Frontal and multi-frontal solvers: Generalization to isogeometric finite element method

Maciej Paszynski

Department of Computer Science

AGH University of Science and Technology, Krakow, Poland

maciej.paszynski@agh.edu.pl

http://home.agh.edu.pl/paszynsk

http://www.ki.agh.edu.pl/en/staff/paszynski-maciej

http://www.ki.agh.edu.pl/en/research-groups/a2s

Main collaborators

Victor Calo (KAUST)
Leszek Demkowicz (ICES, UT)
David Pardo (IKERBASQUE)

INTRODUCTION
B-SPLINES BASED FINITE ELEMENT METHOD

Strong formulation

$$-\frac{d}{dx}\left(A(x)\frac{du}{dx}\right)+B(x)\frac{du}{dx}+C(x)u=f(x)$$

$$u(0)=0$$

$$A(1)\frac{du(1)}{dx}+\beta u(1)=\gamma$$

Weak formulation

$$\text{Find } u\in V=\left\{u\in H^1(0,1):u(0)=0\right\}\text{s.t.}$$

$$b(v,u)=l(v),\quad \forall v\in V=\left\{v\in H^1(0,1):v(0)=0\right\}$$

$$b(v,u)=\int_0^1\left[A(x)\frac{dv}{dx}\frac{du}{dx}+B(x)v(x)\frac{du}{dx}+C(x)v(x)u(x)\right]dx+\beta v(1)u(1)$$

$$l(v)=\gamma v(1)$$

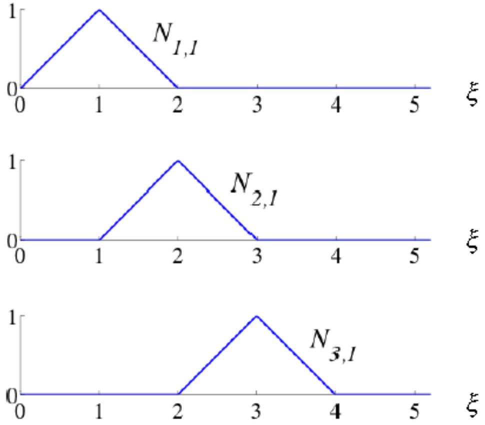
INTRODUCTION
B-SPLINES BASED FINITE ELEMENT METHOD

Using B-splines as basis functions

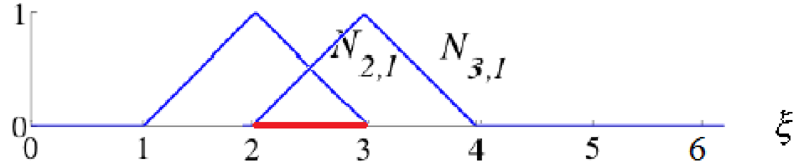
$$\text{Find } u \in V = \left\{ u \in H^1\left(0,1\right) : u\left(0\right) = 0 \right\} \text{s.t.}$$
$$b\left(v,u\right) = l\left(v\right), \quad \forall v \in V = \left\{ v \in H^1\left(0,1\right) : v\left(0\right) = 0 \right\}$$

$$u\left(x\right) \approx \sum_i N_{i,p}\left(x\right) d_i \qquad v\left(x\right) \leftarrow N_{j,p}\left(x\right)$$

$$\sum_i b\left(N_{j,p}\left(x\right), N_{i,p}\left(x\right)\right) a_i = l\left(N_{j,p}\left(x\right)\right), \quad \forall j$$



Linear B-splines



contribution of
b(N2,1;N3,1)

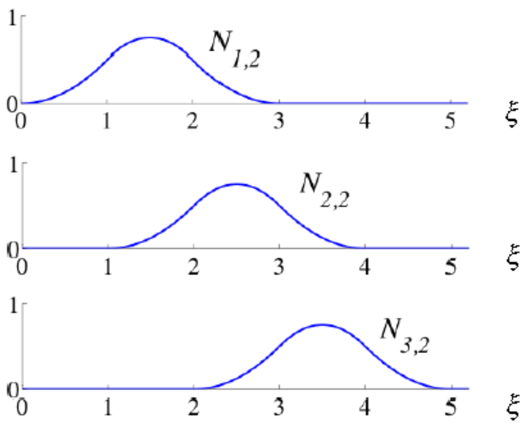
INTRODUCTION
B-SPLINES BASED FINITE ELEMENT METHOD

Using B-splines as basis functions

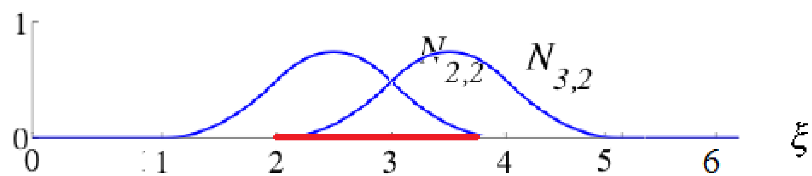
$$\text{Find } u \in V = \left\{ u \in H^1(0,1) : u(0) = 0 \right\} \text{s.t.}$$
$$b(v,u) = l(v), \quad \forall v \in V = \left\{ v \in H^1(0,1) : v(0) = 0 \right\}$$

$$u(x) \approx \sum_i N_{i,p}(x) d_i \qquad v(x) \leftarrow N_{j,p}(x)$$

$$\sum_i b\left(N_{j,p}(x), N_{i,p}(x)\right) a_i = l\left(N_{j,p}(x)\right), \quad \forall j$$

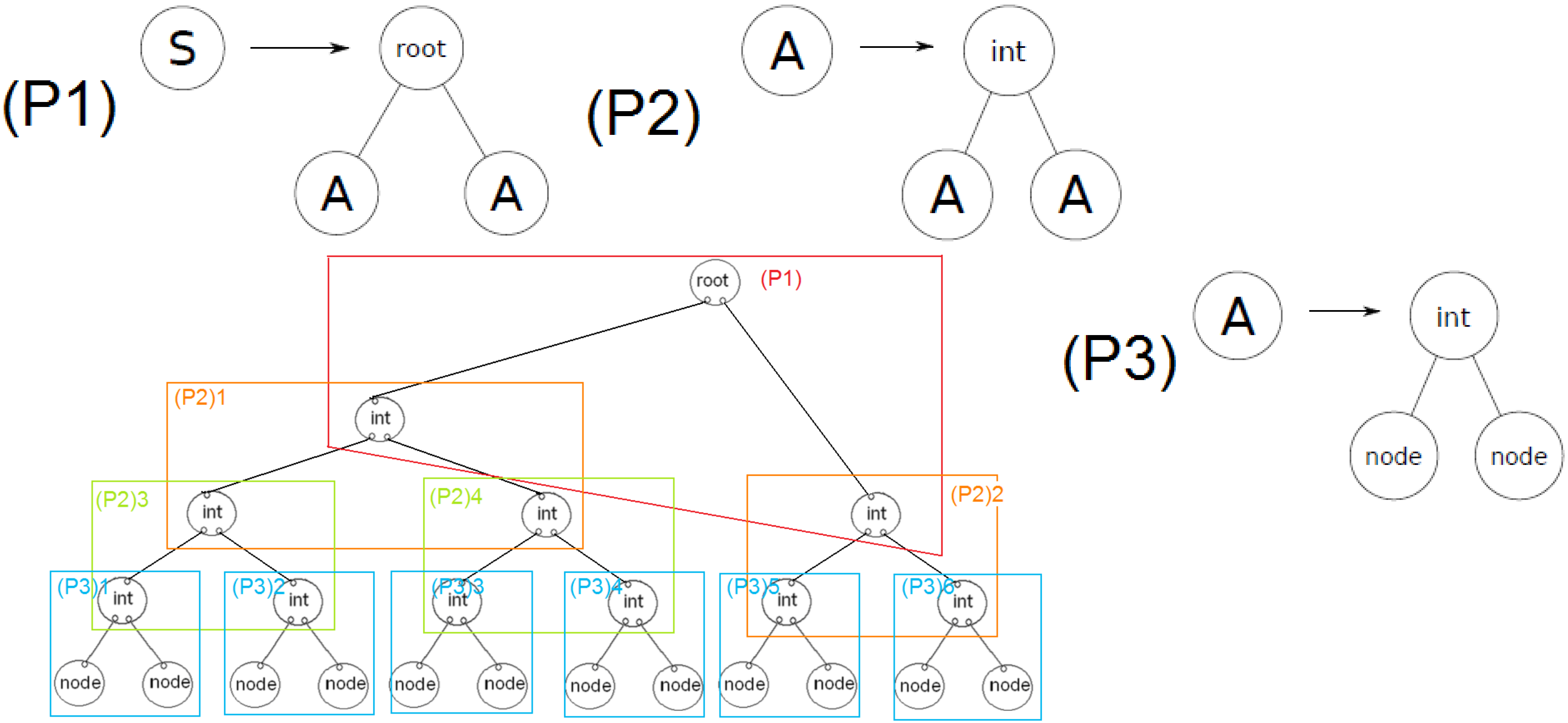


Quadratic B-splines



contribution of b(N2,2;N3,2)

GRAPH GRAMMAR PRODUCTIONS AS ATOMIC TASKS



We assign indices to grammar productions in order to localize the places where the graph grammar productions were fired

The elimination tree obtained by executing the following sequence of productions

$$(P1)\text{-}(P2)_1\text{-}(P2)_2\text{-}(P2)_3\text{-}(P2)_4\text{-}(P3)_1\text{-}(P3)_2\text{-}(P3)_3\text{-}(P3)_4\text{-}(P3)_5\text{-}(P3)_6$$

SCHEDULER BASED ON GRAPH COLORING

Alphabet:

$A = \{(P1), (P2)_1, (P2)_2, (P2)_3, (P2)_4, (P3)_1, (P3)_2, (P3)_3, (P3)_4, (P3)_5, (P3)_6\}$

Dependency relation for construction of the elimination tree

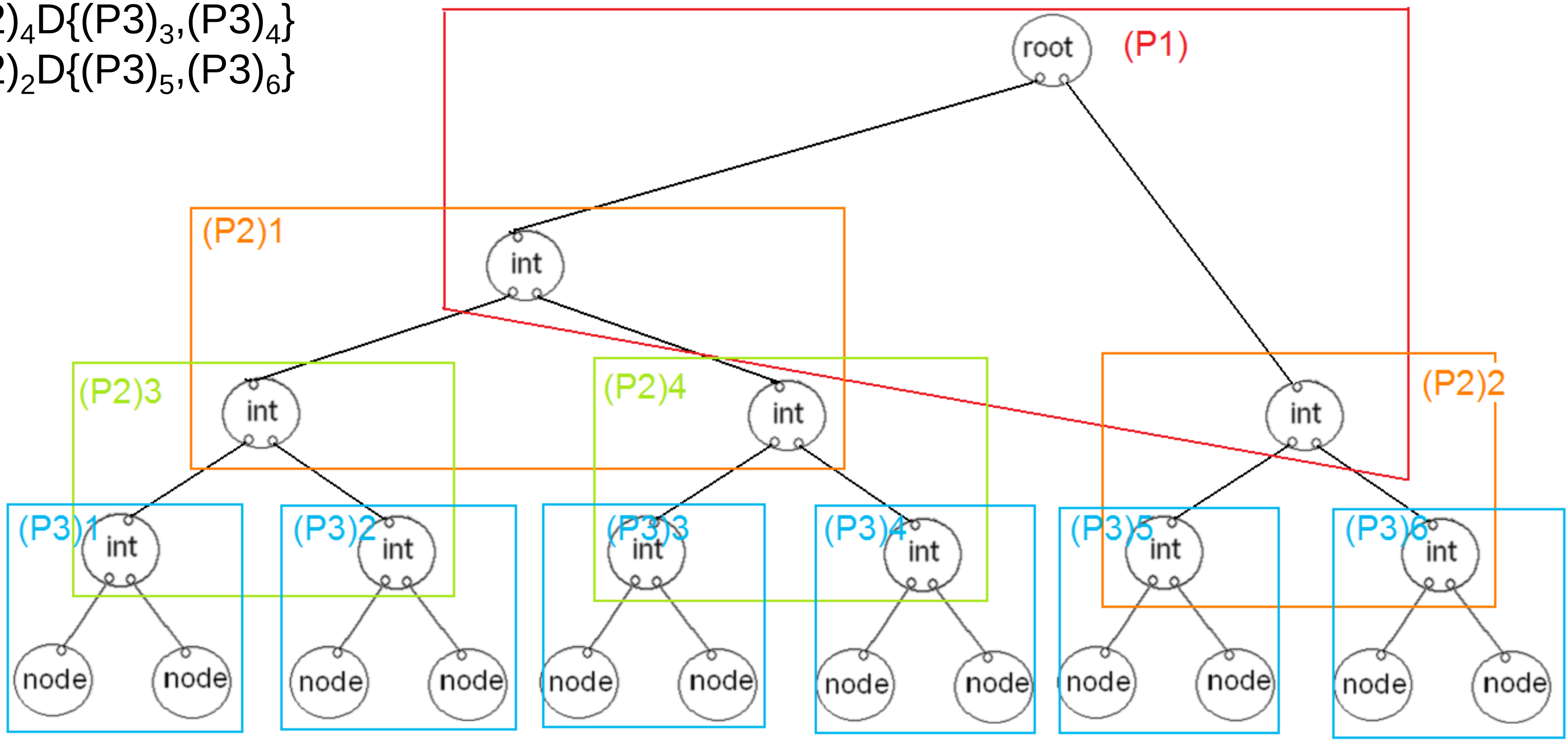
$(P1)D\{(P2)_1,(P2)_2\}$

$(P2)_1D\{(P2)_3,(P2)_4\}$

$(P2)_3D\{(P3)_1,(P3)_2\}$

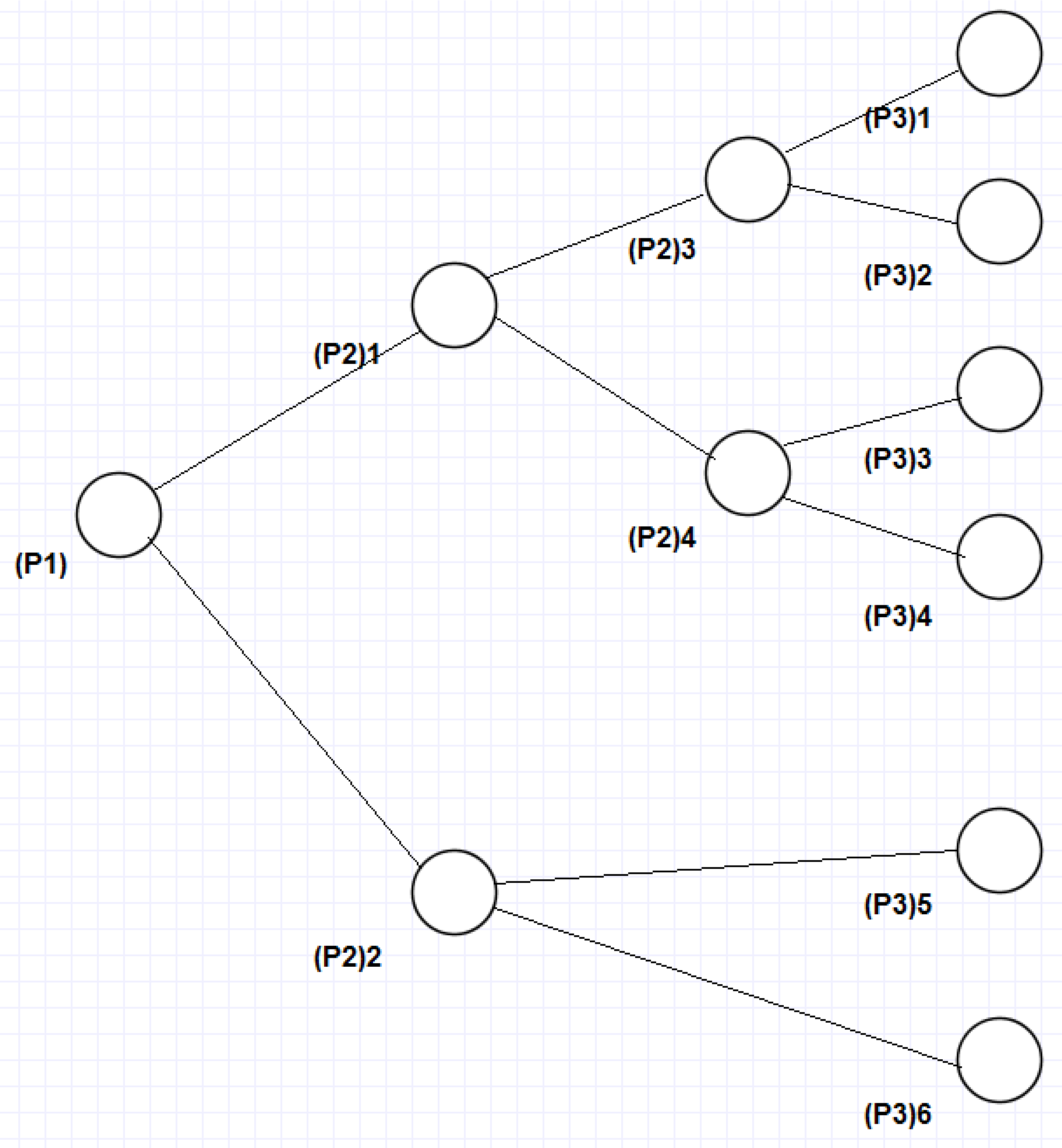
$(P2)_4D\{(P3)_3,(P3)_4\}$

$(P2)_2D\{(P3)_5,(P3)_6\}$



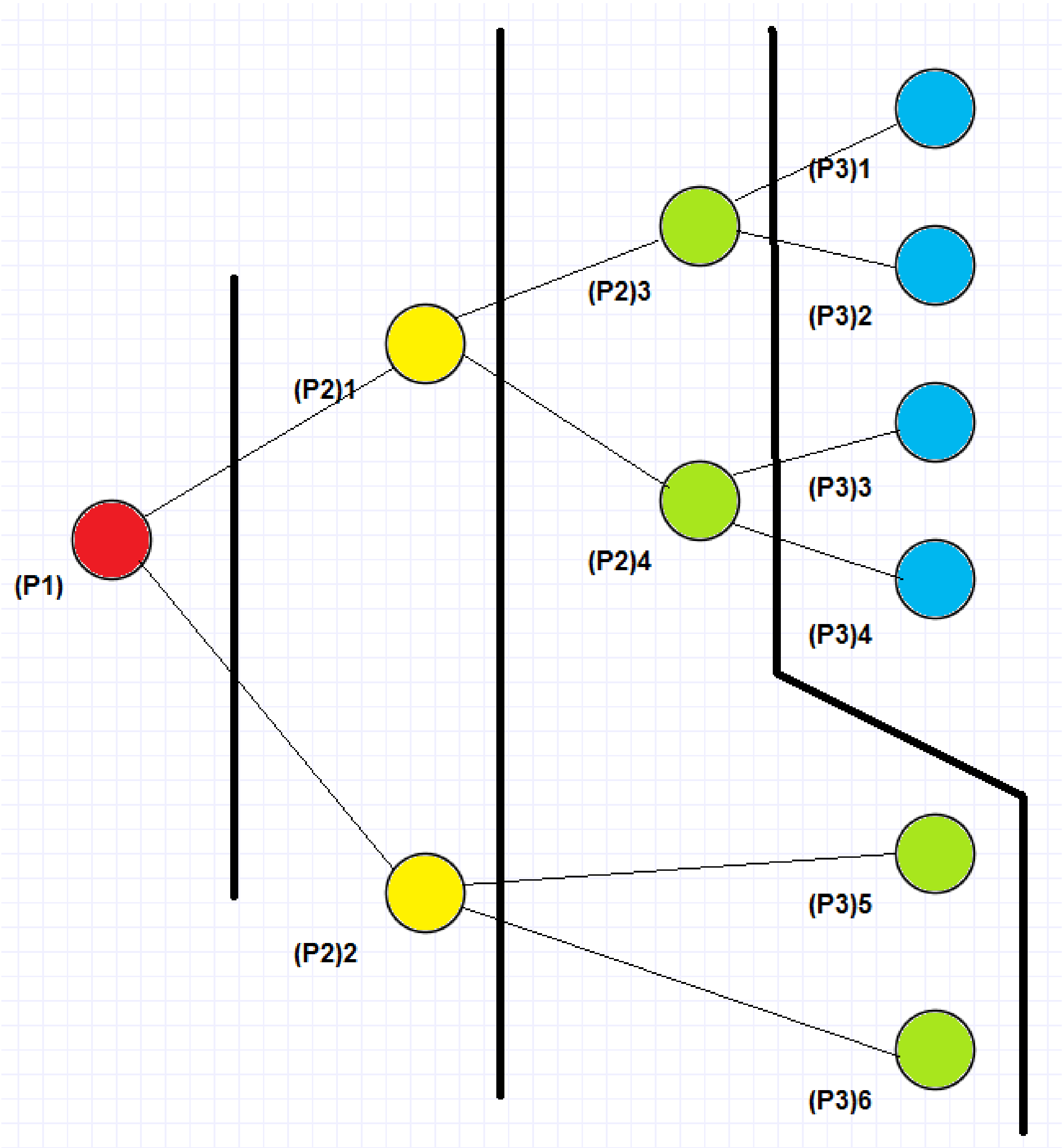
SCHEDULER BASED ON GRAPH COLORING

Dependency graph



SCHEDULER BASED ON GRAPH COLORING

Dependency graph



TRACE THEORY BASED SCHEDULER

$(P1)-(P2)_1-(P2)_2-(P2)_3-(P2)_4-(P3)_1-(P3)_2-(P3)_3-(P3)_4-(P3)_5-(P3)_6$

Foata Normal Form

$$\left[a_1^1 a_2^1 \ldots a_{l_1}^1\right]\left[a_1^2 a_2^2 \ldots a_{l_1}^2\right]\ldots\left[a_1^n a_2^n \ldots a_{l_n}^n\right]$$

$a_i^k \in A$ (alphabet)

$\forall k \; \forall i, j \in \{1,\ldots,l_k\} \quad a_i^k I a_j^k$ i<>j where $I = AxA \backslash D$

$\forall k \; \forall i \in \{1,\ldots,l_k\} \, \exists j \in \{1,\ldots,l_{k-1}\} \quad a_i^{k-1} D a_j^k$

Scheduling according to Foata Normal Form:

$[(P1)][(P2)_1(P2)_2][(P2)_3(P2)_4(P3)_5(P3)_6][(P3)_1(P3)_2(P3)_3(P3)_4]$

Thus, the execution of the solver consists of several steps, where independent tasks are executed in concurrent, interchanged with the synchronization barriers.

