Chapter 3

Iterative Methods

3.1 Iterative method for large system of equation

In this section we consider an iterative method to find the solution of $A\mathbf{x} = \mathbf{b}$. We start from a splitting of A as $A = D - T$. First observe

$$\begin{aligned} (D-T)\mathbf{x} &= \mathbf{b} \\ D\mathbf{x} &= T\mathbf{x} + \mathbf{b} \\ \mathbf{x} &= D^{-1}T\mathbf{x} + D^{-1}\mathbf{b}. \end{aligned}$$

We now define a sequence $\mathbf{x}^{(k)}$ of approximation to $\mathbf{x}$ via

$$\begin{aligned} \mathbf{x}^{(k)} &= D^{-1}T\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b} \\ &= D^{-1}(D-A)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b} \\ &= (I - D^{-1}A)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b}. \end{aligned}$$

Jacobi method

Question. Does $\mathbf{x}^{(k)} \to \mathbf{x}$? and how fast? How to choose $D - T$? The idea is to choose D so that $D\mathbf{x} = \mathbf{b}$ can be solved easily. Diagonal of A is a often a good choice for D(called a **Jacobi method**). If D is diagonal, the Jacobi method is defined as

$$x_i^{(k)} = \left[\sum_{j\neq i} -a_{ij}x_j^{(k-1)} + b_i\right] \Big/ a_{ii}, \qquad i = 1, 2, \ldots, n. \tag{3.1}$$

If we use the splitting $A = D + L + U$, then the scheme is, in matrix form

$$\mathbf{x}^{(k)} = -D^{-1}(U + L)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b}. \tag{3.2}$$

This form is used for analysis only; instead (3.1) is used for actual implementation.

Convergence

Definition 3.1.1. We say a sequence of vectors $\mathbf{x}_k$ **converges to** $\mathbf{x}$ if for some norm $\|\cdot\|$

$$\lim_{k\to\infty} \|\mathbf{x}_k - \mathbf{x}\| \to 0.$$

Similarly, we say a sequence of matrices M_k **converges to** M if for some matrix norm $\|\cdot\|$

$$\lim_{k\to\infty} \|M_k - M\| \to 0.$$

Let us write the Jacobi algorithm as

$$\mathbf{x}^{(k+1)} = M\mathbf{x}^{(k)} + \tilde{\mathbf{b}}. \tag{3.3}$$

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= M\mathbf{x}^{(k)} + \tilde{\mathbf{b}} \\
&= M(M\mathbf{x}^{(k-1)} + \tilde{\mathbf{b}}) + \tilde{\mathbf{b}} \\
&= M^2\mathbf{x}^{(k-1)} + (M + I)\tilde{\mathbf{b}} \\
&= \cdots \\
&= M^{k+1}\mathbf{x}^{(0)} + (M^{(k)} + \cdots + M + I)\tilde{\mathbf{b}}.
\end{aligned}$$

So we need to see the behavior of $M^{(k)}$ as $k \to \infty$. Suppose $\|M\| < 1$ for some norm. Then the eigenvalues of $I - M$ are all nonzero and hence $I - M$ is invertible. We see

$$\begin{aligned}
(M^{(k)} + \cdots + M + I)(I - M) &= I - M^{(k+1)} \\
&\to I.
\end{aligned}$$

Hence the matrix sum $(I + M + \cdots + M^{(k)})$ converges to $(I - M)^{-1}$. More

precisely we see

$$\begin{aligned}(M^{(k)}+\cdots+M+I)-(I-M)^{-1} &= [(M^{(k)}+\cdots+M+I)(I-M)-I](I-M)^{-1}\\ \|(M^{(k)}+\cdots+M+I)-(I-M)^{-1}\| &\leq \|[(M^{(k)}+\cdots+M+I)(I-M)-I]\|\|(I-M)^{-1}\|\\ &\to 0.\end{aligned}$$

Hence we have

$$\sum_{i=0}^{\infty} M^k = (I-M)^{-1}.$$

Thus in this case,

$$\begin{aligned}\lim_{k\to\infty}\mathbf{x}^{(k+1)} &= \lim_{k\to\infty}(M^{k+1}\mathbf{x}^{(0)}+(M^{(k)}+\cdots+M+I)\tilde{\mathbf{b}})\\ &= (I-M)^{-1}\tilde{\mathbf{b}}\end{aligned}$$

for any initial guess $\mathbf{x}^0$. Since the exact solution $\mathbf{x}$ satisfies $\mathbf{x}=M\mathbf{x}+\tilde{\mathbf{b}}$, we subtract it from (3.3) to get

$$\mathbf{x}-\mathbf{x}^{(k)} = M(\mathbf{x}-\mathbf{x}^{(k-1)}). \tag{3.4}$$

Thus

$$\begin{aligned}\mathbf{x}-\mathbf{x}^{(k)} &= M(\mathbf{x}-\mathbf{x}^{(k-1)})\\ &= M^k(\mathbf{x}-\mathbf{x}^{(0)})\\ &\to 0.\end{aligned}$$

Definition 3.1.2. We say a matrix is **diagonally dominant** if

$$|a_{ii}| \geq \sum_{j\neq i}^{n}|a_{ij}|, \text{ for all } i=1,2,\cdots,n.$$

If the inequality is strict for all i, then we say **strictly diagonally dominant** and if the inequality is weak, we say it is **weakly diagonally dominant**

If A is strictly diagonally dominant, then the Jacobi method is convergent. Often weak diagonally dominant matrices are also convergent.

