Numerical Linear Algebra: from Scientific Computing to Data Science Applications

Yousef Saad
University of Minnesota

47th Annual Spring Lecture Series
University of Arkansas

May 4–6, 2022

This tutorial: Topics & Plan

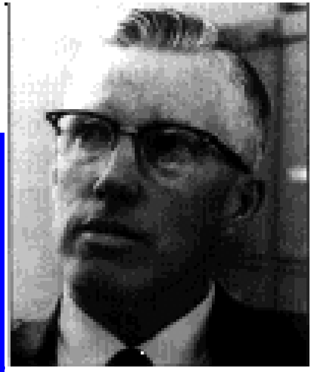
- Current state of advanced Numerical Linear Algebra including:
  - First part: Sparse large matrix problems, linear systems, eigenvalue problems
  - Second: data-related problems: graphs, dimension reduction, ...
  - Prerequisite: senior level course in numerical linear algebra
  - 5 lectures + Matlab demos
  - All materials posted here:

Schedule

Wed.	8:00– 9:00 am	Historical Perspective; Background & Examples; Sparsity; Data structures; Relaxation methods
Wed.	1:00 – 2:00 pm	Projection methods for lin. systems, Krylov methods Eigenvalue Pbs; Proj. Methods; Subs. it.; Lanczos
Thu.	8:00– 9:00 am	Backround on Graphs; Graph representations; Graphs for Data; Networks & Centrality; Graph Laplaceans.
Thu.	1:00 – 2:00 pm	Graph methods; Clustering; Segmentation; Graph embedding; Dimension Reduction; Informtion retrieval.
Fri.	8:00– 9:00 am	Supervised Learning; Neural Networks; Coarsening in scientific computing & in Data Sciences

Introduction: a historical perspective

In 1953, George Forsythe published a paper titled:
"Solving linear systems can be interesting".



- Survey of the state of the art linear algebra in early 50s: direct methods, iterative methods, conditioning, preconditioning, The Conjugate Gradient, acceleration methods, ...

➤ An amazing paper in which the author was urging researchers to start looking at solving linear systems

Introduction: a historical perspective

In 1953, George Forsythe published a paper titled:

"Solving linear systems can be interesting".

- Survey of the state of the art linear algebra in early 50s: direct methods, iterative methods, conditioning, preconditioning, The Conjugate Gradient, acceleration methods, ...

➤ An amazing paper in which the author was urging researchers to start looking at solving linear systems

➤ Nearly 70 years later – we can certainly state that:

"Linear Algebra problems in Machine Learning can be interesting"

Focus of numerical linear algebra changes over time

- Linear algebra took many direction changes in the past

1940s–1950s: Major issue: flutter problem in aerospace engineering $\rightarrow$ eigenvalue problem [cf. Olga Taussky Todd] $\rightarrow$ LR, QR, .. $\rightarrow$ 'EISPACK'

1960s: Problems related to the power grid promoted what we would call today general sparse matrix techniques

1970s– Automotive, Aerospace, ..: Computational Fluid Dynamics (CFD)

Late 1980s: Thrust on parallel matrix computations.

Late 1990s: Spur of interest in "financial computing"

Current: Machine Learning

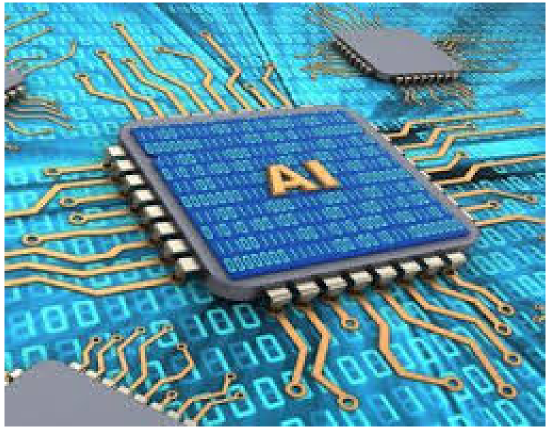
Solution of PDEs (e.g., Fluid Dynamics) and problems in mechanical eng. (e.g. structures) major force behind numerical linear algebra algorithms in the past few decades.

➤ Strong new forces are now reshaping the field today: Applications related to the use of “data”

➤ Machine learning is appearing in unexpected places:

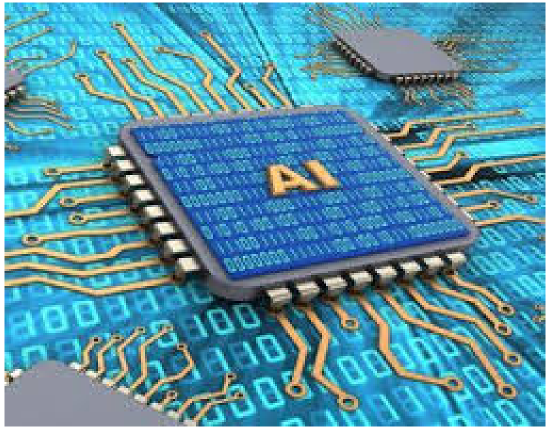
- design of materials
- machine learning in geophysics
- self-driving cars, ..
- ....

