Masterarbeit

Echtzeitsimulation eines Solargenerators für eine hochfliegende Solarplattform

Daniel Ackermann

Prüfer	:	Prof. Dr.-Ing. Harald Michalik
	:	Prof. Dr.-Ing. Stefan Levedag
Betreuer	:	Dipl.-Ing. Andreas Bierig
Eingereicht am	:	1. Januar 2021

Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Masterarbeit selbstständig verfasst sowie alle benutzten Quellen und Hilfsmittel vollständig angegeben habe und dass die Arbeit nicht bereits als Prüfungsarbeit vorgelegen hat.

Braunschweig, den 1. Januar 2021

Kurzfassung

Inhalt dieser Masterarbeit ist die Entwicklung eines Simulationsmodells für den Solargenerator einer hochfliegenden unbemannten Plattform im Rahmen des DLR Querschnittsprojekts HAP (High Altitude Plattform). Das Modell des Solargenerators ist eine benötigte Komponente des Hardware in the Loop Teststands und soll zusammen mit weiteren Modellen im Verbund auf einem dSPACE-Echtzeitrechner simuliert werden, um physische Komponenten zu testen und virtuelle Flüge zu ermöglichen. Gegenüber einfachen Modellen zur Energieertragsbetrachtung eines HAPS (High Altitude Pseudo-Satellite) zeichnet sich dieses Solargeneratormodell durch mehrere Besonderheiten aus. Das energiebringende Licht wird in seinem ganzen Spektrum modelliert, es werden Triple-Junction-Solarzellen abgebildet, die gesamte elektrische Verschaltung wirkt mit ihrem Einfluss und die Verformung des Leichtbauflugzeugs wird mit berücksichtigt. Das Simulationsmodell konnte erfolgreich in das Flugmechanikmodell integriert werden und erste Simulationen zeigen Details, die zuvor in vereinfachten Betrachtungen verborgen blieben. Das gesamte Flugmodell wurde auf das dSPACE-System portiert und kann durch geeignete Maßnahmen die geforderte Simulationsfrequenz von 100 Hz erfüllen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

DLR	Deutsches Zentrum für Luft- und Raumfahrt
EQE	externe Quanteneffizienz
FMU	Functional Mock-up Unit
GZ	gesetzliche Zeit
HAP	High Altitude Plattform
HAPS	High Altitude Pseudo-Satellite
HiL	Hardware in the Loop
MPP	Maximum Power Point
MPPT	Maximum Power Point Tracker
NREL	National Renewable Energy Laboratory
SPA	Solar Position Algorithm
SIC	Simulink Implementation Container
WOZ	wahre Ortszeit
ZG	Zeitgleichung

Symbolverzeichnis

Lateinische Buchstaben

Abkürzung	*Variable*	*Einheit*
a	Tastzeit	-
d	Tag des Jahres	-
E	Bestrahlungsstärke	W/m^2
E_S	Solarkonstante	W/m^2
I	Strom	A
I_K	Kurzschlussstrom	A
I_{mpp}	Strom im MPP	A
I_{Ph}	Photostrom	A
I_{sc}	Short-circuit current	A
p	Luftdruck	Pa
P	Leistung	W
P_{max}	Maximale Leistung	W
R	Widerstand	Ω
T	Temperatur	°C
TK_{Voc}	Temperaturkoeffizient der Leerlaufspannung	mV/K
TK_{Isc}	Temperaturkoeffizient des Kurzschlussstroms	mA/K
U	Spannung	V
U_Kl	Klemmenspannung	V
U_L	Leerlaufspannung	V
U_{mpp}	Spannung im MPP	V
U_{oc}	Open-circuit voltage	V
V_{EAS}	Equivalent airspeed	m/s

Griechische Buchstaben

Abkürzung	*Variable*	*Einheit*
α_S	Topozentrischer Azimutwinkel der Sonne	deg
γ_S	Topozentrischer Elevationswinkel der Sonne	deg
δ	Deklination	deg
η	Wirkungsgrad	%
θ	Zenitwinkel bzw. Einfallswinkel	deg
θ_S	Topozentrischer Zenitwinkel der Sonne	deg
λ	Längengrad	deg
φ	Breitengrad	deg
Φ	Rollwinkel	deg
Θ	Nickwinkel	deg
Ψ	Gierwinkel	deg
ω	Stundenwinkel	deg

1 Einleitung

Im Rahmen des DLR Querschnittsprojekts HAP (High Altitude Plattform) wird ein hochfliegendes solarbetriebenes Flugzeug für langandauernde Flüge in Höhen zwischen 15,5 und 22 km entwickelt (siehe Abbildung 1.1). Dieses Flugzeug soll zukünftig Aufgaben übernehmen, die traditionell von Kleinsatelliten in niedrigen Orbits erfüllt werden. Diese Art von Flugobjekten zählen in die Kategorie eines High Altitude Pseudo-Satellite (HAPS). Das Ziel einer Höhenplattform ist es, eine möglichst lange Stationierbarkeit in großer Höhe zu erreichen, damit sie konkurrenzfähig werden. Dafür ist bereits in der Konzeptentwicklung sowie über den gesamten Entwicklungsprozess hinweg eine energetische Betrachtung des Systems besonders wichtig.

Abbildung 1.1: HAP - Hochfliegende unbemannte Plattform (künstlerische Darstellung)

1.1 Motivation

Der Solargenerator und die Batterien eines HAPS bestimmen weite Teile in der Auslegung der Plattform zum Entwicklungsbeginn und im Betrieb die Fähigkeit der Stationierbarkeit.

Unter der Vorgabe möglicher Mission ergeben sich Szenarien, mit denen sich die maximale bereitgestellte Leistung von der Sonne für das System bestimmen lässt. Der Leistungsumsetzungsgrad des Gesamtsystem lässt sich wirtschaftlich sinnvoll nur mit Simulationen beantworten. Grund sind die sehr hohen Anschaffungskosten der speziellen Solarzellen, die großflächige Test unter Einsatzbedingungen erschweren. Eine Simulation hat neben den geringeren Kosten noch deutlich mehr Vorteile. So sind Tests des Solargenerators unter verschiedensten und reproduzierbaren Bedingungen möglich, die unabhängig vom realen Ort und Wetter sind. Außerdem existieren keinerlei Risiken während der Testdurchführung.

Für die Integrationstests des ersten Prototyps von HAP wird ein Hardware in the Loop (HiL) Teststand aufgebaut. Mit diesem Teststand können dem Flugzeug am Boden zur Simulation eines Fluges realistische Daten übermittelt werden. Hierfür sind die Umweltbedingungen, die Flugdynamik sowie ein Teil der Systeme zu simulieren, zu denen auch der Solargenerator zählt. Die Simulationen werden von einem Echtzeitrechner ausgeführt, der die fehlenden Teile des Prototyps durch Modelle ergänzt. Neben dem Testen der Hardware erlaubt der HiL Teststand mit dem Durchführen der virtuellen Flüge auch das Personal unter den späteren Einsatzbedingungen zu trainieren. Neben möglichen Simulationen zur Energieertragsbestimmung, ist ein Solargeneratormodell demnach auch für den Einsatz in einen HiL Teststand sinnvoll.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung eines Solargeneratormodells für das Flugzeug HAP in MATLAB/Simulink. Dieses soll alle relevanten Abhängigkeiten von der Umwelt erfassen und das interne Verhalten hinreichend genau wiedergeben. Die Funktion des Modells ist die Bestimmung des aktuellen Energieertrags. Die Umsetzung mit Simulink ist dadurch begründet, dass bestehende Modelle der Simulationsumgebung mit Simulink erstellt wurden. Andererseits wird damit die Kompatibilität des Codegenerierungsprozesses für das dSPACE-System gewährleistet. Diese wird benötigt, da das zweite wichtige Ziel dieser Arbeit die Ausführbarkeit und Echtzeitfähigkeit des Modells auf einem dSPACE-Echtzeitrechner ist. Der HiL Teststand soll mit dem Solargeneratormodell vervollständigt werden, damit dessen Systemparameter betrachtet und genutzt werden können. Die Umsetzung des Solargeneratormodells für HAP stellt einen Anwendungsfall dar, wobei das Konzept der Modellbildung des Solargenerators auch auf vergleichbare Höhenplattformen oder Solarflugzeuge angewendet werden kann. Die offizielle Aufgabenstellung ist im Anhang A.1 angehängt.

1.3 Aufbau dieser Arbeit

Diese Arbeit lässt sich in drei wesentliche Teile gliedern. Zunächst werden in Kapitel 2 Grundlagen zu genutzten mathematischen und elektrischen Themen sowie Wissen über allgemeine Modellbildung und Echtzeitfähigkeit vermittelt. Im gleichen Zug wird der Stand der Technik in Bezug auf Solargeneratoren dargestellt und Besonderheiten im flugzeugintegrierten Fall aufgezeigt. Auf Basis dieses Wissens wird in Kapitel 3 nach dem Bottom-Up Ansatz das Solargeneratormodell entwickelt, wobei mit der Modellierung einer einzelnen Solarzelle begonnen wird. Nachdem das Solargeneratormodell vervollständigt und anschließend in das Flugmechanikmodell integriert wurde, ist ein testbares Gesamtmodell entstanden. Dieses wird in Kapitel 4 auf den dSPACE-Echtzeitrechner übertragen und es wird anschließend eine Laufzeitbetrachtung durchgeführt, die die Echtzeitfähigkeit überprüft.

2 Theoretische Grundlagen

Das Kapitel der Theoretischen Grundlagen fasst für das folgende Verständnis dieser Arbeit die wichtigsten Themenbereiche und Begriffe zusammen. Zunächst werden dazu mathematische und physikalische Grundlagen aufgefasst und wiederholt, auf die im Verlauf dieser Arbeit zurückgegriffen werden.

Ein Fokus dieser Arbeit liegt in der Modellierung von Solarzellen. Damit diese korrekt abgebildet werden können, müssen sie auch realistischen Bedingungen ausgesetzt werden. Der wichtigste Einflussfaktor einer Solarzelle ist die Sonne und ihr ausgesendetes Licht. Deshalb wird in diesem Kapitel zunächst die Sonne in der dazu nötigen Tiefe beschrieben.

Anschließend werden die wesentlichen Komponenten eines Solargenerators beschrieben. Dazu wird zunächst eine Einführung in den Aufbau und die Wirkungsweise von Solarzellen gegeben. Dies ist für eine genaue Modellbildung wichtig. Im Weiteren wird darauf eingegangen, wie sich Solarzellen verhalten, wenn sie miteinander in größere Verbünde geschaltet werden. Als Abschluss zum Themengebiet der solaren Energiegewinnung sollen noch die Stromrichter beleuchtet werden, die für den praktischen Einsatz von Solarzellen nötig sind.

Diese Arbeit beschäftigt sich mit der wesentlichen Aufgabe der Modellierung eines Solargenerators für eine spätere Echtzeitsimulation. Dazu wird im letzten Abschnitt dieses Kapitels näher auf die allgemeine Modellbildung eingegangen und anschließend der Bereich der Echtzeitanwendungen näher beschrieben. Da in dieser Arbeit die erstellte Anwendung auf einem HiL-Teststand eingesetzt werden soll, wird das Konzept und der Zweck eines solchen Teststands zum Abschluss erläutert.

2.1 Mathematische und physikalische Grundlagen

Für die Modellierung und spätere Simulation des Solargenerators sind wesentliche Grundlagen aus der Elektrotechnik notwendig, durch die das korrekte Verhalten der verschalteten

Solarzellen aufgezeigt werden kann. Im Verlauf wird eine Bibliothek zur Leistungsflusssimulation implementiert, für die diese Grundlagen essentiell sind. Für eine spätere Berechnung der Einfallswinkel des Sonnenlichts auf die Solarzellen, ist es notwendig, die einzelnen Zellen korrekt im Raum zu orientieren. Die Grundlagen zur Durchführung dieser Rotationen, ebenso wie für Koordinatentransformationen, werden zum Abschluss in diesem Abschnitt vermittelt.

2.1.1 Ohm'sches Gesetz

Elektrischer Strom I entsteht durch bewegte Ladungsträger und ist definiert durch einen Ladungsfluss pro Zeit. Verursacht wird die Bewegung der Ladungsträger, d. h. der Strom, durch die elektrische Spannung U. Eine elektrische Spannung beschreibt die Größe der Potentialdifferenz zwischen 2 Punkten, wobei eine Potentialdifferenz durch voneinander getrennte positive und negative Ladungsträger hervorgerufen wird. Als dritte wichtige elektrische Größe existiert der elektrische Widerstand R. Er ist ein Maß für die Hemmung des Ladungstransports. Diese Größen stehen in einem Zusammenhang, der durch das Ohm'sche Gesetz beschrieben wird:

$$U = R \cdot I \tag{2.1}$$

Eine Aussage des Ohm'schen Gesetzes ist es, dass an einem ohmschen Widerstand sich mit zunehmender Spannung auch der Strom proportional dazu ändert. Metallische Leiter sind das beste Beispiel für ohmsche Widerstände, da sie einen konstanten Widerstandswert besitzen (bei konstanter Temperatur). Wird an einem ohmschen Widerstand eine Spannung angelegt und dazu der Strom gemessen, zeigt sich bei einer Erhöhung der Spannung ein linear steigender Verlauf des Stroms. Deswegen werden ohmsche Widerstände auch als lineare Widerstände bezeichnet. Nichtlineare Verläufe hingegen besitzen Halbleiter [2].

2.1.2 Kirchhoff'sche Regeln

Eine praktische elektrische Schaltung besteht für gewöhnlich aus mehr Bauteilen, als nur einer Spannungsquelle und einem Verbraucher. Elektrische Schaltungen bilden meist Netze durch Abzweigungen. Dadurch entstehen Knoten, an denen Ströme verzweigen und Maschen, die einen möglichen geschlossenen Umlauf für einen Stromfluss darstellen. Das Verhalten der elektrischen Kenngrößen in solchen Netzen wird durch die Kirchhoff'schen Regeln beschrieben, die nachfolgend erklärt werden [2].

Das 1. Kirchhoff'sche Gesetz wird als Knotenregel bezeichnet. Sie beschreibt das Verhalten von n Strömen I an einem Knoten und lautet:

$$\sum_{i=1}^{n} I_i = 0 \tag{2.2}$$

Die Regel besagt, dass an einem Knoten alle eingehenden Strome in der Summe genau so groß sein müssen, wie alle ausgehenden Ströme. Würde dies nicht gelten, wäre das eine Verletzung des Ladungserhaltungssatzes.

Das 2. Kirchhoff'sche Gesetz ist die Maschenregel und beschreibt das Verhalten von n Umlaufspannungen U innerhalb einer Masche. Der dabei gewählte Umlaufsinn kann frei gewählt werden und gibt an, ob die einzelnen Spannungen an den Zählpfeilen positiv oder negativ gezählt werden, je nachdem ob sie mit oder gegen den Umlaufsinn gehen. Die Maschenregel lautet:

$$\sum_{i=1}^{n} U_i = 0 \tag{2.3}$$

Die Maschenregel besagt, das die Summe aller Spannungen innerhalb einer Masche gleich 0 sein muss. Die Maschenregel leitet sich aus dem Energieerhaltungssatz ab, da innerhalb eines geschlossenen Stromkreises (Masche) die zugeführte Arbeit gleich groß der abgeführten Arbeit sein muss [2].

2.1.3 Anwendung der Kirchhoff'schen Gesetze

Aus den Kirchhoff'schen Regeln lassen sich, unter Zuhilfenahme des Ohm'schen Gesetzes Regeln für Reihen- und Parallelschaltungen herleiten. Es soll zunächst eine Reihenschaltung von mehreren Widerständen betrachtet werden. Innerhalb der Reihenschaltung gibt es keine Abzweigungen, durch die sich der Strom aufteilen kann. Der Strom durch alle Elemente muss deshalb der gleiche sein, da es nur einen eingehenden Pfad und einen ausgehenden Pfad zwischen 2 Bauteilen gibt. Weiter lässt sich durch den Maschensatz sagen, dass die Spannung auf der Eingangsseite über alle Widerstände abfallen muss, wodurch gilt:

$$U_0 = U_1 + U_2 + ... + U_n \tag{2.4}$$

Mithilfe des Ohm'schen Gesetzes lässt sich dies umformen und vereinfachen zu:

$$U_0 = IR_1 + IR_2 + ... + IR_n \tag{2.5}$$

$$U_0 = I * (R_1 + R_2 + ... + R_n) \tag{2.6}$$

$$U_0/I = R_{\text{ges}} = R_1 + R_2 + ... + R_n = \sum_{i=1}^{n} R_i \tag{2.7}$$

Damit ergibt sich, dass der Gesamtwiderstand gleich der Summe der einzelnen Widerstände entspricht.

Im nächsten Schritt wird die Reihenschaltung von mehreren Spannungsquellen betrachtet. Sie bewirkt nach der Maschenregel eine Addition der einzelnen Spannungen, sofern sie mit entgegengesetzten Polen zusammengeschaltet werden. Die Strome müssen nach der Knotenregel durch alle in Reihe geschalteten Spannungsquellen gleich sein. Dies hat zur Folge, dass sich an den realen Spannungsquellen nur ein maximaler Strom einstellen kann, der durch die schwächste reale Stromquelle bestimmt wird. Die Begriffe der Strom- und Spannungsquelle lassen sich an dieser Stelle synonym verwenden, da reale Quellen eine maximale Spannung und einen maximalen Strom besitzen. Dies ermöglicht eine Umrechnung einer realen Spannungs- in eine reale Stromquelle und umgekehrt.

Die Parallelschaltung von mehreren Widerständen soll als nächstes betrachtet werden. Hierbei ergibt sich durch die Maschenregel, dass über jedem verzweigtem Pfad die gleiche Spannung abfallen muss, da die gebildete Masche über 2 parallelen Pfaden die gleiche Spannung mit umgekehrten Vorzeichen fordert. Die einzelnen eingehenden Ströme können sich dabei unterscheiden und sind in der Summe gleich dem Strom von der Stromquelle wodurch gilt:

$$I = I_1 + I_2 + ... + I_n \tag{2.8}$$

Durch Anwendung des Ohm'schen Gesetzes und unter der Tatsache, dass die Spannung an allen Pfaden gleich ist, lässt sich dies umformen zu:

$$I = \frac{U_0}{R_1} + \frac{U_0}{R_2} + ... + \frac{U_0}{R_n} \tag{2.9}$$

$$I = U_0 \cdot \left(\frac{1}{R_1} + \frac{1}{R_2} + ... + \frac{1}{R_n} \right) \tag{2.10}$$

$$I = U_0 \frac{1}{R_{\text{ges}}} \tag{2.11}$$

Damit ergibt sich die Formel zur Berechnung des Gesamtwiderstands innerhalb einer Paral-

lelschaltung zu:

$$\frac{1}{R_{\text{ges}}} = \sum_{i=1}^{n} \frac{1}{R_i} \tag{2.12}$$

Abschließend müsste der Vollständigkeit halber die Parallelschaltung von realen Spannungsquellen folgen. Die genaue Betrachtung dazu soll allerdings erst in Abschnitt 3.5 behandelt werden. Vorweg kann bereits gesagt werden, dass sich die Strome der einzelnen realen Spannungsquellen summieren, sofern sie die gleiche Klemmenspannung besitzen. Andernfalls führt dies zu unerwünschten Ausgleichsströmen zwischen den einzelnen Quellen.

2.1.4 Rotationen und Transformationen von Vektoren im $\mathbb{R}^3$

Für spätere Berechnungen von Orientierungen und Rotationen im dreidimensionalen Raum ($\mathbb{R}^3$) werden mathematische Grundlagen zu Vektoren und Koordinatentransformationen benötigt. Dazu zunächst die Definition eines Vektors. Ein Vektor

$$v = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

gibt eine Richtung im $\mathbb{R}^3$ an, die durch seine Längeneigenschaft gleichzeitig einer Position relativ zum Ursprung im Raum entspricht. Er besitzt eine x, y und z Komponente, die dem Vektor eine entsprechende Position auf der jeweiligen Achse zuweist. Mithilfe einer Transformationsmatrix bzw. Drehmatrix ist es möglich, eine Koordinatentransformationen über die Eulerwinkel durchzuführen und eine raumfeste Positionsangabe in eine körperfeste zu überführen oder umgekehrt. Die Transformation ändert nicht den Vektor, sondern nur seine Darstellung, die sich vom alten in das neue Koordinatensystem (auch Frame genannt) ändert. Es ist stattdessen möglich, nicht das Koordinatensystem zu drehen, sondern den Vektor im betrachteten Frame, was später behandelt wird. In Beiden Fällen wird eine 3×3 Matrix A benötigt, die eine Umrechnung des Vektors v_1 mit

$$v_2 = Av_1$$

ermöglicht. Die Drehmatrizen für eine Rotation um den Ursprung entlang der Koordinatenachsen sind bekannt [3, S. 49 f.]. Für A lässt sich eine der nachfolgenden Matrizen R einsetzen, die eine Drehung des Koordinatensystems um die globale x, y oder z-Achse in der

Größe des genutzten Winkels erlaubt:

$$R_x(\Phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\Phi & sin\Phi \\ 0 & -sin\Phi & cos\Phi \end{pmatrix} \tag{2.13}$$

$$R_y(\Theta) = \begin{pmatrix} cos\Theta & 0 & -sin\Theta \\ 0 & 1 & 0 \\ sin\Theta & 0 & cos\Theta \end{pmatrix} \tag{2.14}$$

$$R_z(\Psi) = \begin{pmatrix} cos\Psi & sin\Psi & 0 \\ -sin\Psi & cos\Psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.15}$$

Die gegebenen Drehmatrizen drehen das Koordinatensystem in einem Rechtssystem im mathematisch positivem Sinn, also gegen den Uhrzeigersinn. Das Rechtssystem kann mithilfe der Drei-Finger-Regel hergeleitet werden und der Drehsinn über die Rechte-Hand-Regel. Für die Transformation von einem Koordinatensystem in ein anderes, ist es meist notwendig mehr als eine Drehung durchzuführen, sofern nur entlang der Koordinatenachsen gedreht wird. Diese Reihe von Rotationen wird durch mehrfache Anwendung der gegebenen Matrizen erreicht.

Als Beispiel soll die, für die Luftfahrt übliche, Transformation vom globalen Koordinatensystem in das lokale dienen. Das globale Koordinatensystem wird in der Luftfahrt auch als geodätisches Koordinatensystem bezeichnet und wird über die lokale Horizontalebene der Erdoberfläche aufgespannt. Die Koordinatenachsen zeigen in Richtung Nord (x-Achse), Ost (y-Achse) und in Richtung des Gewichtsvektors nach unten (z-Achse), weswegen es auch mit NED-Frame (für **N**orth, **E**ast, **D**own) abgekürzt wird. Das lokale Koordinatensystem des Flugzeugs wird als flugzeugfestes Koordinatensystem bezeichnet und hat seine X-Achse in Längsrichtung des Flugzeugs nach vorne, die Y-Achse in Richtung der rechten Tragfläche und die Z-Achse nach unten. Die Lagewinkel des Flugzeugs repräsentieren die Abweichungen des flugzeugfesten Koordinatensystems zum geodätischen. Sie heißen Rollwinkel Φ, Nickwinkel Θ und Gierwinkel Ψ und geben die Abweichungen der globalen Achsen zu den lokalen Achsen in dieser Reihenfolge an. Die Bezeichnung der globalen Koordinatenachsen soll im Folgenden über die Kleinbuchstaben xyz erfolgen und die der lokalen Achsen mit den gleichen Großbuchstaben XYZ [4].

Eine Überführung eines Vektors vom geodätischen Koordinatensystem $v_g = (x_g, y_g, z_g)^T$ in das flugzeugfeste Koordinatensystem $v_f = (x_f, y_f, z_f)^T$ ist in der Luftfahrtnorm 9300 definiert. Danach wird begonnen das geodätische Koordinatensystem um die z-Achse um den Gierwinkel Ψ zu drehen. Es entsteht ein neues Hilfskoordinatensystem mit den Achsen x′y′z′, wobei z = z′ entspricht. Danach wird um die neue y′-Achse um den Nickwinkel Θ gedreht, womit zunächst die gesuchte X-Achse entsteht und die Hilfsachsen y″ und z″. Im letzten Schritt wird um den Rollwinkel Φ entlang der X-Achse gedreht, wodurch die Y und Z-Achse entstehen und das flugzeugfeste Koordinatensystem vervollständigt wird [3] [5]. Dies wird auch z-y′-x″-Konvention bezeichnet, da es die Rotationsreihenfolge angibt. Die mathematische Schreibweise lautet:

$$v_f = R_x(\Phi)R_y(\Theta)R_z(\Psi)v_g = R_{fg}v_g \tag{2.16}$$

Wie hier sichtbar wird ist die Reihenfolge der Rotationen von rechts nach links zu lesen, da die Matrizenmultiplikation nicht kommutativ ist. Die Kette von Drehmatrizen lässt sich auch in eine einzelne Matrix zusammenrechnen, die hier R_{fg} genannt wird. Eine Rücktransformation ist über die inverse der Matrix möglich, die bei Drehmatrizen immer der transponierten Matrix entspricht [3] [4]. Daraus ergibt sich:

$$v_g = R_{fg}^T v_f = R_z(\Psi)^T v_f R_y(\Theta)^T R_x(\Phi)^T v_f = R_{gf}v_f \tag{2.17}$$

Beim Auflösen der transponierten Gesamtmatrix muss außerdem darauf geachtet werden, dass sich die Reihenfolge der Drehungen vertauscht. Bei genauerer Betrachtung der ursprünglichen Matrizen aus den Formeln (2.13), (2.14) und (2.15) fällt auf, dass die Transponierte der Drehmatrizen nur ein vertauschtes Vorzeichen bei den Sinus-Einträgen besitzt. Kurzgefasst ist dies gleichzusetzen mit dem negativen Rotationswinkel. Damit ergibt sich:

$$R_x(\Phi)^T = R_x(-\Phi) \qquad R_y(\Theta)^T = R_y(-\Theta) \qquad R_z(\Psi)^T = R_z(-\Psi) \tag{2.18}$$

Nachdem Koordinatentransformationen behandelt wurden, soll noch der Fall der Rotationen von Vektoren erklärt werden. Hierbei bleibt das Koordinatensystem gleich und der Vektor innerhalb des Frames wird im Ursprung entlang einer Koordinatenachse gedreht. Die zu verwendenden Drehmatrizen sind gleich den zuvor verwendeten aus den Formeln (2.13), (2.14) und (2.15), nur dass hier die Matrizen zu transponieren sind. Damit ergibt sich für die Drehung eines Vektors um einen bestimmten Winkel innerhalb des Koordinatensystems das gleiche Ergebnis wie die Drehung des Koordinatensystems um den gleichen Winkel in umgekehrter Richtung.

2.2 Eigenschaften der Sonne

Die Sonne ist die größte Energiequelle für die Erde [6, S. 61]. Mit fortschreitendem technologischem Wissen ist es der Menschheit gelungen diese Energiequelle für sie nutzbar zu machen beispielsweise durch Photovoltaik oder Solarthermie. Es sollte allgemein bekannt sein, dass für eine Solarzelle der wichtigste Faktor zur Leistungserzeugung die Sonne selbst ist. Damit stehen beide in einem derart wichtigen Verhältnis zueinander, dass die Sonne und ihre Strahlungseigenschaften beschrieben werden müssen.

Der Abstand zwischen Erde und Sonne beträgt ungefähr 150 Mio. km. Diese Entfernung unterliegt allerdings einer jährlichen Schwankung, die durch den wechselnden Abstand zwischen Erde und Sonne erklärt wird. Dabei ist die mittlere extraterrestrische Leistungsdichte bzw. Bestrahlungsstärke der Sonne als Solarkonstante E_S mit $1367\frac{W}{m^2}$ definiert [6, S. 41].

Im Folgenden wird genauer auf den örtlichen Verlauf der Sonne am Horizont und das Spektrum der Sonne eingegangen. Die Position und der Verlauf sind wichtig für eine Bestimmung des Einfallswinkels auf die Solarzelle und das Spektrum des Lichts gibt an, welche Leistung mit der Energie des Lichts geleistet werden kann.

2.2.1 Sonnenposition

Die Position der Sonne wird aus topozentrischer Sicht, also von der Eroberfläche aus betrachtet, mit Kugelkoordinaten angegeben. Für die Positionsangabe werden entsprechend zwei Winkel benötigt. Der erste Winkel ist der Azimut α_S und beschreibt die Drehung der Sonne, um die Zenitachse, vom Norden an beginnend und im Uhrzeigersinn laufend. Der Azimut besitzt damit den Wertebereich von 0 bis 360°. Siehe dazu auch Abbildung 2.1. In der Literatur wird der Azimut häufig von Süden an beginnend definiert und kann damit Werte bis ± 180° annehmen.

Für die Höhenangabe der Sonne gibt es zwei Winkel, die angegeben werden können und von denen der eine Winkel jeweils in den Anderen ungerechnet werden kann. Dazu ist zunächst die Elevation γ_S (auch Höhenwinkel) zu nennen, welche die Höhe der Sonne über dem Horizont angibt und +90° bis -90° annehmen kann. Komplementär steht auch die Angabe über den Zenitwinkel θ_S zur Verfügung. Dieser Winkel beschreibt die Abweichung der Sonne vom Zenit im Bereich von 0° bis 180°. Die Elevation und der Zenitwinkel lassen sich mit folgender

Formel ineinander umrechnen [7, S. 15]:

$$\gamma_S = 90 - \theta_S \tag{2.19}$$

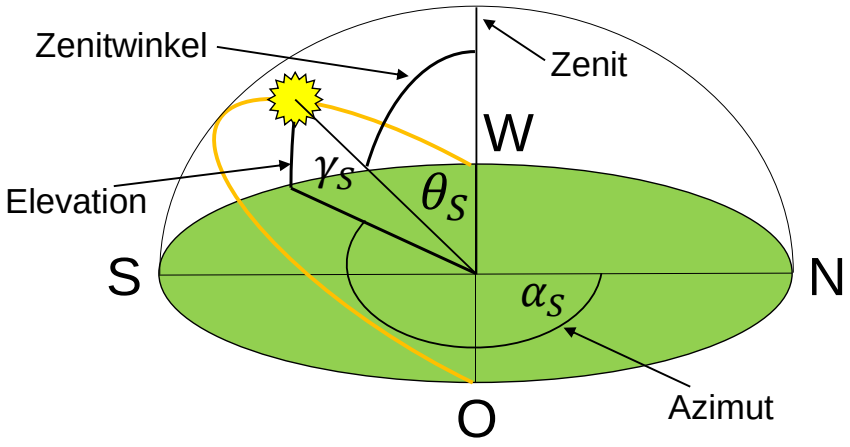


Abbildung 2.1: Positionsangabe der Sonne in Kugelkoordinaten

In Abbildung 2.1 ist des Weiteren in orange der Verlauf der Sonne angedeutet, den sie im Tagesverlauf beschreibt. Dieser Bogen wird als Tagbogen bezeichnet oder kann als Sonnenbahn beschrieben werden. Vereinfachte Formeln zur Beschreibung dieser Sonnenbahn sind gut bekannt [8] [6] [7] [9]. Im Folgenden werden diese Formeln zusammengefasst und beschrieben.

Zur Berechnung der Sonnenposition werden Angaben zum Ort und zur Zeit benötigt. Die Eingangsparameter lauten wie folgt:

Ort: Breitengrad φ
Längengrad λ

Zeit: Datum mit Tag, $Monat$
Uhrzeit in Stunden h

Zunächst wird aus dem Datum die Tagesnummer d über folgende Formel bestimmt:

$$d = 30,3 \cdot (Mon. - 1) + Tag \tag{2.20}$$

Mit der Tagesnummer kann als nächstes die Deklination der Erde annähernd bestimmt werden. Die Deklination δ entsteht durch die Neigung der Erdachse und schwankt mit einer

jährlichen Periode zwischen ± 23,45°. Die Formel zur annähernden Bestimmung lautet:

$$\delta = 23,45° \cdot sin\left(360° \cdot \frac{284 + d}{365}\right) \tag{2.21}$$

Eine weitere wichtige Größe ist der Stundenwinkel ω. Dieser beschreibt den zeitlichen Verlauf der Sonne über einen Tag und ist zum Zeitpunkt des Sonnenhöchststands mit $\omega = 0°$ definiert. Das Vorzeichen des Stundenwinkels ist vormittags positiv und nachmittags negativ. Die wahre Ortszeit (WOZ) ist, wie der Stundenwinkel über den Sonnenhöchststand definiert und nach der WOZ beträgt die Zeit zum Höchststand genau 12 Uhr. Für die WOZ werden keine Zeitzonen betrachtet. Für die Benutzung der Formeln ist die Zeit der WOZ in der Einheit Stunden $[h]$ anzugeben (12:45 Uhr entspricht 12,75 h). Diese Zusammenhänge werden mit folgender Formel zusammengefasst:

$$\omega = (12h - WOZ) * \frac{15°}{h} \tag{2.22}$$

Die einfachste Formel zur Bestimmung der Sonnenstands beschreibt die Elevation γ_S bzw. den Zenitwinkel θ_S der Sonne in Abhängigkeit des Stundenwinkels ω. Dazu werden lediglich eine Angabe des Breitengrads φ und die der Deklination δ benötigt.

$$cos\theta_S = sin\delta \cdot sin\varphi + cos\delta \cdot cos\varphi \cdot cos\omega = sin\gamma_S \tag{2.23}$$

Eine Vervollständigung der Positionsangabe der Sonne durch den Azimut α_S kann durch folgende Formel vorgenommen werden:

$$\alpha_S = \begin{cases} 180° - arccos\left(\frac{sin\gamma_S \cdot sin\varphi - sin\delta}{cos\gamma_S \cdot cos\varphi}\right), & \text{wenn } 0 < WOZ < 12 \text{ h} \\ 180° + arccos\left(\frac{sin\gamma_S \cdot sin\varphi - sin\delta}{cos\gamma_S \cdot cos\varphi}\right), & \text{sonst} \end{cases} \tag{2.24}$$

Bei programmatischer Umsetzung kann auch folgende Formel verwendet werden:

$$\alpha_S = atan2(sin\omega, cos\omega \cdot sin\varphi - tan\delta \cdot cos\varphi) + 180° \tag{2.25}$$

Die hierbei hinzuaddierten 180° sind nötig, um den Azimut im Norden beginnen zu lassen. Ist der Beginn des Azimuts mit Süden definiert, so ist diese Angabe wegzulassen.

Soll die Angabe der Sonnenposition nicht der WOZ, sondern der gesetzlichen Zeit (GZ) entsprechen muss noch eine Korrektur der Zeitangabe erfolgen. Hierbei wird auch die jeweilige Zeitzone berücksichtigt. Dazu muss außerdem die Zeitgleichung (ZG) mit einbezogen werden, die den Unterschied zwischen der wahren und der mittleren Ortszeit beschreibt. Die beiden

Formeln lauten:

$$ZG = \left[0,123 \cdot cos\left(360° \frac{88+d}{365}\right) - 0,167 \cdot sin\left(720° \frac{10+d}{365}\right)\right] h \tag{2.26}$$

und

$$WOZ = GZ + (\lambda_0 - \lambda)\frac{h}{15°} + ZG \tag{2.27}$$

Wobei hier λ_0 den Längengrad der Zeitzone angibt und damit die Differenz zum lokalen Längengrad λ beschreibt.

Die hier gezeigten Formeln folgen damit weitestgehend dem in der DIN 5034 Teil 2 beschriebenen Algorithmus zu Positionsberechnung der Sonne [10]. Dieser unterscheidet sich in den genäherten Formeln zur Bestimmung der Deklination und der Zeitgleichung.

2.2.2 Spektrum der Sonne

Das Spektrum und damit die Bestrahlungsstärke der Sonne ist wesentlich von der Sonnenhöhe und von den atmosphärischen Bedingungen abhängig. Dazu soll zunächst auf den Einfluss der Elevation eingegangen werden. Dieser bestimmt durch wie viele Atmosphärenmassen das Licht auf dem Weg von der Sonne zur Erde durchläuft. Die Abkürzung für Atmosphärenmasse ist AM und leitet sich von der englischen Bezeichnung Air Mass ab. Abbildung 2.2 verdeutlicht dazu den Zusammenhang zwischen AM Zahl und Sonnenstand.

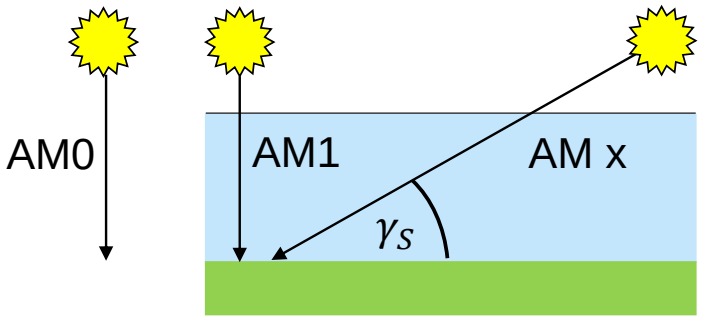


Abbildung 2.2: Verbildlichung der Bedeutung der Atmosphärenmassen (AM)

Ein AM0 Spektrum bezeichnet den Zustand ohne Einfluss der Atmosphäre und damit das extraterrestrische Spektrum. Das AM1 Spektrum liegt dann vor, wenn die Sonne genau eine Atmosphärenmasse durchläuft und dies gilt nur dann, wenn die Sonne im Zenit steht. Dies

ist nur an wenigen Orten auf der Erde der Fall. Ein sehr geläufiges Spektrum ist dagegen das AM1,5. Es tritt bei einem Elevationswinkel von ca. 42° auf und gilt für die meisten Regionen der mittleren Breitengrade, wozu auch Deutschland zählt. Des Weiteren kann auch jede beliebige AM Zahl angegeben werden, wobei der Zusammenhang zur Elevation wie folgt lautet:

$$x \approx \frac{1}{sin(\gamma_S)} \tag{2.28}$$

Die Formel wird jedoch im Bereich eines sehr flachen Sonnenstands zunehmend ungenauer, da hier die runde Form der Erde nicht berücksichtigt wird. Die Formel nimmt eine flache Erde mit einer darüber gleichbleibend dicken Atmosphäre an [8].

