Tutorial für die Benutzung von AnyLogic und Octave im Rahmen der Vorlesung Modellgestützte Analyse und Optimierung

Jessica Bühler

8. Februar 2018

Inhaltsverzeichnis

Abbildung 1: Download der „FREE PLE“ Version von AnyLogic

1 Einführung

Mit dem Ersetzen des Experimentierens am realen Objekt durch rechnergestützte und modellbasierte Analyse werden Tools zum Dürchführen dieser Analysen benötigt. Dieses Tutorial soll in das Tool AnyLogic einführen und die theoretischen Grundlagen, die in der Vorlesung vermittelt werden, vertiefen. In diesem einführenden Kapitel wird zunächst in Unterkapitel 1.1 ein Überblick über die Installation von AnyLogic gegeben. Danach schliesst das Kapitel in Unterkapitel 1.2 mit einem Überblick über die grafische Benutzeroberfläche des Programms ab.

1.1 Installation

Das Programm AnyLogic kann unter `http://www.AnyLogic.de/downloads` für Windows, Mac und Linux runtergeladen werden. Die Version „FREE PLE“ ist eine für Studenten kostenlose Version, die hier genutzt wird. Um diese Version herunterzuladen wird wie in Abbildung 1 zu sehen ist auf das jeweilig gewünschte Betriebssystem und die 32 oder 64 Bit Variante geklickt. Anschließend muss ein Formular der Firma AnyLogic ausgefüllt werden[1], um die kostenlose Version zu erhalten. Nach dem Ausfüllen des Formulars kann die Installationsdatei heruntergeladen werden. Im folgenden Schritt wird diese Datei auf dem System entsprechend der Vorgehensweise des jeweiligen Systems installiert.

1.2 Überblick über den Aufbau von AnyLogic

Die grafische Benutzeroberfläche von AnyLogic teilt sich in verschiedene Fenster auf:

- den Editor
- die Ansicht für die Simulation
- und ein Hilfefenster

[1] Hier ist es nicht notwendig seine Daten herauszugeben.

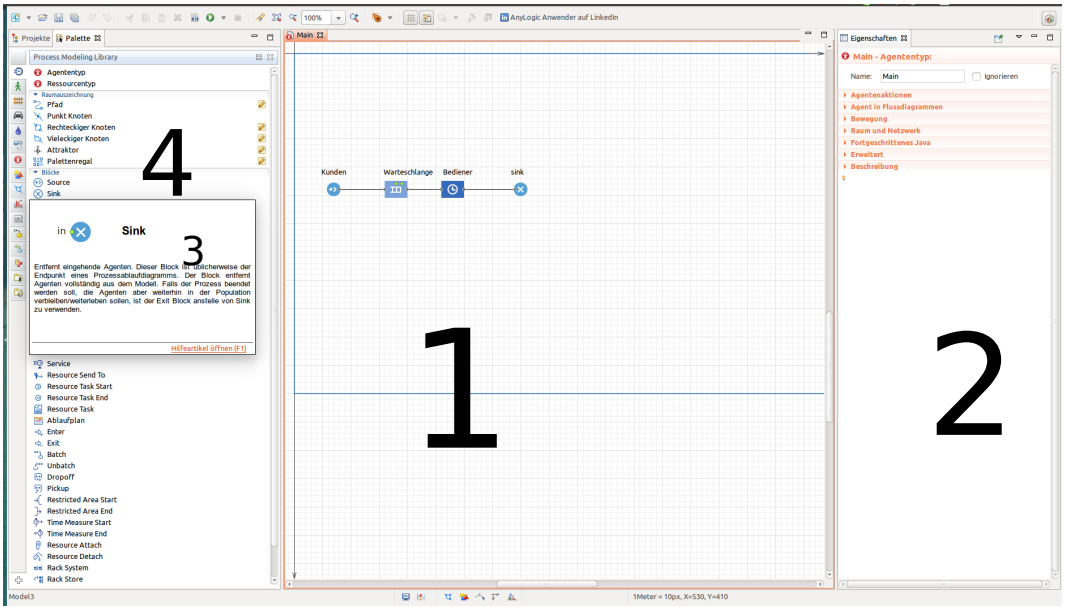


Abbildung 2: Übersicht über den AnyLogic Editor

In diesem Unterkapitel wird zunächst der Aufbau des Editors in Paragraph 1.2.1 gefolgt vom Aufbau der Ansicht für die Simulation in Paragraph 1.2.2 erläutert.

1.2.1 AnyLogic Editor

Den AnyLogic Editor sieht man in Abbildung 2. Zu diesem gelangen wir, indem auf dem „Willkommen“ Bildschirm ein neues Modell erstellt wird. Die Auswahl der Eigenschaften beim Erstellen eines neuen Modells wird später bei den einzelnen Modellen erklärt. Die Zahlen in Abbildung 2 stellen die verschiedenen Elemente der Benutzeroberfläche dar:

1. Hauptfenster, in dem das System nachgebildet wird
2. Eigenschaftsfenster für das aktuell im Hauptfenster ausgewählte Element
3. Erklärmenü, das nach kurzen Zeigen auf ein Element im Palettenmenü auftaucht.
4. Palettenmenü mit den möglichen Modellelementen

Im Palattenmenü ist links zusätzlich eine Auswahlmöglichkeit für verschiedene Modellelemente und Simulationselemente zu finden.

1.2.2 Simulationsansicht

Ist das gewünschte Modell im AnyLogic Editor fertig erstellt, so kann es simuliert werden. Zu diesem Zweck ist in der Editoransicht in der oberen Menüleiste ein grüner Pfeil zu sehen. Alternativ kann die Taste *F5* benutzt werden. Nach dem Klicken

