Esame di Programmazione per la Fisica: toy-model epidemiologico al primo ordine di approssimazione

`https://github.com/PizzaGitHub/Plague.git`

Janko Petkovic - janko.petkovic@studio.unibo.it

Agosto 2020

Indice

1 Introduzione: il modello SIR

La descrizione analitica di un fenomeno infettivo rappresenta da sempre un problema di non banale risoluzione.
Gli approcci matematici che possono essere tentati sono vari e vengono generalmente distinti in due categorie principali: i modelli deterministici, operanti nel limite dei grandi numeri, studiano gli andamenti asintotici delle grandezze epidemiologiche fornendo predizioni in termini di valori medi; i modelli stocastici, al contrario, tengono conto della casualità dei singoli elementi in gioco e vengono principalmente usati per fornire simulazioni del fenomeno studiato con cui testare la bontà dei parametri di malattia ipotizzati.
Per comprendere meglio il funzionamento di questi approcci, consideriamone ora uno fondamentale: il modello SIR (*Susceptible Infected Recoverd*). Rinconducile a una forma particolarmente semplice del modello deterministico Kermack-McKendrick, esso individua nell'epidemia tre gruppi distinti di individui (i compartimenti, appunto) ognuno con precise caratteristiche epidemiologiche:

- **Suscettibili:** rappresentano il bacino di infezione della malattia potendo essere convertiti in infetti a seguito di un incontro con un altro infetto
- **Infettivi:** contagiati che possono convertire i suscettibili infettandoli
- **Rimossi:** elementi non piu' infettivi ne' suscettibili, possono essere rappresentati in prima approsimazione i deceduti e i guariti[1].

L'evoluzione temporale dei tre compartimenti può quindi essere descritto dal seguente sistema di EDO:

$$\begin{cases} \dfrac{dS}{dt} &= -\dfrac{\beta IS}{N}, \\ \dfrac{dI}{dt} &= \dfrac{\beta IS}{N} - \gamma I, \\ \dfrac{dR}{dt} &= \gamma I \end{cases}$$

dove β è il prodotto fra la probabilità di infezione a seguito di contatto e il numero medio di contatti che un infetto ha nell'arco di un giorno e γ è il tasso di conversione giornaliero degli infetti in rimossi.
In questo scenario un gruppo di elementi suscettibili viene gradualmente contagiato da un agente infettivo diventando immediatamente a propria volta fonte di contagio (termine $\beta IS/N$); questo processo viene però bilanciato da un contemporaneo flusso di guariti (o deceduti) che si spostano dal compartimento I al compartimento R (termine γI).
A seguito di qualche manipolazione è possibile trovare le risolventi del sistema in forma implicita

$$\begin{cases} S(t) + I(t) + R(t) = N \\ S(t) = S(0)\ \exp\left\{-R_0\ \dfrac{R(t) - R(0)}{N}\right\} \end{cases}$$

in cui $R_0 = \beta/\gamma$ è il "numero di riproduzione di base", ovvero il numero di elementi infettati in un giorno da un singolo elemento infettante nel limite $S \to \infty$.

[1] Questo potrebbe non essere sempre vero. Ad esempio, l'immunita' ottenuta a seguito dell'infezione potrebbe non essere permanente rendendo possibile una reinfezione dei guariti dopo un qualche periodo di tempo (modello SIS). Oppure potrebbe non essere possibile una gestione sufficientemente sicura ed efficiente dei corpi dei deceduti che quindi continuerebbero ancora a veicolare la malattia provocando nuove infezioni (scenario riscontrabile durante le plurine epidemie di peste in Europa).
Come nota a margine, e' interessante considerare il caso della *Ophiocordyceps unilateralis*, un fungo parassita delle formiche appartenenti alla tribu' *Camponotini*, la cui sporulazione e diffusione può avere luogo solo a seguito della morte della malcapitata portatrice il cui cadavere rappresenta quindi la vera e propria fonte di infezione.

Definendo $s(t) = S(t)/N$ è possibile ottenere il risultato

$$s_\infty = 1 - r_\infty = -\frac{1}{R_0}\,\mathcal{W}\left\{-s_0\,R_0\,e^{-R_0(1-r_0)}\right\}$$

dove $\mathcal{W}$ è la funzione W di Lambert[2]. Si può notare così come l'evoluzione naturale dell'epidemia non porti mai al contagio di tutta la popolazione suscettibile ($s_\infty = 0 \Leftrightarrow\ S(0) = 0$) ma si arresti prima a seguito della "barriera statistica" rappresentata dai rimossi al tempo 0.
E' infine molto interessante riscrivere la (2) nella forma

$$\frac{dI}{dt} = \left(\frac{R_0 S}{N} - 1\right)\gamma I$$

che permette di notare come lo scoppio di un'epidemia può avvenire solo nel caso in cui $R_0 S > N$, essendo altrimenti la variazione degli infetti nel tempo negativa. Possiamo in questo modo calcolare agilmente la frazione di vaccinati (chiamiamola $p = 1 - s$) necessaria a garantire l'immunità di gregge

