

Esercitazione N°2

Luca Zepponi

2 dicembre 2022

1. DESCRIZIONE DEL PROGRAMMA

Il programma si occupa sia di calcolare il prodotto di convoluzione fra due funzioni $s(t)$ e $r(t)$ sia di mandare in stampa il grafico delle tre funzioni (ciascuno opportunamente normalizzato in verticale), rispettivamente $s(t)$, $r(t)$ e $(s * r)(t)$ richiedendo all'utente di premere un tasto qualsiasi prima di procedere alla stampa del nuovo grafico.

Le funzioni $s(t)$ e $r(t)$ non vengono inserite dall'utente ad ogni esecuzione, ma sono definite all'interno del programma.

1.1. DESCRIZIONE DELLE SINGOLE FUNZIONI

Per poter far funzionare il programma occorre caricare le librerie

```
#include <iostream>    #include <cmath>,
```

che, rispettivamente, servono per per gli input e gli output e per utilizzare le funzioni matematiche predefinite.

Le costanti di istanza definite sono tre:

- `#define C 100` per definire i campioni dove la funzione è non nulla;
- `#define T 1` per l'intervallo di campionamento
- `#define ASSEY 40` per impostare la grandezza dell'asse y .

Analogamente le funzioni che useremo in seguito sono:

- `void descrizione();` per mandare in output una descrizione del programma;
- `float funcS(float t);` per definire la funzione $s(t)$;

- `float funcR(float t);` per definire la funzione $r(t)$;
- `float costruisciProdottoConvoluzionale(float tau);` per calcolare il prodotto di convoluzione;
- `float massimoFunzione(float (*funzione)(float t));` per normalizzare l'asse y ;
- `void disegnaFunzione(float(*funzione)(float x));` per disegnare il grafico della funzione.

Il `main` richiama tutte le funzioni che servono per la riuscita del programma. La prima che si incontra è `descrizione()`, la quale contiene solo un'insieme di stringhe che descrivono lo scopo del programma. In tale funzione sono presenti alcuni “unicode escape characters” per stampare a video alcuni simboli particolari. Questi sono:

- `\u2124`: insieme dei numeri interi “ \mathbb{Z} ”;
- `\u03C4`: lettera greca minuscola tau “ τ ”;
- `\u2211`: simbolo di sommatoria “ \sum ”;
- `\u221E`: simbolo di infinito “ ∞ ”;
- `\u2208`: simbolo di appartenenza insiemistica “ \in ”.

Per definire le funzioni $s(t)$ e $r(t)$ sono presenti rispettivamente le funzioni

`float funcS(float t) e float funcR(float t);.`

Le due funzioni, come da richiesta, sono definite non nulle solo su 100 campioni a partire da $t = 0$. Esse prendono in input un `datofloat`, valutano se tale input soddisfa le condizioni richieste, allora verranno valutate le funzioni e restituiranno un altro `datofloat`, se il dato non soddisfa la condizione sopra detta, la funzione varrà zero.

```

1 float funcS(float t){
2   \verb|float| risultato ;
3
4   // controllo se l'input è fuori dall'intervallo [0, C].
5   // Se sì, restituisco 0, altrimenti valuto l'espressione
6   if((t < 0) || (t >= C)) return 0;
7   risultato = t/2;
8

```

```

9 // restituisco il risultato
10 return risultato;
11 }

```

Listing 1: Funzione $s(t)$.

```

1 float funcR(float t){
2     \verb|float| risultato ;
3
4     // controllo se l'input è fuori dall'intervallo [0, C].
5     // Se sì, restituisco 0, altrimenti valuto l'espressione
6     if((t < 0) || (t >= C)) return 0;
7     risultato = t*t;
8
9     // restituisco il risultato
10    return risultato;
11 }

```

Listing 2: Funzione $r(t)$.

