Optimierung (B.Sc.)
Albert-Ludwigs-Universität Freiburg – Wintersemester 2026/27

Übung 1: Optimierungsprobleme formulieren und lösen mit CasADi

Prof. Moritz Diehl, Florian Messerer

LÖSUNG

In dieser Übung lernen wir, wie wir Optimierungsprobleme formulieren und sie am Computer mit Hilfe von CasADi lösen können.

Installation

Für die Übungen benutzen wir Python 3 sowie die Pakete NumPy, Matplotlib und CasADi. Mit dem Paketmanager `pip` können Sie diese mit

```
pip install numpy matplotlib casadi
```

installieren. Wenn Sie möchten, können Sie die Pakete in einer virtuellen Umgebung (`venv`) installieren, um sie von anderen Python-Projekten getrennt zu halten:

```
python3 -m venv venv          # venv anlegen (einmalig)
source venv/bin/activate      # venv aktivieren (Linux/macOS)
# venv\Scripts\activate       # venv aktivieren (Windows)
pip install numpy matplotlib casadi
```

Falls Sie einen anderen Paketmanager bevorzugen, können Sie auch diesen verwenden.

Aufgabe 1: Einführung in CasADi

CasADi[1] ist eine Open-Source-Toolbox zum Formulieren und Lösen von Optimierungsproblemen. Die zur Optimierung benötigten Ableitungen berechnet CasADi automatisch und exakt über Algorithmische Differenzierung (AD).[2] Zum Lösen übergibt CasADi das Problem an einen eigenständigen Löser (Solver). Der Open-Source-Solver IPOPT ist in jeder CasADi-Installation bereits enthalten.

In der Vorlesung haben wir nichtlineare Optimierungsprobleme (nichtlineare Programme, NLP) in der Form

$$\min_{x \in \Omega} \quad f(x) \tag{1}$$

kennengelernt, mit Entscheidungsvariablen x, Zielfunktion $f : \mathbb{R}^n \to \mathbb{R}$, und zulässiger Menge $\Omega \subseteq \mathbb{R}^n$. Bei beschränkten Optimierungsproblemen wird die zulässige Menge durch explizite Nebenbedingungen definiert. Wir verwenden hierfür die Definition

$$\Omega = \{x \in \mathbb{R}^n \mid g_{\text{lb}} \leq g(x) \leq g_{\text{ub}}\}, \tag{2}$$

mit vektorwertiger Funktion $g : \mathbb{R}^n \to \mathbb{R}^m$ sowie unteren und oberen Schranken $g_{\text{lb}}, g_{\text{ub}} \in \mathbb{R}^m$. Fehlende Schranken setzt man auf $\pm\infty$. Um eine Gleichungsbedingung $g(x) = 0$ zu definieren, setzt man $g_{\text{lb}} = g_{\text{ub}} = 0$.

[1] `https://web.casadi.org`
[2] Dazu lernen wir mehr in Vorlesung 3.

Wir können (1) und (2) auch direkt kombinieren:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g_{\text{lb}} \leq g(x) \leq g_{\text{ub}}. \end{aligned} \tag{3}$$

Hierbei ist "s.t." die Abkürzung für "subject to" ("unter der Bedingung"). Um in CasADi ein Optimierungsproblem zu formulieren, müssen wir einen symbolischen Vektor x erstellen und daraus die Funktionen $f(x)$ und $g(x)$ konstruieren. Zum Lösen des Problems müssen wir dann konkrete Werte für die Schranken g_{lb} und g_{ub} übergeben.

Beispiel: Wir betrachten als Beispiel das Optimierungsproblem

$$\begin{aligned} \min_{x_0, x_1 \in \mathbb{R}} \quad & (x_0 - 1)^2 + (x_1 - 2)^2 \\ \text{s.t.} \quad & x_0 + x_1 \leq 2. \end{aligned} \tag{4}$$

Hier ist

$$x = (x_0, x_1), \qquad f(x) = (x_0 - 1)^2 + (x_1 - 2)^2, \qquad g_{\text{lb}} = -\infty,$$
$$g(x) = x_0 + x_1, \qquad g_{\text{ub}} = 2.$$