Another way to check $\rho(M)$. It is well known that

$$\rho(M) \leq \|M\|$$

for any norm.

Gauss-Seidel iterative method

Suggestion. In each computation of Jacobi method, $x_i^{(k-1)}$ was already updated for $j < i$. Why don't we utilize the most recent information? The result is

$$x_i^{(k)} = \left[\sum_{j=1}^{i-1} -a_{ij}x_j^{(k)} + \sum_{j=i+1}^{n} -a_{ij}x_i^{(k-1)} + b_i\right] \Big/ a_{ii}. \tag{3.5}$$

If we use the splitting $A = D + L + U$, then the scheme is, in matrix form

$$\mathbf{x}^{(k)} = -(D+L)^{-1}U\mathbf{x}^{(k-1)} + (D+L)^{-1}\mathbf{b}. \tag{3.6}$$

Block Jacobi or block Gauss-Seidel Method

$$\bar{x}_i^{(k)} = A_{ii}^{-1}\left(\sum_{j\neq i} -A_{ij}\bar{x}_j^{(k-1)} + \bar{k}_i\right)$$

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

$D = \begin{bmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & 0 \\ 0 & 0 & A_{33} \end{bmatrix}$ so that D contain as many element of A as possible.

Comparison between Jacobi and Gauss-Seidel Method

(1) Generally, Gauss-Seidel method is faster than Jacobi method

(2) On parallel machine Jacobi method is faster than Gauss-Seidel method

(3) When p, the number of processor is not large compared to n, the number of unknowns, one might try some combinations.

3.2 Sparsity

The iterative method is usually fast only if the matrix is sparse. Examples of sparse matrix arises in solving partial differential equations.

(1) Structured sparse matrix

(2) Tridiagonal matrix

(3) Block Tridiagonal matrix

(4) Unstructured sparse matrix

Issues: How to perform $A\mathbf{x}$ effectively(fast) ? We **do not want to do something like** because it is costly.

For $i = 1, 2, \ldots, n$
    $temp = 0$
    For $j = 1, 2, \ldots, n$
        $temp = temp + a_{ij}x_j$
        $b_i = temp$
    end
end

Total cost is n^2.

If A is tridiagonal, we have $a_{i,j} = 0$ for $j \notin \{i-1, i, i+1\}$. Thus we can proceed as follows: (with some modification when $j = 1$ or $j = n$)

For $i = 1, 2, \ldots, n$
    $temp = 0$
    For $j = i-1, i, i+1$
        $temp = temp + a_{i,j}x_j$
        $b_i = temp$
    end
end

Total cost is $3n$.

Unstructured sparse matrix

Read the book.

$$\begin{bmatrix} 8 & 7 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 7 \\ 0 & 9 & 0 & 0 & 8 & 0 \\ 8 & 0 & 7 & 0 & 0 & 7 \\ 0 & 0 & 0 & 9 & 0 & 8 \\ 0 & 7 & 9 & 0 & 0 & 0 \end{bmatrix}$$

we store row index vector $\mathbf{r}$, column index vector $\mathbf{c}$, and vector $\mathbf{v}$ of values:

$$\begin{aligned} \mathbf{r} &= (1,1,1,2,3,3,4,4,4,5,5,6,6) \\ \mathbf{c} &= (1,2,6,6,2,5,1,3,6,4,6,2,3) \\ \mathbf{v} &= (8,7,8,7,9,8,8,7,7,9,8,7,9). \end{aligned} \tag{3.7}$$

These are called **packed form**. Assume $\mathbf{x} = (x_1, x_2, \cdots, x_6)$(nonpacked form for simplicity). Then $\mathbf{y} = A\mathbf{x}$ can be computed by

$$\begin{aligned} y_1 &= v_1 x_{(c(1))} + v_2 x_{(c(2))} + v_3 x_{(c(3))} \\ y_2 &= v_4 x_{(c(4))} \\ y_3 &= v_5 x_{(c(5))} + v_6 x_{(c(6))} \\ &= \cdots \\ y_i &= \sum v_j x_{(c(j))} \end{aligned}$$

Example 3.2.1. Assume we have

$$\begin{bmatrix} 2.1 & 3.5 & 0 & 0 & 0 & 0 & 1.7 \\ 0 & 0 & 0 & 2.4 & 0 & 0 & 0 \\ 3.2 & 5 & 0 & 0 & 0 & 0 & 2.7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 9.1 \\ 1 & 3.5 & 0 & 0 & 0 & 0 & 0 \\ 2.4 & 0 & 0 & 6.2 & 5.2 & 0 & 0 \end{bmatrix}$$

We store row index vector $\mathbf{r}$, column index vector $\mathbf{c}$, and vector $\mathbf{v}$ of values:

$$\begin{aligned} \mathbf{r} &= (1,1,1,2,2,3,3,3,5,6,6,7,7,7) \\ \mathbf{c} &= (1,2,7,4,1,2,7,7,1,2,1,4,5) \\ \mathbf{v} &= (2.1,3.5,1.7,\cdots,2.4,6.2,5.2). \end{aligned} \tag{3.8}$$

Complete the multiplication $A\mathbf{x}$.

