**Contents:**

* Struktur von Modell02
* Beispiel von Kollektormodell in openmodelica verstehen
* Kollektordaten und Weatherdaten verändern
* Hauptprogramme beschreiben
* Weatherdaten zu verändern
* Kollektordaten zu verändern
* Vergleichen Modell 02 mit Modell 01
* Simulation zu laufen
* Ergebnisse in OM zu zeigen
* Ergebnisse in Matlab auszugeben
* Struktur von Modell02
* Weatherdateneinlesen:

**Potsdam2019.txt**

**Germany\_Potsdam\_2019\_DWD.txt**

**Modell02.** **Potsdam2019\_Meteonorm\_ASCII**

* Zuordnung und Analyse von Ergebnissen

**‘****Sortieren\_res.mat’ in ‘Sortieren\_output’**

**Modell02.mo**

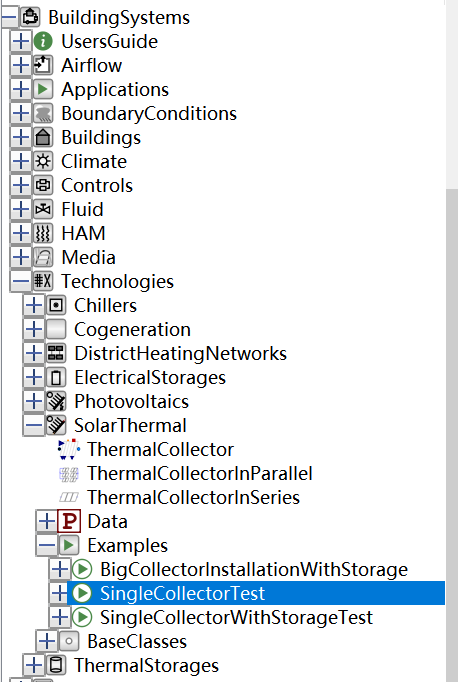
**Beispiel von Kollektormodell in openmodelica verstehen**

**OManalyse02\_mat.mat**

**Analyse von Ergebnissen**

**Zuordnung von Output-Varianten**

In diese openmodelica\_Liberary haben wir eine gute Beispiel ‘SingleCollectorTest‘ gefunden.

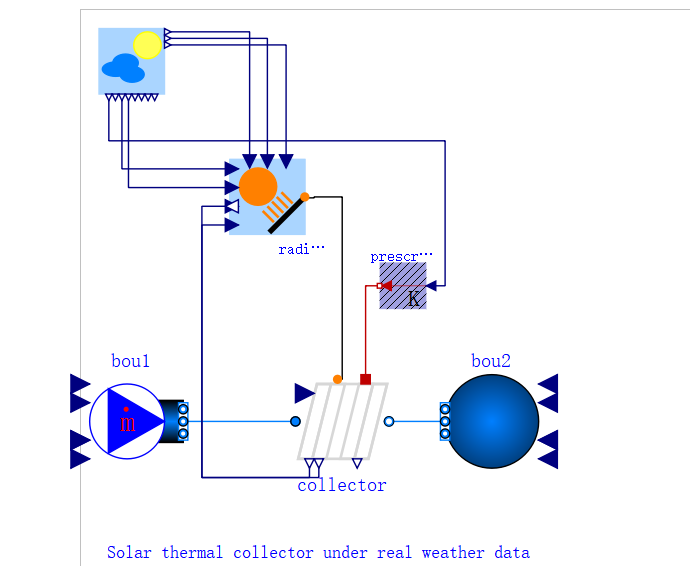


Es gibt 6 Komponenten in diese Schaubild.

Zuerst wird WeatherDataReader **Gb, Gd,** die geogradische Koordinaten von eine Stadt(z.B. Wurzburg) als Eintrittsparameter in ‘SolarRadiationTransformer‘ und **Ta** in ‘Heattransfer‘ importieren.

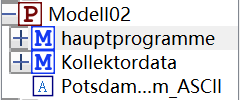
Danach wird ‘SolarRadiationTransfer‘ G an ‘RadiationPort‘ , auch Azimut, tilt angle in Kollektor importieren. **Ta** von ‘Heattransfer‘ importieren an ‘Heatport‘ von Kollektor. ‘MassFlowSource1‘ wird die Massenstrom und Eintrittstemperatur von Kollektor in Kollektor importieren.