In CasADi sieht das so aus (`casadi_intro.py`):

```
import casadi as ca

x = ca.SX.sym('x', 2)                          # Vektor symbolischer Variablen
f = (x[0] - 1)**2 + (x[1] - 2)**2              # Zielfunktion
g = x[0] + x[1]                                # Nebenbedingungsfunktion

nlp = {'x': x, 'f': f, 'g': g}                 # Problemkomponenten zusammenfassen
solver = ca.nlpsol('solver', 'ipopt', nlp)     # Solver erzeugen

lbg = -ca.inf                                  # Untere Schranke auf g(x)
ubg = 2                                        # Obere Schranke auf g(x)
sol = solver(lbg=lbg, ubg=ubg)                 # Problem loesen
print(sol['x'], sol['f'])                      # Loesung und optimaler Wert
```

Aufgaben:

1. Führen Sie `casadi_intro.py` aus. Schauen Sie sich die Ausgabe von IPOPT an. Woran erkennen Sie, dass der Solver erfolgreich war? Welche Lösung erhalten Sie?

   Die letzte Zeile der IPOPT-Ausgabe lautet `EXIT: Optimal Solution Found.` Die Lösung ist $x_0^* = 0.5$, $x_1^* = 1.5$ (bis auf eine kleine numerische Ungenauigkeit).

2. Was passiert, wenn Sie die Nebenbedingung zu $x_0 + x_1 \leq 4$ verändern?

   Die Lösung ist $x_0^* = 1$, $x_1^* = 2$, das Minimum der Zielfunktion ohne Nebenbedingung. Die Nebenbedingung ist an der Lösung nicht "aktiv" und hat daher keinen Einfluss.

3. Überlegen Sie sich eine weitere Veränderung der Nebenbedingung.
   Hinweis: Die bereitgestellte Visualisierungsfunktion unterstützt lineare Nebenbedingungen der Form $g(x) = a_1 x_1 + a_2 x_2$.

Aufgabe 2: Hängende Kette

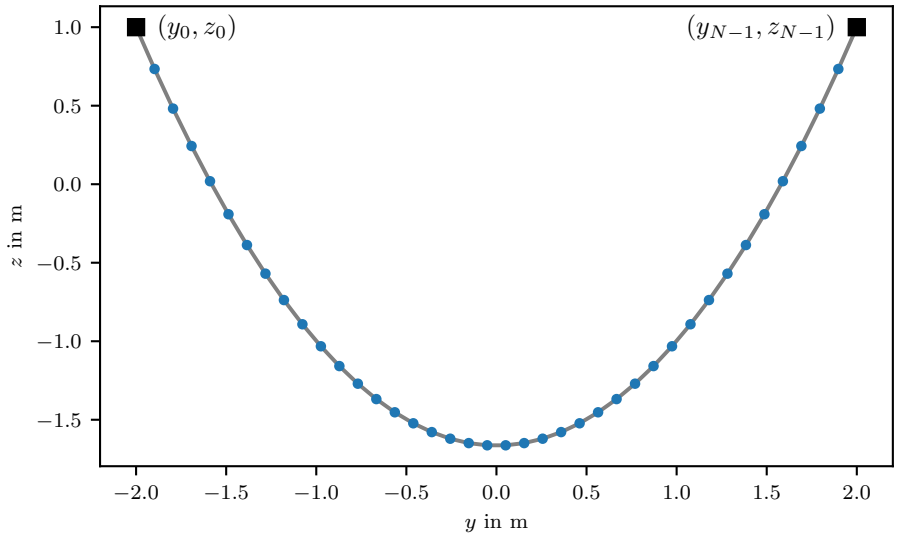


Abbildung 1 – Ruheposition der hängenden Kette (ohne Boden). Die äußersten Massepunkte (schwarz) sind fixiert.