Other ways are also possible, see the book.

Tridiagonal matrix

$$A = \begin{bmatrix} a_1 & c_1 & & \\ b_2 & a_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & b_n & a_n \end{bmatrix} = \begin{bmatrix} \alpha_1 & 0 & & \\ b_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & 0 \\ & & b_n & \alpha_n \end{bmatrix} \begin{bmatrix} 1 & \gamma_1 & & \\ & 1 & \ddots & \\ & & \ddots & \gamma_{n-1} \\ & & & 1 \end{bmatrix}$$

$$\begin{aligned} \alpha_1 = a_1, \qquad \alpha_1 \cdot \gamma_1 &= c_1 \qquad \therefore\ \gamma_1 = \tfrac{c_1}{\alpha_1}, \qquad \alpha_i \neq 0 \\ b_2\gamma_1 + \alpha_2 &= a_2 \qquad \therefore\ \alpha_2 = a_2 - b_2\gamma_1 \\ \alpha_2\gamma_2 &= c_2 \qquad \therefore\ \gamma_2 = c_2/\alpha_2 \end{aligned}$$

In general, with $\gamma_0 = 0$, we have

$$\begin{aligned} \alpha_i &= a_i - b_i\gamma_{i-1}, \qquad i = 1, \ldots, n \\ \gamma_i &= c_i/\alpha_i, \qquad i = 1, \ldots, n-1, \quad 2(n-1) \quad \text{mult. and div.} \end{aligned}$$

Back substitution

$$\begin{aligned} L\mathbf{y} &= \mathbf{b} \cdots \text{forward-elimination} \qquad 2(n-1)+1 \\ U\mathbf{x} &= \mathbf{y} \cdots \text{back-substitution} \qquad n-1 \end{aligned}$$

$$\therefore \quad \text{Total} \quad 5n-4.$$

3.3 Iterative Refinement

Start with some initial guess $\mathbf{x}_0$ and try to improve it by solving the residual equation:

$$\begin{aligned} \mathbf{r} &= \mathbf{b} - A\mathbf{x}_0 \\ &= A(\mathbf{x} - \mathbf{x}_0). \end{aligned}$$

With $\mathbf{e} = \mathbf{x} - \mathbf{x}_0$ we have

$$A\mathbf{e} = \mathbf{r}. \tag{3.9}$$

We can solve (3.9) and have

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{e}.$$

But solving (3.9) is a s costly as solving the original problem and it again introduces some error. So we want to solve it approximately by a cheap method and set

$$\mathbf{x}_1 = \mathbf{x}_0 + \tilde{\mathbf{e}}.$$

This type of method is called an **iterative refinement**. We will not pursue this topic here.

3.4 Preconditioning

$$A\mathbf{x} = \mathbf{b} \tag{3.10}$$

can have an equivalent form(Preconditioning)

$$M_1 A M_2 \mathbf{y} = M_1 \mathbf{b}, \tag{3.11}$$

where $\mathbf{y}$ $M_2^{-1}\mathbf{x}$. If we can choose M_1, M_2 so that

$$\kappa(M_1 A M_2) << \kappa(A)$$

it would be cheaper to solve the latter system. We shall study some of these technique. Assume $M_2 = I$. Consider

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}. \tag{3.12}$$

If $P = D$ the diagonal of A, then it is called a **Jacobi preconditioner**

Example 3.4.1 (Diagonal preconditioner)**.**

$$A = \begin{bmatrix} 2 & 1 \\ 0.1 & 0.01 \end{bmatrix},\ A^{-1} = \frac{-1}{0.08}\begin{bmatrix} 0.01 & -1 \\ -0.1 & 2 \end{bmatrix}.$$

Thus the $\|A\|_\infty = 3$(maximum row sum) and $\|A^{-1}\|_\infty = 26.25$(maximum row sum). Thus $\kappa(A)\doteq 78.75$

$$D^{-1}A = \begin{bmatrix} 1 & 0.5 \\ 10 & 1 \end{bmatrix},\ (D^{-1}A)^{-1} = \frac{-1}{4}\begin{bmatrix} 1 & -0.5 \\ -10 & 1 \end{bmatrix}.$$

Thus $\|D^{-1}A\|_\infty = 11$(maximum row sum) and $\|(D^{-1}A)^{-1}\|_\infty = 11/4$. Thus $\kappa(D^{-1}A)\doteq 30.25$, while in book we have $\kappa(A)\doteq 62$ and $\kappa(D^{-1}A)\doteq 26$.

Thus the diagonal preconditioner is good in this case. In fact, it has the effect of equilibrating(balancing) the rows of a matrix.(This tactic seems useful when the matrix is diagonally dominant.)

ILU Preconditioner

When A is large and sparse what keeps us from using LU -decomposition?

LU are apt to be dense even if A is sparse.

One remedy is to replace the LU by its approximation $\tilde{L}\tilde{U}$. Strategy to compute $\tilde{L}\tilde{U}$:

(1) Allow entries of $\tilde{L}$ or $\tilde{U}$ is nonzero when the corresponding entry of A is nonzero.