Zuletzt wird die Austrittstemperatur von Kollektor in ‘MassFlowSource2‘ exportieren.



* **Kollektordaten und Weatherdaten verändern**

Package als ‘Modell 02‘ einstellen. Block ‘Potsdam2019\_Meteronorm\_ASCII‘ in ‘Modell 02‘ einstellen. Model ‘Kollektordaten‘ und ‘hauptprogramme‘ in ‘Modell 02 ‘ einstellen.



* **Weatherdaten zu verändern**

**Block ‘Potsdam2019\_Meteronorm\_ASCII‘ beschreiben.**

block Potsdam2019\_Meteonorm\_ASCII

extends BuildingSystems.Climate.WeatherData.BaseClasses.WeatherDataFileASCII(info = "Source: Meteonorm 7.0", filNam = Modelica.Utilities.Files.loadResource("modelica://Modell02/Potsdam2019.txt"), **‘Weatherdaten zu lesen’**

final tabNam = "tab1", final timeFac = 1.0 / 3600.0, final deltaTime = 1800.0, final columns = {5, 6, 3, 8, 9, 4, 7}, final scaleFac = {1.0, 1.0, 1.0, 1.0, 1.0, 0.01, 1.0}, final latitudeDeg = 49.47, final longitudeDeg = 9.57, final longitudeDeg\_0 = 1.0);

**‘Einlesensparameter’**

// beam horizontal radiation

// diffuse horizontal radiation

// air temperature

// wind speed

// wind direction

// relative humidity

// cloud cover

annotation(

Documentation(info = "<html>source: Meteonorm 7.0</html>"));

end Potsdam2019\_Meteonorm\_ASCII;

* **Kollektordaten zu verändern**

**‘Kollektordaten‘ beschreiben**

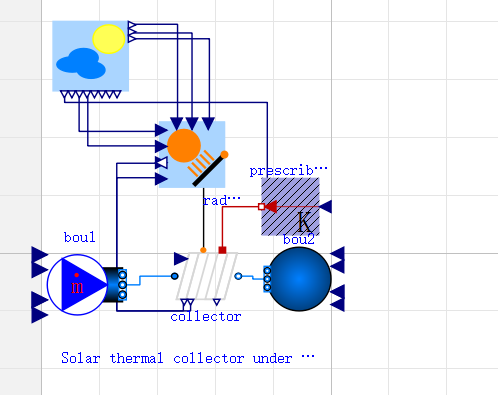
model Kollektordata

record Modell2SolarCollector = BuildingSystems.Technologies.SolarThermal.Data.Collectors.CollectorPartial(final IAMC = 0.92, final V\_A = 1 / 0.1 / 980, final C\_0 = 0.80, final C\_1 = 3.5, final C\_2 = 0.01, A = 2.0) annotation(

uses(BuildingSystems(version = "2.0.0-beta")));

end Kollektordata;

* **model hauptprogramme beschreiben :**



model hauptprogramme

package Medium = BuildingSystems.Media.Antifreeze.PropyleneGlycolWater(X\_a = 0.40, property\_T = 293.15);

Modelica.Thermal.HeatTransfer.Sources.PrescribedTemperature prescribedTemperature annotation(

Placement(visible = true, transformation(origin = {-14, 16}, extent = {{10, -10}, {-10, 10}}, rotation = 0)));

BuildingSystems.Fluid.Sources.MassFlowSource\_T bou1(nPorts = 1, m\_flow = 0.02083, redeclare package Medium = Medium, T = 323.15) annotation(

Placement(visible = true, transformation(origin = {-93, -9}, extent = {{-13, -13}, {13, 13}}, rotation = 0)));

BuildingSystems.Fluid.Sources.Boundary\_pT bou2(redeclare package Medium = Medium, nPorts = 1) annotation(

Placement(visible = true, transformation(origin = {-5, -9}, extent = {{11, -11}, {-11, 11}}, rotation = 0)));

