# Projection and its Importance in Scientific Computing

**Stan Tomov**
*EECS Department*
*The University of Tennessee, Knoxville*

April 12, 2020

- Contact information


office : Claxton 317

phone : (865) 974-6317

email : tomov@cs.utk.edu

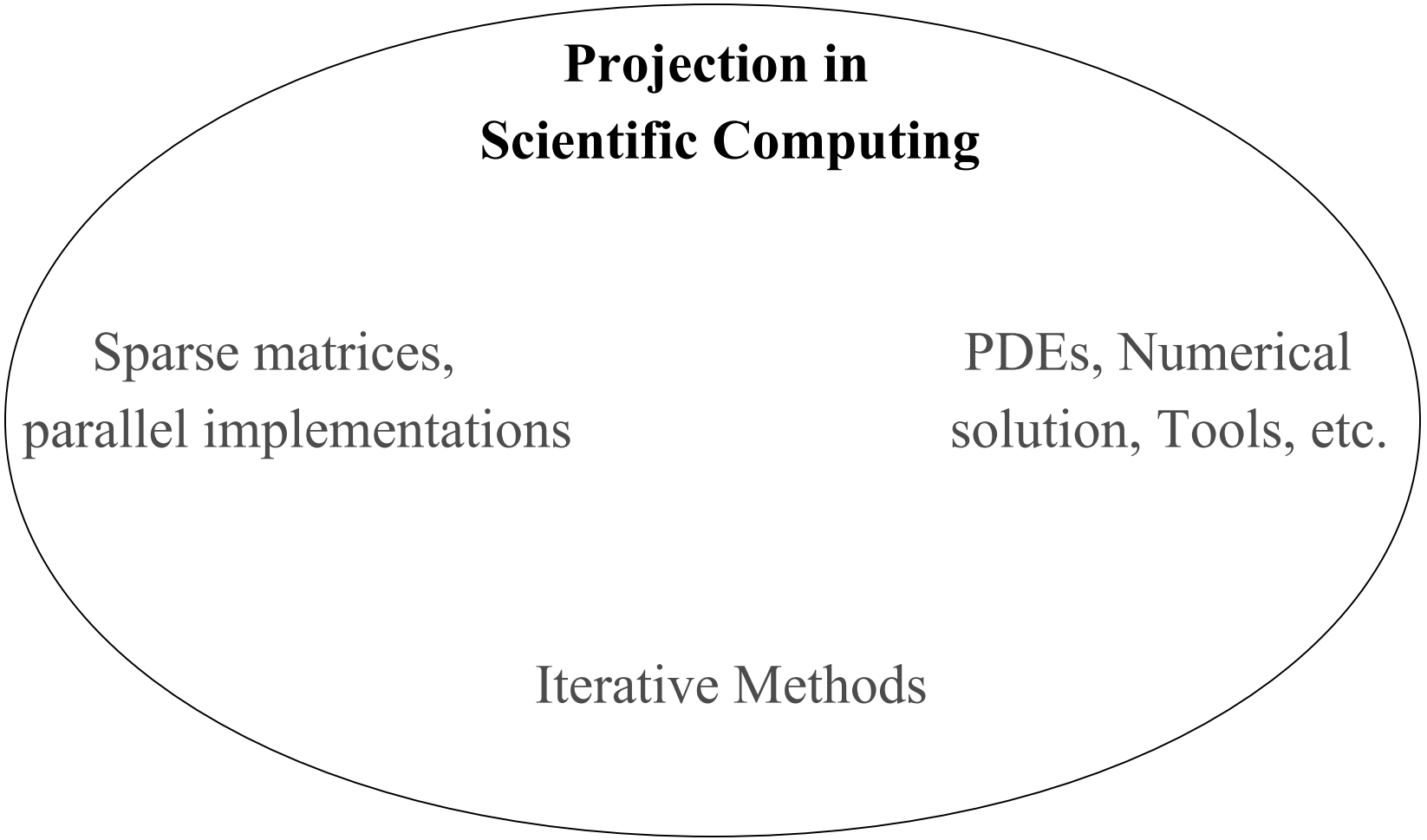

# Additional reference materials:


[1] R.Barrett, M.Berry, T.F.Chan, J.Demmel, J.Donato, J. Dongarra, V. Eijkhout, R.Pozo, C.Romine, and H.Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods (2nd edition)*
http://netlib2.cs.utk.edu/linalg/html_templates/Templates.html

[2] Yousef Saad, *Iterative methods for sparse linear systems (1st edition)*
*http://www-users.cs.umn.edu/~saad/books.html*

