



Universidad Francisco de Paula Santander

Ocaña - Colombia

Vigilada Mineducación

www.ufpso.edu.co

Desarrollo de un Software para la Solución Matemático de la Descomposición de la Forma LU de una Matriz A 3x3 en Python

Juan David Durán Garzón

Cod. 192104

Entregado a:

Orlando Quintero Álvarez

Facultad de Ingenierías, Universidad Francisco de Paula Santander Ocaña

Ingeniería de Sistemas

193504A: Análisis Numérico

21 de Abril de 2024

Desarrollo de un Software para la Solución Matemático de la Descomposición de la Forma LU de una Matriz A 3x3 en Python

Juan David Durán Garzón

Cod. 192104

Entregado a:

Orlando Quintero Álvarez

Facultad de Ingenierías, Universidad Francisco de Paula Santander Ocaña

Ingeniería de Sistemas

193504A: Análisis Numérico

21 de Abril de 2024

Tabla de Contenido

Desarrollo de un Software para la Solución Matemático de la Descomposición de la	
Forma LU de una Matriz A 3x3 en Python	2
Desarrollo de un Software para la Solución Matemático de la Descomposición de la	
Forma LU de una Matriz A 3x3 en Python	3
Tabla de Ilustraciones.....	5
Introducción.....	6
Objetivo General.....	7
Objetivos Específicos.....	7
Marco Teórico.....	8
Justificación	9
Metodología.....	10
Diagrama de Flujo.....	11
Pseudocódigo	12
Pasos Seguidos en el Desarrollo del Software	15
Contextualización	15
Requerimientos	16
Análisis	17
Diseño.....	23
Prueba.....	24
Detalles del Procedimiento con la Metodología Aplicada	25
Resultados.....	26

Conclusiones.....	27
Enlace Código e Imágenes Finales	28
Bibliografía	30

Tabla de Ilustraciones

Ilustración 1. Diagrama de Flujo	11
Ilustración 2. Pseudocódigo	12
Ilustración 3. Código Fuente	13
Ilustración 4. Código Fuente	14
Ilustración 5. Implementación	17
Ilustración 6. Diseño	23
Ilustración 7. Prueba 1	24
Ilustración 8. Prueba 2	24

Introducción

La capacidad de realizar cálculos aritméticos de manera rápida y precisa es fundamental en diversas áreas, desde las finanzas hasta la ingeniería. El objetivo de este proyecto es desarrollar una calculadora de escritorio intuitiva y funcional que facilite la solución de sistemas de ecuaciones lineales de 3×3 . La solución propuesta es una aplicación de escritorio creada en Python usando como Custom Tkinter, que brinda una interfaz familiar y amigable para el usuario.

Esta calculadora está diseñada para ofrecer una experiencia de cálculo eficiente, precisa y sin complicaciones, al tiempo que incorpora características adicionales como la evaluación de expresiones, el formato integrado de los números definido por su cantidad de decimales. A continuación, se detallarán los requisitos, el diseño de la interfaz de usuario, la implementación de la lógica de cálculo y las pruebas realizadas durante el desarrollo de esta calculadora de escritorio.

Objetivo General

- Desarrollar una Calculadora en Python que Permita el desarrollo de una calculadora para la Solución de Ecuaciones 3x3 a través de la descomposición de una matriz de la Forma LU

Objetivos Específicos

- Investigar y comprender el método utilizado para la Solución y realización de la Descomposición de la Matriz A y el Encontrar Sus Valores Pertinentes.
- Implementar un Algoritmo en Python que realice los procesos de manera Grafica mediante una UI para obtener los Valores de las Matriz U.
- Implementar un Algoritmo en Python que realice los procesos de manera Grafica mediante una UI para obtener los Valores de las Matriz Z.
- Implementar un Algoritmo en Python que realice los procesos de manera Grafica mediante una UI para obtener los Valores de las Matriz X.
- Implementar un Algoritmo en Python que realice los procesos de manera Grafica mediante una UI para obtener los Valores de las Matriz B.
- Implementar un Algoritmo en Python que realice los procesos de manera Grafica mediante una UI para obtener los Valores de las Matriz L.
- Implementar un Algoritmo en Python que realice los procesos de manera Grafica mediante una UI para obtener los Valores de las Matriz A según sea el caso.
- Validar la Entrada del Usuario para comprobar que la Matriz sea Valida en este caso una Matriz 3x3.
- Probar la calculadora con distintos casos de Prueba para verificar la Precisión (4 Cifras) y la Robustez del Código.
- Documentar Adecuadamente el Código, incluyendo esto comentarios explicativos y documentación de las respectivas funciones.
- Proporcionar ejemplos de uso de la Aplicación y una guía de usuario para facilitar la comprensión y el uso de la calculadora.

Marco Teórico

La descomposición LU es una técnica fundamental en álgebra lineal que implica factorizar una matriz en el producto de dos matrices triangulares: una matriz triangular inferior (L) y una matriz triangular superior (U). Esta descomposición se representa matemáticamente como ($A = LU$), donde (A) es la matriz original que se desea descomponer.

Existen varios métodos para realizar la descomposición LU, siendo dos de los más comunes el método de Doolittle y el método de Crout. En el método de Doolittle, la matriz (A) se descompone en (L) y (U) sin el uso de la eliminación gaussiana. Este método implica calcular los elementos de (L) y (U) utilizando relaciones algebraicas entre los elementos de (A) y las matrices (L) y (U). Para implementar la descomposición LU en Python, se pueden utilizar las capacidades de programación del lenguaje para crear funciones que realicen el proceso de factorización de una matriz 3x3. Python ofrece una sintaxis clara y concisa, así como estructuras de datos adecuadas, como listas o arrays de NumPy, para representar y manipular matrices. Es importante validar la implementación de la calculadora en Python, verificando su precisión y corrección en una variedad de casos de prueba. Esto puede incluir matrices con diferentes propiedades, como matrices singulares, matrices con elementos cero o matrices diagonales dominantes. Las pruebas también pueden comparar los resultados de la calculadora con soluciones conocidas o calculadas manualmente.

