

Solving Matrix Equations

Q: Solve $x+2y+3z = 5$, $2x-3y-z = 3$, and $4x+y+2z = 8$. A: Put in the **matrix** form $A\underline{x} = \underline{b}$ as shown in *yellow*. Perform **row** operations on A: $R_i := R_i - k_{ij}R_j$ (with $i > j$).

Do these operations to get the matrix with the **zeros** in the bottom left. Then perform R_2-2R_1 and R_3-4R_1 , and so on, to get the final matrix. We shall call this matrix **U**. We aim to factorise A in the form $A = LU$. In fact, L is the final matrix, where the lower triangular entries are the k_{ij} .

The **elementary** operation $R_i := R_i - k_{ij}R_j$ may be performed on A by left multiplying by the elementary matrix E_{ij} , where E_{ij} is an identity matrix with a $-k_{ij}$ in row i and column j. Think of the **product** shown in green, which is a $1 \times n$ matrix multiplied by an $n \times m$ matrix to give a $1 \times m$ matrix. We have performed $E_{32}E_{31}E_{21}A = U$. Hence $A = E_{21}^{-1}E_{31}^{-1}E_{32}^{-1}U$, where $E_{21}^{-1}E_{31}^{-1}E_{32}^{-1} = LU$, and $L = E_{21}^{-1}E_{31}^{-1}E_{32}^{-1}I$. Note that E_{ij}^{-1} is the *same* elementary matrix as before, but with a **sign** change in the k_{ij} position. This is because $R_i := R_i+k_{ij}R_j$ is the **inverse** of $R_i := R_i-k_{ij}R_j$.

So we calculate L as $I = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{matrix} \\ R_3+R_2 \\ \\ \end{matrix} \sim \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{matrix} \\ \\ R_3+4R_1 \\ \end{matrix} \sim \begin{bmatrix} 1 & & \\ & 2 & 1 \\ & & 1 \end{bmatrix} \begin{matrix} \\ R_2+2R_1 \\ \\ \end{matrix}$

and $A = LU$ imply that $LU\underline{x} = \underline{b}$ and $L\underline{c} = \underline{b}$, so that $\underline{Ux} = \underline{c}$. We now have 2 **triangular** systems of equations. First *solve* $L\underline{c} = \underline{b}$ (1st) and then $\underline{Ux} = \underline{c}$ (2nd). So the *solution* is $x = 4/3$, $y = -2/3$ and $z = 5/3$.

Pivoting

The **operations** used are $R_i := R_i - k_{ij}R_j$. If the k_{ij} are large, then round-off errors become **significant**. The aim is to get $-1 \leq k_{ij} \leq 1$. How: move the largest entry in the column on or below the diagonal to the diagonal position. This is called the “**pivot**” element. Note: “largest” means “has maximum modulus”. Looking at the **diagram**, we have $E_{32}P_{22}E_{31}E_{21}P_{13}A = U$. (Note: P_{22} is just I). And so $A = P_{13}^{-1}E_{21}^{-1}E_{31}^{-1}P_{22}^{-1}E_{32}^{-1}IU$. The **right** hand side is L. Note that $P_{ij} = P_{ij}^{-1}$ we — just have a identity matrix with 0’s along diagonal **where** our I and j are and 1’s on the offset — like the diagram in *blue*.

$$A \begin{bmatrix} 1 & 2 & 3 \\ 2 & -3 & -1 \\ 4 & 1 & 2 \end{bmatrix} \underline{b} \begin{bmatrix} 5 \\ 3 \\ 8 \end{bmatrix} \underline{x} \begin{bmatrix} 4/3 \\ -2/3 \\ 5/3 \end{bmatrix}$$

PIVOT

$$\text{swap } (R_1, R_3) \begin{bmatrix} 4 & 1 & 2 \\ 2 & -3 & -1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 4 & 1 & 2 \\ 0 & -7/2 & -2 \\ 0 & 7/4 & 5/2 \end{bmatrix} \begin{matrix} \\ R_2 - 1/2R_1 \\ R_3 - 1/4R_1 \end{matrix}$$

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 1 & 2 \\ 0 & -7/2 & -2 \\ 0 & 0 & 3/2 \end{bmatrix} \begin{matrix} \\ \\ -R_3 + 1/2R_2 \end{matrix} \underline{U}$$

What we get in the *red* on the next page is a row permutation of a *lower triangular matrix*. Next step: solve $A\underline{x} = \underline{b}$ using $L\underline{c} = \underline{b}$ and $\underline{Ux} = \underline{c}$. This is shown in *blue* on the next page. Amazingly, it is still possible to place the k_{ij} ’s in L as they are calculated. *Method*: Use an extra vector, Order.

