Allgemeine Fragen:

* Das Ziel ist es, durch ein neurales Netzwerk eine Funktion zu approximieren, welche vom Ort auf dem Intervall der Kante und der Zeit abhängt, und die Lösung der Drift-Duffusion Gleichung ist, unter den Kirchhoff-Neumann-, Stetigkeits- und Influx-Outflux-Bedingungen?
* Was soll schneller gemacht werden (im Vergleich zur FEM von Max)? Die learning rate (Geschwindigkeit, die Gewichte und Biases zu lernen) oder der forward pass?
* Wir benutzen nicht backpropagation mit stochastic gradient zur Berechnung der weigths und biases?
* Was sind boundary vertices und inner vertices? („Kanten können sich nicht kreuzen, ohne einen Knoten zu bilden.“)
* Die Struktur vom metric graph geht über die boundary conditions bzgl. jeder Kante ein?
* Wir wollen den Gradienten der loss-Funktion (9) im Paper bzgl. Der Modellparameter \theta berechnen/bestimmen?

Jupyter - NetPINNs\_Drift-Diffusion:

* “The idea is to have one vanilla PINN for each edge which are connected via boundary and vertex conditions which are enforced weakly.”?
* Was muss ich alles über tensorflow.keras wissen?
* Was ist eine scaling layer?
* Was ist ein kernel.initializer?
* Gutes Paper für PINNs (mashine learning)?
* Was ist ein MultiDiGraph?
* PINN\_ResNet: wb, A, c? call() -> Forward-pass?
* Was sind dirichletNodes, dirichletAlpha, dirichletBeta, lb, ub (im Paper)?
* “Note that this matrix is not symmetric, as it belongs to a directed graph.” Warum nun ein gerichteter Graph?
* Muss ich etwas über tf.random wissen?
* Was sind die trainable\_variables?
* Was sind Xr, X0, Xl, Xu?
* Was macht \_setupVertexVariables?
* Ich verstehe die short-hand notation of mean-squared loss nicht wirklich?
* “Type conversion is necessary since scipy-routines are written in Fortran which requires 64-bit floats instead of 32-bit floats.”
* Warum der Adam-optimizer?
* Warum L-BFGS-B?
* Was sind unsere box-constraints?
* TF-Doku: “For efficiency, some ops (like ReLU) don't need to keep their intermediate results and they are pruned during the forward pass. However, if you use persistent=True on your tape, nothing is discarded and your peak memory usage will be higher.”
* Warum unterscheiden wir zwischen coarse oder nicht?
* Warum genau diese Optionen?
* Der Solver gibt uns als Ergebnis die Gewichte und biases? Mit diesen können wir dann ein NN definieren, welches die DGL löst?

Jupyter - Semiglobal\_optimal\_feedback\_final:

* Gutes Paper mit background Informationen?
* Wir benutzen ODE-Solver für PDEs?
* Wo passiert die „Magie“?
* Was ist LQR and PSE?
* “Function class representing feedback laws”?
* Was macht der main loop?

Paper:

* Weitere Erklärung zu: homogeneous Neumann–Kirchhoff conditions, flux boundary conditions (Dirichlet conditions?)
* “The model is then trained based on the minimization of a loss function \phi\_theta that incorporates the residual at a set of collocation points as well as a number of misfit terms which enforce both the boundary and initial conditions weakly.”
* “To be precise we assume that we identify a feedforward neural network \rho\_{\theta\_e} of the form (8) with the solution on an individual edge e \in \mathcal{E} where we have \theta\_e as the weights associated with the solution of the drift-diffusion equation defined on edge e \in \mathcal{E}.”

Nächste Schritte:

* Advanced automatic differentiation verwenden?
* Andere activation function (abhängig von der Schicht) benutzen -> ReLu, tanh?
* Einen anderen tensorflow optimizer (Adamax) vor LBFGS benutzen?
* Andere scipy omptimizer (abhängig von Constraints) benutzen?
* Berechnen des Gradienten von \phi bzgl. Modellparameter

27.10.2021

Fragen:

* Wir benutzen momentan für jede Kante ein ResNet, wobei die Architektur gegeben ist durch (29) in Ruthotto et. al? -> Es müssen wb, A, c, ResNet in PINN\_ResNet gelernt werden?
* Anzahl der Layer und Neuronen pro Layer? 1 Hidden-Layer, 16 Neuronen? Warum diese Architektur?
* Wie kommen wir auf die Schrittlänge?
* Wir wollen zur Berechnung der Gradienten Backpropagation nutzen? (So wie in Semiglobal\_optimal\_feedback\_final)
* Kann das überhaupt funktionieren? (Gradient von mehreren Netzwerken gleichzeitig)
* Macht tf.GradientTape das nicht eigentlich schon? (reverse accumulaton)
* In graphPINNSolver: self.trainable\_variables.append(self.graphPINN.NNs[i].trainable\_variables) ?
* <http://neuralnetworksanddeeplearning.com/chap2.html> (assumptions) -> Mittelwerte bzgl. Kanten nehmen? -> tf.reduce\_mean
* <https://stats.stackexchange.com/questions/505116/neural-network-derivative-with-respect-to-input> (derivative with respect to input) -> Ruthotto et. al (33)
* „Backpropagation learning does not require normalization of input vectors; however, normalization could improve performance.“
* Parameter in der Cost function?
* Gewichte in der Cost function (wie in (2.2) von Lu et. al)?
* Ableitung vom Betrag?
* Stochastic L-BFGS? (hier bei zufälliger Auswahl der Kollokationspunkte aufpassen, damit jedes NN jeder Kante trainiert wird -> aber wir nehmen wahrscheinlich eh immer full-batch da Datenmengen noch vernünftig sind, Raissi)
* Discrete Time Models?
* Die neuralen Netzwerke „kommunizieren“ über die Stetigkeitsbedingung an den Knoten (5)?
* Was ist nochmal over-fitting?
* Hättet Ihr ein Büro frei?
* Soll ich schonmal was schreiben?