(2) Specify a range of entries of $\tilde{L}$ and $\tilde{U}$ that can be nonzero(e.g, a few sub and super diagonals).

Any such $\tilde{L}\tilde{U}$ is called an **incomplete LU preconditioner-ILU**. See example 3.4.2 where $\kappa(A)\doteq 103$ but $\kappa((\tilde{L}\tilde{U})^{-1}A)\doteq 1.03$.

Using the Preconditioner

When we use the preconditioner P, we usually do not form $P^{-1}A$. Consider $P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}$. Instead of solving it, we observe an iterative scheme where we need the residual

$$\begin{aligned}\mathbf{r} &= P^{-1}A\mathbf{b} - P^{-1}A\mathbf{x}_0 \\ &= P^{-1}(\mathbf{b} - A\mathbf{x}_0).\end{aligned}$$

Thus we solve the following easier system

$$P\mathbf{r} = \mathbf{b} - A\mathbf{x}_0.$$

3.5 Krylov Space Methods

Many iterative method for solving $A\mathbf{x} = \mathbf{b}$ are based on the **Krylov Space**

$$K_k = span\{\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \cdots, A^{k-1}\mathbf{b}\}. \tag{3.13}$$

Since we have

$$K_n = \mathbb{R}^n$$

in many cases, it appears we can find a good approximation in K_k for $k << n$. Given an initial value $\mathbf{x}_0$ of the solution of $A\mathbf{x} = \mathbf{b}$, we define

$$V_k = \mathbf{x}_0 + K_k.$$

This is an **affine space**.

Minimizing the Residual

A natural way to define an approximation from V_k is to let $\mathbf{x}_k$ be the minimizer of the residual

$$\|\mathbf{b} - A\mathbf{x}\|.$$

Equivalently

$$\mathbf{x}_k = \mathrm{a}rgmin_{\mathbf{x}\in V_k}\|\mathbf{b} - A\mathbf{x}\|^2. \tag{3.14}$$

Let $\mathbf{z}_k = \mathbf{x}_k - \mathbf{x}_0$. Then $\mathbf{z}_k \in K_k$ and

$$\begin{aligned} \mathbf{z}_k &= \mathrm{a}rgmin_{\mathbf{z}\in K_k}\|\mathbf{b} - A(\mathbf{z} + \mathbf{x}_0)\|^2 \\ &= \mathrm{a}rgmin_{\mathbf{z}\in K_k}\|\mathbf{b} - A\mathbf{x}_0 - A\mathbf{z}\|^2 \\ &= \mathrm{a}rgmin_{\mathbf{z}\in K_k}\|\mathbf{r}_0 - A\mathbf{z}\|^2 \end{aligned} \tag{3.15}$$

where $\mathbf{r}_0$ is the initial error.

GMRES

We will solve (3.15) for $k = 1, 2, 3, \cdots$. But the point here is to stop for $k << n$ and for a large class of problems, this yields an acceptable solution, known as **GMRES**(General minimized residual method). One common stopping

criterion is that the relative residual

$$\frac{\|\mathbf{r}_k\|}{\|\mathbf{b}\|}$$

be less than some tolerance. How can we solve least square problem? It is natural to think QR decomposition. Define the Krylov matrix

$$\Gamma_k \equiv [\mathbf{b}\,|A\mathbf{b}\,|A^2\mathbf{b}\,|\cdots|A^{k-1}\mathbf{b}].$$

Since every vector in K_k is a linear combination of columns of Γ_k, there exists some vector $\mathbf{y}$ such that

$$\mathbf{x}_k - \mathbf{x}_0 = \Gamma_k \mathbf{y}.$$

In terms of (3.15) this problem becomes finding the solution $\mathbf{y}$ of

$$\mathbf{y} = \mathrm{a}rgmin_{\mathbf{y}\in\mathbb{R}^k}\|\mathbf{r}_0 - A\Gamma_k\mathbf{y}\|^2. \tag{3.16}$$

This is a standard minimization problem over $\mathbb{R}^k$ which could be solved by QR decomposition.

In practice this process is unstable and inefficient because Krylov matrix Γ_k is very ill-conditioned. A remedy is to use the Gram-Schmidt method to orthogonalize the bases of Krylov space. When the Gram-Schmidt method is used to the Krylov space, it is called the **Arnoldi process**:

Arnoldi -Algorithm:
Set $\mathbf{q}_1 = \frac{\mathbf{b}-A\mathbf{x}_0}{\|\mathbf{b}-A\mathbf{x}_0\|}$.
For $m = 1, 2, \ldots, k-1$
  Set $\mathbf{v}_{m+1} = A\mathbf{q}_m - \sum_{i=1}^{m} < \mathbf{q}_i, A\mathbf{q}_m > \mathbf{q}_i$.
  Set $\mathbf{q}_{m+1} = \dfrac{\mathbf{v}_{m+1}}{\|\mathbf{v}_{m+1}\|}$.
End loop

The resulting set $\{\mathbf{q}_1, \cdots, \mathbf{q}_k\}$ is an orthonormal basis for K_k. With $Span\Gamma_k = SpanQ_k$

$$Q_k \equiv [\mathbf{q}_1\,|\mathbf{q}_2\,|\mathbf{q}_3\,|\cdots|\mathbf{q}_k].$$

Thus we can rephrase as

$$\mathbf{y} = \mathrm{a}rgmin_{\mathbf{y}\in\mathbb{R}^k}\|\mathbf{r}_0 - AQ_k\mathbf{y}\|^2. \tag{3.17}$$

This behaves much better because now Q_k has normalized columns.

