Technische Universität Berlin, Fakulät IV
Institut für Energie und Automatisierungstechnik
Fachgebiet Elektronische Mess- und Diagnosetechnik

Großes Projekt Messdatenverarbeitung (LV 0430 L349)
Betreuung: Daniel Thomanek
SoSe 2022

Entwicklung einer Simulationstoolbox für die Übung zur Lehrveranstaltung Grundlagen der elektronischen Messtechnik

Testing

Juan Nicolas Pardo Martin (397882)
Boris Maurer (409862)
Erik Tröndle (409537)

23.10.2022

Inhaltsverzeichnis

1 Wartungshandbuch

Softwareanforderungen Für die Erstellung der Programme wurde die Python Version 3.9 benutzt. Die folgenden folgende Bibliotheken und deren Versionen wurden benutz:

- Tkinter wird benutzt ist aber keine Bibliothek, die installiert sein muss, kommt mit er Python Version
- Matplotlib 3.5.2
- Numpy 1.21.6
- Control 0.9.2

Als Entwicklungsumgebung wurde Spyder benutzt der durch Anaconda installiert werden kann. Als ein Texteditor kann auch VS Code benutzt werden wo die Programme problemlos ausgeführt werden können. Für das Projekt wurde die VS Code 1.72.2 und Spyder 4.1.5 Version benutzt. Bei Erstellung der 7 Übungen sind die Grundlagen von Messtechnik benötigt, die bei den Entsprechenden Folien von Modul Grundlagen der Messtechnik nachgeholt werden können.

2 Testing

Dieser Code führt verschiedene Tests für 7 Übungen durch:

1. In `test1()` wird das Modul `Ubung_01` importiert und verschiedene Einstellungen werden nacheinander geändert, während das GUI-Eventhandling durch `wait()` fortgesetzt wird. Schließlich wird das Hauptfenster zerstört.
2. `test2()` importiert die Module `Ubung_01` und `Übung_02`. Es ändert die Dropdown-Auswahl und fügt Datenpunkte in die Textfelder ein. Die Funktion `show()` aktualisiert das Plot und wartet. Am Ende wird das Hauptfenster zerstört.
3. In `test3()` wird das Modul `Übung_03` importiert und ein neues `ue3TKClass`-Objekt erstellt. Die Funktionen `button3test()`, `button2test()`, `button1test()`, `slider1test()`, `slider2test()` und `slider3test()` testen verschiedene Schieberegler und Schaltflächen, während das Eventhandling fortgesetzt wird. Schließlich wird das Fenster zerstört.
4. `test4()` importiert das Modul `Ubung_04` und testet zwei Schieberegler (`slider1test()`

und `slider2test()`) in Kombination, während das Eventhandling weiterläuft. Schließlich wird das Hauptfenster zerstört.

5. In `test5()` wird das Modul `Ubung_05` importiert und ein neues `ue5TKClass`-Objekt erstellt. Die Funktion `slider1test()` ändert den Schieberegler und aktualisiert die Darstellung, während das Eventhandling fortgesetzt wird. Schließlich wird das Fenster zerstört.
6. `test6()` importiert das Modul `Übung_06` und erstellt ein neues `ue6TKClass`-Objekt. Es werden verschiedene Einstellungen geändert, das Eventhandling fortgesetzt und schließlich das Fenster zerstört.
7. Schließlich führt `test7()` das Modul `Übung_07` aus, erstellt ein neues `ue7TKClass`-Objekt und ändert die Eingabefelder, während das Eventhandling fortgesetzt wird.

2.1 Übung 1

Dieser Python-Code erstellt und visualisiert Histogramme für zufällig generierte Daten, die einer Normalverteilung und einer Student-t-Verteilung folgen. Die wichtigsten Funktionen des Codes sind:

- Generierung von zufälligen Daten, die einer Normalverteilung oder einer Student-t-Verteilung folgen.
- Anpassung der Anzahl der Bins und Samples für die Histogramme.
- Anpassung des Freiheitsgrads für die Student-t-Verteilung.
- Animation der Histogramme zur Veranschaulichung der Verteilungen.
- Option zum Einfrieren des Datensatzes, sodass die Histogramme nicht mehr aktualisiert werden.
- Anzeige oder Verbergen der Legende für die Histogramme.

