πλατφόρμα μαθηματικών μεθόδων και λογισμικού
για την επίλυση προβλημάτων πολλαπλών πεδίων
σε σύγχρονες υπολογιστικές αρχιτεκτονικές

Αριθμητικές Μέθοδοι Collocation

Απεικόνιση σε Σύγχρονες Υπολογιστικές Αρχιτεκτονικές

ΘΗΣΕΥΣ

European Union
European Social Fund

OPERATIONAL PROGRAMME
EDUCATION AND LIFELONG LEARNING
investing in knowledge society
MINISTRY OF EDUCATION & RELIGIOUS AFFAIRS
MANAGING AUTHORITY

NSRF 2007-2013
programme for development
EUROPEAN SOCIAL FUND

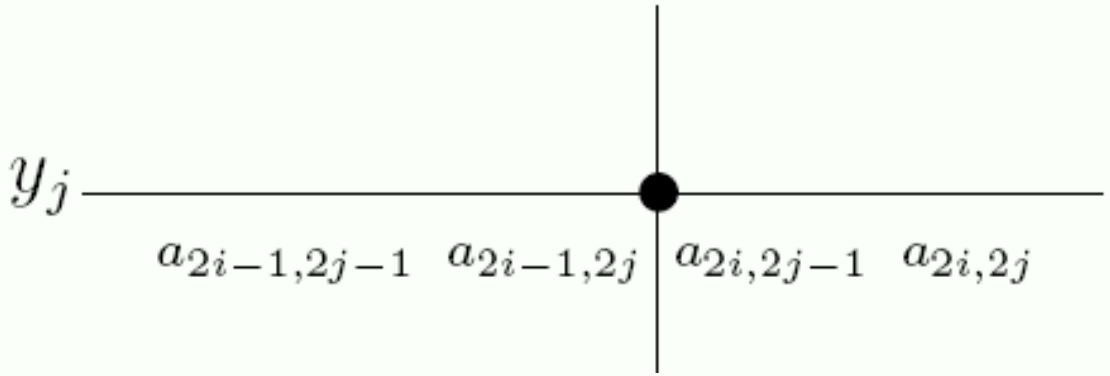
Co-financed by Greece and the European Union

Hermite Collocation Method

BVP

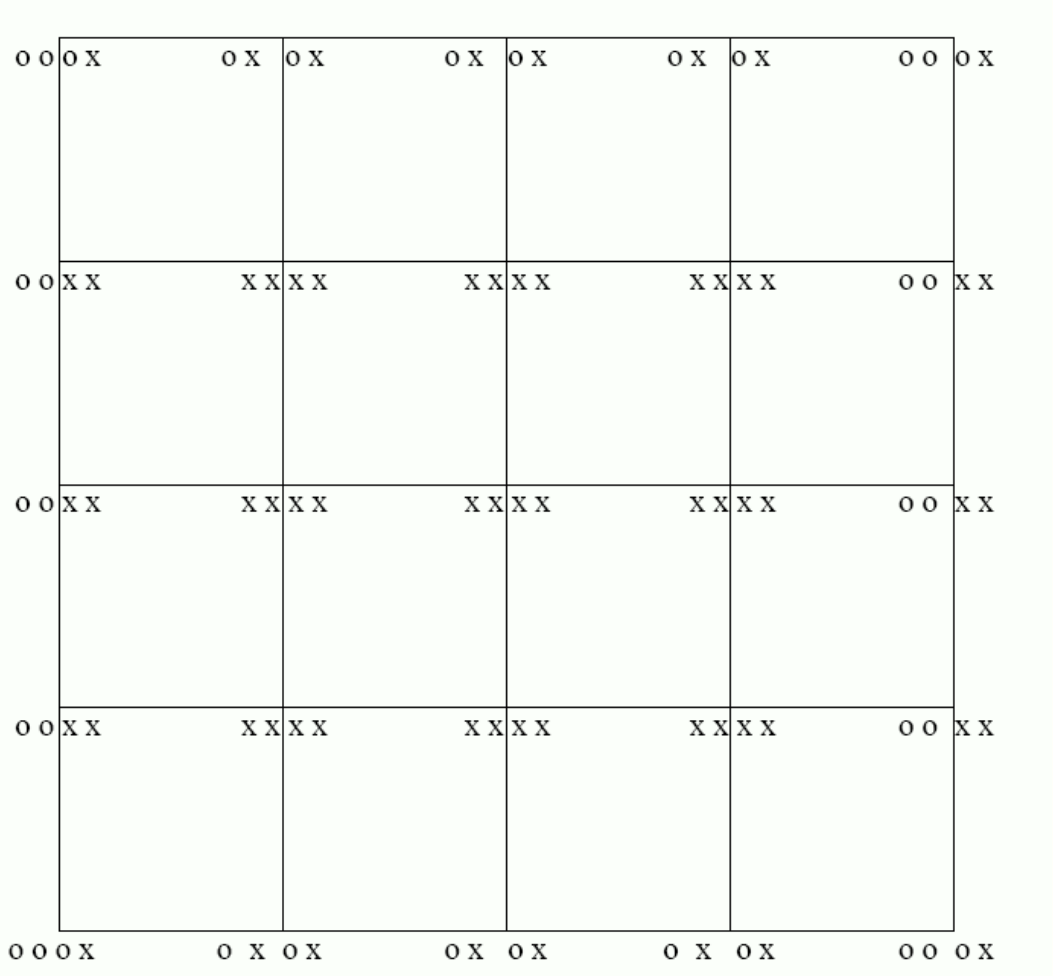
$$\mathcal{L}\,u(x,y) = f(x,y) \quad , \quad (x,y) \in \Omega$$

$$\mathcal{B}\,u(x,y) = g(x,y) \quad , \quad (x,y) \in \partial\Omega$$

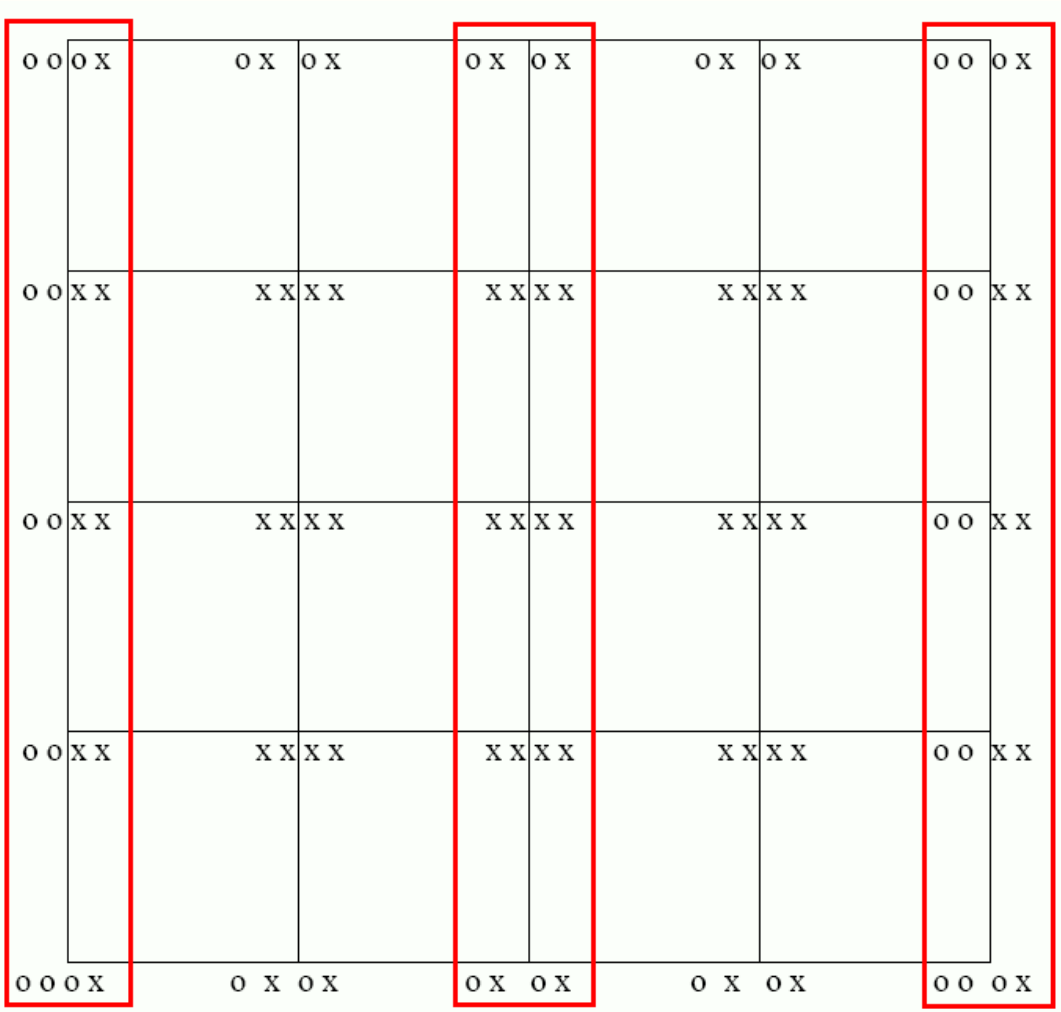


$$\begin{cases} u_n(x_i,y_j) = a_{2i-1,2j-1} \;, & h\frac{\partial}{\partial y}u_n(x_i,y_j) = a_{2i-1,2j} \\ h\frac{\partial}{\partial x}u_n(x_i,y_j) = a_{2i,2j-1} \;, & h^2\frac{\partial^2}{\partial x \partial y}u_n(x_i,y_j) = a_{2i,2j} \end{cases}$$