GRAMMAR BASED NUMERICAL INTEGRATION

$$b\left(N_{j,p}(x), N_{i,p}(x)\right) =$$

$$= \int_0^1 \left[A(x) \frac{dN_{j,p}(x)}{dx} \frac{dN_{i,p}(x)}{dx} + B(x) N_{j,p}(x) \frac{dN_{i,p}(x)}{dx} + C(x) N_{j,p}(x) N_{i,p}(x) \right] dx$$

$$+ \beta N_{j,p}(1) N_{i,p}(1)$$

$$l\left(N_{i,p}(x)\right) = \gamma N_{i,p}(1)$$

using Gaussian quadrature the integration over the domain can be substituted by a weighted summation over Gauss points

$$\int_0^1 \left[A(x) \frac{dN_{i,p}(x)}{dx} \frac{dN_{j,p}(x)}{dx} + B(x) \frac{dN_{i,p}(x)}{dx} N_{j,p}(x) + C(x) N_{i,p}(x) N_{j,p}(x) \right] dx =$$

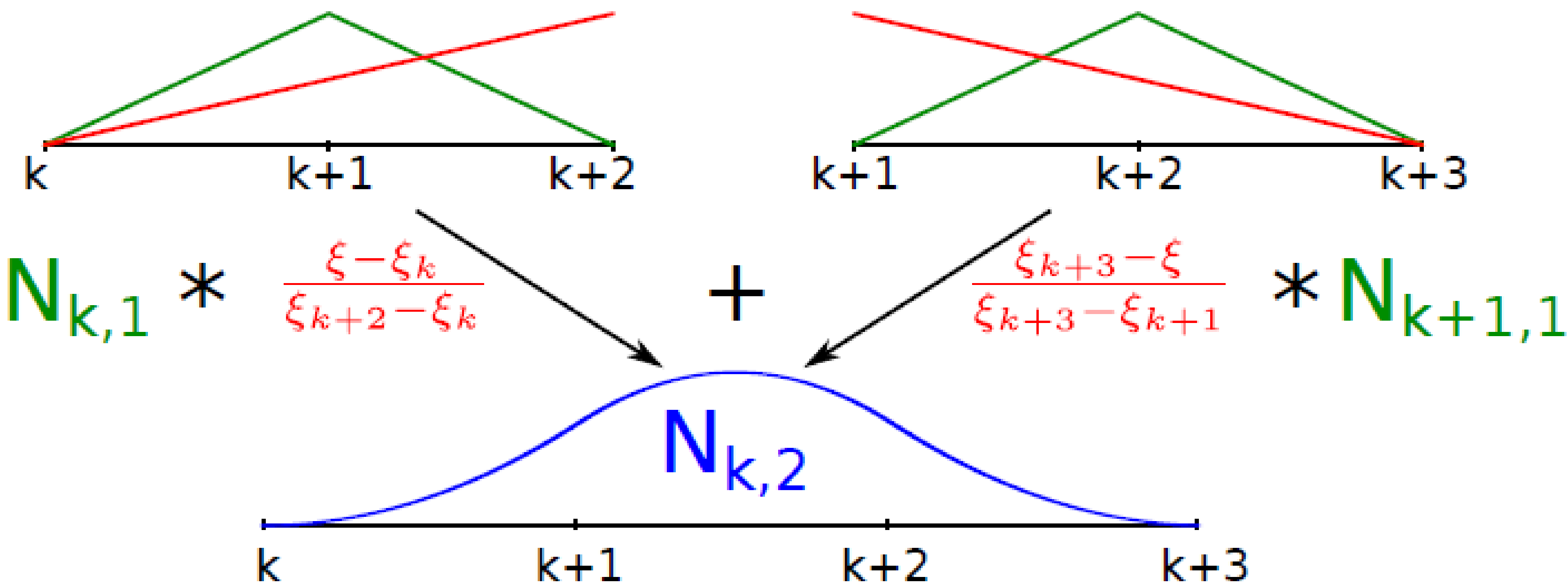
$$\sum_l w_l \left[A(x_l) \frac{dN_{i,p}(x_l)}{dx} \frac{dN_{j,p}(x_l)}{dx} + B(x_l) \frac{dN_{i,p}(x_l)}{dx} N_{j,p}(x_l) + C(x_l) N_{i,p}(x_l) N_{j,p}(x_l) \right]$$

GRAMMAR BASED NUMERICAL INTEGRATION

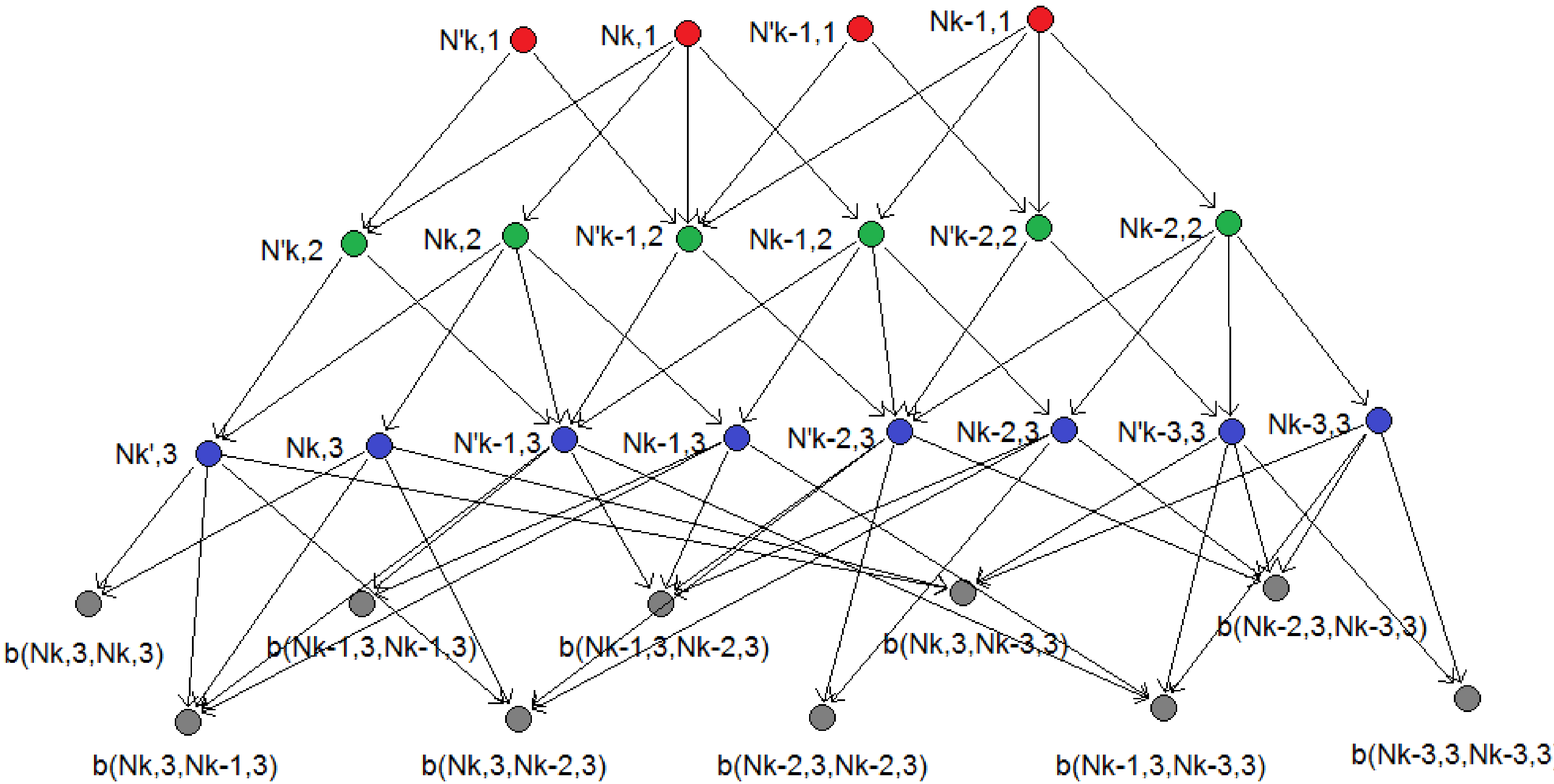
Cox-de Boor formula

$$N_{i,0}(\xi) = I_{[\xi_i,\xi_{i+1}]}$$

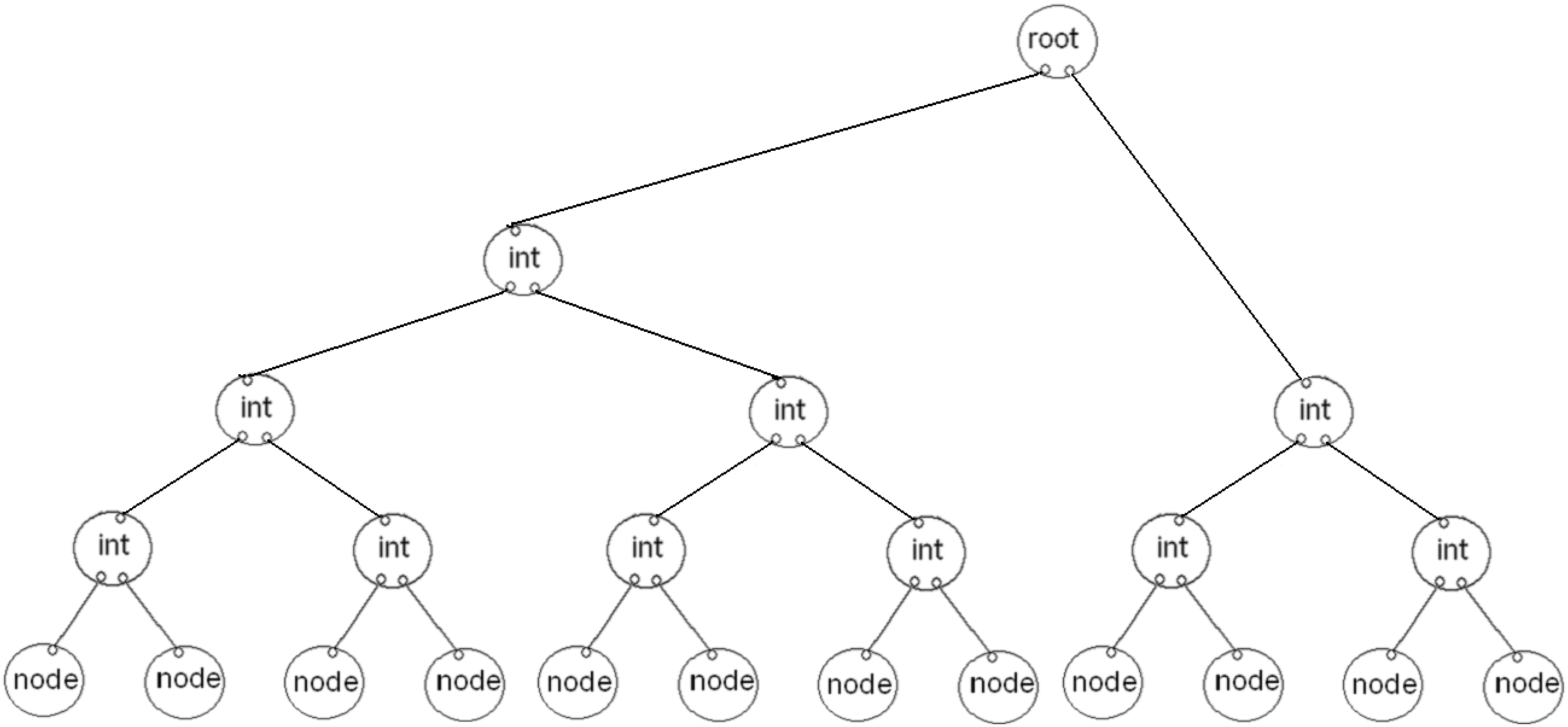
$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{x_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{x_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi)$$



GRAMMAR BASED NUMERICAL INTEGRATION



PROCESS OF THE ELIMINATION EXPRESSED BY GRAPH GRAMMAR PRODUCTIONS



$$\begin{bmatrix} 1 & 0 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \begin{bmatrix} -1/h^2 & 1/h^2 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_2^{II} \\ x_3^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \begin{bmatrix} -1/h^2 & 1/h^2 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_3^{II} \\ x_4^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

$$\begin{bmatrix} -1/h^2 & 1/h^2 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_4^{II} \\ x_5^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \begin{bmatrix} -1/h^2 & 1/h^2 \\ 1/h^2 & -1/h^2 \end{bmatrix} \begin{Bmatrix} x_5^{II} \\ x_6^{I} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \begin{bmatrix} -1/h^2 & 1/h^2 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} x_6^{II} \\ x_7 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 20 \end{Bmatrix}$$

Generation of frontal matrices at leaves of the eliminaton tree expressed as the execution of graph grammar productions (A1)-(A)^4-(AN)

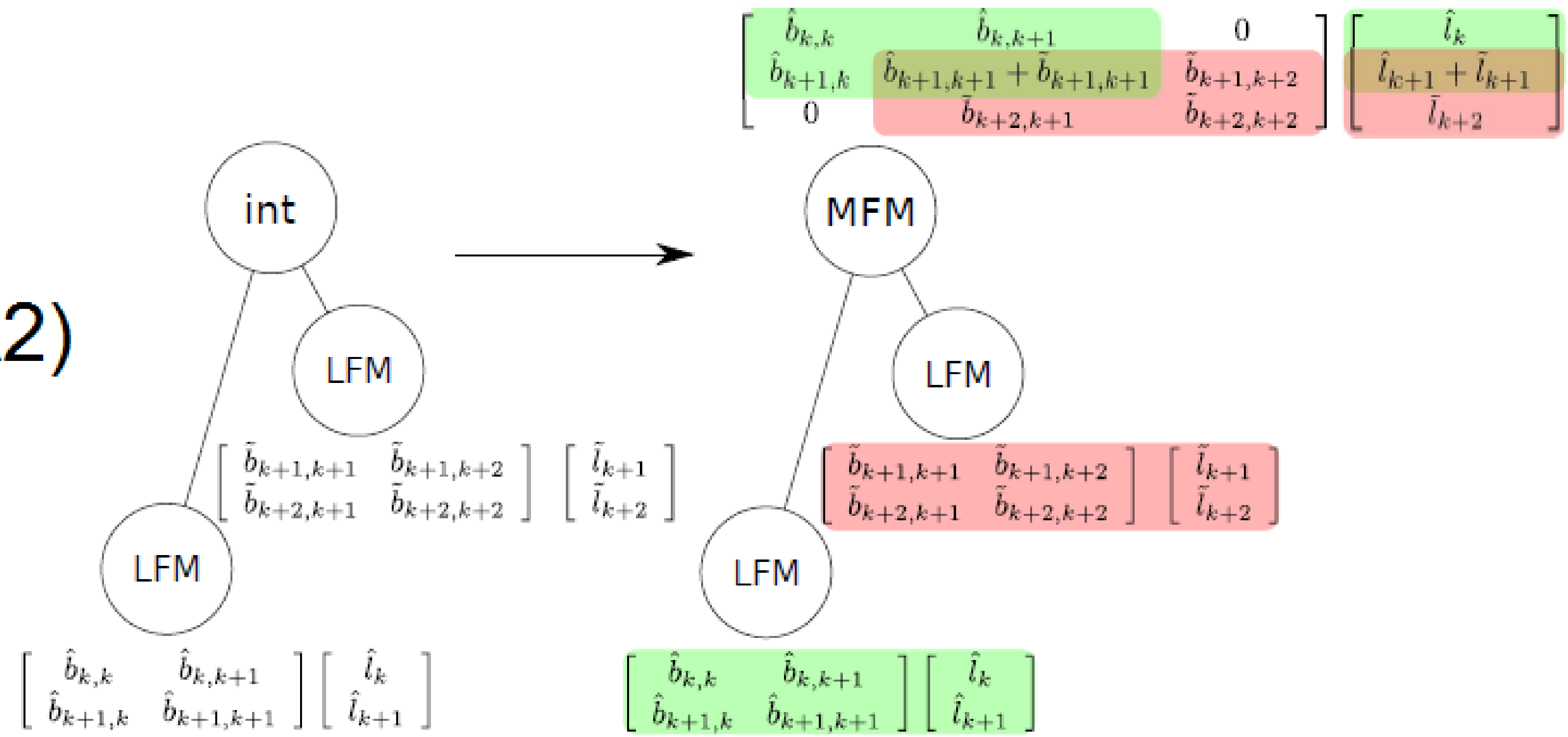
PROCESS OF THE ELIMINATION
EXPRESSED BY GRAPH GRAMMAR PRODUCTIONS

(A1),(A),(AN) node $\longrightarrow$ LFM $\begin{bmatrix} b_{k,k} & b_{k,k+1} \\ b_{k+1,k} & b_{k+1,k+1} \end{bmatrix} \begin{bmatrix} l_k \\ l_{k+1} \end{bmatrix}$

Graph grammar productions generating local frontal matrices for left boundary, interior and right boundary nodes for linear B-splines

PROCESS OF THE ELIMINATION
EXPRESSED BY GRAPH GRAMMAR PRODUCTIONS

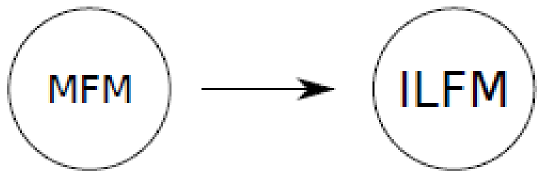
(A2)



Graph grammar productions merging element frontal matrices at parent level

PROCESS OF THE ELIMINATION
EXPRESSED BY GRAPH GRAMMAR PRODUCTIONS

(E2)



$$\begin{bmatrix} b_{k,k} & b_{k,k+1} & 0 \\ b_{k+1,k} & b_{k+1,k+1} & b_{k+1,k+2} \\ 0 & b_{k+2,k+1} & b_{k+2,k+2} \end{bmatrix} \begin{bmatrix} l_k \\ l_{k+1} \\ l_{k+2} \end{bmatrix}$$

$$\begin{bmatrix} b_{k+1,k+1} & b_{k+1,k} & b_{k+1,k+2} \\ b_{k,k+1} & b_{k,k} & 0 \\ b_{k+2,k+1} & 0 & b_{k+2,k+2} \end{bmatrix} \begin{bmatrix} l_{k+1} \\ l_k \\ l_{k+2} \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} 1 & \hat{b}_{k+1,k} & \hat{b}_{k+1,k+2} \\ 0 & \hat{b}_{k,k} & \hat{b}_{k,k+2} \\ 0 & \hat{b}_{k+2,k} & \hat{b}_{k+2,k+2} \end{bmatrix} \begin{bmatrix} \hat{l}_{k+1} \\ \hat{l}_k \\ \hat{l}_{k+2} \end{bmatrix}$$

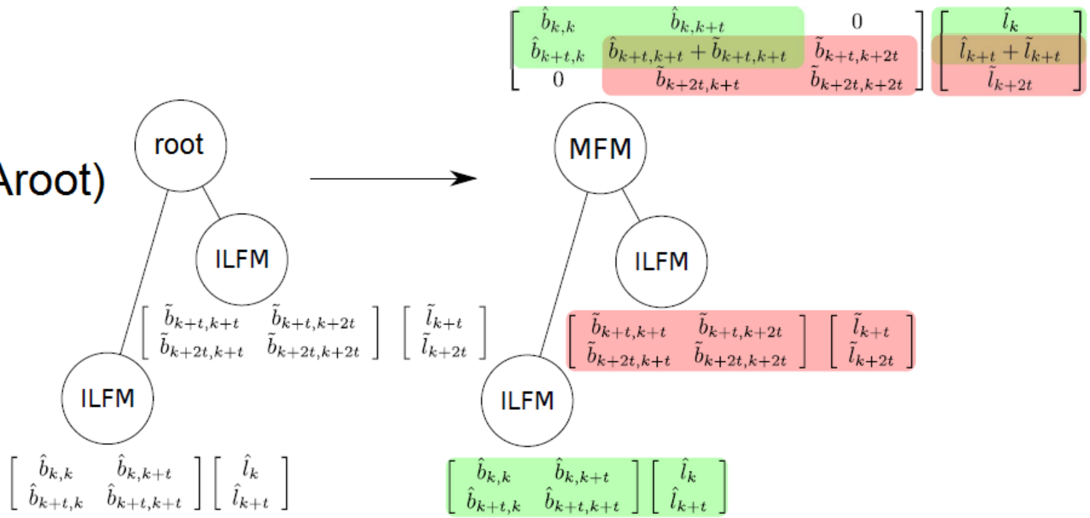
$$\downarrow$$

$$\begin{bmatrix} \hat{b}_{k,k} & \hat{b}_{k,k+2} \\ \hat{b}_{k+2,k} & \hat{b}_{k+2,k+2} \end{bmatrix} \begin{bmatrix} \hat{l}_k \\ \hat{l}_{k+2} \end{bmatrix}$$

Graph grammar production eliminating fully assembled row at parent level

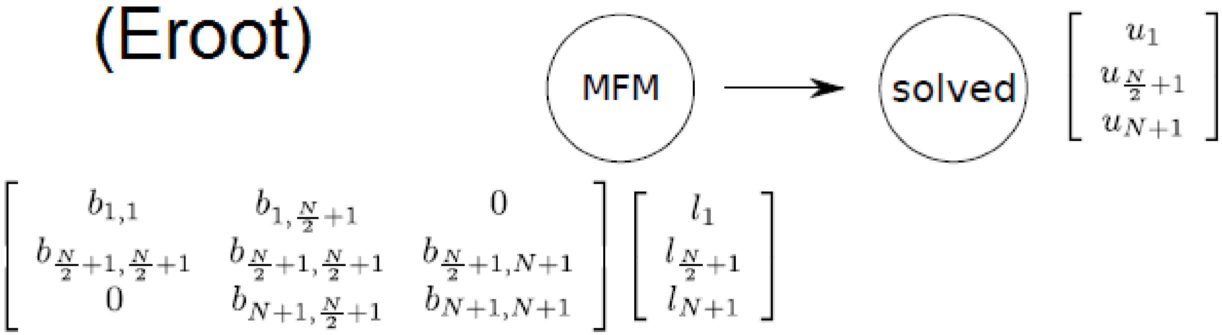
PROCESS OF THE ELIMINATION
EXPRESSED BY GRAPH GRAMMAR PRODUCTIONS

(Aroot)



Graph grammar production for merging element frontal matrices at root level

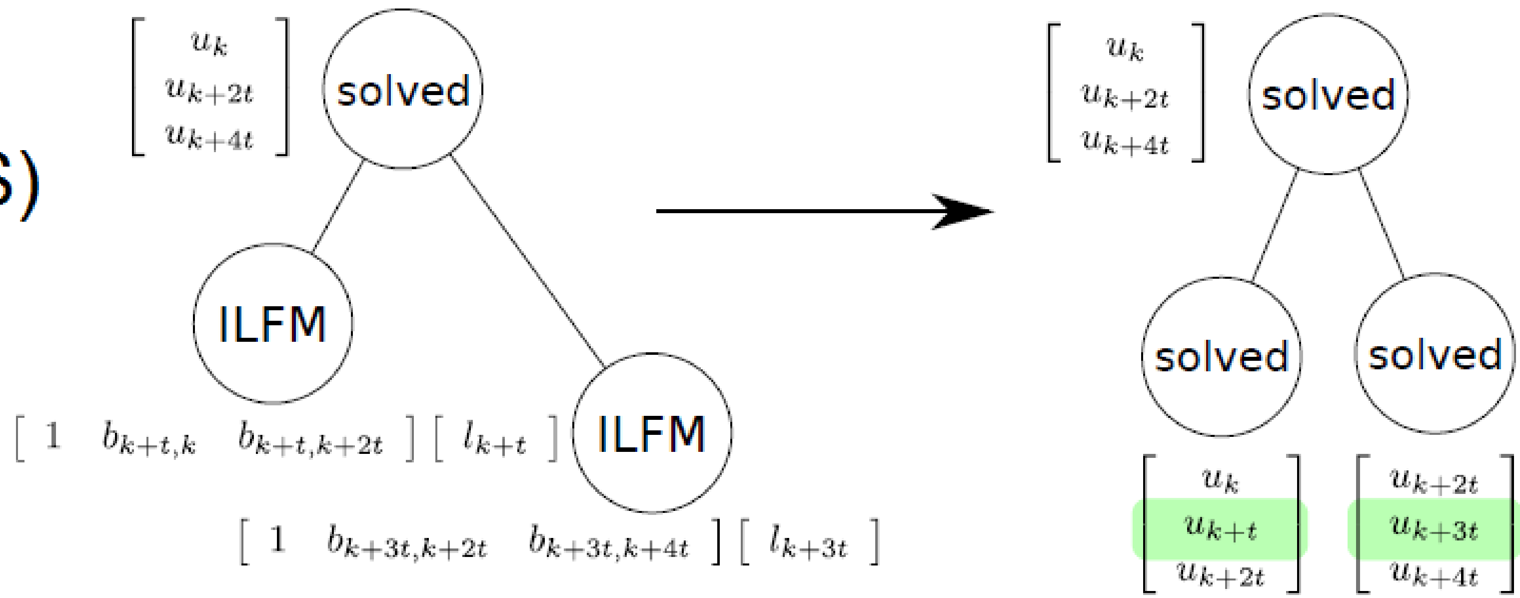
(Eroot)