Wir betrachten eine Kette, bestehend aus N Massepunkten, die durch $N-1$ Federn verbunden sind, siehe Abbildung 1. Die Massepunkte haben Masse m und Positionen (y_i, z_i), mit $i \in \{0, \dots, N-1\}$. Die beiden äußersten Massen sind an den Punkten $(\bar{y}_0, \bar{z}_0)$ bzw. $(\bar{y}_{N-1}, \bar{z}_{N-1})$ fixiert. Wir wollen die Ruheposition der Kette finden. Diese ist die Position, in der die potentielle Energie $V(y, z)$ der Kette minimal ist. Die potentielle Energie setzt sich zusammen aus der Lageenergie der Massen und der Energie der gespannten Federn:

$$V(y,z) = \sum_{i=0}^{N-1} m g_{\text{grav}} z_i + \frac{1}{2} \sum_{i=0}^{N-2} D \left((y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2 \right). \tag{5}$$

Hierbei ist g_{grav} die Erdbeschleunigung und D die Federkonstante. Wir verwenden $N = 40$, $m = 0.1\,\text{kg}$, $D = 70\,\text{N/m}$ und $g_{\text{grav}} = 9.81\,\text{m/s}^2$. Für die Endpunkte verwenden wir $\bar{y}_0 = -2\,\text{m}$, $\bar{z}_0 = 1\,\text{m}$, $\bar{y}_{N-1} = 2\,\text{m}$ und $\bar{z}_{N-1} = 1\,\text{m}$. Die Vektoren $y = (y_0, \dots, y_{N-1})$ und $z = (z_0, \dots, z_{N-1})$ fassen die Positionen aller Massepunkte zusammen.

Wir nehmen alle Positionen als Entscheidungsvariablen, $x = (y, z) \in \mathbb{R}^{2N}$, und fixieren die äußeren Punkte durch Gleichungsbedingungen:

$$\begin{aligned}
\min_{y,\, z \,\in\, \mathbb{R}^N} \quad & V(y,z) && \text{(6a)}\\
\text{s.t.} \quad & y_0 = \bar{y}_0, && \text{(6b)}\\
& z_0 = \bar{z}_0, && \text{(6c)}\\
& y_{N-1} = \bar{y}_{N-1}, && \text{(6d)}\\
& z_{N-1} = \bar{z}_{N-1}. && \text{(6e)}
\end{aligned}$$

Aufgaben (Teil A):

1. Wie viele Entscheidungsvariablen und wie viele Nebenbedingungen hat Problem (6)? Wie sehen $g(x)$, g_{lb} und g_{ub} aus?

   $2N = 80$ Entscheidungsvariablen und 4 Gleichungsbedingungen, mit $g(x) = (y_0, z_0, y_{N-1}, z_{N-1})$ und $g_{\text{lb}} = g_{\text{ub}} = (\bar{y}_0, \bar{z}_0, \bar{y}_{N-1}, \bar{z}_{N-1}) = (-2, 1, 2, 1)$.

2. Implementieren Sie in `chain.py` die Zielfunktion $V(y, z)$.
   Hinweis: Sie können die Summen entweder durch for-loops implementieren, oder die Funktionen ***`ca.sum`*** *(Summe der Elemente eines Vektors),* ***`ca.sumsqr`*** *(Summe der Quadrate) und* ***`ca.diff`*** *(Differenzen aufeinanderfolgender Elemente) nutzen.*

```
V_grav = m * g_grav * ca.sum(z)
V_spring = 0.5 * D * (ca.sumsqr(ca.diff(y)) + ca.sumsqr(ca.diff(z)))
V = V_grav + V_spring
```

3. Implementieren Sie die Nebenbedingungen.
   Tipp: ***`y[-1]`*** *ist das letzte Element von* ***`y`****, also* y_{N-1}.

```
g.append(z[0])
lbg.append(1)
ubg.append(1)
g.append(y[-1])
lbg.append(2)
ubg.append(2)
g.append(z[-1])
lbg.append(1)
ubg.append(1)
```

4. Führen Sie das Skript aus und betrachten Sie den Plot der Lösung. Entspricht die Form der Kette Ihrer Erwartung? Welchen optimalen Wert V^* hat die Zielfunktion?

   Die Kette hängt symmetrisch durch, der tiefste Punkt liegt bei $z \approx -1.66$. Der optimale Wert ist $V^* \approx 19.64$.

5. Was passiert, wenn Sie die Federkonstante D verdoppeln bzw. halbieren? Und was, wenn Sie die Masse m verändern?

   Größeres D (steifere Federn) bzw. kleineres m: die Kette hängt weniger durch. Kleineres D bzw. größeres m: die Kette hängt stärker durch.

