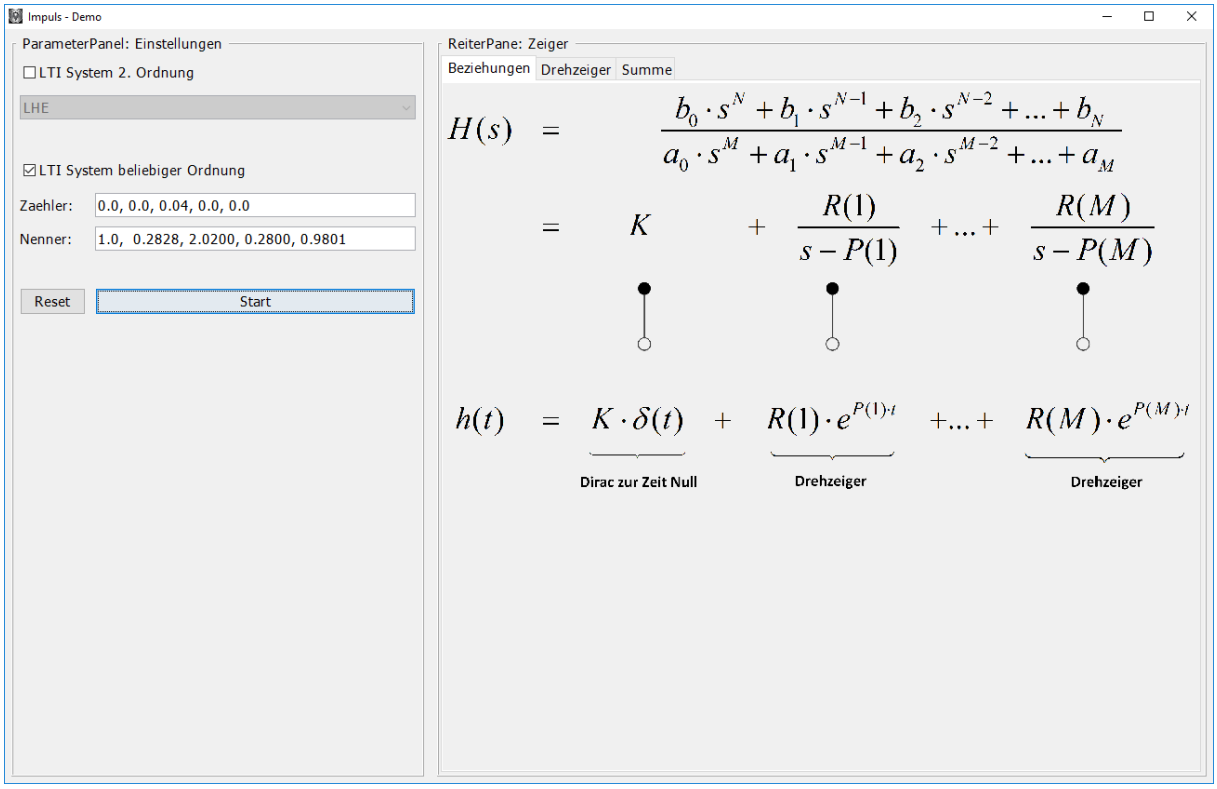
OOP2

PRÜFUNG I

Beschreibung:

Ziel diese Prüfung ist es, ein Demo-Tool zum Thema Impulsantwort eines LTI[1]-Systems zu erstellen. Die Übertragungsfunktion H(s) kann mittels Residuen-Rechnung in eine Konstante K und eine Summe der einzelnen Partialbrüche aufgeteilt werden[2]. Die Konstante und die einzelnen Partialbrüche können dann einfach in den Zeitbereich transformiert werden. Die Residuen R und die Pole P können generell komplex sein. Die zugehörigen Terme im Zeitbereich können als Drehzeiger verstanden werden. Die Anwendung dient zur Visualisierung der Drehzeiger und ihrer Summe. Der Typ des LTI-Systems kann definiert werden: Am Beispiel von LTI-Systemen zweiter Ordnung kann die Wirkung von Polen in der linken Laplace-Halbebene (LHE), Polen auf der $j\omega$ −Achse (*JW*) und Polen in der rechten Halbebene (RHE) gezeigt werden. Darüber hinaus können LTI-Systeme beliebiger Ordnung durch ihre Zähler- und Nenner-Polynome spezifiziert werden.

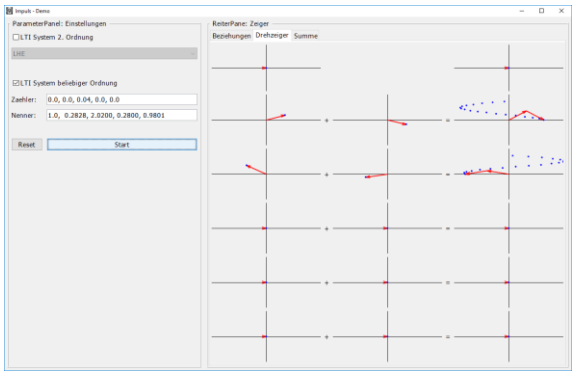


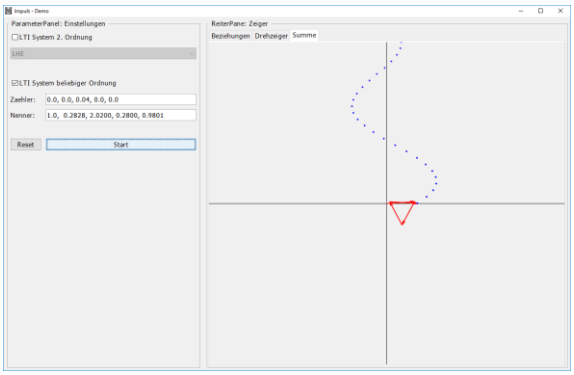
Die Anwendung ist im bekannten MVC-Design-Pattern programmiert und verwendet das Observer/Observable-Pattern zur Aktualisierung der Anzeige. Die Ansicht ist unterteilt in ein Panel zur Eingabe der Parameter und ein