Graph grammar production for solution at root level

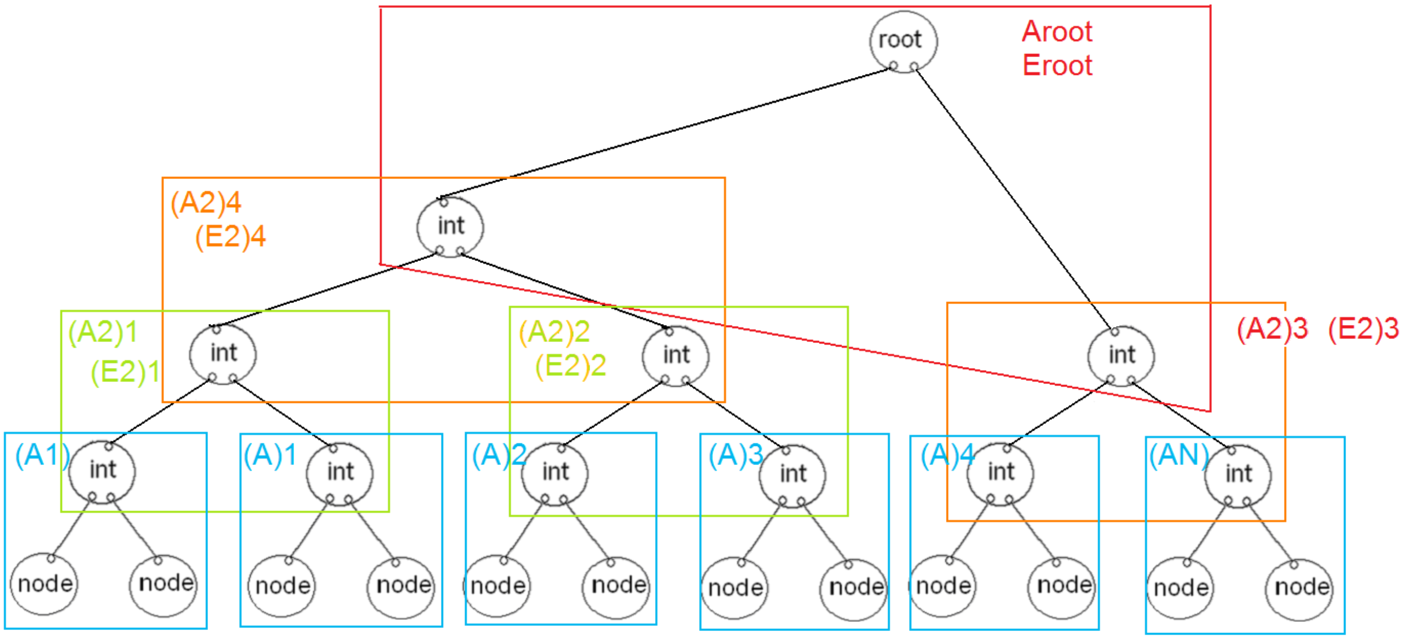
PROCESS OF THE ELIMINATION
EXPRESSED BY GRAPH GRAMMAR PRODUCTIONS

(BS)



Graph grammar production for recursive backward substitution

PROCESS OF THE ELIMINATION EXPRESSED BY GRAPH GRAMMAR PRODUCTIONS



Expression of the solver execution by graph grammar productions

(A1)-(A)^4-(AN) (generation of frontal matrices at leaves of the elimination trees)

(A2)^3 (merging contributions at father nodes)

(E2)^3 (elimination of fully assembled nodes)

(A2) – (E2) (merging at parent node followed by elimination)

(Aroot) – (Eroot) (merging at root node followed by full forward elimination)

(BS)^4 (backward substitutions)

SCHEDULER BASED ON GRAPH COLORING

Alphabet:

A={(A1), $(A)_1$, $(A)_2$, $(A)_3$, $(A)_4$, (AN), $(A2)_1$, $(A2)_2$, $(A2)_3$, $(E2)_1$, $(E2)_2$, $(E2)_3$, $(A2)_4$, $(E2)_4$, (Aroot) , (Eroot) , $(BS)_1$, $(BS)_2$, $(BS)_3$, $(BS)_4$ }

Dependency relation for the solver algorithm

$\{(A1),(A)_1\}D(A2)_1$

$\{(A)_2,(A)_3\}D(A2)_2$

$\{(A)_4,(AN)\}D(A2)_3$

$(A2)_1D(E2)_1$

$(A2)_2D(E2)_2$

$(A2)_3D(E2)_3$

$\{(E2)_1,(E2)_2\}D(A2)_4$

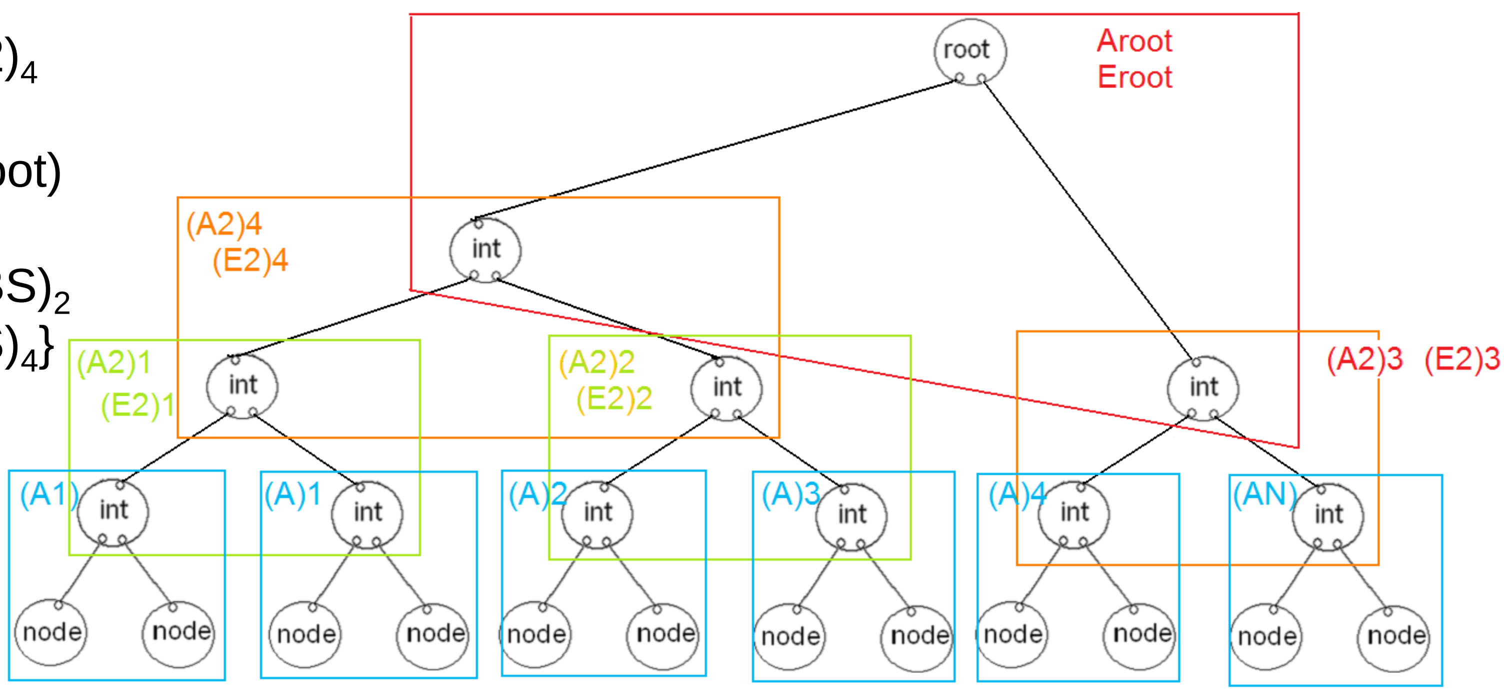
$(A2)_4D(E2)_4$

$\{(E2)_3(E2)_4\}D(Aroot)$

$(Aroot)D(Eroot)$

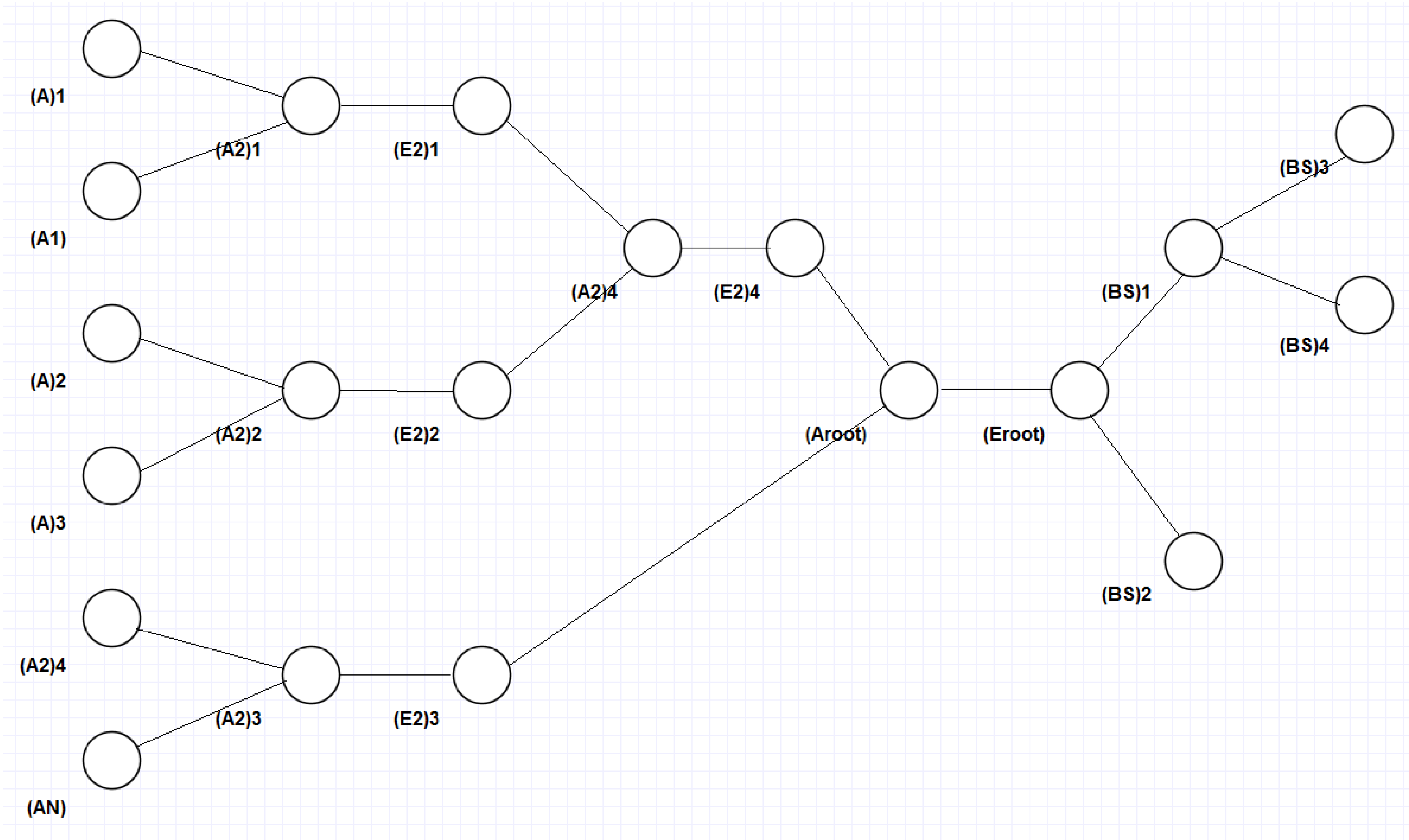
$(Eroot)D\{(BS)_1,(BS)_2$

$(BS)_1D\{(BS)_3,(BS)_4\}$



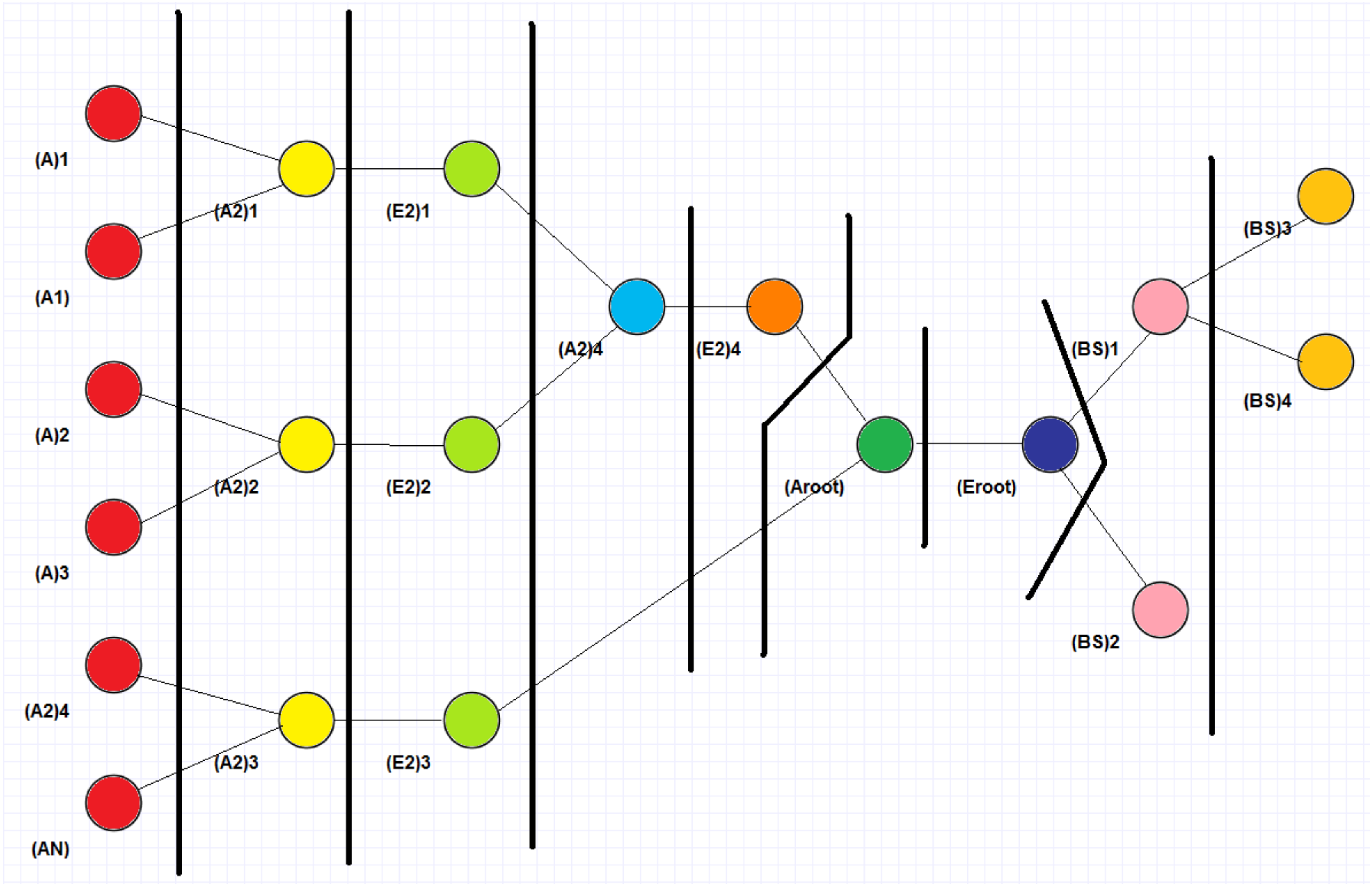
SCHEDULER BASED ON GRAPH COLORING

Dependency graph



SCHEDULER BASED ON GRAPH COLORING

Dependency graph



TRACE THEORY BASED SCHEDULER

(A1)-$(A)_1$-$(A)_2$-$(A)_3$-$(A)_4$- (AN)-$(A2)_1$-$(A2)_2$- $(A2)_3$-$(E2)_1$-$(E2)_2$-$(E2)_3$- $(A2)_4$- $(E2)_4$-
(Aroot)-(Eroot)-$(BS)_1$-$(BS)_2$-$(BS)_3$-$(BS)_4$

Foata Normal Form

$$\left[a_1^1 a_2^1 \ldots a_{l_1}^1\right]\left[a_1^2 a_2^2 \ldots a_{l_1}^2\right]\ldots\left[a_1^n a_2^n \ldots a_{l_n}^n\right]$$

$a_i^k \in A$ (alphabet)

$$\forall k\ \forall i, j \in \{1,\ldots,l_k\} \quad a_i^k I a_j^k$$

$$\forall k\ \forall i \in \{1,\ldots,l_k\}\ \exists j \in \{1,\ldots,l_{k-1}\} \quad a_i^{k-1} D a_j^k$$

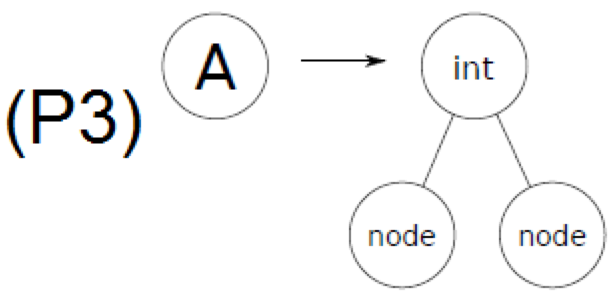
Scheduling according to Foata Normal Form:

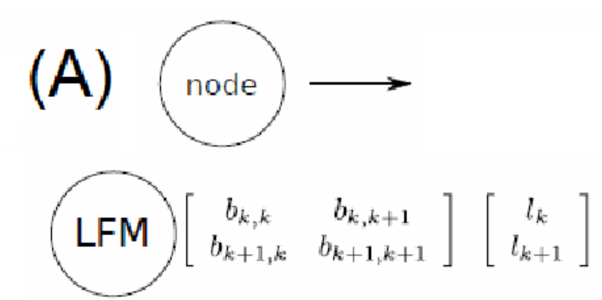
[(A1)$(A)_1$$(A)_2$$(A)_3$$(A)_4$(AN)][$(A2)_1$$(A2)_2$$(A2)_3$][$(E2)_1$$(E2)_2$$(E2)_3$] [$(A2)_4$][$(E2)_4$]
[(Eroot)][(Aroot)][(Eroot)][$(BS)_1$$(BS)_2$][$(BS)_3$$(BS)_4$]

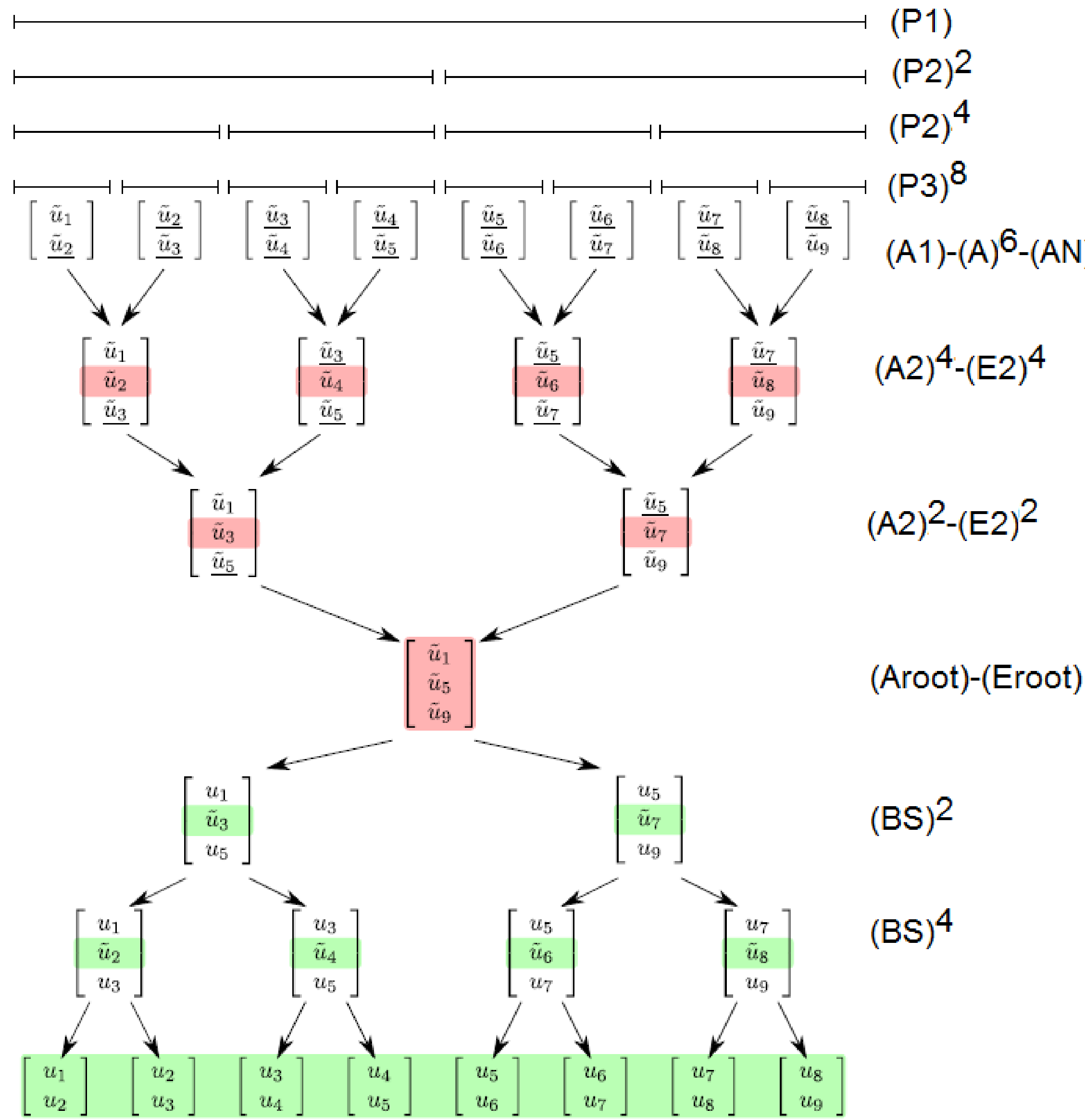
Thus, the execution of the solver consists of several steps, where independent tasks are executed in concurrent, interchanged with the synchronization barriers.