Abbildung 3: Leiste aus der AnyLogic Simulationsansicht

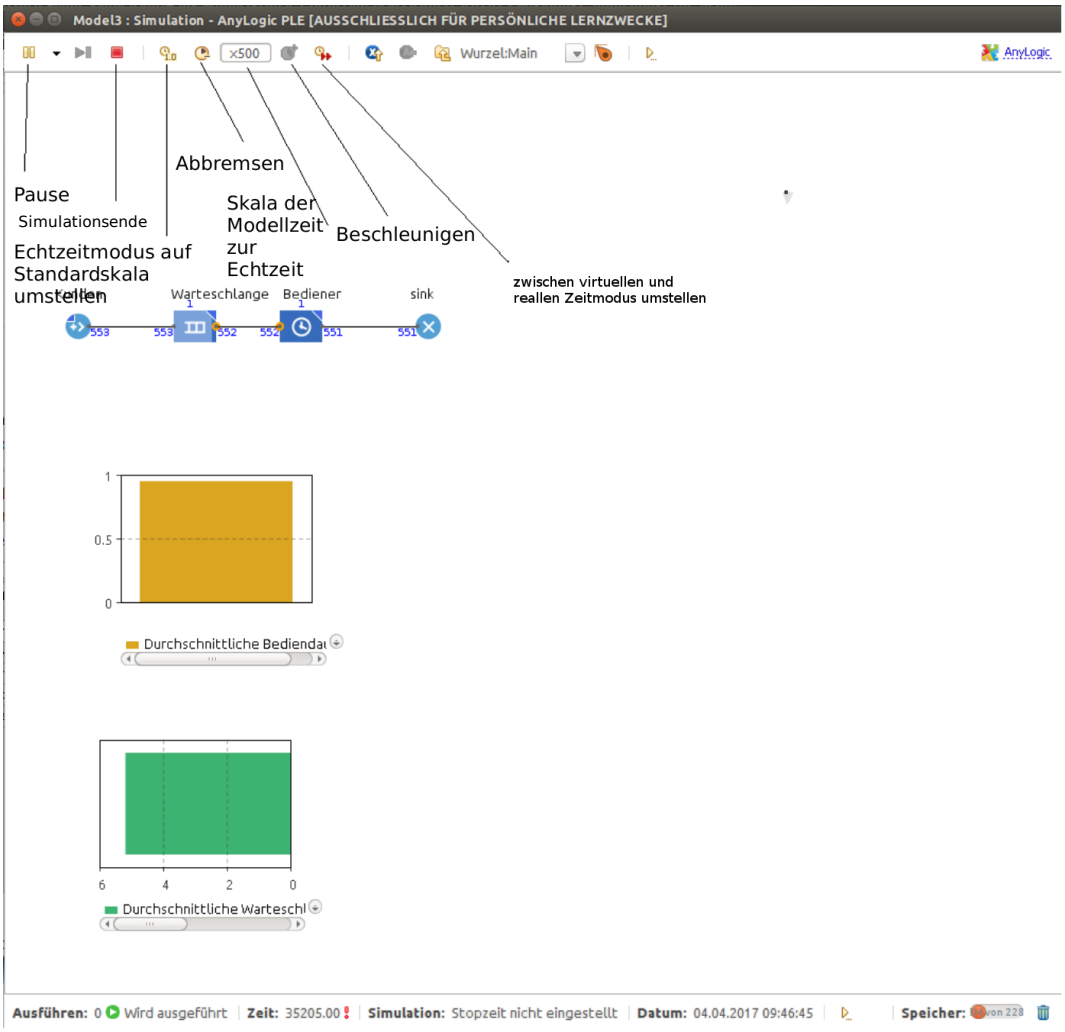


Abbildung 4: Übersicht über die AnyLogic Simulationsansicht

auf diesen Button erscheint entweder in der Statusleiste der Editoransicht im unteren Bereich eine Fehlermeldung (welches es vor der Simulation zu finden gilt) oder es öffnet sich das Fenster der Modellansicht. Um das Modell zu simulieren muss nun noch auf den grünen Pfeil in Abbildung 3 geklickt werden. In Abbildung 4 ist noch zu sehen, wie auf das Tempo der Simulation Einfluss genommen werden kann.

2 Erstes Modell mit AnyLogic

Anhand eines einfachen Beispiels aus dem Skript[2] soll die Funktionsweise von AnyLogic erläutert werden. Wir betrachten als Beispiel einen Schalter an dem Be-

[2] Kapitel 2.1.3

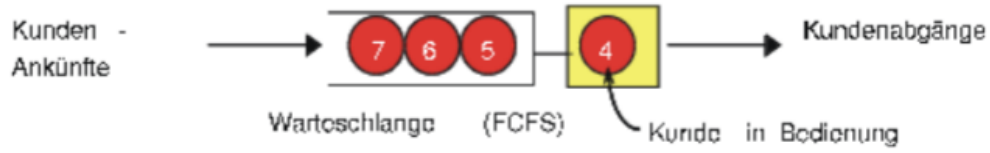


Abbildung 5: Aufbau des Systems

dienungen stattfinden. Es existiert genau ein Bediener, der zu jedem Zeitpunkt maximal einen Kunden bedienen kann und immer arbeitet, d.h. Kunden bedient, wenn Kunden im System sind. Kunden betreten das System (d.h. kommen am Schalter an) und verlangen nach Bedienung. Falls der Schalter belegt ist, so warten die Kunden in einem Warteraum. Wir nehmen an, dass der Warteraum eine potentiell unendliche Kapazität hat und das Kunden nach der Strategie (FCFS) *(first come first served)* bedient werden. Nach Beendigung der Bedienung verlassen die Kunden das System. Die Ankunftszeit der Kunden sei durch eine exponential verteilte Zwischenankunftszeit gegeben.
Abbildung 5 zeigt, wie dieses System aufgebaut ist.
Ziel dieses Kapitels ist es dieses System in AnyLogic zu simulieren. Um diese Aufgabe zu erfüllen muss das System im AnyLogic Editor nachgebildet werden.

2.1 Aufbau des Modells im AnyLogic Editor

Um das System nachzubilden brauchen wir vier Elemente aus dem Menübereich der Palette:

- Source
- Sink
- Delay
- Queue

Diese Elemente müssen im Main-Bereich des Editors positioniert werden, wie in Abbildung 4 zu sehen ist. Dafür muss im Palettenmenü das Untermenü der Process Modelling Library genutzt werden. Anschließend werden diese Elemente untereinander wie in Abbildung 6 zu sehen ist verbunden. Die Verbindung wird durch einen Doppelklick auf den Anfang und das Ende einer gewünschten Verbindung erstellt. Achtung: Schiebt man im Editor zwei Elemente nah zueinander, so werden automatisch Verbindungen erstellt, die aber nicht unbedingt der gewünschten entsprechen und Fehler erzeugen können.

