Anfang:

* Ich heiße Sie herzlichst Willkommen.
* Mein Name ist Tom-Christian Riemer
* Digitale Verteidigung Masterarbeit.

Contents:

* Kurze Einführung weshalb man sich das anschauen sollte.
* Einige Definitionen und Beschreibung des Approximationsproblems.
* Vorstellung eines Ansatzes physics informed neural networks zur Approximation der Lösung zu verwenden.
* Betrachten Einsatz von expliziten Ableitungen.
* Und zum Schluss fasse ich nochmal die Ergebnisse zusammen und gebe einen kurzen Ausblick!

Introduction:

* Ich heize in 2 Wochen mit Gas
* Gas-Pipeline von den Jamal-Halbinseln nach Europa
* Druck mit der Länge der Leitung ab.
* Druck sollte stabil bleiben, d.h. in einem bestimmten Fenster bewegen.
* Dafür werden differential pressure transmitter und Verdichtungsmaschinen in verschiedenen Abständen verwendet
* Die Transmitter senden Daten an eine zentrale Stelle
* Die zentrale Stelle reguliert dann die Verdichtungsmaschinen.
* Wie wird das modelliert bzw. wie kommt man auf die richtigen Regulationsfaktoren?
* Wie viele Netzwerke in der realen Welt wird auch dieses mathematisch mit einem Graphen modelliert, doch weil die Länge der Pipeline eine entscheidende Größe ist, wird diese mit hinzu genommen. Man spricht dann von einem metrischen Grafen.
* Auf so einen Graphen kann man den Druck durch eine partielle Differentialgleichung modellieren.
* Ich bin kein Analytiker, deswegen habe ich keine Ahnung wie man auf eine analytische Lösung kommt.
* Aber ich bin ein bisschen in der Numerik bewandert und da gibt es verschiede Methoden die Lösung einer partiellen Differentialgleichung zu approximieren
* Eine recht frische Idee sind physics informed neural networks, in welchen die Information der Differentialgleichung in der Lernphase mit verwendet wird.
* Wir haben uns gefragt: Klappt das auch in dem Setup, wenn partielle Differentialgleichung auf einem metrischen Graphen definiert ist?

Graphen:

* Kombinatorische Graphen kennt jeder. Ein Tupel aus einer Knotenmenge und einer Kantenmenge
* Gerichteter Graph, jeder Kante ist eine Richtung zugeordnet, d.h. eindeutige Darstellung einer Kante mit einem Tupel von Knoten, ein Anfangsknoten ein Endknoten.
* Die Umkehrung einer Kante ist eine Kante mit entgegen gesetzter Richtung.
* Die Umkehrung ist reflexiv!
* Bei einem metrischen Graphen wird jeder gerichteten Kante eine positive Länge zugeordnet.
* Man kann sogenannte Koordinaten auf dem Intervall einer Kante betrachten, welche mit der Richtung wachsen.
* Definition einer Metrik ist möglich, hier für zwei Knoten, aber das ist auch auf zwei Koordinaten auf zwei unterschiedlichen Kanten erweiterbar.

Funktion spaces:

* Mit dieser Metrik ist man in der Lage verschiedene Funktionenräume zu definieren.
* Die Menge Lagrange-quadrat-integrierbaren Funktionen auf einer Kante.
* Lagrange-quadrat-integrierbaren Funktionen auf dem gesamten Graphen ist dann die Orthogonale Summe der Räume.
* Der Sobolev-Raum auf einer einzelnen Kante ist wie man ihn im Normalfall kennt.
* Beim Sobolev-Raum fordern wir nur noch, dass die Funktionen stetig aus dem gesamten Graphen sind, damit jede Funktion aus diesem Raum an einem Knoten auf allen an diesen Knotenpunkt angrenzenden Kanten denselben Wert an einem Knoten annimmt und somit eindeutig definiert ist.

Traffic flow:

* Karte vom Kaßberg
* Ein kompaktes Straßennetzwerk, welches aus einer endlichen Anzahl an Straßen endlicher Länge und einer endlichen Anzahl an Kreuzungen besteht.
* Das können wir mit einem sog. Kompakten metrischen Graphen modellieren. Endliche Anzahl Kanten mit endlicher Länge und endlicher Anzahl an Knoten.
* In den Fünfziger haben Herr Lighthill und Herr William und dazu unabhängig Herr Richards den Verkehrsfluss mit Gleichungen die den Fluss von Wasser beschreiben verknünpft.
* Die Grundidee dieses Ansatzes besteht darin, große Maßstäbe anzulegen, d. h. einzelne Autos als kleine Teilchen zu betrachten, eine Gruppe von Autos als Masse und ihre Dichte als die wichtigste zu berücksichtigende Größe.
* Dem Ansatz folgen wir auch.
* Es ist sinnvoll, von der Erhaltung der Anzahl der Autos auszugehen, was durch die folgende Kontinuitätsgleichung für jede einzelne Kante ausgedrückt werden kann, wobei J\_e den Strom von Autos beschreibt.