Der Code verwendet die Bibliotheken ‘numpy‘, ‘matplotlib‘ und ‘tkinter‘ für die Erstellung und Animation der Histogramme sowie für die Erstellung der grafischen Benutzeroberfläche (GUI). Es gibt Schieberegler, um die Anzahl der Bins, die Anzahl der Samples und den Freiheitsgrad der Student-t-Verteilung anzupassen. Zudem gibt es Kontrollkästchen, um das Aktualisieren der Samples pro Frame zu steuern und den Datensatz einzufrieren. Der Code enthält auch Kommentare und Dokumentation für die verschiedenen Funktionen und Teile

des Codes.

2.2 Übung 2

Der gegebene Python-Code ist ein Programm zur grafischen Darstellung von Regression, linearer Interpolation und kubischer Spline-Interpolation für eine gegebene Messreihe. Die Daten werden über ein GUI-Fenster eingegeben und der Benutzer kann eine Methode auswählen, um die unbekannten Werte abzuschätzen.

- Zuerst werden einige notwendige Bibliotheken importiert, wie `tkinter` für die GUI, `numpy` für numerische Berechnungen, `random` für Zufallszahlen, `matplotlib` für das Plotten und `scipy` für die Interpolation.
- Anschließend werden Funktionen für das Sortieren von Eingabe- und Ausgabewerten, das Erstellen einer linearen Approximation, die lineare Regression, die lineare Interpolation und die kubische Spline-Interpolation definiert.
- Das Hauptfenster der GUI wird erstellt und verschiedene Eingabefelder und Labels für die Benutzerinteraktion werden hinzugefügt.
- Es werden auch Funktionen definiert, um den ausgewählten Plot zu aktualisieren und Werte für eine unbekannte Größe abzuschätzen, basierend auf der gewählten Methode.
- Schließlich wird das Hauptfenster der GUI angezeigt und auf Benutzereingaben gewartet.

Der Code kann beispielsweise verwendet werden, um den Widerstand in Abhängigkeit von der Temperatur zu approximieren und auszurechnen.

3 Übung 3

Dieses Python-Skript erstellt ein Tkinter-Fenster zur Darstellung und Analyse der statischen Eigenschaften von Messsystemen. Es zeigt die reale und ideale Systemkennlinie mit verschiedenen Fehlertypen, wie Offsetfehler, Verstärkungsfehler und Linearitätsfehler. Es ermöglicht dem Benutzer, diese Fehler mithilfe von Schiebereglern und Kontrollkästchen zu manipulieren.

Der Code besteht aus einer Hauptklasse `TKWindow`, die ein Tkinter-Fenster erstellt und alle

erforderlichen Funktionen und Widgets enthält. Hier ist eine Zusammenfassung der Hauptfunktionen des Codes:

- Erstellen eines Tkinter-Fensters mit dem Titel „Übung 03 - Statische Eigenschaften von Messsystemen“.
- Erstellen und Platzieren von Schiebereglern und Kontrollkästchen, um Offsetfehler, Verstärkungsfehler und Linearitätsfehler zu steuern.
- Initialisieren einer Matplotlib-Figur und zugehöriger Achsen zur Darstellung der Kennlinien.
- Aktualisieren der Kennlinien basierend auf den Werten der Schieberegler und der Auswahl der Kontrollkästchen.
- Animieren der Matplotlib-Figur, um die Änderungen der Kennlinien in Echtzeit darzustellen.
- Starten der Tkinter-Fensterschleife.

