

Note below, NCM="Numerical Computing in MATLAB ", the text for the course.

1. One of the major breakthroughs in computing π occurred in 1671 when James Gregory obtained the infinite series expansion for $\arctan(x)$:

$$\arctan(x) = \sum_{j=0}^{\infty} \frac{(-1)^j x^{2j+1}}{2j+1}. \quad (1)$$

- (a) Using $x = 1$ in (1), obtain an infinite series expansion for computing π . (Note that the series expansion you should obtain was known by Leibniz in 1674).
- (b) Using $x = \frac{1}{\sqrt{3}}$ in (1), obtain a different infinite series expansion for computing π .
- (c) Write a MATLAB function for approximating π using the series you developed in parts a and b.
 - i. Compute the approximation using $N = 11, 101, 1001, 10001$, and 100001 terms and report the results in a table.
 - ii. Compute the error in the approximation using the built-in constant for π from MATLAB (or some other suitable language) and report the results in a table.
 - iii. Plot the absolute value of the error as a function of the number of terms used in the sum. Use a log scale on both the x and y -axis of your plot to obtain a nice picture of the error curves. Include these plots in your assignment.
- (d) You should find that the series in part (b) gives a much better approximation to π than the one in part (a) as the number of terms N increases. Explain why this is the case.

Hint: Recall the domain where a Taylor series is convergent.

- e. A much more efficient method for computing π is based on an *arithmetic geometric mean* (AGM) iteration. The AGM of two positive numbers a_0 and b_0 is $a = \lim_{j \rightarrow \infty} a_j = \lim_{j \rightarrow \infty} b_j$, where

$$a_{j+1} = \frac{a_j + b_j}{2}, \text{ and, } b_{j+1} = \sqrt{a_j b_j}.$$

Gauss first showed how this iteration was related to complete elliptic integrals of the first kind. It was later shown in the 1970's how the method could also be used to compute π as illustrated by the following algorithm:

```

a = 1;
b = 1/√2;
t = 1/4;
j = 1;
while a - b ≥ ε do
  y = a;
  a = (a + b)/2;
  b = √by
  t = t - j(a - y)2;
  j = 2j;
end while
return a2/t;

```

Implement this algorithm in MATLAB . Run the program with $\varepsilon = 10^{-13}$ and report the value of a^2/t through each iteration of the while loop. Compare this value to π .

The following are some MATLAB commands that may be useful for the programs above:

`pi, for, while, sum, loglog, sqrt`

Help on these commands can be obtained by typing

`>>help <command>`

at the MATLAB command prompt.

- NCM, problem 1.38
- Matrices of size m -by- n with random entries can be created in MATLAB with the command `rand(m,n)` (note that if $m = n$ then the command `rand(n)` can be used). Create four random matrices A , B , C , and D , such that A is 2-by-3, B is 3-by-3, C is 3-by-2, and D is 3-by-3. Perform the following operations on these matrices using MATLAB . If the operation cannot be performed, explain (mathematically) why not.

- | | | |
|----------------------------|----------------------------|-------------------|
| (a) $2A + C^T$ | (b) $C - 3B$ | (c) $3B - 2D$ |
| (d) AD | (e) CA | (f) AC |
| (g) BD | (h) DB | (i) BC |
| (j) CB | (k) AB | (l) $2D^T + B$ |
| (m) $\det(D)$ | (n) $\det(A)$ | (o) $C^T D$ |
| (p) BA^T | (q) $-2A^T + 5C$ | (r) $B^T + D$ |
| (s) $\frac{1}{2}(B + B^T)$ | (t) $\frac{1}{2}(B - B^T)$ | (u) AA^T |
| (v) $A^T A$ | (w) $\det(AA^T)$ | (x) $\det(A^T A)$ |
| (y) $B(AD)^T$ | (z) ADB^T | |

Do not turn in the individual output of each of the valid operations, simply turn in the MATLAB statements you used for each of the operations.

- NCM, problem 2.3. Please also explain where the factor $\alpha = 1/\sqrt{2}$ comes from in the set of equations.
- NCM, problem 2.4
- Recall the set of equations introduced in class for modeling an n -stage countercurrent chemical extraction reactor:

$$\begin{aligned} -(W + Sm)x_1 + Smx_2 &= -Wx_{\text{in}}, \\ Wx_{i-1} - (W + Sm)x_i + Smx_{i+1} &= 0 \quad (i = 2, 3, 4, \dots, n-1), \\ Wx_{n-1} - (W + Sm)x_n &= -Sy_{\text{in}}, \end{aligned}$$

where W is the mass flow rate of the incoming water sample containing some chemical with a mass fraction x_{in} , S is the mass flow rate of the solvent containing an incoming mass fraction of the same chemical of y_{in} , and m is a constant that depends on the chemical and solvent.

- Write a MATLAB function that computes the mass fraction of the chemical in the water stream at each of the n stages (i.e. x_i , $i = 1, 2, \dots, n$). Your function should take as input W (in kg/hr), S (in kg/hr), x_{in} , y_{in} , m , and n . No explicit looping structures should be done in your function. Turn-in a print out of your code.

Hint: The following commands may be useful for constructing the matrix for determining x_i : `diag` or `spdiags`. You will want to use the backslash operator `\` for solving the linear system of equations.

- Use your function to solve the n -stage countercurrent chemical extraction problem with $M = 150$ kg/hr, $S = 25$ kg/hr, $x_{\text{in}} = 0.1$, $y_{\text{in}} = 0$, $m = 5$, and $n = 50$. Plot the mass fraction vs. the stages of the extraction. How much has the original chemical in the water been reduced by at the end of the extraction procedure?