

Scipy Trilinos PETSc Eigen

UMFPack Mumps SuperLU

```
from scipy.sparse import linalg

linalg.spsolve(A, b)
```

Introduction to sparse solvers

$$\mathrm{A}\mathbf{x} = \mathbf{b}$$

An iterative solver finds a series of approximate solutions $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \ldots$

- Residual $\mathbf{r}_m = \mathbf{b} - \mathrm{A}\mathbf{x}_m$

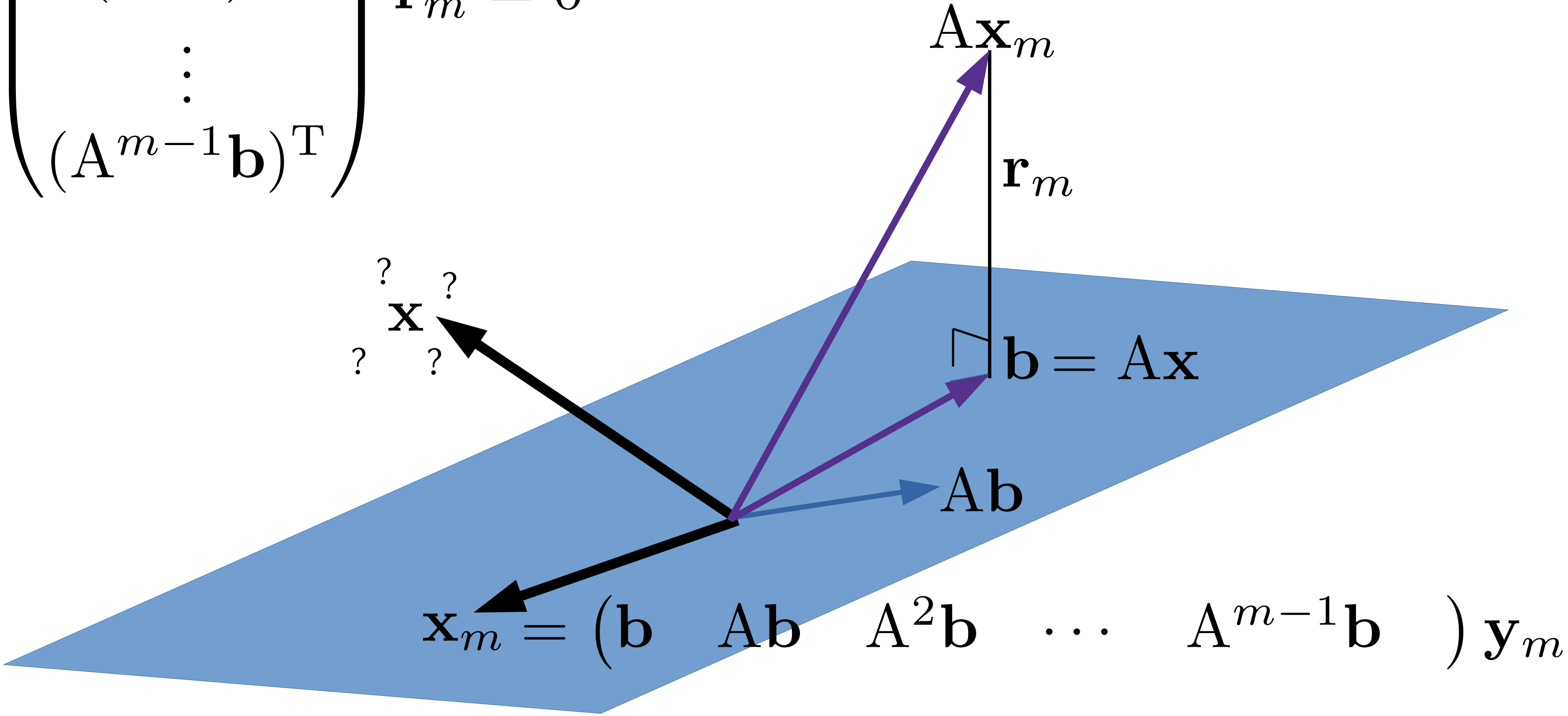
Sparse iterative solvers:
Krylov subspaces

The Krylov subspace is defined by

$$\mathcal{K}_m(\mathrm{A}, \mathbf{b}) := \mathrm{span}\{\mathbf{b}, \mathrm{A}\mathbf{b}, \mathrm{A}^2\mathbf{b}, ..., \mathrm{A}^{m-1}\mathbf{b}\}$$

