INFORME LABORATORIO

Nombres: Nicolás Paila
Catherine Gomez
Carreras: Ing. civil Informática
Ing. civil Quimica
Profesor: Erwin Henriquez

Problema 3: (archivo **problema_1.m**)

1) **Considerando la siguiente función y su respectiva expansión en serie de Taylor truncada alrededor de $x_0 = 0$.**

$$f(x) = \frac{x}{(1+4x)^2} \qquad \frac{x}{(1+4x)^2} \approx \sum_{k=0}^{n} (-1)^k 4^k (k+1) x^{k+1}$$

a) **Programe una función que calcule el valor de la serie para $x \in \mathbb{R}$ y $n \in \mathbb{N}$.**

- La rutina programada es el siguiente, el cual se puede encontrar en el archivo script nombrado ***problema_1.m***:

notes.m | untitled5.m | utils.m | problema_1.m | +

```
function [resultado] = problema_1(x, n)
    % Función que calcula una aproximación de f(x)
    % con una serie de Taylor hasta la n-ésima orden
    resultado = 0; % Variable que contiene el resultado
    for k = [0 : 1 : n] % Bucle que realiza hasta n + 1 iteraciones
        k_termino = (-1)^k*(4^k)*(k+1)*x.^(k+1); % k-ésimo término de la sumatoria
        resultado = resultado + k_termino; % Al resultado se le suma cada término
    end
end
```

- **Pruebe su rutina para los siguientes valores:**

  i) $x =$ 0.01, $n =$ 10
  ii) $x =$ -0.23, $n =$ 30
  iii) $x =$ 3, $n =$ 20

- Para probar esta rutina, se debe abrir el archivo script de nombre ***probador_1.m*** el que se ve de la siguiente manera:

```
clc
format long

% Los resultados son mostrados
% con una precisión decimal de 10 dígitos

% i
x = 0.01;
n = 10;

[resultado] = problema_1(x, n); % Función objetivo
fprintf("i.   %5.10f\n", resultado);

% ii
x = -0.23;
n = 30;

[resultado] = problema_1(x, n); % Función objetivo
fprintf("ii.  %5.10f\n", resultado);

% iii
x = 3;
n = 20;

[resultado] = problema_1(x, n); % Función objetivo
fprintf("iii. %5.10f\n", resultado);
```

- Para utilizar este archivo, solo se debe apretar el botón “RUN” y lograrán observar los siguientes resultados:

```
Command Window
i.    0.0092455621
ii.   -26.5066735319
iii.  2237645477248632755322.0000000000
>>
```

Observaciones:

- Los resultados corresponden a la sumatoria de los $(n+1)$* términos de la sucesión de Taylor centrada en $x_0 = 0$ hasta el n-ésimo orden. Lo equivalente a una aproximación de $f(x)$. *$(n+1)$ términos porque se incluye el término para $n = 0$.

b) Grafique la imagen de la serie en un intervalo de la forma $linspace(-a, a, m)$, donde $a > 0$ y $m \in \mathbb{N}$, la función debe tener como entrada a, x, y n. Considere la rutina del ítem anterior.

- La rutina programada es el siguiente, el cual se puede encontrar en el archivo script nombrado ***problema_1b.m***:

notes.m | problema_1b.m

```
function [resultado] = problema_1b(a, m, n)
    % Método que grafica la secuencia de Taylor
    % dada en función de x en un intervalo dado

    x = linspace(-a, a, m); % x es el conjunto de pre-imágenes de la función
    y = problema_1(x, n); % y es el conjunto de imágenes de la función

    plot(x, y, '-b');  % comando que grafica la función

end
```

- **Pruebe su rutina para los siguientes valores:**

  i) $a = 0.2$, $m = 1000$, $n = 20$
  ii) $a = 0.5$, $m = 100$, $n = 50$
  iii) $a = 2$, $m = 100$, $n = 30$

