Simulation des Betriebs eines Pumpspeicherwerks

Hausarbeit

Khaled Aboelroos
391756

Nachhaltige Rohstoff- und Energieversorgung
Einführung in Matlab

Prof. Dr. Elisabeth Clausen
RWTH Aachen

07.09.2020

1 Skript

```
% Pfad und Dateien manuell eingeben
% Wenn file_paths oder file_extensions leer sind werden diese durch
% Standard-Werte ergaenzt
file_paths      = "";
file_names      = ["Datensatz_1", "Datensatz_2", "Datensatz_3", "Datensatz_4"];
file_extensions = "";

% Bedingungen fuer falsche Daten festlegen
frequency_condition = @(frequency) isnan(frequency) | frequency < 45 | frequency > 55;
intraday_condition  = @(price) isnan(price);

% Erstelle vollen string aus Dateipfad, Dateiname und Dateiendung;
% vervollstaendige unvollstaendige Eingaben
full_file_paths = generate_file_paths(file_paths, file_names, file_extensions);

% Datenstruct erstellen
data = struct;

% Gehe alle Dateien der Reihe nach durch und wende das Program an die Daten
% an
for f=1:length(file_names)
    % Extrahiere Frequenz und Intraday Price aus dem Datensatz
    [frequency , intraday_price] = read_file(full_file_paths(f));
    % Speicher Laenge von Datensatz
    array_length = length(frequency);

    % Speicher alle Daten
    data.(file_names(f)).frequency      = frequency;
    data.(file_names(f)).intraday_price = intraday_price;

    % Frequenz und intraday price korrigieren
    frequency      = correct_data(frequency,     frequency_condition);
    intraday_price = correct_data(intraday_price, intraday_condition);

    % Glaettung der Frequenz
    smoothed_frequency = smoothdata(frequency, 'movmedian', 'SmoothingFactor',0.7);

    % Maxima / Minima von geglaetteten Daten bestimmen
    [max_mask , min_mask] = extrema(smoothed_frequency);


    % Speicher korrigierte Daten
    data.(file_names(f)).corrected_frequency      = frequency;
    data.(file_names(f)).corrected_intraday_price = intraday_price;

    % Speicher geglaettete Frequenz
    data.(file_names(f)).smoothed_frequency = smoothed_frequency;
```

```
% Maxima / Minima speichern
data.(file_names(f)).max_mask = max_mask;
data.(file_names(f)).min_mask = min_mask;


% Alles plotten wie in fig 2
plot_frequency(frequency, smoothed_frequency, max_mask, min_mask, file_names(f));


% Erstelle arrays fuer die Fuellstaende und den Kontostand
fill_level_upper_basin = zeros(array_length,1);
fill_level_lower_basin = zeros(array_length,1);
account_balance = zeros(array_length,1);

% Initialisiere Fuellstaende
fill_level_upper_basin(1) = 250E3;
fill_level_lower_basin(1) = 150E3;


% plotte alles nach fig 3
plot_status(fill_level_upper_basin, fill_level_lower_basin, ...
          frequency, intraday_price, ...
          account_balance, file_names(f));


% bestimme fuer alle Frequenz/Intraday-Preispaare ob Pumpe oder Turbine
% verwendet werden soll und aktualisiere Kontostand entsprechend
for k=1:array_length-1
    % bestimme um wie viel sich der Fuellstand des unteren Beckens
    % aendert
    filling_change_lower = calculate_filling_change_lower(fill_level_upper_basin(k), ...
                                          fill_level_lower_basin(k), ...
                                          intraday_price(k));

    % Berechne Gewinn / Kosten fuer das aktuelle Frequenz/Intraday-Preispaar
    account_change = calculate_earnings(filling_change_lower, ...
                              frequency(k), intraday_price(k), ...
                              max_mask(k), min_mask(k));


    % Aktualisiere Fuellstand der Becken fuer den naechsten Schritt
    fill_level_lower_basin(k+1) = fill_level_lower_basin(k) + filling_change_lower;
    fill_level_upper_basin(k+1) = fill_level_upper_basin(k) - filling_change_lower;

    % Aktualisiere Kontostand
    account_balance(k+1) = account_balance(k) + account_change;
end

% Speicher Ergebnisse in struct
data.(file_names(f)).fill_level_upper_basin = fill_level_upper_basin ;
data.(file_names(f)).fill_level_lower_basin = fill_level_lower_basin ;
data.(file_names(f)).account_balance        = account_balance;
```

```
    % plotte alles nach fig 4
    plot_status(fill_level_upper_basin, fill_level_lower_basin, ...
              frequency, intraday_price, ...
              account_balance, ...
              file_names(f));

end
```

1.1 Variablenerstellung

Zuerst wird dem Nutzer die Option gegeben, einen eigenen Pfad für Datensätze einzutippen.
Sollte kein Pfad angegeben werden, werden die Datensätze in %PROGRAM_PATH%/Datensatz gesucht.
%PROGRAM_PATH% bezeichnet hierbei den Order in welchem sich program.m befindet.
Sollte kein Dateiformat gegeben sein, werden die drei verfügbaren Dateiformate (“.csv”,”.json”,”mat”) der Reihe nach angewandt.

Danach werden die Bedingungen für fehlerhafte Werte festgelegt. `frequency_condition` ist erfüllt wenn die Frequenz fehlerhaft ist. Also falls der Wert `NaN` ist, oder kleiner als 45 Hz oder größer als 55 Hz ist.
Der Intraday-Price ist nur dann fehlerhaft wenn der Wert `NaN` ist, dann ist `intraday_condition` erfüllt.

```
>> data.Datensatz_1

ans =

  struct with fields:

                  frequency: [1000×1 double]
              intraday_price: [1000×1 double]
         corrected_frequency: [1000×1 double]
    corrected_intraday_price: [1000×1 double]
          smoothed_frequency: [1000×1 double]
                    max_mask: [1000×1 logical]
                    min_mask: [1000×1 logical]
      fill_level_upper_basin: [1000×1 double]
      fill_level_lower_basin: [1000×1 double]
             account_balance: [1000×1 double]
```

Abbildung 1: Datenfelder die im Struct “data” für jeden Datensatz gespeichert werden.