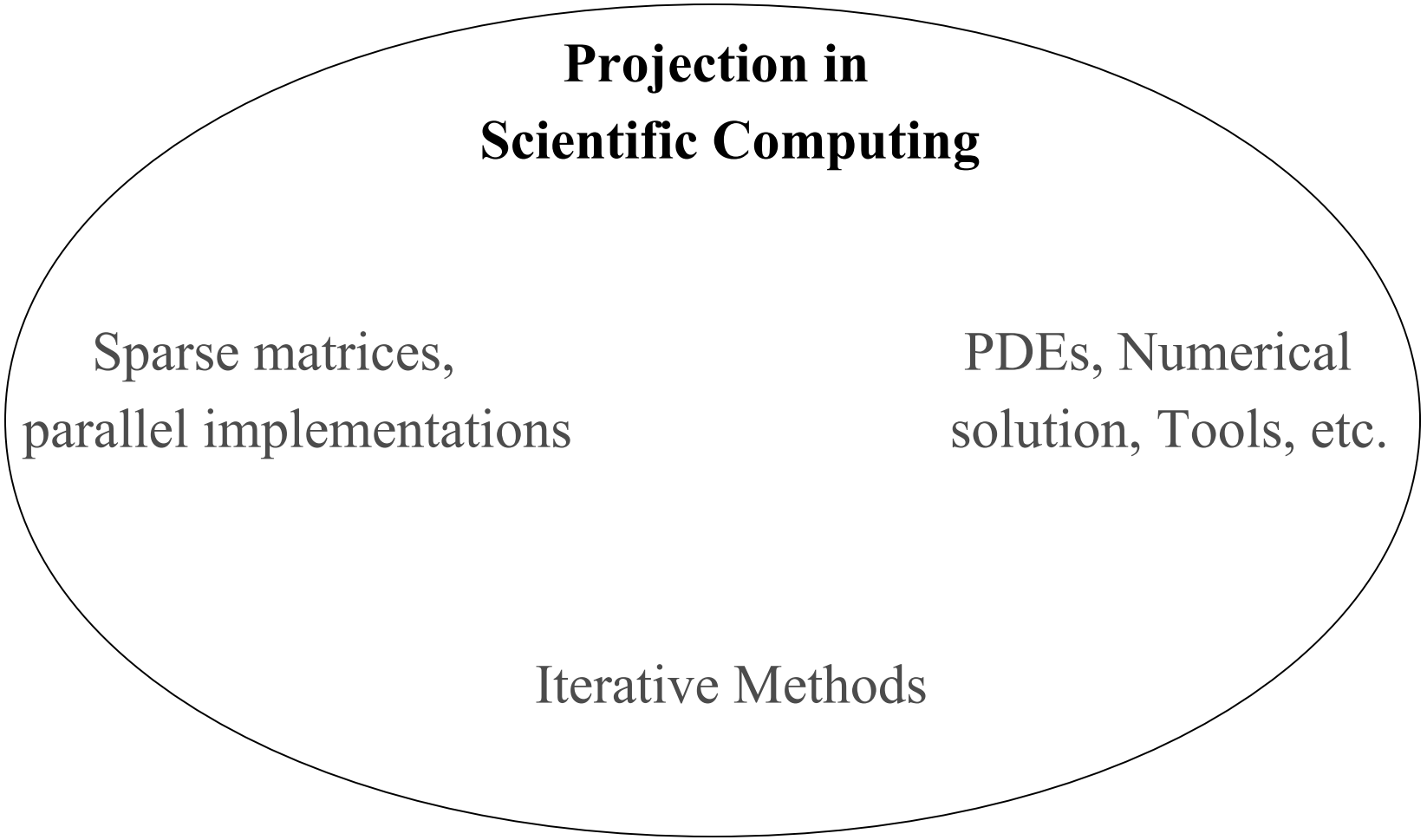
# Topics

as related to **high-performance scientific computing**



# Topics

## on new architectures – multicore, GPUs (CUDA & OpenCL), MIC



# Outline

- Part I
  - Fundamentals
- Part II
  - Projection in Linear Algebra
- Part III
  - Projection in Functional Analysis (e.g. PDEs)
- HPC with Multicore and GPUs

# Part I

# Fundamentals

# Projection in Scientific Computing

## [ an example – in solvers for PDE discretizations ]

### Electronic structure calculations

- Density functional theory

Many-body Schrödinger equation (exact but exponential scaling)

$$\{-\sum_i \frac{1}{2}\nabla_i^2 + \sum_{i,j}\frac{1}{|r_i - r_j|} + \sum_{i,I}\frac{Z}{|r_i - R_I|}\}\Psi(r_1,..r_N) = E\Psi(r_1,..r_N)$$

- Nuclei fixed, generating external potential (system dependent, non-trivial)
- N is number of electrons

**Kohn Sham Equation: The many body problem of interacting electrons is reduced to non-interacting electrons (single particle problem) with the same electron density and a different effective potential (cubic scaling).**

$$\{-\frac{1}{2}\nabla^2 + \int\frac{\rho(r')}{|r - r'|}dr' + \sum_I \frac{Z}{|r - R_I|} + V_{XC}\}\psi_i(r) = E_i\psi_i(r)$$

$$\rho(r) = \sum_i |\psi_i(r)|^2 = |\Psi(r_1,..r_N)|^2$$

- $V_{XC}$ represents effects of the Coulomb interactions between electrons
- $\rho$ is the density (of the original many-body system)

