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

CE-3102: Análisis Numéricos para Ingeniería
Semestre: I - 2023
Valor Porcentual: 20 %

Tareas 3 y 4

Instrucciones generales

- La nota de esta evaluación será asignada tanto para el rubro de la tarea 3 como para el rubro de la tarea 4.
- 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/1s2Q9ODQnhl0CNtUwTZJHvn8gz9dCvQOV`

  El número del grupo está indicado en la primera columna del documento.
- Todos los archivos de esta tarea se encuentran en la carpeta de *One Drive* del curso.
- Los archivos computacionales implementados 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.**
- Los archivos que dan solución a la tarea deben estar en una carpeta principal con nombre `Tarea 3 y 4 - Grupo #`, donde # es el número de cada grupo. Dentro de esta carpeta debe existir dos carpetas con nombres `Parte 1` y `Parte 2`. En cada una de estas carpetas estarán todos los archivos necesarios para el desarrollo de las preguntas mencionadas anteriormente.
- La solución de la tarea que se encuentra en la carpeta `Tarea 3 y 4 - Grupo #` debe comprimirse en un archivo `.zip` y subirlo al formulario que se encuentra en el siguiente enlace:

  `https://forms.gle/nBenrrLuyCarTWku6`

  Observación: Se necesita tener una cuenta de **gmail** para llenar el formulario.
- **Fecha y hora máxima de entrega: Domingo 18 de Junio del 2023, a las 11:59 pm**
- Las entregas tardías se penalizarán con una reducción de la nota obtenida con un 10% por cada hora de atraso. A las tareas que excedan el plazo de entrega en 10 horas o más después de la hora límite, se les asignará la nota de 0.

Parte I: Paquete Computacional *NumInt* en GNU Octave

Descripción General

- Esta parte de la tarea consiste en desarrollar un paquete computacional en GNU Octave que permita aproximar el valor numérico de la integra definida

$$\int_a^b f(x)\,dx, \tag{1}$$

  donde $f : A \to \mathbb{R}$, $A \subseteq \mathbb{R}$, es una función continua en A y $[a, b] \in A$.

Preguntas

1. [**Valor: 40 puntos**] Implemente computacionalmente en GNU Octave los siguientes métodos para aproximar el valor de la integral definida expresada en (1).

    - Regla del Trapecio Compuesta:
        - Nombre de la función: `trapecio_compuesto`.
        - Parámetros iniciales: `f` = función $f(x)$; `a, b` = intervalo $[a, b]$; `N` = número de puntos en los que se divide el intervalo $[a, b]$.
        - Parámetro de Salida: `I` = Aproximación numérica de (1).
    - Regla de Simpson Compuesta:
        - Nombre de la función: `simpson_compuesto`.
        - Parámetros iniciales: `f` = función $f(x)$; `a, b` = intervalo $[a, b]$; `N` = número de puntos en los que se divide el intervalo $[a, b]$.
        - Parámetro de Salida: `I` = Aproximación numérica de (1).
    - Cuadratura Gaussiana Compuesta:
        - Nombre de la función: `gaussiana_compuesta`.
        - Parámetros iniciales: `f` = función $f(x)$; `a, b` = intervalo $[a, b]$; `M` = orden de la cuadratura (Considerar un orden máximo de 10); `N` = número de puntos en los que se divide el intervalo $[a, b]$.
        - Parámetro de Salida: `I` = Aproximación numérica de (1).
    - Regla del Trapecio Compuesta Iterativa:
        - Nombre de la función: `trapecio_compuesto_iterativa`.
        - Parámetros iniciales: `f` = función $f(x)$; `a, b` = intervalo $[a, b]$; $tol > 0$ que es la tolerancia; `iterMax` = iteraciones máximas.
        - Parámetro de Salida: `I` = Aproximación numérica de (1).
    - Regla de Simpson Compuesta Iterativa:
        - Nombre de la función: `simpson_compuesto`.
        - Parámetros iniciales: `f` = función $f(x)$; `a, b` = intervalo $[a, b]$; $tol > 0$ que es la tolerancia; `iterMax` = iteraciones máximas.
        - Parámetro de Salida: `I` = Aproximación numérica de (1).
    - Cuadratura Gaussiana Compuesta iterativa:
        - Nombre de la función: `gaussiana_compuesta_iterativa`.
        - Parámetros iniciales: `f` = función $f(x)$; `a, b` = intervalo $[a, b]$; `M` = orden de la cuadratura (Considerar un orden máximo de 10); $tol > 0$ que es la tolerancia; `iterMax` = iteraciones máximas.

