**Sistema intelligente di difesa antiaerea per la simulazione, previsione e calcolo di traiettorie balistiche**

**Gruppo di lavoro**

**Studente:** Epifania Cristiano

**MAT.:** 766404

**Email:** c.epifania5@studenti.uniba.it

[**Repository**](https://github.com/CittiX/ICON-proj.git)

A.A. 2024-2025

Introduzione

Lo scopo di questo progetto è simulare un sistema intelligente per un dispositivo di difesa antiaerea, capace di prevedere e ottimizzare la precisione, l'affidabilità e l'efficienza delle traiettorie balistiche. Il sistema sarà in grado di gestire le traiettorie di missili nel raggio d'azione del dispositivo, adattandosi a variabili come peso del missile, spinta, resistenza aerodinamica, angolo di lancio, durata della spinta e gravità (quest’ultima costante), prevedendo e applicando principi di balistica.

Sommario

Questo progetto propone l’integrazione di un KBS (*Knowledge Based System*) per la previsione e il controllo delle traiettorie di missili balistici utilizzando reti neurali (non tradizionali, consultare il paragrafo di riferimento). Il KBS combina modelli fisici con l’apprendimento supervisionato per costruire un sistema intelligente in grado di fornire una traiettoria balistica predetta con la maggiore accuratezza possibile. Nel seguito verrà specificata l’architettura, comprendente simulazione e addestramento della rete (d’ora in poi chiamata **PINN**).

# Architettura

Il KBS si compone di due moduli principali:

1. Simulazione (*Simulation*)
2. Apprendimento supervisionato (*Supervised Learning)*

Questi moduli consentono la creazione di un dataset iniziale attraverso la simulazione del lancio di un missile con dati iniziali noti (nel seguito si mostrerà un adattamento della PINN alla non conoscenza dei dati) e al successivo apprendimento.

# Simulazione

Questo modulo integra la simulazione di un lancio di un missile con correlata creazione di dati e successiva aggiunta di rumore. Per entrambe le tipologie di dati, sarà possibile visualizzarne un grafico.

## Simulazione del missile

Le feature scelte per il missile sono le seguenti:

* *gravity*: ossia l’accelerazione gravitazionale, costante fisica con valore **9.81m/s2**;
* *drag\_coefficient*:ossia resistenza aerodinamica. Trattasi della forza opposta al moto del missile causata dall’attrito dell’aria. Questa inficia molto sul moto del missile, anticipando il momento in cui il missile collide con il terreno. Attualmente impostato a **3kg/s**;
* *launch\_angle*: ossia l’angolo di lancio del missile, impostato attualmente a **85rad**;
* *initial\_thrust*: ossia la spinta (boost) iniziale del missile. Un valore più alto sposta il vertice della traiettoria più in alto rispetto all’asse y. È stato scelto un valore pari a **5000N** per non esagerare con la simulazione e renderla più contenuta;
* *thrust\_duration*: ossia la durata della spinta iniziale che in questa configurazione vale **2s**;
* *mass*: peso del missile impostato a **100kg** (un peso maggiore abbassa il vertice della traiettoria. In tal caso è possibile aumentare la spinta iniziale e la sua durata per una traiettoria più stabile).

Si anticipa che solo *gravity, launch\_angle* e *mass* sono parametri noti.

Inizialmente questi dati (eccezione fatta per *gravity*) sono stati inizializzati con valori scelti arbitrariamente. Tuttavia, è possibile eliminare i commenti relativi alla generazione probabilistica di questi valori e generare valori casuali attraverso **np.linspace(*start, end*)**.A computer screen shot of a black screen

Description automatically generated

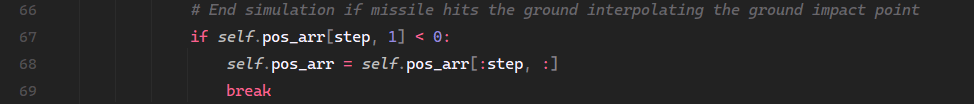
In seguito, è stata definita la classe *Missile* per parametrizzare le chiavi del dizionario e usarle per calcolare variabili utili per la simulazione. Tre componenti sono importanti:

1. *time\_arr*: vettore contenente una sequenza di *num\_time\_steps* (=5000) numeri uniformemente distribuiti da 0 a *max\_sim\_time* (tempo massimo per la simulazione);
2. pos\_arr*:* matrice *num\_time\_steps* × 2 che modella le coppie di coordinate (x, y);
3. *speed\_arr*: matrice *num\_time\_steps* × 2 che modella le coppie di coordinate (dx, dy) dove **dx** indica la velocità sull’asse x e **dy** sull’asse y.

Queste tre componenti sono essenziali per l’applicazione del metodo di Eulero alle equazioni differenziali ordinarie che compongono il sistema fisico di riferimento e all’interpolazione del punto di collisione tra missile e terreno, condizione per la terminazione della simulazione.

