ARIADNE: supporto e integrazione di GNUPLOT

Albanese Mirko
mirko.albanese@studenti.univr.it

Univeristà degli studi di Verona — May 2, 2021

Introduzione

In Ariadne vi è la necessità di implementare un'interfaccia che permetta di disegnare la regione raggiunta di un sistema a derivate parziali. Al suo interno vi è già l'utility Cairo che mediante il plotting png permette di disegnare la traiettoria nel tempo di un sistema ad equazioni differenziali ordinarie. Questo documento descrive l'attività svolta per implementare un supporto a Gnuplot, con lo scopo di poter disegnare sistemi a derivate parziali, ovvero la loro regione raggiunta.

Nella prima sezione viene descritto il funzionamento di Gnuplot come la sua installazione e il suo funzionamento.

Nella seconda sezione vengono illustrati gli obiettivi posti e le motivazioni che hanno spinto a costruire il supporto e viene descritto brevemente i modelli utilizzati per testare il supporto, in particolare `l'equazione dell'onda`.

Nella terza sezione viene descritto l'interfacciamento a Gnuplot all'interno dell'interfaccia di disegno preesistente in Ariadne.

Nella quarta sezione viene illustrato il risultato finale ottenuto, mostrando il file generato dall'interfaccia di disegno Gnuplot.

1 Gnuplot

Gnuplot è un'utilità grafica portable basata su riga di comando per Linux, OS/2, MS Windows, OSX, VMS e molte altre piattaforme. Il codice sorgente è protetto da copyright ma distribuito liberamente. È stato originariamente creato per consentire a scienziati e studenti di visualizzare funzioni e dati matematici in modo interattivo, ma è cresciuto fino a supportare molti usi non interattivi come lo scripting web. Viene anche utilizzato come motore di stampa da applicazioni di terze parti come Octave. Gnuplot supporta molte tipologie di grafici in 2D e 3D. Può disegnare usando linee, punti, riquadri, contorni, campi vettoriali, superfici e vari testi associati.

1.1 Installazione

Per evitare incongruenze con la libreria Ariadne si preferisce installare programmi e utility da pacchetti, perciò si consiglia fortemente di installare la versione che in questo momento, da pacchetto apt è la versione 5.2.8 oppure indirizzarsi al sito internet `http://sourceforge.net/projects/gnuplot` e scaricare la versione 5.2.8.
Per installare la versione da apt occorre digitare il seguente comando da shell:

Command Line

```
$sudo apt-get install gnuplot
```

Altrimenti una volta scaricato, estratto il sorgente e portatosi all'interno della cartella di installazione, l'installazione avviene digitando i seguenti comandi:

Command Line

```
$./configure
$make
$make check
$make install
```

Per verificare la corretta installazione basti digitare sulla shell il comando

`gnuplot`

per poter avviare l'applicazione.

Un ulteriore controllo da effettuare è verificare se gnuplot ha installato tutte le tipologie di output che ci interessano, per fare questo, avviate l'applicazione inserendo il comando

`gnuplot`

succettivamente per visualizzare le tipologie di output installate occore eseguire il comando:

`set terminal`

e tra l'elenco su schermo devono apparire la dicitura gif e png. Se ciò non dovesse accadere allora occorre installare la libreria libgd-dev mediante il comando:

`sudo apt-get install libgd-dev`

e una volta installato ripetere il procedimento di installazione di gnuplot descritto sopra.

1.2 Funzionamento

Come si è potuto facimente notare l'utilizzo di gnuplot è basato sull'esecuzione, da riga di comando, di una serie consecutiva di comandi che, fino alla costruzione del disegno finale, permettono di creare il disegno da una fonti di dati in input come un file oppure una serie di valori. Ora verranno descritti brevemente i comandi che interessano maggiormente per la comprensione del progetto.

Gnuplot permette di creare un canvas di un determinato tipo (png, gif etc) di date dimensioni mediante il comando:

`set terminal [type] size [xvalue], [yvalue]`

dove [type] può essere un elemento tra quelli definiti digitando 'set terminal' dopo aver avviato gnuplot e [xvalue] e [yvalue] sono rispettivamente le dimensioni dell'immagine.

