CS2035 - Assignment 2 - 2019

Out: Monday, January 28th, 2019

In: Sunday, February 17th, 2019 at 11:55pm via Owl

Introduction

This MatLab assignments requires you to write MatLab code using rational functions to approximate e^{-x} for $x \in [0,1]$. A rational function, $f(x)$, is simply the ratio of two polynomials. We are interested in the rational function:

$$f(x) = \frac{1 - \frac{3}{5}x + \frac{3}{20}x^2 - \frac{1}{60}x^3}{1 + \frac{2}{5}x - \frac{1}{20}x^2},$$

which approximates e^{-x} in the interval $x \in [0,1]$. Note that e^0 is 1 and e^{-1} is 0.367879441171442.

In the above equation there are 4 constants which we need to compute only once. As well, `x` is 10,000,000 equally spaced values between 0 and 1. Lastly, we must make sure `y` is not allocated (from say previous runs of your code with `y` being saved in your workspace) by setting it to a single value (0)).

```
% Serialized solution - don't pre-allocate y so that
% it has to by dynamically allocated for each iteration of the for loop
n=10000000;
x=linspace(0,1,n);
% no y pre-allocation using zeros
% pre-compute 4 constants once
c1=3/5;
c2=3/20;
c3=2/5;
c4=1/20;

% De-allocate y from any workspace y from previous runs
% This forces y to be dynamically allocated for each run of your code
% Setting y to a single 0 value destroys y as an array (if it is
% already in the workspace from a previous run)
y=0;
```

A serial MatLab computation of $f(x)$ uses a dynamically allocated `y` array and could be:

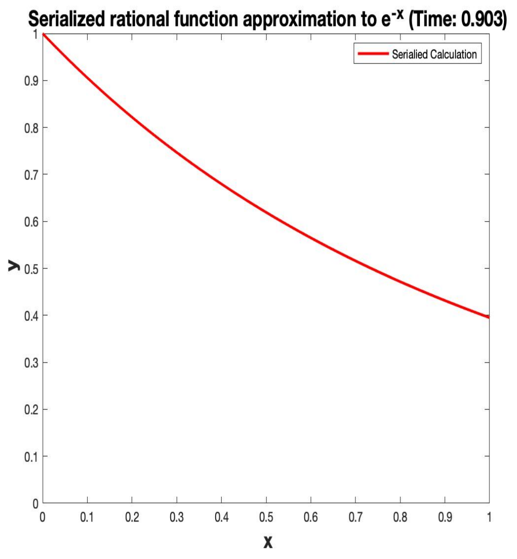
```
start_time=tic;
% could speed up the loop by computing x2=x(k)*x(k) and using that
for k=1:n
  x2=x(k)*x(k);
  y(k) = (1-c1*x(k)+c2*x2-(x(k)/60)*x2)/...
         (1+c3*x(k)+c4*x2);
end % for k=1:n
elapsed_time1=toc(start_time);
fprintf('Computational time for serialized solution: %0.3f\n',elapsed_time1);
figure
plot(x,y,'linewidth',2.0,'color','r') % red thick line
axis([0 1 0 1]);
xlabel('\bf\fontsize{16} x');
ylabel('\bf\fontsize{16} y');
title(['\fontsize{16} Serialized rational function approximation to e^{-x} (Time: ',...
         sprintf('%.3f',elapsed_time1) ')']);
legend('Serialied Calculation');
print serialized_rational_approximation.jpg -djpeg
```

Figure 1a shows the serial plot. Note that the computational time is plotted in the figure's title. The above serialized code will be **not** be compiled by the MatLab as `y` is dynamically allocated. At each iteration, `y` is increased by 1 element.
The JIT (Just In Time) compiler can be invoked by pre-allocating `y`. The rest of the code is the same as for the serial solution. Figure 1b shows the JIT plot and computational time.

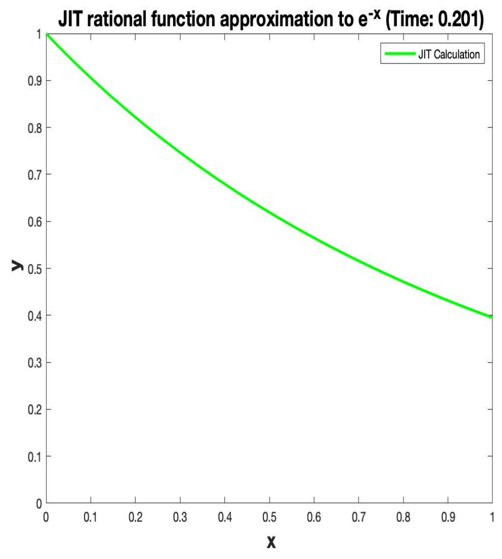
The next and **main** task in this assignment is to performed for this assignment is to vectorize this code and then time that code's execution. Vectorized code does not use loops or array indices (but it does use arrays). Use a single MatLab line of code to do your vectorization. Again, print your timing results in the graph's title. Figure 2a shows the vectorized plot with computational time.

Lastly, we plot and time the use of `exp(x)` (the `y` value here) against the `x`. As expected, the plot in Figure 2b is the same as the other plots.

The computational times for the 4 plots should differ a little from run to run on the same

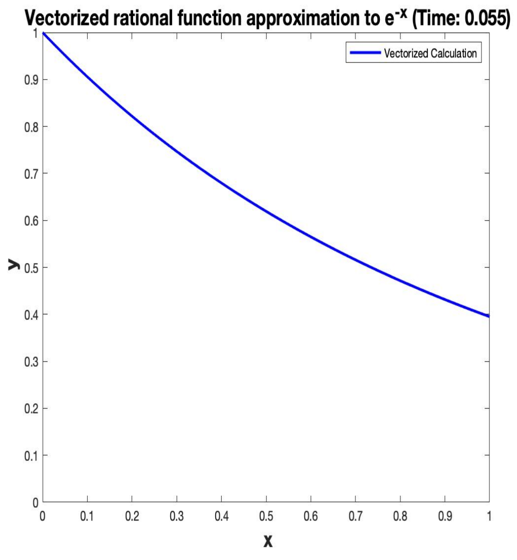


(a)

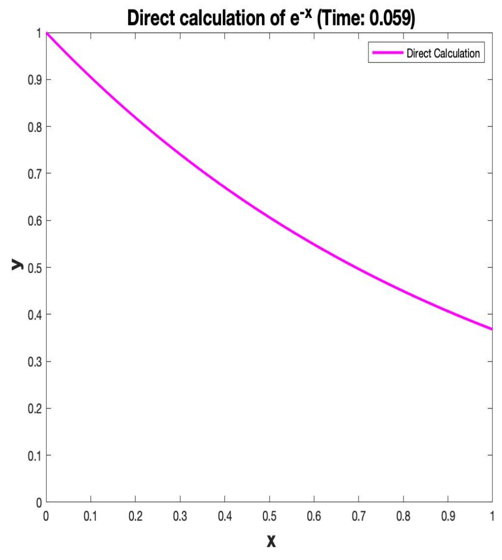


(b)

Figure 1: (a) Serialized and (b) JIT rational function plots of e^{-x} for $x \in [0, 1]$.



(a)



(b)

Figure 2: (a) Vectorized rational function plot of e^{-x} for $x \in [0, 1]$ versus (b) a direct plot of e^{-x} for $x \in [0, 1]$.

machine. On different machines, the time differences may be more significant. Do the times make sense to you?