Ottimizzazione su GPU di modelli di materiali per l'edilizia civile

CANDIDATO Nikolas Sacchi
RELATORE Francesco Leporati
CORRELATORE Emanuele Torti

Università degli studi di Pavia - Facoltà di Ingegneria
Dipartimento di Ingegneria Industriale e dell'Informazione
Ingegneria Elettronica e Informatica

INDICE

- Introduzione
- Obiettivi
- Struttura *mesh*
- Norma H_∞
- Sviluppo di algoritmo seriale e parallelo
- Risultati
- Tempi di elaborazione
- Speed Up
- Conclusioni e sviluppi futuri

INTRODUZIONE

- Realizzazione di elementi architettonici:
  - in maniera automatica
  - rispettando vincoli
- Problema di ottimizzazione dinamica

INTRODUZIONE

- Sistema dinamico in formato **State Space**:

$$\begin{cases} E\dot{x} = Ax + Bw \\ z = Cx \end{cases}$$

- A, B, C, D, E matrici **sparse**
  - costruite tenendo conto del materiale e dei carichi

$$E = \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}, A = \begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix},$$

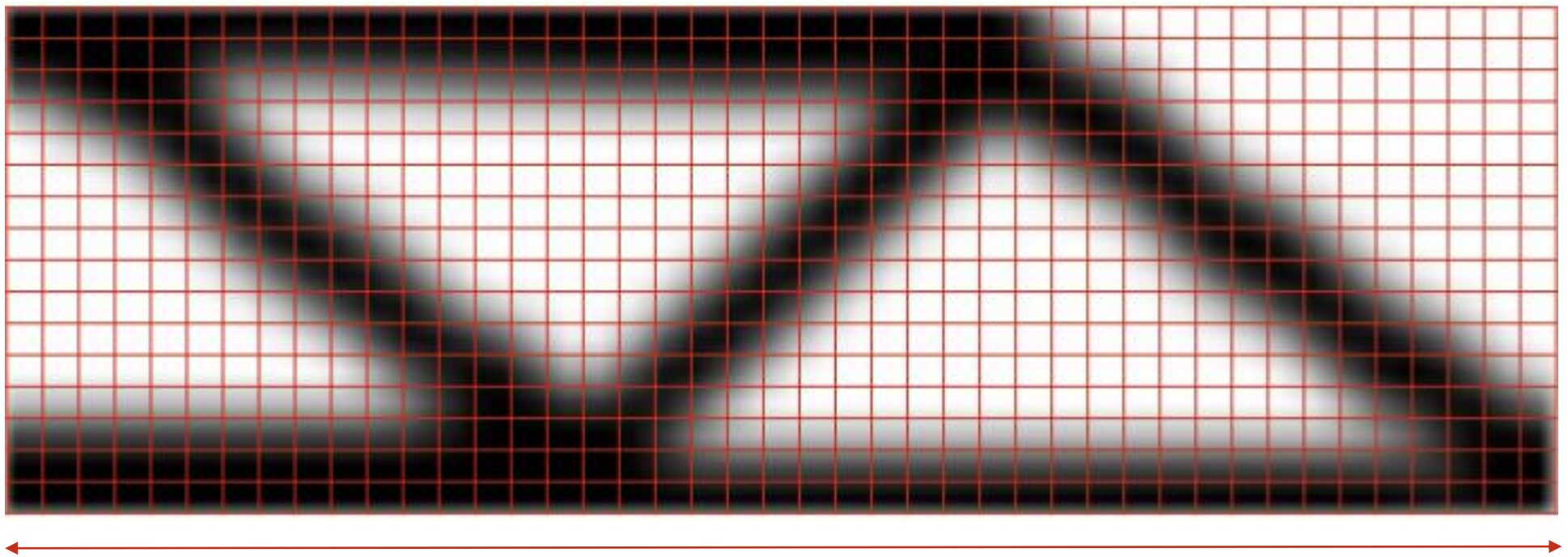
$$x = \begin{bmatrix} u \\ \dot{u} \end{bmatrix}, B = \begin{bmatrix} 0 \\ -M \end{bmatrix},$$