Unter der Kette befindet sich nun ein ebener Boden, der die Kette blockiert. Dies drücken wir durch die *Ungleichungsbedingungen* $z_i \geq 0$ aus. Wir erhalten das Optimierungsproblem

$$\begin{aligned}
\min_{y,\, z \in \mathbb{R}^N} \quad & V(y, z) && && \text{(7a)}\\
\text{s.t.} \quad & y_0 = \bar{y}_0, && && \text{(7b)}\\
& z_0 = \bar{z}_0, && && \text{(7c)}\\
& y_{N-1} = \bar{y}_{N-1}, && && \text{(7d)}\\
& z_{N-1} = \bar{z}_{N-1}, && && \text{(7e)}\\
& z_i \geq 0, && i = 0, \ldots, N-1. && \text{(7f)}
\end{aligned}$$

Aufgaben (Teil B):

1. Wie müssen wir $g(x)$, g_{lb} und g_{ub} erweitern, um die neuen Nebenbedingungen zu berücksichtigen?

   Wir hängen z an die Nebenbedingungsfunktion an: $g(x) = (y_0, z_0, y_{N-1}, z_{N-1}, z)$. Die zugehörigen Schranken sind $g_{\text{lb}} = (\bar{y}_0, \bar{z}_0, \bar{y}_{N-1}, \bar{z}_{N-1}, 0, \ldots, 0)$ und $g_{\text{ub}} = (\bar{y}_0, \bar{z}_0, \bar{y}_{N-1}, \bar{z}_{N-1}, \infty, \ldots, \infty)$, jeweils mit N Einträgen 0 bzw. ∞ für z.

2. Vervollständigen Sie Teil B in `chain.py` und lösen Sie das Problem. Vergleichen Sie die Lösung mit der aus Teil A.

```
g.append(z)
lbg.append(np.zeros(N))
ubg.append(np.inf * np.ones(N))
nlp_B = {'x': x, 'f': V, 'g': ca.veccat(*g)}
solver_B = ca.nlpsol('solver', 'ipopt', nlp_B)
sol = solver_B(lbg=ca.veccat(*lbg), ubg=ca.veccat(*ubg))
```

Da sich die Funktion g geändert hat, muss ein neuer Solver erzeugt werden. Die Kette liegt im mittleren Bereich flach auf dem Boden, dort gilt $z_i = 0$. Die äußeren Massepunkte hängen frei. Der optimale Wert $V^* \approx 30.98$ ist größer als in Teil A.

3. Fügen Sie weitere Nebenbedingung Ihrer Wahl hinzu, z.B. eine Schräge. Sie können auch nichtlineare Funktionen benutzen, z.B. um einen Hügel zu modellieren.

4. Kann es passieren, dass das Hinzufügen einer Nebenbedingung den optimalen Wert verkleinert?

   Nein. Beim Hinzufügen von Nebenbedingungen wird die Menge, über die minimiert wird, kleiner (oder bleibt gleich), der optimale Wert kann also nur größer werden (oder gleich bleiben).

