



UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di Laurea Magistrale in Informatica

Modellazione e Apprendimento di Studi Epidemiologici con Agenti su Grafo

Relatore: Prof. Antoniotti Marco

Correlatore: Prof.

Relazione della prova finale di:

Matteo Stievano
Matricola 829535

Anno Accademico 2022-2023

Abstract

Lo scopo del presente lavoro è l'analisi e lo studio del comportamento di modelli di simulazione e di intervento nell'ambito della gestione di eventi pandemici, utilizzando il linguaggio di programmazione Julia e alcuni dei suoi framework, tra cui Agents.jl, SciML.ai e Lux.jl.

Nello specifico, si propone l'adozione del framework Agents.jl come strumento principale per la simulazione di sistemi complessi e l'integrazione di SciML.ai e Lux.jl per lo sviluppo di un controllore in grado di analizzare e interpretare i dati in ingresso.

L'obiettivo è sviluppare un approccio dinamico e ibrido che combini le vantaggiose caratteristiche dei modelli di simulazione e dei modelli matematici. Poiché il problema affrontato è intrinsecamente complesso, è stato modellato come un grafo sociale, cercando di emulare il comportamento di una rete sociale in cui i nodi del grafo rappresentano gli agenti del modello.

Per migliorare le prestazioni e la stabilità del modello, è stato introdotto un sistema di equazioni differenziali ordinarie (ODE) che gestisce e simula l'andamento epidemico di ciascun nodo (agente) del grafo. Il controllore si basa sull'idea di una Neural ODE, che regola in modo autonomo e automatico il livello di contromisure applicate. Tali contromisure sono identificate come una riduzione del numero di individui infetti. Per evitare l'applicazione di contromisure praticamente insostenibili, è stato introdotto un parametro di "felicità", che agisce come una funzione di costo aggiuntiva per il controllore.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduzione | 1 |
| 2 | Panoramica dello Stato dell'Arte | 3 |
| 2.1 | Epidemiologia | 3 |
| 2.2 | Modelli Compartmentali | 6 |
| 2.2.1 | Il modello Kermack-McKendrick | 8 |
| 2.3 | Modelli basati su Agenti | 12 |
| 2.3.1 | Comportamento Emergente | 13 |
| 2.3.2 | Discretizzazione | 14 |
| 2.4 | Julia | 15 |
| 2.4.1 | Agents.jl | 15 |
| 2.4.2 | Benchmarking e confronto con altri linguaggi | 16 |
| 2.5 | Equazioni Neurali Differenziali | 19 |
| 2.5.1 | Equazioni Differenziali Universali | 22 |
| 2.5.2 | Inserire un'ODE all'interno di una NN | 25 |
| 2.5.3 | Backpropagation tramite ODE solver | 27 |
| 3 | Metodi e Modelli | 28 |
| 3.1 | Approccio con Rete Sociale | 28 |
| 3.2 | Agente | 30 |
| 3.3 | Spazio e Modello | 32 |
| 3.4 | Funzione di Avanzamento Agente | 33 |
| 3.5 | Funzione di Avanzamento Modello | 35 |
| 3.5.1 | Funzione per la Generazione delle Varianti del Virus (VOC) | 36 |
| 3.5.2 | Funzione per la Simulazione della Campagna di Vaccinazione | 37 |
| 3.6 | Monitoraggio e Intervento | 39 |
| 3.6.1 | Implementazione Controllore | 42 |
| 3.6.2 | Esplorazione Funzione di Loss | 44 |
| 3.6.3 | Esplorazione Funzione di Addestramento | 45 |
| 4 | Analisi di Sensibilità | 48 |
| 4.1 | Analisi del Comportamento in Base al Numero Iniziale di Nodi Infetti | 50 |
| 4.2 | Analisi del Comportamento in Base al Tasso di Migrazione | 51 |
| 4.3 | Analisi del Comportamento in Base al Numero di Nodi della Rete | 52 |
| 4.4 | Analisi del Comportamento in Base alla Copertura della Rete | 53 |
| 4.5 | Studio dell'Andamento del Controllore in Relazione all'Intervallo di Intervento | 55 |
| 4.6 | Studio dell'Andamento del Controllore in Relazione all'Intervallo di Tolleranza | 56 |
| 5 | Risultati Ottenuti | 59 |
| 5.1 | Nessun Intervento | 61 |

| | | |
|----------|--|-----------|
| 5.2 | Intervento Non Farmaceutico | 62 |
| 5.3 | Intervento Farmaceutico | 63 |
| 5.3.1 | Grafici di Confronto | 64 |
| 5.4 | Intervento Farmaceutico e Non Farmaceutico | 64 |
| 5.4.1 | Grafici di Confronto | 65 |
| 6 | Conclusioni | 67 |
| 7 | Sviluppi Futuri | 68 |
| 7.1 | Perfezionamento del Controllore | 68 |
| 7.2 | Miglioramento della Funzione di Happiness | 68 |
| 7.3 | Perfezionamento Generale del Modello | 68 |
| 7.4 | Ottimizzazione Generale | 68 |
| A | Altri Approcci | 80 |
| A.1 | Modello ad Agente su Spazio Continuo | 80 |
| A.2 | Modello ad Agente con Spazio a Grafo e Modellazione Singolo Agente . . | 82 |
| A.3 | Controllore Ipopt | 84 |

List of Figures

| | | |
|----|---|----|
| 1 | Esempio di correlazione spuria | 4 |
| 2 | Esempio della struttura del modello SIR | 9 |
| 3 | Esempio di modello SEIR con perdita di immunità [1] | 10 |
| 4 | Esempio di modello SEIR stocastico | 11 |
| 5 | Agent-Based Epidemiological Modeling of COVID-19 in Localized Environment [2] | 13 |
| 6 | Esempio di comportamento emergente nella simulazione degli stormi di uccelli | 13 |
| 7 | Esempio di differenti tipologie di discretizzazione, siano esse nel tempo o nei valori | 14 |
| 8 | Esempio di metriche per il calcolo della distanza su uno spazio a griglia [3] | 16 |
| 9 | Agent Based Modelling and Simulation tools: A review of the state-of-art software [4] | 18 |
| 10 | Differenti approcci di apprendimento di machine learning | 19 |
| 11 | Esempio di funzionamento di una rete neurale | 21 |
| 12 | Esempio comparativo tra funzionamento Machine Learning e Deep Learning | 23 |
| 13 | Comportamento UDE nell'approssimazione di fenomeni non lineari [5] . | 24 |
| 14 | Esempio definizione ODE in Julia | 25 |
| 15 | Esempio implementazione di una Neural ODE in Julia tramite l'utilizzo della libreria <code>DiffEqFlux.jl</code> [6] | 26 |
| 17 | Grafo Sociale di una rete di classe 3 | 29 |
| 18 | Codice Agente | 30 |
| 19 | Esempio di matrice di migrazione in una rete con 8 nodi | 31 |
| 20 | Funzione che crea la matrice di migrazione data la topologia di un grafo . | 31 |
| 22 | Funzione che si occupa di gestire la simulazione di ogni singolo agente presente all'interno del modello | 33 |
| 23 | Funzione atta a calcolare lo spostamento di agenti da un nodo all'altro del grafo | 34 |
| 24 | Funzione atta a calcolare la felicità degli agenti | 34 |
| 25 | Codice per l'aggiornamento del modello ad ogni passo della simulazione . | 36 |
| 26 | Funzione che si occupa di generare la VOC | 37 |
| 27 | Funzione che si occupa di simulare la ricerca di un vaccino e la sua successiva applicazione | 38 |
| 28 | Definizione della funzione di attivazione del controllore all'interno della funzione <code>agent_step!</code> (Figura 22) definita nel modello di simulazione . | 41 |
| 29 | Definizione del controllore mediante Neural ODE | 42 |
| 30 | Definizione delle funzioni di supporto del controllore | 43 |
| 32 | Definizione della funzione di loss per la Rete Neurale | 45 |
| 33 | Definizione delle funzioni di addestramento del controllore | 46 |
| 34 | Funzione di attivazione Swish | 47 |
| 35 | Grafico rappresentante l'analisi di sensitività del modello | 48 |

| | | |
|----|---|----|
| 36 | Parametri usati per l'analisi di sensitività del modello ad agente | 49 |
| 43 | Grafici cumulativi descriventi il numero cumulativo medio di individui Exposed, Infected e Dead insieme al valore di Happiness ottenuto da una simulazione multipla (<code>EnsembleRun</code>) considerando differenti contromisure | 60 |
| 44 | Grafico cumulativo del modello senza alcun tipo di intervento | 61 |
| 45 | Grafico cumulativo del modello con intervento non farmaceutico | 62 |
| 46 | Grafico cumulativo del modello con intervento farmaceutico | 63 |
| 48 | Grafico cumulativo del modello con intervento farmaceutico e non farmaceutico combinato | 65 |
| 50 | Esempio del modello modellato su spazio continuo | 80 |
| 51 | Esempio del comportamento delle curve nel modello continuo | 81 |
| 52 | Comportamento modello ABM su spazio a grafo al variare del parametro R_0 | 82 |
| 53 | Comportamento modello SEIR al variare del parametro R_0 | 83 |
| 54 | Comparison of Ipopt performance over various linear solvers using the two-dimensional partial differential equation test problem set. [7] | 84 |
| 55 | Definizione del controllore tramite Ipopt | 85 |
| 56 | Definizione regole del modello del controller | 86 |
| 57 | Definizione regole del modello del controller per le contromisure | 86 |

List of Tables

| | | |
|---|--|----|
| 1 | Paradosso di Simpson | 5 |
| 2 | A global panel database of pandemic policies Oxford COVID-19 Government Response Tracker [8] | 40 |
| 3 | Benchmarking con BenchmarkTools.jl [9] | 59 |
| 4 | Parametri utilizzati per la simulazione del modello | 60 |

1 Introduzione

La storia umana è segnata da epidemie, ma solo alcune di esse hanno lasciato un'impronta duratura nella memoria collettiva a causa delle loro conseguenze catastrofiche. Tra queste, alcune delle più note includono la peste nera che nel XIV secolo mieté venti milioni di vite in Europa in soli sei anni, l'epidemia di tifo durante le crociate e la Seconda Guerra Mondiale, l'influenza spagnola che causò 50 milioni di morti tra il 1918 e il 1920, e l'epidemia di AIDS, che ha colpito oltre 75 milioni di persone e causato 35 milioni di decessi dal 1981.

Oggi, l'influenza stagionale non suscita più lo stesso timore nei cittadini dei paesi sviluppati, ma la pandemia di COVID-19, iniziata alla fine del 2019, ha condizionato l'umanità per tre anni e continua a farlo, causando finora quasi 7 milioni di vittime accertate. Questa pandemia rimarrà impressa nella memoria umana poiché ha messo in crisi l'intero sistema governativo globale, causando allarmi, panico e, talvolta, isteria, come pochi altri eventi sono stati in grado di fare.

Esaminando le statistiche di questa epidemia, i numeri relativi ai decessi e agli infetti (quasi 7 milioni di morti e quasi 800 milioni di infetti) da soli sono sufficienti a preoccupare qualsiasi lettore. Tuttavia, ci sono altri dati meno evidenti che forniscono informazioni altrettanto inquietanti, come l'impatto economico globale e il concetto di "poverty trap", un circolo vizioso in cui la povertà e le malattie perpetuano un ciclo di bassa salute e crescente povertà.

Questi sono solo alcuni dei problemi che possono emergere durante una pandemia globale come quella del COVID-19, e quindi la comunità scientifica, in particolare gli epidemiologi, cerca costantemente soluzioni più efficaci ed accurate per prevenire, contenere e gestire eventi di questo genere.

L'epidemiologia è una disciplina relativamente giovane che si è evoluta per affrontare emergenze sanitarie. La sua definizione originale risale al 1978 [10] e cita:

"Epidemiology is the study of the prevalence and dynamics of stages of health in populations."

ma nel corso del tempo è cresciuta e si è adattata alle esigenze della società.

Attualmente [11], l'epidemiologia è definita come:

"Epidemiology is the study of the distribution and determinants of health-related states or events (including disease), and the application of this study to the control of diseases and other health problems. Various methods can be used to carry out epidemiological investigations: surveillance and descriptive studies can be used to study distribution; analytical studies are used to study determinants."

Uno strumento ampiamente utilizzato in epidemiologia è la simulazione tramite software, che si basa su modelli matematici per trarre conclusioni sui sistemi analizzati. Questi

sistemi, spesso definiti complessi, coinvolgono molteplici componenti che interagiscono tra loro.

I primi modelli epidemiologici erano basati sul paradigma compartimentale, dove gruppi di individui separati interagivano tra loro e potevano essere descritti mediante equazioni differenziali ordinarie (ODE). Tra i primi modelli utilizzanti le ODEs troviamo il così detto “Susceptible-Infected-Recovered” (SIR) sviluppato da Kermack e McKendrick nel 1927 [12]. Successivamente, sono emersi i modelli ad agente, che consentono la simulazione di sistemi complessi tramite l’interazione di entità autonome.

La simulazione ha fatto notevoli progressi dagli anni ’90 grazie all’espansione delle risorse computazionali. Tuttavia, i modelli di simulazione richiedono compromessi e semplificazioni, ad esempio la discretizzazione degli ambienti di simulazione, ma permettono lo studio di fenomeni definiti *emergenti*. Questi sono comportamenti dell’intero sistema che non possono venire predetti osservando il comportamento dei singoli agenti. Inoltre, è necessario considerare il comportamento umano in situazioni di pericolo, il che può essere complesso da modellare.

L’identificazione delle cause di un fenomeno e la comprensione di come un intervento influenzi tale fenomeno rappresentano una delle sfide più complesse dell’epidemiologia. La causalità, ossia la relazione di causa-effetto tra eventi o variabili, non è sempre evidente e richiede un’analisi rigorosa. La correlazione tra eventi non implica necessariamente causalità.

Le sezioni successive analizzeranno lo stato dell’arte dell’epidemiologia e della simulazione, con un focus sulla pandemia da COVID-19 e sui metodi di monitoraggio e intervento nelle simulazioni epidemiologiche.

2 Panoramica dello Stato dell'Arte

2.1 Epidemiologia

L'epidemiologia rappresenta una disciplina biomedica di fondamentale importanza, focalizzata sull'analisi della distribuzione e dell'incidenza delle malattie e degli eventi sanitari rilevanti all'interno di una popolazione. Questo campo di studio si dedica all'approfondimento delle cause, dei pattern temporali e delle conseguenze delle malattie [13] [14].

L'epidemiologia si divide in quattro ambiti principali [15]: epidemiologia descrittiva, epidemiologia analitica, epidemiologia clinica ed epidemiologia sperimentale.

1. **L'epidemiologia descrittiva** si occupa di studiare la frequenza e la distribuzione delle malattie e dei parametri di salute nelle popolazioni. Descrive eventi sanitari come malattie, cause di morte la presenza di fattori di rischio come, ad esempio, il fumo di tabacco, l'inquinamento atmosferico.
2. **L'epidemiologia analitica** si occupa invece di studiare le cause delle malattie e degli eventi sanitari. In particolare, cerca di identificare i fattori che possono influenzare l'insorgenza o lo sviluppo della malattia.
3. **L'epidemiologia clinica** si concentra sullo studio dei pazienti affetti da una determinata patologia. In particolare, cerca di identificare i fattori che possono influenzare il decorso della malattia e l'efficacia dei trattamenti.
4. **L'epidemiologia sperimentale** si occupa invece di studiare gli effetti delle terapie preventive o curative sulla popolazione.

L'epidemiologia si caratterizza per la sua natura essenzialmente pratica, poiché il suo obiettivo principale consiste nel determinare le cause sottese a un determinato effetto sanitario. Questa disciplina si trova ad affrontare una serie di sfide complesse che definiscono il suo percorso. Tra queste sfide possiamo trovare l'identificazione delle relazioni di causalità tra eventi.

Tali interrogativi possono sembrare elementari, dato che, come specie, abbiamo sviluppato un istinto innato nell'individuare correlazioni causali tra eventi, anche quando queste non esistono effettivamente. Ad esempio, se ci trovassimo in un bosco buio, soli e circondati solo dal sussurro di una leggera brezza estiva e dovessimo udire un rumore provenire dai cespugli, è probabile che lo associeremmo a un pericolo imminente, come la presenza di un predatore, sebbene il rumore sia causato dalla brezza stessa.

Questo adattamento evolutivo ci ha permesso di sopravvivere in situazioni di pericolo, ma purtroppo, quando si tratta di scienza e dati, l'istinto non è sempre una guida affidabile. I dati, per loro natura, sono neutri e non trasmettono automaticamente informazioni significative; spetta a noi, in quanto individui dotati di intelligenza, tecniche e metodi, estrarre significato da questi dati in modo accurato ed inequivocabile.

Un’osservazione chiave è che ciò che sembra ovvio può essere fuorviante. Per esempio, il grafico seguente sembra dimostrare in modo “inequivocabile” una relazione diretta tra la spesa degli Stati Uniti per la ricerca aerospaziale e il numero di suicidi per strangolamento:

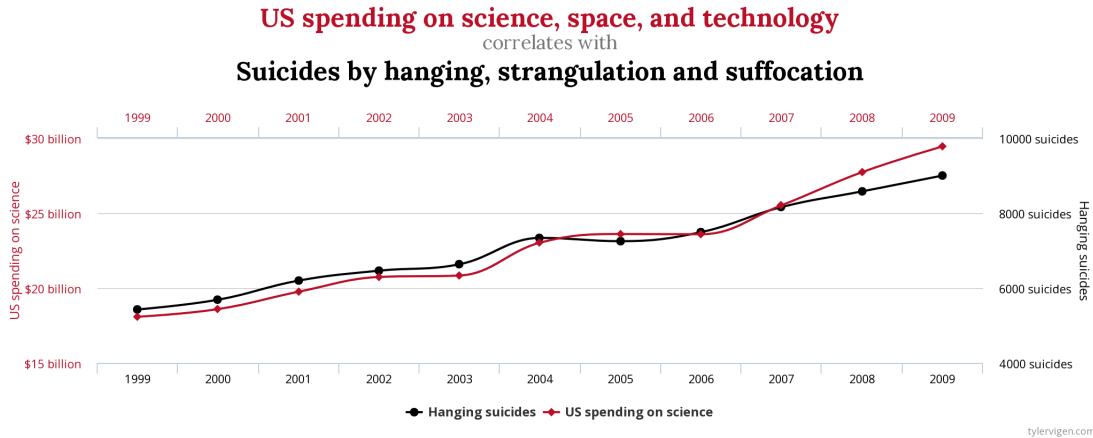


Figure 1: Esempio di correlazione spuria
<https://www.tylervigen.com/spurious-correlations>

Sulla base di questo grafico e dei dati presentati, si potrebbe erroneamente concludere che le due categorie sono in qualche modo correlate e che il governo degli Stati Uniti debba essere accusato di incitare al suicidio. Tuttavia, questo è un esempio di una relazione spuria, in cui due o più variabili sono associate ma non sono causalmente collegate.

È importante sottolineare che l’associazione e la causalità non sono la stessa cosa, e quando si studia una, è fondamentale non confonderla con l’altra. In statistica, una correlazione tra dati rappresenta qualsiasi tipo di relazione tra due o più variabili, indipendentemente dalla sua natura causale o non causale. Nel caso precedente, la correlazione tra le due variabili potrebbe essere semplicemente dovuta al passare del tempo: nel corso degli anni, la spesa media per la ricerca aerospaziale è continuata a crescere a causa di un interesse e di investimenti sempre maggiori in quel settore, mentre nel tempo si è verificato un costante aumento del numero di suicidi.

Il nostro pregiudizio verso la ricerca di collegamenti tra eventi, in modo che sembrino sempre collegati in modo tangibile e che si possa tracciare una chiara e distinta linea causale dall’inizio alla fine, può ingannarci quando tali collegamenti sembrano evidenti ma non lo sono. Spesso, la spiegazione più semplice è anche la meno interessante, sebbene sia corretta: due eventi completamente svincolati tra loro possono avere andamenti simili, e differenti fattori possono portare allo stesso comportamento.

Il problema della causalità è di estrema importanza e rappresenta una delle sfide principali quando si cercano di sviluppare e applicare interventi all’interno di una popolazione per mitigare, ad esempio, la diffusione di un agente patogeno [14].

Come già accennato, i dati per se stessi sono privi di significato; è il nostro compito imparare a interpretare il loro significato. Un esempio eloquente di come, nonostante la consapevolezza del problema delle correlazioni spurie, i dati possano comunque trarre in inganno è il seguente:

Supponiamo di essere medici e di dover decidere se prescrivere un certo farmaco a un paziente. Per prendere questa decisione, abbiamo a disposizione la storia clinica del paziente e i risultati di uno studio su un nuovo farmaco che sembra promettente nel trattamento della sua malattia. Questo farmaco è stato testato su un gruppo di 700 persone, suddivise in due sottogruppi di 350 pazienti ciascuno. I pazienti hanno scelto autonomamente se assumere o meno il farmaco. Ecco i risultati:

Table 1: Paradosso di Simpson

| Paradosso di Simpson | | | | | | |
|----------------------|----------|---------|-----------|----------|---------|-----------|
| Categoria | Pazienti | Guariti | % Guariti | Pazienti | Guariti | % Guariti |
| Uomini | 87 | 81 | 93% | 270 | 234 | 87% |
| Donne | 263 | 192 | 73% | 80 | 55 | 69% |
| Dati combinatori | 350 | 273 | 78% | 350 | 289 | 83% |

Questi risultati sembrano suggerire che la prescrizione di questo nuovo farmaco non abbia un impatto positivo sulla guarigione dei pazienti. Tuttavia, questo risultato è in realtà un esempio del cosiddetto “paradosso di Simpson”, in cui i dati aggregati relativi a un trattamento specifico sembrano indicare una perdita di efficacia, mentre i dati delle singole categorie mostrano risultati opposti. Questo esempio sottolinea il fatto che l’interpretazione di dati aggregati non sempre può essere affidabile, e talvolta può ingannare. In questi casi, è necessario estrarre le informazioni sulla causalità dai dati individuali.

È evidente che comprendere le cause di un determinato effetto o insieme di effetti non è un compito banale. Anche conoscendo l’agente patogeno o almeno la sua natura, non sempre è sufficiente per spiegare la complessità delle interazioni. L’utilizzo di modelli di apprendimento automatico per l’analisi dei dati, la ricerca di correlazioni e la successiva formulazione di politiche di intervento può rappresentare un rischio, ma al contempo offre un alleato potente nella definizione di politiche di intervento in settori estremamente delicati come quello della sanità [16].

2.2 Modelli Compartmentali

In campo epidemiologico, i modelli compartmentali [17] rappresentano una metodologia di modellazione generica che si presta bene all'analisi globale del comportamento di una malattia infettiva. Questa tecnica di modellazione trova applicazione anche in altri ambiti scientifici, tra cui la finanza.

L'approccio matematico dei modelli compartmentali si basa sull'assunzione che, dato un gruppo di individui, sia possibile suddividerli in diverse categorie in base al loro stato di esposizione o infezione da una particolare malattia. Questo processo crea compartimenti ben distinti che possono interagire tra loro, ma rimangono chiaramente separati gli uni dagli altri.

I modelli compartmentali sono di solito basati su equazioni differenziali ordinarie (che sono deterministiche), ma, utilizzando come input delle probabilità, possono anche essere visualizzati in un quadro stocastico che è più realistico ma anche più complicato da analizzare.

Equazioni Ordinarie Differenziali

Nel contesto matematico, le Equazioni Differenziali Ordinarie (ODE) costituiscono una categoria fondamentale di equazioni differenziali che coinvolgono una sola variabile indipendente, generalmente rappresentata come il tempo. All'interno delle numerose tipologie di equazioni differenziali, le ODE lineari occupano una posizione di particolare rilievo. Questo perché molteplici fenomeni fisici e matematici possono essere accuratamente descritti attraverso la risoluzione di ODE lineari.

Le ODE lineari si caratterizzano per la loro forma generale, che si presenta come un polinomio lineare delle derivate successive della funzione incognita rispetto alla variabile indipendente.

$$\alpha_0(x)y + \alpha_1(x)y' + \alpha_2(x)y'' + \dots + \alpha_n(x)y^{(n)} + b(x) = 0$$

In questa formulazione, i coefficienti $\alpha_0(x), \dots, \alpha_n(x)$ e la funzione $b(x)$ possono assumere qualsiasi forma differenziabile, senza la necessità di essere lineari, mentre le derivate $y', \dots, y^{(n)}$ rappresentano le successive derivate della funzione y rispetto alla variabile x .

È da notare che in diversi contesti, le equazioni differenziali non lineari possono essere approssimate attraverso equazioni lineari, semplificando notevolmente il processo di ottenimento delle soluzioni. Naturalmente ci sono dei compromessi da accettare applicando talune approssimazioni.

In generale, quando ci troviamo di fronte a una derivata di una funzione, il nostro obiettivo è comprendere e studiare la funzione stessa. Per fare ciò, è necessario trovare l'antiderivata, o integrale, della funzione data.

Un esempio semplice di questa idea è l'equazione $\frac{dx}{dt}(t) = \cos(t)$, la cui soluzione è $x(t) = \sin(t) + C$, dove C è una costante di integrazione.

Tuttavia, risolvere un'ODE può essere più complesso rispetto al calcolo di un semplice integrale. Anche se il concetto di base è lo stesso (ovvero, risolvere un integrale), la difficoltà principale risiede nella determinazione del tipo di integrazione necessario per ottenere la soluzione desiderata.

Nel caso in cui le equazioni coinvolgano $\frac{dx}{dt}$ e queste dipendano unicamente dalla variabile t , la soluzione risulta relativamente più semplice. Tuttavia, quando le equazioni coinvolgono $\frac{dx}{dt}$ come funzione di $x(t)$, la situazione diventa più complessa e richiede spesso l'applicazione di manipolazioni matematiche avanzate, come l'utilizzo delle regole di derivazione o la sostituzione delle variabili (chain rules o u-substitution). In queste situazioni, la risoluzione delle ODE può richiedere una comprensione approfondita delle tecniche matematiche e dell'applicazione dei principi di calcolo differenziale. Un esempio è l'utilizzo dei metodi di integrazione numerica Runge-Kutta.

Metodi di integrazione numerica Runge-Kutta

I metodi di integrazione di Runge-Kutta [18] [19] sono una famiglia di algoritmi ampiamente utilizzati per risolvere equazioni differenziali ordinarie (ODE). Questi metodi consentono di approssimare la soluzione di un'ODE attraverso passi discreti nel tempo. La forma generale di un metodo di Runge-Kutta è la seguente:

Dato un problema di ODE del primo ordine: $\frac{dy}{dt} = f(t, y)$

Un metodo di Runge-Kutta approssima $y(t)$ nei punti discreti t_n come segue:

$$y_{n+1} = y_n + h \cdot \sum_{i=1}^s b_i \cdot k_i$$

$$k_i = f(t_n + c_i \cdot h, y_n + h \cdot \sum_{j=1}^s a_{ij} \cdot k_j)$$

Dove:

- y_n è l'approssimazione di y al tempo t_n .
- h è la dimensione del passo temporale.
- s è il numero di stadi (spesso denominato ordine del metodo).
- b_i sono i pesi per calcolare l'approssimazione finale.
- c_i sono i coefficienti che specificano la posizione dei punti intermedi nel passo temporale.
- a_{ij} sono i coefficienti che governano l'interazione tra i punti intermedi k_i .

I metodi di Runge-Kutta sono molto flessibili e possono variare notevolmente in termini di complessità e precisione. Alcuni dei metodi di Runge-Kutta più noti includono il metodo di Euler (un metodo di Runge-Kutta di primo ordine), il metodo di Heun (un metodo di Runge-Kutta di secondo ordine), e il metodo di Runge-Kutta a quarto ordine.

Quest'ultimo è uno dei più comuni e ampiamente utilizzati per la sua buona precisione e stabilità.

Un esempio del metodo di Runge-Kutta a quarto ordine (RK4) con $s = 4$:

- $b_i = \frac{1}{6}$ per $i = 1, 2, 3, 4$.
- $c_1 = 0, c_2 = \frac{1}{2}, c_3 = \frac{1}{2}, c_4 = 1$.
- $a_{21} = \frac{1}{2}, a_{31} = 0, a_{32} = \frac{1}{2}, a_{41} = 0, a_{42} = 0, a_{43} = 1$.

I metodi di Runge-Kutta sono ampiamente utilizzati nella simulazione numerica, nella risoluzione di ODEs in fisica, ingegneria e molti altri campi in cui è necessario risolvere problemi che coinvolgono equazioni differenziali. Sono apprezzati per la loro accuratezza e robustezza in una vasta gamma di applicazioni.

2.2.1 Il modello Kermack-McKendrick

Secondo le definizioni sopra riportate, è possibile suddividere la popolazione oggetto di studio in categorie o compartimenti, stabilendo una relazione temporale che descrive il passaggio degli individui da un compartimento all'altro. In base alla tipologia di malattia analizzata si avrà una struttura compartmentale differente. Malattie che conferiscono immunità avranno una struttura compartmentale diversa rispetto a malattie che non la conferiscono.

Il modello più utilizzato come riferimento per lo studio e la modellazione epidemiologica è il modello “Susceptible, Infectious, Recovered” (SIR), ideato nel 1927 da Kermack e McKendrick [12]. Questo modello si basa sull'assunzione che, durante lo sviluppo di una malattia, gli individui possano trovarsi in uno di tre stati:

- **Susceptible (S)**: Rappresenta lo stato iniziale per la maggior parte degli individui all'interno di una popolazione, indicando il numero di persone che possono contrarre la malattia.
- **Infectious (I)**: Questo stato rappresenta gli individui che, partendo dallo stato suscettibile, diventano infetti dopo essere venuti in contatto con individui infetti.
- **Recovered (R)**: Questo stato rappresenta gli individui che, al termine della malattia, sopravvivono o muoiono a causa di essa. A volte, questo stato è chiamato anche “Removed”.