[1] Linear-Time-Invariant System

[2] Es wird davon ausgegangen, dass Zähler- und Nennerpolynom nur einfache Nullstellen enthalten und dass die Ordnung des Zählerpolynoms N kleiner gleich der Ordnung des Nennerpolynoms ist.

Tab - Panel mit der Darstellung verschiedener Grafiken. Das Parameterfeld enthält zwei Kontrollkästchen (*JCheckBox*) zur Auswahl des *LTI-Systems 2. Ordnung* oder eines *LTI-Systems beliebiger Ordnung*. Bei LTI Systemen 2. Ordnung kann zwischen LHE, JW und RHE über ein JComboBox gewählt werden. Wird ein LTI-System beliebiger Ordnung gewählt, können in den beiden Textfeldern Zähler- und Nenner-Polynome eingegeben werden. Die Animation der Drehzeiger kann mittels **Reset** zurückgesetzt und mittels **Start** gestartet werden.





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 befindet sich ein Panel mit einer 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 Verlauf der zugehörigen Impulsantwort des Systems[3].

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 verfügt über einen *Swing-Timer*, der im Abstand von 50ms die Methode *actionPerformed*() wiederholt aufruft. Ist das System nicht angehalten, wird in *actionPerformed*() die Methode *processing*() aufgerufen, die benötigte Berechnung gemacht, neu dargestellt und das Attribut n um eins erhöht. Mittels den Methoden *setSystemType*() resp. *setUTF*() wird die Übertragungsfunktion gesetzt und die Partialbruch-Zerlegung ausgeführt.
Hinweis: Das Frame-Work verfügt über die bekannte TraceV2 – Klasse. In der Methode main() in der Impuls-DemoApplikation wird als erstes `TraceV2.mainCall(false, false, false);` aufgerufen. Damit lassen sich folgende Informationen ein- resp. ausschalten:

```
public static void mainCall(boolean show, boolean showBuild, boolean showMethode) {}
```

show:	generelles on/off.
showBuild:	Attribut-Initialisierung und Konstruktor-Aufrufe on/off.
showMethode:	on/off der Methodenaufrufe.

Beispiele:
TraceV2.*mainCall*(**false**, **true**, **true**); -> Nichts wird dargestellt.
TraceV2.*mainCall*(**true**, **true**, **false**); -> Attribut-Init. und Konstruktor-Aufrufe werden dargestellt.
TraceV2.*mainCall*(**true**, **false**, **true**); -> Methodenaufrufe werden dargestellt.
TraceV2.*mainCall*(**true**, **true**, **true**); -> Alles wird dargestellt.

[3] Sinngemäss zur Darstellung in Matlab, wird ein allfälliger Dirac-Stoss zur Zeit Null nicht gezeigt.

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 (~ 63 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*, *ParameterPanel, ReiterPane*, gemäss Dokumentation und Beilage. Die benötigten Klassen *BildPanel* und *Zeigerpanel* und *SummePanel* sind bereits implementiert.

Aufgabe 2: Aspekt Controller: *Klasse zu Steuerung* (~ 5 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 (~ 43 Pte.)

a) Implementieren Sie den Konstruktor der Klasse und die Methode *actionPerformed*().

Das Model bewerkstelligt, wie eingangs erwähnt, die Partialbruchzerlegung. Dazu steht die Klasse *Residue* zur Verfügung: *Residue* berechnet mittels der Klasse *Matlab* die Konstante *K*, die komplexwertigen Residuen *R* und die zugehörigen komplexwertigen Pole *P*. Um zur richtigen Darstellung zu gelangen, muss die Länge von *R* und *P* immer ungerade sein: Falls *R* und damit *P* gerader Länge sind, wird mittels *Matlab.concat*() sowohl bei *R* wie auch bei *P* eine komplexe Null hinzugefügt. Des Weiteren müssen *R* und *P* richtig sortiert vorliegen. Dazu gilt es die Methode *bubbleSort*() zu implementieren.

b) Implementieren Sie die Methoden *setSystemType*(), *bubbleSort*() und *setUTF*() gemäss den gemachten Angaben.

Die Methode *processing*() berechnet die Drehzeiger. Der m-te Drehzeiger d[m] ist dabei gegeben durch

$$d[m] = R[m] \cdot e^{P[m] \cdot n \cdot T}$$

T bezeichnet dabei ein Zeitintervall und n der Laufindex.

c) Implementieren Sie die Methode *processing*() gemäss Angaben.
d) Implementieren Sie die verbleibenden Methoden.

Aufgabe 4: Clean-up (~? Pte)

Gelegentlich geht etwas vergessen …

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