$$I = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \xrightarrow{R_3 - \frac{1}{2}R_2} \begin{bmatrix} 1 & & \\ & 1 & \\ & & -\frac{1}{2} & 1 \end{bmatrix} \xrightarrow{\substack{R_2 + \frac{1}{2}R_1 \\ R_3 - \frac{1}{4}R_1}} \begin{bmatrix} 1 & & \\ & 1 & \\ & & \frac{1}{4} & -\frac{1}{2} & 1 \end{bmatrix} = L$$

$$Lc = b \begin{bmatrix} \frac{1}{4} & -\frac{1}{2} & 1 \\ \frac{1}{2} & 1 & \\ 1 & & \end{bmatrix} \begin{bmatrix} 8 \\ -1 \\ \frac{5}{2} \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \\ 8 \end{bmatrix}$$

$$Ux = c \begin{bmatrix} 4 & 1 & 2 \\ & -\frac{7}{2} & -2 \\ & & \frac{3}{2} \end{bmatrix} \begin{bmatrix} \frac{4}{3} \\ -\frac{2}{3} \\ \frac{5}{3} \end{bmatrix} = \begin{bmatrix} 8 \\ -1 \\ \frac{5}{2} \end{bmatrix}$$

Whenever we **interchange** a row, $R_i \leftrightarrow R_j$, we apply the **row change** to order as well. Therefore, k_{ij} is placed in L in column j, row Order[i]. The 1 is placed in column j, row Order[j]. In our example, the Order is $[1_{23}]$

initially, and then becomes $[^3_{21}]$ (by $R_1 \leftrightarrow R_3$). So in column 1, the 1 goes in row order[1] = 3. The $\frac{1}{2} = k_{21}$ goes in row **order** [2] = 2. And the $\frac{1}{4} = k_{31}$ goes in row order [3] = 1. In column 2, the 1 goes in row Order[2] = 2; the $-\frac{1}{2} = k_{32}$ goes in row Order [3] = 1; and in column 3, the 1 goes in row Order[3] = 1. Note: If we had $|A_{22}| < |A_{32}|$, do row swap P_{23} .

➤ 11th October 1999

Useful UNIX Commands

>**pwd** print working directory. >**ls** short list. >**ls -l** long list. >**cat <file>** puts the file on the screen. >**rm <file>** remove/delete. >**mv <file> <where to>** moves a file. >**lpr -Pmalaser <file>**. >**cd <dir/dir/...>** change directory. >**mkdir <dir. name>** make directory. >**rm -r <dir>** deletes directory. >**ps ux** process list. >**kill -9 <process>** deletes/terminates process.

Smith Normal Form / Invariant Factor Form

Consider an **integer** matrix with diagonals d_1, d_2, \dots, d_r , where $d_1 | d_2 | d_3 | \dots | d_r$. Now $d_1 = \gcd\{a_{ij}\}$ (it is the gcd of **all** the entries of the *matrix*). $d_2 = \gcd\{2 \times 2 \text{ sub-determinants}\}$, etc. Example: Consider $A = \begin{pmatrix} 30 & 165 \\ 66 & 110 \end{pmatrix}$. Find the g.c.d. of the matrix: start with **two** numbers, i.e. 2nd column, and carry on with the *row* operations: $\sim \begin{pmatrix} -36 & 55 \\ 66 & 110 \end{pmatrix} \sim \begin{pmatrix} -36 & 85 \\ 13 & 50 \end{pmatrix}$. Now we know that $\gcd(165, 110) = 55$. Now find $\gcd(55, -36)$ by doing **column** operations: $[c_2 + c_1] \sim \begin{pmatrix} -36 & 19 \\ 138 & 138 \end{pmatrix}$; $[c_1 + 2c_2] \sim \begin{pmatrix} 2 & 19 \\ 414 & 138 \end{pmatrix}$; $[c_2 - 10c_1] \sim \begin{pmatrix} 2 & -1 \\ 414 & -4002 \end{pmatrix}$; $[c_1 + c_2] \sim \begin{pmatrix} 1 & -1 \\ -3588 & -4002 \end{pmatrix}$; $[c_2 + c_1] \sim \begin{pmatrix} 1 & 0 \\ -3588 & -7570 \end{pmatrix}$. Now **do** $R_2 + 3588R_1$ and $(-c_2)$ to get $\begin{pmatrix} 1 & 0 \\ 0 & 7590 \end{pmatrix} = D$.