3.1 Übung 4

Dieser Python-Code führt eine Analyse eines zweiten Ordnungs LTI-Systems durch. Die wichtigsten Schritte sind wie folgt:

1. Importieren der benötigten Bibliotheken, wie `numpy`, `control`, `matplotlib`, `tkinter` und `yaml`.
2. Einlesen der Kommandozeilenargumente, um zusätzliche Rendering-Optionen zu ermöglichen.
3. Initialisierung des `Tkinter`-Fensters und Erstellung von `Tkinter`-Widgets, wie Labels, Skalen und FigureCanvasTkAgg-Objekten.
4. Definition von Funktionen zur Aktualisierung der Systemparameter (`onVarChange`, `onZetaChange` und `onfrequenzChange`), die bei Änderungen der Schieberegler aufgerufen werden.
5. Erstellung von Bode-Diagramm, Pol-Nullstellen-Diagramm und Sprungantwort-Plot des LTI-Systems und Einbettung dieser Plots in das `Tkinter`-Fenster.
6. Animation der erstellten Plots, um sie bei Änderungen der Systemparameter automatisch zu aktualisieren.

7. Anzeige von Systemeigenschaften, wie Anstiegszeit und Überschwingen, in einem `Tkinter`-Label.
8. Start der `Tkinter`-Hauptschleife und Schließen aller Plots beim Beenden der Anwendung.

3.2 Übung 5

Das Python-Script verwendet das tkinter-Modul, um eine grafische Benutzeroberfläche für ein Übungsprogramm zu erstellen, das die digitale Messdatenerfassung mittels Analog-Digital-Umsetzer (ADU) demonstriert. Die Hauptkomponenten des Codes sind:

- Definition der `TKWindow`-Klasse, die von der `tk.Tk`-Klasse erbt und die Hauptanwendungsfenster erstellt.
- Initialisierung und Platzierung von Widgets, wie Schieberegler zur Einstellung der Bitanzahl und Beschriftung zur Anzeige der Auflösung.
- Erstellung von Matplotlib-Plots innerhalb der tkinter-Benutzeroberfläche, um die ADU-Kennlinie, die ideale Kennlinie und das Quantisierungsrauschen darzustellen.
- Implementierung der `animate`-Funktion, die die Plots basierend auf der aktuellen Einstellung des Schiebereglers für die Bitanzahl aktualisiert.
- Berechnung der Auflösung basierend auf der Anzahl der Bits und Aktualisierung des entsprechenden Beschriftungstexts.

Wenn das Skript ausgeführt wird, wird ein Fenster mit der Bezeichnung „Übung 05 - Digitale Messdatenerfassung mittels ADU“ geöffnet. Der Benutzer kann die Bitanzahl mit dem Schieberegler anpassen, und die Anwendung zeigt die entsprechenden aktualisierten Plots und die berechnete Auflösung an.

3.3 Übung 6

Der Code definiert eine Klasse `TKWindow`, die ein tkinter-Fenster erstellt, um ein Diagramm von Spannung und Strom anzuzeigen. Die Klasse hat mehrere Instanzvariablen zum Speichern verschiedener Attribute im Zusammenhang mit dem Diagramm, wie Amplitude, Frequenz und verschiedene Datenlisten für die Zeit-, Spannungs-, Strom- und Leistungswerte. Es hat mehrere Methoden zum Erstellen und Aktualisieren des Diagramms sowie Hilfsmethoden zum

Erstellen von Schiebereglern, Dropdown-Menüs und Texteingaben.

Die `__init__`-Methode initialisiert die Klasse, während die `createWidgets`-Methode die Benutzeroberflächenkomponenten einrichtet, einschließlich Beschriftungen und Menüs. Die `window_loop`-Methode startet die Hauptschleife für das Fenster, und die `animate`-Methode aktualisiert das Diagramm, indem sie die neuen Datenpunkte berechnet und plottet.