BuildingSystems.Climate.SolarRadiationTransformers.SolarRadiationTransformerIsotropicSky radiation(rhoAmb = 0.2) annotation(

Placement(visible = true, transformation(origin = {-48, 34}, extent = {{-14, -14}, {14, 14}}, rotation = 0)));

**‘Weatherdaten zu lesen’**

BuildingSystems.Climate.WeatherData.WeatherDataReader weatherData(redeclare block WeatherData = Modell02.Potsdam2019\_Meteonorm\_ASCII) "time Gdot\_beam Gdot\_diffuse T\_air\_env" annotation(

Placement(visible = true, transformation(origin = {-83, 68}, extent = {{-13, -12}, {13, 12}}, rotation = 0)));

**‘Kollektordaten zu lesen’** BuildingSystems.Technologies.SolarThermal.ThermalCollector collector(redeclare package Medium = Medium, redeclare Kollektordata.Modell2SolarCollector collectorData, angleDegAzi = 0.0, angleDegTil = 45.0, dp\_nominal = 2.0, m\_flow\_nominal = 0.02083, nEle = 1) annotation(

Placement(visible = true, transformation(origin = {-43, -8}, extent = {{-11, -10}, {11, 10}}, rotation = 0)));

**‘alle Komponenten zu verbinden’**

equation

connect(collector.heatPortCon, prescribedTemperature.port) annotation(

Line(points = {{-37.5, 1}, {-37.5, 16}, {-24, 16}}, color = {191, 0, 0}));

connect(collector.angleDegAzi, radiation.angleDegAzi) annotation(

Line(points = {{-51, -17}, {-51, -20}, {-74, -20}, {-74, 26}, {-59, 26}}, color = {0, 0, 127}));

connect(weatherData.IrrDifHor, radiation.IrrDifHor) annotation(

Line(points = {{-84, 55}, {-84, 37}, {-59, 37}}, color = {0, 0, 127}));

connect(weatherData.longitudeDeg, radiation.longitudeDeg) annotation(

Line(points = {{-69, 76}, {-48, 76}, {-48, 45}}, color = {0, 0, 127}));

connect(radiation.radiationPort, collector.radiationPort) annotation(

Line(points = {{-37, 34}, {-44, 34}, {-44, 1}}));

connect(weatherData.TAirRef, prescribedTemperature.T) annotation(

Line(points = {{-92, 55}, {-92, 52}, {-22, 52}, {-22, 16}, {-2, 16}}, color = {0, 0, 127}));

connect(collector.angleDegTil, radiation.angleDegTil) annotation(

Line(points = {{-48.5, -17}, {-48.5, -20}, {-74, -20}, {-74, 31}, {-59, 31}}, color = {0, 0, 127}));

connect(bou1.ports[1], collector.port\_a) annotation(

Line(points = {{-80, -9}, {-66, -9}, {-66, -8}, {-54, -8}}, color = {0, 127, 255}));

connect(weatherData.longitudeDeg0, radiation.longitudeDeg0) annotation(

Line(points = {{-69, 74}, {-42, 74}, {-42, 45}}, color = {0, 0, 127}));

connect(radiation.latitudeDeg, weatherData.latitudeDeg) annotation(

Line(points = {{-53, 45}, {-53, 79}, {-69, 79}}, color = {0, 0, 127}));

connect(collector.port\_b, bou2.ports[1]) annotation(

Line(points = {{-32, -8}, {-27, -8}, {-27, -9}, {-16, -9}}, color = {0, 127, 255}));

connect(weatherData.IrrDirHor, radiation.IrrDirHor) annotation(

Line(points = {{-87, 55}, {-87, 42}, {-59, 42}}, color = {0, 0, 127}));

annotation(

uses(Modelica(version = "3.2.3"), BuildingSystems(version = "2.0.0-beta")),

Diagram(graphics = {Text(lineColor = {0, 0, 255}, extent = {{-94, -26}, {-4, -46}}, textString = "Solar thermal collector under real weather data")}),

experiment(StartTime = 0, StopTime = 31536000, Tolerance = 1e-6, Interval = 3600));

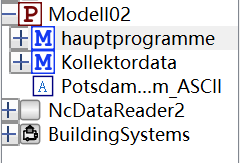
end hauptprogramme;

