Projection and its Importance in Scientific Computing

Stan Tomov
EECS Department
The University of Tennessee, Knoxville

March 4, 2015
Contact information

office: Claxton 317
phone: (865) 974-6317
email: tomov@cs.utk.edu

Additional reference materials:


http://www-users.cs.umn.edu/~saad/books.html
Topics as related to high-performance scientific computing

Projection in Scientific Computing

Sparse matrices, parallel implementations

PDEs, Numerical solution, Tools, etc.

Iterative Methods
Topics on new architectures – multicore, GPUs (CUDA & OpenCL), MIC

Projection in Scientific Computing

Sparse matrices, parallel implementations

PDEs, Numerical solution, Tools, etc.

Iterative Methods
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
Electronic structure calculations

- Density functional theory
  Many-body Schrödinger equation (exact but exponential scaling)
  \[-\sum_{i} \frac{1}{2} \nabla_{i}^{2} + \sum_{\alpha \neq \beta} \frac{1}{|r_{i} - r_{\beta}|} + \sum_{\beta} \frac{Z_{\beta}}{|r_{i} - R_{\beta}|} \psi(r_{1},\ldots,r_{N}) = E\psi(r_{1},\ldots,r_{N})\]
  - 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).

\[-\frac{1}{2} \nabla^{2} + \int \frac{\rho(r')}{|r - r'|} dr' + \sum_{\beta} \frac{Z_{\beta}}{|r - R_{\beta}|} + V_{xc}(r) \psi_{i}(r) = E_{i}\psi_{i}(r)\]
\[\rho(r) = \sum_{i} |\psi_{i}(r)|^{2} = |\Psi(r_{1},\ldots,r_{N})|^{2}\]

- \(V_{xc}\) 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)

\[ P : \text{orthogonal projection of vector } u \text{ on } e \]

The error \((u - Pu)\) to be orthogonal to vector \(e\)

(From functional analysis)

\[ P : \text{best approximation (projection) of } f(x) \text{ in span\{e\} } \subset C[0,1] \]
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 $P: V \to V_1$ is defined for $\forall u \in V$ as $Pu \equiv u_1$
Importance in Scientific Computing

- To compute approximations $P u \approx u$ where $\dim V_1 \ll \dim V$
  
  $$V = V_1 \oplus V_2$$

- 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 $x, y \in R^2$
  \[(x, y) = x_1 y_1 + x_2 y_2 \quad (= y^T x = x \cdot y)\]
  and \[\| x \| = (x, x)^{1/2}\]

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

i.e. for $\| e \| = 1$
Pu = (u, e) e

P : orthogonal projection of vector $u$ on $e$
Projection in $\mathbb{R}^n / \mathbb{C}^n$

- Similarly to $\mathbb{R}^2$

  $P$ : Orthogonal projection of $u$ into $\text{span}\{e_1, \ldots, e_m\}$, $m \leq n$.
  
  Let $e_i$, $i = 1 \ldots m$ is **orthonormal** basis, i.e.
  
  $$
  (e_i, e_j) = 0 \quad \text{for } i \neq j \quad \text{and} \\
  (e_i, e_j) = 1 \quad \text{for } i = j
  $$

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

  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 $x_1, \ldots, x_m$
  
  **Output**: m orthonormal vectors $x_1, \ldots, x_m$

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

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'. \[ \text{do } j = 1, i-1 \]

  \[ x_i = x_i - (x_i, x_j) x_j \]

  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 \textit{vs} MGS

[Results from Julien Langou:]

Accuracy of MGS \textit{vs} CGS on matrices of increasing condition number
QR factorization

Let $A = [x_1, \ldots, x_m]$ be the input for CGS/MGS and $Q = [q_1, \ldots, q_m]$ the output;

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

Then

$$A = Q R$$
Other QR factorizations

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

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

- Does \( Q \) have orthonormal columns (i.e. \( Q^T Q = I \)),
  i.e. \( A = Q L^T \) to be a QR factorization (**Exercise**)

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

- Feasible when $n \gg m$
- Allows efficient parallel implementation: blocking both computation and communication

\[
\begin{align*}
A^T & \\
A & \\
G & \\
\end{align*}
\]

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

Exercise
How is done in LAPACK?