Flux of cars:

* Der Strom von Autos setzt sich zusammen einmal aus einem Diffussionsterm, welcher den Transport durch Diffusion beschreibt, der durch das erste Fick'sche Gesetz gegeben ist, Dafür braucht man noch einen typischerweise kleinen Diffussionskoeffizienten.
* Und der zweite Term ist ein Strömungsterm, welcher den Transport durch Strömung beschreibt. Dieser ist durch die Ableitung eines Potentials bzgl. der Ortskoordinate gegeben, dieses kann von Kante zu Kante unterschiedlich sein, und durch eine positive Mobilitätsfunktion.
* Wir wählen die Mobilität für unser Problem wie angegeben. Der Grund dafür ist, dass in vielen Anwendungen, wie auch die hier betrachtete, die Dichte aufgrund von Begrenzungseffekten eben begrenzt ist und wir mit dieser Wahl sicherstellen, dass die Lösung in diesem vorgegebenen Intervall bleibt.

Drift-Diffussion-equations on a metric graph.

* Die ausgangslage für unser Approximationsproblem ist, dass wir einen Metrischen Graphen haben auf jede Kante die Drift-Diffusion equation lösen möchten.
* Damit das ein wohlformuliertes Problem wird, brauchen wir noch Initial- und Randbedingungen.
* Die Initialbedingungen sind einfach, wir fordern, dass unsere Lösung zum Anfangszeit punkt gleich einer vorgegeben Langrange-quadrat-integrierbaren Funktion ist.
* Die Randbedingungen sind auf einem metrischen Graphen sogenannte Knotenbedingungen.
* Für die Definition müssen wir noch einen Normalenvektor für die Knoten einer Kante definieren. Dieser ist minus eins am Anfangsknoten und plus eins am Endknoten.

Conditions on interior vertices:

* Wir haben einmal Bedingungen für Knoten innerhalb eines Graphens, damit meinen wir knoten, die sowohl mindestens eine reinkommende wie auch eine rausgehende Kante anliegen haben.
* Wir fordern für diese Knoten einmal, dass sie Krichhoff-Neumann Bedingung erfüllen, diese fordert, dass der aufaddierte Strom am Knoten gleich null ist.
* Das bedeutet, dass in jeden inneren Knoten genau so viel Masse hineinfließt wie herausfließt.
* Des Weiteren wollen wir natürlich auch, dass die Lösungen stetig in den Knoten sind, dass heißt, dass die Lösungen, welche in einem Knoten aufeinander treffen auch den gleichen Funktionswert dort haben.

Conditions on interior vertices:

* Was sind äußere Knoten – da haben Jan, Martin und ich des Öfteren darüber diskutiert. Wir definieren diese nun so, dass nur entweder Kanten aus einem äußeren Knoten rausgehen oder Kanten in einen äußeren Knoten reingehen.
* Wir fordern auf diesen Knoten, dass die flux boundary condition gilt, welche durch den hier beschrieben Term definiert ist.
* Ich bemerke, dass Rho an der Stelle v entweder am Anfang oder am Ende des Intervalls ausgewertet wird.
* Alpha ist die Einflussrate und Beta ist die Ausflussrate des Systems.
* In typischen Situationen sind äußere Knoten nur vom Einfluss bzw. Ausflusstyp, das bedeutet das bei solchen Knoten das Produkt von alpha und Beta zu jedem Zeitpunkt gleich Null ist.
* Diese Bedingungen sorgen dafür, dass Masse nur über die äußeren Knoten, für die entweder Alpha bzw. Beta positiv ist, in das System hineinfließen bzw. hinausfließen.
* Bisher ist noch keine analytische Lösung bekannt (mir zumindestens).
* Aber Herr Prof. Pietschmann formuliert noch einen Existenzsatz.
* Wir betrachten die Lösung dieses Systems als unser Approximationspriblem.