$$R_0(1-p) = 1 \qquad \Longrightarrow \qquad p = 1 - \frac{1}{R_0}$$

[2]sia $z \in \mathbb{C}$ allora la funzione di Lambert $\mathcal{W}:\ \mathbb{C} \longrightarrow \mathbb{C}$ è definita dall'equazione

$$z = \mathcal{W}\,e^{\mathcal{W}}$$

2 Costruzione del modello

2.1 Considerazioni preliminari

Il modello SIR classico garantisce la possibilità di un calcolo esplicito di alcune importanti grandezze epidemiologiche (infettività, letalità, ecc..) nonchè una agevole valutazione di parametri fondamentali per l'impostazione di manovre di prevenzione o terapia.
Presenta però alcuni aspetti che possono risultare problematici:

- il formalismo differenziale è legittimato da un passaggio al limite per $T \to \infty$ con T indicante il numero totale di giorni studiati. Questa scelta può essere appropriata nel caso in cui siano studiati lunghi periodi di tempo (qualche mese, ad esempio) ma può risultare scorretta nel caso in cui si effettuino previsioni di minore durata. I vantaggi che presenta da un punto di vista analitico, inoltre, vengono persi nel contesto di una simulazione su macchina per le originali equazioni alle differenze sono molto più agevoli da trattare;

- nel modello SIR le grandezze studiate sono prive di sufficiente "memoria" (un valore al giorno t+1 dipende unicamente da quelli al giorno t). Questo è incompatibile con il concetto di storia naturale di malattia in cui la probabilità' di guarigione o decesso varia lungo il corso della patologia;

- questa mancanza di memoria instaura una paradossale dipendenza fra la probabilità di rimozione dei singoli contagiati e il numero di persone suscettibili nella popolazione (nel limite $S \longrightarrow \infty$ sarebbe ammesso persino un tempo di malattia infinito per un singolo paziente nel caso $dI/dt > 0$).

Con l'intento di aggirare queste criticità proviamo quindi a costruire un prototipo diverso di modello, piu' semplice per alcuni aspetti ma raffinabile a posteriori.

2.2 Ipotesi di intervento precoce

Consideriamo uno scenario in cui a seguito della comparsa di un nuovo agente infettante si sia intervenuto rapidamente per limitare i casi di contagio riuscendo a mantenere il numero complessivo dei colpiti inferiore al 5% del totale della popolazione. In questo caso, l'equazione (2) in forma discreta

$$\begin{aligned} \Delta i_{t+1} &= \beta\, s_t\, i_t - \gamma\, i_t \\ &= (\beta - \gamma) i_t - \beta i_t^2 \end{aligned}$$

presenta un termine quadratico di almeno due ordini di grandezza inferiore a quello lineare e quindi può essere trattata in modo perturbativo trascurando al primo ordine il contributo logistico di s_i (questo equivale ad assumere il limite $S \longrightarrow \infty$).
Fenomeni epidemiologici che soddisfano questo requisito non sono pochi e comprendono, in particolare in Italia, la recente epidemia di COVID-19. Utilizzando un fattore cautelativo di 10, si puo' stimare che con i suoi 250000 contagi circa, essa abbia interessato interessando il 4.2% della popolazione totale l'uso del fattore cautelativo viene particolarmente incentivato anche in seguito alle indagini sierologiche di inizio Agosto).

2.3 Storia naturale: probabilità di rimozione

Consideriamo ora un esempio molto semplice che permette di mettere in evidenza il ruolo della storia naturale della patologia.
Supponiamo che in una popolazione con $S \longrightarrow \infty$ un fenomeno infettivo si propaghi con $\beta = 0.5$; supponiamo, inoltre, che la malattia duri un massimo di due giorni con una probabilità di guarigione distribuita uniformemente lungo questo lasso di tempo, ovvero $G_0 = 0,\ G_1 = 0.5,\ G_2 = 1$.
La nuova dipendenza dello stato $t+1$ dagli stati $t-1$ e $t-2$ oltre che dallo stato t si traduce in due modifiche strutturali del termine $\gamma\, i_t$:

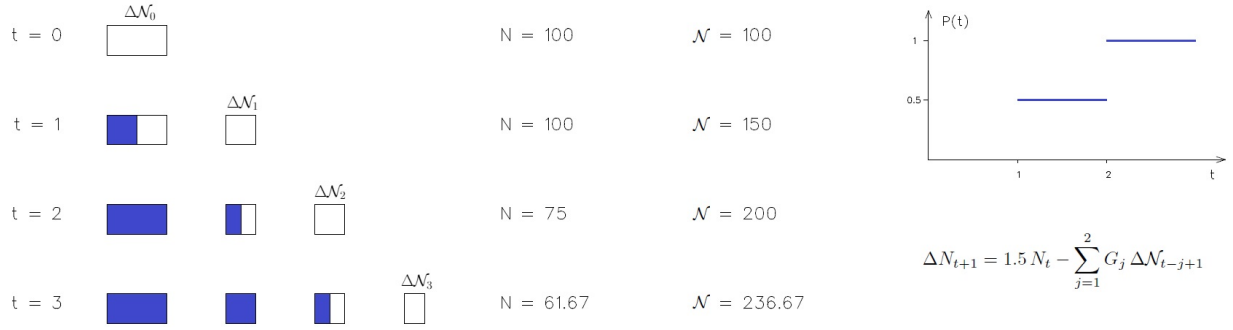

Figura 1: Esempio di fenomeno infettivo con guarigione entro la seconda giornata di malattia.

