Assignment 2: Modellierung und Differentialgleichungen

Angewandte Modellierung 25

Carl Colmant

May 3, 2025

Exercise 1. Matlab Adventure

Zuerst soll die Differentialgleichung $30\cos(t) = LI'(t) + RI(t) = I'(t) + 5I(t)$ gelöst werden mit den Bedingungen $I(0) = 0$. Dazu benutze ich folgenden Matlab Code:

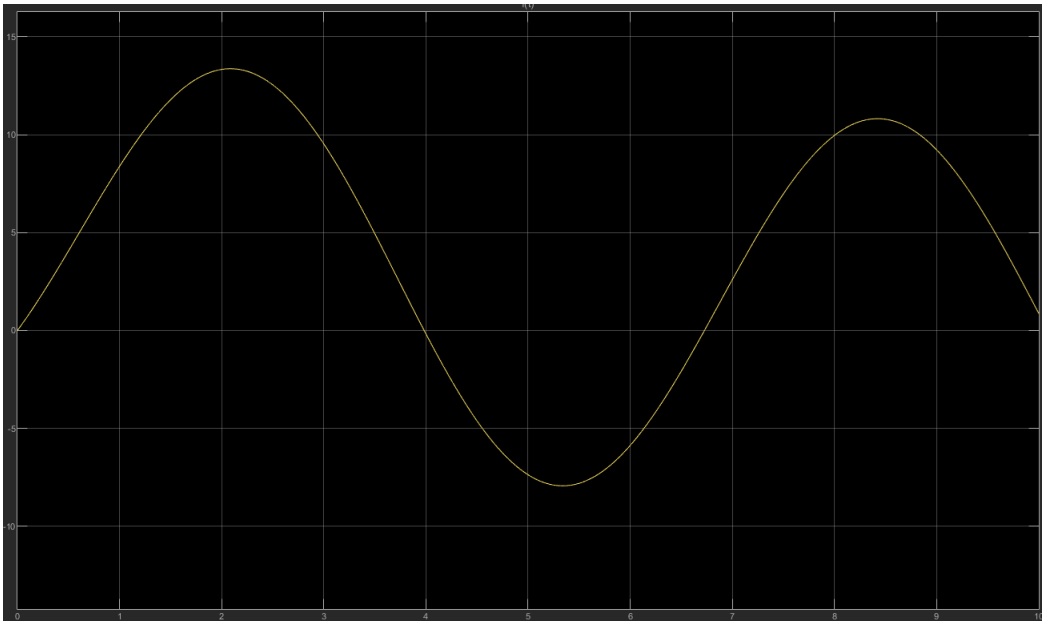
```
syms t I(t)

eq = 30*cos(t) == diff(I(t), t) +5*I(t);

cond= I(0) == 0;

sol = dsolve(eq, cond);
```

Dann soll die Gleichungslösung in punkt 0.5 ausgewertet werden. Und zu guter letzt die Gleichungslösung über den Zeitbereich $[0, 10]$ dargestellt werden. Dazu benutzte ich folgenden Code:

```
x = double(subs(sol, t, 0.5));

%Plot von Chat GPT erstellt
figure
fplot(sol, [0,10], 'LineWidth', 2) %
xlabel('t')
ylabel('I(t)')
```

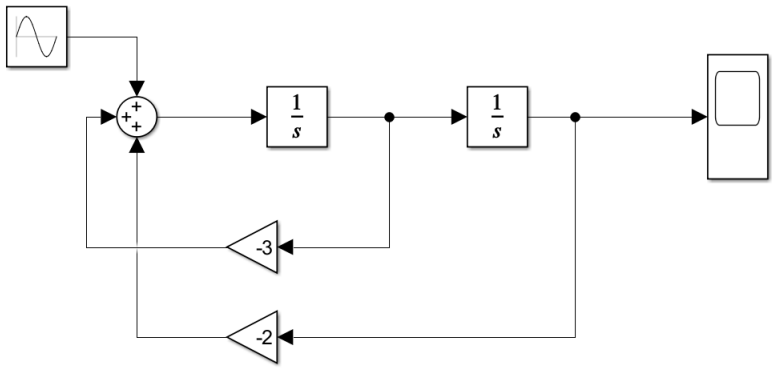
Exercise 2. Resistor–inductor circuit in Simulink