Successivamente gnuplot ci permette di definire il nome del file di output mediante il comando:

`set output "NOME_FILE.xxx"`

Permette di disegnare un elemento sopra un altro oppure rendere il plot univoco all'elemento disegnato mediante il comando:

`set multiplot or unset multiplot`

Permette, inoltre di definire un nome e un range agli assi cartesiani mediante i comandi:

```
set xlabel 'x0'
set ylabel 'x1'
set zlabel 'x2'
set xrange [0:19]
set yrange [0:19]
set zrange [0:1]
```

Permette di definire una color palette (utilizzato nelle proiezioni 3d) mediante i seguenti comandi:

```
set cbrange [-0.5:1]
set cbtics 0.2
set palette defined
```

Permette di definire una trasparenza:

```
set style fill transparent solid #
```

Infine permette di disegnare a 2 dimensioni:

```
plot '-'
```

Oppure a 3 dimensioni:

```
splot '-'
```

Il digit '-' permette a gnuplot di sapere che dopo il seguente comando vi saranno degli input che verranno inseriti (ogni input deve essere confermato con il ritorno a capo) e terminati con la lettera 'e'.

Infine per terminare il suo utilizzo viene digitato il comando:

```
quit
```

2 Obiettivi

2.1 Motivazioni

Durante lo sviluppo del progetto si è posto il problema di poter visualizzare graficamente la regione raggiunta di un sistema a derivate parziali. Questa necessità ha permesso di progettare un supporto a Gnuplot all'interno di Ariadne per gestire, oltre alla stampa di un disegno statico, un'animazione e di conseguenza visualizzare la regione raggiunta nel tempo sulle coordinate spaziali, ma anche una visualizzazione statica, sia di una proiezione di una variabile spaziale che una traiettoria nel tempo. Per fare ciò, ho costruito una simulazione a due dimensioni (x1, t) e a tre dimensioni (x1, x2, t) dell'equazione dell'onda per poter testare il comportamento di Gnuplot.

2.2 Equazione dell'onda

In questa sotto-sezione viene descritta brevemente l'implementazione utilizzata per simulare l'equazione dell'onda a due e a tre dimensioni. In entrambe le metodologie viene utilizzato l'algoritmo delle differenze finite per risolvere numericamente le equazioni differenziali.

2.2.1 Due dimensioni: Spazio x1 - Tempo t

L'equazione dell'onda a 2 dimensioni può essere paragonata al movimento di una corda vibrante. L'equazione che modella questa situazione può essere scritta nel seguente modo (equazione dell'onda):

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{1}$$

dove, $u(x,t)$ è lo spostamento della corda.
Definiamo ora gli elementi principali per definire la simulazione:

- Definire un quadro dimensionale dell'equazione 1, ovvero: $x \in (0, L), t \in (0, T]$
- Definire una condizione iniziale sulla posizione: $u(x, 0) = I(x)$
- Definire una condizione iniziale sulla velocità: $\frac{\partial}{\partial t} u(x, 0) = 0$
- Definire una condizione iniziale al contorno: $x(0, t) = x(L, t) = 0$

Una volta definite le condizioni iniziali occorre discretizzare le variabili di spazio e di tempo definendo una mesh per variabile dove ogni punto è equidistante da quello successivo rendendo la mesh spaziale e temporale uniformi.

Tempo : $0 = t_0 < t_1 < t_2 < ... < t_{N_{t-1}} < t_{N_t} = T$

Spazio : $0 = x_0 < x_1 < x_2 < ... < x_{N_{x-1}} < x_{N_x} = L$

Ogni elemento x_i e t_i ha una spaziatura δx e δt uniforme:

Tempo : $t_i = i\Delta t.i = 0...N_t$

Spazio : $t_i = i\Delta x.i = 0...N_x$

La soluzione all'interno della mesh viene così definita ${u_x}^n = u(x_i, t_n)$.
Sostituendo la soluzione all'interno dell'equazione 1 otteniamo la seguente equazione differenziale.