La funzione

```
float costruisciProdottoConvoluzionale(float tau);
```

accetta un input di tipo `float` e lo usa per calcolare il prodotto di convoluzione fra le funzioni $s(t)$ e $r(t)$, che, nell'ambito di una trattazione di segnali campionati, con funzioni nulle per $t < 0$, associabili a serie numeriche, è definito come

$$(s * r)[\tau] = C_T[\tau] = \frac{1}{T} \sum_{k=0}^{+\infty} (s[k] \cdot r[\tau - k]),$$

dove:

- $s[k]$ e $r[k]$ sono le funzione definite prima;
- τ è il valore che `costruisciProdottoConvoluzionale` prende in input;
- $\tau \in [0, 200]$;
- T è il periodo di campionamento, che è stato impostato uguale ad 1.

Dato che le funzioni $s[k]$ e $r[k]$ sono nulle dopo il `C`-esimo valore, è sufficiente che la sommatoria vada da 0 a `C`.

Dal momento che le funzioni `funcS` e `funcR` accettano in input un `float` e l'indice `indice` del ciclo `for` è stato definito come un intero, è stato effettuato un `static_cast` per convertire l'intero in `float`.

```

1 float costruisciProdottoConvolutionale(float tao){
2     \verb|float| prod = 0;
3
4     for(int indice = 0; indice <= C; indice++){
5         // static_cast<float> per passare un\verb|float| alla funzione
6         prod+=(funcS(static_cast<float>(indice))*funcR(static_cast<float>(tao-indice)));
7     }
8     prod = prod/T;
9     return prod;
10 }

```

Listing 3: Funzione per il prodotto di convoluzione.

Per normalizzare l'asse y è necessario calcolare il massimo valore assunto dalla funzione. Questo compito è svolto da

```
float maxFunc(float (*funzione)(float t)).
```

Tale funzione prende in input il puntatore a funzione della funzione da valutare.

Per calcolare il massimo si inizializza la variabile `max` che assume il valore assoluto della funzione data in input valutata in $t = 0$, ovvero il primo valore non obbligatoriamente nullo. In seguito si valuta la funzione in modulo in ogni suo punto e si confronta il valore ottenuto con il massimo. Ogni volta che la funzione valutata in modulo risulta avere un valore maggiore del massimo, la variabile `max` si aggiorna.

La funzione è stata valutata in valore assoluto per evitare la presenza di valori negativi. Per esempio, se si fosse considerata la funzione

$$f[t] = -t,$$

il suo massimo sarebbe stato

$$\max_{t \in [0, 100[} f[t] = 0,$$

mentre il suo minimo era

$$\min_{t \in [0, 100[} f[t] = 99.$$

Utilizzando il valore assoluto, invece, risulta

$$\max_{t \in [0, 100[} |f[t]| = 99.$$

In questo modo non occorre calcolare anche il minimo e confrontarli per ottenere la corretta normalizzazione.

```

1 float maxFunc(float (*funzione)(float t)){
2     \verb|float| funzValutata;
3
4     // La funzione va in modulo per non dover calcolare anche il minimo
5     \verb|float| max = abs(funzione(0));
6
7     // Cerco il massimo della funzione in modulo
8     for (int t = 1; t < C; t++) {
9         funzValutata = abs(funzione(static_cast<float>(t)));
10        if (funzValutata > max) max = funzValutata;
11    }
12
13    return max;
14 }

```

Listing 4: Funzione per il calcolo del valore massimo.

L'ultima funzione definita è

```
void disegnaFunzione(float(*funzione)(float x)),
```

la quale si occupa di disegnare il grafico della funzione data in input. Questa funzione memorizza il massimo della funzione matematica e la salva nella variabile `max` utilizzando la funzione `maxFunc`. Per disegnare la funzione, lo schermo è stato pensato come una matrice, dove nella prima colonna è presente l'asse y , la riga a metà è l'asse x . Attraverso due cicli annidati `for` ci si sposta lungo le righe (dall'alto verso il basso) e lungo le colonne (da sinistra a destra). Utilizzando una serie di condizioni, ogni elemento della matrice immaginaria viene riempito con un simbolo diverso:

- “*”: rappresenta l'output normalizzato della funzione;
- “+”: rappresenta l'intersezione degli assi;
- “-”: rappresenta l'asse x ;
- “|”: rappresenta l'asse y ;
- “ ”: se l'elemento della matrice non deve contenere nulla.