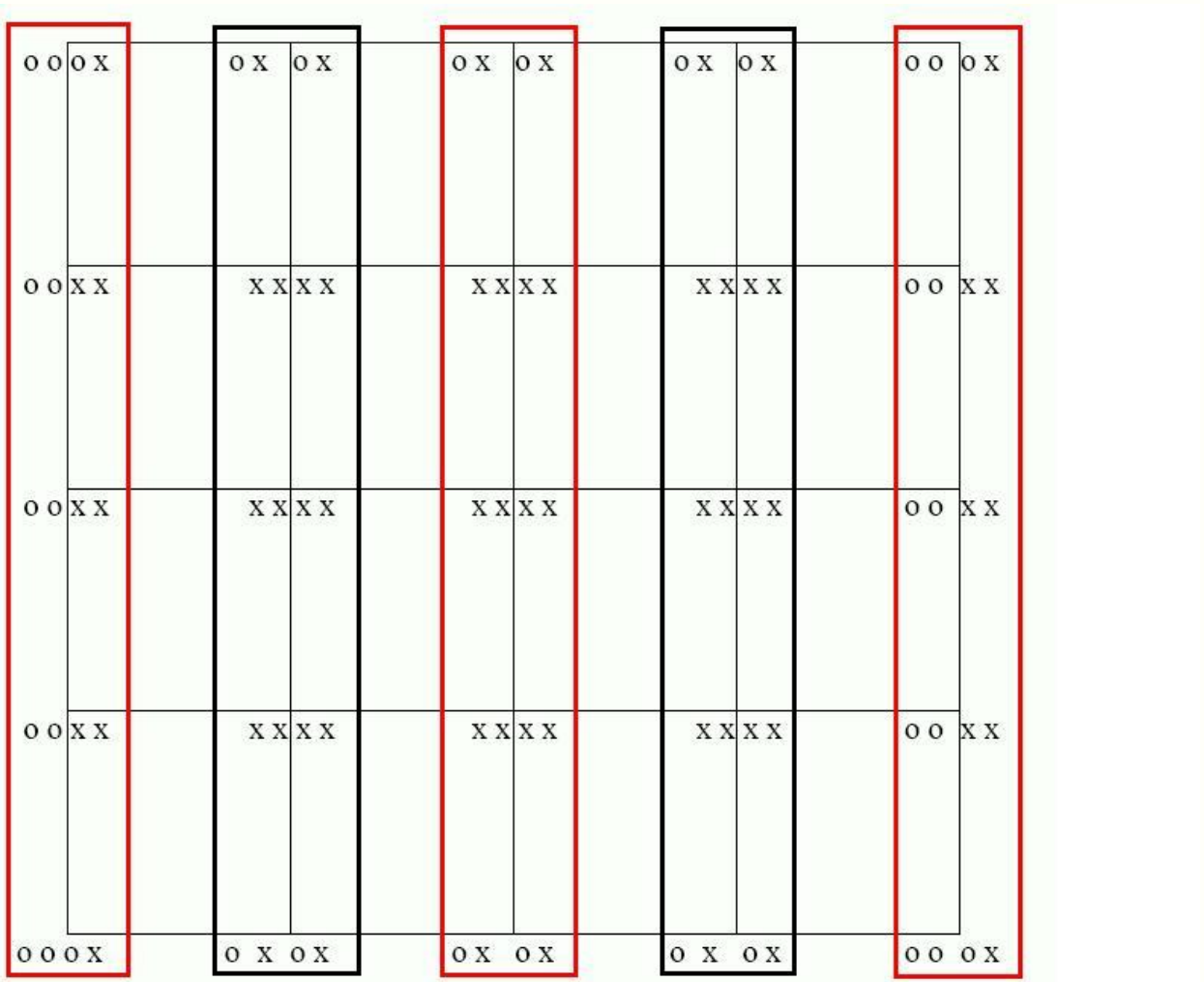
Red – Black Collocation Linear system



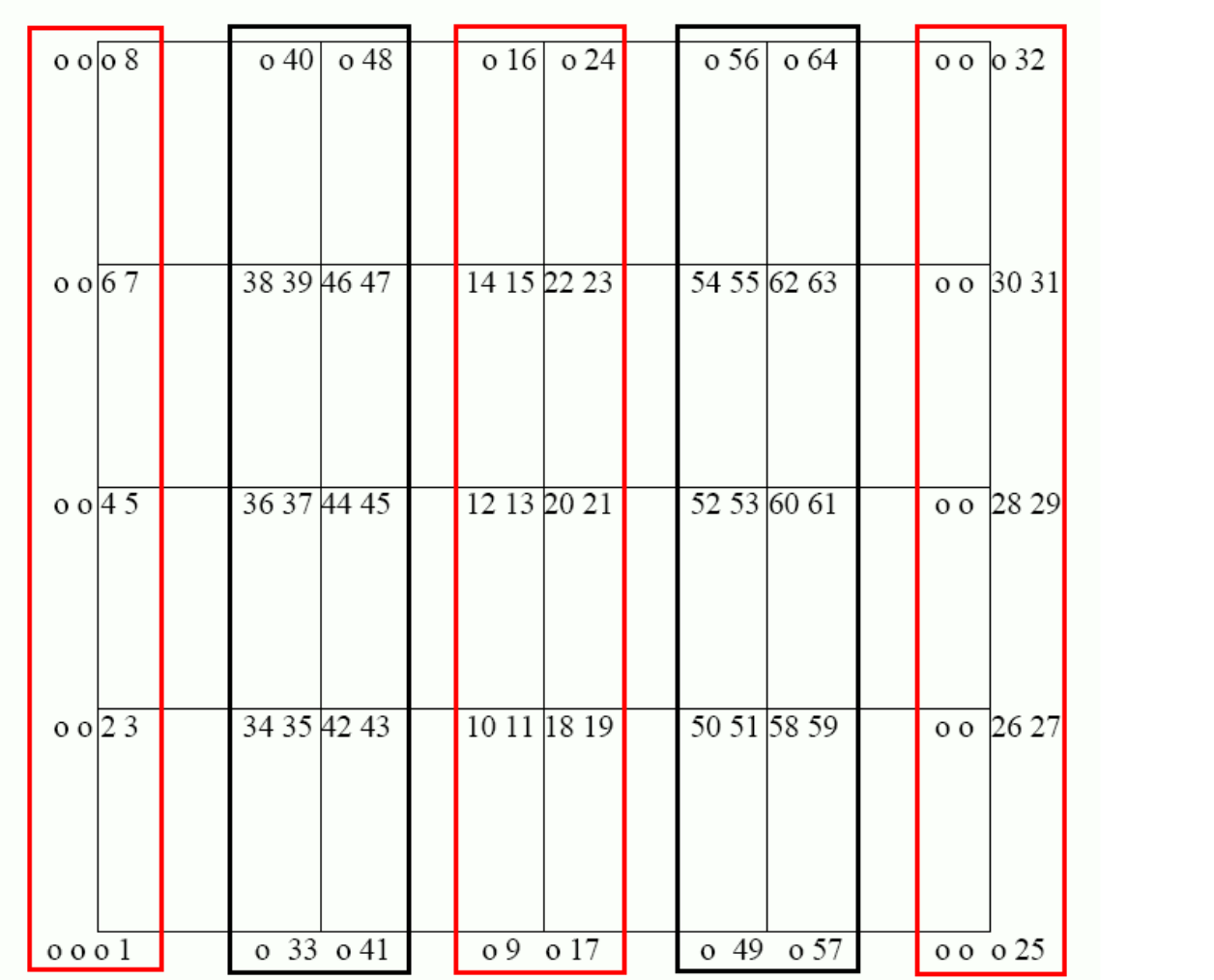
Red – Black Collocation Linear system



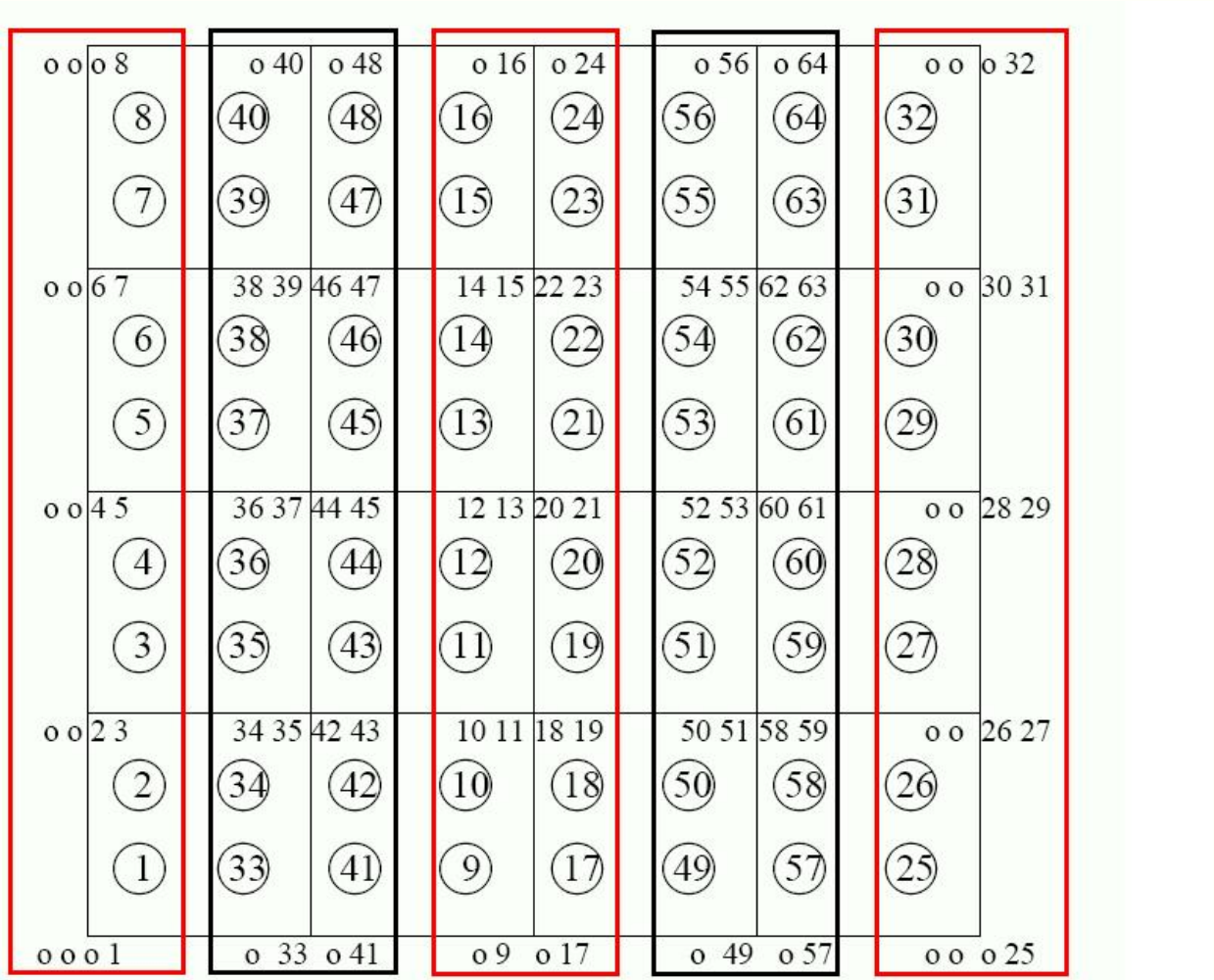
Red – Black Collocation Linear system



Red – Black Collocation Linear system



Red – Black Collocation Linear system

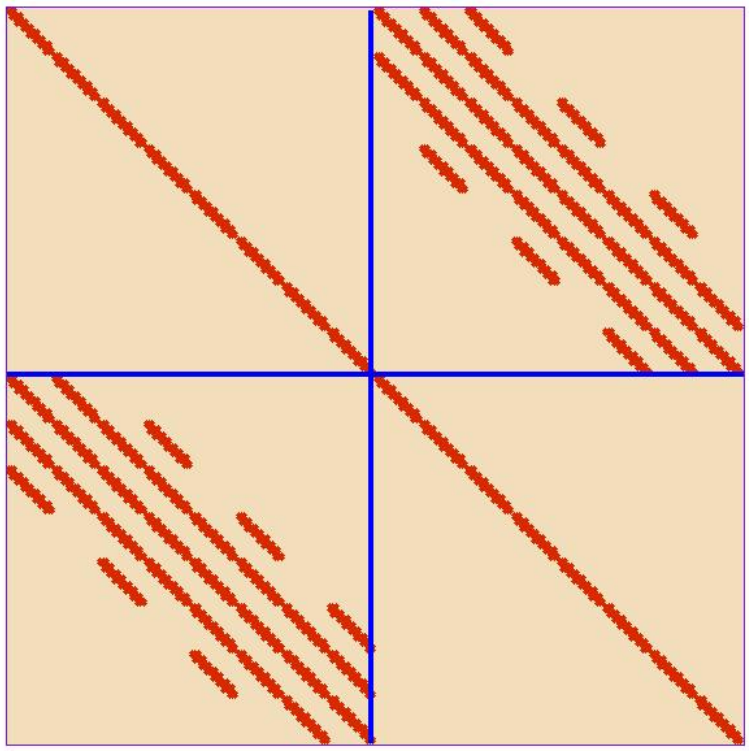


Model Problem

$$\nabla^2 u(x,y) - \lambda u(x,y) = f(x,y) \quad , \quad (x,y) \in \Omega$$

$$u(x,y) = g(x,y) \quad , \quad (x,y) \in \partial\Omega \quad \text{with} \quad \lambda \geq 0$$

$$A = \begin{bmatrix} D_R & H_B \\ H_R & D_B \end{bmatrix}$$



Helmholtz Collocation Matrix

$$A = \begin{bmatrix} D_R & H_B \\ H_R & D_B \end{bmatrix}$$

$$D_R = 2\,\mathrm{diag}[\frac{1}{2}A_2\ A_1\ A_2\ \cdots\ A_1\ A_2\ -\frac{1}{2}A_2]$$

$$D_B = 2\,\mathrm{diag}[A_1\ A_2\ \cdots\ A_1\ A_2]$$

$$H_B = -\begin{bmatrix} A_3 & -A_4 & O & O & \cdots & O & O & O & O \\ A_3 & A_4 & A_3 & -A_4 & \cdots & O & O & O & O \\ -A_3 & -A_4 & A_3 & -A_4 & \cdots & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & A_3 & A_4 & A_3 & -A_4 \\ O & O & O & O & \cdots & -A_3 & -A_4 & A_3 & -A_4 \\ O & O & O & O & \cdots & O & O & A_3 & A_4 \end{bmatrix}$$

$$H_R = -\begin{bmatrix} A_4 & A_3 & -A_4 & O & O & \cdots & O & O & O & O & O \\ -A_4 & A_3 & -A_4 & O & O & \cdots & O & O & O & O & O \\ O & A_3 & A_4 & A_3 & -A_4 & \cdots & O & O & O & O & O \\ O & -A_3 & -A_4 & A_3 & -A_4 & \cdots & O & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \cdots & A_3 & A_4 & A_3 & -A_4 & O \\ O & O & O & O & O & \cdots & -A_3 & -A_4 & A_3 & -A_4 & O \\ O & O & O & O & O & \cdots & O & O & A_3 & A_4 & -A_4 \\ O & O & O & O & O & \cdots & O & O & -A_3 & -A_4 & -A_4 \end{bmatrix}$$

Red – Black Collocation Linear system

$$A_i = \begin{bmatrix} a_2 & a_3 & -a_4 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ a_4 & a_1 & -a_2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & a_1 & a_2 & a_3 & -a_4 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & a_3 & a_4 & a_1 & -a_2 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_1 & a_2 & a_3 & -a_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_3 & a_4 & a_1 & -a_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_1 & a_2 & -a_4 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_3 & a_4 & -a_2 \end{bmatrix}$$

	a_1	a_2	a_3	a_4
A_1	r^+	s^+	q	t^+
A_2	s^+	u^+	t^-	ϵ
A_3	q	t^-	r^-	s^-
A_4	t^+	ϵ	s^-	u^-

with

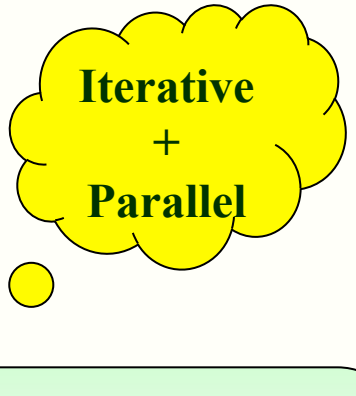
$\epsilon = -\frac{\lambda}{24n_s^2}$, $q = 24 + 22\epsilon$,

$r^{\pm} = 86\epsilon - 24 \pm (48\epsilon - 18)\sqrt{3}$,

$s^{\pm} = 13\epsilon - 12 \pm (7\epsilon - 8)\sqrt{3}$,

$t^{\pm} = 5\epsilon + 3 \pm (\epsilon + 1)\sqrt{3}$,

$u^{\pm} = 2\epsilon - 3 \pm (\epsilon - 2)\sqrt{3}$.



- Ο collocation πίνακας είναι μεγάλης διάστασης, αραιός και δεν έχει κάποια ιδιαίτερα χρήσιμη ιδιότητα (π.χ. συμμετρικός, θετικά διευθετιμένος)

Iterative Solution

$$A = \begin{bmatrix} D_R & H_B \\ H_R & D_B \end{bmatrix}$$

$$A = D_A - L_A - U_A$$

όπου

$$D_A = \begin{bmatrix} D_R & O \\ O & D_B \end{bmatrix}, \; L_A = \begin{bmatrix} O & O \\ -H_R & O \end{bmatrix}, \; U_A = \begin{bmatrix} O & -H_B \\ O & O \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{bmatrix} \text{ and } \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_R \\ \boldsymbol{b}_B \end{bmatrix} .$$

Iterative Solution

$$M_1^{-1} A M_2^{-1} M_2 \boldsymbol{x} = M_1^{-1} \boldsymbol{b}$$

where $M_1 = D_A - L_A = D_A(I - D_A^{-1} L_A)$

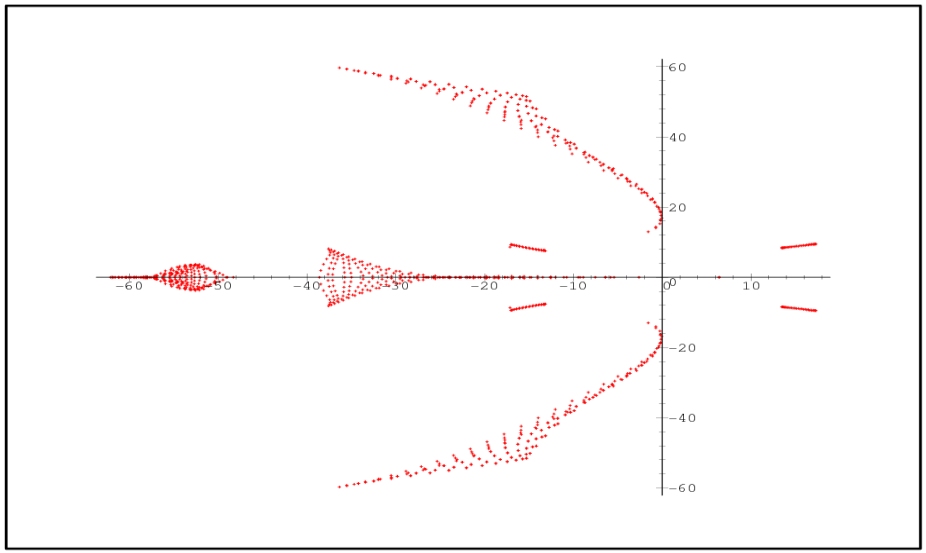
and $M_2 = I - D_A^{-1} U_A$

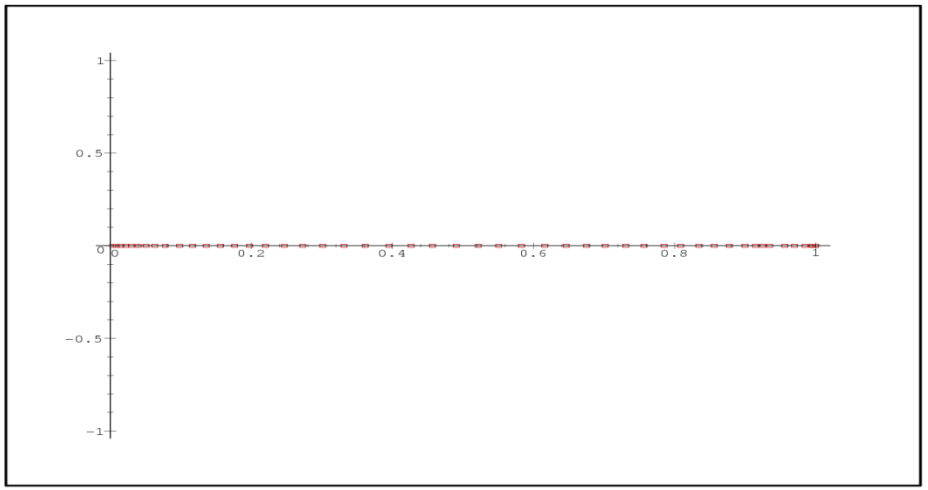
$$\begin{bmatrix} I & O \\ O & S \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_R + D_R^{-1} H_R \boldsymbol{x}_B \\ \boldsymbol{x}_B \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{b}}_R \\ \hat{\boldsymbol{b}}_B \end{bmatrix}$$

where $S = D_B - H_R D_R^{-1} H_B$

$$\hat{\boldsymbol{b}}_R = D_R^{-1} \boldsymbol{b}_R \quad \text{and} \quad \hat{\boldsymbol{b}}_B = \boldsymbol{b}_B - H_R \hat{\boldsymbol{b}}_R$$

Eigenvalues of Collocation matrix





Schur Complement Iterative Solution

S1: $\underline{\textit{Solve}} \quad D_R \hat{\boldsymbol{b}}_R = \boldsymbol{b}_R$

S2: $\underline{\textit{Evaluate}} \quad \hat{\boldsymbol{b}}_B = \boldsymbol{b}_B - H_R \hat{\boldsymbol{b}}_R$

S3: $\underline{\textit{Solve with BiCGSTAB}} \quad S\,\boldsymbol{x}_B = \hat{\boldsymbol{b}}_B$

S4: $\underline{\textit{Evaluate}} \quad \hat{\boldsymbol{x}}_B = H_B \boldsymbol{x}_B$

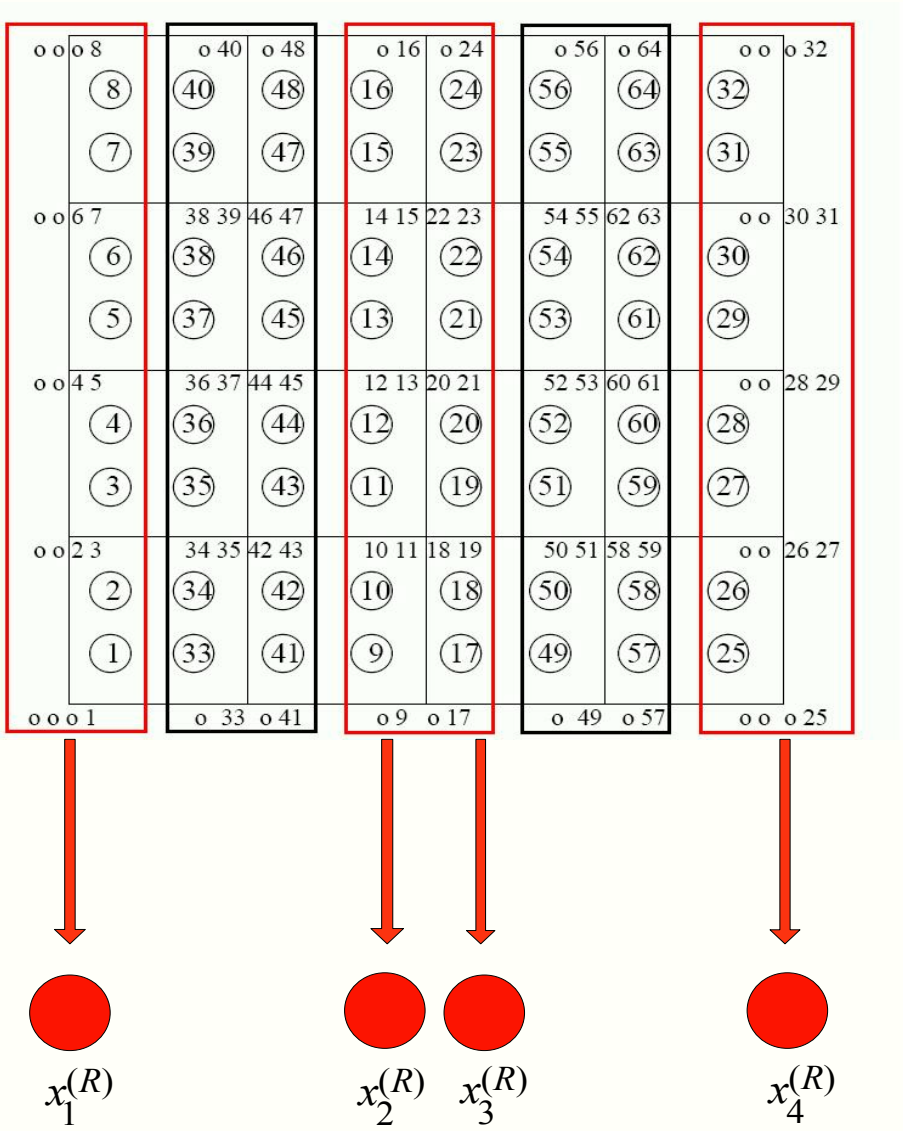
S5: $\underline{\textit{Solve}} \quad D_R \hat{\boldsymbol{x}}_R = \hat{\boldsymbol{x}}_B$