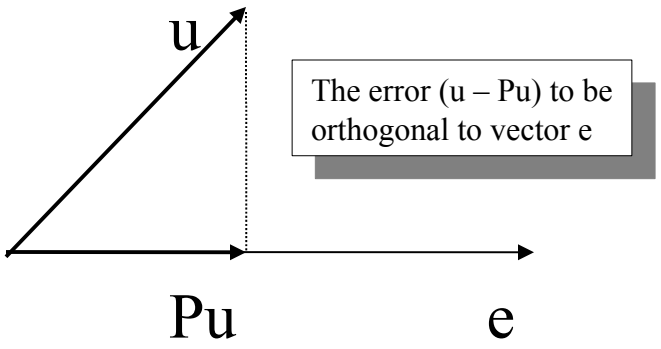
$V_{XC}$ is not known except special cases ⇒ use approximation, e.g. Local Density Approximation (LDA) where $V_{XC}$ depends only on $\rho$

- A model leading to self-consistent iteration with need for high-performance diagonalization and orthogonalization routines

# What is Projection?

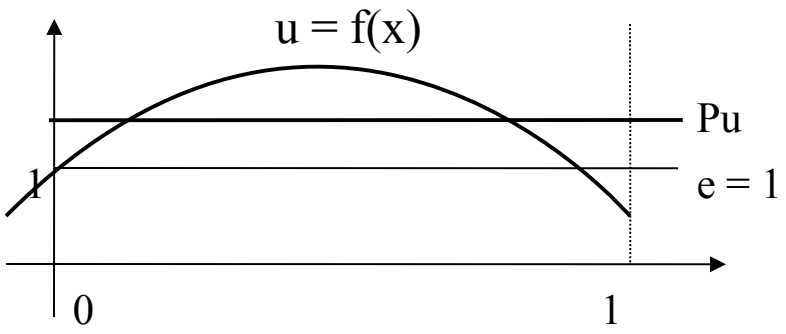
- Here are two examples

(from linear algebra)




P : orthogonal projection of vector u on e

(from functional analysis)




P : best approximation (projection) of f(x) in span{ e } ⊂ C[0,1]

# Definition

- **Projection** is a linear transformation P from a linear space V to itself such that

$$P^2 = P$$

**equivalently**

Let V is direct sum of subspaces $V_1$ and $V_2$

$$V = V_1 \oplus V_2$$

i.e. for $\forall\, u \in V$ there are unique $u_1 \in V_1$ and $u_2 \in V_2$ s.t.

$$u = u_1 + u_2$$

Then $P: V \rightarrow V_1$ is defined for $\forall\, u \in V$ as $Pu \equiv u_1$

# Importance in Scientific Computing

- To compute approximations $Pu \approx u$ where **dim** $V_1$ << **dim** $V$

$$V = V_1 \oplus V_2$$

- When computation directly in V is not feasible or even possible.

A few examples:

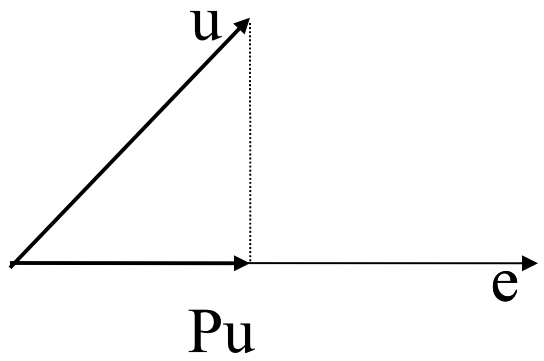
  - Interpolation (via polynomial interpolation/projection)
  - Image compression
  - Sparse iterative linear solvers and eigensolvers
  - Finite Element/Volume Method approximations
  - Least-squares approximations, etc.

# Projection in $R^2$

- In $R^2$ with Euclidean inner-product, i.e. for $x, y \in R^2$

$$\mathbf{(x, y)} = x_1 y_1 + x_2 y_2 \quad ( = y^T x = x \cdot y )$$

and $\quad \| x \| = (x, x)^{1/2}$




P : orthogonal projection of vector u on e

$$Pu = \frac{(u, e)}{\| e \|^2} e \qquad \text{(Exercise)}$$

i.e. for $\| e \| = 1$

$$Pu = (u, e)\, e$$

# Projection in $R^n$ / $C^n$

- Similarly to $R^2$

  P : Orthogonal projection of u into span$\{e_1, \ldots, e_m\}$, $m \leq n$.

  Let $e_i$, $i = 1 \ldots m$ is **orthonormal** basis, i.e.

  $$(e_i, e_j) = 0 \qquad \text{for } i \neq j \text{ and}$$

  $$(e_i, e_j) = 1 \qquad \text{for } i = j$$

  $$\mathbf{P\,u = (u, e_1)\, e_1 + \ldots + (u, e_m)\, e_m} \quad (\text{Exercise})$$

  Orthogonal projection of u on $e_1$

# How to get an orthonormal basis?

- Can get one from every subspace by **Gram-Schmidt** orthogonalization:

  Input : m linearly independent vectors $x_1, ..., x_m$

  Output : m orthonormal vectors $x_1, ..., x_m$

  1. $x_1 = x_1 / \| x_1 \|$
  2. do i = 2, m
  3. $x_i = x_i - (x_i, x_1)\, x_1 - ... - (x_i, x_{i-1})\, x_{i-1}$ (Exercise: $x_i \perp x_1, ..., x_{i-1}$)
  4. $x_i = x_i / \| x_i \|$
  5. enddo

  Orthogonal projection of $x_i$ on $x_1$

  Known as **Classical Gram-Schmidt** (CGM) orthogonalization

# How to get an orthonormal basis?

- What if we replace line 3 with the following (3')?