A screenshot of a computer program

Description automatically generated*Simulazione*

*Termine* simulazione

Il grafico risultante è il seguente: A graph of a missile

Description automatically generated

Completata la simulazione, il KBS ha una struttura del mondo su cui creare il dataset iniziale.

## Generazione di rumore

Per una resa più realistica e per un principio di aleatorietà della generazione dei dati, è stato aggiunto rumore ai dati simulativi, modellando 8 punti: A screen shot of a computer program

Description automatically generated

A graph with dots and numbers

Description automatically generated

Si può notare come i punti rumorosi si discostano da quelli effettivi.

Questo dataset viene salvato in un file denominato *dataset.csv*, non molto utile all’atto pratico di questo progetto sebbene sia efficace per una visualizzazione standard dei dati e per eventuali estensioni future di questo progetto alla stesura di altri.

|  |  |
| --- | --- |
| X Displacement | Y Displacement |
| 0.496714153 | -0.138264301 |
| 3.084300509 | 24.58597317 |
| 9.237681352 | 88.64539848 |
| 22.99205925 | 200.0864531 |
| 33.90199858 | 321.1692631 |
| 46.90556063 | 424.2916264 |
| 60.64722644 | 509.8420756 |
| 69.57840821 | 600.2950869 |

# Apprendimento

Questo modulo prepara la PINN all’addestramento dei dati generati nella fase precedente. Tuttavia va chiarito cosa sia una PINN.

## Physical Informed Neural Network

Sebbene simili, ANN (Artificial Neural Network) e PINN si differenziano per alcuni aspetti. Entrambe sono reti neurali, ossia modelli parametrizzati per predizioni, tipicamente composte da diversi livelli di funzioni lineari parametrizzate e funzioni di attivazione non lineari. Questi modelli (d’ora in poi abbreviate con NN) restituiscono una funzione lineare per una predizione a valori reali. In base al tipo di predizione le NN calcolano funzioni per la predizione, come nel caso di una predizione booleana che fa uso di sigmoide, o softmax per quelle categoriche. Per quanto concerne predizioni strutturate vengono introdotti metodi speciali.

Le NN sono composte da *n - 1* *hidden layers* (livelli/strati nascosti) ed *n* livelli, dove *n* è la profondità della rete, ognuno dei quali mappa un vettore (o strutture dati affini come array o lista). I componenti dei vettori sono detti *unità*. Nel caso di questo progetto la PINN è composta da 4 livelli.

Le NN fanno uso di una funzione di attivazione non lineare che calcola l’output in funzione degli input del livello corrente.

Le ANN sono adatte al riconoscimento di pattern complessi nei dati e all’approssimazione di funzioni non lineari, tuttavia, richiedendo un’importante mole di dati. Le ANN sono addestrate mediante metodi di ottimizzazione come la discesa di gradiente per la minimizzazione di una funzione di perdita, ad esempio MSE (Mean Squared Error, *Errore quadratico medio*, o MSL, Mean Squared Loss) o Cross-Entropia.

In questo progetto sono state adottate le PINN che si differenziano dalle ANN tradizionali per il loro dominio, ossia problemi di carattere fisico. Rispetto alla controparte tradizionale, le PINN sono composte di livelli i cui input sono coordinate spaziali e temporali, integrando equazioni fisiche nella funzione di perdita, inserendo vincoli fisici, per l’addestramento. Non necessita di grandi quantità di dati grazie alla conoscenza incorporata della fisica.

## Preparazione dei dati per l’addestramento

Vengono preparati i dati (rumorosi) per l’addestramento della PINN trattandoli come *target feature*A computer screen shot of a program

Description automatically generated

e definito un dizionario contenente gli iperparametri.A screenshot of a computer

Description automatically generated

## Definizione della PINN feed-forward

Come anticipato, la NN utilizzata in questo progetto è la PINN. Essa è stata definita attraverso la libreria *PyTorch*.

La PINN è composta di 4 livelli, di cui:

1. 1 livello di input;
2. 2 livelli nascosti che fanno uso di una funzione di attivazione detta GELU;
3. 1 livello di output.

Il **livello di input** prende il tempo come input monodimensionale e predice la posizione del missile come **output** bidimensionale.

I **livelli nascosti** sono stati configurati attualmente con 50 neuroni e fanno uso della **GELU** come funzione di attivazione.