S6: $\underline{\textit{Evaluate}} \quad \boldsymbol{x}_R = \hat{\boldsymbol{b}}_R - \hat{\boldsymbol{x}}_R$

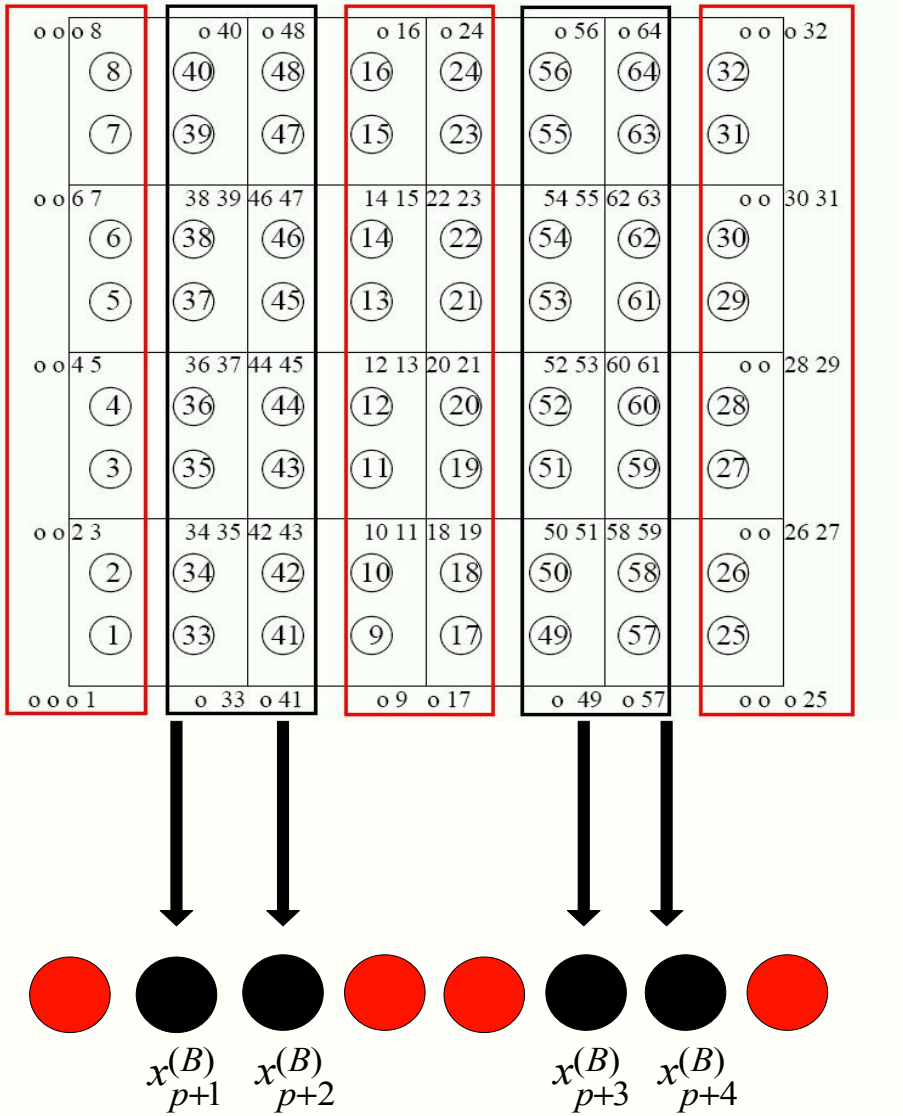
Parallel Iterative Solution of Collocation Linear system on Shared Memory Architectures

- Ομοιόμορφη κατανομή φόρτου εργασίας μεταξύ core threads
- Ελαχιστοποίηση κύκλων αδρανών core threads
- Ελαχιστοποίηση κόστους επικοινωνίας

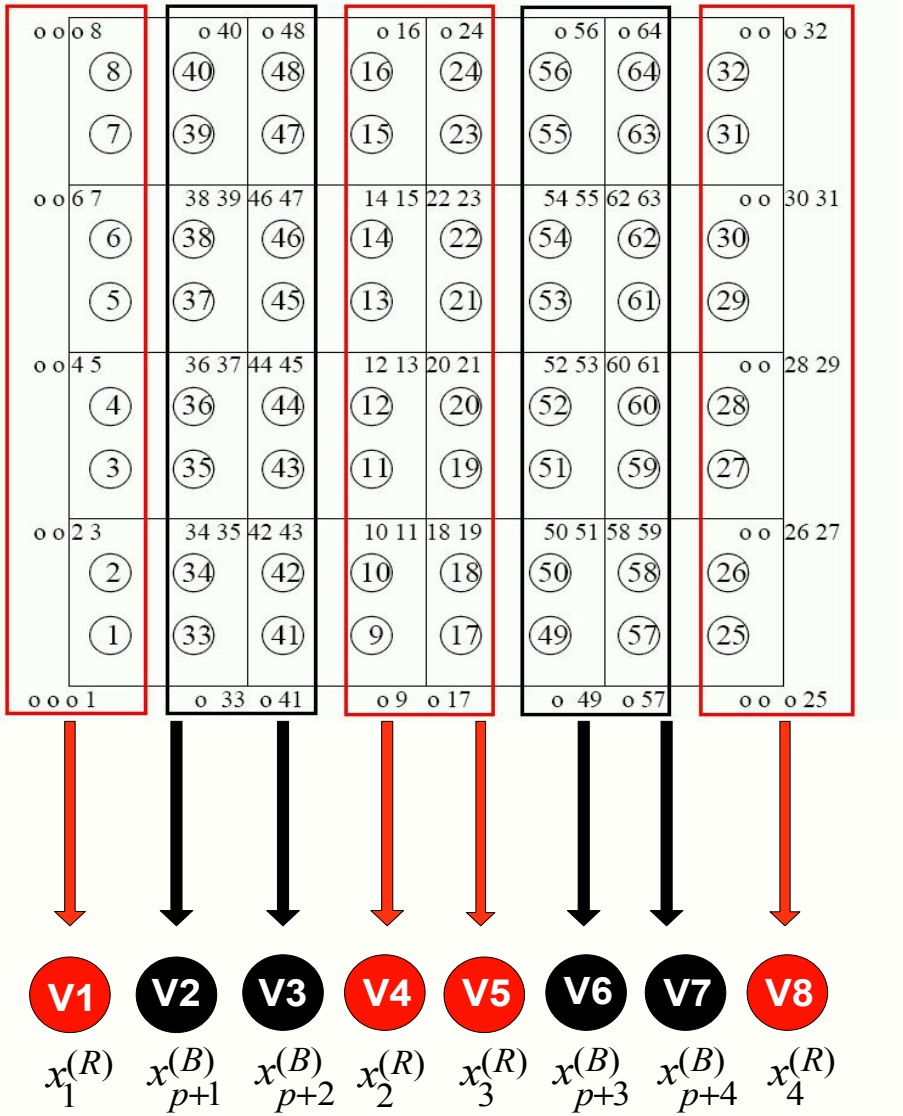
case of $ns = 2p$

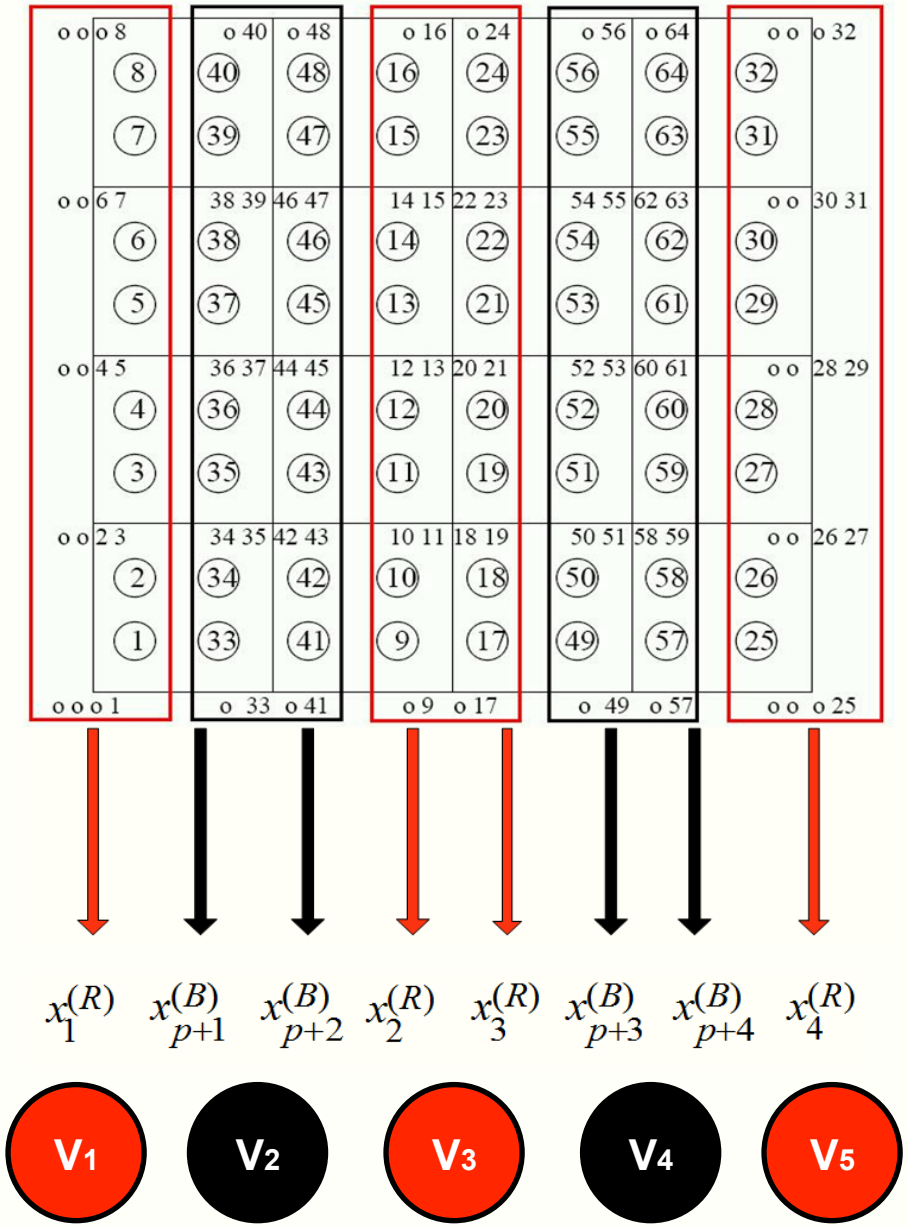

o o o 8
8
7
o 40
o 48
40
48
39
47
o 16
o 24
16
24
15
23
o 56
o 64
56
64
55
63
o o
o 32
32
31
o o 6 7
6
5
38 39 46 47
38
46
37
45
14 15 22 23
14
22
13
21
54 55 62 63
54
62
53
61
o o
30 31
30
29
o o 4 5
4
3
36 37 44 45
36
44
35
43
12 13 20 21
12
20
11
19
52 53 60 61
52
60
51
59
o o
28 29
28
27
o o 2 3
2
1
34 35 42 43
34
42
33
41
10 11 18 19
10
18
9
17
50 51 58 59
50
58
49
57
o o
26 27
26
25
o o o 1
o 33 o 41
o 9 o 17
o 49 o 57
o o o 25
$x_1^{(R)}$
$x_2^{(R)}$
$x_3^{(R)}$
$x_4^{(R)}$

case of $ns = 2p$

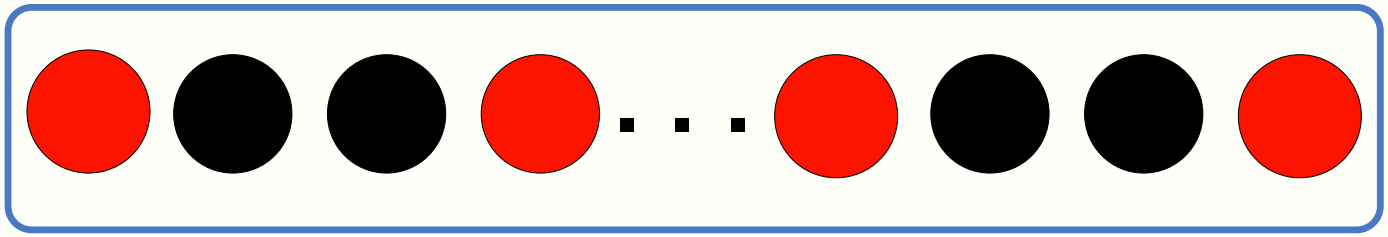

o o o 8
8
7
o 40
o 48
40
48
39
47
o 16
o 24
16
24
15
23
o 56
o 64
56
64
55
63
o o
o 32
32
31
o o 6 7
6
5
38 39 46 47
38
46
37
45
14 15 22 23
14
22
13
21
54 55 62 63
54
62
53
61
o o
30 31
30
29
o o 4 5
4
3
36 37 44 45
36
44
35
43
12 13 20 21
12
20
11
19
52 53 60 61
52
60
51
59
o o
28 29
28
27
o o 2 3
2
1
34 35 42 43
34
42
33
41
10 11 18 19
10
18
9
17
50 51 58 59
50
58
49
57
o o
26 27
26
25
o o o 1
o 33 o 41
o 9 o 17
o 49 o 57
o o o 25
$x^{(B)}_{p+1}$
$x^{(B)}_{p+2}$
$x^{(B)}_{p+3}$
$x^{(B)}_{p+4}$

case of $ns = 2p$


o o o 8
8
7
o 40
o 48
40
48
39
47
o 16
o 24
16
24
15
23
o 56
o 64
56
64
55
63
o o
o 32
32
31
o o 6 7
6
5
38 39 46 47
38
46
37
45
14 15 22 23
14
22
13
21
54 55 62 63
54
62
53
61
o o
30 31
30
29
o o 4 5
4
3
36 37 44 45
36
44
35
43
12 13 20 21
12
20
11
19
52 53 60 61
52
60
51
59
o o
28 29
28
27
o o 2 3
2
1
34 35 42 43
34
42
33
41
10 11 18 19
10
18
9
17
50 51 58 59
50
58
49
57
o o
26 27
26
25
o o o 1
o 33 o 41
o 9 o 17
o 49 o 57
o o o 25
V1
V2
V3
V4
V5
V6
V7
V8
$x_1^{(R)}$
$x_{p+1}^{(B)}$
$x_{p+2}^{(B)}$
$x_2^{(R)}$
$x_3^{(R)}$
$x_{p+3}^{(B)}$
$x_{p+4}^{(B)}$
$x_4^{(R)}$


o o o 8
8
7
o 40
o 48
40
48
39
47
o 16
o 24
16
24
15
23
o 56
o 64
56
64
55
63
o o
o 32
32
31
o o 6 7
6
5
38 39 46 47
38
46
37
45
14 15 22 23
14
22
13
21
54 55 62 63
54
62
53
61
o o
30 31
30
29
o o 4 5
4
3
36 37 44 45
36
44
35
43
12 13 20 21
12
20
11
19
52 53 60 61
52
60
51
59
o o
28 29
28
27
o o 2 3
2
1
34 35 42 43
34
42
33
41
10 11 18 19
10
18
9
17
50 51 58 59
50
58
49
57
o o
26 27
26
25
o o o 1
o 33 o 41
o 9 o 17
o 49 o 57
o o o 25
$x_1^{(R)}$
$x_{p+1}^{(B)}$
$x_{p+2}^{(B)}$
$x_2^{(R)}$
$x_3^{(R)}$
$x_{p+3}^{(B)}$
$x_{p+4}^{(B)}$
$x_4^{(R)}$
V1
V2
V3
V4
V5

Mapping into a fixed size Architecture of N Cores



case of $k = 2p/N$ even

$2k$ virtual threads

$x_l^{(R)}$ $l=(j-1)k+1,\ldots,jk$

$x_l^{(B)}$ $l=2p+(j-1)k+1,\ldots,2p+jk$

P_j $j=1,\ldots,N$

Parallel Schur Complement Iterative Solution

S1: *Solve in parallel on host* $D_R \hat{\boldsymbol{b}}_R = \boldsymbol{b}_R$

S2: *Send matrices* A_3 *and* A_4 *to GPU*

S3: *Evaluate in parallel on GPU* $\hat{\boldsymbol{b}}_B = \boldsymbol{b}_B - H_R \hat{\boldsymbol{b}}_R$

S4: *Solve in parallel with BiCGSTAB* $S\,\boldsymbol{x}_B = \hat{\boldsymbol{b}}_B$

S5: *Evaluate in parallel on GPU* $\hat{\boldsymbol{x}}_B = H_B \boldsymbol{x}_B$

S6: *Solve in parallel on host* $D_R \hat{\boldsymbol{x}}_R = \hat{\boldsymbol{x}}_B$

S7: *Evaluate in parallel on host* $\boldsymbol{x}_R = \hat{\boldsymbol{b}}_R - \hat{\boldsymbol{x}}_R$

