Instituto Tecnológico de Costa Rica
Ingeniería en Computadores
Puntaje Total: 100 puntos

CE-3102: Análisis Numéricos para Ingeniería
Semestre: I - 2022
Valor Porcentual: 15 %

Tarea 1

Instrucciones generales

- La tarea se realiza en grupos de máximo 4 personas. Cada grupo debe escribir el nombre de los integrantes del grupo en la siguiente dirección electrónica:

  `https://docs.google.com/spreadsheets/d/1DvlRnOrb5p8mvOVOFB4mr54QG52j_C4o`

- Todos los archivos de esta tarea se encuentran en la carpeta de One Drive del curso.
- Los archivos computacionales implementados en GNU Octave, Pyhton y C++ deben estar correctamente comentados. Por cada archivo que no este documentado correctamente, se restaran 5 puntos de la nota final. **Si alguna función o archivo computacional está incompleto o genera error al momento de compilar, entonces pierde el 75% del puntaje de la pregunta asignada.**

Parte 1: Paquete Computacional `FunTras` en GNU Octave

Descripción General

Definición: Una función trascendente es una función que no satisface una ecuación polinomial. Ejemplo de funciones trascendentes son e^x, $\ln(x)$, $\sin(x)$ y $\frac{1}{x}$.

Esta parte de la tarea consiste en desarrollar un paquete computacional en GNU Octave que permita aproximar el valor numérico de un conjunto de funciones trascendentes de variable real utilizando **únicamente** las operaciones de suma (`+`), resta (`-`), multiplicación (`*`) y potencia de exponente entero positivo (`^`). **No pueden usar la división (`/`).**

Preguntas

1. [**Valor: 25 puntos**] Implemente computacionalmente en GNU Octave las funciones transcendentes que se encuentran en la siguiente tabla.

Función $f(x)$	Comando en GNU Octave	Función $f(x)$	Comando en GNU Octave
x^{-1}	`div_t(x)`	e^x	`exp_t(x)`
$\sin(x)$	`sin_t(x)`	$\cos(x)$	`cos_t(x)`
$\tan(x)$	`tan_t(x)`	$\ln(x)$	`ln_t(x)`
$\log_a(x)$	`log_t(x,a)`	a^x	`power_t(x,a)`
$\sinh(x)$	`sinh_t(x)`	$\cosh(x)$	`cosh_t(x)`
$\tanh(x)$	`tanh_t(x)`	$\sqrt{x}$	`sqrt_t(x)`
$\sqrt[a]{x}$	`root_t(x,a)`	$\sin^{-1}(x)$	`asin_t(x)`
$\tan^{-1}(x)$	`atan_t(x)`		

   - Para realizar dicha implementación, deben leer el documento `fun_tras.pdf` que se encuentra en la carpeta de One Drive del curso. Este documento contiene los métodos iterativos que deben implementar para aproximar las funciones que se encuentran en la tabla anterior.
   - Cada función debe estar implementada en un archivo por aparte, y el nombre del archivo debe ser el mismo que el nombre de la función.

- Para su implementación, cada método iterativo debe usar una tolerancia de 10^{-8}, además de una cantidad máxima de 2500 iteraciones.
- Algunas de las funciones que se encuentran en la tabla no están en el documento `fun_tras.pdf`. Para la implementación de estas funciones, utilice propiedades matemáticas para re-escribir dichas funciones en términos de las funciones que se encuentran en el documento `fun_tras.pdf` (por ejemplo, $\log_a(x) = \ln(x)/\ln(a)$).
- Cada una de las funciones debe verificar su dominio máximo. En el caso de que el parámetro inicial no se encuentra en el dominio, la función debe enviar un mensaje de error (por ejemplo, la función `sqrt_t(x)` solo debe aceptar parámetros mayores o iguales a 0).
- Algunas de las funciones que se encuentran en el documento `fun_tras.pdf` utilizan la función factorial(`factorial`). Para eso, pueden usar la función factorial que tiene implementada GNU Octave.
- Utilizando las funciones implementadas, desarrolle un *script* con nombre `test_funtras.m` que realice la operación

$$\frac{\sqrt[3]{\sin\left(\frac{3}{7}\right) + \ln(2)}}{\sinh(\sqrt{2})} + \tan^{-1}(e^{-1}).$$

2. [**Valor: 5 puntos**] Para implementar la función $f(x) = \cos^{-1}(x)$, se utiliza la siguiente propiedad:

$$\cos^{-1}(x) = \frac{\pi}{2} - \sin^{-1}(x).$$

   - Investigue como se puede aproximar el valor de π utilizando un método itertaivo, e implemente la función `pi_t()` utilizando dicho método iterativo. Utilice una tolerancia de 10^{-8}, además de una cantidad máxima de 2500 iteraciones.
   - Utilizando la función `pi_t()` y `asin_t(x)`, implemente la función `acos_t(x)`, la cual aproxima el valor de $\cos^{-1}(x)$. Utilice una tolerancia de 10^{-8}, además de una cantidad máxima de 2500 iteraciones.

