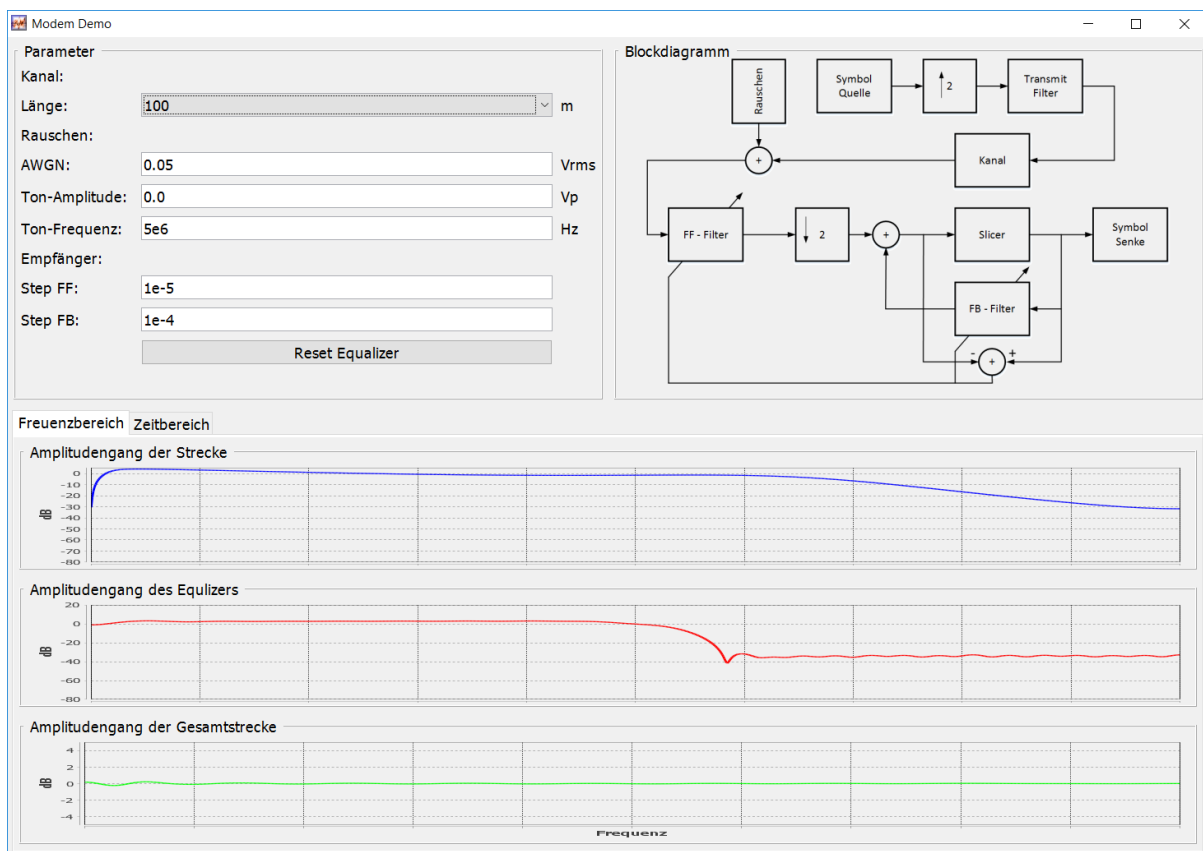


OOP2

MSP FS2018

Beschreibung:

Ziel dieser Prüfung ist es, ein Programm zur Visualisierung des Adaptionsvorganges bei einem Modem zu schreiben. Adaptive Systeme erlauben bei der digitalen Signalverarbeitung die Anpassung an ein unbekanntes oder sich änderndes System. Bei einem Modem wird mittels adaptiven Filtern Das Rauschen weiss gemacht und die Übertragungsstrecke egalisiert. Das zugehörige Blockdiagramm ist oben rechts zu sehen.



Die Applikation ist im klassischen Model - View/Controller und Observable/Observer Entwurfsmuster, gemäss Klassendiagramm in der Beilage, implementiert.