3. $x_i = x_i - (x_i, x_1)\, x_1 - \ldots - (x_i, x_{i-1})\, x_{i-1}$

3'. do j = 1, i-1

$x_i = x_i - (x_i, x_j)\, x_j$

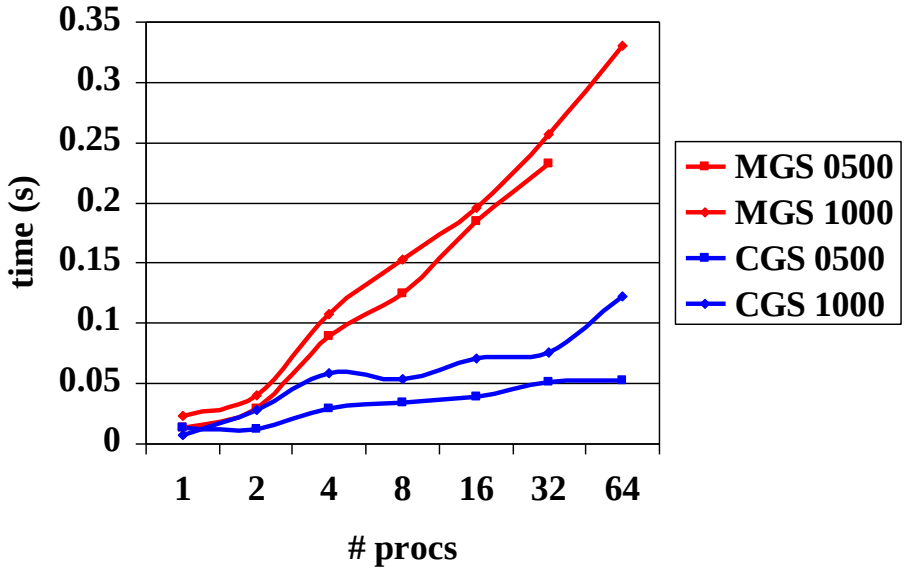
enddo

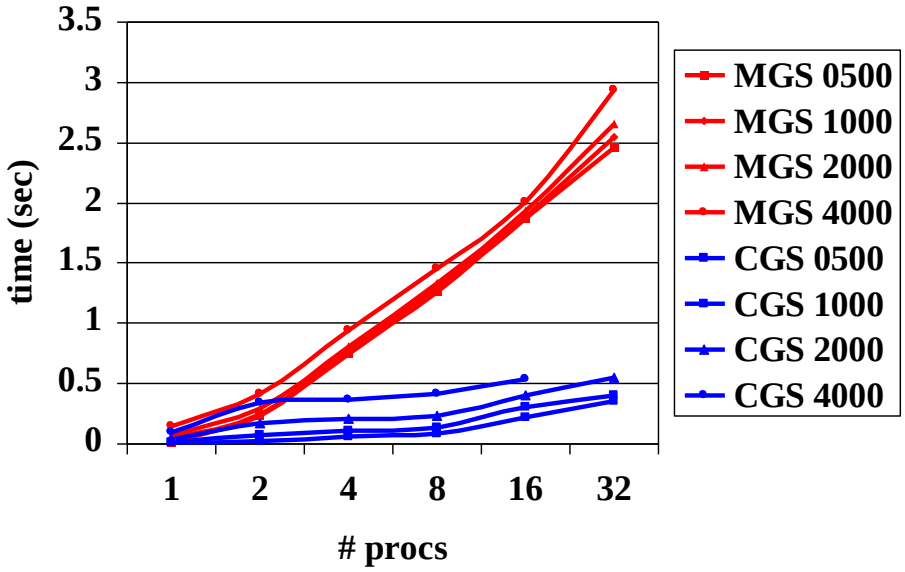

- Equivalent in exact arithmetic (Exercise) but not with round-off errors (next) !
- Known as **Modified Gram-Schmidt** (MGS) orthogonalization

# CGS *vs* MGS

[ Results from Julien Langou: ]

Scalability of MGS and CGS on two different clusters for matrices of various size m=[500 1000 2000 4000] per processor, n = 100