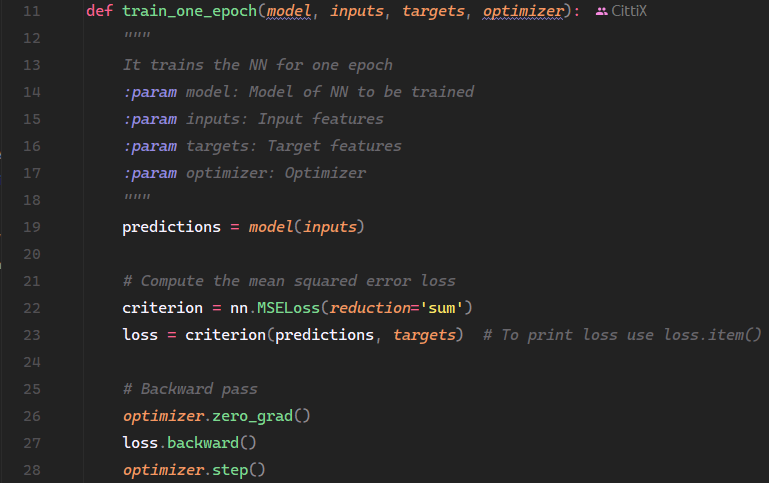
La **funzione di attivazione** utilizzata per questo modello è la *Gaussian Error Linear Unit* (GELU) e non la ReLU. La scelta è stata fatta tenendo in considerazione la gradualità che fornisce rispetto alla ReLU, oltre al fatto che rappresenta una soluzione più moderna e adatta alle PINN, in particolare quella definita in questo progetto che è poco profonda.

Come ottimizzatore è stato scelto **adam** (adaptive moments, *momentum adattivi*) data la presenza di scale differenti tra i parametri, la necessità di una rapida convergenza, la presenza di molti parametri e tempo richiesto per l’addestramento. Adam è un ottimizzatore che usa sia i momentum che il quadrato dei gradienti, apportando correzioni ai parametri da ottimizzare che sono stati eventualmente inizializzati a 0, dato che non rappresentano una buona stima su cui calcolare la media. Rappresenta una soluzione migliore per questo progetto rispetto, ad esempio, alla **SGD** (Stochastic Gradient Descent, *Discesa di Gradiente Stocastica*) standard che converge più lentamente. A screen shot of a computer program

Description automatically generated

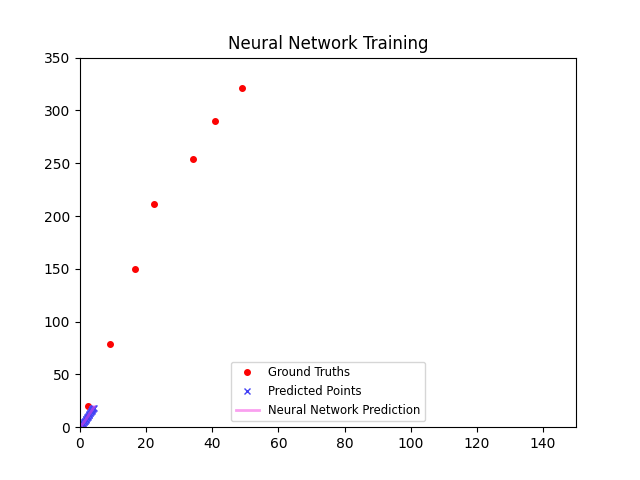
Per una computazione più rapida le feature in input ed output sono state convertite in tensori che integrano il calcolo con GPU.

## Addestramento e Valutazione

La PINN, ogni epoca, addestra le valutazioni sulle predizioni (**train\_one\_epoch***)* e ad una fissata frequenza le registra (**train\_and\_record**) in un array che verrà poi utilizzato per visualizzare i risultati su un grafico. 

L’addestramento del modello in un’epoca usa come funzione di perdita la media quadratica tra predizioni e dati effettivi e ad ogni step vengono calcolati i gradienti. A screen shot of a computer

Description automatically generated

Come primo risultato viene mostrato un grafico animato dell’addestramento della PINN, tenendo conto delle *ground truth* e dei punti predetti:

## Problemi

Dal grafico si può notare come la PINN presenti sovraddattamento e scarsa estrapolazione.

La curva si adatta troppo ai punti di training e questo causerà scarsa capacità di generalizzazione. Inoltre, al di fuori della regione di training, le predizioni sono scorrette, violano leggi fisiche e vincoli e i dati non sono utilizzabili, anche se da un certo punto di vista, forniti i dati di addestramento, questo sia assumibile ma non da un punto di vista cinematico.

## Soluzione

Come è stato già asserito, le PINN non sono totalmente diverse dalle ANN tradizionali. Tuttavia, oltre alle differenze già citate, vi è l’inclusione di operatori differenziali, importanti per il calcolo di leggi fisiche che coinvolgono l’oggetto di questo progetto.

Tuttavia, le PINN non risolvono le leggi fisiche direttamente, bensì imparano ad approssimare una funzione *f(x)* tale che le equazioni coinvolte nel calcolo della fisica vengano minimizzate.