$$\frac{\partial^2}{\partial t^2}u(x_i, t_n) = c^2\frac{\partial^2}{\partial x^2}u(x_i, t_n) \tag{2}$$

dove, $i = 1, ..., N_x - 1$ e $n = 1, ..., N_t - 1$. Per $n = 0$ avremo la condizione iniziale $u = I(x)$ mentre per $i = 0, N_x$ avremo le condizioni al contorno con $u = 0$.
Applicando il metodo delle differenze finite all'equazione 2 possiamo ottenere:

$$\frac{\partial^2}{\partial t^2}u(x_i, t_n) \simeq \frac{u_x^{n+1} - 2{u^n}_x + u_x^{n-1}}{\Delta t^2} \tag{3}$$

e

$$\frac{\partial^2}{\partial x^2}u(x_i, t_n) \simeq \frac{u_{x+1}^n - 2{u^n}_x + u_{x+1}^n}{\Delta x^2} \tag{4}$$

ottenendo la sequente equazione

$$\frac{u_x^{n+1} - 2{u^n}_x + u_x^{n-1}}{\Delta t^2} = c^2\frac{u_{x+1}^n - 2{u^n}_x + u_{x+1}^n}{\Delta x^2} \tag{5}$$

Da qui si deve definire la versione algebrica della condizione iniziale $u_t(x, 0) = 0$ definendo ${u_x}^0 = I(x_i)$ con ${u_x}^{n-1} = u_x^{n+1}$ per $n = 0$
Risolvendo per ${u_x}^{n+1}$ dall'equazione 5 possiamo scrivere la sequente equazione:

$${u_x}^{n+1} = -{u_x}^{n-1} + 2{u_x}^n + C^2(u_{x+1}^n - 2{u_x}^n + u_{x-1}^n) \tag{6}$$

Avendo 3 step u_x^{n-1}, u_x^n, u_x^{n+1}, occorre definire la soluzione all'istante temporale $n = 0$ perchè in questa situazione u_x^{n-1} punterà ad un valore al di fuori dalla mesh. Perciò considerando $u_x^{n-1} = u_x^{n+1}$ possiamo definire

$$u_x^1 = u_x^0 + \frac{1}{2}C^2(u_{i+1}^n - 2u_i^n + u_{i-1}^n \tag{7}$$

dove $C = c\frac{\Delta t}{\Delta x}$ viene denominato Courant Number. In questo modo è possibile trovare la soluzione all'istante di tempo successivo.
L'algoritmo è il seguente:

Algorithm 1: `2D Wave equation solver`

Input: $I(x)$, Condizione iniziale
Input: $u[i]$, Vettore dei ${u_x}^{n+1}$
Input: $u_1[i]$, Vettore dei ${u_x}^n$
Input: $u_2[i]$, Vettore dei ${u_x}^{n-1}$
Result: (u), Soluzione

Risolvere ${u_x}^0 = I(x), \forall x$;
Risolvere ${u_x}^1$;
Forzare le condizioni al contorno ;
$\forall n \rightarrow {u_x}^{n+1}$;

2.2.2 Tre dimensioni: Spazio x1 - Spazio x2 - Tempo t

L'equazione dell'onda a 3 dimensioni può essere paragonata al movimento di un telo elastico. L'equazione che modella questa situazione può essere scritta nel seguente modo, in maniera molto simile a quella a due dimensioni (equazione dell'onda):

$$\frac{\partial^2 u}{\partial t^2} = c^2(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{8}$$

La discretizzazione dell'equazione avviene allo stesso modo del paragrafo in riferimento all'equazione dell'onda a 2 dimensioni, in particolare:

$$\frac{u_{x,y}^{n+1} - 2u_{x,y}^n + u_{x,y}^{n-1}}{\Delta t^2} = C^2 \frac{u_{x+1,y}^n - 2u_{x,y}^n + u_{x-1,y}^n}{\Delta x^2} + c^2 \frac{u_{x,y+1}^n - 2u_{x,y}^n + u_{x,y-1}^n}{\Delta y^2} \tag{9}$$

Risolvendo per $u_{x,y}^{n+1}$ avremo

$$u_{x,y}^{n+1} = 2u_{x,y}^n + u_{i,j}^{n-1} + c^2\Delta t^2(\frac{u_{x+1,y}^n - 2u_{x,y}^n + u_{x-1,y}^n}{\Delta x^2} + \frac{u_{x,y+1}^n - 2u_{x,y}^n + u_{x,y-1}^n}{\Delta y^2}) \tag{10}$$