Definition of **elementary** matrices: An E_{ij} matrix is a matrix with *ones* on the main diagonal and a $-k$ in the i row; j^{th} column. A P_{ij} matrix is a matrix with *ones* on the main diagonal, except as **shown** in the diagram. E_{ij} performs $R_i := R_i - kR_j$. $P_{ij}A$ performs $R_i \leftrightarrow R_j$. AE_{ij} performs $C_j := C_j - kC_i$. And AP_{ij} performs $C_i \leftrightarrow C_j$. Here, we have $B_u \dots B_3 B_2 B_1 A C_1 C_2 \dots C_v = D$, where the B_i and C_j are either E_{ij} , P_{ij} or N_i type matrices. **Hence** $A = (B_1^{-1} B_2^{-1} \dots B_u^{-1}) D (C_v^{-1} \dots c_2^{-1} c_1^{-1})$; $A = LDR$, where L & R are invertible and all integer.

$$E_{ij} = \begin{bmatrix} 1 & & & \\ & \dots & & \\ & & 1 & \\ & & & \dots & \\ & & -k & & 1 & \\ & & & & & \dots & \\ & & & & & & 1 & \\ & & & & & & & \dots & \\ & & & & & & & & & 1 \end{bmatrix} \quad P_{ij} = \begin{bmatrix} 1 & & & \\ & \dots & & \\ & & 1 & \\ & & & \dots & \\ & & & & 0 & \\ & & & & & \dots & \\ & & & & & & 1 & \\ & & & & & & & \dots & \\ & & & & & & & & & 1 \end{bmatrix} \quad N_i = \begin{bmatrix} 1 & & & \\ & \dots & & \\ & & 1 & \\ & & & \dots & \\ & & & & -1 & \\ & & & & & \dots & \\ & & & & & & 1 & \\ & & & & & & & \dots & \\ & & & & & & & & & 1 \end{bmatrix}$$

➤ 18th October 1999

Hessenberg Form

We require the matrix as **shown** in the diagram, where $*$ = "no restriction", and where H is similar to A, i.e. there exists a T such that $H = T^{-1}AT$. Method: $\dots B_2^{-1} B_1^{-1} A B_1 B_2 \dots$ ($B_k = E_{ij}, P_{ij}, N_i$). We can use *partial pivoting* as necessary.

$$A \rightarrow H = \begin{bmatrix} * & * & \dots & * \\ * & * & \dots & * \\ 0 & * & \dots & * \\ 0 & 0 & * & * \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & * & * \end{bmatrix}$$

Example: we can represent H as $H = E_{43}E_{42}P_{23}AP_{23}E_{42}^{-1}E_{43}^{-1}$.

This implies that $A = (P_{23}E_{42}^{-1}E_{43}^{-1}I) \times H(IE_{43}E_{42}P_{23}) = T^{-1}HT$.

$$\begin{aligned}
 A &= \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 4 & 5 \\ 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & 1 \end{bmatrix} \sim \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 4 & 5 \\ 1 & 3 & 2 & 1 \end{bmatrix} \xrightarrow{R_2 \leftrightarrow R_3} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 4 & 5 \\ 0 & 1 & 1 & 0 \end{bmatrix} \xrightarrow{R_4 - R_2} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 4 & 5 \\ 0 & 0 & -1 & -1 \end{bmatrix} \\
 &\sim \begin{bmatrix} 0 & 2 & 1 & 3 \\ 1 & 1 & 1 & 1 \\ 0 & 4 & 0 & 5 \\ 0 & 1 & 1 & 0 \end{bmatrix} \xrightarrow{C_2 \leftrightarrow C_3} \begin{bmatrix} 0 & 5 & 1 & 3 \\ 1 & 2 & 1 & 1 \\ 0 & 9 & 0 & 5 \\ 0 & 1 & 0 & 0 \end{bmatrix} \xrightarrow{C_2 + C_4} \begin{bmatrix} 0 & 5 & 1 & 3 \\ 1 & 2 & 1 & 1 \\ 0 & 9 & 0 & 5 \\ 0 & 0 & 0 & -5/9 \end{bmatrix} \xrightarrow{R_4 - (1/9)R_3} \begin{bmatrix} 0 & 5 & 1 & 3 \\ 1 & 2 & 1 & 1 \\ 0 & 9 & 0 & 5 \\ 0 & 0 & 0 & -5/9 \end{bmatrix} \xrightarrow{C_3 + (1/9)C_4} \begin{bmatrix} 0 & 5 & 4/3 & 3 \\ 1 & 2 & 10/9 & 1 \\ 0 & 9 & 5/9 & 5 \\ 0 & 0 & 5/81 & -5/9 \end{bmatrix}
 \end{aligned}$$

Now do inverse column operations.

Algorithm. Procedure Hessenberg (A). $n = \text{rowdim}(A) (= \text{coldim}(A))$. For each *column* k , ($1 \leq k \leq (n-2)$), Search *column* k and rows $k+1, \dots, n$ to find the **largest** entry — in row I , say. Swap rows $k+1$ and I . **Swap** columns $k+1$ and I . For $k+2 \leq j \leq n$, $R_j := R_j - (A_{k+1,k}/A_{j,k})R_{k+1}$. **For** $k+2 \leq j \leq n$, $C_{k+1} := C_{k+1} + (\text{previous multiple})C_j$.

