Unit 10: S4 Practical solution of the Hermitian Eigenvalue Problem

We know how to find eigenvector associated with the largest in magnitude eigenvalue, to find the next eigenvector v_1 that is associated with the next largest eigenvalue, we do:

$x_0 = x_0 / \| x_0 \|_2$ <normalize eigenvector x_0>

$v_1 =$ random vector

$v_1 = v_1 - x_0^H v_1 x_0$ <make sure the vector is orthogonal to x_0>

$v_1^{(0)} = v_1 / \| v_1 \|_2$

for $k = 0, \ldots$

$v_1 = A v_1^{(k)}$

$v_1^{(k+1)} = v_1 / \| v_1 \|_2$

end for

We can actually do better and compute both x_0 v_0, v_1 simultaneously:

$v_0 =$ random vector

$v_1 =$ random vector

$\left(\left(v_0^{(0)} \mid v_1^{(0)} \right), R \right) = QR\left(\left(v_0 \mid v_1 \right) \right)$

for $k = 0, \ldots$

$\left(\left(v_0^{(k+1)} \mid v_1^{(k+1)} \right), R \right) = QR\left(A \left(v_0^{(k)} \mid v_1^{(k)} \right) \right)$

end for

Now using Rayleigh quotient, we can calculate eigenvalue simultaneously:

$\hat{V} :=$ random $m \times n$ matrix

$(\hat{V}^{(0)}, R) = QR(\hat{V})$

$A^{(0)} = V^{(0)H} A V^{(0)}$

for $k = 0, \ldots$

$(\hat{V}^{(k+1)}, R) = QR(A \hat{V}^{(k)})$

$A^{(k+1)} = \hat{V}^{(k+1)H} A \hat{V}^{(k+1)}$

end for

This is called subspace iteration

A Simple QR algorithm

Subspace iteration

$\hat{V} = I$

for $k = 0, \ldots$

$(\hat{V}, \hat{R}) = QR(A\hat{V})$

$\hat{A} = \hat{V}^H A \hat{V}$

end for

QR algorithm

$V = I$

for $k = 0, \ldots$

$(Q, R) = QR(A)$

$A = RQ$

$V = VQ$

end for

Hw 10.2.1.1

Show that the algorithm on the right

$A^{(k+1)} = Q^{(k+1)H} A^{(k)} Q^{(k+1)}$ <similarity transformation>

Hw 10.2.1.2

Prove that, for all k:

- $\hat{A}^{(k)} = A^{(k)}$
- $\hat{R}^{(k)} = R^{(k)}$
- $\hat{V}^{(k)} = V^{(k)}$

Hw 10.2.1.3

In the above algorithms, show that for all k:

- $V^{(k)} = Q^{(0)} Q^{(1)} \cdots Q^{(k)}$
- $A^{k} = V^{(k)} R^{(k)} \cdots R^{(1)} R^{(0)}$ (A^k means raising to k^{th} power)

Assume that $Q^{(0)} = I$

Observations:

- $A^{(k+1)} = Q^{(k)H} A^{(k)} Q^{(k)}$ means that we are viewing $A^{(k)}$ in the new basis
- $A^{(k+1)} = (Q^{(0)} \cdots Q^{(k)})^H A Q^{(0)} \cdots Q^{(k)}$
  $= V^{(k)H} A V^{(k)}$

This means we can think of $A^{(k+1)}$ as the matrix A but viewed in a new basis

- In each step, we compute:

$$(Q^{(k+1)}, R^{(k+1)}) = QR(A^{(k)})$$

$$\Leftrightarrow (Q^{(k+1)}, R^{(k+1)}) = QR(A^{(k)}, I)$$

This suggest that in each iteration, we perform one step of subspace iteration, but with matrix $A^{(k)}$ and $V = I$:

$$(Q^{(k+1)}, R^{(k+1)}) = QR(A^{(k)} V)$$

Conclusion: From this insight, we can see that QR algorithm is identical to subspace iteration, except that at each step we reorient the problem (express it in a new basis) and restart it with $V = I$

