SDC und Reinforcement Learning

16. Februar 2021

1 Aufgabenstellung

Ziel: Löse das Kollokationsproblem

$$C_f u = u_0, \tag{1}$$

mit Kollokationsoperator

$$C_f(u) := (I_M \otimes I_N - \Delta t(Q \otimes I_N)f)(u), \tag{2}$$

so effizient wie möglich.

Ansatz: Benutze den iterativen Löser SDC (eine vorkonditionierte Fixpunktiteration)

$$P_f(u^{k+1}) = P_f(u^k) + (u_0 - C_f(u^k)) \qquad \text{für } j = 0, 1, 2, ..., \tag{3}$$

mit

$$P_f(u) := (I_M \otimes I_N - \Delta t(Q_\Delta \otimes I_N)f)(u). \tag{4}$$

Frage: Wie ist die diagonale Matrix Q_Δ zu wählen um einen besonders guten Vorkonditioniere P_f zu erzeugen.
Wähle die Einträge $(q_\Delta)_{ii}$ von Q_Δ so, dass die Anzahl der Iterationen von (3) bezüglich einer vorgegebenen Fehler-Schranke minimiert werden.

Im einfachsten Fall ist u skalar und $f(u) = \lambda u$ wir nennen dies unsere Testgleichung und beschränken uns zunächst auf $\lambda \in [-100, 0]$.

2 Q_Δ konstant: eine erste triviale Lösung

Als erstes nehmen wir an, dass Q_Δ nicht von der Iterationszahl k abhängt, sondern lassen unseren Algorithmus für jedes Testbeispiel ein Q_Δ wählen und führen für dieses Beispiel alle Iterationen k mit dieser Wahl aus.

Wie vergleichen die Wahl unseres Algorithmus mit zwei anderen Möglichkeiten Q_Δ zu besetzen:

- RL (unsere Implementierung): Eine Episode entspricht mehreren SDC-Iterationen (3): Für jede Iteration wird der Reward um eins verringert. Die Episode wird beendet falls der Fehler eine vorgegebene Schranke (10^{-10}) unterschritten hat oder dies nach 50 Iterationen nicht der Fall ist.
- MIN (Referenzlösung): Wir sind uns selbst nicht sicher wo diese Zahlen für Q_Δ herkommen
- LU (Referenzlösung): Ist eine verbreitete Wahl von Q_Δ als untere Dreiecksmatrix

Wir benutzen unseren RL Agent nach verschieden intensivem Training (100k oder 1000k Schritte) um Testgeichungen mit verschiedenen $\lambda \in [-100, 0]$ zu Lösen und vergleichen das Ergebnis mit den zwei anderen Lösern:

Alg. und Länge Trainig	durchschn Anz. Iterationen	gef. Lösung für $(q_\Delta)_{11}$ $[min, max]$ $Mittel \pm Abw.$	gef. Lösung für $(q_\Delta)_{22}$ $[min, max]$ $Mittel \pm Abw.$	gef. Lösung für $(q_\Delta)_{33}$ $[min, max]$ $Mittel \pm Abw.$
RL 100k	29.46	$[0.329, 0.481]$ 0.423 ± 0.009	$[0.179, 0.276]$ 0.184 ± 0.013	$[0.0, 0.425]$ 0.401 ± 0.078
RL 1000k	15.22	$[0.255, 0.324]$ 0.318 ± 0.008	$[0.128, 0.14]$ 0.136 ± 0.004	$[0.302, 0.375]$ 0.34 ± 0.009
MIN	14.22	0.320	0.14	0.372
LU	11.53			

Die Tabelle lässt vermuten, dass die von RL gefundenen Werte für Q_Δ gegen die MIN-Lösung konvergieren. Die Abbildungen 1 und 2 zeigen Lernerfolge nach einem Training mit 100k beziehungsweise 1000k Schritten. Dargestellt sind für die Löser die Anzahl der benötigten Iterationen (auf der y-Achse) gegen verschiedene Testbeispiele (λ auf der x-Achse). Nach 1000k Trainingsschritten braucht unser Agent genau wie MIN im Mittel etwa 14 Iterationen.

