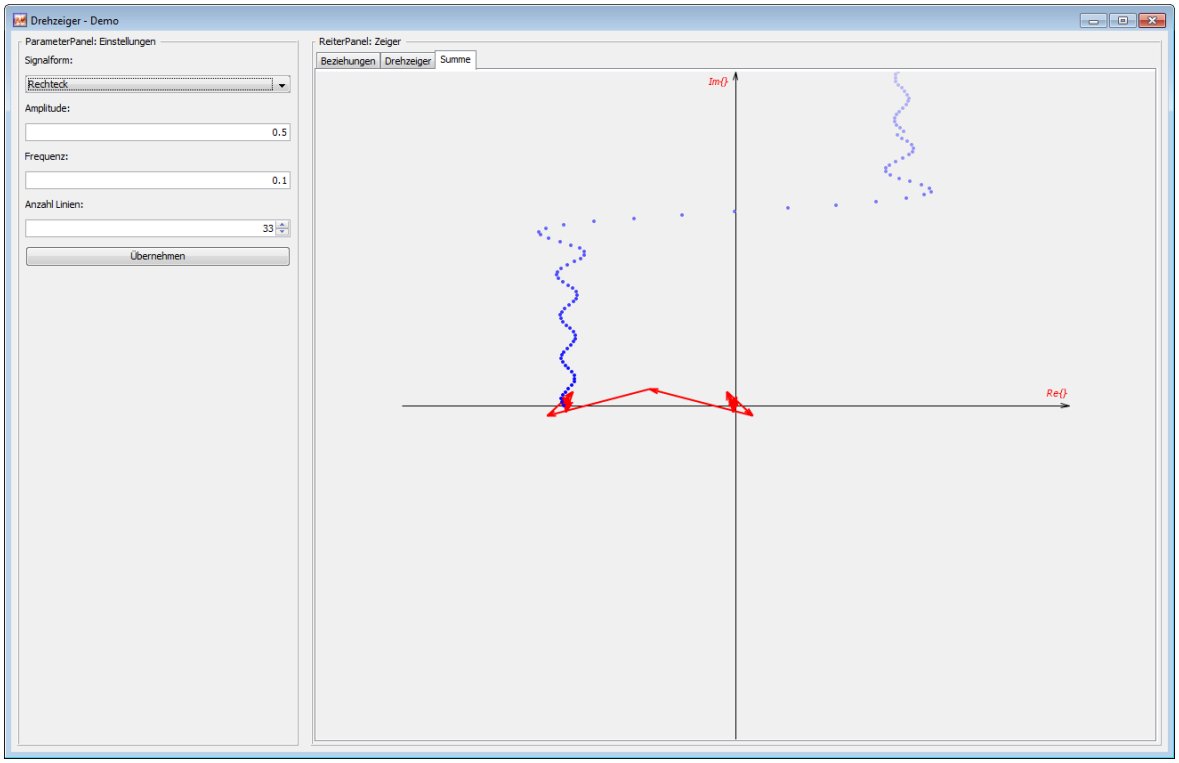
OOP2

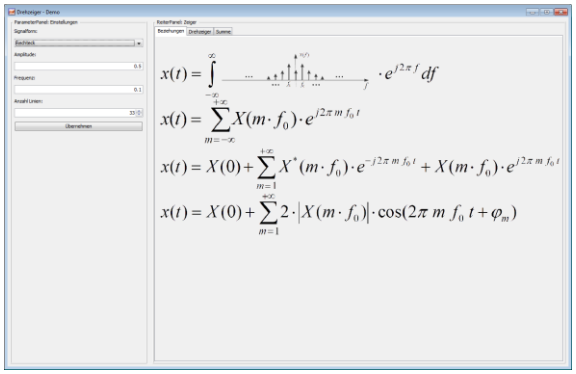
MODULSCHLUSSPRÜFUNG

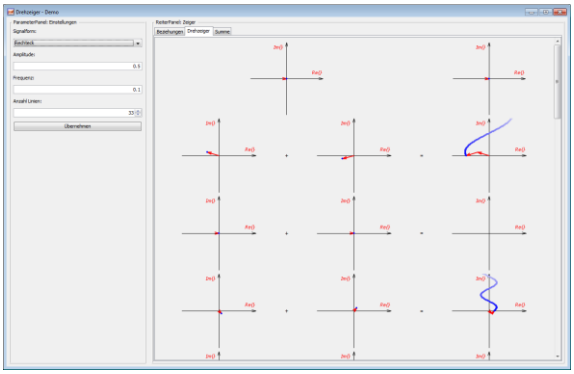
Beschreibung:

Ziel dieser Prüfung ist es, ein Demo - Tool zum Themenkreis Fourier-Transformation / Fourier-Reihe / Drehzeiger zu erstellen. Bei der komplexen Fourier-Reihe ergibt sich für das zugehörige Zeitsignal eine Summe von je paarweise gegenläufigen Drehzeigern plus Gleichanteil. Mit der Applikation sollen die Drehzeiger und deren Summe visualisiert werden. Festgelegt werden kann dabei die Signalform, die Amplitude des Signales, dessen Frequenz sowie die Anzahl verwendeter Spektrallinien.



Die Applikation ist im bekannten MVC-Entwurfsmuster programmiert und bedient sich zum Aufdatieren der Anzeige des Observer/Observable - Musters. Die View ist in ein Panel zur Eingabe der Parameter und ein Reiter - Pane mit der Anzeige verschiedener Grafiken unterteilt. Das Parameter - Panel umfasst eine JComboBox zur Festlegung der Signalform, zwei JTextField zur Spezifikation der Amplitude und der Grundfrequenz f_0, sowie einen JSpinner zur Wahl der Anzahl Linien. Übernommen werden die Werte mittels eines JButton.





Das Reiter - Pane beheimatet drei Reiter. Auf dem zum ersten Reiter zugehörigen Panel sind die grundlegenden mathematischen Zusammenhänge in Form eines Bildes zu sehen. Auf dem zum zweiten Reiter zugehörigen JScrollPane befindet sich ein