Zusätzlich gibt es Methoden zum Erstellen von Schiebereglern (`createSlider`), Dropdown-Menüs (`createDropdown`) und Texteingaben (`createTextInput`). Die `onSelection`-Methode behandelt die Auswahl verschiedener Wellenformtypen, und die `change_frequency`-Methode aktualisiert die Frequenz und Phase der Wellenform. Der Code definiert eine Klasse namens `TKWindow`, die ein grafisches Benutzeroberfläche (GUI) Fenster mit verschiedenen Widgets für eine Simulation im Zusammenhang mit elektrischen Schaltungen erstellt. Die Hauptmerkmale umfassen:

1. Initialisierung des Fensters mit Schiebereglern, Dropdown-Menüs, Texteingaben und Canvas.
2. Bereitstellung von Popup-Menüs für verschiedene elektrische Komponenten wie Kondensatoren, Widerstände, Induktoren, Dioden und DIACs.
3. Anzeige von Spannung, Leistung und Stromwerten in einem Matplotlib-Diagramm mit verschiedenen Ansichten.
4. Ermöglichen von Benutzern, die Anzeige mit Optionen wie Anzeigen von effektiven Werten, symmetrischer Y-Achse und Legenden anzupassen.
5. Zurücksetzen der angezeigten Werte und Animation bei Bedarf.

Der `__main__` Abschnitt des Codes instanziiert die `TKWindow`-Klasse und startet die Hauptschleife für die Anwendung.

3.4 Übung 7

Dieser Code definiert eine Klasse namens `TKWindow`, die ein benutzerdefiniertes Tkinter-Fenster zur Visualisierung von komplexen Signalen und Messbrückenwerten erstellt. Die Klasse hat Attribute zum Speichern von Figurenparametern, Diagrammen, Widgets und Messbrückenwerten.

Die `__init__`-Methode initialisiert das Fenster, die Rahmen, die Widget-Parameter, erstellt die Widgets, initialisiert die Widgets und richtet die Figuren und Achsen ein. Die

Methoden `init_window`, `init_frames`, `init_params`, `create_widgets`, `init_widgets`, `init_figure` und `init_axes` sind für die Initialisierung verschiedener Teile des Fensters verantwortlich.

Die `TKWindow`-Klasse verfügt über mehrere Wörterbücher, die die Eigenschaften und Konfiguration des Fensters definieren, wie z.B. die Figurenparameter, Diagramme für die Signale und Widgets für die Benutzeroberfläche. Das Fenster enthält eine Leinwand zur Anzeige eines Bildes, Beschriftungen zur Beschreibung der Eingabefelder, Eingabefelder für den Benutzer zum Eingeben komplexer Werte und zwei Plots: einen zur Anzeige der Signalverläufe über die Zeit und einen weiteren zur Darstellung eines Signals gegen ein anderes.

Dieser Code definiert eine auf TKinter basierende Anwendung zur Visualisierung und Animation einer Messbrücke mit komplexen Impedanzwerten. Die Hauptkomponenten des Codes sind:

1. `place_widgets()`: Platziert die TKinter-Widgets (grafische Benutzeroberflächen-Elemente) in der Anwendung.
2. `window_loop()`: Startet die Animation und Fensterschleife.
3. `animate(i)`: Aktualisiert die Achsenobjekte und zeichnet die Diagramme, wobei `i` eine Ganzzahl für die Animation ist.
4. `check_for_changed_entry()`: Überprüft, ob Eingabevariablen aktualisiert oder geändert wurden.
5. `entry_changed(entry)`: Vergleicht den aktuellen Wert einer gegebenen Variable mit ihrem alten Wert.
6. `update_measurement_bridge_values()`: Aktualisiert die Messbrückenwerte basierend auf der Benutzereingabe.
7. `update_graphs()`: Aktualisiert die Diagrammwerte.
8. `plot_axes(axes, graph, x=None)`: Zeichnet das gegebene Diagramm im angegebenen Achsenobjekt, mit optionalen x-Achsenwerten.

Die Anwendung erstellt eine Instanz der `TKWindow`-Klasse und startet die Fensterschleife. Sie aktualisiert die Diagramme dynamisch basierend auf der Benutzereingabe und animiert die Messbrückenwerte.