Hier sollte ein etwas complexes System aufgebaut werden. Ich hab die Blöcke genau wie in der Aufgabe beschrieben angeordnet den Summenblock mit dem Minuszeichen unten und den Pluszeichen oben ausgestattet. Da kommt dann folgender Graph über 10 sekunden simulationszeit raus:

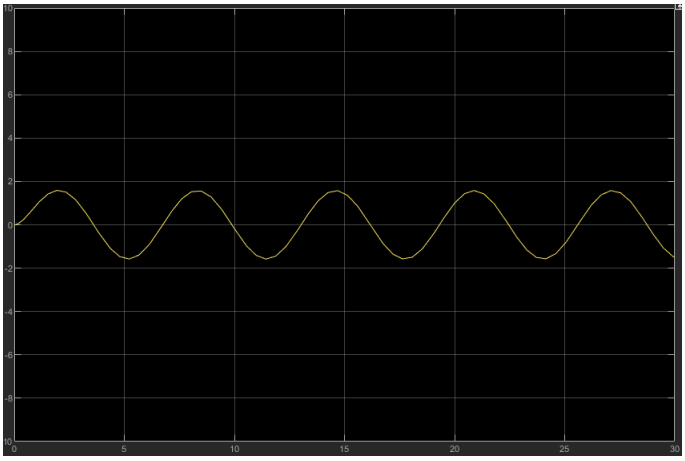


Exercise 3. RLC model in Simulink

Hier ist die Simulation fast identisch aufgebaut, diesmal habe ich aber die Minuszeichen in den Gain-block getan:

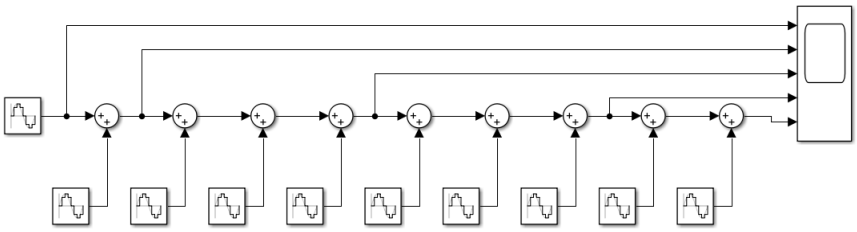


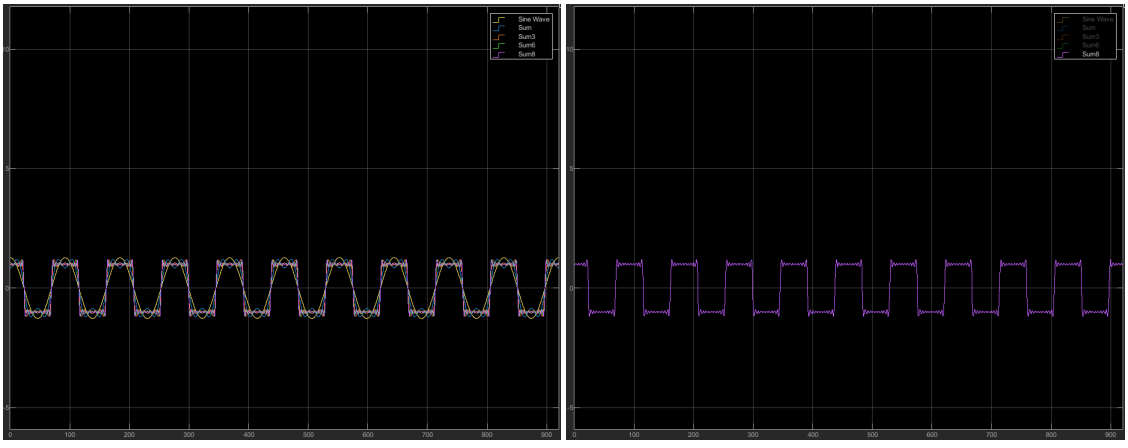
Ich habe die Simulation 30 sekunden laufen lassen und das Ergebnis sieht so aus:



Exercise 4. Fourier series of periodic signals

Ich habe das Modell so wie in der Aufgabe gezeigt aufgebaut. Man soll die Phase in den Sinus Blöcken auf $\frac{\pi}{2}$ das liegt daran dass die Funktion die approximiert werden soll aus cosinus funktionen besteht. Wenn wir alle Sinus Blöcke korrekt eingestellt haben, lassen wir die Simulation für 920 zeiteinheiten laufen. Es entsteht folgendes Signal:





So mehr summen man einbaut desto eckiger und grader wird das Signal. Das heißt die

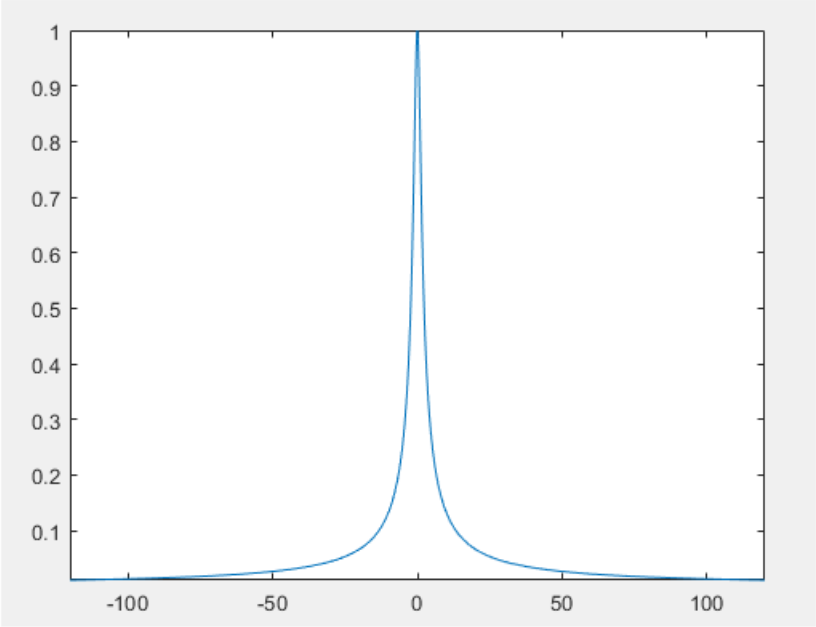
Simulation approximiert die Funktion korrekt.