➤ 25th October 1999

Eigenvalues/Eigenvectors

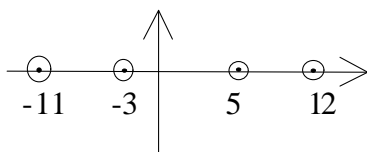
Let A be a **square** matrix, where $A\underline{x} = \lambda\underline{x}$. Solve $|A - \lambda I| = 0$, the *characteristic* equation. If A is symmetric, this implies that all $\lambda \in \mathbf{R}$. Assume that A is an $n \times n$ matrix, with **eigenvalues** $\{\lambda_1, \dots, \lambda_n\}$. If the λ_i are all distinct, this *implies that* $P^{-1}AP = D$ (diagonal). The **columns** of P are the x_i .

Gerschgorin's Theorems

(1) If $A = (a_{ij})$ is an $n \times n$ matrix, define C_i to be the disk in \mathbf{C} with centre $a_{i,i}$ (the i^{th} diagonal entry) and *radius* $\sum_{j \neq i} |a_{ij}|$. (The sum of the moduli of the *non-diagonal entries in row* i). Then every eigenvalue lies in *one of these discs*. For the example matrix A shown, we have the table shown.

$$A = \begin{bmatrix} -11 & 0.1 & 0.1 & 0.2 \\ 0.1 & -3 & 0.1 & 0.1 \\ 0.1 & 0.1 & 5 & 0.1 \\ 0.2 & 0.1 & 0.1 & 12 \end{bmatrix}$$

i	centre	radius
1	-11	$0.1+0.1+0.2 = 0.4$
2	-3	0.3
3	5	0.3
4	12	0.4

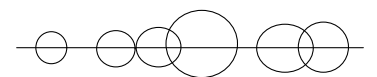


Theorem 2: If k of the C_i form a connected region R disjoint from the *remaining* C_i , then exactly k eigenvalues lie in R . **In Maple**, `charpoly(A, lambda)` returns the characteristic polynomial of A . `eigenvals(A)`; `eigenvects(A)` returns $[[\lambda_1, \{\underline{x}_1\}], \dots, [\lambda_r, \{\underline{x}_r\}]]$.

➤ 1st November 1999

Overlapping Gerschgorin Circles

Suppose we had a diagram as shown. Then the answer to the question “What is the set of connected regions” would be $[[1], [2,3,4,5,6]]$.

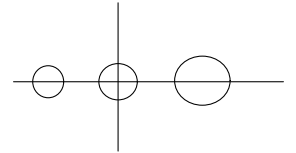


What we are trying to prove is that if d_{ij} is the distance between the *centres of the two circles*, and r_i and r_j are the radii of the two circles, then the two circles **overlap** if $d_{ij} \leq r_i + r_j$. Overlap is not an *equivalence relation*, but “in the same connected component” is. We could store the result of which overlap is with which in a **matrix** as shown. So, find out **which** overlap with which, and then use **transitivity** to find the connected regions.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Power Methods

(1) The **Power Method** finds the greatest *eigenvalue*, if one exists
 Example: Let $A = \begin{bmatrix} 8 & 1 & -5 & 1 \\ 1 & 0 & 2 & 1 \end{bmatrix}$. Then $\underline{u}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$; $\underline{v}_1 = A\underline{u}_0 = \begin{bmatrix} 10 \\ -3 \end{bmatrix}$; $m_1 = 10$.
 Then $\underline{u}_1 = \underline{v}_1/m_1 = \begin{bmatrix} 1 \\ -0.3 \end{bmatrix}$. $\underline{v}_2 = A\underline{u}_1 = \begin{bmatrix} 7.9 \\ 2.7 \end{bmatrix}$, $m_2 = 7.9$ so $\underline{u}_2 = \underline{v}_2/m_2 = \begin{bmatrix} 1 \\ 0.27 \end{bmatrix}$. Now $\underline{v}_3 = A\underline{u}_2 = \dots$



Algorithm: Set $\underline{u} = [1 \ 1 \ \dots \ 1]^t$. Repeat: $\underline{v} = A\underline{u}$; $m =$ greatest element in \underline{v} ; $\underline{u} = \underline{v}/m$ — until successive m 's and \underline{u} 's differ by $\leq 10^{-6}$, say. The final m is the *required eigenvalue*; and the final \underline{u} is the *required eigenvector*. **Suppose** that A has eigenvalues $\lambda_1, \dots, \lambda_n$, with $(|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|)$. Assume that $\{\underline{x}_1, \dots, \underline{x}_n\}$ is a *basis* for \mathbf{C}^n , where $A\underline{x}_i = \lambda_i \underline{x}_i$, and that $[1, 1, \dots, 1]^t = c_1 \underline{x}_1 + c_2 \underline{x}_2 + \dots + c_n \underline{x}_n$, with $c_1 \neq 0$.

