SDC and Reinforcement Learning

15. Februar 2021

1 Aufgabenstellung

Ziel: Löse das Kollokationsproblems

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

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 Was bisher funtioniert: 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 beended 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 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 von RL gefundenen Werte für Q_Δ konvergieren gegen die MIN-Lösung (glaube ich).

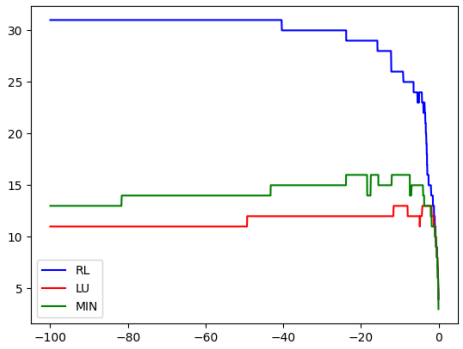

Abbildung 1: RL 100K

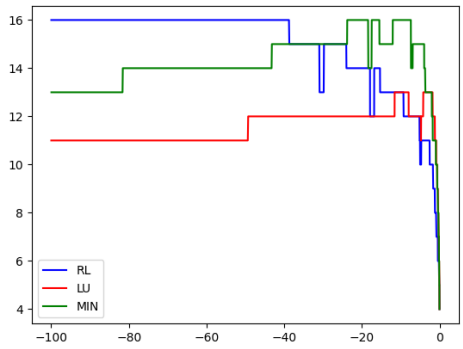

Abbildung 2: RL 1000K

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

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

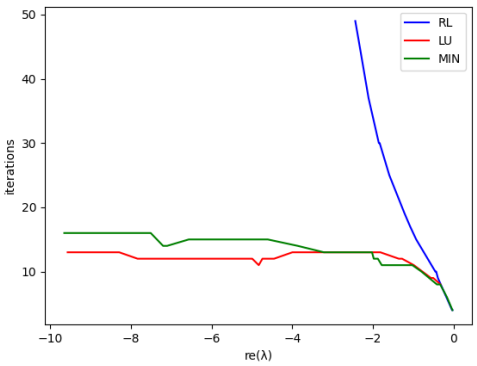

Abbildung 3: Reward wie bisher, 100K

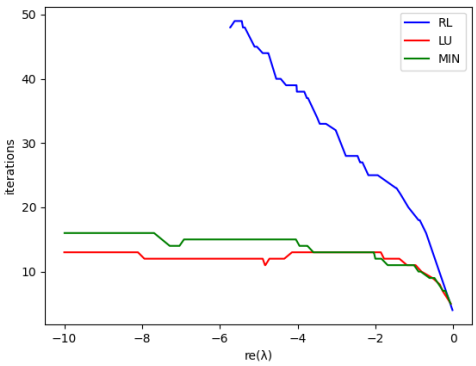

Abbildung 4: Reward wie bisher, 1000K

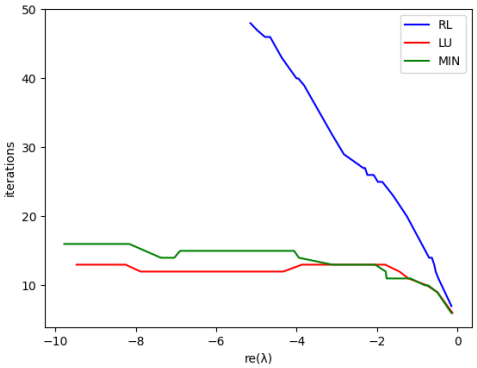

Abbildung 5: neuer Reward, 100K

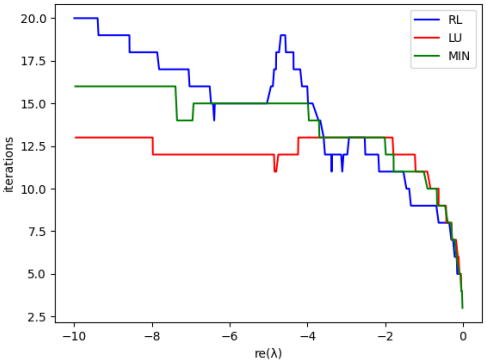

Abbildung 6: neuer Reward, 1000K

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