Ein Datastruct nach Abbildung 1 wird erstellt, in dem für jeden Datensatz alle relevanten Datenreihen gespeichert werden.
Diese umfassen Frequenz (Rohdaten, korrigierte und geglättete Daten), Intraday-Preis (Roh- und korrigierte Daten), eine Maske der Maxima und Minima und die berechneten Füllstandverläufe des oberen und unteren Beckens sowie den Verlauf des Kontostandes.
Das Skript geht nun alle gegebenen Dateien durch und wendet wertet diese aus.

1.2 Einlesen

Die Frequenz- und Intraday-Preisdaten werden aus dem entsprechenden Datensatz extrahiert.
Hierfür wird die Funktion `read_file(filename)` benutzt, der Parameter filename umfasst den gesamten Pfad der Datei, zum Beispiel "`C:/Pfad/Datensatz_1.csv`".
Die Rohdaten werden in dem zuvor erstellten Datenstruct gespeichert.

1.3 Aufbereitung

Nach der Speicherung alle Daten werden die rohen Frequenz und Intraday-Preisdaten mittels der Funktion `correct_data(data,condition)` korrigiert.
Der Funktion werden die Daten und die Bedingungen für fehlerhafte Werte in den Daten übergeben, zurück werden die korrigierten Daten gegeben.
Die korrigierte Frequenz wird weiter mit dem Befehl `smoothdata(frequency, 'movmedian', 'SmoothingFactor',0.7)` geglättet und gespeichert.
Mittels der `extrema(frequency)` Funktion werden die lokalen Maxima und Minima der geglätteten Frequenz ermittelt.
Es wird eine logische Maske der Maxima und Minima in `max_mask` und `min_mask` gespeichert.
Zuletzt werden die gesamten aufbereiteten Daten im Datenstruct des Datensatzes gespeichert.

1.3.1 Erste Visualisierung/Abbildung

Als nächstes wird die korrigierte Frequenz gegen die geglättete Frequenz nach Abbildung 2 in der Aufgabenbeschreibung abgebildet.

Die entstehende Abbildung für Datensatz 1 ist in Abbildung 2 dargestellt. Wegen des höheren Kontrastes wurde die geglättete Frequenz in rot gezeichnet.

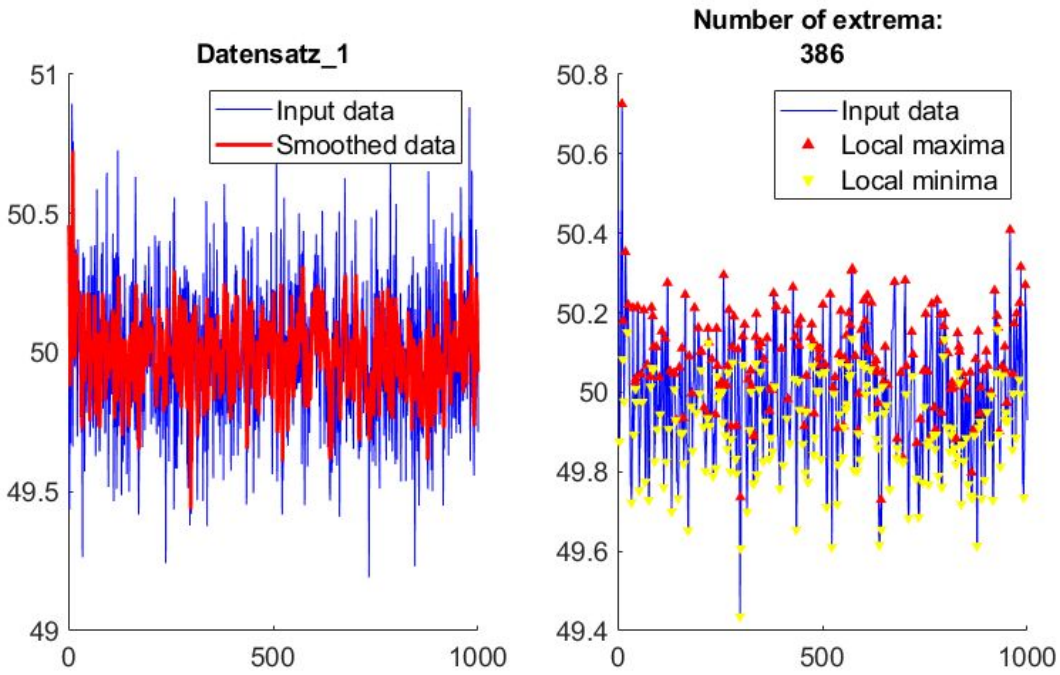


Abbildung 2: Links: Korrigierte Frequenz gegen geglättete Frequenz von Datensatz 1. Rechts: Geglättete Frequenz mit markierten lokalen Maxima und Minima.

1.4 Verarbeitung

Zunächst werden Arrays für die Füllstandverläufe des oberen und unteren Beckens und des Kontostandverlaufs initialisiert.
Die Anfangswerte die Becken werden gesetzt, 250 000m³ für das obere und 150 000m³ für das untere Becken.

1.4.1 Zweite Visualisierung/Abbildung

Hier werden die Anfangswerte vom Füllstand, Kontostand, oberen und unteren Becken sowie die aufbereitete Frequenz und Intraday-Preis nach Abbildung 3 in der Aufgabenbeschreibung abgebildet.
Hierfür wird die Funktion
`plot_status(fill_level_upper_basin, fill_level_lower_basin, frequency, intraday_price, account_balance, figure_title)`
benutzt.
Abbildung 3 zeigt diese Visualisierung für Datensatz 1.