Además, la documentación adecuada del código es esencial. Esto implica incluir comentarios explicativos que describan la lógica detrás de cada función y paso del algoritmo de descomposición LU.

Justificación

Se realiza este proyecto con la intención de abarcar distintos temas como la Aplicación en la Ingeniería y Ciencias teniendo que la descomposición LU es una técnica fundamental en diversos campos de la Ingeniería, las Ciencias Básicas, la Informática y la Economía; Facilitar el análisis de las matrices con las que se trabaja para encontrar una solución llevada a bordo en el mundo real de una manera más sencilla simplificando y agilizando los procesos en los mismos. Un programa como este también busca una ayuda en el Aprendizaje y Enseñanzas de distintas materias como es en este caso el Análisis Numérico permitiendo automatizar los procesos para acelerar el proceso de solución de distintos problemas presentados. En resumen, la creación de una Calculadora para la descomposición de una matriz 3×3 en la Forma L.U en Python ofrece beneficios prácticos, educativos y comunitarios, lo que justifica su desarrollo y aplicación en diversos Contextos. Ahora este proyecto aplica esa misma herramienta para resolver un sistema de ecuaciones Lineales 3×3 .

Metodología

La metodología utilizada en el transcurso de tomas de datos, análisis, desarrollo, y pruebas es una de las más conocidas por su Agilidad en el mundo de la creación y procesos esta fue la Metodología de Kanban que se centra en la visualización del trabajo y la gestión del flujo del mismo en donde se debe mostrar 3 separaciones que son: Cosas por Hacer, En Progreso y Terminados; colocando un límite de tareas que puede haber en cada columna para poder gestionar el flujo y no se genere una sobrecarga esto nos permite que el proyecto se desarrolle de manera Flexible, nos permite una mayor visibilidad y una mejora continua del desarrollo de actividades.

En resumen, Kanban es una metodología ágil que se centra en la visualización del trabajo, la limitación del trabajo en curso y la gestión del flujo de trabajo. Es una herramienta poderosa para mejorar la eficiencia y la productividad en proyectos de desarrollo de software y en una amplia gama de otros contextos empresariales.