* **Simulation zu laufen**

**Um die Simulation zu laufen, wählen Sie das Arbeitsverzeichnis als Aktendeckel’Model\_02\_v15‘.**

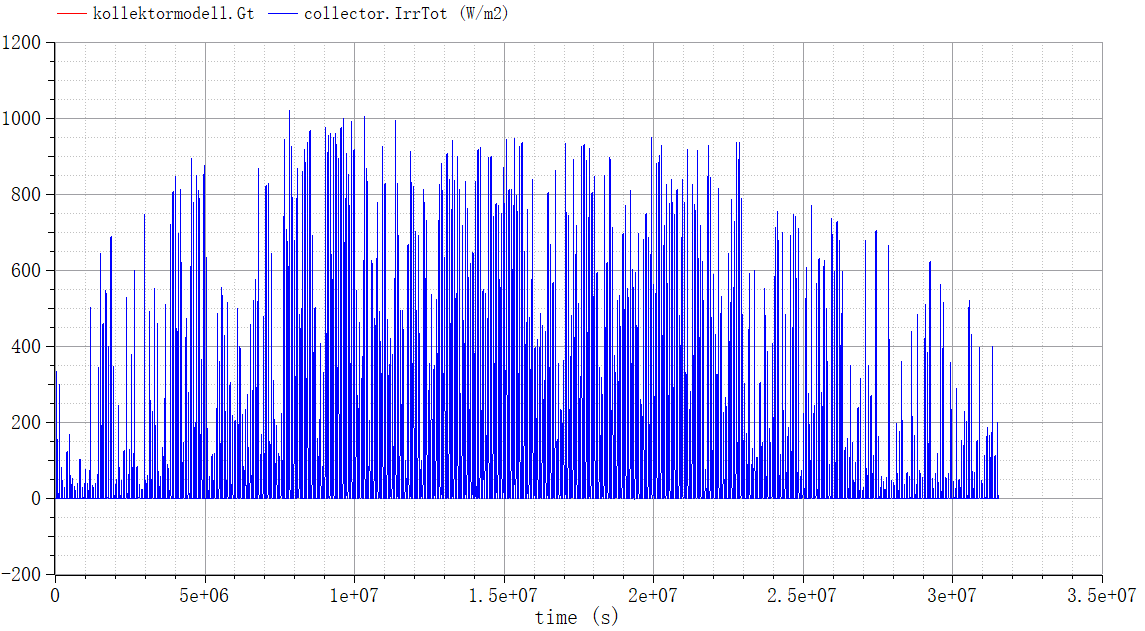


**Wenn man ‚Modell02.hauptprogramme‘ zu laufen bringen möchte, muss man auch Omlibrary ‚Buildingsystem‘ öffnen, um die Komponenten aus dem in Ordnung zu funktionieren.**