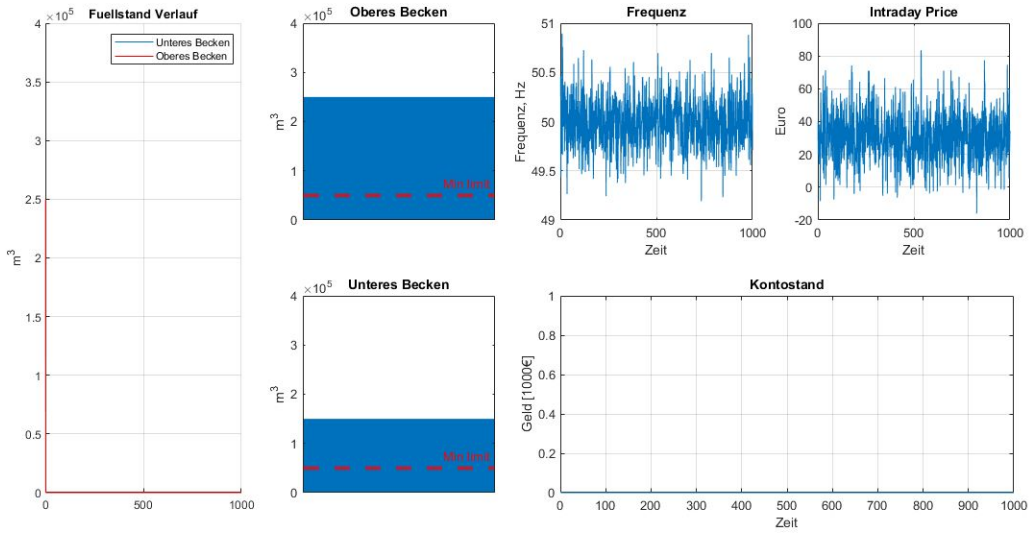


Abbildung 3: Visualisierung der Anfangswerte von Füllstand, Kontostand, oberen und unteren Becken sowie die aufbereitete Frequenz und Intraday-Preis für Datensatz 1

1.5 Entscheidungsfindung

Hier wird eine Schleife über alle Frequenz/Intraday-Preispaare durchgeführt. Entschieden wird, ob die Pumpe oder die Turbine verwendet werden soll. Hierfür wird die Funktion `calculate_filling_change_lower(fill_level_upper_basin, fill_level_lower_basin, intraday_price)` benutzt.
Diese Funktion ermittelt mithilfe der Füllstände der Becken und des aktuellen Intraday-Preises, um wie viele Kubikmeter Wasser der Füllstand des unteren Beckens geändert wird.
Hierbei kann das Becken um 1000m³ befüllt oder geleert werden, oder der Füllstand bleibt unverändert.

Der Gewinn oder die anfallenden Kosten werden mit der Funktion `calculate_earnings(filling_level_change_lower, frequency, intraday_price, is_max, is_min)` berechnet und in der Variable account_change gespeichert. Es wird die Füllstandsänderung des unteren Beckens und der aktuelle Intraday-Preis benutzt um diese Gewinne oder Kosten zu errechnen. Mit der aktuellen Frequenz und zwei logischen Variablen über das aktuelle Extrema der geglätteten Frequenz (is_max, is_min) werden Boni bestimmt.

Nun wird der Füllstand der Becken und der Kontostand für den nächsten Schritt aktualisiert.
Nachdem die Schleife sämtliche Freuqenz/Intraday-Preispaare bearbeitet hat, werden die Verläufe der Füllstände und des Kontostands in dem Datastruct gespeichert.

1.5.1 Dritte Visualisierung/Abbildung

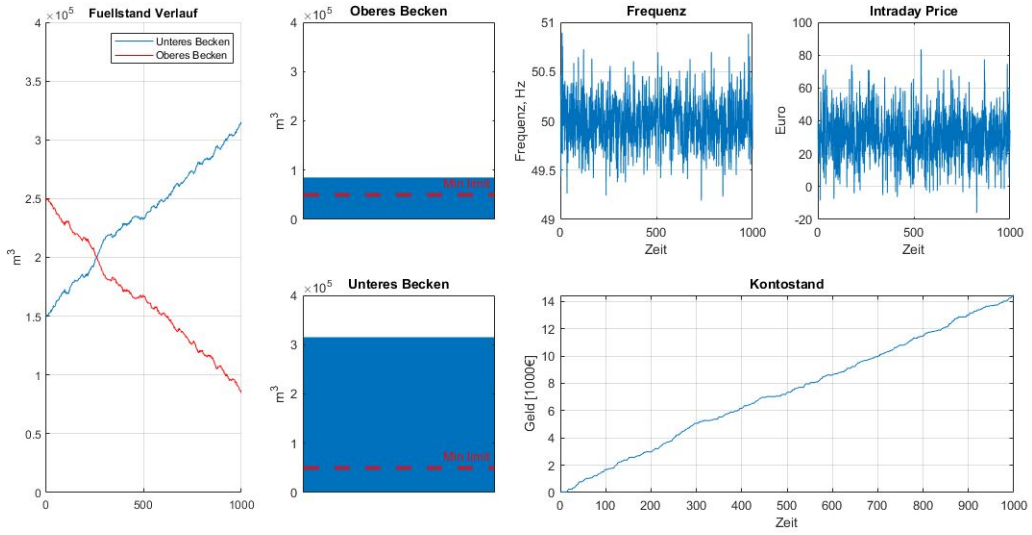


Abbildung 4: Visualisierung der Verläufe von Füllstand, Kontostand, oberen und unteren Becken sowie die aufbereitete Frequenz und Intraday-Preis für Datensatz 1

Hier werden die Verläufe von Füllstand, Kontostand, oberen, unteren Becken sowie die aufbereiteten Frequenz und Intraday-Preis nach der Entscheidung nach Abbildung 3 in der Aufgabenbeschreibung abgebildet.