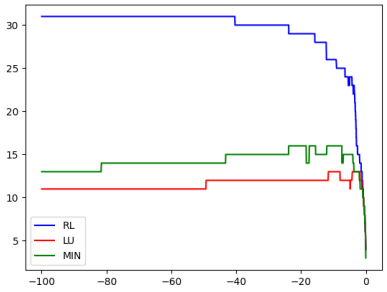

Abbildung 1: RL 100K

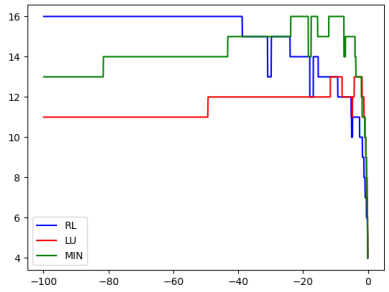

Abbildung 2: RL 1000K

3 Bestimme Q_Δ in jeder Iteration

Bestimmt man ein neues Q_Δ in JEDER Iteration, führt dies zu einer erheblichen Steigerung des Trainingsaufwands und bisher auch nicht zum Erfolg. Daher beschränken wir das Training und die Auswertung zunächst auf $\lambda \in [-10, 0]$ (dies sind auch die einfachen Testfälle).
(Für $\lambda = -20$ funktioniert das wirklich GAR NICHT aber vielleicht lerne ich auch nicht lange genug!)

Expertenrat: Benutze LSTM (unser Netzwerk war vorher vollständig verbunden) und finde eine passende Reward-Funktion:

- bisherige Belohnung: -1 für jede Iteration
- jetzt abhängig vom Residuum r^k und der gewünschten Genauigkeit $r_{tol} = 10^{-10}$ und der Anfangsgenauigkeit r^0

$$0.5 * \frac{log(r^k) - log(r^{k+1})}{log(r^0) - log(r_{tol})} - 0.01 \tag{5}$$

Die Abbildungen 3 und 4 zeigen das Lernen mit der alten Belohnungs-Methode (-1 für jede Iteration). Die Abbildungen 5,6,7,8 zeigen Ergebnisse für das Lernen mit der verbesserten Reward-Funktion (5), welche schneller zu besseren Ergebnissen führt. (Warum es irgendwann wieder schlechter wird habe ich als nicht RL-Experte noch nie verstanden).
Verwunderlich: Ist die Ergebnis-Matrix immer noch konstant?

Die Matrix Q_Δ, die von RL in Abbildung 6 (1000k) verwendet wird lautet: $[0.612 \pm 0.11, 0.307 \pm 0.075, 0.0252 \pm 0.0282]$.

Die Matrix Q_Δ, die von RL in Abbildung 7 (2000k) verwendet wird lautet: $[0.573 \pm 0.072, 0.281 \pm 0.047, 0.015 \pm 0.012]$.