Big impact on the economy



- New economy driven by Google, Facebook, Netflix, Amazon, Twitter, Ali-Baba, Tencent, ..., and even the big department stores (Walmart, ...)
- Huge impact on **Jobs**

Big impact on the economy



➤ New economy driven by Google, Facebook, Netflix, Amazon, Twitter, Ali-Baba, Tencent, ..., and even the big department stores (Walmart, ...)

➤ Huge impact on **Jobs**

➤ Old leaders - e.g., Mining; Car companies; Aerospace; Manufacturing; offer little growth – Some instances of renewal driven by new technologies [e.g. Tesla]

➤ Look at what you are doing under new lenses: DATA

Ax=b

$-\Delta u = f$

Graph Partitioning

Preconditioning

Matlab, PETSc, ...

Model reduction

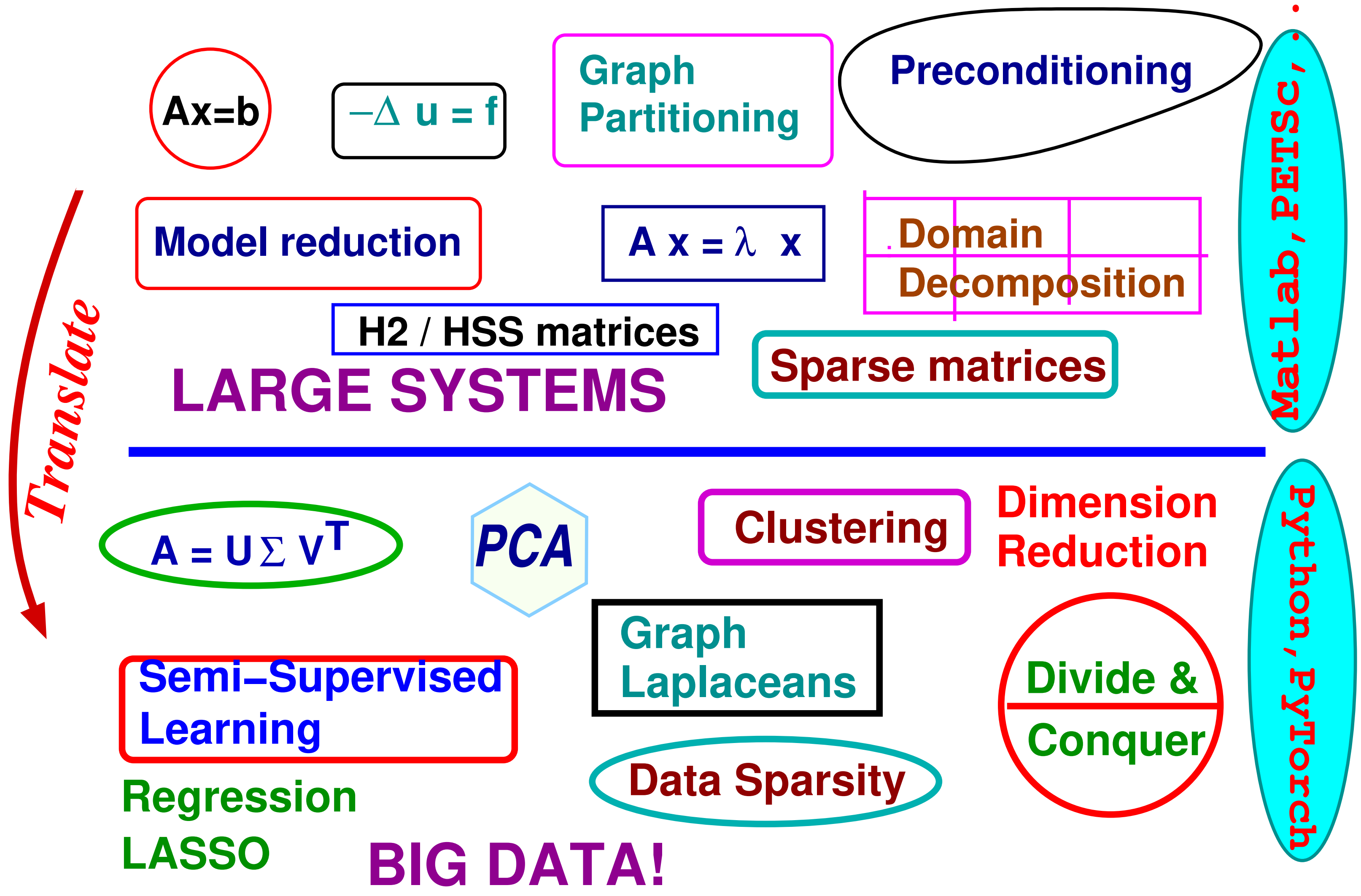
$A x = \lambda x$

Domain Decomposition

H2 / HSS matrices

Sparse matrices

LARGE SYSTEMS



Ax=b
$-\Delta u = f$
Graph Partitioning
Preconditioning
Matlab, PETSC, ..
Model reduction
$A x = \lambda x$
Domain Decomposition
H2 / HSS matrices
Sparse matrices
LARGE SYSTEMS
Translate
$A = U \Sigma V^T$
PCA
Clustering
Dimension Reduction
Python, PyTorch
Graph Laplaceans
Semi–Supervised Learning
Divide & Conquer
Data Sparsity
Regression
LASSO
BIG DATA!