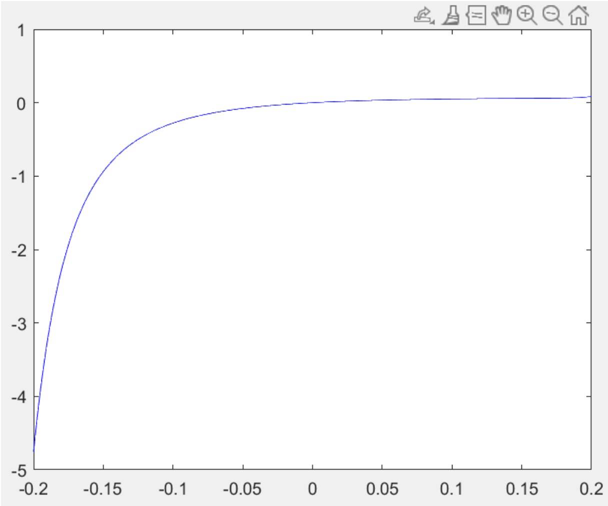
- Para utilizar este rutina, se debe abrir el archivo script ***probador_1b.m*** el cual se ve de la siguiente manera:

untitled5.m | utils.m | probador_1.m | probador_1c.m | probador_1b.m

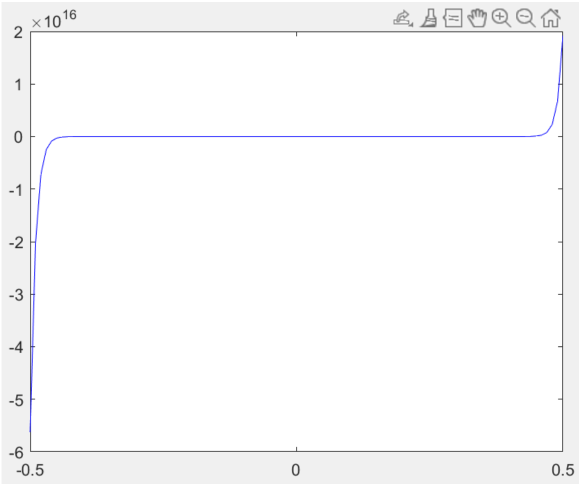
```
clc
format long

% i
a = 0.2; % a es un número real tal que a > 0
m = 1000; % m es un número natural
n = 20; % n es la cantidad de términos de la sumatoria

problema_1b(a, m, n); % Función que grafica

pause(2) % Pausa la ejecución del programa por dos segundos
clf

% ii
a = 0.5; % a es un número real tal que a > 0
m = 100; % m es un número natural
n = 50; % n es la cantidad de términos de la sumatoria

problema_1b(a, m, n); % Función que grafica

pause(2) % Pausa la ejecución del programa por dos segundos
clf

% iii
a = 2; % a es un número real tal que a > 0
m = 100; % m es un número natural
n = 30; % n es la cantidad de términos de la sumatoria

problema_1b(a, m, n); % Función que grafica
```

- Si apretamos el botón “RUN” se logra observar 1 gráfico, que luego de 2 segundos cambia al gráfico 2 y este luego cambia al gráfico 3. Los cuales se pueden ver de forma detallada a continuación:

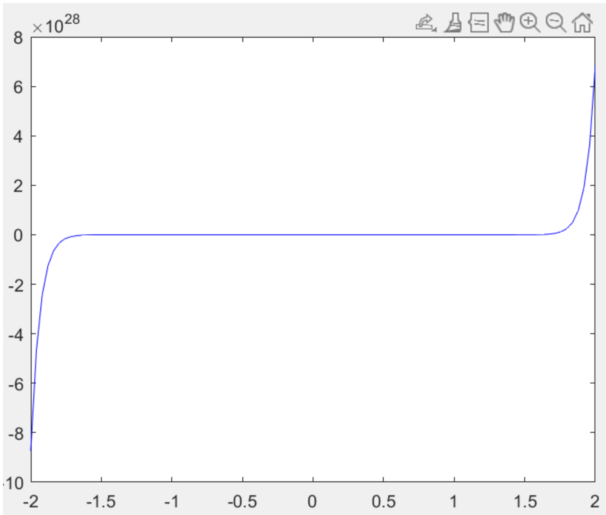
- Gráfico 1



- Gráfico 2



- Gráfico 3



Observaciones (Justificación de los resultados):