K = matrice delle rigidità
D = matrice degli smorzamenti
M = matrice delle masse
I = matrice identità
u = posizione
$\dot{u}$ = velocità

OBIETTIVI

- Minimizzazione della norma H_∞
- Caratterizzazione di una ***mesh*** per il progetto di elementi architettonici

MESH



- Suddivisione dello spazio in elementi discreti :
  - di eguali dimensioni
  - caratterizzati da valori compresi fra 0 e 1
    - ottenuti dalla matrice *xPhys* [$nelx * nely$]

NORMA H_∞

- Formulazione matematica:

$$||\boldsymbol{G}||_\infty = \sup_\omega |\boldsymbol{G}(i\omega)|$$

- Calcolo:
  - modulo della risposta in frequenza $G(i\omega)$ calcolato iterativamente nelle <u>basse frequenze</u>

$$\boldsymbol{G}(i\omega) = \boldsymbol{C}(i\omega\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{B}$$

```
while (current_omega <= omega_max) {
    /*
        RISPOSTA IN FREQUENZA IN current_omega
    */
    if (freq_resp >= hinf_nrm) {
        omega_peak = current_omega;
        hinf_nrm = freq_resp;
    }
    current_omega += omega_step;
}
```

- Parametri:
  - omega_max = 0.1 [rad/s]
  - omega_step = 0.01 [rad/s]
  - hinf_nrm e omega peak opportunamente inizializzati
  - current_omega inizializzato a 0 [rad/s]

ALGORITMO

- L'algoritmo:
  - determina la ***mesh ottima:***
    - garantisce norma H_∞ minima
    - rispetta i vincoli sulla quantità di materiale
  - è **iterativo**

ALGORITMO

- Ad ogni iterazione:
  - *xPhys* elaborata mediante **MMA**[1] (*Method of Moving Asymptotes*)
  - il sistema viene rivalutato secondo *xPhys*
  - norma H_∞ e gradiente vengono calcolati
  - *nuova mesh costruita con xPhys*
  - *le condizioni di terminazione*

