

- 565 students must answer all the questions. 465 students must answer all questions that do not explicitly state [565 only].
- Teamwork is encouraged! Please let me know if you are having trouble finding someone to team up with.

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

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

(Actually, the Indian mathematician Madhava of Sangamagrama had discovered the same formula some 300 years earlier, so Gregory really *rediscovered* it).

- (a) Using $x = 1$ in (1), obtain an infinite series expansion for computing π .
- (b) Using $x = \frac{1}{\sqrt{3}}$ in (1), obtain a different infinite series expansion for computing π .
- (c) Write a computer program to approximate 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 and look at the remainder term in the Taylor expansion.

- (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 = sqrt(b*y)
t = t - j(a - y)^2;
j = 2j;
end while
return a^2/t;

```

Write a program (using for example MATLAB) that implements this algorithm. 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 . What type of convergence does the iteration appear to have.

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.

2. Suppose that you are working with a limited computer that only supports addition, subtraction, multiplication, division, rounding, and computing integer powers of numbers. To add more functionality to your computer, you would like to add a function for computing the exponential function e^x using only the arithmetic operations provided. Recall from Calculus that, for any real (or complex) value of x , the exponential function $f(x) = e^x$ can be represented by the infinite series

$$e^x = \lim_{n \rightarrow \infty} \sum_{j=0}^n \frac{x^j}{j!},$$

which converges for $-\infty < x < \infty$. Since you learned Calculus because you thought it would be useful, you might expect to use this series in your algorithm for computing e^x .

- (a) Create a function (using for example MATLAB) that computes e^x by truncating the series above to n terms. Using your function for $n = 10$, compute an approximation to e^x for $x = \pm 0.5$ and $x = \pm 30\pi$, and report the relative error in your approximation. Repeat these computations for $n = 40$. Comment on your results.
- (b) By exploiting properties of the exponential, design an algorithm for accurately computing the value of e^x using the series above, for a wide range of x (e.g. $-100 \leq x \leq 100$). This algorithm can only use the basic operations of the computer described above. Create a function for your algorithm and use it, with $n = 10$, for computing approximations to e^x for $x = \pm 0.5$ and $x = \pm 30\pi$. Report the relative error in these approximations. If done correctly, your algorithm should produce approximations for these values of x with relative errors $< 10^{-13}$.

Hint: Based on the remainder theorem for Taylor series, for what values of x do you expect the above series to be the most accurate? Exploit the power property of the exponential so that your algorithm only uses the series approximation for e^x for this range of x .

Actual algorithms for computing e^x use a rational function (ratio of two polynomials) approximation instead of a Taylor series approximation or they use an *arithmetic geometric mean* iteration.

- (c) [565 only] Explain how your algorithm from part (b) can be used for approximating $\cos(x)$ and $\sin(x)$.

The following are some MATLAB commands that may be useful for the programs from (a)-(b) above:

```
for, while, if, exp, round, abs
```

The MATLAB function `gamma` can be used for computing factorials using the relationship $k! = \text{gamma}(k+1)$, for some nonnegative integer k .

3. (a) Verify that

$$f(x) = 1 - \tan x \quad \text{and} \quad g(x) = \frac{\cos 2x}{\cos^2 x(1 + \tan x)}$$

are identical functions.

- (b) Which function should be used when evaluating x near $\pi/4$ and $5\pi/4$? Why?
 (c) Which function should be used when evaluating x near $3\pi/4$ and $7\pi/4$? Why?
4. Solve the following linear system of equations by hand in exact arithmetic using Gaussian elimination with partial pivoting and backward/forward substitution:

$$\begin{cases} x_1 - 2x_2 + x_3 - x_4 = -5 \\ x_1 + 5x_2 - 7x_3 + 2x_4 = 2 \\ 3x_1 + x_2 - 5x_3 + 3x_4 = 1 \\ 2x_1 + 3x_2 - 5x_3 = 17 \end{cases}$$

Show all your steps, and describe in words what you are doing in each step.

5. 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.

6. [565 only] Suppose we wish to evaluate the integral

$$I(N) = \int_N^{N+1} \frac{1}{1+x^2} dx$$

using the analytical result

$$I(N) = \arctan(N+1) - \arctan(N). \tag{2}$$

- a. Explain why (2) is unacceptable for computations when N is large, e.g. when $N = 10^8$.
 b. Find some exact way to re-write (2) in an acceptable form for computation when N is large.
 c. Compare your formula in Part b with the formula in (2) for $N = 10^8$.

Hint: When you evaluate $I(N)$ using the formula you derived in Part b, you should obtain a value like $I(N) \approx 9.99 \times 10^{-17}$, not $I(N) = 0$.

7. [565 only] Straightforward block partitioned 2×2 matrix multiplication $C = AB$ can be written

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

where

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

In the scheme discovered by Strassen (1969), the computations are rearranged into

$$\begin{aligned} P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\ P_2 &= (A_{21} + A_{22})B_{11} \\ P_3 &= A_{11}(B_{12} - B_{22}) \\ P_4 &= A_{22}(B_{21} - B_{11}) \\ P_5 &= (A_{11} + A_{12})B_{22} \\ P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\ P_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

$$\begin{aligned} C_{11} &= P_1 + P_4 - P_5 + P_7 \\ C_{12} &= P_3 + P_5 \\ C_{21} &= P_2 + P_4 \\ C_{22} &= P_1 - P_2 + P_3 + P_6 \end{aligned}$$

- Verify analytically (by hand or using *Maple*) that the Strassen algorithm will give the same result as the traditional algorithm. Make sure to check that we never assume that matrix multiplication is commutative.
- Suppose we start Strassen's method with two $n \times n$ matrices where $n = 2^m$ and use the algorithm repeatedly until all matrices have become size 1×1 . Show (for example using induction) that the total arithmetic operation count becomes $7 \cdot 7^m - 6 \cdot 4^m$ (i.e. $O(n^{\log_2 7}) \approx O(n^{2.81})$).
- Verify numerically that Strassen's algorithm works by means of running the following MATLAB code for a few cases with random input matrices of size $n = 2^m$, $m = 4, 5, 6$, and 7, and compare against regular matrix multiplication.

```
00001 function c = strass(a,b)
00002 nmin = 16;
00003 [n,n] = size(a);
00004 if n <= nmin;
00005     c = a*b;
00006 else
00007     m = n/2; u = 1:m; v = m+1:n;
00008     p1 = strass(a(u,u)+a(v,v),b(u,u)+b(v,v));
00009     p2 = strass(a(v,u)+a(v,v),b(u,u));
00010     p3 = strass(a(u,u),b(u,v)-b(v,v));
00011     p4 = strass(a(v,v),b(v,u)-b(u,u));
00012     p5 = strass(a(u,u)+a(u,v),b(v,v));
00013     p6 = strass(a(v,u)-a(u,u),b(u,u)+b(u,v));
00014     p7 = strass(a(u,v)-a(v,v),b(v,u)+b(v,v));
00015     c = [p1+p4-p5+p7,p3+p5; p2+p4, p1-p2+p3+p6];
00016 end
```

- Explain why the MATLAB code for Strassen's method above introduces `nmin=16` and then uses the `if` statement on line 5 to check if `n <= nmin`.

To create a random matrix of size $n = 2^m$ in MATLAB , use the following command

```
>> a = rand(2^m)
```