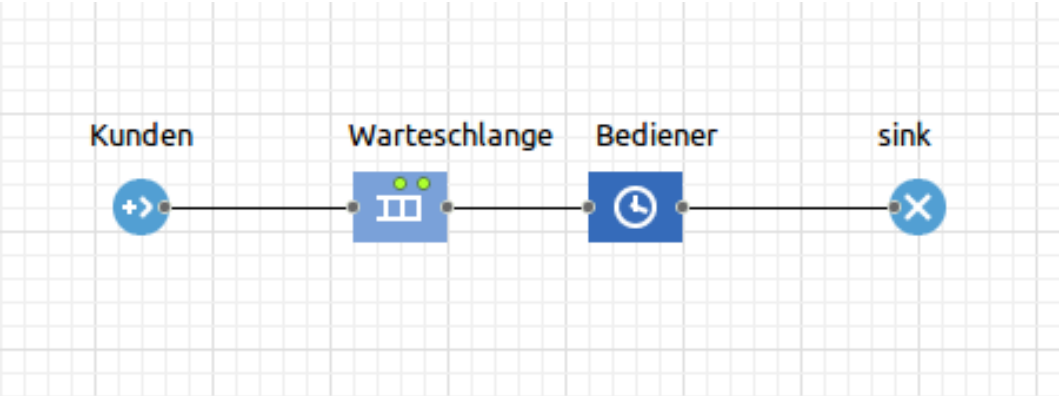


Abbildung 6: Übersicht über das MM1 Modell

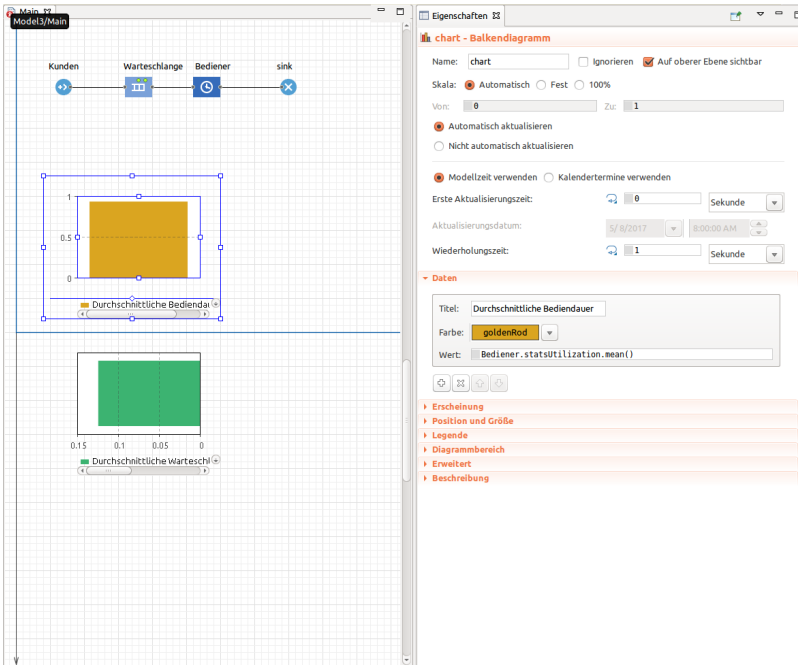


Abbildung 7: Konfiguration des 1. Balkendiagramms

Um die Simulation des Modells auswerten zu können macht es Sinn während der Simulation Daten zu sammeln, welche dann in einem Graph visualisiert werden können. Zu diesem Zweck verwenden wir in diesem Modell zwei Balkendiagramme. Diese Balkendiagramme finden sich im Palettenmenü in der Unterauswahl Analyse. In Abbildung 7 ist zu sehen, wie die durchschnittliche Bediendauer zum aktuellen Zeitpunkt der Simulation angezeigt werden kann. Hierfür wird auf dem Bediener die Funktion statsUtilization und mean() aufgerufen. In Abbildung 8 ist zu sehen wie die durchschnittliche Länge der Warteschlange zum aktuellen Zeitpunkt angezeigt werden kann. Hierfür wird auf der Warteschlange die Funktion statSize und mean() aufgerufen.

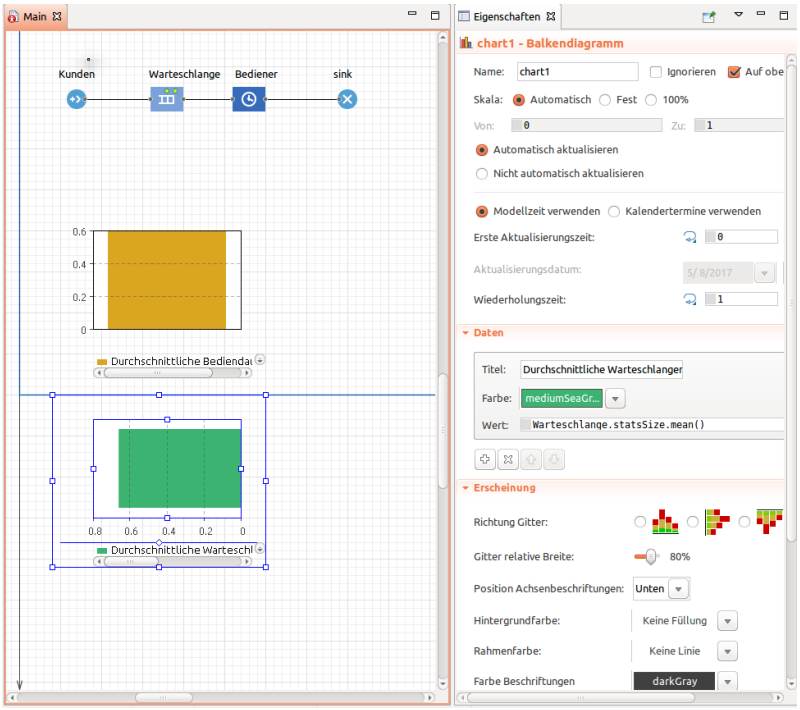


Abbildung 8: Konfiguration des 2. Balkendiagramms

2.2 Simulation des Modells

Bei der Simulation des Modells kann beobachtet werden, wie einzelne Kunden im System ankommen. Zusätzlich kann anhand der Balkendiagramme der aktuelle Wert der Wartschlangengröße und die durchschnittliche Bediendauer abgelesen werden.

3 Optimierung mit AnyLogic

Wie in den vorherigen Kapiteln eingeführt wurde eignet sich AnyLogic zur Modellierung und Simulation. Ein weiteres Aufgabenfeld im Bereich der Modellierung ist das Optimiern von Modellparametern. Dies wollen wir in diesem Kapitel erarbeiten. Dafür benutzen wir das Beispiel mit dem Namen „Oil Terminal“. Diese kann beim Start des Programms aus den Beispielen ausgewählt werden, es befindet sich aber auch in dem zu diesem Tutorial gehörenden Archiv. Es zeigt zusätzlich neben der Optimierung noch, welche grafischen Möglichkeiten AnyLogic bietet. Beim Start von Anylogic muss auf die Schaltfläche „Beispiele öffnen“ und dann auf das Beispiel „Oil Terminal“ geklickt werden oder die Modelldatei aus dem Archiv geladen werden.

