Linear Algebra Topics

- Gram-Schmidt Process
- QR Factorization
- QR Factorization with Column Pivoting
- Randomized QR Factorization with Column Pivoting
- Randomized GECP
- Randomized Subspace Iteration
- Lanczos subspace methods for truncated SVD

Gram-Schmidt Process (I)

Given $A \stackrel{def}{=} \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{pmatrix} \in \mathcal{R}^{m \times n}$ with $m \geq n$.
Gram-Schmidt Process computes an orthonormal basis
$\left\{ \mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_n \right\}$ for **span** $\begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{pmatrix}$

IDEA: MAKE $\left\{ \mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_j \right\}$ ORTHONORMAL BASIS FOR **span** $\begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_j \end{pmatrix}$ FOR $j = 1, \cdots, n$

Algorithm 1 Gram-Schmidt Process

Compute unit vector $\mathbf{q}_1 \in \mathcal{R}^m$ so that $\|\mathbf{a}_1\|_2 \, \mathbf{q}_1 = \mathbf{a}_1$.

for $j = 2, \cdots, n$ **do**

$$\mathbf{u}_j = \mathbf{a}_j - \sum_{k=1}^{j-1} \left(\mathbf{q}_k^T \mathbf{a}_j \right) \mathbf{q}_k, \quad \text{and} \quad \|\mathbf{u}_j\|_2 \, \mathbf{q}_j = \mathbf{u}_j$$

end for

In case $\mathbf{u}_j = \mathbf{0}$ for any $j \geq 2$, choose $\mathbf{q}_j$ to be any unit vector orthogonal to $\left\{ \mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_{j-1} \right\}$.

Gram-Schmidt Process (II)

Re-arrange terms in equations, for $j = 1, \cdots, n$

$$\mathbf{u}_j = \mathbf{a}_j - \sum_{k=1}^{j-1} \left(\mathbf{q}_k^T \mathbf{a}_j \right) \mathbf{q}_k, \quad \text{or}$$

$$\mathbf{a}_j = \sum_{k=1}^{j-1} \left(\mathbf{q}_k^T \mathbf{a}_j \right) \mathbf{q}_k + \|\mathbf{u}_j\|_2 \, \mathbf{q}_j$$

Gram-Schmidt Process (II)

Re-arrange terms in equations, for $j = 1, \cdots, n$

$$\mathbf{u}_j = \mathbf{a}_j - \sum_{k=1}^{j-1} \left(\mathbf{q}_k^T \mathbf{a}_j \right) \mathbf{q}_k, \quad \text{or}$$

$$\mathbf{a}_j = \sum_{k=1}^{j-1} \left(\mathbf{q}_k^T \mathbf{a}_j \right) \mathbf{q}_k + \|\mathbf{u}_j\|_2 \, \mathbf{q}_j \stackrel{def}{=} \begin{pmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_j \end{pmatrix} \cdot \begin{pmatrix} r_{1,j} \\ \vdots \\ r_{j,j} \end{pmatrix}.$$

In matrix form,

$$\begin{aligned} A &= \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_j & \cdots & \mathbf{q}_n \end{pmatrix} \begin{pmatrix} r_{1,1} & \cdots & r_{1,j} & \cdots & r_{1,n} \\ & \ddots & \vdots & \ddots & \vdots \\ & & r_{j,j} & \cdots & r_{j,n} \\ & & & \ddots & \vdots \\ & & & & r_{n,n} \end{pmatrix} \stackrel{def}{=} Q\,R. \end{aligned}$$

QR Factorization for $A \in \mathbb{R}^{m \times n}$ (I)

Partition matrix $A = \begin{pmatrix} \mathbf{a}_1 & \widehat{A}_1 \end{pmatrix}$, with $\mathbf{a}_1 \in \mathbb{R}^m$, $\widehat{A}_1 \in \mathbb{R}^{m \times (n-1)}$.

