

SDC and Reinforcement Learning

16. Februar 2021

1 Aufgabenstellung

Ziel: Löse das Kollokationsproblems

$$C_f u = u_0, \quad (1)$$

mit Kollokationsoperator

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

so effizient wie möglich.

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

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

mit

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

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 Was bisher funktioniert: Konstante Matrix Q_Δ

- 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 (Referenzlöser): Wir sind uns selbst nicht sicher wo diese Zahlen für Q_Δ herkommen

- LU (Referenzlöser): Ist eine verbreitete Wahl von Q_Δ als untere Dreiecksmatrix

Wir benutzen unseren RL Agent (nach verschieden intensivem Training) um Testgleichungen 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 $(q_\Delta)_{11}$ [<i>min, max</i>] <i>Mittel \pm Abw.</i>	gef. Lösung für $(q_\Delta)_{22}$ [<i>min, max</i>] <i>Mittel \pm Abw.</i>	gef. Lösung für $(q_\Delta)_{33}$ [<i>min, max</i>] <i>Mittel \pm Abw.</i>
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 von RL gefundenen Werte für Q_Δ konvergieren gegen die MIN-Lösung (glaube ich).

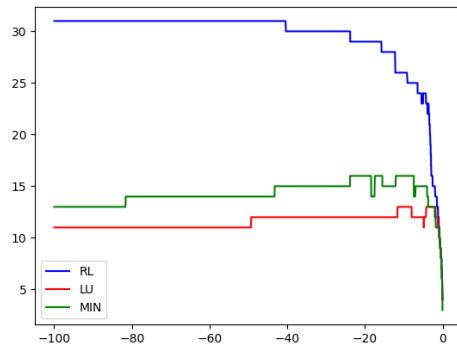


Abbildung 1: RL 100K

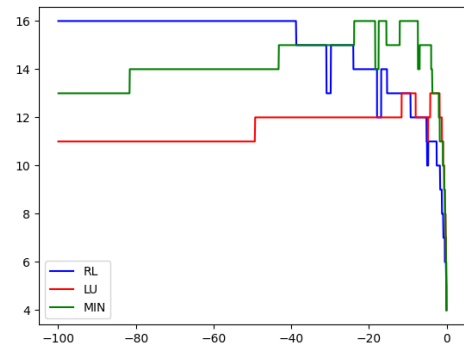


Abbildung 2: RL 1000K

3 Bestimme Q_Δ in jeder Iteration

Beschränke dazu das Training und die Auswertung auf $\lambda \in [-10, 0]$. (Für $\lambda = -20$ funktioniert das wirklich GAR NICHT aber vielleicht lerne ich auch nicht lange genug!)

Expertenrat: Benutze jetzt LSTM (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