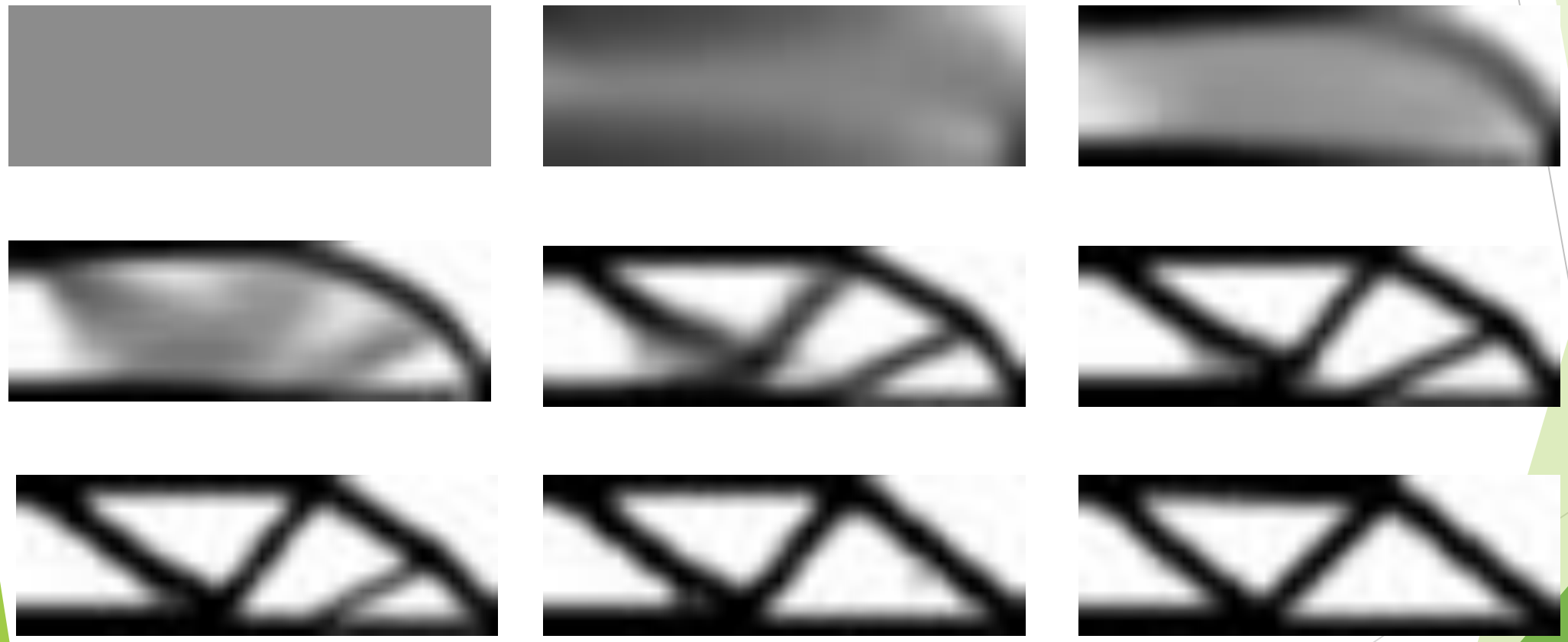
1. K. Svanberg, «The method of moving asymptotes – a new method for structural optimization», International journal for numerical methods in engineering, 24(2) February 1987

ALGORITMO

- Terminazione algoritmo:
  - condizioni di **Karush-Kuhn-Tucker** soddisfatte
  - numero massimo di iterazioni raggiunto