Then $A\underline{u}_0 = \underline{v}_1 = A(c_1 \underline{x}_1 + \dots + c_n \underline{x}_n) = c_1 \lambda_1 \underline{x}_1 + \dots + c_n \lambda_n \underline{x}_n$. **Now** $A^2 \underline{u}_0 = c_1 \lambda_1^2 \underline{x}_1 + \dots + c_n \lambda_n^2 \underline{x}_n$; $A^n \underline{u}_0 = c_1 \lambda_1^n \underline{x}_1 + \dots + c_n \lambda_n^n \underline{x}_n$. And $A^k \underline{u}_0 / \lambda_1^k = c_1 \underline{x}_1 + (\lambda_2/\lambda_1)^k \underline{x}_2 + \dots + (\lambda_n/\lambda_1)^k \underline{x}_n$. As $k \rightarrow \infty$, $(\lambda_i/\lambda_1)^k \rightarrow 0$; $A^k \underline{u}_0 / \lambda_1^k \rightarrow c_1 \underline{x}_1$; and $A(A^k \underline{u}_0 / \lambda_1^k) \rightarrow \lambda_1 (c_1 \underline{x}_1)$. **Example:** $A = \begin{bmatrix} -10 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$. This has 2 *greatest eigenvalues*, and the \underline{u}_i oscillates between 2 combinations of \underline{x}_1 and \underline{x}_2 .

(2) **Inverse Power Method.** $A\underline{x} = \lambda \underline{x} (\lambda \neq 0) \Rightarrow A^{-1} A \underline{x} = \lambda A^{-1} \underline{x} \Rightarrow (1/\lambda) \underline{x} = A^{-1} \underline{x} \Rightarrow A$ and A^{-1} have the **same** eigenvectors, but **inverse** eigenvalues. The greatest eigenvalue of A^{-1} is the *least* eigenvalue of A . BUT, we prefer **not** to invert A . Instead of $\underline{v} := A^{-1} \underline{u}$, we solve $A \underline{v} = \underline{u}$ for \underline{v} using $A = LU$ and $\{L\underline{c} = \underline{u}, U\underline{v} = \underline{c}\}$.

Algorithm: Set $\underline{u} = [1, 1, \dots, 1]^t$. Factorise $A = Lu$. Repeat: Solve $A \underline{v} = \underline{u}$ by $\{L\underline{c} = \underline{u}, U\underline{v} = \underline{c}\}$; $m_i =$ greatest element in \underline{v} ; $\underline{u}_i = \underline{v}/m_i$; — until.... The **final** \underline{u} is the required *eigenvector*; and the final m is $(1/\text{required eigenvalue})$, or $1/m =$ *required eigenvalue*.

(3) Shifted **Inverse Power Method.** This method finds the eigenvalue *nearest to the* chosen μ . Method: $B = A - \mu I$. Apply (2) to B . The **Eigenvalue** we get is $\mu + 1/m$. Recall that $A \underline{x} = \lambda \underline{x}$ — this *implies that* $A^{-1} \underline{x} = \lambda^{-1} \underline{x}$. Similarly, $(A - \mu I) \underline{x} = (\lambda - \mu) \underline{x}$. And $(A - \mu I)^{-1} \underline{x} = (1/(\lambda - \mu)) \underline{x}$. The *smallest* eigenvalue of $A - \mu I$ is the *eigenvalue* of A nearest to $\mu = \mu + 1/m$, where m is the result of applying the *power method* to $(A - \mu I)^{-1}$.

Algorithm: Factorise $(A - \mu I)$ as LU . Proceed as with the *inverse power method*. It works if μ is a reasonable guess. It fails if more than one is “**nearest**” to μ . (Likely to oscillate).