Dazu wird die Funktion `plot_status(fill_level_upper_basin, fill_level_lower_basin, frequency, intraday_price, account_balance, figure_title)` benutzt.

Abbildung 4 zeigt diese Visualisierung für Datensatz 1.

2 Funktionen

2.1 `calculate_earnings`

```
% Bestimme die Gewinne/Kosten des Turbinen-/Pumpenbetriebs fuer das
% aktuelle Frequenz/Intraday-Preispaar
function winnings = calculate_earnings(filling_level_change_lower, frequency,
intraday_price, is_max, is_min)
    % Kosten und Gewinn durch das Pumpen des Wassers
    % 1000 m^3 <-> 1MWh, zum aktuellen intraday Preis gehandelt
    winnings = filling_level_change_lower * intraday_price / 1000;


    % Bonus fuer Entnahme von Strom bei hoher Netzfrequenz (lokalem
    % Frequenzmaxima)
    if(filling_level_change_lower < 0 && frequency > 50 && is_max)
        winnings = winnings + 50;
        return;
    end

    % Bonus fuer Stromversorgung bei niedriger Netzfrequenz (lokalem
    % Frequenzminima)
    if(filling_level_change_lower > 0 && frequency < 50 && is_min)
        winnings = winnings + 50;
        return;
    end
end
```

Abbildung 5: Code von `calculate_earnings(filling_level_change_lower, frequency, intraday_price, is_max, is_min)`

2.1.1 Input

1) `filling_level_change_lower`: double, Änderung des Wasservolumens im unteren Becken.
2) `frequency`: double, Die korrigierte Frequenz des bearbeiteten Frequenz/Intraday-Preispaars vom entsprechenden Datensatz
3) `intraday_price`: double, Der korrigierte Intraday-Preis der der Frequenz zugeordnet ist
4) `is_max`: logical, ist das aktuelle Wertepaar ein lokales Maximum in den geglätteten Frequenzdaten
5) `is_min`: logical, ist das aktuelle Wertepaar ein lokales Minimum in den geglätteten Frequenzdaten

2.1.2 Output

`winnings`: Die Gewinne oder Kosten anhand der Änderung des Füllstandes des unteren Beckens und des Intraday-Preise. Für eine Füllstandsänderung von 1000m³ wird 1MWh verbraucht oder erzeugt, die zum aktuellen Intraday Preis verrechnet werden.
Zusätzlich wird ein Bonus von 50€ addiert für die Stromentnahme bei hoher Frequenz und für Stromeinspeisung bei niedriger Frequenz.

2.2 calculate_filling_change_lower

```
% Bestimme die Aenderung des Fuellstandes des niedrigeren Beckens anhand
% des aktuellen Intraday-Preises sowie der minimalen
% Fuellstaende
% Gibt entweder -1000, 0, +1000 zurueck
function filling_change_lower =
calculate_filling_change_lower(fill_level_upper_basin, fill_level_lower_basin,
intraday_price)
    % Bei Betrieb der Turbine/Pumpe werden 1000m^3 Wasser pro MWh bewegt
    water_delta  = 1000;
    % Der Minimale Fuellstand betraegt 50.000 m^3
    minimum_fill_level_basin = 50E3;

    % Bestimme ob Wasser vom oberen in das untere Becken fliesst
    if(intraday_price > 40 && fill_level_upper_basin >
minimum_fill_level_basin)
        filling_change_lower = + water_delta;
        return;
    end

    % Bestimme ob Wasser vom unteren in das obere Becken gepumpt wird
    if(intraday_price < 10 && fill_level_lower_basin >
minimum_fill_level_basin)
        filling_change_lower = - water_delta;
        return;
    end

    % Sonst veraendert sich der Fuellstand der Becken nicht
    filling_change_lower = 0;
    return
end
```

Abbildung 6: Code von `calculate_filling_change_lower(fill_level_upper_basin, fill_level_lower_basin, intraday_price)`

2.2.1 Input

1) `fill_level_upper_basin`: double, Beschreibt den aktuellen Füllstand vom oberen Becken
2) `fill_level_lower_basin`: double, Beschreibt den aktuellen Füllstand vom unteren Becken
3) `intraday_price`: double, ist der Intraday-Preis vom entsprechen Frequenz/Intraday-Preispaar

2.2.2 Output

`filling_change_lower`: double, ist die Änderung des Füllstands im unteren Becken

Zuerest werden `water_delta` und fill_level_basin initialisiert.
`water_delta` entspricht dem Volumen an Wasser, das in einer Zeiteinheit bewegt wird, also 1000m³.
`minimum_fill_level_basin` ist das kleinste Volumen an Wasser, dass im oberen oder unteren Becken erlaubt ist, was 50 000m³ entspricht.
Damit das Wasser vom oberen in das untere Becken fließt, muss der Intraday-Preis höher als 40 Euro sein und das obere Becken mehr als 50 000m³ Wasser haben. Wenn diese 2 Bedingungen erfüllt sind, dann wird `filling_water_change` der Wert `+water_delta` gegeben. Um das Wasser vom unteren in das obere Becken zu pumpen, muss der Intraday-Preis weniger als 10 Euro sein und das untere Becken mehr als 50 000 m^3 haben. Wenn diese 2 Bedingungen erfüllt sind, dann wird `filling_water_change` den Wert `-water_delta` gegeben.
Wenn keine der Bedingungen erfüllt sind bleibt der Wert von `filling_water_change` bei 0.