A simple shifted QR algorithm

We know that $V^{(k)} = (v_0 | v_1 | \cdots | v_{m-2} | v_{m-1})$ will converge to $(x_0 | x_1 | \cdots | x_{m-2} | x_{m-1})$. And if we only consider the last (smallest) eigenvalue λ_{m-1}:

- The last column of $V^{(k)}$ converges to point in the direction of eigenvector associated with λ_{m-1}
- The rate of convergence of that vector is linear with a constant $|\lambda_{m-1}| / |\lambda_{m-2}|$

This insight suggest that we can accelerate convergence rate with shifting the matrix by an estimate of the eigenvalue that is smallest in magnitude (ρ)

And since we already have an estimation for v_{m-1}, we can use Rayleigh quotient to come up with ρ. However, that same value is already available at the last element of diagonal matrix $A^{(k)}$

```
V = I
for k = 0, ...
  (Q, R) = QR(A - α_{m-1,m-1} I)
  A = RQ + α_{m-1,m-1} I
  V = VQ
endfor
```

The algorithm in details:

Simple shifted QR algorithm

$A^{(0)} = A$, $V^{(0)} = I$, $R^{(0)} = I$

for $k = 0, \dots$

$\mu_k = \alpha^{(k)}_{m-1,m-1}$

$(Q^{(k+1)}, R^{(k+1)}) = QR(A^{(k)} - \mu_k I)$

$A^{(k+1)} = R^{(k+1)} Q^{(k+1)} + \mu_k I$

$V^{(k+1)} = V^{(k)} Q^{(k+1)}$

endfor

Hw 10.2.2.2

Show that:

- $A^{(k+1)} = Q^{(k+1)H} A^{(k)} Q^{(k+1)}$
- $A^{(k+1)} = V^{(k+1)H} A^{(k)} V^{(k+1)}$

$\Rightarrow$ This exercise confirms that the eigenvalues of $A^{(k)}$ equal the eigenvalues of A

Hw 10.2.2.3

Show that:

$(A - \mu_{k-1} I)(A - \mu_{k-2} I) \dots (A - \mu_k I)(A - \mu_0 I)$

$= \underbrace{Q^{(0)} Q^{(1)} \dots Q^{(k)}}_{\text{unitary matrix } V^{(k)}} \underbrace{R^{(k)} \dots R^{(1)} R^{(0)}}_{\text{upper triangular}}$

Deflating the problem

Hw 10.2.3.1

Let $A \in \mathbb{C}^{m \times m}$ be Hermitian matrix and $V \in \mathbb{C}^{m \times m}$ be a unitary matrix such that:

$$V^H A V = \begin{pmatrix} \Lambda_{00} & 0 \\ 0 & A_{11} \end{pmatrix}$$

If V_{00} and V_{11} are unitary matrices such that:

$$\left.\begin{aligned} V_{00}^H A_{00} V_{00} &= \Lambda_0 \\ V_{11}^H A_{11} V_{11} &= \Lambda_1 \end{aligned}\right\} \Rightarrow \text{these are orthogonal}$$

Then:

$$\left(V \begin{pmatrix} V_{00} & 0 \\ 0 & V_{11} \end{pmatrix}\right)^H A \left(V \begin{pmatrix} V_{00} & 0 \\ 0 & V_{11} \end{pmatrix}\right) = \begin{pmatrix} \Lambda_0 & 0 \\ 0 & \Lambda_1 \end{pmatrix}$$

The hw prove that at some point in the algorithm, $A^{(k)}$ will becomes a block diagonal matrix. Then we can proceed to find the spectral decompositions of the blocks on the diagonal

Since the last column of $V^{(k)}$ converges fastest, $A^{(k)}$ will have the form:

$$A^{(k)} = \begin{pmatrix} A^{(k)}_{00} & f^{(k)}_{01} \\ f^{(k)T}_{01} & \alpha^{(k)}_{m-1,m-1} \end{pmatrix}$$

where $f^{(k)}_{01}$ is small. In other words: $A^{(k)} \approx \begin{pmatrix} A^{(k)}_{00} & 0 \\ 0 & \alpha^{(k)}_{m-1,m-1} \end{pmatrix}$

Once $f^{(k)}_{01}$ is small enough, the problem deflate to a smaller problem, it will proceed with $A^{(k)}_{00}$.