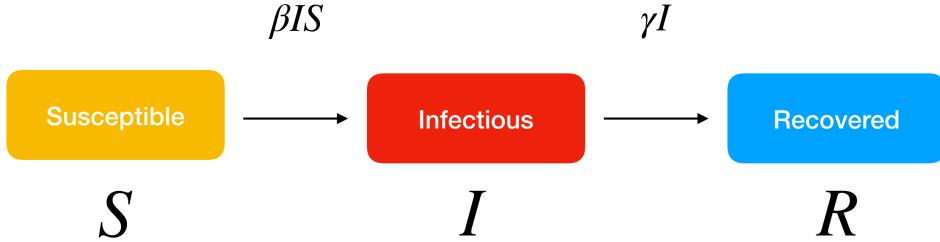


Figure 2: Esempio della struttura del modello SIR

Da questa idea di base, è stato sviluppato un modello matematico che descrive come queste tre categorie, sebbene separate, interagiscano nel tempo. Il tempo è considerato la variabile indipendente del modello, denotata come t , e i tassi di transizione tra i compartimenti sono espressi come derivate rispetto al tempo e alle dimensioni dei compartimenti. Di conseguenza, questo modello è stato inizialmente formulato utilizzando equazioni differenziali ordinarie [20].

Il sistema di equazioni differenziali che descrive il modello SIR è il seguente:

$$\frac{dS}{dt} = -\beta \cdot S \cdot I$$

$$\frac{dI}{dt} = \beta \cdot S \cdot I - \gamma \cdot I$$

$$\frac{dR}{dt} = \gamma \cdot I$$

Questo sistema modella le seguenti assunzioni:

- In media, un individuo nella popolazione ha abbastanza contatti con gli altri da permettere la diffusione dell'infezione, con un tasso di infezione rappresentato da $\beta \cdot N$, dove N è il numero totale di individui nella popolazione.
- Gli individui infetti lasciano il compartimento degli infetti a un tasso di $\gamma \cdot I$.
- Non ci sono entrate o uscite di individui dalla popolazione totale, tranne per le possibili uscite dovute alla morte causata dalla malattia stessa.

L'indice R_0 rappresenta il numero di infezioni secondarie causate da un singolo individuo durante il suo periodo infettivo in una popolazione completamente suscettibile di dimensione $K \approx S(0)$. In questa situazione, un individuo infetto effettua $\beta \cdot K$ contatti per unità di tempo, ognuno dei quali può trasmettere l'infezione a un individuo suscettibile,

producendo nuove infezioni durante il suo periodo medio di infettività di $\frac{1}{\gamma}$. Pertanto, il valore di R_0 è $\beta \cdot \frac{K}{\gamma}$.

Il valore di R_0 è cruciale nel determinare se si verificherà o meno un'epidemia. Se $R_0 < 1$, l'infezione si estinguerà prima di diventare un'epidemia. Se $R_0 > 1$, ci sarà un'epidemia.

La modellazione epidemiologica può diventare più complessa in presenza di una fase di esposizione, in cui gli individui infetti non sono immediatamente infettivi. In questo caso, è possibile aggiungere un compartimento “Exposed (E)” al modello, che influenzera il sistema di equazioni differenziali. Questo tipo di modello è noto come SEIR (Susceptible, Exposed, Infectious, Recovered) ed è largamente utilizzato all'interno dell'ambito epidemiologico e medico.

Il sistema di equazioni differenziali che descrive il modello SEIR è il seguente:

$$\begin{aligned}\frac{dS}{dt} &= -\beta(N) \cdot S \cdot I \\ \frac{dE}{dt} &= \beta(N) \cdot S \cdot I - \kappa \cdot E \\ \frac{dI}{dt} &= \kappa \cdot E - \gamma \cdot I \\ \frac{dR}{dt} &= \gamma \cdot I\end{aligned}$$

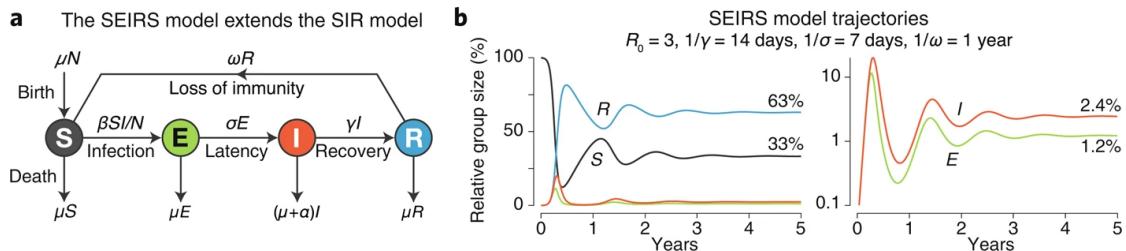


Figure 3: Esempio di modello SEIR con perdita di immunità [1]

Un altro sviluppo importante è l'analisi dei modelli epidemiologici in forma stocastica, che tiene conto della variabilità individuale e delle piccole dimensioni dei compartimenti all'inizio di un'epidemia. Questi modelli stocastici considerano gli effetti casuali nei contatti tra gli individui e possono essere particolarmente rilevanti nelle prime fasi di un'epidemia, quando il numero di infetti è ancora relativamente basso. In queste situazioni, è necessario utilizzare approcci stocastici, come i modelli basati su reti (network models) per catturare la dinamica epidemica in modo più accurato.

In generale, i modelli compartmentali sono strumenti essenziali per la modellazione e la comprensione della diffusione delle malattie infettive, consentendo agli epidemiologi di studiare scenari diversi e sviluppare strategie di controllo più efficaci.

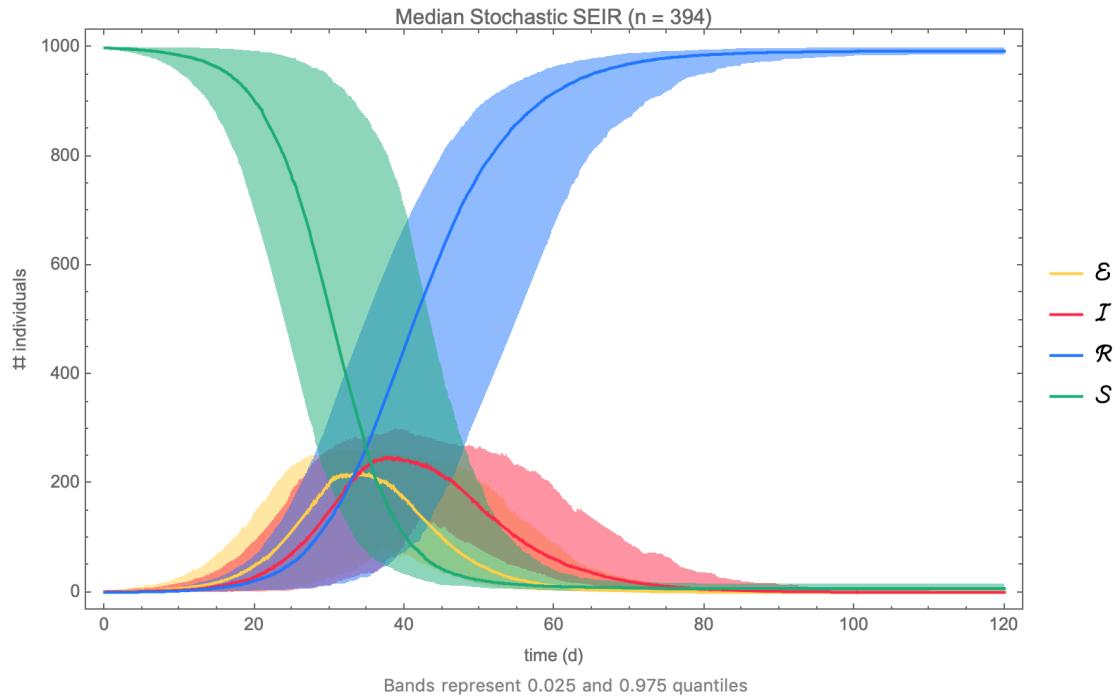


Figure 4: Esempio di modello SEIR stocastico
https://community.wolfram.com//c/portal/getImageAttachment?filename=stochastic_SEIR.png&userId=228444

2.3 Modelli basati su Agenti

I modelli basati su agenti rappresentano una metodologia computazionale utilizzata per simulare le azioni e le interazioni di un insieme di agenti autonomi, che possono essere individui o gruppi di individui. Lo scopo principale di tali modelli è comprendere il comportamento di un sistema e le relazioni che influenzano i suoi risultati [21].

Nella modellazione basata su agenti, un sistema viene suddiviso in un insieme di entità decisionali autonome chiamate “agenti”. Ciascun agente valuta autonomamente la propria situazione e prende decisioni basate su un insieme di regole specifiche. Le interazioni, competitive e cooperative, ripetute tra gli agenti costituiscono una caratteristica distintiva di un modello basato su agenti, sfruttando la potenza computazionale dei computer per esplorare dinamiche altrimenti difficilmente accessibili attraverso la modellazione matematica tradizionale.

In una forma di base, un modello basato su agenti consiste in un insieme di agenti e delle relazioni tra di essi. Anche un modello estremamente semplice può rivelare comportamenti complessi e offrire informazioni preziose sulle dinamiche del mondo reale che stanno modellando.

Generalmente, gli agenti possono evolvere nel tempo, consentendo di osservare comportamenti precedentemente inaccessibili, noti come “comportamenti emergenti”. I modelli più sofisticati spesso incorporano reti neurali, algoritmi evolutivi o altre tecniche di apprendimento per rendere la simulazione il più realistica possibile, tentando di incentivare la comparsa di queste tipologie di comportamenti.

L’uso di modelli basati su agenti in epidemiologia è noto da diversi decenni ed è stato utilizzato per simulare e comprendere una vasta gamma di problemi, spesso centrati sul comportamento umano come parametro chiave [22] [23] [24] [25]. Uno dei parametri più rilevanti che è stato preso in considerazione è stato il comportamento sociale degli individui. Questo insieme di parametri comprende diverse interazioni specifiche, che possono essere raggruppate in macrocategorie se necessario.

Con l’avvento della pandemia di COVID-19, molti ricercatori si sono concentrati sulla creazione di modelli basati su agenti, sia puri che ibridi con equazioni differenziali, al fine di sviluppare simulazioni affidabili del decorso di una pandemia, tenendo conto di variabili stocastiche e imprevedibili come il comportamento umano. Questo approccio ha lo scopo di ottenere intuizioni sul comportamento emergente del sistema.

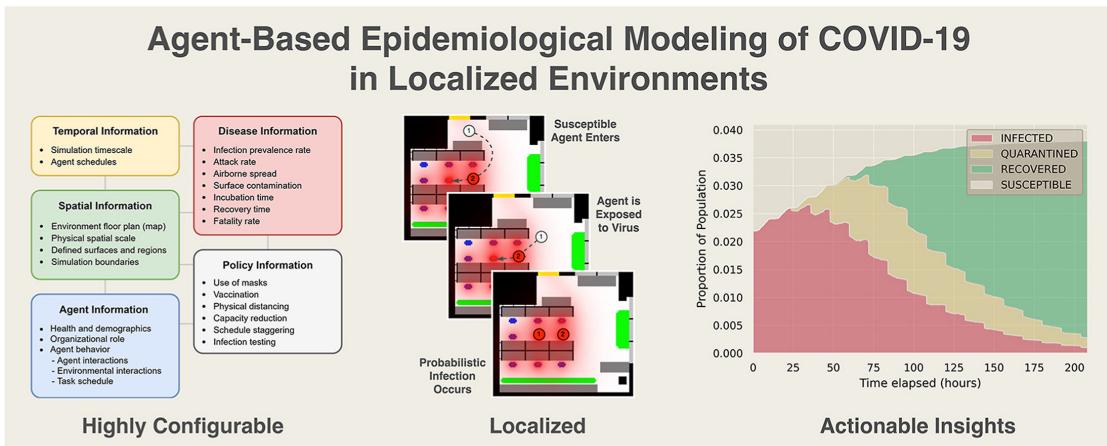


Figure 5: Agent-Based Epidemiological Modeling of COVID-19 in Localized Environment [2]

2.3.1 Comportamento Emergente

I fenomeni emergenti sono il risultato delle interazioni tra singole entità. Non possono essere spiegati completamente attraverso le proprietà intrinseche delle singole parti, poiché dipendono dalle interazioni tra tali parti. Il comportamento emergente può manifestare proprietà separate dalle singole parti, rendendo la comprensione e la previsione del comportamento del sistema complesse, spesso contrarie all'intuito.

Un modello basato su agenti genera comportamenti emergenti dal basso verso l'alto, cioè dai singoli agenti all'intero gruppo. Ciò solleva domande sull'essenza stessa del comportamento emergente.

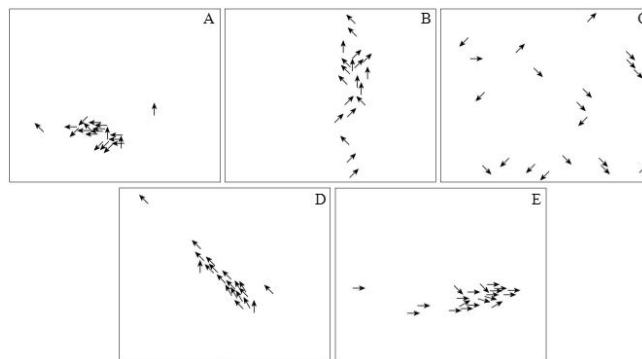


Figure 6: Esempio di comportamento emergente nella simulazione degli stormi di uccelli
<https://www.jasss.org/13/2/8/Figure6b.jpg>

2.3.2 Discretizzazione

La discretizzazione è una delle sfide fondamentali nella simulazione. Il mondo reale è costituito sia da fenomeni continui che da fenomeni discreti; solo i primi di reale interesse in questa sezione.

Gli strumenti a nostra disposizione sono in grado solamente di rappresentare e operare su dati discreti, ed è quindi necessario decidere come adattare la realtà alla simulazione, con il rischio di perdere informazioni nel processo.

La discretizzazione in una simulazione riguarda principalmente due aspetti: lo spazio e il tempo. Molti framework di simulazione offrono la possibilità di specificare come gestire la discretizzazione, tuttavia, la scelta della discretizzazione può influenzare significativamente l'adeguatezza della simulazione per un dato contesto.

Va notato che la discretizzazione non è sempre negativa e che alcune simulazioni possono fornire risultati accurati, se non migliori, anche con una discretizzazione significativa. Un esempio tra tutti è l'algoritmo di Gillespie [26], il quale è più preciso ed è in grado di rivelare comportamenti non visibili tramite l'utilizzo di modelli continui. La scelta della discretizzazione dipende spesso dalla specifica applicazione e dall'obiettivo della simulazione, nonché dai mezzi tecnici a disposizione.

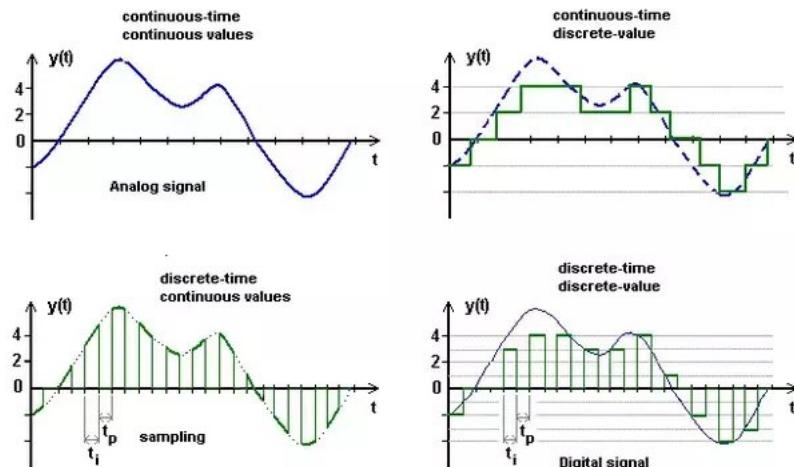


Figure 7: Esempio di differenti tipologie di discretizzazione, siano esse nel tempo o nei valori

<https://qph.cf2.quoracdn.net/main-qimg-100fc99cfe855462247225a07e1dfb7e-pj1q>

2.4 Julia

Il linguaggio di programmazione Julia rappresenta un sistema di alto livello, multi-paradigma e open-source concepito soprattutto per la soluzione di problemi di “Scientific Computing”. Julia è emerso come linguaggio di programmazione nel 2012 con l’obiettivo di fornire uno strumento in grado di rivaleggiare i linguaggi di riferimento nel settore, come C e Fortran, in termini di velocità e affidabilità. Inoltre, Julia è stato progettato anche per essere di facile accesso anche a coloro che non possiedono una solida base di programmazione. Il linguaggio è stato sviluppato principalmente in C++ e Scheme, ma gran parte del suo ecosistema è stato scritto in Julia stesso.

Le principali caratteristiche di Julia sono:

- **Elevate prestazioni:** Julia è stato creato con l’obiettivo di offrire prestazioni eccezionali, con la capacità di compilare programmi in codice nativo per diverse piattaforme, sfruttando LLVM.
- **Dinamismo:** La scelta di adottare una tipizzazione dinamica rende Julia più accessibile a chiunque, anche a coloro che non hanno una solida base di programmazione. Ciò offre anche un alto supporto per l’interazione in tempo reale.
- **Ambiente riproducibile:** Julia mira a garantire che le condizioni di esecuzione di un programma siano ricreabili su qualsiasi macchina.
- **Componibilità:** Julia fa uso dell’approccio del “multiple dispatch” come paradigma di programmazione, consentendo una grande flessibilità nella rappresentazione di una vasta gamma di pattern di programmazione, dall’orientato agli oggetti a quello funzionale.
- **General-purpose:** Julia mira a creare un ecosistema in grado di soddisfare qualsiasi esigenza dell’utente, consentendo la creazione di applicazioni e microservizi senza la necessità di ricorrere a integrazioni con codice non nativo di Julia.
- **Open source:** Julia abbraccia la filosofia open source, con il codice sorgente del linguaggio e di tutte le librerie disponibili su GitHub sotto la licenza MIT. Questo favorisce una crescita eterogenea grazie al contributo di oltre 1000 utenti impegnati nel migliorare il linguaggio.

2.4.1 Agents.jl

In linea con la filosofia di sviluppo di Julia, la libreria Agents.jl [3] è stata progettata con l’obiettivo di essere facilmente apprendibile, estendibile e di offrire modelli di simulazione veloci e scalabili. Numerosi confronti hanno dimostrato che questa libreria offre significativi vantaggi prestazionali rispetto ai principali concorrenti presenti sul mercato, come Mesa, NetLogo e MASON [4].

La facilità d’uso di Agents.jl non va confusa con una mancanza di opzioni di sviluppo, poiché la libreria consente l’integrazione con altre librerie in modo altrettanto semplice

e rapido. In particolare, offre un supporto per l'uso di machine learning, in particolare nel campo del Scientific Machine Learning [27], un campo che ha suscitato un crescente interesse, soprattutto a causa della pandemia da Covid-19.

Agents.jl offre diverse opzioni di configurazione, ma si basa principalmente su due principi:

- **Definizione del tipo di agente:** È generalmente consigliato estendere la tipologia StandardABM, che rappresenta l'implementazione predefinita di un modello basato su agenti. Questo tipo consente la creazione di un `AgentBasedModel`.
- **Definizione del tipo di spazio:** Sono supportati principalmente due tipi di spazio: spazio discreto a grafo [28] e spazio discreto a griglia. Entrambi richiedono attributi specifici per rappresentare la posizione degli agenti nello spazio.

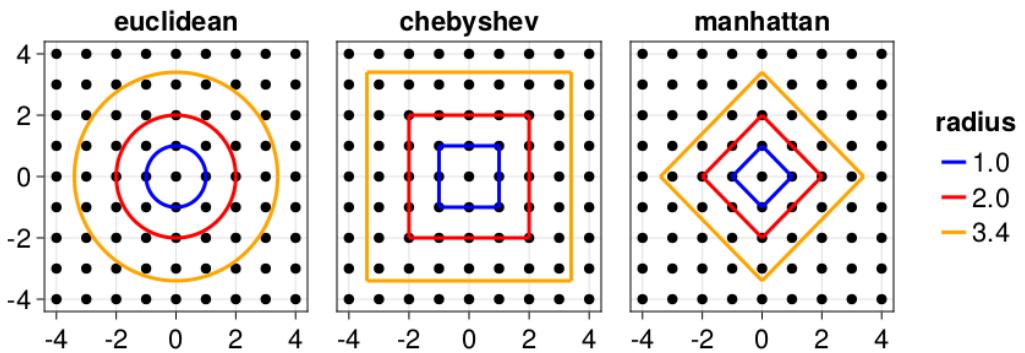


Figure 8: Esempio di metriche per il calcolo della distanza su uno spazio a griglia [3]

Inoltre, è possibile utilizzare uno spazio continuo, che rappresenta uno spazio di dimensione variabile con attributi di posizione e velocità. Infine, esiste uno spazio misto, chiamato `OpenStreetMapSpace`, che rappresenta mappe come entità continue.

2.4.2 Benchmarking e confronto con altri linguaggi

Uno degli aspetti di maggior rilevanza nel contesto di Julia sono le sue eccellenti prestazioni rispetto ad altri linguaggi di programmazione. Questo è stato dimostrato in diversi benchmark, in cui Julia ha superato linguaggi come Python, R e MATLAB [4] in termini di velocità computazionale. Questo vantaggio è particolarmente importante nelle simulazioni basate su agenti, in cui la velocità di esecuzione può fare la differenza tra la fattibilità e l'inapplicabilità di un modello.

Il confronto con Python è particolarmente rilevante, dato che Python è ampiamente utilizzato in campo scientifico e computazionale. Mentre Python offre una vasta gamma di librerie e strumenti, il suo utilizzo in simulazioni basate su agenti può essere limitato dalla sua lentezza. Julia, grazie alla sua architettura e alle sue librerie ottimizzate, riesce

a superare questa limitazione senza compromettere la facilità d'uso [5] [6] [27] [29] [30] [31].

In conclusione, Julia e la libreria Agents.jl rappresentano strumenti potenti per la creazione di modelli di simulazione basati su agenti che richiedono elevate prestazioni computazionali. La combinazione di un linguaggio di alto livello, come Julia, con una libreria specializzata come `Agents.jl`, offre un ambiente ideale per la progettazione e lo sviluppo di simulazioni avanzate in una vasta gamma di settori.

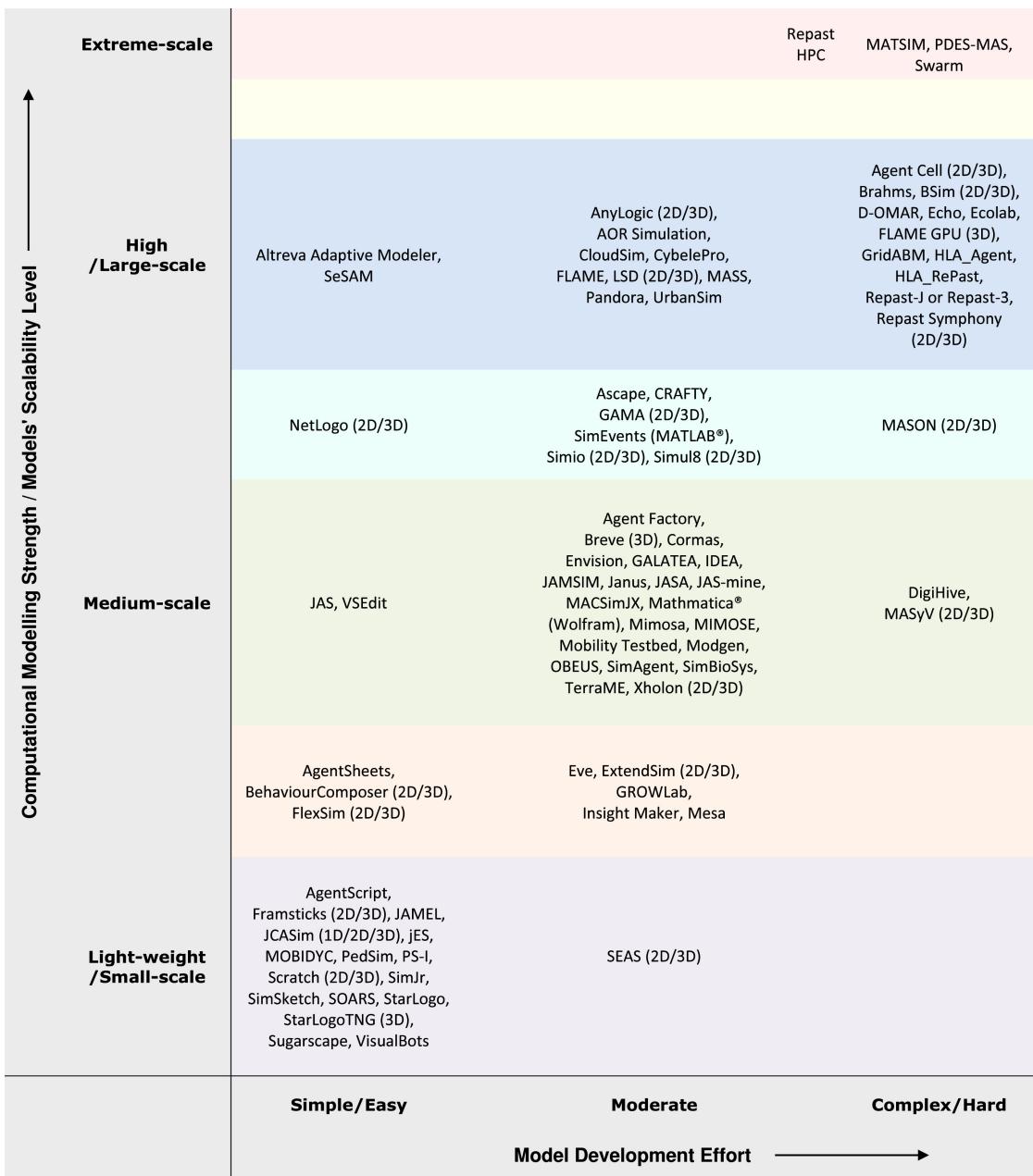


Figure 9: Agent Based Modelling and Simulation tools: A review of the state-of-art software [4]

2.5 Equazioni Neurali Differenziali

La tecnica delle Equazioni Neurali Differenziali (END) ha attirato notevole attenzione, portando all'ibridazione di due paradigmi di modellazione distinti: le Equazioni Differenziali Ordinarie (ODE) e le reti neurali (NN). Questo sforzo mira a sfruttare al meglio entrambi i paradigmi, minimizzando gli effetti indesiderati [32] [33].

Un'Equazione Differenziale è un metodo per specificare una trasformazione non lineare arbitraria, codificando matematicamente le ipotesi strutturali a priori del sistema. Esistono tre approcci comuni per definire tale trasformazione non lineare:

- **Modellazione diretta**
- **Machine learning**
- **Equazioni differenziali**

L'approccio di modellazione diretta funziona solo quando si conosce la funzione esatta che collega l'input con l'output. Tuttavia, nella maggior parte dei casi, questa relazione è sconosciuta a priori, rendendo impossibile applicare questo metodo. In questi casi, l'approccio di machine learning diventa una soluzione valida.

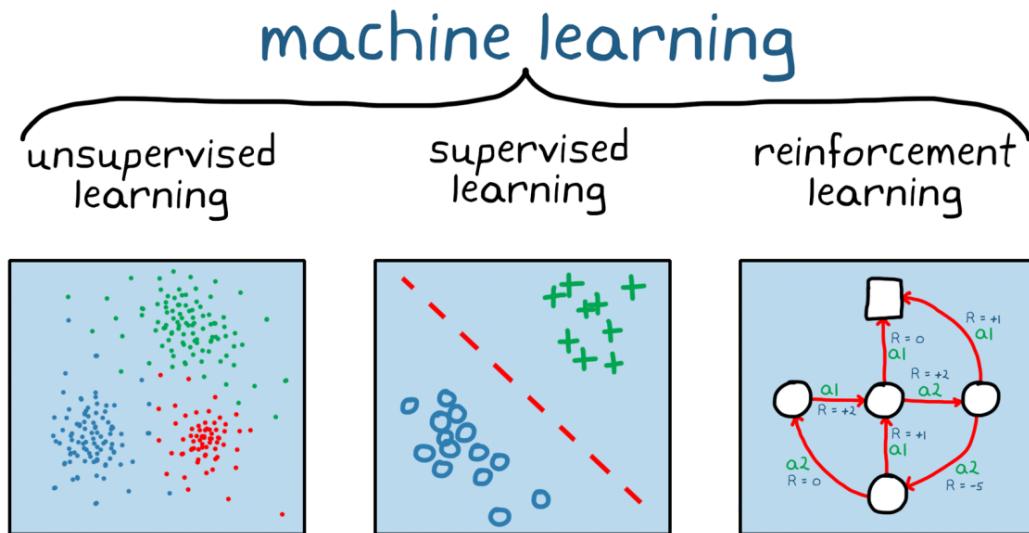


Figure 10: Differenti approcci di apprendimento di machine learning
https://it.mathworks.com/discovery/reinforcement-learning/_jcr_content/mainParsys3/discoverysubsection/mainParsys/image.adapt.full.medium.png/1689243301131.png

Nel contesto generale del machine learning, l'obiettivo è sviluppare algoritmi e modelli che possano apprendere dai dati e migliorare le prestazioni su specifici compiti senza es-

sere programmati esplicitamente. Questa funzione viene comunemente chiamata “modello”. Alcuni degli obiettivi chiave del machine learning sono:

- **Predizione:** Uno degli obiettivi più comuni è creare modelli in grado di fare previsioni accurate. Ad esempio, si possono creare modelli di previsione del tempo, modelli di previsione di vendite, o modelli di previsione della probabilità di un paziente di sviluppare una determinata malattia.
- **Classificazione:** Questo obiettivo riguarda la capacità di assegnare oggetti o dati a categorie specifiche. Ad esempio, un modello di machine learning può essere addestrato per classificare email in “spam” o “non spam”, o per riconoscere oggetti in immagini, come gatti e cani.
- **Clustering:** Gli algoritmi di clustering cercano di trovare pattern nei dati e raggruppare gli oggetti simili. Questo è utile per identificare segmenti di clientela, raggruppare documenti simili e molto altro.
- **Ottimizzazione e pianificazione:** L’obiettivo può essere quello di sviluppare modelli di machine learning che aiutino a prendere decisioni ottimali in situazioni complesse, come la pianificazione dei percorsi per veicoli autonomi o l’ottimizzazione della gestione della catena di approvvigionamento.

Si inizia con una fase di addestramento in cui si cercano di regolare i parametri (iperparametri) del modello per ottenere un modello in grado di generare previsioni accurate. Successivamente, il modello viene utilizzato per inferire dati x mai visti in precedenza.