Diagrama de Flujo

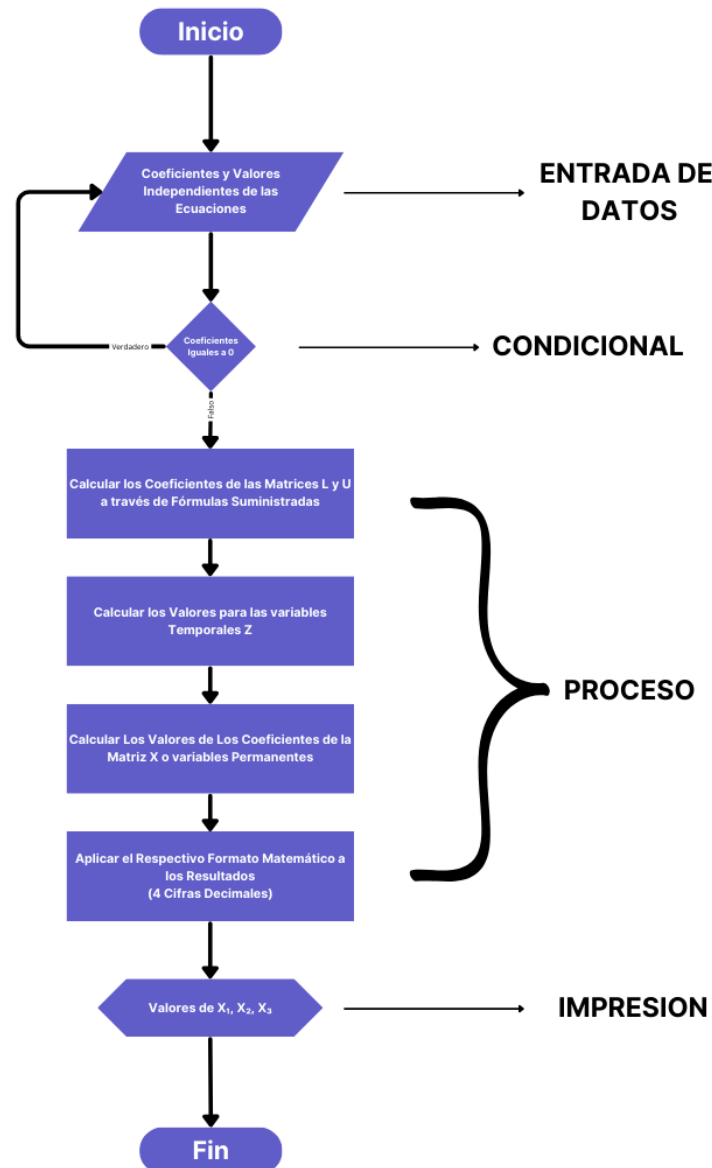


Ilustración 1. Diagrama de Flujo

Pseudocódigo

```

1  Definir Formato(Número):
2      devolver (redondear(Número, 4))
3
4  Definir Matn(X):
5      Declarar Matriz A, Matriz B, Matriz L, Matriz U, Matriz Z, Matriz X
6      Matriz A = Ingresar Datos
7      Matriz B = Datos Ingresados
8      Matriz U = LU(Matriz A, "U")
9      Matriz L = LU(Matriz A, "L")
10     Matriz Z = UB(Matriz L, Matriz B)
11     Matriz X = UZ(Matriz U, Matriz Z)
12
13     si X igual "A":
14         devolver (Matriz A)
15     pero si X igual "L":
16         devolver (Matriz L)
17     pero si X igual "U":
18         devolver (Matriz U)
19     pero si X igual "Z":
20         devolver (Matriz Z)
21     pero si X igual "X":
22         devolver (Matriz X)
23     sino:
24         devolver (Matriz B)
25
26 Definir LU(Matriz A, X):
27     U11 = Formato(Matriz A[0][0])
28     U12 = Formato(Matriz A[0][1])
29     U13 = Formato(Matriz A[0][2])
30
31     L21 = Formato(Matriz A[1][0]/U11)
32     U22 = Formato(Matriz A[1][1]-(L21*U12))
33     U23 = Formato(Matriz A[1][2]-(L21*U13))
34
35     L31 = Formato(Matriz A[2][0]/U11)
36     L32 = Formato(Matriz A[2][1]-(L31*U12)/U22)
37     U33 = Formato(Matriz A[2][2]-(L31*U13) - (L32*U23))
38
39     Matriz L = [[1, 0, 0], [L21, 1, 0], [L31, L32, 1]]
40     Matriz U = [[U11, U12, U13], [0, U22, U23], [0, 0, U33]]
41
42     Si N es Igual a "L":
43         devolver Matriz L
44     sino:
45         devolver Matriz U
46
47 Definir UB(Matriz L, Matriz B):
48     Z1 = Formato(Matriz B[0])
49     Z2 = Formato(Matriz B[1] - (Matriz L[1][0]*Z1))
50     Z3 = Formato(Matriz B[2] - (Matriz L[2][0]*Z1) - (Matriz L[2][1]*Z2))
51
52     Matriz Z = [Z1, Z2, Z3]
53
54     devolver Matriz Z
55
56 #Funcion para Encontrar Valores de X
57 Definir UZ(Matriz U, Matriz Z):
58     X3 = Formato(Matriz Z[2]/Matriz U[2][2])
59     X2 = Formato(Matriz Z[1]-(Matriz U[1][2]*X3)/Matriz U[1][1])
60     X1 = Formato(Matriz Z[0]-(Matriz U[0][2]*X3)-(Matriz U[0][1]*X2)/Matriz U[0][0])
61
62     Matriz_X = [X1, X2, X3]
63
64     devolver Matriz X

```

Ilustración 2. Pseudocódigo

Código Fuente

```
Index.py - Codigos - Visual Studio Code
1 from kivy.lang import Builder
2 from kivy.properties import StringProperty, ListProperty
3 from kivy.config import Config
4 Config.set('graphics','width','550')
5 Config.set('graphics','height','550')
6
7 from kivymd.app import MDApp
8 from kivymd.theming import ThemeableBehavior
9 from kivymd.ui.boxlayout import MDBoxLayout
10 from kivymd.ui.list import MDListItemLeadingIcon, MDList
11
12 ##### List #####
13
14 class ContentNavigationDrawer(MDBoxLayout):
15     pass
16
17 class ItemDrawer(MDListItemLeadingIcon):
18     icon = StringProperty()
19     text_color = ListProperty([0,0,0,1])
20
21 class DrawerList(ThemeableBehavior,MDList):
22     tabnine: test | explain | document | ask
23     def set_color_item(self, instance_item):
24         for item in self.children:
25             if item.text_color == self.theme_cls.primary_color:
26                 item.text_color = self.theme_cls
27                 break
28             instance_item.text_color = self.theme_cls.primary_color
29
30 class NavigationDrawer(MDApp):
31     tabnine: test | explain | document | ask
32     def build(self):
33         return Builder.load_file("Resources/Widget.kv")
```

Ilustración 3. Código Fuente

```
Widget.kv - Codigos - Visual Studio Code
1 <ItemDrawer>:
2     theme_text_color: "Custom"
3     on_release: self.parent.set_color_item(self)
4
5     IconLeftWidget:
6         id: icon
7         icon: root.icon
8         theme_text_color: "Custom"
9         text_color: root.text_color
10
11 <ContentNavigationDrawer>:
12     orientation: "vertical"
13     padding: "8dp"
14
15     AnchorLayout:
16         anchor_x: 'left'
17         size_hint_y: None
18         height: avatar.height
19
20     Image:
21         id: avatar
22         size_hint: None, None
23         size: "75dp", "75dp"
24         source: "img.png"
25
26     MDLabel:
27         text: "Test Validate Themes"
28         font_style: "Button"
29         adaptive_height: True
30
31     MDLabel:
32         text: "Juan David Durán Garzón"
```

```
1 #import Kivy as Kv
2
3 #Funcion de Formato
4 tabnine: test | explain | document | ask
5 def F(numero):
6     return <round(numero,4)>
7
8 #Funcion de Ejecucion Principal
9 tabnine: test | explain | document | ask
10 def Executable(X):
11     Matriz_B = [9,7,12]
12     Matriz_A = [[4,-2,-1],[5,1,-1],[1,2,-1]]
13     Matriz_U = LU(Matriz_A,"U")
14     Matriz_L = LU(Matriz_A,"L")
15     Matriz_Z = UB(Matriz_L,Matriz_B)
16     Matriz_X = UZ(Matriz_U,Matriz_Z)
17
18     if X == "A":
19         return Matriz_A
20     elif X == "U":
21         return Matriz_U
22     elif X == "L":
23         return Matriz_L
24     elif X == "Z":
25         return Matriz_Z
26     elif X == "X":
27         return Matriz_X
28     else:
29         return Matriz_B
30
31 #Funcion Para Encontrar Valores de L y U
32 tabnine: test | explain | document | ask
33 def LU(Matriz_A,N):
```

Ilustración 4. Código Fuente

Pasos Seguidos en el Desarrollo del Software

Contextualización

La descomposición LU de una matriz es un proceso matemático y geométrico que involucra la factorización de una matriz en dos matrices triangulares, una inferior (L) y otra superior (U). Matemáticamente, si tenemos una matriz cuadrada A de tamaño n veces n, la descomposición LU busca encontrar dos matrices L y U tal que $A = LU$.