Come nel caso a 2 dimensioni vi è lo stesso problemo all'instante temporale $n = 0$ in cui $u_{i,j}^{n-1}$ punta ad un valore al di fuori dalla mesh. La stessa situazione viene applicata in questo contesto.

$$u_{x,y}^{n+1} = u_{x,y}^n + \frac{1}{2}c^2\Delta t^2(\frac{u_{x+1,y}^n - 2u_{x,y}^n + u_{x-1,y}^n}{\Delta x^2} + \frac{u_{x,y+1}^n - 2u_{x,y}^n + u_{x,y-1}^n}{\Delta y^2}) \tag{11}$$

L'algoritmo è il seguente

Algorithm 2: `3D Wave equation solver`

Input: $I(x, y)$, Condizione iniziale
Input: $u[i]$, Vettore dei ${u_x}^{n+1}$
Input: $u_1[i]$, Vettore dei ${u_x}^n$
Input: $u_2[i]$, Vettore dei ${u_x}^{n-1}$
Result: (u), Soluzione

Risolvere $u_{x,y}^0 = I(x, y), \forall x, y$;
Risolvere $u_{x,y}^1$;
Forzare le condizioni al contorno x e y;
$\forall n \rightarrow u_{x,y}^{n+1}$;

3 Implementazione

In questa sezione viene descritto il lavoro svolto per integrare al meglio il supporto di Gnuplot all'interno dell'interfaccia di disegno di Ariadne.

3.1 Gnuplot-iostream

Gnuplot permette il suo funzionamento mediante l'invocazione consecutiva di comandi, il tutto mediante un'interazione da shell. Perciò serve un intermediario tra il framework Ariadne (C++) e l'invocazione dei comandi verso l'istanza Gnuplot. Per fare ciò Daniel Stahlke (dan@stahlke.org) ha sviluppato un'interfaccia C++ verso Gnuplot, chiamata gnuplot-iostream. Questa interfaccia permette, tramite pipe, di propagare i comandi gnuplot verso lo standard output mediante l'utilizzo di File. L'interfaccia di Daniel però presentava un enorme problema per la filosofia di Ariadne, ovvero utilizza le librerie Boost per gestire i descrittori dei file e di conseguenza il buffering verso lo standard output. Alcune librerie esterne presentano un problema di compatibilità nel tempo, perciò si è deciso di eliminare Boost dall'interfaccia. L'utilizzo della libreria Boost all'interno di gnuplot-iostream veniva utilizzata solamente per definire il descrittore del file e per poterlo utilizzare verso lo standard output mediante pipe, perciò ho creato la funzionalità che sostituisce il relativo funzionamento utilizzando le classi `std::streambuf` e `std::ostream`.

3.2 Interfacciamento verso la libreria Ariadne

Dopo aver integrato l'interfaccia C++ - Gnuplot all'interno di Ariadne, era fondamentale capire come doveva essere gestito l'utilizzo di Gnuplot. L'interfaccia di disegno in Ariadne permette di disegnare all'interno di un canvas una serie di coordinate appartenenti ad una figura, in cui vi si può associare determinate proprietà come ad esempio il colore e lo spessore della linea, il riempimento, l'opacità e così via. I metodi di disegno sono inseriti in una classe virtuale chiamata `CanvasInterface` permettendo a diverse "utility" di interfacciarsi e appoggiarsi a questi metodi per poter disegnare una determinata figura. Ad oggi Ariadne utilizza Cairo come utility predefinita di un disegno statico e Gnuplot come utility aggiuntiva per disegnare sia disegni statici che disegni dinamici.

Questi aspetti sono stati fondamentali da capire perchè Gnuplot doveva utilizzare l'interfaccia preesistente e di conseguenza l'implementazione doveva essere il più generica possibile. Dal versante Gnuplot, invece, è stato fondamentale capire ogni suo punto debole e ogni suo punto di forza, andando ad analizzare ogni sua modalità di utilizzo.

Queste analisi mi hanno portato a creare una classe adibita all'utilizzo di Gnuplot chiamata `GnuplotCanvas` dove al suo interno sono stati definiti tutti i metodi provenienti da `CanvasInterface` per la creazione del disegno e inoltre i metodi proprietari di gnuplot per l'invocazione dei comandi utilizzati per definire ogni proprietà del disegno.