L’uso del machine learning si basa su un concetto estremamente semplice ma potente: adattare il modello ai dati forniti in modo efficace. Questa idea si estende ulteriormente alle reti neurali (NN), che sono essenzialmente insiemi di moltiplicazioni tra matrici seguite dall’applicazione di una funzione di attivazione. Ad esempio, una semplice rete neurale a tre strati è definita come:

$$\text{ML}(x) = \sigma(W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x)))$$

Dove W_1 , W_2 , e W_3 sono parametri apprendibili. L’obiettivo è selezionare questi parametri in modo che $\text{ML}(x) = y$ si comporti in modo simile alla funzione incognita che si desidera adattare. Grazie al teorema di approssimazione universale [34], si afferma che con un numero sufficientemente grande di parametri o strati, è possibile approssimare qualsiasi funzione non lineare in modo sufficientemente preciso.

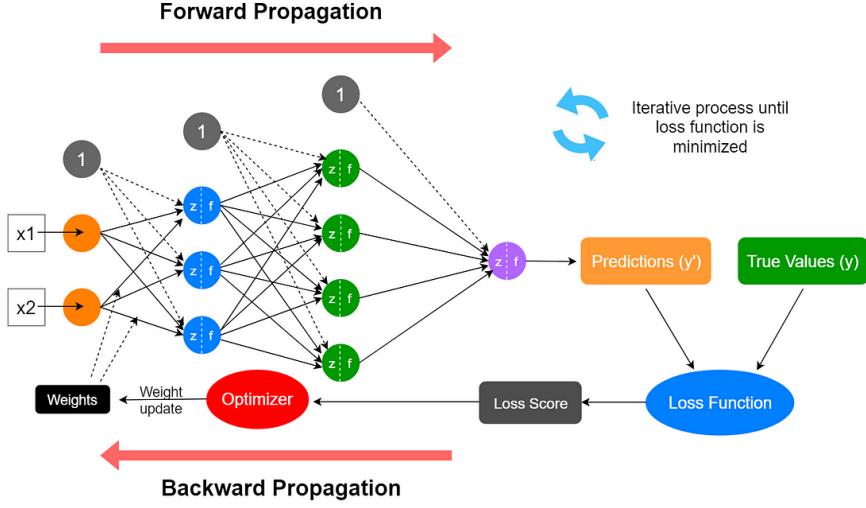


Figure 11: Esempio di funzionamento di una rete neurale
https://miro.medium.com/v2/resize:fit:4800/format:webp/1*ZXAOUqmlyECgfvA81Sr6Ew.png

Questo approccio richiede l'apprendimento di ogni aspetto della trasformazione non lineare direttamente dai dati disponibili. In molti casi, però, non è possibile conoscere l'intera equazione non lineare, ma forse se ne conosce la struttura generale. Un modo per utilizzare questa conoscenza pregressa - la struttura generale della funzione o i suoi vincoli - è di ricorrere di nuovo alle Equazioni Differenziali (DE). Un metodo immediato è quello di definire un modello matematico in cui si cerca di apprendere una costante associata al comportamento di un insieme di dati:

$$y'(t) = \alpha \cdot y(t)$$

Questo approccio non richiede la conoscenza della soluzione dell'equazione differenziale per convalidare la correttezza del modello. La struttura del modello e la matematica stessa sono incorporate nel modello stesso, il quale successivamente produce una soluzione. Questo tipo di modelli è essenzialmente un insieme di equazioni che descrivono il cambiamento delle variabili nel tempo, con le soluzioni che dipendono dal parametro α . Per riuscire ad adattare il modello ai dati si utilizzano varie tecniche di *parameter sweep* [35].

I metodi di approssimazione, predizione o classificazione basati su ML richiedono una grande mole di dati per poter essere addestrati in modo soddisfacente, eseguendo, de-facto, quella che è un'operazione di parameter sweep. In questo contesto, l'uso delle equazioni differenziali neurali è diventato una scelta interessante per specificare la non

linearità in modo apprendibile (tramite parametri) in modo più efficiente. Questo permette di incorporare la conoscenza specifica di un dominio nelle relazioni strutturali tra input e output del modello.

Un'Equazione Differenziale Neurale (Neural ODE o NDE) rappresenta un modo per collegare i mondi del machine learning e delle equazioni differenziali. L'approccio generale è quello di non apprendere solo la trasformazione non lineare tra i dati, ma anche la struttura stessa della trasformazione non lineare. Invece di avere il modello $y = \text{ML}(x)$, si ha il modello $y' = \text{ML}(x)$, e successivamente si tenta di risolvere l'equazione differenziale associata. Le NDE utilizzano una rete neurale per approssimare la funzione $f(x, t)$ nell'ODE. Invece di specificare esplicitamente $f(x, t)$, si definisce una rete neurale che apprende questa funzione. Quindi, l'ODE diventa: $\frac{dx}{dt} = \text{NN}(x, t; \theta)$, dove $\text{NN}(x, t; \theta)$ rappresenta la rete neurale con i suoi parametri θ .

L'approccio fondamentale qui è che definendo il modello in questo modo e utilizzando un risolutore semplice e propenso agli errori come il metodo di Eulero, è possibile ottenere risultati equivalenti a quelli ottenuti con una ResNet [36]. L'idea di base è che invece di modellare una rete neurale con sempre più strati, diventando sempre più profonda, è sufficiente modellare direttamente il sistema di equazioni differenziali e risolverlo con un risolutore specifico.

Le NDE si basano sull'idea di integrazione continua, il che significa che piuttosto che discretizzare il tempo o la variabile indipendente, l'ODE viene risolto in modo continuo. Ciò consente una rappresentazione più fluida dei dati temporali e una maggiore flessibilità nel modellare cambiamenti graduati nel tempo.

L'addestramento di una NDE comporta la soluzione dell'ODE usando un algoritmo di ottimizzazione come la discesa del gradiente. I dati osservati sono utilizzati per calibrare i parametri della rete neurale in modo che l'ODE approssimi correttamente la dinamica dei dati di input. Ciò significa che le NDE possono essere addestrate su dati sequenziali o temporali per prevedere future evoluzioni.

Le NDE sono particolarmente adatte per affrontare problemi in cui la dinamica dei dati può variare in modo non uniforme nel tempo e in cui i dati sequenziali possono essere incompleti o rumorosi, ha una forte conoscenza a priori dello spazio del modello, è in grado di approssimare sia funzioni lineari che non lineari e si basa su solide basi teoriche derivanti da entrambi i lati.

Sono state utilizzate in vari campi, tra cui l'elaborazione del linguaggio naturale, la previsione del traffico e la modellazione di processi biologici. La loro flessibilità e capacità di modellare dati continui le rendono una potente aggiunta al campo del machine learning.

2.5.1 Equazioni Differenziali Universali

Recentemente, gli sviluppi nel campo del machine learning sono stati guidati dalle tecniche di deep learning, che richiedono grandi quantità di dati, noti come “big data”, per

risolvere problemi precedentemente considerati difficili e complessi, come il riconoscimento di immagini o il processing del linguaggio naturale.

Tuttavia, in molti settori, specialmente nella medicina e in altre discipline correlate, è difficile ottenere un insieme di dati sufficientemente ampio e diversificato per applicare queste tecniche. In questi contesti, i modelli meccanicistici rimangono la scelta principale. L'approccio data-driven dei modelli di machine learning, tuttavia, offre maggiore flessibilità e la possibilità di evitare ipotesi semplificative richieste dai modelli teorici.

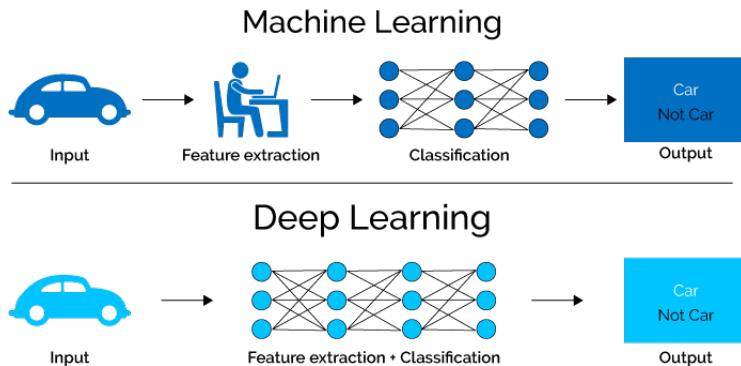


Figure 12: Esempio comparativo tra funzionamento Machine Learning e Deep Learning
https://goodboychan.github.io/images/copied_from_nb/image/fe.png

Un'Equazione Differenziale Universale (UDE) è definita da un “approssimatore universale”, un oggetto parametrico in grado di rappresentare qualsiasi funzione dati un certo numero di parametri. Gli approssimatori universali includono, ad esempio, le serie di Fourier o di Chebyshev per spazi a basse dimensioni e le reti neurali per spazi ad alta dimensione [5].

Un'UDE è un sistema di equazioni differenziali o algebriche non triviali che può approssimare qualsiasi funzione continua su un intervallo a qualsiasi livello di precisione desiderato. In altre parole, un'UDE è in grado di rappresentare qualsiasi sistema dinamico continuo con qualsiasi grado di dettaglio richiesto.

L'approccio delle UDE può essere accoppiato con tecniche data-driven, come l'algoritmo SINDy (Sparse Identification of Non-linear Dynamics), per l'identificazione sparsa di dinamiche non lineari in sistemi complessi come dinamiche dei fluidi o reti biologiche [37].

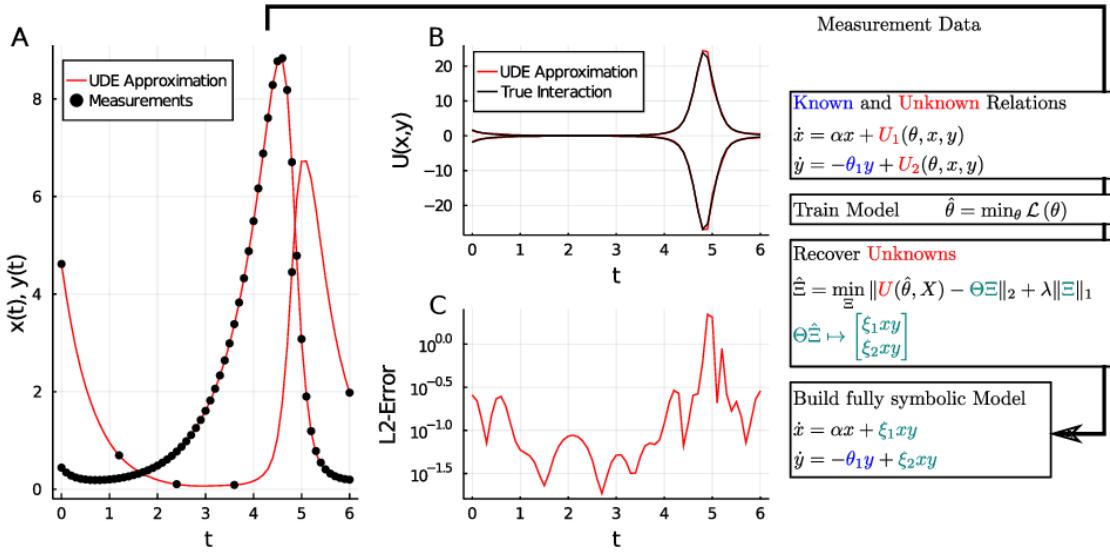


Figure 13: Comportamento UDE nell'approssimazione di fenomeni non lineari [5]

Risolvere un'ODE in Julia

Per risolvere una ODE in Julia si definisce un oggetto `ODEProblem` come una funzione che descrive il sistema di ODE attraverso la specifica delle derivate delle equazioni nella forma $u' = f(u, p, t)$, fornendo un set di condizioni iniziali u , un intervallo di tempo t , e un insieme di parametri p .

Successivamente, è possibile risolvere il sistema di ODE utilizzando la funzione “solve”. È possibile specificare diversi metodi per la risoluzione del sistema. Ad esempio, il metodo “Tsit5” è un metodo esplicito di quinto ordine di tipo Runge-Kutta con un stimatore di errore integrato di tipo Tsitouras [38]. La libreria `DifferentialEquations.jl` offre una vasta gamma di parametri aggiuntivi per un approccio più dettagliato [27].



```

1  function seir!(du, u, p, t)
2      S, E, I, R, D = u
3      R₀, γ, σ, ω, δ, η, ξ = p
4      μ = δ / 1111
5      du[1] = μ * sum(u) - R₀ * γ * (1 - η) * S * I + ω * R - ξ * S - μ * S # dS
6      du[2] = R₀ * γ * (1 - η) * S * I - σ * E - μ * E # dE
7      du[3] = σ * E - γ * I - δ * I - μ * I # dI
8      du[4] = (1 - δ) * γ * I - ω * R + ξ * S - μ * R # dR
9      du[5] = δ * γ * I # dD
10 end
11 u = [0.999, 0.0, 0.001, 0.0, 0.0]
12 p = [3.54, 1 / 14, 1 / 5, 1 / 280, 0.01, 0.0, 0.0]
13 tspan = (0.0, 30.0)
14 prob = ODEProblem(seir!, u, tspan, p)
15 sol = solve(prob, Tsit5())

```

Figure 14: Esempio definizione ODE in Julia

2.5.2 Inserire un’ODE all’interno di una NN

Per capire meglio cosa significhi inserire un’ODE all’interno di una rete neurale (NN), è necessario esaminare come è definito un layer di una NN. Un layer è essenzialmente una funzione differenziabile che accetta un vettore di dimensione n come input e restituisce un nuovo vettore di dimensione m come output. Questo si allinea con l’idea che i risolutori di DE rientrano nella categoria delle funzioni differenziabili, il che significa che possono essere incorporati direttamente in un programma più grande basato su differenziazione automatica [39], come una rete neurale [29] [31].

Successivamente, è possibile addestrare la NN per un numero specifico di integrazioni per ottenere i risultati desiderati. È importante sottolineare che il sistema non sta apprendendo una soluzione all’equazione differenziale, ma piuttosto sta apprendendo il sistema di equazioni differenziali da cui è generata la soluzione. La NN sta effettivamente apprendendo una rappresentazione compatta del comportamento della serie di dati nel tempo e può estrapolare facilmente cosa potrebbe accadere con diverse condizioni iniziali.

La suite `DiffEqFlux.jl` offre un wrapper comodo per definire una “NeuralODE” [6].

```

● ● ●

1  using ComponentArrays, Lux, DiffEqFlux
2  using DifferentialEquations, Optimization
3  using OptimizationOptimJL, OptimizationFlux
4  using Random, Plots
5
6  rng = Random.default_rng()
7  u0 = Float32[2.0; 0.0]
8  datasize = 30
9  tspan = (0.0f0, 1.5f0)
10 tsteps = range(tspan[1], tspan[2], length=datasize)
11
12 function trueODEfunc(du, u, p, t)
13     true_A = [-0.1 2.0; -2.0 -0.1]
14     du .= ((u .^ 3)'true_A)'
15 end
16
17 prob_trueode = ODEProblem(trueODEfunc, u0, tspan)
18 ode_data = Array(solve(prob_trueode, Tsit5(), saveat=tsteps))
19
20 dudt2 = Lux.Chain(x -> x .^ 3,
21                     Lux.Dense(2, 50, tanh),
22                     Lux.Dense(50, 2))
23 p, st = Lux.setup(rng, dudt2)
24 prob_neuralode = NeuralODE(dudt2, tspan, Tsit5(), saveat=tsteps)
25
26 function predict_neuralode(p)
27     Array(prob_neuralode(u0, p, st)[1])
28 end
29
30 function loss_neuralode(p)
31     pred = predict_neuralode(p)
32     loss = sum(abs2, ode_data .- pred)
33     return loss, pred
34 end
35
36 pinit = ComponentArray(p)
37
38 # use Optimization.jl to solve the problem
39 adtype = Optimization.AutoZygote()
40
41 optf = Optimization.OptimizationFunction((x, p) -> loss_neuralode(x), adtype)
42 optprob = Optimization.OptimizationProblem(optf, pinit)
43
44 result_neuralode = Optimization.solve(optprob,
45                                         Adam(0.05),
46                                         callback=callback,
47                                         maxiters=300)
48
49 optprob2 = remake(optprob, u0=result_neuralode.u)
50
51 result_neuralode2 = Optimization.solve(optprob2,
52                                         Optim.BFGS(initial_stepnorm=0.01),
53                                         callback=callback,
54                                         allow_f_increases=false)

```

Figure 15: Esempio implementazione di una Neural ODE in Julia tramite l'utilizzo della libreria `DiffEqFlux.jl` [6]

2.5.3 Backpropagation tramite ODE solver

Il cuore di ogni rete neurale è la capacità di propagare all'indietro le derivate lungo l'intera rete per calcolare il gradiente della funzione di perdita rispetto ai parametri della rete. La sfida sta nel trovare un modo per applicare lo stesso principio quando si utilizzano risolutori di ODE all'interno di una rete neurale.

Esistono vari approcci per affrontare questo problema, ma il più comune è attraverso l'analisi di sensitività definita come “Forward Adjoint” [32] [40]. Questo approccio definisce una nuova ODE il cui obiettivo è calcolare il gradiente della funzione di costo rispetto ai parametri e successivamente risolvere questa seconda ODE.

L'approccio dell'analisi di sensitività (Forward Adjoint) comporta la definizione di una nuova ODE, chiamata anche “ODE aggiunta” o “ODE ausiliaria”, il cui scopo principale è calcolare il gradiente della funzione di costo rispetto ai parametri della rete. Questa ODE ausiliaria è progettata in modo specifico per catturare come varia la funzione di costo al variare dei parametri, consentendo così il calcolo efficiente del gradiente.

Il processo può essere riassunto in modo generale nei seguenti passaggi:

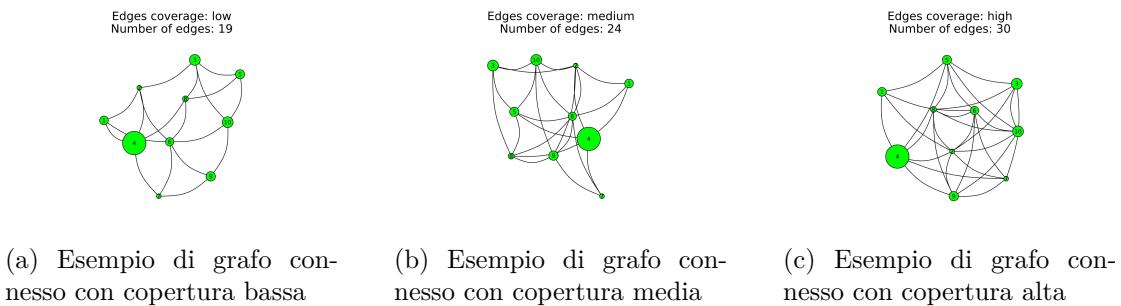
- **Forward Pass:** Durante la fase di forward pass della rete neurale, vengono registrati i valori intermedi necessari per risolvere l'ODE ausiliaria.
- **Calcolo della ODE Ausiliaria:** Durante la fase di backward pass, la ODE ausiliaria viene risolta. Questa ODE tiene conto dei valori intermedi registrati durante il forward pass e calcola il gradiente della funzione di costo rispetto ai parametri.
- **Backward Pass:** Il gradiente calcolato dalla ODE ausiliaria viene quindi utilizzato per aggiornare i parametri della rete neurale durante la fase di backward pass, consentendo l'addestramento della rete mediante la discesa del gradiente.

Questo approccio consente di calcolare il gradiente in modo efficiente anche quando sono presenti risolutori di ODE all'interno della rete. È uno strumento fondamentale nella progettazione e nell'addestramento di reti neurali che includono componenti basate su ODE, come le reti neurali differenziali (DNN).

In generale, l'analisi di sensitività è un concetto cruciale per consentire l'addestramento efficiente di modelli complessi e ibridi che combinano reti neurali con risoluzione di ODE o altre funzioni non lineari.

3 Metodi e Modelli

Nell'ambito di questa tesi, l'approccio adottato è stato incentrato sulla prospettiva di modellare il problema specifico - la modellazione di epidemie - all'interno del contesto di una rete sociale. L'obiettivo principale di questa indagine è stato quello di analizzare e comprendere il processo di diffusione di una pandemia virale all'interno di un ambiente concreto, il quale è principalmente costituito da punti di interesse (PdI) interconnessi tra loro attraverso una struttura di collegamenti, rappresentata graficamente sotto forma di un grafo.



Questo approccio si avvicina a una prospettiva di tipo mesoscopico, orientata al macroscopico. La scelta di tale approccio deriva dalla necessità di modellare il sistema in maniera più ampia, concentrandosi sulla risposta collettiva di un agente astratto agli interventi mirati per contrastare l'epidemia in modo localizzato, piuttosto che sulla risposta di ogni singolo agente concreto alle stesse tipologie di intervento.

3.1 Approccio con Rete Sociale

Il modello sviluppato fa uso del framework `Agents.jl` [3] e definisce un modello ad agente (ABM). In questo contesto, la struttura spaziale, o topologia, del modello non è rilevante, ma le connessioni tra gli agenti lo sono, per questo è stato scelto di utilizzare una rappresentazione tramite grafo sociale. Gli agenti vengono considerati come nodi all'interno di un grafo, andando a rappresentare un punto di interesse specifico, sia esso una stazione ferroviaria, un terminal aeroportuale, oppure una città, in cui viene simulato il ciclo di vita della pandemia attraverso un modello SEIR deterministico. Gli agenti sono definiti come oggetti di tipo `ContinuousAgent`, poiché lo spazio degli agenti è considerato continuo, seppur la struttura del modello sia a grafo e quindi discreta.

Grafo Sociale

Un grafo sociale è una struttura dati utilizzata per rappresentare in maniera formale le complesse relazioni sociali che sussistono tra diverse entità all'interno di una comunità o di un sistema. In questo specifico contesto, i nodi del grafo fungono da rappresentazioni delle entità coinvolte, spesso individui, mentre gli archi tra questi nodi simboleggiano le relazioni sociali che legano tali individui tra loro.

Questa metodologia di rappresentazione è comunemente adottata per modellare le reti sociali, che costituiscono un settore di grande interesse in diverse discipline, come sociologia, informatica, e analisi delle reti [41] [42] [43]. Nell'ambito delle reti sociali, gli individui stessi sono considerati i nodi del grafo, mentre le connessioni tra di essi vengono tradotte in archi. Questo approccio consente di analizzare e studiare una vasta gamma di fenomeni sociali, inclusi processi di diffusione dell'informazione, dinamiche di influenza reciproca, formazione di comunità, e molte altre interazioni sociali complesse.

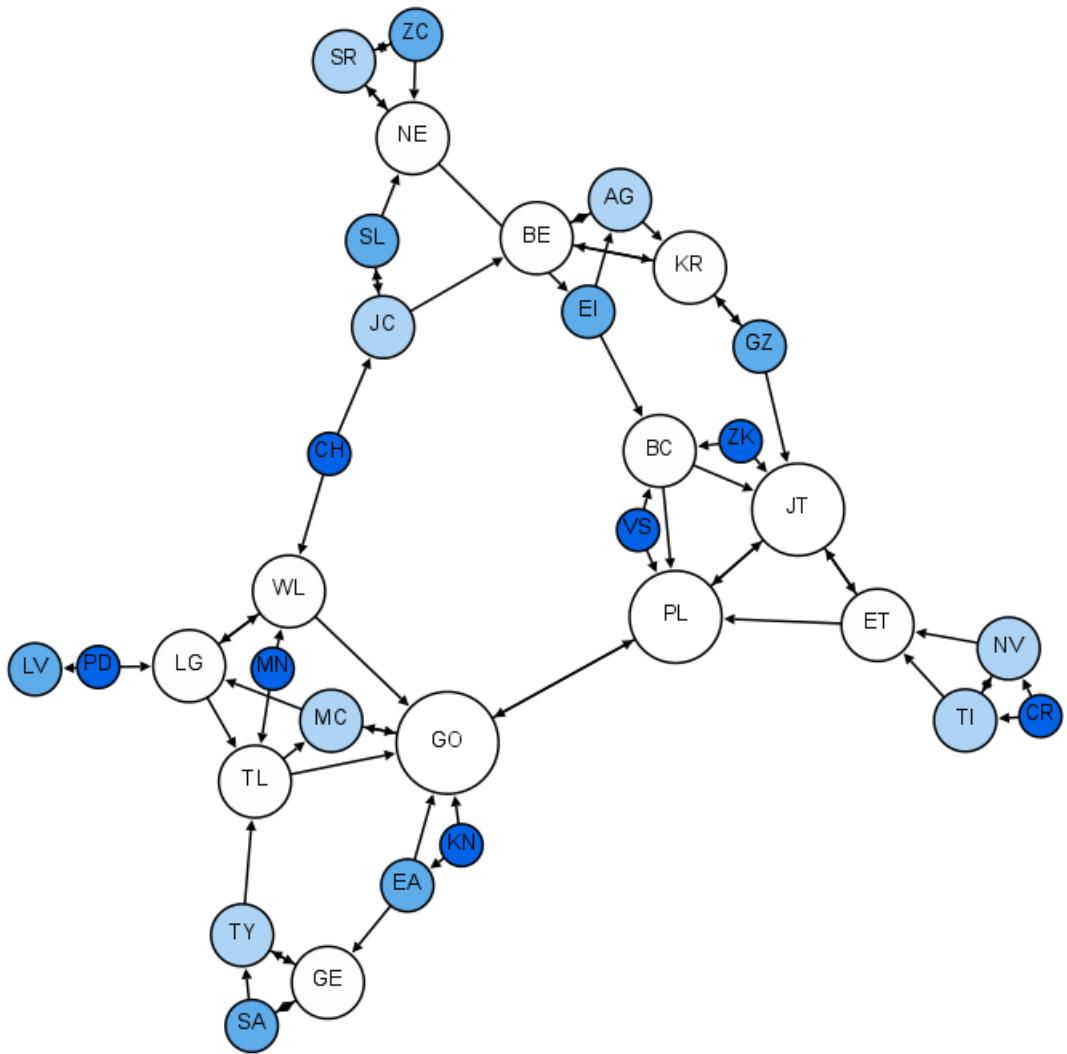


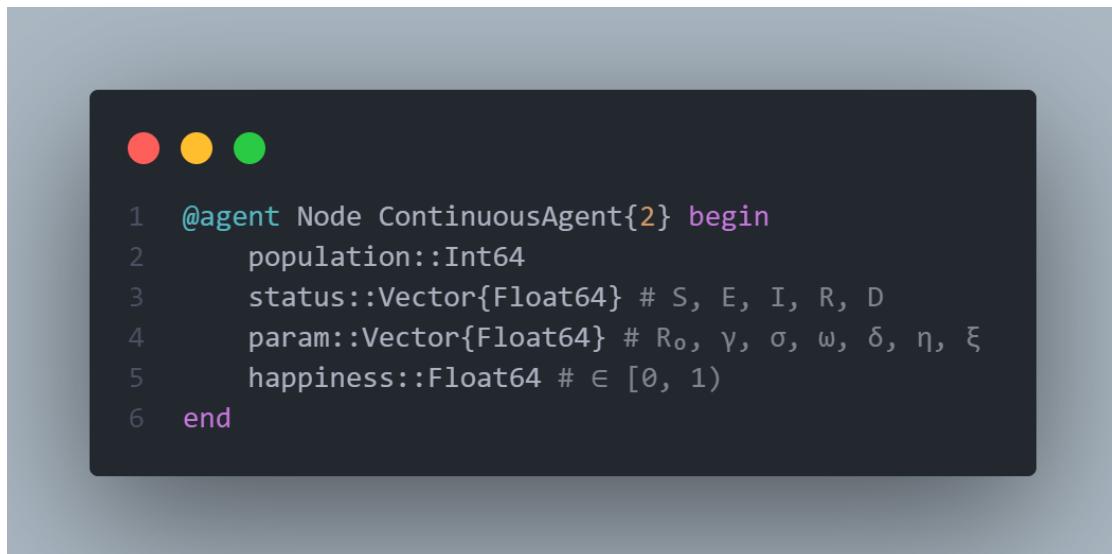
Figure 17: Grafo Sociale di una rete di classe 3
https://en.wikipedia.org/wiki/Sociogram#/media/File:Moreno_Sociogram_3rd_Grade.png

3.2 Agente

Per una comprensione più approfondita dei concetti esposti, si può esaminare ulteriormente l'implementazione dell'agente in questione. Quest'ultimo è stato concepito con una struttura minimale, ridotta ai soli attributi strettamente essenziali per il suo corretto funzionamento. Tali attributi costituiscono l'ossatura di base che consente all'agente di interagire con l'ambiente circostante e con altri agenti all'interno del sistema.

La struttura dell'agente è essenziale e include i seguenti campi principali:

- **population**: un intero che rappresenta il numero totale di individui presenti nel nodo.
- **status**: un vettore di numeri decimali che rappresenta lo stato della popolazione nel nodo, espressa come percentuale di individui.
- **param**: un vettore di numeri decimali che contiene i parametri specifici del modello SEIR per il nodo.
- **happiness**: un campo di supporto per il bilanciamento delle contromisure applicate dal controllore.



The screenshot shows a code editor window with a dark theme. At the top left, there are three colored circular icons: red, yellow, and green. The main area contains the following code:

```
1 @agent Node ContinuousAgent{2} begin
2     population::Int64
3     status::Vector{Float64} # S, E, I, R, D
4     param::Vector{Float64} # R₀, γ, σ, ω, δ, η, ξ
5     happiness::Float64 # ∈ [0, 1)
6 end
```

Figure 18: Codice Agente

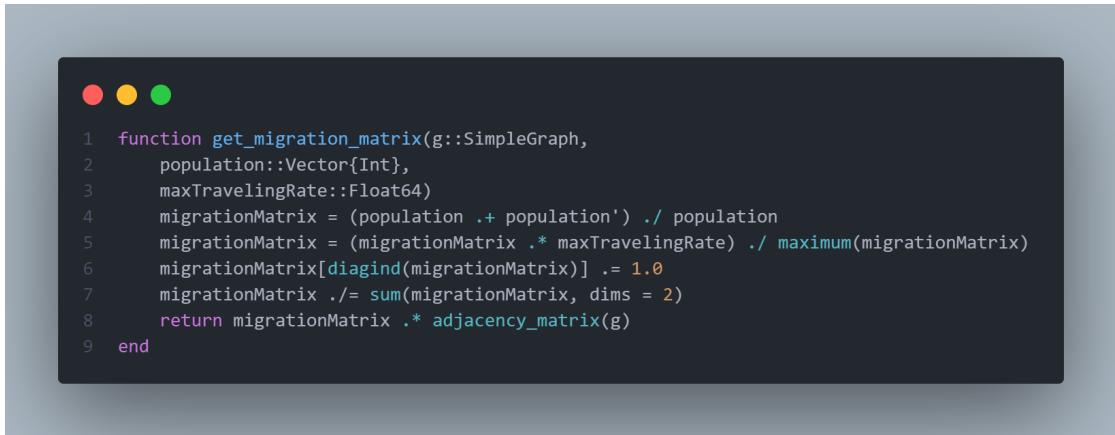
Nel contesto dell'evoluzione degli agenti, è fondamentale notare che questa si svolge in maniera indipendente rispetto agli altri agenti presenti nel medesimo contesto. Tale indipendenza è regolamentata da un sistema di equazioni differenziali ordinarie (ODE), che rappresenta il cuore del processo decisionale dell'agente. Tuttavia, è cruciale sottolineare che, nonostante questa indipendenza, l'agente non opera in un isolamento completo.

