Math 3607: Homework 4

Wednesday, February 16, 2022

TOTAL: 30 points

- Problems marked with ✎ are to be done by hand; those marked with 💻 are to be solved using a computer.
- **Important note.** Do not use *Symbolic Math Toolbox*. Any work done using `sym` or `syms` will receive NO credit.
- **Another important note.** Starting from this assignment, you will be asked to write MATLAB functions. Instead of writing an external function m-file, include all your functions at the end of your live script.

1. (Sliders moving along grooves; adapted from **LM** 2.1–12 and Sample HW01) The mechanical device shown in Figure 1 consists of two grooves in which sliders slide. These sliders are connected to a straight rod.

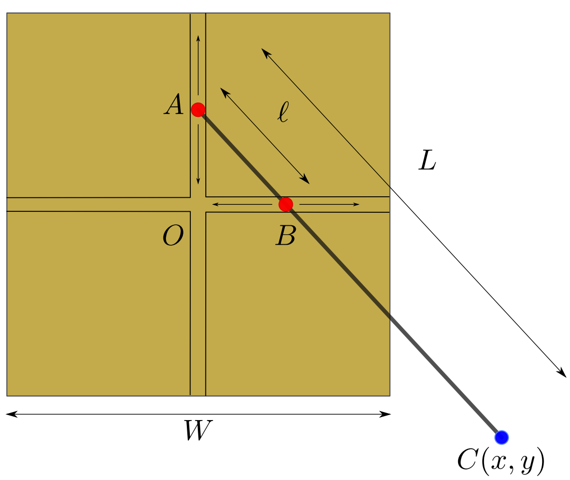

Figure 1: The bronze square is a piece of metal with two grooves cut out of it as shown. There are sliders at the points A and B which slide in these grooves. The slider at A can only slide vertically, and the one at B can only slide horizontally. There is a straight rod attached at A and B, which extends to C. As the point C moves around the block, it traces out a closed curve.

   (a) ✎ Analytically, determine the curve which is traced out by C in one rotation.

   Suggestion. Let (x, y) be the coordinates of the point C. Express the variables x, y in terms of L, ℓ, and θ, where $\theta \in [0, 2\pi)$ is the angle from the part of the horizontal groove which is to the right of B to the rod BC.

   (b) 💻 Using the previous result, plot the trajectory of C in one rotation for $\ell = 2$ and $L = 7$.

2. (Spiral triangles to spiral polygons; adapted from **LM** 5.9–7, 6.8–34) 💻 The following script[1] generates spirals using equilateral triangles as shown in the figure below.

[1] It is slightly modified from the code included in Lecture 9 slides. Note the introduction of a new variable `d_rot`, which is accountable for the rotation of the innermost triangle.

```
m = 21; d_angle = 4.5; d_rot = 90;
th = linspace(0, 360, 4) + d_rot;
V = [cosd(th);
     sind(th)];
C = colormap(hsv(m));
s = sind(150 - abs(d_angle))/sind(30);
R = [cosd(d_angle) -sind(d_angle);
     sind(d_angle) cosd(d_angle)];
hold off
for i = 1:m
  if i > 1
      V = s*R*V;
  end
  plot(V(1,:), V(2,:), 'Color', C(i,:))
  hold on
end
set(gcf, 'Color', 'w')
axis equal, axis off
```

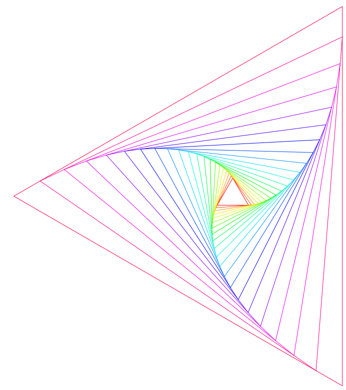

Figure 2: Spiral triangles with $m = 21$ and $\theta = 4.5°$.

(a) Write a function named `spiralgon` by modifying the script so that it generates spirals using m regular n-gons for any $n \geqslant 3$. Your function must be written at the end of your homework live script (.mlx) file. Begin the function with the following header and comments.

```
function V = spiralgon(n, m, d_angle, d_rot)
% SPIRALGON plots spiraling regular n-gons
% input:    n = the number of vertices
%           m = the number of regular n-gons
%           d_angle = the degree angle between successive n-gons
%           (can be positive or negative)
%           d_rot = the degree angle by which the innermost n-gon
%                   is rotated
% output:   V = the vertices of the outermost n-gon
....
```

(b) Run the statements below to generate some aesthetic shapes.

```
clf
subplot(2, 2, 1), spiralgon(3, 41, 4.5, -90);
subplot(2, 2, 2), spiralgon(4, 37, -2.5, 45);
subplot(2, 2, 3), spiralgon(5, 61, 3, -90);
subplot(2, 2, 4), spiralgon(8, 91, -4, 22.5);
```

Note. Copy the five lines, paste them inside a single code block, and run it. This code block must *precede* your function(s).