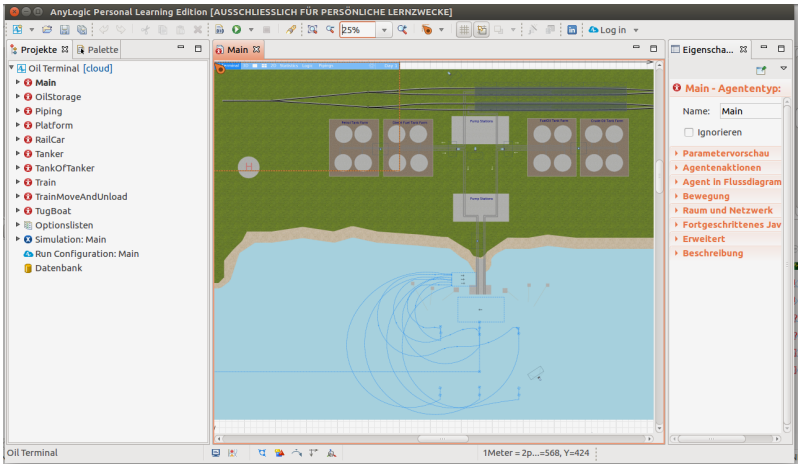


Abbildung 9: Übersicht über Beispiel Oil Terminal

Abbildung 10: 3D Ansicht Oil Terminal

3.1 Überblick über das Beispiel

In Abbildung 9 ist die Simulation des Öl-Terminals nach dem Öffnen zu sehen. Wie in Abbildung 10 zu sehen ist, ermöglicht AnyLogic auch das Erstellen von 3D Ansichten[3].

Das Modell simuliert das Verladen von Öl auf die Schiene und den Wasserweg sowie das Zwischenlager in Öltanks. Bestandteile des Modells sind

- Bahnsteige zum Verladen und Empfangen von Ölprodukten
- Tanks zum Lagern von Ölprodukten
- Öltanker zum Verschiffen von Ölprodukten

[3] Wie genau diese 3D Modellierung funktioniert würde dieses Tutorial sprengen und ist in einem AnyLogic Video Tutorial unter `https://www.youtube.com/watch?v=L5SD_QdUUKE&feature=youtu.be&list=PLUJJN9tmVTj1czFMt9IKi6wge9GnmMTN3` zu finden.

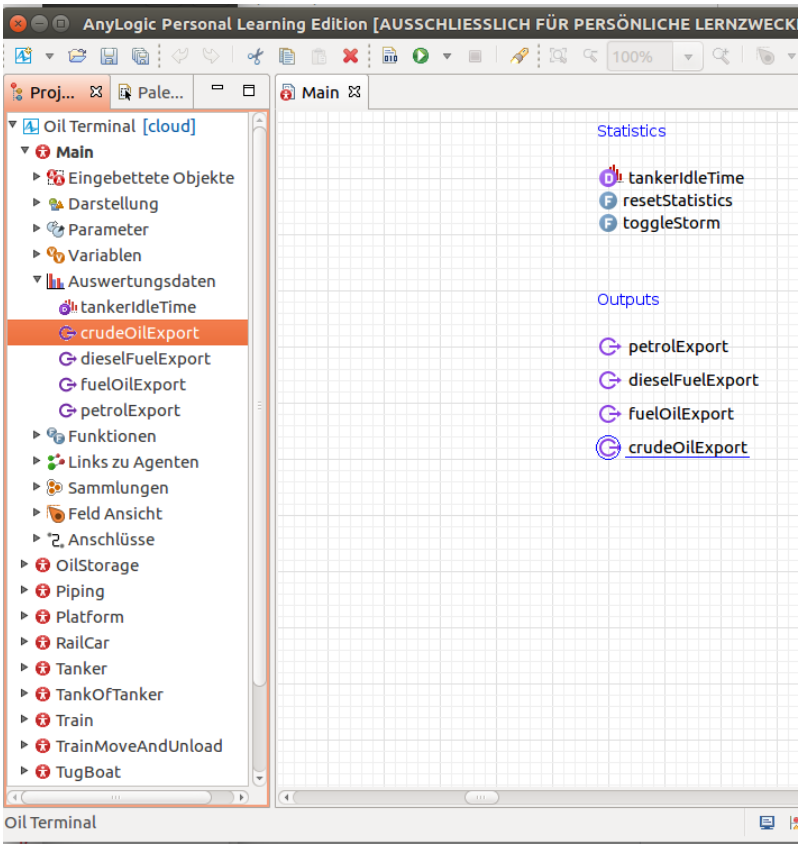


Abbildung 11: Auswertungsdaten anzeigen lassen

3.2 Optimierungsexperiment

Der Hauptagent des Beispielmodells hat folgende unterschiedliche, veränderbare Parameter:

- Zwischenankunftszeit der Tanker
- Anzahl an Tanks in einem Tanker
- Kapazität eines Tanks
- Zugkapazität

Das Optimierungsexperiment wird genutzt um für diese Parameter optimale Wertebelegungen herauszufinden. Dafür müssen Parameter unter den vorhandenen bestimmt werden, die veränderlich sind. Die Parameter werden anhand einer Kostenfunktion optimiert. In diesem Fall will die Kostenfunktion die Menge an exportiertem Öl maximieren. Dafür wird zunächst ein Ausdruck gebraucht, der die absolute Menge an exportierten Ölprodukten berechnet. Um diesen Ausdruck anzulegen muss man wie in Abbildung 11 zu sehen ist unter dem Mainagenten im Projektfenster auf „Auswertungsdaten“ klicken und dann einen der Unterpunkte auswählen.