3. [**Valor: 10 puntos**] Utilizando las funciones implementadas en las Preguntas 1 y 2, desarrolle un paquete computacional en GNU Octave, basándose en los siguientes indicaciones:

   - El nombre del paquete debe ser `FunTras`.
   - Para crear el paquete computacional, utilice la información que se encuentra en la siguiente dirección electrónica:

     `https://octave.org/doc/v4.2.2/Creating-Packages.html`.

     También pueden usar de referencia el documento `paquetes_octave.pdf` que se encuentra en la carpeta de One Drive del curso.
   - Para este paquete computacional, se debe elaborar un manual de usuario. El manual de usuario debe contener lo siguiente:
     - Portada con nombre del paquete, nombre del TEC, nombre del curso y el nombre de los miembros del grupo.
     - Tabla de contenidos
     - Una sección donde se explique en que consiste el paquete computacional.
     - Una sección que explique como instalar el paquete computacional y que requisitos se necesitan para su uso.
     - Una sección donde explique el uso de las funciones implementadas, con su formulación matemática (método iterativo) y ejemplos ilustrativos. Esta sección debe contener todo lo necesario para saber utilizar las funciones implementadas.
   - El nombre del manual deben ser `manual_FunTras.pdf`. La estructura del manual se puede basar en el manual desarrollado para el paquete NumPy de Python, el cual encuentra en la carpeta de One Drive del curso, con el nombre `userguide_numpy.pdf`. **Se tomará en cuenta la apariencia, aspecto y calidad del manual en el puntaje de esta pregunta.**

Parte 2: Generalización del Método de Newton-Rapshon

Descripción General

Esta parte de la tarea consiste en el estudio e implementación en Python de varios métodos iterativos que representan una variación del método de Newton-Raphson para aproximar una solución de una ecuación no lineal. Al final, se utilizarán estos métodos para resolver un problema del área de la ingeniería.

Preguntas

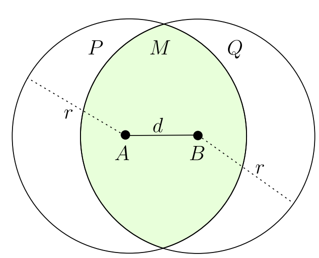
1. [**Valor: 25 puntos**] En el artículo científico *One-point Newton-type iterative methods: A unified point of view*, desarrollado por los investigadores A. Cordero, C. Jordán y J.R. Torregrosa, se desarrolla una generalización del método de Newton-Rapshon para aproximar una solución de la ecuación $f(x) = 0$. Esta generalización se representan en las ecuaciones (4) y (12) del artículo. Las fórmulas iterativas en (4) y (12) utiliza funciones de peso $H(u(x_k))$ y $G(w(x_k))$, respetivamente. Los posibles valores de dichas funciones se encuentran en las Tablas 1 y 2 del artículo. El artículo se encuentra en el la carpeta de One Drive del curso en el archivo `one_newton.pdf`.

   (a) Implemente computacionalmente en Python las ecuaciones (4) y (12) del artículo, escogiendo solo 2 funciones de peso de cada una de las Tablas 1 y 2.

   - El nombre de las funciones en Python debe seguir el siguiente formato: `newton_H_m1` y `newton_H_m2` para los métodos basados en la ecuación (4) y la Tabla 1; `newton_G_m1` y `newton_G_m2` para los métodos basados en la ecuación (12) y la Tabla 2.
   - Los parámetros de entrada de las funciones son los siguientes: un *string* `fun` que representan a la función f, un valor inicial `x0`, una tolerancia `tol` > 0 e iteraciones máximas `iterMax`.
   - Los parámetros de salida son los siguientes: `xk` que representa la aproximación a la solución de la ecuación $f(x) = 0$, `k` que representa el número de iteraciones realizadas y `error` que representa al error $|f(x_k)|$.
   - Cada función implementada debe realizar el cálculo de las derivadas. Para eso utilicen el paquete `SymPy`.
   - Los criterios de parada para cada uno de los métodos implementados es que se cumpla $|f(x_k)| < tol$ o $k > iterMax$.
   - Algunos de los métodos del artículo tienen parámetros extra (por ejemplo, β, λ, A). Cada grupo debe seleccionar un valor particular de estos parámetros para utilizarlos en las funciones implementadas.
   - Todas las funciones deben estar implementadas en un archivo con nombre `metodos_p2.py`.

2. [**Valor: 5 puntos**]. En la Sección 5 del artículo científico, se encuentra un conjunto de funciones para probar la eficiencia de los métodos implementados. Entre las funciones f_1, f_2, f_3, f_4, f_5 que se encuentran en esta sección, seleccione una de ellas para probar todos los métodos implementados en la Pregunta 1. Dicha prueba se debe guardar en un archivo con nombre `prueba_metodo_p2.py`. Al ejecutar cada uno de los métodos, debe aparecer un mensaje indicando: (1) la función a la cual se le desea calcular el cero, (2) el nombre del método, (3) la aproximación del cero y (4) el error. Utilice una tolerancia de 10^{-5} y una cantidad máxima de 500 iteraciones.