- Let $\mathbf{u}_1 \in \mathbb{R}^m$ be the unit vector in the Householder Reflection matrix $G_1 \stackrel{def}{=} \widehat{G}_1 = I - 2\,\mathbf{u}_1\,\mathbf{u}_1^T$ so that

$$\widehat{G}_1\,\mathbf{a}_1 = \begin{pmatrix} \pm\,\|\mathbf{a}_1\|_2 \\ \mathbf{0} \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} r_{1,1} \\ \mathbf{0} \end{pmatrix}. \quad \text{and}$$

$$G_1\,A = \begin{pmatrix} \widehat{G}_1\,\mathbf{a}_1 & \widehat{G}_1\,\widehat{A}_1 \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} r_{1,1} & r_{1,2} & \mathbf{r}_1^T \\ \mathbf{0} & \mathbf{a}_2 & \widehat{A}_2 \end{pmatrix}.$$

QR Factorization for $A \in \mathbb{R}^{m \times n}$ (II)

$$G_1\, A = \begin{pmatrix} r_{1,1} & r_{1,2} & \mathbf{r}_1^T \\ \mathbf{0} & \mathbf{a}_2 & \widehat{A}_2 \end{pmatrix}, \quad \widehat{A}_2 \in \mathbb{R}^{(m-1)\times(n-2)}.$$

- Let $\mathbf{u}_2 \in \mathbb{R}^{m-1}$ be the unit vector in $\widehat{G}_2 = I - 2\,\mathbf{u}_2\,\mathbf{u}_2^T$ so that

$$\widehat{G}_2\,\mathbf{a}_2 \;=\; \begin{pmatrix} \pm\,\|\mathbf{a}_2\|_2 \\ \mathbf{0} \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} r_{2,2} \\ \mathbf{0} \end{pmatrix}. \quad \text{With} \quad G_2 = \begin{pmatrix} 1 & \\ & \widehat{G}_2 \end{pmatrix},$$

$$G_2\, G_1\, A \;=\; \begin{pmatrix} r_{1,1} & r_{1,2} & \mathbf{r}_1^T \\ \mathbf{0} & \widehat{G}_2\,\mathbf{a}_2 & \widehat{G}_2\,\widehat{A}_2 \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & \overline{r}_1^T \\ 0 & r_{2,2} & r_{2,3} & \mathbf{r}_2^T \\ \mathbf{0} & \mathbf{0} & \mathbf{a}_3 & \widehat{A}_3 \end{pmatrix}.$$

QR Factorization for $A \in \mathbb{R}^{m \times n}$ (III)

- After $m-1$ Householder Reflections,

$$G_{m-1} \cdots G_2 \, G_1 \, A = \begin{pmatrix} r_{1,1} & \cdots & r_{1,j} & \cdots & r_{1,n} \\ & \ddots & \vdots & \ddots & \vdots \\ & & r_{j,j} & \cdots & r_{j,n} \\ & & & \ddots & \vdots \\ & & & & r_{n,n} \\ \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix}.$$

- Partition $(G_{m-1} \cdots G_2 \, G_1)^T \stackrel{def}{=} \begin{pmatrix} Q & Q^{\perp} \end{pmatrix}$ with $Q \in \mathcal{R}^{m \times n}$. Then $A = Q \, R$.

QR Factorization for $A \in \mathbb{R}^{m \times n}$ (IV)

Algorithm 2 QR Factorization

---

for $j = 1, \cdots, m-1$ **do**

Compute Householder Reflection $\widehat{H}_j$ on vector $A(j:m, j)$.

$A(j:m, j:n) = \widehat{H}_j \, A(j:m, j:n)$.

end for

Inconvenience: $r_{j,j} = 0$ in R if $A(j:m, j) = \mathbf{0}$ for some j.

QR Factorization with column pivoting

Algorithm 3 QR Factorization with column pivoting

for $j = 1, \cdots, m-1$ **do**

$\jmath = \mathbf{argmax}_{j \leq k \leq n} \, \|A(j:m, k)\|_2$

swap columns j and $\jmath$ in A.

Compute Householder Reflection $\widehat{H}_j$ on vector $A(j:m, j)$.

$A(j:m, j:n) = \widehat{H}_j \, A(j:m, j:n)$.

end for

Let Π be the accumulation of column permutations,

$$A\,\Pi = Q\,R.$$

rank$(A) = \#$ of non-zero diagonal entries in R.