Le interazioni tra agenti sono agevolate dalle connessioni esistenti tra i nodi del grafo, che possono essere interpretate come canali di comunicazione o canali attraverso i quali fluiscono influenze e informazioni. A tal proposito è stata aggiunta in supporto, una matrice detta “di migrazione”, utile per rappresentare il flusso di informazioni (dati) che fluiscono da un agente ad un altro durante ogni passo della simulazione.

```
8x8 SparseArrays.SparseMatrixCSC{Float64, Int64} with 36 stored entries:
 0.000260307 . 0.000229864 0.00029455 0.000240596 .
 0.000198612 . 0.000267355 . 0.000210296 . 0.000193921
 . 0.000194772 . 0.000213806 . 0.000127024 .
 0.000220886 . 0.000287449 0.000235602 0.000137423 0.000214978
 0.00018246 . 0.000238272 0.000185299 . 0.000191947 .
 0.000211826 0.000242663 . 0.000215861 0.000272811 . 0.000135356 0.000206413
 . 0.000994875 0.000622588 . 0.000669306 . 0.00057587
 . 0.000268816 . 0.000236618 0.000305034 0.000247968 0.000139906 .
```

Figure 19: Esempio di matrice di migrazione in una rete con 8 nodi

La matrice di migrazione rappresenta un elemento importante in questo contesto. Essa definisce come l’agente può essere influenzato da altri agenti all’interno della stessa rete. Questo meccanismo consente un grado di adattamento e cooperazione tra agenti, sebbene ognuno di essi mantenga la propria autonomia decisionale.

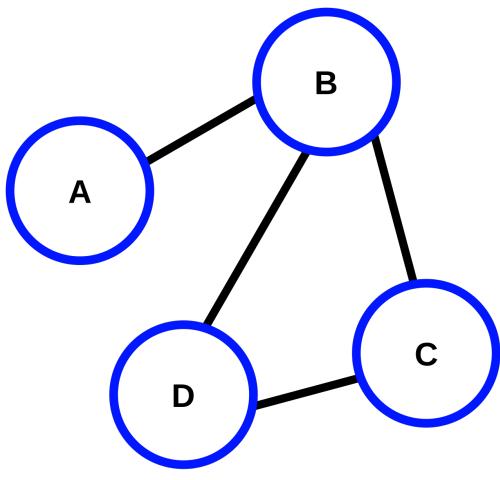


```
1 function get_migration_matrix(g::SimpleGraph,
2     population::Vector{Int},
3     maxTravelingRate::Float64)
4     migrationMatrix = (population .+ population') ./ population
5     migrationMatrix = (migrationMatrix .* maxTravelingRate) ./ maximum(migrationMatrix)
6     migrationMatrix[diagind(migrationMatrix)] .= 1.0
7     migrationMatrix ./= sum(migrationMatrix, dims = 2)
8     return migrationMatrix .* adjacency_matrix(g)
9 end
```

Figure 20: Funzione che crea la matrice di migrazione data la topologia di un grafo

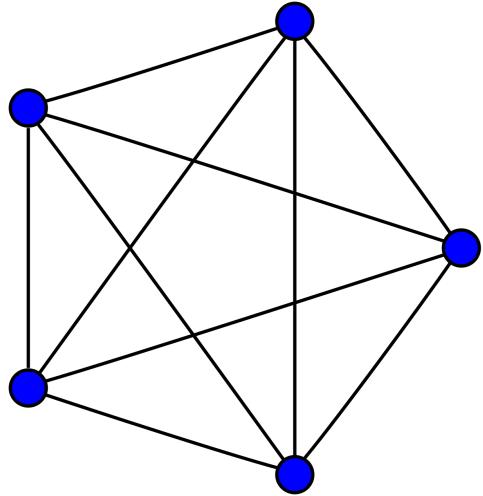
3.3 Spazio e Modello

Lo spazio del modello è fondato su un grafo connesso, il cui numero di archi è determinato in base alla copertura desiderata degli archi. La topologia specifica del grafo viene creata in conformità alle preferenze dell'utente, rispettando un limite inferiore imposto dalla connessione minima e un limite superiore rappresentato dalla connessione completa.



(a) Esempio di grafo connesso

https://it.wikipedia.org/wiki/Grafo_connesso#/media/File:CPT-Graphs-undirected-unweighted-ex1.svg



(b) Esempio di grafo completo

https://en.wikipedia.org/wiki/Complete_graph#/media/File:4-simplex_graph.svg

In altre parole, il grafo che costituisce il contesto spaziale del modello è progettato in modo dinamico per adattarsi alle esigenze e alle scelte dell'utente. Queste preferenze determinano sia il grado di interconnessione tra gli agenti (nodi del grafo) sia la forma generale del grafo stesso. Il limite inferiore garantisce una connessione minima tra gli agenti, mentre il limite superiore consentirebbe la massima interconnessione possibile, ovvero una situazione in cui ogni agente è connesso a tutti gli altri.

Questo approccio flessibile alla creazione della topologia del grafo permette agli utenti di configurare il modello in modo adatto alle specifiche condizioni e alle variabili in gioco, fornendo così un livello di personalizzazione che può essere fondamentale per la comprensione e la modellazione dei fenomeni legati alla diffusione della pandemia virale all'interno del sistema.

3.4 Funzione di Avanzamento Agente

Ogni agente all'interno del modello adotta un ciclo di comportamento che incorpora tre compiti fondamentali:

- **Spostamento tra i nodi del grafo** [44]: Gli agenti seguono un sistema di spostamento all'interno della rete, il quale può essere influenzato da fattori variabili, come ad esempio la topologia del grafo, la preferenza dell'agente, o dinamiche specifiche del modello. Questo aspetto del comportamento dell'agente è influenzato da un processo di spostamento tra i nodi del grafo, e tali movimenti possono essere utilizzati per modellare gli spostamenti fisici degli individui in un contesto di pandemia.
- **Calcolo del proprio “livello di felicità”**: Gli agenti effettuano un calcolo che valuta il loro “livello di felicità”. Questo parametro può essere influenzato da molteplici fattori, inclusi i dati relativi allo stato di salute, le misure di intervento adottate, le connessioni sociali e le condizioni ambientali. Il livello di felicità costituisce un indicatore importante per comprendere la percezione e il benessere degli agenti nell'ambiente simulato.
- **Chiamata al controllore per valutare le misure di intervento necessarie**: Gli agenti comunicano con il controllore al fine di valutare le misure di intervento necessarie. Questo processo di valutazione considera l'evoluzione della pandemia, il comportamento degli agenti e i parametri specifici del modello SEIR, tra altri fattori. Le misure di intervento possono comprendere restrizioni, vaccinazioni, consigli alla popolazione o altre strategie atte a mitigare la diffusione del virus.

Questi tre compiti costituiscono una sequenza chiave di attività che ogni agente esegue ciclicamente all'interno del modello, contribuendo in modo significativo all'analisi della diffusione della pandemia e alla valutazione delle misure di intervento.



```
1  function agent_step!(agent, model::ABM)
2      migrate!(agent, model)
3      happiness!(agent)
4      model.control ? control!(agent, model; model.control_options...) : nothing
5      agent.param[7] > 0.0 ? spread_vaccine!(agent, model) : nothing
6  end
```

Figure 22: Funzione che si occupa di gestire la simulazione di ogni singolo agente presente all'interno del modello

```

● ● ●

1  function migrate!(agent, model::ABM)
2      # get the connections and weights of the matrix
3      network = model.migrationMatrix[agent.id, :]
4      tidxs, tweights = findnz(network)
5
6      for i in eachindex(tidxs)
7          ap = agent.population
8          as = agent.status
9
10         out = as .* tweights[i] .* (1 - agent.param[6])
11         outp = out .* ap
12         new_status = as - out
13         new_population = sum(new_status .* ap)
14         agent.status = new_status .* ap ./ new_population
15         agent.population = round(Int64, new_population)
16
17         objective = filter(x -> x.id == tidxs[i], [a for a in allagents(model)])[1]
18         os = objective.status
19         op = objective.population
20
21         new_status = (os .* op) + outp
22         new_population = sum(new_status)
23         objective.status = new_status ./ new_population
24         objective.population = round(Int64, new_population)
25     end
26 end

```

Figure 23: Funzione atta a calcolare lo spostamento di agenti da un nodo all'altro del grafo

```

● ● ●

1  function happiness!(agent)
2      agent.happiness = (1 - agent.param[6]) * (agent.status[1] + agent.status[4]) -
3                      Distributions.cdf(Distributions.Beta(2, 5),
4                      agent.status[2] + agent.status[3] + agent.status[5])
5      agent.happiness = agent.happiness < 0.0 ? 0.0 :
6                      agent.happiness > 1.0 ? 1.0 : agent.happiness
7  end

```

Figure 24: Funzione atta a calcolare la felicità degli agenti

3.5 Funzione di Avanzamento Modello

Il modello svolge un ruolo di coordinamento nell’evoluzione degli agenti e si occupa di operazioni che coinvolgono l’intero sistema, anziché focalizzarsi sugli agenti individuali. Queste operazioni includono principalmente tre aspetti chiave:

- **Aggiornamento delle equazioni differenziali ordinarie (ODE):** Il modello si fa carico dell’aggiornamento delle equazioni differenziali ordinarie associate a ciascun agente. Le ODE rappresentano il modello SEIR deterministico che guida l’evoluzione della pandemia all’interno di ciascun agente. L’aggiornamento di queste equazioni è cruciale per monitorare e prevedere il comportamento del virus e dei suoi effetti sulla popolazione.
- **Gestione delle varianti del virus di interesse:** Il modello si occupa anche della gestione delle varianti del virus di interesse. Questo include il monitoraggio delle mutazioni e delle caratteristiche delle varianti, nonché l’aggiornamento dei parametri del modello in base all’emergere di nuove varianti. La gestione efficace delle varianti è essenziale per valutare in modo realistico l’andamento della pandemia e determinare l’efficacia delle misure di intervento.
- **Simulazione della campagna vaccinale:** Il modello si occupa della simulazione della campagna vaccinale. Questo include la simulazione del tempo di ricerca di un vaccino, il calcolo della sua efficacia e della copertura minima per contrastare l’avanzata pandemica, la simulazione della diffusione del vaccino all’interno della rete. Questa simulazione è interessante e soprattutto utile per osservare i differenti comportamenti del modello con varie opportunità farmacologiche.



```

1  function model_step!(model::ABM)
2      for agent in allagents(model)
3          # notify the integrator that the condition may be altered
4          model.integrator[agent.id].u = agent.status
5          model.integrator[agent.id].p = agent.param
6          OrdinaryDiffEq.u_modified!(model.integrator[agent.id], true)
7          OrdinaryDiffEq.step!(model.integrator[agent.id], 1.0, true)
8          agent.status = model.integrator[agent.id].u
9      end
10     voc!(model)
11     model.vaccine ? vaccine!(model) : nothing
12     model.step += 1
13 end

```

Figure 25: Codice per l’aggiornamento del modello ad ogni passo della simulazione

3.5.1 Funzione per la Generazione delle Varianti del Virus (VOC)

La generazione delle Varianti di Interesse (VOC, acronimo di Variant of Concern) nel contesto del modello si basa su alcune assunzioni chiave. Tra queste assunzioni figurano:

- **Tasso di mutazione casuale delle basi del virus:** Si assume che le VOC si sviluppino principalmente a causa di mutazioni casuali nelle basi del virus. Queste mutazioni possono verificarsi durante la replicazione del virus, portando a nuove varianti con caratteristiche genetiche leggermente diverse. Tuttavia, questa rappresentazione semplificata del processo di mutazione è un’approssimazione del comportamento reale delle varianti, che coinvolge una gamma più ampia di fattori e processi genetici.
- **Distribuzione dei parametri pandemici:** La generazione delle VOC considera una distribuzione dei parametri pandemici, il che significa che le varianti possono differire in termini di infettività, gravità dei sintomi, periodo di incubazione, e così via. Questa variabilità nei parametri pandemici riflette la diversità delle varianti del virus nel mondo reale.

È importante notare che questa implementazione è volutamente semplificata per scopi di modellazione e simulazione. La realtà dei processi di mutazione e dell’evoluzione delle varianti è estremamente complessa e coinvolge una serie di meccanismi biologici, evolutivi e epidemiologici. Pertanto, l’uso di queste assunzioni semplificate consente al

modello di fornire una rappresentazione approssimativa del comportamento delle varianti all'interno del contesto della simulazione [45] [46] [47]



```
1  import numpy as np
2  from abm import ABM
3
4  def voc():
5      model = ABM()
6
7      if np.random.rand() < 8e-3:
8          agent = random_agent(model)
9
10         if agent.status[3] != 0.0:
11             agent.param[1] = np.random.uniform(3.3, 5.7)
12             for i in range(2, 5):
13                 agent.param[i] = np.random.normal(agent.param[i], agent.param[i] / 10)
14
15         return agent
16
17     else:
18         return None
```

Figure 26: Funzione che si occupa di generare la VOC

3.5.2 Funzione per la Simulazione della Campagna di Vaccinazione

La simulazione della campagna di vaccinazione si fonda su un approccio in cui viene vaccinata una percentuale fissa di popolazione ad ogni passo del modello. Questa percentuale è scelta in modo tale da tenere conto dell'obiettivo di raggiungere l'immunità di gregge entro un periodo di tempo specifico. Tuttavia, è importante notare che questo approccio è basato su alcune assunzioni semplificative, le quali comprendono:

- **Effetto delle mutazioni del virus sulla distribuzione dei parametri pandemici:** Nel modello, si suppone che le mutazioni del virus possano influenzare la distribuzione dei parametri pandemici, come l'infettività e la gravità dei sintomi. Tuttavia, il modo in cui le mutazioni influenzano questi parametri è approssimato in modo semplificato, mentre nella realtà, le interazioni tra le mutazioni e i parametri pandemici sono estremamente complesse e variabili.
- **Meccanismo di vaccinazione basato su dosi fisse:** Nel contesto della simulazione, il processo di vaccinazione è modellato come somministrazione di dosi fisse di vaccino a una percentuale della popolazione. Questa rappresentazione semplificata non tiene conto di fattori come la distribuzione dei vaccini, la copertura effettiva, la risposta immunitaria individuale e la necessità di dosi di richiamo, che sono elementi importanti nella gestione di una campagna di vaccinazione nel

mondo reale.

Queste semplificazioni nel modello consentono una simulazione più gestibile e comprensibile, ma è fondamentale comprendere che non riflettono completamente il comportamento complesso e variabile del mondo reale. Le simulazioni sono uno strumento utile per l'analisi e la previsione, ma devono essere interpretate con consapevolezza delle limitazioni delle assunzioni adottate.

```
● ● ●
```

```
1  function vaccine!(model::ABM)
2      if rand(model.rng) < 1 / 365
3          R = mean(agent.param[1] for agent in allagents(model))
4          vaccine = (1 - (1 / R)) / rand(model.rng, Normal(0.83, 0.083))
5          vaccine *= mean(agent.param[4] for agent in allagents(model))
6          agent = random_agent(model)
7          agent.param[7] = vaccine
8      end
9  end
10
11 function spread_vaccine!(agent, model::ABM)
12     network = model.migrationMatrix[agent.id, :]
13     tidxs, tweights = findnz(network)
14     for i in eachindex(tidxs)
15         objective = filter(x -> x.id == tidxs[i], [a for a in allagents(model)])[1]
16         objective.param[7] = agent.param[7]
17     end
18 end
```

Figure 27: Funzione che si occupa di simulare la ricerca di un vaccino e la sua successiva applicazione

3.6 Monitoraggio e Intervento

Il sistema di controllo e intervento adottato segue il principio del controllo autonomo mediante l’impiego di una Neural ODE. Questa strategia permette di definire un sistema di Equazioni Differenziali Ordinarie (ODE) che descriva il sistema del mondo reale con le sue relazioni, e di inserire all’interno di esso in modo mirato una rete neurale che controlli il dosaggio delle contromisure [48] [49] [50].

Questa tecnica ha dimostrato di produrre risultati promettenti, ma è fortemente influenzata dalla corretta modellazione del sistema sottostante e dalla selezione della funzione di costo o di controllo utilizzata per addestrare il modello. La sfida principale in questo contesto è garantire una modellazione accurata del sistema di partenza e una comprensione approfondita dei risultati generati dal controllore. Questo perché il controllore restituisce valori nel range $\in [0, 1]$, i quali possono risultare difficilmente interpretabili. Pertanto, è fondamentale definire chiaramente in anticipo il significato di questo intervallo di valori nel contesto del mondo reale.

Nell’elaborato è stato scelto di utilizzare un approccio aggregato per definire le contromisure non farmaceutiche risultanti dall’applicazione del controllore. Queste contromisure indicano in maniera compatta e aggregata la rigidezza di un insieme di contromisure prese ad esempio dal seguente articolo [8]. Nell’articolo vengono presentate 8 differenti contromisure applicate dai vari governi durante l’intera gestione della pandemia da COVID-19 insieme alla loro rigidezza (strictness), e non insieme alla loro efficacia una volta applicate.

Table 2: A global panel database of pandemic policies Oxford COVID-19 Government Response Tracker [8]

| ID | Name | Type | Targeted / General? |
|----|-----------------------------------|---------|---------------------|
| C1 | School closing | Ordinal | Geographic |
| C2 | Workplace closing | Ordinal | Geographic |
| C3 | Cancel public events | Ordinal | Geographic |
| C4 | Restrictions on gathering size | Ordinal | Geographic |
| C5 | Close public transport | Ordinal | Geographic |
| C6 | Stay-at-home requirements | Ordinal | Geographic |
| C7 | Restrictions on internal movement | Ordinal | Geographic |
| C8 | Restrictions on internal travel | Ordinal | No |

La scelta di utilizzare questo approccio deriva dal fatto che in questo modo, si può avere un risultato compatto e immediato dell'efficacia delle contromisure applicate data una media generale della loro rigidezza. Tuttavia non è possibile comprendere l'efficacia singola di ogni tipologia di intervento relativa alla propria rigidezza, essendo questi presi come media generale.

Inizialmente, la funzione di controllo incorporata nel modello agente è definita come illustrato in figura 28. È possibile notare come siano inseriti dei parametri di controllo per gestire l'invocazione effettiva del controllore.

Questi parametri di controllo sono principalmente associati a quando è necessario, dal punto di vista pratico, richiamare la funzione di controllo. Da un lato, servono per ridurre il numero di chiamate al controllore, contribuendo così a migliorare le prestazioni, e dall'altro lato, rendono più realistica l'idea di avere accesso a un sistema di controllo solo in determinati momenti temporali, anziché in modo continuo. Ciò riflette anche il ritardo tra la raccolta dei dati, la formulazione di una contromisura appropriata e la sua attuazione.

In seguito, vengono definite ulteriori regole di attivazione, questa volta legate alla prima attivazione del controllore. Retrospettivamente, è evidente come avere regole di prevenzione già attive possa essere di grande aiuto nella gestione di una pandemia. Tuttavia, a volte può esserci un ritardo nel riconoscere il pericolo, determinato dal valore di toller-

anza, il quale evita l'attivazione del controllore fintanto che la situazione sembra rimanere al di sotto della soglia minima.

Altri parametri definiscono principalmente i fattori necessari affinché la funzione di controllo operi correttamente.



The screenshot shows a Jupyter Notebook cell with three colored icons (red, yellow, green) at the top. The cell contains the following Julia code:

```
1  function control!(agent,
2      model::ABM;
3      tolerance = 1e-3,
4      dt = 14.0,
5      step = 5.0,
6      maxiters = 30,
7      patience = 3,
8      doplot::Bool = false,
9      verbose::Bool = false)
10     if agent.status[3] ≥ tolerance && model.step % dt == 0
11         agent.param[6] = controller(agent.status,
12             vcat(agent.param[1:5], agent.param[7]));
13         h = agent.happiness,
14         timeframe = (0.0, dt),
15         step = step,
16         maxiters = maxiters,
17         patience = patience,
18         doplot = doplot,
19         verbose = verbose,
20         id = agent.id,
21         rng = model.rng)
22     end
23 end
24
```

Figure 28: Definizione della funzione di attivazione del controllore all'interno della funzione `agent_step!` (Figura 22) definita nel modello di simulazione

3.6.1 Implementazione Controllore

Il controllore effettivo è principalmente strutturato come già accennato nella sezione relativa alle Neural ODE.

Una rete neurale è definita utilizzando il framework **Lux.jl** [31] e viene utilizzata esclusivamente per fare una stima del valore di controllo η durante la fase di addestramento, il quale sarà alla fine utilizzato come valore di controllo per il modello. Successivamente, viene definito il sistema di ODE che governa il fenomeno che si desidera analizzare, nello specifico, il sistema è un modello **SEIR** che tiene conto della perdita di immunità nel tempo.



```
● ● ●
1 CUDA.allowscalar(false)
2
3 ann = Lux.Chain(Lux.Dense(5, 64, swish), Lux.Dense(64, 1))
4 p, state = Lux.setup(rng, ann)
5 p = p |> ComponentArray |> Lux.gpu_device()
6 state = state |> Lux.gpu_device()
7
8 function dudt_(du, u, p, t)
9     S, E, I, R, D = u
10    Ro, γ, σ, ω, δ, ξ = p_true
11    η = abs(ann(u, p, state)[1][1])
12    μ = δ / 1111
13    du[1] = μ * sum(u[1:5]) - Ro * γ * (1 - η) * S * I + ω * R - ξ * S - μ * S # dS
14    du[2] = Ro * γ * (1 - η) * S * I - σ * E - μ * E # dE
15    du[3] = σ * E - γ * I - δ * I - μ * I # dI
16    du[4] = (1 - δ) * γ * I - ω * R + ξ * S - μ * R # dR
17    du[5] = δ * γ * I # dD
18 end
19
```

Figure 29: Definizione del controllore mediante Neural ODE

Successivamente, il problema viene istanziato utilizzando l’interfaccia **ODEProblem**, a partire dalla quale il modello può iniziare ad essere addestrato. Prima di procedere con l’addestramento effettivo, vengono definite funzioni per la predizione, il calcolo della perdita del modello (loss) e una funzione di callback ausiliaria utile per monitorare l’andamento dell’apprendimento del modello.

```

1  function predict(p)
2      _prob = remake(prob, u0 = initial_condition, tspan = timeframe, p = p)
3      Array(solve(_prob,
4          Tsit5(),
5          saveat = ts,
6          abstol = 1e-10,
7          reltol = 1e-10,
8          verbose = false)) # suppress unwanted warning. Always active
9  end
10
11 function loss(p)
12     pred = predict(p)
13     (sum(abs2, pred[3, :]) + sum(abs2, pred[5, :]) + sum(abs2, pred[2, :])) / h
14 end
15
16 patience_temp = 0
17 losses = Float64[]
18 callback = function (p, l; lstep = loss_step)
19     push!(losses, l)
20     # Exit early if not improving...
21     if length(losses) > 1 && (abs(l - losses[end - 1]) < 1e-4 || isnan(l))
22         patience_temp += 1
23         if patience_temp > patience
24             return true
25         end
26     else
27         patience_temp = 0
28     end
29     return false
30 end
31

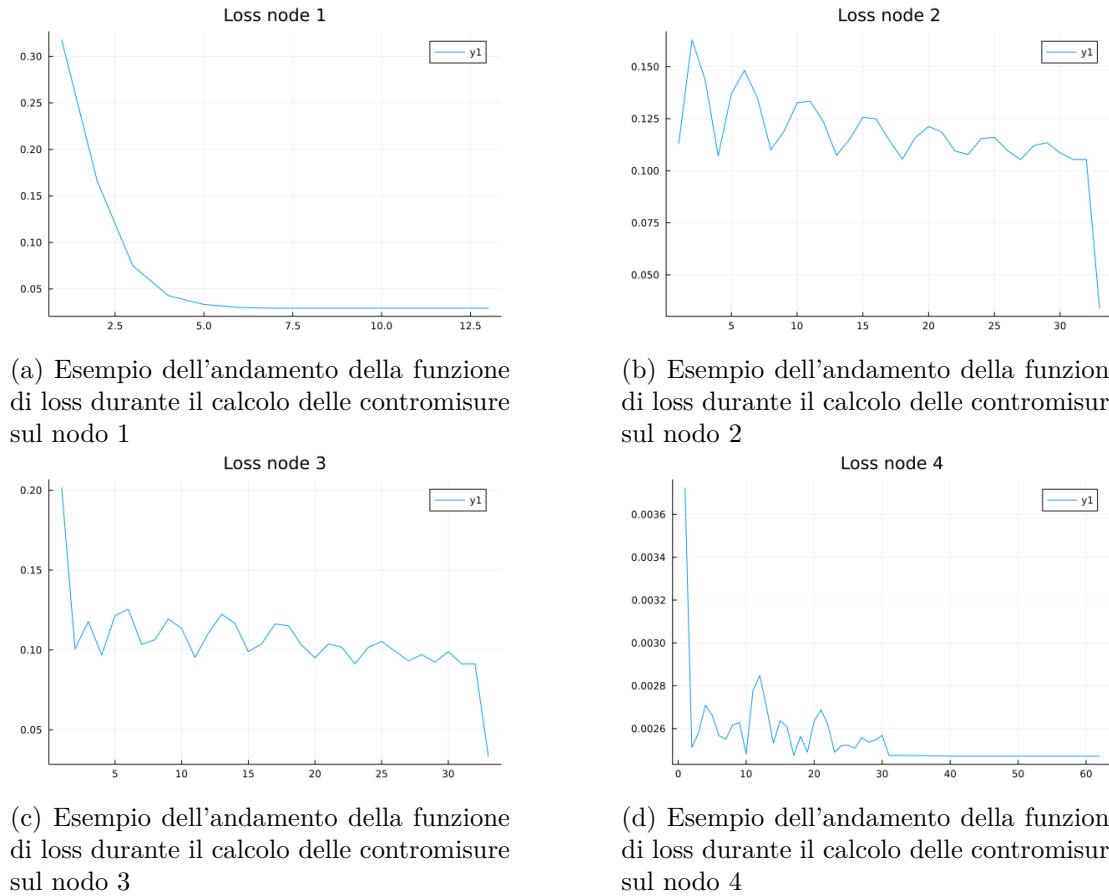
```

Figure 30: Definizione delle funzioni di supporto del controllore

Come mostrato nella figura 30, la funzione di callback implementa una forma di “early stopping” per evitare l’addestramento eccessivo una volta che il modello ha raggiunto un risultato ottimale. Questo approccio si basa sull’osservazione che il valore della loss tende a stabilizzarsi una volta che il modello ha raggiunto una buona soluzione.

3.6.2 Esplorazione Funzione di Loss

La definizione della funzione di perdita, come illustrato nella Figura 32, rivela un approccio che enfatizza la massimizzazione del numero di individui in uno stato di salute ottimale, mentre cerca di minimizzare il numero di individui affetti da patologie. In aggiunta a questa considerazione, la funzione di perdita prende in considerazione il valore complessivo di felicità dell'agente, cercando di massimizzarlo.



(c) Esempio dell'andamento della funzione di loss durante il calcolo delle contromisure sul nodo 3

(b) Esempio dell'andamento della funzione di loss durante il calcolo delle contromisure sul nodo 2

(d) Esempio dell'andamento della funzione di loss durante il calcolo delle contromisure sul nodo 4

L'obiettivo ottimale da perseguire è rappresentato da un valore prossimo allo zero. È importante notare che un valore leggermente superiore a zero indica comunque che il modello ha identificato una soluzione accettabile, seppur non ottimale. Tale approccio mette in evidenza l'importanza di equilibrare la salute degli individui con il loro benessere complessivo, creando così una soluzione che tiene conto di entrambi questi aspetti fondamentali.



```

1 function loss(p)
2     pred = predict(p)
3     (sum(abs2, pred[3, :]) + sum(abs2, pred[5, :]) + sum(abs2, pred[2, :])) / h
4 end
5

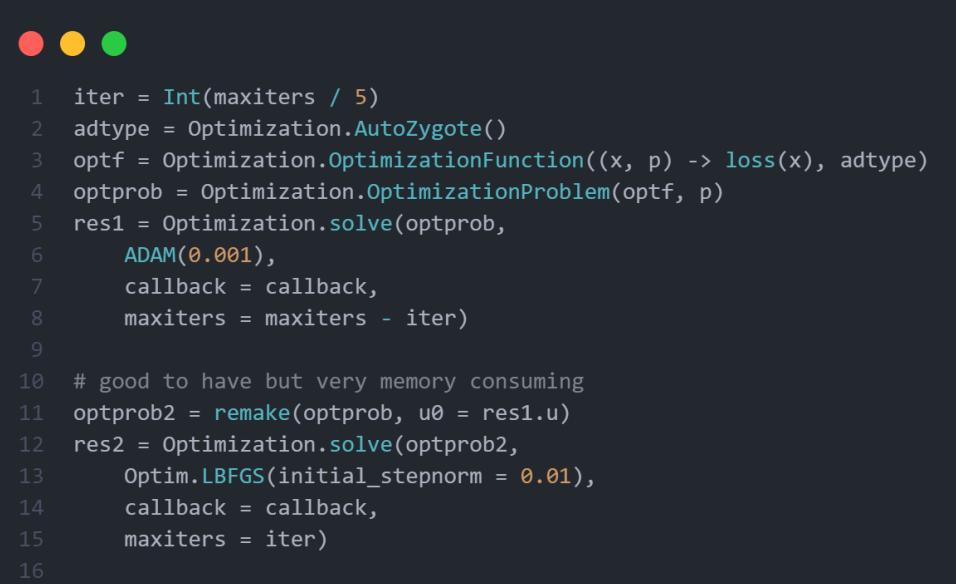
```

Figure 32: Definizione della funzione di loss per la Rete Neurale

3.6.3 Esplorazione Funzione di Addestramento

Il ciclo di addestramento può essere idealmente suddiviso in due parti, ma in pratica, entrambe potrebbero non essere necessarie. La prima parte, come mostrato nella figura 33, impiega un ciclo di addestramento con l'uso dell'ottimizzatore **ADAM** [51]. Se i risultati sono soddisfacenti, è possibile concludere l'addestramento. Tuttavia, è comune utilizzare due cicli di addestramento differenti per massimizzare le prestazioni. In particolare, si associano due cicli di addestramento con due ottimizzatori diversi al fine di sfruttare al massimo le peculiarità di ognuno. Questo approccio è utile per massimizzare il processo di ottimizzazione dei parametri. Solitamente, si utilizza un ottimizzatore tipo **ADAM** per la maggior parte dell'addestramento, in quanto è in grado di trovare rapidamente un buon spazio di iperparametri. Successivamente, si passa a un secondo ottimizzatore, come ad esempio **BFGS** [52] [53] [54], che eccelle nel trovare rapidamente un minimo locale all'interno dello spazio dei parametri.