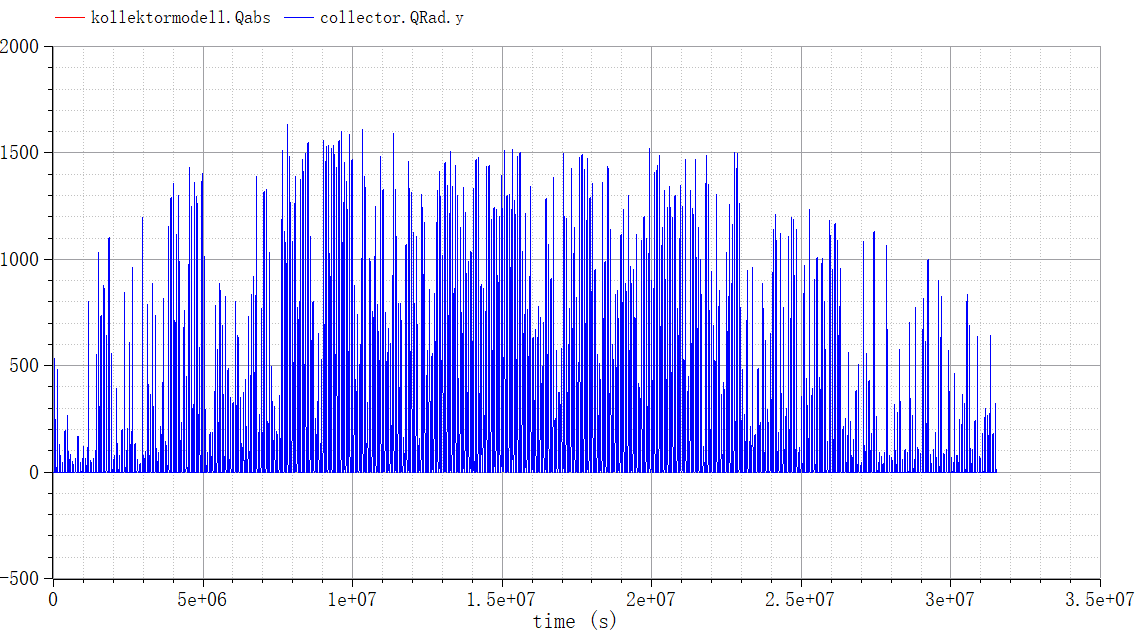


* **Vergleich Modell 02 mir Modell 01**

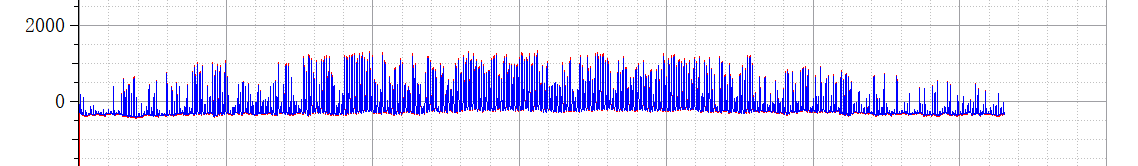
G：



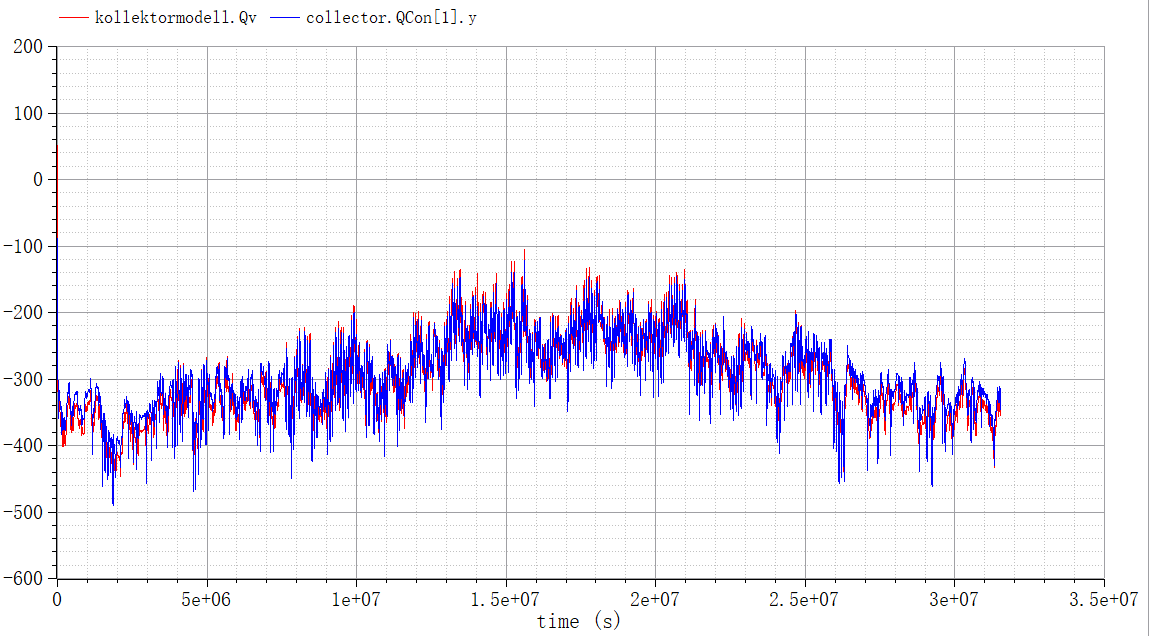
Qabs:



Quse:



Qverlust:

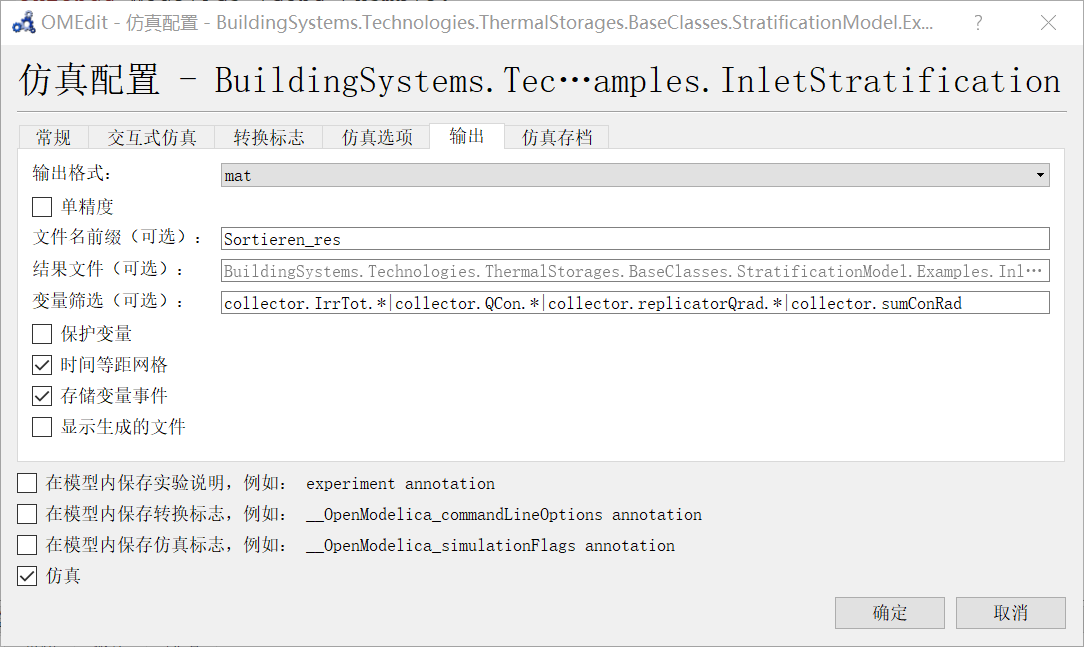


* **Ergebnisse:**

Modell01 und Modell02 haben die gleiche Weatherdatenquelle (‚Germany\_Potsdam\_2019\_DWD.txt ‘) benutzt. Die Ergebnisse zeigt, Modell01 und Modell02 kann gut miteinander anpassen. So können wir weiter gehen.

* Ergebnisse in Matlab auszugegebn

Zuerst gebe ich die Simulationsergebnisse von OpenModelica als Mat-File aus. Die ausgegebene Mat-Dateien sind als ‚Sortieren\_res.mat‘ in Aktiendeckel ‚Sortieren\_output‘ von ‚Modell02\_v15‘ gespeichert.



Danach darstelle und analysiere ich die ausgegebene Ergebnissevarianten in Matlab durch folgende Matlab-Code.

clc

clear all

%%

% load Sim\_out.mat % individueller Name derAusgabedatei, alle Var: .\*

load 'Modell02.hauptprogramme/Sortieren\_res.mat' % alternativ

%% Datenaufbereitung

[r,c] = size(data\_2);

%% Header

col1 = string(name(:,1:r)');

col2 = string(description(:,1:r)');

data = (data\_2);

res\_0 = table(col1,col2,data);

res\_0.Properties.VariableNames{1} = 'Bez\_var';

res\_0.Properties.VariableNames{2} = 'OM\_var';

%% Dateneinlesen

res\_0.OM\_var

ts0 = res\_0.data(1,:)';

% Zeile 3: "collector.IrrTot " "Total solar radiation on collector's absorber surfcace [W/m2]

GT0 = res\_0.data(3,:)';

% Zeile 4: "collector.QCon[1].y " "Value of Convective heat flow Real output

Qv0 = res\_0.data(4,:)';

