A Fast Recursive Algorithm for Constructing Matrices with Prescribed Eigenvalues and Singular Values

by

Moody T. Chu

North Carolina State University

Outline

- Background
  - ◇ Schur-Horn Theorem
  - ◇ Mirsky Theorem
  - ◇ Sing-Thompson Theorem
  - ◇ Weyl-Horn Theorem
- A Recursive Algorithm
  - ◇ The Building Block — 2×2 Case
  - ◇ The Original Proof by Induction
  - ◇ An Innocent Mistake
  - ◇ A Recursive Clause in Programming
- The Matrix Structure
  - ◇ A Modified Proof
  - ◇ A Symbolic Example
- Numerical Experiment

Schur-Horn Theorem

- Given arbitrary Hermitian matrix H,
  - ◇ Let $\lambda = [\lambda_i] =$ eigenvalues.
  - ◇ Let $a = [a_i] =$ diagonal entries.
  - ◇ Assume

$$\begin{aligned} a_{j_1} &\le \ldots \le a_{j_n}, \\ \lambda_{m_1} &\le \ldots \le \lambda_{m_n}. \end{aligned}$$

  - ◇ Then

$$\begin{aligned} \sum_{i=1}^{k} \lambda_{m_i} &\le \sum_{i=1}^{k} a_{j_i}, \quad \text{for } k = 1, \ldots n, \\ \sum_{i=1}^{n} \lambda_{m_i} &= \sum_{i=1}^{n} a_{j_i}. \end{aligned}$$

    - ▷ Known as *a majorizing* λ.
- Given vectors $a, \lambda \in R^n$,
  - ◇ Assume a majorizes λ.
  - ◇ Then a Hermitian matrix H with eigenvalues λ and diagonal entries a exists.
- How to solve the *inverse eigenvalue problem* numerically?

Mirsky Theorem

- Any similar connection between eigenvalues and diagonal entries of a general matrix?
- A matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ and main diagonal elements $a_1, \ldots, a_n$ exists if and only if

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} \lambda_i.$$

Sing-Thompson Theorem

- Any connection between singular values and diagonal entries of a general matrix?
- Given vectors $d, s \in R^n$,
  - ◇ Assume

$$\begin{array}{rl} s_1 & \geq s_2 \;\; \geq \ldots s_n, \\ |d_1| & \geq |d_2| \;\; \geq \ldots |d_n|. \end{array}$$

  - ◇ Then a real matrix with singular values s and main diagonal entries d (possibly in different order) exists if and only if

$$\begin{array}{rcl} \sum_{i=1}^{k} |d_i| & \leq & \sum_{i=1}^{k} s_i, \quad \text{for } k = 1, \ldots, n, \\ \left(\sum_{i=1}^{n-1} |d_i|\right) - |d_n| & \leq & \left(\sum_{i=1}^{n-1} s_i\right) - s_n. \end{array}$$

- How to solve the *inverse singular value problem* numerically?

Weyl-Horn Theorem

- Any connection between singular values and eigenvalues of a general matrix?
  - ◇ singular value = |eigenvalue| for Hermitian matrices.
- Given vectors $\lambda \in C^n$ and $\alpha \in R^n$,
  - ◇ Assume

$$
\begin{aligned}
|\lambda_1| \ &\geq \ldots \ \geq |\lambda_n|, \\
\alpha_1 \ &\geq \ldots \ \geq \alpha_n.
\end{aligned}
$$

  - ◇ Then a matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ and singular values $\alpha_1, \ldots, \alpha_n$ exists if and only if

$$
\begin{aligned}
\prod_{j=1}^{k} |\lambda_j| \ &\leq \ \prod_{j=1}^{k} \alpha_j, \quad k = 1, \ldots, n-1, \\
\prod_{j=1}^{n} |\lambda_j| \ &= \ \prod_{j=1}^{n} \alpha_j.
\end{aligned}
$$

    - ▹ If $|\lambda_n| > 0$, then $\log \alpha$ majorizes $\log |\lambda|$.
- How to solve the *inverse eigenvalue (singular value) problem* numerically?