Geoméricamente, este proceso puede entenderse como una transformación de la matriz original en dos componentes que representan información sobre cómo se combinan las operaciones elementales que se aplican a A.

La matriz L representa las transformaciones realizadas en A para llevarla a una forma triangular inferior, mientras que la matriz U representa las transformaciones realizadas en A para llevarla a una forma triangular superior.

Cada elemento en L y U proporciona información sobre cómo se ha afectado cada componente de A durante la factorización. Los elementos no diagonales de L y U muestran cómo las operaciones elementales de fila han afectado la matriz original, mientras que los elementos diagonales de L y U representan los multiplicadores que se utilizan durante el proceso.

En términos geométricos, la descomposición LU puede visualizarse como una serie de transformaciones lineales que se aplican a la matriz original para llevarla a una forma triangular inferior (L) y superior (U). Estas transformaciones representan una descomposición del espacio en subespacios generados por las columnas de A, lo que facilita el análisis y la resolución de sistemas de ecuaciones lineales y otras operaciones matriciales.

Haciendo aplicación de este proceso de descomposición se nos permitirá realizar las operaciones con las fórmulas otorgadas para encontrar los valores correspondientes de las variables temporales de la Matriz Z, y por consiguiente conseguir los datos de las variables permanentes de la Matriz X.

En resumen, la descomposición LU es un proceso matemático y geométrico que descompone una matriz en dos componentes triangulares, proporcionando información sobre las transformaciones lineales que se aplican a la matriz original para llevarla a una forma triangular inferior y superior. Esto facilita el análisis y la manipulación de matrices en diversas aplicaciones matemáticas y científicas.

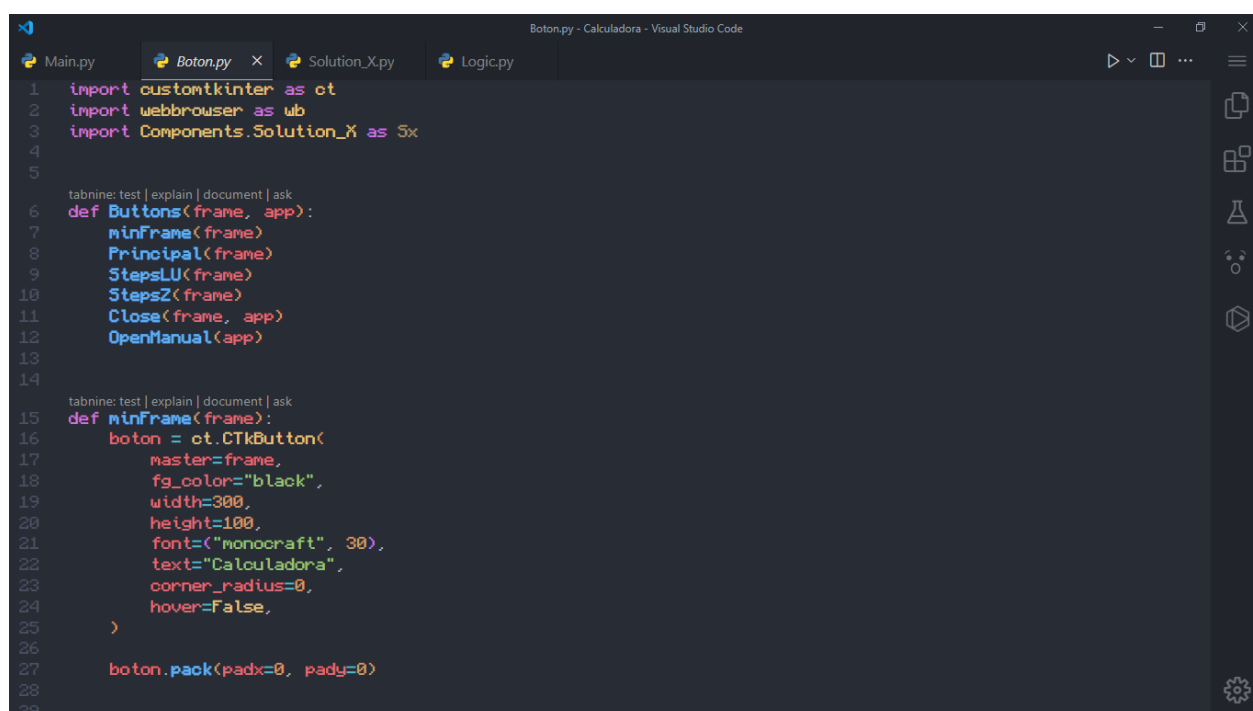
Requerimientos

- Pedir al Usuario los Datos de la Matriz A.
- Pedir al Usuario los Datos de la Matriz U y la Matriz L.
- Preguntar al Usuario si se desea Hallar los Valores de la Matriz A, o en su respectivo caso de la Matriz L y la Matriz U.
- Se debe mostrar a través de una UI los Procesos, Ingreso de Valores, Pasos y Respectivos resultados para el problema Ingresado por el Usuario.
- Se debe presentar un manual o una forma en la que el usuario sea capaz de entender rápidamente el uso o Implementación de la Aplicación.
- Pedir al usuario la cantidad de decimales que desea observar en el proceso según la exactitud que el mismo desee.
- Se desea que, si la persona desea realizar otra matriz, el ingreso actual sea “Limpiado” para el ingreso de una nueva matriz o matrices, según sea el caso.

