FEM

Stato attuale esempio funzionante di solutore FEM (per PDE).

Caratteristiche generazione parallela della matrice del problema, soluzione parallela.

Potenzialità integrazione con CfL (manca parsing latex)

Speedup da provare (mostrare anche un test su gpu01 durante il seminario?)

Identificazione

Stato attuale codice Pyhton/C di un solutore basato su GN (ma abbastanza complesso), scheletro implementazione GPU, implementazione CUDA della costruzione matrice Ψ

Idea ciclo esterno (elaborazione dei *job*) su CPU (eventualmente in parallelo alla GPU), ciclo interno (generazione matrice Ψ, fit ai minimi quadrati con GN) su GPU; semplificare molto il codice Python/C a disposizione (implementare un semplice fit ai minimi quadrati)

Criticità 1) riuscire a *riempire* i thread della GPU, per ora parallelismo poco spinto su parametri e $\pm\delta$, es: $N_p = 6, N_d = 6 \implies P = 6 \cdot 6 \cdot 2 = 72$, dove 2 è dovuto al segno di δ. 2) Gestione della memoria (del trasferimento di dati) impegnativo per la complessità dell'algoritmo. 3) Se l'identificazione è tempo-variante si può aggiungere il parallelismo sugli intervalli temporali? Sono indipendenti le sezioni o devono combaciare agli estremi?

Speedup solutore da implementare, speed-up teorico molto alto?

Potenzialità speed-up teoricamente molto alto se buona implementazione, possibile integrazione con CfL

Interfaccia

Stato attuale per il FEM: PyCuda (per codice CUDA) e ctypes (per importare .dll); da replicare per l'identificazione?

Caratteristiche con PyCuda si evita la compilazione degli script CUDA e vengono forniti metodi per semplificare il trasferimento di dati da/a GPU.

Criticità librerie come cuBLAS, CULA non forniscono metodi *device-callable*, ma solo *host-callable*