The 2×2 Case

- The Weyl-Horn Condition:

$$\begin{cases} |\lambda_1| & \leq & \alpha_1, \\ |\lambda_1||\lambda_2| & = & \alpha_1\alpha_2. \end{cases}$$

$$\Downarrow$$

$$\begin{cases} \alpha_2 \leq |\lambda_2| \leq |\lambda_1| \leq \alpha_1 \\ |\lambda_1|^2 + |\lambda_2|^2 \leq \alpha_1^2 + \alpha_2^2. \end{cases}$$

- The building block — A triangular matrix

$$A = \begin{bmatrix} \lambda_1 & \mu \\ 0 & \lambda_2 \end{bmatrix}$$

has singular value $\{\alpha_1, \alpha_2\}$ if and only if

$$\mu = \sqrt{\alpha_1^2 + \alpha_2^2 - |\lambda_1|^2 - |\lambda_2|^2}.$$

  - ◇ A is complex-valued when eigenvalues are complex.
  - ◇ A stable way of computing μ:

$$\mu = \begin{cases} 0, & \text{if } |(\alpha_1 - \alpha_2)^2 - (|\lambda_1| - |\lambda_2|)^2| \leq \epsilon \\ \sqrt{|(\alpha_1 - \alpha_2)^2 - (|\lambda_1| - |\lambda_2|)^2|}, & \text{otherwise.} \end{cases}$$

Ideas in Horn's Proof

- Reduce the original inverse problem to two problems of *smaller sizes.*
- Problems of smaller sizes are guaranteed to be solvable by the *induction hypothesis.*
- The subproblems are *affixed* together by working on a suitable 2×2 *corner*.
- The 2×2 problem has an explicit solution.

Key to the Algorithmic Success

- The eigenvalues and singular values of each of the two subproblems can be derived *explicitly.*
- Each of the two subproblems can further be down-sized.
- The original problem is *divided* into subproblems of size 2×2 or 1×1.
- The smaller problems can be *conquered* to build up the original size.
- In an environment that allows a subprogram to invoke itself recursively, only one-step of the divide-and-conquer procedure will be enough.
- Very similar to the radix-2 FFT $\Longrightarrow$ fast algorithm.

Outline of Proof

- Suppose $\alpha_i > 0$ for all $i = 1, \ldots, n$. So $\lambda_i \neq 0$ for all i.
  - ◇ The case of zero singular values can be handled in a similar way.
- Define

$$\begin{cases} \sigma_1 & := & \alpha_1, \\ \sigma_i & := & \sigma_{i-1}\frac{\alpha_i}{|\lambda_i|}, \quad \text{for } i = 2, \ldots, n-1. \end{cases}$$

  - ◇ Assume $\sigma := \min_{1 \leq i \leq n-1} \sigma_i$ occurs at the index j.
- Define

$$\rho := \frac{|\lambda_1 \lambda_n|}{\sigma}.$$

- The following three sets of inequalities are true. The numbers satisfy the Weyl-Horn conditions.

$$\begin{cases} |\lambda_1| & \geq & |\lambda_n|, \\ \sigma & \geq & \rho. \end{cases}$$

$$\begin{cases} \sigma & \geq & |\lambda_2| & \geq & \ldots & \geq & |\lambda_j|, \\ \alpha_1 & \geq & \alpha_2 & \geq & \ldots & \geq & \alpha_j. \end{cases}$$

$$\begin{cases} |\lambda_{j+1}| & \geq & \ldots & \geq & |\lambda_{n-1}| & \geq & \rho, \\ \alpha_{j+1} & \geq & \ldots & \geq & \alpha_{n-1} & \geq & \alpha_n. \end{cases}$$

- By induction hypothesis,
  - ◇ There exist unitary matrices $U_1, V_1 \in C^{j\times j}$ and *triangular* matrices A_1 such that

$$U_1 \begin{bmatrix} \alpha_1 & 0 & \ldots & 0 \\ 0 & \alpha_2 & & 0 \\ \vdots & \ddots & & \\ 0 & 0 & \ldots & \alpha_j \end{bmatrix} V_1^* = A_1 = \begin{bmatrix} \sigma & \times & \times & \ldots & \times \\ 0 & \lambda_2 & & & \times \\ & & & & \\ \vdots & & & \ddots & \\ 0 & 0 & & & \lambda_j \end{bmatrix}.$$