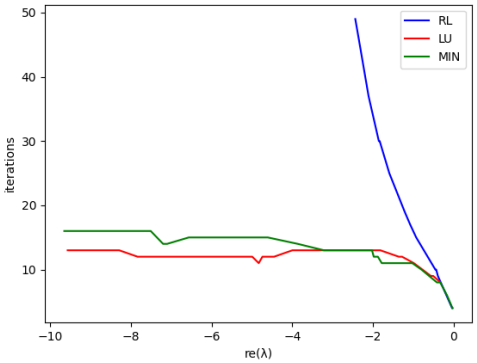

Abbildung 3: Reward wie bisher, 100k

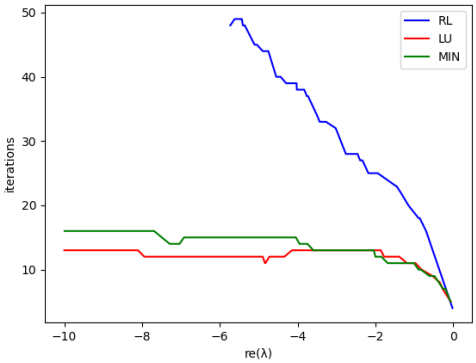

Abbildung 4: Reward wie bisher, 1000k

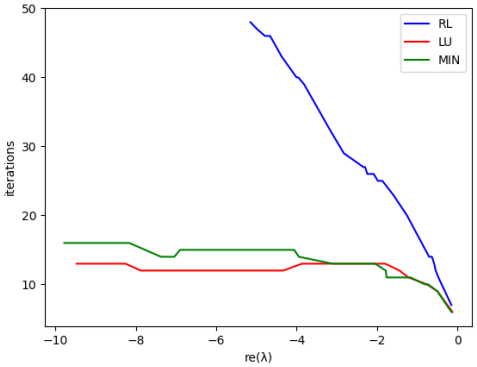

Abbildung 5: neuer Reward, 100k

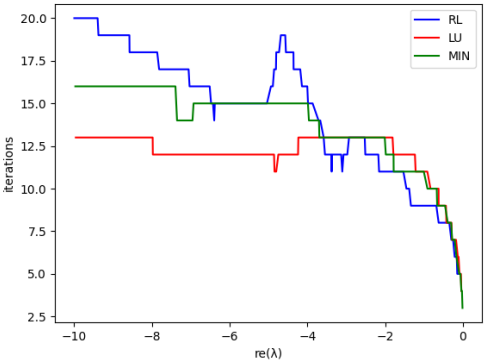

Abbildung 6: neuer Reward, 1000k

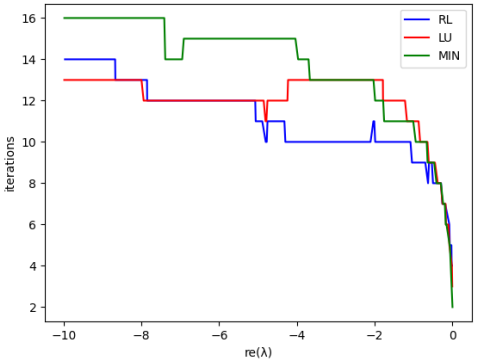

Abbildung 7: neuer Reward, 2000k

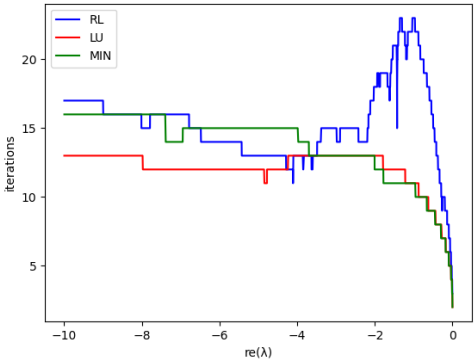

Abbildung 8: neuer Reward, 5000k

4 Was noch nicht klappt

Als nächstes untersuchen wir ein schweres Beispiel nämlich den Fall $\lambda = -20$ (alles mit betragsmäßig großen λ bereitet dem RL Agent bei schrittweiser Neuwahl von Q_Δ Probleme). Ziel ist erst mal nur mit diesem einen Beispiel zu trainieren und es zu Lösen. Dazu soll das Residuum kleiner als 10^{-2} werden (bei den letzten Beispielen wurde $< 10^{-10}$ gefordert).

Verwende ich die Werte von MIN denn steigt das Residuum zunächst von 20 auf 25 bevor es dann in insgesamt 6 Schritten unter die gegebene Schranke fällt. Das MUSS besser gehen!

Ansatz: Härtere Bestrafung/Abbruch bei Residuums-Vergrößerung!

Problem: Nach 1000k Iterationen ist RL immer noch deutlich schlechter als die MIN-Lösung!!!?