Artificial neurons:

* In den letzten Jahren ist eine Methode zum Approximmieren verschiedener Abbildungen immer mehr ins Rampenlicht der angewandten Mathematik gerückt: nämlich Maschinelles Lernen.
* Das hat alles damit angefangen, dass man die Neuronen vom tierischen Gehirn mathematisch modellieren wollte.
* Das resultierte sogenannte Artificial Neurons. Hier bildet man mit dem Input, was irgendwelche Daten sein können eine Linearkombination, verschiebt diese mit einem Bias und wendet dann darauf eine sogenannte Aktivierungsfunktion an.
* Die Aufgabe der Aktivierungsfunktion besteht lediglich darin, aus der Linearkombination einen nichtlinearen Output zu. Daher kann jede nicht-lineare, nicht-konstante Funktion eine Aktivierungsfunktion sein.
* Aber sind die im Allgemeinen verwendeten Aktivierungsfunktionen zusätzlich noch monoton steigend, stetig und zumindesten stückweise glatt.
* Die bekannteste ist wahrscheinlich die rectified linear unit, kurz ReLU, aber es werden auch häufig der Tangens hyperbolicus oder die Sigmoid funktion verwendet.

Feed-Forward Neural Network:

* Leider ist aber ein einzelnes Neuron unfähig einen mehrdimensionalen Output zu erzeugen oder etwas komplexere Abbildungen zu approximieren.
* Es gibt viele verschiedene Arten Neuronen in irgendeiner Weise miteinander zu verbinden. Man spricht dann von einem neuronalen Netzwerk. Eines der Bekanntesten ist das sogenannte Feed-Forward Neural Network oder kurz FNN.
* Deshalb hat man mehrere Neuronen in Schichten organisiert, wobei die Neuronen einer Schicht ihren Output an alle Neuronen der Darauf folgenden Schicht weitergeben. Darauf leitet sich auch der Name ab, weil Information vorwärts durch das Netzwerk.
* Das ganze wird durch diese Iteration umgesetzt, wir bilden eine mehr-dimensionale Linearkombination mit Gewichts Matrizen und verschieben diesen mit einem mehrdimensionalen Bias und wenden darauf eine Aktivierungsfunktion komponentenweise an.
* Dabei sind die Anzahl der Schichten und die Anzahl der Neuronen pro Schicht hyperparameter die der Nutzer angibt.
* Die Gewichte und Biase sind sogenannte trainierbare Parameter, d.h. diese müssen durch eine Lernprozess noch angepasst werden. Und oben steht es schon
* Der Neuropsychologe Donald Hebb benutzte diesen Ausdruck erstmals 1949, um zu beschreiben, wie sich die Bahnen im Gehirn durch Wiederholung bilden und verstärken. Je öfter das Gehirn eine bestimmte Aufgabe ausführt, desto stärker wird das neuronale Netz und desto effizienter wird der Prozess bei jeder Wiederholung.

Machine Learning:

* Es gibt verschiedene Lernmethoden, die bekannteste ist supervised learning.
* Hier hat man eine Trainingsmenge an Daten, wo jedes Element ein Tupel aus einem Input einem gewünschten Output, z.B. einen zu approximierenden Funktionswert, besteht.
* Man definiert dann eine Kostenfunktion zum Beispiel mit dem Mean-squared-Error und minimiert das ganze bzgl. der trainierbaren Parameter.
* Das wird natürlich auch mit iterativen verfahren umgesetzt. Das meist verwendete ist eine Variante des Stochastic Gradient descent. Wobei man als Basis das Gradientenverfahren hat, aber man den Gradienten in jeder Iteration an einem zufällig ausgewählten Trainingspaar auswertet.
* Der Gradient wird dann mit Backpropagation berechnet.

PINNs:

* Gut, wie werden nun Neuronale Netzwerke zur Approximation der Lösung einer partiellen Differentialgleichung verwendet.
* 2017, also vor kurzen, aber noch vor der Pandemie, haben einen Methode vorgestellt, in welcher die Information einer Differentialgleichung, welche zum Beispiel ein Physikalisches System beschreibt, in die Lernphase eines Neuronalen Netzwerkes mit einarbeitet.
* Man hat dafür ein sog. Network, dessen Aufgabe es ist die Differentialgleichung in dem vorgegebenen Gebiet zu approximieren, und man setzt dieses surrogate network in die linke Seite der Differentialgleichung ein, und verwendet für die darin auftretenden partiellen Ableitungen Automatic Differentiation der jeweilig verwendeten Programmiersprache ein.