GRAPH GRAMMAR PRODUCTIONS EXPRESSING THE SOLVER ALGORITHM

Linear B-splines



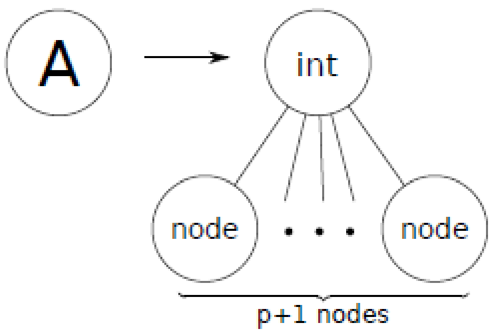




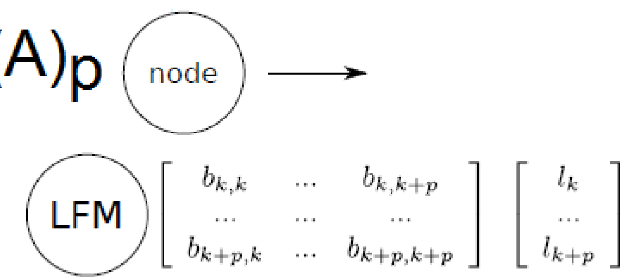
GRAPH GRAMMAR PRODUCTIONS EXPRESSING THE SOLVER ALGORITHM

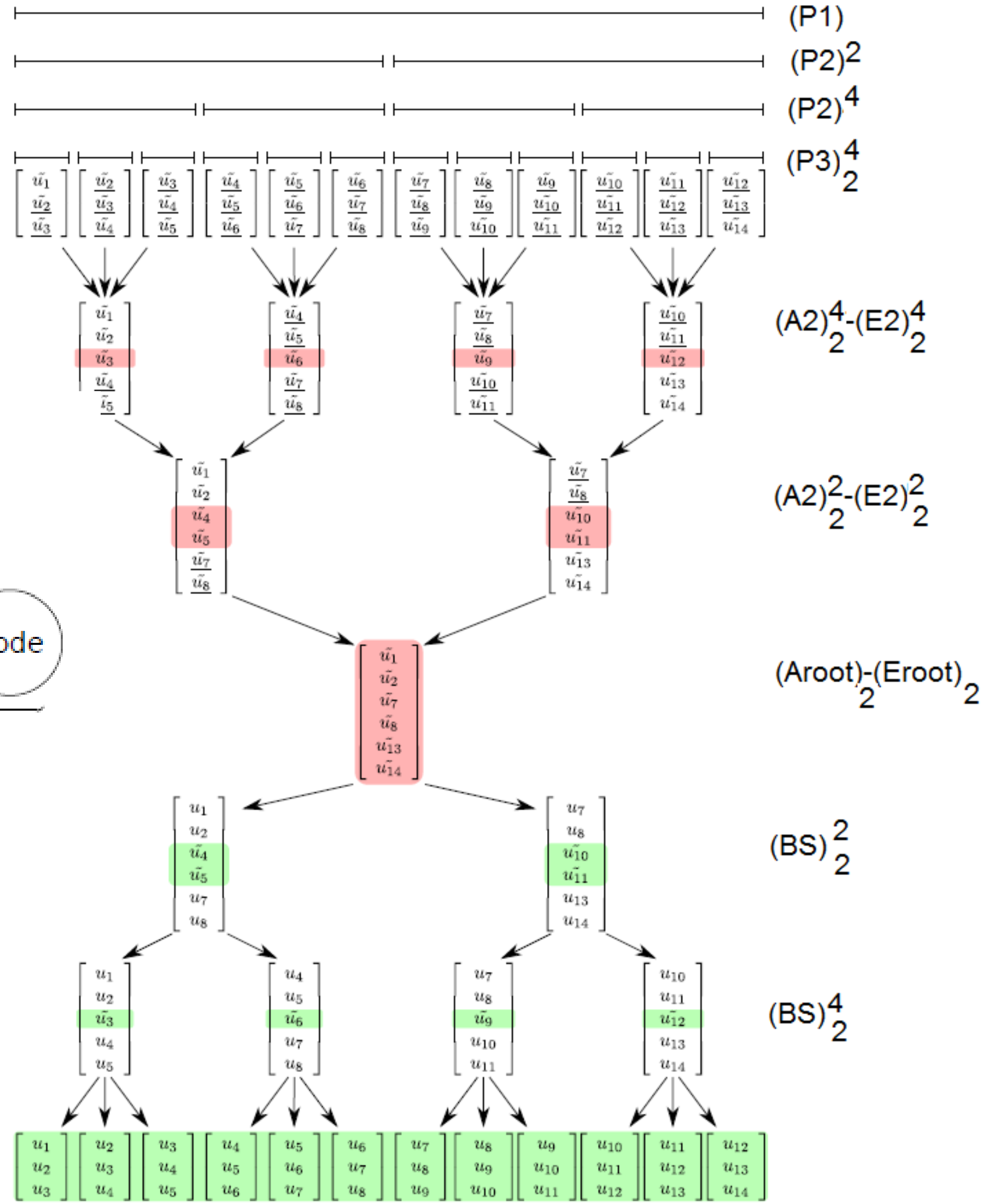
Quadratic B-splines

$(P3)_p$



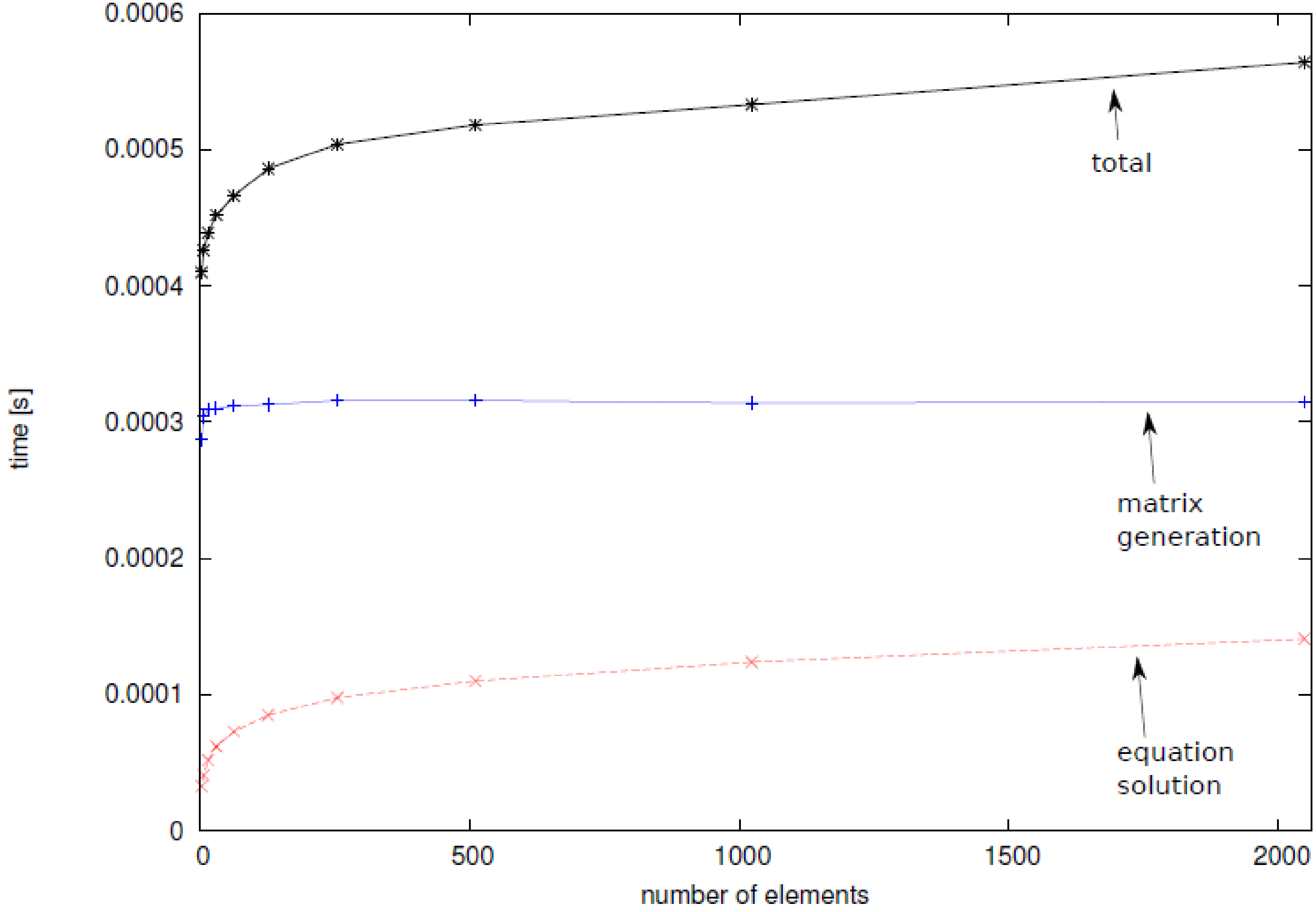
$(A)_p$





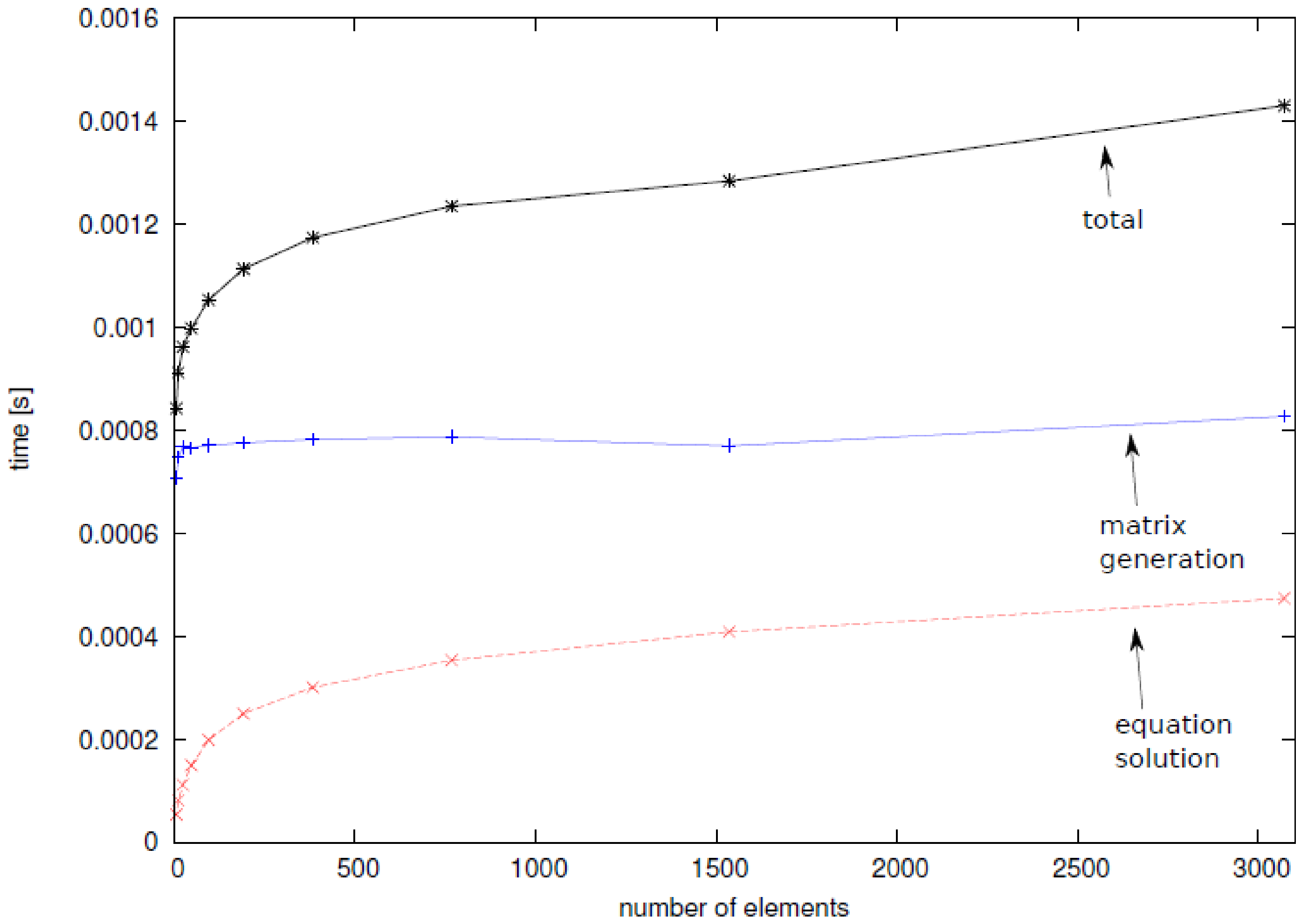
NUMERICAL EXPERIMENTS

1D NUMERICAL RESULTS
LINEAR B-SPLINES



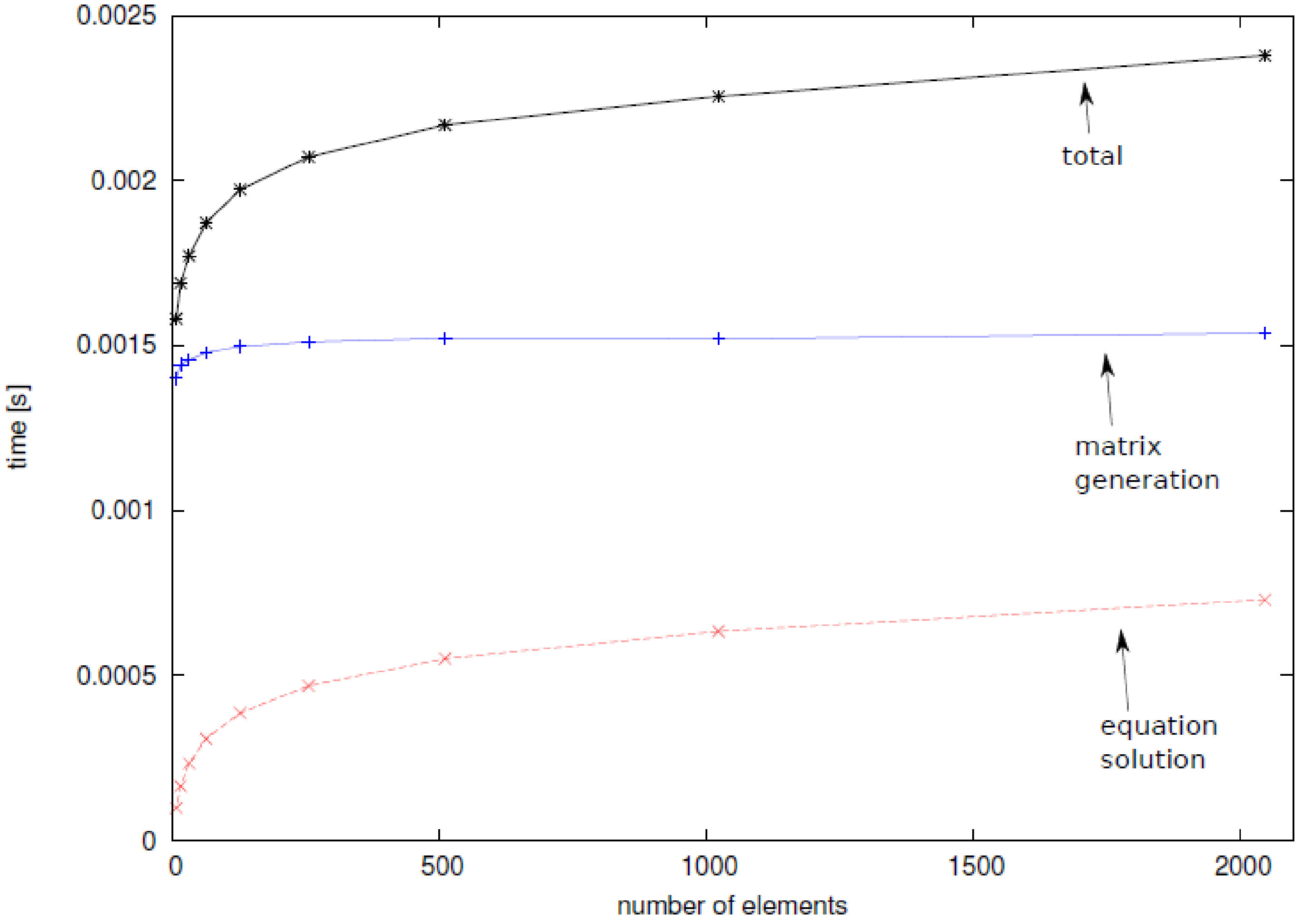
NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock

1D NUMERICAL RESULTS
QUADRATIC B-SPLINES



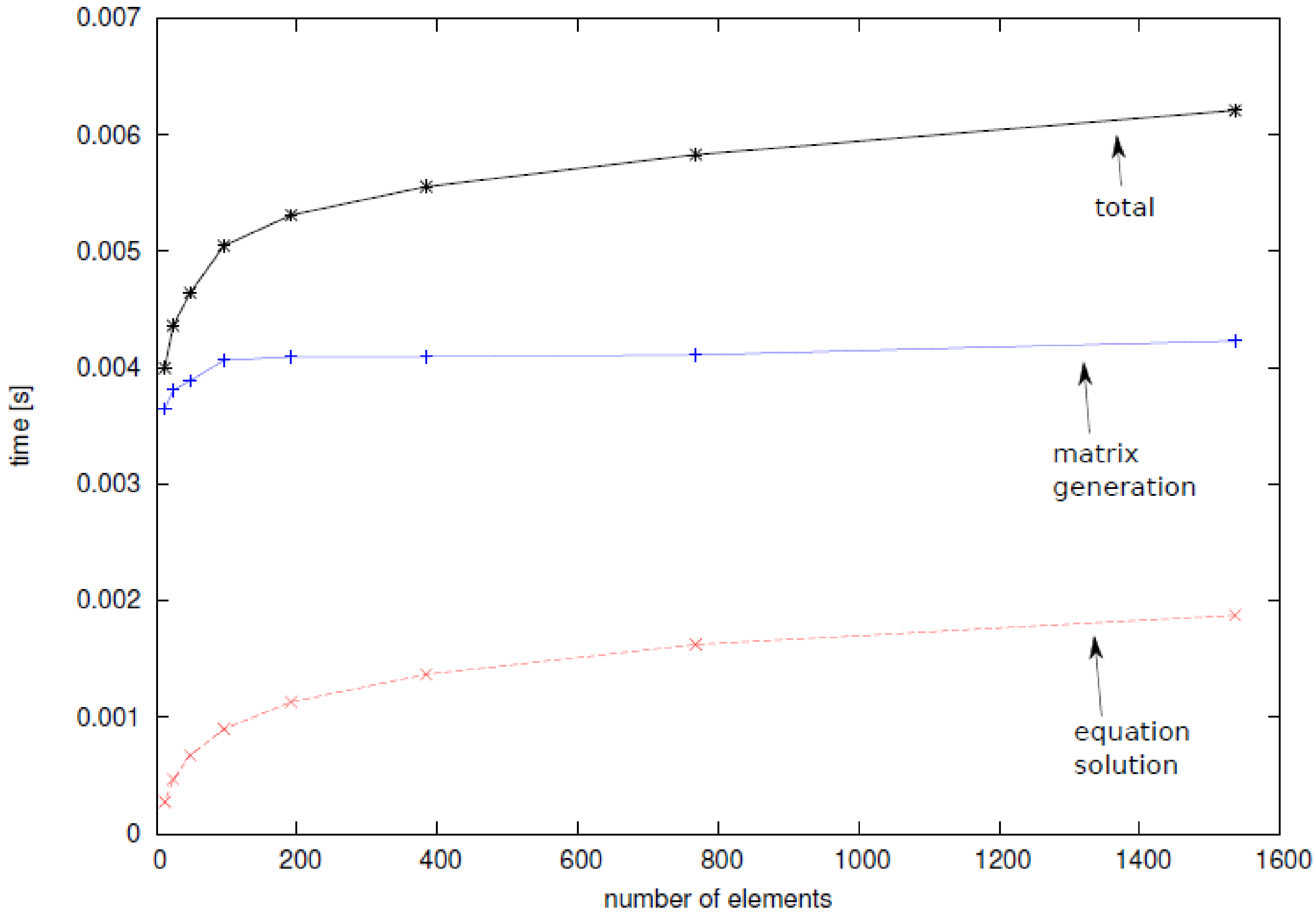
NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock

1D NUMERICAL RESULTS
CUBIC B-SPLINES



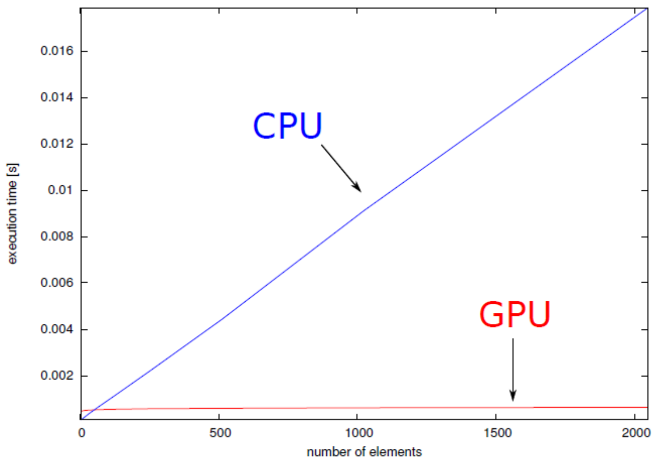
NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock

1D NUMERICAL RESULTS
QINTIC B-SPLINES

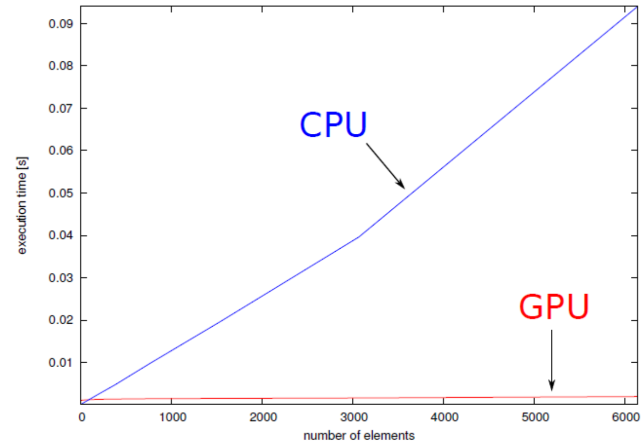


NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock

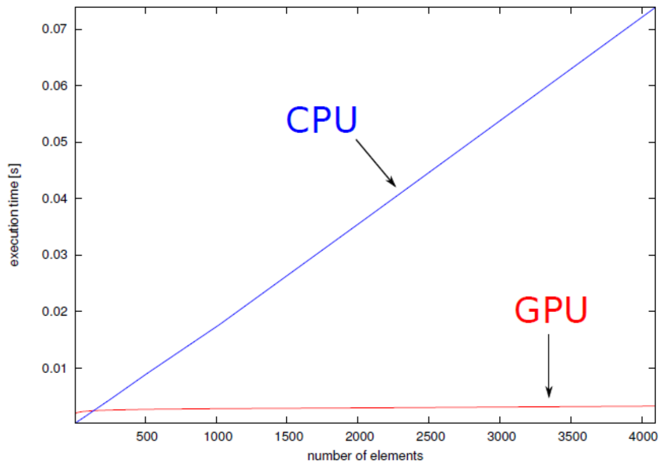
COMPARISON WITH CPU MUMPS SOLVER



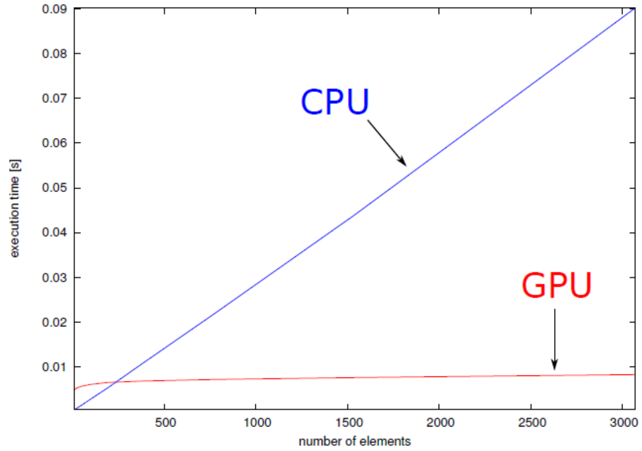
Linear B-splines



Quadratic B-splines



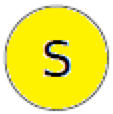
Cubic B-splines

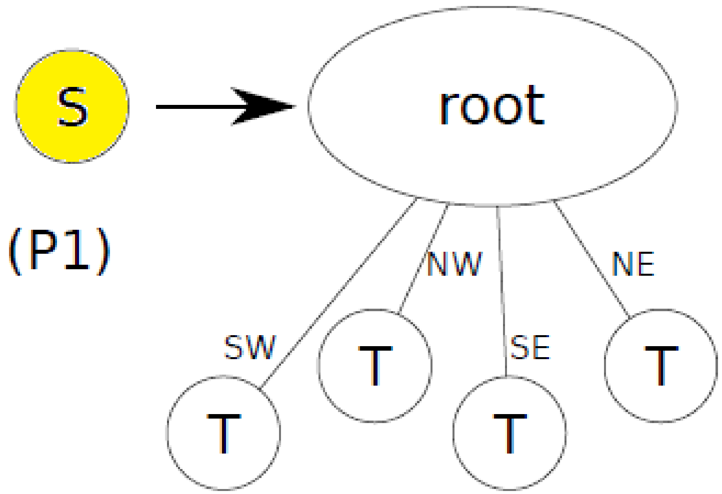


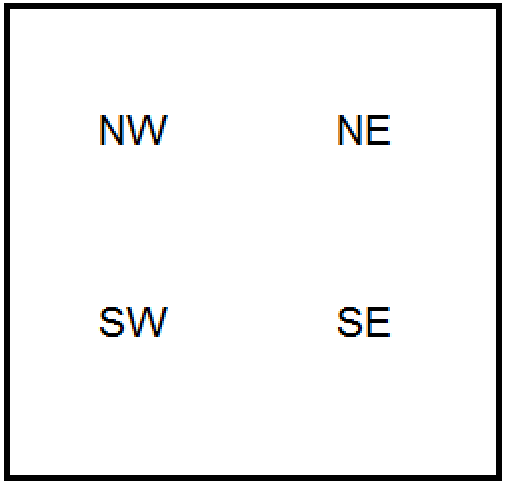
Qintic B-splines

NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock
Intel(R) Core(TM)2 Quad CPU Q9400 with 2.66GHz clock, 8GB of memory