Die *View* beinhaltet das *ParameterPanel* zur Einstellung der Parameter, das *BDGPanel* mit dem Blockdiagramm sowie ein *ReiterPanel* mit der *ViewFrequenzbereich* zur Darstellung dreier Amplitudengänge und der *ViewZeitbereich* zur Darstellung dreier Impulsantworten. Amplitudengänge und Impulsantworten werden mittels der Klasse *XYPlot* repräsentiert. Das zugehörige Layout ist in der Beilage zu finden.

Der *Controller* liest die Information aus dem *ParameterPanel* aus und delegiert allfällige Aufgaben an das *Model*. Der *Controller* verfügt dazu sowohl über die *View* wie auch über das *Model*.

Das *Model* beinhaltet die *SymbolQuelle*, die mittels *ScheduledExecutorService* in regelmässigen Abständen die Methode *processSymbol()* des *Models* aufruft. Es verfügt weiter, entsprechend dem Blockdiagramm, über die Verzögerung *delay*, das Transmit- und Kanalfilter in der Form von *FIRFilter* und

die beiden adaptiven Filter *FFilter* und *FFilter*, in der Form von *LMSFilter*. *FIRFilter* basiert auf *Fil-*
ter, das *LMSFilter* ist ein Derivat der Klasse *FIRFilter*. Für die notwendigen Berechnungen steht die
Klasse *Matlab* und weitere Klassen zur Verfügung. Das zugehörige Klassendiagramm ist in der Beilage
zu finden.

Die Aufgabenstellung führt Sie schrittweise zum Ziel. Folgen Sie daher beim Lösen der Prüfung der
Aufgabenstellung.

Achten Sie darauf, dass Ihr Code kompilierbar bleibt!

Aufgabe 1: Aspekt View: Klassen des User - Interfaces (~ 48 Pte.)

View:

Die View ist wie beschrieben aufgebaut und erlaubt mittels Reiter zwischen der Darstellung des Fre-
quenzbereichs und des Zeitbereichs zu wechseln. Allfällige Updates seitens des Models gibt die View
entsprechend weiter.

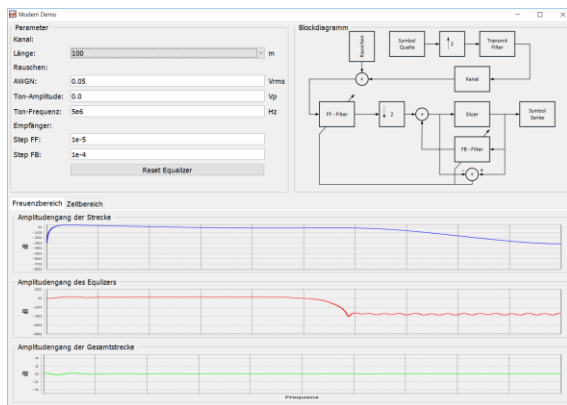


Abbildung 1: Frequenzbereich

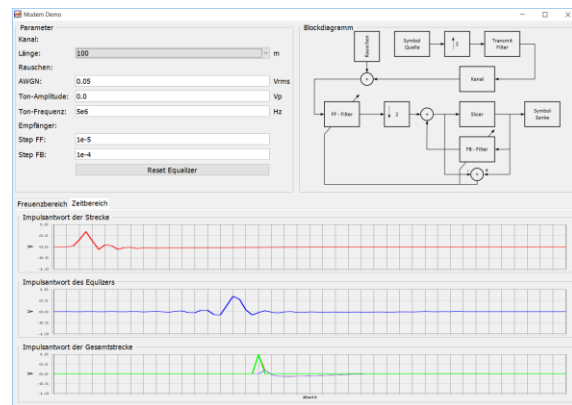


Abbildung 2: Zeitbereich

- Implementieren Sie die Klassen *View*, *ParameterPanel* und *ReiterPanel* gemäss Dokumentation und Beilage. Die Klassen *BDGPanel*, *ViewFrequenzbereich* und *ViewZeitbereich* sind bereits gegeben.