2.3 plot_status

```
% Erstelle Abbildung nach fig. 3, 4 in Aufgabenbeschreibung
% Verlauf von Fuellstand, Frequenz, Intraday Price und Kontostand
% aktueller Fuellstand des oberen und unteren Beckens
function plot_status(fill_level_upper_basin, fill_level_lower_basin, ...
                     frequency, intraday_price, account_balance, figure_title)
    % Farbcode von "Matlab Blau"
    mat_blue = [0 0.4470 0.7410];

    % Definiere Grenzen fuer die subplots
    time_limits    = [0 length(frequency)];
    basin_limits   = [0 4E5];
    account_limits = [0 Inf];

    % Erstelle Fenster fuer die figure
    figure('Name',figure_title,'NumberTitle','off');
    set(gcf, 'Position',  [100, 100, 1300, 600])

    % Erstelle subplot fuer den Fuellstandverlauf
    subplot(2,4,[1,5]);
    hold on
    plot(fill_level_lower_basin,'Color',mat_blue);
    plot(fill_level_upper_basin,'r');
    xlim(time_limits);
    ylim(basin_limits);
    legend("Unteres Becken", "Oberes Becken");
    title("Fuellstand Verlauf");
    ylabel("m^3");
    grid on;

    % Erstelle subplot fuer das obere Becken
    subplot(2,4,2);
    % Zeige letzten Fuellstand an der nicht-Null ist (da das Becken nicht
    % komplett geleert werden kann)
    last_nonzero = find(fill_level_upper_basin,1,'last');
    area([0 1], [fill_level_upper_basin(last_nonzero
fill_level_upper_basin(last_nonzero)],'LineStyle','none');
    xlim([0 1]);
    ylim(basin_limits);
    title("Oberes Becken");
    ylabel("m^3");
    set(gca,'xtick',[])
    yline(50E3,'r--','Min limit','Linewidth',3);




% Erstelle subplot fuer das obere Becken
```

```
    subplot(2,4,6);
    % Zeige letzten Fuellstand an der nicht-Null ist (da das Becken nicht
    % komplett geleert werden kann)
    area([0 1], [fill_level_lower_basin(last_nonzero)
fill_level_lower_basin(last_nonzero)],'LineStyle','none');
    xlim([0 1]);
    ylim(basin_limits);
    title("Unteres Becken");
    ylabel("m^3");
    set(gca,'xtick',[])
    yline(50E3,'r--','Min limit','Linewidth',3);


    % Erstelle subplot fuer die Frequenz
    subplot(2,4,3);
    plot(frequency);
    xlim(time_limits);
    title("Frequenz");
    ylabel("Frequenz, Hz");
    xlabel("Zeit");
    grid on;

    % Erstelle subplot fuer den Intraday-Price
    subplot(2,4,4);
    plot(intraday_price);
    xlim(time_limits);
    title("Intraday Price");
    ylabel("Euro");
    xlabel("Zeit");
    grid on;

    % Erstelle subplot fuer den Kontostandverlauf
    subplot(2,4,[7,8]);
    plot(account_balance / 1000);
    xlim(time_limits);
    ylim(account_limits);
    title("Kontostand");
    ylabel("Geld [1000€]");
    xlabel("Zeit");
    grid on;

    % Oeffne Fenster mit figure
    shg
end
```

Abbildung 7: Code von `plot_status(fill_level_upper_basin, fill_level_lower_basin, frequency, intraday_price, account_balance, figure_title)`

2.3.1 Input

1) `fill_level_upper_basin`: [1000x1 double], ist der Füllstandverlauf des oberen Becken
2) `fill_level_lower_basin`: [1000x1 double], ist der Füllstandverlauf des unteren Becken
3) `frequency`: [1000x1 double], ist der Verlauf der Frequenz
4) `intraday_price`: [1000x1 double], ist der Verlauf des Intraday-Preises
5) `account_balance`: [1000x1 double], ist der Kontostandverlauf
6) `figure_title`: string, ist der Titel der Abbildung

2.3.2 Output

Statt einer Rückgabe erstellt diese Funktion ein Fenster mit den Verläufen von Füllstand der Becken, Frequenz und Intraday Preis und Kontostand dargestellt.
Hierfür wird zunächst mit dem figure() Befehl ein neues Fenster für eine Abbildung geöffnet. Im Folgenden wird mit dem subplot() Befehl die Visualisierung der jeweiligen Daten in dem Fenster eingeordnet.
Diese Funktion wird zwei mal verwendet um den Anfangszustand und den Verlauf der Variablen abzubilden.

2.4 plot_frequency

```
% Erstelle Abbildung nach fig. 2 in Aufgabenbeschreibung
% Verlauf von Frequenz gegen geglaettete Frequenz
% Lokale Maxima und Minima der geglaetteten Frequenz
function plot_frequency(frequency, smoothed_frequency, max_mask, min_mask,
frequency_title)
    % Erstelle Fenster fuer die figure
    figure();
    % Erstelle subplot fuer die Frequenz gegen geglaettete Frequenz
    subplot(1,2,1);
    hold on;
    plot(frequency,"-b","LineWidth",0.5);
    plot(smoothed_frequency,"-r","LineWidth",2);
    legend("Input data","Smoothed data");
    title(frequency_title, 'Interpreter', 'none');

    % Erstelle subplot fuer die Extrema der geglaetteten Frequenz
    subplot(1,2,2);
    hold on;
    plot(smoothed_frequency,"-b","LineWidth",0.5);
    indices = 1:length(smoothed_frequency);
    scatter(indices(max_mask),
smoothed_frequency(max_mask),25,"^r","filled");
    scatter(indices(min_mask),
smoothed_frequency(min_mask),25,"vy","filled");
    title(["Number of extrema: ", num2str(sum(max_mask) + sum(min_mask))]);
    legend("Input data", "Local maxima", "Local minima");

    % Oeffne Fenster mit figure
    shg
end
```

Abbildung 8: Code von `plot_frequency(frequency, smoothed_frequency, max_mask, min_mask, frequency_title)`

2.4.1 Input

1) `frequency`: [1000x1 double], ist die aufbereitete aber nicht geglättete Frequenz
2) `smoothed_frequency`: [1000x1 double], ist die geglättete Frequenz
3) `max_mask`: [1000x1 logical], Maske der lokalen Maxima
4) `min_mask`: [1000x1 logical], Maske der lokalen Minima
5) `frequency_title`: ist der Name des entsprechenden Datensatzes, der als Titel für die Abbildung benutzt wird

2.4.2 Output

Statt einer Rückgabe erstellt diese Funktion ein Fenster mit dem Frequenzverlauf im Vergleich zum geglätteten Frequenzverlauf dargestellt. Weiterhin werden die Maxima und Minima des geglätteten Frequenzverlaufes angezeigt. Das Format des Fensters orientiert sich nach Abbildung 2 in der Aufgabenbeschreibung.