What criteria should we use to deflate?

If the active matrix is $m \times m$, then:

$$\| a_{0,1} \|_1 \leq \varepsilon_{mach} \left(|\alpha_{0,0}^{(k)}| + \cdots + |\alpha_{m-1,m-1}^{(k)}| \right).$$

Remark 10.2.3.1

It is possible that deflation can happen anywhere in the matrix and one should check for that. However, it is most likely happen in the last row and column of the active part of the matrix

Cost of a simple QR algorithm

- $A \to QR$ (QR factorization): $\frac{4}{3}m^3$ flops
- $A = RQ$: m^3 (naive implementation) / $\frac{1}{3}m^3$
- $V = VQ$: $2m^3$ flops (calculate eigenvectors)

Thus the cost per iteration equals approximately $\left(\frac{4}{3} + \frac{1}{2}\right)m^3 = \frac{11}{6}m^3$ flops if only the eigenvalues are computed.

If the eigenvectors are also included, the ~~the~~ cost is $\frac{11}{6}m^3 + 2m^3 = \frac{23}{6}m^3$ flops.

Now consider deflation, let says it takes k iterations before an eigenvalue is found, then the problem deflates:

$$\sum_{i=m}^{1} k \frac{23}{6} m i^3 \approx k \frac{23}{6} \int_0^m x^3 dx = k \frac{23}{6} \frac{1}{4} x^4 \Big|_0^m = k \frac{23}{24} m^4$$

The bottom line is that the computation requires $O(m^4)$ flops, which is very expensive!

A Pratical Hermitian QR Algorithm

Reduction to tridiagonal form

Algorithm for reducing a Hermitian matrix A to tridiagonal form:

- Partition $A \to \begin{pmatrix} \alpha_{11} & * \\ a_{21} & A_{22} \end{pmatrix}$. $(*)$ means part of the matrix that is neither stored or updated.
- Update $[a_{21}, \tau] = \text{Housev1}(a_{21})$. This will:
  - Overwrites first element of a_{21} with $\pm \|a_{21}\|_2$
  - The remainder with all but first element of Householder vector u
- Update $A_{22} = H(a_{21}) A_{22} H(a_{21})$ ($H(x)$ means $(I - \frac{1}{\tau} u u^H)$ where u, τ are computed by $\text{Housev1}(x)$)
- Continue the process with updated A_{22}

It can be proven that:

$$A_{22} := H(a_{21}) A_{22} H(a_{21}) = A_{22} - u_{21} w_{21}^H - w_{21} u_{21}^H$$

Hermitian rank 2 update

This algorithm has 2 advantages:

- fewer computations
- doesn't generate intermediate result that is not Hermitian

Simple ~~Triang~~ Tridiagonal Reduction

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|cc} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ A_{20} & a_{21} & A_{22} \end{array}\right), \quad \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ t_2 \end{array}\right)$$

$[a_{21}, \tau_1] := \text{Housev1}(a_{21})$

$u_{21} = a_{21}$ with first element replaced with 1

Update $A_{22} := H(a_{21}) A_{22} H(a_{21})$ via the steps

$$\begin{cases} y_{21} := A_{22} u_{21} \\ \beta := u_{21}^H y_{21} / 2 \\ w_{21} := (y_{21} - \beta u_{21} / \tau_1) / \tau_1 \\ A_{22} := A_{22} - \text{tril}(u_{21} w_{21}^H + w_{21} u_{21}^H) \end{cases}$$

...