  - ◇ There exist unitary matrices $U_2, V_2 \in C^{(n-j)\times(n-j)}$, and *triangular* matrix A_2 such that

$$U_2 \begin{bmatrix} \alpha_{j+1} & 0 & \ldots 0 \\ 0 & \alpha_{j+2} & 0 \\ \vdots & & \ddots \\ 0 & 0 & \ldots \alpha_n \end{bmatrix} V_2^* = A_2 = \begin{bmatrix} \lambda_{j+1} & \times & \ldots & \times & \times \\ 0 & \lambda_{j+2} & & & \times \\ \vdots & & \ddots & & \vdots \\ & & & \lambda_{n-1} & \times \\ 0 & 0 & \ldots & 0 & \rho \end{bmatrix}.$$

- Horn's claim: The block matrix

$$\begin{bmatrix} A_1 & \bigcirc \\ \bigcirc & A_2 \end{bmatrix}$$

can be *permuted* to the triangular matrix

$$\begin{bmatrix} \lambda_2 & \times & \dots & \times & \times & & & & & \\ 0 & & & & \times & & & & & \\ \vdots & & \ddots & & \vdots & & & \bigcirc & & \\ & & & \lambda_j & \times & & & & & \\ 0 & & \dots & 0 & \boxed{\begin{matrix} \sigma & 0 \\ 0 & \rho \end{matrix}} & & & & & \\ 0 & 0 & \dots & 0 & & \times & \times & \dots & \times \\ & & & & & \lambda_{j+1} & & & \times \\ & & \bigcirc & & & & & & \\ & & & & \vdots & & & \ddots & \\ & & \dots & & 0 & 0 & & & \lambda_{n-1} \end{bmatrix}$$

- The 2×2 corner can now be glued together by

$$U_0 \begin{bmatrix} \sigma & 0 \\ 0 & \rho \end{bmatrix} V_0^* = A_0 = \begin{bmatrix} \lambda_1 & \mu \\ 0 & \lambda_n \end{bmatrix}.$$

- How to do the permutation, or is it a mistake?
  - ◇ It takes more than permutation to rearrange the diagonals of a triangular matrix.

A MATLAB Program

```
function [A]=svd_eig(alpha,lambda);
n = length(alpha);
if n == 1                                            % The 1 by 1 case
   A = [lambda(1)];
elseif n == 2                                        % The 2 by 2 case
   [U,V,A] = two_by_two(alpha,lambda);
else                                                 % Check zero singular values
   tol = n*alpha(1)*eps;
   k = sum(alpha > tol); m = sum(abs(lambda) > tol);
   if k == n                                         % Nonzero singular values
      j = 1; s = alpha(1); temp = s;
      for i = 2:n-1
         temp = temp*alpha(i)/abs(lambda(i));
         if temp < s, j = i; s = temp; end
      end
      rho = abs(lambda(1)*lambda(n))/s;
      [U0,V0,A0] = two_by_two([s;rho],[lambda(1);lambda(n)]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      [A1] = svd_eig(alpha(1:j),[s;lambda(2:j)]);          % RECURSIVE %
      [A2] = svd_eig(alpha(j+1:n),[lambda(j+1:n-1);rho]); %  CALLING  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      A = [A1,zeros(j,n-j);zeros(n-j,j),A2];
      Temp = A;
         A(1,:)=U0(1,1)*Temp(1,:)+U0(1,2)*Temp(n,:);
         A(n,:)=U0(2,1)*Temp(1,:)+U0(2,2)*Temp(n,:);
      Temp = A;
         A(:,1)=V0(1,1)*Temp(:,1)+V0(1,2)*Temp(:,n);
         A(:,n)=V0(2,1)*Temp(:,1)+V0(2,2)*Temp(:,n);
   else                                              % Zero singular values
      beta = prod(abs(lambda(1:m)))/prod(alpha(1:m-1));
      [U3,V3,A3] = svd_eig([alpha(1:m-1);beta],lambda(1:m));
      A = zeros(n); A(1:m,1:m) = V3'*A3*V3;
      for i = m+1:k, A(i,i+1) = alpha(i); end
      A(m,m+1) = sqrt(abs(alpha(m)^2-beta^2));
   end
end
```