Impact on what we teach...

- My course: *CSCI 8314: Sparse Matrix Computations*
  [url: my website - follow teaching]

... Has changed substantially in past 4-6 years

Before: —*PDEs, solving linear systems, Sparse direct solvers, Iterative methods, Krylov methods, Preconditioners, Multigrid,..*

$\longrightarrow$

Now: — a little of sparse direct methods + Applications of graphs, dimension reduction, Krylov methods.. Examples in: PCA, Information retrieval, Segmentation, Clustering, ...

General Introduction and Background

➤ This tutorial is about Numerical Linear Algebra – both the *classical* kind and the *new*:

- Standard matrix computations (e.g. solving linear systems, eigenvalue/SVD problems, ...)
- Graph algorithms and tools (Sparse graphs, graph coarsening, graphs and sparse methods). ..
- Dimension reduction methods; Graph embeddings;
- Specific machine learning algorithms; unsupervised/ supervised learning;
- Graph coarsening methods in scientific computing and machine learning

Example: Fluid flow

Physical Model

$\downarrow$

Nonlinear PDEs

$\downarrow$

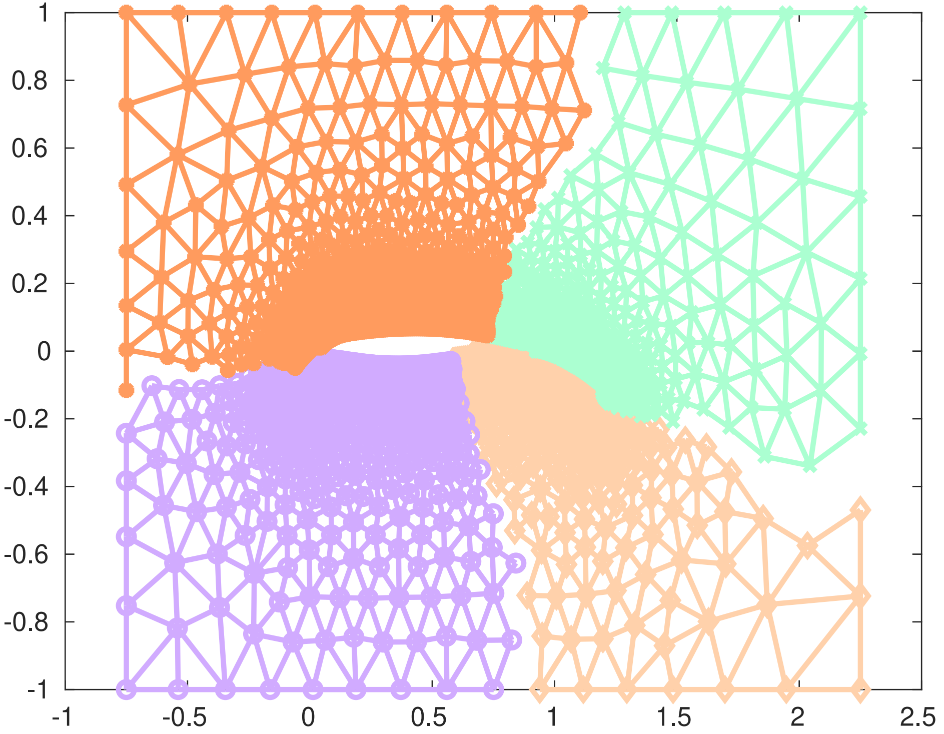
Discretization

$\downarrow$

Linearization (Newton)

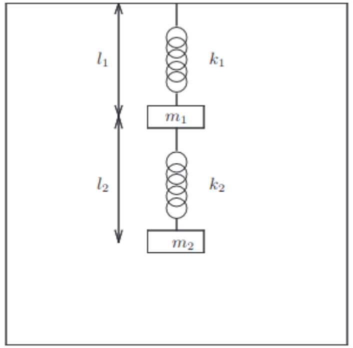
$\downarrow$

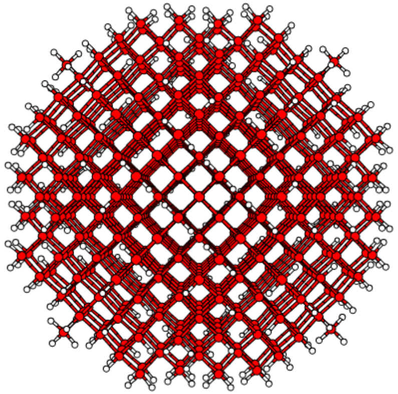
Sparse Linear Systems $Ax = b$

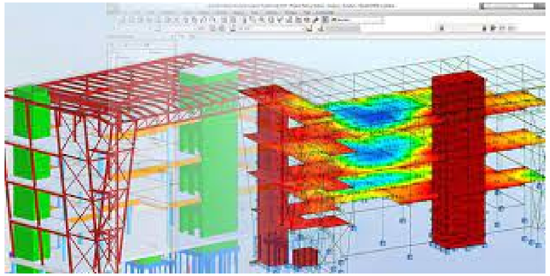


Example: Eigenvalue Problems

➤ Many applications require the computation of a few eigenvalues + associated eigenvectors of a matrix A