Example 3.5.1 (Simple GMRES with Arnoldi process)**.** Let $A = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 2 & 2 \\ -2 & 1 & 2 \end{bmatrix}$ and $\mathbf{b} = [2, 4, 1]$. We solve $A\mathbf{x} = \mathbf{b}$ with $\mathbf{x}_0 = 0$..... Then $\mathbf{r}_0 = \mathbf{b}$ and $\Gamma_1 = \mathbf{b}$. So the first vector is $\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\| = \frac{1}{\sqrt{21}}[2, 4, 1]^T$. Hence for $k = 1$, we must solve

$$\begin{aligned} \mathbf{y}_1 &= argmin_{\mathbf{y}\in\mathbb{R}^k} \|\mathbf{r}_0 - AQ_1\mathbf{y}\|^2 \\ &= argmin_{\mathbf{y}\in\mathbb{R}^k} \|\mathbf{r}_0 - A\mathbf{q}_1\mathbf{y}\|^2. \end{aligned}$$

Here $k = 1$ so $\mathbf{y}_1 \in \mathbb{R}$ and $A\mathbf{q}_1 = \frac{1}{\sqrt{21}}[7, 10, 2]^T \doteq (1.5275, 2.1822, 0.4364)^T$.

sol. We need to solve least square problem:

$$\begin{bmatrix} 1.527 \\ 2.183 \\ 0.436 \end{bmatrix} \mathbf{y}_1 = \begin{bmatrix} 2 \\ 4 \\ 1 \end{bmatrix}$$

We can use QR decomposition of $(1.5275, 2.1822, 0.4364)^T$, which is

$$Q \doteq \begin{bmatrix} -0.5659 \\ -0.8085 \\ -0.1617 \end{bmatrix} \tag{3.18}$$

and $R \doteq (-2.6992)$(From Matlab). To solve this least square problem, see QR in section 2.6, 2.7.

$$\mathbf{y}_1 \doteq R^{-1}Q^T\mathbf{r}_0.$$

We can continue to iterate. We need $\mathbf{v}_2$ (in Arnoldi) ... The only skipped details are QR decomposition.

Improving Efficiency

From the third line of Arnoldi algorithm, we see

$$\mathbf{v}_{m+1} = A\mathbf{q}_m - \sum_{i=1}^{m} < \mathbf{q}_i, A\mathbf{q}_m > \mathbf{q}_i.$$

Or

$$A\mathbf{q}_m = \sum_{i=1}^{m} \mathbf{q}_i^T A\mathbf{q}_m \mathbf{q}_i + \mathbf{v}_{m+1}. \tag{3.19}$$

The summation on the right hand side is nothing but a linear combination of columns of $\mathbf{q}_i, (i = 1, 2, \cdots, n)$ with coefficients $h_{im} = \mathbf{q}_i^T A\mathbf{q}_m$. Hence

$$\sum_{i=1}^{m} \mathbf{q}_i^T A\mathbf{q}_m \mathbf{q}_i = [\mathbf{q}_1 \,|\mathbf{q}_2 \,|\mathbf{q}_3 \,|\cdots|\mathbf{q}_m] Q_m^T A\mathbf{q}_m = Q_m Q_m^T A\mathbf{q}_m.$$

Thus

$$A\mathbf{q}_m = Q_m Q_m^T A\mathbf{q}_m + \|\mathbf{v}_{m+1}\|\mathbf{q}_{m+1}. \tag{3.20}$$

Hence with $m = k$

$$\begin{aligned} AQ_k &= [AQ_{k-1}, A\mathbf{q}_k] = Q_k[Q_k^T AQ_{k-1}, Q_k^T A\mathbf{q}_k] + \|\mathbf{v}_{k+1}\|\mathbf{q}_{k+1}\mathbf{e}_k^T \\ &= Q_k[Q_k^T AQ_k] + \|\mathbf{v}_{k+1}\|\mathbf{q}_{k+1}\mathbf{e}_k^T \\ &= [Q_k, \mathbf{q}_{k+1}] \begin{bmatrix} H_k \\ \|\mathbf{v}_{k+1}\|\mathbf{e}_k^T \end{bmatrix} \\ &= Q_{k+1}\tilde{H}_k, \end{aligned}$$

where $Q_{k+1} = [Q_k, \mathbf{q}_{k+1}]$ and $\tilde{H}_k$ is obtained by augmenting $[0, \cdots, 0, h_{k+1,k}]$, $h_{k+1,k} = \|\mathbf{v}_{k+1}\|$ after the last row of H_k. Substituting in (3.17) we get

$$\mathbf{y}_k = \mathrm{a}rgmin_{\mathbf{y}\in\mathbb{R}^k} \|\mathbf{r}_0 - Q_{k+1}\tilde{H}_k\mathbf{y}\|^2.$$