2.5 `extrema`

```
% Erstelle Maske fuer die lokalen Maxima und Minima der gegebenen Frequenz
% Beispiel: max_mask = [0,0,0,1,0,0,1,0] fuer Maxima an Index 4 und 7
function [max_mask, min_mask] = extrema(frequency)
    max_mask = islocalmax(frequency);
    min_mask = islocalmin(frequency);
end
```

Abbildung 9: Code von `extrema(frequency)`

2.5.1 Input

`frequency`: [1000x1 double], ist die geglättete Frequenz

2.5.2 Output

1) `max_mask`: [1000x1 logical], Maske der lokalen Maxima
2) `min_mask`: [1000x1 logical], Maske der lokalen Minima

Die Funktion bestimmt eine logische Maske der lokalen Maxima und Minima mithilfe der in Matlab integrierten Funktion `islocalmax(frequency)` oder `islocalmin(frequency)`.

2.6 `read_file`

```
% Liest die gegebene Datei ein und extrahiert Frequenz und Intraday-Preis
function [frequency, intraday_price] = read_file(filename)
    % Oeffne die Datei mit verschiedenen Methoden abhaengig von dem
    % Dateiformat
    [~,~,ext] = fileparts(filename);
    if(ext == ".mat")
        % hier matlab Datei einlesen
        file = load(filename);
        frequency = file.data.frequency;
        intraday_price = file.data.intraday_price;
        return
    end
    if(ext == ".csv")
        % hier .csv Datei einlesen
        file = readmatrix(filename);
        frequency = file(:,1);
        intraday_price = file(:,2);
        return
    end
    if(ext == ".json")
        % hier .json Datei einlesen
        text = fileread(filename);
        values = jsondecode(text);
        frequency = values.frequency;
        intraday_price = values.intraday_price;
        return
    end
end
```

Abbildung 10: Code von `read_file(filename)`

2.6.1 Input

`filename`: ist der volle Pfad und Name des aufzubereitenden Datensatzes

2.6.2 Output

1) `frequency`: die nicht aufbereiteten vom Datensatz extrahierten Frequenzdaten
2) `intraday_price`: die nicht aufbereiteten vom Datensatz extrahierten Intraday-Preisdaten

Die Funktion read_file extrahiert aus der gegebenen Datei die Frequenz und Intraday-Preisdaten. Hierfür wird je nach Dateiformat andere Matlabfunktionen verwendet die für das jeweilige Dateiformat angemessen sind.

2.7 correct_data

```
% Pruefe Daten nach Bedingung, ersetze fehlerhafte Werte mit dem
% arithmetischen Mittel der ersten nicht-fehlerhaften Werte die links und
% rechts von dem fehlerhaften Wert sind
function corrected_data = correct_data(data, condition)
    corrected_data = data;

    % Ueberpruefe den ersten Wert und ersetze den (falls fehlerhaft) durch
    % den ersten nicht-fehlerhaften Wert
    if(condition(data(1)))
        k = 2;
        while(condition(data(k)))
            k = k+1;
        end
        corrected_data(1) = data(k);
    end
    % Ueberpruefe den letzten Wert und ersetze den (falls fehlerhaft) durch
    % den letzten nicht-fehlerhaften Wert
    if(condition(data(end)))
        k = size(data,1)-1;
        while(condition(data(k)))
            k = k-1;
        end
        corrected_data(end) = data(k);
    end

    % Maske mit fehlerhaften Werten
    mask = condition(corrected_data);
    % Indexe der fehlerhaften Werte
    indx = find(mask);
    % Bestimme Indexe der naechsten Nachbarn von fehlerhaften Werten, die
    % nicht selbst fehlerhaft sind
    [correct_left_indx, correct_right_indx] = correct_data_index(mask, indx);

    % Ersetze fehlerhafte Werte durch das arithmetische Mittel der
    % naechsten nicht-fehlerhaften Werte
    corrected_data(indx) = (data(correct_left_indx) +
data(correct_right_indx))/2;
    return
end
```

Abbildung 11: Code von `correct_data(data, condition)`

2.7.1 Input

1) `data` : [1000 x 1 double] sind die Rohdaten (entweder Frequenz oder intraday-Preis)
2) `condition`: [function_handle], die Bedingung nach der die Rohdaten (Frequenz oder intraday-Preis) fehlerhaft sind.

2.7.2 Output

`corrected_data`: die korrigierten Daten

Zunächst wird geprüft ob die Werte an den Rändern (1, end) der gegebenen Datenliste die condition erfüllen und damit fehlerhaft sind.
Sollte das der Fall sein, werden diese durch den ersten nicht-fehlerhaften Wert neben denen ersetzt.
Danach wird eine Maske mit allen Werten erstellt die nach Bedingung condition fehlerhaft sind.
Die Indexe der fehlerhaften Werte und die Maske wird an `correct_data_index(mask, index)` weitergereicht.
Mit dieser Funktion werden die Indizes aller nicht fehlerhaften Werte links und rechts von den fehlerhaften Werten bestimmt.
Im Folgenden werden alle fehlerhaften Werte durch das arithmetische Mittel der nächsten ermittelten nicht fehlerhaften Werte ersetzt und als `corrected_data` zurückgegeben.