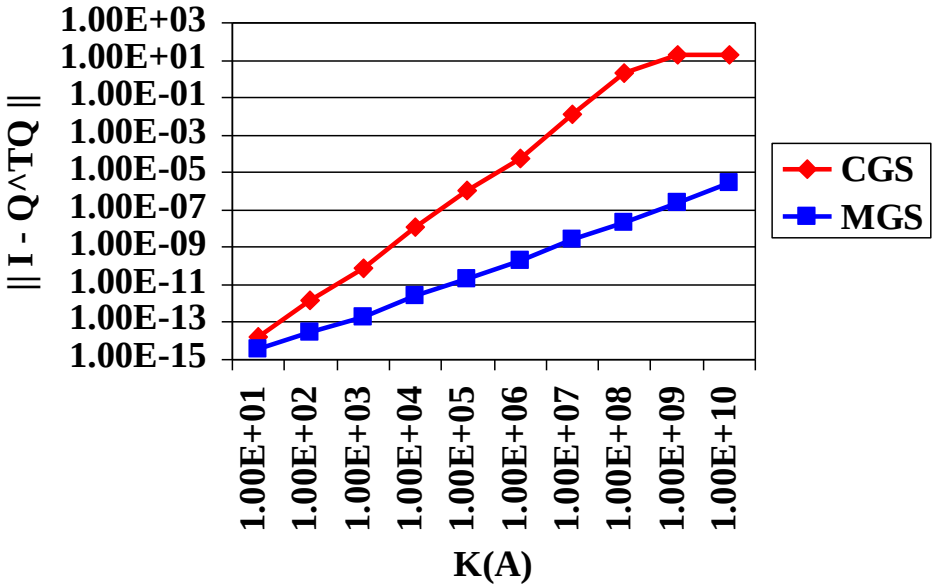


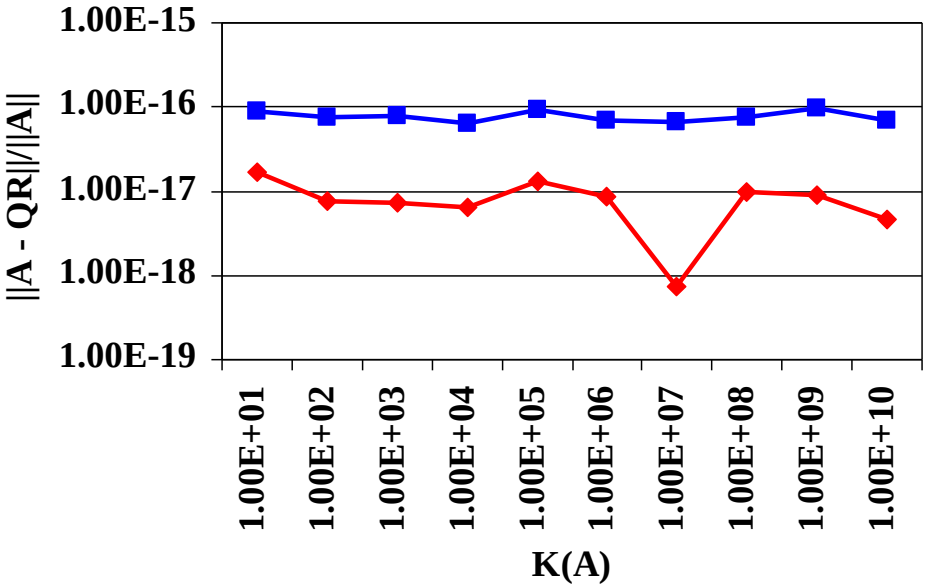



# CGS *vs* MGS

[ Results from Julien Langou: ]

Accuracy of MGS *vs* CGS on matrices of increasing condition number



# QR factorization

- Let $\mathbf{A} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$ be the input for CGS/MGS and

  $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_m]$ the output;

  R : an upper m $\times$ m triangular matrix defined from the CGR/MGS.

  Then

$$\mathbf{A} = \mathbf{Q}\,\mathbf{R}$$

# Other QR factorizations

- **What about the following?**
  **[**known as Cholesky QR]

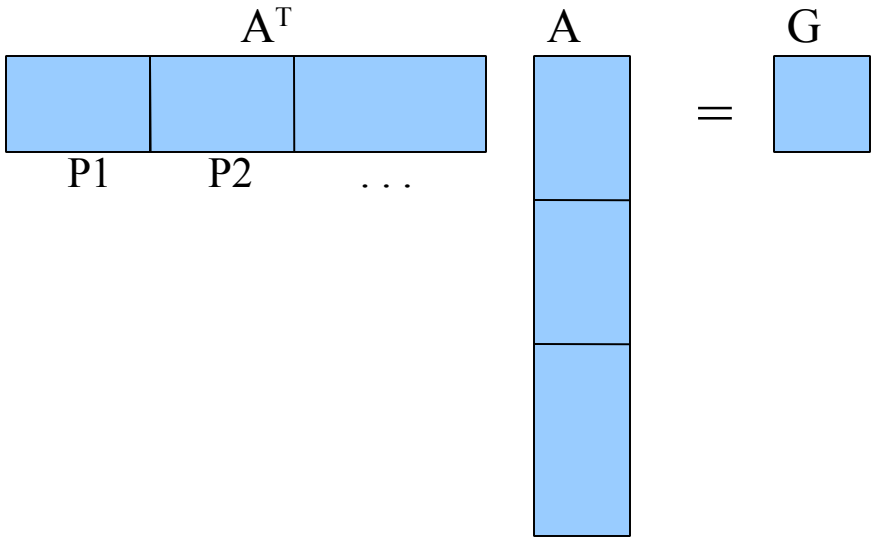
1. $G = A^T A$
2. $G = L\,L^T$ (Cholesky factorization)
3. $Q = A\,(L^T)^{-1}$

- Does Q have orthonormal columns (i.e. $Q^TQ=I$),
  i.e. $A = QL^T$ to be a QR factorization (Exercise)

- When is this feasible and how compares to CGS and MGS?