3. [**Valor: 10 puntos**]. En el artículo científico *Estimating distances via received signal strength and connectivity in wireless sensor networks*, los autores explican una técnica para estimar la distancia en redes de sensores inalámbricos. En general, el problema se puede formular de la siguiente manera: Determinar la distancia d entre dos sensores A y B, el cuál se muestra en la siguiente figura:



En la figura anterior, r representa el radio de cobertura de cada sensor (nodo). Se asume que cada nodo tiene el mismo radio de cobertura. P, Q, M representan los vecindarios (regiones) de interés.

El valor d que calcula la distancia entre dos sensores se obtiene de encontrar la intersección con el eje d de la función

$$F(d) = \frac{\log_{10}(x_1/d)}{\sigma_R^2 \ln(10)} + \frac{d(x_2 - d)}{\sigma_c^2}, \tag{1}$$

donde:

- $\sigma_R^2 = {\sigma_{dB}}^2/(10\alpha)^2$
- $\sigma_c^2 = \frac{g^2(d)}{2\lambda k^2}\left(\frac{1}{g(d)} + \frac{1}{S}\right)$
- $k = 10\alpha/\ln(10)$
- $S = \pi r^2$
- $g(d) = \frac{2S}{\pi}\arccos\left(\frac{d}{2r}\right) - d\sqrt{r^2 - \frac{d^2}{4}}$
- $\alpha, \lambda, r, \sigma_{dB}, x_1, x_2$ son parámetros conocidos.

- Utilizando los valores $r = 10$, $\alpha = 4$, $\sigma_{dB} = 4$, $\lambda = 1$, $x_1 = 7$ y $x_2 = 6$, aproxime en Python el valor d que calcula la distancia entre los sensores usando uno de los métodos implementados en la Pregunta 1. Para esto, modifique dicho método para no tener que ingresar la función `func` como un *string*, sino que acepte una función en formado simbólico.
- Utilice una tolerancia de 10^{-5} y una cantidad máxima de 100 iteraciones.
- El valor inicial x_0 debe definirlo cada grupo.
- La implementación computacional en Python se debe realizar en un archivo con nombre `aplicacion_p2.py`.
- Al ejecutar el método, debe aparecer un mensaje indicando la aproximación del cero y el error calculado.

Parte 3: Implementación en C++

Descripción General

Esta parte de la tarea consiste en implementar computacionalmente en C++ algunas de las preguntas desarrolladas en las Partes 1 y 2.

Preguntas

1. [**Valor: 20 puntos**] Cada grupo debe escoger entre la **Pregunta 1 de la Parte 1** o las **Preguntas 1 y 2 de la Parte 2** de esta tarea para realizar la implementación en C++.

   (a) Si el grupo selecciona la **Pregunta 1 de la Parte 1**, entonces deben implementar todas las funciones en un solo archivo con nombre `parte_1.cpp`. En el `main` de este archivo, debe estar la operación numérica que se presenta al final de la Pregunta 1.

   (b) Si el grupo selecciona la **Pregunta 1 de la Parte 2**, entonces deben implementar todas las funciones en un solo archivo con nombre `parte_2.cpp`. En este caso, no deben escribir la función como un *string*, sino que puede definir la función y sus derivadas como funciones auxiliares. Por ejemplo, para definir la función $f(x) = x^2 + 3x + 1$ y su derivada $f'(x) = 2x + 3$, se puede definir en C++ de la siguiente forma

```
float fn(float x)
{
     return  pow(x,2)+(3*x)+1 ;
}
float de(float x)
{
     return  2*x + 3 ;
}
```

   En el `main` de este archivo, debe probar los métodos con la función seleccionada en la Pregunta 2.

Información de la Entrega

- **Fecha y hora límite**: Domingo 20 de Marzo del 2022 a las 11:59 pm.
- Los documentos deben estar en una carpeta principal con nombre `Tarea 1 - Grupo #`, donde `#` es el número de cada grupo. Dentro de esta carpeta debe existir tres carpetas con nombres `Parte 1`, `Parte 2` y `Parte 3` . En cada una de estas carpetas estarán todos los archivos necesarios para el desarrollo de las preguntas mencionadas anteriormente.
- Deben enviar la carpeta `Tarea 1 - Grupo #` en formato `zip` al correo `jusoto@tec.ac.cr`, con el encabezado `Entrega Tarea 1 - Grupo # - ANPI`. En el cuerpo del correo deben indicar el nombre completo de los miembros del grupo.
- **OBSERVACIÓN IMPORTANTE: Las entregas tardías se penalizarán con una reducción del 25% de la nota máxima por día de atraso. A las tareas que excedan el plazo de entrega en 3 días o más después de la fecha límite, se les asignará la nota de 0**.

Defensa

- Cada grupo debe defender esta tarea frente al profesor. Para eso deben seleccionar un horario de la siguiente dirección electrónica:

  `https://docs.google.com/spreadsheets/d/1DvlRnOrb5p8mv0V0FB4mr54QG52j_C4o`

- Deben escribir el número de las posibles horas de defensa.
- Todos los miembros del grupo deben estar presentes para defender cada una de las preguntas. Si un estudiante no esta presente, entonces el estudiante perderá 35 puntos de la nota final.