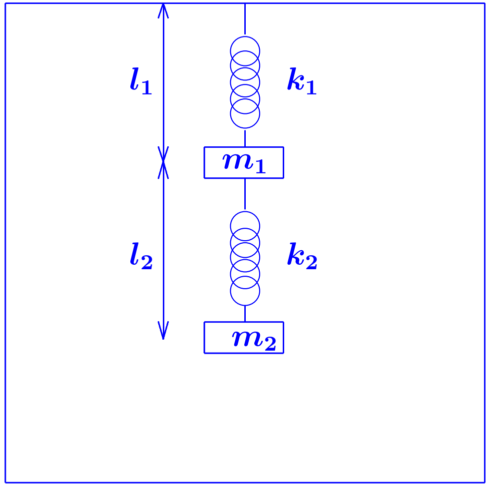
- Structural Engineering – (Goal: frequency response)
- Electronic structure calculations [Schrödinger equation..] – Quantum chemistry
- Stability analysis [e.g., electrical networks, mechanical system,..]
- ...

Example: Vibrations

- Vibrations in mechanical systems. See:

www.cs.umn.edu/~saad/eig_book_2ndEd.pdf

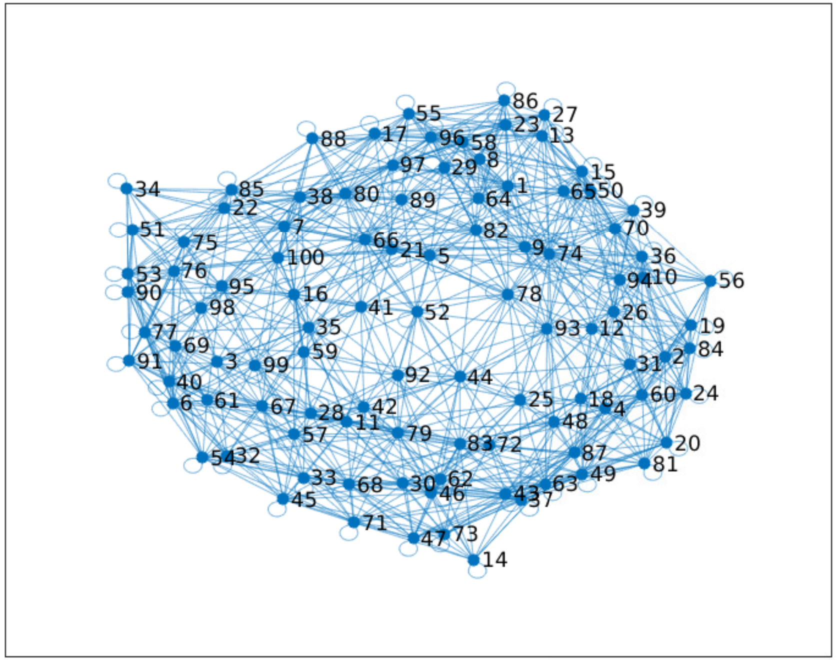
Problem: Determine the vibration modes of the mechanical system [to avoid resonance]. See details in Chapter 10 (sec. 10.2) of above reference.



- Problem type: Eigenvalue Problem

Example: Google Rank (pagerank)

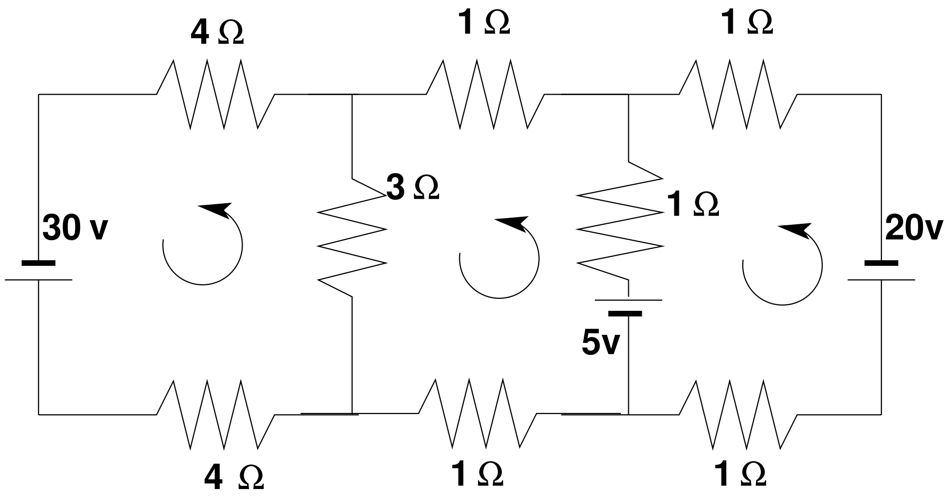
If one were to do a random walk from web page to web page, following each link on a given web page at random with equal likelihood, which are the pages to be encountered this way most often?