Aufgabe 3: Energiegemeinschaft

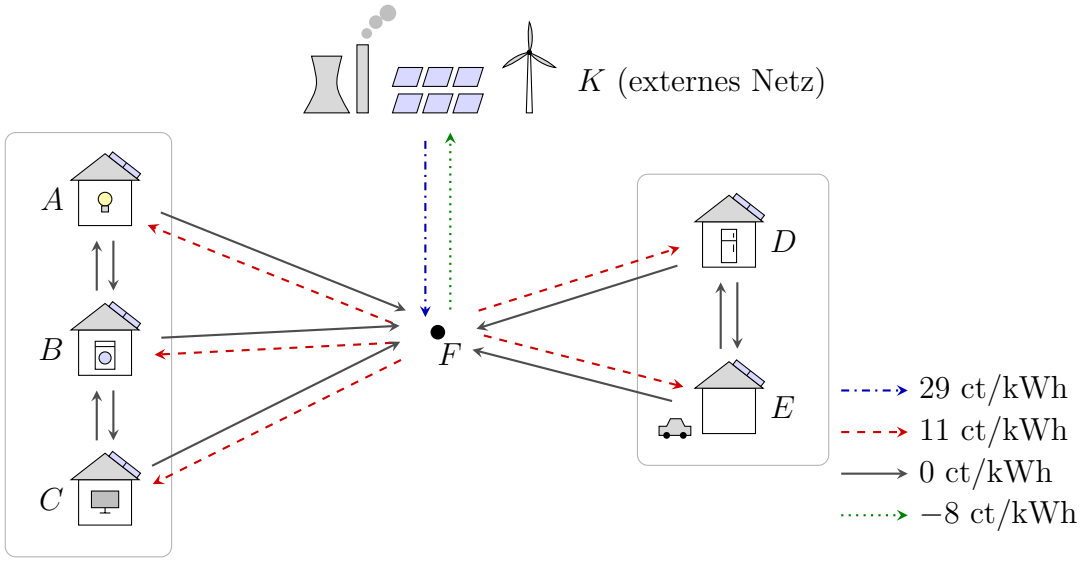


Abbildung 2 – Schema der Energiegemeinschaft. Jeder Haushalt hat einen Energiebedarf, aber erzeugt auch Strom durch PV-Module (Solarstrom). Die Haushalte können sowohl Strom vom Netz beziehen als auch untereinander verteilen.

Wir betrachten eine Energiegemeinschaft aus fünf Haushalten A bis E, siehe Abbildung 2. Die Haushalte sind über den Knotenpunkt F verbunden, der außerdem am externen Stromnetz K angeschlossen ist. Die Haushalte A, B, C sowie D, E sind zusätzlich direkt durch private Leitungen verbunden.

Wir betrachten einen Zeitraum von einer Stunde. Jeder Haushalt i hat einen Strombedarf und eine Photovoltaikanlage (PV). Die Differenz aus Bedarf und Solarstrom ergibt die Residuallast r_i. Negative Last, $r_i < 0$, entspricht einem Stromüberschuss. In der betrachteten Stunde ist die Residuallast der Haushalte wie folgt gegeben:

Haushalt i	A	B	C	D	E
r_i in kWh	-4	1	1.5	3	-0.5

Wir möchten die Stromflüsse so wählen, dass die Gesamtkosten der Gemeinschaft minimal sind.

Wir nummerieren die 9 Leitungen in der Reihenfolge $AB, BC, DE, AF, BF, CF, DF, EF, KF$. Entscheidungsvariablen sind die Stromflüsse in kWh, getrennt nach Richtung: $x_{\text{fwd}} = (x_{AB}, x_{BC}, \ldots, x_{KF}) \in \mathbb{R}^9$ in Vorwärtsrichtung ($A \to B$, …, $K \to F$) und $x_{\text{bwd}} = (x_{BA}, x_{CB}, \ldots, x_{FK}) \in \mathbb{R}^9$ in Rückwärtsrichtung ($B \to A$, …, $F \to K$).

Kosten. Einkauf aus dem Netz ($K \to F$) kostet $c_{\text{kauf}} = 29\,\text{ct/kWh}$, Verkauf ($F \to K$) bringt $c_{\text{verkauf}} = 8\,\text{ct/kWh}$. Von F zu einem Haushalt fällt das Netzentgelt $c_{\text{netz}} = 11\,\text{ct/kWh}$ an, alle anderen Flüsse sind kostenlos. Die Gesamtkosten sind

$$f(x) = c_{\text{fwd}}^\top x_{\text{fwd}} + c_{\text{bwd}}^\top x_{\text{bwd}} \tag{8}$$

mit $c_{\text{fwd}} = (0, \ldots, 0, c_{\text{kauf}})$ und $c_{\text{bwd}} = (0, 0, 0, c_{\text{netz}}, \ldots, c_{\text{netz}}, -c_{\text{verkauf}})$.

Bilanzen. Jeder Haushalt muss seinen Bedarf decken: Zufluss minus Abfluss muss mindestens r_i sein. Haushalt A ist über die Leitungen AB und AF verbunden. Für A lautet die Bedingung also

$$x_{BA} + x_{FA} - x_{AB} - x_{AF} \geq r_A. \tag{9}$$

Wir fassen die Bilanzen für alle Haushalte zusammen in der Ungleichung $M(x_{\text{fwd}} - x_{\text{bwd}}) \geq r$ mit $r = (r_A, \ldots, r_E)$ und Matrix $M \in \mathbb{R}^{5\times 9}$. Jede Zeile von M gehört zu einem Haushalt, jede Spalte zu einer Leitung. Der Eintrag ist -1, wenn die Leitung (in Vorwärtsrichtung) vom Haushalt wegführt, $+1$, wenn sie zu ihm hinführt, und sonst 0. Die ersten drei Zeilen sind bereits ausgefüllt, die erste Zeile entspricht (9):

$$M = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccccccc} AB & BC & DE & AF & BF & CF & DF & EF & KF \end{array} \\ \left[\begin{array}{ccccccccc} -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ & & & & & & & & \\ & & & & & & & & \end{array}\right] \end{array} \tag{10}$$