Análisis

Se presenta una Actividad en la cual se requiere realizar una Calculadora en el Lenguaje de Programación Python el cual permita encontrar los valores de la Descomposición de una matriz 3x3 de la Forma L.U; lo cual implica realizar el proceso interno y mostrar de manera grafica los resultados de la Matriz U y de la Matriz L siguiendo los pasos presentados en las Horas de Clase, en caso de que la persona cuente con las Matrices L y U, se necesita que la persona pueda obtener los Valores de la Matriz A, osea realizar el proceso de una manera inversa a la cual se presentó en un comienzo.

Implementación



```
1 import customtkinter as ct
2 import webbrowser as wb
3 import Components.Solution_X as Sx
4
5
6 tabnine: test | explain | document | ask
7 def Buttons(frame, app):
8     minFrame(frame)
9     Principal(frame)
10    StepsLU(frame)
11    StepsZ(frame)
12    Close(frame, app)
13    OpenManual(app)
14
15 tabnine: test | explain | document | ask
16 def minFrame(frame):
17     boton = ct.CTkButton(
18         master=frame,
19         fg_color="black",
20         width=300,
21         height=100,
22         font=("monocraft", 30),
23         text="Calculadora",
24         corner_radius=0,
25         hover=False,
26     )
27     boton.pack(padx=0, pady=0)
28
29
```

Ilustración 5.Implementacion

```
1 import customtkinter
2 from Components import Containter
3
4 customtkinter.set_appearance_mode("dark")
5 customtkinter.set_default_color_theme("green")
6
7 app = customtkinter.CTk(fg_color="black")
8 app.wm_attributes("-fullscreen", True)
9
10 Containter.Ejecucion(app)
11
12
13 app.mainloop()
14
```

```
5 Lg.Matriz_B = [4, 3, 10]
6 Lg.Imprimir()
7
8
9 def FrameX(screen):
10     global Matriz_U
11
12     # Parte pasos En X
13     pasosX = ct.CTkFrame(
14         master=screen,
15         fg_color="black",
16         bg_color="black",
17     )
18
19     pasosX.pack(expand=True)
20     pasosX.pack(padx=2)
21     pasosX.pack(fill=ct.BOTH)
22
23     # Pasos donde Se Ingresan Datos
24     frameInsert = ct.CTkFrame(
25         master=pasosX,
26         fg_color="green",
27         height=300,
28     )
29
30     frameInsert.pack(expand=True)
31     frameInsert.pack(fill=ct.X)
32     frameInsert.pack_propagate(False)
33
34     # Pasos donde se Realiza Toda el Proceso
```

```
Logicpy - Calculadora - Visual Studio Code
Main.py Boton.py Solution_X.py Logicpy X
1 # Funcion de Formato
  tabnine: test | explain | document | ask
2 def F(numero):
3     return round(numero, 4)
4
5
6 Matriz_A = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
7 Matriz_B = [0, 0, 0]
8
9
10 tabnine: test | explain | document | ask
11 def Imprimir():
12     print(Matriz_A)
13     print(Matriz_B)
14     return 0
15
16 # Funcion de Ejecucion Principal
17 tabnine: test | explain | document | ask
18 def Executable(X):
19     Matriz_U = LU(Matriz_A, "U")
20     Matriz_L = LU(Matriz_A, "L")
21     Matriz_Z = UB(Matriz_L, Matriz_B)
22     Matriz_X = UZ(Matriz_U, Matriz_Z)
23
24     if X == "A":
25         return Matriz_A
26     elif X == "U":
27         return Matriz_U
28     elif X == "L":
29         return Matriz_L
```

```
Menu.py - Calculadora - Visual Studio Code
Main.py Solution_X.py Logicpy Menu.py X
1 import customtkinter as ct
2 import Components.Boton as Boton
3
4
5 tabnine: test | explain | document | ask
6 def MainFrame(container, app):
7     frame = ct.CTkFrame(
8         master=container,
9         fg_color="black",
10         bg_color="black",
11     )
12
13     frame.pack(side=ct.LEFT)
14     frame.pack(fill=ct.Y)
15
16     Boton.Buttons(frame, app)
```

```
Screen.py - Calculadora - Visual Studio Code
Main.py Solution_X.py Logic.py Screen.py X
1 import customtkinter as ct
2 import Components.Solution_X as SX
3
4
5 tabnine: test | explain | document | ask
6 def PrincipalShowData(container):
7     frame = ct.CTkFrame(
8         master=container,
9         fg_color="blue",
10     )
11     frame.pack(side=ct.LEFT)
12     frame.pack(expand=True)
13     frame.pack(fill=ct.BOTH)
14
15     SX.FrameX(frame)
16
```

```
Container.py - Calculadora - Visual Studio Code
Main.py Solution_X.py Logic.py Container.py X
3 import Components.Screen as Screen
4
5
6 tabnine: test | explain | document | ask
7 def Ejecution(app):
8     frame = ct.CTkFrame(
9         master=app,
10         fg_color="pink",
11     )
12     frame.pack(expand=True, fill=ct.BOTH)
13
14     Menu.MainFrame(frame, app)
15     Screen.PrincipalShowData(frame)
16
```

```

Design.py - Calculadora - copia - Visual Studio Code
Main.py  Logic.py  Design.py x  ContentZ.py  ContentMain.py

1 import customtkinter as ct
2 import Components.ContentMain as Content
3 import Components.ContentLU as ContentLU
4 import Components.ContentZ as ContentZ
5
6
7 subinter test | main | document | ask
8 def MenuBotones(app):
9     # Definición de Funciones que Llanaran los Botones
10
11     def Close():
12         app.destroy()
13
14     def OrganizaBotones(Boton):
15         Boton.pack(side=ct.TOP)
16         Boton.pack(fill=ct.X)
17         Boton.pack(pady=1)
18
19     def ScreensPosittions(Screen):
20         Screen.pack(expand=True)
21         Screen.pack(fill=ct.BOTH)
22         Screen.pack(padx=2)
23
24     def OpenMainScreen():
25         ScreensPosittions(ScreenMainFrame)
26         ScreenLuFrame.pack_forget()
27         ScreenZFrame.pack_forget()
28
29     def OpenLuScreen():
30         ScreensPosittions(ScreenLuFrame)
31         ScreenMainFrame.pack_forget()
32         ScreenZFrame.pack_forget()
33
34     def OpenZScreen():
35         ScreensPosittions(ScreenZFrame)
36         ScreenLuFrame.pack_forget()
37         ScreenMainFrame.pack_forget()
38
39     # Frame donde se Ubican los Botones Correspondientes del Menu
40
41     MenuFrame = ct.CTkFrame(
42         master=app,
43         fg_color="#FFFFFF",
44         width=300