Als nächstes soll näher auf die Zusammensetzung des Spektrums selbst eingegangen werden. In Abbildung 2.3 sind die nach dem ASTM Standard definierten Spektren für AM0 und

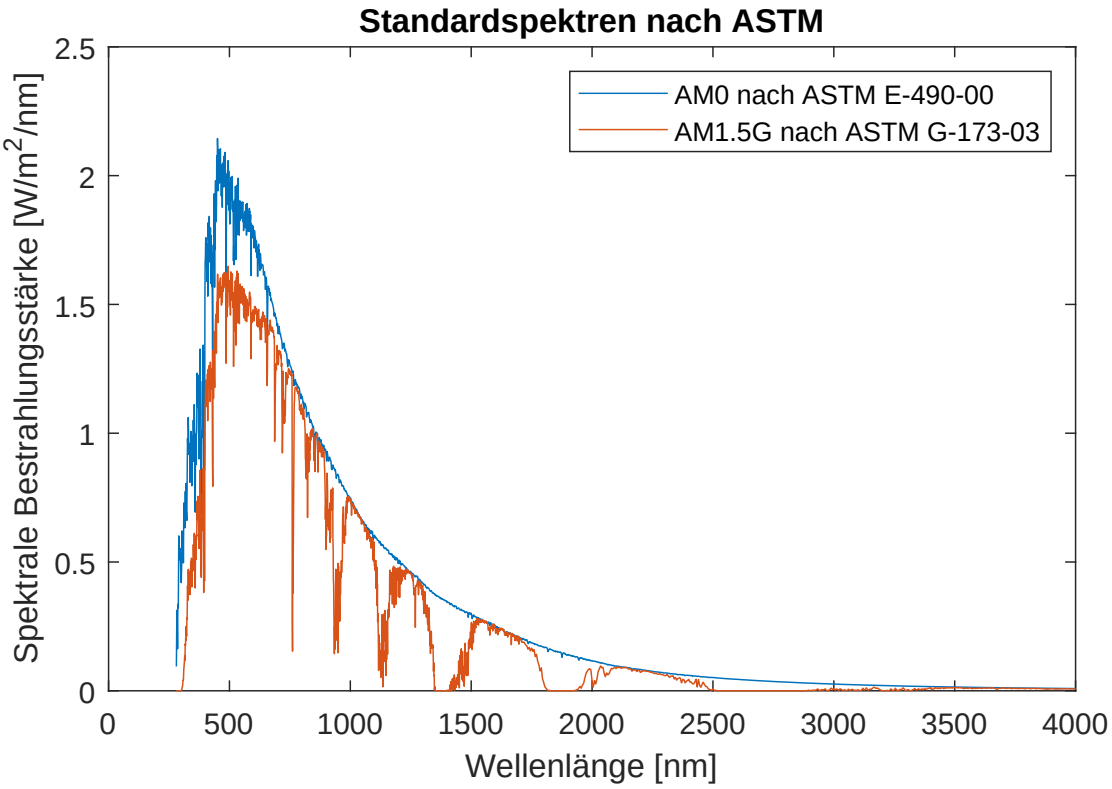


Abbildung 2.3: Standardspektren nach dem ASTM Standard

AM1,5 Bedingungen gezeigt. Diese werden meist für Solarzellen als Standardtestbedingungen genommen, um untereinander vergleichbare Werte in den Datenblättern der Solarzellen zu schaffen. Eine Aufintegration des Spektrums über alle Wellenlängen ergibt für das AM0 Spektrum, wie es in [11] zu finden ist, eine Bestrahlungsstärke von $E_{\mathrm{AM0\_ASTM\_E490}} = 1366,1\ \frac{W}{m^2}$.

Dieses Spektrum ist allerdings für einen Wellenlängenbereich von $0,1195\ \mu m$ bis $1000\ \mu m$ angegeben und übersteigt den abgebildeten Bereich deutlich.

Das AM1,5 Spektrum wie es in [12] zu sehen ist, beschreibt dabei eine Bestrahlungsstärke, wie sie in mittleren Breitengraden im Jahresmittel bei gutem Wetter herrschen. Der Buchstabenzusatz G steht dabei für die Angabe Global und umfasst die Summe aus direkter und diffuser Strahlung. Eine Integration der AM0 und AM1,5 Spektren über einen Wellenlängenbereich von $0,28\ \mu m$ bis $4,0\ \mu m$ ergibt $E^{\star}_{\text{AM0\_ASTM\_E490}} = 1347,6\ \frac{W}{m^2}$ und $E_{\text{AM1.5\_ASTM\_G173}} = 1000,4\ \frac{W}{m^2}$ als Bestrahlungsstärken. Dieser Wellenlängenbereich stellt den in Abbildung 2.3 abgebildeten Bereich dar.

Wie erwähnt, wird das Licht durch einen direkten und einen diffusen Anteil unterschieden. Der direkte Anteil ist der Teil des Lichts, der ohne Umwege und aus der direkten Sichtlinie von der Sonne zum Betrachter scheint. Das diffuse Licht entsteht erst durch das Vorhandensein der Atmosphäre. Die Gase, die sie aufbauen und darin enthaltene Partikel sorgen für Absorptionen und Streuungen. Dieser gestreute Anteil bildet das diffuse Licht. Außerdem gibt es noch Reflektionen, die am Boden entstehen. Dieser direkt reflektierte Anteil soll aber nicht weiter betrachtet werden, da die Solarzellen von HAP auf der Flügeloberseite sind und damit das von unten kommende Licht nicht oder nur schwer einfangen könnten.

Die Einflüsse der Atmosphäre werden nachfolgend genauer kategorisiert [6] [13]:

1. Reflexion von Licht: In der Atmosphäre treten Reflektionen auf, die einen Teil des Lichts wieder in den Weltraum zurück werfen. Hauptsächlich geschieht dies durch flüssiges oder gefrorenes Wasser in den Wolken. Dies ist stark von der Wolkendichte abhängig.

2. Absorption von Licht: Bei einem Vergleich der beiden Spektren in Abbildung 2.3 fällt auf, dass das AM1,5 Spektrum mehrere Intensitätseinbrüche bei bestimmten Wellenlängen aufweist (z. B. 1400 nm). Dies ist durch Absorptionen von Molekülen wie Ozon, Sauerstoff, Wasserdampf oder CO2 zu erklären, die an den entsprechenden Stellen ihre Spektrallinien besitzen.

3. Rayleigh-Streuung: Eine Ursache für das diffuse Licht ist die Rayleigh-Streuung. Sie entsteht, wenn Licht auf Moleküle oder Teilchen fällt, die kleiner als die Wellenlänge sind. Da dieses Phänomen stark wellenlängenabhängig ist und die Streuung niedriger Wellenlängen durch die Erdatmosphäre begünstigt wird, tritt es besonders im bläulichen auf und lässt den Himmel damit auch blau erscheinen.

4. Streuung an Aerosolen und Staubteilchen: Die zweite Ursache für diffuse Strahlung ist die Mie-Streuung. Die Erklärung dieser Streuung sind sphärische Objekte mit etwa gleichem

oder größerem Durchmesser gegenüber der Wellenlänge, wie etwa Staub, Rauch oder Pollen. Partikel dieser Größe existieren stark ortsabhängig und sind vermehrt in urbanen Gegenden mit Industrie zu finden.

2.3 Komponenten eines Solargenerators

Da der Kern dieser Arbeit die Erstellung eines Solargeneratormodells ist, muss zunächst geklärt werden, was ein Solargenerator ist und wie dieser aufgebaut ist. Allgemein bezeichnet ein Solargenerator einen Verbund aus mehreren Solarzellen, die insgesamt als Gleichstromquelle arbeiten. Die einzelne Solarzelle liefert elektrische Energie, indem die Energie des Sonnenlichts umgewandelt wird. Durch das Zusammenschalten mehrerer Solarzellen zu einem Solarmodul können die Leistungen um Größenordnungen gesteigert werden. Da allerdings die elektrischen Eigenschaften der Solarzellen durch Schwankungen der Bestrahlung meist nicht konstant sind, wird noch ein Stromrichter benötigt, der gleichbleibende Bedingungen für den Abnehmer schafft.

2.3.1 Solarzellen

Eine Solarzelle ist im Prinzip wie eine großflächige Photodiode aufgebaut und besteht aus zwei dotierten Halbleitern, die den photoelektrischen Effekt ausnutzen. Üblicherweise ist dabei die obere sonnenzugewandte Seite n-dotiert (negativ) und die untere p-dotiert (positiv). Durch die Berührung der beiden dotierten Halbleiterkristalle rekombinieren die ausgebildeten Elektronen und Löcher der Grenzschicht und bilden eine Sperrschicht aus. Sie entsteht durch Fehlstellen im Kristall, da die dotierten Atome ihr Elektron abgegeben haben bzw. eins aufgenommen haben und nun ionisiert sind. Die entgegengesetzt geladenen Ionen bilden damit ein elektrisches Feld aus, welches die Raumladungszone genannt wird. Zur Vervollständigung der Zelle sind auf der Ober- und Unterseite noch Metallische Kontakte angebracht. Abbildung 2.4 hilft dabei bei der Verdeutlichung.

Wenn nun ein Photon mit genug Energie in die Solarzelle eindringt und absorbiert wird, hebt es ein Elektron des Halbleitermaterials vom Valenz- und das Leitungsband. Dazu muss die quantisierte Energie des Photons größer gleich der Bandlücke des Materials sein (z. B. kristallines Silizium $\Delta E = 1,23$ eV [6, S. 84]). Dieses angeregte Elektron kann sich nun frei im Kristall bewegen, bis es an einer Fehlstelle rekombiniert.

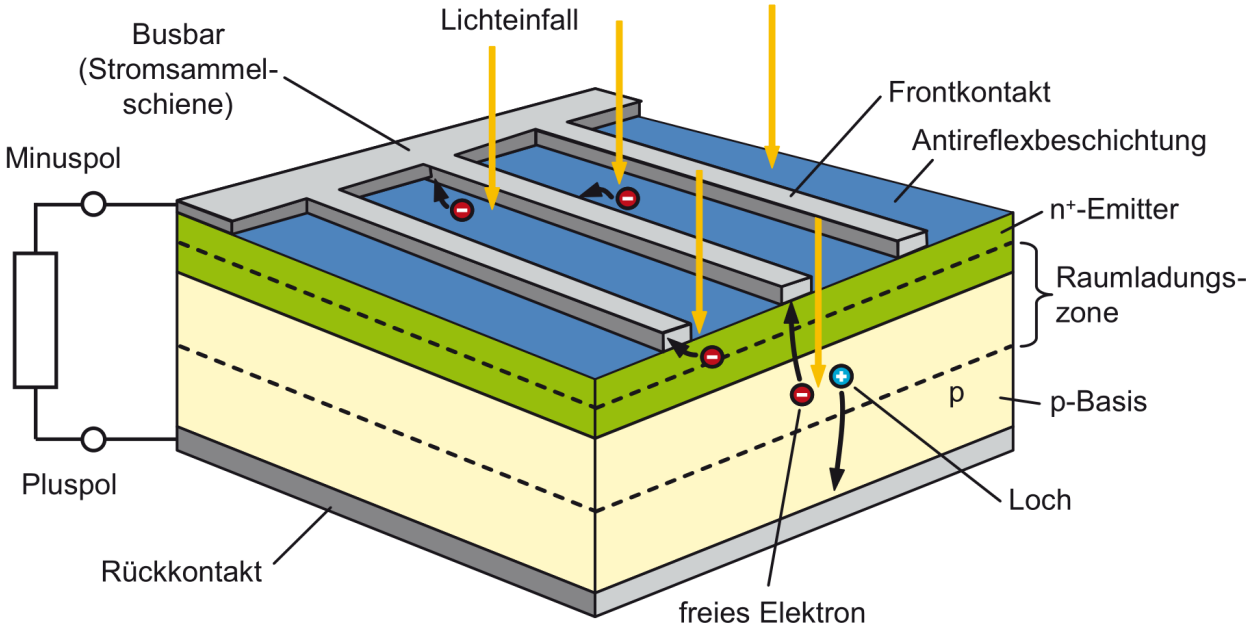


Abbildung 2.4: Aufbau und Funktion einer Solarzelle nach [1]

Für die Stromerzeugung der Solarzelle ist es nun wichtig, an welcher Stelle im Kristall das Photon absorbiert wurde und ein Elektron ausbildet. Geschieht die Absorption genau in der Raumladungszone wird das Elektron durch das elektrische Feld nach oben in die n-dotierte Zone beschleunigt und steht als freier Ladungsträger der Zelle zur Verfügung. Dies stellt den gewünschten Ablauf dar, da das Elektron und das Loch voneinander durch die Raumladungszone getrennt werden und nicht wieder rekombinieren können.

Wird das Photon in der unteren p-dotierten Zone absorbiert, kann es sein, dass das Elektron bis zur Grenzschicht diffundiert und damit auch einen Ladungsträger bereitstellen kann. Andernfalls rekombiniert das ausgebildete Elektronen-Loch-Paar wieder. Gleiches gilt auch für eine Absorption in dem oberen n-dotierten Teil, nur dass es hier extrem unwahrscheinlich ist, dass das Elektronen-Loch-Paar später einen freien Ladungsträger ausbildet [6].

Dies zeigt, dass nicht alle Photonen, die in die Solarzelle eindringen auch freie Ladungsträger ausbilden können. Zum einen weil nicht alle Photonen genug Energie besitzen, um Elektronen aus dem Halbleiter zu lösen. Zum anderen, da Solarzellen nicht perfekt gefertigt werden können und damit Möglichkeiten zur Rekombination der Elektronen bleiben. Diese Verluste bzw. dieser Wirkungsgrad wird durch den externen Quantenwirkungsgrad beschrieben, der auch externe Quanteneffizienz (EQE) heißt. Die EQE gibt für eine Solarzelle an, wie hoch die Wahrscheinlichkeit ist, dass eingedrungenes Licht einer bestimmten Wellenlänge freie Ladungsträger ausbilden kann. Mithilfe einer EQE kann entsprechend analysiert werden, welche Auswirkungen das Ändern des Spektrums auf das Ausbilden von Ladungsträgern

hat. Ein Graph einer EQE folgt später in Abbildung 3.9.

Sofern das Spektrum unverändert bleibt, ist die Bestrahlungsstärke E proportional zur Anzahl der erzeugten Ladungsträger, dem Photostrom I_{Ph}. Es gilt also:

$$I_{Ph} \sim E \tag{2.29}$$

Wenn eine Solarzelle kurzgeschlossen wird, fließt damit der größtmögliche Strom, der Kurzschlussstrom I_{sc}. Im deutschen Sprachgebrauch wird dieser meist mit I_K angegeben. Im Rahmen dieser Arbeit wird überwiegend die englische Bezeichnung mit I_{sc} verwendet werden, da die genutzten Herstellerangaben ebenfalls auf Englisch vorliegen. Der Kurzschlussstrom ist gleich dem Photostrom der Solarzelle.

Die größtmögliche Spannung der Solarzelle heißt Leerlaufspannung U_{oc} und sie liegt an, wenn die Kontakte der Zelle offen gelassen werden. Im Deutschen ist die Bezeichnung U_L. Die Leerlaufspannung ändert sich lediglich mit dem natürlichen Logarithmus der Bestrahlungsstärke E. Es gilt also:

$$U_{oc} \sim ln(E) \tag{2.30}$$

Eine typische Kennlinie einer Solarzelle hat den Verlauf wie er in Abbildung 2.5 gezeigt ist.

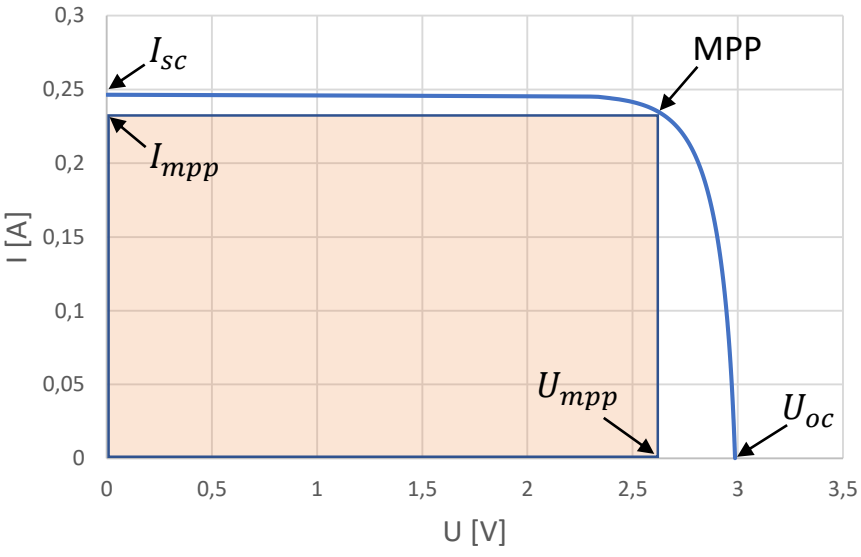


Abbildung 2.5: Kennlinie und Kennwerte einer Solarzelle

Da die elektrische Leistung P das Produkt aus Strom und Spannung ist, kann dies in der Abbildung durch eine Fläche dargestellt werden. Daher ist die Leistung der Solarzelle an dem Punkt maximal, an dem der Arbeitspunkt die größte Fläche ausbildet. Der Punkt maximaler

Leistung wird auch als Maximum Power Point (MPP) bezeichnet und wird durch die Werte U_{mpp} und I_{mpp} angegeben.

Neben der Bestrahlungsstärke ist die Temperatur eine weitere wichtige Größe, die das elektrische Verhalten der Solarzelle beeinflusst. Kurzgefasst bewirkt eine Erhöhung der Temperatur das Absenken der Leerlaufspannung und im geringen Maße das Ansteigen des Kurzschlussstroms. Die Änderung des Kurzschlussstroms liegt allerdings ca. eine Größenordnung unter der der Leerlaufspannung, weswegen letztlich mit steigender Temperatur der Wirkungsgrad der Solarzelle sinkt. Deshalb wird ein möglichst kühler Betrieb von Solarzellen angestrebt [14] [6].

Mit dem Ziel möglichst hohe Wirkungsgrade für eine Solarzelle zu erreichen, wurden in der Vergangenheit sogenannte Multi-Junction-Solarzellen entwickelt. Diese Zellen bestehen im Aufbau aus mehreren Solarzellen, die übereinander gelegt werden und jede der Subzellen ist aus einem anderen Halbleitermaterial gefertigt. Durch die verschiedenen Bandlücken in den Halbleitern sind die Subzellen auf die Absorption verschiedener Bereiches des Spektrums optimiert und erlangen so einen insgesamt höheren Wirkungsgrad als gewöhnliche Single-Junction-Solarzellen. Für eine Separierung der einzelnen Subzellen sorgen Tunnel Junctions, die gleichzeitig für das Licht durchlässig sind. Unter den Multi-Junction-Solarzellen sind die Triple-Junction-Solarzellen die häufigsten Vertreter. Neben diesen sind noch Zellen mit zwei oder vier Subzellen geläufig. Darüber hinaus sind die Multi-Junction-Solarzellen nur im Laboreinsatz [6].

Weitere Informationen zu Tandem- und Multi-Junction-Solarzellen lassen sich in dem Buch von Shah [15] finden. Umfassendere Literatur zur mathematischen Beschreibung von Solarzellen steht mit Stadler [16] und Lewerenz [17] zur Verfügung.

2.3.2 Solarmodul

Solarzellen ermöglichen es, elektrische Leistung bereitzustellen, doch liegt diese bestenfalls in der Größenordnung von 1 W. Damit können nur die wenigsten Verbraucher betrieben werden. Deswegen werden einzelne Zellen meist zu Solarmodulen zusammengeschaltet. Dies wird in erster Linie über Reihenschaltungen realisiert, damit die Spannungen der einzelnen Zellen sich addieren und so die Nennspannung des Verbrauchers erreicht wird. Ein weiterer Vorteil dabei ist, dass der gemeinsame Strom dabei klein gehalten wird und so die Verluste im Leiter gering bleiben. Eine Reihenschaltung von mehreren Solarzellen wird meistens als String bezeichnet. Soll die Leistung eines Solarmoduls weiter gesteigert werden und ist

die Nennspannung bereits erreicht, bleibt die Möglichkeit mehrere Strings parallel zu schalten. Dadurch addieren sich die Ströme der Strings, sofern alle Zellen von der Sonne gleich bestrahlt werden.

Mögliche Probleme beim Verschalten von Solarzellen entstehen erst, sobald die einzelnen Zellen unterschiedlich bestrahlt werden. Dies muss nicht nur dadurch entstehen, dass die Zellen unterschiedlich ausgerichtet sind, sondern hat am Boden meist den Grund das Teile des Solarmoduls verschattet werden. Bei stationären Photovoltaikanlagen, wie sie zunehmend auf Häuserdächern installiert werden, kann dies durch feste Hindernisse wie Bäume verursacht werden, aber auch durch saisonale Ereignisse wie Schnee.

Die Auswirkung der Verschattung nur einer Zelle innerhalb eines Strings bewirkt, dass der gesamte String gehemmt wird und im schlechtesten Fall keine Leistung bereitstellen kann. Dies hängt von der Bestrahlung, dem Zelltyp, der Zellanzahl und dem aktuellen Arbeitspunkt ab. Da die Bestrahlungsstärke proportional zum Kurzschlussstrom ist, wird dieser zu null solange auch die Bestrahlung gleich null ist. Durch die Reihenschaltung kann nur der geringste Strom durch den String fließen, was in diesem Fall bedeutet, dass kein Strom fließen kann. Sobald sich aber über einer verschatteten Zelle eine negative Spannung aufgebaut hat, die groß genug ist, agiert sie nicht mehr als eine Quelle, sondern als Verbraucher. Dadurch entstehen Hotspots, die zu der thermischen Zerstörung der verschatteten Zelle führen können.

Als Lösung dazu lassen sich Bypassdioden integrieren, die parallel zu einer oder n Solarzellen geschaltet werden. Die Dioden ermöglichen einen „Ausweichpfad“ für den Strom im String, falls der Widerstand einer verschatteten Solarzelle zu hoch wird. Sie bewirken damit eine Steigerung des Gesamtwirkungsgrads im Betrieb und schützen die Zellen vor Zerstörung. Innerhalb von flugzeugintegrierten Solarzellen werden Bypassdioden deshalb meist schon während der Produktion mit in die einzelnen Solarzellen integriert. Nachteile entstehen durch Bypassdioden während des Betriebs nicht.

Liegt eine Parallelschaltung von mehreren Strings vor, von denen ein String teil- oder vollverschattet ist, können Ausgleichsströme durch den verschatteten Teil fließen. Da die Spannung der unverschatteten Strings größer als die des Verschatteten ist, bildet sich über diesem ein negativer Spannungsabfall. Wie im vorherigen Fall sorgt dies für eine thermische Belastung der verschatteten Solarzellen, die zu Schäden führen kann. Abhilfe schaffen in Reihe geschaltete Dioden zu den einzelnen Strings, die Stringdioden oder im englischen Blockingdioden genannt werden. Sie verhindern das Ausgleichsströme fließen können. Allerdings besitzen sie den Nachteil, dass im Betrieb die Spannung des Strings über der Blockingdiode anliegt und damit an der Diode für gewöhnlich 0,7 V abfallen [6] [18] [14].

In der noch folgenden Abbildung 3.19 ist der Aufbau eines flugzeugintegrierten Solargenerators gezeigt, in dem auch Bypass- und Blockingdioden integriert sind.

2.3.3 Stromrichter

Bisher wurde der Aufbau und die Funktion eines Solargenerators umfassend beschrieben, doch wie sieht die abnehmende Seite für die elektrische Leistung aus? Im einfachsten Fall wird ein ohmscher Widerstand am Solargenerator angeschlossen. Nach einer Arbeitspunktbestimmung wird sichtbar, wie viel Leistung der Solargenerator abgibt. Im Normalfall wird dieser Punkt nicht gleich dem MPP sein und es wird damit nicht die maximale Leistung aus dem Solargenerator entnommen.

Die Funktion des Solargenerators liegt darin, elektrische Leistung bereitzustellen, die möglichst maximal sein sollte. Dazu müsste ein variabler Widerstand angeschlossen werden, der die Funktion besitzt sich selbst, auch unter wechselnden Bedingungen, dem MPP nachzuführen. Dieses Verhalten wird als MPP-Tracking (MPPT) bezeichnet. Ein Verbraucher, der diese regelnde Funktion besitzt, müsste dabei mit einer ständig ändernden Eingangsspannung arbeiten können. Da normale Verbraucher ein relativ festes Spannungsniveau benötigen und auch keine MPPT Funktion besitzen, werden Solargeneratoren nicht direkt mit den Verbrauchern verbunden. Zwischen beiden wird eine flexible elektronische Anpassungsschaltung eingesetzt, die die Eingangsspannung des Solargenerators von der Nennspannung des Verbrauchers entkoppelt und auf der Eingangsseite ein MPP-Tracking betreibt.

Die Funktion der Spannungsumwandlung erfüllt ein Stromrichter, der die Gleichspannung der Solarzellen, je nach Anwendungsgebiet, in eine andere Gleichspannung (Gleichspannungswandler) oder eine Wechselspannung (Wechselrichter) umwandelt. Im Fall eines Solarflugzeugs wird nur eine Gleichspannung benötigt, mit der die Batterien geladen werden oder das Bordnetz versorgt wird. Die Elektromotoren besitzen bei Bedarf ihre eigenen Wechselrichter. Der Gleichspannungswandler oder auch DC/DC-Wandler, besitzt damit im Anwendungsfall einer autarken Solaranlage gleichzeitig die Funktion eines MPP-Ladereglers [6, S. 234]

Das Prinzip eines Maximum Power Point Trackers ist in Abbildung 2.6 gezeigt. Der verwendete DC/DC-Wandler, zur Entkopplung der Spannungen, ist meistens als Tiefsetzsteller (Buck Converter) ausgeführt. Demnach ist die Spannung des Solargenerators höher als die benötigte Spannung des Verbrauchers. Der Tiefsetzsteller besitzt einen Schalter, der hochfrequent geschaltet wird und damit die höhere Eingangsspannung zerhackt. Mithilfe von geeigneten

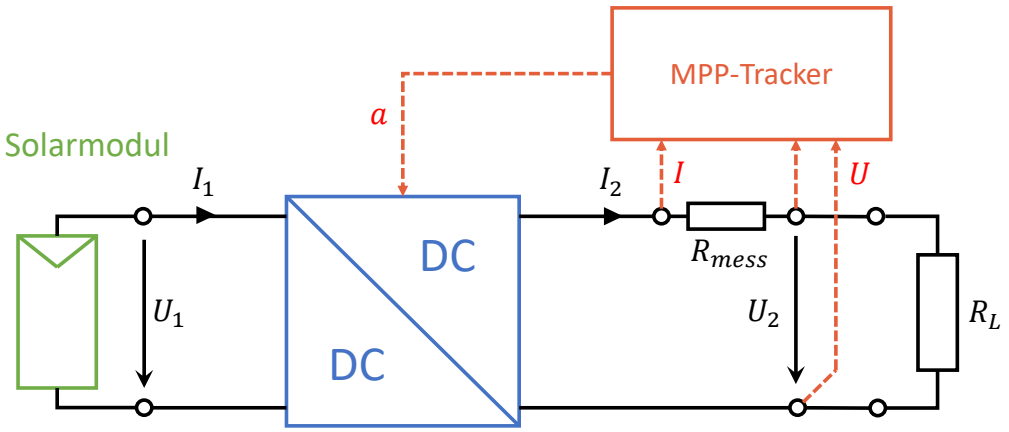


Abbildung 2.6: Prinzip eines MPPTs zur maximalen Leistungsabnahme am Solargenerator

Filterbausteinen entsteht so eine niedrigere Ausgangsspannung, die über die Tastzeit a eingestellt werden kann. Der MPPT misst auf der Ausgangsseite des Gleichspannungswandlers die Spannung, den Strom und damit die Leistung. Durch Variation der Tastzeit a ändert sich die Last für den Solargenerator und der MPPT kann so den Solargenerator im MPP betreiben und gleichzeitig eine konstante Ausgangsspannung bereitstellen. Leistungsänderungen auf der Eingangsseite äußern sich auf der Ausgangsseite darin, dass der abgegebene Strom variiert.

Obwohl genau genommen nur der Regler der MPPT ist wird meistens der Gleichspannungswandler mit hinzugezählt, da Beide eine Einheit bilden. Moderne MPPTs besitzen Wirkungsgrade von 98-99 % bei der Entnahme der Leistung am Solargenerator bis zur Abgabe an den Verbraucher [14] [6] [19] [8].

2.4 Simulation Engineering

Das Ziel dieser Arbeit ist, das Verhalten des Solargenerators von HAP in einer Echtzeitsimulation wiederzugeben bzw. vorherzusagen. Dabei stellt der Solargenerator ein *Real World System* dar oder kurz ein System. Ein System besteht aus einer Menge von Komponenten, die miteinander interagieren und ist durch eine Systemgrenze abgeschlossen. Die Systemgrenze markiert, welche Komponenten zum System gehören und schafft definierte Schnittstellen zur Umwelt. Die einzelnen Komponenten können wiederum selbst Systeme sein [20].

Mit dem Ziel, das System Solargenerator zu simulieren, ist zunächst der Prozess der Modellbildung notwendig. Dabei wird durch Abstraktion eine Beschreibung des Systems geschaffen, das Modell. Dieses erlaubt im Kontext der digitalen Datenverarbeitung eine Simulation des

Systems für einen konkreten Anwendungsfall. Nachdem ein berechenbares Modell erstellt wurde, kann dies, bei Einhaltung von Kriterien, auch als eine Echtzeitsimulation ausgeführt werden und das reale System ggf. ersetzen. Wenn dies mit der Absicht für einen Teststand geschieht und mit der Echtzeitsimulation physische Komponenten vervollständigt und getestet werden sollen, dann handelt es sich um einen Hardware in the Loop (HiL) Teststand. Die Entwicklung des Solargeneratormodells soll in diesem Anwendungsfall münden. Die einzelnen Themengebiete, die dazu genannt wurden, werden in den folgenden Abschnitten beschrieben.

2.4.1 Modellbildung

Die Modellbildung ist der Prozess der Erstellung eines Modells von einem System. Allgemein gesprochen ist ein Modell die physikalische, mathematische oder logische Repräsentation eines Systems, in einer abstrakten oder reduzierten Form [21, S. 648] [22, S. 4]. Im Kontext dieser Arbeit bezeichnet ein Modell ein berechenbares Objekt, das das Verhalten eines realen Systems wiedergibt. Dabei bildet ein Modell nur einen Teil des realen Systems ab. Am Beispiel einer Solarzelle könnte dies das elektrische Verhalten sein. Die thermischen Eigenschaften sind in diesem Fall nicht von Interesse und werden deshalb vernachlässigt. Daher muss zu Beginn festgelegt werden, welcher Teil des *Real World Systems* modelliert werden soll. Das relevante Wissen dazu zu finden und zusammenzutragen, sowie deren Abbildung, ist die Hauptingenieursarbeit [20].

Für den Prozess der Modellbildung ist zunächst ein Verständnis des Systems notwendig, das wiederum aus einer Analyse des Systems gewonnen wird. Als Analysemethode wird von Mobus und Kalton [21, S. 595] die Dekomposition des Systems vorgeschlagen. Dabei wird geschaut, aus welchen Komponenten ein System aufgebaut ist, wie diese funktionieren und welche Schnittstellen die Komponenten zueinander besitzen. Damit wird das System als White Box betrachtet. Ist die Funktion einer Komponente zu komplex und wird diese selbst als Subsystem angesehen, dann wird dieses Vorgehen für die einzelnen Subsysteme rekursiv wiederholt, um auch deren Funktion genau genug zu verstehen. Das Ziel der Analyse soll nicht der reine Wissensaufbau sein, sondern ein weitreichendes Verständnis zu schaffen, mit dem selbst Hypothesen aufgestellt werden können.

Am Ende der Dekomposition eines Systems gibt es nach Mobus und Kalton [21, S. 622] drei Produkte, die als Ausgabe gebildet werden und die für die nächste Phase der Modellbildung benötigt werden. In erster Linie ist dies die Wissensbasis über das System. Mit ihr sei jedes Objekt innerhalb des Systems erfasst und jeder Zusammenhang. Aus dem verknüpften

Wissen lassen sich die anderen beiden Produkte erzeugen, die systematische Sichten auf das System darstellen. Dies sind eine Baumstruktur des hierarchischen Aufbaus und ein Plan der Flüsse und Prozesse.

Für die Modellbildung selbst müssen nun Übertragungsfunktionen gefunden werden, die die Ausgaben und damit das Verhalten des System, in Abhängigkeit der Eingangsgrößen beschreiben. Dies kann auf jeder Hierarchieebene des Systems geschehen und gibt damit den Detailgrad des erstellen Modells an. So kann ein angenähertes Modell erstellt werden, indem nur für das Gesamtsystem versucht wird, eine Übertragungsfunktion zu finden, aber dann werden nur angenäherte Ergebnisse möglich sein. Je komplexer ein System ist, desto schwieriger wird es sein, eine solche Funktion zu finden. Für genauere Modelle ist es daher notwendig, auch die Subsysteme zu modellieren. Zur Umsetzung der Übertragungsfunktionen werden die Wissensbasis und die Übersicht der Flüsse und Schnittstellen genutzt. Die Beschreibung selbst kann über mathematische, physikalische oder logische Methoden erfolgen und liegt im Ermessen des Modellierers.

Für die Validierung eines Modells ist es üblich, reale Messwerte des Systems mit der Simulation zu vergleichen. Dabei gilt es Qualitätskriterien für die Vergleiche festzulegen und die Testbedingungen zu hinterfragen: Wurden relevante Eingangsgrößen getestet? Passen die Umgebungsbedingungen unter denen das Modell zum Einsatz kommt? Sind die Eingangswerte in ihrer Größenordnung und Wertebereich realistisch? Die korrekte Validierung zu bewerten und einzuschätzen, kann nur ein Experte des Fachbereichs [20].

2.4.2 Echtzeitsysteme

Echtzeitsysteme sind Rechensysteme, die aufgrund ihres Umfelds und ihres Aufgabenbereichs Echtzeitanforderungen besitzen. Diese zeitlichen Anforderungen ergeben sich aus dem Kontext des Rechensystems und der Abhängigkeiten zu den äußeren Systemen, in dem die Abläufe der Abhängigkeiten zeitliche Limits besitzen. Echtzeit selbst bedeutet, dass das Ergebnis der Berechnung die das Rechensystem zu erledigen hat innerhalb einer festgelegten Zeit garantiert vorliegt.

Abbildung 2.7 zeigt das Grundmodell eines Echtzeitsystems, wie es häufig in der Literatur angegeben wird [23] [22]. In diesem erhält das Rechensystem Eingangsdaten durch Sensoren und muss basierend auf diesem Input die Handlung für den nächsten Schritt berechnen. Das Rechenergebnis kann durchaus die Form eines Stellkommandos annehmen, mit dem ein Aktor (auch Aktuator) kommandiert wird. Durch diese Handlung soll ein technisches

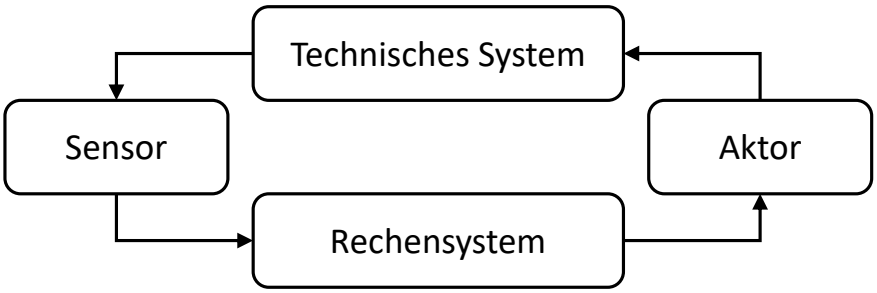


Abbildung 2.7: Grundmodell eines Echtzeitsystems

System in einem gewünschtem Zustand gehalten werden. Dem Echtzeitsystem liegt damit eine periodische Aufgabe zugrunde, die für gewöhnlich mit einer festen Frequenz ausgeführt wird.

Die Anwendungen der Echtzeitsysteme werden in der Literatur häufig mit Regelungsaufgaben im Kontext der Prozesstechnik und Automatisierung angegeben, wie bei Wörn [24]. Dies muss nicht der Fall sein. Die Rolle des Sensors und Aktors kann ebenso durch das Ein- und Auslesen einer digitalen Kommunikationsschnittstelle gefüllt werden. Damit erfüllt der Echtzeitrechner Kommunikationsaufgaben, wie dies auf einem HiL Teststand der Fall sein kann, indem noch nicht entwickelte Komponenten eines Systems simuliert werden. Genaueres folgt im nächsten Abschnitt.

Die unbedingte Einhaltung der zeitlichen Limits ergibt sich aus den möglichen Konsequenzen bei Verletzung dieser. Als anschauliches Beispiel sei hier ein Steuergerät für das ABS oder ESP im Auto genannt. Sollte die Berechnung für die Stellkommandos zu lange dauern, kann dies zu einem Ausbruch des Autos aus seiner Spur führen und damit zu einer Gefahr für Leib und Leben. Nicht immer müssen die Folgen einer Nichteinhaltung so dramatisch sein, aber der Kern dabei ist, dass es zu einem nicht vorhersehbaren Verhalten kommen kann. Wenn das Echtzeitsystem mit mechanisch bewegten Objekten operiert, kann dies zu Beschädigungen dieser führen, die es zu vermeiden gilt.

2.4.3 Hardware in the Loop Teststand

Mithilfe einer Hardware in the Loop (HiL) Simulation können Hardware oder Software auf Subsystemebene getestet werden, noch bevor ein testbarer Prototyp zur Verfügung steht. HiL Simulationen werden mit einem Echtzeitrechner realisiert, der I/O-Funktionen übernimmt und verschiedenste Software bzw. Simulationsmodelle ausführen kann. Angeschlossen ist dieser an einer Hardware des zu testenden Systems, die so in eine Umgebung versetzt

wird, dass sie denkt, es entspreche den späteren Einsatzumgebungen. Dabei können auf dem zu testenden System verschiedene Software- oder Firmwarestände ausprobiert werden, als auch auf dem Echtzeitrechner, der die Umgebung simuliert [22].