% Zeile 6: "collector.replicatorQrad.y[1] " "Value of Radiative heat flow Real output signals

Qabs0 = res\_0.data(6,:)';

% Zeile 7: "collector.sumConRad[1].y " "Value of Useenergy Real output signal

Quse0 = res\_0.data(7,:)';

%% Filter für Stundenwerte

p = ts0/3600 - fix(ts0/3600); %

pp = p ==0; % wenn pp ==1, dann voller Stundenwert

ts = ts0(pp)/3600; % Auwahl der vollen Stunden

ts = ts(2:end-1,1); % Eliomination erster / letzter Wert

GT = GT0(pp);

GT = GT(2:end-1,1);

Qv = Qv0(pp);

Qv = Qv(2:end-1,1);

Qabs = Qabs0(pp);

Qabs = Qabs(2:end-1,1);

Quse = Quse0(pp);

Quse = Quse(2:end-1,1);

%% Eingabeswert

Tci = 50 ;

Quse(Quse<0)=0;

ctr=Quse./Quse;

ctr(isnan(ctr))=0;

%% Zuordnung der Ergebnissen

figure(1)

subplot(2,1,1)

plot(ts/3600 ,GT.\*ctr ,'.-r');

hold on

plot(ts/3600 ,Qabs/2.\*ctr,'.-b');

hold off

legend('GT','Qabs/2')

grid

ylabel('w/m^2')

subplot(2,1,2)

plot(ts/3600 ,Qv/2.\*ctr,'.-r');

hold on

plot(ts/3600 ,Quse/2.\*ctr,'.-g');

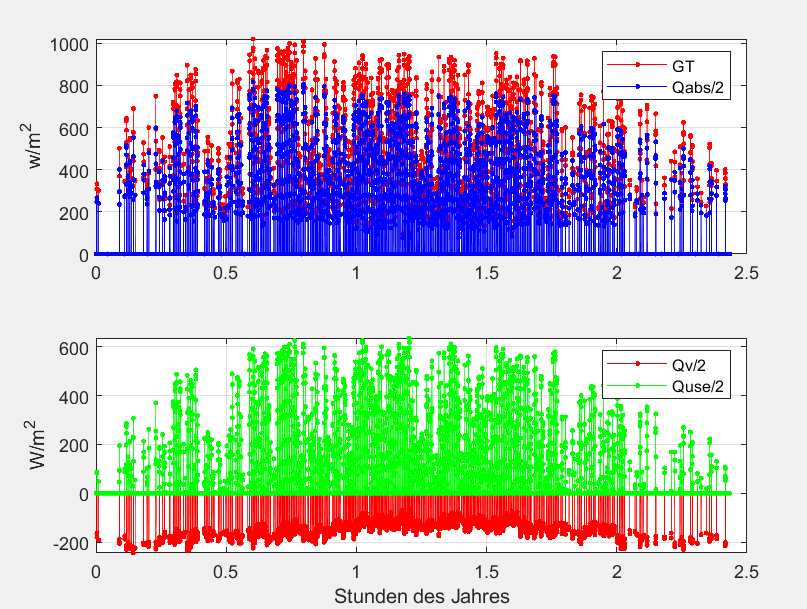
hold off

grid

legend('Qv/2','Quse/2')

xlabel('Stunden des Jahres')

ylabel('W/m^2')



%%

QuseD=sum(reshape(Quse(1:8760,1),[24,365]))./1000;

QuseM(1,1)=sum(QuseD(1:31));

QuseM(1,2)=sum(QuseD(32:59));

QuseM(1,3)=sum(QuseD(60:90));

QuseM(1,4)=sum(QuseD(91:120));

QuseM(1,5)=sum(QuseD(121:151));

QuseM(1,6)=sum(QuseD(152:181));

QuseM(1,7)=sum(QuseD(182:212));

QuseM(1,8)=sum(QuseD(213:243));

QuseM(1,9)=sum(QuseD(244:273));

QuseM(1,10)=sum(QuseD(274:304));

QuseM(1,11)=sum(QuseD(305:334));

QuseM(1,12)=sum(QuseD(335:365));