# Other QR factorizations

- Feasible when n >> m
- Allows efficient parallel implementation:
  **blocking** both computation and communication




**Investigate numerically accuracy and scalability**
**(compare to CGS and MGS)**
**Exercise**

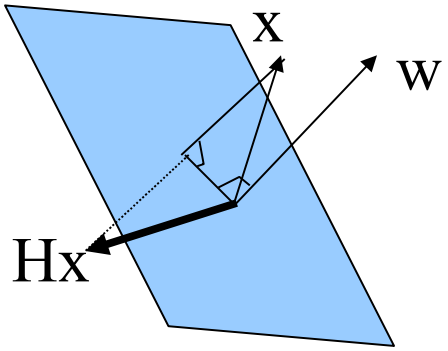
# How is done in LAPACK?

- Using **Householder reflectors**

  $H = I - 2\, w\, w^T$

- w = ?  so that

  $H\, x_1 = \alpha\, e_1$




$\Rightarrow$  w = ... (compute or look at the reference books)

- Allows us to construct

$$X_k \equiv \underbrace{H_{k-1} \ldots H_1}_{Q} X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & \cdots & \cdots & x_{1m} \\ & x_{22} & x_{23} & \cdots & \cdots & \cdots & x_{2m} \\ & & x_{33} & \cdots & \cdots & \cdots & x_{3m} \\ & & & \ddots & \cdots & \cdots & \vdots \\ & & & & x_{kk} & \cdots & \vdots \\ & & & & x_{k+1,k} & \cdots & x_{k+1,m} \\ & & & & \vdots & \vdots & \vdots \\ & & & & x_{n,k} & \cdots & x_{n,m} \end{pmatrix}$$

(Exercise(

LAPACK implementation : "**delayed update**" of the trailing matrix +

"**accumulate transformation**" to apply it as **BLAS 3**

# Part II

# Projection in Linear Algebra

# Projection into general basis

- How to define projection without orthogonalization of a basis?
  - Sometimes is not feasible to orthogonalize
  - Often the case in functional analysis
    (e.g. Finite Element Method, Finite Volume Method, etc.)
    where the basis is "linearly independent" functions (more later, and Lecture 2)
- We saw if $X = [x_1, \ldots, x_m]$ is an orthonormal basis

  (*) $\mathbf{P\,u = (u, x_1)\,x_1 + \ldots + (u, x_m)\,x_m}$
- How does (*) change if X are just linearly independent ?

  $\mathbf{P\,u =}$ **?**

# Projection into a general basis

- The problem:

Find the coefficients $C = (c_1 \ldots c_m)^T$ in

$$P\,u = c_1\,x_1 + c_2\,x_2 + \ldots + c_m\,x_m = X\,C$$

so that

$$u - Pu \perp \text{span}\{x_1, \ldots, x_m\}$$

or $\Leftrightarrow$ so that the **error e** in

(1) $$\mathbf{u} = \mathbf{P}\,\mathbf{u} + \mathbf{e}$$

is $\perp$ $\text{span}\{x_1, \ldots, x_m\}$

# Projection into a general basis

- (1) $\mathbf{u} = \mathbf{P}\,\mathbf{u} + \mathbf{e} = \mathbf{c}_1\,\mathbf{x}_1 + \mathbf{c}_2\,\mathbf{x}_2 + \ldots + \mathbf{c}_m\,\mathbf{x}_m + \mathbf{e}$
- Multiply (1) on both sides by "***test***" vector/function $x_j$ (terminology from functional analysis) for $j = 1,\ldots, m$

$$(\mathbf{u}, \mathbf{x}_j) = \mathbf{c}_1\,(\mathbf{x}_1, \mathbf{x}_j) + \mathbf{c}_2\,(\mathbf{x}_2, \mathbf{x}_j) + \ldots + \mathbf{c}_m\,(\mathbf{x}_m, \mathbf{x}_j) + \cancel{(\mathbf{e}, \mathbf{x}_j)}^{\,0}$$

- i.e., **m** equations for **m** unknowns
- In matrix notations $\Leftrightarrow (\mathbf{X}^T\,\mathbf{X})\,\mathbf{C} = \mathbf{X}^T\,\mathbf{u}$ (Exercise)

  $\Leftrightarrow \mathbf{X}^T\,\mathbf{P}\mathbf{u} = \mathbf{X}^T\,\mathbf{u}$
- $X^TX$ is the so called Gram matrix (nonsingular; why?) => there exists a unique solution C