➤ 8th November 1999

Results on Eigenvalues

Take A to be *complex* (even if it is real). $A = [a_{ij}] \Rightarrow \bar{A} = [\bar{a}_{ij}]$ is the *conjugate* of A . $A \in \mathbf{R} \Rightarrow \bar{A} = A$ ($(3+4i) = 3-4i$). **Transpose:** $A^t = [a_{ji}]$. If $\underline{x} \in \mathbf{C}^n$, it follows that $|\underline{x}| = \sqrt{(\sum_{i=1}^n |\underline{x}_i|^2)} = \sqrt{(\underline{x} \bar{\underline{x}}_1 + \dots + \underline{x}_n \bar{\underline{x}}_n)}$. ($|z| = \sqrt{(\bar{z}z)}$) ($|3+4i| = \sqrt{[(3+4i)(3-4i)]} = \sqrt{(9+16)} = 5$). So $\underline{x} = \begin{pmatrix} 3+4i \\ 1-i \end{pmatrix}$ has *length* $\sqrt{(25+2)} = 3\sqrt{3}$. The *eigenvalues* of A are roots of the characteristic polynomial $\text{ch}(A)$. When A is real, this set of roots may contain **complex** conjugate pairs. A complex polynomial equation of degree n has n *complex* roots.

λ	$p_0(\lambda)$	$p_1(\lambda)$ $-6-\lambda$	$p_2(\lambda)$ $(-2-\lambda)p_1-1$	$p_3(\lambda)$ $(3-\lambda)p_2-4p_1$	$p_4(\lambda)$ $(6-\lambda)p_3-p_2$	$f(\lambda)$	<i>sign changes</i>
-7	1						
-6	1						
.....	1						
3	1	-9	44	36	64	2	+ - + + +
4	1	-10	59	-19	-97	3	+ + + - -
...	1						
7	1						

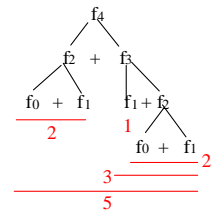
Conclusion: the eigenvalue lies *between 3 and 4*.

➤ 22nd November 1999

Recursion

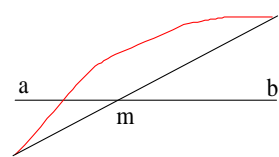
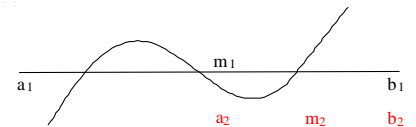
$n! = n(n-1)(n-2)...1$ or $n! = 1.2.3...(n-1)n$, or $n! = (\text{recursive}) = 1$ when $n = 0$, and $n(n-1)$ otherwise. **Compute** this: **f := 1, for i from 2 to n do f := f*i; od: print(f)**. Or, we could use a *procedure*: **fact := proc(n), if n = 0 then 1: else n*fact(n-1): fi: end:**. Analysing this, we start *with* $\text{fact}(4)$, then go to $4 \times \text{fact}(3)$, then $4 \times 3 \times \text{fact}(2)$, etc., to get the answer of 24. A recursive procedure calls itself. **Short** to code, but **hard** to Debug. *Multiple* copies of local variables.

Fibonacci numbers: $f_0 = f_1 = 1, f_n = f_{n-1} + f_{n-2}$. We could have the *following* code: **fib := proc(n), option remember, if n = 0 or n = 1 then 1: else fib(n-1)+fib(n-2): fi: end:**. A diagram showing how $\text{fib}(4)$ is calculated is shown. But note that f_2 is calculated twice! It gets *worse*: 5 calculations for f_6 !



That is why we use the **option remember** — it causes a *remember* table to be initialised which stores all calculated values of fib — so no value is calculated twice. Therefore, part of the tree is not required. (Note: there are more *efficient* quadratic methods for f_n).

Legendre Polynomials: $P_n(x)$ is of degree n in x . $P_0(x) = 1, P_1(x) = x, P_n(x) = (2n-1/n)P_{n-1}(x) - (n-1/n)P_{n-2}(x)$. **Solve** $f(x) = 0$ by *bisection*. Assume that $a < b$, and that $f(a)$ & $f(b)$ have opposite sign. This implies that $f(x) = 0$ has a solution in $[a,b]$. **Method:** Let m be the mid-point of $[a,b]$, $m = (a+b)/2$. Either $f(a)$ and $f(m)$ have *opposite* signs, or $f(m)$ and $f(b)$ have *opposite* signs. So the root is in one half or the other. **Iterate** until the size of interval becomes $< 10^{-10}$, say.

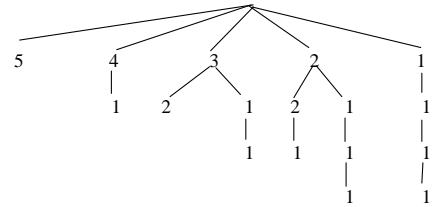


Aside: the secant method gives a *better* m : $y - f(a) = \frac{f(b) - f(a)}{b - a}(x - a)$. At $(m, 0)$, $-(b-a)f(a) = (f(b) - f(a))(m - a)$. **Solve** for m . Now we could *code* the above example as follows: **f := x → x⁵ - 3x³ + ...; eps := 10⁻¹⁰; bisect := proc(a,b); local m; global f, eps; <add checks here>, m := (a+b)/2; if (b-a < eps) then m: elif f(m) = 0 then m: elis sign(f(a))*sign(f(m)) < 0; bisect(a,m); else bisect(m,b); fi: end:**. Note that a, b and m are needed for *each* recursive call.