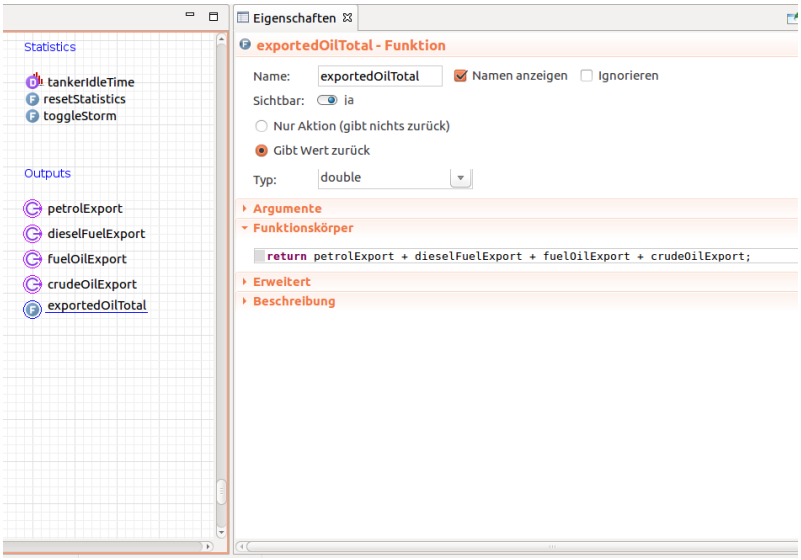


Abbildung 12: Funktion zur Ausgabe des totalen Exports an Ölprodukten

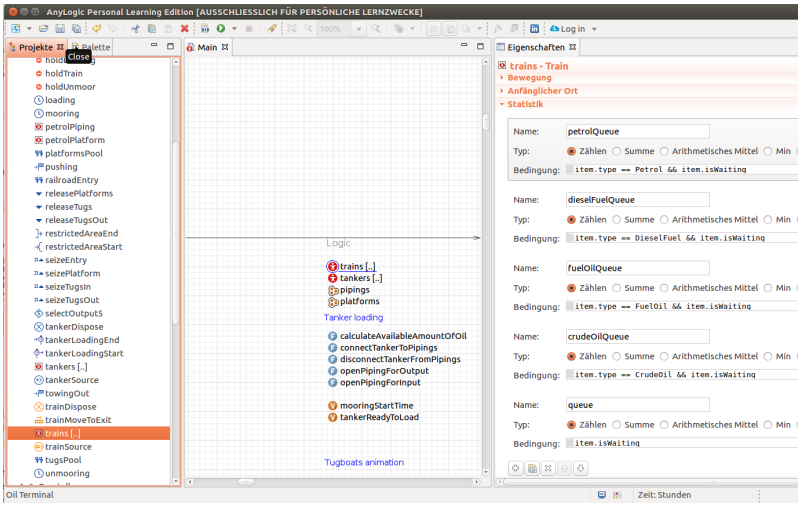


Abbildung 13: Übersicht über die Züge

Nun muss man in der Palette unter dem Reiter Agent ein Funktionselement auswählen und auf die in Abbildung 11 sichtbare Fläche ziehen. In Abbildung 12 ist zu sehen, welche Einstellungen für dieses Element geändert werden müssen.

Als zusätzliche Anforderung soll in diesem Beispiel modelliert werden, dass Züge nicht in einer Warteschlange warten bevor sie abgefertigt werden. Das heißt die Warteschlange soll eine maximale durchschnittliche Länge von Null haben. Zu diesem Zweck werden im Projektfenster unter dem Menüpunkt Eingebettete Objekte die Züge (trains) ausgewählt. Nach der Auswahl erscheint im Hauptfenster der in Abbildung 13 angezeigte Inhalt.

In diesem Fenster werden einige statistische Funktionen für Zugpopulation angezeigt. Die Queue Statistik gibt zum Beispiel zurück wieviele Züge sich gerade in der Warteschlange aufhalten. Um die maximale durchschnittliche Warteschlangen-

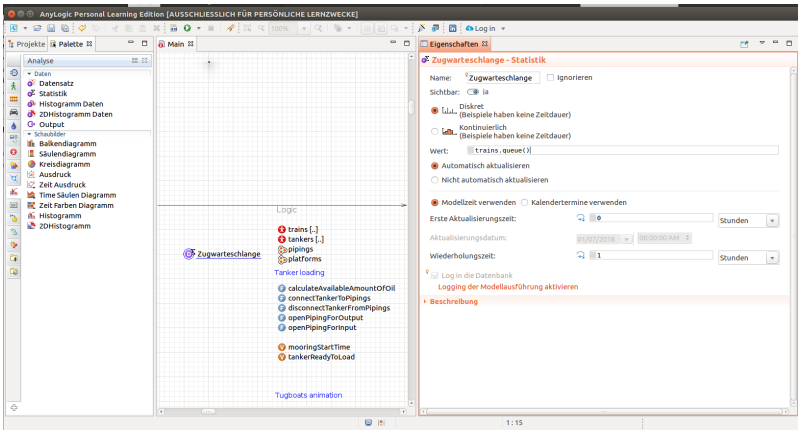
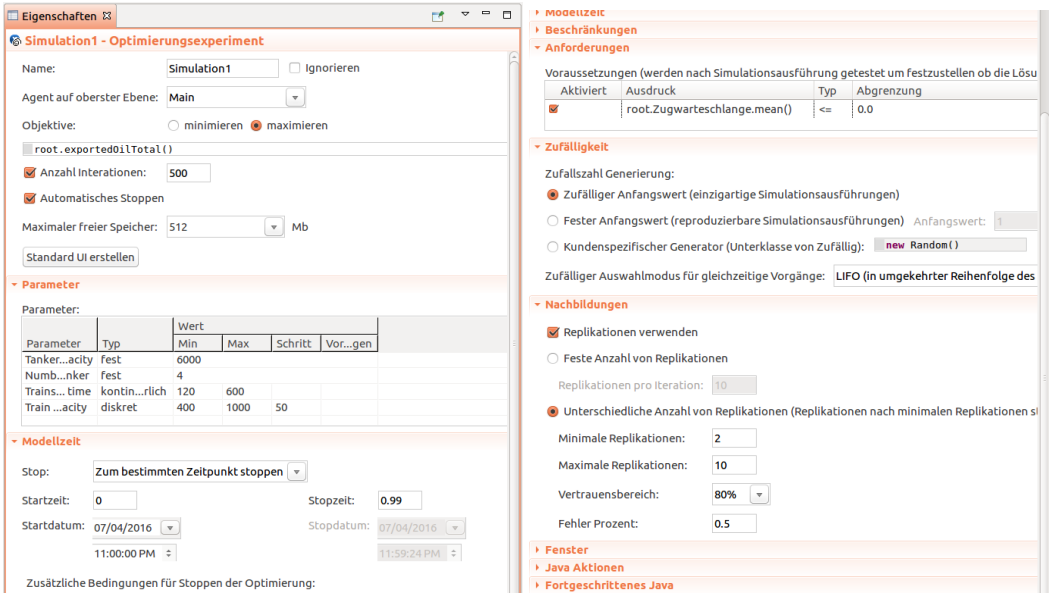


Abbildung 14: Einstellungen der Zugwarteschlange

länge während der gesamten Simulation abzurufen wird ein Statstikelement hinzugefügt. Dafür wird im Palettenfenster unter dem Analysereiter der Punkt Statistik ausgewählt und in das Hauptfenster gezogen. In Abbildung 14 ist zu sehen, welche Einstellungen dieses Statistikelement haben muss.