```

```

Main.py - Calculadora - Visual Studio Code
Main.py x  Solution_X.py  Logic.py  Containter.py

1 import customtkinter
2 from Components import Containter
3
4 customtkinter.set_appearance_mode("dark")
5 customtkinter.set_default_color_theme("green")
6
7 app = customtkinter.CTk(fg_color="black")
8 app.wm_attributes("-fullscreen", True)
9
10 Containter.Ejecucion(app)
11
12
13 app.mainloop()
14

```

EXPLORADOR: CALCULADORA

- Archives
- Components
 - __pycache__
 - Boton.py
 - Containter.py
 - Logic.py
 - Menu.py
 - Screen.py
 - Solution_X.py
 - Main.py
- Seudocodigo.txt

```
ContentMain.py - Calculadora - copia - Visual Studio Code
Main.py  Logic.py  Design.py  ContentZ.py  ContentMain.py x
1 import customtkinter as ct
2 import Components.Logio as Lg
3
4 Lg.Matriz_A = [[43, 2, 32], [-32, 23, 40], [15, 20, -19]]
5 Lg.Matriz_B = [12, 32, 9]
6
7
8 tablinerest | explain | document | ask
9 @staticmethod
10 def PeHake():
11     Lg.Matriz_A = [[51, 2, 31], [3, 5, 9], [12, 5, -3]]
12     Lg.Matriz_B = [5, 3, 9]
13
14 tablinerest | explain | document | ask
15 def ScreenData(frame):
16     # Function to Archive
17     def OrdenContentMain(Cuadro):
18         Cuadro.pack(expand=True)
19         Cuadro.pack(side=ct.TOP)
20         Cuadro.pack(fill=ct.BOTH)
21
22     FrameInputData = ct.CTkFrame(
23         master=frame,
24         fg_color="#293C4B",
25         corner_radius=0,
26     )
27
28     LabelKey = ct.CTkLabel(
29         master=FrameInputData,
30         corner_radius=0,
31         fg_color="#293C4B",
32         text="(",
33         text_color="white",
34         font=("Times New Roman", 200),
35         anchor="center",
36     )
37
38     FrameInputInvisibleBig = ct.CTkFrame(
39         master=FrameInputData,
40         fg_color="#293C4B",
41         height=200,
42         width=250,
43         corner_radius=0,
```

Diseño

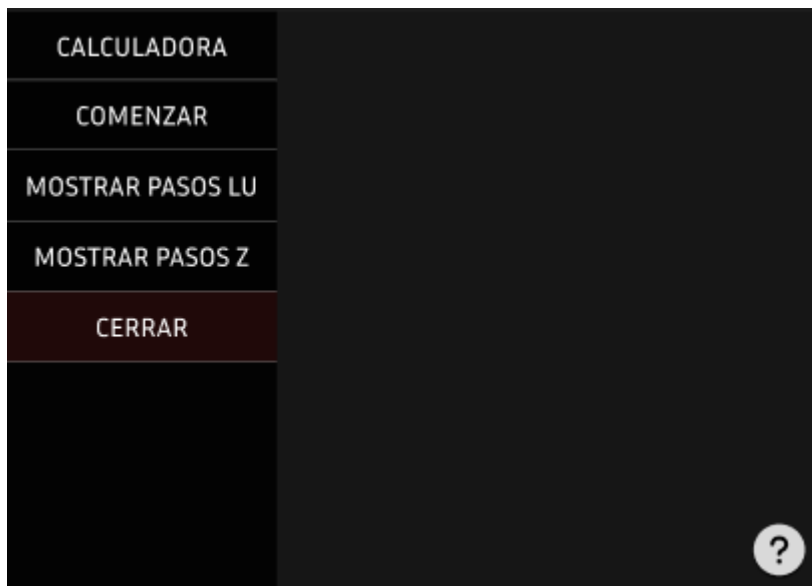
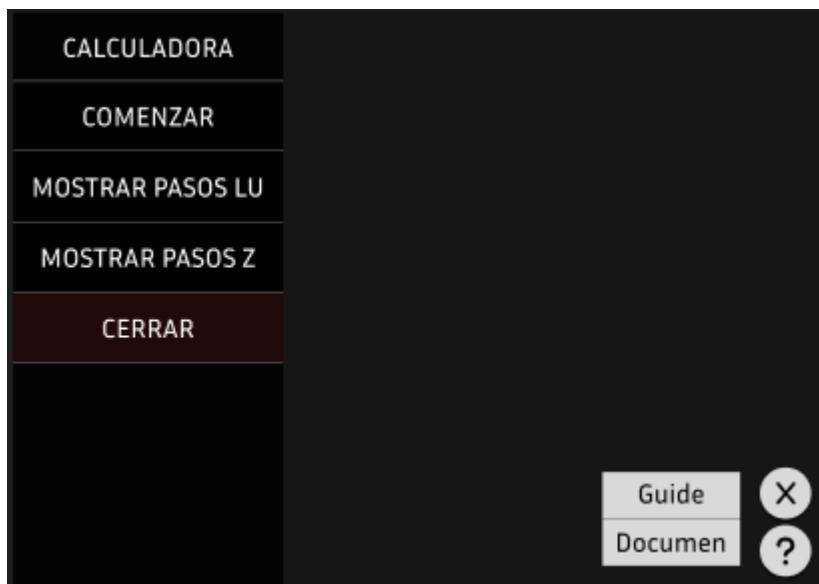