# Normal equations

- System

$$(X^T X)\, C = X^T u$$

  is known also as **Normal Equations**

- The ***Method of Normal Equations***:
  Finding the projection (approximation) Pu ≡ XC (approximation) of u in X by solving the Normal Equations system

# Least Squares (LS)

- Equivalently, system

$$(X^T X)C = X^T u \qquad P\,u$$

gives also the solution of the LS problem

$$\min_{C \in R^m} \| X C - u \|$$

since $\| v_1 - u \|^2 = \| (v_1 - Pu) - e \|^2 = \| v_1 - Pu \|^2 + \| e \|^2$

$$\geq \| e \|^2 = \| Pu - u \|^2 \quad \text{for} \quad \forall\, v_1 \in V_1$$

# LS

- Note that the usual notations for LS is: For $A \in R^{n \times m}$, $b \in R^n$ find

$$\min_{x \in R^m} \| A x - b \|$$

- Solving LS with QR factorization

Let $A = Q R$, $Q^T A = R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$ , $Q^T b = \begin{pmatrix} c \\ d \end{pmatrix}$ $\begin{matrix} m \\ n - m \end{matrix}$

Then

$$\| Ax - b \|^2 = \| Q^T Ax - Q^T b \|^2 = \| R_1 x - c \|^2 + \| d \|^2$$

i.e. we get minimum if x is such that

$$R_1 x = c$$

# Projection and iterative solvers

- The problem : Solve
  $$Ax = b \qquad \text{in } R^n$$
- Iterative solution: at iteration **i** extract an approximate $\mathbf{x}_i$ from just a subspace $V = \text{span}[v_1, ..., v_m]$ of $R^n$
- How? As on slide 22, impose constraints:
  $b - Ax \perp$ subspace $W = \text{span}[w_1,...,w_m]$ of $R^n$, i.e.
  $$(*) \qquad (Ax, w_i) = (b, w_i) \qquad \text{for } \forall\ w_i \in W = \text{span}[w_1,...,w_m]$$
- Conditions (*) known also as **Petrov-Galerkin conditions**
- Projection is ***orthogonal*****:** V and W are the same (Galerkin conditions) or
  ***oblique*** **:** V and W are different

# Matrix representation

- Let $V = [v_1, \ldots, v_m]$, $W = [w_1, \ldots, w_m]$

  Find $y \in R^m$ s.t. $\mathbf{x = x_0 + V y}$ solves $Ax = b$, i.e.

$$\mathbf{A V y} = b - Ax_0 = \mathbf{r_0}$$

subject to the orthogonality constraints:

$$\mathbf{W^T A V y} = \mathbf{W^T r_0}$$

- The choice for V and W is crucial and determines various methods (more in Lectures 13 and 14)

# A General Projection Algorithm

- Prototype from Y.Saad's book

1. *Until convergence, Do:*
2. *Select a pair of subspaces* $\mathcal{K}$ *and* $\mathcal{L}$
3. *Choose bases* $V = [v_1, \ldots, v_m]$ *and* $W = [w_1, \ldots, w_m]$ *for* $\mathcal{K}$ *and* $\mathcal{L}$
4. $r := b - Ax$
5. $y := (W^T AV)^{-1} W^T r$
6. $x := x + Vy$
7. *EndDo*

# Projection and Eigen-Solvers

- The problem : Solve
  $Ax = \lambda x$ in $R^n$
- As in linear solvers: at iteration **i** extract an approximate
  $\mathbf{x}_i$ from a subspace $V = span[v_1, ..., v_m]$ of $R^n$
- How? As on slides 22 and 26, impose constraints:
  $\lambda x - Ax \perp$ subspace $W = span[w_1,...,w_m]$ of $R^n$, i.e.

  (*) $(Ax, w_i) = (\lambda x, w_i)$ for $\forall\ w_i \in W = span[w_1,...,w_m]$
- This procedure is known as **Rayleigh-Ritz**
- Again projection can be ***orthogonal*** or ***oblique***

# Matrix representation

- Let $V = [v_1, \ldots, v_m]$, $W = [w_1, \ldots, w_m]$

  Find $y \in R^m$ s.t. $\mathbf{x = V y}$ solves $Ax = \lambda x$, i.e.

  $$\mathbf{A V y = \lambda V y}$$

  subject to the orthogonality constraints:

  $$\mathbf{W^T A V y = \lambda W^T V y}$$

- The choice for V and W is crucial and determines various methods (more in Lectures 4 and 5)

# Part III

# Projection in PDEs

# Projection in Functional Spaces

- The discussion so far can be applied to any functional inner-product space (examples to follow)
- An important space is **C[a, b]**, the space of continuous functions on [a, b], with inner-product

$$(f, g) = \int_a^b f(x)\, g(x)\, dx$$

and induced norm

$$\| f \| = (f, f)^{1/2}$$

# Projection in Functional Spaces

- Projection P: $V \rightarrow V_1$ where $V = V_1 \oplus V_2$
- In functional analysis and scientific computing $V_1$ is usually taken as
  - Piecewise polynomials
    - In PDE approximation (FEM/FVM), Numerical integration, etc.
  - Trigonometric functions