- γ deve poter assumere i valori di una distribuzione di probabilità discreta g_k ($g_k \geq 0\ \forall k,\ \sum g_k = 1$);
- non si potrà più tenere conto semplicemente di i_t ma sarà necessario considerare separatamente i contributi dati dai nuovi contagi al giorno $t - k$:

$$\gamma\, i_t \qquad \longrightarrow \qquad \sum_{j=1}^{m} g_j\, \Delta\mathcal{N}_{t-j+1}$$

dove m indica l'ultimo giorno possibile di malattia (nel nostro esempio 2).

Passando per comodità dalle frazioni ai conteggi e indicando con N_t il numero di contagiati attivi al tempo t otteniamo quindi che al primo ordine:

$$\begin{cases} \Delta N_{t+1} &= \beta\, N_t - \sum_{j=1}^{m} g_j\, \Delta\mathcal{N}_{t-j+1} \\ \Delta\mathcal{N}_{t+1} &= \beta\, N_t \end{cases}$$

una forma un po' ridondante da cui, ricordando come $N_{t+1} = N_t + \Delta N_{t+1}$, si ottiene

$$N_{t+1} = (\beta + 1)\, N_t - \beta \sum_{j=1}^{m} g_j\, N_{t-j}. \tag{1}$$

Quest'ultima equazione assume una forma particolarmente semplice ed efficientemente implementabile se si definiscono le grandezze tensoriali

$$\mathbf{N}_t = (N_t,\, N_{t-1},\, ...,\, N_{t-m})$$

$$\mathcal{M} = \begin{pmatrix} c, & -g_1, & -g_2, & \dots, & -g_m \\ 1, & 0, & \dots & \dots & 0 \\ 0, & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0, & \dots & 0, & 1, & 0 \end{pmatrix} \beta, \qquad c = 1 + \frac{1}{\beta}$$

potendo essere riscritta in forma rispettivamente aperta e chiusa come

$$\mathbf{N}_{t+1} = \mathcal{M}\, \mathbf{N}_t \tag{2}$$

$$\mathbf{N}_t = \mathcal{M}^t\, \mathbf{N}_0 \tag{3}$$

(avendo definito a priori il significato dei termini $N_{-1},\, ...\, , N_{-m-1}$).
Notiamo inoltre che la linearità dell'equazione permette di considerare $\mathcal{M}$ nelle sue due componenti di riproduzione e rimozione

$$\mathcal{R} = \begin{pmatrix} c, & 0, & \dots & \dots, & 0 \\ 1, & 0, & \dots & \dots & 0 \\ 0, & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0, & \dots & 0, & 1, & 0 \end{pmatrix}, \qquad \mathcal{G} = \begin{pmatrix} 0, & g_1, & g_2, & \dots, & g_m \\ 0, & \dots & \dots & \dots & 0 \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ 0, & \dots & \dots & \dots & 0 \end{pmatrix} \tag{4}$$

evidenziando nella (2) i due contributi infettivo e di guarigione

$$\begin{aligned} \mathbf{N}_{t+1} &= \beta\, (\mathcal{R} - \mathcal{G})\, \mathbf{N}_t \\ &= \beta\, \mathcal{R}\, \mathbf{N}_t - \Delta \mathbf{G}_{t+1} \end{aligned}$$

e mostrando esplicitamente l'incremento cumulativo degli infetti e dei rimossi all'iterazione t+1.

2.4 Storia naturale: possibili outcome

Consideriamo ora il termine di rimozione $\Delta\mathbf{G}$.
Nell'esempio precedente questo indicava il numero di elementi guariti ad ogni interazione. La guarigione, tuttavia, è solo uno dei possibili outcome o, più generalmente, solo uno dei possibili cambi di compartimento che un contagiato può effettuare. Raffinando il modello potremmo infatti decidere di includere scenari come:

- l'ospedalizzazione del paziente (con probabile aggravamento della mortalità ma riduzione del coefficiente di riproduzione);
- la quarantena (riduzione di R_0 verosimilmente senza particolari modifiche dei coefficienti di morte/guarigione);

- il decesso;

- etc.

