Praktikum 6

Markus Roos

06.02.2023

Aufgaben:

1 Industrieroboter

1.1 Lernziele

- Sie können ein einfaches System von nichtlinearen Gleichungen mit Hilfe des Newtonverfahrens lösen.
- Sie können mit Hilfe von Konvergenzbetrachtungen die Implementierung der Jacobimatrix kontrollieren.
- Sie sind in der Lage, dieses Lösungsverfahren als MATLAB oder PYTHON Routine zu implementieren, welche als Übergabeparameter minimal einen Handle auf das zu lösende Gleichungssystem (d.h. eine vektorwertige Funktion) und einen Handle zur Berechnung der Jacobimatrix (d.h. eine matrixwertige Funktion) aufweist. Hinweis: Sie werden diese Routine in künftigen Praktika wiederverwenden können. Achten Sie deshalb darauf, dass das eigentliche Lösungsverfahren bis auf die Parameterübergabe *unabhängig* von der Roboterarmproblemstellung ist!

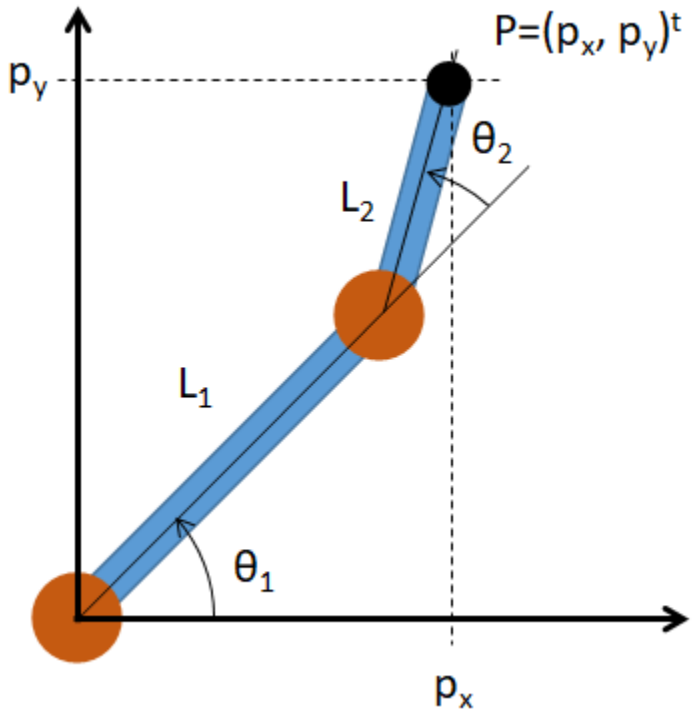
1.2 Theorie

Aufbau von Industrierobotern

Ein Industrieroboter ist eine universelle, programmierbare Maschine zur Handhabung, Montage oder Bearbeitung von Werkstücken. Diese Roboter sind z.B. für den Einsatz in der Automobilfertigung konzipiet. Der Industrieroboter besteht im Allgemeinen aus dem Manipulator (Roboterarm), der Steuerung und einem Effektor (Werkzeug, Greifer, etc.). Oft werden Roboter auch mit Sensoren ausgerüstet. Einmal programmiert ist die Maschine in der Lage, einen Arbeitsablauf autonom durchzuführen, oder die Ausführung der Aufgabe abhängig von Sensorinformationen in Grenzen zu variieren (aus wiki/Industrieroboter)

Einfaches Beispiel: 2-achsiger Roboterarm

Wir befassen uns mit der einfachsten nicht-trivalen Form eines Roboterarms bestehend aus 2 Gliedern. Diese können unabhängig voneinander gesteuert werden können und bewegen sich immer in derselben Ebene. Es handelt sich also um ein 2-achsiges System, das zwei Freiheitsgrade aufweist. Zur mathematischen Beschreibung müssen die Zustände dieses Systems parametriert werden, d.h. wir wählen zwei Winkel θ_1, θ_2, die eindeutig zu einer Position des Arms in der Ebene führen. Bei der Parametrierung hat man viel Freiheit und die optimale Wahl muss gut überdacht werden. In unserem Beispiel ist die Position des zweiten Armteiles relativ zur Orientierung des ersten Teilarmes gewählt. Neben den Orientierungen ist dieses System durch die fixen Längen der beiden Teilarme charakterisiert, d.h. $L_1 = 2\,m$ und $L_2 = 1\,m$.



Diese Parametrierung kann direkt technisch realisiert werden: der Aktuator im Gelenk zwischen den beiden Teilarmen kann die Orientierung des 2. Armes relativ zum 1. Arm einstellen, resp. messen. Eine Paramtrierung, welche auf absoluten Winkeln beruht, z.B. relativ zur x-Achse gemessen, könnte nicht direkt technisch umgesetzt werden.

Funktionale Abhängigkeiten

In einem ersten Schritt muss die Abhängigkeit des Zielpunktes (schwarzer Kreismittelpunkt) als Funktion der Parameterwinkel θ_1, θ_2 bestimmt werden. Konkret suchen wir eine *vektorwertige* Funktion $\vec{p} = \vec{f}(\theta_1, \theta_2)$, welche das Manipulatorende am Punkt mit den kartesischen Koordinaten $P = (p_x, p_y)^t$ positioniert (dabei enthält diese Funktion neben den Winkeln auch die Längen der Arme als weitere, konstante Parameter).