Parallel BiCGSTAB

Choose initial approximation $x^{(0)}$ of the solution x_B

$r^{(0)} = b_B - Sx^{(0)}$

Choose $\hat{r}$ (usually $\hat{r} = r^{(0)}$)

for $i = 1, 2, \ldots$

$\rho_{i-1} = \hat{r}^T r^{(i-1)}$

if $\rho_{i-1} = 0$ method fails

if $i = 1$

$p^{(1)} = r^{(0)}$

else

$\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}} \frac{\alpha_{i-1}}{\omega_{i-1}}$

$p^{(i)} = r^{(i-1)} + \beta_{i-1} (p^{(i-1)} - \omega_{i-1} v^{(i-1)})$

endif

$v^{(i)} = S p^{(i)}$

$\alpha_i = \frac{\rho_{i-1}}{\hat{r}^T v^{(i)}}$

$s = r^{(i-1)} - \alpha_i v^{(i)}$

if $\| s \|$ is small enough **then**

$x_B^{(i)} = x_B^{(i-1)} + \alpha_i p^{(i)}$ stop

$t = S s$

$\omega_i = \frac{s^T t}{t^T t}$

$x_B^{(i)} = x_B^{(i-1)} + \alpha_i p^{(i)} + \omega_i s$

Check for Convergence

if $\omega_i = 0$ stop

$r^{(i)} = s - \omega_i t$

end

Evaluation of $\boldsymbol{t} = S\boldsymbol{p}$

S1: *Send* $\boldsymbol{p}$ *from host to GPU*

S2: *Evaluate in parallel on GPU* $\boldsymbol{t} = H_B\,\boldsymbol{p}$

S3: *Send* $\boldsymbol{t}$ *from GPU to host*

S4: *Solve in parallel on host* $D_R\boldsymbol{s} = \boldsymbol{t}$

S5: *Send* $\boldsymbol{s}$ *from host to GPU*

S6: *Evaluate in parallel on GPU* $\boldsymbol{q} = H_R\,\boldsymbol{s}$

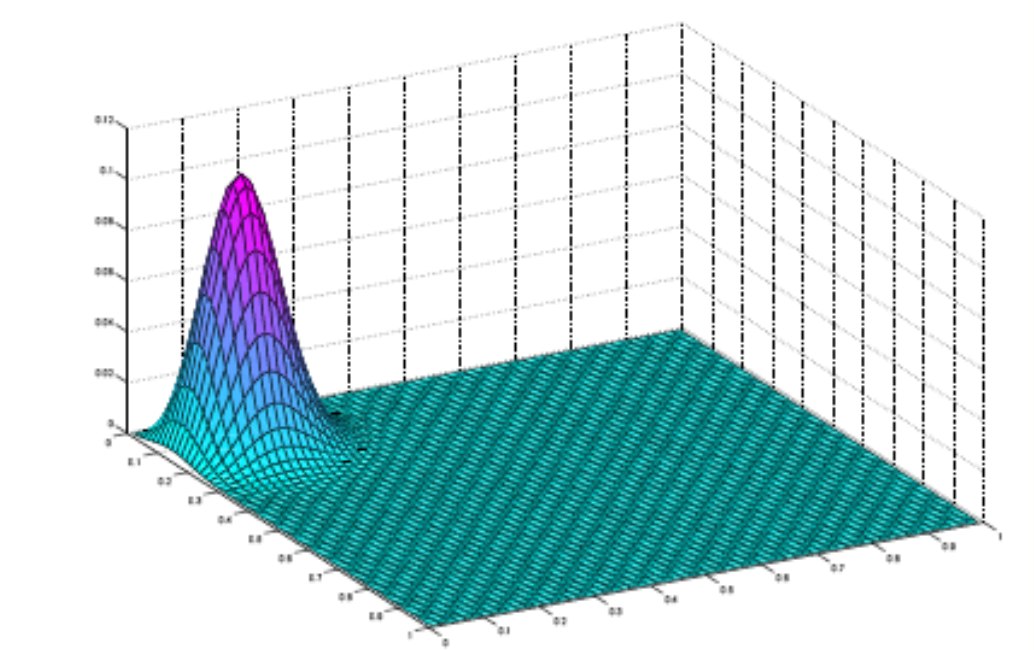
S7: *Send* $\boldsymbol{q}$ *from GPU to host*

S8: *Evaluate in parallel on host* $\boldsymbol{t} = D_B\,\boldsymbol{p} - \boldsymbol{q}$

The Dirichlet Helmholtz Problem

$$u(x,y)=10\varphi(x)\varphi(y) \quad , \quad (x,y)\in[0,1]\times[0,1]$$

with $\varphi(x)=(x^2-x)e^{-100(x-0.1)^2}$



HP SL390s – Tesla M2070 GPUs

HP SL390s

\+ 2 x

6 core Xeon@2.8GHz
24GB memory
Oracle Linux 6.3 x64
PGI 13.5 Fortran
PCI-e gen2 x16

TECHNICAL SPECIFICATIONS

	Tesla M2070 / M2075
Peak double precision floating point performance	515 Gigaflops
Peak single precision floating point performance	1030 Gigaflops
CUDA cores	448
Memory size (GDDR5)	6 GigaBytes
Memory bandwidth (ECC off)	150 GBytes/sec

PGI Accelerator Compilers

The Portland Group

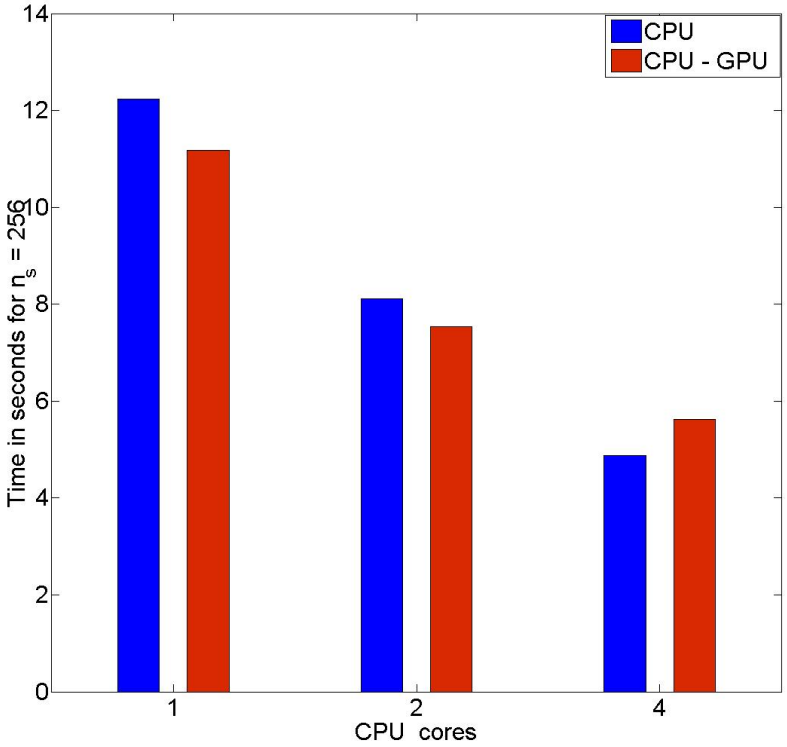
OpenACC. DIRECTIVES FOR ACCELERATORS

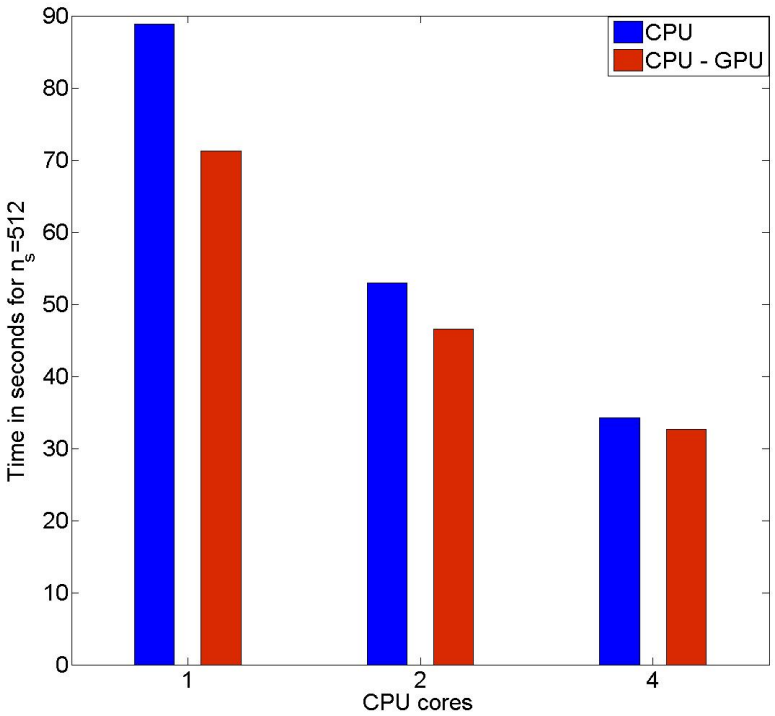
OpenMP

Realization on HP SL390s Tesla GPU machine
Iterations / Error measurements

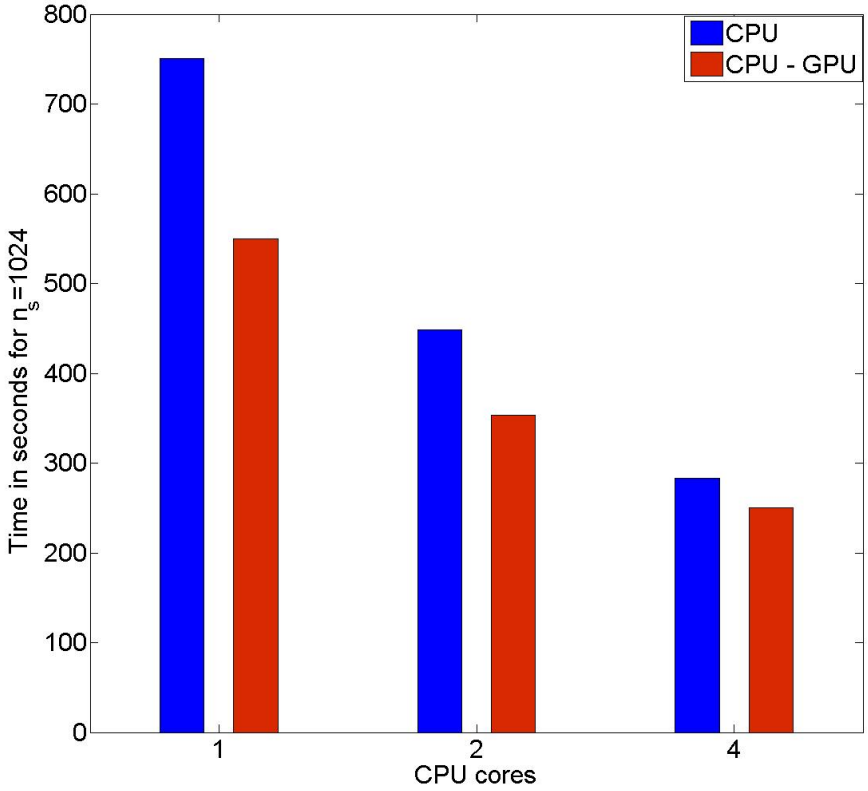
n_s	BiCGSTAB Iterations	$\| \mathbf{b} - A\mathbf{x}_n \|_2$
256	294	6.06e-11
512	589	2.85e-11
1024	1161	1.39e-11
2048	3726	9.59e-12

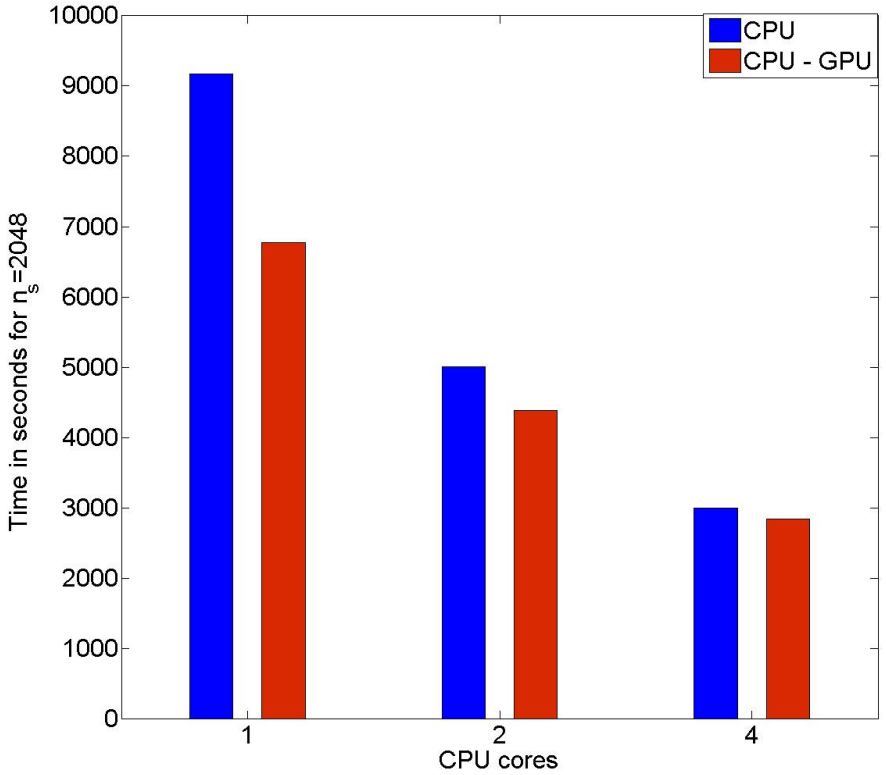
Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης



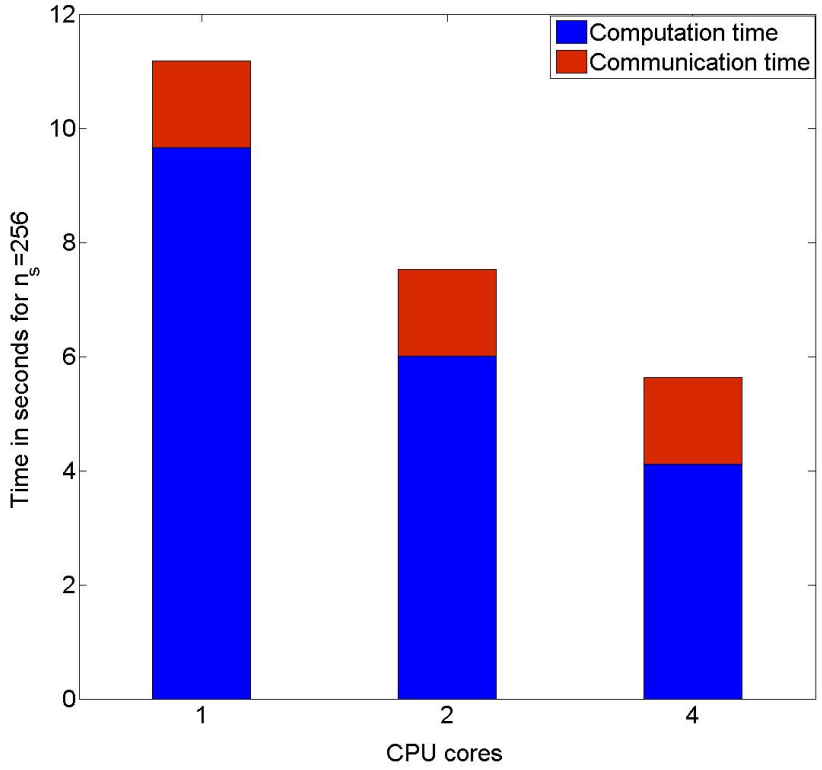


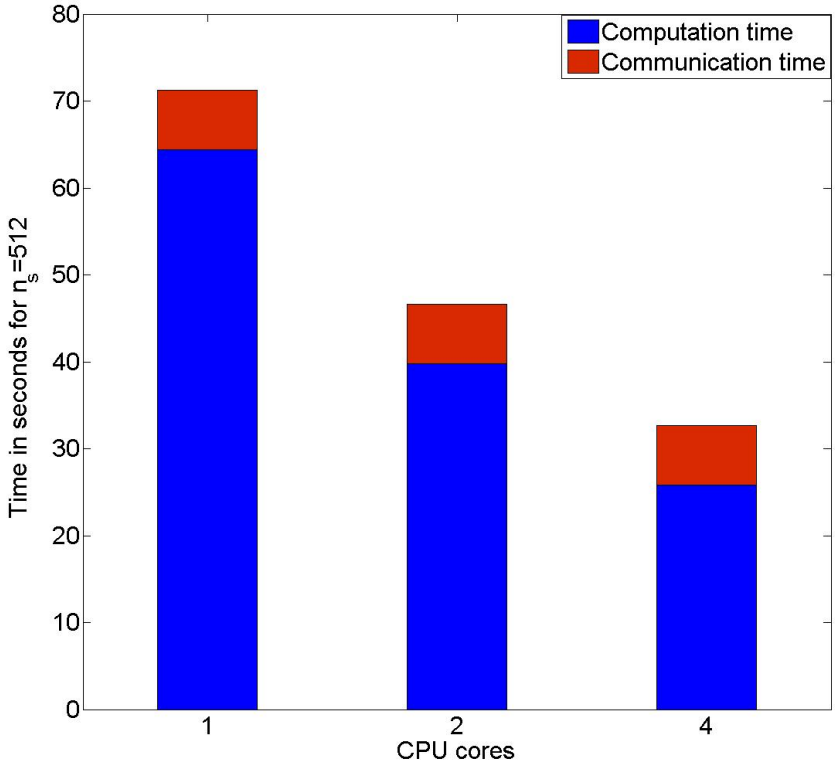
Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης



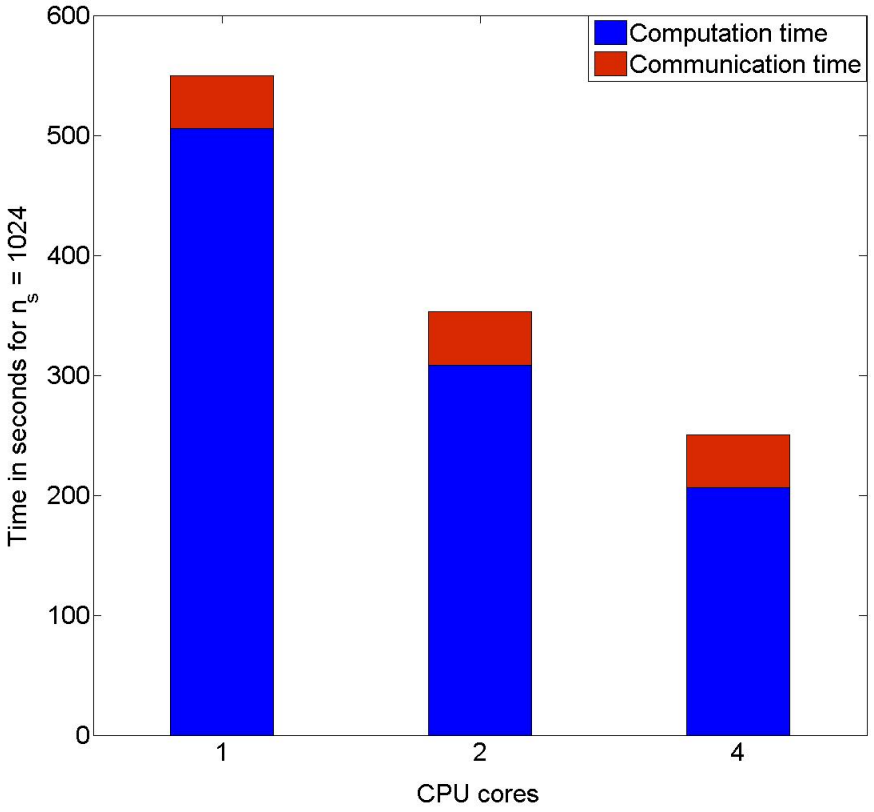


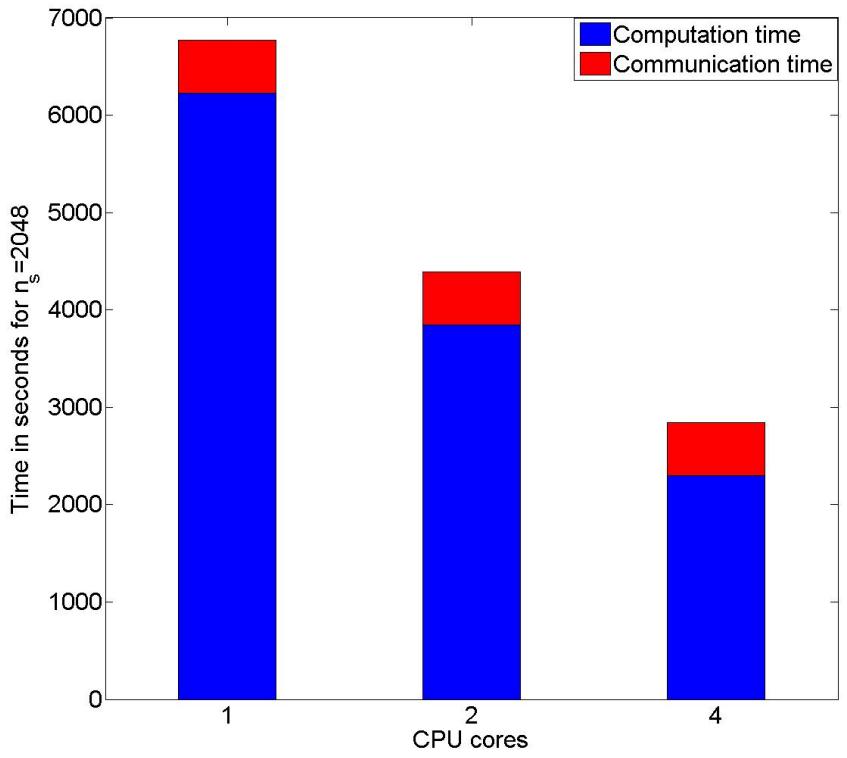
Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης



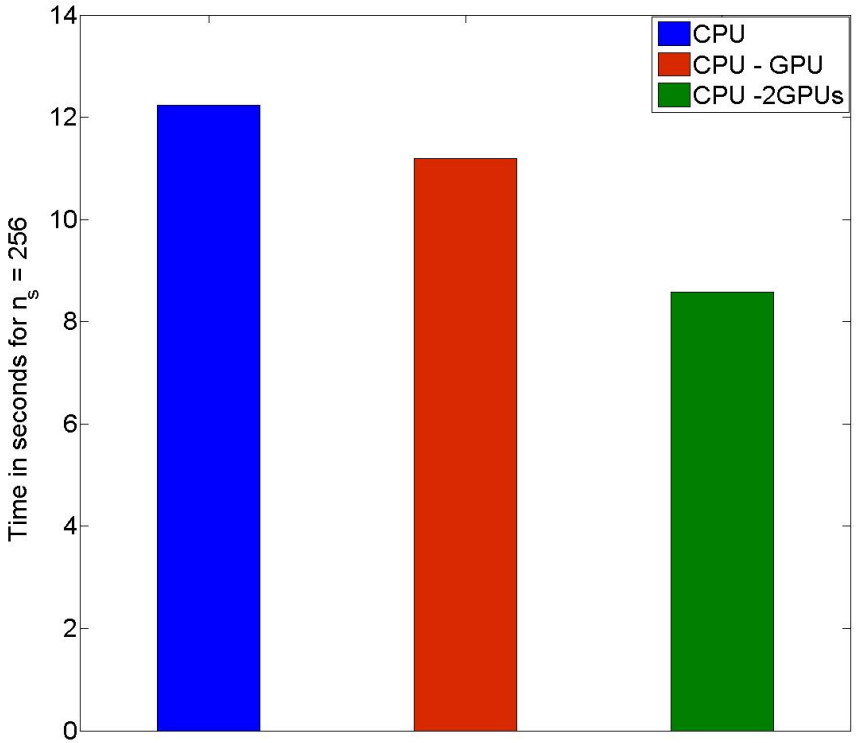


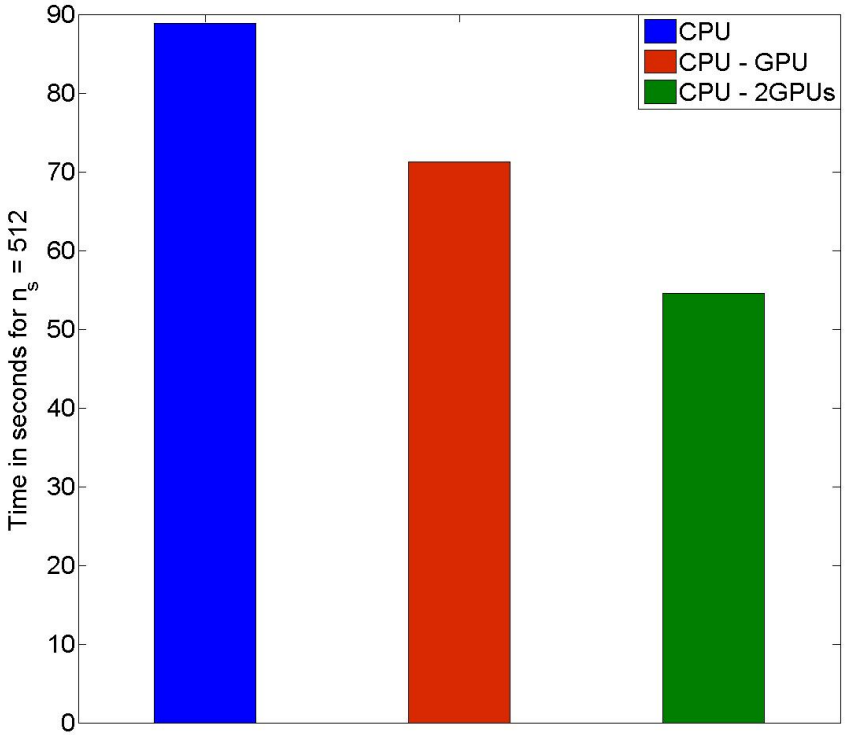
Realization on HP SL390s Tesla GPU machine
Time measurements



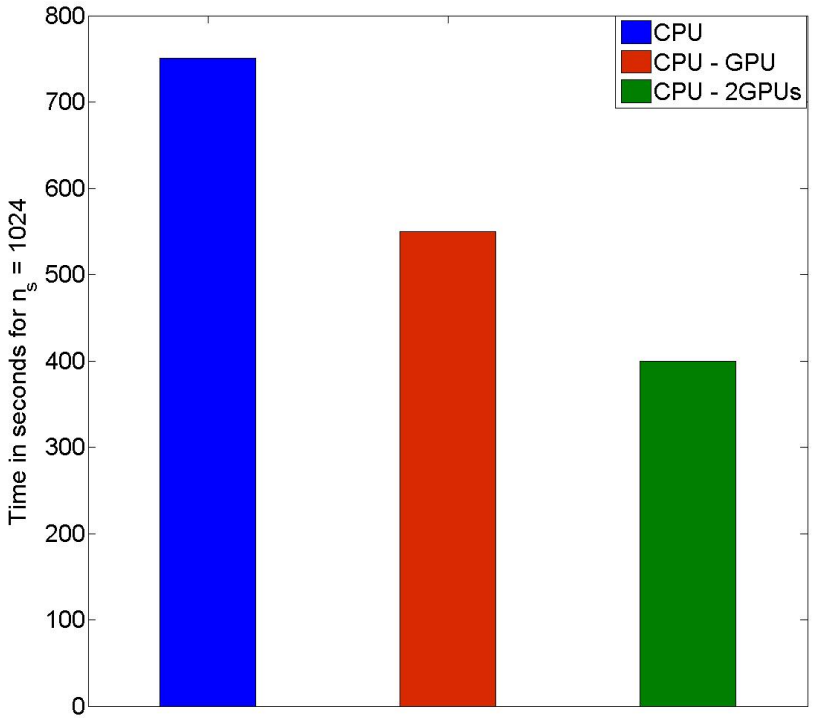


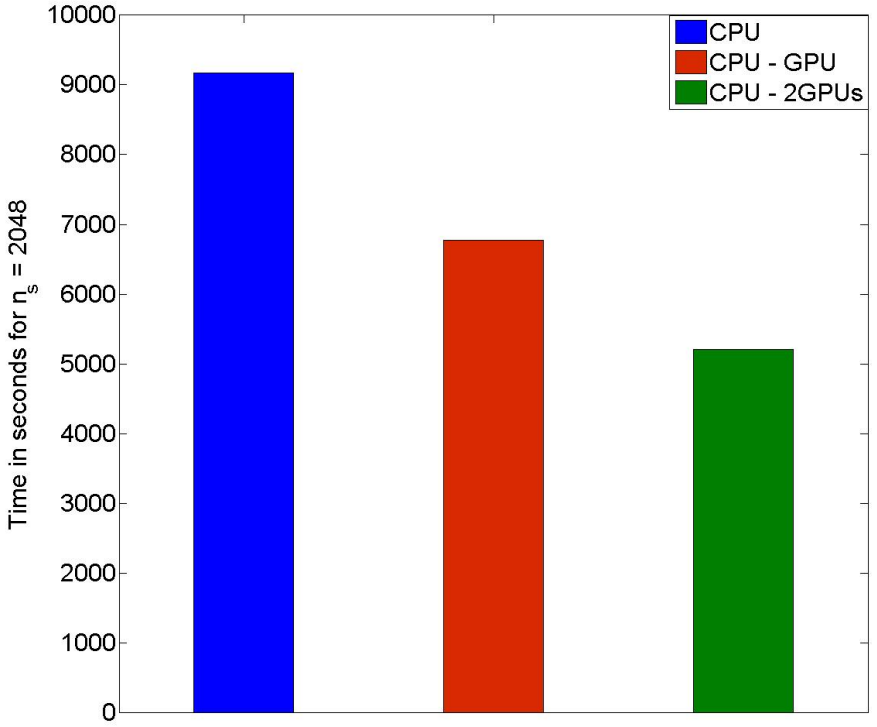
Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης





Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης





Συμπεράσματα

- Σχεδίαση νέου παράλληλου αλγορίθμου της Schur complement μεθόδου με την BiCGSTAB για την Hermite Collocation αριθμητική μέθοδο.
- Υλοποίηση σε αρχιτεκτονικές υπολογισμών Κοινής Μνήμης για multi-core μηχανήματα με επιταχυντές (GPUs) .
- Επιτάχυνση μέχρι 30%.

Επόμενα βήματα

- Κατασκευή αποδοτικού αλγορίθμου της μεθόδου Discontinuous Hermite Collocation για αρχιτεκτονικές με Multiprocessor /Grid μηχανήματα με επιταχυντές υπολογισμών.

Semi-Implicit Discontinuous Hermite Collocation Method

DHC system of ODEs

$$\left(C_x^0 \otimes C_y^0\right) \dot{\mathbf{a}} = \left(D_x C_x^2 \otimes C_y^0\right) \mathbf{a} + \left(D_x C_x^0 \otimes C_y^2\right) \mathbf{a}$$

where C_x^* and C_u^* denotes the $1d$ Discontinuous and Continuous Collocation matrices

$$D_x = diag(\gamma, \ldots, \gamma, 1, \ldots, 1, \gamma, \ldots, \gamma)$$

$$A_0 := C_x^0 \otimes C_y^0 \qquad B := D_x C_x^2 \otimes C_y^0 + D_x C_x^0 \otimes C_y^2$$

$$A_0 \dot{\mathbf{a}} = B\mathbf{a}$$

Semi-Implicit Discontinuous Hermite Collocation Method

$$A_0 \dot{\mathbf{a}} = B\mathbf{a}$$

DHC method coupled with an optimal 2-step and 3rd order DIRK

$$A\,\mathbf{a}^{(n,1)} = A_0\,\mathbf{a}^{(n)}$$
$$A\,\mathbf{a}^{(n,2)} = A_0\,\mathbf{a}^{(n)} + \tau(1-2\lambda)B\,\mathbf{a}^{(n,1)}$$
$$A_0\,\mathbf{a}^{(n+1)} = A_0\,\mathbf{a}^{(n)} + \frac{\tau}{2}\left[B\,\mathbf{a}^{(n,1)} + B\,\mathbf{a}^{(n,2)}\right]$$

where $A := A_0 - \tau\lambda B$, τ is the time spacing and $\lambda = \frac{1}{2} + \frac{\sqrt{3}}{2}$