Le probabilità di transizione in questi nuovi compartimenti saranno distribuite ognuna secondo una propria una propria funzione discreta ma dovrà sempre essere rispettato il vincolo

$$\sum_{j=0}^{t} u_j \;=\; \sum_{j=0}^{t} g_j + m_j + h_j + q_j + ...$$

$$U_t = G_t + M_t + H_t + Q_t + ...$$

essendo l'evento "uscita al tempo t" costituito dalla somma degli outcome con cui l'uscita si può manifestare (guarigione, ospedalizzazione, morte, ecc.).
E' interessante notare come deceduti e guariti non vadano a incidere sulla forma della (4) se non in un frazionamento interno di ΔG; al contrario, compartimenti come ospedalizzati o isolati necessitano dell'introduzione di nuovi termini di riproduzione e nuove distribuzioni di rimozione specifiche. La conseguente generalizzazione del modello non è però banale e nel programma esposto in seguito si sceglierà di implementare un modello con i soli compartimenti Infettivi, Guariti e Deceduti, cercando di rendere il codice quanto piu' accessibile per successive ottimizzazioni.

3 Implementazione: `Plague_simulator`

3.1 Struttura generale

Lo scheletro del questo programma è organizzato in due cartelle principali: `Plague_simulator/Core`, contenente gli algoritmi del modello e la `Plague_simulator/GUI`, la cartella pertinente all'interfaccia grafica, adottata in quanto permette un utilizzo dell'appicazione piu' agevole e (forse soprattutto) una limitazione limitazione più consistente del range di input inseribili all'utente eccessivamente entusiasta.

Coding conventions: non sono stati utilizzati particolari norme durante la stesura del codice se non alcuni accorgimenti per favorirne una più chiara comprensione alla lettura diretta:

- classi e funzioni sono identificati da nomi con iniziali maiuscole (`class MyClass; void MyFunction()`)
- attributi privati di una classe sono identificati da nomi preceduti da un underscore (`int _privateInt`)
- gli attributi di una classe sono generalmente privati e vi si può accedere mediante apposite funzioni getter e setter, dal nome facilmente intuibile;

```
<...>
private:
    int _privateInt;
public:
    int PrivateInt() const { return _privateInt; }
    void SetPrivateInt(int privateInt) { _privateInt = privateInt; }
<...>
```

Librerie Oltre a vari elementi della STL nel programma vengono utilizzati due pacchetti aggiuntivi:

- **`Armadillo`** (http://arma.sourceforge.net): libreria specificamente progettata e ottimizzata per il calcolo matriciale, viene implementata in (`PlagueModel::DetPredict()`, cfr. di seguito).
- **`Qt`** (https://www.qt.io): toolkit multipiattaforma adibito alla realizzazione di programmi con interfaccia grafica. Oltre a librerie include anche un proprio IDE, offrendo non solo una gestione automatica di tutto l'albero di inclusioni e del suo Makefile ma anche la possibilita' di progettare graficamente l'interfaccia generandone autonomamente in seguito il sorgente C++.

Il codice è estesamente commentato in tutte le sue parti (forse anche troppo); nelle prossime sezioni ci limiteremo, quindi, a esporne le varie componenti descrivendole concettualmente, senza soffermarci eccessivamente sui particolari implementativi per i quali si rimanda direttamente alla consultazione del sorgente.

3.2 Core/Plague.h

Due sono gli oggetti definiti in questo header con relativi metodi ed attributi:

`struct statusChange`

Usata per la costruzione della classe `PlagueModel`, raccoglie le caratteristiche definenti un determinanto processo di rimozione (guarigione o decesso, nel nostro caso). In particolare:

`int peak`	giornata di picco della distribuzione considerata
`double finalProb`	probabilità a priori di conludere la propria storia naturale con l'outcome in questione
`string distrType`	tipo di distribuzione del processo, gaussiano o uniforme. Si è scelto di codificare questa variabile come stringa (decodificandola a valle secondo un dizionario) in modo da rendere più rapidi debugging ed eventuali future implementazioni in altri progetti.

`class PlagueModel`

La classe contiene gli attributi dei parametri epidemiologici e i metodi per la simulazione.

Costruttori

`PlagueModel()`

Costruttore di default. Il modello istanziato con questo costruttore non è pronto per effettuare previsioni.

`PlagueModel(const int dOI, const statusChange death, const statusChange recov, const double beta)`

Costruttore parametrizzato. Inizializza immediatamente i parametri caratteristici del modello (numero dei giorni di malattia, tasso di riproduzione β e probabilità cumulative di guarigione e decesso). L'istanza costruita con questo metodo e pronta per effettuare previsioni.

Attributi

`int` **`_dOI`**

Days of Illness: massimo numero di giornate per cui un paziente può risultare infetto influenzando l'evoluzione dell'epidemia.