    Observaciones:

    - Cada función implementada en GNU Octave debe tener su propia ayuda. Esta ayuda debe indicar en que consiste la función, cuales son los parámetros iniciales y cuales con los parámetros de salida.
    - Cada método debe estar desarrollado como una función, y cada función debe estar en un archivo por aparte. El nombre del archivo debe ser el mismo que el nombre de la función.
    - Crear un archivo `ejecutable.m` el cual pruebe numéricamente todas las funciones implementadas para aproximar el valor de la integral
    $$\int_{0.1}^{0.9} \log_x(\arcsin(x))dx,$$
    utilizando los valores $N = 20$, $tol = 10^{-6}$, `iterMax` $= 2500$, cuando corresponda.

2. En el libro *Numerical Analysis* de R. Burden y J. Faires, Novena Edición, en la sección 4.5, página 213, es explica el **método de Romberg** para aproximar el valor de la integral definida expresada en (1).

   - [**Valor: 5 puntos**] En un documento con nombre `parte1_p2.pdf`, realice una breve explicación sobre el método Romberg, mostrando su formulación matemática, además de presentar un ejemplo numérico. La formulación matemática debe indicar los valores iniciales y el valor de salida.
   - [**Valor: 10 puntos**] Implemente computacionalmente en GNU Octave el método de Romberg, como una función con el nombre `romberg`. Este método debe tener su propia ayuda. Esta ayuda debe indicar en que consiste la función, cuales son los parámetros iniciales y cuales con los parámetros de salida. El nombre del archivo debe ser el mismo que el nombre de la función.

3. [**Valor: 15 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 `NumInt`.
   - 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 y ejemplos ilustrativos. Esta sección debe contener todo lo necesario para saber utilizar las funciones implementadas.

     El nombre del manual deben ser `manual_NumInt.pdf`. La estructura del manual se puede basar en el manual desarrollado para el paquete NumPy de Python, el cual se 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: Ecuación Diferencial Ordinaria de Segundo Orden

Descripción General

Esta tarea consiste en resolver el siguiente problema, utilizando el lenguaje de programación Python.

Problema A: Sea $y = y(x)$ una función continua en un intervalo $[a, b]$. Considere la ecuación diferencial

$$\begin{cases} y'' = p(x)y' + q(x)y + f(x) \\ \\ y(a) = y_0, \;\; y(b) = y_n \end{cases}$$

donde $p(x), q(x), f(x)$ son funciones continuas de variable real. Definimos $x_0 = a$, $x_n = b$, y un conjunto soporte $\mathcal{S} = \{x_0, x_1, ..., x_n\}$, tal que $h = x_{i+1} - x_i$, para todo $i = 0, 1, ..., n-1$. El problema consiste en calcular un conjunto de puntos $\{(x_0, y_0),\ (x_1, y_1),\ ....,\ (x_n, y_n)\}$ que aproximen numéricamente la función y en el intervalo $[a, b]$.

Preguntas

1. [**Valor: 30 puntos**] Implemente computacionalmente en Python el método de diferencias finitas, explicado en el documento `diferencias_finitas.pdf`, que da una solución numérica al **Problema A** . Para eso, elabore una función con nombre `edo2`, cuyos parámetros iniciales son las funciones p, q y f, el tamaño de paso h, los valores a, b del intervalo y los valores iniciales y_0, y_n. Los parámetros de salida son los vectores $x = [x_0, x_1, ..., x_n]^T$ y $y = [y_0, y_1, ..., y_n]^T$.

   Nota: La función `edo2` necesita resolver un sistema de ecuaciones lineal cuya matriz de coeficientes es una matriz tridiagonal. Para resolver dicho sistema, utilice el comando `numpy.linalg.solve`.

   Luego, implemente computacionalmente un *script* para aproximar la solución del problema

$$\begin{cases} y'' = -\dfrac{1}{x}y' + \left(\dfrac{1}{4x^2} - 1\right) y \\ \\ y(1) = 1, \;\; y(6) = 0 \end{cases} \tag{2}$$

   Para esto, debe generar una animación en la cual debe aparecer cada 2 segundo una nueva gráfica que representa una aproximación de la solución a la ecuación diferencial (2) con diferentes valores de h. Para eso, utilicen los valores $h = 10^{-i}$, donde $i = 1, 2, ..., 10$. Adicionalmente, al inicio de la animación debe aparecer la solución exacta del problema

$$y(x) = \frac{\sin(6 - x)}{\sin(5)\sqrt{x}}.$$

   La animación debe indicar una leyenda para cada gráfica. Un ejemplo de la animación solicitada se encuentra en el video `ejemplo_animacion.mp4` (El video no corresponde a este ejercicio). Todo lo anterior debe estar implementado en un solo archivo con nombre `pregunta2.py`.

   Sugerencia: En Python, utilice el comando `time.sleep` del modulo `time`.