This can be simplified as

$$\begin{aligned} \mathbf{y}_k &= \mathrm{a}rgmin_{\mathbf{y}\in\mathbb{R}^k} \|\mathbf{r}_0 - Q_{k+1}\tilde{H}_k\mathbf{y}\|^2 \\ &= \mathrm{a}rgmin_{\mathbf{y}\in\mathbb{R}^k} \|Q_{k+1}(Q_{k+1}^T\mathbf{r}_0 - \tilde{H}_k\mathbf{y})\|^2 \\ &= \mathrm{a}rgmin_{\mathbf{y}\in\mathbb{R}^k} \|Q_{k+1}^T\mathbf{r}_0 - \tilde{H}_k\mathbf{y}\|^2 \\ &= \mathrm{a}rgmin_{\mathbf{y}\in\mathbb{R}^k} \|\rho\mathbf{e}_1 - \tilde{H}_k\mathbf{y}\|^2. \end{aligned}$$

Here $\mathbf{e} = (1, 0, \cdots, 0) \in \mathbb{R}^{k+1}$, and we used the form

$$Q_{k+1}^T\mathbf{r}_0 = \begin{bmatrix} Q_k^T \\ \mathbf{q}_{k+1}^T \end{bmatrix} \mathbf{r}_0 = \|\mathbf{r}_0\|\mathbf{e}_1 = \rho\mathbf{e}_1.$$

We can use QR decomposition to solve the least square problem

$$\mathbf{y} = argmin_{\mathbf{y}\in\mathbb{R}^k} \|\rho\mathbf{e}_1 - \tilde{H}_k\mathbf{y}\|^2 \tag{3.21}$$

and set $\mathbf{x}_k = \mathbf{x}_0 + Q_k\mathbf{y}_k$.

Remark 3.5.2.

$$\begin{aligned}
\mathbf{r}_k &= \mathbf{b} - A\mathbf{x}_k \\
&= \mathbf{b} - A(\mathbf{x}_0 + Q_k\mathbf{y}_k) \\
&= \mathbf{r}_0 - AQ_k\mathbf{y}_k \\
&= \mathbf{r}_0 - Q_{k+1}\tilde{H}_k\mathbf{y}_k \\
&= Q_{k+1}(\rho\mathbf{e}_1 - \tilde{H}_k\mathbf{y}_k).
\end{aligned}$$

So in checking the residual, we do not need to form $\mathbf{x}_k$ until the convergence is obtained.

GMRES -Algorithm:
Set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0, \rho = \|\mathbf{r}_0\|, \mathbf{q}_1 = \frac{\mathbf{r}_0}{\rho}$.
For $k = 1, 2, \ldots, n$
    For $i = 1, 2, \ldots, k$
        Set $h_{ik} = \mathbf{q}_i^T A\mathbf{q}_k$
    End i-loop
    Set $\mathbf{v}_{k+1} = A\mathbf{q}_k - \sum_{i=1}^{k} h_{ik}\mathbf{q}_i$.
    Set $h_{k+1,k} = \|\mathbf{v}_{k+1}\|$.
    Set $\mathbf{q}_{k+1} = \frac{\mathbf{v}_{k+1}}{h_{k+1,k}}$.
    Find the minimizer of $\|\rho\mathbf{e}_1 - \tilde{H}_k\mathbf{y}_k\|$ by finding QR -factorization of $\tilde{H}_k$
    If stopping criterion is met, then set $\mathbf{x}_k = \mathbf{x}_0 + Q_k\mathbf{y}_k$.
End k-loop

Do not confuse the Q with Q_k, which is a factor of Krylov matrix Γ_k.

Reorthogonalization

We should use the Modified Gram Schmidt in the Arnoldi. Even then we lose orthogonality. Thus we reorthogonalize the basis for Γ_k periodically (applying the MGS again to existing (semi) orthogonal basis).

Advantage: Applicable wide class of matrices without having special properties.

Disadvantage: We have to store a basis for K_k during the computation.(In the CPU memory! not in the disk)

Restarting

We periodically stop the process (empty the memory except the current $\mathbf{x}_k$), which we take as a new initial guess and start over.

Example 3.5.3. $A = [-1, 3; 2, 6; 2, 3]$

$$Q = \begin{pmatrix} -0.3333 & -0.8666 & 0.3714 \\ 0.6667 & -0.4952 & -0.5571 \\ 0.6667 & 0.0619 & 0.7428 \end{pmatrix} \quad R = \begin{pmatrix} 3.0000 & 5.0000 \\ 0 & -5.3852 \\ 0 & 0 \end{pmatrix}$$

Updating QR

See p. 606 Golub's book. Let B has QR decomposition $B = QR$ and B is updated by $B + \mathbf{u}\mathbf{v}^T$.

3.6 Eigenvalue Problems

$$A\mathbf{x} = \lambda\mathbf{x}.$$

Here λ is the root of a characteristic polynomial

$$c(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \cdots a_1\lambda + a_0.$$

3.6.1 Power method to find a few extreme eigenvalues and eigenvectors

Hypothesis :

(1) A is diagonalizable, i.e, $A\mathbf{v}^{(i)} = \lambda_i\mathbf{v}^{(i)}$ for linearly independent eigenvectors $\mathbf{v}^{(i)}$.