Semi-Implicit Discontinuous Hermite Collocation Method

INPUT $\boldsymbol{a}_{old}$, B , A_0 , A , A_b

for $t = \tau$ *to* t_{max} *with step* τ *do*

compute $\boldsymbol{a}_0 = A_0\, \boldsymbol{a}_{old}$

if $t \leq 2\tau$ *then*

solve $A_b\, \boldsymbol{a}_{new} = \boldsymbol{a}_0$ *with BiCGSTAB*

else

solve $A\, \boldsymbol{a}_1 = \boldsymbol{a}_0$ *with BiCGSTAB*

compute $\boldsymbol{a}_3 = \boldsymbol{a}_0 - \tau \frac{\sqrt{3}}{3} B\, \boldsymbol{a}_1$

solve $A\, \boldsymbol{a}_2 = \boldsymbol{a}_3$ *with BiCGSTAB*

compute $\boldsymbol{a}_2 = \boldsymbol{a}_0 + \frac{\tau}{2} B\, (\boldsymbol{a}_1 + \boldsymbol{a}_2)$

solve $A_0\, \boldsymbol{a}_{new} = \boldsymbol{a}_2$ *with BiCGSTAB*

endif

compute $\boldsymbol{a}_{old} = \boldsymbol{a}_{new}$

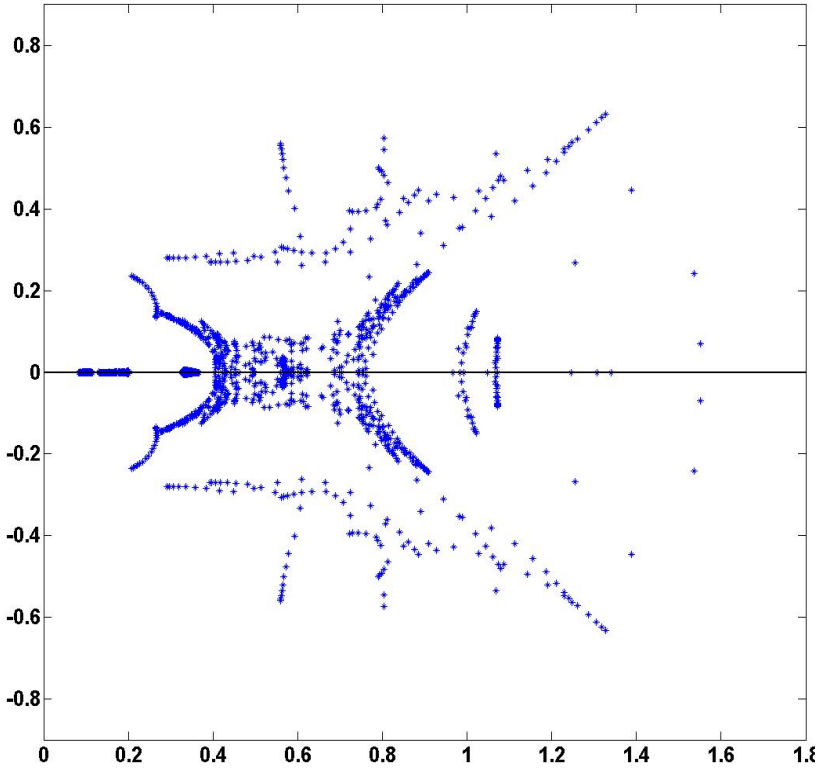
enddo

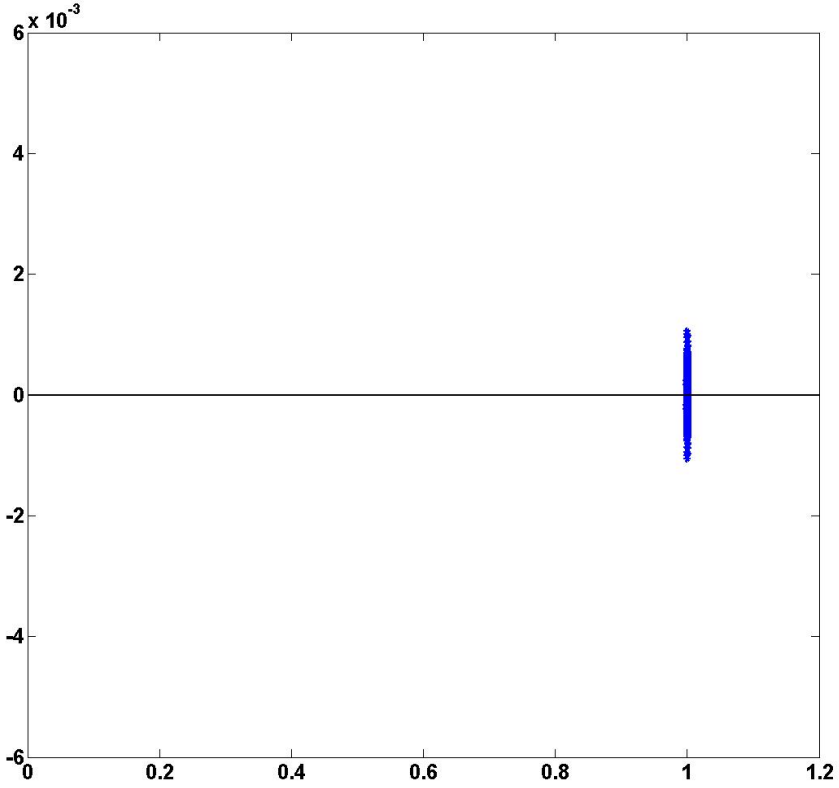
Preconditioned BiCGSTAB Method...

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
Choose $\tilde{r}$ (for example, $\tilde{r} = r^{(0)}$)
for $i = 1, 2, \ldots$
  $\rho_{i-1} = \tilde{r}^T r^{(i-1)}$
  if $\rho_{i-1} = 0$ **method fails**
  if $i = 1$
    $p^{(i)} = r^{(i-1)}$
  else
    $\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$
    $p^{(i)} = r^{(i-1)} + \beta_{i-1}(p^{(i-1)} - \omega_{i-1} v^{(i-1)})$
  endif
  solve $M\hat{p} = p^{(i)}$
  $v^{(i)} = A\hat{p}$
  $\alpha_i = \rho_{i-1}/\tilde{r}^T v^{(i)}$
  $s = r^{(i-1)} - \alpha_i v^{(i)}$
  check norm of s; if small enough: set $x^{(i)} = x^{(i-1)} + \alpha_i \hat{p}$ and stop
  solve $M\hat{s} = s$
  $t = A\hat{s}$
  $\omega_i = t^T s / t^T t$
  $x^{(i)} = x^{(i-1)} + \alpha_i \hat{p} + \omega_i \hat{s}$
  $r^{(i)} = s - \omega_i t$
  check convergence; continue if necessary
  for continuation it is necessary that $\omega_i \neq 0$
end

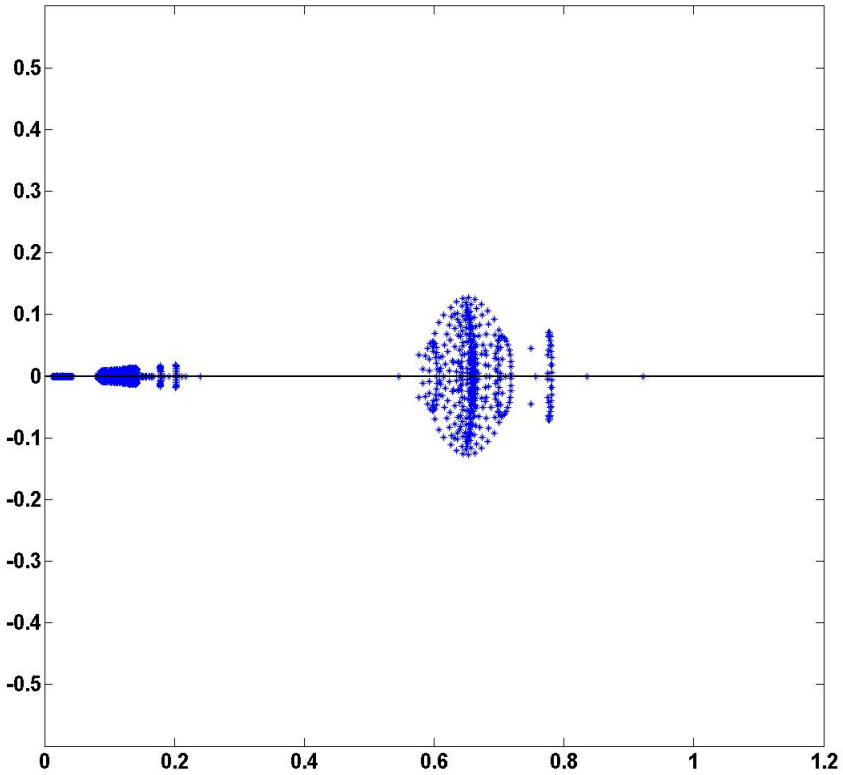
$A \sim M = ilu(A)$

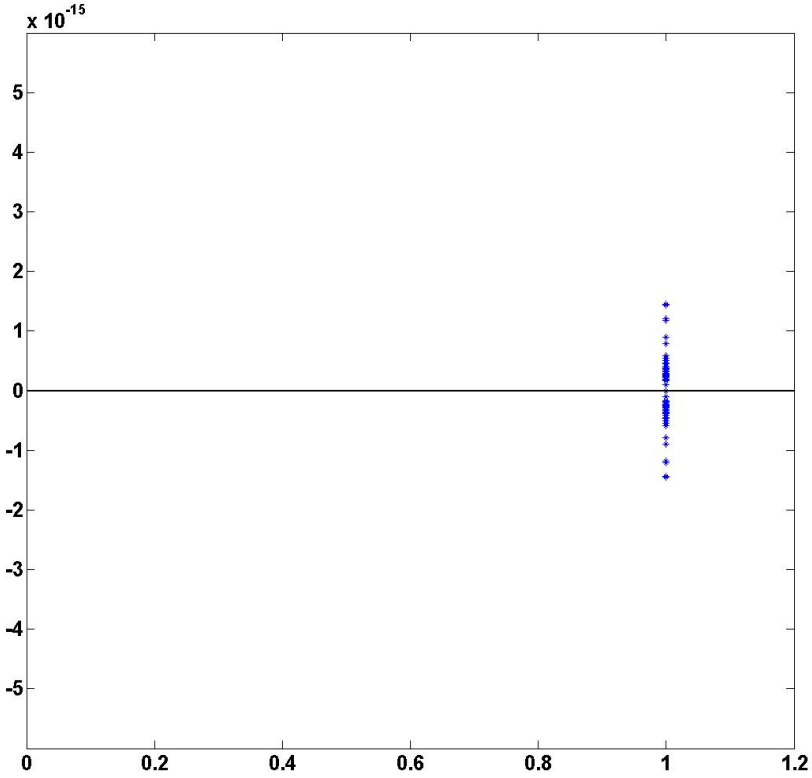
Matrix A


0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


x 10^-3
6
4
2
0
-2
-4
-6
0
0.2
0.4
0.6
0.8
1
1.2

Matrix A_0


0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.2
0.4
0.6
0.8
1
1.2


x 10^-15
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.2
0.4
0.6
0.8
1
1.2

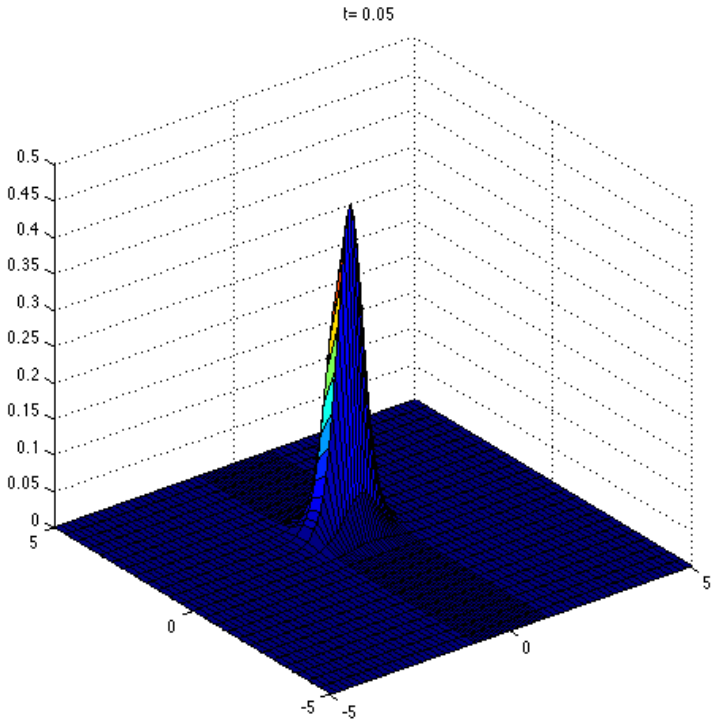
Parallel BiCGSTAB

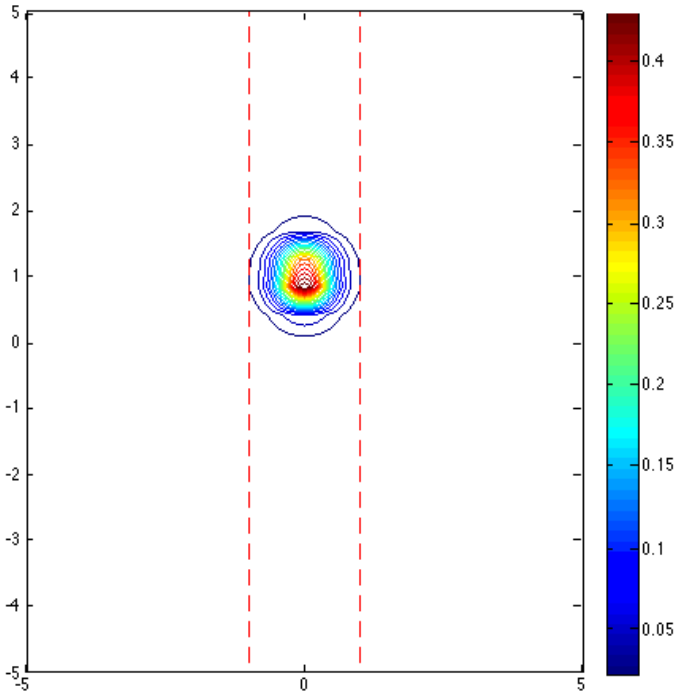
- All basic linear algebra operations are performed on the GPU

- The preconditioning procedure $M z = t$ with $M=LU$ is performed on the CPU

  Step 1 : GPU sends to CPU vector t
  Step 2 : CPU solution of $L y = t$
  Step 3 : CPU solution of $U z = y$
  Step 4 : CPU sends to GPU vector z

The test problem ...





HP SL390s - Tesla M2070 GPUs

HP SL390s

+

6 core Xeon@2.8GHz
24GB memory
Oracle Linux 6.3 x64
PGI 14.5 Cuda Fortran
Cuda toolkit 6.0
PCI-e gen2 x16

The Portland Group

TECHNICAL SPECIFICATIONS

	Tesla M2070 / M2075
Peak double precision floating point performance	515 Gigaflops
Peak single precision floating point performance	1030 Gigaflops
CUDA cores	448
Memory size (GDDR5)	6 GigaBytes
Memory bandwidth (ECC off)	150 GBytes/sec

PGI Accelerator Compilers

NVIDIA® CUDA™
Parallel Programming and Computing Platform
NVIDIA.

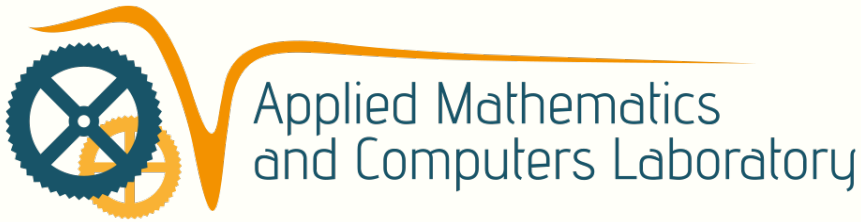
Development tools …

- MatLab R2012b
- PGI 14.5 Cuda Fortran
  Cuda toolkit 6.0 – cuBLAS - cuSRARSE for GPU operations
  SparseKit for CPU operations

Development tools ...

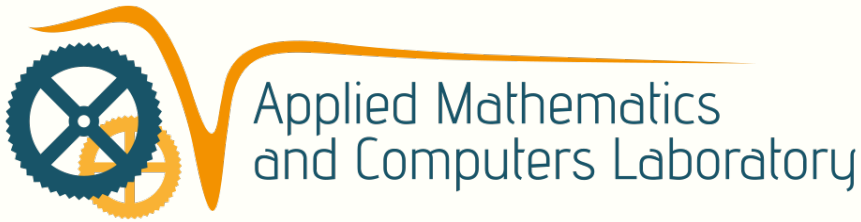
```
   module bicgstabGPU
   contains
    subroutine bicgstabGPU(n,x,b,A,L,U,iA,iL,jA,jL,jU,
+                          nzA,descrA,handle,error,r,rh,pi,ph,
+                          t,s,sh,ui,istep,temp,ttt)
 implicit real*8 (a-h,o-z)
 real*8 ttt(n),L(*),U(*),temp(n)
 integer*8 handle,descrA,iA,jA,A,
+           x,b,r,rh,pi,ph,s,sh,ui,t
 integer   cusparse_dcsrmv,iL(n+1),jL(*),jU(*),
+   cuda_memcpy_c2fort_real,cuda_memcpy_fort2c_real


 imaxstep=istep
 tol=error
 istep=0
 dnrmb=cublas_dnrm2(n,b,1)
 call cublas_dcopy(n,b,1,x,1)
 istatus=cusparse_dcsrmv(handle,0,n,n,nzA,1.0d0,descrA,A,iA,jA,
+         x,0.0d0,r)
```



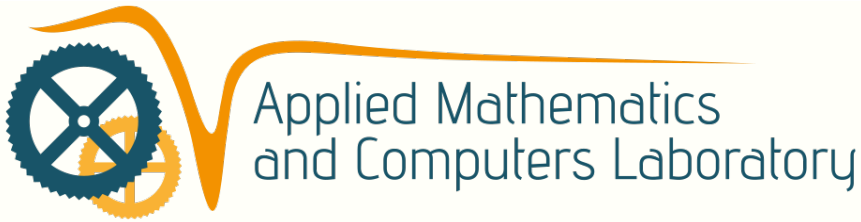
Development tools ...

```
      call cublas_dscal(n,-1.0d0,r,1)
      call cublas_daxpy(n,1.0d0,b,1,r,1)
      call cublas_dcopy(n,r,1,rh,1)
999   continue
      istep=istep+1
      if (istep.gt.1) roip2=roip1
       roip1=cublas_ddot(n,rh,1,r,1)
      if (istep.eq.1) then
       call cublas_dcopy(n,r,1,pi,1)
      else
       bi=(roip1/roip2)*(ai/wi)
       call cublas_daxpy(n,-wi,ui,1,pi,1)
       call cublas_dcopy(n,r,1,t,1)
       call cublas_daxpy(n,bi,pi,1,t,1)
       call cublas_dcopy(n,t,1,pi,1)
      endif
c-------------------------------------
```