In particolare:

- se la riga y coincide con il valore della funzione opportunamente calcolato (vedi in seguito per maggiori dettagli), si stampa *;

- se la riga y coincide con la metà di `ASSEY`, si stamperà il trattino orizzontale “-” perché rappresenta l’ordinata $y = 0$;
- se la colonna x coincide con la prima, si stamperà il trattino verticale “|” perché rappresenta l’ascissa $x = 0$;
- se andrebbero stampati contemporaneamente i due trattini, si procederà a mandare in output il simbolo “+”.

Per valutare la funzione è stato necessario fare delle accortezze:

- l’indice x è un intero, mentre la funzione matematica da disegnare accetta valori di tipo `float`, quindi è stato effettuato un casting statico;
- per normalizzare l’asse y è stato necessario dividere il valore della funzione valutata in x per il massimo della stessa funzione (calcolato precedentemente), in questo modo si ottengono tutti valori compresi fra 0 e 1, e poi moltiplicare per `ASSEY/2`, per scalare correttamente i valori;
- il risultato di `funzione(static_cast<float>(x))/max*ASSEY/2` in generale è un numero `float` e non un intero, quindi la condizione

`y == round(funzione(static_cast<float>(x))/max*ASSEY/2),`

in generale, non verrà mai soddisfatta, pertanto è stato necessario arrotondare il valore;

- è stato necessario traslare il valore della funzione verso l’alto di `ASSEY/2` righe perché l’asse delle x è rappresentato sulla `ASSEY/2`-esima riga in modo da permettere la rappresentazione anche di valori negativi.

```

1 void disegnaFunzione(float (*funzione)(float x)) {
2     // Calcolo massimo funzione
3     \verb|float| max = maxFunc(funzione);
4
5     // Mi sposto sull'asse y
6     for (int y = ASSEY; y >= 0; y--)
7     {
8         // Mi sposto sull'asse x
9         for (int x = 0; x < C; x++)
10        {
11            if (y == round(funzione(static_cast<float>(x))/max*ASSEY/2) + ASSEY/2)
12            {
13                cout << '*';

```

```

14     }
15     else if ((y == ASSEY/2) && (x == 0))
16     {
17         // se si trova al centro del grafico, inserisci +
18         cout << '+';
19     }
20     else if (y == ASSEY/2)
21     {
22         // se si trova sull'asse delle x, inserisci -
23         cout << '-';
24     }
25     else if (x == 0)
26     {
27         // se si trova sull'asse delle y, inserisci |
28         cout << '|';
29     } else {
30         // Se non c'è nulla da rappresentare, inserisci spazio vuoto
31         cout << ' ';
32     }
33 }
34 // Fine y-esima riga
35 cout << '\n';
36 }
37 }

```

Listing 5: Funzione per il calcolo del valore massimo.

2. CODICE SORGENTE

```

1 // Esercitazione 2 : Convoluzione
2 // Studente: Luca Zepponi
3 //
4 // Il programma effettua il prodotto di convoluzione tra due funzioni reali
5 // limitate s(t) e r(t) campionate ad intervallo fisso T.
6 // In particolare, la funzione campionata ha valori nulli ovunque tranne in un
7 // intervallo di 100 campioni, a partire da t = 0.
8
9 #include <iostream>
10 #include <cmath>
11
12 using namespace std;

```

```

13
14 #define DPRINT(VAR) cout << #VAR << " _=" << (VAR) << endl;
15
16 // definizione costanti di istanza
17 #define C 100 // Campione
18 #define T 1 // Intervallo campionamento
19 #define ASSEY 40
20
21
22 // dichiarazione funzioni che useremo
23 void descrizione ();
24 float funcS(float t);
25 float funcR(float t);
26 float costruisciProdottoConvoluzionale(float tau);
27 float maxFunc(float (*funzione)(float t));
28 void disegnaFunzione(float (*funzione)(float x));
29
30 int main(){
31     setlocale(LC_ALL, "");
32
33     // Presentazione programma
34     descrizione ();
35
36     // Disegno funzione s
37     cout << "Il grafico della funzione s_è:" << endl;
38     disegnaFunzione(funcS);
39
40     cout << '\n';
41
42     char pausa;
43     cout << "Premi un pulsante_(e_poi_invio)_per_continuare:_";
44     cin >> pausa;
45
46     // Disegno funzione r
47     cout << "Il grafico della funzione r_è:" << endl;
48     disegnaFunzione(funcR);
49
50     cout << '\n';
51
52     cout << "Premi un pulsante_(e_poi_invio)_per_continuare:_";
53     cin >> pausa;
54
55     // Disegno funzione s*r