Ilustración 6. Diseño



Prueba

Matriz A = [[34, 2, 3], [3, 5, 9], [12, 5, -3]]

Matriz B = [5, 3, 9]

$$L \begin{pmatrix} 1.0000 + 0.0000 + 0.0000 \\ 0.0882 + 1.0000 + 0.0000 \\ 0.3529 + 0.8902 + 1.0000 \end{pmatrix} * U \begin{pmatrix} 34.0000 + 2.0000 + 3.0000 \\ 0.0000 + 4.8236 + 8.7354 \\ 0.0000 + 0.0000 + -11.8350 \end{pmatrix} = B \begin{pmatrix} 5.0000 \\ 3.0000 \\ 9.0000 \end{pmatrix}$$

$Z_1 = 5.0000$
 $Z_2 = 2.5590$
 $Z_3 = 4.9575$

$X_1 = 0.1082$
 $X_2 = 1.2891$
 $X_3 = -0.4189$

① $(34.0000 * 0.1082) + (2.0000 * 1.2891) + (3.0000 * -0.4189) = 5.0000$; $5.0000 = 5.0000$
 ② $(3.0000 * 0.1082) + (5.0000 * 1.2891) + (9.0000 * -0.4189) = 3.0000$; $3.0000 = 3.0000$
 ③ $(12.0000 * 0.1082) + (5.0000 * 1.2891) + (-3.0000 * -0.4189) = 9.0000$; $9.0000 = 9.0000$

Ilustración 7. Prueba 1

Matriz A = [[43, 2, 32], [-32, 23, 40], [15, 20, -19]]

Matriz B = [12, 32, 9]

$$L \begin{pmatrix} 1.0000 + 0.0000 + 0.0000 \\ -0.7442 + 1.0000 + 0.0000 \\ 0.3488 + 0.7882 + 1.0000 \end{pmatrix} * U \begin{pmatrix} 43.0000 + 2.0000 + 32.0000 \\ 0.0000 + 24.4884 + 63.8144 \\ 0.0000 + 0.0000 + -80.4601 \end{pmatrix} = B \begin{pmatrix} 12.0000 \\ 32.0000 \\ 9.0000 \end{pmatrix}$$

$Z_1 = 12.0000$
 $Z_2 = 40.9304$
 $Z_3 = -27.4469$

$X_1 = -0.0112$
 $X_2 = 0.7825$
 $X_3 = 0.3411$

① $(43.0000 * -0.0112) + (2.0000 * 0.7825) + (32.0000 * 0.3411) = 12.0000$; $12.0000 = 12.0000$
 ② $(-32.0000 * -0.0112) + (23.0000 * 0.7825) + (40.0000 * 0.3411) = 32.0000$; $32.0000 = 32.0000$
 ③ $(15.0000 * -0.0112) + (20.0000 * 0.7825) + (-19.0000 * 0.3411) = 9.0000$; $9.0000 = 9.0000$

Ilustración 8. Prueba 2

Detalles del Procedimiento con la Metodología Aplicada

Con la aplicación de la metodología de Kanban se inició colocando las tareas relacionadas con la documentación del proyecto en la columna "Por Hacer" del tablero Kanban. A medida que se avanzaba, algunas tareas se movieron a "En Progreso", como la búsqueda de librerías y herramientas necesarias para el desarrollo en Python, así como programas como Visual Studio Code, Figma, Canva, Diagramas de Flujo de Datos, Excel, Word, Tabnine, FontAwesome y Pinterest. Luego, la programación de la calculadora se convirtió en la tarea principal "En Progreso", tomando alrededor de 4 o 5 días para completarse. Una vez finalizada la funcionalidad básica, la tarea se movió a "Hecho" y se iniciaron nuevas tareas como la documentación del proyecto y el manual de usuario. Paralelamente, se incorporó GitHub como controlador de versiones, moviendo esa tarea a "En Progreso" mientras se realizaban los commits y se probaba la aplicación con diferentes valores. Finalmente, después de las pruebas satisfactorias y la finalización de la documentación, el proyecto se marcó como "Hecho", entregando una calculadora funcional y bien documentada. El método Kanban permitió una visualización clara del flujo de trabajo, limitando el trabajo en progreso y promoviendo la colaboración y la entrega incremental.

Resultados

El proyecto “Desarrollo de un Software para la Solución Matemático de la Descomposición de la Forma LU de una Matriz A 3x3 en Python” se ha realizado de manera satisfactoria, logrando con énfasis el desarrollo de los objetivos planteados al inicio del proyecto mediante un análisis de requerimientos necesarios. Se pudo conseguir el correcto funcionamiento del Software para la Solución de Ecuaciones 3x3 mediante la descomposición LU de una Matriz, se consiguió de manera efectiva la implementación grafica de Bibliotecas además de Diseño UI/UX del aplicativo. El Trabajo ha dado un minimo porcentaje de error estimado, consagrándolo asi como un programa matemático eficiente en el Area de la Ingeniería anticipando su uso a cualquier usuario por la implementación de manuales de uso y la utilización de la documentación del Proyecto mismo.