Ein HiL Teststand bezeichnet eine einzelne HiL Simulationsumgebung oder kann auch eine Mehrzahl von angeschlossenen Systemen beschreiben. Sind auf dem Teststand zwei oder mehr Systeme involviert, die später miteinander arbeiten sollen, so können sie direkt aneinander angeschlossen werden. Dann muss der Echtzeitrechner nicht mehr alle Schnittstellen für eine Komponente simulieren, sondern sie können zunehmend durch die echten ersetzt werden. Eine schrittweise Integration der Subsysteme schafft eine kontrollierte Testumgebung. Ein HiL Teststand eignet sich besonders für die Entwicklung komplexer, eingebetteter Systeme wie sie beispielsweise in der Luft- und Raumfahrt vorkommen.

Das Ziel eines HiL Teststands ist es, die zu testenden Systeme möglichst realistischen Bedingungen auszusetzen. Mit dem zunächst einzelnen Testen und dem späteren Zusammenschluss der Systeme zum Verbund, werden gleichzeitig die verschiedenen Testphasen im Entwicklungsprozess durchlaufen, bis letztendlich möglichst das Gesamtsystem getestet werden kann. Dies ist vor allem in der Bewältigung der Komplexität des Gesamtsystems von Vorteil. Mithilfe der kontrollierten Umgebung ist es nicht nur möglich, das Gesamt- oder Teilsystem innerhalb der zu erwartenden Umgebungen zu testen, sondern auch bis an die Grenzbereiche oder darüber hinaus. Sofern es sich nicht um mechanische Teile handelt, die beansprucht werden, schafft dies eine sichere Umgebung ohne zu erwartende Schäden. Innerhalb der Testumgebung können beliebige und auch seltene Szenarien in wiederholbarer Form getestet werden. Aufgrund der Digitalisierung erlaubt dies meist auch eine Automatisierung der Testzyklen.

Für den Aufbau einer HiL Simulation ist es die direkteste Vorgehensweise mit einer nicht echtzeitfähigen Simulation zu starten und diese nach Fertigstellung echtzeitfähig zu machen [22, S. 108]. Für die Überführung müssen Integratoren an eine feste Schrittweite angepasst und deren Genauigkeit betrachtet werden. Es müssen die I/O-Funktionen des Modells an die Hardware angepasst werden und es sollten Speicherzugriffe auf persistente Speicher nach Möglichkeit vermieden werden. Bei Bedarf und Nichteinhaltung der zeitlichen Vorgaben müssen Optimierungen oder Vereinfachungen am Modell vorgenommen werden. Eine Alternative dazu stellt die Möglichkeit der Verwendung einer leistungsfähigeren Recheneinheit dar.

3 Entwicklung des Solargeneratormodells

Der erste große Teilaufgabe dieser Arbeit ist es, einen flugzeugintegrierten Solargenerator mit möglichst hoher Güte zu modellieren. Die Umsetzung geschieht am Beispiel des HAP Prototyps. Dazu werden zunächst Anforderungen an das Modell aufgestellt und eine Softwarearchitektur abgeleitet. Noch vor der Implementierung des Solargeneratormodells, werden zunächst Modelle für die Sonne implementiert, die den Verlauf und das Spektrum der Sonne abbilden können. Damit erhält der Solargenerator später Eingangsgrößen mit sinnvollen Werten. Für den Aufbau des Solargeneratormodells wird zunächst ein Modell für eine einzelne Solarzelle erstellt, welches auf physikalischen Zusammenhängen und realen Messwerten aufbaut. Mithilfe einer selbst entwickelten Leistungsflussbibliothek können im nächsten Schritt die einzelnen Solarzellen verschaltet werden und damit die Verschaltungseffekte berücksichtigt werden. Vervollständigt wird das Solargeneratormodell anschließend durch die Berechnung der Einfallswinkel des Sonnenlichts für die einzelnen Solarzellen. Ergänzt wird das Solargeneratormodell abschließend durch ein MPPT-Modell, das die aufnehmende Seite der Leistung darstellt.

3.1 Vorgehen zum Modellaufbau

Als Vorgehensmodell zur Gesamtmodellentwicklung wurde sich am V-Modell orientiert. Zuerst soll damit begonnen, Anforderungen an das zu entwickelnde Gesamtsystem zu definieren, was im Kern das Solargeneratormodell ist. Danach wird das Gesamtsystem mit seinen Subsystemen betrachtet und deren Verbindungen, woraus sich eine grobe Architektur des Gesamtsystems ergibt. Mit einer nachfolgenden Subsystembetrachtung ergibt sich eine Feinarchitektur. Nach der Implementierung der Subsysteme kann aus diesen das Gesamtsystem aufgebaut werden. Dabei muss jedes System und Subsystem auf korrekte Funktion getestet werden, damit sich für das Gesamtsystem das richtige Verhalten ergibt.

Die Abbildung 3.1 zeigt nun eine Einordnung dieser Arbeit in den Kontext von HAP und bezugnehmend auf die beschriebenen Ausgangsbedingungen und Ziele ergibt sich die Vorgehensweise zur Erstellung des Solargeneratormodells.

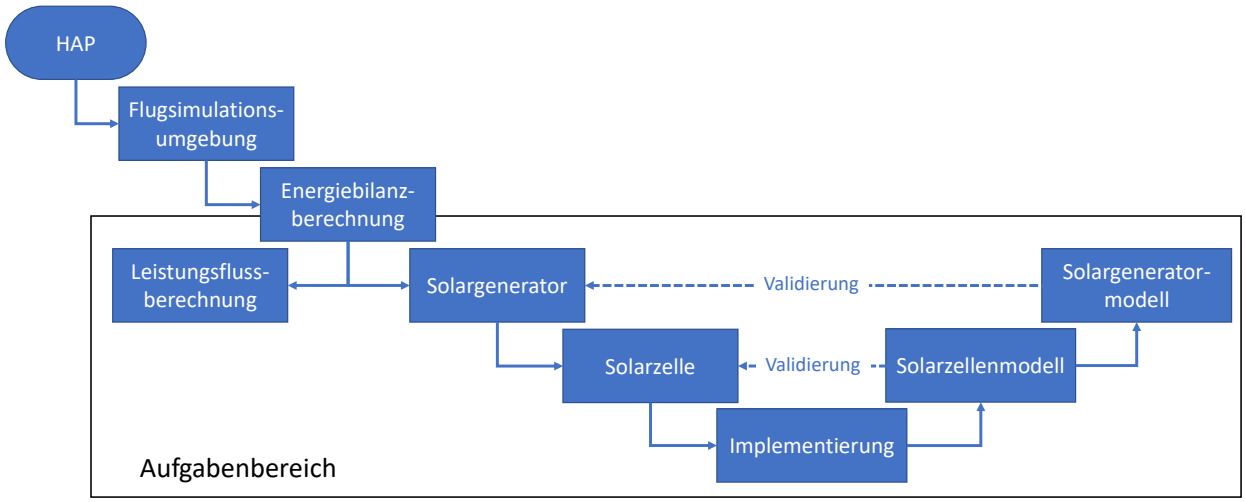


Abbildung 3.1: Einordnung und Umfang des Modellaufbaus

Mit dem Top-Down Ansatz lässt sich zunächst die Energiebilanzrechnung von HAP in zwei zu erfüllende Teile unterteilen. Diese benötigt den energieliefernden Solargenerator und eine Leistungsflussverschaltung, die alle Energieflüsse abbildet. Zu der Energiebilanzrechnung gehören u. a. auch die Batterien des Systems, allerdings liegen diese außerhalb des Aufgabenbereichs dieser Arbeit. Die Unterteilung des Solargenerators ergibt eine Reihe miteinander verschalteter Solarzellen, die dessen Subsysteme sind. Die einzelne Solarzelle ist als kleinste Einheit des Modells anzusehen und wird nicht weiter unterteilt.

Es folgt die Modellierung einer einzelnen Solarzelle. Zur Qualitätssicherung muss das Solarzellenmodell nach der Implementierung auf geeignete Weise validiert werden z. B. durch Vergleiche mit Messwerten einer realen Solarzelle. Ausgehend von validierten Modell einer Solarzelle wird der Bottom-Up Ansatz verwendet, um aus den Solarzellenmodell schrittweise die nächstgrößeren Strukturen des Solargenerators abzubilden. Sobald der gesamte Solargenerator mitsamt korrekter interner Verschaltung modelliert wurde, müsste auch hier eine Validierung folgen, um die Funktion des Modells zu bestätigen. Eine Validierung des Solargeneratormodells ist zu diesem Entwicklungszeitpunkt des Projekts nicht möglich, da noch kein realer Solargenerator vorhanden ist, mit dem Kontrollmessungen durchgeführt werden könnten.

Sofern bei der Verschaltung der validierten Solarzellen keine Fehler gemacht wurden und auch kein systematischer Fehler vorliegt, kann davon ausgegangen werden, dass das Gesamtmodell

nur eine Ungenauigkeit aufweist, die sich aus der Summe der Teilungenauigkeiten ergibt. Die Gesamtungenauigkeit müsste im Einzelnen durch eine detaillierte Rechnung mit dem Fehlerfortpflanzungsgesetz bestimmt werden, soll aber nicht mehr Teil dieser Arbeit sein.

3.1.1 Aufstellen von Anforderungen

Vor Beginn jeder Implementierung sollten Anforderung an ein zu entwickelndes System gestellt werden, gegen die zum Ende des Entwicklungszyklus getestet werden kann. Für das weitere Vorgehen wurden dazu folgende Anforderungen an das Solargeneratormodell gestellt:

Das Solargeneratormodell ...

1. muss den Solargenerator von HAP vollständig abbilden.
2. muss Schnittstellen für den Austausch mit anderen Modulen der Simulationsumgebung besitzen.
3. muss den Leistungsoutput in geeigneter Weise ausgeben können.
4. muss auf einem Echtzeitrechner von dSPACE lauffähig sein.
5. soll auf dem Echtzeitrechner mit einer Frequenz von 100 Hz simuliert werden können.

Dabei ist auf die Schlagwörter „muss“ und „soll“ zu achten. Das Wort „muss“ steht dabei für eine unbedingt notwendige Erfüllung dieser Anforderung, wohingegen das Wort „soll“ nur eine optionale Erfüllung vorsieht.

Die erste Anforderung beschreibt dabei die Hauptmotivation und den Kern dieser Arbeit. Die Vollständigkeit bezieht sich dabei auf physische und elektrische Vollständigkeit des Solargenerators. Das Modell soll dabei gleichzeitig eine korrekte Abbildung der Wirkung auf äußere Einflüsse ermöglichen. Die zweite Anforderung leitet sich daraus ab, dass das Modell später zusammen mit weiteren Simulationsmodellen als eine Gesamtsimulation laufen muss und dabei auch ein Datenaustausch zwischen diesen Modellen stattfinden muss.

Anforderung Nummer drei definiert die erzeugte Leistung als Ausgangsgröße des Solargeneratormodells und lässt dabei den Lösungsraum offen, indem die Schnittstelle zu nachfolgenden Modulen nicht genauer eingegrenzt wird. Die beiden letzten Anforderungen nehmen Bezug auf den Echtzeitrechner, der die spätere ausführende Hardware darstellt. Dabei wird zuerst festgelegt, dass eine Kompatibilität mit einem System von dSPACE gewährleistet werden

muss. Dies ist als eine Designentscheidung begründet, da es bereits ein vorhandenes System gibt. Die Simulationsfrequenz von 100 Hz ist dadurch begründet, dass das Flugmechanikmodell aktuell mit 100 Hz simuliert wird und als schnellstes Modul gilt. Alle anderen Module an diese Frequenz anzupassen hat aus Sicht der Softwareentwicklung Vorteile, ist aber nicht notwendig. Eine spätere Reduktion der Frequenz ist jederzeit möglich.

3.1.2 Abgeleitete Softwarearchitektur

Für das weitere Vorgehen wurde ein Übersichtsdiagramm erstellt. Dieses zeigt die logische Architektur der Softwaremodule und die Schnittstellen, die sie besitzen. Abbildung 3.2 zeigt einen Ausschnitt der Softwarearchitektur der Gesamtsimulation und die Einordnung des Solargenerators darin. Im Anhang ist die detailliertere Abbildung A.1 zu finden.

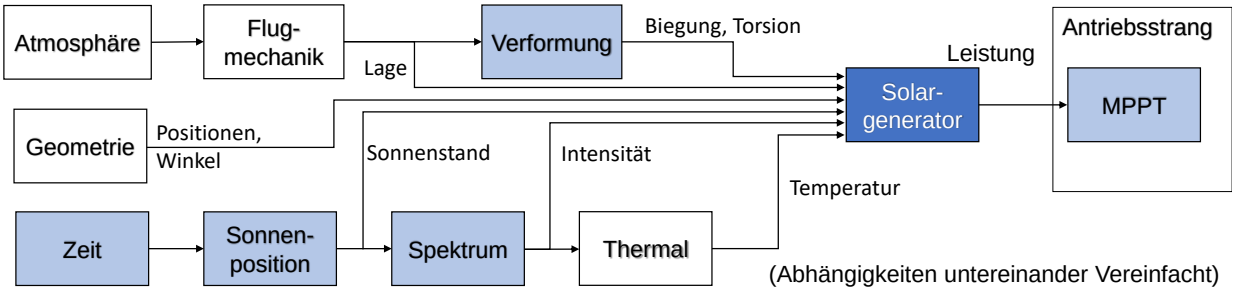


Abbildung 3.2: Ausschnitt der Softwarearchitektur des Simulationssystems mit Fokus auf dem Solargenerator – erstellte Modelle in blau

Die im vorherigen Abschnitt definierte Anforderung 2 fordert, dass der Solargenerator über Schnittstellen für die Kommunikation mit anderen Modulen besitzen muss. Die hier in Abbildung 3.2 gezeigte Struktur lässt die Schnittstellen hieraus ableiten. Dabei wäre zunächst die Leistung als Ausgangsgröße zu nennen, wie es ebenfalls von der Anforderung 3 festgelegt wurde. Die Leistung wird an den Antriebsstrang weitergegeben, wobei hier der MPPT das aufnehmende Modul sein wird.

Auf der Eingangsseite des Solargenerators sind Verbindungen zur Sonnenposition und zum Spektrum zu sehen. Diese sollen dem Solargenerator die aktuelle Position und die spektrale Bestrahlungsstärke der Sonne mitteilen. Das Thermalmodell berechnet für den Solargenerator seine mittlere Temperatur und leitet sie an ihn weiter.

Die verbleibenden Abhängigkeiten haben alle einen Einfluss auf den Einfallswinkel der Sonne auf die einzelnen Solarzellen. Da wäre zunächst die durch die Struktur vorgegebene Platzierung und Orientierung der Solarzellen. Dies sind statische Werte, die vor Simulationsbeginn

festgelegt werden und im einfachsten Fall durch globale Variablen zur Verfügung gestellt werden. Wesentliche Größen, die von dem Flugmechanikmodell gebraucht werden, sind die Lagewinkel des Flugzeugs. Als Besonderheit dieser Energieertragsberechnung sollen auch die Verformungen des Flugzeugs mit berücksichtigt werden. Diese äußern sich durch Biege- und Torsionswinkel, die an mehreren Stellen auf dem Flügel berechnet werden müssen.

Da das solare Spektrum oder die Verformung von keinem anderen Modell als dem Solargenerator in dieser Simulationsumgebung gebraucht werden, ist eine Implementierung durch einen anderen Mitarbeiter nicht innerhalb der Bearbeitungszeit zu erwarten. Dies hat zur Folge, dass die in Abbildung 3.2 hellblau gefärbten Module zusätzlich zum Solargenerator im Rahmen dieser Arbeit entstanden sind, damit entsprechende Eingangsdaten bereitgestellt werden können.

In den folgenden Abschnitten wird auf die Implementierung der erstellten Modelle eingegangen und es werden die durchgeführten Tests und Ergebnisse dargelegt.

3.2 Sonnenpositionsmodell

Für einen simulierten Flug mit einem Solarflugzeug ist es notwendig, dass die Sonne über den Tagesverlauf auch einen korrekten Tagesbogen macht. Funktion des Sonnenpositionsmodells soll es sein, zu einem gegebenen Ort und einer Zeit den Positionsstand der Sonne zu berechnen. Wie bereits in Abschnitt 2.2.1 gezeigt gibt es eine relativ geringe Anzahl an Formeln, mit deren Hilfe sich der Verlauf annähernd beschreiben lässt. Dabei wurden auch einige Vereinfachungen angenommen, die das Ergebnis des errechneten Energieertrags beeinflussen könnten.

Sofern mit den beschriebenen Formeln nur mit einem Stundenwinkel gerechnet wird und die wahre Ortszeit ungenutzt bleibt, tritt eine zeitliche Verschiebung auf, die für die Energieberechnung keine Auswirkungen hat. Abweichungen dagegen können die Annahmen bringen, dass die Deklination während eines Tages gleich bleibt und dass die Bahn der Erde um die Sonne jedes Jahr gleich ist.

Interessanter dagegen sind möglich Auswirkungen auf den Sonnenverlauf dadurch, dass HAP ein hochfliegendes Flugzeug ist und in mehreren km Höhe fliegen soll. Eine Berücksichtigung der Höhe wird in den vereinfachten Formeln nicht vorgenommen. Die Höhe zusammen mit der Vereinfachung, dass die Erde eine Kugel und kein Ellipsoid ist, könnte Auswirkungen haben. Die letzte und bedeutendste Vernachlässigung sind Einflüsse der Atmosphäre auf

den Elevationswinkel. Die Refraktion beschreibt dabei den Effekt, dass die Sonne höher am Himmel erscheint, als sie tatsächlich ist und ist durch den vom Vakuum verschiedenen Brechungsindex der Atmosphäre zu erklären. Der Effekt ist bei einem Zenitwinkel von 80 bis 90°, also nah über dem Horizont, am stärksten [7, S. 15]. Dies mag für Flüge weit im Norden, wie in Kiruna, sehr relevant sein, da die Sonne dann mehrere Stunden knapp am Horizont steht.

Zur Vermeidung von möglichen Simulationsungenauigkeiten soll gleich von Beginn an diese Fehlerquelle ausgeschlossen werden und ein bestehendes Modell zur Bestimmung des Sonnenstands genutzt werden.

3.2.1 Bestehende Modelle

Bei einer Recherche lassen sich mehrere Sonnenpositionsmodelle finden und es fällt dabei auf, dass viele dieser auf den Solar Position Algorithm (SPA) des National Renewable Energy Laboratory (NREL) [25] referenzieren und sich mit diesem vergleichen. Dieser zeichnet sich durch besonders hohe Genauigkeit aus und ist für astronomische Aufgabenbereiche geeignet. Im Gegenzug benötigt er dafür eine relativ hohe Rechenzeit.

Ein weiteres Modell liefert der SG2 Algorithmus von Ph. Blanc und L. Wald [26] der für die vergangenen und aktuellen Dekaden bis 2030 seine hohe Genauigkeit beibehalten kann. Dieser Algorithmus setzt auf Approximationen und ist deshalb nur für den genannten Zeitraum gültig, hat aber den großen Vorteil, dass er auf Schnelligkeit optimiert ist. Seine Genauigkeit liegt im Vergleich zum SPA bei einer Abweichung in der Größenordnung von Gradsekunden. Die Kombination aus Genauigkeit und Schnelligkeit macht ihn besonders attraktiv für die spätere Anwendung auf dem Echtzeitrechner, weshalb dieses Modell implementiert wurde.

3.2.2 Implementierung des Sonnenpositionsmodells

Aus genannten Gründen wurde der SG2 Algorithmus in MATLAB implementiert. Hierbei wird zunächst die gegebene Zeit in ein Julianisches Datum umgerechnet, um aus astronomischer Sicht eine definierte Zeitangabe zu besitzen. Diese wird im nächsten Schritt dazu genutzt heliozentrische Koordinaten für die Erde zu berechnen, also aus Sicht des Sonnenzentrums. Darauf folgt eine Umrechnung der vorhandenen Werte in geozentrische Parameter, die entsprechend den Stand der Sonne mit Sicht aus dem Zentrum der Erde angeben. Im letzten

Schritt folgt die Berechnung der gesuchten topozentrischen Koordinaten, die die Position der Sonne aus Sicht von der Erdoberfläche aus beschreiben.

Bereits während der Implementierung ist aufgefallen, dass in der angegebenen Berechnung des Julianischen Datums ein Fehler vorhanden ist und Schaltjahre die durch 400 teilbar sind nicht, also falsch, gezählt wurden. Weiter ist aufgefallen, dass für die Berechnung von ω^g keine Formel oder fester Wert angegeben wird, welcher allerdings in den Formeln (33) und (35) benötigt wird. Durch Treffen einer Annahme konnte diese Lücke geschlossen werden und der Algorithmus vervollständigt werden.

Berechnungen von Tagesverläufen der Sonne haben ein widersprüchliches Verhalten gezeigt. Eine Änderung der besagten Annahme z. B. durch verschiedene Konstanten haben keine merklichen Veränderungen bewirkt. Es konnte weiter festgestellt werden, dass die Berechnung der Sonnenstands zu einer festen Uhrzeit mit Variation der Jahre zu einer periodischen Schwingung mit einer Periode von 50 Jahren und einer Amplituden von knapp 45° am Elevationswinkel führt. Dies ist ein weiteres unrealistisches Verhalten, das als Fehler gewertet wird aber die vorherige Beobachtung erklärt. Bei Berechnungen innerhalb eines Jahres, in dem die besagte Jahresdrift bei 0 liegt, konnte als weiterer Fehler beobachtet werden, dass der Effekt der Refraktion deutlich zu stark ist.

Eine mehrfache Kontrolle der implementierten Formeln und abgeschriebenen Konstanten konnte keine fehlerhafte Implementierung aufdecken. Aufgrund des deutlich zu großen zeitlichen Aufwands für dieses Modell und des mangelnden Erfolgs, wurde eine Weiterarbeit gestoppt und der Fortschritt verworfen.

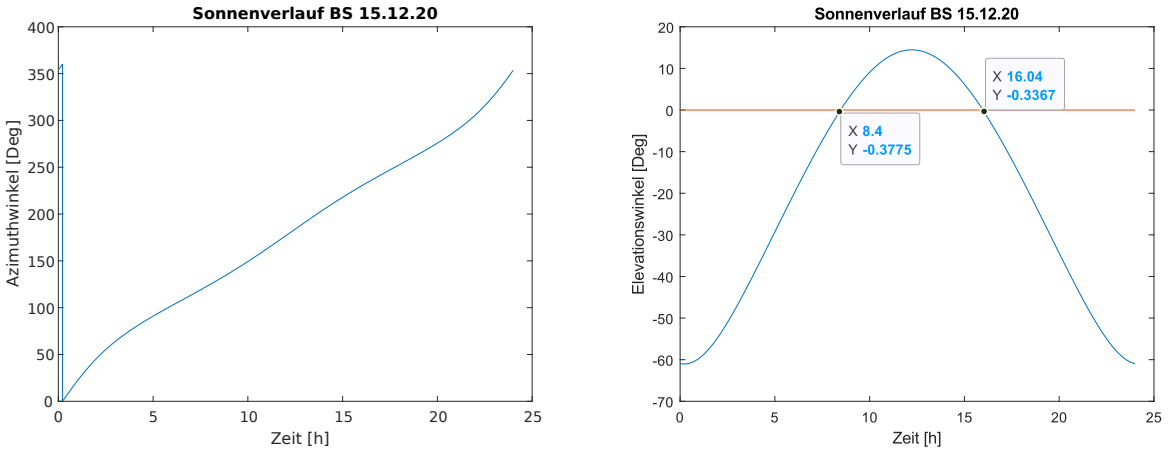


Abbildung 3.3: Vom SPA berechneter Sonnenverlauf in Braunschweig am 15.12.2020

Als Alternative wurde eine bestehende Implementierung eines Sonnenpositionsmodells als MATLAB-Code gesucht, um den Entwicklungsaufwand minimal zu halten. Dabei konnte

eine fertige SPA Implementierung im MATLAB File Exchange gefunden werden [27]. Die fertige Funktion liegt als MATLAB-Skript vor und musste für den eigenen Gebrauch in ein passendes Format für einen MATLAB-Function Block umgeschrieben werden. Diese Umgebung verbietet einige genutzte Funktionen mit den Datenstrukturen, da sie bei einer späteren Umwandlung in C-Code nicht möglich wären. Eine Formatanpassung konnte durchgeführt werden und der MATLAB-Function Block wurde für Simulink zugänglich gemacht. Ein beispielhaftes Simulationsergebnis ist in Abbildung 3.3 gezeigt.

3.2.3 Validierung des Sonnenpositionsmodells

Es mag zu erwarten sein, dass ein fertig implementierter Algorithmus, der seit mehreren Jahren online ist, korrekt funktioniert, doch wie das Beispiel des SG2 Modells gezeigt hat, sollte im jedem Fall eine Prüfung vorgenommen werden. Eine Überprüfung ist durch den Vergleich der aktuellen lokalen Sonnenposition mit der berechneten Position möglich. Dies kann ohne genaue Messinstrumente nur als eine Abschätzung gewertet werden.

Eine andere Möglichkeit stellt den Vergleich der eigenen Simulationsergebnisse mit einem von mehreren online verfügbaren Implementierungen dar. Ein als vertrauenswürdig einzustufenden Online-Rechner mag der von der National Oceanic and Atmospheric Administration (NOAA) des US Handeisministeriums gelten. Dieser ist zu finden unter https://www.esrl.noaa.gov/gmd/grad/solcalc/. Ein weiterer zum unabhängigen Vergleich kann der von Sonnenverlauf.de unter https://www.sonnenverlauf.de sein.

Als Vergleichstage wurden Sommer und Wintertage genommen, sowie der 21.9. als mögliche Tagundnachtgleiche. Diese Tage wurden auf verschiedenen Breitengraden hinsichtlich dem Sonnenhöchststand, Sonnenauf- und untergang verglichen. Ein Beispiel wurde bereits in Abbildung 3.3 gezeigt. Das Ergebnis war jeweils eine maximale Abweichung in der Größenordnung von Gradminuten. Die korrekte Funktion des SPA aus dem MATLAB File Exchange gilt damit als bestätigt.

3.3 Spektralmodell

Aufgabe des Spektralmodells ist die Berechnung der einzelnen spektralen Intensitäten des Lichts an dem gegebenen Ort unter Berücksichtigung des Sonnenstands und der Atmosphärischen Eigenschaften. Das Modell und sein Spektrum als Ausgabe wird benötigt, um die Solarzellen passend auf die Änderungen des Lichts reagieren zu lassen. Dies ermöglicht auch

detailliertere Untersuchungen in Hinsicht der Auswirkungen der Atmosphäre oder des Sonnenstands auf die Leistung der Solarzellen.

Ähnlich wie zum Sonnenstandmodell soll auch hier ein existierendes Modell recherchiert und implementiert werden. Eine Suche hat ein passendes Paper hervorgebracht, das ein einfaches Spektralmodell für direktes und diffuses Licht unter klarem Himmel beschreibt [28]. Die Einschränkung des wolkenlosen Himmels stellt in Bezug auf HAP keine Problem dar, da das Flugzeug entweder über dem Wettergeschehen fliegen wird oder für Start- und Landemanöver gutes Wetter vorausgesetzt wird. Das modellierte Spektrum umfasst dabei die Wellenlängen von 0,3 bis 4,0 μm und berücksichtigt die wichtigsten atmosphärischen Eigenschaften. Im Folgenden wird die Implementierung des Modells SPCTRAL2 von Bird kurz beschrieben.

3.3.1 Implementierung des Spektralmodells

Im ersten Schritt des Modells werden Konstanten definiert die das extraterrestrische Ausgangsspektrum beschreiben sowie Koeffizienten für Ozon, gemischtes Gas und Wasserdampf, die deren Einflüsse auf das Spektrum spezifizieren. Dabei ist jede beschriebene Formel und auch die genannten Konstanten eine von der Wellenlänge abhängige Größe. MATLAB bietet hier den Komfort das Ausgangsspektrum als Array bzw. Vektor zu definieren und die mathematischen Operatoren direkt auf das Array anzuwenden, wobei es intern auf die Elemente das Arrays angewendet wird. Dies sorgt auch für eine bessere programmatische Übersicht.

Die zentrale Formel zur Berechnung der direkten Strahlung $I_{d\lambda}$ lautet wie folgt:

$$I_{d\lambda} = H_{o\lambda} D T_{r\lambda} T_{a\lambda} T_{w\lambda} T_{o\lambda} T_{u\lambda} [\frac{W}{m^2 \mu m}] \tag{3.1}$$

Dabei steht $H_{o\lambda}$ für das extraterrestrische Spektrum, welches mit einem normierten Abstand zur Sonne D und weiteren Transmissionsfunktionen multipliziert wird. Diese beschreiben respektive die Rayleigh-Streuung $T_{r\lambda}$, die Aerosoldämpfung $T_{a\lambda}$, die Wasserdampfabsorption $T_{w\lambda}$, die Ozonabsorption $T_{o\lambda}$ und die Absorption durch Mischgase $T_{u\lambda}$ [28, S. 87]. Die einzelnen Funktionen werden beschrieben und die direkte Strahlung kann berechnet werden. Im nächsten Schritt wird die diffuse Strahlung $I_{s\lambda}$ über folgende Formel errechnet:

$$I_{s\lambda} = (I_{r\lambda} + I_{a\lambda} + I_{g\lambda}) C_s \tag{3.2}$$

Sie setzt sich dabei aus den Anteilen der Rayleigh-Streuung $I_{r\lambda}$, der Aerosolstreuung $I_{a\lambda}$ und die Komponente $I_{g\lambda}$ zusammen, wobei letztere die mehrfachen Reflektionen des Licht

am Boden und mit der Luft beschreibt [28, S. 90]. Die Summe dieser diffusen Teilstrahlungen wird anschließend mit dem Korrekturfaktor C_s multipliziert, der nur das Licht niedrigerer Wellenlängen beeinflusst.

Zum Schluss wird die Gesamtstrahlung $I_{T\lambda}$ auf einer ebenen Fläche berechnet, die sich aus der direkten und diffusen Strahlung zusammensetzt und die den Zenitwinkel der Sonne (hier Z) als abhängige Größe besitzt. Die Formel lautet:

$$I_{T\lambda} = I_{d\lambda} cos(Z) + I_{s\lambda} \tag{3.3}$$

Nach der Implementierung aller Formel wurden die nötigen Eingangsgrößen des Modells sichtbar. Diese sind im wesentlichen der Tag im Jahr d, der Zenitwinkel der Sonne θ_S (bzw. hier Z), der lokale Luftdruck p, die Ozonmenge O_3, die optische Dicke der Aerosole bei 500 nm β_n (im Code $tau500$) und die Menge an Wasserdampf in der vertikalen Bahn W.

Als Anmerkung sei noch erwähnt, dass die alleinige Befolgung des Papers an wenigen Formeln zu Unklarheiten oder offenen Stellen führt, die allerdings durch einen Vergleich mit einer bestehenden C Implementierung beseitigt werden konnten [29].

3.3.2 Validierung des Spektralmodells

Zur Validierung des Spektralmodells soll ein Vergleich mit den bereits genannten Spektren des ASTM Standards erfolgen (siehe Abschnitt 2.2.2). Dies hat nebenbei den Hintergrund, dass die Funktion des Spektralmodells besonders in dem Bereich Gültigkeit beweisen muss unter denen die Solarzellen von HAP ausgesetzt sind. Da das Flugzeug auf einer Höhe von 15,5 bis über 20 km Höhe fliegen soll, unterliegt es damit einem Licht, dass im Tagesverlauf zwischen dem von AM0 und AM1,5 liegt.

Der erste Vergleich soll mit dem AM0 Spektrum nach dem ASTM E-490 Standard erfolgen. Am Spektralmodell müssen dazu die passenden Eingangsgrößen eingestellt werden. Zunächst soll für d der Tag im Jahr angegeben werden, an dem zwischen Erde und Sonne der normierte Abstand von $D = 1$ herrscht. Eine Iterationsrechnung mit dem Modell ergab die Ergebnisse $d_1 = 278$ und $d_2 = 93,5$ für das Jahr 2020, wobei hier mit einem geradzahligem Tag weitergerechnet werden soll. Damit ergibt sich für $d = 278$ der 4.10.2020. Als nächster Parameter wird der Zenitwinkel der Sonne auf $\theta_S = 0$ festgelegt, da die sichtbare Sonne im Weltall immer direkt auf den Betrachter scheint und es keine Bezugsfläche gibt, mit der ein Zenitwinkel bestimmt werden kann. Die weiteren atmosphärischen Größen p, O_3, β_n und W

werden ebenfalls auf 0 gesetzt, da es im extraterrestrischen Raum keine Atmosphäre gibt. Abbildung 3.4 zeigt nun den Vergleich der beiden AM0 Spektren.

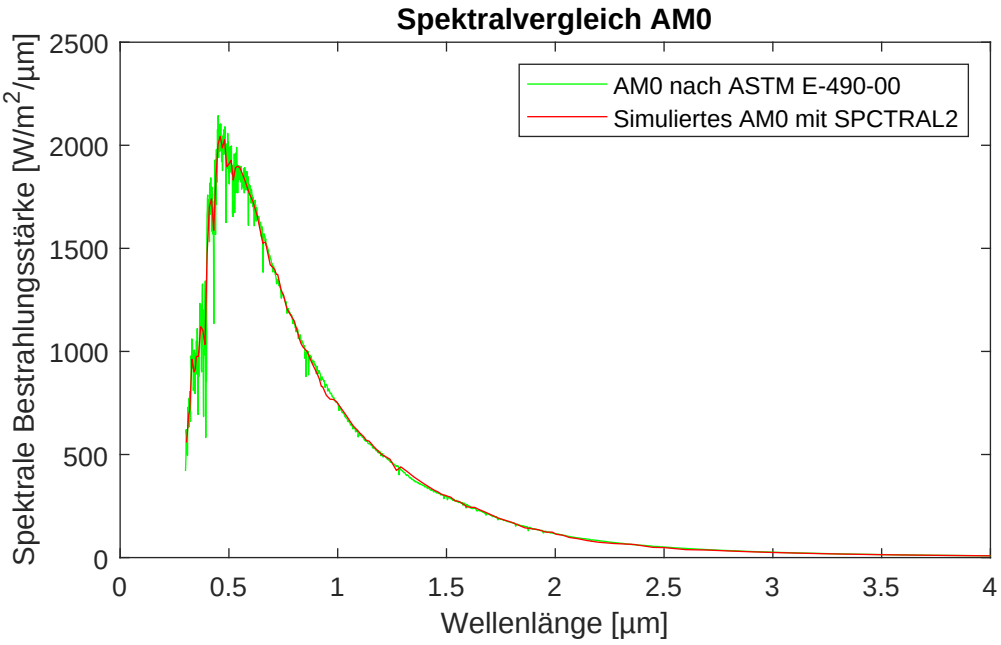


Abbildung 3.4: Spektralvergleich des definierten und simulierten AM0 Spektrums

Der Vergleich der beiden Graphen zeigt einen fast identischen Verlauf, was für ein sehr gutes Simulationsergebnis spricht. Der auffälligste Unterschied ist in den niedrigen Wellenlängen zu sehen, da der Standard hier ein deutlich feiner aufgelöstes Spektrum hinterlegt hat mit fast 1700 Datenpunkten über das gesamte Spektrum. Wohingegen das Modell nur 122 Datenpunkte benutzt, um das Spektrum wiederzugeben. Einzig kleine Ausreißer sind sonst bei ca. 0,9 μm und bei 1,25 μm zu sehen.

Ein mit Zahlen ausdrückbarer Vergleich soll über die Bestimmung der jeweiligen Bestrahlungsstärken erfolgen. Dazu werden alle spektralen Bestrahlungsstärken über den Wellenlängenbereich von $0,3\ \mu m$ bis $4,0\ \mu m$ integriert, da das Modell nur diesen Bereich abbildet. Es ergibt sich eine Bestrahlungsstärke von $1339,4\ W/m^2$ für das AM0 der ASTM Definition und $1336,1\ W/m^2$ für das SPCTRAL2 Modell. Die Abweichung des Modells liegt in diesem Fall bei ca. $0,25\%$, was als sehr genau gewertet wird.

Dabei muss angemerkt werden, dass es sich bei dem beschriebenen Spektrum nur um den direkten Anteil des Lichts handelt. Das diffuse Licht wirkt zusätzlich mit ca. 17 W/m^2, was ohne Atmosphäre nicht möglich sein sollte. Die Erklärung liegt darin, dass das Modell hier trotzdem die Reflektionen von der Erdoberfläche wiedergibt, da der Albedo als fester Wert

im Code hinterlegt wurde. Aufgrund der Tatsache, dass der ASTM Standard für AM0 nur den direkten Anteil des Lichts von der Sonne vorsieht, stimmt der beschriebene Vergleich.

Der zweite Vergleich soll mit dem AM1,5 Spektrum nach dem ASTM G-173 Standard erfolgen. Im Gegensatz zum AM0 Spektrum können hier für die atmosphärischen Bedingungen keine Annahmen getroffen werden. Die benötigten Parameter werden in [12] beschrieben und sie sind mit $\theta_S = 41,18°$, $O_3 = 0,34$ cm, $\beta_n = 0,084$ c und $W = 1,42$ cm festgelegt. Es wird geschrieben, dass die 1976 U.S. Standardatmosphäre genutzt wird, allerdings nicht von welcher Höhe ausgegangen wird. Hier wird nun die Annahme von 0 m NHN gemacht, womit sich der Luftdruck zu $p = 1013$ hPa ergibt. Der letzte Parameter wird mit $d = 278$ angenommen, da sich hier die mittlere Entfernung von Erde zu Sonne einstellt wie im vorherigen Vergleich beschrieben. Der Vergleich der beiden AM1,5 Spektren ist in Abbildung 3.5 zu sehen.