`int` **`_beta`**

Tasso di riproduzione giornaliero (Cfr. eq. Sez. 1)

`vector<double>` **`_deathCumFunc`**

Funzione di probabilità cumulativa del processo di decesso lungo un periodi di `_dOI` giorni

vector<double>	**_recovCumFunc** Funzione di probabilità cumulativa del processo di guarigione lungo un periodi di _dOI giorni

Getter

double	**Beta**() const;
int	**DOI**() const;
vector<double>	**RecovCumDistr**() const;
vector<double>	**DeathCumDistr**() const;

Setter

void	**SetBeta**(const double);
void	**DOI**(const int)
void	**RecovCumDistr**(const vector<double>);
void	**DOI**(const vector<double>);

Metodi predittivi

vector<vector<double>>	**DetPredict**(const int daysOfPred, const vector<double> Nvec) const Metodo che effettua la previsione deterministica sull'evoluzione del contagio. Accetta come input due parametri (numero di giorni da prevedere e numero di contagiati nei dOI giorni precedenti al momento considerato); l'output consiste in una matrice di double con quattro righe e daysOfPred colonne in cui: - l'elemento [0][t] contiene il numero di persone contagiatesi entro il giorno t compreso; - l'elemento [1][t] contiene il numero dei contagiati attivi al giorno t; - l'elemento [2][t] contiene il numero delle persone decedute entro il giorno t compreso; - l'elemento [3][t] contiene il totale delle persone guarite entro il giorno t compreso.
vector<vector<double>>	**DetPredict**(const int daysOfPred, const int N0) const Overload del metodo precedente, utilizzato per una previsione dal giorno 0. Si limita a costruire un vettore idoneo a partire da N0 e poi chiama a propria volta DetPreict(const int, const vector<vector<double>>)

3.3 Core/utils.cpp

File di implementazione contenente funzioni ausiliarie utilizzate dalla classe PlagueModel.

`arma::dmat` **`BuildDistribMat(vector<double> probCumVector);`**

Funzione utilizzata nel metodo `PlagueModel::DetPredict(...)`: dato un vettore contentente la funzione cumulativa discreta di un certo processo di rimozione essa ne restituisce la matrice di propagazione $\mathcal{G}$ (cfr. (3) sez. 2.2).

`arma::dmat` **`CastVecMat(const vector<double>& vec);`**

Chiamata dalla `BuildDistribMat(...)` esegue il broadcasting del vettore `vec` in una matrice quadrata delle stesse dimensioni con la prima riga uguale agli elementi di `vec` e zero in ogni altra posizione.

`double` **`CumGauss(const double& x)`**

Utilizzata nella successiva `CumProbFunc` fornisce il valore dell'integrale gaussiano da $-\infty$ a x.

`vector<double>` **`CumProbFunc(const int dOI, const int peakDay, const double finalProb, const std::string type);`**

Utilizzata nel costruttore parametrizzato `PlagueModel`, costruisce la funzione di probabilità cumulativa di un fenomeno di rimozione a partire da:

- `dOI`: massima durata di malattia in giorni;
- `peakDay`: giorno di picco della distribuzione di probabilità del fenomeno in esame;
- `finalProb`: probabilità totale del singolo paziente subire l'esito considerato;
- `type`: tipo di distribuzione, può accettare i valori "Gaussian" e "Uniform".

`int` **`MinDist(const int dOI, const int peakDay)`**

Chiamata in `CumProbFunc(...)`: dato un intervallo di tempo di `dOI` giorni e un giorno di picco `peakDay` interno a questo intervallo restituisce la minore fra le distanze (in giorni) tra il giorno di picco e i due estremi dell'intervallo.

`bool` **`Peakcheck(const int peakDay, const int dOI)`**

Utilizzata nel costruttore parametrizzato di `PlagueModel` in particolare durante i controlli preliminari di conformità degli input. Restituisce 0 se `peakDay` cade all'interno dell'intervallo `dOI` (estremi esclusi) e 1 in caso contrario.

3.4 GUI

L'interfaccia del programma è stata progettata utilizzando Qt 5.15, una libreria appositamente concepita per la realizzazione di front-end grafici fornita di un proprio IDE (QtCreator). Questo IDE, oltre a fornire un generico ambiente per la programmazione in C++ comprende anche una salvifica modalità ≪Design≫ in cui l'utente può configurare con mouse la struttura della GUI che sta progettando. Questa configurazione viene automaticamente salvata come file .ui in formato XML 1.0 e verra' poi tradotta in codice C++ dal makemake qmake durante il building del programma.

Event handling L'interazione delle componenti dell'interfaccia fra loro e con l'utente è gestita da Qt mediante un sistema a meta-oggetti. Questo si basa su tre componenti principali:

- **QObject class:** classe antenata universale, permette di definire nelle sue sottoclassi gli oggetti *slots* e *signals*;
- **Q_OBJECT:** macro istanziata all'interno della sezione privata di ogni classe, ne permette l'utilizzo del meccanismo *signal-slot*;
- **Meta-object compiler:** tool incluso in qmake, genera il codice C++ di implementazione del meccanismo *signal-slot* per ogni classe in cui è inclusa la macro Q_OBJECT[3].