Si introduce dunque la funzione di perdita definita come segue:

Dove:

1. : termine per l’errore della predizione, in questo caso l’errore quadratico;
2. : termine per l’errore calcolato sulla fisica, indicatore di quanto l’output della PINN soddisfi le leggi fisiche;
3. : regolarizzatore dell’errore fisico.

Un’altra importante differenza che emerge tra PINN e ANN tradizionali è nell’errore fisico che è calcolato sull’intero dominio in input e non solo dove ci sono i dati di training. Queste importanti differenze rendono le PINN più adatte al problema affrontato in questo progetto rispetto alle ANN tradizionali, fornendo ottime capacità di estrapolazione e facendo emergere ottime capacità nell’apprendere dati al di fuori della regione di addestramento.

## Seconda Legge di Newton e Applicazione della Funzione d’Errore Fisico

La traiettoria del missile è calcolata dalla Seconda Legge di Newton che considera forze come accelerazione gravitazionale, resistenza aerodinamica e spinta:

,

Dove:

1. è la resistenza aerodinamica,
2. è la spinta,
3. è la massa,
4. è l’accelerazione gravitazionale,
5. è l’angolo di lancio del missile.

Queste due equazioni descrivono il comportamento dell’accelerazione lungo l’asse x ed y.

Dunque, la PINN ha lo scopo di imparare una funzione *f(t)* che predice le coordinate (x, y) al tempo *t*.

La retropropagazione necessita di una funzione differenziabile per il calcolo dei gradienti. La funzione utilizzata in questo progetto, *step(thrust)* coinvolge la resistenza aerodinamica ed è attiva solo per i primi 2 secondi (*t <= 2*). La funzione è cosí definita:

che non è differenziabile in *t=3*. Viene dunque fornita una funzione logistica come approssimazione della precedente:

Dove:

1. , una costante che controlla la precisione dello step
2. , il momento in cui la resistenza aerodinamica cede,
3. , momento attuale,
4. , una piccola costante che adatta meglio la posizione dello step.

sono stati scelti attraverso dei test empirici. A computer screen with text

Description automatically generated

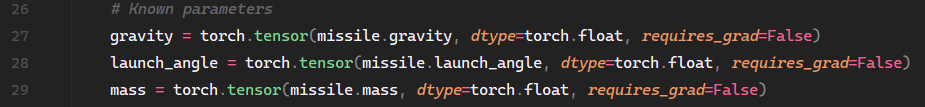
Ne risulta quindi: A graph with a line graph

Description automatically generated

La funzione di perdita fisica è cosí definita:

Tutte le derivate sono calcolate dalla PINN, il resto invece sono costanti fisiche. Le derivate vengono predette attraverso un calcolo di gradienti: A screen shot of a computer program

Description automatically generated

Mentre le caratteristiche note del missile vengono così calcolate: 

Tuttavia il modello non conosce parametri come *drag\_coefficient* e *thrust\_magnitude*, dunque vengono inizializzati a 1 e calcolati attraverso gradienti: A screen shot of a computer code

Description automatically generated

Il risultato dell’introduzione della funzione di errore fisica è dato dal seguente grafico animato: A graph with red dots

Description automatically generated

È stato così risolto il sovradattamento e la potenza di estrapolazione della PINN.

Il modello è stato poi reinizializzato insieme all’ottimizzatore: A screen shot of a computer program

Description automatically generated

In seguito all’aggiunta della funzione di errore, è stato adattato l’addestramento di un’epoca: A computer screen shot of a program

Description automatically generated

## Risultato Finale

Si mostra il risultato finale del progetto: A graph with numbers and text

Description automatically generated

Come si può notare, la PINN sottostima leggermente le due forze ignote, spinta e resistenza aerodinamica, risultando più “piatta” rispetto a quella effettiva. Viene anche anticipata la collisione del missile con il terreno; tuttavia, è poco meno di un secondo.

### Conclusioni

In questo progetto si è dimostrata la potenza e l’utilità di un modello di rete neurale feedforward noto come PINN per casi che coinvolgono la fisica. Utilizzando le PINN si è scavalcata la difficoltà del sovradattamento e della scarsa capacità di estrapolazione del modello risultando in uno affidabile, credibile e potente, nonostante la presenza di rumore e la non conoscenza di due parametri fisici.

Elenco argomenti di interesse

* Feature Engineering
* Reti neurali feed forward
* Apprendimento dei parametri
* Migliore l’ottimizzazione
* Apprendimento supervisionato
* Sovradattamento
* Regolarizzatori
* Valutare le predizioni
* Modelli di conoscenza incerta
* Ricerca di soluzioni in spazi di stati

# Strumenti utilizzati

* PyTorch
* NumPy
* MatPlotLib
* Pandas
* IPython
* Jupyter

IDE: PyCharm