2.8 `correct_data_index`

```
% Hilfsfunktion von correct_data(data, condition)
% Findet naechste benachbarte Indexe von fehlerhaften Werten die nicht
% fehlerhaft sind
function [left_indices, right_indices] = correct_data_index(mask, indices)
    % Anzahl der zu durchsuchenden Werte
    array_length = length(mask);
    % Anzahl als fehlerhaft markierten Werte
    index_length = length(indices);
    % Indexe der naechsten nicht-fehlerhaften Werte
    left_indices  = zeros(index_length,1);
    right_indices = zeros(index_length,1);

    % Fuer jeden fehlerhaften Wert...
    for i = 1:index_length
        left_index = indices(i)-1;
        right_index = indices(i)+1;
        % Diese Schleifen werden nur dann ausgefuehrt wenn zwei fehlerhafte
        % Werte nebeneinander sind, gehe so viele Indexe nach links
        % (rechts) bis ein nicht-fehlerhafter Wert gefunden ist oder die
        % Grenzen der Werteliste erreicht sind
        while(left_index > 1 && mask(left_index) == 1)
            left_index = left_index - 1;
        end
        while(right_index < array_length && mask(right_index) == 1)
            right_index = right_index + 1;
        end

        left_indices(i)  = left_index;
        right_indices(i) = right_index;
    end
    return
end
```

Abbildung 12: Code von `correct_data_index(mask, indices)`

2.8.1 Input

1) `mask` : [1000 x 1 logical], Liste aller als fehlerhaft markierten Werte
2) `indices`: [N x 1 double], Liste aller Indizes mit fehlerhaften Werten. Hierbei entspricht N der Anzahl der fehlerhaften Werte

2.8.2 Output

1) `left_indices` : [N x 1 double], Liste der ersten nicht-fehlerhaften Indizes links von den fehlerhaften
2) `right_indices`: [N x 1 double], Liste der ersten nicht-fehlerhaften Indizes rechts von den fehlerhaften

Generell gilt `left_indices(i) < indices(i) < right_indices(i)` für alle i von 1 bis N, mit N der Anzahl fehlerhafter Werte.
Es wird für jeden fehlerhaften Index eine Schleife verwendet, um den ersten Index links (beziehungsweise rechts) von dem fehlerhaften Index zu finden für den gilt: mask(index) == 0, also dass der Wert mit dem Index nicht fehlerhaft ist oder die Grenzen der Werteliste erreicht sind.

Diese Funktion wird als Hilfsfunktion von `correct_data(data, condition)` aufgerufen nachdem die fehlerhaften Werte an den Grenzen bereits ersetzt wurden, daher werden alle fehlerhaften Werte die in dieser Funktion bearbeitet werden immer zwei richtige Werte jeweils links und rechts haben.

2.9 generate_file_paths

```
% Fuehrt Aneinanderkettung der strings durch, falls kein Pfad oder kein
% Dateiformat gegeben ist, benutze Standards
function full_file_paths = generate_file_paths(file_paths, file_names,
file_extensions)
    % Valide Dateiformate
    extensions = [".csv", ".json", ".mat"];

    % Anzahl gegebener Dateien
    file_count = length(file_names);

    % dynamisch den Ordner des Programms als Stammpfad aller Dateien nehmen
    % falls keiner gegeben ist
    if(file_paths == "")
        [program_path, ~, ~] =
fileparts(matlab.desktop.editor.getActiveFilename);
        file_paths = strings(file_count,1);
        for f = 1:file_count
            file_paths(f) = program_path + "\Datensatz";
        end
    end

    % dynamisch ein zufaelliges valides Dateiformat waehlen falls keins
    % gegeben ist (nimmt an dass alle Dateien in allen Dateiformaten
    % vorhanden sind)
    if(file_extensions == "")
        file_extensions = strings(file_count,1);
        % Waehle beliebiges Dateiformat aus den vorgegebenen
        for f = 1:file_count
            file_extensions(f) = extensions(1 + mod(f,3));
        end
    end

    % Baue vollen Dateipfad aus dem Pfad der Datei, dem Dateinamen und des
    % Dateiformats
    full_file_paths = strings(file_count,1);
    for f = 1:file_count
        full_file_paths(f) = file_paths(f) + '\' + file_names(f) +
file_extensions(f);
    end
end
```

Abbildung 13: Code von `generate_file_paths(file_paths, file_names, file_extensions)`

2.9.1 Input

1) `file_paths`: [N x 1 string], ist ein Array an Pfaden, die vom Nutzer eingegeben werden können. Kann als `""` leer gelassen werden um den Pfad des Programms zu nutzen.
2) `file_names`: [N x 1 string], die Namen der Dateien, die verarbeitet werden sollen
3) `file_extentions`: [Nx1 string], die Endungen der Dateien, die im Stampfad liegen. Kann als `""` leer gelassen werden um der Reihe nach `".csv"`, `".json"`, `".mat"` Dateiformate anzuwenden