3. (Machine epsilon; adapted from **LM** 9.3–3(a)) Recall that the number in the computer which follows 1 is $1 + \boxed{\text{eps}}$, which can be verified in MATLAB by

```
>> format long
```

```
>> (1 + 0.5*eps) - 1
ans =
     0
>> (1 + 0.51*eps) - 1
ans =
     2.220446049250313e-16
```

In the same manner:

(a) Verify that the number in the computer which follows 8 is $8 + 8\boxed{\text{eps}}$ by numerically calculating $8 + 4\boxed{\text{eps}}$ and $8 + 4.01\boxed{\text{eps}}$.

(b) Verify that the number in the computer which precedes 16 is $16 - 8\boxed{\text{eps}}$ by numerically calculating $16 - 4.01\boxed{\text{eps}}$ and $16 - 4\boxed{\text{eps}}$.

(c) What are the numbers in the computer that precedes and follows $2^{10} = 1024$, respectively? Verify your claims in MATLAB by carrying out appropriate calculations.

Note. Begin with `format long` as shown in the example above. This is needed only once before the beginning of part (a).

Note. Answer each part of the problem in a single code block. No external script needs to be written.

4. (Catastrophic cancellation; **LM** 9.3–10) We revisit the function from Problem 3 of Homework 3. Consider the function

$$f(x) = \begin{cases} \dfrac{e^x - 1}{x} & \text{if } x \neq 0 \\ 1 & \text{if } x = 0, \end{cases}$$

Here we explore the catastrophic cancellation which occurs as $x \to 0$ since $e^x \to 1$ as $x \to 0$.

(a) Use the Taylor series expansion of e^x to prove that f is continuous at 0.

(b) Now calculate $f(x)$ numerically for $x = 10^{-k}$ where $k \in \mathbb{N}[1, 20]$ in three slightly different ways:

i. Calculate $f(x)$ as written.

ii. Calculate it as

$$f_1(x) = \frac{e^x - 1}{\log e^x}, \quad \text{for } x \neq 0.$$

(You and I know that analytically $f_1(x) \equiv f(x)$ for all nonzero x – but MATLAB doesn't.)

iii. MATLAB has a function which analytically subtracts 1 from the exponential to avoid catastrophic cancellation before the result is calculated numerically. So define the function $f_2(x)$ to be the same as $f(x)$ except that $e^x - 1$ is replaced by `expm1(x)`.

Tabulate the results using `disp` or `fprintf`. The table should have four columns with the first being x, the second using $f(x)$, the third using $f_1(x)$, and the fourth using $f_2(x)$, with all shown to full accuracy. Do it as efficiently as you can, without using a loop.

Note. Write your code for this part in a single code block. No external script needs to be written.

(c) Comment on the results obtained in the previous part. Explain why certain methods work well while others do not.

5. (Inverting hyperbolic cosine; **FNC** 1.3.6) The function

$$x = \cosh(t) = \frac{e^t + e^{-t}}{2}$$

can be inverted to yield a formula for $\operatorname{acosh}(x)$:

$$t = \log\left(x - \sqrt{x^2 - 1}\right). \tag{$\star$}$$

In MATLAB, let `t=-4:-4:-16` and `x=cosh(t)`.

(a) Find the condition number of the problem $f(x) = \operatorname{acosh}(x)$ by hand. (You may use Equation ($\star$), or look up a formula for f' in a calculus book.) Then evaluate κ_f at the elements of `x` in MATLAB.

(b) Evaluate the right-hand side of Equation ($\star$) using `x` to approximate `t`. Record the accuracy of the answers (by displaying absolute and/or relative errors), and explain. (Warning: Use `format long` to get enough digits or use `fprintf` with a suitable format.)

(c) An alternate formula for $\operatorname{acosh}(x)$ is

$$t = -2\log\left(\sqrt{\frac{x+1}{2}} + \sqrt{\frac{x-1}{2}}\right). \tag{$\dagger$}$$

Apply Equation ($\dagger$) to `x` and record the accuracy as in part (b). Comment on your observation.

(d) Based on your experiments, which of the formulas ($\star$) and ($\dagger$) is unstable? What is the problem with that formula?

Note. Write your code for each of parts (a), (b), and (c) in a single code block. No external script needs to be written.