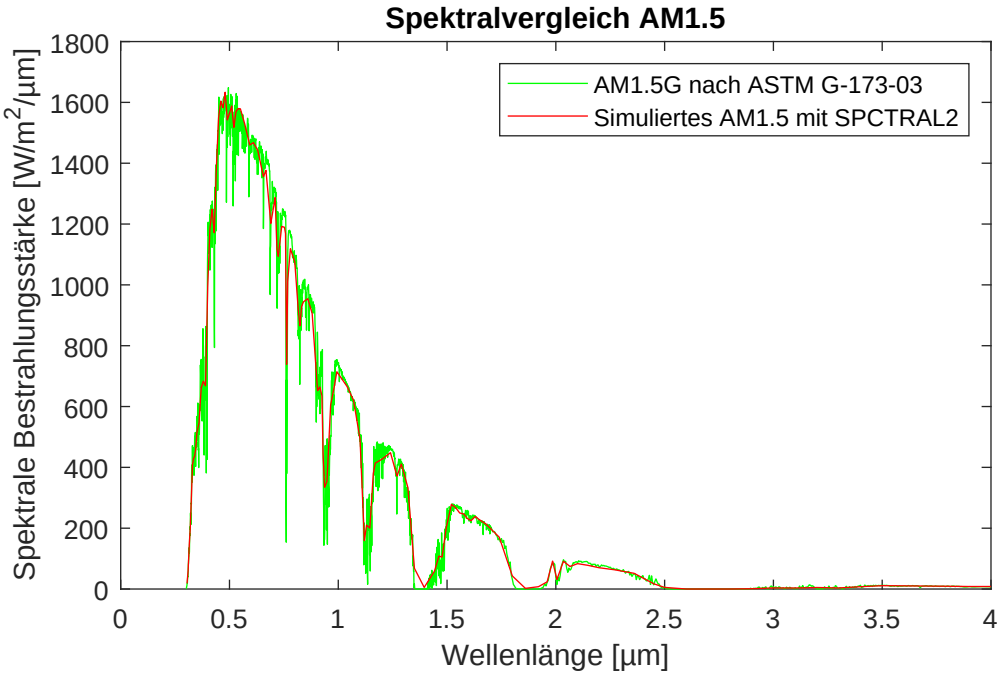


Abbildung 3.5: Spektralvergleich des definierten und simulierten AM1,5 Spektrums

Wie im vorherigen Vergleich ist auch hier eine sehr gute Übereinstimmung beider Graphen zu sehen, die keine merklichen Ausreißer aufweist. Einzige Unterschiede des Verlaufs sind auf die geringere Auflösung des Modells zurückzuführen. Ein Vergleich der beiden Bestrahlungsstärken im Bereich von $0,3$ μm bis $4,0$ μm ergibt für das ASTM Spektrum $1000,4$ W/m^2 und $1003,8$ W/m^2 für das modellierte Spektrum. Dies entspricht einer Abweichung von ca. $0,33\%$ und wird wieder als sehr gutes Ergebnis gewertet.

Mit den beiden Vergleichen konnte gezeigt werden, dass das implementierte Spektralmodell

funktionsfähig ist und in den beiden Testfällen sehr genaue Ergebnisse liefert. Es ist davon auszugehen, dass sofern das Modell korrekte Atmosphärische Eingangsdaten bekommt auch für verschiedene Szenarien die korrekten Spektralverhältnisse voraussagen kann.

3.4 Aufbau des Solarzellenmodells

Für die Entwicklung des Solarzellenmodells muss zunächst die reale Solarzelle in ihrem Aufbau und ihren Eigenschaften verstanden werden. Danach kann ein Solarzellenmodell entwickelt werden, das möglichst hohe Güte bei einer gleichzeitig schnellen Ausführungsgeschwindigkeit verspricht. In der Umsetzung nutzt das Zellenmodell physikalische Zusammenhänge und Eigenschaften der Zelle, die über Messungen ermittelt wurden. Eine Validierung des Solarzellenmodells erfolgte über einen Vergleich mit realen Messungen, die in der Simulation nachgestellt wurden. Die betrachteten Vergleiche der Messungen spiegeln die Einsatzbedingungen der Solarzellen auf dem Flugzeug wieder.

3.4.1 Solarzellen von HAP

Vor Beginn der Implementierung muss zunächst verstanden werden, wie die abzubildende reale Solarzelle aufgebaut ist und welche Eigenschaften sie besitzt. Dazu werden in diesem Abschnitt die Solarzellen von HAP näher beschrieben. Es handelt sich um Triple-Junction-Solarzellen auf Galliumarsenid Basis der Firma MicroLink. Sie wurden ausgewählt, da sie hocheffizient sind mit $\eta \approx 30\%$, sehr dünn mit unter 30 μm Dicke und damit relativ leicht und zuletzt weil sie flexibel sind und damit für das hochflexible Flugzeug geeignet sind. Die Materialien der einzelnen Subzellen sind InGaP (Top), GaAs (Middle) und InGaAs (Bottom).

Abbildung 3.6 a) zeigt nun die Solarzellen von MicroLink samt Verkapselung zum Schutz vor äußeren Einflüssen und als zwei Strings, die je aus fünf Solarzellen bestehen. Ein String bezeichnet dabei eine Reihenschaltung mehrerer Zellen. In Abbildung 3.6 b) ist ein Ersatzschaltbild einer einzelnen Solarzelle gezeigt. Da es sich um eine Triple-Junction-Solarzelle handelt, besteht sie aus drei Subzellen. In diesem Fall sind die Subzellen in Reihe geschaltet, um ihre Spannungen zu addieren. Aus dem Ersatzschaltbild wird auch ersichtlich, dass jede Zelle eine Bypass-Diode besitzt, durch die Verschattungsverluste im Verbund reduziert werden. Diese Dioden werden bei der Produktion bereits mit in die Zelle integriert. Da im

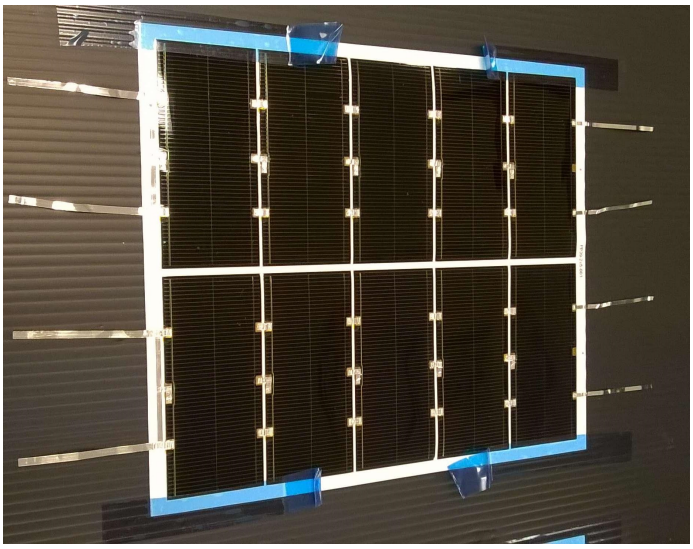
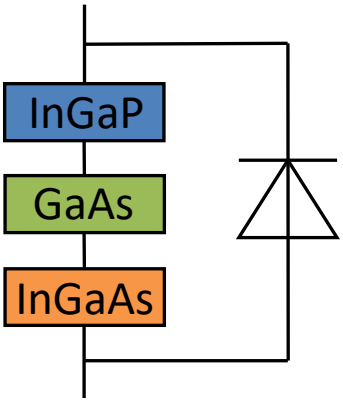
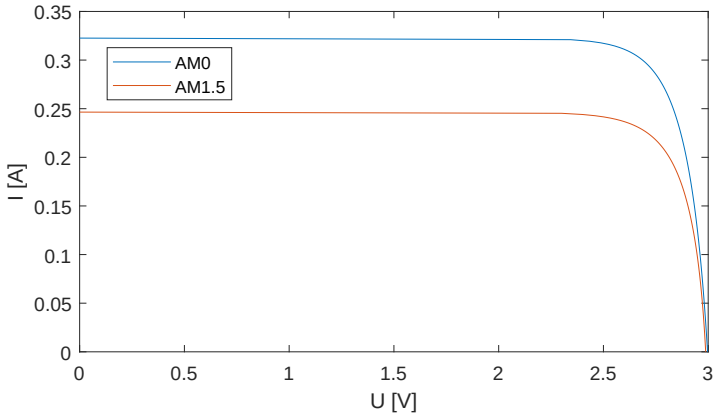
a)  b) 

Abbildung 3.6: a) Triple-Junction-Zellen von Microlink (2x5 Zellen) b) Ersatzschaltbild

Rahmen dieser Arbeit keine Betrachtung von Verschattungen stattfindet und sie sonst keine Auswirkungen auf die Verschaltung haben, können diese Dioden ignoriert werden.



	AM0	AM1.5
U_{oc} [V]	2,997	2,988
I_{sc} [A]	0,3226	0,2465
U_{mpp} [V]	2,632	2,634
I_{mpp} [A]	0,3075	0,2339
P_{max} [W]	0,8093	0,616
η [%]	29,62	30,79

Abbildung 3.7: Kennlinie und Kennwerte der Zellen von MicroLink

In der Abbildung 3.7 sind die Kennlinien und Kennwerte der Solarzellen für AM0 und AM1,5 Bedingungen aufgeführt. Dabei ist zu erwähnen, dass die gezeigten Kennwerte für AM0 aus einer Lieferung mit 40 Solarzellen stammen und den Mittelwert dieser darstellen. Jede der Zellen wurde einzeln bestrahlt und dabei vermessen [30]. Die einzelnen Messwerte sind im Anhang in Abbildung A.2 zu finden. Die zugehörige Kennlinie wurde mit dem Programm *PV-Teach* berechnet welches unter www.lehrbuch-photovoltaik.de kostenlos heruntergeladen werden kann [6, S. 109].

Da die gekauften und vermessenen Zellen keine Messwerte für das AM1,5 Spektrum beinhalten, wurden diese separat beim Hersteller angefragt. Als Antwort wurden die Werte mit-

geteilt, die im Anhang in Abbildung A.3 zu sehen sind. Aus den dortigen AM1,5 Messwerten wurde ebenfalls der Mittelwert gebildet und die dazugehörige Kennlinie mit *PV-Teach* berechnet, welche in Abbildung 3.7 dargestellt sind. Da es sich in diesem Fall nur um 4 Messwerte für das AM1,5 Spektrum handelt und die Zellen aus einer anderen Charge stammen, variieren sie leicht gegenüber denen des ersten Messsatzes. Dies wird besonders beim Vergleich der AM0 Messwerte sichtbar.

3.4.2 Implementierung des Solarzellenmodells

Dieses Solarzellenmodell unterscheidet sich von den sonst in der Literatur beschriebenen Modellen dadurch, dass kein mathematisches Modell zur Beschreibung der Zelle genutzt wird. Dies ist einerseits dadurch begründet, dass die dafür nötigen Formeln unbestimmt sind und auf beiden Seiten des Gleichheitszeichens entweder den Strom I oder die Spannung U stehen haben. Zur Lösung dieser Gleichungen müssen numerische Verfahren angewendet werden. Dies kann aus Sicht der Echtzeitberechnung problematisch werden, da die Lösungsberechnung iterativ erfolgt und damit nicht deterministisch und allgemein rechenaufwändig ist.

Der zweite Grund ist das für dieses mathematische Ersatzschaltbild sehr genaue Charakteristika der Zelle bekannt sein müssen, wie der Innenwiderstand, der Sättigungsstrom und der Idealitätsfaktor der dazu angenommenen Diode. Die Problematik ergibt sich daraus, dass diese Modelle, soweit bekannt, nie für Multi-Junction-Zellen angegeben oder genutzt werden. Theoretisch müssen dazu nur drei Modelle gleichzeitig berechnet werden, doch damit ergibt sich, dass für jede Subzelle die Charakteristika bekannt sein müssen. Diese sind an der Gesamtzelle von außen nicht einzeln messbar und können somit unter normalen Umständen nur vom Hersteller bereitgestellt werden.

Das implementierte Solarzellenmodell nutzt stattdessen Messwerte und die physikalischen Zusammenhänge dieser, um ein deterministisch berechenbares Modell zu schaffen. Die genutzten Messwerte stammen dabei direkt vom Hersteller, könnten aber auch in jedem Labor mit geeigneter Messausrüstung ermittelt werden. Die verwendete Methode ist darüber hinaus auch auf jede Art von Solarzelle anwendbar.

Zu Beginn wird im Solarzellenmodell das einfallende Licht betrachtet. Im Fall dieser modellierten Solarzellen trifft das Licht zunächst auf die Verkapselung der Zelle. Unabhängig davon ob es sich zuerst um eine Verkapselung handelt oder die reine Zelle, der folgende Brechungsindex wird ein anderer sein, als der vorherige (normalerweise Luft). Bei diesem

Materialübergang gibt es Anteile des Lichts, die in die Verkapselung bzw. Solarzelle transmittieren und Anteile die reflektieren. Dies kann durch die Fresnelschen Formeln beschrieben werden [31]. Sind soweit alle Materialparameter bekannt bestehen für den Transmissions- und Reflektionsgrad weiterhin Abhängigkeiten vom Einfallswinkel und der Wellenlänge des einfallenden Lichts. Im Fall dieser Arbeit wurden entsprechende Informationen von MicroLink bereitgestellt und sind in Abbildung 3.8 in graphischer Form dargestellt.

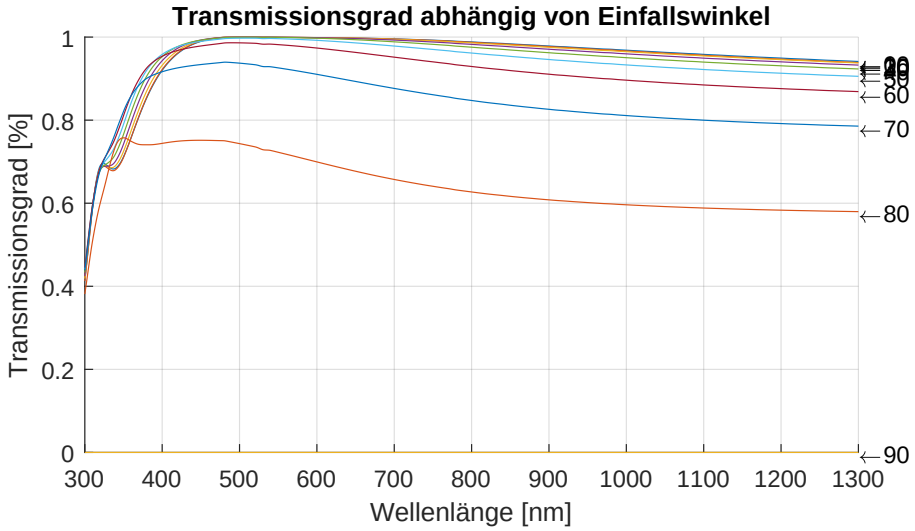


Abbildung 3.8: Transmissionsgrad der Solarzellen von Microlink

Die in Abbildung 3.8 gezeigten Funktionen wurden in einer Look-Up-Tabelle genutzt, um so die Transmissionen des einfallenden Lichts genau abzubilden und gleichzeitig eine schnelle Berechnung zu ermöglichen. Die Eingangsgrößen für die Transmissionsberechnung sind das aus dem Spektralmodell kommende Spektrum und der Einfallswinkel. Als Ausgangsgröße folgen die transmittierten spektralen Anteile des Lichts.

Im nächsten Schritt der Berechnung wird das transmittierte Spektrum mit der EQE der Solarzelle gefaltet, die als Look-Up-Tabelle vorliegt. Die dazugehörigen Daten stammen ebenfalls von MicroLink. Es ist dabei wichtig zu erwähnen, dass diese EQE an einer Solarzelle gemessen wurde, die keine Verkapselung besaß. Nur deshalb dürfen beide genannten Berechnungen hintereinander ausgeführt werden. Die EQE der MicroLink Triple-Junction-Solarzellen ist in Abbildung 3.9 gezeigt. Hierbei ist sichtbar, dass es sich genau genommen um drei EQEs handelt, die jeweils für die einzelnen Subzellen gelten. Nachdem das verbliebene Spektrum mit den einzelnen EQEs gefaltet wurde, ergibt sich als Ausgangsgröße für jede Subzelle ein Array von spektralen Leistungsdichten, das für die Ausbildung von freien Ladungsträgern sorgt. Diese unanschauliche Größe wird im nächsten Abschnitt genutzt, um für jede Wellenlänge die Anzahl an Photonen auf der gegebenen Fläche der Solarzelle zu

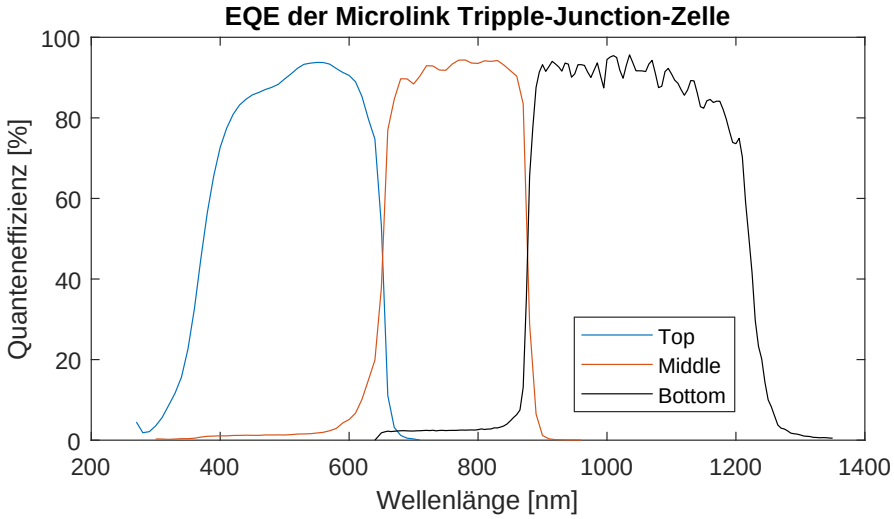


Abbildung 3.9: Quantenwirkungsgrad der Solarzellen von Microlink

berechnen. Dadurch, dass eine Leistung pro Wellenlänge gegeben ist lässt sich mit

$$E_{\text{Photon}} = h \cdot c/\lambda \tag{3.4}$$

und einer Division durch die Leistung die genaue Anzahl der Photonen dieser Wellenlänge berechnen. Da nach der EQE jedes dieser Photonen einen Ladungsträger erzeugt, lässt sich durch eine Aufintegration über die Wellenlängen die Gesamtanzahl der zur Verfügung stehenden Elektronen einer Subzelle berechnen. Dies entspricht gleichzeitig dem Kurzschlussstrom I_{sc} der Subzelle (vergleiche Quellcode A.1).

Der nächste Schritt ist durch den Aufbau der MicroLink Zellen begründet. Da die Triple-Junction-Zellen aus drei in Reihe geschaltete Subzellen bestehen kann nur der kleinste der drei Subzellenströme fließen. Deshalb gilt für den Kurzschlussstrom der Gesamtzelle das eine Minimum-Funktion über die Subzellenströme gebildet werden muss. Dieser Zusammenhang ist in Abbildung 3.10 verdeutlicht.

In der Abbildung wird ebenfalls die Strategie zur Berechnung der Gesamtkennlinie angedeutet. Es wird das physikalische Prinzip ausgenutzt, dass der Kurzschlussstrom einer Solarzelle proportional zur Bestrahlungsstärke ist, also $I_{sc} \sim E$. Diese Proportionalität ist bereits im Modell enthalten, dadurch dass nur Multiplikationen durchgeführt wurden. Wenn nun eine Kennlinie einer Solarzelle vorhanden ist, von der bekannt ist, unter welcher Bestrahlungsstärke sie gemessen wurde, kann diese als Vorlage für die weitere Berechnung benutzt werden.

Der nächste Berechnungsblock nutzt nun die bekannte Kennlinie der Solarzelle unter AM0 Bedingungen und verschiebt diese in vertikaler Richtung um die Differenz des aktuellen I_{sc}

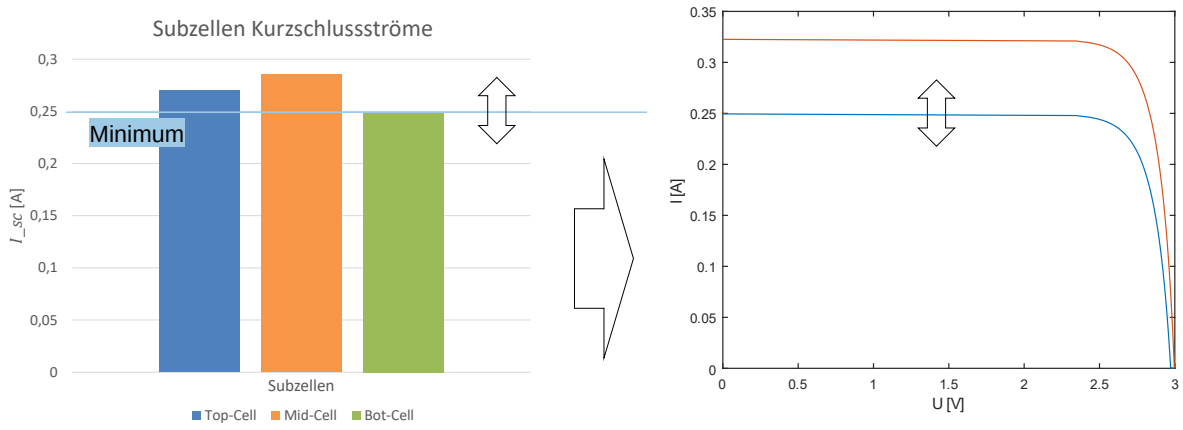


Abbildung 3.10: Kurzschlussstromberechnung im Solarzellenmodell

zu dem I_{sc} der hinterlegten Kennlinie. Dieses Delta der Ströme wird auf jeden Messwert der Kennlinie addiert, wodurch sich die Verschiebung ergibt (vergleiche Quellcode A.2). Ein zweiter Zusammenhang der Bestrahlungsstärke besteht zu der Leerlaufspannung. Diese ändert sich nur mit dem natürlichen Logarithmus der Bestrahlungsstärke also $U_{oc} \sim ln(E)$. Eine Bestätigung, dass sich bei dieser Methode die Leerlaufspannung korrekt mit verschiebt, wird später in der Validierung gebracht. Zu diesem Punkt kann also die Kennlinie der Solarzelle in Abhängigkeit zum Spektrum und dem Einfallswinkel berechnet werden. Allerdings gilt diese Kennlinie nur für 25°C.

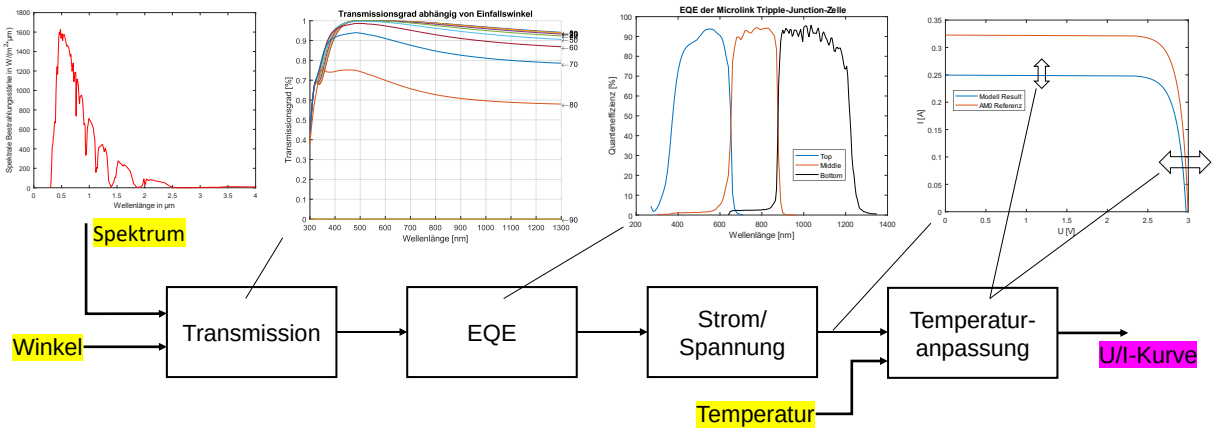


Abbildung 3.11: Zusammenfassung des Solarzellenmodells

Im letzten Schritt des Solarzellenmodells folgt deshalb noch eine Temperaturanpassung der Kennlinie. Dies geschieht mithilfe der Temperaturkoeffizienten der Zelle, die von Hersteller mit $TK_{Voc} = -6\frac{mV}{K}$ und $TK_{Isc} = 0,26\frac{mA}{K}$ angegeben werden. Mithilfe dieser wird im letzten Schritt eine Verschiebung und Streckung bzw. Stauchung der Kennlinie durchgeführt (vergleiche Quellcode A.3).

Die Abbildung 3.11 zeigt noch einmal zusammengefasst welche Berechnungsschritte mit welchen Daten durchgeführt werden.

3.4.3 Validierung des Solarzellenmodells

Die Validierung des Solarzellenmodells soll zeigen, dass das Modell das Verhalten der echten Solarzelle korrekt abbilden kann. Dazu sollen mehrere Vergleiche mit realen Messdaten des Herstellers angestellt werden, die in der Simulation nachzustellen sind. Die gewählten Messungen stellen Einsatzbedingungen dar, unter denen die Solarzellen im späteren Betrieb stehen werden. Sie sind somit repräsentativ für den Arbeitsbereich des Solarzellenmodells.

Vorweg soll die angekündigte Bestätigung folgen, dass eine Verschiebung einer bestehenden Kennlinie ein valides Mittel ist, um eine neue und weiterhin korrekte Kennlinie zu erzeugen. Wie bereits gesagt, gilt $I_{sc} \sim E$ und $U_{oc} \sim ln(E)$. Daraus lässt sich folgern, dass

$$\implies U_{oc} \sim ln(I_{sc}) \tag{3.5}$$

ebenfalls gelten muss. Dieser Zusammenhang wird nur für diesen Test verwendet, da es eine deutlich leichtere Umsetzung des Tests mit der bestehenden Architektur der Software erlaubt. Für die Durchführung wurde der Kurzschlussstrom I_{sc} der Kennlinie schrittweise gesenkt und dabei jeweils die neue Leerlaufspannung U_{oc} aufgenommen, die sich am Nulldurchgang der Kennlinie einstellt. Das Simulationsergebnis ist in Abbildung 3.12 zu sehen.

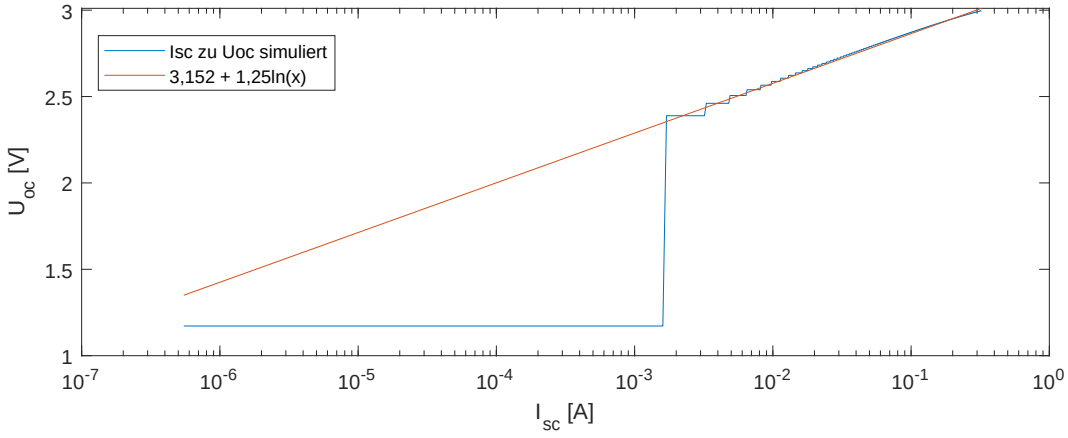


Abbildung 3.12: Logarithmischer Zusammenhang zwischen U_{oc} und I_{sc} bzw. E

Wie nun zu sehen ist befindet sich auf der X-Achse der logarithmisch aufgetragene Kurzschlussstrom I_{sc} und auf der Y-Achse die Leerlaufspannung U_{oc}. Die sich in diesem Zusam-

menhang einstellende Spannung ergibt aufgetragen eine annähernde Gerade. Die stufigen Abweichungen lassen sich dadurch erklären, dass diese Kennlinie berechnet wurde und die generierten Messwerte nicht gleich verteilt sind. Besonders die große Stufe am Anfang kommt dadurch zustande, dass nach dem ersten Messpunkt bei 0 V der zweite erst bei ca. 2,3 V folgt. In der Praxis wird diese Ungenauigkeit erst erreicht, wenn I_{sc} bei unter 2 mA liegt und die Zelle damit kaum relevante Leistung mehr bereitstellen kann.

Als erster Test wurde eine Messung des Kursschlussstrom-Einstrahlwinkel-Zusammenhangs nachgestellt. Hierbei wird die Solarzelle einem gleichbleibenden Spektrum ausgesetzt und es wird der Einfallswinkel des Lichts von 0 bis 90° variiert. Gemessen wird dabei der Kurzschlussstrom der Zelle. Die hierbei eingestrahlte Leistung ist allgemein proportional zum Cosinus des Einfallswinkels also $E \sim cos(\theta)$, da die relative Fläche immer kleiner wird. Mit flacher werdendem Einfallswinkel treten vermehrt zusätzliche Verluste durch verringerte Transmissionsgrade auf.

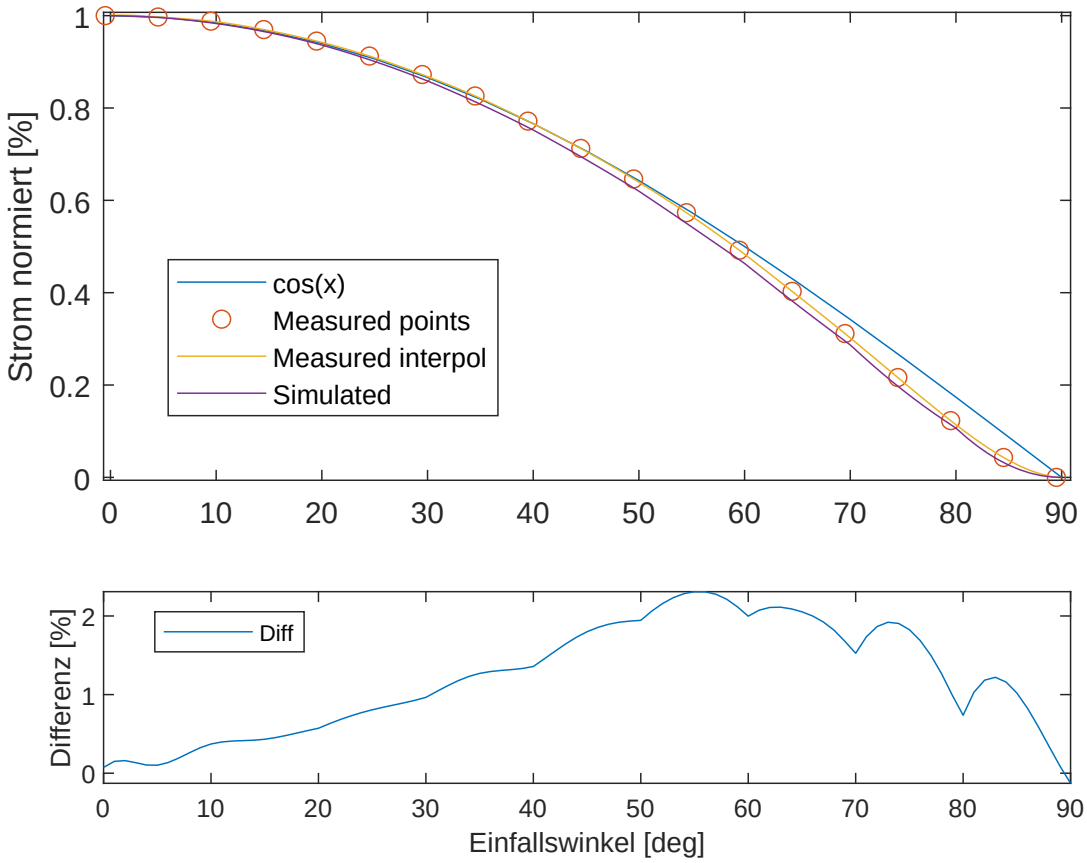
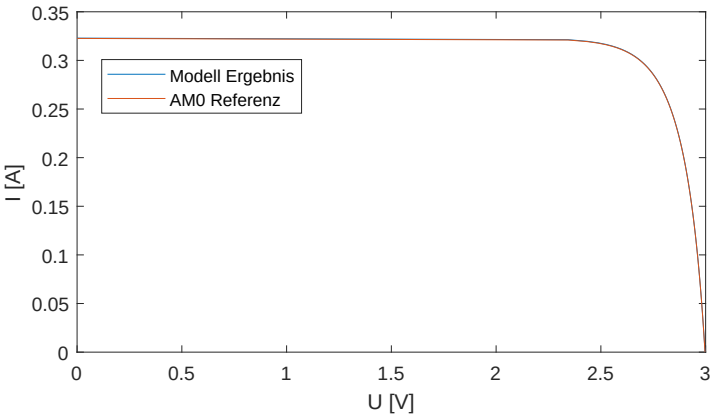


Abbildung 3.13: Kurzschlussstrom in Abhängigkeit zum Einfallswinkel

Die durchgeführte Messung und der simulierte Verlauf sind im oberen Graphen in Abbildung

3.13 zu sehen. Beide Verläufe folgen dem Cosinus bis der gemessene bei ca. 55° beginnt davon abzuweichen. Der simulierte Verlauf beginnt bereits deutlich früher bei ca. 30° abzuweichen. Dennoch bleibt die Differenz zwischen dem gemessenen und simulierten Graphen durchgehend gering, was im unteren Graphen zu sehen ist. Die maximale Differenz beträgt dabei 2,3 %. Dies zeigt, dass der Einfluss des Einfallswinkels ziemlich gut abgebildet werden kann.

Der zweite Test sah nun vor, das Solarzellenmodell einem AM0 Spektrum auszusetzen und die dabei entstehenden Kennwerte mit denen der realen Messungen zu vergleichen. Das Ergebnis ist in Abbildung 3.14 zu sehen.



	Real	Simuliert
U_{oc} [V]	2,997	2,997
I_{sc} [A]	0,3226	0,3229
U_{mpp} [V]	2,632	2,63
I_{mpp} [A]	0,3075	0,3083
P_{max} [W]	0,8093	0,8109
η [%]	29,62	29,66

Abbildung 3.14: Vergleich der realen und simulierten Zellkennwerte unter AM0 Bedingungen

Wie sich anhand der Kennlinie und den Kennwerten sehen lässt, sind beide Ergebnisse nahezu identisch. In allen Werten liegt die maximale Abweichung unter einem Prozent. Das Modell kann entsprechend das Verhalten unter AM0 Bedingungen perfekt wiedergeben.

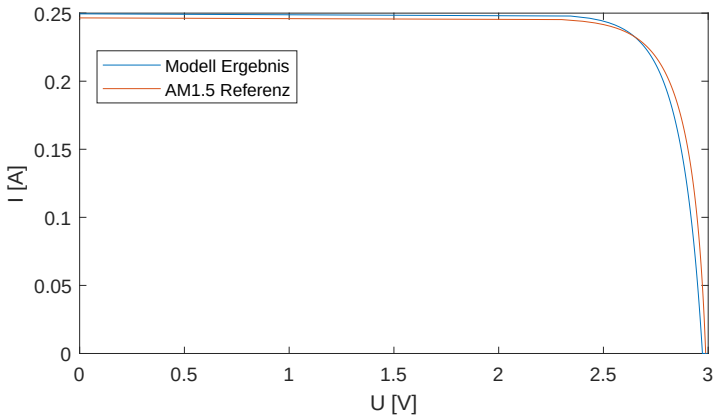
Als dritter und letzter Test sollten die Zellparameter des Modells unter AM1,5 Bedingungen ermittelt werden. Das Simulationsergebnis hat hierbei zunächst deutliche Abweichungen von den Messwerten gezeigt. Deshalb musste eine Modellanpassung vorgenommen werden. Dabei wurden durch zusätzliche Faktoren die Wirkungsgrade der EQE gesenkt. Auf Nachfrage beim Hersteller hat sich herausgestellt, dass die Angaben der EQE aufgrund des genutzten Messverfahrens höhere Werte aufweisen, als sie in der Praxis auftreten. Es wurde von Seiten des Herstellers geraten, die Werte der EQE um 5,5 % zu senken. Dies hat das Simulationsergebnis verbessert, doch es blieben merkliche Unterschiede.

Für eine Optimierung des Testergebnisses wurden die Subzellenströme bei den AM0 und AM1,5 Tests genauer betrachtet und die Faktoren für die EQEs individuell angepasst. Dieses Modell hat nicht zum Ziel die Solarzelle als physikalisch vollständig korrektes Modell

darzustellen, sondern soll nur dessen elektrische Performance möglichst gut wiedergeben. Deshalb werden die EQEs nun mit folgenden Werten multipliziert:

- $f_{\text{top}} = 0,945$
- $f_{\text{middle}} = 0,99$
- $f_{\text{bottom}} = 0,89$

Anschließende Regressionstests lieferten für den AM0 Test gleichzeitig bessere Ergebnisse, welche die bereits gezeigten sind. Das Ergebnis des AM1,5 Tests ist nun in Abbildung 3.15 zu sehen.



	Real	Simuliert
U_{oc} [V]	2,988	2,972
I_{sc} [A]	0,2465	0,2495
U_{mpp} [V]	2,634	2,598
I_{mpp} [A]	0,2339	0,2381
P_{max} [W]	0,616	0,6186
η [%]	30,79	30,91

Abbildung 3.15: Vergleich der realen und simulierten Zellkennwerte bei AM1,5 Bedingungen

Ein Vergleich beider Kennlinien zeigt, dass sie einen sehr ähnlichen Verlauf besitzen, doch im Gegensatz zum perfekten AM0 Ergebnis sind hier leichte Abweichungen zu sehen. Dies mag dadurch begründet werden, dass die Kennwerte zu beiden Spektren aus unterschiedlichen Chargen kommen und das die AM1,5 Messwerte von nur zwei verschiedenen Solarzellen gemittelt wurden. Ein Vergleich der maximalen Leistungsausgabe und des Wirkungsgrads zeigt hingegen, dass die Ergebnisse sehr nah aneinander liegen und unter 1 % Abweichung besitzen.