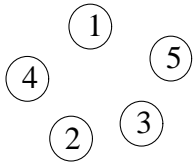
Backtrack Trees

Problem: Make a sequence of choices. Method: To make the *next* choice, order the allowed choices. Make each one in turn iterate; print it if it is the solution; then undo this choice and carry on. The nodes at level i in a backtrack tree represent a **partial** solution with i choices already made.

Example: Combinatorics: *integer* partitions. 5 can be represented by the following (decreasing) sequences. $5 = 5 = 4+1 = 3+2 = 3+1+1 = 2+2+1 = 2+1+1+1 = 1+1+1+1+1$. In the i choice, choose the i^{th} path in a partition which *cannot* be larger than the previous part, and cannot be too big. The tree for this problem is shown on the **right**.



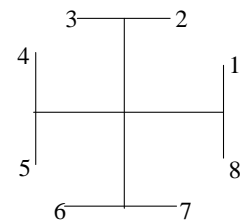
Another example: we have a ring of numbers as shown on the left. We choose one, and then go around (clockwise or anticlockwise) by the amount of steps indicated by the number on the circle we are on. Question: Is it possible to visit *each* number in exactly 5 steps, i.e. no number appears **twice**? We draw a tree as shown *above* for this problem. The solutions are 3-1-4-2-5 and 3-4-2-1-5.



Maple notes: "term" has form [scalar, monomial]. "polynomial" has form [term, term, term]. Example: $[[-17, x_1^2x_2], [3, x_2^{19}]]$ represents $-17x_1^2x_2 + 3x_2^{19}$.

Knight's Tour

A knight in the *centre of a chess board* has eight possible moves. (A move is: "go 2 moves in one direction, then one perpendicularly"). Problem start at a **chosen** square, and visit each square exactly once. In a 3×3 board, we can't do this — we can't reach the centre. Produce a program with a *recursive* algorithm to see if it is possible with *other* sized boards.



Matrix Multiplication

Consider $AB = C$, where A is an $m \times p$ matrix, B is a $p \times n$ matrix, and C is an $m \times n$ matrix. A **position** in C , C_{ij} , is given by $\sum_{k=1}^p a_{ik}b_{kj}$. This can be *coded up*. Restricted problem: calculate UAU , where U is an $n \times n$ identity matrix with an $m \times n$ matrix in the bottom right hand corner. Note: H is **symmetric**. General example: $U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & a_c & 0 \\ 0 & 0 & b_d \end{bmatrix}$.

We solve this by *block multiplication*. Partition the matrices into parts, and we get what is shown on the right. (B is an $(n-m) \times (n-m)$ matrix, C is an $(n-m) \times 1$ matrix, D is an $m \times (n-m)$ matrix, and E & H are $m \times 1$ matrices).

$$\begin{aligned}
 UAU &= \begin{bmatrix} I & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} B & C \\ D & E \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & H \end{bmatrix} \\
 &= \begin{bmatrix} B & C \\ HD & HE \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & H \end{bmatrix} \\
 &= \begin{bmatrix} B & CH \\ HD & HEH \end{bmatrix}
 \end{aligned}$$

Exercise: If $\underline{x} = ({}^5_3\sqrt{2})$ and $z = ({}^1_0_0)$, $\sigma = \|\underline{x}\|$ and $v = \underline{x} + \sigma \underline{z}$, calculate $H = I_3 - \frac{2}{\|v\|^2} vv^t$. **A:** Here, $\sigma = \sqrt{(25+9+2)} = \sqrt{36} = 6$. So $\underline{v} = ({}^5_3\sqrt{2}) + ({}^6_0_0) = ({}^{11}_3\sqrt{2})$. And $\|\underline{v}\|^2 = 11^2 + 9 + 2 = 132$. So $H = ({}^1_0_0 \ 0 \ 0_1) - \frac{2}{132} ({}^{11}_3\sqrt{2})({}^{11} \ 3 \ \sqrt{2}) = ({}^1_0_0 \ 0 \ 0_1) - \frac{2}{132} ({}^{121}_{33} \ 33 \ 11\sqrt{2} \ 33 \ 9\sqrt{2} \ 11\sqrt{2} \ 3\sqrt{2} \ 2) = -\frac{1}{132} ({}^{-110} \ -66 \ -22\sqrt{2} \ -66 \ 114 \ -6\sqrt{2} \ -22\sqrt{2} \ -6\sqrt{2} \ 128)$. **This is symmetric and orthogonal!** **Lemma:** $H\underline{x} = ({}^\sigma_0 \dots_0) = \sigma \underline{z}$.

