Chap 7. Computation with Matrices

7.1 Introduction

In numerical analysis, there is a survival of the fittest, and we describe some ideas that have survived.

① Techniques for solving $Ax=b$.

Elimination is a perfect algorithm, except when the particular problem has special properties — as almost every problem has. For sparse matrices, we develop iterative methods for solving $Ax=b$. The object is to get close more quickly than elimination. In some problems, that can be done; in many others, elimination is safer and faster if it takes advantage of the zeros. The competition is far from over.

② Techniques for solving $Ax=\lambda x$.

Until recently no one knew how to solve the eigenvalue problem. Dozens of algorithms have been suggested. Two or three ideas have superseded others: the QR algorithms, the power methods, and the preprocessing of a symmetric matrix to make it tridiagonal. The last is a direct method, but it produces a much simpler matrix to use in the iterative steps.

③ The Condition Number of a Matrix.

The condition number measures the sensitivity of A by multiplying the norms of A and A^{-1}.

The matrices in this chapter are square.

7.3 Computation of Eigenvalues

The ordinary power method starts with u_0 and each step $u_{k+1} = A u_k$ is a matrix-vector multiplication. $\Rightarrow u_k = A^k u_0$. The multiplication by A should be easy, i.e., a large matrix A should be sparse. Assuming A has a full set of eigenvectors $x_1, \cdots, x_n$, the vector u_k will be

$$u_k = c_1 \lambda_1^k x_1 + \cdots + c_n \lambda_n^k x_n.$$

Suppose $|\lambda_1| \le |\lambda_2| \le \cdots \le |\lambda_{n-1}| < |\lambda_n|$. When u_0 contained some component of x_n, i.e., $c_n \neq 0$, the component $c_n x_n$ will dominate in u_k:

$$\frac{u_k}{(\lambda_n)^k} = c_1 \left(\frac{\lambda_1}{\lambda_n}\right)^k x_1 + \cdots + c_{n-1}\left(\frac{\lambda_{n-1}}{\lambda_n}\right)^k x_{n-1} + c_n x_n$$

The convergence factor is the ratio $r = |\lambda_{n-1}|/|\lambda_n|$. Often we divide each u_k by its first component α_k before taking the next step: $u_{k+1} = A u_k / \alpha_k$, where α_k will approach λ_n.

Ex 1:

$u_k \to \begin{bmatrix} 2/3 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 0.667 \\ 0.333 \end{bmatrix}$ when $A = \begin{bmatrix} 0.9 & 0.2 \\ 0.1 & 0.8 \end{bmatrix}$

$u_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $u_1 = \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}$, $u_2 = \begin{bmatrix} 0.83 \\ 0.17 \end{bmatrix}$, $u_4 = \begin{bmatrix} 0.781 \\ 0.219 \end{bmatrix}$, $u_5 = \begin{bmatrix} 0.\ldots \\ 0.253 \end{bmatrix}$

If $r = |\lambda_{n-1}|/|\lambda_n|$ is close to 1, then convergence is slow

If $r = 1$ (i.e., $|\lambda_{n-1}| = |\lambda_n|$), then convergence will not occur.

There are several ways to get around this limitation.

① The block power method works with several vectors at once. If we multiply p orthonormal vectors by A, and then apply Gram-Schmidt to orthogonalize them again, the convergence ratio becomes $r' = |\lambda_{n-p}|/|\lambda_n|$. We obtain approximations to p different eigenvalues and their eigenvectors.

② The inverse power method operates with A^{-1} instead of A. A single step is $v_{k+1} = A^{-1}v_k$ by solving $Av_{k+1} = v_k$. (Save the factors L and U!) Now we converge to the smallest eigenvalue λ_1 and its eigenvector x if $|\lambda_1| < |\lambda_2|$. Often it is λ_1 that is wanted in the applications.