Im Rahmen dieser Validierung des Solarzellenmodells konnte gezeigt werden, dass es nur sehr geringe Unterschiede zwischen den realen und simulierten Messwerten gibt. Damit eignet sich dieses Modell sehr gut dafür, als Basis für den Aufbau eines Solargeneratormodells zu dienen.

3.5 Leistungsflusssimulation in Gleichstromnetzen

Nachdem ein Solarzellenmodell erstellt wurde, wurden Überlegungen angestellt, wie die Einflüsse der Verschaltung am besten abgebildet werden können. Im einfachsten Fall wird dazu aus der modellierten Kennlinie der Solarzelle der maximale Leistungswert entnommen und zu den Leistungen der anderen modellierten Zellen hinzuaddiert. Dabei stellt sich die Frage, wie die Einflüsse der Blockingdioden mit eingebunden werden können. Da sich Strom- und Spannungswerte der Solarzellen ändern, kann kein fester Wert vom String abgezogen werden. Passender ist da die Subtraktion der von der Diode benötigten Spannung von der Gesamtspannung. Dieser Grund führt dazu, die einzelnen Ströme und Spannungen an allen Elementen des Netzes zu modellieren. Entsprechend wäre es nun möglich immer U_{mpp} und I_{mpp} an den einzelnen Zellen ausgeben zu lassen.

Letztlich arbeiten die einzelnen Solarzellen nicht von alleine im MPP, sondern sie werden vom MPPT in ihrem Betriebspunkt gehalten. Dieser kann außerdem nur den MPP des gesamten Stacks finden und einstellen und nicht für jede Zelle individuell. Dies ist ein Grund, die elektrische Verschaltung als eine Schaltungssimulation abzubilden. Ein anderer Grund ist die Tatsache, dass es nicht immer der Fall sein muss, dass der Solargenerator seine volle Leistung abgeben kann. Dieses Verhalten soll im Folgenden durch eine Leistungsflusssimulation bzw. eine Schaltungssimulation abgebildet werden.

Eine Möglichkeit zur Umsetzung eines solchen Netzes wird durch die *Simscape* Erweiterung von MathWorks ermöglicht [32]. Allerdings stellt dies eine proprietäre Lösung dar, da jeder Nutzer eine Lizenz erwerben müsste. Eine Machbarkeit der Integration des erstellten Solarzellenmodells ist ebenso fragwürdig. Eine allgemeine Möglichkeit zur Modellierung dynamischer Leistungsflüsse stellt die Methode der Bondgraphen dar [33] [20]. Sie eignet sich nicht nur für den Einsatz in der elektrischen Domäne, sondern auch für die Mechanik, Hydraulik, Thermodynamik und weitere. Eine Transformation der Leistungsflüsse zwischen den Domänen ist problemlos möglich und könnte sich für ein Mehrdomänensystem wie HAP gut eignen. Die einzige gefundene Bibliothek, die Bondgraphen in Simulink modellieren lässt, ist die in Geitners Aufsatz [33] beschriebene Eigenentwicklung [34]. Die Bedienung der Bibliotheksblöcke und der Methode hat sich als nicht intuitiv herausgestellt, was die Nutzung für andere Mitarbeiter stark erschweren würde. Außerdem bestehen Zweifel an der Skalierbarkeit der Funktionsblöcke für den gesamten Solargenerator. Aus diesen Gründen wurde entschieden eine eigene Leistungsflussbibliothek in Simulink zu implementieren.

3.5.1 Konzept der Leistungsflusssimulation

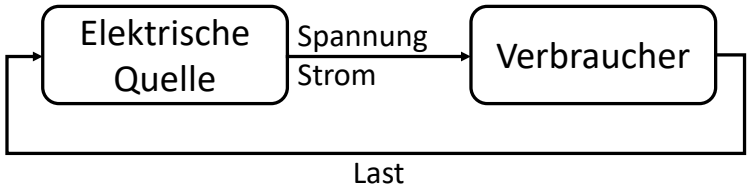


Abbildung 3.16: Prinzip der Leistungsflusssimulation

Das Konzept zur Leistungsflusssimulation dazu sieht vor, elektrische Quellen und Verbraucher als eigenständige Blöcke zu implementieren. Ein Quellblock soll die Funktion besitzen, seine Spannung U und seinen Strom I auszugeben. Aus diesen beiden Größen lässt sich über

$$P = U \cdot I \tag{3.6}$$

gleichzeitig die abgegebene Leistung P der Quelle bestimmen [2]. Der Verbraucher soll einen festen Widerstandswert besitzen oder einen variablen, der sich durch seinen Leistungsbedarf einstellen soll. Die Last des Verbrauchers hängt dabei von der anliegenden Spannung ab:

$$R = U^2/P \tag{3.7}$$

Durch eine Rückführung des Widerstandswerts an die Quelle ist es für die Quelle möglich, auf die eingestellte Last zu reagieren. Indem für die Quelle eine Kennlinie hinterlegt wird, ist es für sie möglich, mithilfe des Widerstands eine Arbeitspunktbestimmung durchzuführen und so die Spannung und den Strom für die nächste Iteration zu berechnen. Durch diesen iterativen Prozess ist es möglich, Anfangswertprobleme zu lösen und auf die Dynamik des System zu reagieren und dessen Verlauf abzubilden. Die Anfangswerte für den Quellblock sollen sich immer aus dem Leerlauffall ergeben. Dies ist dadurch begründet, dass die Spannung die Ursache darstellt und der Strom nur die Wirkung dessen ist.

Für den vollständigen Aufbau eines Netzes ist es außerdem nötig, Blöcke für Verschaltungen zu implementieren. Zum einen sind die beiden Fälle von Parallel- und Reihenschaltung zu unterscheiden, aber auch, ob es sich bei der Verschaltung um zwei Verbraucher oder zwei Quellen handelt. Dies ergibt insgesamt vier verschiedene Knotenblöcke, die zu implementieren sind. Am Ende erlauben die Knotenblöcke das Netz, auf den eingangs beschriebenen Fall von einer Quelle und einem Verbraucher, zu reduzieren.

3.5.2 Implementierung der Funktionsblöcke

Während der Implementierung der Leistungsflussblöcke werden die elektrotechnischen Grundlagen der Abschnitte 2.1.1 bis 2.1.3 genutzt und auf deren Formeln zurückgegriffen. Es soll zunächst mit dem Block für den Widerstand bzw. den Verbraucher begonnen werden.

Der erstellte **Verbraucherblock** besitzt als Eingangsgrößen die anliegende Spannung U und den eingehenden Strom I. Als Ausgangsgröße besitzt der Block einen Widerstand. Dieser kann entweder fest sein oder variiert nach der Formel 3.7, um als Verbraucher eine feste Leistung abzunehmen. Entsprechendes Verhalten kann über eine Maske eingestellt werden.

Als nächstes wurde der **Quellenblock** erstellt, wobei dieser als MATLAB-Function Block implementiert wurde. Die Funktion nimmt als Eingangsgrößen die Kennlinie und den Angeschlossenen Widerstand auf. Die Kennlinie wird dabei als ein 2-dimensionales Array übergeben, das in der ersten Zeile die Spannungswerte enthält und in der zweiten Zeile die zugehörigen Werte für den Strom. Die so diskretisierte Kennlinie wird im relevanten Bereich linearisiert und es wird mit der Geraden des Widerstands eine Schnittpunktberechnung durchgeführt. Der gefundene Schnittpunkt entspricht dem Arbeitspunkt der Quelle und es können die gefundenen Werte für die Spannung und den Strom ausgegeben werden. Eine Veranschaulichung der Arbeitspunktbestimmung zeigt Abbildung 3.17. Der zugehörige Quellcode A.5 ist im Anhang aufgeführt.

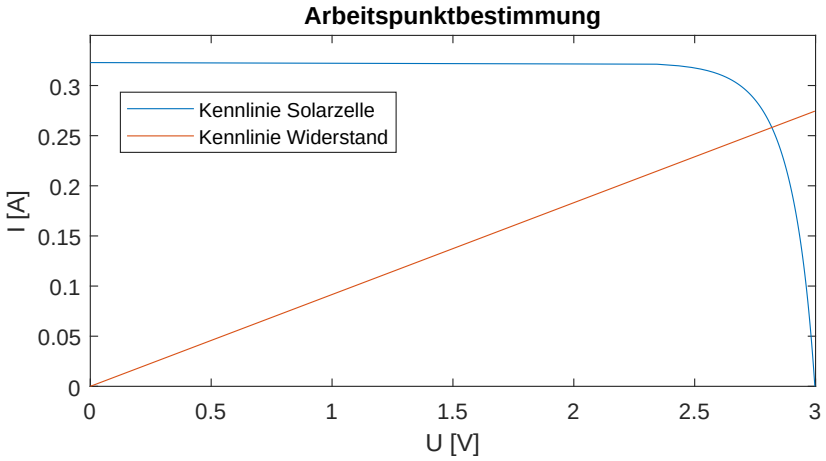


Abbildung 3.17: Arbeitspunktbestimmung des Quellenblocks

Damit die Quell- und Verbraucherblöcke erst ein Netz für Leistungsflüsse aufbauen können, sind die Knotenblöcke essentiell. Es wurde zunächst begonnen, die Funktionen für die Verbraucher aufzubauen. Für die **Reihenschaltung von Verbrauchern** benötigt der Block

die Spannung und den Strom der Quelle, sowie die Größen der einzelnen Widerstände. Wie in Formel 2.7 gezeigt wurde, lässt sich für die Quelle der Widerstand aus der Summe der Einzelwiderstände berechnen. Da es sich um eine Reihenschaltung handelt, erhalten alle Verbraucher den gleichen Strom wie der eingehende Strom. Für die Berechnung der Spannungen an den einzelnen Lasten lässt sich die Maschenregel zusammen mit dem Ohm'schen Gesetz anwenden:

$$U_E = U_1 + U_2 + ... + U_n \tag{3.8}$$

$$U_E = R_{\text{ges}} I = R_1 I + R_2 I + ... + R_n I \tag{3.9}$$

$$\Longrightarrow \frac{U_n}{U_E} = \frac{R_n}{R_{\text{ges}}} \tag{3.10}$$

Es wird genutzt, dass das Verhältnis des Teilwiderstands zum Gesamtwiderstand gleich der Teilspannung zur Gesamtspannung ist. Damit ergibt sich für die einzelnen Teilspannungen:

$$U_n = U_E \frac{R_n}{R_{\text{ges}}} \tag{3.11}$$

Der Block zur **Parallelschaltung von Lasten** besitzt die gleichen Ein- und Ausgänge wie der zur Reihenschaltung, nur ändern sich intern die Formeln. Für die Quelle errechnet sich der Gesamtwiderstand nach der Formel 2.12. Aufgrund der Parallelschaltung liegt an allen Widerständen die gleiche Spannung an. Die jeweiligen Ströme für die Pfade lassen sich aus den einzelnen Widerständen und den gleichen Spannung über das Ohm'sche Gesetz errechnen:

$$I_n = \frac{U_E}{R_n} \tag{3.12}$$

Nachdem die Knotenblöcke für die Lasten abgeschlossen sind, müssen noch die Verschaltungen von elektrischen Quellen genauer betrachtet und umgesetzt werden. Zunächst soll die **Reihenschaltung von Quellen** aufgebaut werden. Hierbei muss der Block die Ströme und Spannungen der einzelnen Quellen aufnehmen und für den abnehmenden Verbraucher zusammenfassen. Gleichzeitig ist der Widerstand des Verbrauchers aufzunehmen und in scheinbare Teilwiderstände für die Quellen aufzuteilen, damit eine korrekte Arbeitspunktbestimmung durchgeführt werden kann. Hier sind keine Scheinwiderstände des Wechselstroms gemeint! In Abschnitt 2.1.3 wurde bereits festgestellt, dass sich bei der Reihenschaltung die einzelnen Spannungen summieren und nur der geringste Strom der Quellen fließen kann. Damit gilt

für den Ausgang:

$$U_A = U_1 + U_2 + ... + U_n \tag{3.13}$$

$$I_A = \min\{I_1, I_2, ..., I_n\} \tag{3.14}$$

Als letztes müssen noch die scheinbaren Teilwiderstände für die Quellen bestimmt werden. Als Lösung dazu lässt sich die Formel 3.10 nutzen, die die Teilspannungen in das Verhältnis zu deren Teilwiderständen setzt. Umgeformt für diesen Fall ergibt sich:

$$R_n = R \cdot \frac{U_n}{U_{\text{ges}}} \tag{3.15}$$

Zum Schluss bleibt noch, den Block für die **Parallelschaltung von Quellen** zu implementieren. Dieser besitzt die gleichen Schnittstellen wie der Block für die Reihenschaltung. Allgemein wird die Parallelschaltung in Schaltungen genutzt, um Ströme zu variieren. Im Fall von Stromquellen, um deren Ströme zu erhöhen. Sofern die Spannungen der beiden Quellen genau gleich groß sind, addieren sich die Ströme.

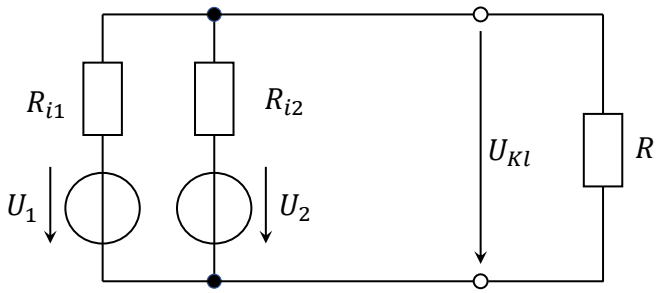


Abbildung 3.18: Parallelschaltung von 2 realen Spannungsquellen

Das Problem entsteht, sobald sich die Spannungen der realen Quellen unterscheiden. Dazu lässt sich gedanklich eine Masche um zwei parallelgeschaltete Spannungsquellen bilden, die verschiedene Spannungen besitzen (vergl. Abb. 3.18). Es zeigt sich, dass auf den ersten Blick eine Spannung fehlen muss, die die Summe beider entgegengesetzter Spannungen auf 0 bringen kann. Diese fehlende Spannung fällt über den Innenwiderständen der beiden Spannungsquellen ab. Durch den Spannungsabfall unter den Quellen ergibt sich auch, dass damit ein Strom entstehen muss. Dieser Ausgleichsstrom sorgt für eine gleiche Spannung an den Klemmen beider Quellen. Sobald eine Last an den Quellen anliegt, die dafür sorgt, dass die Klemmenspannung unter die kleinste Leerlaufspannung gefallen ist, löst sich der Ausgleichsstrom auf. Für die Betrachtung des Funktionsblockes kann damit gesagt werden,

dass der abgegebene Strom der Summe der Einzelströme entspricht:

$$I_A = I_1 + I_2 + ... + I_n \tag{3.16}$$

Da am Knotenblock keine Information über die Innenwiderstände der Quellen vorhanden ist, soll für die abgegebene Spannung die Durchschnittsspannung der anliegenden gebildet werden.

$$U_A = (U_1 + U_2 + ... + U_n)/n \tag{3.17}$$

Das korrekte Ergebnis stellt sich mit fortlaufenden Iterationen automatisch ein. Als letztes muss der scheinbare Teilwiderstand für die einzelnen Quellen bestimmt werden. Als Vermutung wurde zunächst angenommen, dass die gesuchte Formel komplementär zu Formel 3.15 sein muss und dass der scheinbare Widerstand R_n damit über den eigenen Teilstrom I_n zusammenhängen muss. Mithilfe einer Beispielrechnung konnten die Verhältnisse zwischen Strom und Widerstand bestimmt werden, wodurch sich folgende Formel ableiten ließ:

$$R_n = R \cdot \left(\frac{I_n}{I_{\text{ges}}} \right)^{-1} \tag{3.18}$$

Die Korrektheit der Formel konnte in der nachfolgenden Validierung bestätigt werden. Die Beispielrechnung ist im Anhang A.3 angeben.

Mit Ausnahme des Quellenblocks, wurden alle Funktionsblöcke mit diskreten Standardblöcken aus Simulink aufgebaut. Eine Validierung der Funktionen wurde anschließend durchgeführt.

3.5.3 Validierung der Bibliotheksfunktionen

Für eine Funktionsprüfung der einzelnen Bibliotheksblöcke wurden Testschaltungen aufgebaut, die diese Funktionen im Verbund testen. Ausgehend von den einzelnen Knotenfunktionen, wurden diese um Quellen und Verbraucher ergänzt, wodurch vier Testschaltungen entstanden. Die Abbildungen zu den durchgeführten Tests sind im Anhang angeführt.

Für die Reihenschaltung von zwei Lasten haben diese zwei verschiedene Widerstandswerte (100 Ω und 20 Ω) erhalten. Eine Reihenschaltung von zwei festen Leistungen ist nicht sinnvoll. Der Testaufbau ist in Abbildung A.4 zu sehen. Das simulierte Ergebnis entspricht dem gleichen wie bei einem angeschlossenen Widerstand mit dem Wert der summierten Teilwiderstandswerte (120 Ω) und ist damit korrekt.

Der nächste Aufbau testet die Parallelschaltung von zwei Lasten. Die Verbraucher besitzen zwei verschiedene Leistungsaufnahmen (20 W und 100 W). In Abbildung A.5 ist der Testaufbau zu sehen. Die Leistungsabgabe der Quelle entspricht der Summe der beiden Leistungen (120 W) und die einzelnen Verbraucher erhalten die jeweils verlangen Leistungen. Als Anmerkung dazu: Das gleiche Szenario findet sich in jedem Haushaltsstromnetz. Hier beträgt die Spannung für alle Verbraucher 230 V und durch Parallelschaltung können beliebig viele Lasten an dem Netz angeschlossen werden und jeder Verbraucher erhält seine benötigte Leistung.

Im dritten Testaufbau wurden zwei Quellen mit jeweils unterschiedlichem U_L und I_K in Reihe geschaltet (U_{L1}=10 V, I_{K1}=100 A, U_{L2}=9 V, I_{K2}=110 A). Die angeschlossene Last soll 400 W aufnehmen. Die Schaltung ist in Abbildung A.6 zu sehen. Das Ergebnis zeigt eine korrekt abgegebene Last mit gleichen Strömen für beide Quellen und deren Spannungen haben sich entsprechend ihrer Leistungsfähigkeit eingestellt. Im Gegensatz zu den Verschaltungen mit den Widerständen hat sich hier gezeigt, dass sich die Spannungen und Ströme nicht im ersten Iterationsschritt berechnen lassen, sondern dass diese sich erst einregeln müssen.

Die letzte Testschaltung schaltet zwei Quellen parallel und ist in Abbildung A.7 zu sehen. Hier besitzen beide Quellen die gleiche Leerlaufspannung (U_L=10 V) und unterschiedliche Kurzschlussströme (I_{K1}=100 A, I_{K2}=50 A). Die Last hat eine Leistungsaufnahme von 100 W. Die Simulation hat ergeben, dass eine korrekte Last abgegeben wird und dass beide Quellen sich auf eine gleiche Spannung eingestellt haben. Die abgegebenen Ströme ergeben sich entsprechend ihrer Leistungsfähigkeit. Hier ergibt sich der stationäre Fall für die Spannungen und Ströme erst nach mehreren Iterationen, genau wie in der Reihenschaltung.

Ein Vorausgriff auf die spätere Entwicklung dieser Arbeit soll bereits jetzt erfolgen. Es hat sich ergeben, dass für die Parallelschalung von Quellen ein Block mit variabel vielen Eingängen benötigt wird und dass die Lösung mit den diskreten Blöcken aus Simulink nicht ausreichend ist. Dazu wurde eine S-Funktion geschrieben, die dies ermöglicht. Zum Testen des Blocks wurde der gleiche Testaufbau wie der des Vorgängerblocks genutzt. Da die gleichen Formeln verwendet wurden, hat sich als Ergebnis die gleiche Ausgangssituation eingestellt, die in Abbildung A.8 zu sehen ist. Damit konnte die korrekte Funktion des S-Function Blocks bestätigt werden.

3.6 Aufbau des Solargeneratormodells

Nachdem bislang ein Solarzellenmodell als Grundlage erstellt wurde und eine Library für die elekrische Verschaltung, kann nun das Modell des Solargenerators aufgebaut werden. Dazu muss zunächst bekannt sein, aus welchen Einzelteilen sich der Solargenerator zusammensetzt. Als anschauliche Hilfe ist Abbildung 3.19 gegeben.

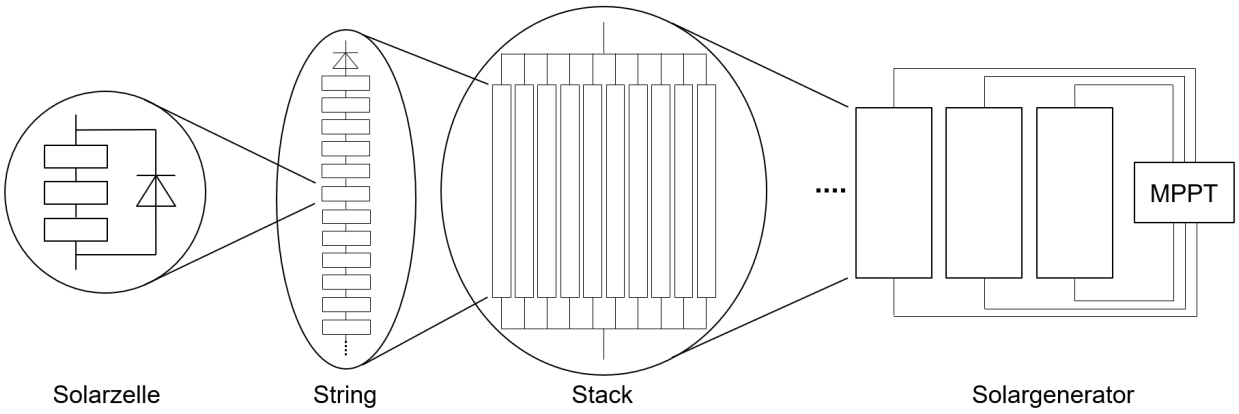


Abbildung 3.19: Aufbau und Verschaltung des Solargenerators von HAP

Wie zu Beginn gesagt wurde, wird nach einem Bottom-Up Ansatz gearbeitet. Deshalb muss die Solarzelle in die nächstgrößere Einheit überführt werden, die sich String nennt. Ein String besteht aus 24 in Reihe geschalteten Solarzellen, um auf die nötige Arbeitsspannung zu kommen und aus einer Blockingdiode zum Abschluss. Die nächstgrößere Einheit des Stacks setzt sich wiederum aus 10 parallel geschalteten Strings zusammen, um deren Ströme zu kombinieren. Zum Abschluss wird aus mehreren elektrisch unabhängigen Stacks der Solargenerator gebildet.

Im Folgenden soll zunächst damit begonnen werden, ein Solarstack-Modell aufzubauen. Da dieses eine Schnittstelle zur Verformungsberechnung besitzt, die nicht vorhanden war, wurde eine erste Version implementiert, welche die offenen Schnittstellen füllt. Der letzte Absatz behandelt die Umsetzung des Solargeneratormodells.

3.6.1 Solarstack-Modell

Wie gesagt, besteht ein Stack aus 24 in Reihe geschaltete Solarzellen, die anschließend 10-fach parallel geschaltet werden. Für eine Umsetzung des Stack-Modells wäre es nun nicht korrekt die Strom und Spannungswerte einer einzelnen Zelle entsprechend der Anzahl zu multiplizieren, da sich die elektrischen Eigenschaften der einzelnen Zellen unterscheiden. Durch die

Form des Flügelprofils, auf dem sie auflaminiert sind und die möglichen Winkel bei Verformungen, unterscheiden sich die Orientierungen der Zellen zueinander, was für unterschiedliche Einfallswinkel in der Bestrahlung sorgt. Durch diese unterschiedlichen Bestrahlungen ergeben sich verschiedene maximale Leistungen der Zellen. Eben diese Ungleichheiten in der elektrischen Verschaltung sorgen für Verluste, die mit berücksichtigt werden müssen.

Der Aufbau des Stack-Modells soll zunächst in die elektrische Verschaltung untergliedert werden und im Anschluss wird die Berechnung der Einfallswinkel von der Sonne zu den Zellen näher beschrieben. Diese Verschaltung von unabhängigen Zellinstanzblöcken mit den Leistungsflussblöcken liefert implizit das Stack-Verhalten und bildet damit den Solarstack als Modell ab.

Elektrische Verschaltung Wie aus Abbildung 3.19 ersichtlich wird, müssen für den elektrischen Aufbau eines Stacks zunächst 24 Solarzellenmodelle mithilfe der Leistungsflussknotenblöcke in Reihe geschaltet werden. Hierbei wurden zwei Tatsachen festgestellt, die eine Vereinfachung ermöglichen. Auf die Einzelheiten dieser Optimierungen soll später in Abschnitt 4.4 eingegangen werden. Für die jetzige Beschreibung reicht es aus zu sagen, dass nur eine Solarzelle modelliert wird, die in ihrer Spannung mit dem Faktor 24 multipliziert wird. Dies ist durch die Reihenschaltung begründet, die die Spannungen summiert und die Ströme auf dem gleichen Niveau hält. Die neue Kennlinie wird einem Quellenblock aus der Leistungsflussbibliothek zugeführt.

Nachdem die prinzipielle Kennlinie des Strings bereitgestellt wird und auch eine Arbeitspunktberechnung durchgeführt werden kann, folgt die Berücksichtigung der Blockingdiode. Sie ist durch eine einfache if-else-Logik umgesetzt wurden, nach der die Spannung um $0,7$ V reduziert wird, sofern sie größer als $0,7$ ist. Wenn dies nicht der Fall ist, werden Strom und Spannung am Ausgang auf 0 gesetzt. Auf eine Nutzung der genauen Kennlinie wurde an dieser Stelle verzichtet. Damit wird soweit das Verhalten eines Strings modelliert.

Für die Umsetzung des Stack-Modells müssen 10 String-Modelle parallel geschaltet werden. Eine Schwachstelle hierbei stellen die Knotenblöcke für Leistungsflussverschaltung dar. Sie besitzen in der bisherigen Umsetzung nur die Möglichkeit, zwei Elemente parallel oder in Reihe zu schalten. Entsprechend müssten neun Blöcke verwendet werden, um alle 10 Strings parallel zu schalten. Die Schwachstelle entsteht durch die darin enthaltenen delay-Funktionen, die genutzt werden müssen, um eine algebraische Schleife zu verhindern. Dies würde zu einer unnötig langen Trägheit der Netzberechnung führen.

Als Lösung wurde ein überarbeiteter Block für die Parallelschaltung von Quellen implementiert, der als S-Function in Simulink umgesetzt wurde. Nur darüber kann die Möglichkeit realisiert werden, über eine Parameterangabe in der Maske eine variable Anzahl von Eingängen zu ermöglichen. Im Kern werden weiterhin die gleichen Formeln genutzt, die für eine korrekte Aufteilung der Widerstände sorgen und die für die Summierung der Spannungen sorgen (vergleiche Quellcode A.6). Mithilfe dieser neuen Möglichkeit zur Parallelschaltung wurden 10 String-Modelle miteinander verbunden. Das Solarstack-Modell besitzt damit die Ausgabewerte von Strom und Spannung, die in Abhängigkeit zur angeschlossenen Last stehen. Damit ermöglicht sich auch der Anschluss eines MPPT-Modells, das damit seinen Suchalgorithmus realistisch testen kann. Diese Schnittstelle erfüllt gleichzeitig die Anforderung 3.

Einfallswinkel der Sonne Nachdem die elektrische Verschaltung eines Stacks modelliert wurde, muss noch die Berechnung der jeweiligen Einfallswinkel der Zellen folgen, die erst für die Ungleichheiten der einzelnen Zellen sorgt. Da sich das gesamte System in einem dreidimensionalen Raum befindet, muss hier mit Vektoren und Rotationsmatrizen gerechnet werden.

Im folgenden soll zur Veranschaulichung und besseren Übersicht eine Notation genutzt werden, die stark angelehnt an der von Kuipers ist [3, S. 78]. Sie gibt in knapper Form die Möglichkeit eine Sequenz von Rotationen zu veranschaulichen (Abbildung 3.20).

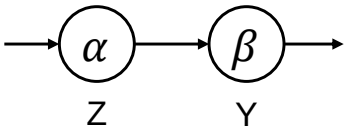


Abbildung 3.20: Notation für Rotationsketten

Ein dargestellter Kreis steht für eine durchzuführende Rotation. Der untere Buchstabe gibt an, um welche Achse des Koordinatensystems gedreht werden soll und der Winkel im inneren des Kreises gibt an, um wie viel Grad gedreht werden soll. Die hier dargestellte Folge beschreibt damit zunächst eine Drehung mit dem Winkel α um die Z-Achse und danach eine Drehung mit dem Winkel β um die Y-Achse. Die Reihenfolge erfolgt also von links nach rechts. Die mathematische Schreibweise der dargestellten Folge entspricht: $R_Y(\beta)R_Z(\alpha)$

Für die Berechnung des Einfallswinkels auf die einzelne Zelle muss der Normalenvektor der Zelle im globalen Koordinatensystem außerhalb des Flugzeugs bekannt sein. Dazu muss eine

Koordinatentransformation beginnend vom lokalen Koordinatensystem der Zelle durchgeführt werden, da die Matrizenmultiplikationen nicht kommutativ sind. Die Rotationsreihenfolge muss dabei von außen nach innen erfolgen, wie in Abbildung 3.21 gezeigt.

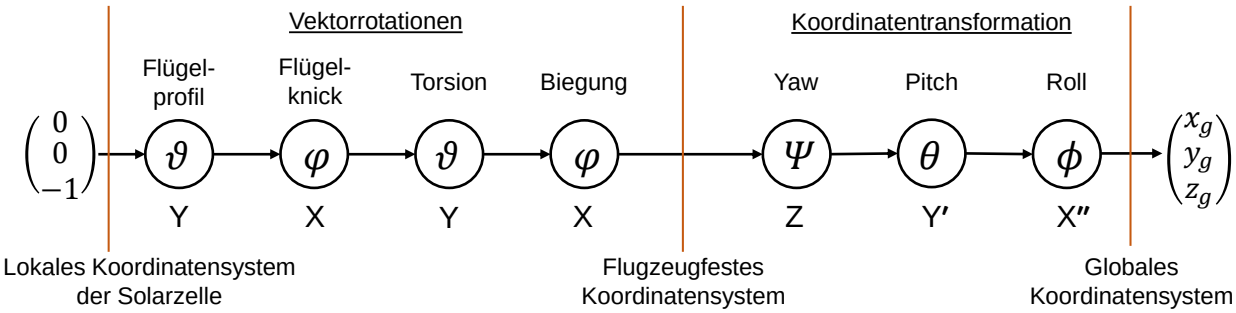


Abbildung 3.21: Rotationskette zur Berechnung des Zellnormalenvektors im globalen Koordinatensystem

Es wird für jede Zelle und durch die Vereinfachung gleichzeitig für jeden String ein Normalenvektor gebildet, der nach oben zeigt. Dieser Vektor wird als erstes bestimmend durch die Position auf dem Flügelprofil um die Y-Achse gedreht. Danach kann eine mögliche Drehung um die X-Achse folgen, sofern sich der Stack außen am Flügel und hinter dem Flügelknick befindet. Es folgen danach mögliche Rotationen aufgrund der Verformung des Flugzeugs, die sich durch Biegungen und Torsionen des Flügels äußern. Nach Durchführung der genannten Rotationen um die flugzeugfesten Koordinatenachsen, befindet sich der Normalenvektor der Zelle im flugzeugfesten Koordinatensystem.

Für eine weitere Überführung in das globale Koordinatensystem, ist es notwendig, den Normalenvektor um die Lagewinkel des Flugzeugs zu drehen und bei der Rotation die flugzeugfesten Koordinatenachsen mit zu drehen. Dadurch entstehen die Hilfsachsen X′Y′Z′ und X″Y″Z″, wobei hier implizit gilt Z = Z′, Y′ = Y″ und X″ = X‴ = x was im letzten Fall der globalen Koordinatenachse entspricht. Damit wurde der Normalenvektor der Solarzelle in das geodätische Koordinatensystem überführt. Die Grundlagen für Rotationen und Transformationen wurden in Abschnitt 2.1.4 behandelt.

Nachdem der Normalenvektor der Solarzelle bzw. des Strings bekannt ist, kann dieser zusammen mit der Positionsangabe der Sonne in einen Einfallswinkel umgerechnet werden. Dazu müssen zunächst die Kugelkoordinaten der Sonnenposition in kartesische Koordinaten umgerechnet werden mit folgenden Formeln [35, S.351]:

$$x_1 = 1 \cdot sin(\theta_S) \cdot cos(\alpha_S) \quad x_2 = 1 \cdot sin(\theta_S) \cdot sin(\alpha_S) \quad x_3 = 1 \cdot cos(\theta_S) \tag{3.19}$$

Mit dem Normalenvektor der Solarzelle und dem der Sonne, die in kartesischen Koordinaten

vorliegen kann nachfolgend mithilfe des Skalarprodukts der gesuchte Einfallswinkel berechnet werden [35, S. 361]. Der mögliche Wertebereich liegt bei 0 bis 180°. Es wird gerechnet

$$\theta = arccos\left(\frac{\vec{v_1} \cdot \vec{v_2}}{|\vec{v_1}||\vec{v_2}|}\right) \tag{3.20}$$

Mit der Nutzung der Solarzellenmodelle, den Leistungsflussblöcken zur elektrischen Verknüpfung und der jeweiligen Berechnung der Einfallswinkel auf die Solarzellen, konnte das Solarstack-Modell abschließend vervollständigt werden.

3.6.2 Verformung

Die Verformung des Leichtbauflugzeugs hat Auswirkungen auf die Orientierung der Solarzellen und damit auf den Einfallswinkel der Sonne. Wie im vorherigen Abschnitt genannt wurde, werden die Biege- und Torsionswinkel des Flugzeugs in der Einfallswinkelberechnung berücksichtigt. Dazu müssen diese allerdings bereitgestellt werden. Da zum Entwicklungszeitpunkt dieser Arbeit keine Verformungen durch das Flugmechanikmodell bereitgestellt werden konnten, wurde eine eigene erste Implementierung umgesetzt.

Die umgesetzte Verformungsberechnung nutzt Arbeitsergebnisse von der Aeroelastik, die entsprechende Verformungen mit der aktuellen Flugzeugkonfiguration für einen Geradeausflug berechnet haben. In diesem Fall hängt die Verformung nur von aktuellen Fluggeschwindigkeit ab, die als V_{EAS} angegeben wird, damit sie höhenunabhängig ist. Als Ausgangsgrößen sind die Biege- und Torsionswinkel über die gesamte Flügellänge angegeben. Die errechneten und genutzten Daten der Aeroelastik sind in Abbildung 3.22 dargestellt.

Wie in den Graphen zu sehen ist, wurden die Verformungen für vier ausgewählte Geschwindigkeitsfälle berechnet. Im oberen Graphen wird die Biegung des Flügels um die X-Achse angegeben und im unteren Graphen die Torsion des Flügels um die Y-Achse.

Das nun umgesetzte Verformungsmodul nutzt die gezeigten Daten als eine Look-Up Tabelle und interpoliert zwischen den Geschwindigkeiten. Die Eingangsgrößen sind die Positionen der Solarstacks und die aktuelle equivalent airspeed V_{EAS}. Damit können die Biege- und Torsionswinkel der Stacks an ihren Positionen ermittelt werden. Die angegebene Position stellt dabei die mittlere Position des Stacks dar, um den mittleren Winkel zwischen Anfangs- und Endposition zu erhalten.

Diese Umsetzung stellt zunächst einen Platzhalter dar, bis das Flugmechanikmodell weiterentwickelt wurde und die Verformungen während der Laufzeit berechnen kann. Allerdings konnten so schon die Schnittstellen im Stack-Modell vorab getestet werden.

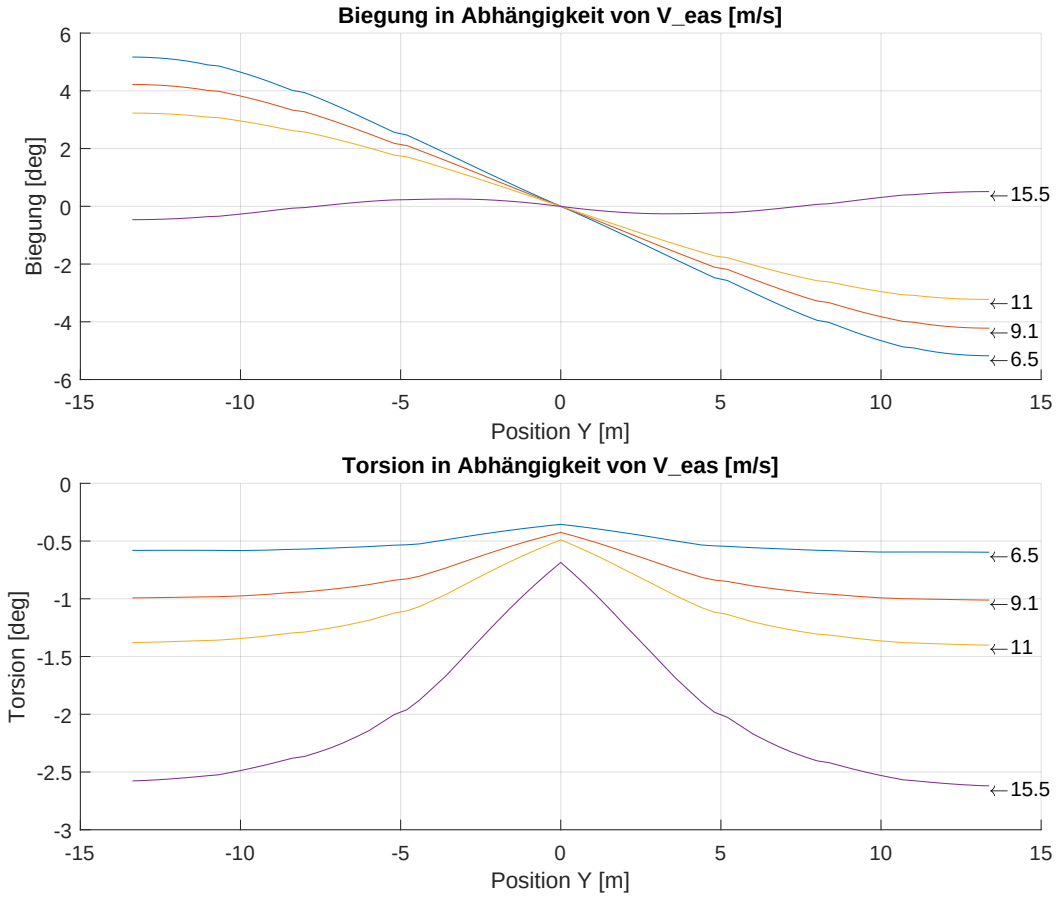


Abbildung 3.22: Verformung von HAP in Abhängigkeit von V_{EAS}

3.6.3 Solargeneratormodell

Nachdem das Solarstack-Modell fertiggestellt wurde, besteht der letzte Schritt zur Vervollständigung des Solargeneratormodells darin, mehrere der Stack-Modelle in einem Modell zu bündeln. Da die einzelnen Stacks elektrisch gesehen unabhängig voneinander sind, müssen in diesem Modell nur die Unterschiede der Positionierung und Orientierung auf dem Flügel an die einzelnen Stacks weitergegeben werden. Dabei sorgt die Positionierung im Endeffekt durch die unterschiedliche Verbiegung auch für eine neue Orientierung, die auf Ebene des Stack-Modells verrechnet wird.