Im Knotenpunkt F muss der Zufluss exakt den Abfluss decken. Dementsprechend verlangen wir $m_F^\top(x_{\text{fwd}} - x_{\text{bwd}}) = 0$ mit Vektor $m_F \in \mathbb{R}^9$:

$$m_F^\top = \begin{array}{c} \begin{array}{ccccccccc} AB & BC & DE & AF & BF & CF & DF & EF & KF \end{array} \\ \left[\begin{array}{ccccccccc} & & & & & & & & \end{array}\right] \end{array} \tag{11}$$

Kapazitäten. Alle Flüsse sind positiv. Die privaten Leitungen übertragen höchstens $\bar{x} = 2\,\text{kWh}$ pro Richtung, die anderen sind unbeschränkt. Dies fassen wir in $x_{\text{ub}} = (\bar{x}, \bar{x}, \bar{x}, \infty, \ldots, \infty) \in \mathbb{R}^9$ zusammen.

Damit lautet das Optimierungsproblem

$$\begin{aligned} \min_{x_{\text{fwd}}, x_{\text{bwd}} \in \mathbb{R}^9} \quad & c_{\text{fwd}}^\top x_{\text{fwd}} + c_{\text{bwd}}^\top x_{\text{bwd}} && \text{(12a)} \\ \text{s.t.} \quad & M(x_{\text{fwd}} - x_{\text{bwd}}) \geq r, && \text{(12b)} \\ & m_F^\top(x_{\text{fwd}} - x_{\text{bwd}}) = 0, && \text{(12c)} \\ & 0 \leq x_{\text{fwd}} \leq x_{\text{ub}}, && \text{(12d)} \\ & 0 \leq x_{\text{bwd}} \leq x_{\text{ub}}. && \text{(12e)} \end{aligned}$$

Aufgaben:

1. Vervollständigen Sie M in (10) und m_F in (11).

$$M = \begin{bmatrix} -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$

$$m_F^\top = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

2. Bringen Sie (12) in die Form (3), indem Sie x, $f(x)$, $g(x)$, g_{lb} und g_{ub} definieren.

   $x = (x_{\text{fwd}}, x_{\text{bwd}})$, $f(x) = c_{\text{fwd}}^\top x_{\text{fwd}} + c_{\text{bwd}}^\top x_{\text{bwd}}$ und

$$g(x) = \begin{bmatrix} M(x_{\text{fwd}} - x_{\text{bwd}}) \\ m_F^\top (x_{\text{fwd}} - x_{\text{bwd}}) \\ x_{\text{fwd}} \\ x_{\text{bwd}} \end{bmatrix}, \qquad g_{\text{lb}} = \begin{bmatrix} r \\ 0 \\ \mathbf{0}_9 \\ \mathbf{0}_9 \end{bmatrix}, \qquad g_{\text{ub}} = \begin{bmatrix} \infty \cdot \mathbf{1}_5 \\ 0 \\ x_{\text{ub}} \\ x_{\text{ub}} \end{bmatrix},$$

   wobei $\mathbf{0}_n \in \mathbb{R}^n$ der Nullvektor und $\mathbf{1}_n \in \mathbb{R}^n$ der Vektor aus Einsen ist.
   Hinweis: wir haben in (12) eine kompakte Notation gewählt. Wenn wir die Matrix-Vektor-Produkte explizit ausmultiplizieren erhalten wir:

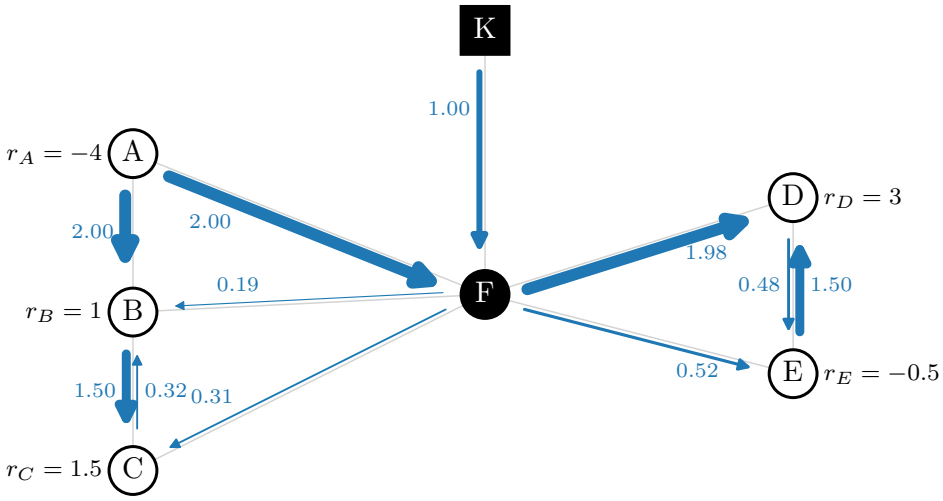
$$\begin{aligned} \min_{x_{\text{fwd}}, x_{\text{bwd}} \in \mathbb{R}^9} \quad & c_{\text{kauf}} x_{KF} + c_{\text{netz}}(x_{FA} + x_{FB} + x_{FC} + x_{FD} + x_{FE}) - c_{\text{verkauf}} x_{FK} \\ \text{s.t.} \quad & x_{BA} + x_{FA} - x_{AB} - x_{AF} \geq r_A, \\ & x_{AB} + x_{CB} + x_{FB} - x_{BA} - x_{BC} - x_{BF} \geq r_B, \\ & x_{BC} + x_{FC} - x_{CB} - x_{CF} \geq r_C, \\ & x_{ED} + x_{FD} - x_{DE} - x_{DF} \geq r_D, \\ & x_{DE} + x_{FE} - x_{ED} - x_{EF} \geq r_E, \\ & x_{AF} + x_{BF} + x_{CF} + x_{DF} + x_{EF} + x_{KF} \\ & -x_{FA} - x_{FB} - x_{FC} - x_{FD} - x_{FE} - x_{FK} = 0, \\ & 0 \leq x_{\text{fwd}} \leq x_{\text{ub}}, \\ & 0 \leq x_{\text{bwd}} \leq x_{\text{ub}}. \end{aligned}$$

3. Die Vorlage `energy_community.py` enthält bereits das vollständige Problem bis auf die fehlenden Einträge von M und m_F. Ergänzen Sie diese, lösen Sie das Problem und betrachten Sie den Plot. Welche Gesamtkosten ergeben sich? Erscheint Ihnen die Lösung sinnvoll? Fällt Ihnen etwas auf?

```
[0, 0, -1, 0, 0, 0, -1, 0, 0], # D
[0, 0, 1, 0, 0, 0, 0, -1, 0]]) # E
m_F = np.array([[0, 0, 0, 1, 1, 1, 1, 1, 1]]).T
```

   Die Gesamtkosten sind $f^* = 62\,\text{ct}$. Die Haushalte haben zusammen 1 kWh zu wenig, die aus dem Netz gekauft wird (29 ct). Über die privaten Leitungen wird so viel wie möglich kostenlos geteilt, für die restlichen 3 kWh, die von F an Haushalte fließen, fällt das Netzentgelt an (33 ct). Die Kosten sind also sinnvoll.
   Auffällig ist aber: Auf manchen Leitungen fließt gleichzeitig Strom in beide Richtungen (z.B. zwischen D und E), und E bezieht Strom von F, obwohl E einen Überschuss hat. Das liegt daran, dass die Lösung nicht eindeutig ist: Solche Flüsse ändern die Kosten nicht, da Hin- und Rückfluss sich aufheben bzw. das Netzentgelt für $F \to E$ statt $F \to D$ anfällt. Der optimale Wert f^* ist eindeutig, die optimalen Stromflüsse nicht. Je nach Solver bzw. Version kann der Plot daher anders aussehen.