Si è deciso di utilizzare un solo ottimizzatore, in quanto **ADAM** in questo caso, pur con un numero limitato di iterazioni, consente di raggiungere un risultato ottimale in tempi brevi, senza la necessità di utilizzare un secondo ottimizzatore, il quale in questo caso non apporterebbe miglioramenti significativi comportando solo un aumento delle risorse computazionali.



```

1 iter = Int(maxiters / 5)
2 adtype = Optimization.AutoZygote()
3 optf = Optimization.OptimizationFunction((x, p) -> loss(x), adtype)
4 optprob = Optimization.OptimizationProblem(optf, p)
5 res1 = Optimization.solve(optprob,
6     ADAM(0.001),
7     callback = callback,
8     maxiters = maxiters - iter)
9
10 # good to have but very memory consuming
11 optprob2 = remake(optprob, u0 = res1.u)
12 res2 = Optimization.solve(optprob2,
13     Optim.LBFGS(initial_stepnorm = 0.01),
14     callback = callback,
15     maxiters = iter)
16

```

Figure 33: Definizione delle funzioni di addestramento del controllore

Funzione di Attivazione della Rete Neurale

La scelta della funzione di attivazione all'interno delle reti neurali ha un notevole impatto sulle dinamiche di apprendimento. Attualmente, la funzione più comunemente utilizzata e di successo è la funzione **ReLU** (Rectified Linear Unit), definita come $f(x) = \max(0, x)$. Tuttavia, sono state proposte diverse alternative, ma nessuna di esse è riuscita a superare la popolarità della ReLU, principalmente a causa di risultati prestazionali variabili.

La funzione di attivazione che ho scelto di utilizzare è chiamata **Swish** ed è stata proposta dal Google Brain Team [55]. La funzione Swish è definita come $f(x) = x \cdot \text{sigmoid}(x)$. In esperimenti condotti, è stato osservato che sostituire la funzione di attivazione ReLU con Swish ha portato a miglioramenti nelle prestazioni sui task di classificazione “top-1” su dataset come **ImageNet**, con guadagni fino al 0.9% utilizzando **NASNetA** e fino al 0.6% utilizzando **Inception-ResNet-v2**.

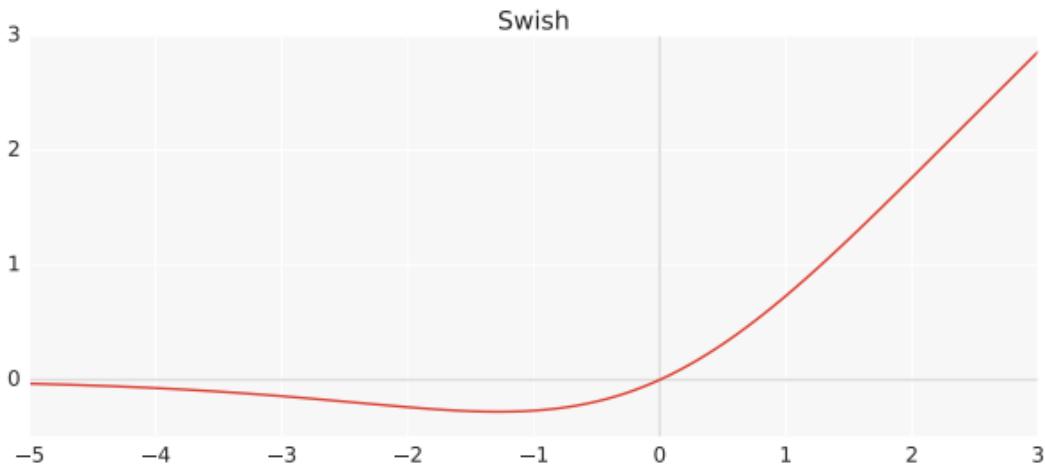


Figure 34: Funzione di attivazione Swish
<https://lazyprogrammer.me/wp-content/uploads/2017/10/Screen-Shot-2017-10-18-at-2.39.55-PM.png>

La peculiarità distintiva di Swish risiede nella sua semplicità e nella notevole somiglianza con la funzione di attivazione ReLU, il che la rende estremamente agevole da implementare come sua alternativa. Un'importante problematica associata all'utilizzo di ReLU è rappresentata dal fatto che il valore della sua derivata è pari a zero per la metà dei valori di input x .

È stato dimostrato, mediante l'impiego del dataset **MNIST**, che le prestazioni di Swish e ReLU sono sostanzialmente comparabili quando si utilizzano reti neurali con un numero di strati (layer) che si aggira intorno a 40. Tuttavia, emerge chiaramente che Swish supera in modo significativo ReLU in termini di prestazioni, specialmente quando il numero di strati si avvicina ai 40-50, situazione in cui l'ottimizzazione tende a diventare più complessa. Ciò suggerisce, ed è stato successivamente confermato, che in reti neurali molto profonde, Swish mostra prestazioni superiori nei test di accuratezza rispetto alla funzione ReLU. Tuttavia, è importante notare che entrambe le funzioni sperimentano un declino delle prestazioni all'aumentare delle dimensioni dei batch, ma in ogni caso, Swish ottiene risultati migliori.

Alla luce di questi dati e considerando la struttura relativamente poco profonda del modello, ho preso la decisione di utilizzare Swish come funzione di attivazione all'interno della rete neurale. È importante notare che questa scelta è attesa per contribuire positivamente alle prestazioni complessive, anche se ci si aspetta che il guadagno di accuratezza non sia eccessivamente sensibile, dato il contesto dell'applicazione illustrata nella figura 29.

4 Analisi di Sensibilità

L'analisi di sensitività è stata condotta sul modello SEIR, come illustrato in Figura 14. Questa analisi mirava a valutare come il modello reagisse a variazioni specifiche dei suoi parametri, le quali potrebbero condurre a comportamenti inattesi o peculiari.

Per effettuare l'analisi di sensitività, sono stati impiegati i metodi forniti dalla suite SciML.ai. Ciò ha permesso di identificare i parametri ai quali il modello è più sensibile.

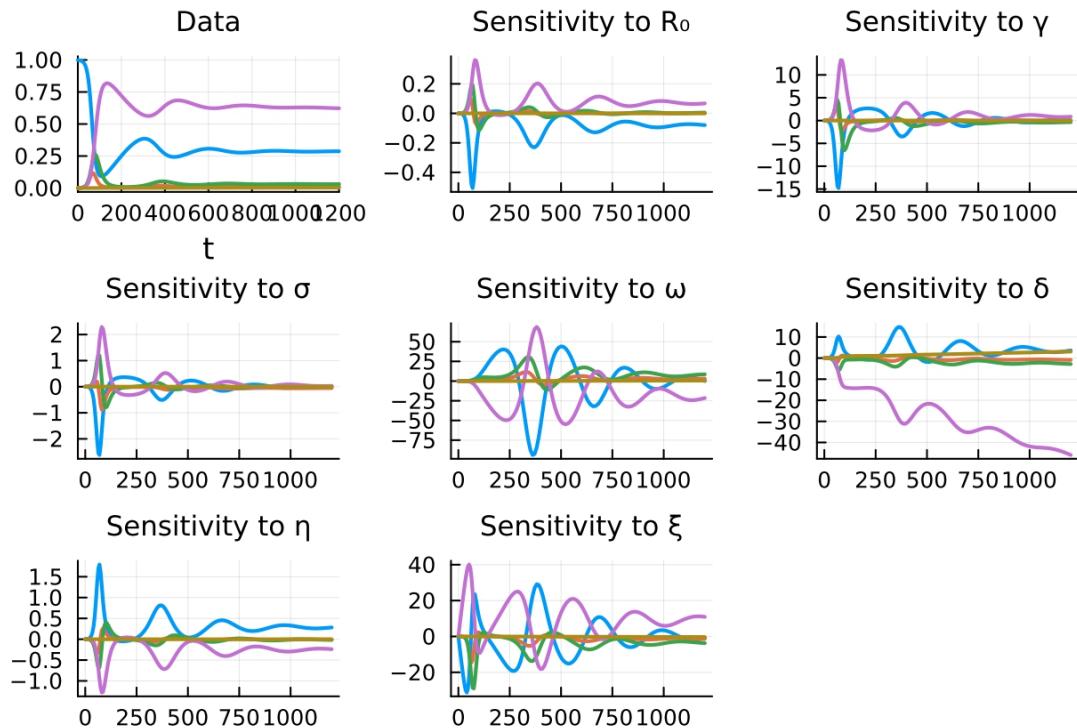


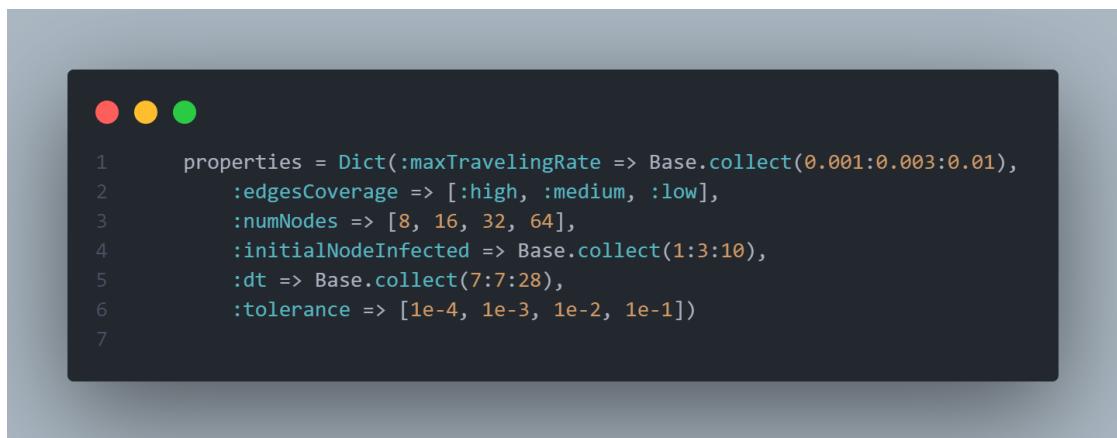
Figure 35: Grafico rappresentante l'analisi di sensitività del modello

Come evidenziato nella Figura 35, il modello dimostra una sensitività uniforme rispetto ai parametri R_0 , γ , e σ , il che è coerente, dato che essi influiscono sull'andamento dell'epidemia. La sensitività ai parametri ω e δ era attesa, ma è risultata più marcata di quanto previsto. La sensitività al parametro η , che influisce sulle contromisure, è inversamente correlata a quella del parametro R_0 , come previsto, poiché η è direttamente legato alle misure di contenimento. La sensitività al parametro ξ mostra un comportamento singolare, probabilmente dovuto alla sua correlazione con ω , entrambi legati al compartimento “R” del modello, che gestisce gli individui guariti e vaccinati.

In una fase successiva dell'analisi, è stata condotta un'analisi di sensitività sui parametri specifici del modello ad agenti, utilizzando la funzione **paramscan** fornita dalla libreria

Agents.jl. Sono stati individuati e selezionati quei parametri che potrebbero avere un notevole impatto sul comportamento generale del modello. Questa selezione è stata fatta in base al potenziale di tali parametri di innescare cambiamenti significativi nei risultati della simulazione.

È importante notare che alcuni parametri, specificamente quelli legati al controllore e alla campagna di vaccinazione, sono stati esclusi da questa analisi. Ciò è dovuto al fatto che variazioni in questi parametri possono comportare cambiamenti sostanziali nella simulazione, e pertanto sono stati considerati separatamente e successivamente analizzati in modo dettagliato.



```
properties = Dict(:maxTravelingRate => Base.collect(0.001:0.003:0.01),
                  :edgesCoverage => [:high, :medium, :low],
                  :numNodes => [8, 16, 32, 64],
                  :initialNodeInfected => Base.collect(1:3:10),
                  :dt => Base.collect(7:7:28),
                  :tolerance => [1e-4, 1e-3, 1e-2, 1e-1])
```

Figure 36: Parametri usati per l'analisi di sensitività del modello ad agente

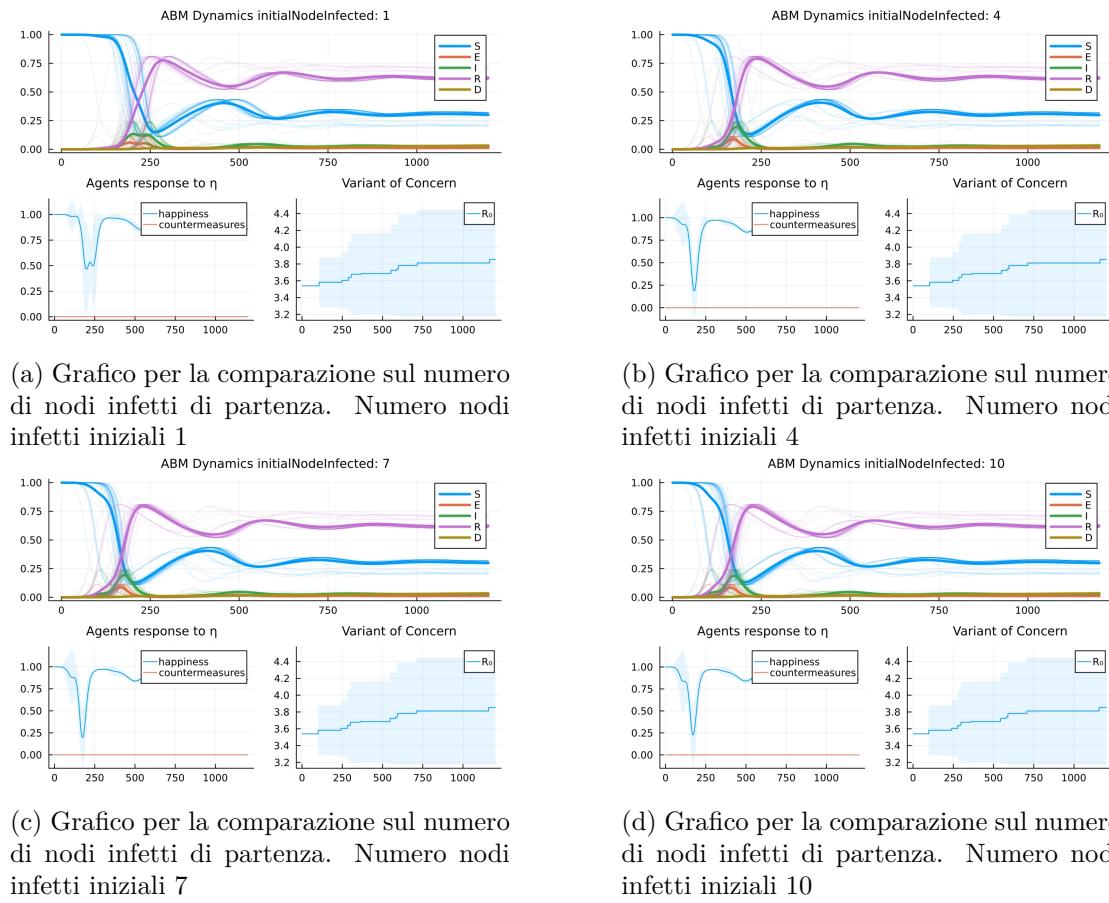
L'analisi di sensitività sui parametri rimanenti mira a esplorare come variazioni specifiche in ciascuno di essi possano influenzare il comportamento complessivo del modello. Questo tipo di analisi è fondamentale per comprendere la sensibilità del sistema a variazioni nei parametri e per valutare le risposte del modello a differenti condizioni.

In definitiva, questa analisi di sensitività fornisce un quadro più completo della dinamica del modello, consentendo di identificare i parametri chiave che influenzano le sue prestazioni e fornendo indicazioni importanti per la calibrazione e l'ottimizzazione del modello stesso.

4.1 Analisi del Comportamento in Base al Numero Iniziale di Nodi Infetti

L'analisi condotta ha investigato l'impatto del numero iniziale di nodi infetti sul comportamento complessivo del modello. I risultati hanno dimostrato che un numero maggiore di nodi infetti iniziali conduce a un raggiungimento più rapido del picco di infetti nel corso della simulazione. Tuttavia, non sono stati osservati altri cambiamenti significativi nel comportamento del modello.

Questo risultato era atteso e coerenza con le dinamiche epidemiologiche conosciute. Infatti, è noto che un aumento del numero iniziale di individui infetti in una popolazione può accelerare l'espansione della malattia e portare a un picco di infetti più precoce. Tuttavia, le altre dinamiche del modello, come il tasso di diffusione del virus, la progressione della pandemia e le misure di intervento, non sono state influenzate in modo significativo da questa variazione.



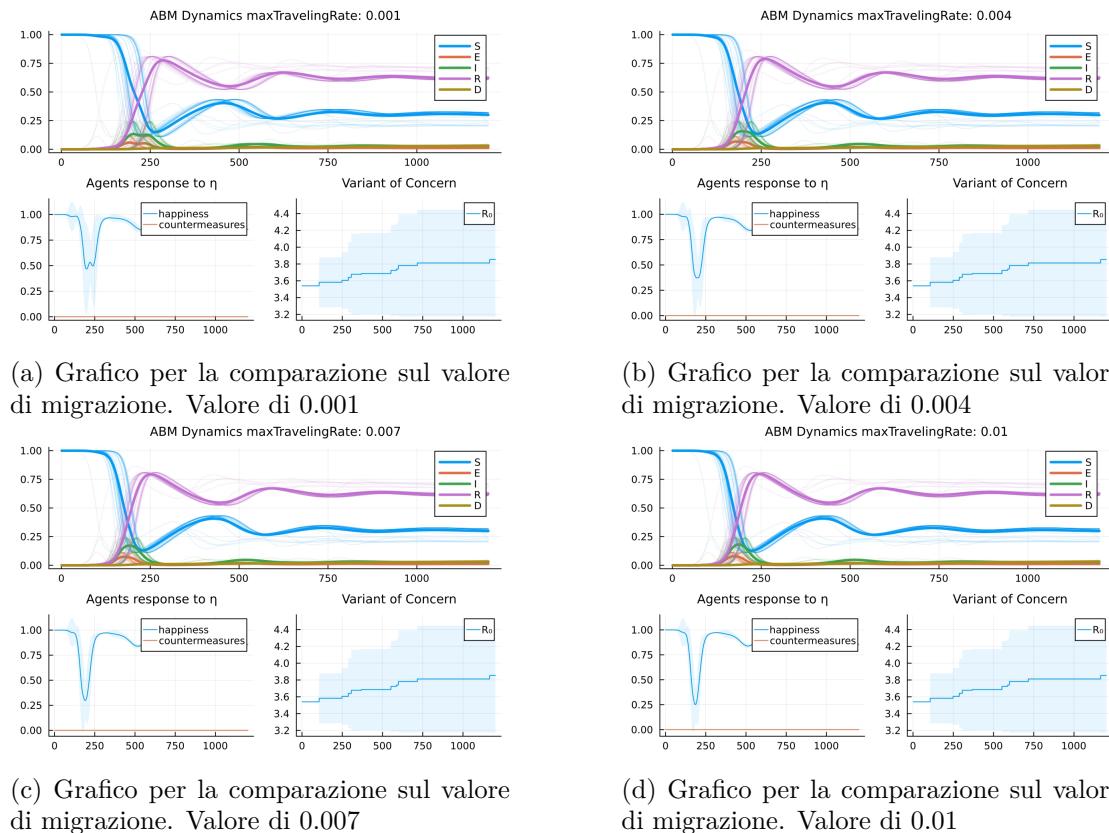
In definitiva, questo risultato conferma la validità delle dinamiche epidemiologiche incorporate nel modello e suggerisce che il modello sia in grado di riflettere realisticamente

l'effetto del numero iniziale di infetti sulla diffusione della pandemia, come si verifica anche nella realtà.

4.2 Analisi del Comportamento in Base al Tasso di Migrazione

L'analisi condotta ha esplorato l'influenza del tasso di migrazione sul comportamento del modello. I risultati hanno evidenziato che un tasso di migrazione più elevato ha generato un picco di individui infetti più alto, e ciò è avvenuto in un periodo di tempo più breve rispetto ai casi in cui il tasso di migrazione era più basso.

Questo risultato è coerente con le conoscenze consolidate riguardo all'effetto della mobilità sulla diffusione delle malattie infettive. Infatti, un aumento del tasso di migrazione implica un maggiore movimento delle persone tra i nodi del grafo (rappresentanti aree geografiche o comunità), il che può facilitare la trasmissione del virus da un luogo all'altro. Di conseguenza, si verifica un aumento più rapido e pronunciato nel numero di individui infetti, portando a un picco più alto in un tempo più breve.



Questi risultati sottolineano l'importanza della mobilità e dei flussi di persone nel contesto della diffusione di malattie infettive, come le pandemie. Essi sottolineano anche la capacità del modello di catturare in modo realistico queste dinamiche epidemiologiche,

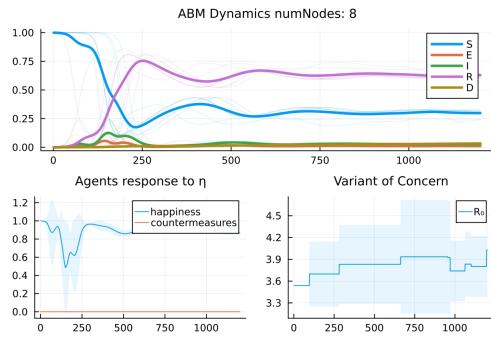
consentendo una valutazione approfondita degli effetti del tasso di migrazione sul comportamento del sistema.

4.3 Analisi del Comportamento in Base al Numero di Nodi della Rete

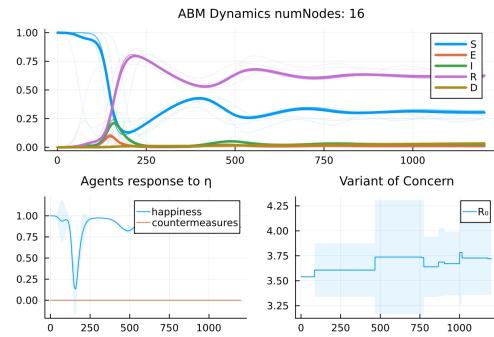
L'analisi condotta ha investigato come il numero di nodi nella rete influenzi il comportamento complessivo del modello. Dai risultati è emerso che un numero maggiore di nodi nella rete ha comportato una diffusione più lenta della pandemia e ha generato picchi meno ripidi ma più prolungati nel tempo.

Questo risultato è influenzato da diversi fattori, tra cui la topologia delle connessioni tra i nodi e la posizione del focolaio iniziale della pandemia. Quando ci sono più nodi nella rete, la trasmissione del virus può richiedere più tempo per propagarsi attraverso l'intera popolazione, dato che ci sono più "tappe" attraverso cui deve passare. Inoltre, la posizione iniziale del focolaio può influenzare la velocità di diffusione, in quanto i nodi vicini al focolaio avranno una probabilità maggiore di essere infettati inizialmente rispetto ai nodi più lontani.

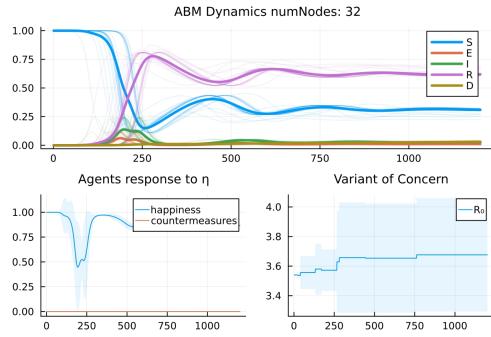
Di conseguenza, si osserva una diffusione più graduale della pandemia nel corso del tempo, con picchi meno ripidi ma più prolungati. Questo risultato è coerente con l'effetto della dimensione della popolazione e della struttura della rete sulla diffusione di malattie infettive.



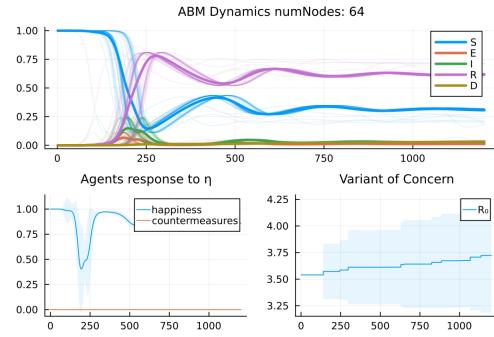
(a) Grafico per la comparazione sul numero di nodi della rete. Numero nodi 8



(b) Grafico per la comparazione sul numero di nodi della rete. Numero nodi 16



(c) Grafico per la comparazione sul numero di nodi della rete. Numero nodi 32



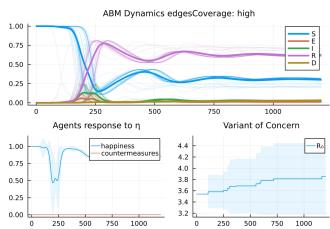
(d) Grafico per la comparazione sul numero di nodi della rete. Numero nodi 64

In sintesi, l'analisi ha dimostrato come il numero di nodi nella rete, insieme a variabili come la topologia delle connessioni e la posizione iniziale del focolaio, possa influenzare in modo significativo le dinamiche della pandemia e i profili temporali di infezione.

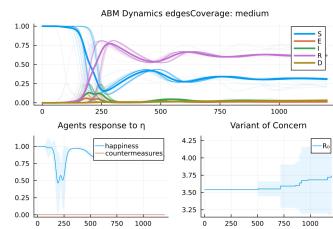
4.4 Analisi del Comportamento in Base alla Copertura della Rete

L'analisi condotta ha esaminato l'effetto della copertura della rete, rappresentata dal numero di archi, sul comportamento del modello. Dai risultati è emerso che una copertura bassa, ovvero un numero limitato di connessioni tra i nodi del grafo, ha comportato un rallentamento significativo nella diffusione del virus, analogamente all'effetto di misure di contenimento come il lockdown. Questo risultato è coerente con l'importanza della mobilità nella diffusione delle malattie infettive e sottolinea il ruolo cruciale delle misure di contenimento nel limitare la propagazione del virus.

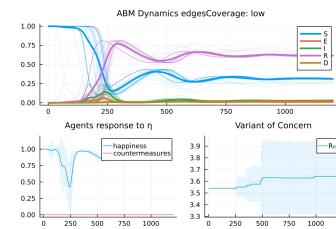
Inoltre, è stato osservato che il numero di Varianti di Interesse (VOC) tende ad aumentare con il numero di nodi nella rete. Questo suggerisce che in una popolazione più ampia, c'è una maggiore probabilità di emergere nuove mutazioni. Tuttavia, è interessante notare che la presenza di una bassa copertura, cioè una limitata interconnessione tra i nodi, può limitare la diffusione delle varianti stesse, poiché la trasmissione tra gli individui è ridotta.



(a) Grafico per la comparazione sulla copertura della rete. Copertura alta



(b) Grafico per la comparazione sulla copertura della rete. Copertura media



(c) Grafico per la comparazione sulla copertura della rete. Copertura bassa

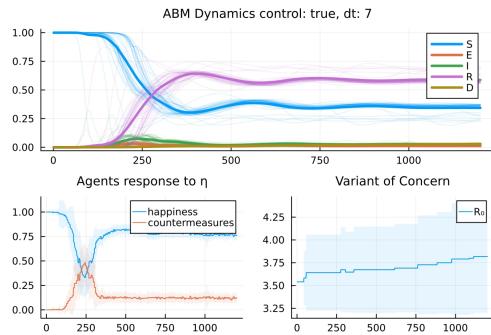
In sintesi, l'analisi ha rivelato come vari parametri e condizioni, tra cui la copertura della rete, la mobilità, le misure di contenimento e il numero di nodi, possono influenzare in modo significativo il comportamento del modello epidemiologico. Questi risultati forniscono importanti informazioni per la comprensione e la gestione delle epidemie, evidenziando l'importanza delle strategie di mitigazione e della considerazione della struttura della rete nella prevenzione della diffusione del virus.

4.5 Studio dell’Andamento del Controllore in Relazione all’Intervallo di Intervento

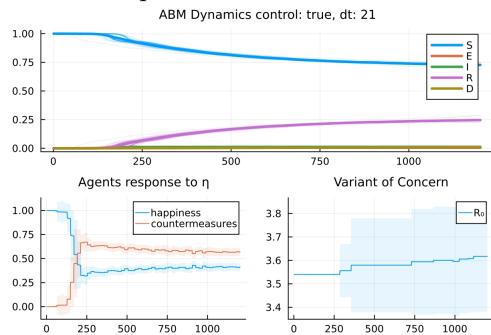
L’analisi condotta nel presente studio ha investigato l’influenza dell’intervallo di intervento del controllore in relazione al controllo della pandemia e all’andamento della curva di “happiness”. I risultati ottenuti hanno rivelato che un incremento nel valore associato all’intervallo di intervento provoca un notevole sbilanciamento nella curva di felicità (“happiness”) e, in generale, comporta un repentino aumento nei valori associati alle contromisure adottate.