Plan: look for an approximation of the solution in a Krylov subspace.

$$\begin{pmatrix} \mathbf{b}^{\mathrm{T}} \\ (\mathrm{A}\mathbf{b})^{\mathrm{T}} \\ (\mathrm{A}^2\mathbf{b})^{\mathrm{T}} \\ \vdots \\ (\mathrm{A}^{m-1}\mathbf{b})^{\mathrm{T}} \end{pmatrix} \mathbf{r}_m = 0$$



$$\underbrace{\begin{pmatrix} \mathbf{b}^{\mathrm{T}} \\ (\mathrm{A}\mathbf{b})^{\mathrm{T}} \\ (\mathrm{A}^2\mathbf{b})^{\mathrm{T}} \\ \vdots \\ (\mathrm{A}^{m-1}\mathbf{b})^{\mathrm{T}} \end{pmatrix}}_{\mathrm{V}_m^{\mathrm{T}}} \mathbf{r}_m = 0$$

$$\mathbf{x}_m = \underbrace{\begin{pmatrix} \mathbf{b} & \mathrm{A}\mathbf{b} & \mathrm{A}^2\mathbf{b} & \cdots & \mathrm{A}^{m-1}\mathbf{b} \end{pmatrix}}_{\mathrm{V}_m} \mathbf{y}_m$$

$$(\mathbf{b} - \mathrm{A}\mathbf{x}_m)$$

$$\mathrm{V}_m^{\mathrm{T}} \mathbf{r}_m = 0$$

$$\mathbf{x}_m = \mathrm{V}_m \mathbf{y}_m$$

$$V_m^T A \mathbf{x}_m = V_m^T \mathbf{b}$$

$$\mathbf{x}_m = V_m \mathbf{y}_m$$

$$\boxed{V_m^T A V_m} \mathbf{y}_m = V_m^T \mathbf{b}$$

m by *m* matrix

$$\mathbf{x}_m = V_m \mathbf{y}_m$$

[live Python demo]

Arnoldi iteration

$$\mathcal{K}_m(\mathrm{A}, \mathbf{b}) := \mathrm{span}\{\mathbf{b}, \mathrm{A}\mathbf{b}, \mathrm{A}^2\mathbf{b}, \ldots, \mathrm{A}^{m-1}\mathbf{b}\}$$

orthogonalise

This gives us the **full orthogonalisation method** (FOM) for solving A**x**=**b**.

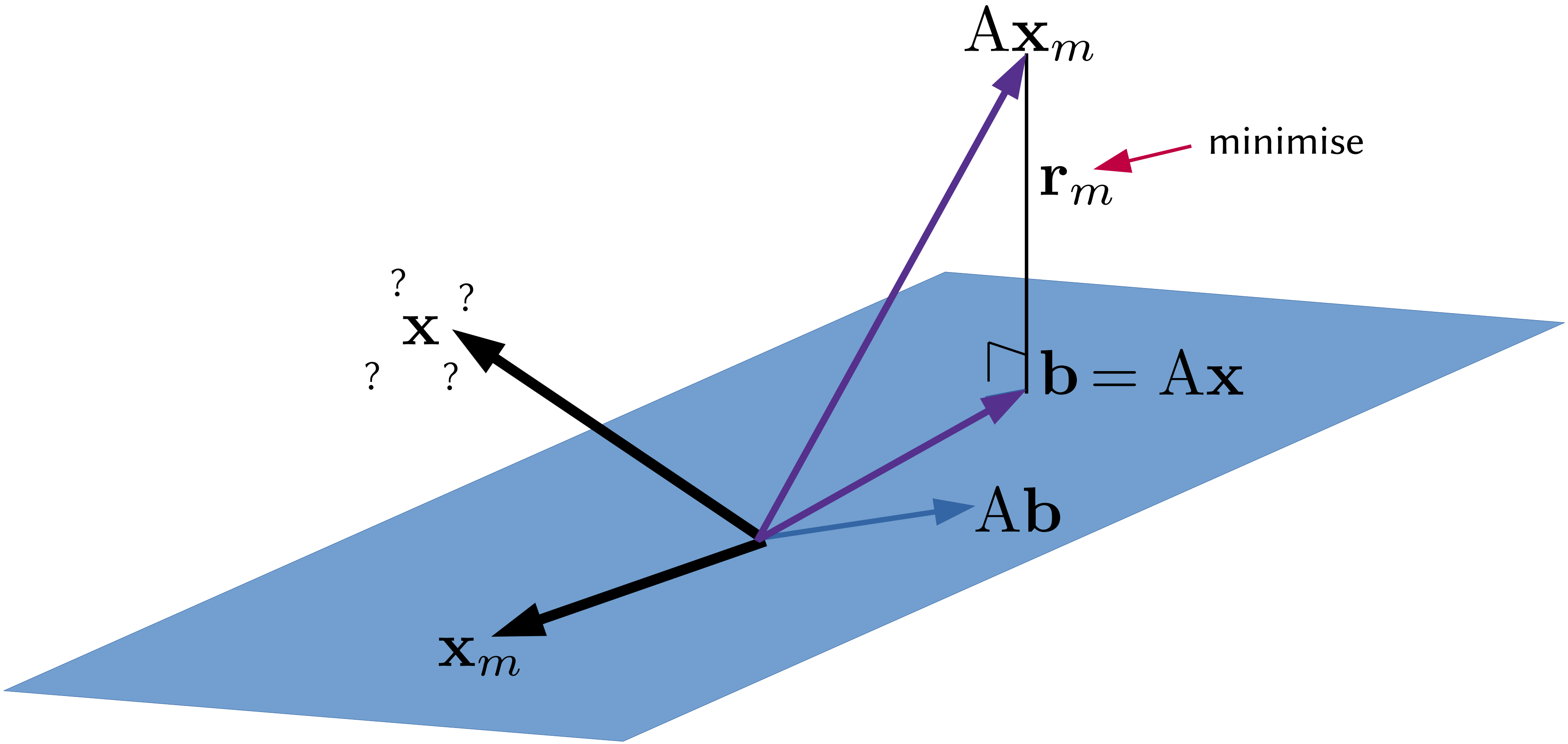
[live Python demo]