QR with column pivoting (QRCP)

Inputs: $A \in \mathbf{R}^{m \times n}$, target rank k
Outputs: Π, Q, R such that $A\Pi = QR$

$\Pi = 1 : n, r_i = ||A(1 : m, i)||_2 \quad (1 \leqslant i \leqslant n)$
for $j = 1, k$ **do**
  Find $i_{\max} = argmax_{j \leqslant i \leqslant n} r_i$
  Swap jth column and $i_{\max}$th column in A, update Π
  $[\widehat{Q}, \widehat{R}] = qr(A(j : m, j))$
  $A(j : m, j + 1 : n) \leftarrow \widehat{Q}^T A(j : m, j + 1 : n)$
  Update $r_i = ||A(j + 1 : m, i)||_2 \quad (j + 1 \leqslant i \leqslant n)$
end for

Randomized QRCP (RQRCP)

Inputs: $A \in \mathbf{R}^{m \times n}$, target rank k, block size b, oversampling p
Outputs: Π, Q, R such that $A\Pi = QR$

$\Pi = 1 : n$, $\Omega \in \mathcal{N}(0,1)^{(b+p)\times m}$, $B = \Omega A \in \mathbf{R}^{(b+p)\times n}$
for $j = 1, k, b$ **do**
  $b = min(b, k - j + 1)$
  b-step partial QRCP on $B(:, j : n)$ to obtain b pivots
  Swap the corresponding columns in A, update Π
  $[\widehat{Q}, \widehat{R}] = qr(A(j : m, j : j + b - 1))$
  $A(j : m, j + b : n) \leftarrow \widehat{Q}^T A(j : m, j + b : n)$
  if $j + 1 - b < k$ **then**
    Update $B(:, j + b : n)$
  end if
end for

Instead of computing a random projection of the trailing matrix of A, we efficiently update B.

Idea of RQRCP

We recursively find b pivots on B and apply these pivots on A until we reach the target rank k.

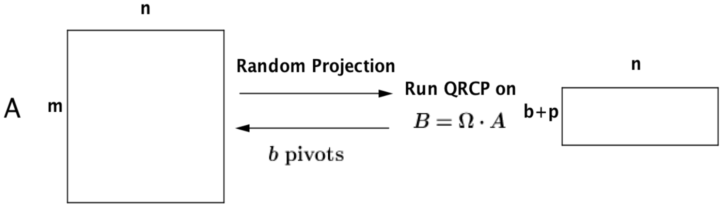


Figure: Use random projection to find b pivots in each block step.

Spectrum revealing QR factorization (SRQR)

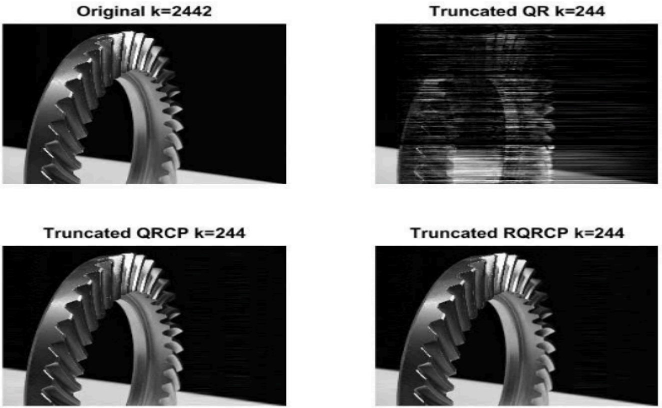
$A \in \mathbf{R}^{m \times n}$. Consider a partial QR factorization $A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix}$, with $R_{11} \in \mathbf{R}^{\ell \times \ell}$. Define $\widetilde{R} = \begin{pmatrix} R_{11} & R_{12} \end{pmatrix}$. For any $1 \leqslant k \leqslant \ell$, denote $\widetilde{R}_k$ the rank-k truncated SVD of $\widetilde{R}$. There exists permutation Π such that

$$\sigma_j(\widetilde{R}) \geqslant \frac{\sigma_j(A)}{\sqrt{1 + \left(\frac{\|R_{22}\|_2}{\sigma_j(\widetilde{R})}\right)^2}} = \frac{\sigma_j(A)}{\sqrt{1 + O\left(\left(\frac{\sigma_{\ell+1}(A)}{\sigma_j(A)}\right)^2\right)}} \quad (1 \leqslant j \leqslant \ell),$$

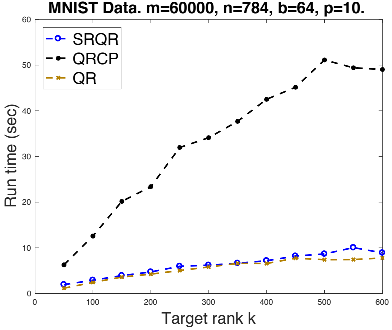
$$\left\| A\Pi - Q \begin{pmatrix} \widetilde{R}_k \\ 0 \end{pmatrix} \right\|_2 \leqslant \sigma_{k+1}(A) \sqrt{1 + O\left(\left(\frac{\sigma_{\ell+1}(A)}{\sigma_{k+1}(A)}\right)^2\right)}.$$