(2) $|\lambda_1| > |\lambda_j|$, $j = 2, \ldots, n$(i.e, λ_1 is a dominant eigenvalue.)

Power method consists of generating sequences of vector converging to an eigenvector and a sequence of scalar converging to dominant eigenvalue λ_1. We assume $\{\mathbf{v}^{(i)}\}_{i=1}^n$ is a normalized sets with respect to $\|\cdot\|_\infty$.

Let $\mathbf{y} \in \mathbb{C}^n$ be any vector whose projection against $\{\mathbf{v}^{(1)}\}$ is nonzero. Then one write $\mathbf{y} = \sum_{i=1}^n c_i \mathbf{v}^{(i)}$ with $c_1 \neq 0$(In practice, we may always assume this). With $\mathbf{y}^{(0)} = \mathbf{y}$, set

$$\mathbf{y}^{(k)} = A\mathbf{y}^{(k-1)}, \qquad k = 1, 2, \ldots$$

Then

$$\begin{aligned} \mathbf{y}^{(k)} &= A^k \mathbf{y}^{(0)} = A^k(\sum c_i \mathbf{v}^{(i)}) = \sum c_i \lambda_i^k \mathbf{v}^{(i)} \\ &= \lambda_1^k \left[c_1 \mathbf{v}^{(1)} + \sum_2^n c_i (\frac{\lambda_i}{\lambda_1})^k \mathbf{v}^{(i)} \right]. \end{aligned}$$

Since $|\lambda_1| > |\lambda_i|$, $i = 2, \ldots, n$,

$$\frac{y_j^{(k)}}{y_j^{(k-1)}} = \lambda_1 \frac{c_1 v_j^{(1)} + \varepsilon_j^{(k)}}{c_1 v_j^{(1)} + \varepsilon_j^{(k-1)}} \quad \text{if } y_j^{(k)} \neq 0.$$

Thus $y_j^{(k)}/y_j^{(k-1)} \to \lambda_1$ as $k \to \infty$.

But, this process is unstable numerically because $\mathbf{y}^{(k)} \to 0$ or ∞. To avoid it, we normalize the vector at each step. Since $\frac{\mathbf{y}^{(k-1)}}{\|\mathbf{y}^{(k-1)}\|_\infty} \approx \mathbf{v}^{(1)}$, we define

$$\mathbf{y}^{(k)} = A(\frac{\mathbf{y}^{(k-1)}}{\|\mathbf{y}^{(k-1)}\|_\infty}) \doteq A\mathbf{v}^{(1)} = \lambda_1 \mathbf{v}^{(1)}.$$

Then $\|\mathbf{y}^{(k)}\|_\infty \doteq |\lambda_1|$. If $\mathbf{y}^{(k-1)} \doteq \mathbf{y}^*$, then $\mathbf{y}^{(k)} \doteq \lambda_1 \mathbf{y}^*$ and $\lambda_1 \mathbf{y}^* \doteq A\mathbf{y}^*$, i.e, $\mathbf{y}^*$ is an approximate eigenvector corresponding to λ_1. Power method is useful when the matrix is sparse.

In practice we use the following variation(Rayleigh quotient):

for $k = 1, 2, \cdots, n$ do
$\mathbf{z}^{(k)} = A\mathbf{y}^{(k-1)}$
$\mathbf{y}^{(k)} = \frac{\mathbf{z}^{(k)}}{\|\mathbf{z}^{(k)}\|_2}$
$\lambda^{(k)} = {\mathbf{y}^{(k)}}^T A\mathbf{y}^{(k)}$

Roots of same modulus

We assume full linearly independent eigenvectors exist corresponding to the multiple eigenvalues.

The case $|\lambda_1| = |\lambda_2|$

(Faddeev, Faddeeva.) Suppose $|\lambda_1| = |\lambda_2| > |\lambda_3| \cdots$ and $\lambda_2 = \pm\lambda_1$. Then

$$\frac{y_j^{(k+2)}}{y_j^{(k)}} = \frac{\lambda_1^{k+2}[c_1\mathbf{v}^{(1)} + c_2(-1)^{k+2}\mathbf{v}^{(2)} + \varepsilon^{(k+2)}]_j}{\lambda_1^{k}[c_1\mathbf{v}^{(1)} + c_2(-1)^{k}\mathbf{v}^{(2)} + \varepsilon^{(k)}]_j} \to \lambda_1^2.$$

If three eigenvalues have same modulus, and $(\frac{\lambda_i}{\lambda_1})^3 = 1$, $i = 1, 2$ then

$$\frac{y_j^{(k+3)}}{y_j^{(k)}} = \frac{\lambda_1^{k+3}[c_1\mathbf{v}^{(1)} + c_2(\frac{\lambda_2}{\lambda_1})^{k+3}\mathbf{v}^{(2)} + c_3(\frac{\lambda_3}{\lambda_1})^{k+3}\mathbf{v}^{(3)} + \varepsilon(k+3)]_j}{\lambda_1^{k}[c_1\mathbf{v}^{(1)} + c_2(\frac{\lambda_2}{\lambda_1})^{k}\mathbf{v}^{(2)} + c_3(\frac{\lambda_3}{\lambda_1})^{k}\mathbf{v}^{(3)} + \varepsilon(k)]_j} \to \lambda_1^3.$$

If $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = -1$, try $y_j^{(k+6)}/y_j^{(k)}$.