Als nächstes muss nun ein Optimierungsexperiment erstellt werden, mit dem für die gewünschten Parameter optimale Werte gefunden werden. Dafür muss im Projektfenster ein Element des Modells mit einem Rechtsklick ausgewählt werden. Dabei öffnet sich ein Menü, wo der Menüpunkt Neu und der Untermenüpunkt Experiment ausgewählt werden muss. Im nächsten Schritt wird in dem nun vorhandenen Fenster das Optimierungsexperiment gewählt und mit Fertigstellen das Optimierungsexperiment erstellt. In Abbildung 15 sind die notwendigen Einstellungen zu sehen. Die Eigenschaft „Automatische Stoppen in Abbildung 15 bezieht sich darauf, dass ein Simulationsdurchlauf beendet werden kann, wenn durch in diesem Durchlauf keine Verbesserung mehr zu finden ist. Die Parameter sind so eingestellt, dass die beiden Parameter Zwischenankunftszeit und Zugkapazität veränderlich sind und die anderen Parameter fest sind. Bei der Zwischenankunftszeit ist sicher trivial zu erklären wieso diese Größe im Gegensatz zu den festen veränderlich sein muss. Auch die Zugkapazität lässt sich relativ einfach durch Anhängen weiterer Wagons verändern. Die Modellzeit ist hier so gewählt, dass eine Stunde Simulationszeit nicht überschritten wird, da dies eine Begrenzung der freien Version von AnyLogic ist. Unter dem Menüpunkt Anforderungen wird die gewünschte Länge der Zugwarteschlange modelliert. Da jeder Simulationslauf in diesem Modell von stochastischen Größen bestimmt wird, wird hier ein zufälliger Anfangswert für jeden Durchlauf gewählt. Der Punkt Nachbildungen erlaubt es, zu entscheiden wieviele Wiederholungen bei der Entscheidung für die optimalen Parameter durchgeführt werden.

Nachdem Klicken auf den Pfeil neben dem Run-Button um das Optimierungsexperiment auszuführen wird im Simulationsfenster ein Optimierungsexperiment angezeigt (siehe Abbildung 16), welches die optimalen Werte für die veränderlichen



(a) (b)

Abbildung 15: Einstellungen des Optimierungsexperiments

Parameter berechnet. Ist dieses Experiment durchgelaufen, so können mittels des „Copy“ Buttons die gefundenen Parameter in die Zwischenablage kopiert werden. In Abbildung 17 ist zu sehen wie die Parameter in das normaöle Modell eingefügt werden können um weiter für die Simulation genutzt werden zu können.

4 Lösung von linearen Optimierungsproblemen mit Octave

4.1 Einführung

Zur computergestützten Lösung linearer Optimierungsprobleme gibt es viele Programme. Einige dieser Programme sind teure Lösungen für die kommerzielle Verwendung. Andere Lösungsmöglichkeiten sind in webunterstützte Möglichkeiten kleinere Optimierungsprobleme zu Lösen. Im Rahmen dieses Tutorials wird Octave als Programm zur Lösung von linearen Optimierungsproblemen vorgestellt, da es zum einen kostenlos erhältlich ist und zum anderen auch für etwas grössere Optimierungsprobleme innerhalb einer angemessenen Zeitspanne Lösungen errechnet. Wir wollen in diesem Kapitel lernen, wie lineare Optimierungsprobleme mit Octave zu lösen sind und wie die Ausgaben von Octave zu interpretieren sind. Die Installationsdateien von Octave und die dazu gehörigen Anleitungen finden sich unter `https://www.gnu.org/software/octave/download.html`. In Abbildung 18 ist die Oberfläche von Octave zu sehen. Im Befehlsfenster werden die Eingaben unseres Optimierungsproblems eingegeben und die Ausgaben unse-