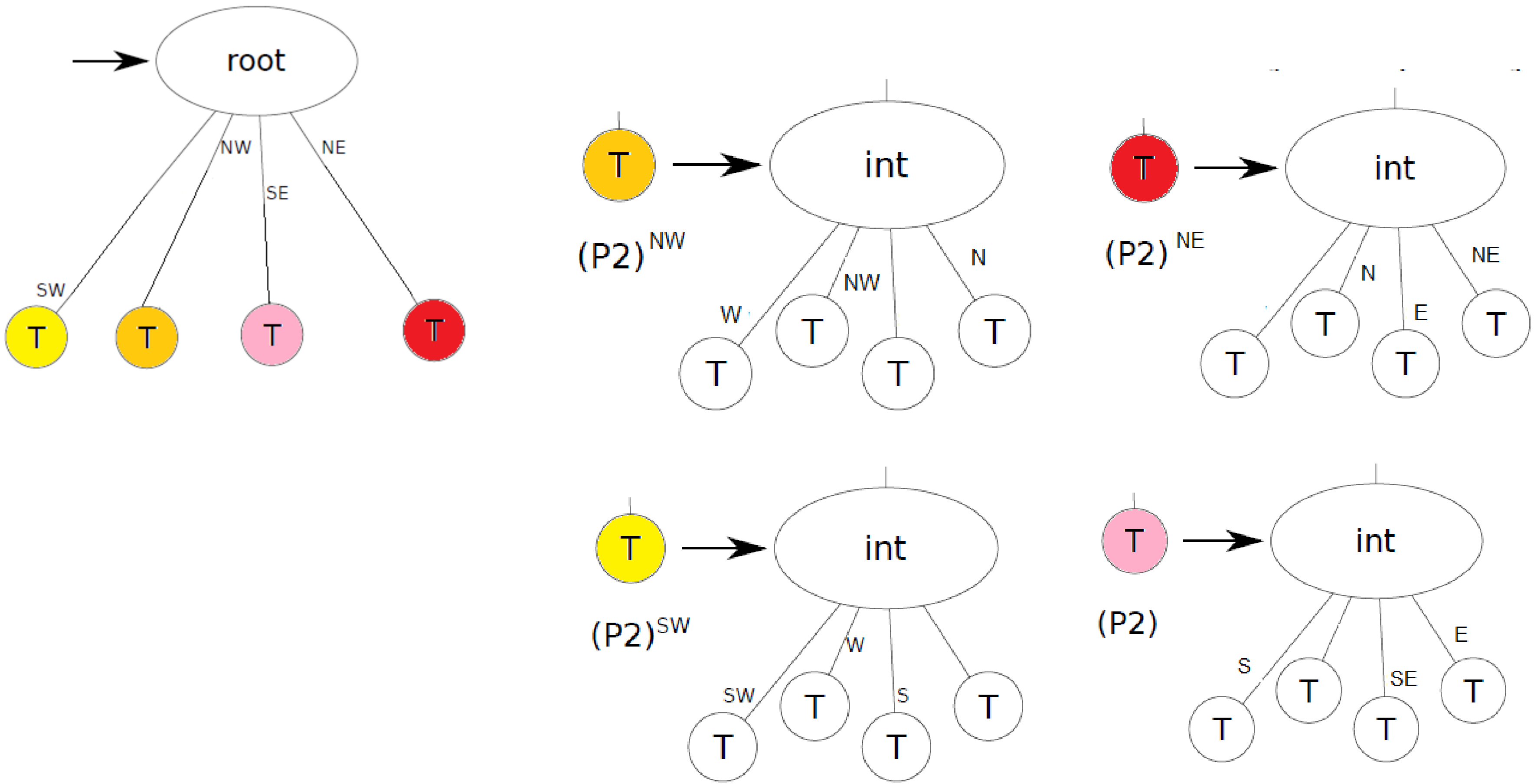
GENERATION OF 2D ELIMINATION TREE



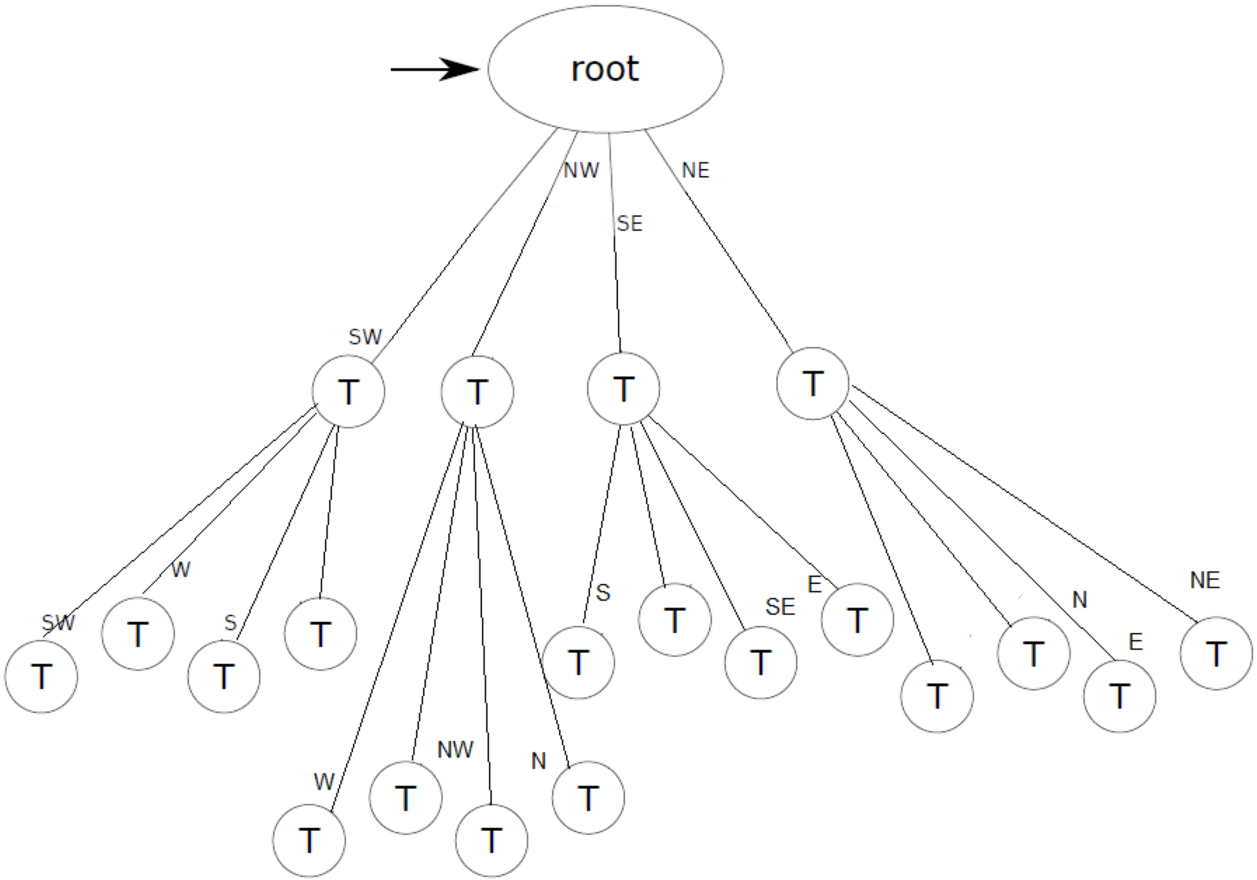


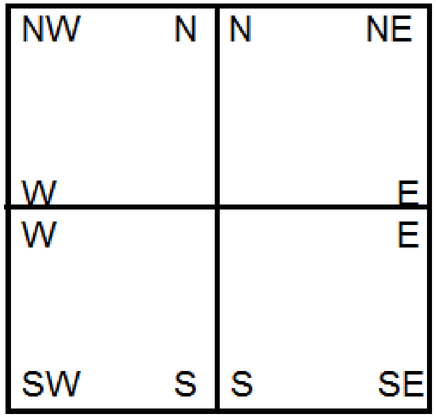


GENERATION OF 2D ELIMINATION TREE

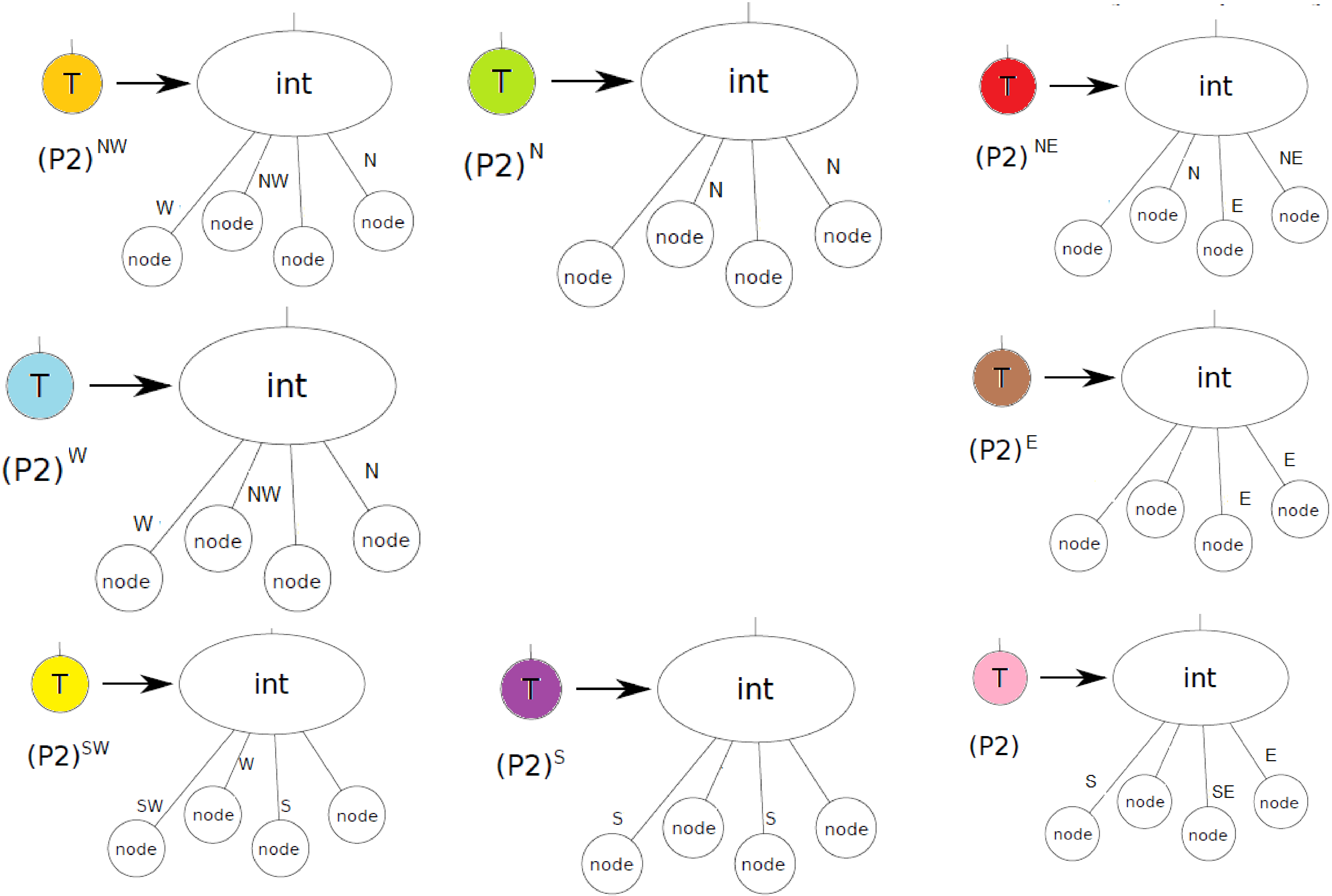


GENERATION OF 2D ELIMINATION TREE

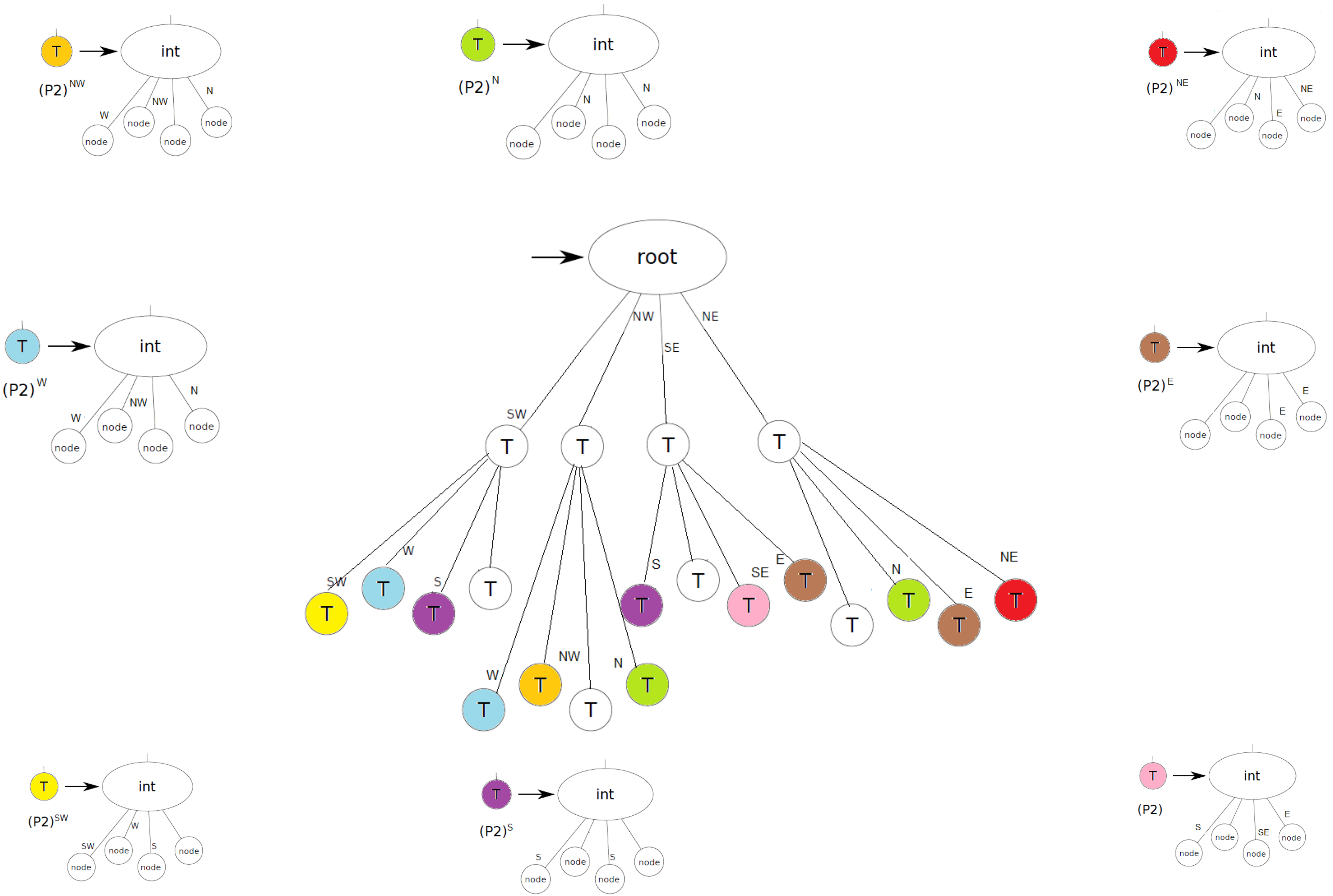




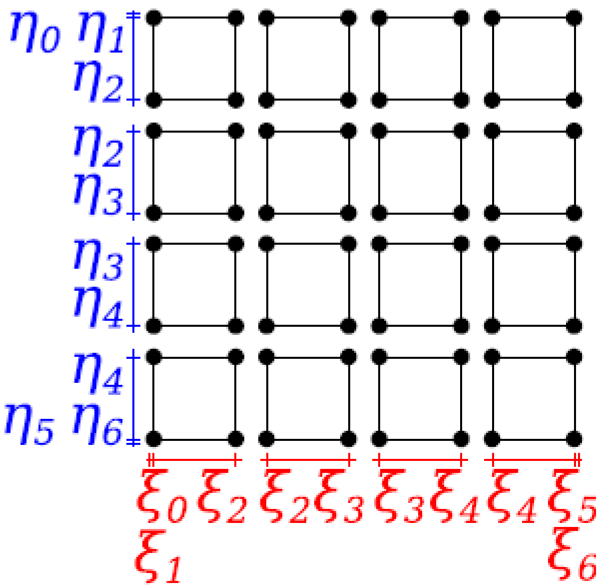
GENERATION OF 2D ELIMINATION TREE

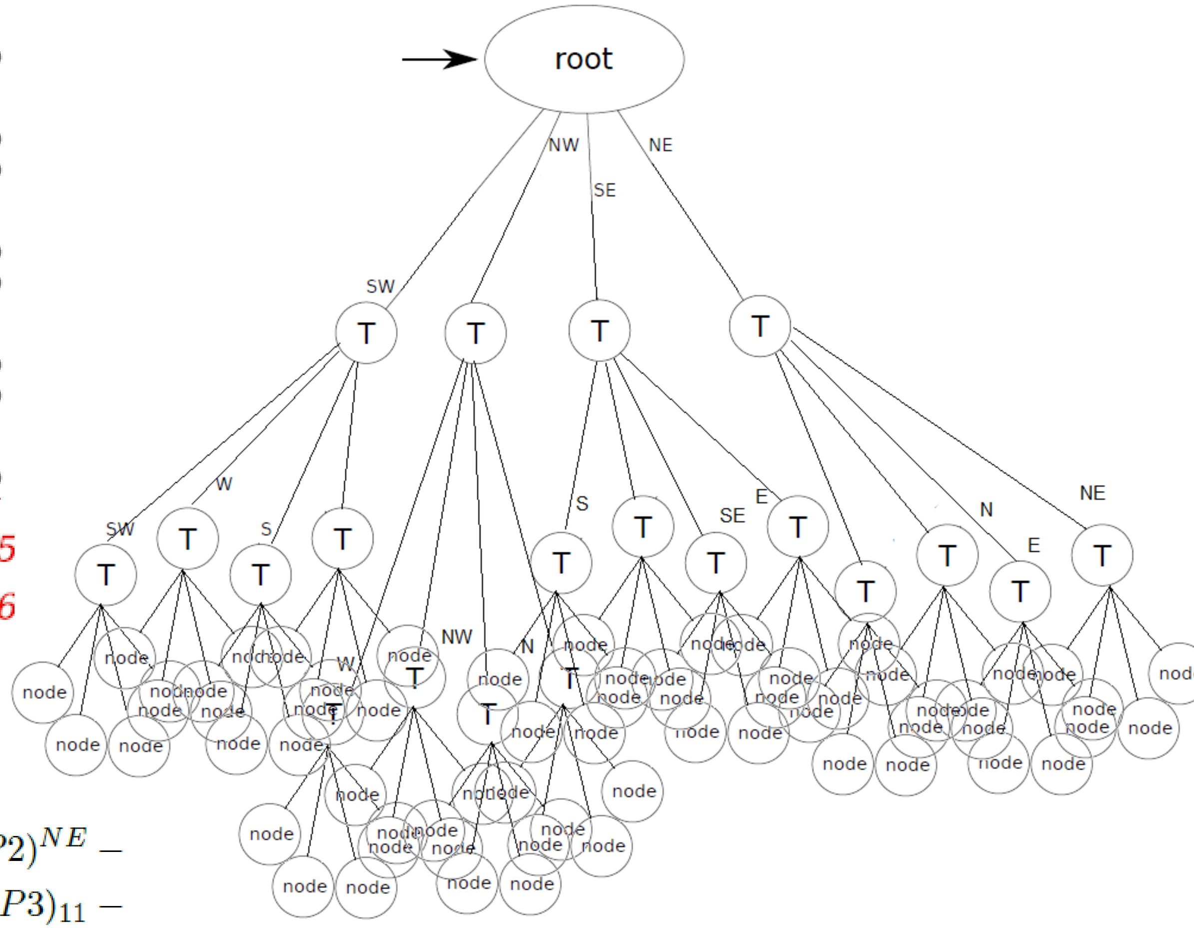


GENERATION OF 2D ELIMINATION TREE



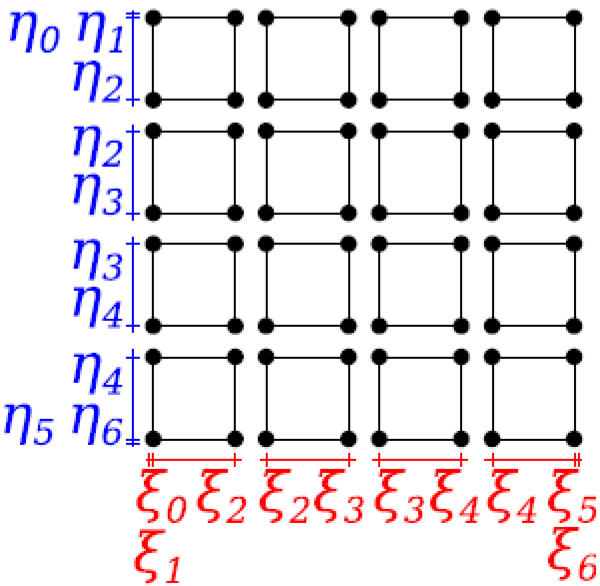
GENERATION OF 2D ELIMINATION TREE

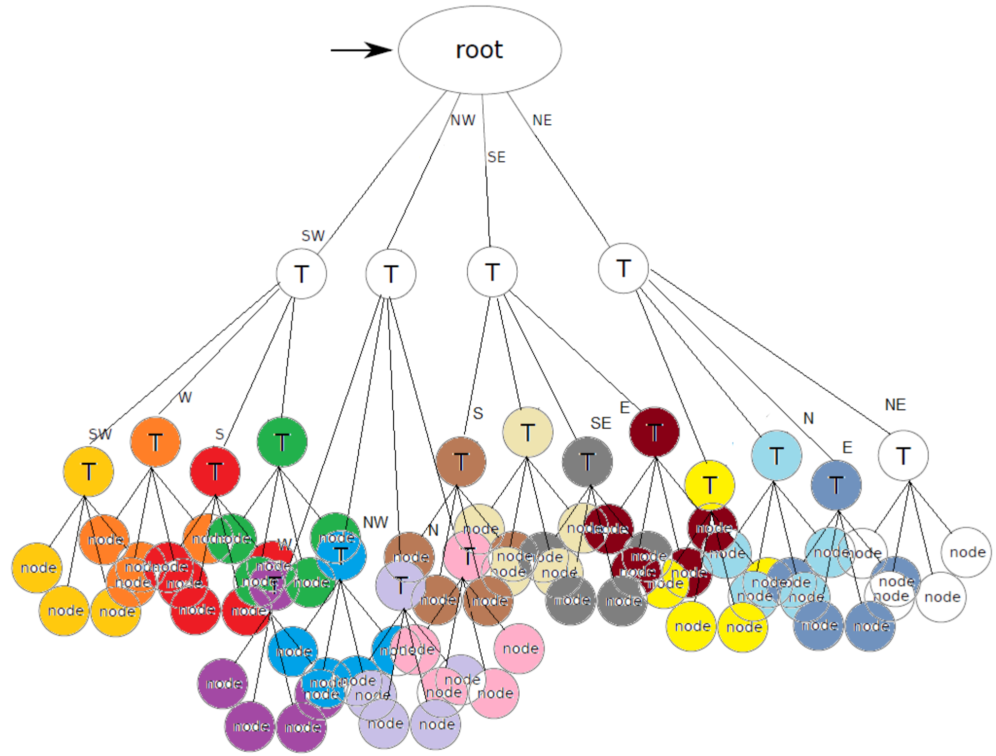


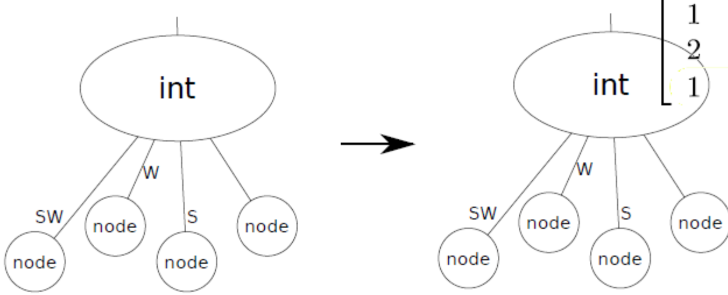


$$
\begin{aligned}
&(P1) - (P2)^{SW} - (P2)^{NW} - (P2)^{SE} - (P2)^{NE} - \\
&(P3)^{SW} - (P3)^{S} - (P3)^{W} - (P3)_{11} - \\
&(P3)^{SE} - (P3)^{S} - (P3)^{E} - (P3)_{12} - \\
&(P3)^{NE} - (P3)^{E} - (P3)^{N} - (P3)_{22} - \\
&(P3)^{NW} - (P3)^{W} - (P3)^{N} - (P3)_{21}
\end{aligned}
$$