3.3 Estensione della classe DrawableInterface e integrazione di Tensor

Gnuplot può permettere una stampa 3D fornendo comandi adatti alla stampa a tre dimensioni, perciò vi è stata la necessità di integrare la classe `DrawableInterface` in modo che determinati oggetti possano essere identificati e disegnati come oggetti a due dimensioni oppure a tre dimensioni, dove questi ultimi possono essere proiezioni a due dimensioni. La classe `DrawableInterface` viene sostituita con due classi: `Drawable2dInterface` che definisce e disegna gli oggetti a 2 dimensioni, la classe `Drawabe2d3dInterface` che estende `Drawable2dInterface` definendo e disegnando gli oggetti a 3 dimensioni ma che permettono anche proiezioni a 2 dimensioni.

Viene quindi integrata e modificata la classe `Tensor` estendedola con la classe `Drawabe2d3dInterface` per poter definire così l'oggetto Tensore che può assumere forme a tre dimensioni ma anche proiezioni su due dimensioni.

3.4 Estensione della classe Figure

Dopo aver aggiunto la modalità di disegno in 3 dimensione, è necessario estendere la classe 3D inserendo i vari metodi relativi alla gestione di una figura a tre dimensioni e aggiungendo e estendendo la chiamata al costruttore dell'utility Gnuplot in aggiunta alla già esistente chiamata al costruttore per l'utility Cairo.

3.5 Creazione del file di dump

Ad ogni stampa di un disegno, l'interfaccia genera automaticamente un file con estensione `.gnu` con al suo interno la lista dei comandi Gnuplot che l'interfaccia ha eseguito e che ha passato alla pipe in standard output. Questo file può essere utilizzato come metodo di debug per verificare il corretto funzionamento oppure può essere importato da un software che utilizza un engine Gnuplot per poter visualizzare il disegno da un tool esterno.

4 TEST

Una volta conclusa l'implementazione e l'integrazione ho utilizzato le simulazioni dell'equazione dell'onda create per costruire dei semplici casi di test per poter testare l'interfacciamento a Gnuplot. I seguenti test sono presenti all'interno del file `test_gnuplot.cpp`, eccone i relativi risultati:

4.0.1 Generazione di un punto

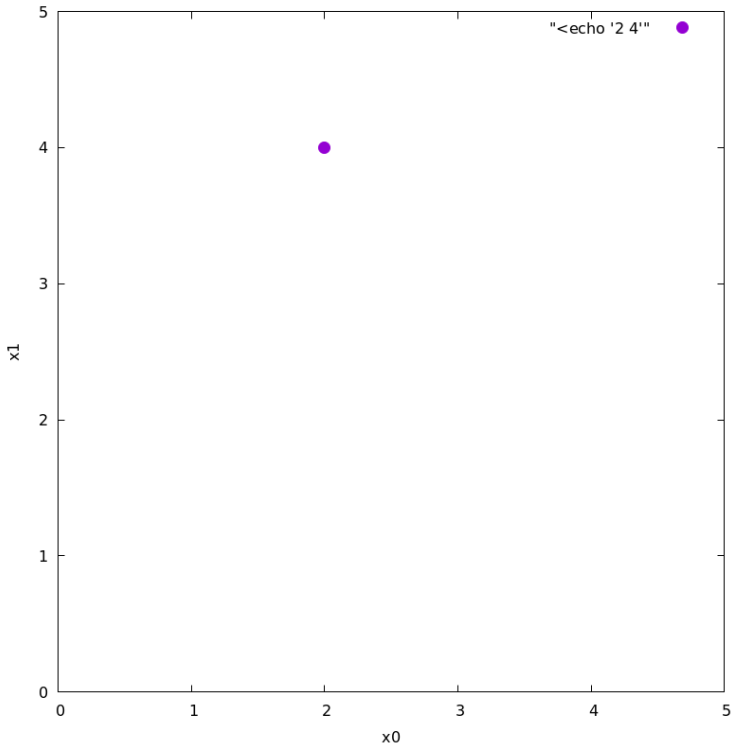


Figure 1: Stampa di un punto nello spazio bidimensionale

4.0.2 Generazione di una linea

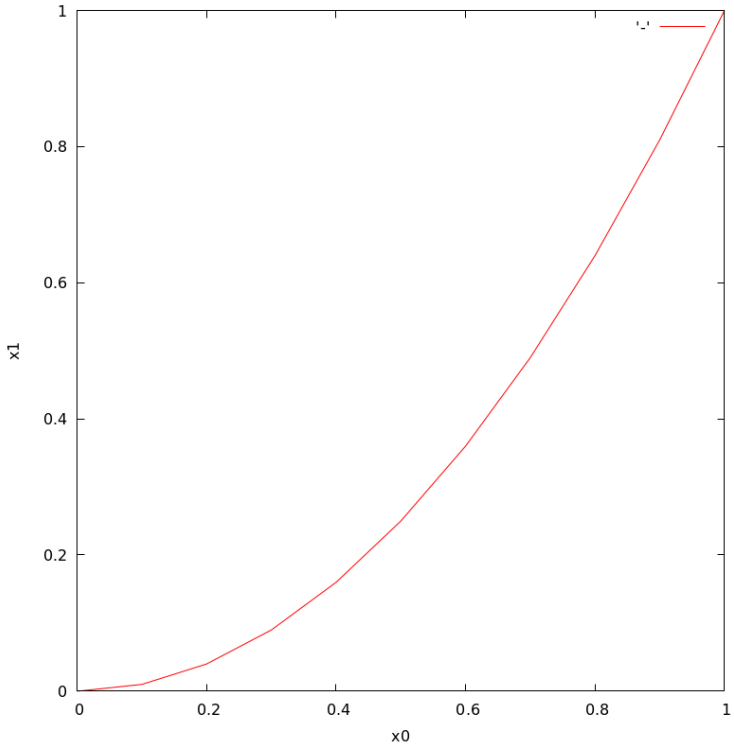
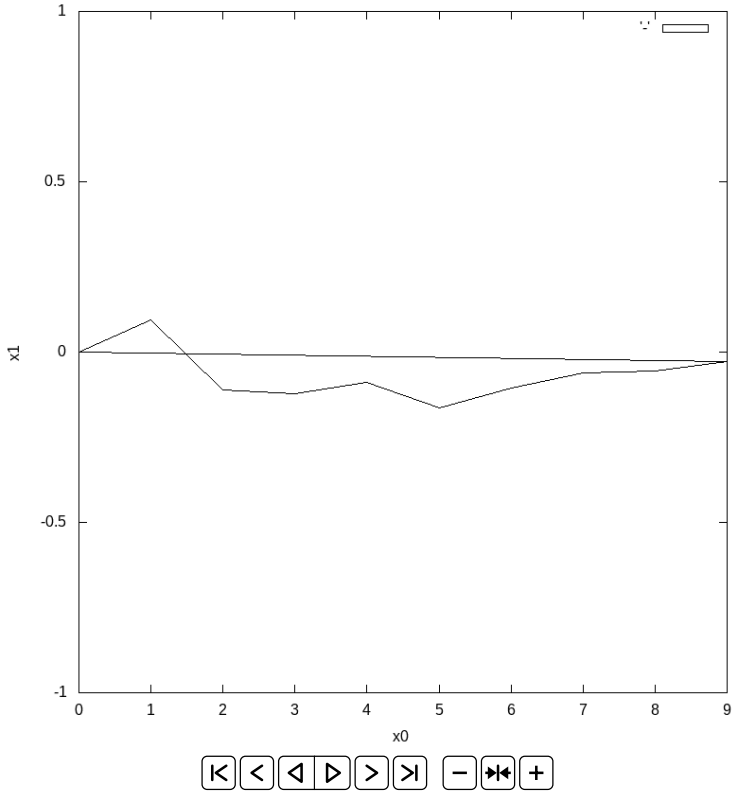