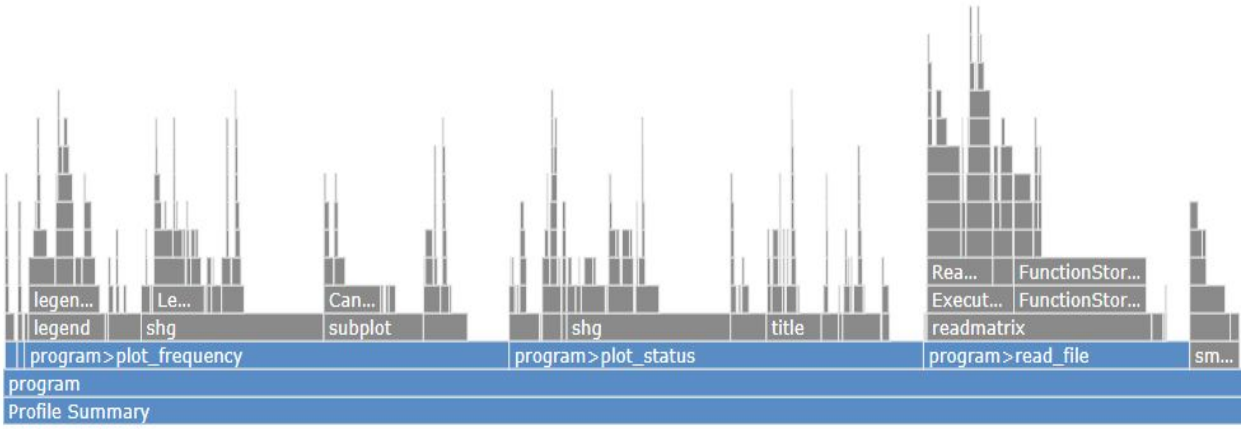
2.9.2 Output

Diese Funktion ist da, um die Benutzung des Programms zu vereinfachen. Sie ermöglicht das Bestimmen eines eigenen Dateipfades für jeden Datensatz und auch das Auslassen von dem Standard-Pfad (`program_path + "\Datensatz"`). Sollte kein Dateiformat angegeben sein werden die drei vorgegebenen Dateiformate der Reihe nach angewandt.

3 Laufzeitanalyse

Profile Summary (Total time: 24.500 s)

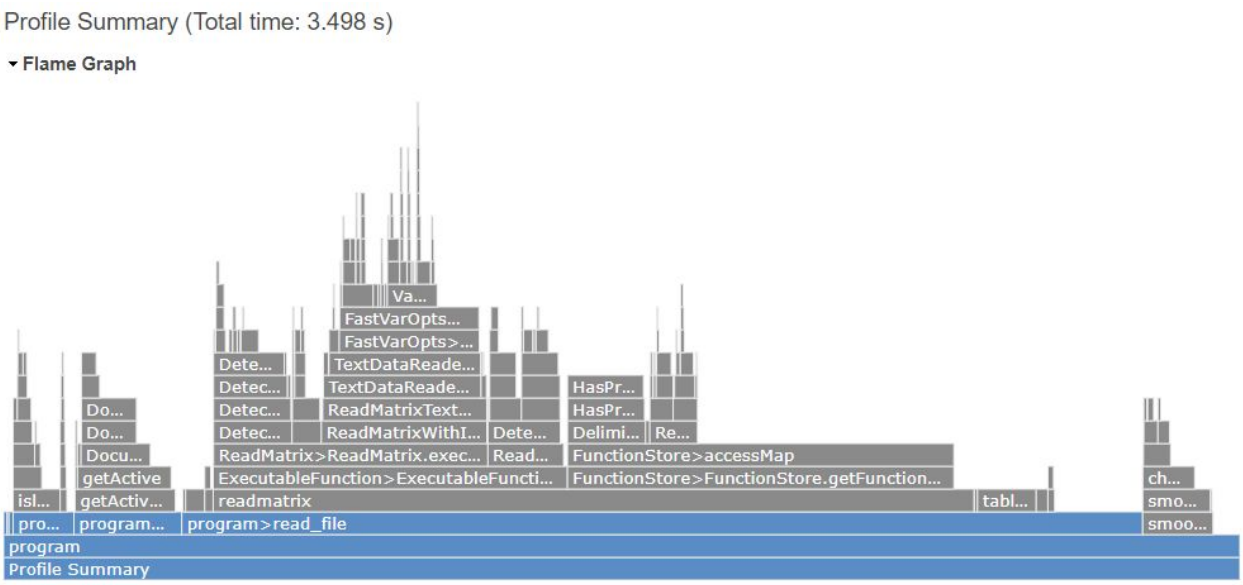
▾ **Flame Graph**



Generated 07-Sept.-2020 22:33:56 using performance time.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
program	1	24.500	0.103	
program>plot_frequency	4	9.566	0.822	
program>plot_status	8	8.178	0.666	
shg	12	6.833	2.917	
program>read_file	4	5.252	0.450	
readmatrix	1	4.427	0.092	
subplot	56	2.677	0.863	

Abbildung 14: Laufzeitanalyse des Programms



Generated 07-Sept.-2020 22:41:32 using performance time.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
program	1	3.498	0.072	
program>read_file	4	2.718	0.250	
readmatrix	1	2.152	0.056	
FunctionStore>FunctionStore.getFunctionByName	1	1.093	0.000	
FunctionStore>accessMap	1	1.093	0.724	
ExecutableFunction>ExecutableFunction.validateAndExecute	1	1.002	0.011	

Abbildung 15: Laufzeitanalyse des Programms ohne die Funktionen die für die Abbildungen gebraucht werden.

Wie in Abbildung 14 erkennbar ist, wird die mit Abstand größte Zeit des Programms mit dem Erstellen und Darstellen der Abbildungen in Anspruch genommen.
Also insbesondere `plot_frequency` und `plot_status`.

Abbildung 15 zeigt die Laufzeitanalyse wenn diese beiden Funktionen nicht aufgerufen werden.
Hier ist erkennbar, dass die meiste Zeit des Programms mit dem Einlesen der Dateien verbraucht wird.
Die nächste Funktion die die meiste Zeit verbraucht ist die `generate_filepaths` Funktion.

Da das Erstellen der Abbildungen die meiste Zeit benötigt, können diese nur bei Bedarf erstellt werden, oder alternativ nach dem gesamten restlichen Programm.
Das Einlesen der Dateien kann durch das Verzichten auf in matlab verfügbaren Funktionen beschleunigt werden. Dies könnte erfolgen, indem eine selbsterstellte Funktion geschrieben wird die die bestimmten Dateiformate effizienter nur nach dem bekannten Frequenz/Intraday-Preis Muster durchsucht und diese Daten extrahiert.
Das Verbinden der Dateipfade kann ebenfalls optimiert werden, da es sich hierbei im Wesentlichen nur um die Aneinanderreihung von strings handelt. Durch direktes Verwenden von dem ganzen Dateipfad in einem string ohne die Aufteilung in Pfad, Dateiname und Format können so langsame string-Operationen gespart werden.