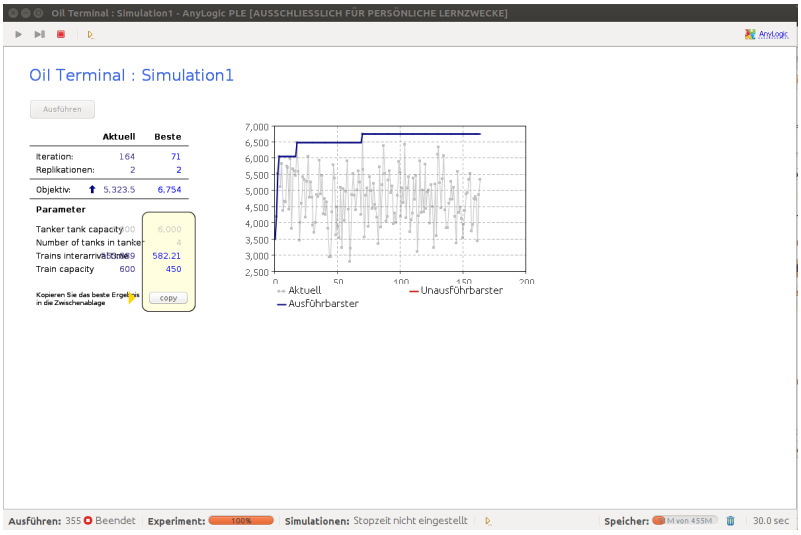


Abbildung 16: Simulationsfenster mit dem Optimierungsexperiment

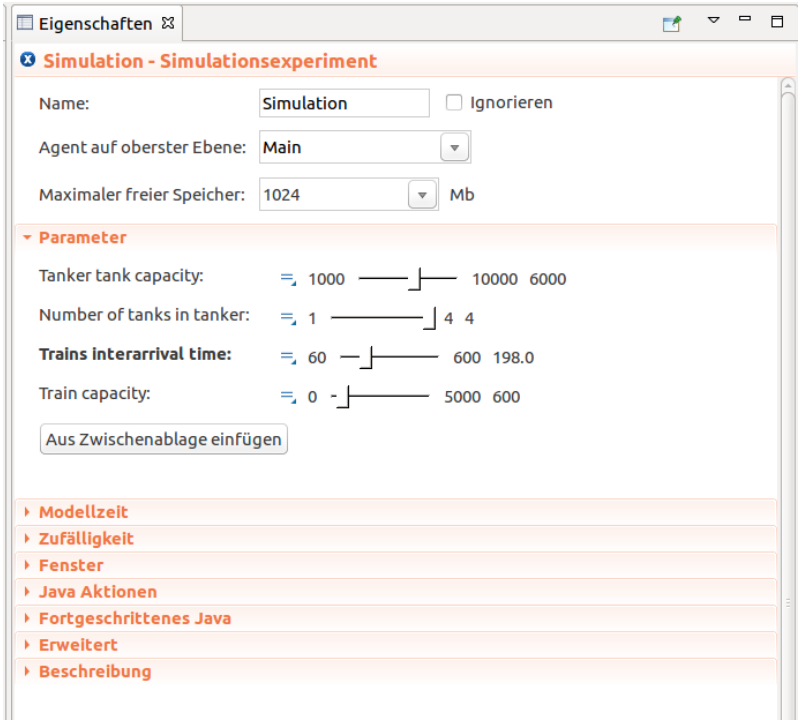


Abbildung 17: Simulationsfenster mit dem Optimierungsexperiment

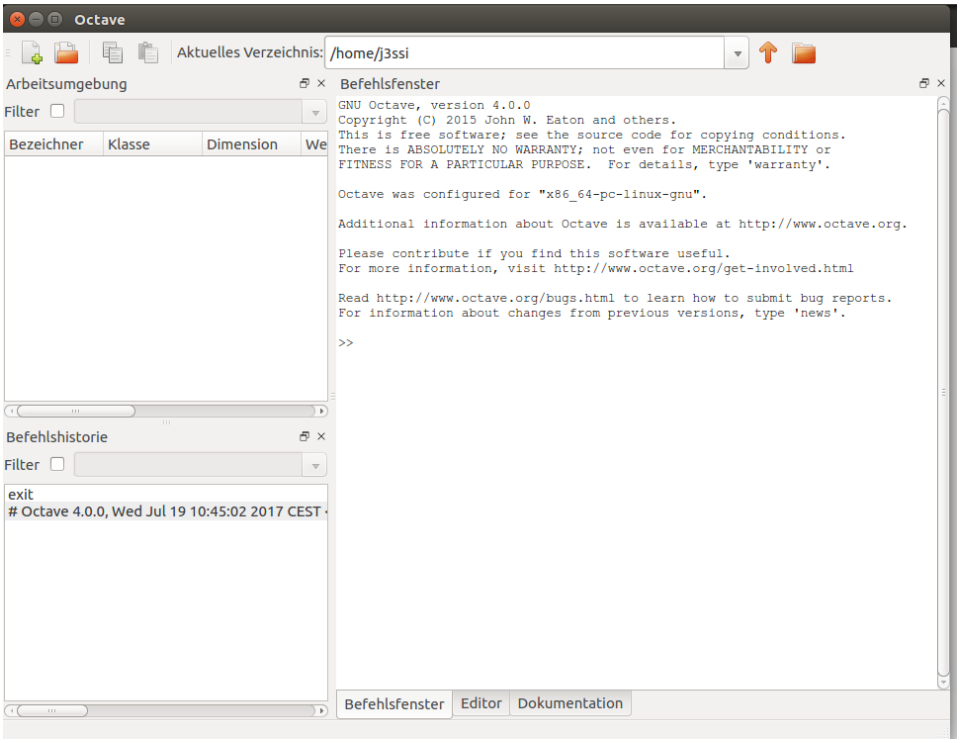


Abbildung 18: Grafische Benutzeroberfläche von Octave

res Programmes ausgegeben.
Nun muss ein lineares Optimierungsproblem in die Form gebracht werden, in der Octave es akzeptiert und eine Lösung berechnet.

4.2 Formulierung des Optimierungsproblems in Octave

Gegeben ist ein lineares Optimierungsproblem. Die Funktion

$$f(\mathbf{x}) = \sum_{i=1}^{n} c_i \cdot x_i \tag{1}$$

soll unter den Nebenbedingungen

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_j \; (i = 1, \ldots, m) \text{ und } x_j \geq 0$$

minimiert werden. Die verschiedenen Arten von Lösungen die für dieses Problem möglich sind umfassen

- eine optimale Lösung
- unendlich viele optimale Lösungen
- einen leeren Lösungsraum
- keine optimale Lösung bei unbeschränktem Lösungsraum

Diese Lösungmöglichkeiten sollen nun anhand von Beispielen in Octave aufgezeigt werden. Zunächst betrachten wir das Optimierungsproblem

$$\min -2x_1 - x_2 \tag{2}$$

mit den Nebenbedingungen

$$x_1 + x_2 \leq 5 \tag{3}$$
$$x_1 - 3x_2 \leq 1 \tag{4}$$

Dieses Optimierungsproblem besitzt eine optimale Lösung. Diese Optimierungsproblem wird nun in die Vektor/Matrix- Schreibweise überführt, da Octave diese Form als Eingabe erwartet. Aus

$$\min -2x_1 - x_2 \tag{5}$$

wird

$$\min \left([-2-1]^T \cdot [x_1 x_2] \right). \tag{6}$$

Und aus den Nebenbedingungen erhalten wir

$$\begin{matrix} x_1 + x_2 \leq 5 \\ x_1 - 3x_2 \leq 1 \end{matrix} \Rightarrow \begin{bmatrix} 1 & 1 \\ 1 & -3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \quad x_1, x_2 \geq 0$$

Diese Gleichungen in Matrix- /Vektor Schreibweise wird nun in Programmcode für Octave überführt. Aus

$$\min \left([-2-1]^T \cdot [x_1 x_2] \right). \tag{7}$$

wird

$$c = [-2-1]' \tag{8}$$

Und aus

$$\begin{matrix} x_1 + x_2 \leq 5 \\ x_1 - 3x_2 \leq 1 \end{matrix} \Rightarrow \begin{bmatrix} 1 & 1 \\ 1 & -3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \quad x_1, x_2 \geq 0$$

wird

$$a = [1,1;1,-3], b = [5,1]' \tag{9}$$

Zur Spezifikation der Nichtnegativbedingungen von x_1 und x_2 wird folgendes eingegeben:

$$lb = [0,0]' \tag{10}$$

Dies ist eine Spezifikation, die nicht nur zum Setzen einer Nichtnegativbedingung geeignet ist sondern es ermöglicht, eine untere Schranke für die Lösung einzugeben. Zusätzlich können auch noch obere Schranken angegeben werden. Sind obere Schranken im Optimierungsproblem nicht spezifiziert, so geben wir

$$ub = [] \tag{11}$$

ein. Zusätzlich erwartet die Funktion *glpk*, die in Octave für das Lösen von linearen Optimierungsproblemen verantwortlich ist, noch einige andere Aussagen über die Beschaffenheit von Parametern des Optimierungsproblems. So ist es möglich für die Nebenbedingungen verschiedene Vergleichsoperatoren zu verwenden $(\leq, <, =, \geq, >)$. Für jede einzelne Gleichung muss der Typ der Ungleichung angegeben werden. In unseren Fall müssen wir daher

$$ctype = \text{“}UU\text{”} \tag{12}$$

eingeben[4].