```



```

56     cout << "Il grafico della funzione s*r è:" << endl;
57     disegnaFunzione(costruisciProdottoConvoluzionale);
58
59     cout << '\n';
60 }
61
62 // Descrizione programma
63 void descrizione(){
64     // Elenco "unicode escape characters" utilizzati
65     // • \u2124: insieme dei numeri interi ;
66     // • \u03C4: tau (lettera greca minuscola);
67     // • \u2211: simbolo di sommatoria;
68     // • \u221E: simbolo di infinito ;
69     // • \u2208: simbolo di appartenenza insiemistica.
70
71     cout << endl;
72
73     cout << "%-----%" << endl;
74     cout << "Il programma effettua il prodotto di convoluzione tra" << endl;
75     cout << "due funzioni reali limitate s(t) e r(t) campionate ad" << endl;
76     cout << "intervallo fisso T." << endl;
77     cout << "In particolare, la funzione campionata ha valori nulli" << endl;
78     cout << "ovunque, tranne in un intervallo di 100 campioni, a" << endl;
79     cout << "partire da t=0." << endl;
80
81     cout << endl;
82
83     cout << "Convoluzione discreta." << endl;
84     cout << "Date due funzioni s[t] e r[t] definite sull'insieme dei" << endl;
85     cout << "numeri interi \u2124, il prodotto di convoluzione per" << endl;
86     cout << "segnali campionati, nulle per t<0, associate a" << endl;
87     cout << "serie numeriche, abbiamo" << endl;
88     cout << " $C_T[\tau] = (s * r)[\tau]$ " << endl;
89     cout << " $= \sum_{k=0}^{\infty} (s[k] r[\tau - k]) / T,$ " << endl;
90     cout << "in cui T è il periodo di campionamento." << endl;
91     cout << "In particolare, si può notare che" << endl;
92     cout << " $\tau \in [-100, 100]$ ." << endl;
93     cout << "Al di fuori dell'intervallo  $C_T[\tau]$  è nulla." << endl;
94
95     cout << "%-----%" << endl;
96
97     cout << endl;
98 }

```

```

99
100 // Funzione s[t]
101 float funcS(float t){
102     \verb|float| risultato ;
103
104     // controllo se l'input è fuori dall'intervallo [0, C].
105     // Se sì, restituisco 0, altrimenti valuto l'espressione
106     if((t < 0) || (t > C)) return 0;
107     risultato = t/2;
108
109     // restituisco il risultato
110     return risultato;
111 }
112
113 // Funzione r[t]
114 float funcR(float t){
115     \verb|float| risultato ;
116
117     // controllo se l'input è fuori dall'intervallo [0, C].
118     // Se sì, restituisco 0, altrimenti valuto l'espressione
119     if((t < 0) || (t > C)) return 0;
120     risultato = t*t;
121
122     // restituisco il risultato
123     return risultato;
124 }
125
126 // Calcolo prodotto convoluzionale
127 float costruisciProdottoConvoluzionale(float tao){
128     \verb|float| prod = 0;
129
130     for(int indice = 0; indice <= C; indice++){
131         // static_cast<float> per passare un\verb|float| alla funzione
132         prod+=(funcS(static_cast<float>(indice))*funcR(static_cast<float>(tao-indice)));
133     }
134     prod = prod/T;
135     return prod;
136 }
137
138 // Valuta la funzione data in input (puntatore)
139 float maxFunc(float (*funzione)(float t)){
140     \verb|float| funzValutata;
141