GENERATION OF 2D ELIMINATION TREE

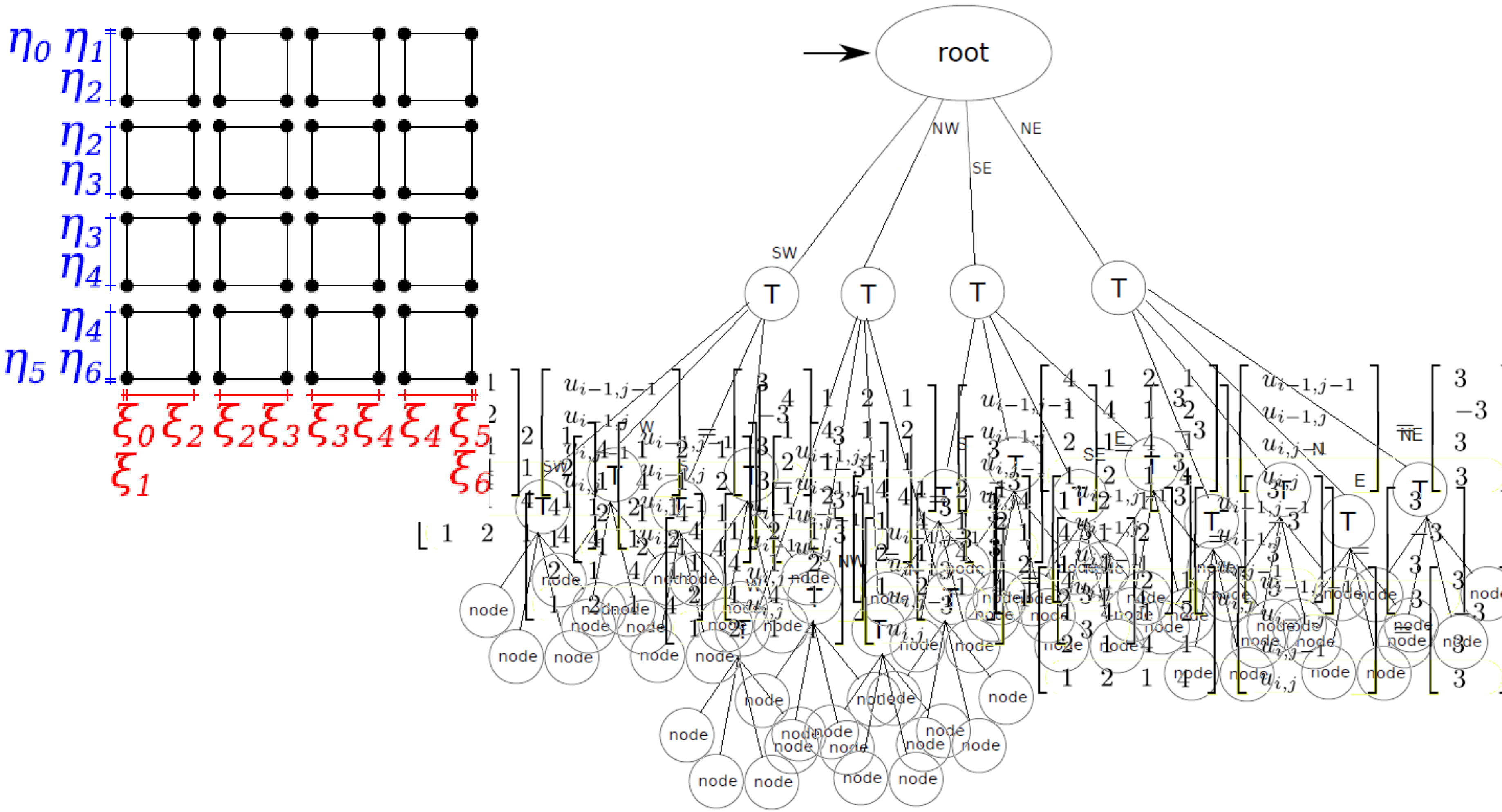




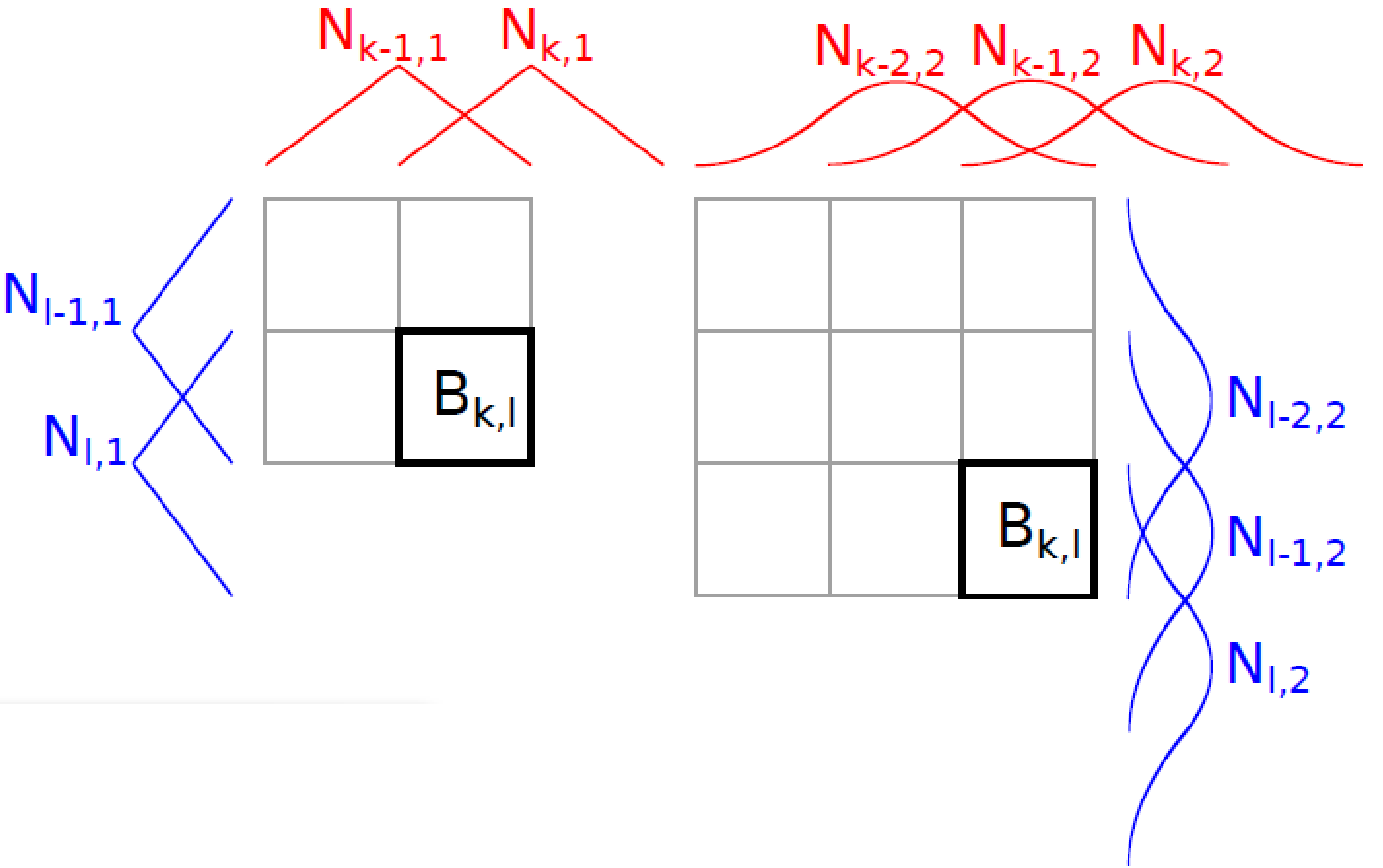


$$\begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i-1,j-1} \\ u_{i-1,j} \\ u_{i,j-1} \\ u_{i,j} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 3 \end{bmatrix}$$

GENERATION OF 2D ELIMINATION TREE



2D NUMERICAL INTEGRATION



2D NUMERICAL INTEGRATION

For p=2 there are 3x3=9 two dimensional B-splines

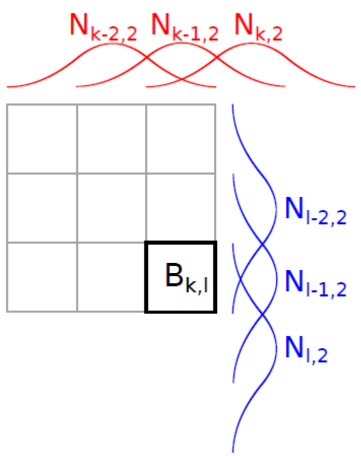
$b(B_{k-2,l-2;1},B_{k-2,l-2;1})$	…	$b(B_{k,l;1},B_{k-2,l-2;1})$
…	…	…
$b(B_{k-2,l-2;1},B_{k,l;1})$	…	$b(B_{k,l;1},B_{k,l;1})$

so we need to compute 3x3=9 two dimensional B-splines

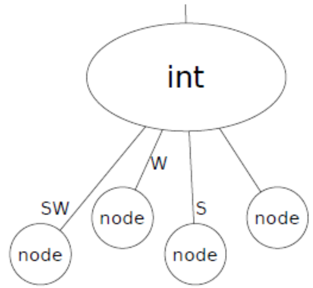
$B_{k,l;1}(x_1,x_2)=N_{k;1}(x_1)N_{l;1}(x_2)$	$B_{k,l-1;1}(x_1,x_2)=N_{k;1-1}(x_1)N_{l;1}(x_2)$	$B_{k,l-2;1}(x_1,x_2)=N_{k;1}(x_1)N_{l-2;1}(x_2)$
$B_{k-1,l;1}(x_1,x_2)=N_{k-1;1}(x_1)N_{l;1}(x_2)$	$B_{k-1,l-1;1}(x_1,x_2)=N_{k-1;1-1}(x_1)N_{l;1}(x_2)$	$B_{k-1,l-2;1}(x_1,x_2)=N_{k-1;1-1}(x_1)N_{l-2;1}(x_2)$
$B_{k-2,l;1}(x_1,x_2)=N_{k-2;1}(x_1)N_{l;1}(x_2)$	$B_{k-2,l-1;1}(x_1,x_2)=N_{k-2;1-1}(x_1)N_{l;1}(x_2)$	$B_{k-2,l-2;1}(x_1,x_2)=N_{k-2;1-1}(x_1)N_{l-2;1}(x_2)$

so we need to compute 3+3=6 one dimensional B-splines

$N_{k;1}(x_1)$	$N_{k-1;1}(x_1)$	$N_{k-2;1}(x_1)$
$N_{l;1}(x_2)$	$N_{l-1;1}(x_2)$	$N_{l-2;1}(x_2)$



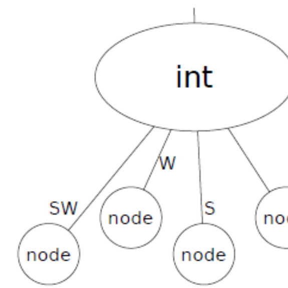
GENERATION OF 2D ELIMINATION TREE

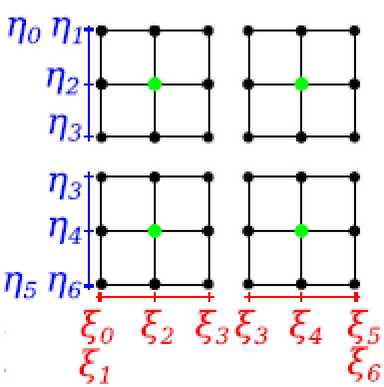


$$\begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i-1,j-1} \\ u_{i-1,j} \\ u_{i,j-1} \\ u_{i,j} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 3 \end{bmatrix} \qquad \begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i-1,j} \\ u_{i-1,j+1} \\ u_{i,j} \\ u_{i,j+1} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i,j-1} \\ u_{i,j} \\ u_{i+1,j-1} \\ u_{i+1,j} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 3 \end{bmatrix} \qquad \begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \\ u_{i+1,j} \\ u_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 3 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 16 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 8 & 2 & 0 & 2 & 1 & 1 & 0 & 0 \\ 2 & 2 & 8 & 2 & 0 & 0 & 1 & 1 & 0 \\ 2 & 0 & 2 & 8 & 2 & 0 & 0 & 1 & 1 \\ 2 & 2 & 0 & 2 & 8 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 1 & 4 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 0 & 0 & 4 & 0 \\ 2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i-1,j} \\ u_{i,j+1} \\ u_{i+1,j} \\ u_{i,j-1} \\ u_{i-1,j-1} \\ u_{i-1,j+1} \\ u_{i+1,j+1} \\ u_{i+1,j-1} \end{bmatrix} = \begin{bmatrix} 9 \\ 0 \\ 3 \\ 6 \\ 6 \\ 3 \\ -3 \\ 3 \\ 3 \end{bmatrix}$$



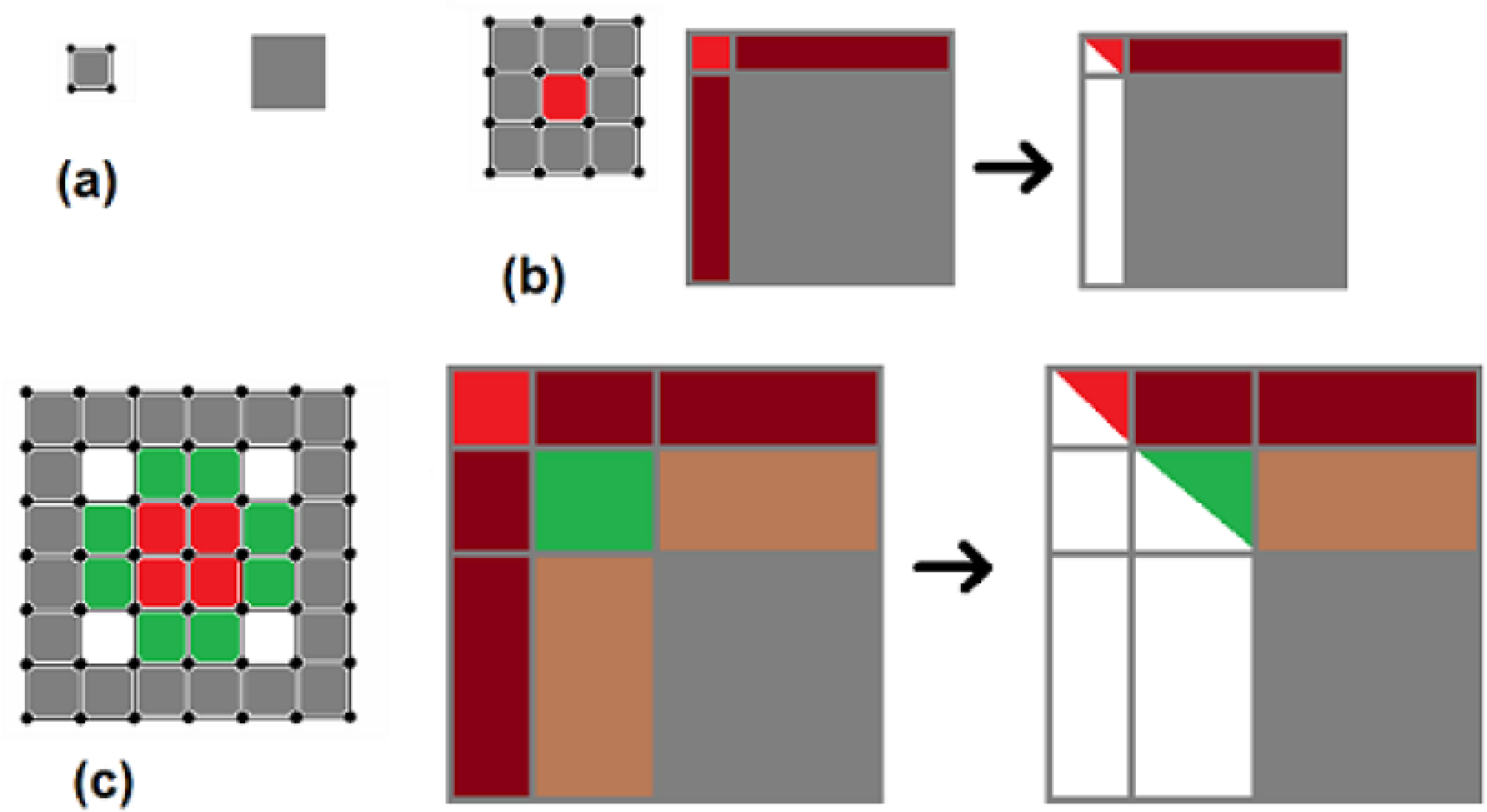


$$\begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i-1,j-1} \\ u_{i-1,j} \\ u_{i,j-1} \\ u_{i,j} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 3 \end{bmatrix} \qquad \begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i,j-1} \\ u_{i,j} \\ u_{i+1,j-1} \\ u_{i+1,j} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 16 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 8 & 2 & 0 & 2 & 1 & 1 & 0 & 0 \\ 2 & 2 & 8 & 2 & 0 & 0 & 1 & 1 & 0 \\ 2 & 0 & 2 & 8 & 2 & 0 & 0 & 1 & 1 \\ 2 & 2 & 0 & 2 & 8 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 1 & 4 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 0 & 0 & 4 & 0 \\ 2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i-1,j} \\ u_{i,j+1} \\ u_{i+1,j} \\ u_{i,j-1} \\ u_{i-1,j-1} \\ u_{i-1,j+1} \\ u_{i+1,j+1} \\ u_{i+1,j-1} \end{bmatrix} = \begin{bmatrix} 9 \\ 0 \\ 3 \\ 6 \\ 6 \\ 3 \\ -3 \\ 3 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i-1,j} \\ u_{i-1,j+1} \\ u_{i,j} \\ u_{i,j+1} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 3 \end{bmatrix} \qquad \begin{bmatrix} 4 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \\ u_{i+1,j} \\ u_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 3 \end{bmatrix}$$

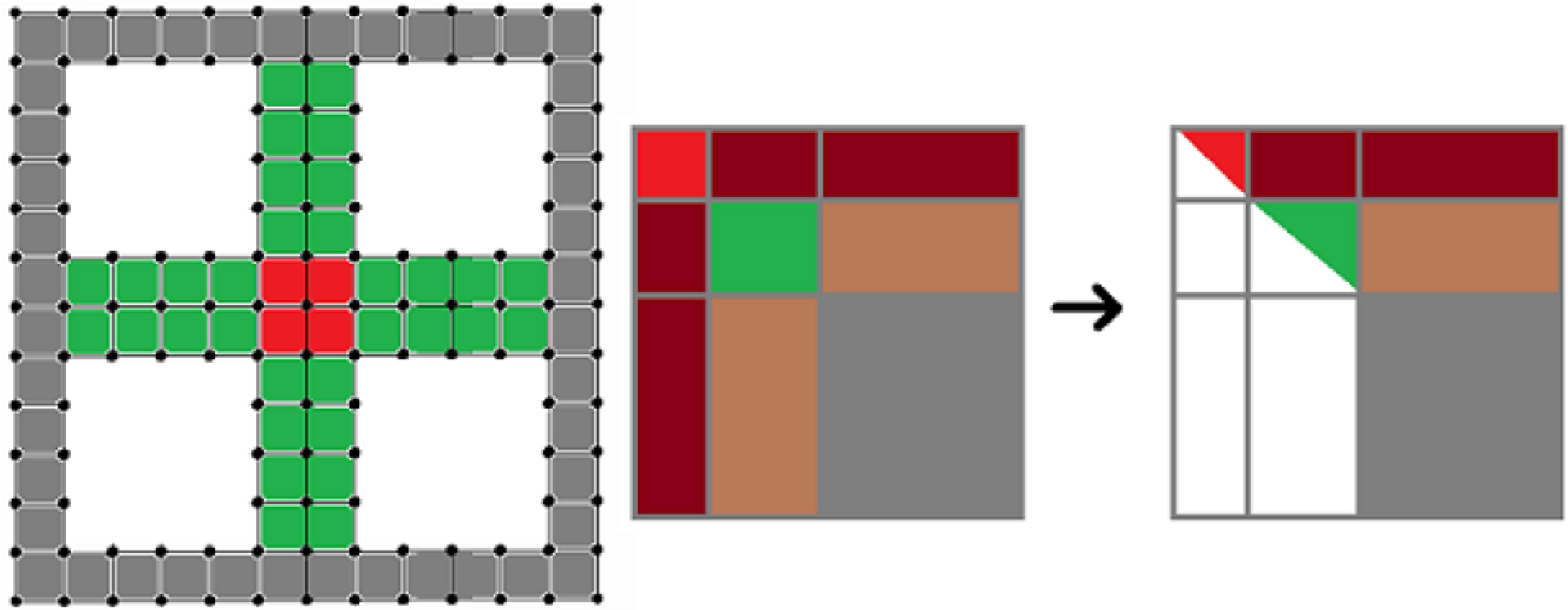
2D ELIMINATION



Tasks related to single row subtractions

2D ELIMINATION

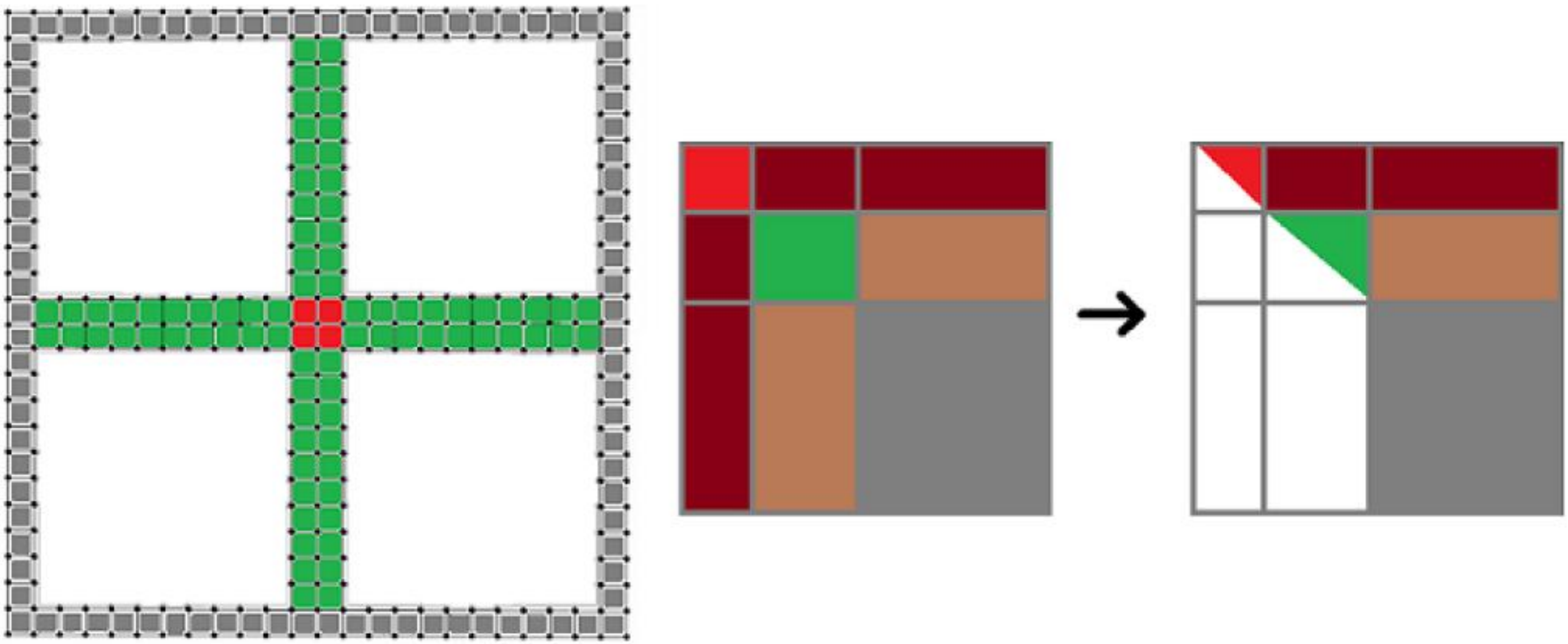
(d)