$$\mathrm{V}_m^{\mathrm{T}} \mathrm{V}_m$$

$$\begin{pmatrix} \mathbf{b}^{\mathrm{T}} \\ \mathrm{A}\mathbf{b}^{\mathrm{T}} \\ \mathrm{A}^2\mathbf{b}^{\mathrm{T}} \\ \vdots \\ \mathrm{A}^{m-1}\mathbf{b}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} \mathbf{b} & \mathrm{A}\mathbf{b} & \mathrm{A}^2\mathbf{b} & \cdots & \mathrm{A}^{m-1}\mathbf{b} \end{pmatrix}$$

GMRES

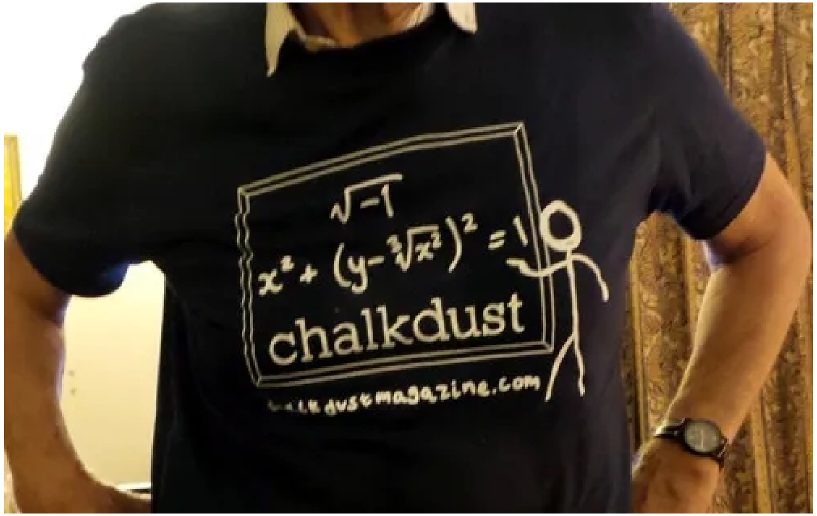
(Generalised Minimum RESidual method)



[live Python demo]

Reading week competition

- Write a function that computes 1000 digits of pi. Make it as fast as you can.
- I will run all the functions on the same computer and measure the time they take. Prize for the code that takes the least time.
- Deadline: Monday 14 November 5pm
- There is no course credit available for this, it’s just for fun / extra practice. **There is no penalty for not doing this.**



Fastest code wins a T-shirt!