```

```

142 // La funzione va in modulo per non dover calcolare anche il minimo
143 \verb|float| max = abs(funzione(0));
144
145 // Cerco il massimo della funzione in modulo
146 for (int t = 1; t < C; t++) {
147     funzValutata = abs(funzione(static_cast<float>(t)));
148     if (funzValutata > max) max = funzValutata;
149 }
150
151 return max;
152 }
153
154 void disegnaFunzione(float (*funzione)(float x)) {
155     // Calcolo massimo funzione
156     \verb|float| max = maxFunc(funzione);
157
158     // Mi sposto sull'asse y
159     for (int y = ASSEY; y >= 0; y--)
160     {
161         // Mi sposto sull'asse x
162         for (int x = 0; x < C; x++)
163         {
164             if (y == round(funzione(static_cast<float>(x))/max*(ASSEY/2) + ASSEY/2))
165             {
166                 cout << '*';
167             }
168             else if ((y == ASSEY/2) && (x == 0))
169             {
170                 // se si trova al centro del grafico, inserisci +
171                 cout << '+';
172             }
173             else if (y == ASSEY/2)
174             {
175                 // se si trova sull'asse delle x, inserisci -
176                 cout << '-';
177             }
178             else if (x == 0)
179             {
180                 // se si trova sull'asse delle y, inserisci |
181                 cout << '|';
182             }
183             else
184             {

```

```

185         // Se non c'è nulla da rappresentare, inserisci spazio vuoto
186         cout << ' ';
187     }
188 }
189 // Fine y-esima riga
190 cout << '\n';
191 }
192 }

```

3. RISULTATO

L'algoritmo è stato testato con due prove:

1. con $s(t) = t/2$ e $r(t) = t^2 = t * t$, si ottiene l'output raffigurato nelle figure 1 a fronte, 2 nella pagina successiva, 3 a pagina 14 e 3 a pagina 14;
2. con $s(t) = 1$ e $r(t) = -t$, si ottiene l'output raffigurato nelle figure 5 a pagina 15, 6 a pagina 15, 7 a pagina 16 e 7 a pagina 16.

4. OSSERVAZIONI E CONCLUSIONI

Un'osservazione debita da fare riguarda il grafico del prodotto di convoluzione. Tale grafico è stato troncato e non viene rappresentato per intero. Una prima modifica che si potrebbe fare è normalizzare l'asse x per evitare questo, inoltre i grafici in output non presentano le frecce “^” e “>” per indicare l'asse y e l'asse x .

Per quanto riguarda il codice, invece, si potrebbe implementare un'altra funzione che calcola il valore della funzione e memorizza al suo interno il suo grafico per poi essere richiamata da un'altra funzione che provvede solo alla stampa della matrice riga per riga in modo da rendere il codice estendibile con maggiore facilità, condizione necessaria per future implementazioni.

5. RIFERIMENTI

Esercizi sulla convoluzione: https://didattica-2000.archived.uniroma2.it//SenTra/deposito/esercizi_convoluzione.pdf (ultimo accesso il 02/12/2022).

```
Last login: Fri Dec 2 00:05:36 on ttyS001
/Users/lucazepponi/Documents/VSC/c++/Esercitazioni/Prova/ex2 ; exit;

The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT288050.
Air-di-Luca:~ lucazepponi$ /Users/lucazepponi/Documents/VSC/c++/Esercitazioni/Prova/ex2 ; exit;

%-----%
Il programma effettua il prodotto di convoluzione tra
due funzioni reali limitate s(t) e r(t) campionate ad
intervallo fisso T.
In particolare, la funzione campionata ha valori nulli
ovunque tranne in un intervallo di 100 campioni, a
partire da t = 0.

Convoluzione discreta.
Date due funzioni s[t] e r[t] definite sull'insieme dei
numeri interi Z, il prodotto di convoluzione per
segnali campionati, nulle per t < 0, associate a
serie numeriche, abbiamo

$$C_T[r] = (s * r)[r]$$


$$= (\sum_{k=0}^{\infty} (s[k]r[r-k]))/T,$$

in cui T è il periodo di campionamento.
In particolare, si può notare che

$$\tau \in [-100, 100].$$

Al di fuori dell'intervallo C_T[r] è nulla.
%-----%
```

Figura 1: Prima prova, 1/4.