Panel mit einer grossen Zahl von Zeiger-Plots, die die einzelnen Drehzeiger und deren paarweise Summe darstellen. Auf dem zum dritten Reiter zugehörigen Panel wird die Summe aller Drehzeiger dargestellt. Alle Zeiger sind animiert und zeigen den zur Signalform zugehörigen Verlauf.
Wie aus dem UML – Klassendiagramm in der Beilage ersichtlich, ist die Applikation nach dem klassischen MVC – Muster programmiert: Entsprechend dem ausgelösten Ereignis, wird seitens des GUI die zugehörige Methode des Controllers aufgerufen. Der Controller enthält die Logik der Applikation und bewirkt entsprechend die Änderung im Model. Das Model macht die Berechnungen und löst via Observer - Interface der *View* das Aufdatieren des User–Interfaces aus.
Das Model hat zwei Arrays mit je 127 komplexen Zahlen, die die Spektrallinien und die Drehzeiger beinhalten. Die Spektrallinien sind die per se komplexwertigen Amplituden der entsprechenden Drehzeiger. Die Drehzeiger enthalten den jeweiligen Momentanwert der Zeiger. Um das regelmässige Aufdatieren der Drehzeiger zu machen, verfügt das Model weiter über einen Swing-Timer, der dann die entsprechende Berechnung auslöst.

Die Aufgabenstellung führt Sie schrittweise zum Ziel, - das einfache Sammeln von billigen Punkten wird nicht honoriert! Halten Sie sich an die im Klassendiagramm festgelegten Bezeichnungen.

Aufgabe 1: Aspekt View: Klassen des User - Interfaces (~ 67 Pte.)