Figure 2: Stampa di una linea nello spazio bidimensionale

4.0.3 Animazione di una corda vibrante secondo l'equazione dell'onda



Animazione di una corda vibrante

4.0.4 Generazione di una gaussiana a tre dimensioni

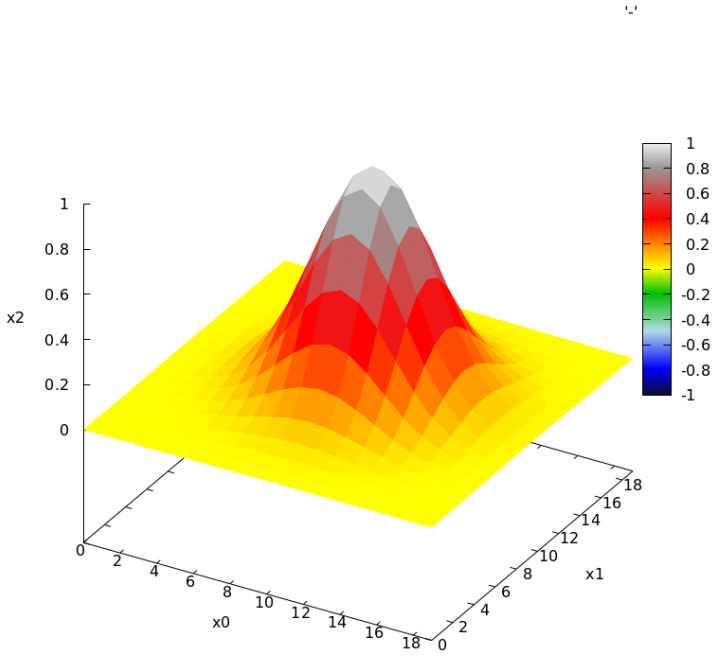


Figure 3: Stampa di una gaussiana nello spazio tridimensionale

4.0.5 Proiezione X0X1 di una gaussiana a tre dimensioni

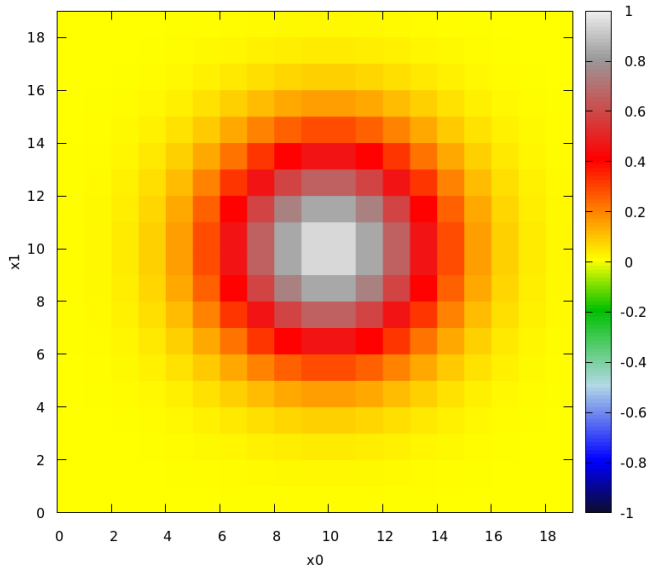


Figure 4: Stampa di una proiezione delle dimensioni x0 e x1 di una gaussiana nello spazio tridimensionale

4.0.6 Proiezione X0X2 di una gaussiana a tre dimensioni

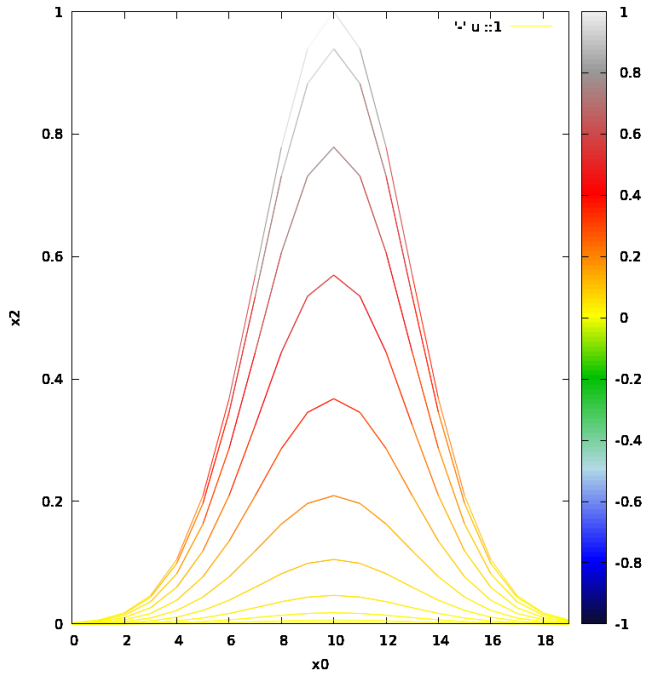


Figure 5: Stampa di una proiezione delle dimensioni x0 e x2 di una gaussiana nello spazio tridimensionale