RQRCP can be used to compute an SRQR quickly in practice

Approximation effectiveness



Run times



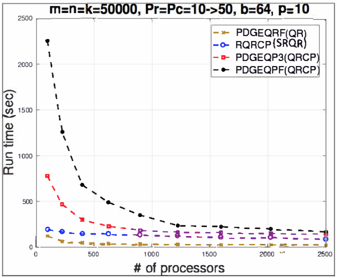


Figure: Comparative performance of the spectrum-revealing QR factorization (SSRQP) on sequential (leftmost) and distributed memory (rightmost) computers, respectively.

GEPP, GECP, and randomized GECP

- GEPP performs row pivoting in LU factorization, cheaper but less reliable.
- GECP performs complete pivoting in LU factorization, more expensive but more reliable.
- randomized GECP as cheap as GEPP, as reliable as GECP.

GEPP

Input: $n \times n$ matrix $\mathbf{A}$

Output: lower triangular L with unit diagonal, upper triangular U, row permutation Π_r.

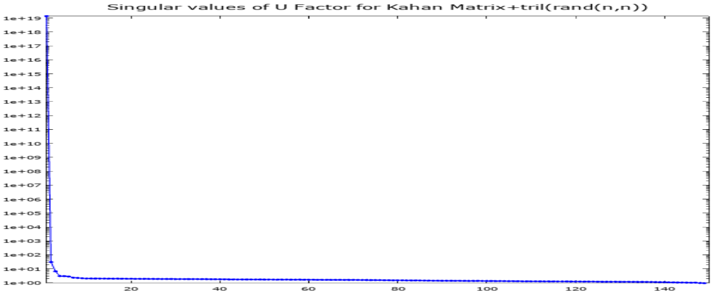
---

for $k = 1, \cdots, n-1$ **do**

- **compute** $\beta =$ **argmax**$_{k \leq j \leq n} |A(j,k)|$.
  swap rows k and β of A.
- **compute** $A(k+1:n,k) = A(k+1:n,k)/A(k,k)$;
- **update** $A(k+1:n,k+1:n) \;-=\; A(k+1:n,k) * A(k,k+1:n)$;

```
>> n = 150;
>> A = 2*eye(n)-tril(ones(n,n));
>> A(1:n-1,n) = 1; %Kahan Matrix
>> A = A + tril(rand(n,n)); %plus random tril
>> [L,U,P] = lu(A);
>> semilogy(svd(U),'b.-')
>> title('Singular values of U Factor for Kahan Matrix+tril(rand(n,n))','FontSize',15)
>> axis tight
>> x = randn(n,1); b = A * x;
>> norm(P-eye(n))
ans = 0
>> xx = A \ b;
>> norm(b-A*xx)/norm(b)
ans =  14.529
```

```
>> n = 150;
>> A = 2*eye(n)-tril(ones(n,n));
>> A(1:n-1,n) = 1; %Kahan Matrix
>> A = A + tril(rand(n,n)); %plus random tril
>> [L,U,P] = lu(A);
>> semilogy(svd(U),'b.-')
>> title('Singular values of U Factor for Kahan Matrix+tril(rand(n,n))','FontSize',15)
>> axis tight
>> x = randn(n,1); b = A * x;
>> norm(P-eye(n))
ans = 0
>> xx = A \ b;
>> norm(b-A*xx)/norm(b)
ans =  14.529
```



GE with column-norm based Complete Pivoting (GECP)

Input: $n \times n$ matrix $\mathbf{A}$
Output: lower triangular L with unit diagonal, upper triangular U, row permutation Π_r, and column permutation Π_c.

---

for $k = 1, \cdots, n-1$ **do**

- **compute** $\alpha =$ **argmax**$_{k \le j \le n} \|A(k:n, j)\|_2$.
  swap columns k and α of A.
- **compute** $\beta =$ **argmax**$_{k \le j \le n} |A(j, k)|$.
  swap rows k and β of A.
- **compute** $A(k+1:n, k) = A(k+1:n, k)/A(k, k)$;
- **update** $A(k+1:n, k+1:n) \;-=\; A(k+1:n, k) * A(k, k+1:n)$;

GE with Randomized Complete Pivoting

Input: $n \times n$ matrix A, sampling dimension $r > 0$
Output: lower and upper triangular L, U, permutations Π_r and Π_c.