Innerhalb des Solargeneratormodells wird noch eine Gruppierung der jeweils rechten und linken Flügelhälfte vorgenommen, um für eine bessere Übersicht zu sorgen. Innerhalb einer Flügelhälfte sind dann 12 Solarstack-Modelle enthalten, die die aktuelle Konfiguration des Flugzeugs wiedergeben. Jede Flügelhälfte besitzt als Ausgangsgrößen die 12 Spannungen und Ströme der einzelnen Stacks, die wiederum auf die jeweils angeschlossene Last reagieren. Der damit ermöglichte Anschluss eines MPPT-Modells wurde durch ein zusätzlich erstelltes einfaches MPPT-Modell auf Funktionalität getestet. Darauf wird gesondert im nächsten Abschnitt 3.6.4 eingegangen.

Für weitere Funktionstests und eine Gesamtbewertung des Solargeneratormodells wurde anschließend eine Integration des Modells in das Flugmechanikmodell vorgenommen. Dadurch können alle Schnittstellen getestet werden, die in Anforderung 2 gefordert wurden und die Funktion des Modells kann unter Verwendung von plausiblen Eingangsdaten betrachtet werden. Auf eine Ausführung und Bewertung wird nachfolgend in Abschnitt 3.7 eingegangen.

3.6.4 MPPT-Modell

Der Maximum Power Point Tracker (MPPT) stellt die nachfolgende Einheit nach dem Solargenerator dar. Dieser hat einerseits die Funktion eines Spannungswandler und soll auf der Ausgangsseite eine konstante Spannung bereitstellen. Doch in erster Linie soll der MPPT in der Kennlinie des angeschlossenen Solarstacks nach dem MPP suchen. Dadurch, dass der MPP in seinem Spannungspunkt variieren kann, muss die Eingangsspannung von der Ausgangsspannung des MPPT entkoppelt werden. Dazwischen findet die DC/DC-Wandlung statt.

Aus Softwaresicht ist im Wesentlichen die Funktion der maximalen Leistungsfindung interessant. Solange dies fehlt, kann am Solarstack nur ein fester Widerstand eingestellt werden, was nicht repräsentativ für die Funktion des Solargenerators ist. Deswegen wurde im Rahmen dieser Arbeit ein einfaches MPPT-Modell implementiert. Dieses besitzt als Kernstück einen Greedy-Algorithmus, der das Ziel hat die entnommene Leistung zu maximieren, indem der eigene Widerstand dazu variiert wird. Mit dem implementierten MPPT-Modell konnten die Schnittstellen des Solargenerators und die Leistungsflussblöcke getestet werden.

Abbildung 3.23 zeigt nun den Beginn eines simulierten Kreisfluges der Gesamtsimulation. Dabei ist im rechten Graphen sichtbar, wie die entnommene Leistung von dem Anfangswert aus stark ansteigt, bis sie einem Sinusverlauf folgt. Der linke Graph zeigt den dazu eingestellten Widerstand des MPPTs. Hier ist gut sichtbar, wie der Widerstandswert besonders

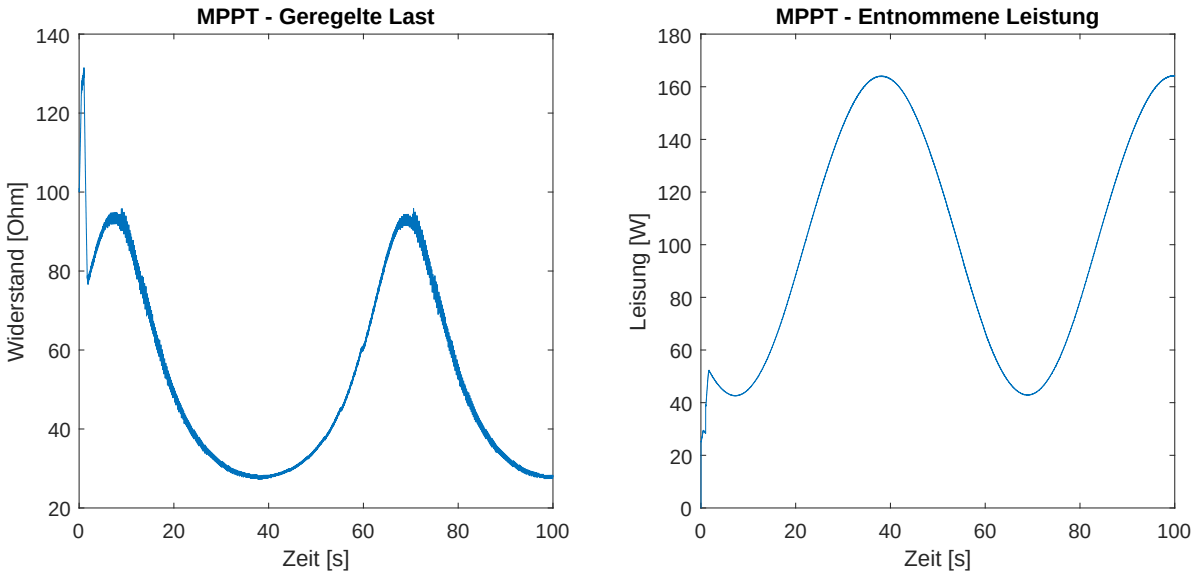


Abbildung 3.23: Lastregelung und Leistungsaufnahme des MPPT-Modells

im Maximum und in den fallenden Kurven einen dickeren Verlauf annimmt. Dies liegt an der Arbeitsweise des implementierten MPPTs.

Der MPPT-Algorithmus des einfachen Modells nutzt Fuzzylogik, um in den MPP zu regeln (vergleiche Quellcode A.7). Hierbei wird die Last solange erhöht bzw. verringert, bis die entnommene Leistung nicht weiter steigt. Ab diesem Punkt wird der Widerstandswert zurück in die andere Richtung geregelt. Sobald der MPP gefunden wurde, tritt damit ein ständiges Toggeln auf, das dadurch Änderungen im Verhalten des Solargenerators detektiert. Dieses Toggeln um den MPP herum erzeugt im Widerstandsgraphen den dickeren Verlauf der Kurve. Wie bei jedem Regler lässt sich das Verhalten auf schnelles Einschwingen oder genaues Regelverhalten optimieren. Dies geschieht in diesem Falls durch eine Veränderung der Schrittweite der Last.

Bei erneuter Betrachtung der entnommenen Leistung ist also ein sehr genaues Regelverhalten zu sehen. Die Regelung kann sich in diesem Fall gleichzeitig schnell einregeln, da die Simulationsfrequenz bei 100 Hz liegt und damit die Änderungen pro Zeitschritt klein bleiben. Der sinusförmige Verlauf ergibt sich aus dem Kurvenflug des Flugzeugs, da es sich hierbei der Sonne zu- und abwendet. Wie hiermit gezeigt werden konnte, funktioniert das implementierte MPPT-Modell und das Solargeneratormodell in Kombination.

3.7 Simulationsergebnisse und Bewertung

Nach dem Abschluss der Implementierungen folgt eine Simulation eines möglichen Missionsszenarios. Hierbei soll einerseits die Funktion des Solargenerators genauer betrachtet werden. Andererseits soll eine Bewertung über einen Vergleich erfolgen, der mit der Vorentwurfsumgebung von HAP durchgeführt werden soll. Diese nutzt eine Python-PARADISE-Toolchain und wurde zur Konfigurationsfindung des Flugzeugs eingesetzt. Mit einer Flugzeugkonfiguration sind zur Bewertung in der Python-Umgebung auch simulierte Testflüge möglich. Dabei werden jedoch nur einfache Formeln genutzt, da hierbei der Fokus auf möglichst schneller Rechenzeit lag und eine Konfiguration aus einer riesigen Lösungsmenge gefunden werden sollte.

Folgende Missionsparameter wurden für den simulierten Testflug ausgewählt:

- Dauer: 24 h
- Beginn: 21.6.20 (Sommersonnenwende)
- Ort: Kiruna in Schweden (φ=67,83°, λ=20,34°)
- Flughöhe: 12192 m (ISO Standardatmosphäre)
- Route: Kontinuierlicher Kreisflug
- Solargeneratorfläche: ca. 11,5 m^2

Es wurde versucht die Parameter beider Simulationsumgebungen möglichst gleich einzustellen. Dabei bleiben Unterschiede in der Bestrahlungsstärke der Sonne und in der Temperatur, die im Anhang in Abbildung A.9 und A.10 genauer betrachtet werden können. Die Bestrahlungsstärke variiert, da das Spektralmodell genaue Werte für die Atmosphäre benötigt, die nur abgeschätzt werden können. Die Temperatur ist in der Simulink-Umgebung hingegen konstant, da soweit kein Thermalmodell implementiert ist. Die Verläufe der einfachen und genauen Modellierung der Sonnenposition sind in Abbildung A.11 und A.12 im Anhang zu sehen. Die Unterschiede fallen sehr gering aus.

Das Simulationsergebnis der ausgewählten Mission ist in Abbildung 3.24 und 3.25 zu sehen, wobei hier die erzeugte Leistung des Solargenerators das Hauptaugenmerk darstellt.

Zunächst sollen hierbei die Rohdaten der erzeugten Leistung betrachtet werden. Allgemein lässt sich sagen, dass beide Verläufe einen ähnlichen Verlauf besitzen und sich in der gleichen Größenordnung bewegen. Über den Tagesverlauf ergibt sich eine Erhöhung der erzeugten

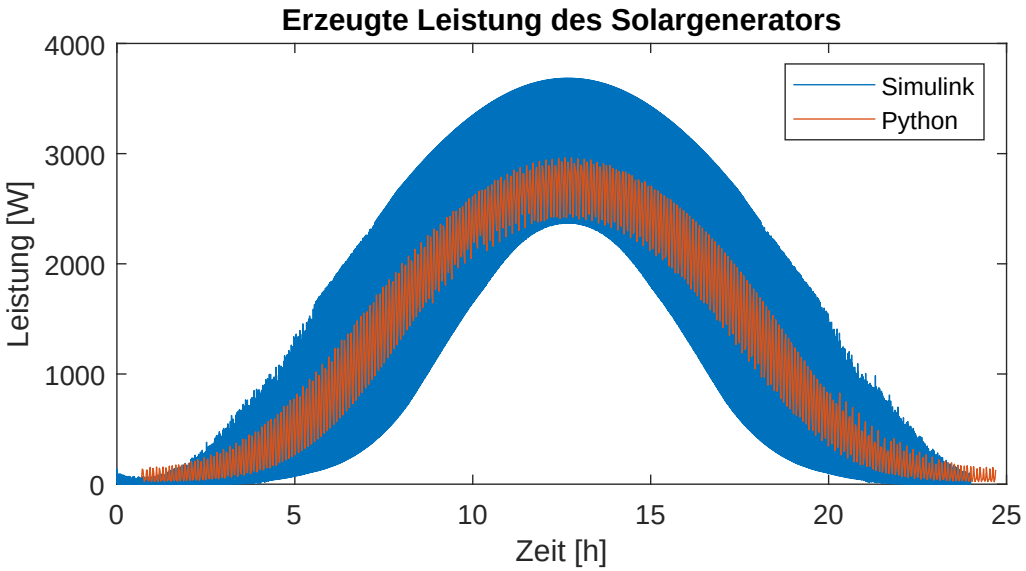


Abbildung 3.24: Modellvergleich zum Leistungsertrag der Missionssimulationen (Rohdaten)

Leistung, bis die Sonne zur Mittagszeit ihren Höchststand erreicht und damit auch die Leistung des Solargenerators. Es fällt besonders auf, dass beide Graphen eine breite farbliche Füllung aufweisen. Diese lässt sich durch den Kurvenflug erklären, bei dem sich das Flugzeug immer wieder der Sonne zu- und abwendet. Dies wurde bereits in Abbildung 3.23 im Zusammenhang mit dem MPPT festgestellt und ist dort im Verlauf deutlich sichtbar.

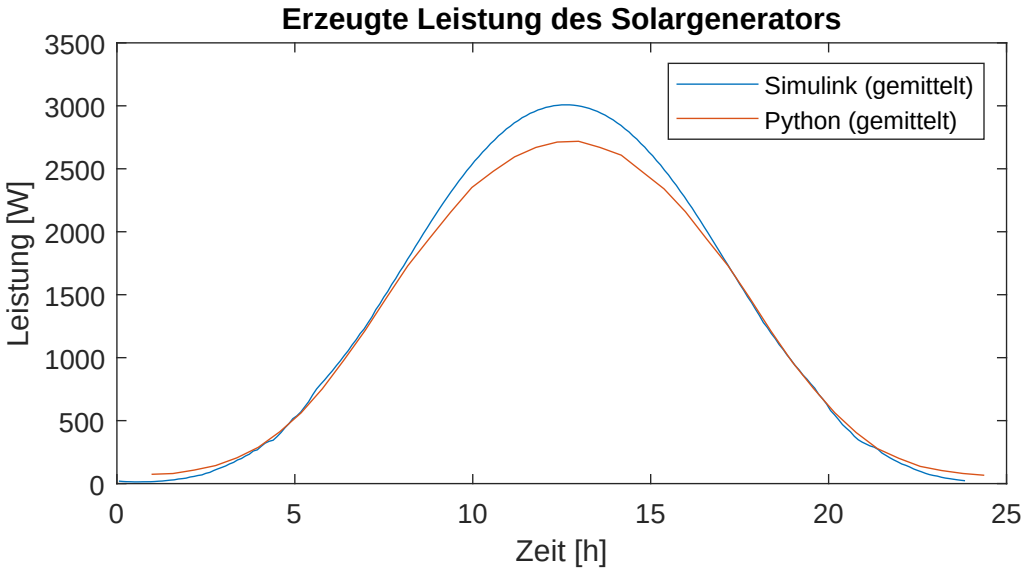


Abbildung 3.25: Modellvergleich zum Leistungsertrag der Missionssimulationen (Gemittelt)

Für einen genaueren Vergleich der erzeugten Leistungen wurde eine zeitliche Mittlung vorgenommen, durch die die Streuung eliminiert wurde und eine geglättete Kurve entsteht. Die

in Abbildung 3.25 zu sehenden gemittelten Graphen zeigen einen ähnlicheren Verlauf als zunächst erkennbar war. Eine weitere Annäherung der beiden Graphen liegt wahrscheinlich dann vor, wenn beide Temperaturverläufe gleich wären. So wirkt sich die höhere Temperatur in der Python-Umgebung negativ auf die Leistungserzeugung zur Mittagszeit aus.

Aufgrund dieser großen Ähnlichkeit, kann davon ausgegangen werden, dass in der komplexeren Simulink-Modellierung des Solargenerators keine schwerwiegenden Fehler vorhanden sind. Eine Aufintegration der erzeugten Leistungen ergibt für Simulink 31,4 kWh und für Python 29,9 kWh gesammelte Energie. Ein weiteres positives Ergebnis aufgrund der hohen Gleichheit ist die Bestätigung der hohen Genauigkeit der Vorentwurfsumgebung. Trotz der einfacheren Formeln und der schnelleren Rechenzeit können ähnliche Ergebnisse vorausgesagt werden. Dies kräftigt hiermit die Entscheidungen, die in der Vergangenheit aufgrund der Simulationsergebnisse getroffen wurden.

Bei einer erneuten Betrachtung der Simulationsergebnisse aus Abbildung 3.24 lässt sich feststellen, dass in der genaueren Simulink-Umgebung deutlich mehr Details abgebildet werden. In erster Linie ist dies auf die unterschiedlichen Simulationsfrequenzen zurückzuführen. Die hier gezeigte Simulink-Simulation wurde mit 10 Hz ausgeführt, wohingegen in Python nur mit 0,02 h Schritten simuliert wurde, was 1/72 Hz entspricht. Dadurch werden in Simulink die einzelnen Kreisflüge genau sichtbar.

Wird in den Zeitbereich einer Kreisbahn in den Morgenstunden herein gezoomt, dann werden bei Betrachtung eines einzelnen Stacks unter anderem die Schaltvorgänge des MPPTs und die Wirkung der Blockingdioden sichtbar. Solange kein Strom fließt hält der MPPT einen festgelegten Startwert für die Last. Sobald in dem Verlauf die Spannung des Stacks über den Schwellwert von 0,7 V steigt, fließt Strom und der MPPT beginnt zu arbeiten. Diese Vorgänge wurden zuvor außer Acht gelassen.

Da das Regelverhalten des MPPT und die Schaltvorgänge eine gewisse Verzögerungszeit besitzen wurde eine Variation der Simulationsfrequenz und dessen Einfluss betrachtet. Die Graphen der Ergebnisse sind in den Abbildungen A.15 und A.16 im Anhang zu finden. Dem bereits gezeigten Simulationsergebnis mit 10 Hz wird ein Ergebnis bei 1 Hz gegenübergestellt. Die ungefähr ersten und letzten 4 Stunden der Simulation sind beide Verläufe gleich. Dazwischen sind starke Einbrüche im Verlauf mit 1 Hz zu sehen. Diese sind durch das zu träge Schaltverhalten zu erklären.

Bei einer ungefähren Rotationsdauer von 60 Sekunden des Flugzeugs ergibt sich eine Winkeländerung von ca. 6 °/s für das Flugzeug. Da in dem Block zur Parallelschaltung noch ein Verzögerungsglied von 1 Takt ist und diese Verzögerung auch vom MPPT berücksichtigt

wird, ergibt sich damit nach 2 Takten also 2 Sekunden bei 1 Hz eine Winkeländerung von bis zu 12° im Einfallswinkel der Sonne. Damit sind starke Änderungen in den Kennwerten der Solarzellen möglich, auf die der MPPT reagieren muss. Dies verdeutlicht, das für die Zukunft eine genaue Abstimmung der Simulationsfrequenz und des Reglerverhaltens berücksichtigt werden muss.

Die Rohdaten und ihre Mittelwerte der jeweiligen Simulationsumgebungen sind im Anhang in den Abbildungen A.13, A.14 zu sehen.

4 Portierung des Gesamtmodells auf einem Echtzeitrechner

Ein Echtzeitrechner der Firma dSPACE bildet den Kern des Hardware in the Loop Teststands, an dem mehrere Hardware-Komponenten getestet werden sollen und bei dem Softwaremodule fehlende Teile des Gesamtsystems ersetzen. Das Solargeneratormodell stellt eines dieser fehlenden Module dar. Da der dSPACE-Rechner von äußeren Prozessen abhängig ist und damit echtzeitfähig sein muss, betrifft dies auch das Solargeneratormodell.

Im Folgenden wird in diesem Kapitel zunächst auf die Eigenschaften der ausführenden Hardware, dem dSPACE-Rechner, eingegangen. Anschließend wird der Prozess der Codegenerierung für den Echtzeitrechner erklärt, damit das erstellte Modell auf der Hardware ausgeführt werden kann. Nachdem das Gesamtmodell auf das dSPACE-System geladen wurde und simuliert wird, kann eine Laufzeitbetrachtung durchgeführt werden. Zum Abschluss werden die Optimierungsmaßnahmen zusammengefasst, die während des Entwicklungsprozesses stattgefunden haben.

4.1 Gegebenheiten des Echtzeitrechners

Das zur Verfügung stehende System von dSPACE ist von der neusten Produktserie die SCALEXIO genannt wird. Es ist eine sogenannte LabBox mit 19 Steckplatzeinheiten für diverse austauschbare Schnittstellenkarten. Zur Auswahl stehen Karten mit analogen und digitalen Ein- und Ausgänge sowie verschiedenste für serielle Schnittstellen zur Kommunikation. Ein Bild der im Rack verbauten LabBox ist in Abbildung 4.1 zu sehen.

Die LabBox selbst stellt nur die Schnittstellen für die Kommunikation mit dem Host Rechner und den Anschluss der diversen Karten bereit. Sie verfügt selbst über keine Rechenkapazität und kann damit keine Simulationen durchführen. Die Rechenleistung kommt entweder über

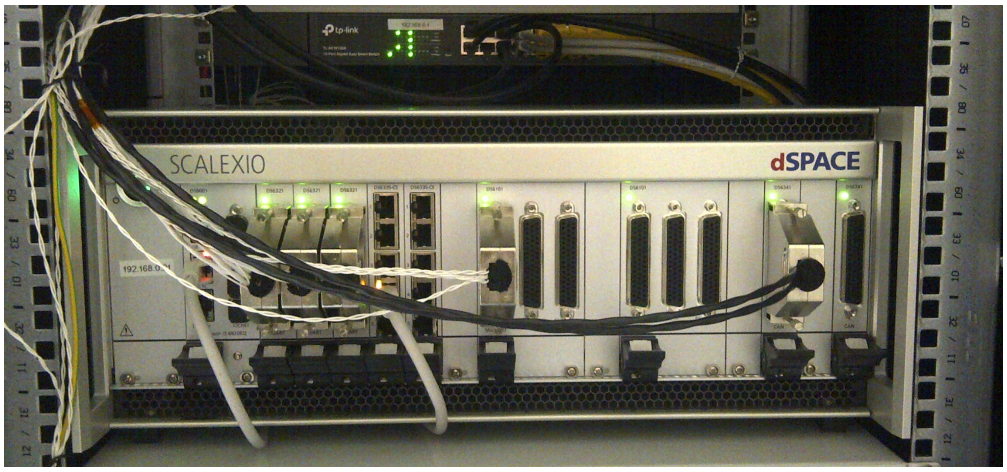


Abbildung 4.1: dSPACE Echtzeitrechner der SCALEXIO Reihe

eine separate „Processing Unit“ oder wie in diesem Fall über eine DS6001 Rechenkarte, die als Steckkarte in der LabBox eingesetzt wird.

Für den Anschluss des Host Rechners werden gewöhnliche Netzwerkkabel verwendet, die auch durch ein Subnetz geroutet werden dürfen. Die interne Buskommunikation mit den Karten wird durch das eigens entwickelte IOCNET ermöglicht, was die wesentliche Neuerung des SCALEXIO Produktserie darstellt. Sie ermöglicht auch die Erweiterung des Systems über die Gehäusegrenze hinaus durch die erwähnten Processing Units oder durch weitere LabBoxen. Die Rechenleistung der DS6001 Karte wird durch einen Intel Core i7-6820EQ Quad-Core Prozessor mit 2.8 GHz erzeugt [36].

4.2 Portierung des Modells

Der Prozess der Codegenerierung für das dSPACE-System beginnt mit einem Simulink-Modell. Es ist ebenso möglich MATLAB oder Stateflow-Modelle als grundlegende Verhaltensmodelle zu nutzen. Des Weiteren ist es möglich, als eine MATLAB unabhängige Lösung auch Functional Mock-up Units FMU zu nutzen. Das ausgewählte Modell wird im ersten Schritt in des Programm Configuration Desk von dSPACE eingelesen. Nach einer Modellanalyse werden mögliche Verletzungen angezeigt und es wird die Modellstruktur dahingehend übernommen, dass die Ein- und Ausgänge des Modells mit I/O-Funktionen des Systems verknüpft werden können. Diese Verknüpfung der Schnittstellen kann in diesem Fall übersprungen werden, da keine I/O-Funktionen genutzt werden.

Sobald das Simulink-Modell in Configuration Desk eingebunden wurde und eine Verbindung zum Zielsystem besteht, kann der Codegenerierungsprozess gestartet werden. Mit der Ausführung des *Build* Befehls wird zunächst ein Fernzugriff auf MATLAB gestartet, durch den dann eine Codegenerierung des Simulink-Modells mit dem Simulink-Coder beginnt. Dadurch werden die Block-Funktionen in C-Code übersetzt und zusammen mit extern eingebundenen Artefakten, wie vorkompilierte Libraries in einem Simulink Implementation Container (SIC) gebündelt.

Der SIC wird nach der Erstellung vom Simulink Coder in Configuration Desk eingelesen und mithilfe des eigenen Compilers in ausführbaren Maschinencode übersetzt. Nachdem dies abgeschlossen ist, beginnt die Übertragung des Gesamtsimulationsmodells auf das dSPACE-System. Mit Abschluss des Downloads wird sofort mit der Modellausführung begonnen. Da dieser Vorgang ohne Komplikationen funktionierte, gilt hiermit Anforderung 4 als erfüllt, die forderte, dass das Solargeneratormodell auf einem dSPACE-Rechner lauffähig sein muss.

Für eine Betrachtung und Anpassung der aktuellen Laufzeitparameter muss das Programm Control Desk von dSPACE verwendet werden. Dies ist für den Betrieb des laufenden Systems oder des Teststands gedacht und erlaubt die Erstellung benutzerdefinierter graphischer Oberflächen zur Überwachung und Steuerung der aktuellen Funktionen. Zu diesem Zweck wurde eine erste Benutzeroberfläche für das Gesamtsimulationsmodell von HAP gebaut, die eine Überwachung der Flug- und Systemparameter erlaubt.

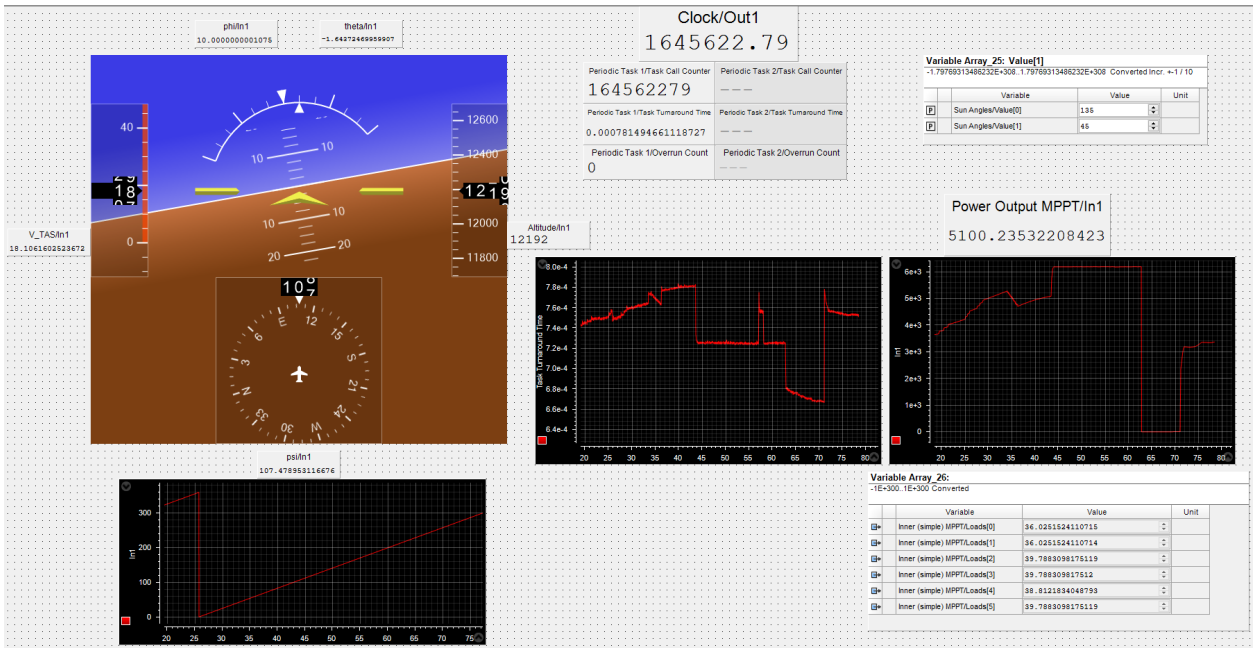


Abbildung 4.2: Exemplarische Benutzeroberfläche in Control Desk

Die erstellte Benutzeroberfläche ist in Abbildung 4.2 zu sehen und zeigt links das Basic-T zur Anzeige der Flugparameter. Im äußerst rechten Graph wird der aktuelle Leistungsoutput des Solargeneratormodells angezeigt. Ein besonderes Interesse liegt bei einer Echtzeitanwendung in der Auslastung des Rechners bzw. in der benötigten Ausführungszeit. Dazu wird im mittleren Graphen die aktuelle Taskdauer angegeben. Auf die Laufzeitbetrachtung soll im nächsten Abschnitt eingegangen werden. Für eine Anzeige der Parameter müssen diese aus der hinterlegten Struktur des Simulink-Modells aus einer Liste ausgewählt werden und dann mit einem Anzeigeelement der Wahl verknüpft werden.

4.3 Laufzeitbetrachtungen

Mit der Anforderung 5 wurde zu Beginn gefordert, dass das Solargeneratormodell mit einer Simulationsfrequenz von 100 Hz lauffähig sein soll. Dies stellt die Obergrenze der möglichen Simulationsfrequenzen dar und würde für die weitere Softwareentwicklung alle Möglichkeiten offen lassen. In diesem Abschnitt wird auf die Laufzeitbetrachtung und die Überprüfung der Erfüllung eingegangen.

Wie im vorherigen Abschnitt gesagt, ermöglicht Control Desk die Ausgabe der aktuellen Taskdauer der Simulation. Bei einer geforderten Frequenz von 100 Hz ergibt sich eine maximale Taskdauer von 10 ms, die nicht überschritten werden darf. Zunächst wurde die Laufzeit des alleinigen Flugmechanikmodells als Referenz betrachtet. Die Taskdauer der Flugmechanik liegt sehr stabil bei ca. 60 μs bzw. 0,06 ms. Damit liegt die Auslastung des von einem Kern des dSPACE-Systems bei ca. 0,6%.

Als erste Annäherung wurde ein alleinstehendes Solargeneratormodell mit insgesamt 8 Stacks ausgeführt statt der vollen 24. Die Taskdauer lag hier bei weniger als 340 μs. Dabei enthielt dieses Modell aus Analysegründen noch Hunderte Display Elemente, die eine Parameterbetrachtung in Simulink zur Laufzeit ermöglichen. Eine Entfernung dieser Elemente hat für Simulink eine dramatische Verbesserung der Performance bewirkt. Die Ausführungszeit auf dem dSPACE System hat sich nach Entfernung dieser nur marginal verringert.

In einer nächsten Betrachtung wurde das Gesamtmodell ausgeführt. Dieses besteht aus dem Flugmechanikmodell, dem vereinfachten Solargeneratormodell und allen weiteren Modellen, die in dieser Arbeit entwickelt wurden. Dabei wurde eine Ausführung der Modelle in separaten Tasks betrachtet, wobei das Flugmechanikmodell den Task 1 mit 100 Hz bekam und die anderen Modelle den Task 2 mit 10 Hz. Da der Solargenerator und Teile der anderen

Modelle Schnittstellen zum Flugmechanikmodell besitzen, muss ein Downsampling der Verbindungen stattfinden. Dies wurde durch „Rate Transition“ Elemente umgesetzt und durch einen „Function Call“ Block, der die langsameren Modelle periodisch aufruft. Die Taskdauer von Task 1 liegt bei ca. 55 μs und die von Task 2 bei ca. 410 μs. Entsprechend der Zahlen müsste in Task 2 das Gesamtmodell ausgeführt werden und es fand in der Kompilierung keine Aufteilung des Modells in 2 Prozesse statt, die dann auf 2 Kernen ausgeführt werden. Stattdessen müsste das Modell auf nur einem Kern rechnen.

Nach diesem Ausblick zur Task-Aufteilung mit unterschiedlichen Frequenzen wurde das Gesamtmodell mit vollständigem Solargenerator (24 Stacks) und bei einheitlichen 100 Hz ausgeführt. Die Taskdauer liegt bei maximal 830 μs. Damit sind die geforderten 100 Hz ohne zusätzliche Maßnahmen erfüllt. Während aller Laufzeitbetrachtungen mit enthaltenem Solargenerator konnte beobachtet werden, dass die Taskdauer in ihrer Größe mit dem Leistungsoutput des Solargenerators korreliert. Das bedeutet mit größerem Leistungsoutput ist auch die Taskdauer gestiegen.

Dieses Verhalten ist auf eine Codestelle im Solarzellenmodell zurückzuführen, bei der die Leerlaufspannung im Graph gesucht wird. Dadurch steigt die Anzahl an Schleifendurchläufen mit höher liegender Kennlinie und damit auch mit der Gesamtleistung der einzelnen Zelle. Mit der aktuellen Umsetzung ist die Taskdauer entsprechend dann am höchsten, wenn die Solarzellen voll bestrahlt werden. Wie gesagt, konnten unter verschiedensten Parametereinstellungen maximal 830 μs beobachtet werden. Die Auswirkungen der genannten Schleifendurchläufe können dabei die Taskdauer im zweistelligen Mikrosekundenbereich variieren.

4.4 Maßnahmen zur Laufzeitoptimierung

Mit dem Ziel, eine Echtzeitsimulation umzusetzen, wurden bereits während der Modellbildung Optimierungsmaßnahmen umgesetzt, durch die die Laufzeit verringert werden konnte. Die getroffenen Maßnahmen während dieser Arbeit werden in diesem Abschnitt behandelt und zusammengefasst.

Die vermutlich stärkste Optimierung betrifft die Modellierung der Solarzellen selbst. Es wird nämlich ausgenutzt, dass alle Solarzellen den gleichen Bedingungen ausgesetzt sind, bis auf den Einfallswinkel der Sonne. Bereits in der Validierung des Solarzellenmodells wurde der Strom-Winkel-Zusammenhang der Solarzelle vorgestellt. Dieser wird genutzt, um nicht jede Solarzelle durch das vollständige Modell abzubilden. In der Umsetzung des Solargeneratormodells ist das vollständige Solarzellenmodell nur ein einziges Mal vorhanden und agiert hier

als Idealzelle. Diese Solarzelle wird ideal bestrahlt, also ohne einen Einfallswinkel und repräsentiert damit die bestmögliche Solarzelle unter den gegebenen Umständen. Alle anderen Zellen sind Zellinstanzen dieser Idealzelle, indem sie deren Kennlinie erben. Nachdem in den jeweiligen Zellinstanzen eine Berechnung des Einfallswinkels der Sonne stattgefunden hat, wird die Idealkennlinie entsprechend des Zusammenhangs vom Einfallswinkel zum Kurzschlussstrom reduziert. Dazu ist der real gemessene Zusammenhang als Look-Up-Tabelle hinterlegt und der aktuelle prozentuale Wert wird genutzt, um die Kennlinie ausgehend vom Kurzschlussstrom vertikal zu verschieben (vergleiche Quellcode A.4). Dies entspricht dem gleichen Prinzip, das im vollständigen Solarzellenmodell genutzt wird. Durch diese fundamentale Maßnahme konnte die Komplexität deutlich reduziert werden.

Die zweite und ebenfalls sehr starke Vereinfachung wird durch die Zusammenfassung aller Zellen innerhalb eines Strings erreicht. Dies ist deshalb gültig, weil die Solarzellen des Strings sich nur darin unterscheiden könnten, wie sie orientiert sind und die Unterschiede in diesem Fall zu gering sind. Auf dem Flügel ist als einzige Ursache die Verformung der Oberfläche dazu fähig, die Ausrichtung der Zellen innerhalb eines Strings zu verändern. Da allerdings die maximalen Verdrehungen innerhalb der Ausmaße eines Strings (80 cm) zu gering sind mit unter einem Grad können diese mit dem Mittelwert gleichgesetzt werden.

Bei der Implementierung des Solarstack-Modells war es nötig, die zehn einzelnen Strings durch eine Parallelschaltung zu verbinden, um den Stack zu bauen. Mit den ursprünglich erstellten Leistungsflussknotenblöcken wären dazu 9 Blöcke nötig gewesen, da ein Block nur 2 Eingänge besitzt. Zur Reduktion der Komplexität wurde dazu für die Parallelschaltung von Quellen eine neue Implementierung auf Basis einer S-Function umgesetzt. Sie erlaubt über einer Eingabe in der Maske, die Anzahl von Eingängen beliebig einzustellen. Diese Funktionalität lässt sich nur auf unterster Ebene mit entsprechendem C Code umsetzen. Die dadurch eingesparte Laufzeit mag im Vergleich zu den anderen Maßnahmen relativ gering sein, aber der große Vorteil entsteht durch eine starke Delay Reduktion in der Leistungsflussberechnung. Jeder einzelne Knotenblock muss ein Delay Element besitzen, um für das gesamte Netz eine algebraische Schleife zu verhindern. Durch die Verwendung der S-Function konnte so die Anzahl der Delay Elemente von 9 auf 1 reduziert werden, was eine enorme Verbesserung bedeutet.

Die letzte Optimierung ist zufälliger Natur entstanden, aber dennoch sehr bedeutsam. Es handelt sich um die Rate Transition Blöcke, die zwischen das Flugmechanik- und dem Solargeneratormodell eingesetzt wurden, um eine Variation der beiden Ausführungsfrequenzen zu erlauben. Während der Implementierungsarbeiten betrug zu einem Zwischenstand die Taskdauer für das Gesamtmodell ca. 800 μs mit den Rate Transition Blöcken. Trotz dessen

war die Frequenz für beide Domänen auf einheitliche 100 Hz gesetzt. Durch das Entfernen der zusätzlichen Blöcke ergab sich keine funktionelle Änderung, aber eine Änderung der Performance. Die Taskdauer ohne Rate Transition ist auf ca. 3,5 ms angestiegen. Das entspricht einem Zuwachs von über 430 %. Die Begründung dafür ist nicht genau bekannt, soll aber in der Zukunft genauer untersucht werden.