➤ 6th December 1999

Tridiagonalise a Symmetric Matrix A

After performing what is shown on the **right** iterate on the 2nd row and column, and so on, to get to the solution. Algorithm for a symmetric A: Let T = the *tridiagonal* form of A, done by a Householder transformer. Let [lower, upper] be the Gerschgorin range for the eigenvalues.

1	0	A ₁₁	x ^t	1	0
0	H ₁	x ₁	A _{1'}	0	H ₁

A ₁₁	x ^t	1	0
-σ ₁	H ₁ A _{1'}	0	H ₁
0			
0			
0			
..			
0			

A ₁₁	-σ ₁	0	0	0
-σ ₁	0	0	0	0	A ₁ = H ₁ A ₁ H ₁
0					
0					
0					
..					
0					

Use Sturm sequences with bisection to locate all the **eigenvalues**, to 10⁻³, say. The shifted inverse power method then improves these estimates and gives *eigenvectors*. This uses factorisation (T-(estimate)I) = LU).

Note: the algorithm fails with an (e.g.) matrix with zeros in the 2nd and 3rd quadrants — we have division by zero.

➤ 8th December 1999

Propagation of Round-off Errors

Addition: $(a+\delta a) + (b+\delta b) = (a+b) + (\delta a + \delta b)$. **Multiplication:** $(a+\delta a)(b+\delta b) = ab + (a\delta b + b\delta a) + \delta a\delta b$. We ignore the *third* term (it is small) — the *second* term is the actual error. The relative error is $\frac{a\delta b + b\delta a}{ab} = \frac{\delta b}{b} + \frac{\delta a}{a}$, the sum of the *relative* errors in a and b. **Consider** a 2x2 determinant, $|\begin{smallmatrix} a+\delta a & b+\delta b \\ c+\delta c & d+\delta d \end{smallmatrix}|$, which is approximately $(ad-bc) + (a\delta d - b\delta c - c\delta b + d\delta a)$. If a, b, c and d are “large”, and ad is approximately equal to bc, the error may become *significant*.

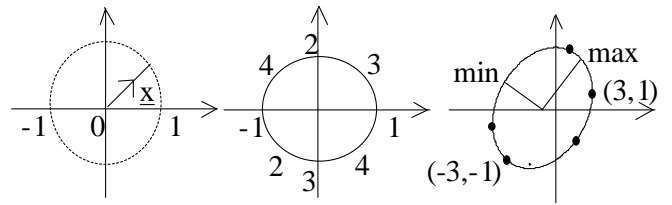
Exercise: consider the equations $x+y = 2$ and $x+1.0001y = 2.0001$. Solution: $x = 1$ and $y = 1$. If we have equations $x+y = 2$ and $x+1.0001y = 1.9999$, then we have the **solutions** $x = 3$ and $y = -1$. This is an *ill-conditioned system of equations* (the determinant is 0.0001). The eigenvalues are approximately 2.0005 & 0.00005.

Note: the system $A\underline{x} = \underline{b}$ is *ill-conditioned* if small changes in A and/or \underline{b} produce relatively *large* changes in \underline{x} .

Use the **condition number** of A, $C_A = \sqrt{(\lambda_{\max})/(\lambda_{\min})}$, where λ_{\max} & λ_{\min} are the greatest and least *eigenvalues* of A^tA (a symmetric & +ve definite matrix). The norm of A is $\max_{\underline{x} \neq 0} \|\underline{Ax}\|/\|\underline{x}\| (= \sqrt{\lambda_{\max}})$. What direction is *most* expanded by A? — use $\|\underline{x}\| = \sqrt{\underline{x} \cdot \underline{x}}$.

Example: Let $A = \begin{pmatrix} 3 & 1 \\ 1 & 4 \end{pmatrix}$. C_A for $\begin{pmatrix} 1 & 1 \\ 1.0001 & 1 \end{pmatrix}$ is approximately 4000. When $A\underline{x} = \underline{b}$ is perturbed to $(A+\delta A)(\underline{x}+\delta\underline{x}) = \underline{b}+\delta\underline{b}$, then

$$\|\delta\underline{x}\|/\|\underline{x}\| \leq C_A \left(\frac{\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta\underline{b}\|}{\|\underline{b}\|}}{1 - C_A \frac{\|\delta A\|}{\|A\|}} \right).$$



So we want C_A to be *small*. If C_A is approximately 1, then the *relative* error in \underline{x} is approximately (the **relative** error in A) + (the **relative** error in b).

Tip: When working with matrices to '*d*' *significant digits*, use 2d digits to perform dot products (because the vectors may be **nearly** orthogonal).