➤ Problem type: (homogeneous) Linear system. Eigenvector problem.

Example: Power networks

➤ Electrical circuits .. [Kirchhiff's voltage Law]



Problem: Determine the loop currents in a an electrical circuit - using Kirchhoff's Law ($V = RI$)

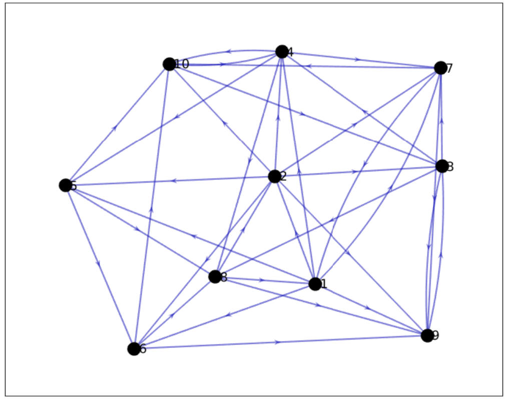
➤ Problem: Sparse Linear Systems [at the origin Sparse Direct Methods]

Example: Economics/ Marketing/ Social Networks

➤ Given: an influence graph G: g_{ij} = strength of influence of j over i

➤ Goal: charge member i price p_i in order to maximize profit

➤ Utility for member i: [x_i = consumption of i]



$$u_i = ax_i - bx_i^2 + \sum_{j \neq i} g_{ij} x_j - p_i x_i$$

- 1: 'Monopolist' fixes prices; 2: agent i fixes consumption x_i

Result: Optimal pricing proportional to Bonacich centrality: $(I - \alpha G)^{-1} \mathbb{1}$ where $\alpha = \frac{1}{2b}$ [*Candogan et al., 2012* + many refs.]

- ’centrality’ defines a measure of importance of a node (or an edge) in a graph
- Many other ideas of centrality in graphs [degree centrality, betweenness centrality, closeness centrality,
- Important application: Social Network Analysis

Example: Method of least-squares

- First use of least squares by Gauss, in early 1800's:

A planet follows an elliptical orbit according to $ay^2 + bxy + cx + dy + e = x^2$ in cartesian coordinates. Given a set of noisy observations of (x, y) positions, compute a, b, c, d, e, and use to predict future positions of the planet. This least squares problem is nearly rank-deficient and hence very sensitive to perturbations in the observations.

- Problem type: Least-Squares system

Read Wikipedia's article on planet ceres:
http://en.wikipedia.org/wiki/Ceres_(dwarf_planet)

Example: Dynamical systems and epidemiology

A set of variables that fill a vector y are governed by the equation

$$\frac{dy}{dt} = Ay$$

Determine $y(t)$ for $t > 0$, given $y(0)$ [called 'orbit' of y]

➤ Problem type: (Linear) system of ordinary differential equations.

Solution:

$$y(t) = e^{tA} y(0)$$

➤ Involves exponential of A [think Taylor series], i.e., a matrix function

- This is the simplest form of dynamical systems (linear).
- Consider the slightly more complex system:

$$\frac{dy}{dt} = A(y)y$$

- Nonlinear. Requires 'integration scheme'.
- Next: a little digression into our interesting times...

Example: The SIR model in epidemiology

A population of $\boldsymbol{N}$ individuals, with $\boldsymbol{N = S + I + R}$ where:

S Susceptible population. These are susecptible to being contaminated by others (not immune).

I Infectious population: will contaminate susceptible individuals.

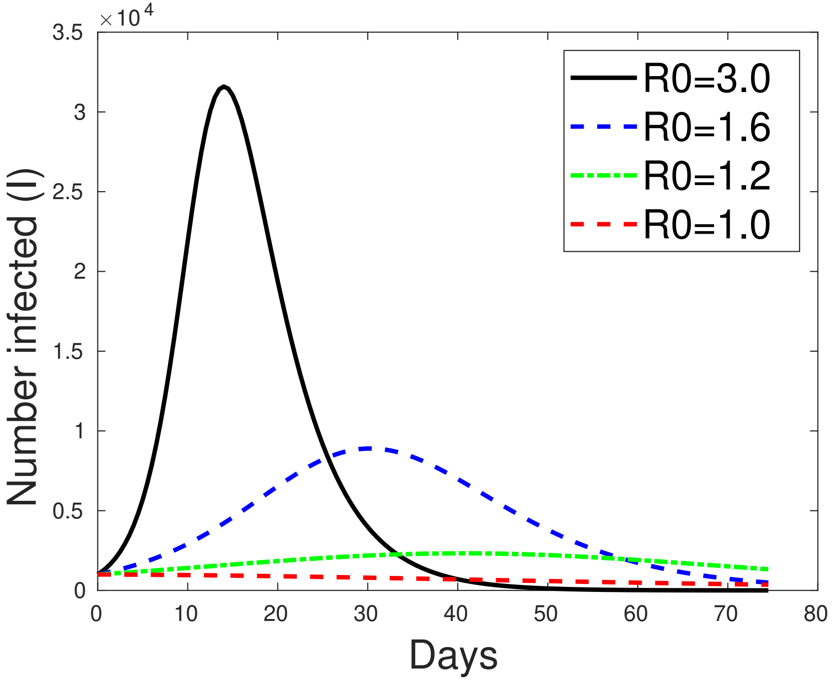
R 'Removed' population: either deceased or recovered. These will no longer contaminate others.