3.6.2 Inverse power method

Assume $0 < |\lambda_1| < |\lambda_2| \leq |\lambda_3| \cdots$. Then the power method applied to A^{-1} lead to the computation of the smallest eigenvalue. Start from any $\mathbf{y}^{(0)}$ and define

$$A\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)}, \qquad k = 1, 2, \cdots .$$

Then

$$\begin{aligned} \mathbf{y}^{(k)} &= A^{-k}\mathbf{y}^{(0)} = A^{-k}(\sum c_i\mathbf{v}^{(i)}) = \sum c_i\lambda_i^{-k}\mathbf{v}^{(i)} \\ &= \lambda_1^{-k}\left[c_1\mathbf{v}^{(1)} + \sum_2^n c_i(\frac{\lambda_i}{\lambda_1})^{-k}\mathbf{v}^{(i)}\right]. \end{aligned}$$

Since $|\lambda_1| < |\lambda_i|$, $i = 2, \ldots, n$,

$$\frac{y_j^{(k)}}{y_j^{(k-1)}} = \frac{1}{\lambda_1}\frac{c_1v_j^{(1)} + \varepsilon_j^{(k)}}{c_1v_j^{(1)} + \varepsilon_j^{(k-1)}} \quad \text{if } y_j^{(k-1)} \neq 0.$$

Thus $y_j^{(k)}/y_j^{(k-1)} \to \frac{1}{\lambda_1}$ as $k \to \infty$.

In practice, we always normalize $\mathbf{y}^{(k-1)}$ each step, so that if $\|\mathbf{y}^{(k-1)}\|_\infty = |y_j^{(k-1)}| = 1$ for some j, then $y_j^{(k)} \approx \frac{1}{\lambda_i} y_j^{(k-1)}$.

The following algorithm is based on Rayleigh Quotient:

Algorithm: Inverse power method

For $k = 1, 2, \cdots$, do

Solve $A\mathbf{z}^{(k)} = \mathbf{y}^{(k-1)}$

$\mathbf{y}^{(k)} = \frac{\mathbf{z}^{(k)}}{\|\mathbf{z}^{(k)}\|_2}$

Solve $A\mathbf{w}^{(k)} = \mathbf{y}^{(k)}$

$\frac{1}{\lambda^{(k)}} = \mathbf{y}^{(k)}\mathbf{w}^{(k)} = \mathbf{y}^{(k)} A^{-1}\mathbf{y}^{(k)}$

Inverse power method with origin shift

Now we present a method to find any eigenvalue by shifting the origin and then using the inverse power method. Assume we know $q \approx \lambda_i$ and $\mathbf{y}^{(0)}$ has a nonzero component along $\mathbf{v}^{(i)}$. Then

$$\begin{aligned}
\mathbf{y}^{(k)} &= (A - qI)^{-1}\mathbf{y}^{(k-1)} \\
&\cdots \\
&= (A - qI)^{(-k)}\mathbf{y}^{(0)} = (A - qI)^{-k}(\sum c_i \mathbf{v}^{(i)}) = \sum_1^n \frac{c_i}{(\lambda_i - q)^k}\mathbf{v}^{(i)} \\
&= \frac{1}{(\lambda_i - q)^k}\left[c_1 \left(\frac{\lambda_i - q}{\lambda_1 - q}\right)^k \mathbf{v}^{(1)} + \cdots + c_i\mathbf{v}^{(i)} + \cdots + c_n \left(\frac{\lambda_i - q}{\lambda_n - q}\right)^k \mathbf{v}^{(n)}\right] \\
&\quad \text{where we assumed } \lambda_i \text{ is closest to } q \\
&= \frac{c_i}{(\lambda_i - q)^k}[\mathbf{v}^{(i)} + \varepsilon^{(k)}].
\end{aligned}$$

$$\therefore \qquad \frac{y_j^{(k)}}{y_j^{(k-1)}} \approx \frac{1}{\lambda_i - q}, \qquad j = 1, \ldots, n.$$

This method finds an eigenvalue closest to q. Thus inverse power method with origin shift useful when one knows an approximate value.

In practice, we use the following variation:

Algorithm: Inverse power method with origin shift

For $k = 1, 2, \cdots$, do

Solve $(A - qI)\mathbf{z}^{(k)} = \mathbf{y}^{(k-1)}$

$\mathbf{y}^{(k)} = \frac{\mathbf{z}^{(k)}}{\|\mathbf{z}^{(k)}\|_2}$

Solve $(A - qI)\mathbf{w}^{(k)} = \mathbf{y}^{(k)}$

$\frac{1}{\lambda^{(k)} - q} = \mathbf{y}^{(k)}\mathbf{w}^{(k)} = \mathbf{y}^{(k)}(A - qI)^{-1}\mathbf{y}^{(k)}$

Remark 3.6.1. One can use an updated shift by taking $q \approx \lambda^{(k)}$. This yields a rapid convergence.

Advantage.

(1) We can find any eigenvalue.

(2) Each iteration takes more time than the direct power method. However, if we have an approximate value of λ_i, the inverse power method will yields λ_i in a few steps.