Exercise 5. Low pass filter

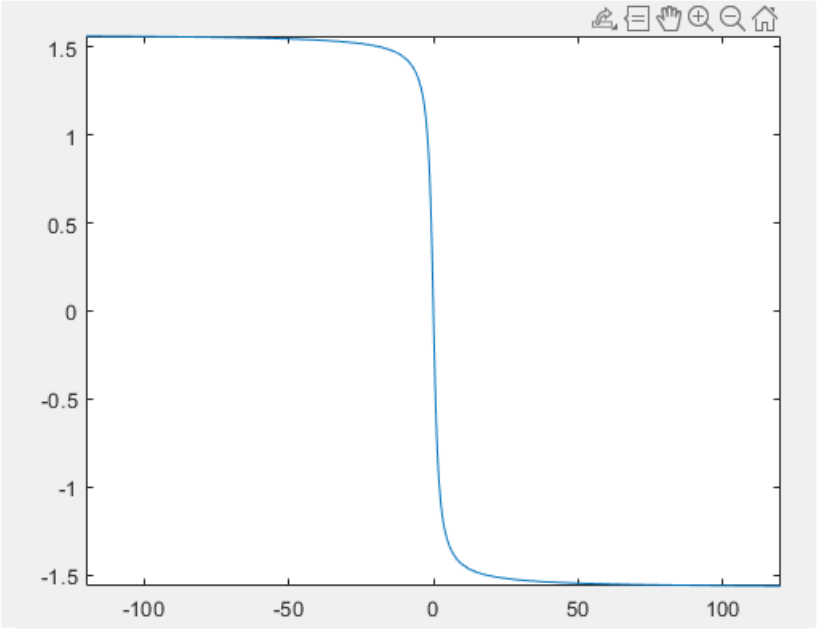
Wenn man die Funktion die in der Aufgabe aufgeführt ist in Matlab umsetzt erhält man:

```
%numeric calc
syms omega

H(omega) = 1/(1+ 1j*omega* 500*(1492e-6)); %Funktion

amp(omega) = abs(H(omega));

amp_num = matlabFunction(amp); %turn symbolick funktion into numeric one

amp(0); % Amplitude at omega = 0

phase = matlabFunction(angle(H));

phase(0) % Phase at omega = 0

%plot
figure
fplot(amp(omega), [-120, 120]);           % Betrag (Amplitude)
figure
fplot(angle(H(omega)), [-120, 120]);         % Phase
```

Die Funktion sieht dann so aus:



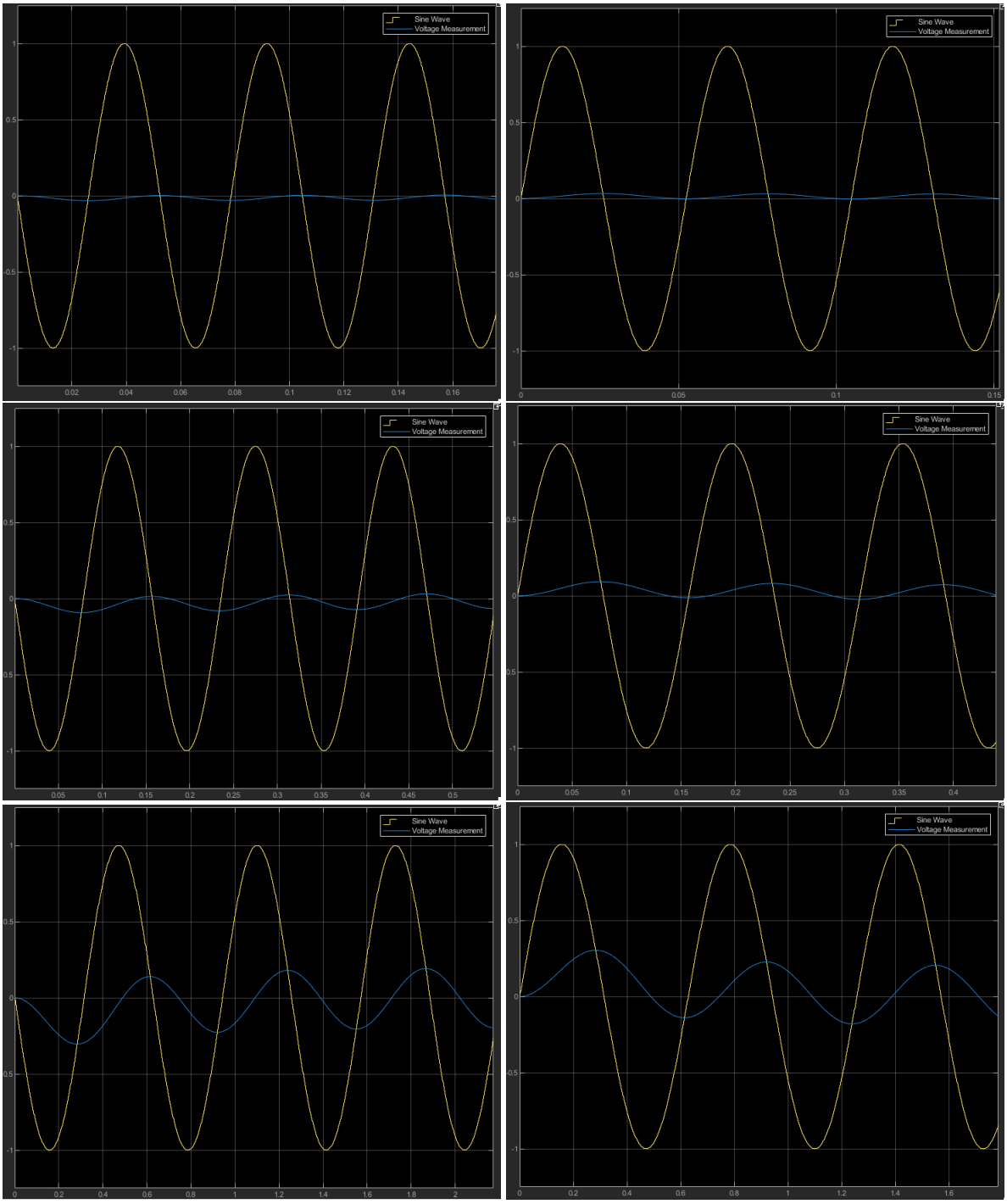
Die Funktion sieht deshalb so aus weil sie die niedrigen Frequenzen unverändert lassen soll (mit 1 multipliziert) und die hohen Frequenzen gedämpft werden sollen(mit <1 multipliziert). Man sieht dass aber noch nicht so gut bei einem so kleinen omega. Hier noch die Phase über omega:



Exercise 6. Low pass filter in Simulink

a)

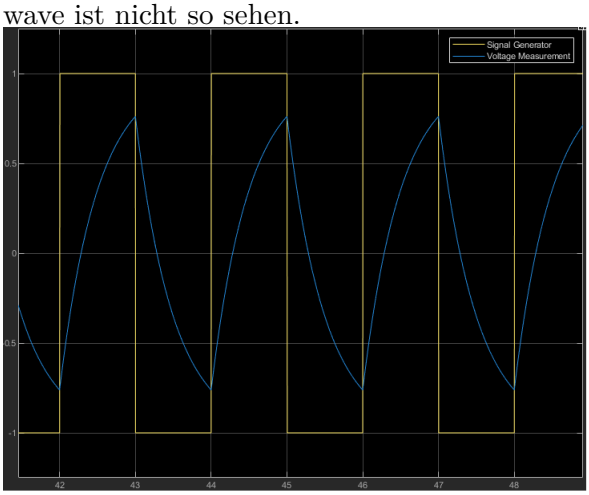
Die Frequenz der Sinusfunktion soll in der Aufgabe nun auf $\omega^* t$ gesetzt werden. Hier nun die Ausgaben der Simulation für Frequenzen $-120, -40, -10, 10, 40, 120$:



Wir sehen das die entstehende Welle nicht gans symetrisch zur x-Achse ist also ein höherer teil ist unter 0 oder über null abhängig davon ob ω positiv oder negativ ist. Ausßerdem ist die Dämpfung etwas stärker als bei der Matlab Berechnung.

b)

Ich habe die Simulation mit dem Signal-Generator erweitert. Für die Theoretische berechnung muss man wissen dass $\omega = 2\pi f$ ist. Das heißt bei einer Frequenz von $\frac{1}{2}$Hz ist $\omega = \pi$. Daraus ergibt sich ein Wert von 0.3925 für die Amplitude.
Das können wir aber in der simulation weil diese keine Sinuswelle sondern eine Square-wave ist nicht so sehen.



Diese Gedämpfte Welle hat eher einen Amplitude von 0.6. Das kann natürlich an einem Fehler meinerseits liegen.

Github