    $\{ \sin(n\,x), \cos(n\,x) \}_{n=0,\ldots}$ , $x \in [0, 2\pi]$

    Orthogonal relative to

$$(f, g) = \int_0^{2\pi} f(x)\, g(x)\, dx$$ (Exercise)

# Normal equations / LS

- **Exercise:**

$$f(x) = \sin(x)$$

Find the projection in $V_1 = \text{span}\{x, x^3, x^5\}$ on interval [-1, 1] using inner-product

$$(f, g) = \int_{-1}^{1} f(x)\, g(x)\, dx$$

and norm $\| f \| = (f,f)^{1/2}$

# Normal equations / LS

- Leads to Gram matrix that is very ill-conditioned
  (called Hilbert matrix: Gram matrix for polynomials 1, x, $x^2$, $x^3$, ...)
- For numerical stability is better to orthogonalize the polynomials
- There are numerous examples of orthonormal polynomial sets

  * Legendre, Chebyshev, Hermite, etc.
  * Check the literature for more if interested

# Integration via Polynomial Interpolation

- Take

$$\int f(x)\ dx \approx \int p(x)\ dx$$

where p is a polynomial approximation to f

- Taking p a polynomial interpolating f at n+1 fixed nodes $x_i$ leads to quadrature formulas

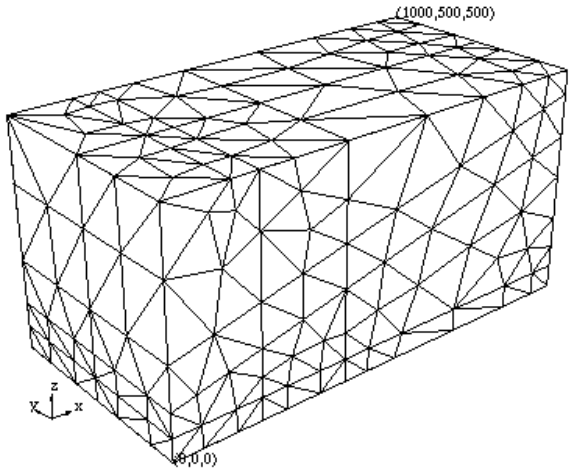
$$\int f(x)\ dx \approx A_0\ f(x_0) + \ldots + A_n\ f(x_n)$$

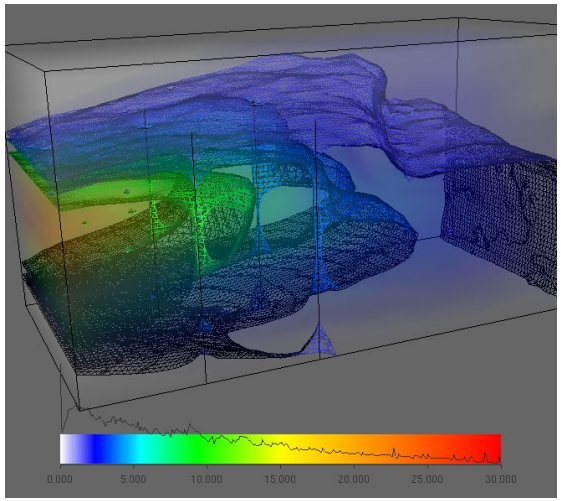
that are exact exact for polynomials of degree $\leq n$

- Smart choice of the nodes $x_i$ (Gaussian quadrature) leads to formulas that are exact for polynomials of degree $\leq 2n+1$

# Galerkin Projection

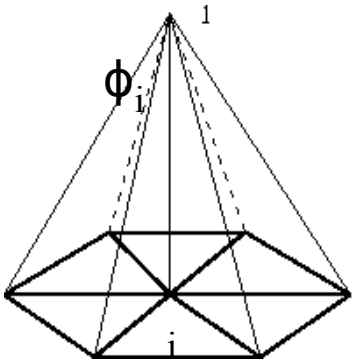
- Numerical PDE discretizations have a common concept:
  - Represent computational domain with mesh
  - Approximate functions and operators over the mesh



# Galerkin Projection

- Finite dimensional spaces (e.g. $V_1$) can can be piecewise polynomials defined over the mesh, e.g.




- Numerical solution of PDE (e.g. FEM)
  - Boundary value problem: $\boldsymbol{Au} = \boldsymbol{f}$, subject to boundary conditions
  - Get a "weak" formulation: $(Au, \phi) = (f, \phi)$ - multiply by test function $\phi$ and integrate over the domain

$$a(u, \phi) = \langle f, \phi \rangle \text{ for } \forall \phi \in S$$

  - Galerkin (FEM) problem: Find $u_h \in S_h \subset S$ s.t.

$$a(u_h, \phi_h) = \langle f, \phi_h \rangle \text{ for } \forall \phi_h \in S_h$$

# Learning Goals

- To refresh some linear algebra essentials that are of fundamental importance for scientific computing
- The idea and application of **Petrov-Galerkin conditions** as a way of defining computationally feasible formulations (approximations)
- Some generic examples demonstrating the ideas in
  - Linear algebra
  - Functional analysis
    (to get more specific in the following lectures)