Die Grundaufgabe der Robotersteuerung ist das Anfahren eines beliebigen Punktes $P = (p_x, p_y)^t$ in der Ebene (hier definiert durch dessen Ortsvektor). Mit anderen Worten, man sucht die beiden Winkel der gewählten Parametrierung als Funktion der Zielpunktkoordinaten. Im allgemeinen ist diese Aufgabe nicht geschlossen lösbar, d.h. man ist auf numerische Verfahren angewiesen. Mathematischer ausgedrückt: man sucht die Umkehrfunktion zur Parametrierungsfunktion $\vec{f}(\theta_1, \theta_2)$, nämlich $\vec{\theta} = \vec{g}(p_x, p_y)$ mit $\vec{\theta} = (\theta_1, \theta_2)^t$.

1.3 Aufträge

1. **Parametrierung**

   Bestimmen Sie die Funktion $\vec{p} = \vec{f}\,(\theta_1,\, \theta_2)$, wozu Sie nur etwas Trigonometrie und Vektorgeometrie benötigen.

2. **Steuerung**

   Entwerfen Sie einen Algorithmus zur Bestimmung der Funktion $(\theta_1, \theta_2)^t = \vec{g}\,(p_x, p_y)$ basierend auf dem Newtonverfahren für Systeme. Die Lösung soll numerisch effizient und genau sein, d.h. die eruierten Winkel sollen in mindestens 14 bit Auflösung berechnet werden. Testen Sie Ihren Algorithmus mit bekannten Werten. Dabei sollen Sie zusätzlich folgende Aspekte untersuchen:

   - **Konvergenzverhalten**: Zeichnen Sie die erzielte Genauigkeit in Abhängigkeit der Anzahl Iterationen auf (Hinweis. wählen Sie eine optimale Darstellungsart, d.h., logarithmisch oder halblogarithmisch). Welches Konvergenzverhalten erwarten Sie für Ihren Rechenvorgang? Experimentieren Sie mit willentlich eingebrachten Fehlern: Ändern Sie ein Vorzeichen eines Ausdrucks oder variieren Sie einen Vorfaktor. Wie ändert sich die Konvergenz? Was schliessen Sie daraus für die Entwicklung von effizienten Verfahren?
   - **Robustheit**: Untersuchen Sie die Robustheit des Algorithmus in Bezug auf die Wahl der Startwerte. Konvergiert das Verfahren immer? Gibt es eine effiziente Strategie zur Wahl der Startwerte? Was passiert, wenn Sie Punkte eingeben, für welche überhaupt keine Lösung exisitiert. Wie sind die Punkte *mit* exisitierender Lösung charakterisiert?
   - **Ein- oder Mehrdeutigkeit der Lösung**: Ist die Lösung eindeutig? Konkret: sind die Lösungswinkel für einen gegebenen Punkt $P = (p_x, p_y)^t$ immer identisch, unabhängig vom Startwert?

3. **Abfahren einer Trajektorie**

   Nun soll auf der Basis des Algorithmus unter 2. mit dem schwarzen Punkt eine Bahn abgefahren werden: Diese Bahn entspricht einem Geradenstück in der parametrischen Darstellung $\vec{p}(t) = \vec{p}_0 + t\,\vec{d}$ mit $\vec{p}_0 = (-2, 1)$ und $\vec{d} = \frac{1}{\sqrt{17}}\,(1, -4)^t$. Der Geradenparameter t variiert dabei gemäss $t \in [0, 4]$ mit einer Auflösung von 0.01, alle Zahlenwerte werden in m gemessen.

   - **Graphische Darstellung**: Stellen Sie den Verlauf der beiden Winkel in Abhängigkeit des Geradenparameters t graphisch dar.
   - **Optimierung des Startwertes**: Wählen Sie den Startwert optimal aufgrund der Struktur der vorliegenden Aufgabenstellung.

1.4 Abgabe

Geben Sie Ihre Bearbeitung der Aufträge bis zu den Praktikumslektionen in der kommenden Woche ab.

2 Template für Roboterarm Movie

```python
# Python Template fuer Movie

import matplotlib
matplotlib.use("Agg")
from matplotlib.animation import FFMpegWriter

metadata = dict(title='Trajektorie', artist='Your Name',
                comment='Movie')
writer = FFMpegWriter(fps=60, metadata=metadata)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
fig = plt.figure(figsize=(6,6))
l1, = plt.plot([], [])
l2, = plt.plot([], [],'o')

# p: Funktion zur Berechnung der Punkte auf der Trajektorie
# ti: wird auch für die Berechnung der Winkel benutzt
ti = np.linspace(0,4,int(4/.01+1))
plt.plot(*np.array([p(tii) for tii in ti]).T)

plt.xlim(-3,3)
plt.ylim(-3,3)
plt.gca().set_aspect(1)
plt.gca().add_patch(mpatches.Circle((0,0), 2-1,alpha=0.1))
plt.gca().add_patch(mpatches.Circle((0,0), 2+1,alpha=0.1))
plt.grid()
plt.xlabel('x')
plt.ylabel('y')

# si: Liste der Winkel fuer die Trajektorie
# PG: liefert Drehpunkte und Endpunkt des Roboters. Bsp:
#       array([[0.        , 0.        ],
#              [0.58856217, 1.91143783],
#              [1.        , 1.        ]])
with writer.saving(fig, 'Trajektorie.mp4',400):
    for s in si:
        l1.set_data(*PG(*s).T)
        l2.set_data(*PG(*s).T)

        writer.grab_frame()
```

Downloads:

- PDF-Dokumentation:
  - Anleitung Praktikum 6
  - Python Movie Template