# Pakete & JuMP

## Grundlagen Operations Research

Auf dieser Seite bauen wir Schritt für Schritt ein vollständiges Optimierungsmodell in **JuMP** auf, vom leeren Modell bis zur ausgewerteten Lösung. Als Beispiel dient das Produktionsproblem aus Vorlesung 1:

$$\begin{aligned} \text{maximiere } & F = 20X_1 + 30X_2 \\ \text{u.d.N.} \quad & X_1 + X_2 \leq 5 \\ & X_1 + 3X_2 \leq 9 \\ & X_1, X_2 \geq 0 \end{aligned}$$

## Pakete laden

Julia-Funktionalität steckt in **Paketen**. Installiert haben Sie die Kurs-Pakete bereits in Installation & Setup (einmalig, mit `] add JuMP HiGHS Plots`); in jedem Skript werden sie dann mit `using` geladen:

```
using JuMP, HiGHS
```

- **JuMP** ist die Modellierungssprache: Sie schreiben Variablen, Zielfunktion und Nebenbedingungen fast wie auf dem Papier.
- **HiGHS** ist der Solver, der das Modell tatsächlich löst, ein Open-Source-Verfahren, das den (dualen) Simplex-Algorithmus und Branch & Bound implementiert.

## Das Modell aufbauen

**Schritt 1: leeres Modell anlegen.** Dabei geben wir an, welcher Solver später rechnen soll:

```
modell = Model(HiGHS.Optimizer)
```

**Schritt 2: Entscheidungsvariablen deklarieren.** `@variable` legt eine Variable an; Schranken schreiben Sie direkt dazu:

```
@variable(modell, X1 >= 0)   # Abnahmemenge Auftrag 1
@variable(modell, X2 >= 0)   # Abnahmemenge Auftrag 2
```

**Schritt 3: Zielfunktion.** `Max` oder `Min`, danach der Term:

```
@objective(modell, Max, 20X1 + 30X2)   # Umsatzerlöse in EUR
```

**Schritt 4: Nebenbedingungen.** Jede Nebenbedingung bekommt einen Namen (hier `Paletten` und `Gewicht`), so können wir später gezielt auf sie zugreifen:

```
@constraint(modell, Paletten, X1 + X2 <= 5)    # max. 5 Paletten
@constraint(modell, Gewicht,  X1 + 3X2 <= 9)   # max. 9 Tonnen
```

Mit `print(modell)` können Sie jederzeit prüfen, ob das Modell so aussieht wie auf dem Papier:

```
print(modell)
```

```
Max 20 X1 + 30 X2
Subject to
 Paletten : X1 + X2 ≤ 5
 Gewicht : X1 + 3 X2 ≤ 9
 X1 ≥ 0
 X2 ≥ 0
```

## Lösen und die Solver-Ausgabe lesen

`optimize!` übergibt das Modell an HiGHS. Beim ersten Mal lassen wir die Solver-Ausgabe bewusst eingeschaltet:

```
optimize!(modell)
```

```
Running HiGHS 1.15.1 (git hash: 04024d701f): Copyright (c) 2026 under MIT
licence terms
Includes third-party software components, see THIRD_PARTY_NOTICES.md for full
details
Using BLAS: Apple Accelerate
LP has 2 rows; 2 cols; 4 nonzeros
Coefficient ranges:
  Matrix  [1e+00, 3e+00]
  Cost    [2e+01, 3e+01]
  Bound   [0e+00, 0e+00]
  RHS     [5e+00, 9e+00]
Presolving model
2 rows, 2 cols, 4 nonzeros 0s
2 rows, 2 cols, 4 nonzeros 0s
Presolve reductions: rows 2(-0); columns 2(-0); nonzeros 4(-0) - Not reduced
Problem not reduced by presolve: solving the LP
Using dual simplex solver
  Iteration        Objective     Infeasibilities num(sum)
          0    -4.9999960786e+01 Ph1: 2(6); Du: 2(50) 0.0s
          2     1.2000000000e+02 Pr: 0(0) 0.0s

Model status        : Optimal
```

```
Simplex   iterations: 2
Objective value     :  1.2000000000e+02
P-D objective error :  0.0000000000e+00
HiGHS run time      :          0.00
```

Die wichtigsten Zeilen dieser Ausgabe:

- **Presolve**: HiGHS vereinfacht das Modell vor dem eigentlichen Lösen (entfernt z. B. überflüssige Zeilen und Spalten).
- **Simplex iterations**: so viele Basistausche hat der Simplex-Algorithmus gebraucht (vgl. Vorlesung 2).
- **Model status: Optimal**, es wurde eine optimale Lösung gefunden.
- **Objective value**: der optimale Zielfunktionswert $F^*$.

> Tipp
>
> Im Kurs schalten wir die Ausgabe fast immer mit `set_silent(modell)` ab. Direkt nach `Model(...)` aufgerufen, bleibt die Konsole sauber und nur unsere eigenen `println`-Zeilen erscheinen.

## Ergebnisse abfragen

Nach dem Lösen prüfen Sie **zuerst den Status** und lesen dann die Werte aus:

```
termination_status(modell)
```

```
OPTIMAL::TerminationStatusCode = 1
```

`OPTIMAL` heißt: Die Lösung ist optimal. Erst dann sind die folgenden Abfragen sinnvoll:

```
println("F* = ", objective_value(modell),
        " mit X1 = ", value(X1), ", X2 = ", value(X2))
```

```
F* = 120.0 mit X1 = 3.0, X2 = 2.0
```

Die drei Statuswerte, die Ihnen in diesem Kurs begegnen (vgl. Vorlesung 5):

| Status | Bedeutung |
|---|---|
| `OPTIMAL` | optimale Lösung gefunden |
| `INFEASIBLE` | das Modell ist unzulässig (kein zulässiger Punkt) |
| `DUAL_INFEASIBLE` | das Modell ist unbeschränkt (Zielfunktion wächst ohne Grenze) |

Bei Variablen-Containern wie `@variable(modell, X[1:4] >= 0)` liefert `value.(X)` (mit Broadcasting-Punkt) die Werte aller Variablen auf einmal.

## Das Muster für jedes Kurs-Modell

Alle Modelle des Kurses folgen genau diesem Aufbau:

```
using JuMP, HiGHS

modell = Model(HiGHS.Optimizer)
set_silent(modell)

@variable(modell, ...)      # 1. Entscheidungsvariablen
@objective(modell, ..., ...) # 2. Zielfunktion (Max/Min)
@constraint(modell, ...)    # 3. Nebenbedingungen

optimize!(modell)           # 4. Lösen

termination_status(modell)  # 5. Status prüfen, dann:
objective_value(modell)     #    Zielfunktionswert und
value.(...)                 #    Variablenwerte auslesen
```

## Wie geht es weiter?

Damit können Sie jede Julia-Zelle des Kurses lesen und selbst ausführen. Alle weiteren JuMP-Bausteine (Container, Summen-Nebenbedingungen, Sensitivitätswerte wie `shadow_price` und `reduced_cost`) finden Sie kompakt im Cheatsheet JuMP und in Aktion ab Übung 1.

## Bibliografie