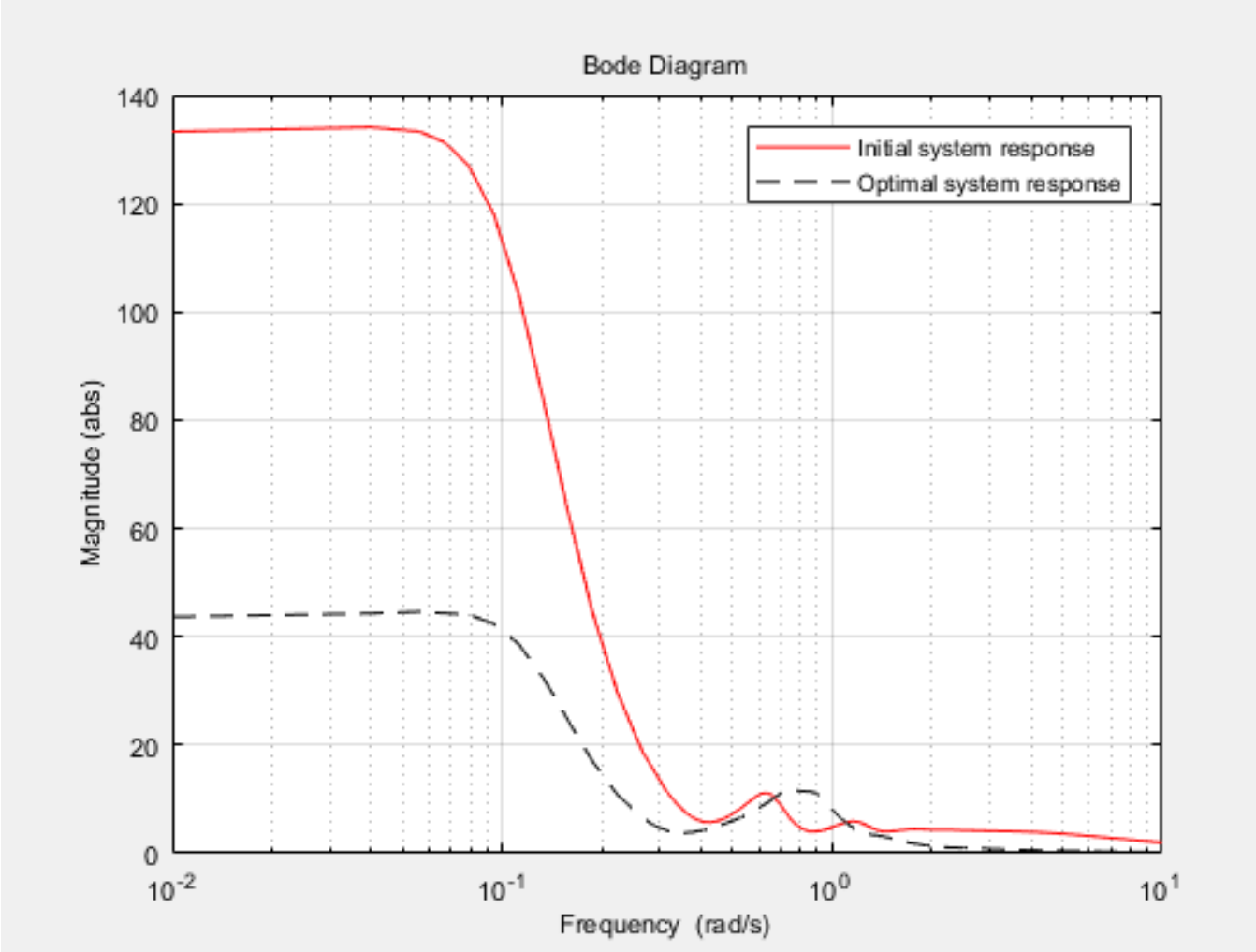
RISULTATI

- Realizzazione della *mesh 48x16*:

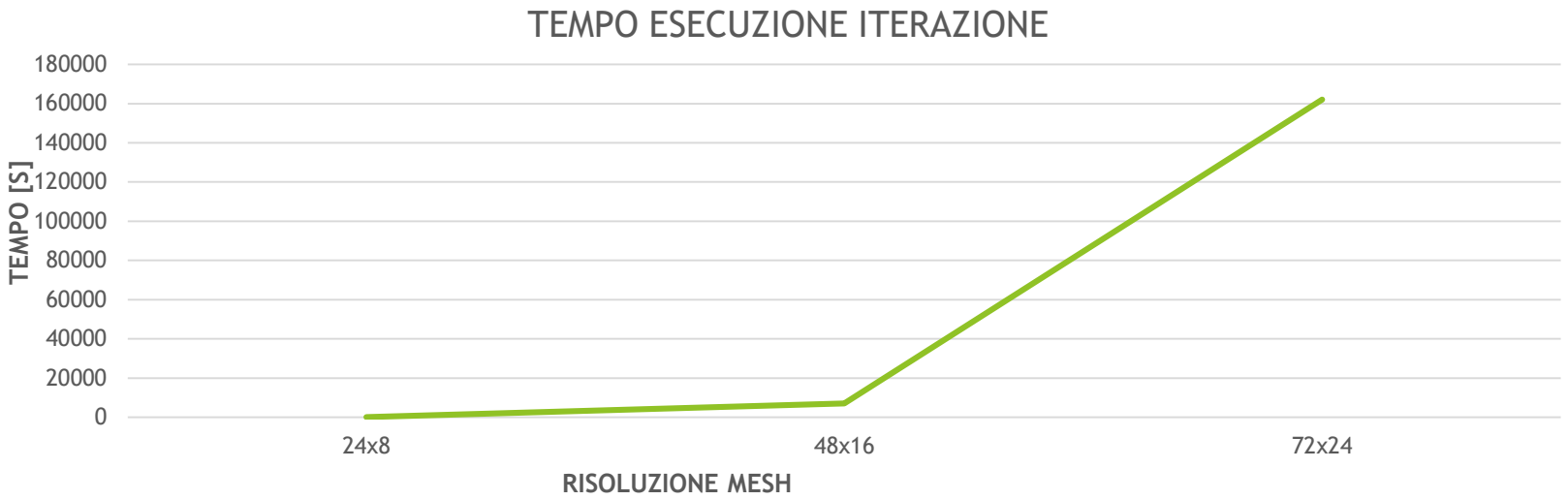


RISULTATI

- Minimizzazione della norma H_∞:



TEMPI DI ESECUZIONE SERIALE

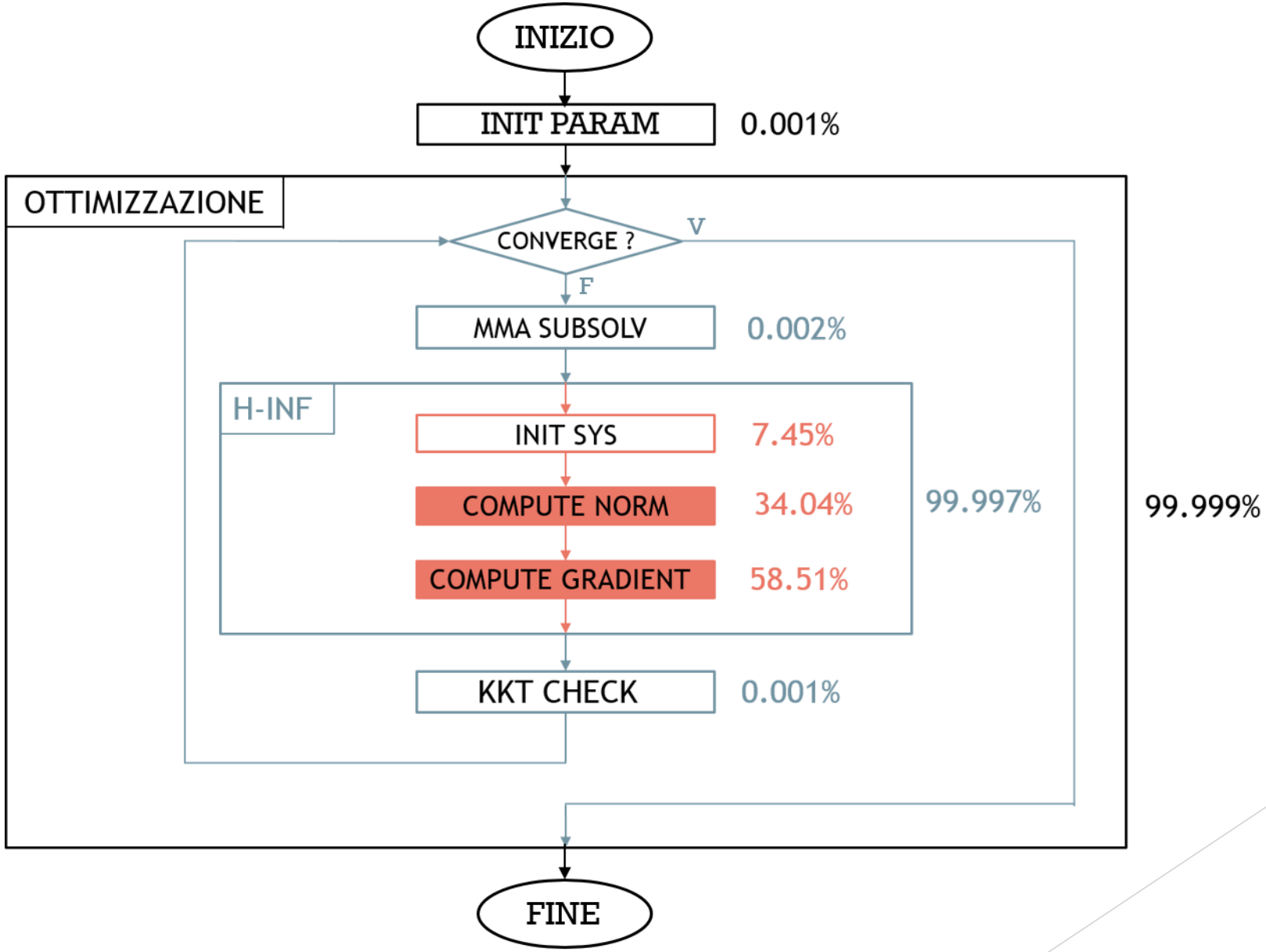


RISOLUZIONE MESH	ITERAZIONE	WORST CASE
24 x 8	≈ 90 s	≈ 12,5 ORE
48 x 16	≈ 7000 s	≈ 40 GIORNI
72 x 24	≈ 162000 s	≈ 2 ANNI 6 MESI

- **WORST CASE**: le condizioni KKT non vengono soddisfatte, l'algoritmo compie 500 iterazioni

- **HARDWARE**
  - **CPU**: Intel Core i7-3770 @3.40 GHz
  - **RAM**: 8 GB DDR3

FLUSSO DI ESECUZIONE



PARALLELIZZAZIONE

- I blocchi evidenziati sono stati parallelizzati su GPU:
  - CUDA 9.0

- Motivazioni:
  - prodotti tra matrici di grandi dimensioni
    - Complessità $O(n^3)$
  - inverse di matrici di grandi dimensioni

- Scelta della GPU:
  - rapidità di computazione di operazioni fra matrici
  - presenza di librerie dedicato

PARALLELIZZAZIONE

- Kernel custom:
  - differenza fra matrici
  - conversione matrici (sparse to dense)

- Libreria cuSOLVER:
  - fattorizzazione LU di matrici
    - *cusolverDnDgetrf()*
  - risoluzione di sistemi lineari $Ax = b$ necessaria per il calcolo delle matrici inverse
    - *cusolverDnDgetrs()*
  - funzioni gestite da *cusolverDnHandle_t*

- Libreria cuBLAS:
  - prodotto matrice-matrice
    - *cublasDgemm()*
  - funzioni gestite da *cublasHandle_t*
  - alte prestazioni con matrici di grandi dimensioni

GESTIONE DELLA MEMORIA

- Fondamentale la gestone delle memorie HOST e DEVICE:
  - HOST: utilizzata durante l'esecuzione seriale (CPU)
  - DEVICE: utilizzata durante l'esecuzione parallela (GPU)