Tale andamento, coerente e previsto, può essere spiegato dalla complessità intrinseca nel formulare contromisure a lungo termine rispetto a quelle a breve termine, oltre al fatto che le prime comportano costi maggiori. È interessante notare che l’andamento delle curve, come mostrato nelle Figure 41c e 41d, ha rivelato comportamenti inaspettati che non erano stati previsti inizialmente. Questo comportamento inusuale è probabilmente riconducibile alla definizione della funzione di loss, come illustrato nella Figura 32, e alle successive iterazioni di addestramento della rete.

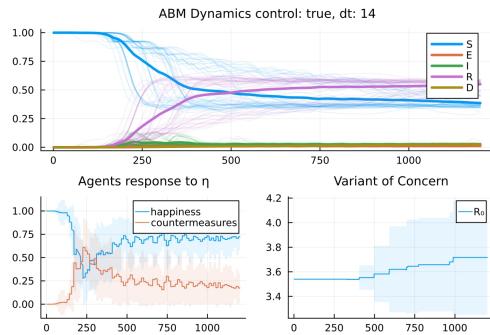
In conclusione, dai risultati ottenuti emerge che attualmente le curve più affidabili sono quelle associate a un intervallo di intervento più breve. Tuttavia, è importante notare che non sono stati esplorati intervalli troppo granulari a causa del loro significativo impatto sulle prestazioni del sistema e, soprattutto, della loro limitata rappresentatività nella realtà.



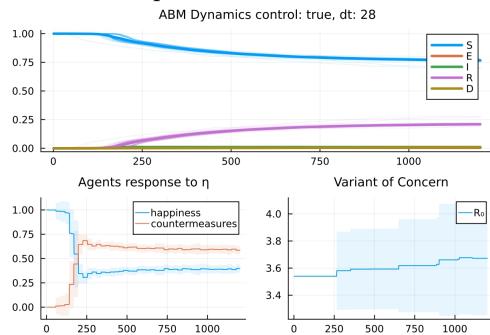
(a) Grafico per la comparazione sul valore dell'intervallo di intervento del controllore. Valore 7 step



(c) Grafico per la comparazione sul valore dell'intervallo di intervento del controllore. Valore 21 step



(b) Grafico per la comparazione sul valore dell'intervallo di intervento del controllore. Valore 14 step



(d) Grafico per la comparazione sul valore dell'intervallo di intervento del controllore. Valore 28 step

4.6 Studio dell'Andamento del Controllore in Relazione all'Intervallo di Tolleranza

L'analisi condotta nel corso di questa indagine ha approfondito l'influenza dell'intervallo di tolleranza del controllore in relazione al controllo della pandemia e all'andamento della curva di "happiness". Dalla nostra analisi emerge chiaramente che il valore di tolleranza del controllore svolge un ruolo cruciale soprattutto durante il picco pandemico.

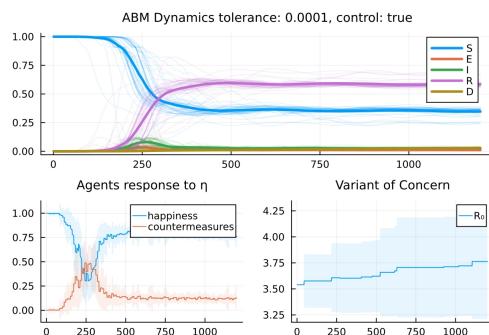
Tuttavia, è degno di nota che un valore di tolleranza estremamente elevato incide in maniera significativa sull'andamento complessivo del modello, ritardando l'attuazione delle contromisure e costringendo il controllore ad adottare un comportamento molto più drastico rispetto alle aspettative.

È fondamentale sottolineare che tali risultati sono ampiamente condizionati dal numero di individui presenti in ciascun nodo del modello. Il test è stato condotto su un modello composto da cinquanta nodi, ognuno dei quali ospitava un numero di individui distribuiti secondo una distribuzione esponenziale con un parametro medio pari a 10.000. Questo implica che l'interpretazione dei risultati deve essere attentamente valutata tenendo

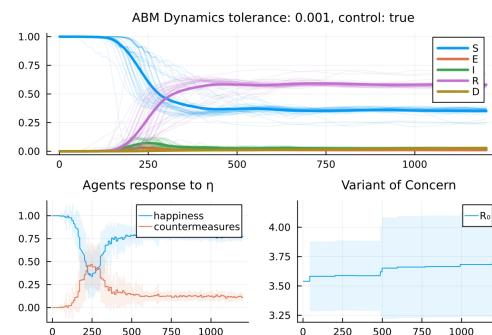
conto del valore medio di popolazione all'interno del modello.

Evidentemente, è cruciale riconoscere come l'aumento del valore di popolazione generi risultati differenti. In scenari caratterizzati da valori di popolazione medio relativamente bassi, è possibile adottare una soglia di tolleranza più elevata, mentre in situazioni in cui la popolazione è notevolmente elevata, diventa necessario aumentare la tolleranza al fine di garantire un adeguato controllo.

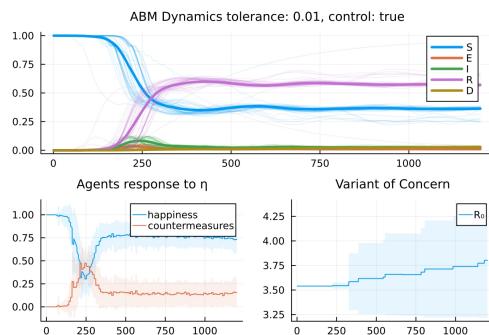
In conclusione, l'analisi effettuata dimostra che l'intervallo di tolleranza del controllore rappresenta un fattore chiave nella gestione della pandemia e dell'andamento della "happiness". La sua ottimizzazione deve essere attentamente bilanciata in base al contesto specifico, inclusi il numero di individui e le dinamiche di popolazione, al fine di garantire un controllo efficace e tempestivo della pandemia.



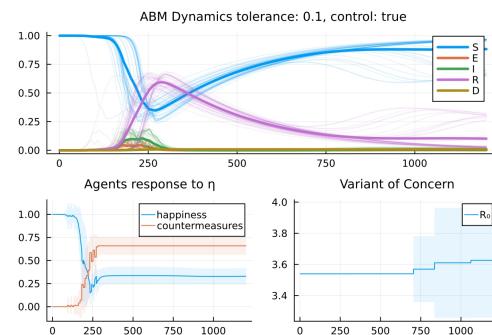
(a) Grafico per la comparazione sul valore di attivazione del controllore. Valore di tolleranza 0.0001



(b) Grafico per la comparazione sul valore di attivazione del controllore. Valore di tolleranza 0.001



(c) Grafico per la comparazione sul valore di attivazione del controllore. Valore di tolleranza 0.01



(d) Grafico per la comparazione sul valore di attivazione del controllore. Valore di tolleranza 0.1

In entrambi i contesti in cui sono stati esaminati e testati alcuni parametri fondamentali del controllore, è evidente come la curva di "happiness" mostri un comportamento apparentemente opposto rispetto alla curva delle contromisure adottate. Questo fenomeno è chiaramente attribuibile alla natura estremamente semplicistica con cui è stata definita

la curva di “happiness”, come illustrato nella Figura 24.

Va notato che questa semplificazione nella rappresentazione della felicità è stata scelta consapevolmente per scopi specifici di questo studio, sebbene sia riconosciuta come irrealistica. Tuttavia, questa semplice rappresentazione ha dimostrato di essere adeguata per gli obiettivi attuali della ricerca.

Nonostante l'apparente discrepanza tra i comportamenti delle due curve, è importante sottolineare che questa semplificazione sarà oggetto di ulteriori raffinamenti in futuro. Tale rifinitura mirerà a renderla più realistica e coerente con le dinamiche reali delle emozioni umane e delle reazioni della società a situazioni di emergenza.

In conclusione, la discrepanza osservata tra le curve di “happiness” e le curve delle contromisure è attribuibile alla semplificazione deliberata della curva di “happiness” utilizzata in questo studio. Tale semplificazione, sebbene irrealistica, è adeguata agli scopi attuali, ma verrà affinata in future ricerche al fine di migliorare la rappresentazione delle dinamiche umane in contesti pandemici e di emergenza.

5 Risultati Ottenuti

I risultati ottenuti, che verranno analizzati in dettaglio nelle sezioni successive, possono essere riassunti in termini generali come risultati positivi, sia in termini di prestazioni del modello che di efficacia delle politiche di intervento adottate. In particolare, è evidente che il tempo totale di simulazione è principalmente influenzato dall'implementazione del controllore e, di conseguenza, dal calcolo delle Neural ODE e da tutte le operazioni connesse.

I lunghi tempi di calcolo riscontrati sono principalmente attribuibili al processo di calcolo delle contromisure non farmaceutiche applicate alla popolazione. Diversamente, le contromisure farmaceutiche, se applicate con tempismo adeguato, oltre a contribuire in modo significativo alla riduzione della diffusione della pandemia (vedi Figura 47a, Figura 47b, Figura 47c, Figura 47d), consentono anche una significativa riduzione dei tempi di calcolo.

Table 3: Benchmarking con BenchmarkTools.jl [9]

| Singlerun | | | | |
|-------------------|-------------------|-----------------------------|-------------------------|-------------------------------|
| | Nessun Intervento | Intervento Non Farmaceutico | Intervento Farmaceutico | Intervento Farmaceutico e Non |
| Evaluation Time | 10.43 s | 994.81 s | 11.46 s | 333.8 s |
| GC Time | 5.07% GC | 11.04% GC | 5.65% GC | 10.39% GC |
| Memory Estimation | 4.4 GiB | 995.35 GiB | 6.05 GiB | 320.6 GiB |
| # Allocation | 67248133 | 11170934613 | 74275343 | 3592984901 |

Si può notare come la riduzione dei tempi di calcolo ottenuta con l'applicazione delle misure farmaceutiche e non, possa essere un indicatore per misurare la "bontà" dei risultati ottenuti dall'applicazione del controllore.

Questo aspetto sottolinea l'importanza delle strategie di vaccinazione e delle misure farmaceutiche nella gestione delle pandemie, non solo dal punto di vista della salute pubblica, ma anche dal punto di vista della computazione e dell'efficienza dei modelli epidemiologici. Ciò nonostante, i risultati per essere considerabili affidabili è stato osservato che devono essere usati congiuntamente; l'unione delle contromisure farmaceutiche e non, permette di ottenere un miglioramento delle prestazioni del modello unitamente ad un miglioramento dei risultati complessivi della simulazione.

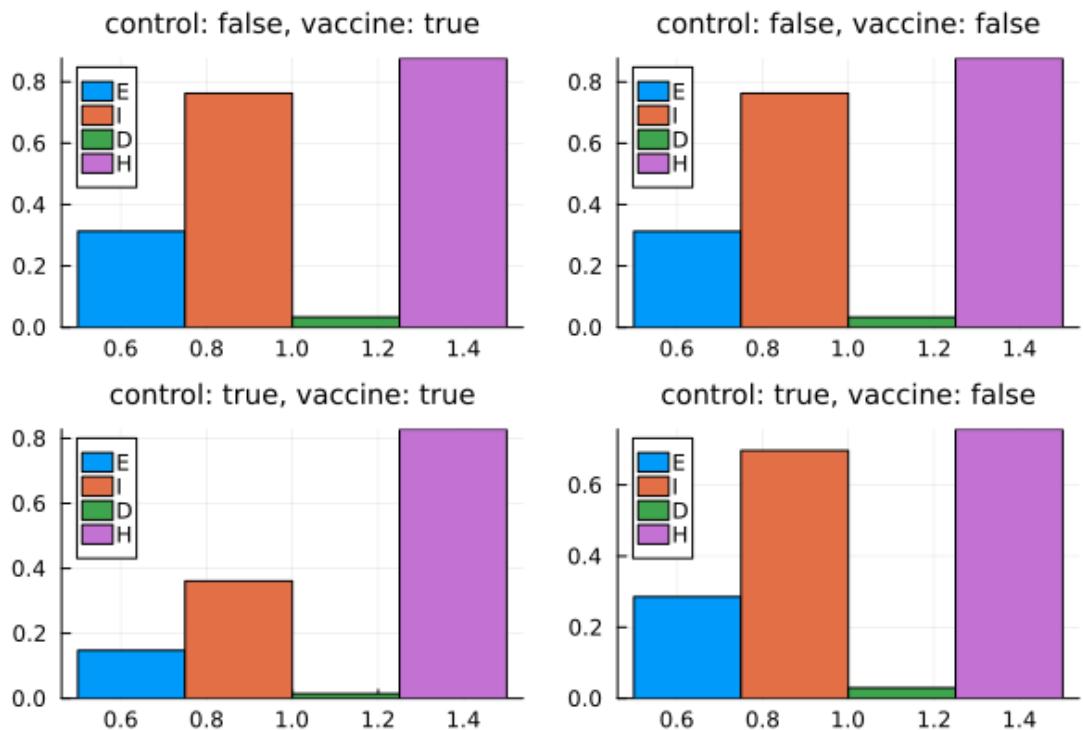


Figure 43: Grafici cumulativi descriventi il numero cumulativo medio di individui Exposed, Infected e Dead insieme al valore di Happiness ottenuto da una simulazione multipla (EnsembleRun) considerando differenti contromisure

Parametri Simulazione

Table 4: Parametri utilizzati per la simulazione del modello

| Parametri | Valori |
|--|--|
| Numero Nodi | 50 |
| Copertura Archi | Alta |
| Numero Nodi Infetti Iniziali | 1 |
| Parametri Pandemici | $R_0 = 3.54$, $\gamma = \frac{1}{14}$, $\sigma = \frac{1}{5}$, $\omega = \frac{1}{280}$, $\delta = 0.01$ |
| Popolazione Media | 10000 |
| Tasso di Migrazione Massimo | 0.001 |
| Tolleranza Controllore | $1e - 3$ |
| Finestra di Intervento | 10 |
| Massimo Numero di Iterazioni Training Loop | 100 |
| Massimo Numero di Iterazioni Senza Miglioramento | 3 |

5.1 Nessun Intervento

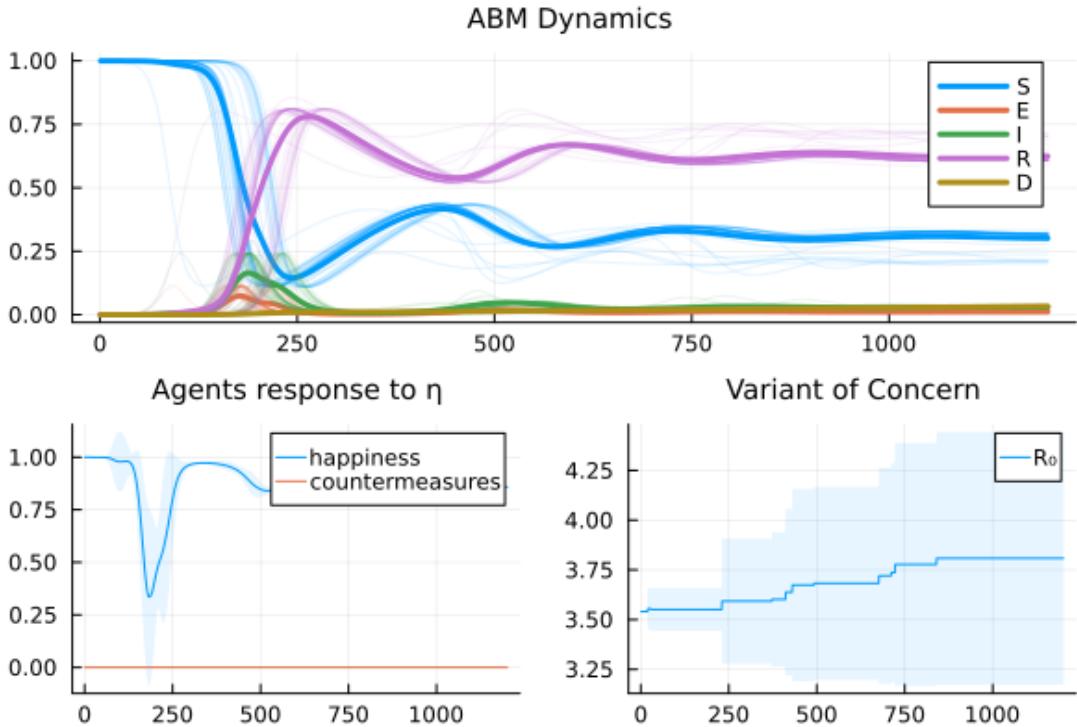


Figure 44: Grafico cumulativo del modello senza alcun tipo di intervento

Il grafico (Figura 44) mostra l'andamento delle curve del modello quando questo viene eseguito senza alcuna tipologia di intervento. Questo andamento è rappresentato in modo cumulativo rispetto all'andamento dei singoli agenti, i quali possono mostrare comportamenti differenti tra loro.

Complessivamente, l'andamento del modello è simile a quello standard di un modello di tipo SEIR, con alcune variazioni dovute ai fattori di stocasticità intrinseci del modello. Il grafico mostra le traiettorie più comuni delle curve cumulative del modello, mettendo in evidenza i percorsi più frequenti.

Come è possibile notare, il numero di individui suscettibili diminuisce drasticamente a causa della rapida diffusione esponenziale del virus. Questo è accompagnato da una crescita altrettanto rapida del numero di individui guariti (recovered). Tuttavia, a causa della definizione delle condizioni per i guariti, questi individui non sono immuni alle varianti del virus, il che permette di modellare una possibile ciclicità dell'epidemia dovuta alla perdita di immunità della popolazione. Queste proprietà contribuiscono all'andamento ciclico delle curve. Inoltre, si nota che la curva associata all'andamento degli individui nella classe D ha una crescita lineare, sebbene non molto evidente.

La curva associata alla variabile di “happiness” del modello, utilizzata per bilanciare la severità delle misure di controllo per evitare un ciclo insostenibile, mostra un comportamento piuttosto peculiare. Questo è principalmente dovuto alla definizione della funzione di controllo della felicità (vedi Figura 24). Inoltre, la curva tende ad oscillare in modo ciclico a causa della definizione delle misure di controllo.

In generale, il modello senza interventi mostra un’epidemia in rapida crescita seguita da periodi ciclici di riduzione della diffusione del virus.

5.2 Intervento Non Farmaceutico

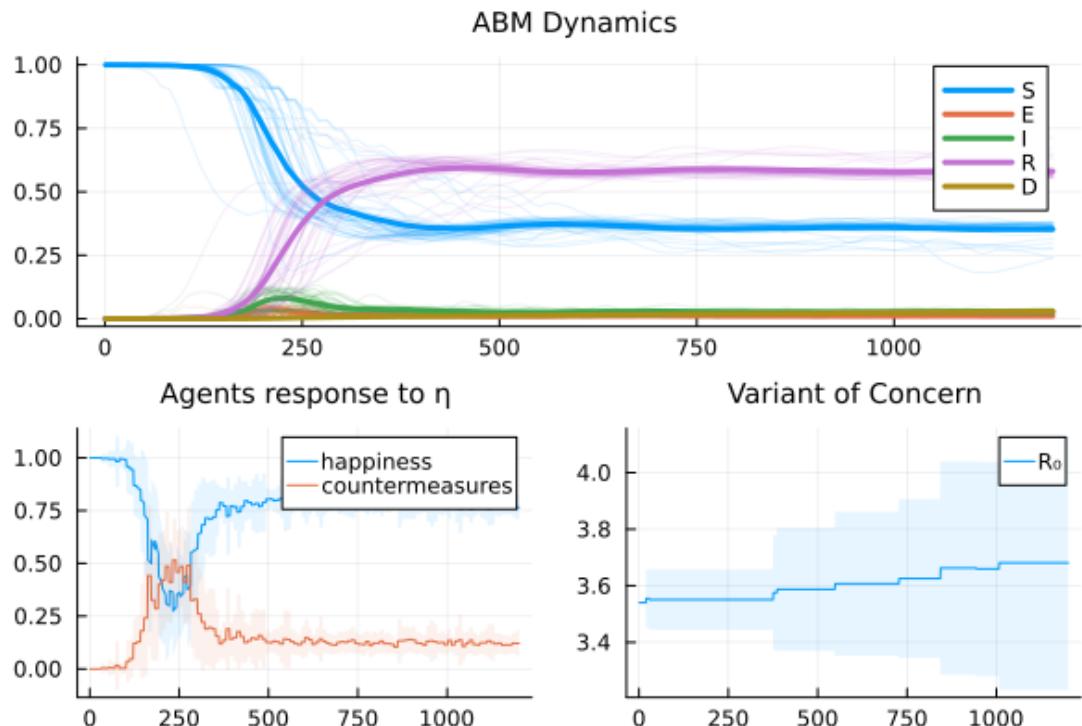


Figure 45: Grafico cumulativo del modello con intervento non farmaceutico

Nel grafico (Figura 45) è rappresentato l’andamento delle curve quando viene applicato un intervento non farmaceutico, come il distanziamento sociale, l’uso di maschere, ecc. L’obiettivo di questo tipo di intervento è quello di rallentare la diffusione del virus senza utilizzare farmaci o vaccini.

Come si può vedere dal grafico, l’intervento non farmaceutico ha un impatto significativo sulla diffusione del virus. In particolare, il numero di individui suscettibili rimane più stabile nel tempo rispetto al caso senza interventi, e il picco dell’epidemia è notevolmente ridotto. Questo è dovuto all’efficacia delle misure di distanziamento sociale e al fatto che esse riducono il tasso di contatto tra gli individui infetti e suscettibili. Tuttavia, il

numero di individui guariti è ancora ciclico a causa della perdita di immunità, e la curva di felicità mostra comunque oscillazioni dovute alle misure di controllo cicliche.

In generale, l'intervento non farmaceutico riesce a contenere l'epidemia e a ridurne l'entità, ma non riesce a eliminarla completamente.

5.3 Intervento Farmaceutico

Il seguente grafico (Figura 46) rappresenta l'andamento delle curve quando viene applicato un intervento farmaceutico, come la somministrazione di vaccini o farmaci antivirali. L'obiettivo di questo tipo di intervento è quello di ridurre significativamente la diffusione del virus e, idealmente, di eliminare l'epidemia.

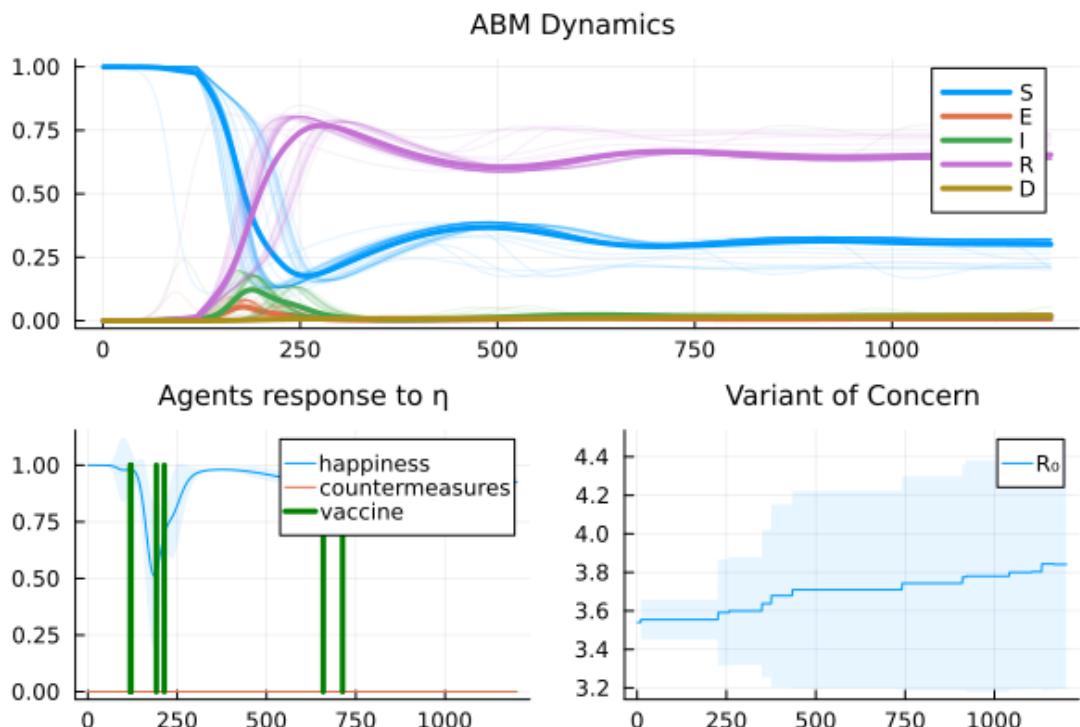


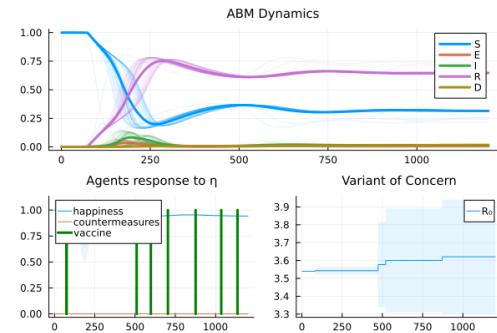
Figure 46: Grafico cumulativo del modello con intervento farmaceutico

Come si può notare, l'intervento farmaceutico ha un impatto molto positivo sulla diffusione del virus. Il numero di individui suscettibili rimane stabile nel tempo, e l'epidemia viene rapidamente contenuta. Questo è dovuto alla somministrazione di farmaci o vaccini, che riducono la capacità di trasmissione del virus e proteggono gli individui suscettibili. Inoltre, il numero di individui guariti non mostra più l'andamento ciclico, poiché i soggetti guariti attraverso l'intervento farmaceutico acquisiscono immunità duratura.

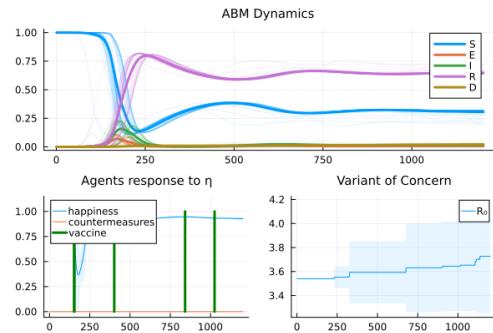
La curva di felicità mostra ancora alcune oscillazioni dovute alle misure di controllo cicliche, ma queste oscillazioni sono meno pronunciate rispetto ai casi precedenti. In generale, l'intervento farmaceutico è estremamente efficace nel contenere e ridurre l'epidemia.

5.3.1 Grafici di Confronto

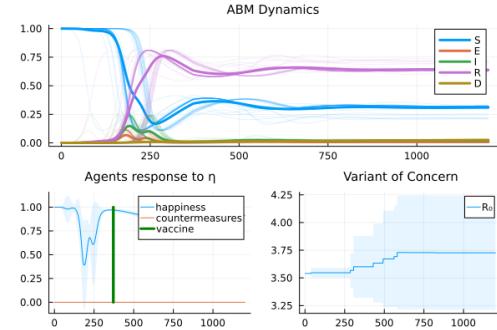
I grafici di confronto mostrano l'impatto del momento di inizio della campagna di vaccinazione su diversi scenari. Questi grafici illustrano come l'efficacia della campagna vaccinale possa dipendere dal suo avvio, oltre al numero di persone vaccinate.



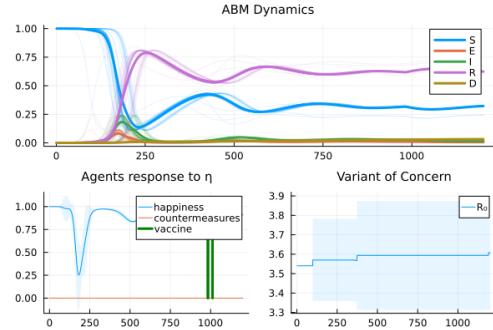
(a) Grafico per la comparazione sul periodo di inizio della campagna vaccinale. Immediata



(b) Grafico per la comparazione sul periodo di inizio della campagna vaccinale. Durante prima ondata



(c) Grafico per la comparazione sul periodo di inizio della campagna vaccinale. Poco dopo la prima ondata



(d) Grafico per la comparazione sul periodo di inizio della campagna vaccinale. In ritardo

5.4 Intervento Farmaceutico e Non Farmaceutico

Il seguente grafico (Figura 48) rappresenta l'andamento delle curve quando vengono applicati sia un intervento farmaceutico che un intervento non farmaceutico. L'obiettivo di questa combinazione di interventi è quello di massimizzare la riduzione della diffusione del virus e di contenere l'epidemia in modo ottimale.

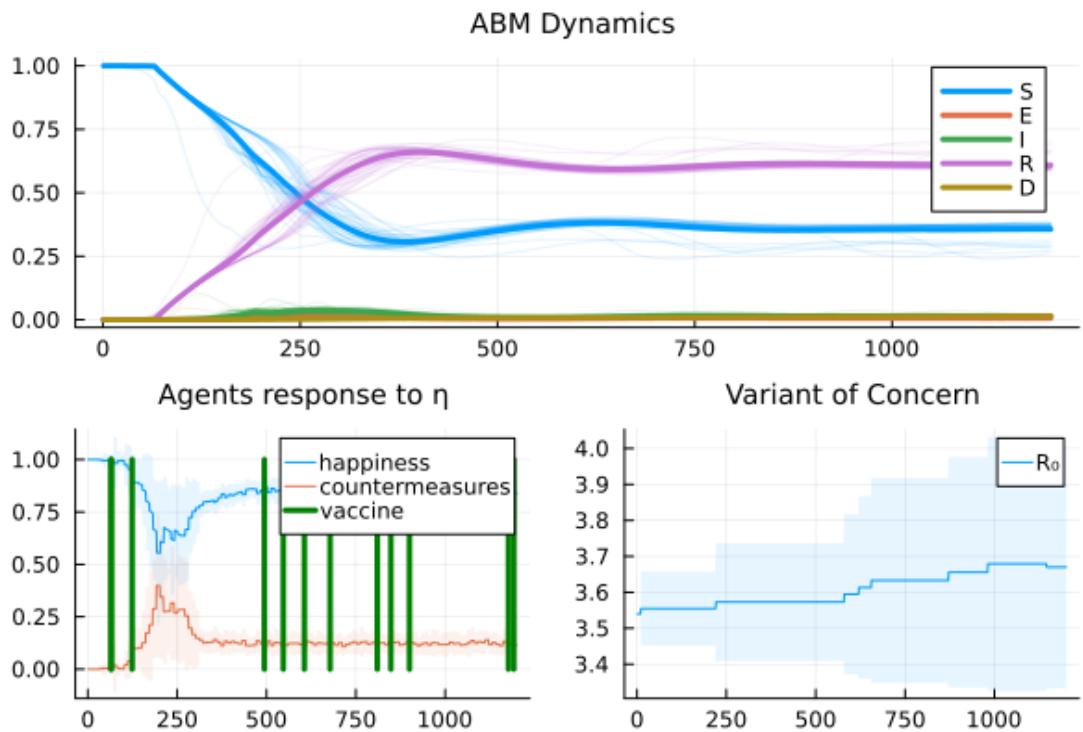


Figure 48: Grafico cumulativo del modello con intervento farmaceutico e non farmaceutico combinato

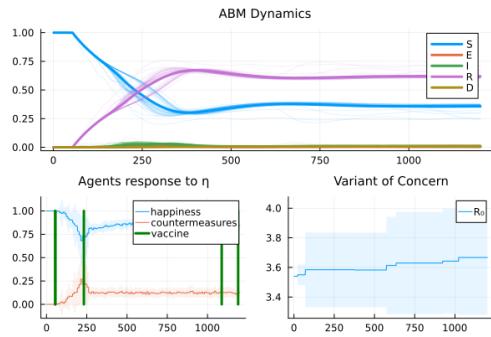
Come mostrato nel grafico, l'effetto combinato di intervento farmaceutico e non farmaceutico è estremamente efficace nel contenere l'epidemia. Il numero di individui suscettibili rimane stabile nel tempo, e l'epidemia viene rapidamente ridotta a livelli molto bassi. La curva di individui guariti mostra un aumento costante nel tempo, poiché l'intervento farmaceutico garantisce immunità duratura.