5 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit konnte ein Simulink-Modell des Solargenerators von HAP erstellt werden, welches außerdem Echtzeitfähigkeit auf einem dSPACE-System demonstriert hat. Zur Erfüllung dieser Aufgabe wurden zunächst Anforderungen definiert und eine Softwarearchitektur daraus abgeleitet, die den Solargenerator und alle relevante Verbindungen zu anderen Modulen abbildet, woraus sich deren Schnittstellen ableiten ließen. Dabei wurden fehlende Module identifiziert, wie Modelle für die Sonne, dessen Eingangsgrößen zum Testen und Bewerten des eigentlichen Solargeneratormodells wichtig sind. Daraufhin wurde zunächst ein Sonnenpositionsmodell implementiert, das den Solar Position Algorithm umsetzt und für Simulink zugänglich macht. Als Ergänzung für die Sonne wurde das Spektralmodell SPCTRAL2 in Simulink implementiert, welches sich für ein einfaches Modell als äußerst genau erwiesen hat. Beide Modelle bilden das Verhalten der Sonne für die Simulationsumgebung hinreichend genau ab.

Für den Aufbau des Solargeneratormodells wurde ein Bottom-Up Ansatz gewählt, wonach zunächst die Solarzellen möglichst genau modelliert werden mussten. Das erstellte Solarzellenmodell nutzt das Spektrum des Lichts, den Einfallswinkel und die Temperatur als Eingangsgrößen, um eine äußerst genaue Kennlinie der Solarzelle auszugeben. Aufbauend auf dem Solarzellenmodell wurde als nächstgrößere Struktur ein Stack des Solargenerators modelliert. Hier fließen als weitere Abhängigkeiten die Orientierung der Zellen auf dem Flügel mit ein, die lokale Verformung des Flügels, die Lage des Flugzeugs und die Sonnenposition zur Einfallswinkelberechnung. Parallel wird im Modell die elektrische Verschaltung durch Reihen- und Parallelschaltung der einzelnen Zellen samt Dioden als Leistungsflusssimulation berücksichtigt. Wegen der unterschiedlichen Bestrahlung der Solarzellen entstehen Verluste, die hiermit berücksichtigt werden. Intern liegt eine Kennlinie des Stacks bereit, durch die das Verhalten auch in Abhängigkeit von der Last wiedergegeben werden kann. Die für die Verschaltung genutzten Leistungsflussblöcke wurden für diesen Zweck im Rahmen dieser Arbeit entwickelt und implementiert.

Eine anschließende Bündelung von mehreren unabhängigen Solarstacks ergibt den Solargenerator von HAP, die sich jeweils von ihrer Positionierung und Orientierung auf dem Flügel

unterscheiden. Das Solargeneratormodell wurde zusätzlich noch um ein einfaches MPPT-Modell erweitert und in das bestehende Flugmechanikmodell integriert. Dieses Gesamtmodell konnte im nächsten Schritt über eine Codegenerierung auf den dSPACE-Echtzeitrechner übertragen werden und es wurde eine Laufzeitbetrachtung durchgeführt. Diese ergab das die Taskdauer des Gesamtmodells stets unter 1 ms liegt und es konnte damit die letzte gestellte Anforderung erfüllt werden.

Aufgrund begrenzter Bearbeitungszeit konnte mit der Gesamtsimulation nur ein erster simulierter Kreisflug durchgeführt werden, der nur in wesentlichen Aspekten analysiert wurde. Dennoch wurde mit der Vorentwurfsumgebung des Projekts, welche vereinfachte Formeln nutzt, das gleiche Missionsszenario geflogen und es konnte ein Vergleich durchgeführt werden. Beide Simulationen zeigen einen ähnlichen Verlauf der Leistungserzeugung des Solargenerators, wobei das erstellte Simulink-Modell deutlich mehr Details erfasst.

Eine detailliertere Analyse, die den Mehrgewinn und die Fähigkeiten des Solargeneratormodells feststellt, wird folgen. Dabei sollen insbesondere Betrachtungen von Missionseinflüssen und deren Auswirkungen zum Energieertrag eines Solarflugzeugs angestellt werden. Hierbei ist angedacht, einen simulierten Testflug als Referenz durchzuführen und anschließend folgende Missionsparameter zu verändern, um deren Einflüsse zu erfassen: der Flugtag, die Flughöhe, den Missionsort, die Atmosphärentemperatur und die Fluggeschwindigkeit.

Eine weitere Untersuchung wird darin bestehen, in wie weit Vereinfachungen im Modell Auswirkungen auf die Leistungsberechnung haben. Umgekehrt kann damit ermittelt werden, wie hoch der Mehrgewinn dieser detaillierten Modellierung beispielsweise im Vergleich zur Vorentwurfsumgebung ist. Dazu sollen schrittweise Teile aus dem Modell entnommen werden oder vereinfacht werden, bis letztendlich nur noch eine Solarfläche mit einem festen Wirkungsgrad vorhanden ist.

Zum Schluss bleiben noch manche offene Fragen, die in Zukunft aufgefasst werden können. Zum einen konnte noch nicht geklärt werden, welche Werte für die Atmosphäre in der großen Höhe angenommen werden können und ob es eine Methode gibt diese für beliebige Orte auf der Erde anzunähern. Dann kann noch untersucht werden, in wie weit eine genauere Betrachtung der lokalen Temperaturen auf dem Solargenerator die Leistung beeinflussen würde anstatt nur eine Temperatur für alle Zellen anzunehmen. Eine sehr interessante Erweiterung des Modells würde auch die Berücksichtigung der eigenen Verschattung bei niedrigem Sonnenstand darstellen. Dabei kann auf Grundlagen von Quaschning [18] zurückgegriffen werden und es wurde bereits ein Konzept zur Umsetzung ausgedacht. Ferner in der Zukunft steht noch eine Validierung des Gesamtmodells und der genutzten Methoden durch einen Vergleich mit dem Solargenerator des real fliegenden Flugzeugs an.

Literaturverzeichnis

[1] Mertens, Konrad: *Dreidimensionale Ansicht einer Solarzelle.* https://www.lehrbuch-photovoltaik.de/auflage_5/abbildungen_5/hp_bild_4.4_dreidimensionale-ansicht-einer-solarzelle.png. – Aufgerufen am 29.12.2020

[2] Hering, Ekbert (Hrsg.) ; Bressler, Klaus (Hrsg.) ; Gutekunst, Jürgen (Hrsg.): *Elektronik für Ingenieure und Naturwissenschaftler.* 7. Springer Berlin Heidelberg, 2017. http://dx.doi.org/10.1007/978-3-662-54214-9. http://dx.doi.org/10.1007/978-3-662-54214-9

[3] Kuipers, Jack B.: *Quaternions and Rotation Seqences – A Primer with Applications to Orbits, Aerospace, and Virtual Reality.* Princeton University Press, 1999. – ISBN 978–0–691–10298–8

[4] Hecker, Peter: *Grundlagen der Flugführung.* 6. Auflage. TU BS – Institut für Flugführung, 2018

[5] Koks, Don: Using Rotations to Build Aerospace Coordinate Systems / Defence Science and Technology Organisation. DSTO Systems Sciences Laboratory Australia, August 2008 (DSTO–TN–0640). – techreport

[6] Mertens, Konrad: *Photovoltaik – Lehrbuch zu Grundlagen,Technologie und Praxis.* 4., aktualisierte Auflage. Fachbuchverlag Leipzig, 2018. – ISBN 978–3–446–44863–6

[7] Iqbal, Muhammad: *An Introduction To Solar Radiation.* Academic Press Canada, 1983. – ISBN 0–12–373750–8

[8] Wehmann, Hergo-Heinrich: *Vorlesungsskript – Solarzellen.* Oktober 2019. – TU BS – Institut für Halbleitertechnik

[9] Wagner, Andreas: *Photovoltaik Engineering.* 4. Auflage. Springer Berlin Heidelberg, 2015. http://dx.doi.org/10.1007/978-3-662-48640-5. http://dx.doi.org/10.1007/978-3-662-48640-5. – ISBN 978–3–662–48639–9

[10] *DIN 5034-2:1985-02, Tageslicht in Innenräumen;*. Beuth Verlag GmbH. – Berlin

[11] NREL: *2000 ASTM Standard Extraterrestrial Spectrum Reference E-490-00*. https://www.nrel.gov/grid/solar-resource/spectra-astm-e490.html. Version: September 2020. – National Renewable Energy Laboratory

[12] NREL: *Reference Air Mass 1.5 Spectra*. https://www.nrel.gov/grid/solar-resource/spectra-am1.5.html. Version: September 2020. – National Renewable Energy Laboratory

[13] Shah, Arvind (Hrsg.): *Solar Cells and Modules*. Springer International Publishing, 2020. http://dx.doi.org/10.1007/978-3-030-46487-5. http://dx.doi.org/10.1007/978-3-030-46487-5. – ISBN 978–3–030–46485–1

[14] Wesselak, Viktor ; Voswinckel, Sebastian: *Photovoltaik – Wie Sonne zu Strom wird*. 2. Auflage. Springer Berlin Heidelberg, 2016. http://dx.doi.org/10.1007/978-3-662-48906-2. http://dx.doi.org/10.1007/978-3-662-48906-2. – ISBN 978–3–662–48905–5

[15] Shah, Arvind (Hrsg.): *Thin-Film Silicon Solar Cells*. EPFL Press, 2010. – ISBN 978–1–4398–0810–8

[16] Stadler, Andreas: *Photonik der Solarzellen*. 2. Auflage. Springer Fachmedien Wiesbaden, 2017. http://dx.doi.org/10.1007/978-3-658-18965-5. http://dx.doi.org/10.1007/978-3-658-18965-5. – ISBN 978–3–658–18964–8

[17] Lewerenz, H.-J. ; Jungblut, H.: *Photovoltaik*. Springer Berlin Heidelberg, 1995. http://dx.doi.org/10.1007/978-3-642-79334-9. http://dx.doi.org/10.1007/978-3-642-79334-9. – ISBN 978–3–642–79334–9

[18] Quaschning, Volker: *Simulation der Abschattungsverluste bei solarelektrischen Systemen*, Technische Universität Berlin, phdthesis, September 1996

[19] Keidel, Bernhard: *Auslegung und Simulation von hochfliegenden,dauerhaft stationierbaren Solardrohnen*, Technische Universität München, phdthesis, 2000

[20] Schumacher, Walter: *Vorlesungsskript – Systemics*. 2019. – TU BS – Institut für Regelungstechnik

[21] Mobus, George E. ; Kalton, Michael C.: *Principles of Systems Science*. Springer New York, 2015. http://dx.doi.org/10.1007/978-1-4939-1920-8. http://dx.doi.org/10.1007/978-1-4939-1920-8. – ISBN 978–1–4939–1919–2

[22] Ledin, Jim: *Simulation Engineering – Build Better Embedded Systems Faster*. CMP Books, 2001. – ISBN 1–57820–080–6

[23] Zöbel, Dieter: *Echtzeitsysteme - Grundlagen der Planung*. Springer, 2008. http://dx.doi.org/10.1007/978-3-540-76396-3. http://dx.doi.org/10.1007/978-3-540-76396-3. – ISBN 978–3–540–76395–6

[24] Wörn, Heinz ; Brinkschulte, Uwe: *Echtzeitsysteme - Grundlagen, Funktionsweisen, Anwendungen*. Springer, 2005. – ISBN 978–3–540–20588–3

[25] Reda, Ibrahim ; Andreas, Afshin: Solar Position Algorithm for Solar Radiation Applications / National Renewable Energy Laboratory. 2008. – techreport. – NREL/TP-560-34302

[26] Blanc, Ph. ; Wald, L.: The SG2 algorithm for a fast and accurate computation of the position of the Sun for multi-decadal time period. In: *Solar Energy* 86 (2012), oct, Nr. 10, S. 3072–3083. http://dx.doi.org/10.1016/j.solener.2012.07.018. – DOI 10.1016/j.solener.2012.07.018

[27] Mahooti, Meysam: *NREL's Solar Position Algorithm (SPA)*. MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/59903-nrel-s-solar-position-algorithm-spa. Version: 2020

[28] Bird, Richard E. ; Riordan, Carol: Simple Solar Spectral Model for Direct and Diffuse Irradiance on Horizontal and Tilted Planes at the Earth's Surface for Cloudless Atmospheres. In: *Journal of climate and applied meteorology* 25 (1986), Januar, S. 87–97

[29] NREL: *Bird Simple Spectral Model*. https://www.nrel.gov/grid/solar-resource/spectral.html. Version: September 2020. – National Renewable Energy Laboratory

[30] MicroLink Devices, Inc.: *Shipment Summary: Laminated Solar Panels (Part - TSP-DD-2x5-A1-AA-BA0)*. Februar 2020

[31] Enders, Achim: *Basistext Elektromagnetische Felder*. 1. Auflage. TU BS – Institut für Elektromagnetische Verträglichkeit, 2011

[32] MathWorks: *Simscape – Modellieren und simulieren von physikalischen mehrdomän Systemen*. https://de.mathworks.com/products/simscape.html. – Aufgerufen am 29.12.2020

[33] Geitner, Gert-Helge: Modellbildung dynamischer Systeme mittels Leistungsfluß. In: *Tagung Verarbeitungsmaschinen und Verpackungstechnik*, Technische Universität Dresden, 2009, S. S. 343–372. – ISBN 978-3-86780-110-2

[34] Geitner, Gert-Helge: *Erweiterungsbibliothek BG V.2.1 zur graphischen Programmierung von Bondgraphen mittels Simulink*. https://eeiwzg.et.tu-dresden.de/ae2_files/ae_8_1.htm. – Aufgerufen am 29.12.2020

[35] Zeidler, Eberhard (Hrsg.): *Springer-Handbuch der Mathematik I*. Springer Fachmedien Wiesbaden, 2013. http://dx.doi.org/10.1007/978-3-658-00285-5. http://dx.doi.org/10.1007/978-3-658-00285-5. – ISBN 978–3–658–00284–8

[36] dSPACE GmbH: *SCALEXIO*. https://www.dspace.com/shared/data/pdf/2020/dSPACE-SCALEXIO_Product-information_08-2020_English.pdf. – Aufgerufen am 29.12.2020

A Anhang

A.1 Aufgabenstellung

Institut für Flugsystemtechnik

Deutsches Zentrum
DLR für Luft- und Raumfahrt

DLR e. V. Institut für Flugsystemtechnik
Lilienthalplatz 7, 38108 Braunschweig

Ihr Gesprächspartner Andreas Bierig

Telefon 0531 295- 2403
Telefax 0531 295- 2931
E-Mail Andreas.Bierig@dlr.de

30. Juni 2020

Aufgabenstellung für Masterarbeit

für Herr Daniel Ackermann

Matrikel-Nr.: 4967237 Studiengang: Elektronische Systeme

E-Mail: daniel.ackermann@tu-bs.de

Sprache der Arbeit: deutsch

Titel (deutsch):

Echtzeitsimulation eines Solargenerators für eine hochfliegende Solarplattform

Titel (englisch):

Real-time simulation of a solar generator for a high-altitude solar-driven platform

Erläuterung:

Lilienthalplatz 7
38108 Braunschweig
Telefon 0531 295-0
Internet DLR.de/ft

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Aufgabenstellung:

Im Rahmen des DLR Querschnittsprojekts HAP (High Altitude Plattform) wird ein hochfliegendes solarbetriebenes Flugzeug für langandauernde Flüge in Höhen zwischen 15,5 und 22km entwickelt. Dieses Flugzeug soll Aufgaben, die traditionell von Kleinsatelliten in niedrigen Orbits erfüllt werden, zukünftig übernehmen. Für die Integrationstests des ersten Prototyps wird ein Hardware in the Loop (HiL) Teststand aufgebaut. Mit diesem Teststand können dem Flugzeug am Boden zur Simulation eines Fluges realistische Daten übermittelt werden. Hierfür sind die Umweltbedingungen, die Flugdynamik sowie ein Teil der Systeme zu simulieren.

Für die Simulation des Solargenerators ist im Rahmen einer Masterarbeit ein echtzeitfähiges Modell des Systems zu erstellen. Die Herausforderungen einer realitätsnahen Simulation liegen in der Konzipierung und Umsetzung zur Berechnung des Energieertrags aller Solarzellen, der Berücksichtigung von elastischen Verformungen des Leichtbauflugzeugs und der Echtzeitfähigkeit des Moduls. Zu den Rahmenbedingungen eines realen Antriebstrangs gehört es außerdem die elektrische Verschaltung der Solarzellen als Stränge zu betrachten und gegebenenfalls auch die Schaltvorgänge und elektrischen Abläufe an Maximum-Power-Point-Trackern (MPPTs) mit zu simulieren. Das Simulationsmodul muss dabei direkte Schnittstellen für den Datenaustausch mit anderen Modulen bieten (etwa dem Flugmechanikmodell) und in die Gesamtsimulation integriert werden.

Die Aufgabenstellung umfasst im Einzelnen:

- Erarbeiten eines Überblicks zu Flugzeugintegrierten Solargeneratoren
- Erarbeiten der Anforderungen an das Simulationsmodell
- Entwickeln von Strategien, die Echtzeitfähigkeit mit möglichst hoher Güte der Simulation zu verknüpfen
- Umsetzung des Simulationsmodells in MATLAB/Simulink
- Integration, Erprobung auf einem dSPACE Echtzeitsimulationssystem
- Bewertung der Lösung und Aufzeigen von Erweiterungspotentialen

Die Arbeit wird im DLR am Institut für Flugsystemtechnik abgefasst. Die Bearbeitungszeit beträgt 6 Monate.

Betreuer: Andreas Bierig

Ausgabedatum: 01.07.2020

Abgabedatum: 01.01.2021

Dipl.-Ing. Andreas Bierig

A.2 Abbildungen

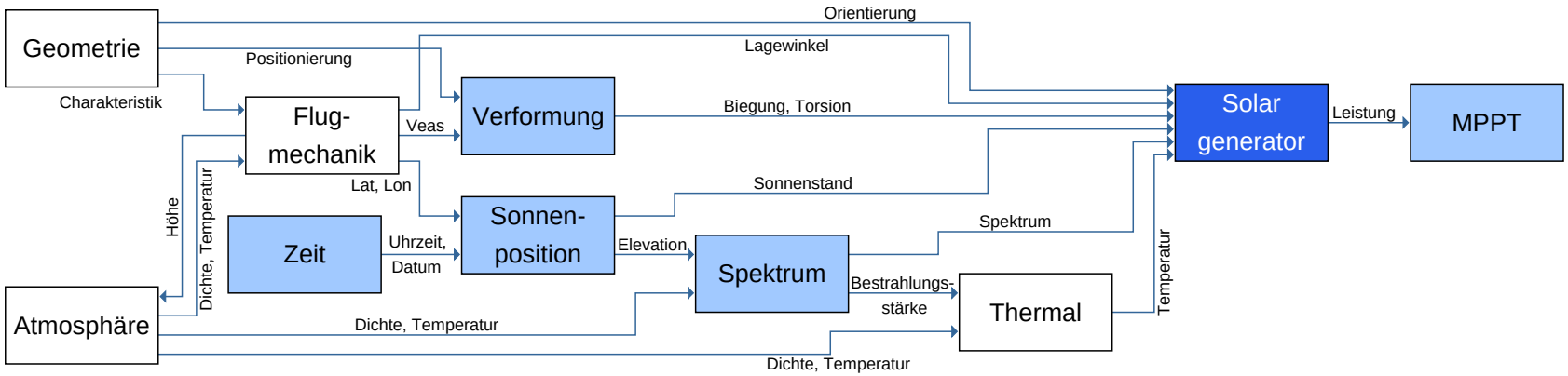


Abbildung A.1: Ausschnitt der Softwarearchitektur des Simulationssystems mit Fokus auf dem Solargenerator –erstellte Modelle in blau

Array ID	Position	Wafer	Cell	Voc [V]	FF [%]	Eff [%]	Isc [mA]	Vmax [V]	Imax [mA]	Pmax [mW]
PP39-2x5-001	A01	2-25256-1	I06	2.985	83.16	29.23	322.0	2.615	305.6	799.2
	A02	2-24986-2	I02	3.013	84.62	29.52	316.5	2.639	305.7	807.0
	A03	2-25255-3	I05	3.012	83.17	29.51	322.0	2.639	305.7	806.8
	A04	2-25520-1	I04	3.001	84.84	29.73	319.2	2.660	305.5	812.7
	A05	2-25520-4	I01	3.003	84.83	29.81	319.9	2.662	306.2	815.1
	B01	2-25255-3	I03	2.978	82.35	28.85	321.6	2.579	305.8	788.7
	B02	2-25176-2	I04	3.015	83.86	29.93	323.6	2.673	306.2	818.2
	B03	2-25253-4	I06	2.999	84.50	29.79	321.3	2.659	306.3	814.3
	B04	2-25176-1	I03	2.983	83.12	29.28	322.8	2.613	306.3	800.5
	B05	2-25253-2	I02	2.981	82.80	29.19	323.3	2.611	305.6	798.0
PP39-2x5-002	A01	2-25520-3	I02	3.000	84.76	29.90	321.5	2.659	307.4	817.4
	A02	2-25200-1	I01	2.940	82.95	28.89	323.9	2.576	306.7	789.8
	A03	2-25520-3	I03	2.998	84.60	29.77	321.0	2.657	306.3	814.0
	A04	2-25520-3	I04	3.005	85.19	29.82	318.5	2.664	306.1	815.3
	A05	2-25208-2	I05	2.990	83.50	29.77	325.9	2.651	307.0	813.8
	B01	2-25253-5	I03	2.957	83.93	29.08	320.3	2.591	306.8	795.0
	B02	2-25253-4	I05	3.019	84.29	30.07	323.1	2.676	307.2	822.1
	B03	2-25208-2	I04	2.997	82.65	29.49	325.5	2.626	307.1	806.3
	B04	2-25256-3	I05	2.996	83.45	29.48	322.3	2.625	307.0	806.0
	B05	2-25057-2	I03	2.970	82.85	28.88	320.9	2.572	307.1	789.7
PP39-2x5-003	A01	2-25342-4	I02	2.986	82.56	29.14	323.2	2.585	308.2	796.6
	A02	2-25057-3	I05	3.010	82.89	29.73	325.8	2.636	308.3	812.9
	A03	2-25176-2	I03	3.002	83.43	29.70	324.2	2.630	308.7	812.0
	A04	2-25027-4	I02	2.978	84.29	29.39	320.2	2.609	308.1	803.6
	A05	2-25253-1	I02	2.960	83.49	29.28	324.0	2.593	308.7	800.5
	B01	2-25256-5	I02	2.992	83.27	29.50	323.7	2.621	307.7	806.5
	B02	2-25057-3	I04	3.009	83.48	29.70	323.2	2.636	308.0	812.0
	B03	2-25057-3	I06	2.967	82.73	29.37	327.1	2.600	308.9	803.0
	B04	2-25182-1	I02	2.952	82.44	28.91	324.8	2.556	309.2	790.4
	B05	2-25342-2	I06	3.017	84.11	29.77	320.7	2.644	307.9	813.9
PP39-2x5-004	A01	2-25208-1	I05	3.023	82.86	29.83	325.6	2.649	307.9	815.5
	A02	2-25520-1	I05	3.024	85.78	30.28	319.2	2.680	308.9	828.0
	A03	2-25208-2	I02	3.019	82.72	29.91	327.5	2.645	309.2	817.8
	A04	2-25208-1	I01	3.004	82.41	29.73	328.3	2.632	308.8	812.8
	A05	2-25520-4	I02	2.983	84.59	29.61	320.9	2.613	309.8	809.6
	B01	2-25176-2	I02	3.017	84.08	30.15	324.9	2.674	308.2	824.2
	B02	2-25208-2	I06	3.005	83.48	30.00	327.0	2.663	308.0	820.3
	B03	2-25386-5	I01	3.053	85.68	30.49	318.6	2.707	308.0	833.6
	B04	2-25520-3	I05	3.013	84.89	29.87	319.3	2.639	309.4	816.7
	B05	2-25520-1	I01	3.009	84.86	29.85	319.6	2.636	309.6	816.1

Abbildung A.2: Solarzellenmesswerte einer Lieferung von 40 Solarzellen

Cell#1

AM	Intensity
1.5	0.1
1.5	0.1
0	0.1367
0	0.1367

Voc (V)	Jsc (mA/cm2)	Fill Factor	Eff	Isc (mA)	Vmax (V)	Imax (mA)	Pmax (mW)
2.965	12.42	84.19	31.00	248.40	2.598	238.70	620.03
2.965	12.36	84.41	30.94	247.26	2.597	238.24	618.77
2.988	15.77	82.49	28.43	315.34	2.587	300.44	777.32
2.988	15.78	82.15	28.33	315.51	2.587	299.39	774.60

Cell#2

AM	Intensity
1.5	0.1
1.5	0.1
0	0.1367
0	0.1367

Voc (V)	Jsc (mA/cm2)	Fill Factor	Eff	Isc (mA)	Vmax (V)	Imax (mA)	Pmax (mW)
3.012	12.26	82.92	30.62	245.22	2.670	229.41	612.46
3.011	12.26	82.89	30.60	245.20	2.669	229.31	612.09
3.034	15.54	82.23	28.36	310.71	2.658	291.62	775.23
3.034	15.51	82.34	28.34	310.10	2.658	291.46	774.78

Abbildung A.3: Aktuelle Zellmesswerte des Herstellers

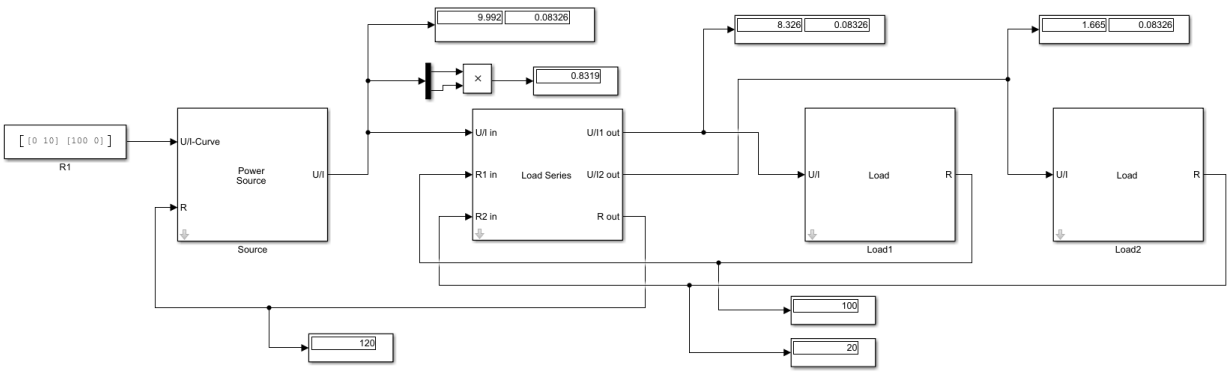


Abbildung A.4: Testaufbau zur Validierung der Reihenschaltung von Lasten

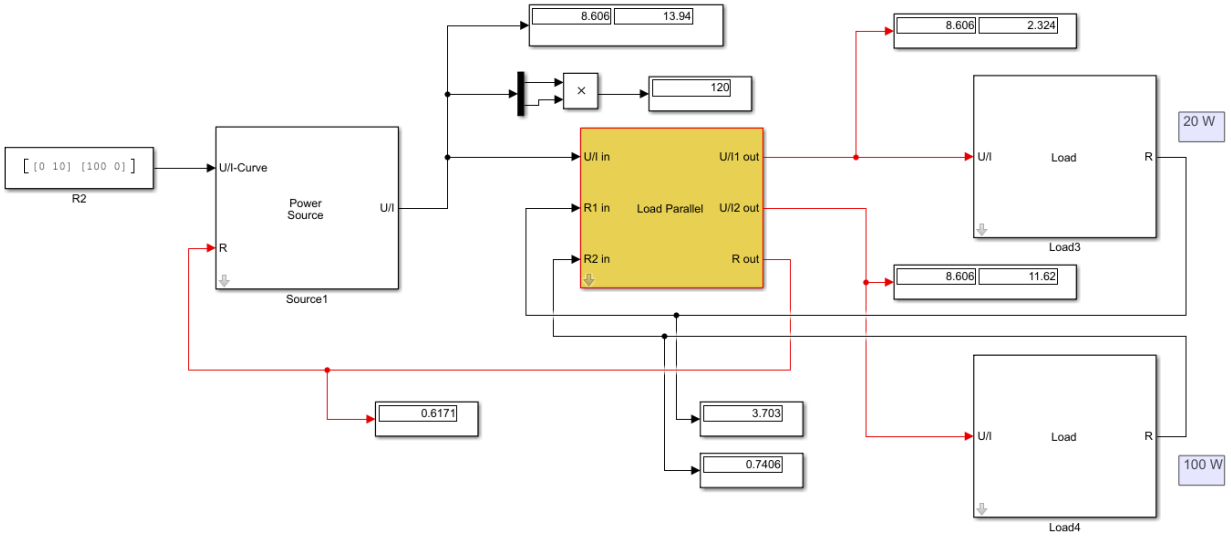


Abbildung A.5: Testaufbau zur Validierung der Parallelschaltung von Lasten

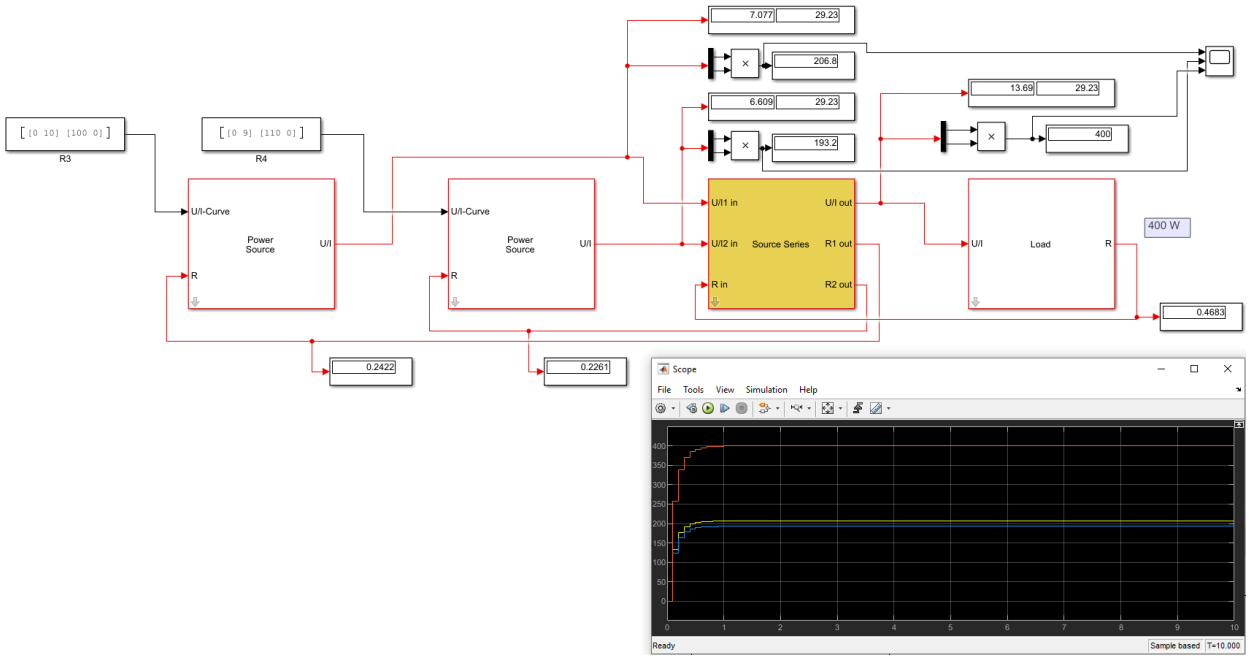


Abbildung A.6: Testaufbau zur Validierung der Reihenschaltung von Quellen

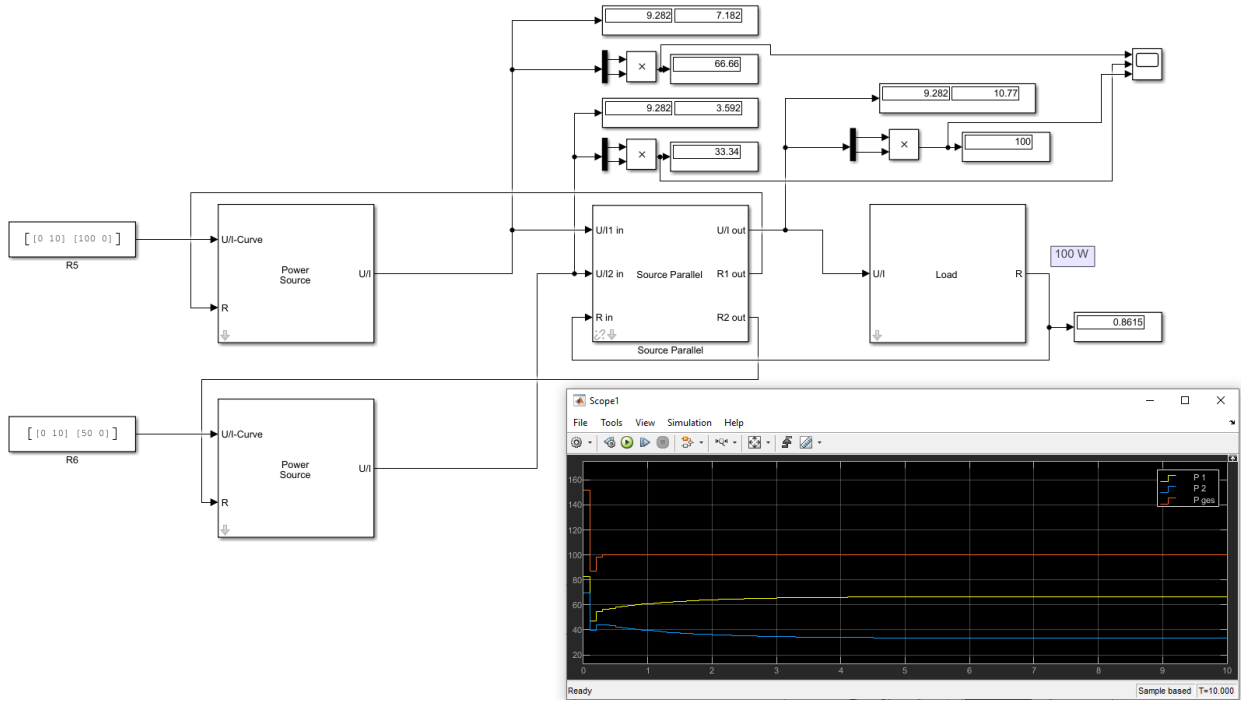


Abbildung A.7: Testaufbau zur Validierung der Parallelschaltung von Quellen

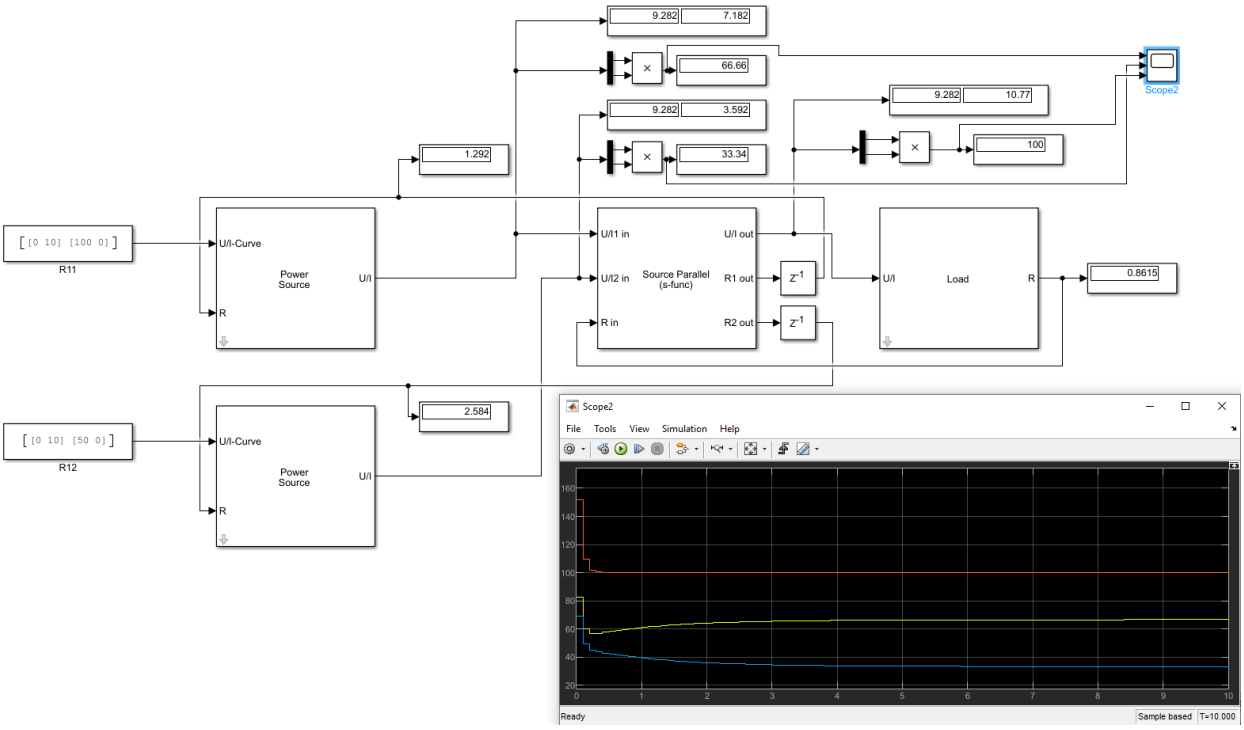


Abbildung A.8: Testaufbau zur Validierung der Parallelschaltung (S-Function) von Quellen