4.0.7 Proiezione X1X2 di una gaussiana a tre dimensioni

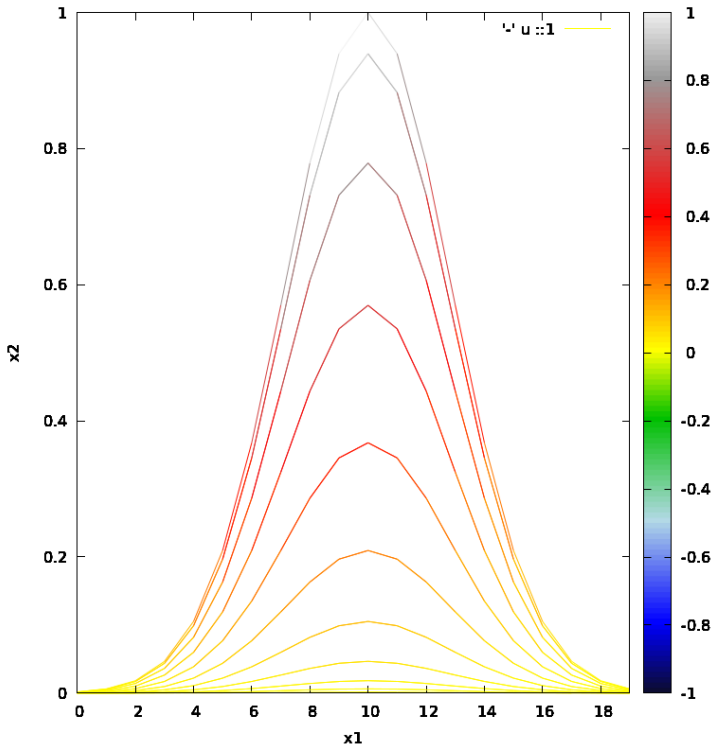
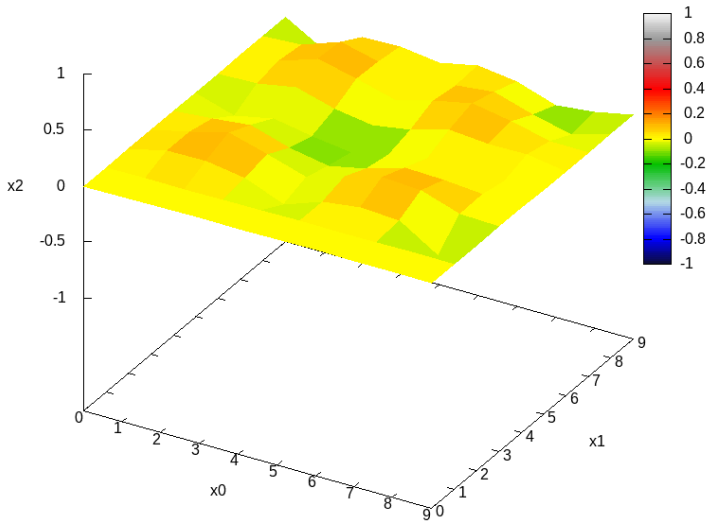


Figure 6: Stampa di una proiezione delle dimensioni x1 e x2 di una gaussiana nello spazio tridimensionale

4.0.8 Animazione di una gaussiana a tre dimensioni secondo l'equazione dell'onda



Animazione di una gaussiana

5 CONCLUSIONE E SVILUPPI FUTURI

In conclusione questo lavoro permette di avere una base, all'interno del framework C++ Ariadne per poter visualizzare la regione raggiunta di un sistema a derivate parziali andando a rappresentare sia nel tempo con disegni dinamici (animazioni) che nello spazio mediante disegni statici. Inoltre permette di poter definire una modalità di stampa definita da Gnuplot, oltre alla già esistente Cairo.

In merito ai sviluppi futuri sicuramente sarà presente il mantenimento dell'interfaccia e l'espansione della stessa in base alle esigenze dettate da Ariadne.