La curva di felicità mostra ancora alcune oscillazioni dovute alle misure di controllo cicliche, ma queste oscillazioni sono molto meno pronunciate rispetto ai casi precedenti.

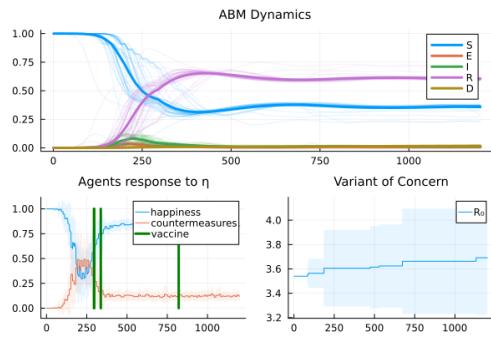
In generale, la combinazione di intervento farmaceutico e non farmaceutico è altamente efficace nel contenere e ridurre l'epidemia in modo ottimale.

5.4.1 Grafici di Confronto

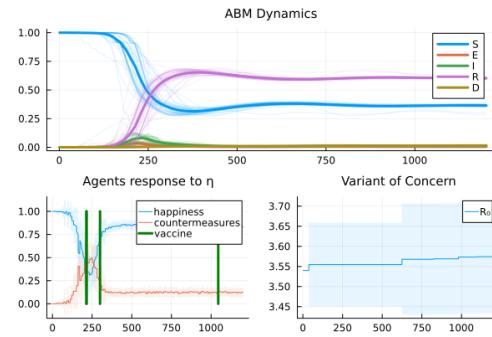
I grafici di confronto mostrano l'impatto del momento di inizio della campagna di vaccinazione su diversi scenari. Questi grafici illustrano come l'efficacia della campagna vaccinale possa dipendere dal suo avvio, oltre al numero di persone vaccinate, e come questa influenzi attivamente le contromisure non farmaceutiche.



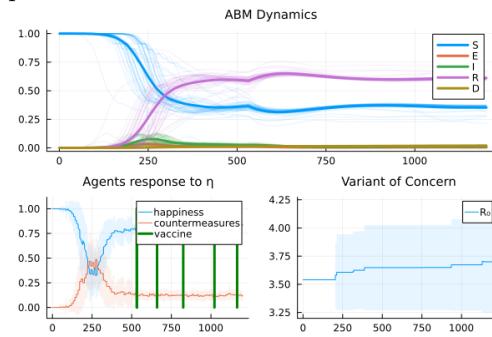
(a) Grafico per la comparazione sul periodo di inizio della campagna vaccinale. Immediata



(c) Grafico per la comparazione sul periodo di inizio della campagna vaccinale. Poco dopo la prima ondata



(b) Grafico per la comparazione sul periodo di inizio della campagna vaccinale. Durante prima ondata



(d) Grafico per la comparazione sul periodo di inizio della campagna vaccinale. In ritardo

6 Conclusioni

In conclusione, i risultati ottenuti dimostrano l'efficacia delle misure di controllo, sia farmaceutiche che non farmaceutiche, nel contenere e ridurre la diffusione del virus. L'intervento farmaceutico, in particolare, si è dimostrato altamente efficace nel contrastare l'epidemia e nel prevenire il ciclo insostenibile di infezioni. Tuttavia, da queste simulazioni, è importante sottolineare che l'intervento farmaceutico da solo potrebbe non essere sufficiente per l'eliminazione completa dell'epidemia. Al contrario, l'adozione di una combinazione di interventi farmaceutici e non farmaceutici offre una protezione ottimale e una strategia più completa per la gestione delle pandemie.

Va notato che i tempi di calcolo delle simulazioni sono notevolmente influenzati dall'applicazione delle misure di controllo non farmaceutiche. Tuttavia, questa è un compromesso necessario per ottenere risultati accurati e realistici che tengano conto della complessità delle dinamiche sociali ed epidemiologiche. Inoltre, l'osservazione di oscillazioni nelle curve di felicità, causate dall'uso di misure di controllo cicliche, non rappresenta un problema insormontabile e può essere gestita adeguatamente.

In generale, il modello di simulazione dell'epidemia su una rete sociale, unitamente all'utilizzo di modelli matematici compartmentali di tipo SEIR con l'applicazione di un controllore gestito tramite Neural ODE, si configura come una piattaforma estremamente utile per lo studio degli effetti delle misure di controllo e per lo sviluppo di strategie ottimali per la gestione delle epidemie. I risultati ottenuti contribuiscono alla comprensione e alla preparazione per affrontare situazioni epidemiologiche complesse e dinamiche come quelle delle pandemie.

7 Sviluppi Futuri

7.1 Perfezionamento del Controllore

Un possibile sviluppo futuro di rilevanza è il perfezionamento del controllore, mirando a migliorarne sia le capacità di individuazione che l'applicazione delle policy. Inoltre, si potrebbe cercare di rendere le policy generate dal controllore più comprensibili per gli operatori umani, in modo da agevolare la loro traduzione e comprensione. Questa evoluzione apre la porta a un'approfondita analisi di causalità, che è un elemento di estrema importanza, come precedentemente discusso. In particolare, sarà necessario esplorare la relazione tra l'applicazione di specifiche contromisure e il loro impatto sulla riduzione di una pandemia.

Tale approccio si presenta come una sfida complessa ma altamente promettente, poiché potrebbe rappresentare un notevole avanzamento tecnologico.

7.2 Miglioramento della Funzione di Happiness

Un'altra importante area di sviluppo futuro riguarda la funzione di happiness, che gioca un ruolo cruciale nel controllo del controllore. Attualmente, questa funzione è utilizzata in modo semplice per evitare l'attivazione di contromisure insostenibili nella vita reale. Tuttavia, è possibile migliorarla ulteriormente, tenendo conto dell'effettivo impatto che ciascuna tipologia di contromisura può avere sulla vita delle persone, sia nel breve che nel lungo periodo. Questo miglioramento potrebbe rappresentare una pietra miliare per l'accuratezza delle policy sviluppate, fornendo un quadro di valutazione più preciso.

Tuttavia, anche questa area di sviluppo richiede un'analisi attenta della causalità degli eventi, in particolare per stimare l'effetto delle contromisure sull'umore complessivo di una popolazione. L'umore può variare a causa di numerosi fattori, comprese le contromisure e gli stimoli esterni, e l'analisi richiederà il supporto di diversi benchmark con modelli di simulazione diversificati.

7.3 Perfezionamento Generale del Modello

Il modello attualmente implementato si basa su assunzioni relativamente realistiche, derivanti da osservazioni empiriche. Tuttavia, il modello potrebbe beneficiare di un maggiore grado di generalizzazione e flessibilità per adattarsi a una varietà di problemi. Ciò comporterebbe la necessità di superare alcune rigidità strutturali attuali che ne limitano l'utilità in contesti diversi.

7.4 Ottimizzazione Generale

Un ulteriore sviluppo importante riguarda l'ottimizzazione generale del codice. Attualmente, il codice potrebbe essere ulteriormente migliorato in termini di efficienza, scal-

abilità e ottimizzazione delle performance. Questo sforzo mirerebbe a rendere il codice più performante e adattabile alle esigenze future.

References

- [1] O. N. Bjørnstad, K. Shea, M. Krzywinski, and N. Altman, “The SEIRS model for infectious disease dynamics,” *Nature Methods*, vol. 17, no. 6, pp. 557–558, Jun 2020. [Online]. Available: <https://doi.org/10.1038/s41592-020-0856-2>
- [2] P. Ciunkiewicz, W. Brooke, M. Rogers, and S. Yanushkevich, “Agent-based epidemiological modeling of covid-19 in localized environments,” *Computers in Biology and Medicine*, vol. 144, p. 105396, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482522001883>
- [3] G. Datseris, A. R. Vahdati, and T. C. DuBois, “Agents.jl: a performant and feature-full agent-based modeling software of minimal code complexity,” *SIMULATION*, p. 003754972110688, jan 2022. [Online]. Available: <https://doi.org/10.1177%2F00375497211068820>
- [4] S. Abar, G. K. Theodoropoulos, P. Lemarinier, and G. M. O’Hare, “Agent based modelling and simulation tools: A review of the state-of-art software,” *Computer Science Review*, vol. 24, pp. 13–33, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1574013716301198>
- [5] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, and A. Ramadhan, “Universal differential equations for scientific machine learning,” *arXiv preprint arXiv:2001.04385*, 2020.
- [6] C. Rackauckas, M. Innes, Y. Ma, J. Bettencourt, L. White, and V. Dixit, “Diffeqflux.jl-a julia library for neural differential equations,” *arXiv preprint arXiv:1902.02376*, 2019.
- [7] B. Tasseeff, C. Coffrin, A. Wächter, and C. Laird, “Exploring benefits of linear solver parallelism on modern nonlinear optimization applications,” 2019.
- [8] T. Hale, N. Angrist, R. Goldszmidt, B. Kira, A. Petherick, T. Phillips, S. Webster, E. Cameron-Blake, L. Hallas, S. Majumdar, and H. Tatlow, “A global panel database of pandemic policies (Oxford COVID-19 government response tracker),” *Nature Human Behaviour*, vol. 5, no. 4, pp. 529–538, Apr 2021. [Online]. Available: <https://doi.org/10.1038/s41562-021-01079-8>
- [9] J. Chen and J. Revels, “Robust benchmarking in noisy environments,” 2016.
- [10] R. R. Frerichs and R. Neutra, “RE: “definitions of epidemiology”,” *Am J Epidemiol*, vol. 108, no. 1, pp. 74–75, Jul. 1978.
- [11] H. Ward, M. B. Toledano, G. Shaddick, B. Davies, and P. Elliott, *Oxford handbook of epidemiology for clinicians*. OUP Oxford, 2012.
- [12] W. O. Kermack and A. G. McKendrick, “A contribution to the mathematical theory of epidemics,” *Proceedings of the royal society of london. Series A, Containing*

papers of a mathematical and physical character, vol. 115, no. 772, pp. 700–721, 1927.

- [13] S. Galea, M. Riddle, and G. A. Kaplan, “Causal thinking and complex system approaches in epidemiology,” *Int J Epidemiol*, vol. 39, no. 1, pp. 97–106, Oct. 2009.
- [14] M. Parascandola and D. L. Weed, “Causation in epidemiology,” *J Epidemiol Community Health*, vol. 55, no. 12, pp. 905–912, Dec. 2001.
- [15] K. Rothman, S. Greenland, and T. Lash, *Modern Epidemiology*. Wolters Kluwer Health/Lippincott Williams & Wilkins, 2015. [Online]. Available: <https://books.google.it/books?id=MSTgnQAACAAJ>
- [16] P. Sanchez, J. P. Voisey, T. Xia, H. I. Watson, A. Q. O’Neil, and S. A. Tsafaris, “Causal machine learning for healthcare and precision medicine,” *Royal Society Open Science*, vol. 9, no. 8, p. 220638, 2022. [Online]. Available: <https://royalsocietypublishing.org/doi/abs/10.1098/rsos.220638>
- [17] J. D. Enderle, “Chapter 7 - compartmental modeling,” in *Introduction to Biomedical Engineering (Third Edition)*, third edition ed., ser. Biomedical Engineering, J. D. Enderle and J. D. Bronzino, Eds. Boston: Academic Press, 2012, pp. 359–445. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123749796000071>
- [18] M. L. Abell and J. P. Braselton, “Chapter 6 - systems of ordinary differential equations,” in *Differential Equations with Mathematica (Fifth Edition)*, fifth edition ed., M. L. Abell and J. P. Braselton, Eds. Academic Press, 2023, pp. 283–384. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780128241608000115>
- [19] S. Gottlieb and D. Ketcheson, “Chapter 21 - time discretization techniques,” in *Handbook of Numerical Methods for Hyperbolic Problems*, ser. Handbook of Numerical Analysis, R. Abgrall and C.-W. Shu, Eds. Elsevier, 2016, vol. 17, pp. 549–583. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1570865916300072>
- [20] F. Brauer, *Compartmental Models in Epidemiology*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 19–79. [Online]. Available: https://doi.org/10.1007/978-3-540-78911-6_2
- [21] C. W. Weimer, J. O. Miller, and R. R. Hill, “Agent-based modeling: An introduction and primer,” in *2016 Winter Simulation Conference (WSC)*, 2016, pp. 65–79.
- [22] K. R. Bissett, J. Cadena, M. Khan, and C. J. Kuhlman, “Agent-based computational epidemiological modeling,” *Journal of the Indian Institute*

of Science, vol. 101, no. 3, pp. 303–327, Jul 2021. [Online]. Available: <https://doi.org/10.1007/s41745-021-00260-2>

- [23] A. M. El-Sayed, P. Scarborough, L. Seemann, and S. Galea, “Social network analysis and agent-based modeling in social epidemiology,” *Epidemiol Perspect Innov*, vol. 9, no. 1, p. 1, Feb. 2012.
- [24] E. R. Groff, S. D. Johnson, and A. Thornton, “State of the art in agent-based modeling of urban crime: An overview,” *Journal of Quantitative Criminology*, vol. 35, no. 1, pp. 155–193, Mar 2019. [Online]. Available: <https://doi.org/10.1007/s10940-018-9376-y>
- [25] M. Tracy, M. Cerdá, and K. M. Keyes, “Agent-Based modeling in public health: Current applications and future directions,” *Annu Rev Public Health*, vol. 39, pp. 77–94, Jan. 2018.
- [26] W. Dubitzky, O. Wolkenhauer, K.-H. Cho, and H. Yokota, Eds., *Gillespie Algorithm*. New York, NY: Springer New York, 2013, pp. 839–839. [Online]. Available: https://doi.org/10.1007/978-1-4419-9863-7_100581
- [27] C. Rackauckas and Q. Nie, “Differentialequations.jl—a performant and feature-rich ecosystem for solving differential equations in julia,” *Journal of Open Research Software*, vol. 5, no. 1, p. 15, 2017.
- [28] J. Fairbanks, M. Besançon, S. Simon, J. Hoffiman, N. Eubank, and S. Karpinski, “JuliaGraphs/Graphs.jl: an optimized graphs package for the Julia programming language,” 2021. [Online]. Available: <https://github.com/JuliaGraphs/Graphs.jl/>
- [29] M. Innes, E. Saba, K. Fischer, D. Gandhi, M. C. Rudilosso, N. M. Joy, T. Karmali, A. Pal, and V. Shah, “Fashionable modelling with flux,” 2018.
- [30] M. Innes, “Flux: Elegant machine learning with julia,” *Journal of Open Source Software*, vol. 3, no. 25, p. 602, 2018. [Online]. Available: <https://doi.org/10.21105/joss.00602>
- [31] A. Pal, “Lux: Explicit Parameterization of Deep Neural Networks in Julia,” 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.7808904>
- [32] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural ordinary differential equations,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf
- [33] S. Kim, W. Ji, S. Deng, Y. Ma, and C. Rackauckas, “Stiff neural ordinary differential equations,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 9, p. 093122, sep 2021. [Online]. Available: <https://doi.org/10.1063%2F5.0060697>

- [34] T. Nishijima, “Universal approximation theorem for neural networks,” 2021.
- [35] D. Fitzpatrick, “Chapter 5 - parametric sweep,” in *Analog Design and Simulation Using OrCAD Capture and PSpice (Second Edition)*, second edition ed., D. Fitzpatrick, Ed. Newnes, 2018, pp. 79–98. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780081025055000057>
- [36] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [37] JuliusMartensen, C. Rackauckas *et al.*, “Datadrivendiffeq.jl,” Jul. 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.5083412>
- [38] C. Tsitouras, “Runge-kutta pairs of order 5(4) satisfying only the first column simplifying assumption,” *Comput. Math. Appl.*, vol. 62, no. 2, p. 770–775, jul 2011. [Online]. Available: <https://doi.org/10.1016/j.camwa.2011.06.002>
- [39] M. Bartholomew-Biggs, S. Brown, B. Christianson, and L. Dixon, “Automatic differentiation of algorithms,” *Journal of Computational and Applied Mathematics*, vol. 124, no. 1, pp. 171–190, 2000, numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377042700004222>
- [40] Q. Wang, “Forward and adjoint sensitivity computation of chaotic dynamical systems,” *Journal of Computational Physics*, vol. 235, pp. 1–13, 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999112005360>
- [41] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, “Network analysis in the social sciences,” *Science*, vol. 323, no. 5916, pp. 892–895, 2009. [Online]. Available: <https://www.science.org/doi/abs/10.1126/science.1165821>
- [42] B. Wellman, “The development of social network analysis: A study in the sociology of science,” *Contemporary Sociology*, vol. 37, no. 3, pp. 221–222, 2008. [Online]. Available: <https://doi.org/10.1177/009430610803700308>
- [43] W. de Nooy, *Social Network Analysis, Graph Theoretical Approaches to*. New York, NY: Springer New York, 2012, pp. 2864–2877. [Online]. Available: https://doi.org/10.1007/978-1-4614-1800-9_176
- [44] X. Ding, S. Huang, A. Leung, and R. Rabbany, “Incorporating dynamic flight network in seir to model mobility between populations,” *Applied Network Science*, vol. 6, no. 1, p. 42, Jun 2021. [Online]. Available: <https://doi.org/10.1007/s41109-021-00378-3>
- [45] M. Abavisani, K. Rahimian, B. Mahdavi, S. Tokhanbigli, M. Mollapour Siasakht, A. Farhadi, M. Kodori, M. Mahmanzar, and Z. Meshkat, “Mutations in sars-cov-2 structural proteins: a global analysis,” *Virology Journal*, vol. 19, no. 1, p. 220, Dec 2022. [Online]. Available: <https://doi.org/10.1186/s12985-022-01951-7>

- [46] P. V. Markov, M. Ghafari, M. Beer, K. Lythgoe, P. Simmonds, N. I. Stilianakis, and A. Katzourakis, “The evolution of sars-cov-2,” *Nature Reviews Microbiology*, vol. 21, no. 6, pp. 361–379, Jun 2023. [Online]. Available: <https://doi.org/10.1038/s41579-023-00878-2>
- [47] S. Wang, X. Xu, C. Wei, S. Li, J. Zhao, Y. Zheng, X. Liu, X. Zeng, W. Yuan, and S. Peng, “Molecular evolutionary characteristics of sars-cov-2 emerging in the united states,” *Journal of Medical Virology*, vol. 94, no. 1, pp. 310–317, 2022. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jmv.27331>
- [48] L. Böttcher and T. Asikis, “Near-optimal control of dynamical systems with neural ordinary differential equations,” *Machine Learning: Science and Technology*, vol. 3, no. 4, p. 045004, oct 2022. [Online]. Available: <https://doi.org/10.1088%2F2632-2153%2Fac92c3>
- [49] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V. B. Shah, and W. Tebbutt, “A differentiable programming system to bridge machine learning and scientific computing,” 2019.
- [50] I. O. Sandoval, P. Petsagkourakis, and E. A. del Rio-Chanona, “Neural odes as feedback policies for nonlinear optimal control,” 2022.
- [51] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.
- [52] C. G. BROYDEN, “The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations,” *IMA Journal of Applied Mathematics*, vol. 6, no. 1, pp. 76–90, 03 1970. [Online]. Available: <https://doi.org/10.1093/imamat/6.1.76>
- [53] D. Goldfarb, “A family of variable-metric methods derived by variational means,” *Mathematics of Computation*, vol. 24, no. 109, pp. 23–26, 1970. [Online]. Available: <http://www.jstor.org/stable/2004873>
- [54] D. F. Shanno, “Conditioning of quasi-newton methods for function minimization,” *Mathematics of Computation*, vol. 24, no. 111, pp. 647–656, 1970. [Online]. Available: <http://www.jstor.org/stable/2004840>
- [55] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” 2017.
- [56] H. Xin, Y. Li, P. Wu, Z. Li, E. H. Y. Lau, Y. Qin, L. Wang, B. J. Cowling, T. K. Tsang, and Z. Li, “Estimating the Latent Period of Coronavirus Disease 2019 (COVID-19),” *Clinical Infectious Diseases*, vol. 74, no. 9, pp. 1678–1681, 09 2021. [Online]. Available: <https://doi.org/10.1093/cid/ciab746>
- [57] C. C. Kerr, R. M. Stuart, D. Mistry, R. G. Abeyseuriya, K. Rosenfeld, G. R. Hart, R. C. Núñez, J. A. Cohen, P. Selvaraj, B. Hagedorn, L. George, M. Jastrzębski, A. S. Izzo, G. Fowler, A. Palmer, D. Delport, N. Scott, S. L. Kelly, C. S. Bennette, B. G. Wagner, S. T. Chang, A. P. Oron, E. A. Wenger, J. Panovska-Griffiths,

- M. Famulare, and D. J. Klein, “Covasim: An agent-based model of covid-19 dynamics and interventions,” *PLOS Computational Biology*, vol. 17, no. 7, pp. 1–32, 07 2021. [Online]. Available: <https://doi.org/10.1371/journal.pcbi.1009149>
- [58] A. Asanjarani, A. Shausan, K. Chew, T. Graham, S. G. Henderson, H. M. Jansen, K. R. Short, P. G. Taylor, A. Vuorinen, Y. Yadav, I. Ziedins, and Y. Nazarathy, “Emulation of epidemics via bluetooth-based virtual safe virus spread: Experimental setup, software, and data,” *PLOS Digital Health*, vol. 1, no. 12, pp. 1–23, 12 2022. [Online]. Available: <https://doi.org/10.1371/journal.pdig.0000142>
- [59] M. Frérot, A. Lefebvre, S. Aho, P. Callier, K. Astruc, and L. S. Aho Glélé, “What is epidemiology? changing definitions of epidemiology 1978-2017,” *PLOS ONE*, vol. 13, no. 12, pp. 1–27, 12 2018. [Online]. Available: <https://doi.org/10.1371/journal.pone.0208442>
- [60] L. J. S. Allen, *An Introduction to Stochastic Epidemic Models*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 81–130. [Online]. Available: https://doi.org/10.1007/978-3-540-78911-6_3
- [61] N. Altman and M. Krzywinski, “Association, correlation and causation,” *Nature Methods*, vol. 12, no. 10, pp. 899–900, Oct 2015. [Online]. Available: <https://doi.org/10.1038/nmeth.3587>
- [62] M. H. Bonds, D. C. Keenan, P. Rohani, and J. D. Sachs, “Poverty trap formed by the ecology of infectious diseases,” *Proc Biol Sci*, vol. 277, no. 1685, pp. 1185–1192, Dec. 2009.
- [63] R. Dandekar, S. G. Henderson, H. M. Jansen, J. McDonald, S. Moka, Y. Nazarathy, C. Rackauckas, P. G. Taylor, and A. Vuorinen, “Safe blues: The case for virtual safe virus spread in the long-term fight against epidemics,” *Patterns*, vol. 2, no. 3, p. 100220, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666389921000349>
- [64] C. I. Siettos and L. Russo, “Mathematical modeling of infectious disease dynamics,” *Virulence*, vol. 4, no. 4, pp. 295–306, 2013, pMID: 23552814. [Online]. Available: <https://doi.org/10.4161/viru.24041>
- [65] G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. Di Filippo, A. Di Matteo, and M. Colaneri, “Modelling the covid-19 epidemic and implementation of population-wide interventions in italy,” *Nature Medicine*, vol. 26, no. 6, pp. 855–860, Jun 2020. [Online]. Available: <https://doi.org/10.1038/s41591-020-0883-7>
- [66] B. Ebbinghaus, L. Lehner, and E. Naumann, “Welfare state support during the covid-19 pandemic: Change and continuity in public attitudes towards social policies in germany,” *European Policy Analysis*, vol. 8, no. 3, pp. 297–311, 2022. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/epa2.1152>

- [67] A. Godio, F. Pace, and A. Vergnano, “Seir modeling of the italian epidemic of sars-cov-2 using computational swarm intelligence,” *International Journal of Environmental Research and Public Health*, vol. 17, no. 10, 2020. [Online]. Available: <https://www.mdpi.com/1660-4601/17/10/3535>
- [68] M. Jalayer, C. Orsenigo, and C. Vercellis, “Cov-abm: A stochastic discrete-event agent-based framework to simulate spatiotemporal dynamics of covid-19,” 2020.
- [69] J. Angevaare, Z. Feng, and R. Deardon, “Pathogen.jl: Infectious disease transmission network modeling with julia,” *Journal of Statistical Software*, vol. 104, no. 4, p. 1–30, 2022. [Online]. Available: <https://www.jstatsoft.org/index.php/jss/article/view/v104i04>
- [70] G. Konstantinoudis, D. Schuhmacher, H. Rue, and B. D. Spycher, “Discrete versus continuous domain models for disease mapping,” *Spatial and Spatio-temporal Epidemiology*, vol. 32, p. 100319, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877584519301297>
- [71] J. Ladyman, J. Lambert, and K. Wiesner, “What is a complex system?” *European Journal for Philosophy of Science*, vol. 3, no. 1, pp. 33–67, Jan 2013. [Online]. Available: <https://doi.org/10.1007/s13194-012-0056-8>
- [72] S. Marzban, R. Han, N. Juhász, and G. Röst, “A hybrid PDE-ABM model for viral dynamics with application to SARS-CoV-2 and influenza,” *R Soc Open Sci*, vol. 8, no. 11, p. 210787, Nov. 2021.
- [73] S. Mwalili, M. Kimathi, V. Ojiambo, D. Gathungu, and R. Mbogo, “Seir model for covid-19 dynamics incorporating the environment and social distancing,” *BMC Research Notes*, vol. 13, no. 1, p. 352, Jul 2020. [Online]. Available: <https://doi.org/10.1186/s13104-020-05192-1>
- [74] J. T. Nardini, R. E. Baker, M. J. Simpson, and K. B. Flores, “Learning differential equation models from stochastic agent-based model simulations,” *J R Soc Interface*, vol. 18, no. 176, p. 20200987, Mar. 2021.
- [75] H. Nishiura, “Correcting the actual reproduction number: a simple method to estimate $r(0)$ from early epidemic growth data,” *Int J Environ Res Public Health*, vol. 7, no. 1, pp. 291–302, Jan. 2010.
- [76] A. Novakovic and A. H. Marshall, “The cp-abm approach for modelling covid-19 infection dynamics and quantifying the effects of non-pharmaceutical interventions,” *Pattern Recognition*, vol. 130, p. 108790, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320322002710>
- [77] S. P. Prajapati, R. Bhaumik, and T. Kumar, “An intelligent abm-based framework for developing pandemic-resilient urban spaces in post-covid smart cities,” *Procedia Computer Science*, vol. 218, pp. 2299–2308, 2023, international

- Conference on Machine Learning and Data Engineering. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050923002053>
- [78] E. Silverman, U. Gostoli, S. Picascia, J. Almagor, M. McCann, R. Shaw, and C. Angione, “Situating agent-based modelling in population health research,” *Emerging Themes in Epidemiology*, vol. 18, no. 1, p. 10, Jul 2021. [Online]. Available: <https://doi.org/10.1186/s12982-021-00102-7>
- [79] W. B. Group, “Wdr 2022 chapter 1. introduction,” Mar 2023. [Online]. Available: <https://www.worldbank.org/en/publication/wdr2022/brief/chapter-1-introduction-the-economic-impacts-of-the-covid-19-crisis>
- [80] W. H. Organization, “Who coronavirus (covid-19) dashboard.” [Online]. Available: <https://covid19.who.int/>
- [81] J. H. Zaccai, “How to assess epidemiological studies,” *Postgrad Med J*, vol. 80, no. 941, pp. 140–147, Mar. 2004.
- [82] C. Rackauckas and Q. Nie, “Confederated modular differential equation apis for accelerated algorithm development and benchmarking,” *Advances in Engineering Software*, vol. 132, pp. 1–6, 2019.
- [83] Y. Ma, V. Dixit, M. J. Innes, X. Guo, and C. Rackauckas, “A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions,” in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, 2021, pp. 1–9.
- [84] S. Gowda, Y. Ma, V. Churavy, A. Edelman, and C. Rackauckas, “Sparsity programming: Automated sparsity-aware optimizations in differentiable programming,” 2019.
- [85] N. Wu, K. Joyal-Desmarais, P. A. B. Ribeiro, A. M. Vieira, J. Stojanovic, C. Sanuade, D. Yip, and S. L. Bacon, “Long-term effectiveness of COVID-19 vaccines against infections, hospitalisations, and mortality in adults: findings from a rapid living systematic evidence synthesis and meta-analysis up to december, 2022,” *Lancet Respir. Med.*, vol. 11, no. 5, pp. 439–452, May 2023.
- [86] R. Dandekar, K. Chung, V. Dixit, M. Tarek, A. Garcia-Valadez, K. V. Vemula, and C. Rackauckas, “Bayesian neural ordinary differential equations,” 2022.
- [87] T. Britton and L. Leskelä, “Optimal intervention strategies for minimizing total incidence during an epidemic,” *SIAM Journal on Applied Mathematics*, vol. 83, no. 2, pp. 354–373, mar 2023. [Online]. Available: <https://doi.org/10.1137%2F22m1504433>
- [88] P.-A. Bliman and M. Duprez, “How best can finite-time social distancing reduce epidemic final size?” *Journal of Theoretical Biology*, vol. 511, p. 110557, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0022519320304124>

- [89] F. Córdova-Lepe and K. Vogt-Geisse, “Adding a reaction-restoration type transmission rate dynamic-law to the basic seir covid-19 model,” *PLOS ONE*, vol. 17, no. 6, pp. 1–21, 06 2022. [Online]. Available: <https://doi.org/10.1371/journal.pone.0269843>
- [90] C. Rackauckas, M. Innes, Y. Ma, J. Bettencourt, L. White, and V. Dixit, “Diffeqflux.jl - A julia library for neural differential equations,” *CoRR*, vol. abs/1902.02376, 2019. [Online]. Available: <https://arxiv.org/abs/1902.02376>
- [91] E. Roesch, C. Rackauckas, and M. P. H. Stumpf, “Collocation based training of neural ordinary differential equations,” *Statistical Applications in Genetics and Molecular Biology*, vol. 20, no. 2, pp. 37–49, 2021. [Online]. Available: <https://doi.org/10.1515/sagmb-2020-0025>
- [92] E. M. Turan and J. Jaschke, “Multiple shooting for training neural differential equations on time series,” *IEEE Control Systems Letters*, vol. 6, pp. 1897–1902, 2022. [Online]. Available: <https://doi.org/10.1109%2Flcsls.2021.3135835>
- [93] P. G. Fennell, S. Melnik, and J. P. Gleeson, “Limitations of discrete-time approaches to continuous-time contagion dynamics,” *Phys. Rev. E*, vol. 94, p. 052125, Nov 2016. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevE.94.052125>
- [94] M. Lubin, O. Dowson, J. D. Garcia, J. Huchette, B. Legat, and J. P. Vielma, “Jump 1.0: Recent improvements to a modeling language for mathematical optimization,” *Mathematical Programming Computation*, 2023.
- [95] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006. [Online]. Available: <https://doi.org/10.1007/s10107-004-0559-y>
- [96] V. K. Dixit and C. Rackauckas, “Optimization.jl: A unified optimization package,” Mar. 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.7738525>
- [97] R. Fletcher, “A new approach to variable metric algorithms,” *The Computer Journal*, vol. 13, no. 3, pp. 317–322, 01 1970. [Online]. Available: <https://doi.org/10.1093/comjnl/13.3.317>
- [98] S. L. Brunton and J. N. Kutz, *7 Data-driven methods for reduced-order modeling*. Berlin, Boston: De Gruyter, 2021, pp. 307–344. [Online]. Available: <https://doi.org/10.1515/9783110671490-007>
- [99] D. J. Gardner, D. R. Reynolds, C. S. Woodward, and C. J. Balos, “Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers,” *ACM Transactions on Mathematical Software (TOMS)*, 2022.
- [100] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, “SUNDIALS: Suite of nonlinear and differential/algebraic

- equation solvers,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 363–396, 2005.
- [101] C. Rackauckas, M. Innes, Y. Ma, J. Bettencourt, L. White, and V. Dixit, “Diffeqflux.jl - a julia library for neural differential equations,” 2019.
 - [102] C. Rackauckas, “A Comparison Between Differential Equation Solver Suites In Matlab, R, Julia, Python, C, Mathematica, Maple, and Fortran,” *The Winnower*.
 - [103] W. Ahrens, K. Krickeberg, and I. Pigeot, *An Introduction to Epidemiology*. New York, NY: Springer New York, 2014, pp. 3–41. [Online]. Available: https://doi.org/10.1007/978-0-387-09834-0_42
 - [104] M. Frérot, A. Lefebvre, S. Aho, P. Callier, K. Astruc, and L. S. Aho Glélé, “What is epidemiology? changing definitions of epidemiology 1978-2017,” *PLoS One*, vol. 13, no. 12, p. e0208442, Dec. 2018.