E' così possibile definire per una classe `ReceiverClass` un set di metodi da eseguire (*slots*) nel caso in cui un'altra classe `SenderClass` subisca una qualche modifica (*signal*), collegando poi le due istanze utilizzate con il comando `QBoject::connect` e le macro `SIGNAL` e `SLOT`:

```
...
ReceiverClass receiver;
SenderClass sender;
...
QObject::connect(sender, SIGNAL(senderSignal(type)),
                 receiver, SLOT(receiverSlot(type)));
...
```

Si noti come segnale e slot devono avere la stessa segnatura affinchè la connessione sia valida.

Descriviamo ora brevemente le componenti dell'interfaccia del programma, rimandando alla lettura del sorgente per i dettagli implementativi.

- **GUI/qsliderfromzero.h**
  Classe ausiliaria, erede di QSlider. Utilizzata per gli input del giorno di picco delle distribuzioni, e' contraddistinta da uno specifico slot:

  `void setMaximum(int val)`

  imposta il valore massimo dello slider dato un intero in ingresso.

[3]Per maggiori informazioni sulla gestione degli eventi in Qt ed in particolare sul sistema MOC consultare:
https://doc.qt.io/qt-5/metaobjects.html - The Meta-Object System
https://doc.qt.io/qt-5/signalsandslots.html - Signals & slots

- **GUI/scaledvaluelabel.h**

Classe ausiliaria, erede di QLabel. E' dotata di due specifici slot:

`void` **setScaledValue0001(int val)**

imposta il testo dell'etichetta a un millesimo del valore di input. Viene usata come display dello slider di probabilità di decesso/guarigione essendo in Qt presenti solamente slider di valore intero ed essendo necessario come input del modello un numero compreso fra 0 e 1 di tre cifre decimali

`void` **setValueMinus1(int val)**

imposta il testo dell'etichetta al valore dato in unput meno uno; utilizzato per aggiornare il valore dell'etichetta di valore massimo degli slider del giorno di picco.

- **GUI/chartwindow.h**

Classe finestra erede di QMainWindow usata per il display dei grafici generati. Possiede un costruttore specifico:

`explicit` **ChartWindow(QWidget* parent, vector<vector<double>>&, vector<vector<double>>&, int)**

Oltre all'argomento di default per tutte le classi Qt `parent` riceve due matrici di double contenenti i dati previsti, una per ogni grafico da generare. Come ultimo argomento accetta il numero di giorni previsti.

- **GUI/window.h**

Classe erede di QMainWindow, definisce la finestra principale del programma.
Oltre gli attributi privati necessari a istanziarne una classe `PlagueModel`, a invocarne il metodo `DetPredict(...)` e ai loro setter sono presenti i seguenti elementi:

Window(QWidget* parent)

Costruttore parametrizzato dall'argomento default di widget antenato, si limita a inizializzare tutti gli attributi presenti nella classe con i valori osservabili sulla GUI all'avvio.

~Window()

Distruttore, si occupa del cancellamento dei puntatori `ui` e `_chartW`.

`ChartWindow*` `_chartW`

Puntatore a ChartWindow, necessario ad avere più grafici aperti nello stesso momento.

`int` `_predictionType`

Attributo privato, indica il tipo di predizione da effettuare in accordo con i metodi presenti in `PlagueSimulator`.

`bool` `_logging`

Attributo privato, regola la stampa su SO della matrice di dati graficati oltre che di alcuni parametri epidemiologici medi finali.

`void` **set_logging(bool)**

`void` **set_predictionType(int)**

`void` **`simulate()`**

Al click del bottone "SIMULATE" il metodo chiama la funzione di predizione corrispondente al valore di `_predictionType` (per adesso è disponibile solo l'opzione deterministica, quella stocastica non è ancora stata dotata di codice). Ottenuta la matrice di dati, genera i valori percentuali dei guariti/deceduti rispetto ai rimossi totali nel tempo, esegue la graficazione e se `_logging==1` effettua la stampa su SO.

- **`GUI/chartutils.cpp`**
  File di implementazione contenente i metodi chiamati da `Window::simulate()` per la graficazione dei dati.

3.5 Usare `Plague_simulator`

L'utilizzo dell'applicazione dovrebbe risultare abbastanza semplice grazie alla GUI su cui ogni input può essere inserito mediante l'apposito campo/slider opportunamente limitato; un unico fatto da avere presente è che l'opzione riguardante il tipo di previsione non ha ancora un metodo implementato per l'opzione "Stocastica" e non genera alcun output.
Una volta premuto il tasto "RUN" la predizione viene effettuata e due grafici vengono mostrati in una nuova finestra: un istogramma a barre segmentate (sopra) con le numerosità dei tre compartimenti studiati e un grafico a linee (sotto) con l'andamento delle percentuali di guariti e deceduti rispetto ai rimossi totali in funzione del tempo.
Nel caso in cui l'opzione "Logging della simulazione" sia spuntata, i dati grezzi usati per la generazione dei grafici vengono aggiuntivamente tabulati sul terminale (o su un `file.txt` di testo nel caso in cui il programma sia eseguito mediante `$./Plague_simulator > file.txt`) insieme ad alcuni valori epidemiologici medi della previsione (tasso di letalità e tasso di guarigione).