Die View ist wie beschrieben aufgebaut und erlaubt mittels *ParameterPanel* die Steuerung des Models. Das *ReiterPane* bringt die genannten Daten zur Darstellung.

a) Implementieren Sie die Klassen *View*, *ReiterPane*, *ParameterPanel, ZeigerPanel* und *SummePanel* gemäss Dokumentation und Beilage. Die benötigten Klassen *BildPanel* und *ZeigerPlot* sind bereits implementiert. **Der Challenge muss erst später implementiert werden.**

Aufgabe 2: Aspekt Controller: *Klasse zu Steuerung* (~ 2 Pte.)

Der *Controller* leitet allfällige Aufgaben an das *Model* weiter.

a) Implementieren Sie die Klasse *Controller* gemäss Dokumentation.

Aufgabe 3: Aspekt Model: Klassen zur Berechnung (~ 16 Pte.)

Das Model berechnet wie eingangs erwähnt, die zur Signalform zugehörigen Spektrallinien, deren Zahl durch *anzahlLinien* begrenzt ist sowie für jeden Zeitindex n die zugehörigen Drehzeiger.

Der Methode process() kommt die Berechnung der komplexwertigen Drehzeiger zu.
Für m gleich *-(anzahlLinien - 1)/2* bis und mit *+(anzahlLinien - 1)/2* gilt:

$$Z[m] = e^{j \cdot 2\pi \cdot m \cdot f_0 \cdot n \cdot T} \cdot X[m]$$

wobei Z[m] dem Drehzeiger drehZeiger$\left[m + \frac{\text{spektralLinie.length}-1}{2}\right]$ und $X[m]$ der Spektrallinie spektralLinie$\left[m + \frac{\text{spektralLinie.length}-1}{2}\right]$ entspricht. Der Term f_0 widerspiegelt die Grundfrequenz des periodischen Signals. Für negative m ergeben sich somit Zeiger die in negativer Richtung drehen, für positive m solche, die in positiver Richtung drehen.

a) Implementieren Sie die Methode process() gemäss den gemachten Angaben.

Der Methode setParameter() kommt die Berechnung der per se komplexwertigen Spektrallinien X[m] zu.

Rechtecksignal:
Für ein Rechteck ist für m gleich *-(anzahlLinien - 1)/2* bis und mit *+(anzahlLinien - 1)/2* die Spektrallinie X[m] gegeben durch:

$$X[m] = \begin{cases} 0 & \text{falls (m mod 2) gleich 0} \\ \frac{2 \cdot A}{|m| \cdot \pi} \cdot (-1)^{\frac{|m|-1}{2}} & \text{sonst} \end{cases}$$

Wobei A der Amplitude entspricht.

b) Implementieren Sie die Methode setParameter() gemäss den gemachten Angaben für das Rechtecksignal.

Aufgabe 4: Challenge League: (~ 22 Pte)

a) Implementieren Sie die Methode setParameter() gemäss den gemachten Angaben für die nachfolgenden Signale sinngemäss.

Dreiecksignal:

$$X[m] = \begin{cases} 0 & \text{falls (m mod 2) gleich 0} \\ \frac{2 \cdot A}{m^2 \cdot \pi} & sonst \end{cases}$$

Sägezahnsignal:

$$X[m] = \begin{cases} j\frac{A}{|m| \cdot \pi} & \text{für m kleiner 0} \\ 0 & \text{für m gleich 0} \\ -j\frac{A}{|m| \cdot \pi} & \text{für m grösser 0} \end{cases}$$

Aus der Sicht des Benutzers ist es angenehm, wenn sich Spinner mittels Mausrad ändern lassen.

b) Ergänzen Sie den Code so, dass sich der Spinner in der Klasse *ParameterPanel* mit dem Mausrad verändern lässt.

Aufgabe 5: Clean Up (~ ? Pte)

Gelegentlich geht etwas vergessen …

a) Ergänzen Sie den Code um noch allfällig fehlenden Code.