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

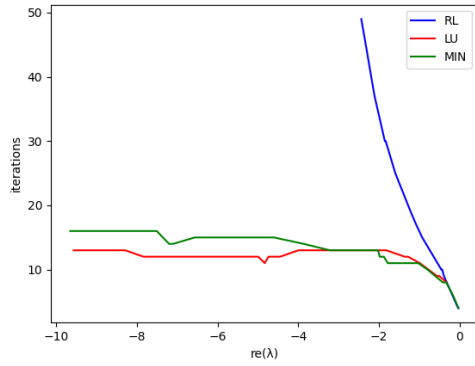


Abbildung 3: Reward wie bisher, 100k

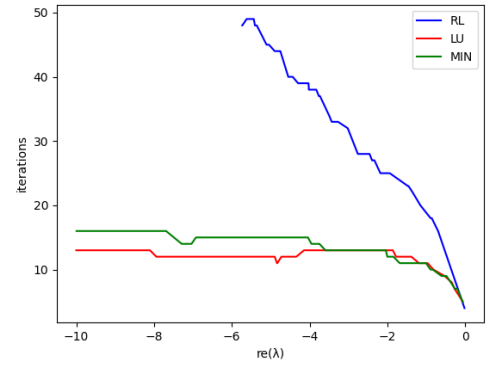


Abbildung 4: Reward wie bisher, 1000k

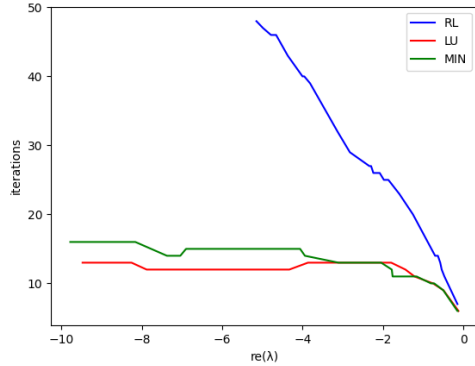


Abbildung 5: neuer Reward, 100k

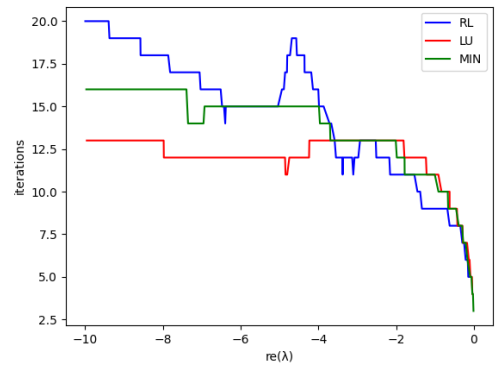


Abbildung 6: neuer Reward, 1000k

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 $[0.573 \pm 0.072, 0.281 \pm 0.047, 0.015 \pm 0.012]$.

Die Matrix Q_{Δ} , die von RL in Abbildung 8 (5000k) verwendet wird lautet $[0.587 \pm 0.15, 0.377 \pm 0.224, 0.073 \pm 0.178]$.

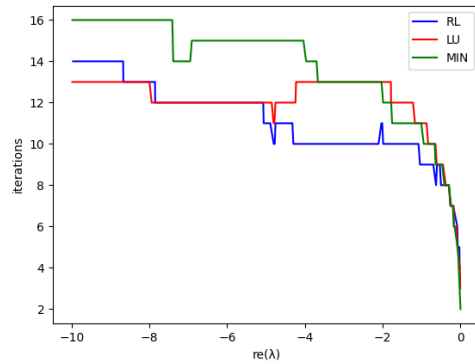


Abbildung 7: neuer Reward, 2000k

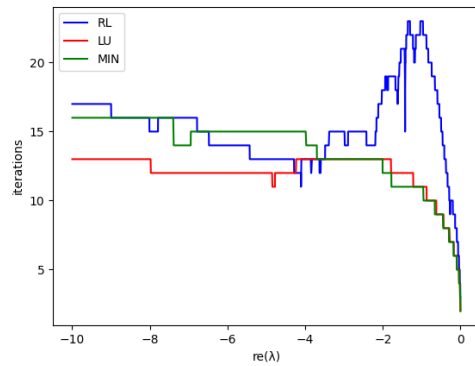


Abbildung 8: neuer Reward, 5000k

Abbildung 4,6,7,8 zeigen im Vergleich zu Abbildung 3 und 4: ein passender Reward kann zu besseren Ergebnissen führen. Mehr lernen führt zu besseren Ergebnissen wenn das Optimum überschritten wurde wird das Ergebnis wieder schlechter?

Ist die Ergebnis-Matrix konstant?

4 Schweres Beispiel: Untersuche $\lambda = -20$

Ziel Residuum kleiner als 10^{-2}

Verhalten MIN:

20.0

25.5911443268809

11.213730768763197

2.2076483639801614

0.10695687245110186

0.028374270509724964

$< 10^{-2}$

Achtung: in der zweiten Iteration steigt das Residuum. Geht das nicht besser? Härtere Bestrafung/Abbruch bei Residuumsvergrößerung!

Problem nach 1000k Iterationen ist RL immer noch schlechter als MIN (braucht mehr Iterationen zur Konvergenz)!!