Tasks related to single row subtractions

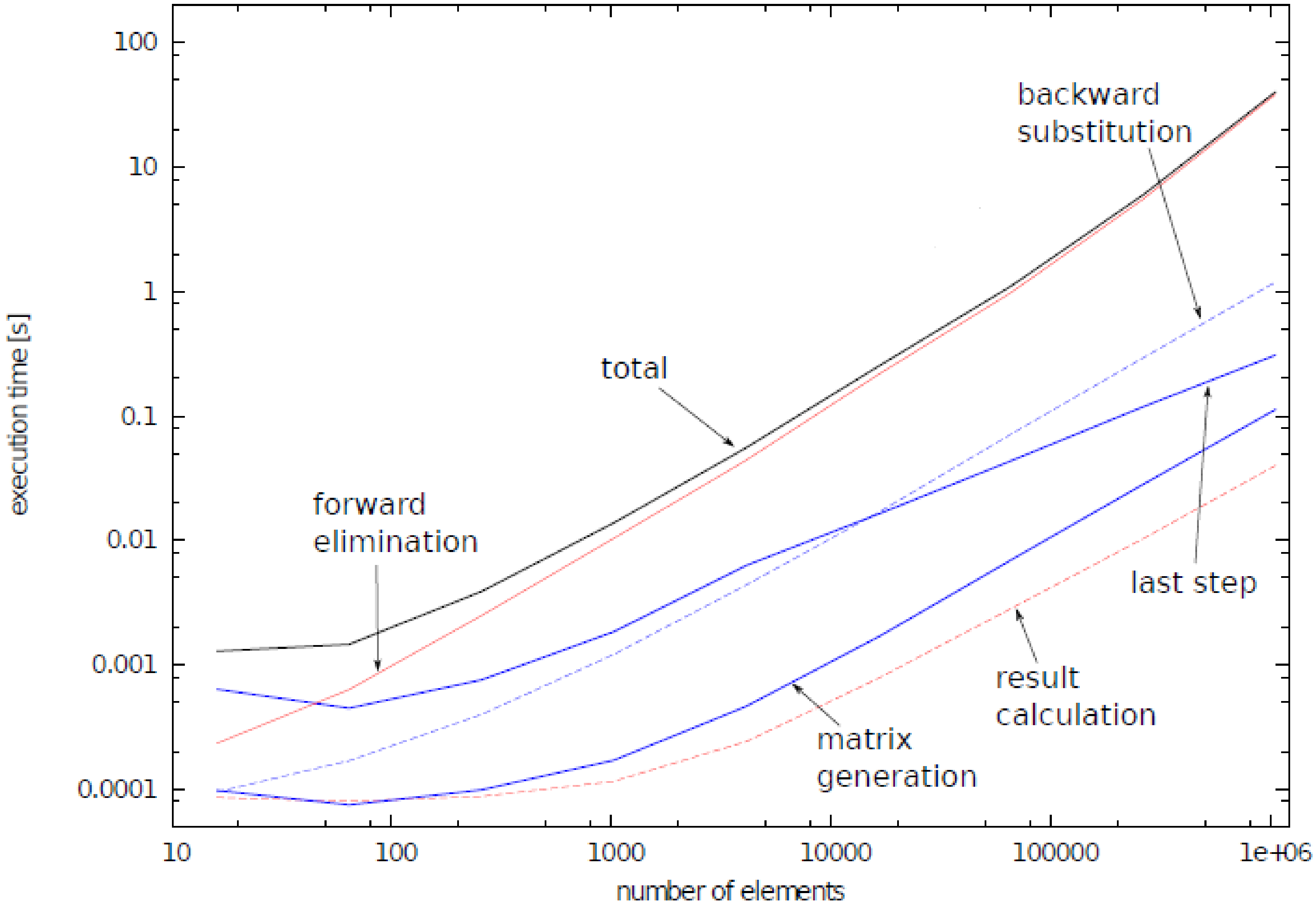
2D ELIMINATION

(e)



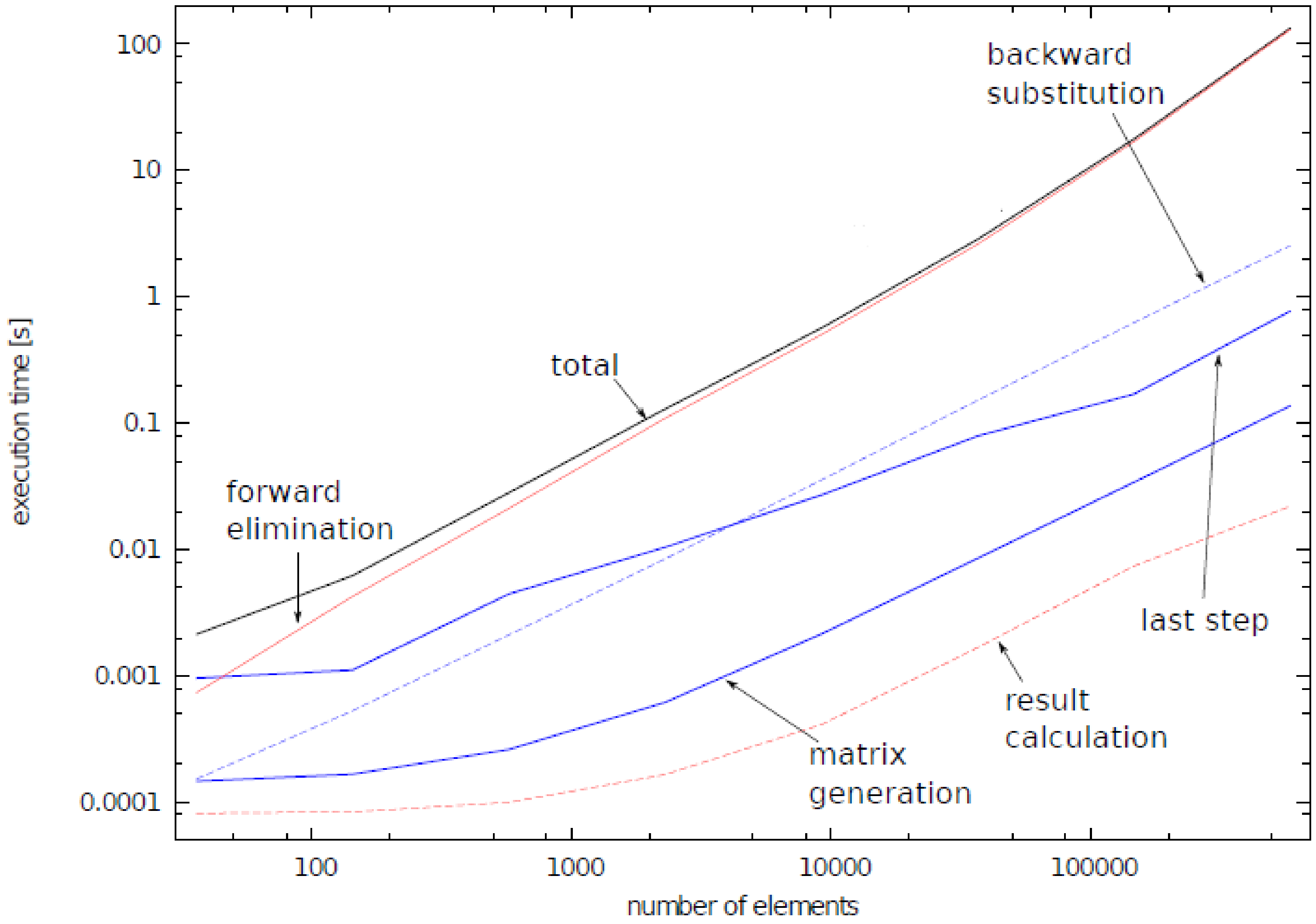
Tasks related to single row subtractions

2D NUMERICAL RESULTS
LINEAR B-SPLINES



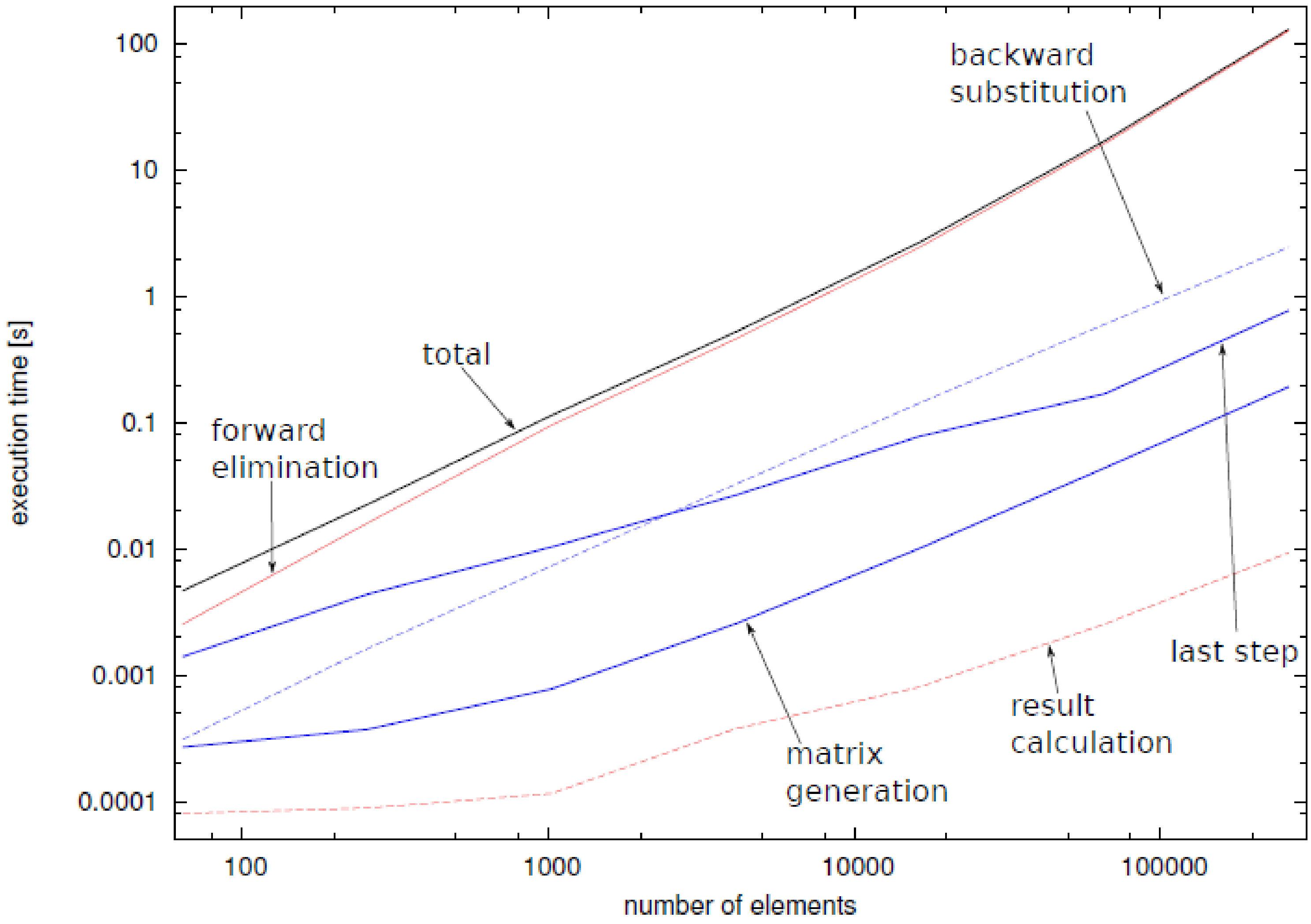
NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock

2D NUMERICAL RESULTS
QUADRATIC B-SPLINES



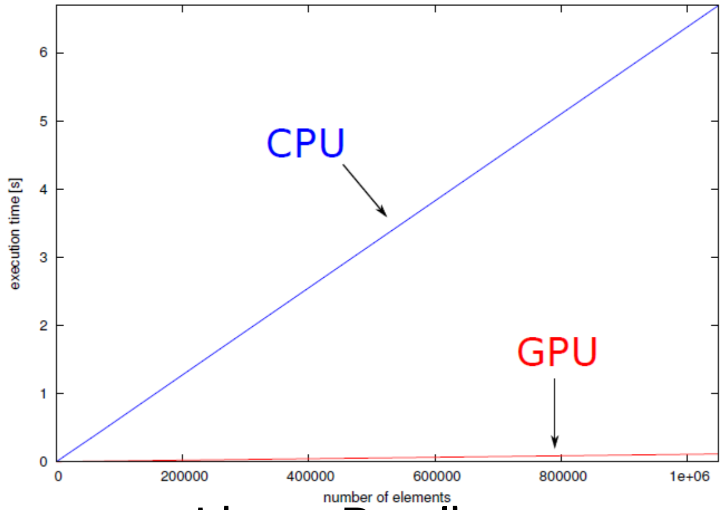
NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock

2D NUMERICAL RESULTS
CUBIC B-SPLINES

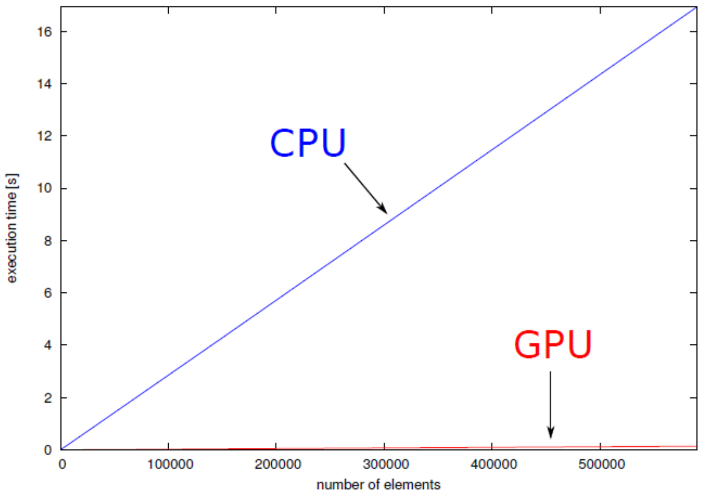


NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock

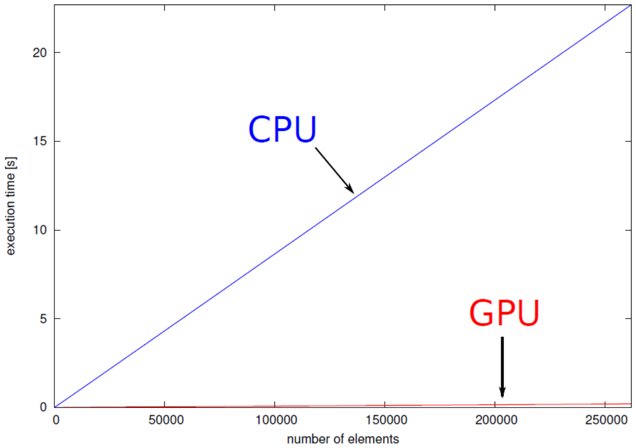
COMPARISON WITH CPU MUMPS ASSEMBLY



Linear B-splines



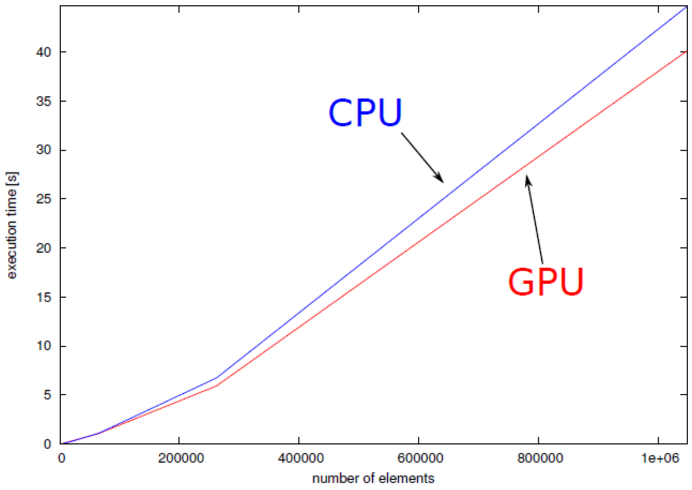
Quadratic B-splines



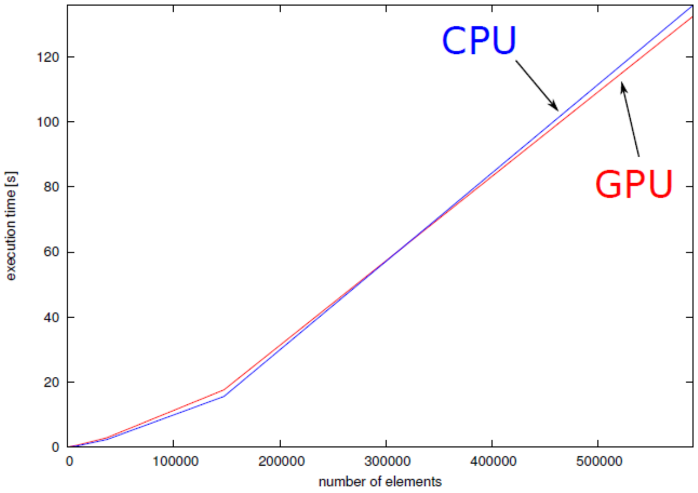
Cubic B-splines

NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock
Intel(R) Core(TM)2 Quad CPU Q9400 with 2.66GHz clock, 8GB of memory

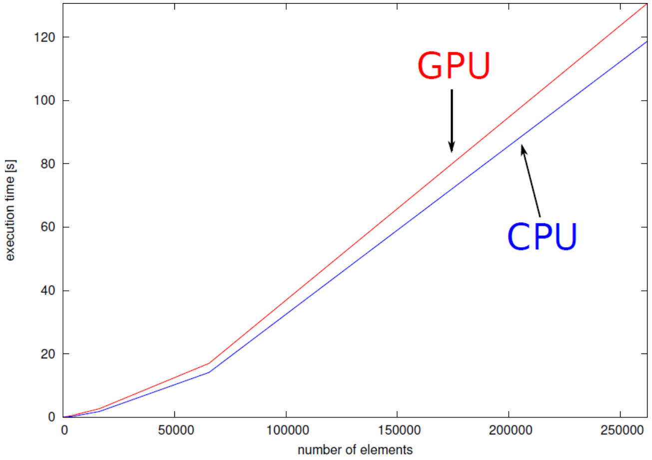
COMPARISON WITH CPU MUMPS FACTORIZATION



Linear B-splines



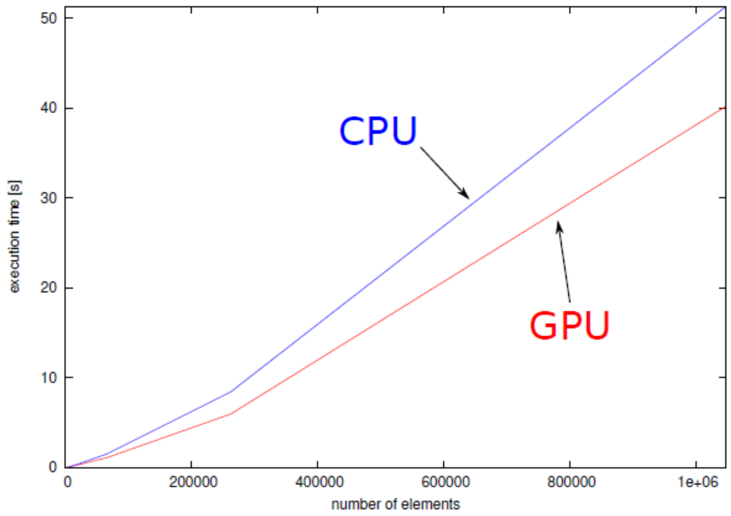
Quadratic B-splines



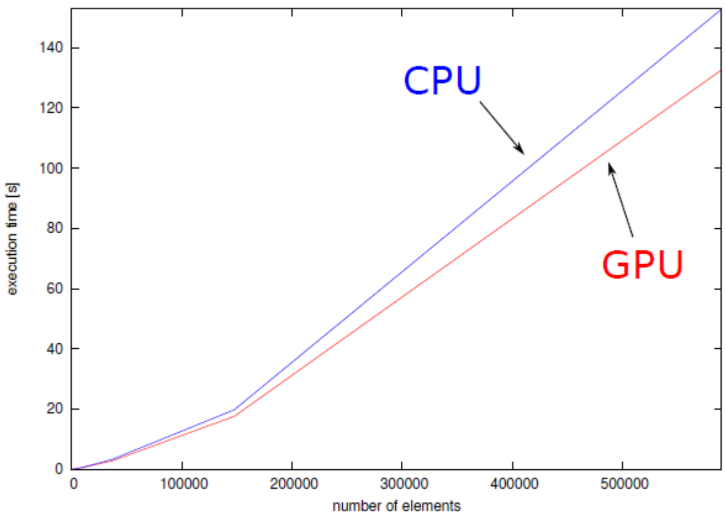
Cubic B-splines

NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock
Intel(R) Core(TM)2 Quad CPU Q9400 with 2.66GHz clock, 8GB of memory