Three equations:

$$\boldsymbol{\frac{dS}{dt} = -\beta IS; \quad \frac{dI}{dt} = (\beta S - \mu)I; \quad \frac{dR}{dt} = \mu I}$$

$\boldsymbol{1/\mu}$ = infection period; $\boldsymbol{\beta = \mu R_0/N}$; $\boldsymbol{R_0}$ = reproduction number.

- The importance of reducing R_0 (a.k.a. “social distancing”):



- See the latest on this ($R_0 \approx 8.2$ for variant BA.1 and ≈ 12 for BA.2 !!)
- ... and keep away from each other

Problems in Numerical Linear Algebra

- Linear systems: $Ax = b$. Often: A is large and sparse
- Least-squares problems $\min \|b - Ax\|_2$
- Eigenvalue problem $Ax = \lambda x$. Several variations -
- SVD .. and
- ... Low-rank approximation
- Tensors and low-rank tensor approximation
- Matrix equations: Sylvester, Lyapunov, Riccati, ..
- Nonlinear equations – acceleration methods
- Matrix functions and applications
- Many many more ...

SPARSE MATRICES ; DATA STRUCTURES

What are sparse matrices?

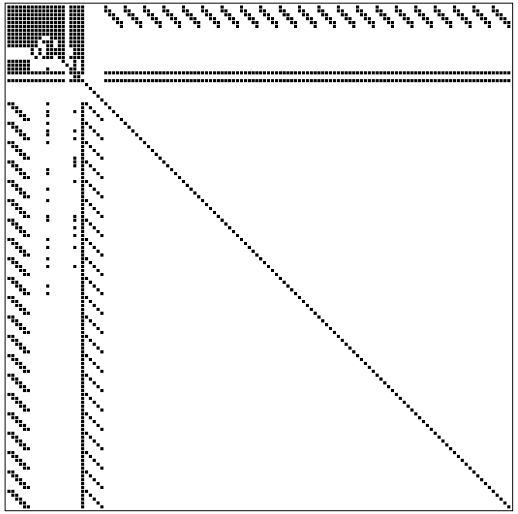
Vague definition: "..matrices that allow special techniques to take advantage of the large number of zero elements and the structure."

A few applications of sparse matrices: Structural Engineering, Reservoir simulation, Electrical Networks, optimization problems, ...

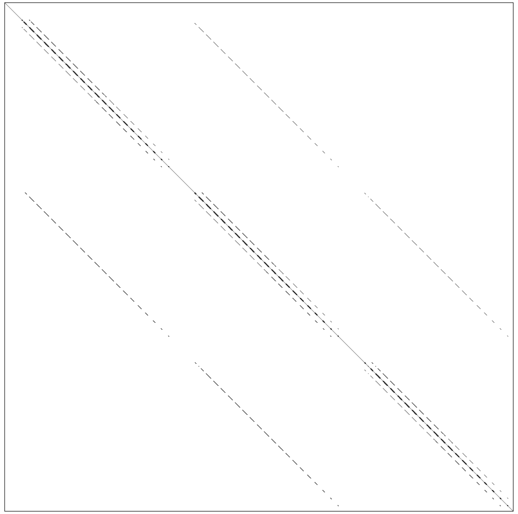
Goals: Much less storage and work than dense computations.

Observation: A^{-1} is usually dense, but L and U in the LU factorization may be reasonably sparse (if a good technique is used).

Sample sparsity patterns



ARC130: Unsymmetric matrix from laser problem. a.r.curtis, oct 1974



SHERMAN5: fully implicit black oil simulator 16 by 23 by 3 grid, 3 unk

Sparse matrices in Matlab

✍ Explore the scripts `Lap2D`, `mark` (provided in matlab suite) for generating sparse matrices

✍ Explore the command `spy`

✍ Explore the command `sparse`

✍ Run the demos titled `demo_sparse0` and `demo_sparse1`

✍ Load the matrix `can_256.mat` from the SuiteSparse collection. Show its pattern

Sparse matrices - continued

➤ *Main goal of Sparse Matrix Techniques:* To perform standard matrix computations economically, i.e., without storing the zeros

➤ *Example:* To add two square dense matrices of size n requires $O(n^2)$ operations. To add two sparse matrices A and B requires $O(nnz(A) + nnz(B))$ where $nnz(X)$ = number of nonzero elements of a matrix X.

➤ For typical Finite Element /Finite difference matrices, number of nonzero elements is $O(n)$.