4 Testing, risultati e considerazioni finali

A un primo testing il programma sembra funzionare correttamente, prevendendo senza errori l'andamento dell'esempio trattato nella Sez. 2.3.
Inoltre, aggiustando i parametri in ingresso (ridotto DOI oppure picchi di rimozione molto precoci), possono anche essere riprodotti i due andamenti ottenibili con un modello SIR nelle fasi iniziali di epidemia: l'«esplosione» esponenziale (alto β omologo a un $R_0 > 1$) e l'esaurimento progressivo dei contagiati (basso β omologo a $R_0 < 1$).

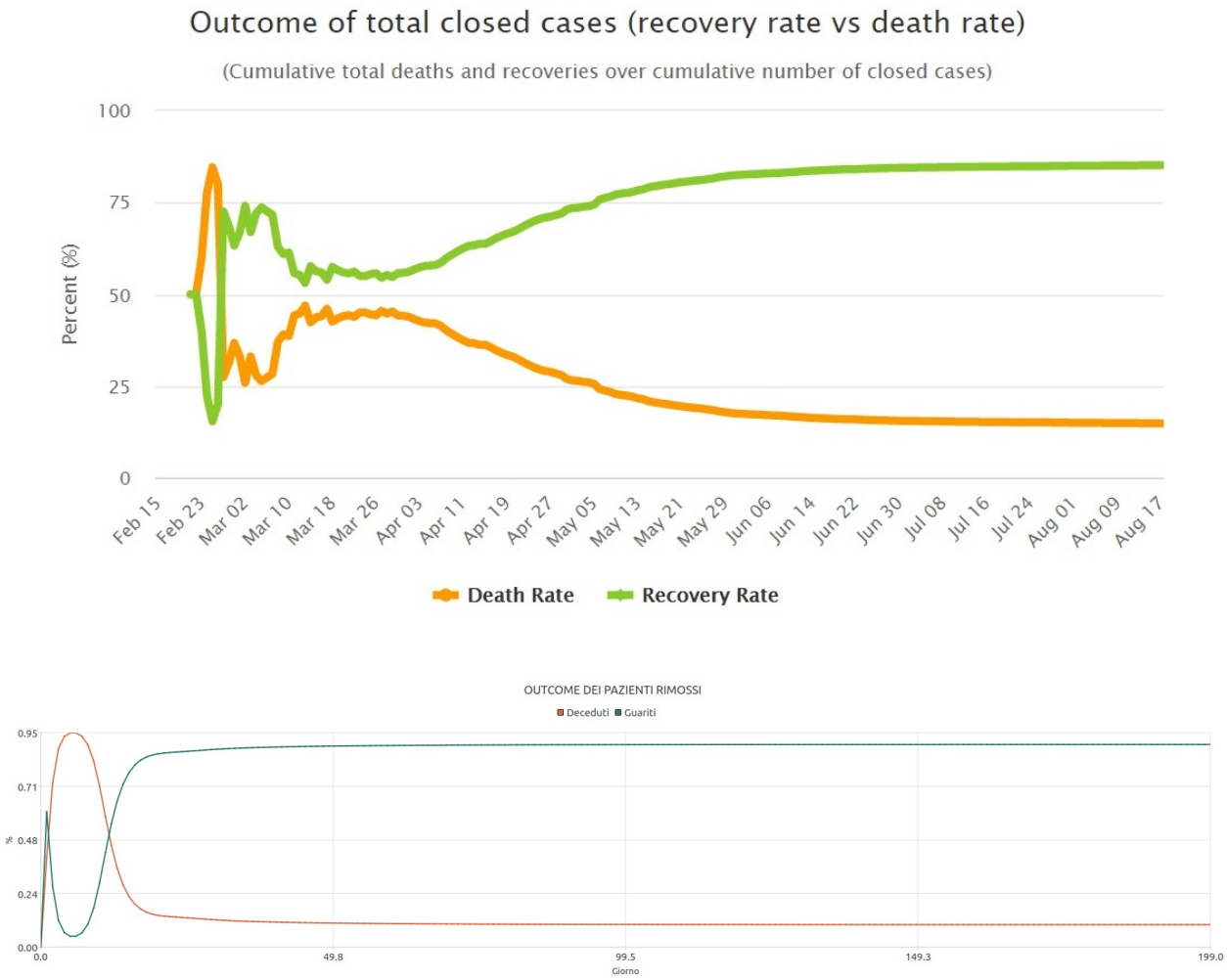

Figura 2: Percentuali di guariti (verde) e deceduti (arancione/rosso) rispetto ai rimossi totali: epidemia COVID-19 in Italia nel periodo Febbraio-Agosto (sopra) vs simulazione (sotto; DOI = 30, picco guarigione 15 giorno, picco decessi 7 giorno, probabilità guarigione 0.95, $\beta = 0.05$)