③ The shifted inverse power method is best of all. Replace A by $A - \alpha I$. Each eigenvalue is shifted by α, and the convergence factor for the inverse method will change to $r'' = |\lambda_1 - \alpha| / |\lambda_2 - \alpha|$. If α is a good approximation to λ_1, r'' will be very small and the convergence is enormously accelerated. Each solves $(A - \alpha I) w_{k+1} = w_k$

$$w_k = \frac{c_1 x_1}{(\lambda_1 - \alpha)^k} + \frac{c_2 x_2}{(\lambda_2 - \alpha)^k} + \cdots + \frac{c_n x_n}{(\lambda_n - \alpha)^k}$$

When α is close to λ_1, the first term dominates. If a good approximation to any eigenvalue has been computed by another algorithm (such as QR), this is a good way to find the eigenvector.

If λ_1 is not already approximated, the shifted inverse power method has to generate its own choice of α. We can vary $\alpha = \alpha_k$ at every step: $(A - \alpha_k I) w_{k+1} = w_k$. When A is symmetric, a very accurate choice is the Rayleigh quotient:

$$\text{shift by } \alpha_k = R(w_k) = \frac{w_k^T A w_k}{w_k^T w_k}$$

This quotient $R(x)$ has a minimum at the true eigenvector x_1. The error $\lambda_1 - \alpha_k$ is roughly the square of the error in the eigenvector. The convergence factors $|\lambda_1 - \alpha_k| / |\lambda_2 - \alpha_k|$ converge to zero. Then these Rayleigh quotient shifts give cubic convergence of α_k to λ_1.

Tridiagonal and Hessenberg Forms

- The power method is reasonable only for a large sparse matrix. Our goal is to find a simple way to create zeros. After computing a similar matrix $Q^{-1}AQ$ with more zeros than A, we don't go back to the power method. The QR algorithm is a much more powerful variant. (The shifted inverse power method has its place at the very end, in finding the eigenvector.) The first step is to produce quickly many zeros, using Q. If A is symmetric, then so is Q^TAQ. (Q: an orthogonal matrix)
- The Hessenberg form accepts one nonzero diagonal below the main diagonal. If a Hessenberg matrix is symmetric, it only has three nonzero diagonals. A Householder matrix is a reflection matrix determined by one vector v:

Householder matrix: $$H = I - 2\cdot\frac{v\cdot v^T}{\|v\|^2} = I - 2uu^T,$$

where $u = v/\|v\|$. H is both symmetric and orthogonal.

$$H^TH = (I-2uu^T)(I-2uu^T) = I - 4uu^T + 4uu^Tuu^T = I$$

Thus $H = H^T = H^{-1}$.

- Householder's plan was to produce zeros with these matrices. Its success depends on the following identity $Hx = -\alpha z$:

TE Suppose $z = (1,0,\cdots,0)^T$, $\alpha = \|x\|$, and $v = x + \alpha z$. Then $Hx = -\alpha z = (-\alpha, 0, \cdots, 0)^T$.

<Pf> $$Hx = x - \frac{2v\cdot v^T}{\|v\|^2}x = x - (x+\alpha z)\frac{2(x+\alpha z)^Tx}{(x+\alpha z)^T(x+\alpha z)}$$

$$= x - (x+\alpha z) \quad (\because x^Tx = \alpha^2)$$

$$= -\alpha z$$ ∎

This identity can be used right away, on the first column of A. The final $Q^{-1}AQ$ is allowed one nonzero diagonal below the main diagonal (Hessenberg form). Therefore *only the entries strictly below the diagonal will be involved*:

$$x = \begin{bmatrix} a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix}, \quad z = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad Hx = \begin{bmatrix} -\sigma \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{3}$$

At this point Householder's matrix H is only of order $n-1$, so it is embedded into the lower right-hand corner of a full-size matrix U_1:

$$U_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & & & & \\ 0 & & H & & \\ 0 & & & & \\ 0 & & & & \end{bmatrix} = U_1^{-1}, \quad \text{and} \quad U_1^{-1}AU_1 = \begin{bmatrix} a_{11} & * & * & * & * \\ -\sigma & * & * & * & * \\ \mathbf{0} & * & * & * & * \\ \mathbf{0} & * & * & * & * \\ \mathbf{0} & * & * & * & * \end{bmatrix}.$$