Correct that "Mistake"

- Horn's requirement:
  - ◇ Both intermediate matrices A_1 and A_2 are upper triangular matrices.
  - ◇ Diagonal entries are arranged in a certain order.
    - ▷ Valid from the Schur decomposition theorem.
    - ▷ More than permutation, not easy for computation.
    - ▷ To rearrange diagonal entries via unitary similarity transformations while maintaining the upper triangular structure is expensive.
- Our contribution:
  - ◇ The triangular structure is entirely unnecessary.
  - ◇ The matrix A produced from our algorithm is generally not triangular.
  - ◇ Do not need to rearrange the diagonal entries
  - ◇ Modifying the first and the last rows and columns of the block diagonal matrix $\begin{bmatrix} A_1 & \bigcirc \\ \bigcirc & A_2 \end{bmatrix}$, *as if nothing happened*, is enough.

- Algorithm:
  - ◇ Denote $U_0 = [u_{0,st}]$ and $V_0 = [v_{0,st}]$.
  - ◇ Then

$$\begin{bmatrix} u_{0,11} & 0 & u_{0,12} \\ 0 & I_{n-1} & 0 \\ u_{0,21} & 0 & u_{0,22} \end{bmatrix} \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & & 0 \\ \vdots & & \ddots & \\ 0 & & & \alpha_n \end{bmatrix} \begin{bmatrix} V_1^* & 0 \\ 0 & V_2^* \end{bmatrix} \begin{bmatrix} v_{0,11} & 0 & v_{0,12} \\ 0 & I_{n-1} & 0 \\ v_{0,21} & 0 & v_{0,22} \end{bmatrix}^*$$

    is the desired matrix.

- A has the structure

$$A = \begin{bmatrix} \lambda_1 & \otimes & \dots & \otimes & \otimes & * & * & & & \mu \\ \otimes & \lambda_2 & & & \times & 0 & 0 & & & * \\ \vdots & & \ddots & & \vdots & & & \bigcirc & & \\ & & & \lambda_{j-1} & \times & & & & & \\ \otimes & \times & \dots & \times & \lambda_j & & & & & * \\ * & 0 & \dots & 0 & 0 & \lambda_{j+1} & \times & \times & \dots & \otimes \\ * & & & & 0 & \times & \lambda_{j+2} & & & \otimes \\ & & \bigcirc & & & & & & & \\ & & & & & \vdots & & & \ddots & \\ 0 & * & \dots & * & * & \otimes & \otimes & & & \lambda_n \end{bmatrix}.$$

  - ◇ $\times$ = unchanged, original entries from A_1 or A_2.
  - ◇ $\otimes$ = entries of A_1 or A_2 that are modified by scalar multiplications.
  - ◇ $*$ = possible new entries that were originally zero.

A Variation of Horn's Proof

- Does the algorithm really works?
  - ◇ Clearly, A has singular values $\{\alpha_1, \ldots, \alpha_n\}$.
  - ◇ Need to show that A has eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$.
- What has been changed?

(P1) Diagonal entries of A_1 and A_2 are in fixed orders, $\sigma, \lambda_2, \ldots, \lambda_j$ and $\lambda_{j+1}, \ldots, \lambda_{n-1}, \rho$, respectively.

(P2) Each A_i is similar through permutations, which need not to be known, to a lower triangular matrix whose diagonal entries constitute the same set as the diagonal entries of A_i. (Thus, each A_i has precisely its own diagonal entries as its eigenvalues.)

(P3) The first row and the last row have the same zero pattern except that the lower-left corner is always zero.

(P4) The first column and the last column have the same zero pattern except that the lower-left corner is always zero.

- Use graph theory to show that the affixed matrix A has exactly the same properties.