---

sample $\Omega(i, j) \sim \mathcal{N}(0, 1)$ for all $1 \leq i \leq r$ and $1 \leq j \leq n$
compute $\Psi = \Omega \cdot A$
for $k = 1, \cdots, n-1$ **do**

- **compute** $\ell = \textbf{argmax}_{k \leq j \leq n} \|\Psi(:, j)\|_2$.
- **set** $\alpha = \begin{cases} k & \text{, if } \|\Psi(:, k)\|_2 \geq \|\Psi(:, \ell)\|_2 \text{,} \\ \ell & \text{, otherwise.} \end{cases}$
- **swap** columns k and α of A and Ψ.
- **compute** $\beta = \textbf{argmax}_{k \leq i \leq n} |A(i, k)|$.
  swap rows k and β of A.
- **compute** $A(k+1 : n, k) = A(k+1 : n, k)/A(k, k)$;
- $A(k+1 : n, k+1 : n) \; -= \; A(k+1 : n, k) \cdot A(k, k+1 : n)$;
- **update** $\Psi(:, k : n)$

Run times

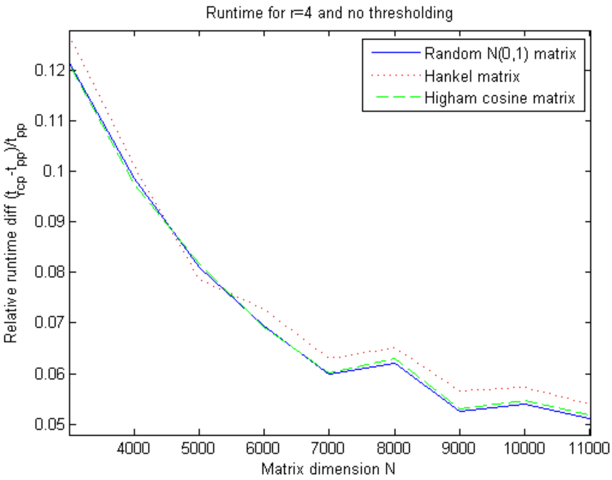


Figure: Relative run times of GERCP and GEPP Fortran code, each averaged over 10 separate trials

Subspace Iteration

- Basic Subspace Iteration as extension of Power Iteration, for approximate truncated SVD.
- Randomized Subspace Iteration

Basic Subspace Iteration

Input:	$m \times n$ matrix A with $n \leq m$, integers $0 < k \leq \ell < n$, and $n \times \ell$ start matrix Ω.
Output:	a rank-k approximation.

- Compute $Y = \left(AA^T\right)^q A\,\Omega$.
- Compute an orthogonal column basis Q for Y.
- Compute $B = Q^T A$.
- Compute B_k, the rank-k truncated SVD of B.
- Return QB_k.

Randomized Subspace Iteration

Input:	$m \times n$ matrix A with $n \leq m$, integers $0 < k \leq \ell$,
Output:	a rank-k approximation.

- Draw a random $n \times \ell$ start matrix Ω.
- Compute a rank-k approximation with Algorithm **??**.

Thm: Let $A = U\Sigma V^T$ be the SVD of A, and $0 \leq p \leq \ell - k$. Further let QB_k be a rank-k approximation computed by RSI. Given any $0 < \Delta \ll 1$, define

$$\mathcal{C}_\Delta = \frac{e\sqrt{\ell}}{p+1}\left(\frac{2}{\Delta}\right)^{\frac{1}{p+1}}\left(\sqrt{n-\ell+p}+\sqrt{\ell}+\sqrt{2\log\frac{2}{\Delta}}\right).$$

We must have for $j = 1, \cdots, k$,

$$\sigma_j\left(QB_k\right) \quad \geq \quad \frac{\sigma_j}{\sqrt{1+\mathcal{C}_\Delta^2\left(\frac{\sigma_{\ell-p+1}}{\sigma_j}\right)^{4q+2}}},$$

and

$$\|\|A - QB_k\|_2 \quad \leq \quad \sqrt{\sigma_{k+1}^2 + k\mathcal{C}_\Delta^2\sigma_{\ell-p+1}^2\left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^{4q}}.$$

with exception probability at most Δ.

RSI vs. Randomized Block Lanczos Algorithm