Data structures: The coordinate format (COO)

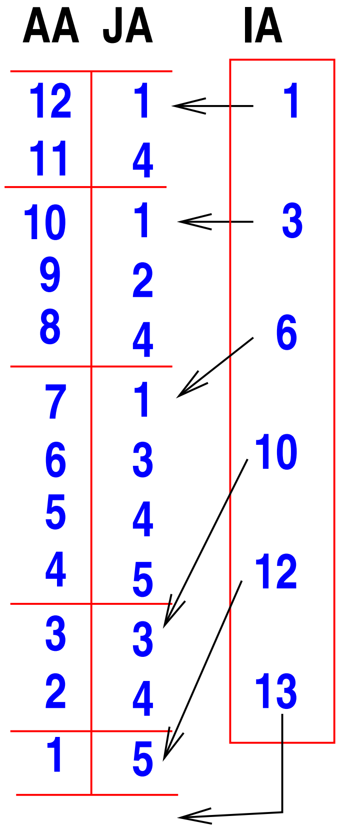
$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

AA	JR	JC
12.	5	5
9.	3	5
7.	3	3
5.	2	4
1.	1	1
2.	1	4
11.	4	4
3.	2	1
6.	3	1
4.	2	2
8.	3	4
10.	4	3

- Also known as 'triplet format'
- Simple data structure - Often used as 'entry' format in packages
- Variant used in matlab
- Note: order of entries is arbitrary [in matlab: sorted by columns]

Compressed Sparse Row (CSR) format

$$A = \begin{pmatrix} 12. & 0. & 0. & 11. & 0. \\ 10. & 9. & 0. & 8. & 0. \\ 7. & 0. & 6. & 5. & 4. \\ 0. & 0. & 3. & 2. & 0. \\ 0. & 0. & 0. & 0. & 1. \end{pmatrix}$$



- IA(j) points to beginning or row j in arrays AA, JA
- Related: Compressed Sparse Column format, Modified Sparse Row format (MSR).
- Used predominantly in Fortran & portable codes [e.g. Metis] – what about C?

CSR (CSC) format - C-style

* CSR: Collection of pointers of rows & array of row lengths

```
typedef struct SpaFmt {
/*---------------------------------------------
| C-style CSR format - used internally
| for all matrices in CSR/CSC format
|---------------------------------------------*/
  int n;           /* size of matrix          */
  int *nzcount;    /* length of each row      */
  int **ja;        /* to store column indices */
  double **ma;     /* to store nonzero entries */
} SparMat;
```

`aa[i][*]` == entries of i-th row (col.);

`ja[i][*]` == col. (row) indices,

`nzcount[i]` == number of nonzero elmts in row (col.) `i`

Data structure used in Csparse [T. Davis' SuiteSparse code]

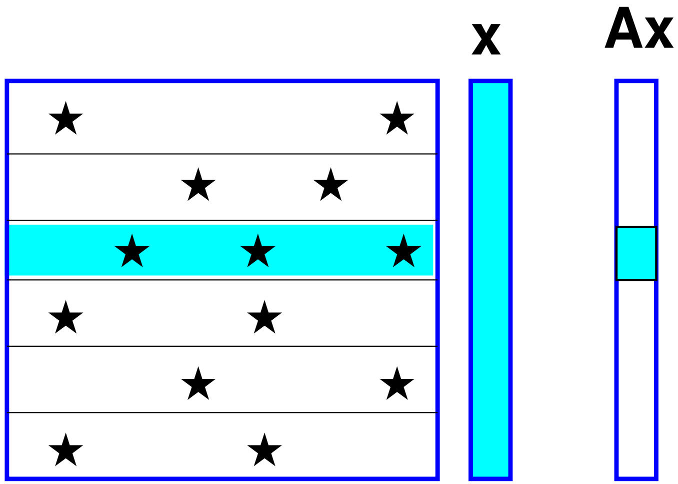
```
typedef struct cs_sparse
{/* matrix in compressed-column or triplet form */
 int nzmax ; /* maximum number of entries */
 int m ;     /* number of rows */
 int n ;     /* number of columns */
 int *p ;    /* column pointers (size n+1) or
                col indices (size nzmax) */
 int *i ;    /* row indices, size nzmax */
 double *x ; /* numerical values, size nzmax */
 int nz ;    /* # of entries in triplet matrix,
                -1 for compressed-col */
} cs ;
```

- Can be used for CSR, CSC, and COO (triplet) storage
- Easy to use from Fortran

Computing $y = Ax$; row and column storage

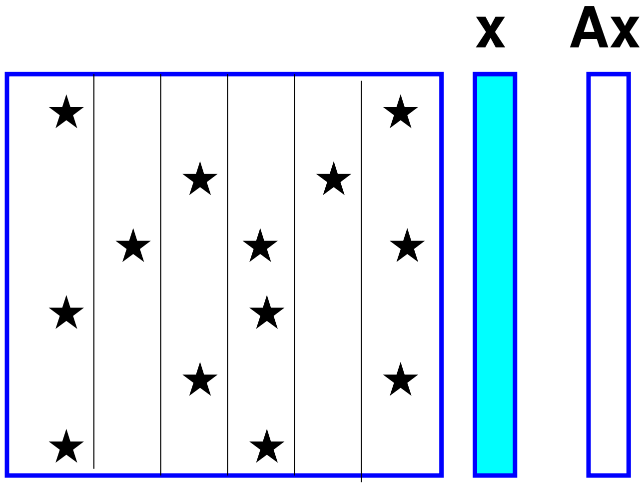
Row-form:

Dot product of $A(i,:)$ and x gives y_i



Column-form:

Linear combination of columns $A(:,j)$ with coefficients x_j yields y



Matvec – row version

```
void matvec( csptr mata, double *x, double *y )
{
    int i, k, *ki;
    double *kr;
    for (i=0; i<mata->n; i++) {
        y[i] = 0.0;
        kr = mata->ma[i];
        ki = mata->ja[i];
        for (k=0; k<mata->nzcount[i]; k++)
            y[i] += kr[k] * x[ki[k]];
    }
}
```

➤ Uses sparse dot products (sparse SDOTS)

✍ Operation count

Matvec – Column version

```
void matvecC( csptr mata, double *x, double *y )
{
  int n = mata->n, i, k, *ki;
  double *kr;
  for (i=0; i<n; i++)
    y[i] = 0.0;
  for (i=0; i<n; i++) {
    kr = mata->ma[i];
    ki = mata->ja[i];
    for (k=0; k<mata->nzcount[i]; k++)
      y[ki[k]] += kr[k] * x[i];
  }
}
```

➤ Uses sparse vector combinations (sparse SAXPY)

✍ Operation count

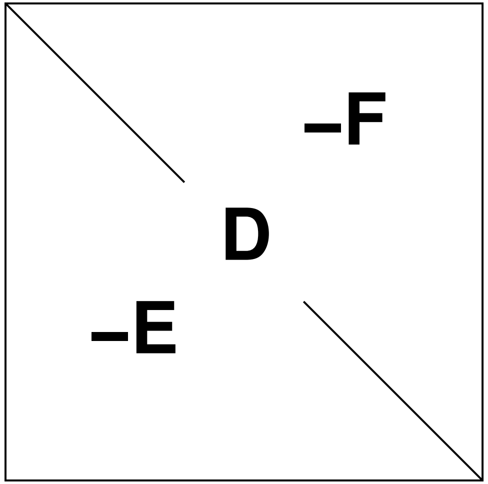
- Using the CS data structure from Suite-Sparse:

```
int cs_gaxpy (cs *A, double *x, double *y) {
 int p, j, n, *Ap, *Ai;
 n = A->n; Ap = A-> p; Ai = A->i; Ax = A->x;
 for (j=0; j<n; j++) {
   for (p=Ap[j]; p<Ap[j+1];p++)
     y[Ai[p]] += Ax[p]*x[j];
 }
return(1)
}
```

BASIC RELAXATION METHODS

Linear Systems: Basic Relaxation Schemes

Relaxation schemes: based on the decomposition $A = D - E - F$



D = diag(A), $-E$ = strict lower part of A and $-F$ its strict upper part.

- For example, Gauss-Seidel iteration :

$$(D - E)x^{(k+1)} = Fx^{(k)} + b$$

- Most common techniques 60 years ago.
- Now: used as smoothers in Multigrid or as preconditioners

Note: If $\rho_i^{(k)} = i$th component of current residual $b - Ax$ then relaxation version of GS is:

$$\xi_i^{(k+1)} = \xi_i^{(k)} + \frac{\rho_i^{(k)}}{a_{ii}}$$

for $i = 1, \cdots, n$

Iteration matrices

➤ Jacobi, Gauss-Seidel, SOR, & SSOR iterations are of the form

$$x^{(k+1)} = Mx^{(k)} + f$$

- $M_{Jac} = D^{-1}(E+F) = I - D^{-1}A$
- $M_{GS}(A) = (D-E)^{-1}F = I - (D-E)^{-1}A$

SOR relaxation: $\xi_i^{(k+1)} = \omega\xi_i^{(GS,k+1)} + (1-\omega)\xi_i^{(k)}$

- $M_{SOR}(A) = (D-\omega E)^{-1}(\omega F + (1-\omega)D)$
  $= I - (\omega^{-1}D - E)^{-1}A$

✍ Matlab: take a look at: *gs.m*, *sor.m*, and *sorRelax.m* in iters/

An observation & Introduction to Preconditioning

➤ The iteration $x^{(k+1)} = Mx^{(k)} + f$ is attempting to solve $(I - M)x = f$. Since M is of the form $M = I - P^{-1}A$ this system can be rewritten as

$$P^{-1}Ax = P^{-1}b$$

where for SSOR, we have

$$P_{SSOR} = (D - \omega E)D^{-1}(D - \omega F)$$

referred to as the SSOR 'preconditioning' matrix.

In other words:

Relaxation Scheme $\Longleftrightarrow$ *Preconditioned Fixed Point Iteration*