Conclusiones

El proyecto “Desarrollo de un Software para la Solución Matemático de la Descomposición de la Forma LU de una Matriz A 3x3 en Python” el cual buscaba consagrar la realización de la solución mediante un software tras haber completado su correcto funcionamiento lo cual era clave en el desarrollo de la actividad. En el transcurso de la documentación e implementación se hizo uso de diferentes aplicativos y bibliotecas para el desarrollo de las mismas, aplicaciones de ayuda en documentación como lo es Word, DFD, Canva; para la implementación se utilizaron tecnologías como Tkinter, Python y otras librerías utilizando como herramienta de IDE la aplicación Visual Studio Code.

Se espera que con este proyecto se pueda llegar a su eficiente uso por parte de usuarios aun sin experiencia del tema, facilitando el desarrollo de actividades referentes al proyecto para otros participantes. El mantenimiento y soporte del programa se llevará a mediano tiempo según como se sigan desarrollando las pautas otorgadas por la entidad a cargo (Universidad Francisco de Pausa Santander, Seccional Ocaña) para su debido uso frente a las personas encargadas.

Enlace Código e Imágenes Finales

CALCULADORA	$\begin{cases} 654 X_1 + 6 X_2 + 15 X_3 = 4 \\ 465 X_1 + 4 X_2 + 16 X_3 = 6 \\ 54 X_1 + 64 X_2 + 1 X_3 = 654 \end{cases}$ <p>Ejecutar Limpiar</p>
Ingreso Datos	
Proceso 'LU'	
Proceso 'Z'	
Cerrar	

$$L \begin{pmatrix} 1.0000 + 0.0000 + 0.0000 \\ 0.7110 + 1.0000 + 0.0000 \\ 0.0826 + -238.7383 + 1.0000 \end{pmatrix} * U \begin{pmatrix} 654.0000 + 6.0000 + 15.0000 \\ 0.0000 + -0.2660 + 5.3350 \\ 0.0000 + 0.0000 + 1.273.4298 \end{pmatrix} = B \begin{pmatrix} 4.0000 \\ 6.0000 \\ 654.0000 \end{pmatrix}$$

$Z_1 = 4.0000$
 $Z_2 = 3.1560$
 $Z_3 = 1.407.1277$

$X_1 = -0.1137$
 $X_2 = 10.2977$
 $X_3 = 1.1050$

① $(654.0000 * -0.1137) + (6.0000 * 10.2977) + (15.0000 * 1.1050) = 4.0000$; $4.0000 = 4.0000$
 ② $(465.0000 * -0.1137) + (4.0000 * 10.2977) + (16.0000 * 1.1050) = 6.0000$; $6.0000 = 6.0000$
 ③ $(54.0000 * -0.1137) + (64.0000 * 10.2977) + (1.0000 * 1.1050) = 654.0000$; $654.0000 = 654.0000$

CALCULADORA	$\begin{cases} X_1 + X_2 + X_3 = \\ X_1 + X_2 + X_3 = \\ X_1 + X_2 + X_3 = \end{cases}$ <p>Ejecutar Limpiar</p>
Ingreso Datos	
Proceso 'LU'	
Proceso 'Z'	
Cerrar	

CALCULADORA	$L \begin{pmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{pmatrix} * Z \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} = B \begin{pmatrix} B_1 \\ B_2 \\ B_3 \end{pmatrix}$ <p>Despues de la Respectiva Formula Despejamos, Obteniendo:</p> $Z_1 = B_1 / L_{11} ; Z_1 = 4.0000$ $Z_2 = B_2 - (L_{21} * Z_1) ; Z_2 = 6.0000 - (0.7110 * 4.0000) = 3.1560$ $Z_3 = B_3 - (L_{31} * Z_1) - (L_{32} * Z_2) ; Z_3 = 654.0000 - (0.0826 * 4.0000) - (-238.7383 * 3.1560) = 1,407.1277$ <p>Comenzamos a Realizar el templazo y Encontramos las Matrices:</p> $L \begin{pmatrix} 1.0000 & 0.0000 & 0.0000 \\ 0.7110 & 1.0000 & 0.0000 \\ 0.0826 & -238.7383 & 1.0000 \end{pmatrix} * Z \begin{pmatrix} 4.0000 \\ 3.1560 \\ 1,407.1277 \end{pmatrix} = B \begin{pmatrix} 4.0000 \\ 6.0000 \\ 654.0000 \end{pmatrix}$
Ingreso Datos	
Proceso 'LU'	
Proceso 'Z'	
Cerrar	

?

Enlace:

<https://github.com/TheOddProgrammer/Calculator.git>

Bibliografía

J. E. Peña Rodríguez, M. R. Rojas Silva, and R. L. Soto Montero, “Una nota sobre el Método Simplex y la descomposición LU”, *Proyecciones (Antofagasta, Online)*, vol. 7, no. 14, pp. 71-84, Mar. 2018.

Tursunbek Sadridinovich Jalolov. (2023). TEACHING THE BASICS OF PYTHON PROGRAMMING. *International Multidisciplinary Journal for Research & Development*, 10(11). Retrieved from <https://www.ijmrd.in/index.php/imjrd/article/view/443>

Ulloa, R. (2013). Kivy: Interactive Applications in Python - Roberto Ulloa - Google Libros. Packt Publishing, 45–53. Retrieved 25 September. 2013, from <https://doi.org/https://books.google.es/books?id=8Y79AAAAQBAJ>

Schreiber, Andreas (2013) Developing Apps for Android and Other Platforms with Kivy and Python. droidcon 2013, 7.-10. Apr. 2013, Berlin.