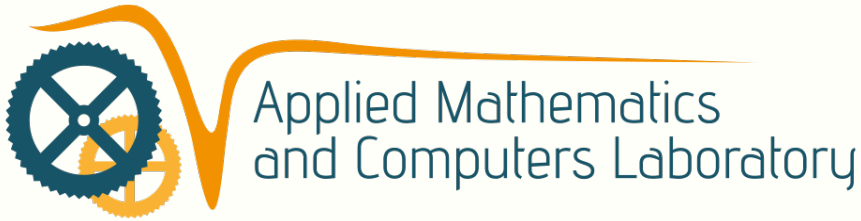
Development tools ...

```
      icudaStat = cuda_memcpy_c2fort_real(ttt,pi,n*8,2)
      call lsol(n,temp,ttt,L,jL,iL)
      call udsol(n,ttt,temp,U,jU)
      icudaStat3 = cuda_memcpy_fort2c_real(ph,ttt,n*8,1)
C-------------------------------------
      istatus=cusparse_dcsrmv(handle,0,n,n,nzA,1.0d0,descrA,A,iA,jA,
     +          ph,0.0d0,ui)
      ai=roip1/cublas_ddot(n,rh,1,ui,1)
      call cublas_dcopy(n,r,1,s,1)
      call cublas_daxpy(n,-ai,ui,1,s,1)
      if (cublas_dnrm2(n,s,1).lt.1.0d-25) then
      call cublas_daxpy(n,ai,ph,1,x,1)
       print*,'BiCGSTAB incomplete'
      else
C-------------------------------------
       icudaStat = cuda_memcpy_c2fort_real(ttt,s,n*8,2)
       call lsol(n,temp,ttt,L,jL,iL)
       call udsol(n,ttt,temp,U,jU)
       icudaStat3 = cuda_memcpy_fort2c_real(sh,ttt,n*8,1)
```



Development tools ...

```
      istatus=cusparse_dcsrmv(handle,0,n,n,nzA,1.0d0,descrA,A,iA,jA,
     +          sh,0.0d0,t)
      wi=cublas_ddot(n,t,1,s,1)/cublas_ddot(n,t,1,t,1)
      call cublas_daxpy(n,ai,ph,1,x,1)
      call cublas_daxpy(n,wi,sh,1,x,1)
      call cublas_daxpy(n,-wi,t,1,s,1)
      call cublas_dcopy(n,s,1,r,1)
       dnrmr=cublas_dnrm2(n,s,1)
       error=dnrmr/dnrmb
c      print*,error,istep
      if (wi.ne.0.0d0.and.error.gt.tol.and.istep.lt.imaxstep)
     + goto 999
      endif
      return
      end
      end  module
```