The first stage is complete, and $U_1^{-1}AU_1$ has the required first column. At the second stage, x consists of the last $n-2$ entries in the second column (three bold stars). Then H_2 is of order $n-2$. When it is embedded in U_2, it produces

$$U_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & & & \\ 0 & 0 & & H_2 & \\ 0 & 0 & & & \end{bmatrix} = U_2^{-1}, \quad U_2^{-1}\left(U_1^{-1}AU_1\right)U_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ \mathbf{0} & * & * & * & * \\ \mathbf{0} & \mathbf{0} & * & * & * \\ \mathbf{0} & \mathbf{0} & * & * & * \end{bmatrix}.$$

U_3 will take care of the third column. For a 5 by 5 matrix, the Hessenberg form is achieved (it has six zeros). In general Q is the product of all the matrices $U_1U_2\cdots U_{n-2}$, and the number of operations required to compute it is of order n^3.

Example 2

(to change $a_{13} = a_{31}$ to zero)

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \quad x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad v = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad H = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.$$

Embedding H into Q, the result $Q^{-1}AQ$ is tridiagonal:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad Q^{-1}AQ = \begin{bmatrix} 1 & -1 & \mathbf{0} \\ -1 & 0 & 1 \\ \mathbf{0} & 1 & 1 \end{bmatrix}.$$

Two other applications of the Householder matrix H

① The Gram-Schmidt factorization $A = QR$.

The first step is to work with the first column of A:

$$x = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix}, \quad z = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow \quad v = x + \|x\| z$$

$$H_1 = I - 2\frac{v \cdot v^T}{\|v\|^2}$$

The first column of $H_1 A$ equals $-\|x\| z$, which is also the first column of R. The second step works with the second column of $H_1 A$, from the pivot on down, and produces an $H_2 H_1 A$ which is zero below that pivot. The result of $(n-1)$ steps is an upper triangular R.

$$H_{n-1} H_{n-2} \cdots H_1 A = R \Rightarrow A = (H_1 \cdots H_{n-1}) R = QR$$

The product $Q = H_1 \cdots H_{n-1}$ can be stored in the factored form (keep only the v's) and never computed explicitly.

② The Singular Value Decomposition $U^T A V = \Sigma$.

The first step is as above. The next step is to multiply on the right by an $H^{(1)}$, which produces zeros as indicated along the first row:

$$A \to H_1 A = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} \to H_1 A H^{(1)} = \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix}.$$

Repeating similar steps produces the bidiagonal form

$$H_2 H_1 A H^{(1)} = \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & * \\ 0 & 0 & * & * \end{bmatrix} \quad \text{and}$$

$$H_2 H_1 A H^{(1)} H^{(2)} = \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \end{bmatrix}.$$

The QR Algorithm for Computing Eigenvalues

It starts with $A_0 = Q_0 R_0$ and reverses the factors: $A_1 = R_0 Q_0$.
A_1 is similar to A_0 ($\because Q_0^{-1} A_0 Q_0 = Q_0^{-1}(Q_0 R_0) Q_0 = A_1$)
The process continues with no change in the eigenvalues.

All A_k are similar: $\underline{A_k = Q_k R_k \text{ then } A_{k+1} = R_k Q_k}$

Almost always A_k approaches a triangular form. Its diagonal entries approach its eigenvalues. If there was some processing to a tridiagonal form A_0, then A_0 is connected to the original A by $Q^T A Q = A_0$.

To speed up the QR algorithm, we must allow shifts to $A_k - \alpha_k I$, and the QR factorization should be very quick.

① The Shifted Algorithm. If α_k is close to an eigenvalue,

$$A_k - \alpha_k I = Q_k R_k \text{ and then } A_{k+1} = R_k Q_k + \alpha_k I$$

A_{k+1} is similar to A_k (always the same eigenvalues):

$$Q_k^{-1} A_k Q_k = Q_k^{-1}(Q_k R_k + \alpha_k I) Q_k = R_k Q_k + \alpha_k I = A_{k+1}.$$

The (n,n) entry of A_k is the first to approach an eigenvalue. The entry is chosen for α_k. Normally it produces quadratic convergence, and in the symmetric case even cubic convergence to the smallest eigenvalue. After three or four steps,

$$A_k = \left[\begin{array}{ccc|c} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ \hline 0 & 0 & \epsilon & \lambda_1' \end{array}\right], \quad \text{with } \epsilon \ll 1.$$

We accept $\lambda_1' \approx \lambda_1$. To find λ_2, we continue with the upper-left $(n-1)\times(n-1)$ submatrix. Another two steps are sufficient to find λ_2. ... To find the eigenvectors, a single inverse power step is usually good enough.