PINN Learning:

* Wie man auf diesem Bild sieht, sind die beiden Netzwerke miteinander verflochten und in der Lernphase minimiert man wieder eine Kostenfunktion bzgl. der trainierbaren Variablen des Surrogate Networks.
* Dafür betrachtet man den Mean-squared-error des Residual networks über einer Menge von Kollokationspunkten und falls man noch zusätzliche Nebenbedingungen hat, kann man diese durch einen entsprechenden Misfit-Term über einer Datenmenge der Kostenfunktion hinzufügen.
* Z.B. man hat eine Initialbedingung, dann kann man die entsprechenden Werte zum Anfang des Beobachtungszeitraums einfügen.

A PINN Approach:

* Alles klar, nach diesen vielen Definition können wir nun endlich einen PINN Ansatz für unser Approximationsproblem vorstellen.
* Was natürlich sofort klar wird, wie brauchen genauso viele surrogate Netzwerke wie wir kanten auf dem betrachteten Graphen haben.
* Dabei beschreibt theta\_e aber nur die trainierbaren Parameter, welche direkt für die Approximation auf der jeweiligen Kante verwendet werden.
* Unser Ansatz haben wir „All at once“-Approach genannt, weil wir bei diesem in der Lernphase eine einzige Kostenfunktion Phi\_theta betrachten, welche die Abweichung aller surrogate Networks zu der Menge der Drift-Diffussionsgleichungen auf jeder Kante, zu den Initialbedingungen auf jeder Kante und den Knotenbedingungen auf jedem Knoten.
* Dementsprechend wird unsere Kostenfunktion aus mehreren Termen bestehen und diese kontruieren wir jetzt.

Deviation tot he Drift-Diffussion equation:

* Die Abweichung des surrogate networks zur Drift-Diffussionsgleichung, da gehen wir straight-forward vor.
* Das ist die Differentialgleichung. Surrogate Network einsetzen. Gleichung umstellen und wir haben das residual network.
* Der Residual Misfit Term ist dann der Mean-suqared-error des residual networks über einer Menge kollokationspunkte.

Deviation to the initial condition:

* Hier das gleiche Spiel. Die Initialbedingungen sind gegeben durch die obere Gleichung.
* Der Initial Misfit Term für eine einzelne Kante ist gegeben durch den Mean-squared-Error dieser umgestellten Gleichung über einer Menge von Kollokationspunkten zum Zeitpunkt null.

Deviation to the Kirchhoff-Neumann condition:

* Die Kirchhoff-Neumann-Bedingung für einen inneren Knoten ist, dass die Summe des Strom multipliziert mit zugehörigen Normalnvektor gleich null ist.
* Hier passiert die Magie, indem wir einfach das Surrogate Network in den Strom stecken und dann den MSE über einer Menge von Time-Snapshots betrachten.

Deviation to the Continuity condition:

* Bei der Continuity Condition fordern wir, dass die Approximationen, die alle am gleichen Knoten verbunden sind, dort den gleichen Funktionswert haben.
* Der Continuity Misfit Term ist gegeben durch folgenden Ausdruck, wobei wir aber für jeden inneren Knoten nochmal genauso viele trainierbare Parameter wie kollokationspunkte hinzufügen.
* Und da wir durch die Minimierung dieses Terms erreichen, dass sich diese Trainierbaren Parameter an die Werte aller an diesem Knoten angebunden Surrogate Networks ausgewertet an den entsprechenden Kollokationspunkten annähert und umgekehrt die Surrogate Networks sich an die trainierbaren Parameter durch Anpassung ihrer eigenen trainierbaren Parameter annähern, kann man es schaffen, dass an diesen Kollokationspunkte die Werte aller Surrogate Netzwerke gleich werden.

Deviation to the flux boundary condition:

* Wie immer

Finale Kostenfunktion:

* Um jetzt nun die Kostenfunktion zu erstellen, dieeren wir die Misfit Terme an den betreffenden Einheiten auf, d.h. residual und initial Misfit Term für jede Kante, Kirchhoff-Neumann und Continuity misifit term für jeden inneren Knoten und den flux boundary misfit term für jeden äußeren Knoten und kombinieren sie in dieser Kostenfunktion.
* Die Minimierung bzgl. der dafür notwendigen trainierbaren Parameter sollte dann sicherstellen, dass die Surrogate Networks die Lösungen auf den Kanten unter den gegebenen Bedingungen zu einem bestimmten grad approximieren.
* Doch was ist jetzt eigentlich Theta und Theta\_e?

Folie 21:

* Was ist eigentlich theta und theta\_e?