A Symbolic Example

- Dividing process:

$$\boxed{\begin{cases} \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 & \lambda_6 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 \end{cases}}$$

$$j_1 = 5 \quad \Downarrow \quad \begin{cases} \lambda_1 & \lambda_6 \\ \sigma_1 & \rho_1 \end{cases}$$

$$\boxed{\begin{cases} \sigma_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 \end{cases}} \; \boxed{\begin{cases} \rho_1 \\ \alpha_6 \end{cases}}$$

$$j_2 = 2 \quad \Downarrow \quad \begin{cases} \sigma_1 & \lambda_5 \\ \sigma_2 & \rho_2 \end{cases}$$

$$\boxed{\begin{cases} \sigma_2 & \lambda_2 \\ \alpha_1 & \alpha_2 \end{cases}} \; \boxed{\begin{cases} \lambda_3 & \lambda_4 & \rho_2 \\ \alpha_3 & \alpha_4 & \alpha_5 \end{cases}} \; \boxed{\phantom{\begin{cases} \rho_1 \\ \alpha_6 \end{cases}}}$$

$$j_3 = 1 \quad \Downarrow \quad \begin{cases} \lambda_3 & \rho_2 \\ \sigma_3 & \rho_3 \end{cases}$$

$$\boxed{\phantom{\begin{cases} \sigma_3 \\ \alpha_3 \end{cases}}} \; \boxed{\begin{cases} \sigma_3 \\ \alpha_3 \end{cases}} \; \boxed{\begin{cases} \lambda_4 & \rho_3 \\ \alpha_4 & \alpha_5 \end{cases}} \; \boxed{\phantom{\begin{cases} \sigma_3 \\ \alpha_3 \end{cases}}}$$

- Conquering process:

$$\begin{bmatrix} \lambda_1 & \otimes & 0 & 0 & \otimes & * \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 \\ \otimes & 0 & \lambda_3 & 0 & \times & * \\ \otimes & 0 & \times & \lambda_4 & \times & * \\ 0 & \times & 0 & 0 & \lambda_5 & 0 \\ 0 & * & 0 & 0 & * & \lambda_6 \end{bmatrix}$$

$$j_1 = 5 \quad \Uparrow \quad \begin{bmatrix} \sigma_1 & \times & 0 & 0 & \times & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 \\ \times & 0 & \lambda_3 & 0 & \times & 0 \\ \times & 0 & \times & \lambda_4 & \times & 0 \\ 0 & \times & 0 & 0 & \lambda_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho_1 \end{bmatrix}$$

$$\begin{bmatrix} \sigma_1 & \otimes & 0 & 0 & * \\ 0 & \lambda_2 & 0 & 0 & 0 \\ * & 0 & \lambda_3 & 0 & \otimes \\ * & 0 & \times & \lambda_4 & \otimes \\ 0 & * & 0 & 0 & \lambda_5 \end{bmatrix} \qquad \begin{bmatrix} \rho_1 \end{bmatrix}$$

$$j_2 = 2 \quad \Uparrow \quad \begin{bmatrix} \sigma_2 & \times & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & \times \\ 0 & 0 & \times & \lambda_4 & \times \\ 0 & 0 & 0 & 0 & \rho_2 \end{bmatrix}$$

$$\begin{bmatrix} \sigma_2 & * \\ 0 & \lambda_2 \end{bmatrix} \qquad \begin{bmatrix} \lambda_3 & 0 & * \\ * & \lambda_4 & \otimes \\ 0 & 0 & \rho_2 \end{bmatrix}$$

$$j_3 = 1 \quad \Uparrow \quad \begin{bmatrix} \sigma_3 & 0 & 0 \\ 0 & \lambda_4 & \times \\ 0 & 0 & \rho_3 \end{bmatrix}$$

$$\begin{bmatrix} \sigma_3 \end{bmatrix} \qquad \begin{bmatrix} \lambda_4 & * \\ 0 & \rho_3 \end{bmatrix}$$

Computational Cost

- The divide-and-conquer feature brings on fast computation.
- The overall cost is estimated at the order of $O(n^2)$.
- A numerical experiment:

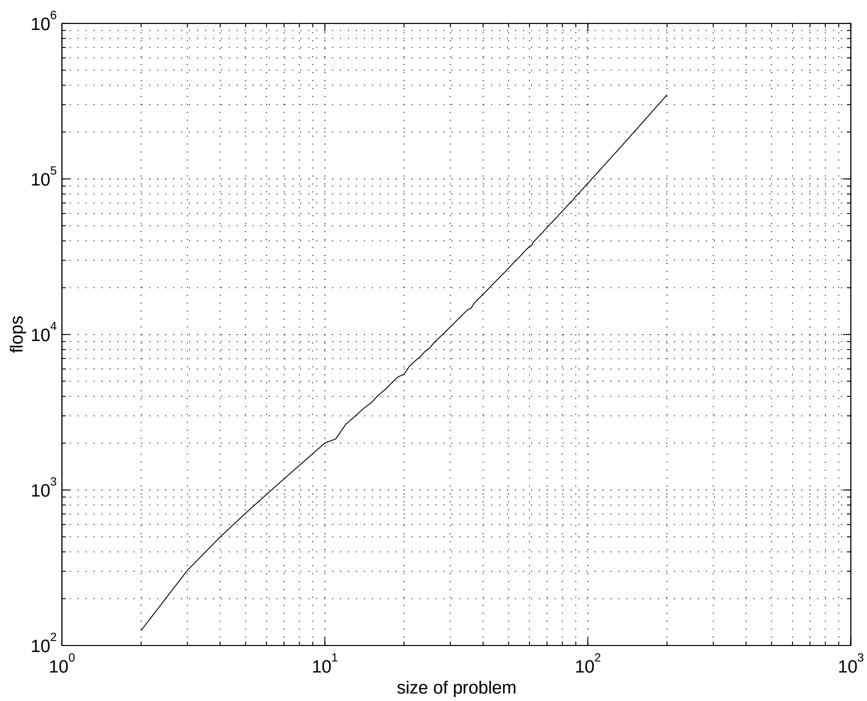

Figure 1: log-log plot of computational flops versus problem sizes

Rosser Test

- Rosser matrix R:

$$R = \begin{bmatrix} 611 & 196 & -192 & 407 & -8 & -52 & -49 & 29 \\ 196 & 899 & 113 & -192 & -71 & -43 & -8 & -44 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 52 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -71 & 61 & 8 & 411 & -599 & 208 & 208 \\ -52 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ -49 & -8 & 8 & 59 & 208 & 208 & 99 & -911 \\ 29 & -44 & 52 & -23 & 208 & 208 & -911 & 99 \end{bmatrix}.$$

  ◇ Has one double eigenvalue, three nearly equal eigenvalues, one zero eigenvalue, two dominant eigenvalues of opposite sign and one small nonzero eigenvalue.

  ◇ The *computed* eigenvalues and singular values of R are

$$\lambda = \begin{bmatrix} -1.020049018429997e{+}03 \\ 1.020049018429997e{+}03 \\ 1.020000000000000e{+}03 \\ 1.019901951359278e{+}03 \\ 1.000000000000001e{+}03 \\ 9.999999999999998e{+}02 \\ 9.804864072152601e{-}02 \\ 4.851119506099622e{-}13 \end{bmatrix}, \alpha = \begin{bmatrix} 1.020049018429997e{+}03 \\ 1.020049018429996e{+}03 \\ 1.020000000000000e{+}03 \\ 1.019901951359279e{+}03 \\ 1.000000000000000e{+}03 \\ 9.999999999999998e{+}02 \\ 9.804864072162672e{-}02 \\ 1.054603342667098e{-}14 \end{bmatrix}.$$

- Using the above λ and α,
  - ◇ A nonsymmetric matrix is produced:

$$\begin{bmatrix} 1.0200e{+}03 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1.0200e{+}03 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0200e{+}03 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0199e{+}03 & 0 & 0 & 1.4668e{-}09 & 0 \\ 0 & 0 & 0 & 0 & 1.0000e{+}03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000e{+}03 & 0 & 0 \\ 0 & 0 & 0 & -1.5257e{-}05 & 0 & 0 & 9.8049e{-}02 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.4045e{-}07 & 0 \end{bmatrix}.$$

  - ◇ The re-computed eigenvalues and singular values of A are

$$\hat{\lambda} = \begin{bmatrix} -1.020049018429997e{+}03 \\ 1.020049018429997e{+}03 \\ 1.020000000000000e{+}03 \\ 1.019901951359278e{+}03 \\ 1.000000000000001e{+}03 \\ 9.999999999999998e{+}02 \\ 9.80486407215721e{-}02 \\ 0 \end{bmatrix}, \hat{\alpha} = \begin{bmatrix} 1.020049018429997e{+}03 \\ 1.020049018429997e{+}03 \\ 1.020000000000000e{+}03 \\ 1.019901951359279e{+}03 \\ 1.000000000000001e{+}03 \\ 9.999999999999998e{+}02 \\ 9.804864072162672e{-}02 \\ 0 \end{bmatrix}.$$

  - ◇ The re-computed eigenvalues and singular values agree with those of R up to the machine accuracy.