2. When A_0 is tridiagonal or Hessenberg, each QR step is very fast. The Gram–Schmidt process (factoring into QR) takes $O(n^3)$ operations for a full matrix A. For a Hessenberg matrix this becomes $O(n^2)$, and for a tridiagonal matrix it is $O(n)$. Fortunately, each new A_k is again in Hessenberg or tridiagonal form:

Q_0 is Hessenberg
$$Q_0 = A_0 R_0^{-1} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{bmatrix} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}.$$

You can easily check that this multiplication leaves Q_0 with the same three zeros as A_0. *Hessenberg times triangular is Hessenberg.* So is triangular times Hessenberg:

A_1 is Hessenberg
$$A_1 = R_0 Q_0 = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{bmatrix}.$$

The symmetric case is even better, since $A_1 = Q_0^{-1} A_0 Q_0 = Q_0^{\mathrm{T}} A_0 Q_0$ stays symmetric. By the reasoning just completed, A_1 is also Hessenberg. So A_1 must be *tridiagonal*. The same applies to $A_2, A_3, \ldots$, and *every QR step begins with a tridiagonal matrix.*

The last point is the factorization itself, producing the Q_k and R_k from each A_k (or really from $A_k - \alpha_k I$). We may use Householder again, but it is simpler to annihilate each subdiagonal element in turn by a "plane rotation" P_{ij}. The first is P_{21}:

Rotation to kill a_{21}
$$P_{21} A_k = \begin{bmatrix} \cos\theta & -\sin\theta & & \\ \sin\theta & \cos\theta & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} a_{11} & * & * & * \\ a_{21} & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{bmatrix} \tag{7}$$

The (2, 1) entry in this product is $a_{11} \sin\theta + a_{21} \cos\theta$, and we choose the angle θ that makes this combination zero. The next rotation P_{32} is chosen in a similar way, to remove the (3, 2) entry of $P_{32} P_{21} A_k$. After $n - 1$ rotations, we have R_0:

Triangular factor
$$R_k = P_{nn-1} \cdots P_{32} P_{21} A_k. \tag{8}$$

7.4 Iterative Methods for $Ax=b$

An iterative method is easy to invert, by splitting the matrix A.
If $A=S-T$, then $Ax=b$ is the same as $Sx=Tx+b$.

Iteration: $Sx_{k+1}=Tx_k+b$

① The new vector x_{k+1} should be easy to compute. Thus S should be a simple (and invertible) matrix – diagonal or triangular.

② The sequence x_k should converge to the true solution.

Error equation: $Se_{k+1}=Te_k$, where $e_k=x_k-x$.

It starts with the initial error e_0, and $e_k=(S^{-1}T)^k e_0$.

The convergence $x_k\to x$ exactly when $e_k\to 0$.

7H

The iterative method is convergent if and only if every eigenvalue of $S^{-1}T$ satisfies $|\lambda|<1$. Its rate of convergence depends on the maximum size of $|\lambda|$:

Spectral radius "rho": $\rho(S^{-1}T)=\max_i|\lambda_i|$

A typical solution to $e_{k+1}=S^{-1}Te_k$ is a combination of eigenvectors:

$$e_k=c_1\lambda_1^k x_1+\cdots+c_n\lambda_n^k x_n.$$

The largest $|\lambda_i|$ will eventually be dominant, so the spectral radius $\rho=|\lambda_{max}|$ will govern the rate at which e_k converges to zero. We certainly need $\rho<1$. We start with three possibilities

① S = diagonal part of A (Jacobi's method)

② S = triangular part of A (Gauss-Seidel method)

③ S = combination of ①&② (Successive overrelaxation or SOR)

S is also called a preconditioner, and its choice is crucial in numerical analysis.