Cost of this algorithm:

Starting at $(m-1) \times (m-1)$ matrix A_{22}:

- $y_{21} := A_{22} u_{21}$ $\quad 2(m-1)^2$ flops
- $A_{22} := A_{22} - (u_{21} w_{21}^H + w_{21} u_{21}^H)$ $\quad 2(m-1)^2$ flops

Reducing $m \times m$ matrix A to tridiagonal form total cost:

$$\sum_{k=0}^{m-1} 4(m-k-1)^2 \quad \text{flops}$$

$$= 4 \sum_{j=0}^{m-1} j^2 \qquad \langle \text{subs } j = m-k-1 \rangle$$

$$\approx 4 \int_0^m x^2 \, dx$$

$$= \frac{4}{3} m^3 \text{ (flops)}$$

$\Rightarrow$ This equals (approximately) one QR factorization of matrix A.

Note:

The diagonal elements of a Hermitian matrix are real. Hence the tridiagonal has real values on its diagonal. A post process can be used to turn the subdiagonal elements to real values as well.

If this is done, then the subsequent computation, that computes eigenvalues and eigenvectors will be performed on real values, which is good!

Givens' Rotations

Givens' Rotation is a type of orthogonal matrix such that:

Given vector $x = \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} \in \mathbb{R}^2$, $\exists$ an orthogonal matrix G such that $G^T x = \begin{pmatrix} \pm \|x\|_2 \\ 0 \end{pmatrix}$ where $G = \begin{pmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{pmatrix}$, $\gamma^2 + \sigma^2 = 1$

The Householder transformation is an example of such matrix G.

It can be proven that $G^T G = I$, which means that Givens' Rotation is an orthogonal matrix.

Ahw 10.3.2.1

Formulas for γ and σ s.t:

$$\begin{pmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{pmatrix}^T \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} = \begin{pmatrix} \|x\|_2 \\ 0 \end{pmatrix}, \text{ where } \gamma^2 + \sigma^2 = 1$$

Take $\gamma = \frac{\chi_1}{\|x\|_2}$ and $\sigma = \frac{\chi_2}{\|x\|_2}$, then $\gamma^2 + \sigma^2 = 1$

and $\begin{pmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{pmatrix}^T \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} = \begin{pmatrix} \|x\|_2 \\ 0 \end{pmatrix}$

Remark 10.3.2.1

We only discuss real-valued Givens' rotations and how they transform real-valued vectors, since the output of reduction to tridiagonal form, after postprocessing, yields real-valued tridiagonal symmetric matrix

Remark 10.3.3.1

An important observation is that if A is tridiagonal, then $A \to QR$ (QR factorization) followed by $A := RQ$ again yields a tridiagonal matrix.

In other words, any QR algorithm previously discussed (simple, shifted, with deflation) when started with a tridiagonal matrix will generate a succession of tridiagonal matrices.

The implicit Q theorem

Definition 10.3.4.1 Upper Hessenberg matrix

A matrix is said to be upper Hessenberg if all entries below its first subdiagonal equal zero.

Example:

$$A = \begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,m-1} \\ \alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,m-1} \\ 0 & \alpha_{2,1} & \cdots & \alpha_{2,m-1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_{m-1,m-1} \end{pmatrix}$$

A tridiagonal matrix is a special case of Upper Hessenberg matrix.

Theorem 10.3.4.2 Implicit Q Theorem

Let B be upper Hessenberg and has only (real) positive elements on its first subdiagonal.
Assume exist unitary matrix Q s.t. $Q^H A Q = B$. Then Q and B are uniquely determined by A and first column of Q.