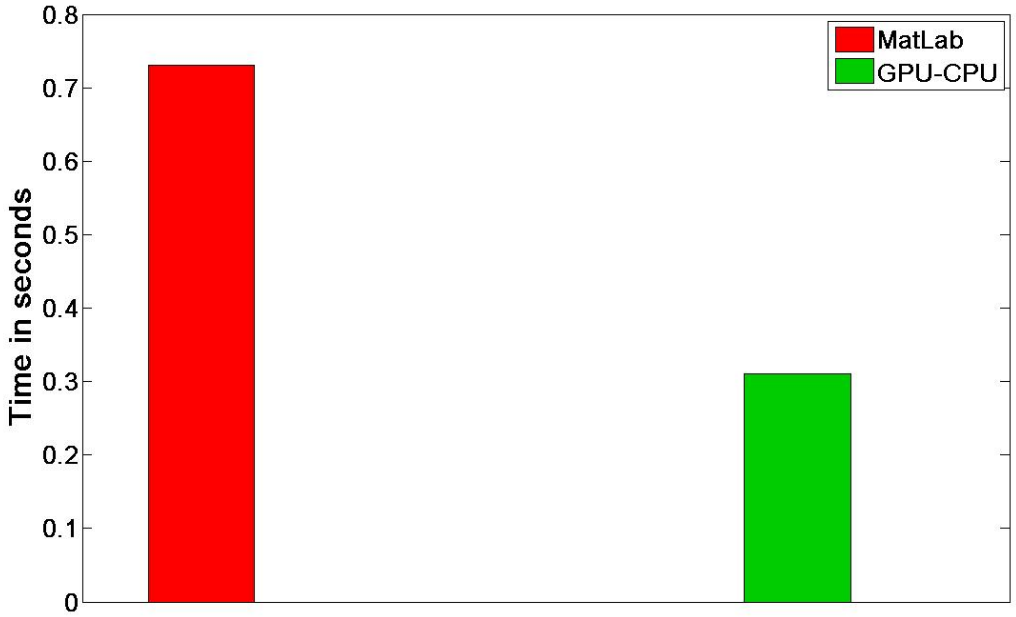


Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης

Finite Elements : 400

Unknowns : 1.600

Dofs : 6.400

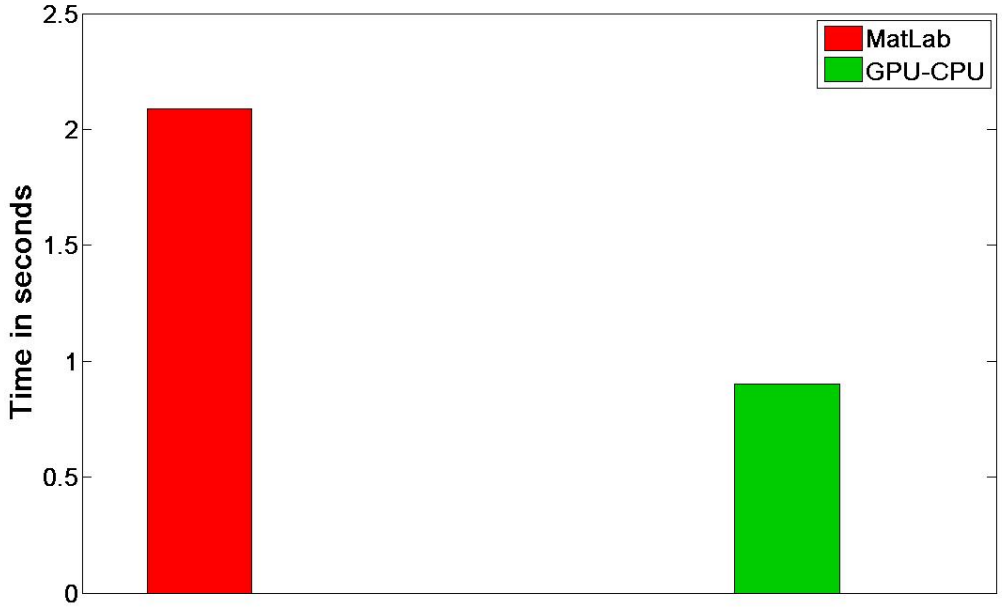


Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης

Finite Elements : 1.600

Unknowns : 6.400

Dofs : 25.600

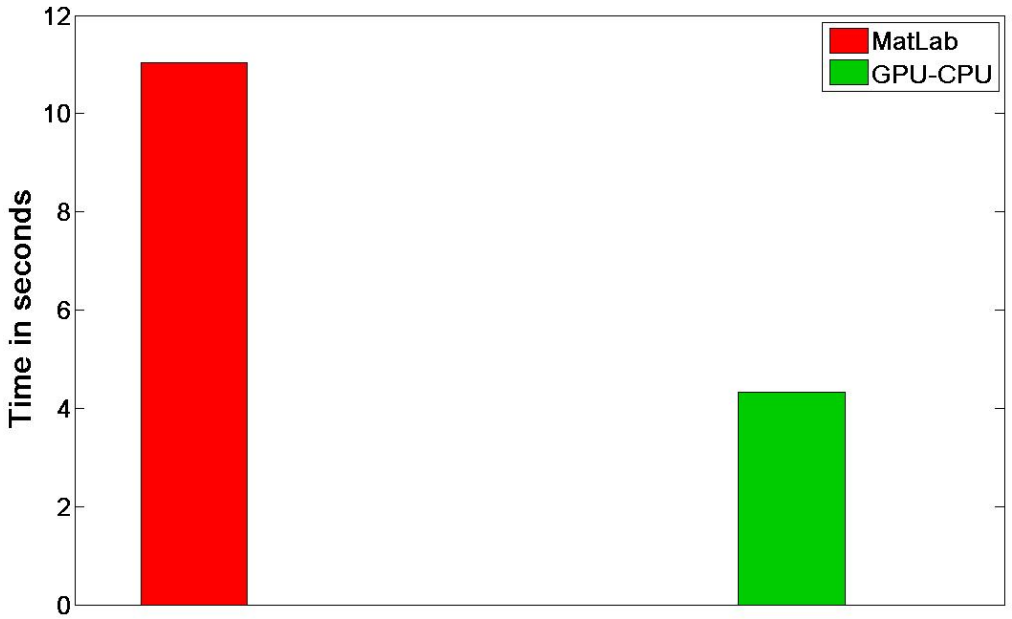


Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης

Finite Elements : 6.400

Unknowns : 25.600

Dofs : 102.400

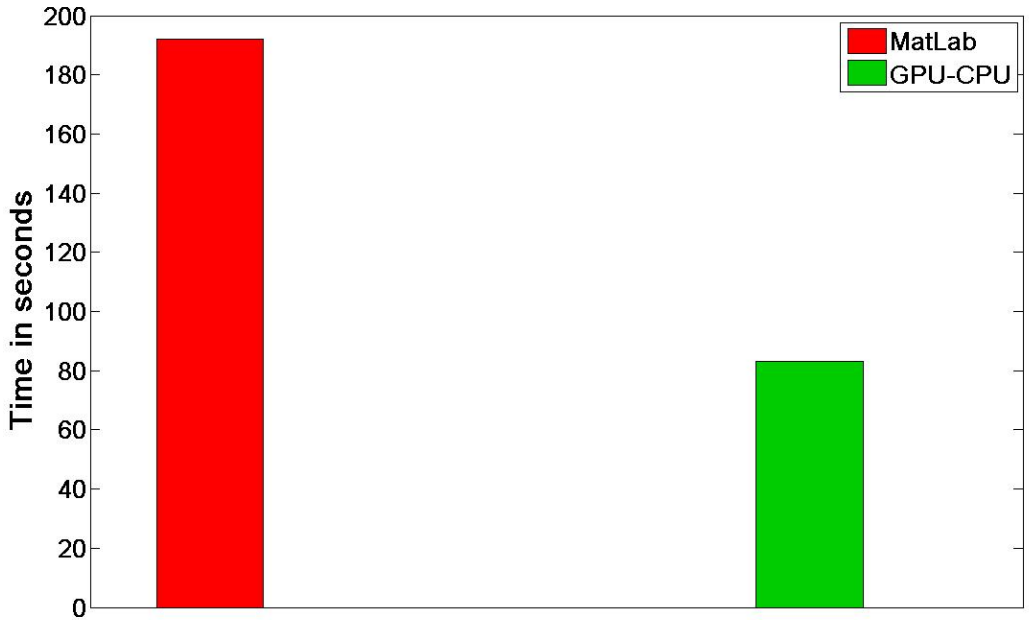


Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης

Finite Elements : 25.600

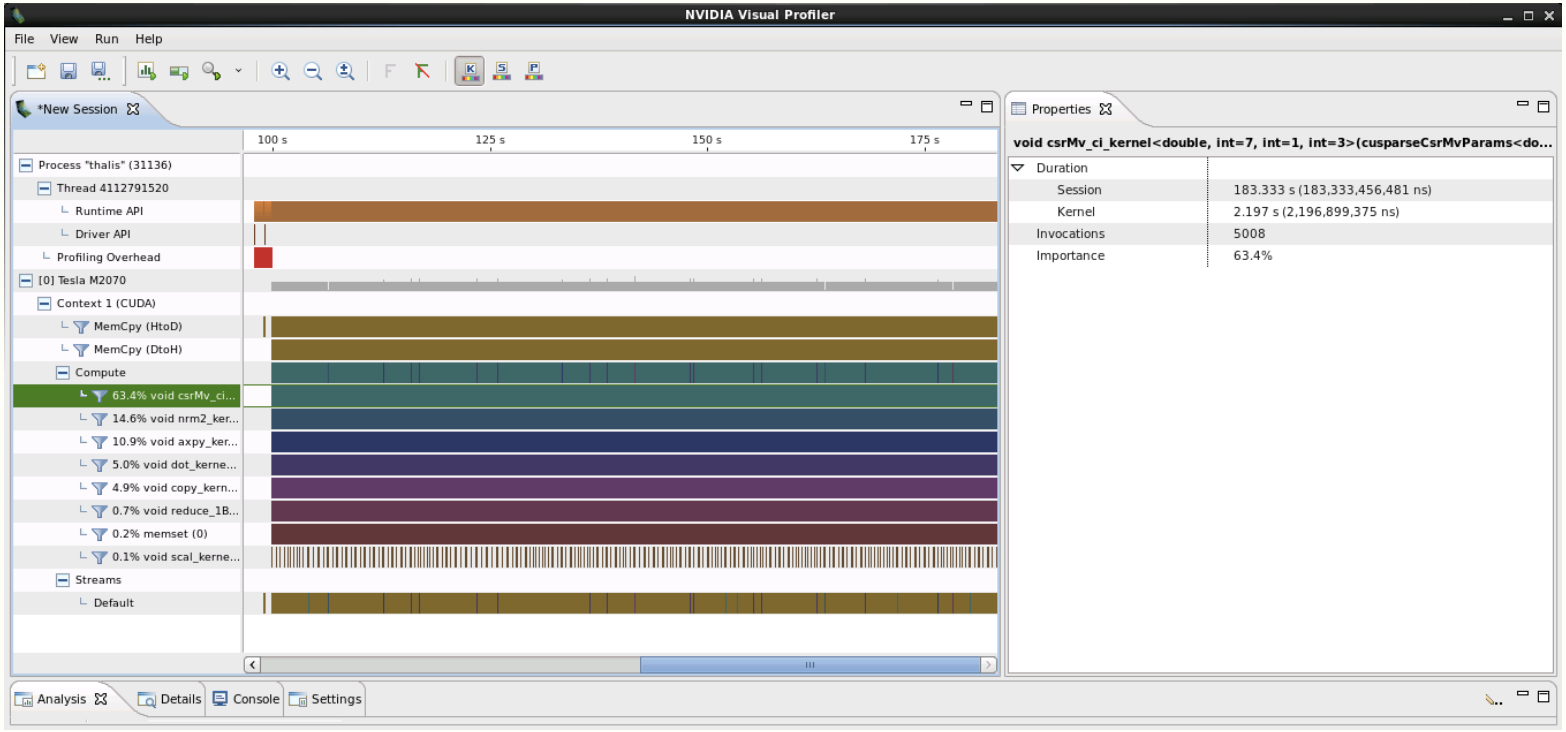
Unknowns : 102.400

Dofs : 409.600



Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης

Finite Elements : 25.600
Unknowns : 102.400
Dofs : 409.600

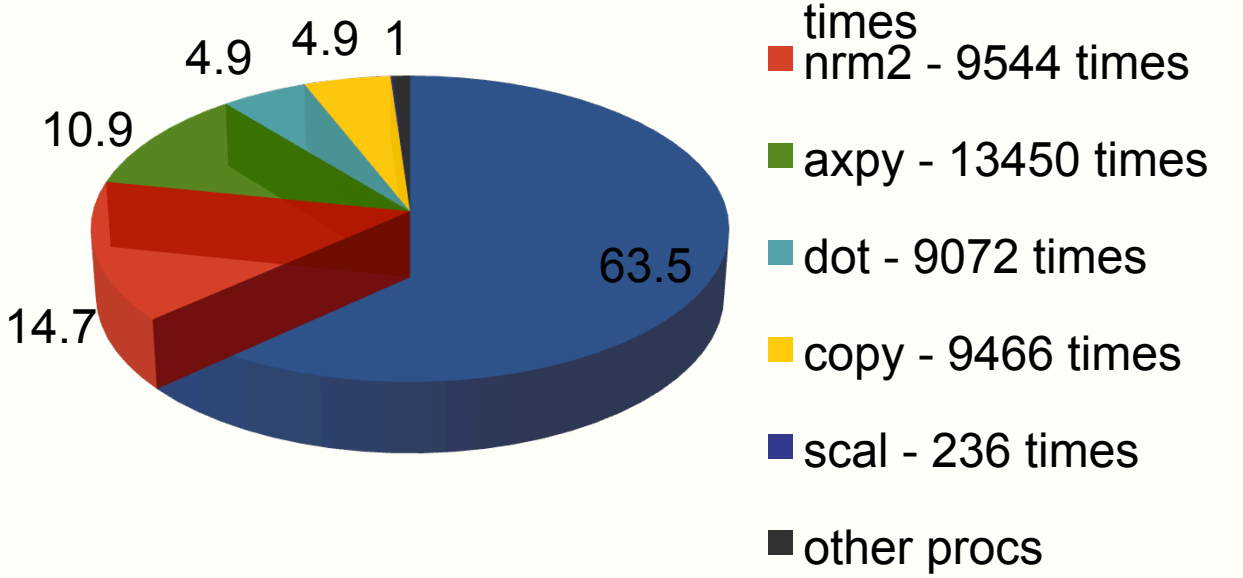


Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης

Finite Elements : 25.600

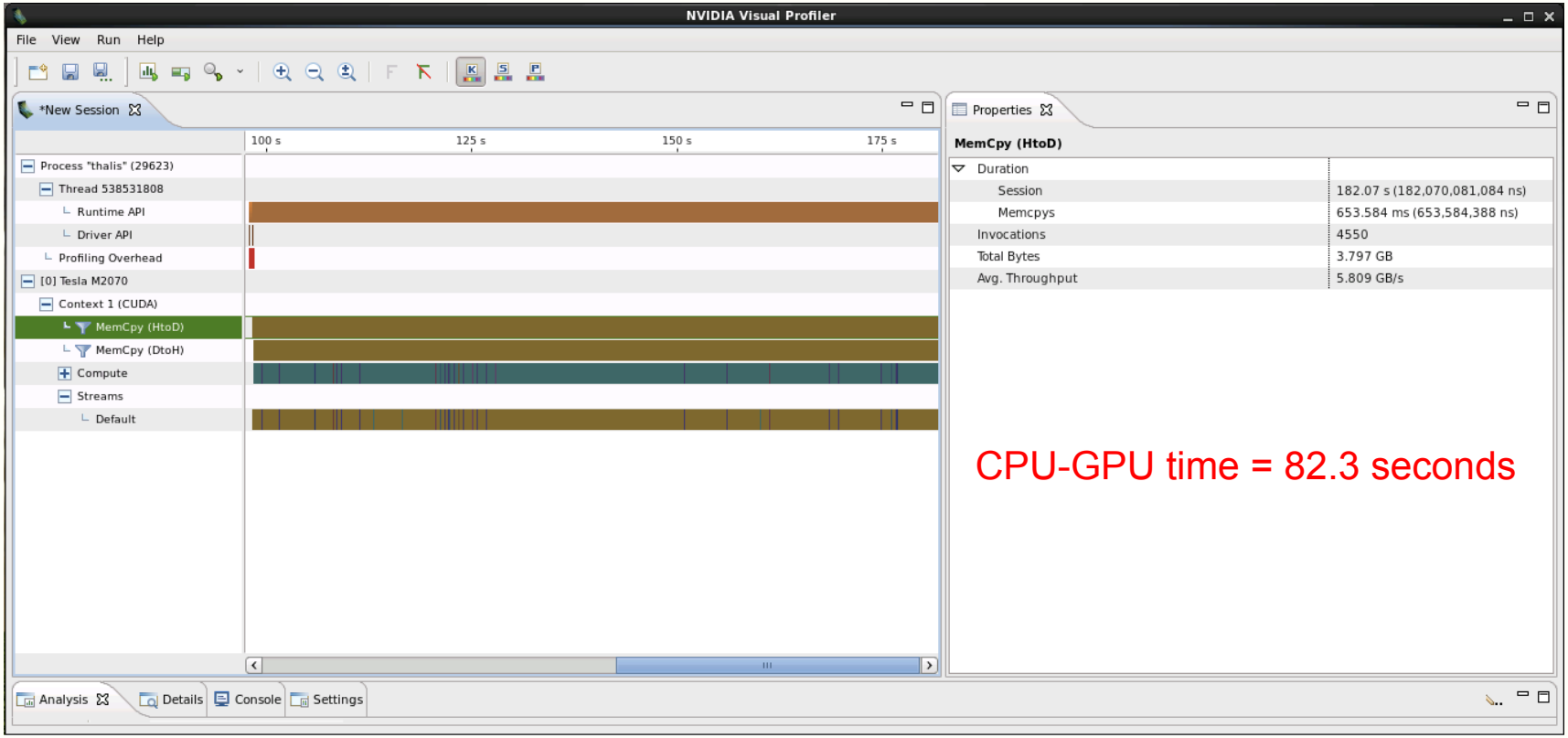
Unknowns : 102.400

Dofs : 409.600



Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης

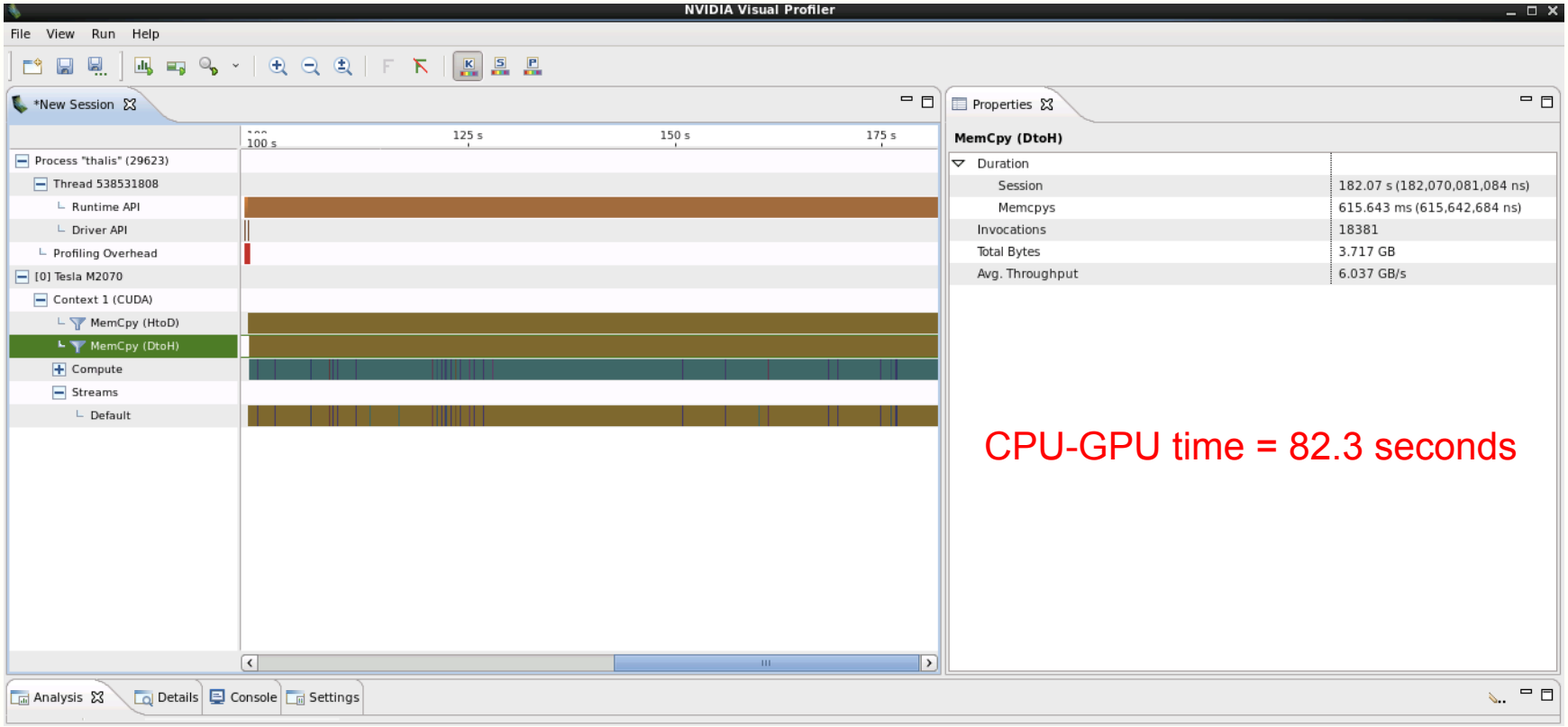
Finite Elements : 25.600 Unknowns : 102.400 Dofs : 409.600



CPU-GPU time = 82.3 seconds

Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις χρόνου εκτέλεσης

Finite Elements : 25.600 Unknowns : 102.400 Dofs : 409.600



CPU-GPU time = 82.3 seconds

Συμπεράσματα

- Σχεδίαση νέου παράλληλου αλγορίθμου της αριθμητικής μεθόδου Discontinuous Hermite Collocation.
- Υλοποίηση σε αρχιτεκτονικές υπολογισμών Κοινής Μνήμης για μηχανήματα με επιταχυντές (GPUs) .
- Παρατηρήσαμε επιτάχυνση **2x** για πυκνές διακριτοποιήσεις σε σχέση με την MatLab multithread υλοποίηση.

Semi-Implicit Discontinuous Hermite Collocation Method στην περίπτωση θεραπείας

Create matrices $A_0, A_u, A_c, A_r, A_{rc}, B_u, B_c, B_r, B_{rc}$ *and initial vector* $\boldsymbol{a_{old}}$

for $t = dt$ **to** t_{max} **with time step** dt

Compute $a_0 = A_0 \boldsymbol{a_{old}}$

if *radiotherapy* **then**

Solve $A_r \, \boldsymbol{a_1} = \boldsymbol{a_0}$

Compute $\boldsymbol{a_0} = \boldsymbol{a_0} - dt\frac{\sqrt{3}}{3} B_r \boldsymbol{a_1}$

Solve $A_r \, \boldsymbol{a_2} = \boldsymbol{a_0}$

Compute $\boldsymbol{a_2} = \boldsymbol{a_0} + \frac{dt}{2} B_r (\boldsymbol{a_1} + \boldsymbol{a_2})$

else if *chemotherapy* **then**

Solve $A_c \, \boldsymbol{a_1} = \boldsymbol{a_0}$

Compute $\boldsymbol{a_0} = \boldsymbol{a_0} - dt\frac{\sqrt{3}}{3} B_c \boldsymbol{a_1}$

Solve $A_c \, \boldsymbol{a_2} = \boldsymbol{a_0}$

Compute $\boldsymbol{a_2} = \boldsymbol{a_0} + \frac{dt}{2} B_c (\boldsymbol{a_1} + \boldsymbol{a_2})$

else if *radiotherapy* **and** *chemotherapy* **then**

Solve $A_{rc} \, \boldsymbol{a_1} = \boldsymbol{a_0}$

Compute $\boldsymbol{a_0} = \boldsymbol{a_0} - dt\frac{\sqrt{3}}{3} B_{rc} \boldsymbol{a_1}$

Solve $A_{rc} \, \boldsymbol{a_2} = \boldsymbol{a_0}$

Compute $\boldsymbol{a_2} = \boldsymbol{a_0} + \frac{dt}{2} B_{rc} (\boldsymbol{a_1} + \boldsymbol{a_2})$

else

Solve $A_u \, \boldsymbol{a_1} = \boldsymbol{a_0}$

Compute $\boldsymbol{a_0} = \boldsymbol{a_0} - dt\frac{\sqrt{3}}{3} B_u \boldsymbol{a_1}$

Solve $A_u \, \boldsymbol{a_2} = \boldsymbol{a_0}$

Compute $\boldsymbol{a_2} = \boldsymbol{a_0} + \frac{dt}{2} B_u (\boldsymbol{a_1} + \boldsymbol{a_2})$

endif

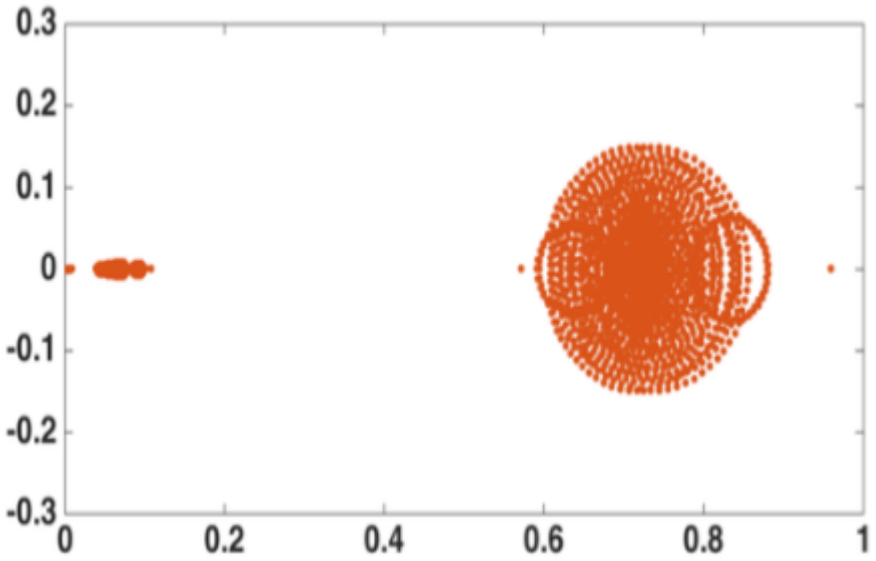
Solve $A_0 \, \boldsymbol{a_{new}} = \boldsymbol{a_2}$

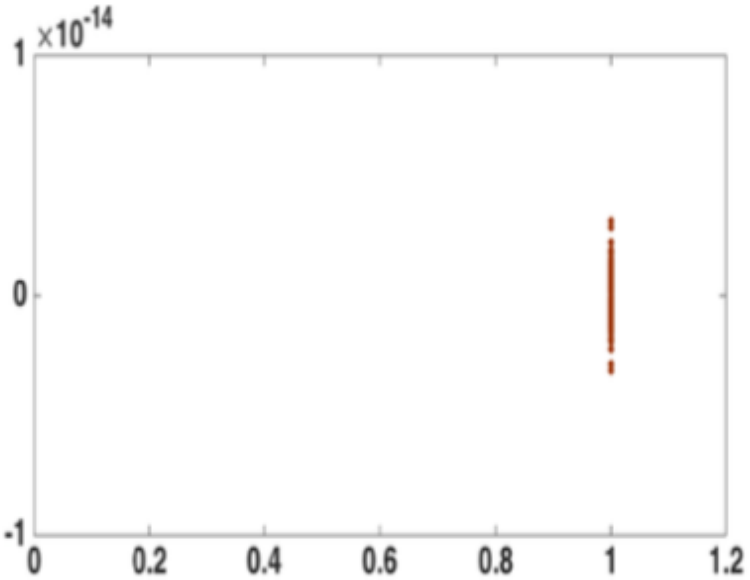
Compute $\boldsymbol{a_{old}} = \boldsymbol{a_{new}}$

endfor

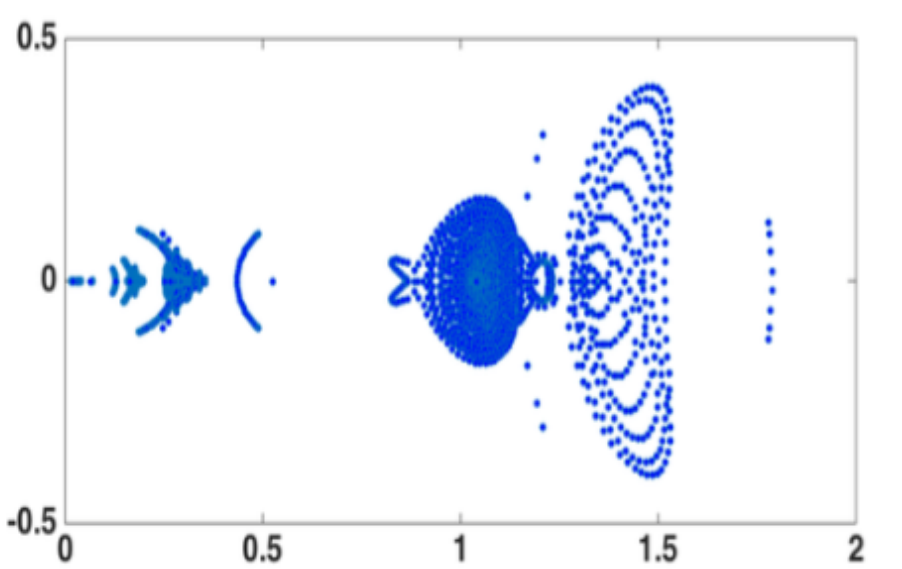
Eigenvalue Distribution ...

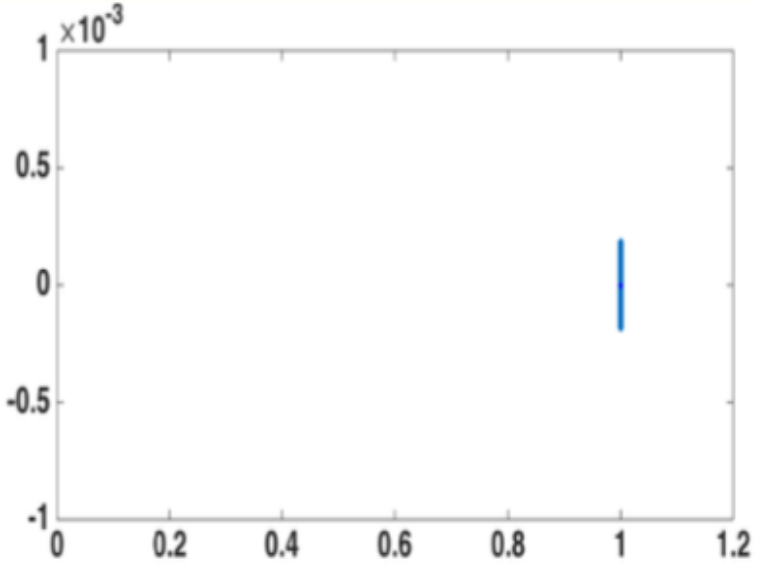
Matrix A_0





Matrix A_{rc}





Υλοποίηση σε HP SL390s Tesla GPU μηχάνημα
Μετρήσεις Speedup

n_s	MatLab - CPU	MatLab - GPU	CPU - GPU
40 x 40	5.2	5.6	1.1
80 x 80	5.9	7.9	1.4
160 x 160	5.3	6.6	1.3

Δημοσιεύσεις

- Athanasakis I., Vilanakis N. and Mathioudakis E., "Solving Discontinuous Collocation Equations for a Class of Brain Tumor Models on GPUs", Lecture Notes in Engineering and Computer Science, (2217), pp. 529-534, IAENG, 2015.

- Vilanakis N, Mathioudakis E, "Parallel iterative solution of the Hermite Collocation equations on GPUs II", Journal of Physics: Conference Series, vol. 490, 012097, 2014.

- Mathioudakis E., Vilanakis N., Papadopoulou E., Saridakis Y., "Parallel iterative solution of the hermite collocation equations on GPUs", Lecture Notes in Engineering and Computer Science, vol. 2205, IAENG, pp. 1281-1286, 2013.