Ex 1: (Jacobi) Here S is the diagonal part of A

$$A=\begin{bmatrix}2&-1\\-1&2\end{bmatrix},\quad S=\begin{bmatrix}2&0\\0&2\end{bmatrix},\quad T=\begin{bmatrix}0&1\\1&0\end{bmatrix},\quad S^{-1}T=\begin{bmatrix}0&\frac{1}{2}\\\frac{1}{2}&0\end{bmatrix}$$

For $x=\begin{bmatrix}v\\w\end{bmatrix}$, the Jacobi step is $Sx_{k+1}=Tx_k+b$

$$\begin{aligned}2v_{k+1}&=w_k+b_1\\2w_{k+1}&=v_k+b_2\end{aligned}\quad\text{or}\quad\begin{bmatrix}v\\w\end{bmatrix}_{k+1}=\begin{bmatrix}0&\frac{1}{2}\\\frac{1}{2}&0\end{bmatrix}\begin{bmatrix}v\\w\end{bmatrix}_k+\begin{bmatrix}b_1/2\\b_2/2\end{bmatrix}$$

The decisive matrix $S^{-1}T$ has eigenvalues $\pm\frac{1}{2}$, which means that the error is cut in half at every step.

The Gauss-Seidel method starts using each component of the new x_{k+1} as soon as it is computed and thus requires only half as much space as the Jacobi.

$$\begin{cases}\text{New } x_1: & a_{11}(x_1)_{k+1}=(-a_{12}x_2-a_{13}x_3-\cdots-a_{1n}x_n)_k+b_1\\ \text{New } x_2: & a_{22}(x_2)_{k+1}=-a_{21}(x_1)_{k+1}+(-a_{23}x_3-\cdots-a_{2n}x_n)_k+b_2\\ \quad\cdots & \\ \text{New } x_n: & a_{nn}(x_n)_{k+1}=(-a_{n1}x_1-a_{n2}x_2-\cdots-a_{n,n-1}x_{n-1})_{k+1}+b_n\end{cases}$$

Ex 2: (Gauss-Seidel) Here $S^{-1}T$ has smaller eigenvalues:

$$A=\begin{bmatrix}2&-1\\-1&2\end{bmatrix},\quad S=\begin{bmatrix}2&0\\-1&2\end{bmatrix},\quad T=\begin{bmatrix}0&1\\0&0\end{bmatrix},\quad S^{-1}T=\begin{bmatrix}0&\frac{1}{2}\\0&\frac{1}{4}\end{bmatrix}$$

$$\begin{cases}2v_{k+1}=w_k+b_1\\2w_{k+1}=v_{k+1}+b_2\end{cases}\quad\text{or}\quad\begin{bmatrix}2&0\\-1&2\end{bmatrix}\begin{bmatrix}v\\w\end{bmatrix}_{k+1}=\begin{bmatrix}0&1\\0&0\end{bmatrix}\begin{bmatrix}v\\w\end{bmatrix}_k+\begin{bmatrix}b_1\\b_2\end{bmatrix}$$

The eigenvalues of $S^{-1}T$ are $\frac{1}{4}$ and 0. The error is divided by 4 every time, so a single Gauss-Seidel step is worth two Jacobi steps. Both methods require the same number of operations. The Gauss-Seidel is better and saves storage.

There are examples in which Jacobi converges and Gauss-Seidel fails (or conversely). The symmetric case is straightforward: When all $a_{ii}>0$, Gauss-Seidel converges $\Leftrightarrow$ A: positive definite.

Convergence is faster if we go beyond the Gauss-Seidel correction $x_{k+1} - x_k$. Roughly speaking, those approximations stay on the same side of the solution x. An overrelaxation factor ω moves us closer to the solution. When $\omega = 1$, we recover Gauss-Seidel. With $\omega > 1$, the method is known as SOR, Successive Overrelaxation. The optimal choice of ω never exceeds 2. It is often ≈ 1.9.