Octave kann als Lösungsraum sämtliche reelle Zahlen akzeptieren als auch den Lösungsraum auf ganze Zahlen einengen. Da in unseren Fall der Lösungsraum die gesamten reellen Zahlen umfassen soll geben wir

$$vartype = \text{“}CC\text{”} \tag{13}$$

ein. Zusätzlich muss noch angegeben werden, ob die Zielfunktion minimiert $(s = 1)$ oder maximiert $(s = -1)$ werden soll.

Diese Informationen spezifizieren das lineare Optimierungsproblem nun vollständig und können so an die Funktion *glpk* weitergegeben werden:

$$[xmin, fmin, status, extra] = glpk(c, a, b, lb, ub, ctype, vartype, s); \tag{14}$$

Nach diesem Aufruf finden sich in

$$[xmin, fmin, status, extra] \tag{15}$$

[4] Für die anderen Fälle stehen die zu wählenden Parameter in https://www.gnu.org/software/octave/doc/v4.2.0/Linear-Programming.html

die Ergebnisse. Diese können durch das Aufrufen der Variablen ausgelesen werden:

$$>> xmin \tag{16}$$

$$xmin = \tag{17}$$

$$4 \tag{18}$$

$$1 \tag{19}$$

$$\tag{20}$$

Ein realistisches anwendungsnahes Optimierungsproblem hat in der Regel keine zwei sondern eventuell sogar hunderte Nebenbedingungen und Variablen. In diesem Fall ist es dann sicher nicht mehr sinnvoll das Programm so direkt einzugeben. Octave unterstützt hier, dass der Code des Optmierungsproblems in einer Datei gespeiuchert werden kann[5]. Die Funktion *glpk* beruht auf dem Simplexalgorithmus. Dieser einen Ausgabe von *xmin* sehen wir nicht an, ob das zugehörige Optimierungsproblem eine oder unendlich viele Lösungen besitzt. Daher soll nun ein Optimierungsproblem mit Octave gelöst werden von dem wir wissen, dass es unendlich viele optimale Lösungen hat:

$$\min -3x_1 - 4x_2 \tag{21}$$

$$x_1, x_2 \geq 0 \tag{22}$$

$$-3x_1 - 4x_2 \geq -5 \tag{23}$$

An diesem Beispiel ist besonders, dass die Nebenbedingung eine Isogewinngerade ist.
Wie im ersten Beispiel wird aus diesem Optimierungsproblem Octave-Programmcode generiert. An dieser Stelle wird der Leser dazu aufgerufen dieses Optimierungsproblem selbstständig wie im ersten Beispiel mit Octave zu lösen.
Der Octave-Programmcodee sieht folgendermassen aus:

```
>> c=[-3  -4]'
>> a=[-3, -4]
>> b=[-5]
>> lb=[0,0]'
>> ub=[]
>> ctype="L"
>> vartype="CC"
>> s=1
>> [xmin,f,status]=
   glpk(c,a,b,lb,ub,ctype,vartype,s)
```

[5] Eine Erklärung dazu findet sich unter

```
xmin =
 0.00000
 1.25000
f = -5
```

Auf den ersten Blick würde man hier nur eine Lösung vermuten, es lässt sich aber durch Veränderung der unteren Grenze zeigen, dass es mindestens zwei Lösungen gibt[6]:

```
>> lb=[1  0]'
>> [xmin,f,status]=
     glpk(c,a,b,lb,ub,ctype,vartype,s)
xmin =
 1.00000
 0.50000
f = -5
```

Als nächste wird ein Optimierungsproblem betrachtet, bei dem der Lösungsraum durch einen Widerspruch bei den Nebenbedingungen leer ist:

$$\min -3x_1 - 4x_2 \tag{24}$$

$$x_1, x_2 \geq 0 \tag{25}$$

$$x_1 \leq -3 \tag{26}$$

$$x_2 \leq -4 \tag{27}$$

Das lineare Optimierungsproblem ändert sich dann zu:

```
>> c=[-3  -4]'
>> a=[1,  0;0,  1]
>> b=[-3,-4]
>> lb=[0,0]'
>> ub=[]
>> ctype="UU"
>> vartype="CC"
>> s=1
>> [xmin,f,status]=
    glpk(c,a,b,lb,ub,ctype,vartype,s)
xmin =
 NA
 NA
```

[6] Hier wird das Optimierungsproblem der Art geändert, dass die untere Grenze von $x_1 \leq 0, x_2 \leq 0 \Rightarrow x_1 \leq 1, x_2 \leq 0$ geändert wird

```
f  =  NA
status  =  10 %No  primal  feasible  solution.
```

Der Inhalt “NA” von *xmin* gibt an, dass keine optimale Lösung vorhanden ist. Eine Optimierungsproblem, bei dem das primale Problem nicht lösbar ist ein Optimierungsproblem mit leerem Lösungsraum.
Mit einem unbeschränktem Lösungsraum ergibt sich ein weiterer Fall, bei dem “NA” als Lösung raus kommt. Ein Beispiel für ein solches Optimierungsproblem sieht folgendermassen aus:

$$\min -3x_1 - 4x_2 \tag{28}$$
$$x_1, x_2 \geq 0 \tag{29}$$
$$x_1 \geq 3 \tag{30}$$
$$x_2 \geq 4 \tag{31}$$

```
>> c=[-3  -4]'
>> a=[1,  0;0,  1]
>> b=[3,4]
>> lb=[0,0]'
>> ub=[]
>> ctype="LL"
>> vartype="CC"
>> s=1
>> [xmin,f,status]=
    glpk(c,a,b,lb,ub,ctype,vartype,s)
xmin  =
 NA
 NA
f  =  NA
status  =  11 %No  dual  feasible  solution.
```

Der Unterschied zum vorherigen Optimierungsproblem ist hier, dass das duale Problem des Optimierungsproblem eine optimale Lösung hat. Es ist möglich zu beweisen, dass dieser Fall nur eintritt, wenn das primale Optimierungsproblem unbeschränkt ist.