The Francis implicit QR step

Recall the shifted QR algorithm:

for $k = 0, \ldots$

$$(Q, R) := QR(A^{(k)} - \mu^{(k)} I) \quad \left.\begin{array}{l} \\ \\ \end{array}\right\} A^{(k+1)} = Q^T A^{(k)} Q$$

$$A^{(k+1)} := RQ + \mu^{(k)} I$$

end for

To better visualize this; the algorithm essentially does:

~~$G_{m-2}^T \cdots G_1^T G_0^T (A^{(k)} - \mu^{(k)} I) G_0$~~

$$A^{(k+1)} = \underbrace{G_{m-2}^T \cdots G_1^T G_0^T}_{Q^T} \underbrace{(A^{(k)} - \mu^{(k)} I)}_{\text{tridiagonal}} \underbrace{G_0 G_1 \cdots G_{m-2}}_{Q} + \mu^{(k)} I$$

$A^{(k+1)}$: tridiagonal

The implicit Q theorem said:

- Q is determine by G_0 (since only G_0 has 1st column of Q)
- If we use a different set of Givens' Rotation, and apply them to $A^{(k)}$ instead of $(A^{(k)} - \mu^{(k)} I)$:

$$\hat{G}_{m-2}^T \cdots \hat{G}_1^T \left(G_0^T A^{(k)} G_0\right) \hat{G}_1 \cdots \hat{G}_{m-2}$$

Such that
- the result is a tridiagonal matrix $A^{(k+1)}$
- the additional Rotations $G_1 \ldots G_{m-2}$ have the properties that the 1st column of $(G_0 \hat{G}_1 \ldots \hat{G}_{m-2})$ is the same as the 1st column of $(G_0 \ldots G_{m-2})$

Then the resulting $A^{(k+1)}$ is the same as the shifted $A^{(k+1)}$

That means we don't have to calculate $(A^{(k)} - \mu^{(k)} I)$, we only need to do ~~G~~ to find G_0 so that ~~apply to A give a resulted in~~ computed from $\begin{pmatrix} \alpha_{0,0} - \mu^{(k)} \\ \alpha_{1,0} \end{pmatrix}$, then proceed to form $A^{(k+1)}$ as tridiagonal matrix.

This is know as "introduce the bulge".

A complete algorithm

The algorithm follow these steps:

1. Apply a bunch of householder transformations to Hermitian matrix A:

$$A^{(0)} = H_{m-2} \cdots H_1 H_0 \, A \, \underbrace{H_0 H_1 \cdots H_{m-2}}_{Q \text{ (householder vectors as columns)}}$$

$A^{(0)}$: tridiagonal

2. Start iterative process:

Starting with: $A^{(0)} = Q^H A Q$

$$\Rightarrow A = Q A^{(0)} Q^H$$

After n iterations: $A = (QG_0 \ldots) \ldots G_0^T A^{(n)} G_0 \ldots (QG_0 \ldots)^H$

$$\Leftrightarrow A = (QG_0 \ldots) A^{(n)} (QG_0 \ldots)^H$$

If you look closely, we have done a spectral decomposition and $A^{(n)}$ is a diagonal matrix where the diagonal elements are eigenvalues and $(Q G_0 \dots)$ matrix consists of eigenvectors of A.

How to choose the shift μ_k?

The shift μ_k can be chosen to equal the last diagonal element $\alpha_{m-1,m-1}$. In practice, choosing the shift to be an eigenvalue of bottom right 2×2 matrix works better. (Wilkinson shift)

Cost of the complete algorithm?

- Let analyze this:
  - Reducing matrix A to tridiagonal form: $O(m^3)$ flops
  - Forming Q from householder vectors: $O(m^3)$ flops
  - Iterative process:
    - Apply a Givens' rotation to a pairs of columns of Q require $O(m)$ per rotation.
    - Each Francis implicit ~~step~~ QR step compute $O(n)$ Givens' rotations

⇒ Application of Givens rotations to Q cost $O(m^2)$ per iteration of the implicitly shift QR algorithm

Clarification: Each iteration, n Givens rotations are generated, applied to Q and Q^T. In the next iteration, another set of n Givens rotations are generated.

Typically, a few (2-3) iterations are ~~needed~~ per ~~eigenvalu~~ eigenvalue that is uncovered (when deflation is incorporated) meaning that $O(n)$ iterations are needed.

⇒ Total cost of Implicitly shift QR algorithm with a tridiagonal matrix that accumulates eigenvectors requires $O(m^3)$ flops