Let D, L, U be the parts of A on, below, and above the diagonal. (They are different from the $A = LDU$.) In fact, we now have $A = L + D + U$. The Jacobi method has $S = D$ and $T = -L - U$, and the Gauss-Seidel chose $S = D + L$ and $T = -U$. To accelerate the convergence,

$$\text{Overrelaxation: } \underbrace{[D + \omega L]}_{\text{lower triangular}} x_{k+1} = \underbrace{[(1-\omega)D - \omega U]}_{\text{upper triangular}} x_k + \omega b$$

x_{k+1} can still replace x_k, component by component, as soon as it is computed. A typical step is

$$a_{ii}(x_i)_{k+1} = a_{ii}(x_i)_k + \omega\left[(-a_{i1}x_1 - \cdots - a_{i,i-1}x_{i-1})_{k+1} + (-a_{ii}x_i - \cdots - a_{in}x_n)_k + b_i\right]$$

Ex 3: (SOR) For the same matrix A, each SOR step is

$$\begin{bmatrix} 2 & 0 \\ -\omega & 2 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}_{k+1} = \begin{bmatrix} 2(1-\omega) & \omega \\ 0 & 2(1-\omega) \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}_k + \omega \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

The crucial matrix $L = S^{-1}T$ is

$$L = \begin{bmatrix} 2 & 0 \\ -\omega & 2 \end{bmatrix}^{-1} \begin{bmatrix} 2(1-\omega) & \omega \\ 0 & 2(1-\omega) \end{bmatrix} = \begin{bmatrix} 1-\omega & \frac{1}{2}\omega \\ \frac{1}{2}\omega(1-\omega) & 1 - \omega + \frac{1}{4}\omega^2 \end{bmatrix}$$

The optimal ω makes the largest eigenvalue of L (its spectral radius) as small as possible. The product of the eigenvalues of L equals $\det L = \det T / \det S$:

$$\lambda_1 \lambda_2 = \det L = (1-\omega)^2$$

always $\det S = \det D$ and $\det T = \det(1+\omega)D$. Their product is $\det L = (1+\omega)^n$. This explains why $\omega < 2$. At the optimal ω, the two eigenvalues are equal to $\omega - 1$. Their sum agrees to the sum of diagonal entries:

Optimal ω: $\lambda_1 + \lambda_2 = 2(\omega_{opt} - 1) = 2 - 2\omega_{opt} + \frac{1}{4}\omega_{opt}^2$

$\therefore \omega_{opt} = 4(2-\sqrt{3}) \approx 1.07$ and $\omega_{opt} - 1 \approx 0.07$

This is a major reduction from the Gauss-Seidel: $\lambda = \frac{1}{4}$ at $\omega = 1$. In this example, the choice of $\omega_{opt} = 1.07$ has again doubled the rate of convergence, because $(\frac{1}{4})^2 \approx 0.01$.

Young's 1950 thesis solved a simple formula for the optimal ω. The key step is to connect the eigenvalue λ of L to the eigenvalue μ of the original Jacobi matrix $D^{-1}(-L-U)$:

Formula for ω: $$(\lambda + \omega - 1)^2 = \lambda\omega^2\mu^2 \qquad (7)$$

When $\omega = 1$ (Gauss-Seidel), it yields $\lambda^2 = \lambda\mu^2 \Rightarrow \lambda = 0$ and $\lambda = \mu^2$. (In Ex 2, $\mu = \pm\frac{1}{2}$ and $\lambda = 0, \lambda = \frac{1}{4}$.) The important problem is to choose ω so that λ_{max} will be minimized. Young's Eq (7) is exactly our 2x2 example! The best ω makes the two roots λ both equal to $\omega - 1$

$$(\omega-1) + (\omega-1) = 2 - 2\omega + \mu^2\omega^2 \quad \text{or} \quad \omega = \frac{2(1-\sqrt{1-\mu^2})}{\mu^2}$$

For a large matrix, this pattern will be repeated for a number of different pairs $\pm\mu_i$. The largest μ gives the largest value of ω and of $\lambda = \omega - 1$. Since our goal is to make λ_{max} as small as possible, that extremal pair specifies

Optimal ω: $$\omega_{opt} = \frac{2(1-\sqrt{1-\mu_{max}^2})}{\mu_{max}^2} \quad \text{and} \quad \lambda_{max} = \omega_{opt} - 1.$$