PHYS 371 - Assignment I

Celil Buğra Karacan
Deepartment of Physics, Bilkent University
(Dated: October 15, 2018)

I. INTRODUCTION

A. Question I

Given function (below), as it turns out represents an ellipse with eccentricity ϵ. Eccentricity being ratio of distances from focus and semi-major axis to center of the ellipse.

$$r(\theta) = \frac{1-\epsilon^2}{1-\epsilon * cos(theta)} \qquad \epsilon = \frac{f}{a}$$

B. Question II

It is asked to calculate absolute fractional error for partial Taylor expension of e^x consisting `N+1` terms. For $x > 0$ exponential decay is expected, however, for $x < 0$ the alternating sum would make error "bounce" to max out at $x = N$, since after that point demoninator(x^j) grows faster than nominator($j!$).

II. METHOD

A. Question I

There seemed two possible solutions to Part-A, creating an array using `linspace()` or defining a symbolic variable θ. Latter had been adapted and `r(`θ`)`, `x(`θ`)`, `y(`θ`)` were defined accordingly. `ezplot()` was used, since function to be drawn were implemented symbolically rather than as arrays. For Part-B $x = r(\theta) * cos(\theta)$ and $y = r(\theta) * sin(\theta)$ were opted to convert from polar to cartesian coordinates. Lastly `subplot()` function was utilized to draw plots for different ϵ values all in one figure.

B. Question II

Two similar functions, namely `err_plot` and err_plot_mod, were defined for Part-A and Part-B respectively. As in Question I, `subplot()` within a `for` loop was utilized here also to provide a comparative figure.
`err_plot` creates an array of integers from 0 to `N` with `linspace()`, sets `sum` as 1, and within a `for` loop increments `sum` $S(x,N) = \sum \frac{x^j}{j!} \approx e^x$ while calculating absolute fractional error ($err = \frac{|S(x,N)-e^x|}{e^x}$) in each step. Then, `plot()` draws `err` versus `n`.
`err_plot_mod` does the above all for $x < 0$, otherwise it uses following function to calculate `err` instead:

$$\text{err } = \frac{|\frac{1}{S(-x,N)} - e^x|}{e^x}, \qquad \frac{1}{S(-x,N)} \approx e^x$$

Where $-x$ is positive number which eliminates fault seen in prior algorithm.

III. VERIFICATION OF THE PROGRAM

A. Question I

Code works as expected for both Part-A and Part-B. Notice that for $\theta = 0$ and $\theta = 2 * \pi$, $r = 1 + \epsilon$ holds at Part-A, and $\epsilon = 0$ gives a circle at Part-B (See: FIG.1.).

B. Question II

Code works for $x = 10$ and $x = -10$, however at the case of $x = 2$ and $x = -2$ there seems to be a "cavity" at $N = 2$ (See: FIG.2.). Exponential decay is present in each case after correcting alternation problem, yet the reason of `err` approaching 0 at $N = 2$ for $x = 2$ (should have been ≈ 0.84) and $x = -2$ (should have been ≈ 0.63) remains a mystery. Also for those cases mentioned, as N increases err converges to a positive number.

IV. ANALYSIS

A. Question I

As ϵ gets larger, ellipse should get more stretched.

B. Question II

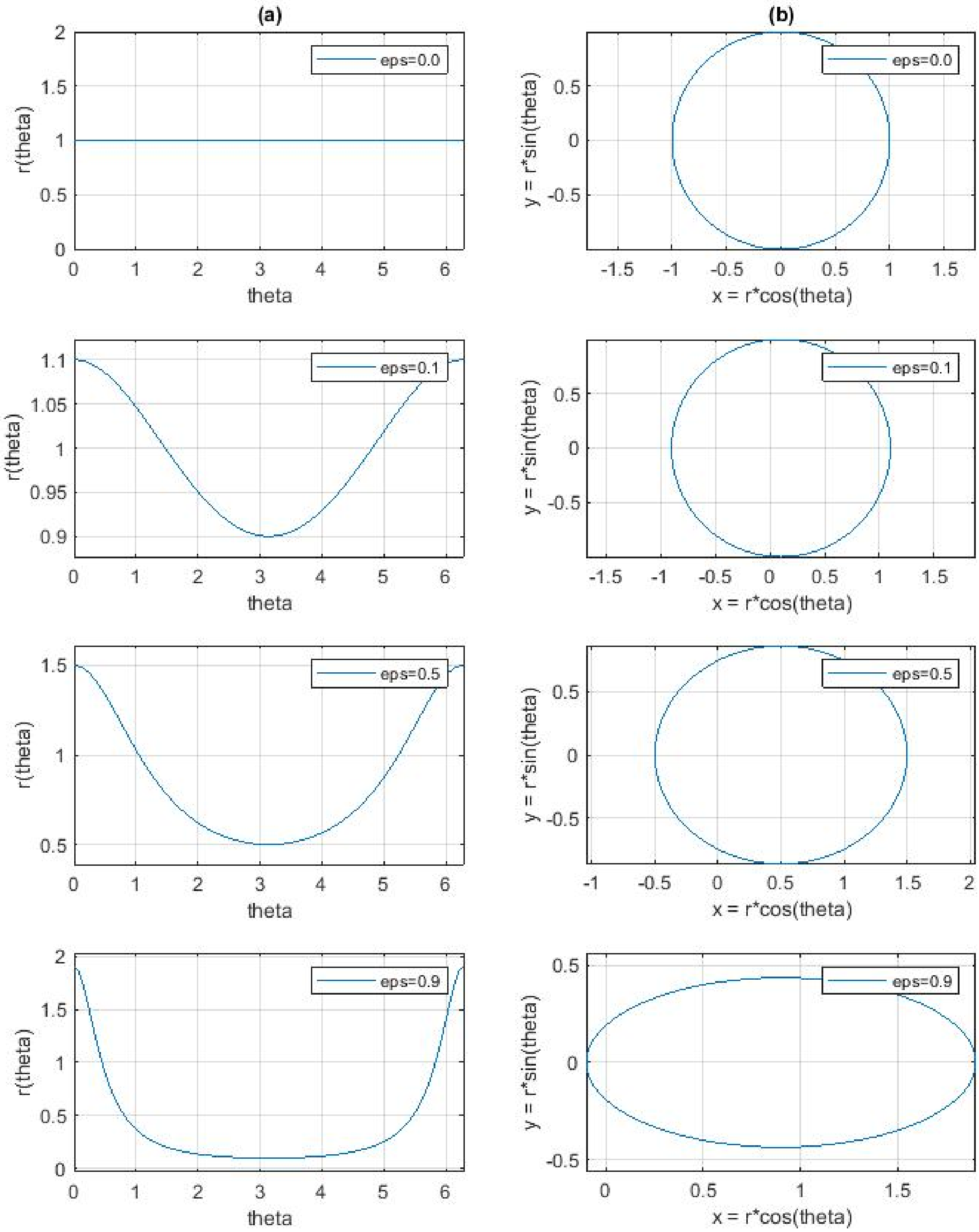
Contarary to problematic results mentioned earlier, `err` should converge to 0, as approximation gets better and better as `N` increases.