A Altri Approcci

A.1 Modello ad Agente su Spazio Continuo

L'approccio iniziale prevedeva la modellazione di una popolazione attraverso l'utilizzo di uno spazio di modello continuo. Gli agenti sarebbero stati rappresentati come individui reali, con un'effettiva presenza all'interno dello spazio. Questo spazio di modello continuo poteva essere visualizzato come una griglia di dimensioni $N \times M$ in cui gli agenti venivano posizionati in modo casuale e rappresentati come punti colorati.

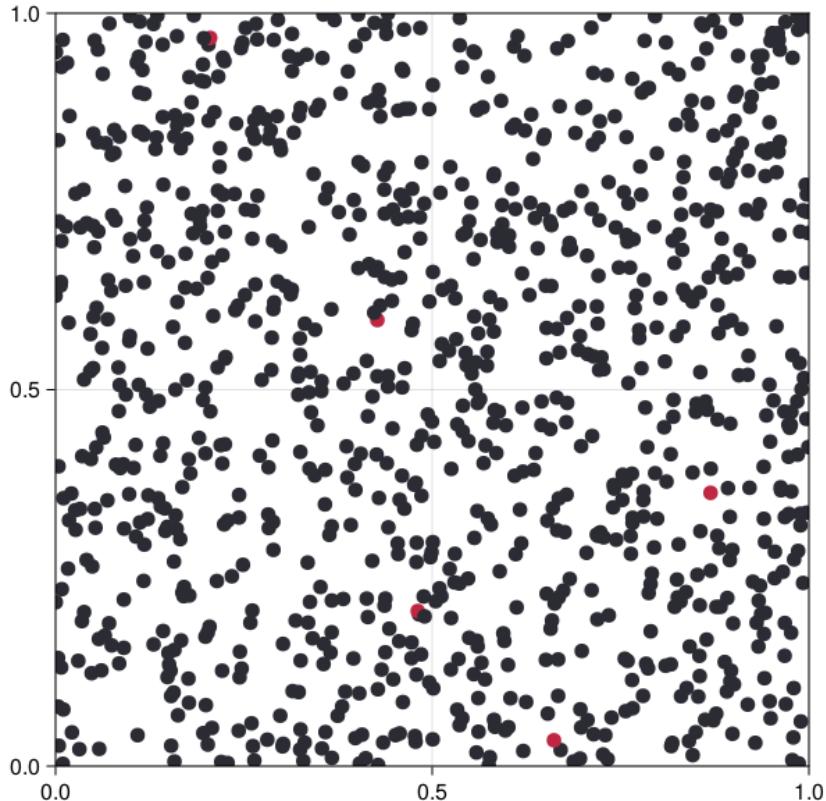


Figure 50: Esempio del modello modellato su spazio continuo

Questo approccio prendeva spunto dalla fisica delle palline da biliardo per modellare le interazioni tra gli agenti all'interno dello spazio del modello. Ogni agente poteva casualmente infettare un vicino solo se interagivano, e questa interazione veniva modellata come un urto elastico tra corpi, anche se non rappresentava un vero simulatore di urti elastici, bensì ne prendeva spunto.

Tuttavia, questo approccio è risultato eccessivamente granulare e poco adatto agli obiettivi del progetto. Inoltre, richiedeva una quantità considerevole di risorse computazionali.

ali e tempo, il che ha portato alla sua sostituzione con un approccio più astratto e flessibile.

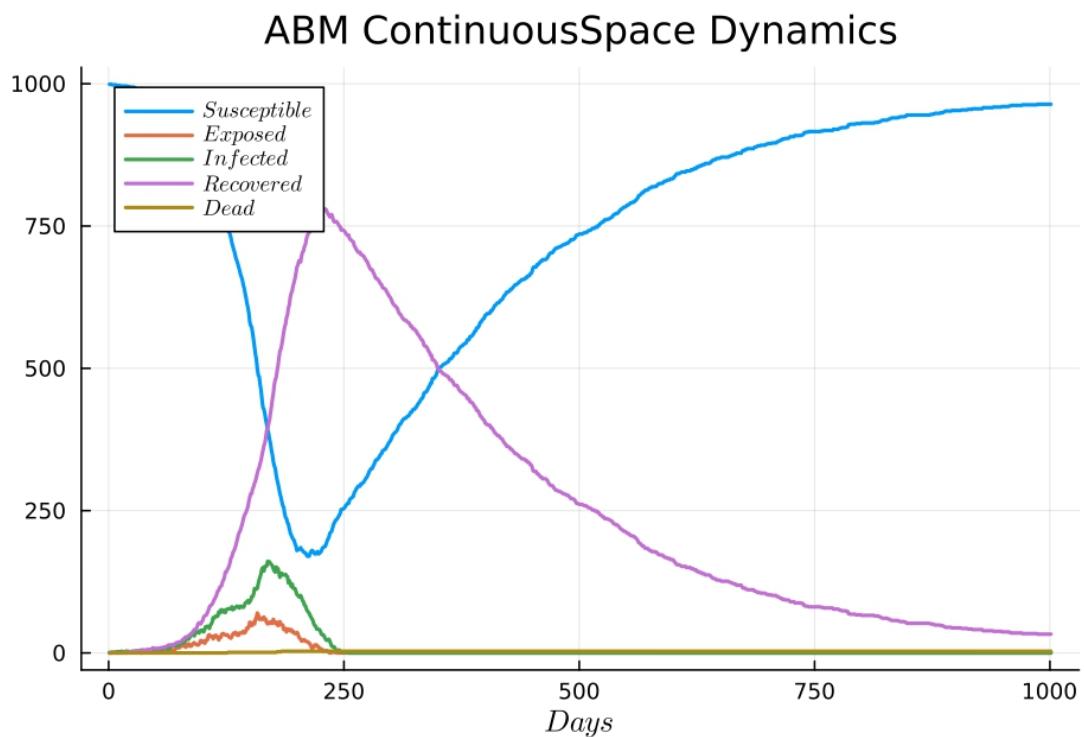


Figure 51: Esempio del comportamento delle curve nel modello continuo

A.2 Modello ad Agente con Spazio a Grafo e Modellazione Singolo Agente

Un secondo approccio è stato progettato per consentire un maggiore controllo sullo spazio di modello e la sua evoluzione locale, piuttosto che sugli agenti individuali. Tuttavia, questo approccio ha incontrato problemi significativi, tra cui tempi di esecuzione estremamente lunghi e un comportamento delle curve epidemiologiche completamente divergente rispetto al modello SEIR deterministico. Questo comportamento anomalo è stato osservato in presenza di variazioni nel parametro R_0 , e la causa rimane sconosciuta.

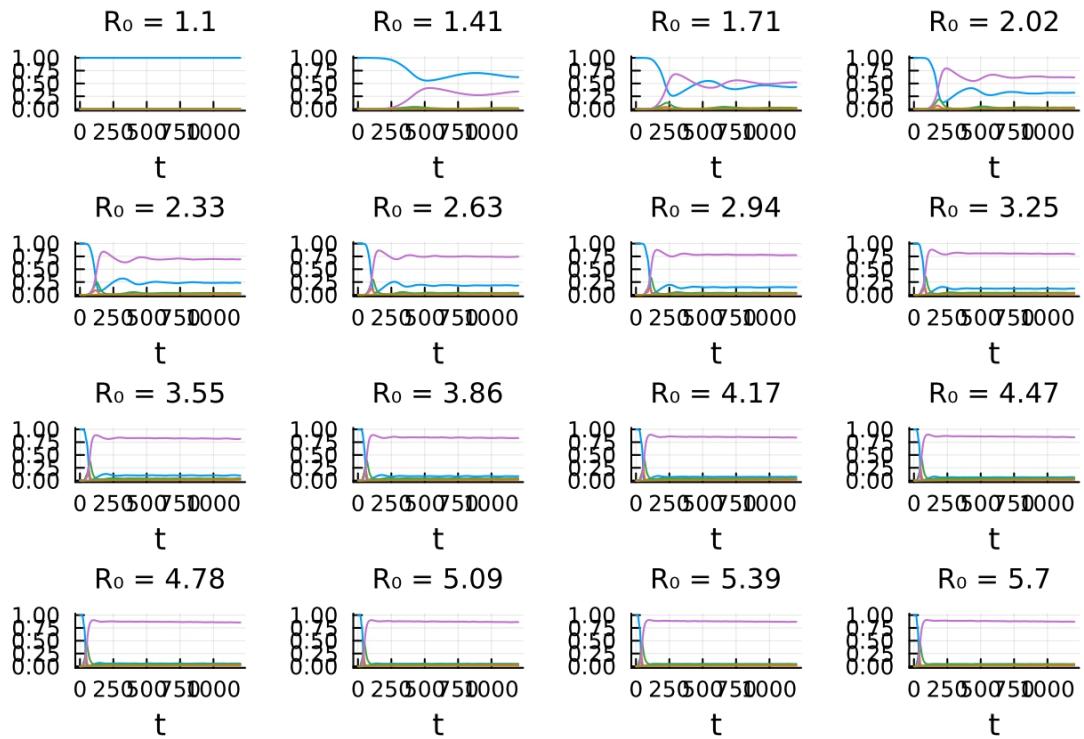


Figure 52: Comportamento modello ABM su spazio a grafo al variare del parametro R_0

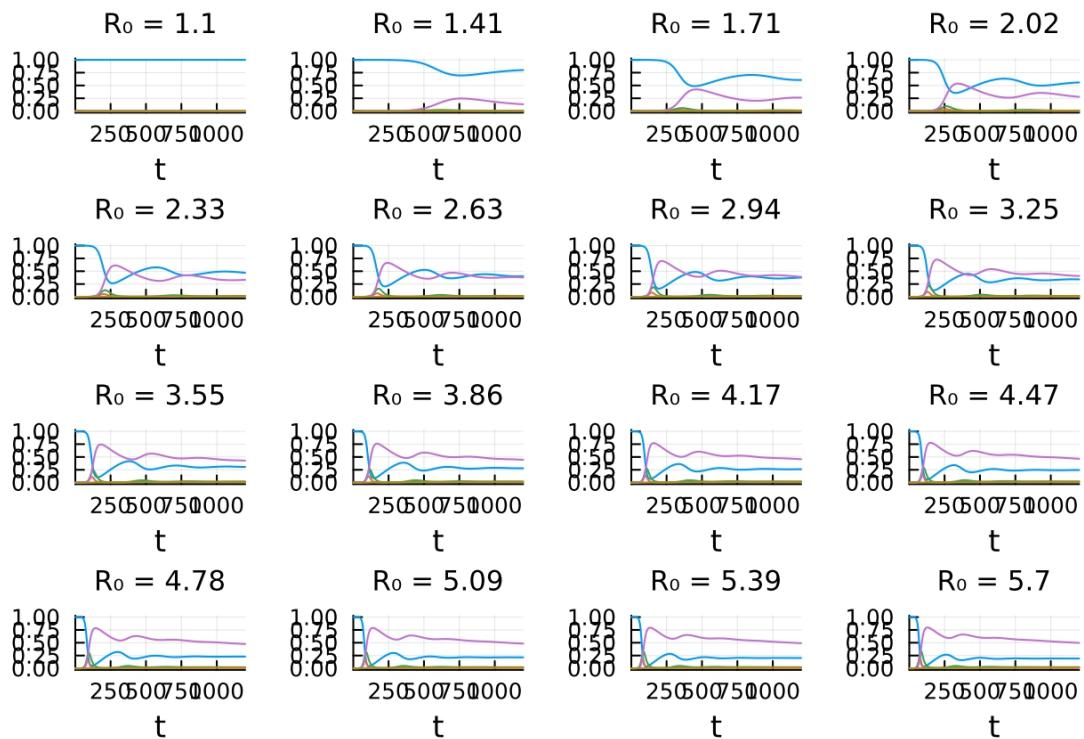


Figure 53: Comportamento modello SEIR al variare del parametro R_0

Anche se è stata formulata una formula approssimativa per descrivere la relazione tra i risultati del modello ABM e SEIR, questo comportamento rimane inspiegabile.

$$ADAPTEDR_0 = 1.1730158534328545 + 0.21570538523224972 * x$$

L'uso di un approccio basato su un grafo e la modellazione del singolo agente ha portato a risultati incoerenti e ha reso necessario abbandonare questa metodologia a favore di alternative più promettenti.

A.3 Controllore Ipopt

È stato esplorato un terzo approccio basato su Ipopt (Interior Point OPTimizer), un pacchetto software per l'ottimizzazione non lineare su larga scala. Ipopt è progettato per trovare soluzioni locali a problemi di ottimizzazione matematica, con funzioni obiettivo e vincoli non lineari. [95]

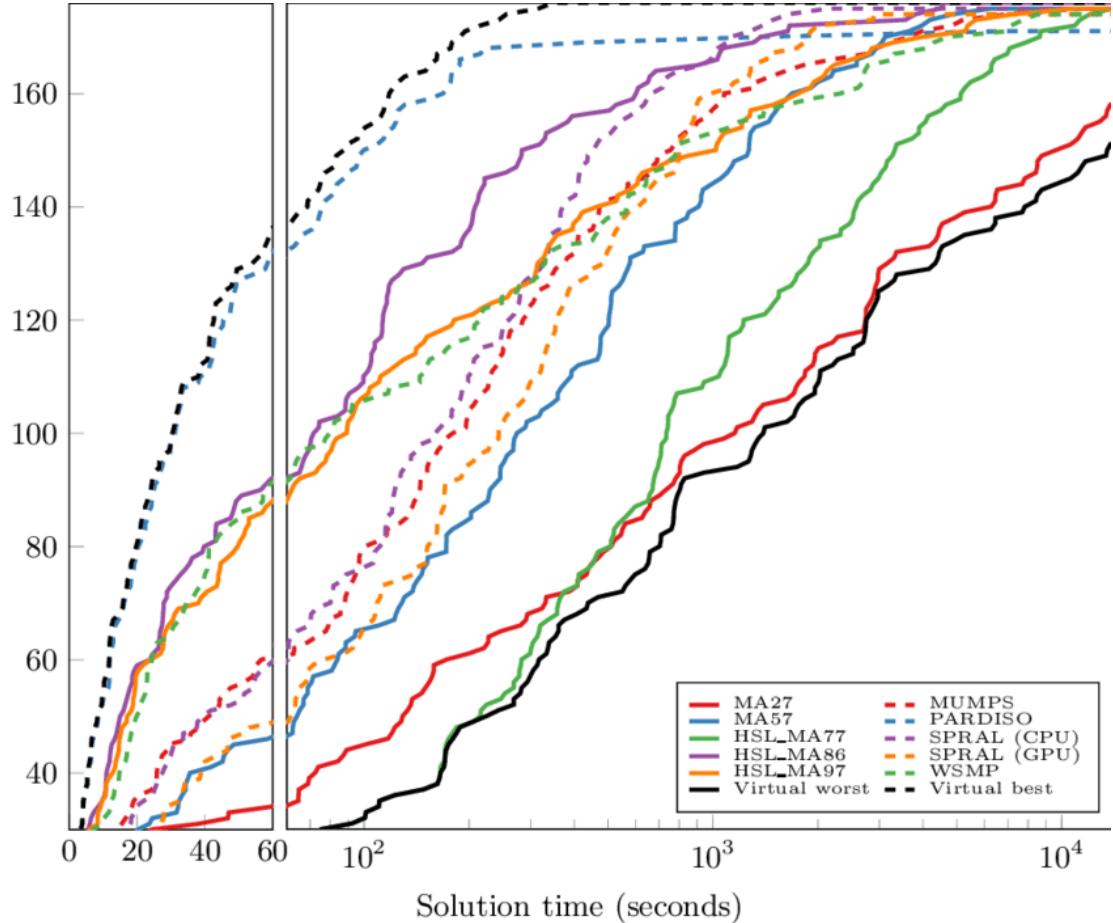


Figure 54: Comparison of Ipopt performance over various linear solvers using the two-dimensional partial differential equation test problem set. [7]

L'approccio ha previsto l'applicazione di un sistema di monitoraggio e intervento all'interno del modello di simulazione, con l'obiettivo di ridurre il numero di individui infetti. Le regole per il comportamento del modello sono state definite, inclusa una rappresentazione degli stati SEIR e un nuovo stato per il controllo delle risorse.

```

● ● ●

1 extra_ts = collect(δt:δt:timeframe[2]-δt)
2 model = InfiniteModel(Ipopt.Optimizer)
3 set_optimizer_attribute(model, "print_level", 0)
4
5 @infinite_parameter(model, t ∈ [timeframe[1], timeframe[2]], num_supports = length(extra_ts) + 2,
6   derivative_method = OrthogonalCollocation(2))
7 add_supports(t, extra_ts)
8
9 @variable(model, S ≥ 0, Infinite(t))
10 @variable(model, E ≥ 0, Infinite(t))
11 @variable(model, I ≥ 0, Infinite(t))
12 @variable(model, R ≥ 0, Infinite(t))
13 @variable(model, D ≥ 0, Infinite(t))
14 @variable(model, C ≥ 0, Infinite(t))
15
16 @variable(model, 0 ≤ u ≤ u_max, Infinite(t), start = 0.0)
17 @constraint(model, u_total_constr, ∫(u, t) ≤ u_total)
18
19 @objective(model, Min, C(timeframe[2]))
20
21 @constraint(model, S(0) == initial_condition[1])
22 @constraint(model, E(0) == initial_condition[2])
23 @constraint(model, I(0) == initial_condition[3])
24 @constraint(model, R(0) == initial_condition[4])
25 @constraint(model, D(0) == initial_condition[5])
26 @constraint(model, C(0) == C₀)
27
28 @constraint(model, S_constr, ∂(S, t) == -(1 - u) * parameters[1] * parameters[2] * S * I + parameters[4] * R - parameters[7] * S)
29 @constraint(model, E_constr, ∂(E, t) == (1 - u) * parameters[1] * parameters[2] * S * I - parameters[3] * E)
30 @constraint(model, I_constr, ∂(I, t) == parameters[3] * E - parameters[2] * I - parameters[5] * I)
31 @constraint(model, R_constr, ∂(R, t) == (1 - parameters[5]) * parameters[2] * I - parameters[4] * R + parameters[7] * S)
32 @constraint(model, D_constr, ∂(D, t) == parameters[5] * parameters[2] * I)
33 @constraint(model, C_constr, ∂(C, t) == parameters[3] * E)
34
35 optimize!(model)
36 @info termination_status(model)
37
38 S_opt = value(S, ndarray=true)
39 E_opt = value(E, ndarray=true)
40 I_opt = value(I, ndarray=true)
41 R_opt = value(R, ndarray=true)
42 D_opt = value(D, ndarray=true)
43 C_opt = value(C, ndarray=true)
44 u_opt = value(u, ndarray=true)
45 obj_opt = objective_value(model)
46 ts = value(t)
47
48 # ritorno il vettore di quando applicare le contromisure
49 ut = unique(map((x) -> trunc(Int, x), findall(x -> x > 1e-3, u_opt) * 0.1))
50 filter!(e -> e ≠ 0, ut)
51 # ritorno il vettore del valore delle contromisure utilizzate durante il periodo specifico
52 u_opt_t = u_opt[trunc(Int, ut[1] / δt):trunc(Int, 1 / δt):trunc(Int, ut[end] / δt)]
53 mean(u_opt_t)

```

Figure 55: Definizione del controllore tramite Ipopt



```

1  @constraint(model, S(0) == initial_condition[1])
2  @constraint(model, E(0) == initial_condition[2])
3  @constraint(model, I(0) == initial_condition[3])
4  @constraint(model, R(0) == initial_condition[4])
5  @constraint(model, D(0) == initial_condition[5])
6  @constraint(model, C(0) == C_0)
7
8  @constraint(model, S_constr, ̈(S, t) == -(1 - u) * parameters[1] * parameters[2] * S * I + parameters[4] * R - parameters[7] * S)
9  @constraint(model, E_constr, ̈(E, t) == (1 - u) * parameters[1] * parameters[2] * S * I - parameters[3] * E)
10 @constraint(model, I_constr, ̈(I, t) == parameters[3] * E - parameters[2] * I - parameters[5] * I)
11 @constraint(model, R_constr, ̈(R, t) == (1 - parameters[5]) * parameters[2] * I - parameters[4] * R + parameters[7] * S)
12 @constraint(model, D_constr, ̈(D, t) == parameters[5] * parameters[2] * I)
13 @constraint(model, C_constr, ̈(C, t) == parameters[3] * E)

```

Figure 56: Definizione regole del modello del controller



```

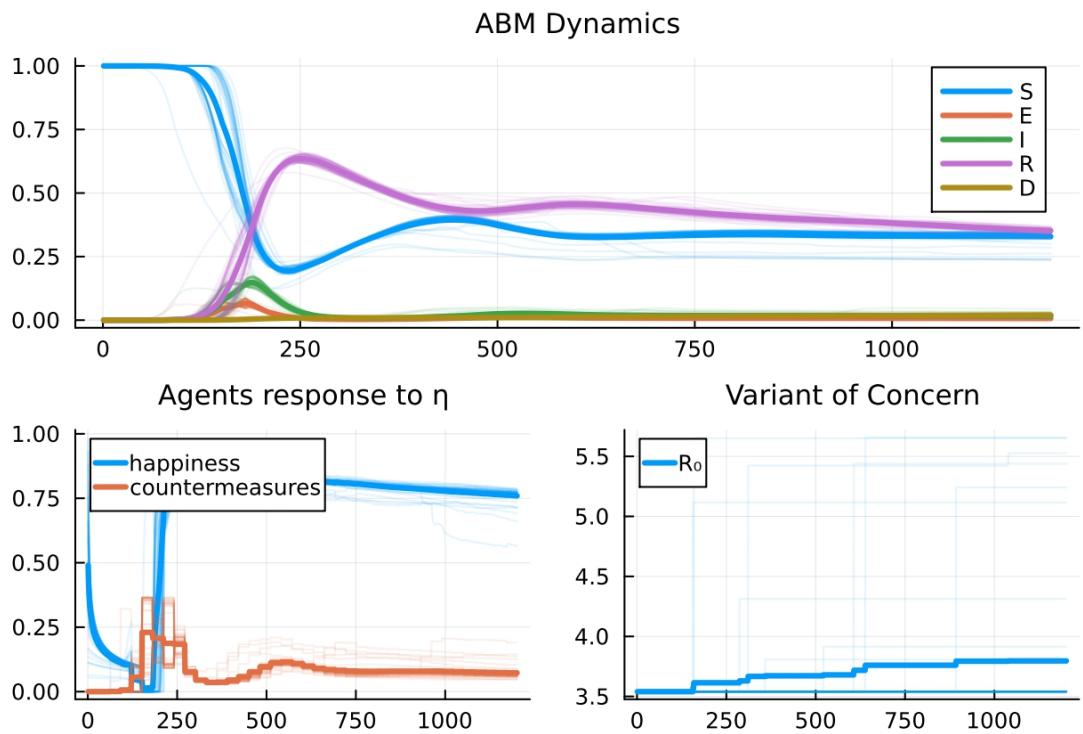
1  @variable(model, 0 ≤ u ≤ u_max, Infinite(t), start = 0.0)
2  @constraint(model, u_total_constr, ∫(u, t) ≤ u_total)

```

Figure 57: Definizione regole del modello del controller per le contromisure

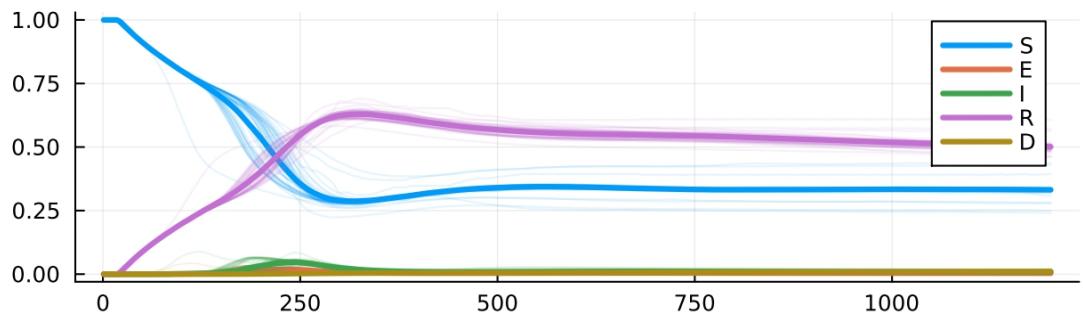
Sono state stabilite anche regole per il consumo di risorse, rappresentando le contromisure con il loro costo associato. L'ottimizzazione del modello è stata eseguita e sono stati restituiti i valori medi delle contromisure applicate.

I risultati ottenuti con Ipopt sono stati confrontati con quelli dell'implementazione personalizzata, rivelando somiglianze significative. Tuttavia, la possibilità di convertire facilmente il codice da CPU a GPU è stata determinante nella decisione di utilizzare l'implementazione personalizzata, in previsione di futuri miglioramenti delle prestazioni del codice.

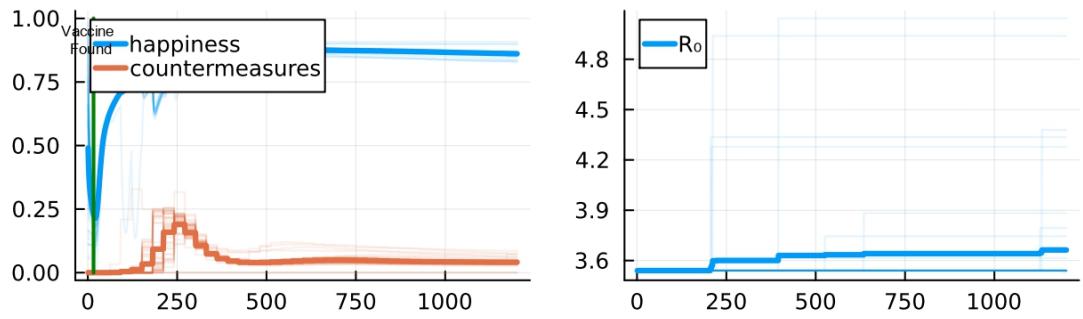


(a) Risultato applicazione controllore tramite la suite Ipopt

ABM Dynamics



Agents response to η



(b) Risultato applicazione controllore tramite la suite Ipopt

L'approccio ibrido del controllore è stato preferito per la sua semplicità e la possibilità di mascherare la funzione obiettivo tramite una rete neurale, semplificando notevolmente la definizione delle regole rispetto all'implementazione esplicita di Ipopt.