Oltre a questi, però, è possibile generare un ulteriore gruppo di soluzioni (tipicamente definite da un alto DOI, basso β, distribuzioni gaussiane di rimozinoe con picchi situati verso la metà della storia naturale), caratterizzate da un andamento cummulativo simil logistico. Questo, almeno secondo l'autore, è un fatto

particolarmente interessante.
Il calo dei nuovi contagi, infatti, non viene più ad essere giustificato solamente dall'effetto limitante del compartimento suscettibile o dalla riduzione in itinere dal tasso di contagio β. Esso diventa un preciso momento evolutivo di un nuovo tipo di fenomeno (bi o, possibilmente, multifasico) che naturalmente, senza modificazioni delle condizioni iniziali o pressione statistica da compartimenti esterni, passa da una crescita esponenziale ad un progressivo smorzamento.

Un ulteriore andamento non riproducibile da un modello SIR è quello delle percentuali di guarigione e decesso nella popolazione rimossa. Supponiamo il caso di una patologia di alto DOI con una probabilità di guarigione di 0.9, distribuzioni di rimozione gaussiane e un picco della probabilità di decesso che anticipa di qualche giorno quello di restitutio. Effettuando la simulazione si può osservare che inizialmente, nonostante la probabilità di morte sia solo dell'1%, i deceduti rappresenteranno la maggioranza dei rimossi. Questa percentuale continua a restare alta nel tempo, in alcuni casi persino aumentando per qualche giorno per poi crollare improvvisamente e portarsi al proprio valore asintotico.
Se ora si considera il grafico reale delle percentuali di outcome relative all'epidemia di COVID-19 in Italia (figura in alto, https://www.worldometers.info/coronavirus/country/italy/) si può individuare un'inversione degli esiti molto simile[4] a quella ottenibile con il simulatore.

In conclusione, pur rimanendo un toy-model dal minimo numero di compartimenti e valido solo come primo grado di approssimazione nel limite $S \longrightarrow \infty$, questo modello offre alcuni interessanti spunti apparentemente degni di un ulteriore approfondimento ed in particolare sembra mettere in rilievo che un utilizzo di equazioni di ordine superiore al primo nell'approccio formale al problema epidemiologico sia fondamentale.

5 Bibliografia

1. https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology - *Compartmental models in epidemiology*
2. https://www.worldometers.info/coronavirus/ - Worldometer: Coronavirus Update (Live)
3. Conrad Sanderson and Ryan Curtin - *Armadillo: a template-based C++ library for linear algebra* - Journal of Open Source Software, Vol. 1, pp. 26, 2016.
4. Conrad Sanderson and Ryan Curti - *A User-Friendly Hybrid Sparse Matrix Class in C++* - Lecture Notes in Computer Science (LNCS), Vol. 10931, pp. 422-430, 2018.
5. https://wiki.qt.io/Main - Qt Wiki

[4]nella loro generalità, l'andamento simulato e quello effettivo sono molto lontani fra loro. Questa differenza potrebbe essere rivalutata però se si tiene in considerazione la saturazione del sistema ospedaliero avuta luogo proprio nel mese di Marzo e che una volta migliorata riporta le percentuali di decesso e guarigione a tendere di nuovo ai valor asintotici - apparentemente 15% e 85%.

6 Appendice

6.1 Ricerca sperimentalmente dei parametri distributivi

Riscrivendo la (1) (Sez. 2.3) rendendo esplicite le distribuzioni di guarigione e decesso ed introducendo, inoltre, le equazioni cumulative per i rimossi otteniamo

$$\begin{cases} N_{t+1} &= (\beta+1)\, N_t - \beta \sum_{j=1}^{m} (\gamma_j + \mu_j)\, N_{t-j} \\ M_{t+1} &= \beta \sum_{j=1}^{m} \mu_j\, N_{t-j} \\ G_{t+1} &= \beta \sum_{j=1}^{m} \gamma_j\, N_{t-j} \end{cases}$$

un sistema sovradeterminato di equazioni lineari alle differenze valutabile numericamente nei coefficienti β, μ_j e γ_j noti i valori dei contagiati, dei guariti e dei deceduti in un intervallo di m+1 giorni.

Nonostante la semplicità del modello presentato e della estrema variabilità dei dati rilevati empiricamente durante un fenomeno epidemico si può supporre che durante una fase di relativa stazionarietà (supponiamo quella che si è passata in Italia durante il mese di Giugno) in cui le metodiche di rilevamento della malattia e il comportamento della popolazione non sono soggette a cambiamenti troppo repentini e significativi, sarebbe interessante vedere se i le soluzioni ottenute per il sistema soddisfano alcuni degli assunti supposti per i coefficienti del modello.
In particolare, sarebbe interessante analizzare la stabilita' del valore di β in confronto a quelli di R_0 ed R_t, la cui evasività potrebbe cosi non dipendere dall'inaccuratezza delle misure effettuate ma dall'impossibilita di utilizzare un modello differenziale al primo ordine.