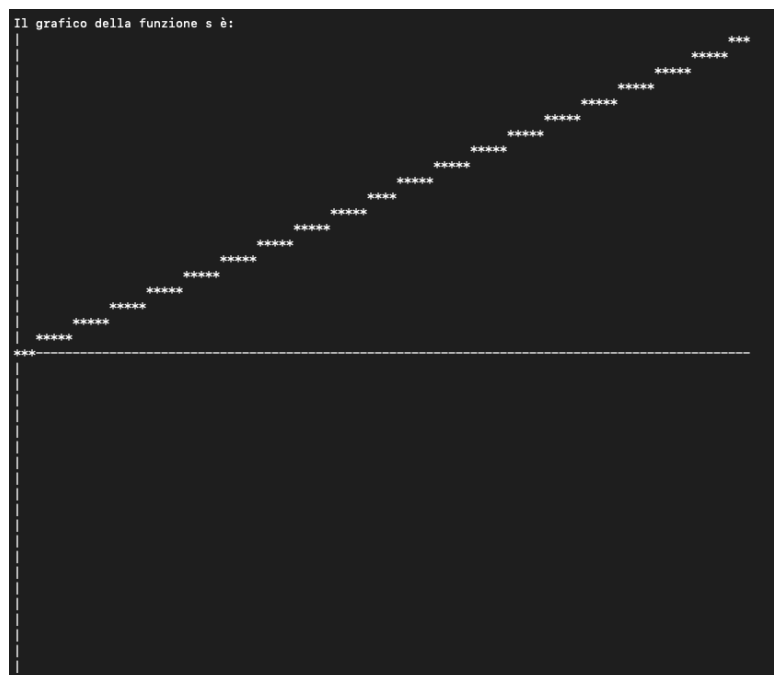


Figura 2: Prima prova, 2/4.

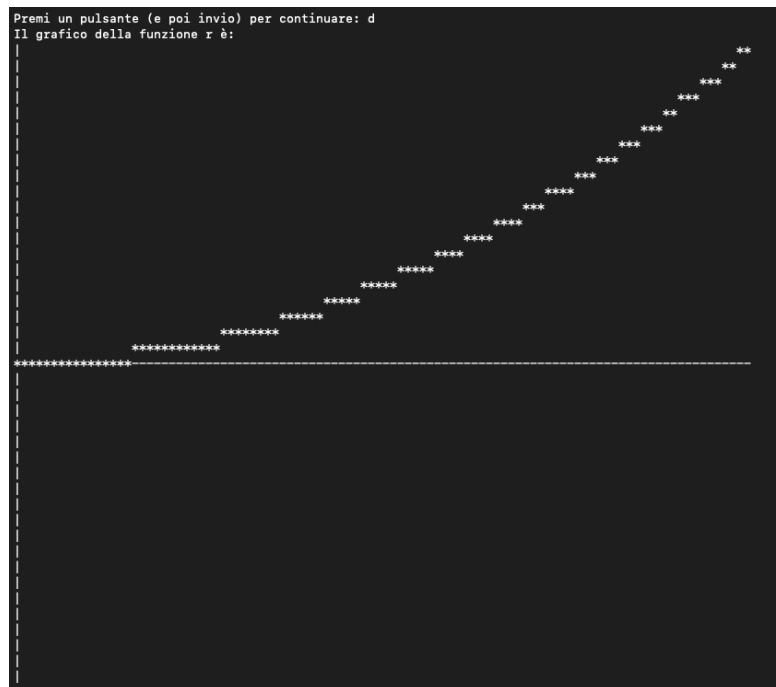


Figura 3: Prima prova, 3/4.