Wilkinson Test

- Wilkinson's matrices:
  - ◇ All are symmetric and tridiagonal.
  - ◇ Have nearly, but not exactly, equal eigenvalue pairs.
- Using these data:
  - ◇ Discrepancy in eigenvalues and singular values between our constructed matrices and Wilkinson's matrices.

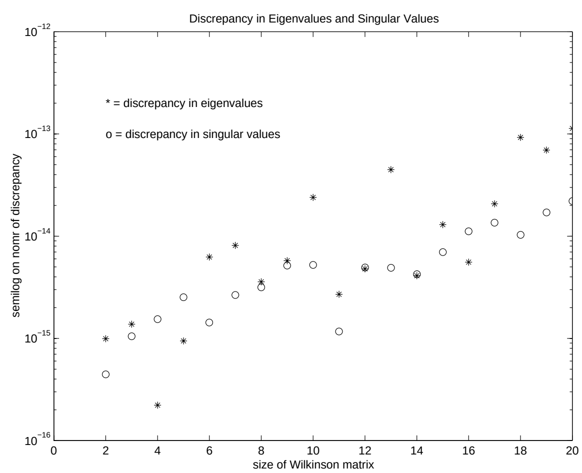

Figure 2: L_2 norm of discrepancy in eigenvalues and singular values.

◇ Matrices constructed are nearly but not symmetric.

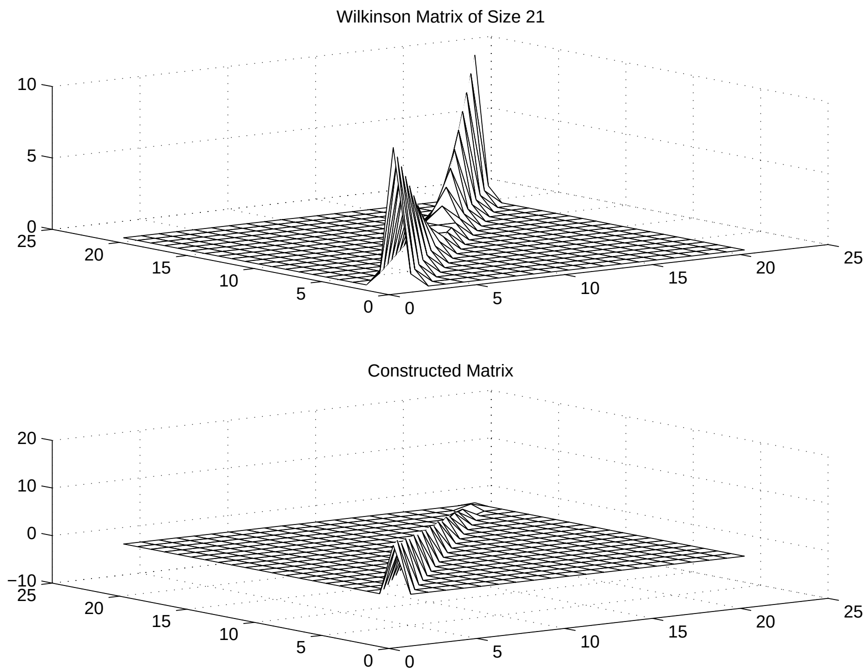

Figure 3: 3-D mesh representation of 21×21 matrices

Conclusion

- Weyl-Horn Theorem completely characterizes the relationship between eigenvalues and singular values of a general matrix.
- The original proof has been modified.
- With the aid of programming languages that allow a subprogram to invoke itself recursively, an induction proof can be implemented as a recursive algorithm.
- The resulting algorithm is fast. The cost of construction is approximately $O(n^2)$.
- The matrix being constructed usually is not symmetric and is complex-valued, if complex eigenvalues are present.
- Numerical experiment on some very challenging problems suggests that our method is quite robust.