V. INTERPRETATION

Function of ellipses drawn could not be interpreted.

VI. DATA

FIG. 1. θ vs $\rho(\theta)$ and
$x(\theta) = \rho(\theta) * cos(\theta)$ vs $y(\theta) = \rho(\theta) * sin(\theta)$



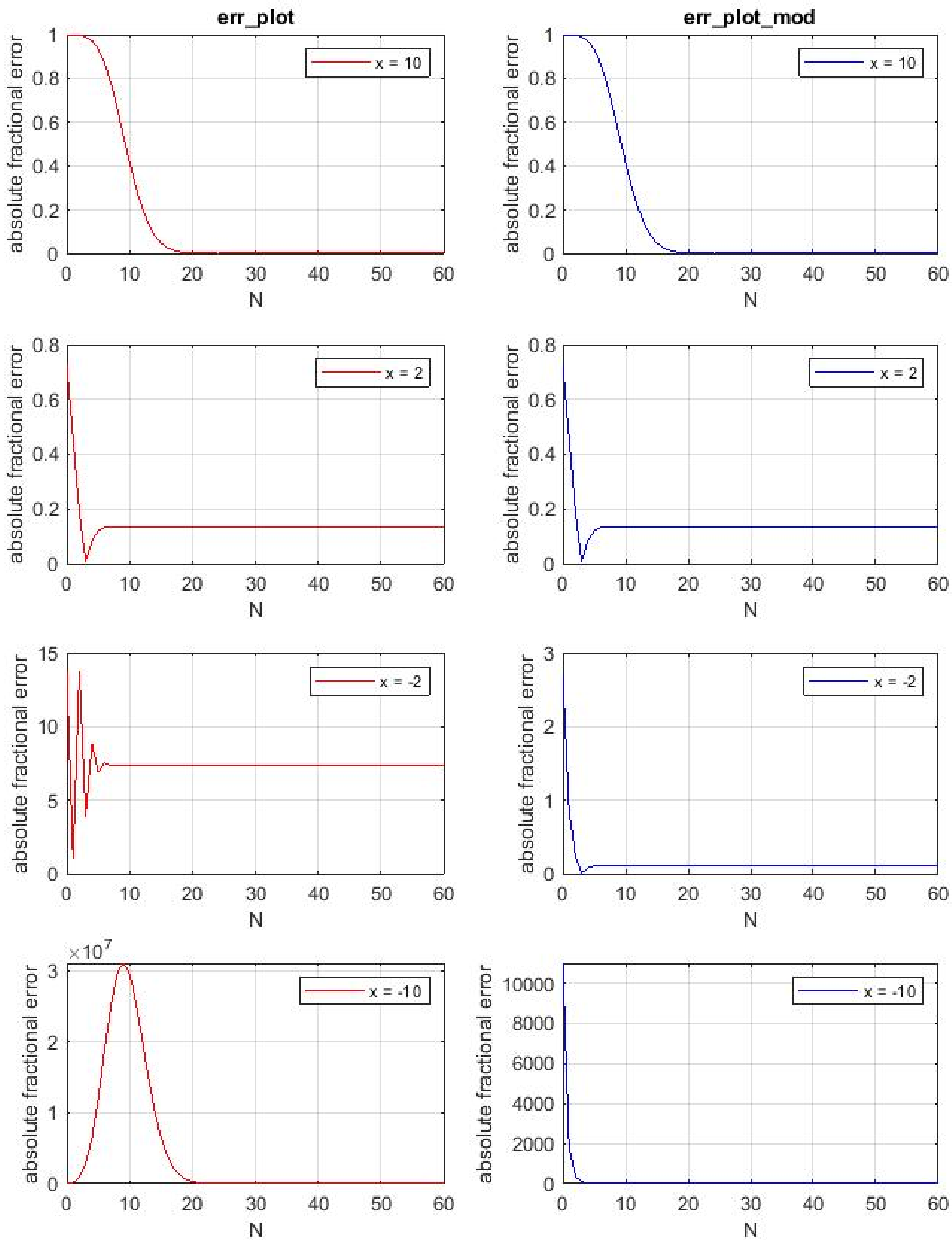


FIG. 2. Absolute fractional error plotted for
$x = [10, 2, -2, 10]$