In dieser Sektion zeigen wir, wie die in den vorherigen numerischen Tests benutzten neuronale Netzwerke explizit abgeleitet werden können, damit man auf die Anwendung des automatischen Differenzierens in einem PINN verzichten kann. Wir vergleichen, die benötigte Zeit bis zur Konvergenz und die benötigten Computerressourcen eines in Python mit expliziten Ableitungen programmiertes PINN mit den zuvor erhaltenen Ergebnissen eines PINNs, welches natürlich automatische Differenzierung verwendet.

in Python mit expliziten Ableitungen programmiertes PINN

der weniger Ressourcen und Zeit benötigt

Dadurch soll ersichtlich werden ob eine explizite Berechnung der Ableitung für unsere Problemstellung Rechenvorteile mit sich bringt und sich dadurch die Performance eines PINNs verbessern lässt.

Wir bemerken, dass dies nur möglich ist, weil der Graph equilateral ist und somit auch rho für x in [0,1] auf jeder Kante definiert ist.

Natürlich sind damit nur die Gewichte und Biase dieses Netzwerkes zu lernen.

Im folgenden bezeichnet rho(z) die Modellvorhersage für den eingabewert z, welche gegeben ist durch

Kommen wir zurück auf unseren Fall in dem ein FNN rho die Lösung der Drift-Diffussionsgleichung auf einer Kante approximiert. Der Gradient ist aufgrund von Gleichung 3 und Gleichung 4 gegeben durch

Berechnung des Gradienten und einer Hesse von einem L-schichtigen feed-forwarard neural network.

Dafür kann die folgende iterative Berechnung verwendet werden

In dieser Arbeit haben wir zwei FNNs mit unterschiedlichen Topologien eingesetzt. Zuerst wurde ein in Sektion 1 ein FNN für alle Kanten auf den Graphen verwendet und dann wurde ein FNN für jede Kante eingesetzt. Dabei

Wir sind daran interessiert eine explizite Rechenformel für den Gradienten von dem für eine individuelle Kante eingesetzten Output eines dieser beiden neuronalen Netzwerke aufzustellen. Dafür berechnen wir zunächst die Ableitung erster Ordnung bzgl. eines vektoren eines allgemeinen FNN welches wir mit f bezeichnen.

Den Gradienten für eine individuelle erhalten wir im Fall von dem Netzwerk gegeben durch gleichung 3 durch die Berechnung der Jacobi

Dabei ist der analytische Grundgedanke der gleiche. Weswegen wir zunächst von allgemeinen Fall, welcher durch FNN1 gegeben ist, auf den Fall FFN2

Wir berechnen zunächst den Gradienten dieser Netzwerke Im folgenden bezeichne einfach rho den Output eine

Wir benutzen für die Berechnung der Jacobi Matrix die Kettenregel um Schicht für Schicht das Netzwerk abzuleiten.

Wir führen für den durch diese Kostenfunktion gegeben Ansatz numerische Experimente mit verschiedenen Arten neuronaler Netzwerke durch.

Im folgenden werden wir die dadurch erhaltenen netzwerke