%%

figure(2)

title 'Monatssummen'

bar(QuseM,'grouped');

grid

legend(['T{ci}=',num2str(Tci),'¡æ'])

xlim([1 12])

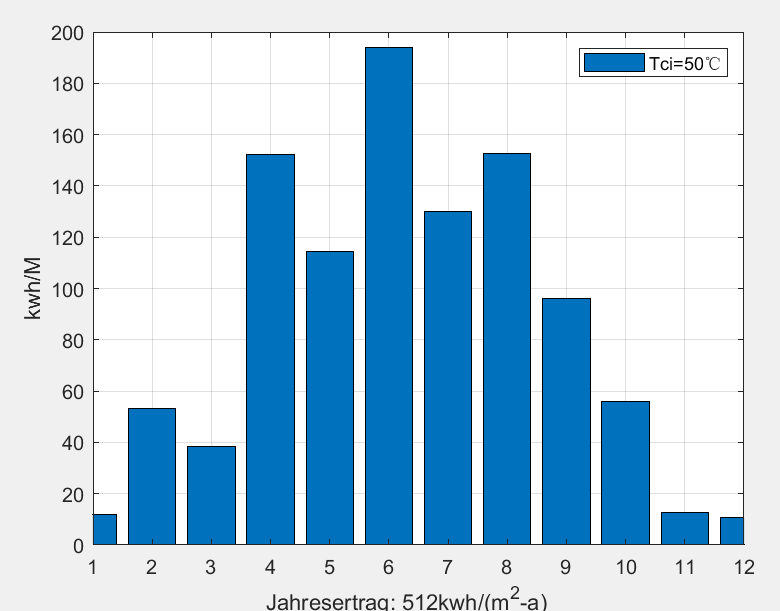
q\_sol=sum(QuseM)/2;

xlabel(['Jahresertrag: ',num2str(ceil(q\_sol)),...

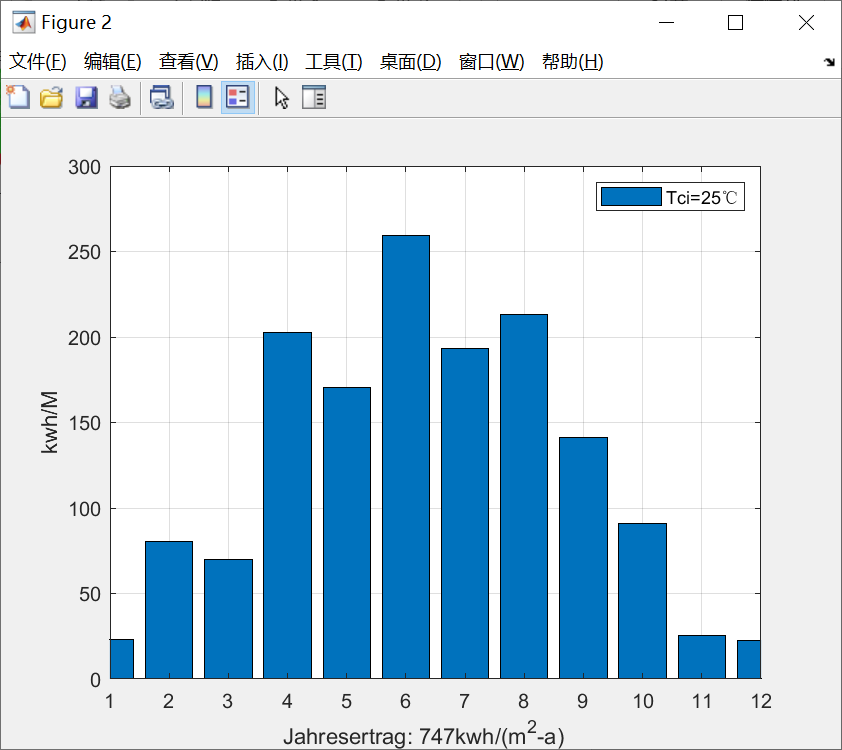
'kwh/(m^2-a) : '])

ylabel('kwh/M')

Wenn Tci=50



Wenn Tci=25



Wenn Tci=75