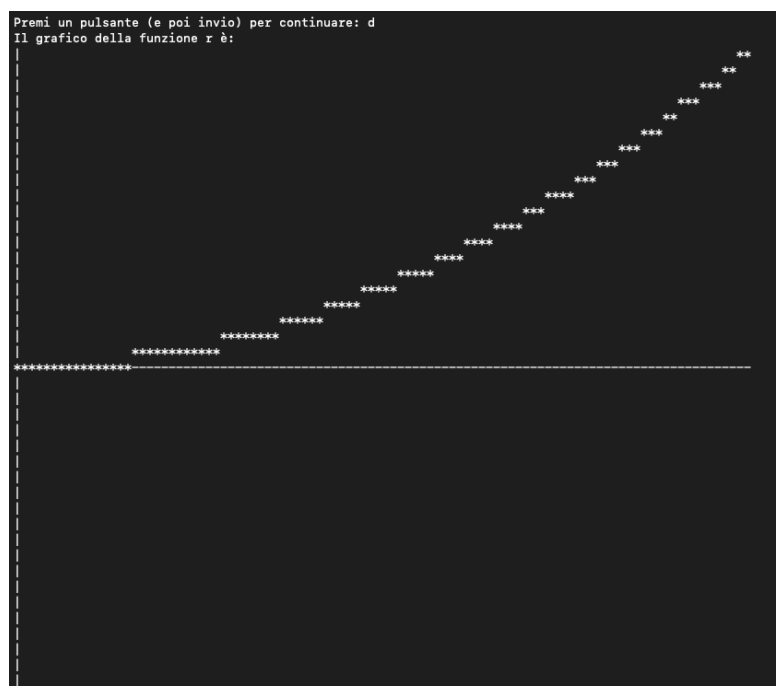


Figura 4: Prima prova, 4/4.

```
Last login: Fri Dec 2 00:06:25 on tty001

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
/Users/lucazepponi/Documents/VSC/c++/Esercitazioni/Prova/ex2 ; exit;
Air-di-Luca:~ lucazepponi$ /Users/lucazepponi/Documents/VSC/c++/Esercitazioni/Prova/ex2 ; exit;

%-----%
Il programma effettua il prodotto di convoluzione tra
due funzioni reali limitate s(t) e r(t) campionate ad
intervallo fisso T.
In particolare, la funzione campionata ha valori nulli
ovunque tranne in un intervallo di 100 campioni, a
partire da t = 0.

Convoluzione discreta.
Date due funzioni s[t] e r[t] definite sull'insieme dei
numeri interi Z, il prodotto di convoluzione per
segnali campionati, nulle per t < 0, associate a
serie numeriche, abbiamo

$$C_T[t] = (s * r)[t] = \left( \sum_{k=0}^{\infty} (s[k]r[t-k]) \right) / T,$$

in cui T è il periodo di campionamento.
In particolare, si può notare che
 $t \in [-100, 100]$ .
Al di fuori dell'intervallo  $C_T[t]$  è nulla.

%-----%
```

Figura 5: Seconda prova, 1/4.

Il grafico della funzione s è:

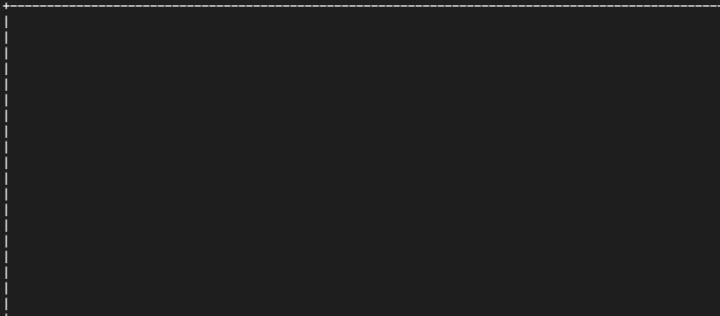
A blank coordinate system with dashed horizontal and vertical axes. The horizontal axis (x-axis) and vertical axis (y-axis) are represented by dashed lines intersecting at the origin. The axes extend across the width and height of the plot area, which is currently empty.

Figura 6: Seconda prova, 2/4.