- Memoria HOST:
  - allocazione: *malloc()*
  - deallocazione: *free()*

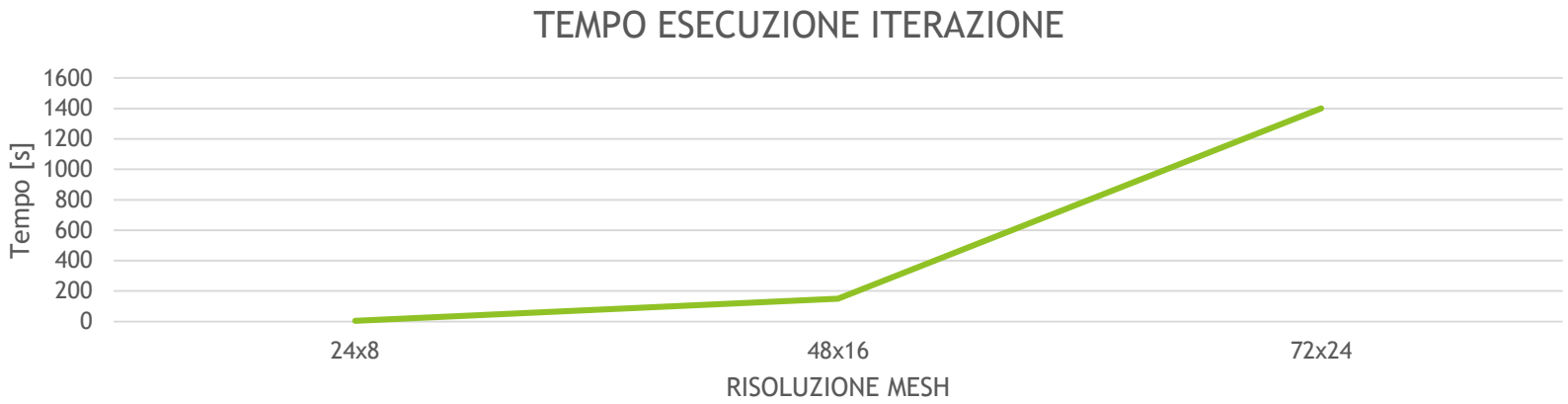
- Memoria DEVICE:
  - allocazione: *cudaMalloc()*
  - deallocazione: *cudaFree()*

GESTIONE DELLA MEMORIA

- Per essere utilizzati nel calcolo parallelo, gli elementi su HOST vengono copiati su DEVICE:
  - matrici A, B, C, D, E calcolate su HOST e copiate su DEVICE per il calcolo della norma H_∞
  - *cudaMemCpy(*dest_D, *src_H, size_t dimension,* ***cudaMemcpyHostToDevice)***

- Per essere utilizzati dalla CPU e visualizzati, i risultati calcolati sul DEVICE devono essere copiati su HOST:
  - la norma viene copiata su HOST per essere utilizzata nell'algoritmo di ottimizzazione H_∞
  - *cudaMemCpy(*dest_H, *src_D, size_t dimension,* ***cudaMemcpyDeviceToHost)***