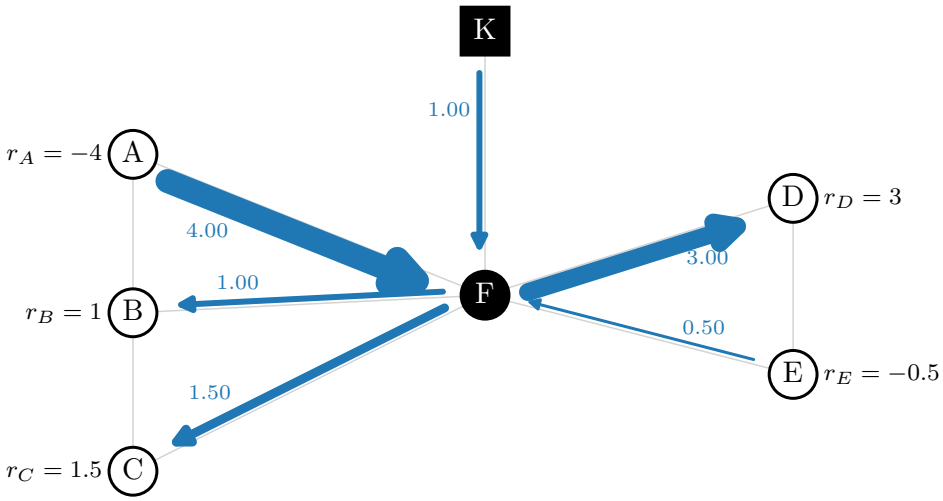
Optimale Stromflüsse in kWh, Kosten 62 ct



4. Wie verändern sich die Kosten, wenn es die privaten Leitungen nicht gäbe? Passen Sie den Code an und berechnen Sie die entsprechende Lösung.
   Hinweis: Es ist nur eine kleine Änderung der Nebenbedingungen notwendig.

   Wir setzen die Kapazität der privaten Leitungen auf null, also `x_max = 0`. Dadurch ändert sich nur die obere Schranke g_{ub}, nicht $g(x)$. Die Kosten steigen von 62 ct auf 89.5 ct. Ohne die privaten Leitungen muss der gesamte Austausch über F laufen, und für die 5.5 kWh Bedarf von B, C und D fällt das Netzentgelt an: $29 \cdot 1 + 11 \cdot 5.5 = 89.5$. Die privaten Leitungen sparen also das Netzentgelt für 2.5 kWh, d.h. 27.5 ct.

Ohne private Leitungen, Kosten 89.5 ct



5. Wie hängen die optimalen Kosten f^* von der Kapazität $\bar{x}$ der privaten Leitungen ab? Lösen Sie das Problem für viele (z.B. 51) Werte im Intervall $\bar{x} \in [0, 5]$ in einer Schleife und plotten

Sie f^* über $\bar{x}$. Erklären Sie den Verlauf der Kurve.
Hinweis: Zum Plotten können Sie die Funktion `plot_pareto_front` *verwenden.*

Da sich nur g_{ub} ändert, kann der Solver in der Schleife wiederverwendet werden. Die Kurve ist stückweise linear und fällt mit wachsendem $\bar{x}$ (mehr Kapazität kann die Kosten nie erhöhen). Jede kWh, die über private Leitungen statt über F fließt, spart das Netzentgelt von 11 ct. Für $\bar{x} \leq 0.5$ begrenzen zwei Leitungen: AB (Richtung $A \to B$) und DE (Richtung $E \to D$), beide werden voll ausgelastet. Die Kosten sinken daher mit $2 \cdot 11 = 22$ ct pro kWh. Ab $\bar{x} = 0.5$ fließt der gesamte Überschuss von E über DE, nur noch die Leitung AB begrenzt, und die Kosten sinken mit 11 ct pro kWh. Ab $\bar{x} = 2.5$ deckt der Fluss über AB den gesamten Bedarf von B und C, und zusätzliche Kapazität bringt nichts mehr: $f^* = 89.5 - 11 \cdot 3 = 56.5$ ct. Es bleiben nur der Zukauf von 1 kWh aus dem Netz und das Netzentgelt für die 2.5 kWh, die D über F bezieht. Die Kurve zeigt den Kompromiss zwischen Investition in Leitungskapazität und Betriebskosten.