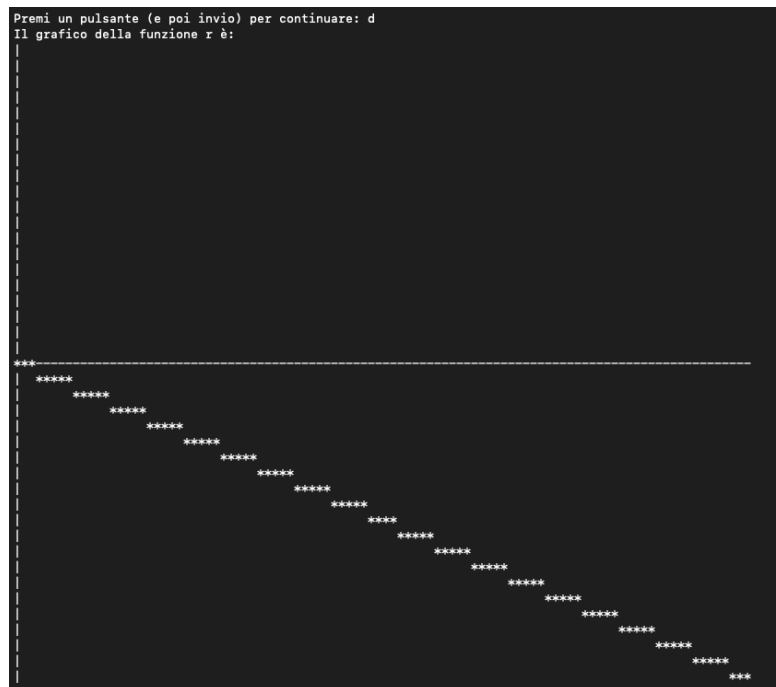


Figura 7: Seconda prova, 3/4.

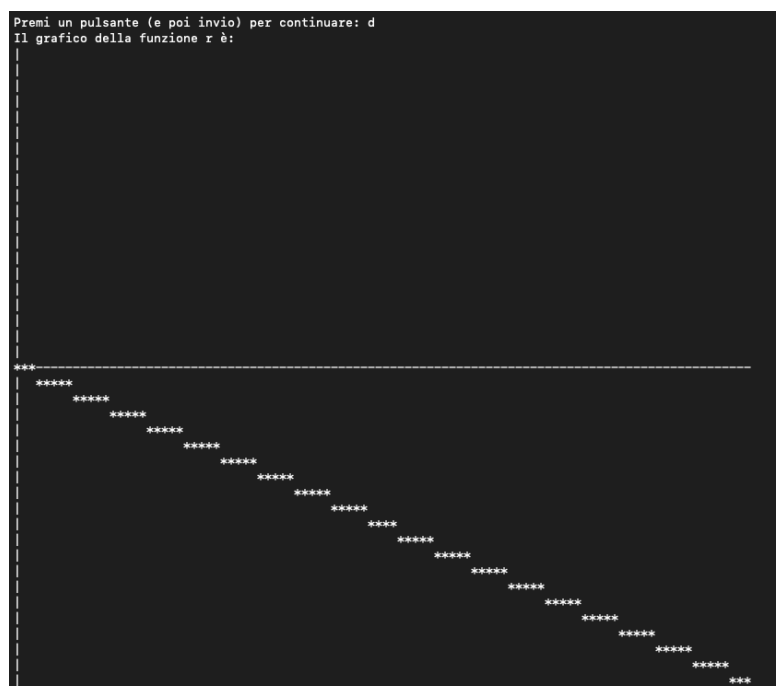


Figura 8: Seconda prova, 4/4.

Convoluzione: <https://it.wikipedia.org/wiki/Convoluzione> (ultimo accesso il 02/12/2022).

Libreria cmath: <https://www.codingcreativo.it/libreria-cmath/> Approfondimento puntatori a funzione: <http://lia.deis.unibo.it/Courses/InfoChim1112/lucidi/18-puntatoriAFunzione.pdf> (ultimo accesso il 02/12/2022).

Unicode characters: <https://www.unicodepedia.com/unicode/latin-1-supplement/97/control-0097/> e <https://stackoverflow.com/questions/40272566/how-to-print-greek-letter-delta-in-c> (ultimo accesso il 23/11/2022).