Aufgabe 2: Aspekt Controller: Klasse zu Steuerung (~ 8 Pte.)

Controller: Der *Controller* leitet allfällige Aufgaben an das *Model* weiter. Die allenfalls benötigten
Informationen werden vom *Controller* via *View* ausgelesen. Der Controller verfügt über einen *Schedule-*
dExecutorService der nach einer anfänglichen Verzögerung von 1500 ms alle 100 ms ausgeführt wird.

- Implementieren Sie die Klasse *Controller* gemäss Dokumentation und Pseudo-Code.

Aufgabe 3: Aspekt Model: Klassen zur Berechnung (~ 41 Pte.)

Model: Das *Model* hat, wie eingangs erwähnt, die *SymbolQuelle* sowie die weiteren Elemente des
Blockdiagramms. Notwendige Berechnungen werden allenfalls mit der Klassen *Matlab* durchgeführt.

Die Signalverarbeitung wird gemäss Anhang in der Methode *processSymbol()* durchgeführt.

- Implementieren Sie die Klasse *Model* gemäss Beschreibung und Pseudo-Code.
- Implementieren Sie die Klasse *SymbolQuelle* gemäss Beschreibung und Pseudo-Code.

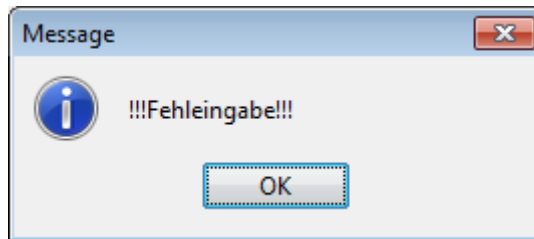
Aufgabe 4: Clean Up (~ ? Pte)

Gelegentlich geht etwas vergessen ...


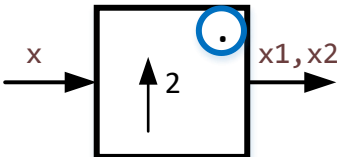
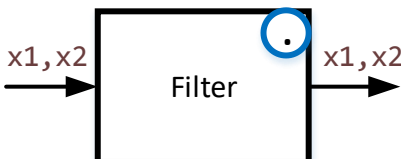
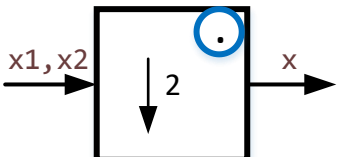
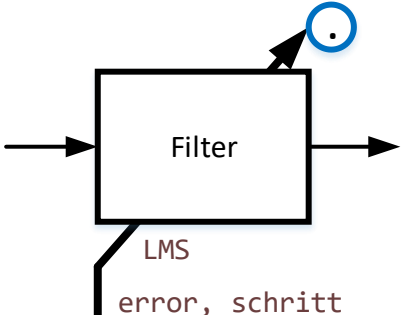
- a) Ergänzen Sie den Code um noch allfällig fehlenden Code.

Aufgabe 5: Challenge League (5 Pte)

- a) Ändern Sie den Controller der Art, dass bei einer Fehleingabe in einem der Textfelder der folgende Dialog erscheint:



Erläuterungen zum Blockdiagramm

Signalverarbeitungsblock	Funktion
	Abfolge: Die Zahlen in den Kreisen zeigen die Abfolge der Signalverarbeitungsschritte.
	Upsampler: Bildet den Eingangswert x in $x_1 = x, x_2 = 0$ ab.
	Filter: Filtert die Eingangswerte x_1, x_2 durch zweimaligen Aufruf der entsprechenden Methode des Filters und schreibt die Resultate gleich wieder in die gleichen Variablen x_1, x_2 .
	Downsampler: Verwirft den Ersten der beiden Eingangswerte und verwendet den Zweiten weiter.
	LMS-Algorithmus: Ruft die Methode <code>lms()</code> des entsprechenden Filters mit dem Fehler <i>error</i> und der entsprechenden Schrittweite auf.

Korrigenda