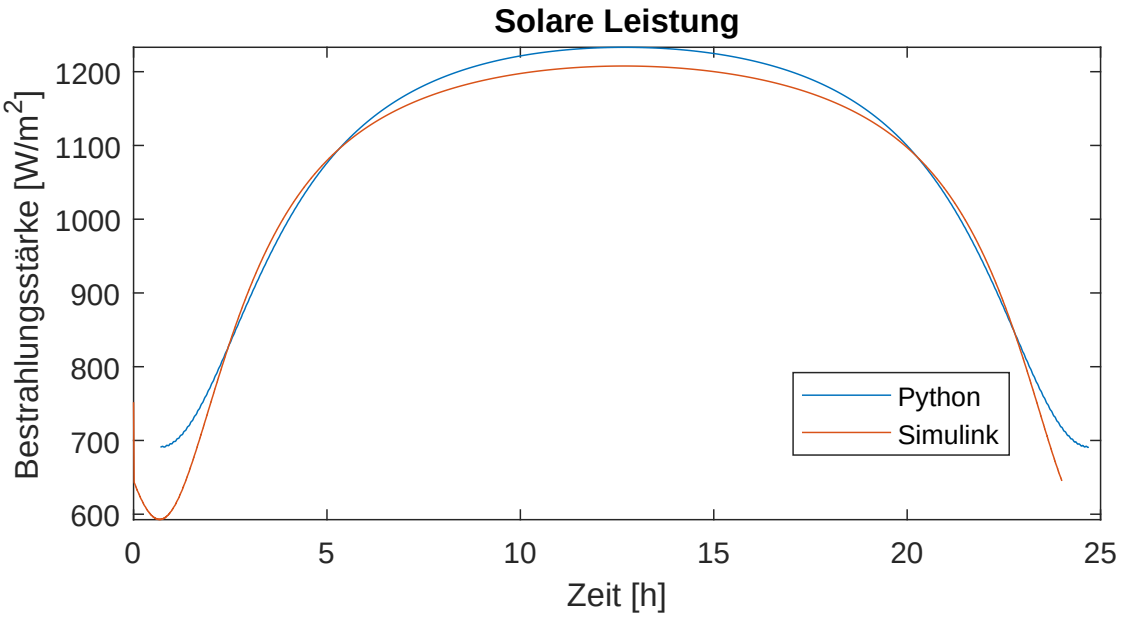


Abbildung A.9: Vergleich der Bestrahlungsstärken der Missionssimulationen

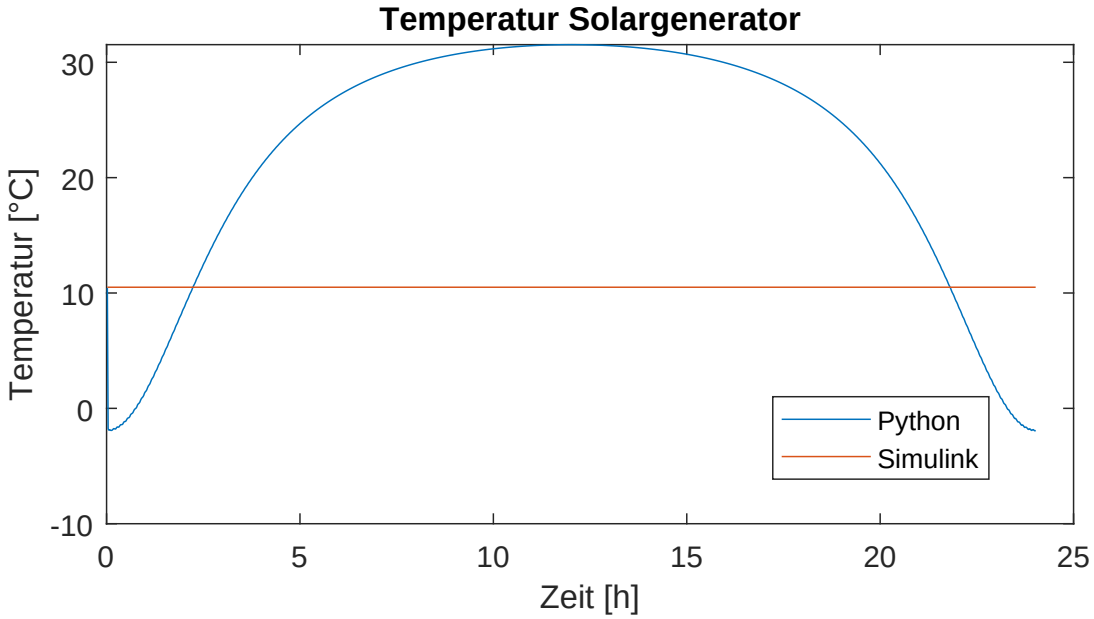


Abbildung A.10: Vergleich der Zelltemperaturen der Missionssimulationen

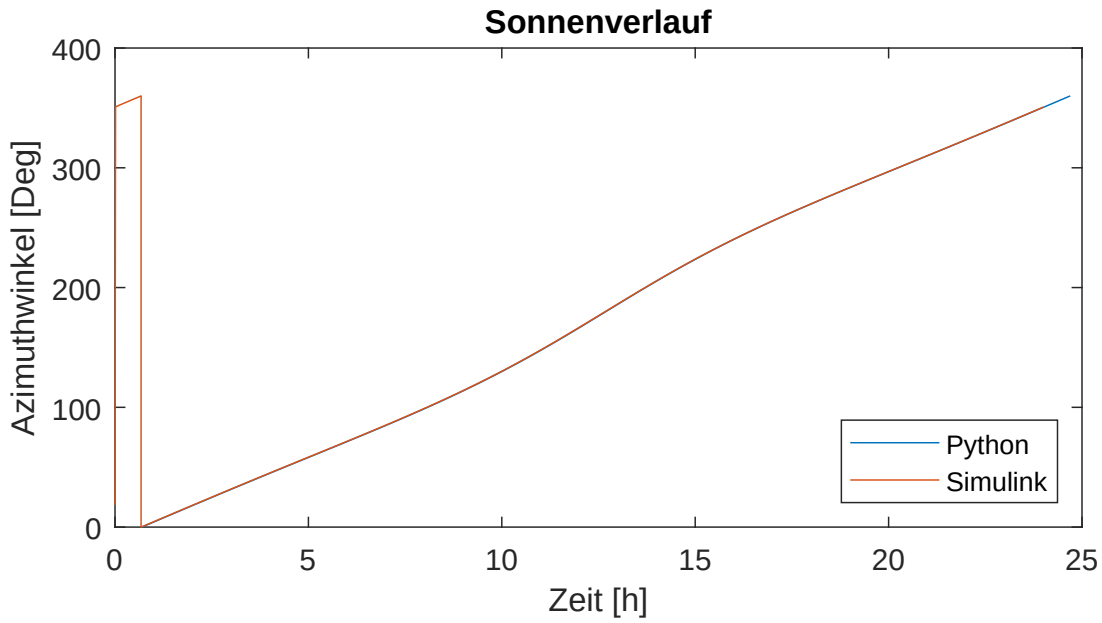


Abbildung A.11: Vergleich des Azimut über Missionssimulationenzeit

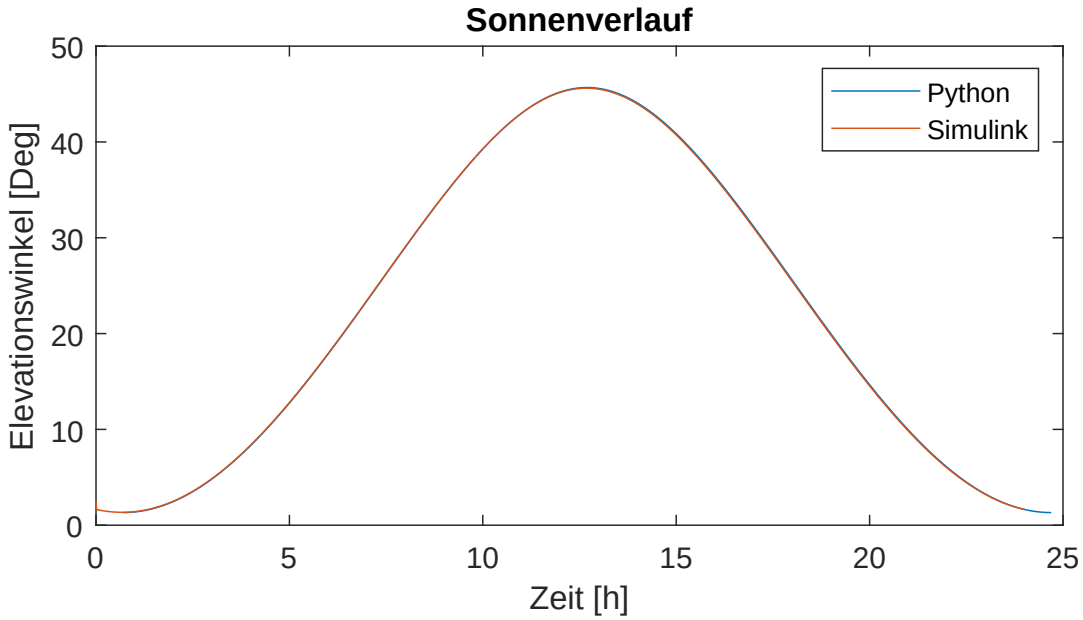


Abbildung A.12: Vergleich der Elevation über Missionssimulationenzeit

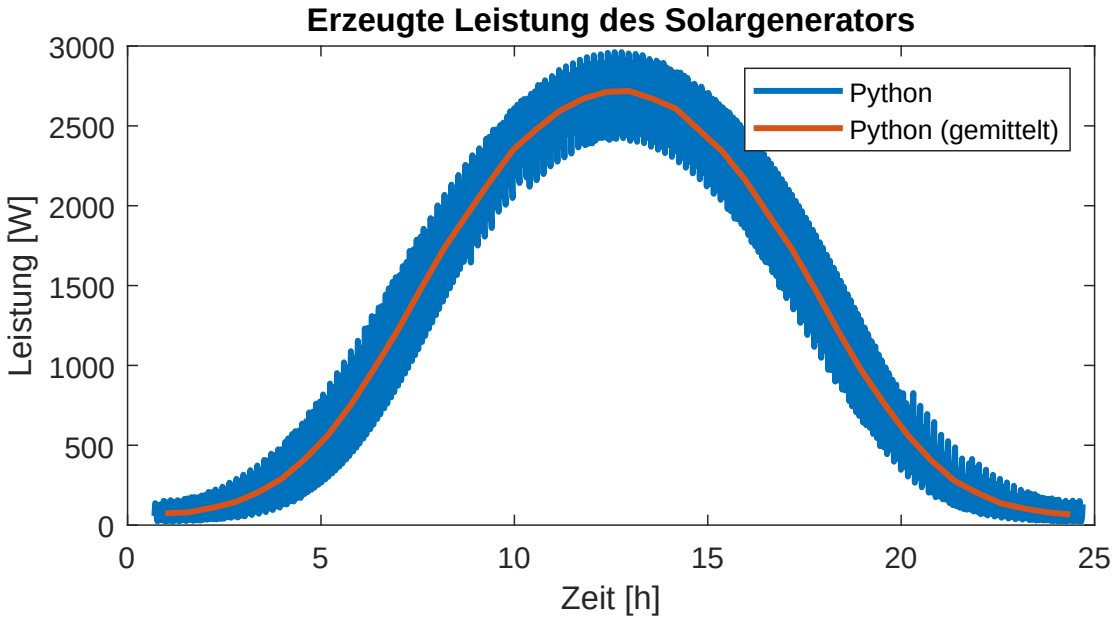


Abbildung A.13: Leistungsertrag während der Python-Missionssimulationen

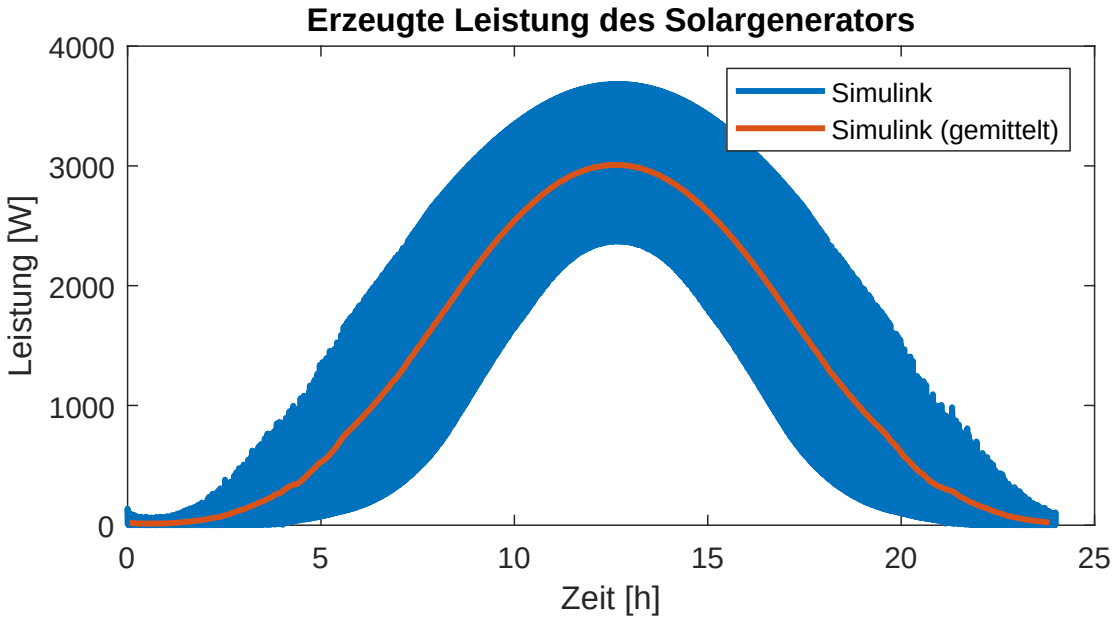


Abbildung A.14: Leistungsertrag während der Simulink-Missionssimulationen

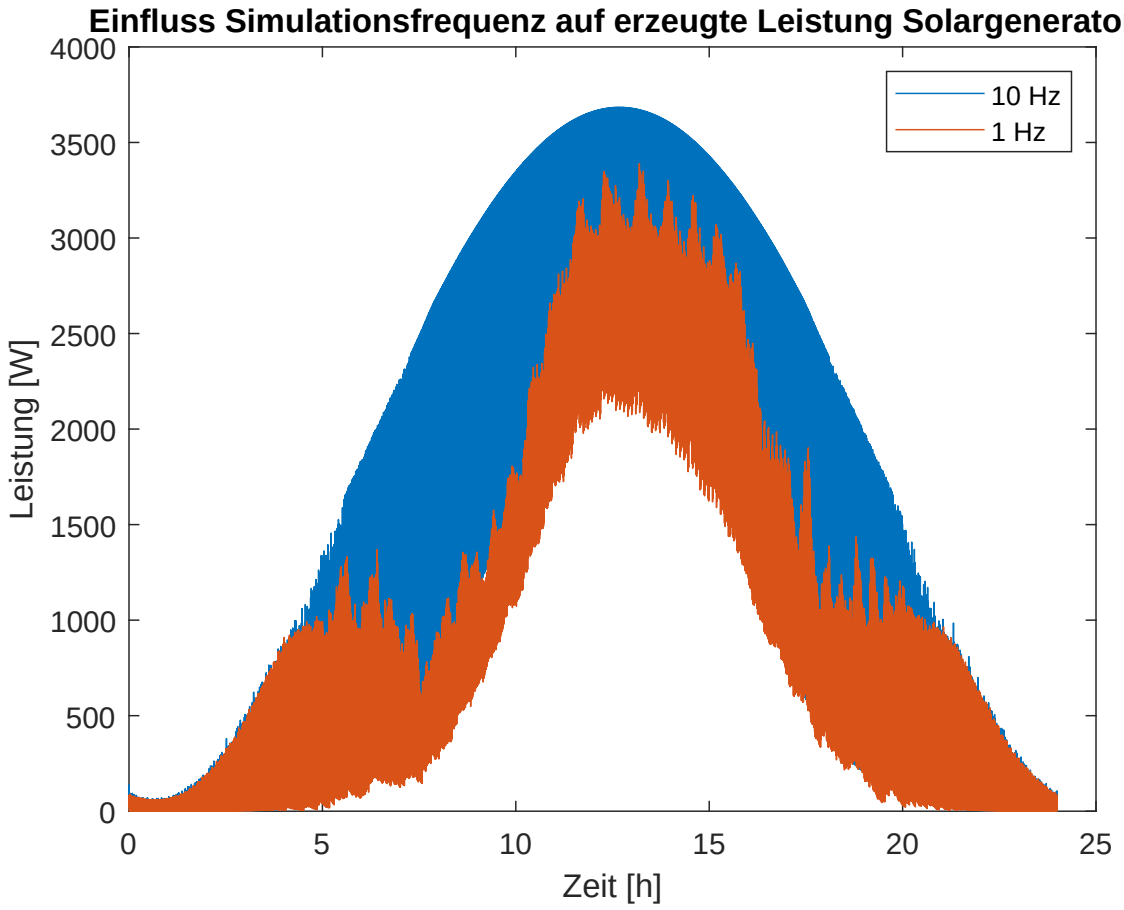


Abbildung A.15: Einfluss der Simulationsfrequenz auf den Leistungsertrag in Simulink

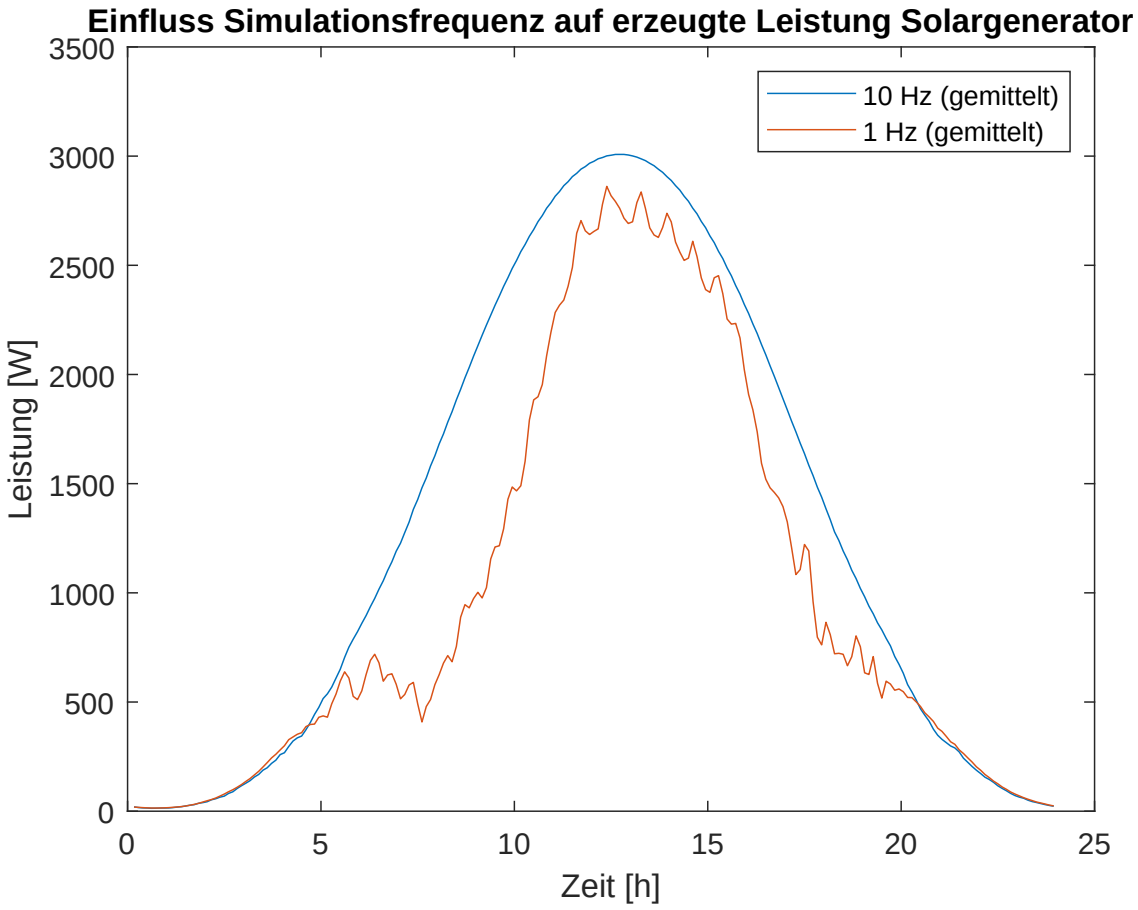


Abbildung A.16: Einfluss der Simulationsfrequenz auf den Leistungsertrag (Gemittelt)

A.3 Beispielrechnung: Parallelschaltung von 2 Spannungsquellen

Beispielrechnung zur Stromermittlung bei 2 parallelgeschalteten Spannungsquellen. Abbildung 3.18 zeigt die betrachtete Schaltung.

In der Rechnung werden folgende Formeln genutzt, die Ersatzspannungs- und Ersatzstromquelle beschreiben [2].

Klemmenspannung an einer realen Spannungsquelle

$$U_{Kl}(I) = U_0 - I \cdot R_i \tag{A.1}$$

Strom von einer realen Stromquelle

$$I(U_{Kl}) = I_K - U_{Kl} \cdot \frac{1}{R_i} \tag{A.2}$$

Gegeben: $U_{L1} = 10$ V, $I_{K1} = 100$ A, $U_{L2} = 10$ V, $I_{K1} = 50$ A, $R = 1\ \Omega$

Die Innenwiderstände der realen Spannungsquellen berechnen sich mit:

$$R_i = U_L / I_K$$

Es ergeben sich

$$R_{i1} = 10\text{ V}/100\text{ A} = 0,1\ \Omega$$

$$R_{i2} = 10\text{ V}/50\text{ A} = 0,2\ \Omega$$

Der Gesamtinnenwiderstand einer Ersatzspannungsquelle errechnet sich mit:

$$R_{i\text{ ges}} = \left(\frac{1}{R_{i1}} + \frac{1}{R_{i2}}\right)^{-1} = \left(\frac{1}{0,1\ \Omega} + \frac{1}{0,2\ \Omega}\right)^{-1} = \frac{1}{15}\ \Omega$$

Der Gesamtwiderstand der Schaltung ist:

$$R_{\text{ges}} = R + R_{i\text{ ges}} = 1\ \Omega + \frac{1}{15}\ \Omega = \frac{16}{15}\ \Omega$$

Durch Betrachtung der idealen Spannungsquelle zusammen mit dem angeschlossenen Widerstand ergibt sich der fließende Strom in der Schaltung zu:

$$I_{\text{ges}} = U_{\text{ges}}/R_{\text{ges}} = 10\ \text{V}/(16/15)\ \Omega = 9,375\ \text{A}$$

Die Klemmenspannung der Ersatzspannungsquelle ergibt sich mit der Formel A.1:

$$U_{Kl}(9,375\ \text{A}) = 10\ \text{V} - 9,375\ \text{A} \cdot \frac{1}{15}\Omega = 9,375\ \text{V}$$

Da sich bei der Parallelschaltung von 2 Spannungsquellen an beiden Quellen die gleiche Spannung einstellt, entspricht dies auch der Spannung der beiden einzelnen realen Spannungsquellen. Über eine Betrachtung der realen Spannungsquellen als realen Stromquellen lassen sich mit der Formel A.2 die Teilströme der einzelnen Quellen errechnen zu:

$$I_1(9,375\ \text{V}) = 100\ \text{A} - 9,375\ \text{V} \cdot \frac{1}{0,1} = 6,25\ \text{A}$$

$$I_2(9,375\ \text{V}) = 50\ \text{A} - 9,375\ \text{V} \cdot \frac{1}{0,2} = 3,125\ \text{A}$$

Die Verhältnisse der Teilströme zum Gesamtstrom ist:

$$6,25\ \text{A}/9,375\ \text{A} = 2/3$$

$$3,125\ \text{A}/9,375\ \text{A} = 1/3$$

Als nächstes soll ermittelt werden welcher Widerstand an den einzelnen Quellen angeschlossen sein müsste, damit sich der berechnete Strom einstellt. Es wird umgestellt:

$$\frac{U_0}{I} = R_{\text{ges}} = R_i + R$$

$$R = \frac{U_0}{I} - R_i$$

Für die einzelnen Fälle ergibt sich:

$$R_1 = \frac{10\ \text{V}}{6,25\ \text{A}} - 0,1\ \Omega = 1,5\ \Omega$$

$$R = \frac{10\ \text{V}}{3,125\ \text{A}} - 0,2\ \Omega = 3\ \Omega$$

Um von dem angeschlossenen Widerstand der Gesamtschaltung auf den scheinbaren Wider-

stand einer einzelnen Quelle zu kommen muss für die erste Quelle gerechnet werden:

$$1\ \Omega \cdot x = 1,5\ \Omega$$

$$x = 1,5\ \Omega / 1\ \Omega = 3/2$$

Dies entspricht dem reziproken Verhältnis des Teilstroms zum Gesamtstrom. Für den zweiten Fall ist der gesuchte Faktor:

$$x = 3\ \Omega / 1\ \Omega = 3 = 3/1$$

Erneut ergibt sich, dass das Verhältnis des scheinbaren Widerstands zum angeschlossenen Widerstand gleich dem des reziproken Teilstroms zum Gesamtstrom ist. Damit lässt sich folgende allgemeine Formel aufstellen:

$$\frac{R_n}{R} = \left(\frac{I_n}{I_{\text{ges}}}\right)^{-1} \tag{A.3}$$

Umgeformt ergibt sich die für die Leistungsflussberechnung fehlende Formel

$$R_n = R\left(\frac{I_n}{I_{\text{ges}}}\right)^{-1} \tag{A.4}$$

A.4 Quellcode

Quellcode A.1: Subzellen Kurzschlussstromberechnung

```
function i_SC = calc_subcell_power(wavelength, intensity, area)

h = 6.626e-34;      % [Js]
c = 299792458;      % [m/s]
eV = 1.602e-19;     % [J]

i_SC = 0;

for i = 1:(length(wavelength))

    if i == 1
        power_input = intensity(i) * (wavelength(i+1)-wavelength(i))/2;
    elseif i == length(wavelength)
        power_input = intensity(i) * (wavelength(i)-wavelength(i-1))/2;
    else
        power_input = intensity(i) * (wavelength(i+1)-wavelength(i))/2 + intensity(i) * (
            wavelength(i)-wavelength(i-1))/2;
    end

    power_input = power_input * area;

    E_photon = h * c / (wavelength(i) * 10^-6);      % [J]

    n_photon = power_input / E_photon;

    i_SC = i_SC + n_photon * eV;

end
```

Quellcode A.2: Berechnung der Solarzellen Kennlinie und Kennwerte

```
function [actual_UI_curve, x, y, u_OC, p_max, u_MPP, i_MPP]= fcn(i_SC, actual_area,
    measured_UI_curve, measured_cell_area)
size_factor = actual_area / measured_cell_area;

measured_I_SC = measured_UI_curve(2, 1);
correction_current = measured_I_SC - i_SC;

if correction_current < 0
    increase_line = true;
else
    increase_line = false;
end

new_i_line = (measured_UI_curve(2, :) - correction_current) * size_factor;
actual_UI_curve = cat(1, measured_UI_curve(1, :), new_i_line);

len = length(actual_UI_curve(1, :));
power = zeros(1, len);
u_OC = 0;
if increase_line
    missing_y = actual_UI_curve(2, len);
    dx = actual_UI_curve(1, len) - actual_UI_curve(1, len-1);
    dy = actual_UI_curve(2, len) - actual_UI_curve(2, len-1);
    factor = abs(missing_y / dy);
    extra = [actual_UI_curve(1, len) + dx*factor; actual_UI_curve(2, len) + dy*factor];
    actual_UI_curve(:, len) = extra;
end
for i = 1:len
    if actual_UI_curve(2, i) == 0
        u_OC = actual_UI_curve(1, i);
        actual_UI_curve(2, i+1:len) = zeros(1, len-i);
        power(i:len) = zeros(1, len-i+1);
        break;
    end
    if ~(i == len)
        if (actual_UI_curve(2, i) > 0) && ( actual_UI_curve(2, i+1) < 0)
            u_OC = (actual_UI_curve(1, i) + actual_UI_curve(1, i+1)) / 2.0;
            actual_UI_curve(1, i+1) = u_OC;
            actual_UI_curve(2, i+1:len) = zeros(1, len-i);
            power(i+1:len) = zeros(1, len-i);
            break;
        end
    end
    power(i) = actual_UI_curve(1, i) * actual_UI_curve(2, i);
end

p_max = max(power);
index_max = find(power == p_max);

u_MPP = actual_UI_curve(1, index_max(1));
i_MPP = actual_UI_curve(2, index_max(1));

x = transpose(actual_UI_curve(1,:));
y = transpose(actual_UI_curve(2,:));
```

Quellcode A.3: Temperaturkorrektur der Kennlinie

```
function [uicurve_T, x, y, i_sc, u_oc, p_max, u_MPP, i_MPP]= fcn(uicurve, T, T_0, alpha_voc,
     alpha_isc)
U = uicurve(1, :);
I = uicurve(2, :);

delta_T = T - T_0;
tc_u = alpha_voc;               % [V/K]
tc_i = alpha_isc;               % [A/K]

len = length(uicurve(1, :));
power = zeros(1, len);
last_valid = len-1;

not_0 = ones(1, len);
for i = 1:len
    if I(i) == 0
        last_valid = i-1;
        if last_valid < 1
            last_valid = 1;
        end
        break;
    end
end
not_0(last_valid+1:end) = zeros(1, len-last_valid);

if tc_i * delta_T > 0
    increase_line = true;
else
    increase_line = false;
end

% move curve up or down on I axis
new_I = (I + tc_i * delta_T) .* not_0;

if increase_line
    missing_y = new_I(last_valid);
    dx = U(last_valid+1) - U(last_valid);
    dy = new_I(last_valid+1) - new_I(last_valid);
    factor = abs(missing_y / dy);
    extra = [U(last_valid) + dx * factor; new_I(last_valid) + dy * factor];
    U(last_valid) = extra(1);
    new_I(last_valid) = extra(2);
end

% stretch or compress curve at U axis
new_U = U * (1 + tc_u * delta_T);

uicurve_T = cat(1, new_U, new_I);

i_sc = new_I(1);
u_oc = new_U(last_valid);
for i = 1:last_valid
    if ~(i == last_valid)
        if (new_I(i) > 0) && ( new_I(i+1) < 0)
```

```
            u_oc = (new_U(i) + new_U(i+1)) / 2.0;
            new_U(i+1) = u_oc;
            new_I(i+1:len) = zeros(1, len-i);
            power(i+1:len) = zeros(1, len-i);
            break;
        end
    end
    power(i) = new_U(i) * new_I(i);
end

p_max = max(power);
index_max = find(power == p_max);

u_MPP = new_U(index_max(1));
i_MPP = new_I(index_max(1));

x = transpose(new_U);
y = transpose(new_I);
```

Quellcode A.4: Prozentuale Reduktion der Kennlinie

```
function new_UI_curve = move_curve_down(UI_curve, normalised_current)

measured_I_SC = UI_curve(2, 1);
correction_current = measured_I_SC * (1 - normalised_current);

new_i_line = (UI_curve(2, :) - correction_current);

new_UI_curve = cat(1, UI_curve(1, :), new_i_line);

% set everythink behind last valid to 0
len = length(new_UI_curve(1, :));
for i = 1:len
    if new_UI_curve(2, i) == 0

        new_UI_curve(2, i+1:len) = zeros(1, len-i);
        break;
    end
    if ~(i == len)
        if (new_UI_curve(2, i) > 0) && ( new_UI_curve(2, i+1) < 0)
            u_OC = (new_UI_curve(1, i) + new_UI_curve(1, i+1)) / 2.0;    % kleine
                ungenauigkeit
            new_UI_curve(1, i+1) = u_OC;
            new_UI_curve(2, i+1:len) = zeros(1, len-i);
            break;
        end
    end
end
```

Quellcode A.5: Arbeitspunktbestimmung des Quellenblocks

```
function [U, I]= calc_power_output(uicurve, R)

x = uicurve(1, :);
y = uicurve(2, :);

len = length(x);
last_valid = len;
for j = 1:len
    if y(j) == 0
        last_valid = j;
        break;
    end
end

    % first finding correct curve interval (line 1)
    if R == 0
        U = x(last_valid);
        I = 0;
    else

        for j = 2:1:last_valid
            if x(j)/R > y(j)
                break;
            end
        end

        % description of line 1
        a1 = (y(j) - y(j-1)) / (x(j) - x(j-1));
        b1 = y(j) - a1*x(j);

        % description of line 2
        a2 = 1/R;
        % b2 = 0;

        % Intersection of both lines is desired Voltage
        U = (b1) / (a2 + (-1*a1));
        I = U / R;

    end
```

Quellcode A.6: Ausschnitt der S-Function zum Parallelschalten von Quellen

```
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *nSources = mxGetPr(ssGetSFcnParam(S, 0));
    int n = *nSources;
    real_T I_ges = 0;
    real_T U_ges = 0;
    int i;

    for(i=0; i<n; i++)
    {
        const real_T *ui_n = (const real_T *) ssGetInputPortSignal(S, i);
        I_ges += ui_n[1];           // sum of all currents
        U_ges += ui_n[0];           // sum of all voltages
    }
    real_T *ui = (real_T *) ssGetOutputPortSignal(S, 0);

    ui[0] = U_ges / n;
    ui[1] = I_ges;

    // R_n = R * (I_ges / I_n)          IF: I_ges != 0

    const real_T *R = (const real_T *) ssGetInputPortSignal(S, n);
    for(i=1; i<n+1; i++)
    {
        const real_T *ui_n = (const real_T *) ssGetInputPortSignal(S, i-1);
        real_T I_n = ui_n[1];

        real_T *R_n = (real_T *) ssGetOutputPortSignal(S, i);

        if(I_n != 0){
            *R_n = *R * (I_ges / I_n);
        } else {
            real_T U_n = ui_n[0];

            if(U_n != 0){
                *R_n = *R * (U_ges / U_n);
            } else {
                *R_n = 0;
            }
        }
    }
}
```

Quellcode A.7: MPPT Reglelalgorithmus

```
function R = MPPT(U, I)

% Constants
f1 = 1.01;
f2 = - 0.005;
start_r = 100;          % Start value of Load
delay = 1;              % One step delay of circuit

P = U * I;

persistent last_p;
persistent last_r;
persistent more_r;
persistent frame;

if isempty(last_p)              % Init first call
    last_p = P * ones(1, 1+delay);
    last_r = start_r * ones(1, 1+delay);
    more_r = true;
    R = last_r(end) * f1;
    frame = delay + 1;
else
    if frame == delay + 1       % Normal call frame
        if P == 0
            R = start_r;
        elseif P > last_p(end)
            if more_r
                R = last_r(end) * f1;
            else
                R = last_r(end) / (f1-f2);
            end
        else
            if more_r
                R = last_r(end) / (f1-f2);
                more_r = false;
            else
                R = last_r(end) * f1;
                more_r = true;
            end
        end
        frame = 1;
    else                        % Wait frame
        R = last_r(frame);
        frame = frame + 1;
    end

    for i = delay+1 : -1 : 2    % Shift all last values in memory
        last_p(i) = last_p(i-1);
        last_r(i) = last_r(i-1);
    end
    last_p(1) = P;
    last_r(1) = R;

end
```

Quellcode A.8: Zeitmodul für eine globale Zeitangabe

```
function time = calcTime(sec, min, hour, day, month, year, stepkind, stepsize, counter,
    timezone)

% cumulative number of days prior to beginning of month
month_days = [ 0,  31,  59,  90, 120, 151, 181, 212, 243, 273, 304, 334, 365];
month_days_leap = [ 0,  31,  60,  91, 121, 152, 182, 213, 244, 274, 305, 335, 366];

total_sec = sec + (min * 60) + (hour * 3600) - (timezone * 3600);
total_day = day + month_days(month);
leap_year = 0;
if mod(year, 4) == 0 && (mod(year, 100) ~= 0 || mod(year, 400) == 0) && month > 2
    leap_year = 1;
end
total_day = total_day + leap_year;

if total_sec < 0            % Because of timezone
    total_sec = total_sec + (24 * 3600);
    total_day = total_day - 1;
    if total_day <= 0
        if leap_year
            total_day = total_day + 366;
        else
            total_day = total_day + 365;
        end
        year = year - 1;
    end
end

counter = double(counter);

switch stepkind
    case 1              % Second
        total_sec = total_sec + counter * stepsize * 1;
    case 2              % Min
        total_sec = total_sec + counter * stepsize * 60;
    case 3              % Hour
        total_sec = total_sec + counter * stepsize * 3600;
    case 4              % Day
        total_day = total_day + counter * stepsize;
    case 5              % Month
        month = month + counter * stepsize * 1;
        year = year + floor((month - 1) / 12);
        month = mod(month - 1, 12) + 1;
        time = [sec; min; hour; day; month; year];
        return;
    otherwise           % Year
        year = year + counter * stepsize * 1;
        time = [sec; min; hour; day; month; year];
        return;
end

total_day = total_day + floor(total_sec / (3600*24));   % ganze Tage Ã¼berlauf an Sekunden
total_sec = mod(total_sec, 3600*24);                    % Rest ist h/m/s des Tages
hour = floor(total_sec / 3600);                         % ganze Stunden der Sekunden
```

```
54 total_sec = mod(total_sec, 3600);                          % Rest ist m/s des Tages
55 min = floor(total_sec / 60);                               % ganze Minuten der Sekunden
56 sec = mod(total_sec, 60);                                  % Rest sind Sekunden
57
58 weiter = 1;
59 while weiter
60     leap_year = 0;
61     if mod(year, 4) == 0 && (mod(year, 100) ~= 0 || mod(year, 400) == 0) && month > 2
62         leap_year = 1;
63     end
64
65     if total_day > (365 + leap_year)
66         year = year + 1;
67         total_day = total_day - (365 + leap_year);
68     else
69         if leap_year
70             for i = 1:13
71                 if total_day <= month_days_leap(i)
72                     break;
73                 end
74             end
75             month = i-1;
76             day = total_day - month_days_leap(i-1);
77         else
78             for i = 1:13
79                 if total_day <= month_days(i)
80                     break;
81                 end
82             end
83             month = i-1;
84             day = total_day - month_days(i-1);
85         end
86         weiter = 0;
87     end
88 end
89
90 time = [sec; min; hour; day; month; year];
```