- Using **Householder reflectors**
  
  \[ H = I - 2w w^T \]

- \( w = ? \) so that

  \[ H x_1 = \alpha e_1 \]

\[ \Rightarrow \quad w = ... \quad \text{(compute or look at the reference books)} \]

- Allows us to construct

  \[ X_k \equiv H_{k-1} \ldots H_1 \quad X = Q \]

  \[(\text{Exercise})\]

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
  \[ P \mathbf{u} = (\mathbf{u}, x_1) x_1 + \ldots + (\mathbf{u}, x_m) x_m \]  

- How does (*) change if \( X \) are just linearly independent?  
  \[ P \mathbf{u} = \text{?} \]
Projection into a general basis

• The problem:

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

$$Pu = c_1 x_1 + c_2 x_2 + \ldots + c_m x_m = X C$$

so that

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

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

$$u = Pu + e$$

is $\perp \text{span}\{x_1, ..., x_m\}$
Projection into a general basis

(1) \[ u = P u + e = c_1 x_1 + c_2 x_2 + \ldots + c_m x_m + e \]

Multiply (1) on both sides by “test” vector/function \( x_j \)
(terminology from functional analysis) for \( j = 1, \ldots, m \)

\[ (u, x_j) = c_1 (x_1, x_j) + c_2 (x_2, x_j) + \ldots + c_m (x_m, x_j) + (e, x_j) \]

i.e., \( m \) equations for \( m \) unknowns

In matrix notations \( \leftrightarrow (X^T X) C = X^T u \) (Exercise)

\( \leftrightarrow X^T P u = X^T u \)

\( X^T X \) is the so called Gram matrix (nonsingular; why?) \( \Rightarrow \) 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) \(P_u \approx X C\) (approximation) of \(u\) in \(X\) by solving the Normal Equations system
Least Squares (LS)

- Equivalently, system\[
\begin{align*}
\begin{pmatrix} X^T \end{pmatrix} C &= X^T u
\end{align*}
\]
gives also the solution of the LS problem
\[
\min_{C \in \mathbb{R}^m} \| X C - u \|
\]
since
\[
\begin{align*}
\| v_1 - u \|^2 &= \| (v_1 - Pu) - e \|^2 = \| v_1 - Pu \|^2 + \| e \|^2 \\
&\geq \| e \|^2 = \| Pu - u \|^2 \quad \text{for } \forall v_1 \in V_1
\end{align*}
\]
LS

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

- Solving LS with QR factorization
  Let $A = QR$, $Q^T A = R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$, $Q^T b = \begin{pmatrix} c \\ d \end{pmatrix}$
  Then
  \[
  \| A x - b \|^2 = \| Q^T Ax - Q^T b \|^2 = \| R_1 x - c \|^2 + \| d \|^2
  \]
  i.e. we get minimum if $x$ is such that
  \[
  R_1 x = c
  \]
Projection and iterative solvers

- The problem : Solve
  \[ Ax = b \] in \( \mathbb{R}^n \)

- Iterative solution: at iteration \( i \) extract an approximate \( x_i \) from just a subspace \( V = \text{span}[v_1, ..., v_m] \) of \( \mathbb{R}^n \)

- How? As on slide 22, impose constraints:
  \[ b - Ax \perp \text{subspace } W = \text{span}[w_1, ..., w_m] \] of \( \mathbb{R}^n \), i.e.
  \[ (Ax, w_i) = (b, w_i) \quad \text{for } \forall w_i \in W = \text{span}[w_1, ..., w_m] \]

- 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 \( V = [v_1, ..., v_m], \ W = [w_1, ..., w_m] \)

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

\[
A V y = b - Ax_0 = r_0
\]

subject to the orthogonality constraints:

\[
W^T A V y = W^T r_0
\]

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. $x := x + V y$
7. EndDo
Projection and Eigen-Solvers

- The problem: Solve
  \[ Ax = \lambda x \quad \text{in} \quad \mathbb{R}^n \]
- As in linear solvers: at iteration \( i \) extract an approximate \( x_i \) from a subspace \( V = \text{span}[v_1, ..., v_m] \) of \( \mathbb{R}^n \)
- How? As on slides 22 and 26, impose constraints:
  \[ \lambda x - Ax \perp \text{subspace} \ W = \text{span}[w_1, ..., w_m] \] of \( \mathbb{R}^n \), i.e.
  \[ (*) \quad (Ax, w_i) = (\lambda x, w_i) \quad \text{for} \quad \forall \ w_i \in W = \text{span}[w_1, ..., w_m] \]
- This procedure is known as Rayleigh-Ritz
- Again projection can be orthogonal or oblique
Matrix representation

Let $V = [v_1, ..., v_m]$, $W = [w_1, ..., w_m]$

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

$$A V y = \lambda V y$$

subject to the orthogonality constraints:

$$W^T A V y = \lambda W^T V y$$

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
    - $2\pi$
    - $(f, g) = \int_{0}^{2\pi} f(x) \ g(x) \ dx \quad (Exercise)$
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 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 be piecewise polynomials defined over the mesh, e.g.

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

  $$a(\ u, \phi) = <f, \phi> \text{ for } \phi \in S$$

  - Galerkin (FEM) problem: Find $u_h \in S_h$ s.t.
    $$a(\ u_h, \phi_h) = <f, \phi_h> \text{ for } \phi_h \in S_h$$
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)