Cada término de la sumatoria representa un término de orden k de la sucesión de Taylor, la sumatoria en este contexto representa la suma de todos los términos de la serie de Taylor de la función $f(x)$ hasta el orden n, así mismo, x es la distancia desde el centro de la serie de Taylor (en nuestro caso el centro es $x_0 = 0$).

Luego, cada gráfica representa la sumatoria de la serie de Taylor hasta el n-ésimo orden en función de la distancia x desde el centro de la serie de Taylor. En otras palabras, variando el valor de n las gráficas nos indican el comportamiento de las aproximaciones de $f(x)$ vecindarias al centro de la serie de Taylor. Mientras mayor sea n, mayor será la precisión y exactitud de estas aproximaciones.

c) En un archivo.m programe la función que calcule el error E_M, dado por:

$$E_M = \max_{x \in S} \left| \frac{x}{(1+4x)^2} - \sum_{k=0}^{n} (-1)^k 4^k (k+1) x^{k+1} \right|$$

- donde $\boldsymbol{S} = \{\boldsymbol{x}/\boldsymbol{x} \in \boldsymbol{linspace}(-\boldsymbol{a}, \boldsymbol{a}, \boldsymbol{m})\}$, la función debe tener como parámetros de entrada $\boldsymbol{a}$, $\boldsymbol{m}$ y $\boldsymbol{n}$ y como parámetros de salida el error máximo.

- La rutina programa para este ítem se puede encontrar en el archivo script de nombre ***problema_1c.m,*** el cual se ve de la siguiente manera:

problema_1.m | probador_1b.m | problema_1c.m | probador_1c.m

```
function error_maximo = problema_1c(a, m, n)
    % Función que calcula el error máximo entre la función original
    % y su expansión de Taylor de orden 'n' dado un intervalo 'a'
    S = linspace(-a, a, m); % Conjunto acotado de pre-imágenes de la función
    error_maximo = 0;
    % El algoritmo procesa cada elemento de S tal que si un número es mayor
    % que el máximo guardado este pasa a ser el nuevo número máximo
    % Trata cada termino calculado como un potencial número máximo
    for x = S
        error = abs(x/(1 + 4*x)^2 - problema_1(x, n));
        if error > error_maximo
            error_maximo = error;
        end
    end
end
```

- **Pruebe su rutina para los siguientes valores:**

i) a = 0.2, m = 100, n = 10
ii) a = 0.25, m = 1000, n = 5
iii) a = 1, m = 100, n = 5

- Para probar esta rutina, se debe abrir el archivo script de nombre ***probador_1c.m*** el que se ve de la siguiente manera:

untitled5.m | utils.m | probador_1.m | probador_1c.m | probador_1b.m

```matlab
clc
format long
% Los resultados son mostrados
% con una precisión decimal de 10 dígitos
% i
a = 0.2;
m = 100;
n = 10;

error_maximo = problema_1c(a, m, n); % Función objetivo
fprintf("i.   %5.10f\n", error_maximo);

% ii
a = 0.25;
m = 1000;
n = 5;

error_maximo = problema_1c(a, m, n); % Función objetivo
fprintf("ii.  %5.10f\n", error_maximo);

% iii
a = 1;
m = 100;
n = 5;

error_maximo = problema_1c(a, m, n); % Función objetivo
fprintf("iii. %5.10f\n", error_maximo);
```

- Para utilizar esta rutina se debe abrir el archivo script ***probador_1c.m*** y apretar ‘’RUN’’ y aparecerán los siguientes resultados:

Command Window

```
i.   1.3743895347
ii.    Inf
iii. 7736.8888888889
fx >>
```

Observaciones:

- El segundo ítem nos dá un valor infinito, lo que nos indica que el error no se acota en al menos un punto en el intervalo $[-0.25, 0.25]$ para $n = 5$.

$$E_M = \max_{x \in S} \left| \frac{x}{(1+4x)^2} - \sum_{k=0}^{n} (-1)^k 4^k (k+1) x^{k+1} \right|$$

- Como se alcanza a apreciar, la expresión en el argumento de $máx$ es el valor absoluto de una función multivariable (con variables independientes x y n) que representa un continuo de errores absolutos entre valores de la función $f(x)$, con preimágenes x del conjunto S, y su aproximaciones en secuencia de Taylor (centrada en x_0) hasta el orden n-ésimo.

2) Su misión consiste en escribir un fichero en Matlab que contenga una rutina que calcule una aproximación del número e usando términos de la sucesión a_n. El programa debe devolver un vector de aproximaciones de e, un vector de errores absolutos y otro de errores relativos correspondiente a cada n. Trabaje con $n \in N$, $1 \leq n \leq 50$. Utilice como valor real de e aquel obtenido en MatLab por instrucción exp(1). Use formato de alta precisión decimal para visualizar los resultados.

- La rutina elaborada se puede encontrar en el archivo script ***problema_2.m*** y se ve de la siguiente manera:

```
clc
format long

rango_n = 1 : 1 : 50; % Conjunto de números naturales n tal que 1 <= n <= 50

% Preasigna el tamaño de los vectores para mayor eficiencia
aproximaciones_e = zeros(1,50);
errores_absolutos = zeros(1,50);
errores_relativos = zeros(1,50);

for n = rango_n % Bucle que itera sobre cada elemento del conjunto
    e_aproximado = (1 + 1/n)^n; % Variable que contiene el valor aproximado
    aproximaciones_e(n) = e_aproximado; % Vector con los valores aproximados
    errores_absolutos(n) = ...
        abs(exp(1) - e_aproximado); % Vector con los errores absolutos
    errores_relativos(n) = ...
        abs(exp(1) - e_aproximado)/abs(exp(1)); % Vector con los errores relativos
end

% Los vectores resultantes se muestran en su
% forma transpuesta para una mejor visualización
disp("Vector de aproximaciones de e:")
disp(aproximaciones_e')
disp("Vector de errores absolutos:")
disp(errores_absolutos')
disp("Vector de errores relativos:")
disp(errores_relativos')

% Además, lo tres vectores también son representados en formato
% de columnas adyacentes (en formato de visualización short)
% Para su visualización, favor de hacer click en el nombre de la variable
% 'matriz_resultados' en el Workspace
matriz_resultados = [aproximaciones_e; errores_absolutos; errores_relativos]';
```

- Para utilizar esta rutina solo se debe apretar el botón ''RUN'' y aparecerán los siguientes resultados en consola, 3 vectores de 50 números naturales cada uno, pero para efectos de este informe solo se mostrarán los primeros elementos de cada vector, además se puede apreciar que están en formato de alta precisión.

```
Vector de aproximaciones de e:
   2.000000000000000
   2.250000000000000
   2.370370370370370
   2.441406250000000
   2.488319999999999
   2.521626371742113
   2.546499697040712
   2.565784513950348
   2.581174791713198
```

```
Vector de errores absolutos:
   0.718281828459045
   0.468281828459045
   0.347911458088675
   0.276875578459045
   0.229961828459046
   0.196655456716932
   0.171782131418333
   0.152497314508697
   0.137107036745847
   0.124539368359043
```

```
Vector de errores relativos:
   0.264241117657115
   0.172271257364255
   0.127989472778804
   0.101856833077533
   0.084598228944277
   0.072345499520340
   0.063195114509416
   0.056100626841605
   0.050438860058735
```

Observaciones:

- Como bien dicho en el enunciado, a medida que crece n la tasa de crecimiento de las aproximaciones de e van disminuyendo, de igual manera pasa algo similar con los errores absolutos y errores relativos.