TEMPI DI ESECUZIONE PARALLELO

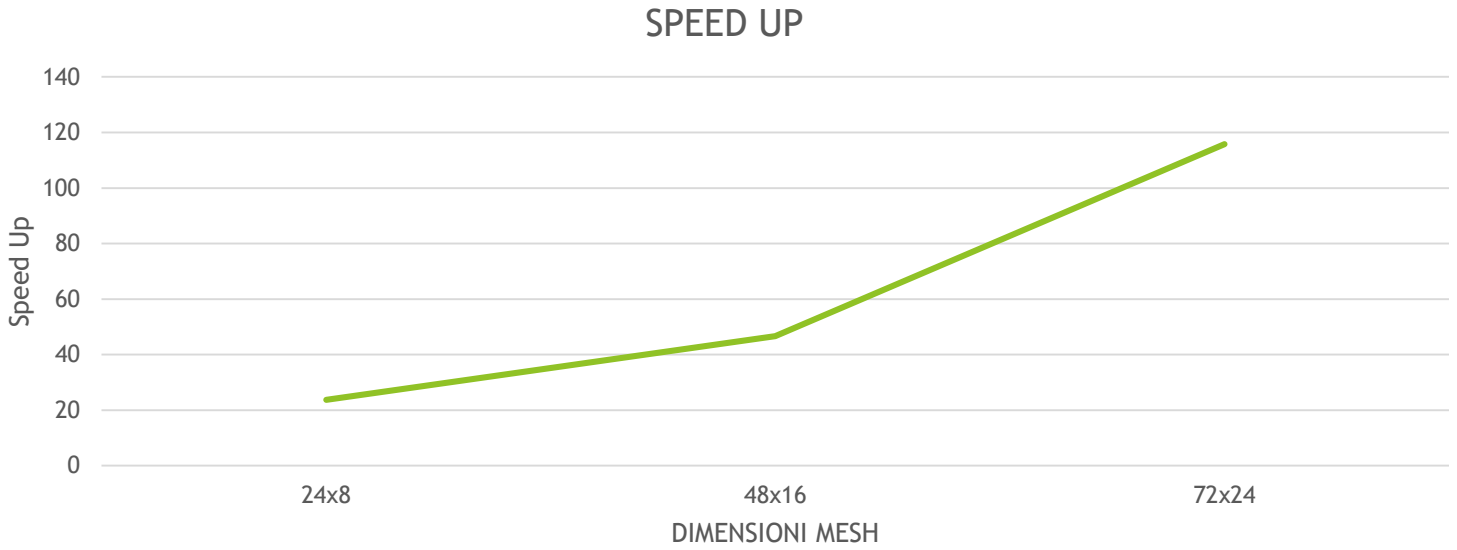


RISOLUZIONE MESH	ITERAZIONE	WORST CASE
24 x 8	≈ 3.8 s	≈ 31 MINUTI
48 x 16	≈ 150 s	≈ 21 ORE
72 x 24	≈ 1400 s	≈ 8 GIORNI

- **WORST CASE**: le condizioni KKT non vengono soddisfatte, l'algoritmo compie 500 iterazioni

- **GPU**
  - Nvidia Tesla K40c
  - 2880 CUDA cores @745 MHz
  - 12 GB DDR4 memory @3004 MHz

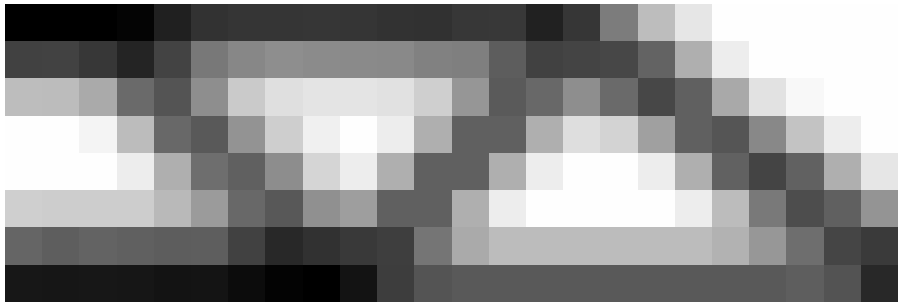
SPEED UP



RISOLUZIONE MESH	SERIALE	PARALLELO	SPEED UP
24 x 8	≈ 90 s	≈ 3.8 s	≈ 23,7
48 x 16	≈ 7000 s	≈ 150 s	≈ 46,67
72 x 24	≈ 162000 s	≈ 1400 s	≈ 115,75

VANTAGGI

- L'esecuzione parallela riduce notevolmente i tempi d'esecuzione:
  - realizzazione di *mesh* a risoluzione maggiore
    - migliore conoscenza della geometria
  - realizzazione di elementi architettonici più complessi
    - piloni, ponti



24 x 8

48 x 16

VANTAGGI

- Una migliore conoscenza della geometria:
  - permette una migliore distribuzione del materiale nell'intorno di singolarità di carico
  - semplifica l'ottimizzazione di altre caratteristiche
    - dissipazione di calore
  - facilita la stampa 3D dell'elemento
    - migliore definizione dei bordi

SVILUPPI FUTURI

- Sviluppi futuri:
  - metodo più efficiente per il calcolo di H_∞
  - condizioni di convergenza meno restrittive
  - realizzazione *mesh* multi materiale

GRAZIE PER L'ATTENZIONE

Ottimizzazione su GPU di modelli di materiali per l'edilizia civile

CANDIDATO Nikolas Sacchi
RELATORE Francesco Leporati
CORRELATORE Emanuele Torti

Università degli studi di Pavia - Facoltà di Ingegneria
Dipartimento di Ingegneria Industriale e dell'Informazione
Ingegneria Elettronica e Informatica