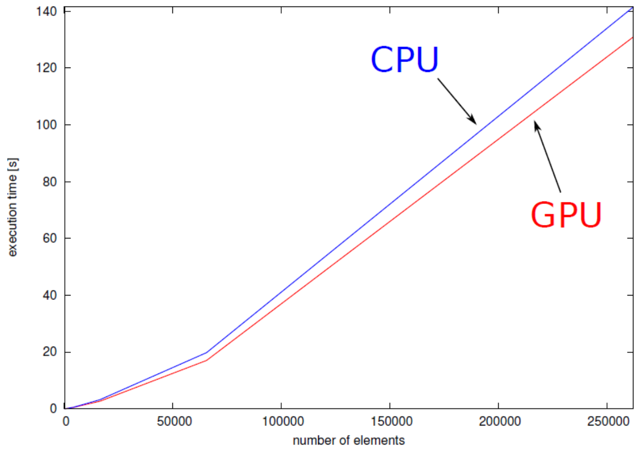
COMPARISON WITH CPU MUMPS
TOTAL TIME



Linear B-splines



Quadratic B-splines



Cubic B-splines

NVidia Tesla c2070, 6GB memory, 448 CUDA cores, each one with 1.15GHz clock
Intel(R) Core(TM)2 Quad CPU Q9400 with 2.66GHz clock, 8GB of memory

COMPUTATIONAL COST OF SERIAL C^0 and C^{p-1} SOLVERS

COMPUTATIONAL COST OF SERIAL C^0 and C^{p-1} SOLVERS

List of crimes

- Consider a regular grid with same number of elements in each direction
- Number of elements assumed to be sufficiently large
- Ignore orthogonality between basis functions sharing same support
- To simplify analysis, only consider limiting cases of continuity C^0 and C^{p-1}
- Numerical results consider Laplace equation over a unit cube geometry

COMPUTATIONAL COST OF SERIAL C^0 and C^{p-1} SOLVERS

FLOPS (Time) Estimates

FLOPS (2D, C^0) $= 2^{2s}p^6 + \sum_{i=1}^{s-1} 2^{2(s-i)}2^{3i}p^3 = \mathcal{O}(Np^4) + \mathcal{O}(N^{1.5})$

FLOPS (2D, C^{p-1}) $= 2^{2s}p^4 + \sum_{i=1}^{s-1} 2^{2(s-i)}2^{3i}p^6 = \mathcal{O}(N^{1.5}p^3)$

FLOPS (3D, C^0) $= 2^{3s}p^9 + \sum_{i=1}^{s-1} 2^{3(s-i)}2^{6i}p^6 = \mathcal{O}(Np^6) + \mathcal{O}(N^2)$

FLOPS (3D, C^{p-1}) $= 2^{3s}p^6 + \sum_{i=1}^{s-1} 2^{3(s-i)}2^{6i}p^9 = \mathcal{O}(N^2p^3)$

For large problems, C^0 continuity is p^3 faster

N.Collier, D. Pardo, L. Dalcin, M. Paszynski, V.Calo; (2012) The cost of continuity: a study of performance of isogeometric finite elements using direct solvers, **Computer Methods in Applied Mechanics and Engineering,** 213-216, p. 353-361

COMPUTATIONAL COST OF SHARED MEMORY C^0 and C^{p-1} SOLVERS

COMPUTATIONAL COST OF SHARED MEMORY C^0 and C^{p-1} SOLVERS

List of parallel crimes

- Assume infinitely large number of processors.
- Assume zero communication cost.
- Assume infinite memory.

COMPUTATIONAL COST OF SHARED MEMORY C^0 and C^{p-1} SOLVERS

$$\text{FLOPS (1D, } C^0) = p^2 + \sum_{i=1}^{s-1} 1 = \mathcal{O}(p^2) + \mathcal{O}(\log(N/p)),$$

$$\text{FLOPS (1D, } C^{p-1}) = p + \sum_{i=1}^{s-1} p^2 = \mathcal{O}(p^2 \log(N/p)),$$

$$\text{FLOPS (2D, } C^0) = p^4 + \sum_{i=1}^{s-1} 2^{2i} p^2 = \mathcal{O}(p^4) + \mathcal{O}(N)$$

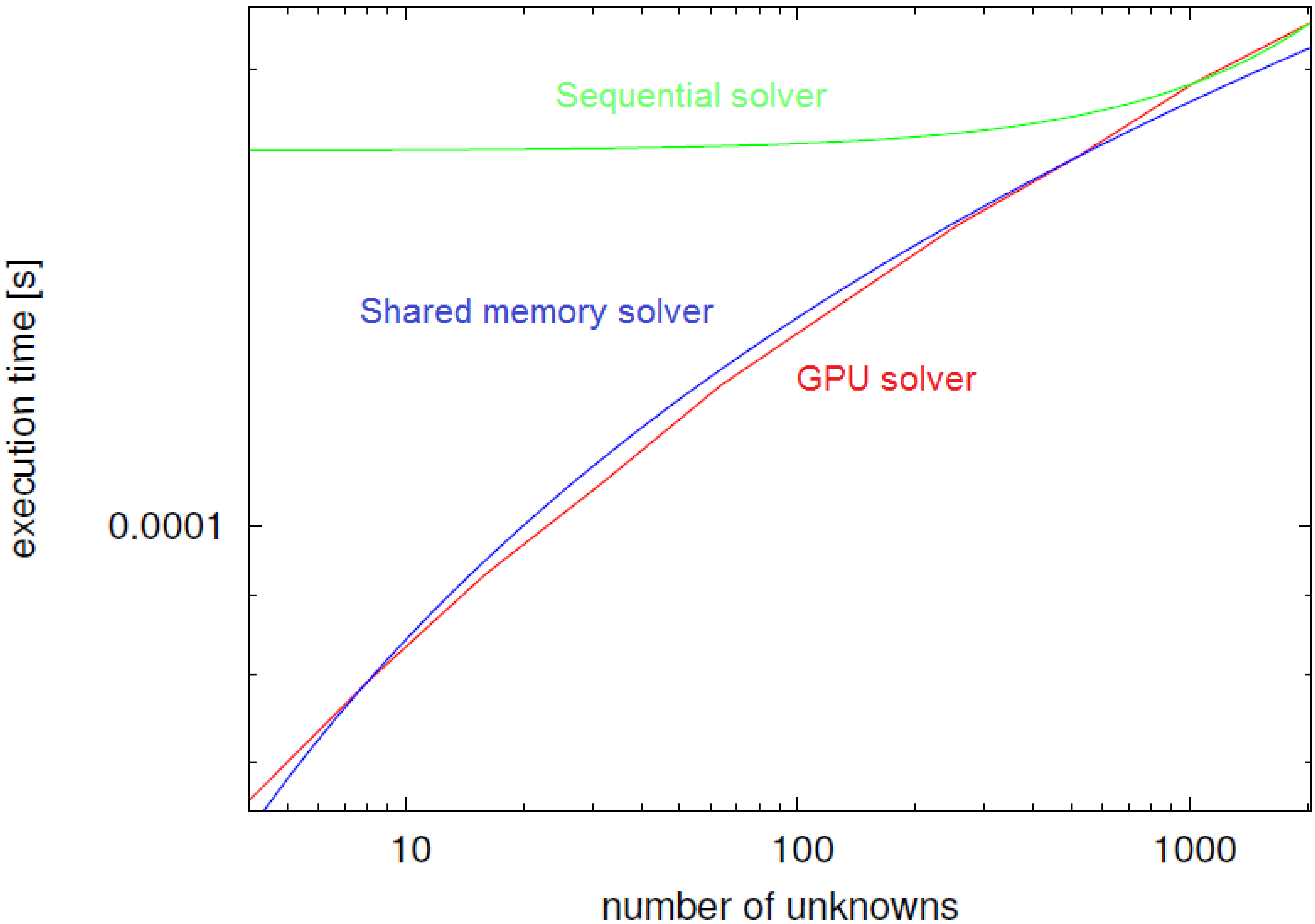
$$\text{FLOPS (2D, } C^{p-1}) = p^2 + \sum_{i=1}^{s-1} 2^{2i} p^4 = \mathcal{O}(Np^2)$$

$$\text{FLOPS (3D, } C^0) = p^6 + \sum_{i=1}^{s-1} 2^{4i} p^4 = \mathcal{O}(p^6) + \mathcal{O}(N^{1.33})$$

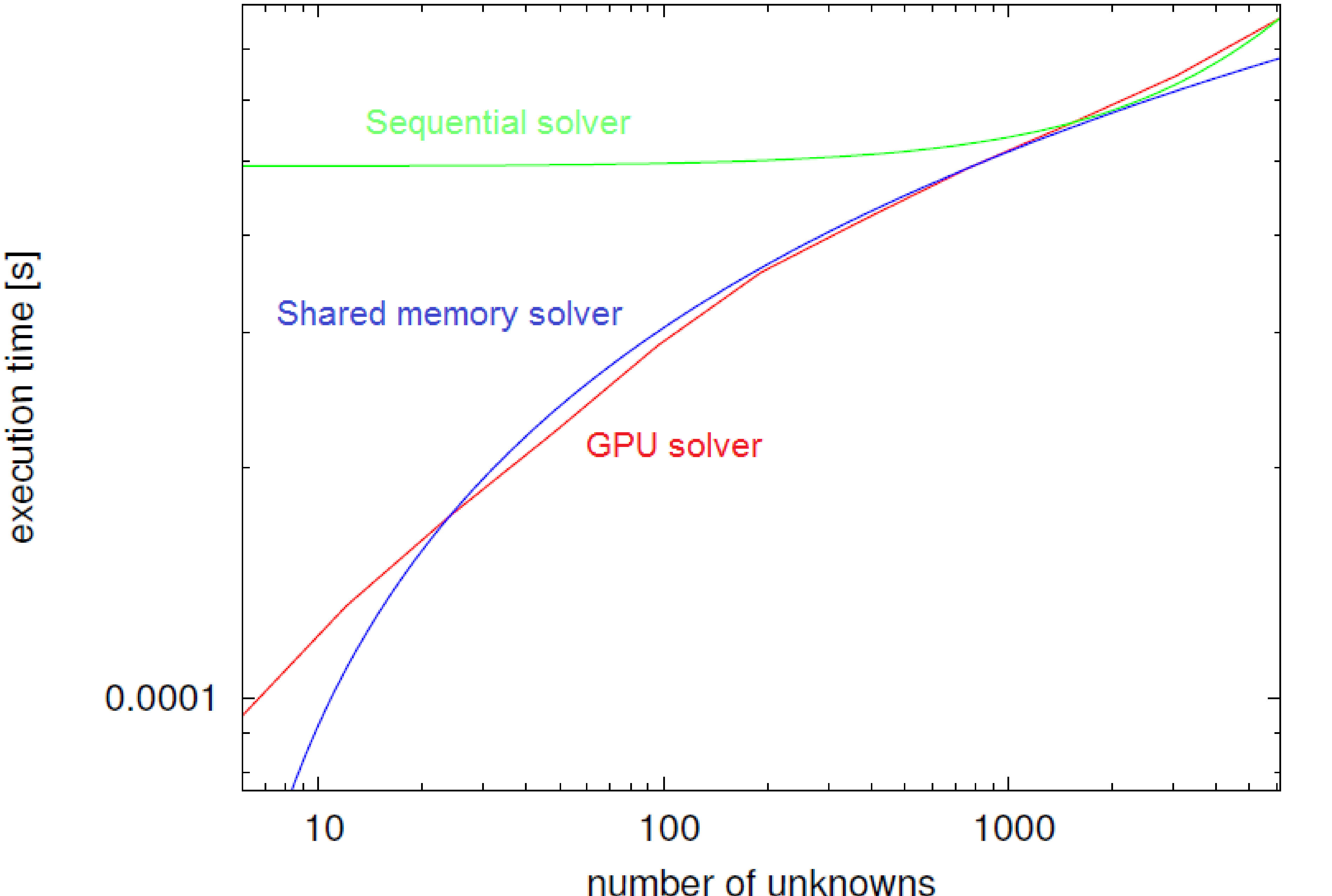
$$\text{FLOPS (3D, } C^{p-1}) = p^4 + \sum_{i=1}^{s-1} 2^{4i} p^6 = \mathcal{O}(N^{1.33} p^2)$$

For large problems, C^0 continuity is p^2 faster

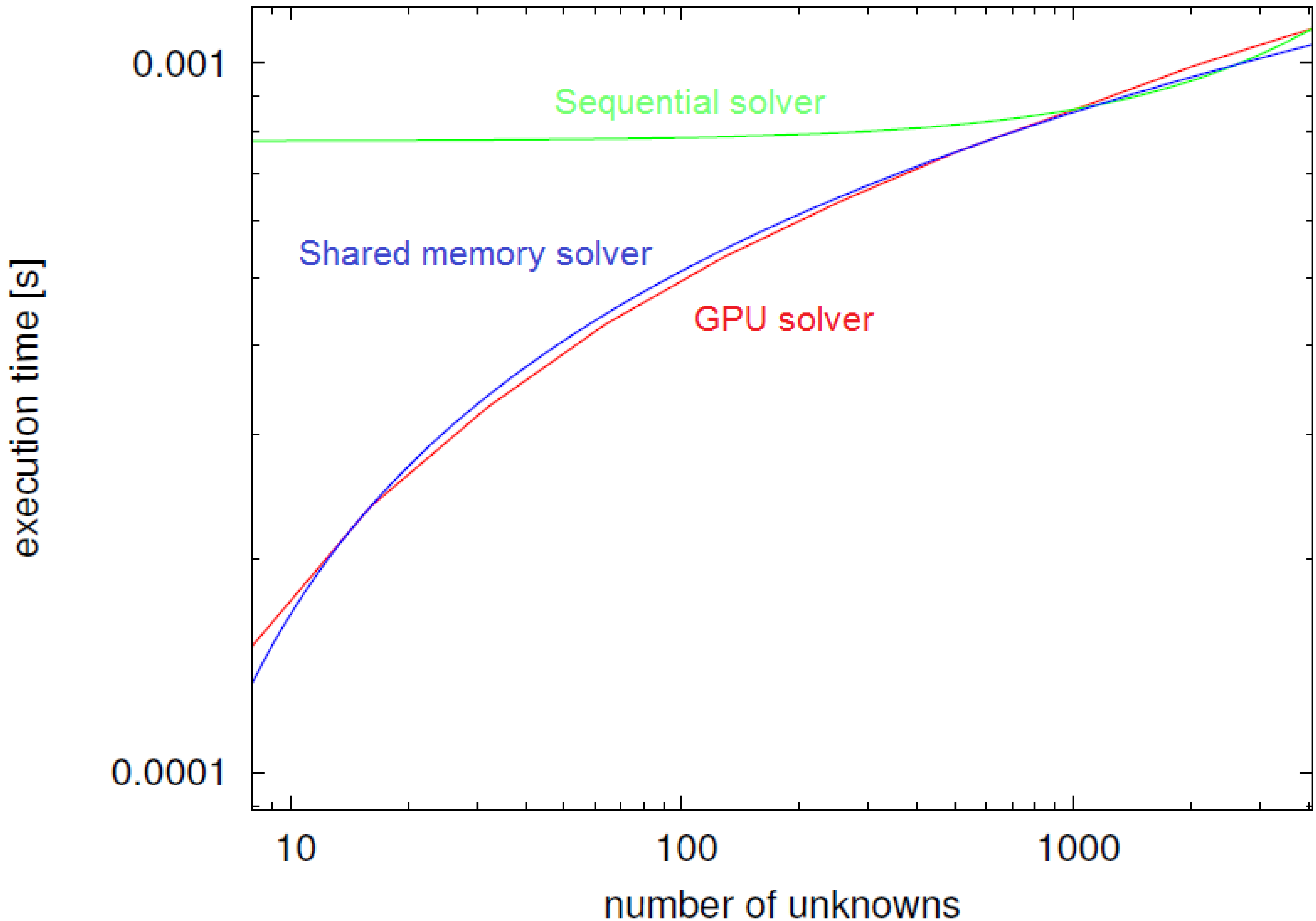
1D LINEAR B-SPLINES



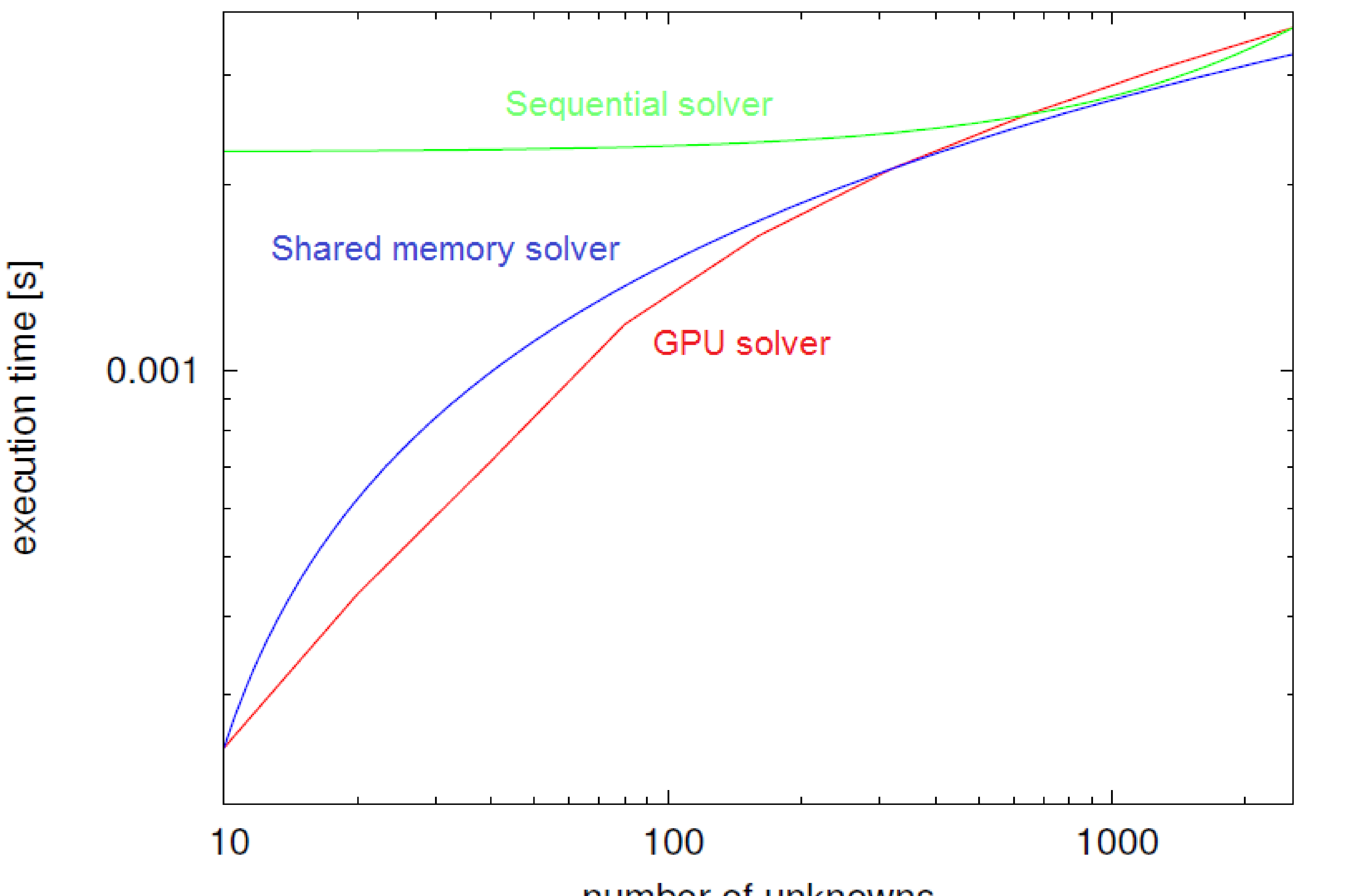
1D QUADRATIC B-SPLINES



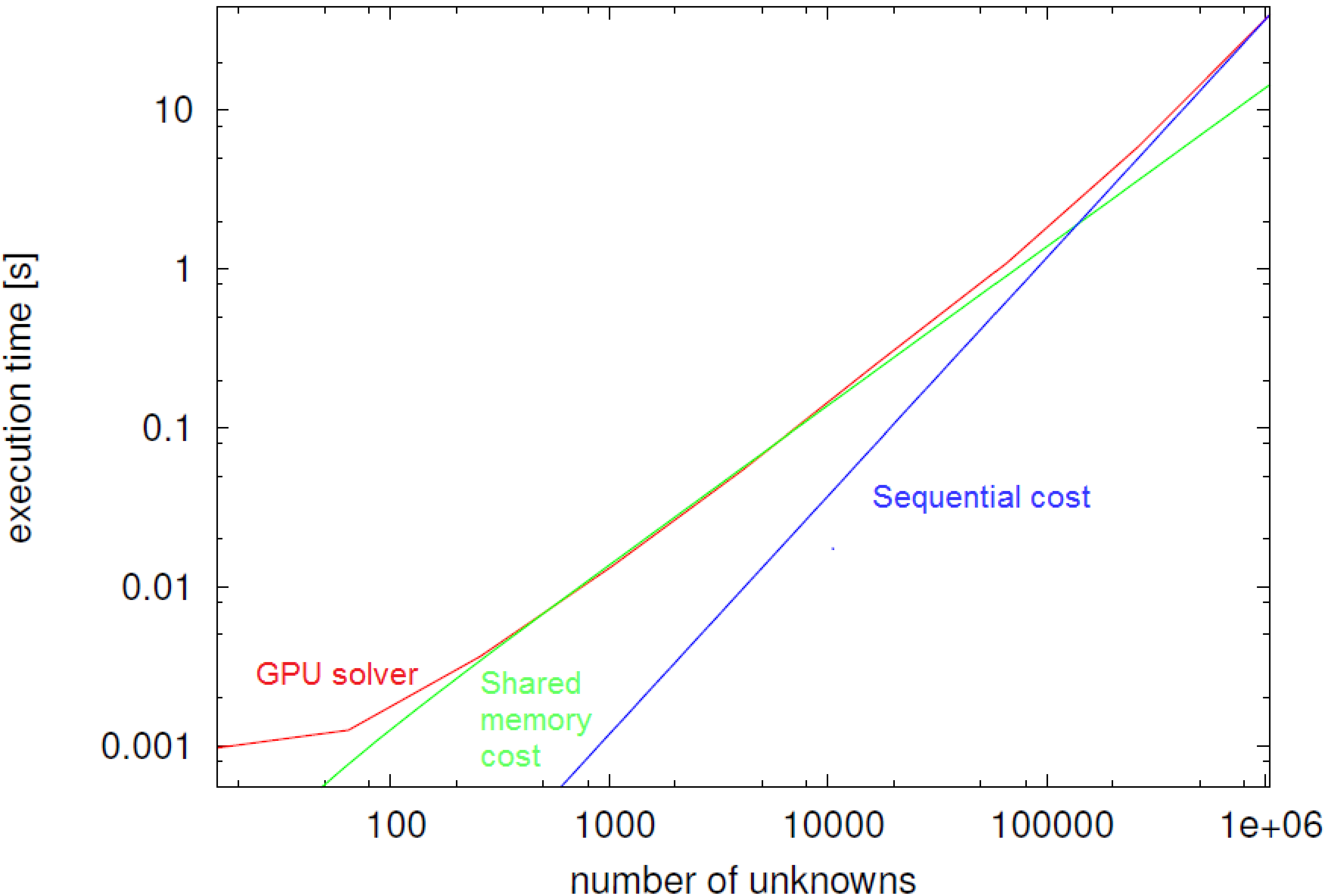
1D CUBIC B-SPLINES



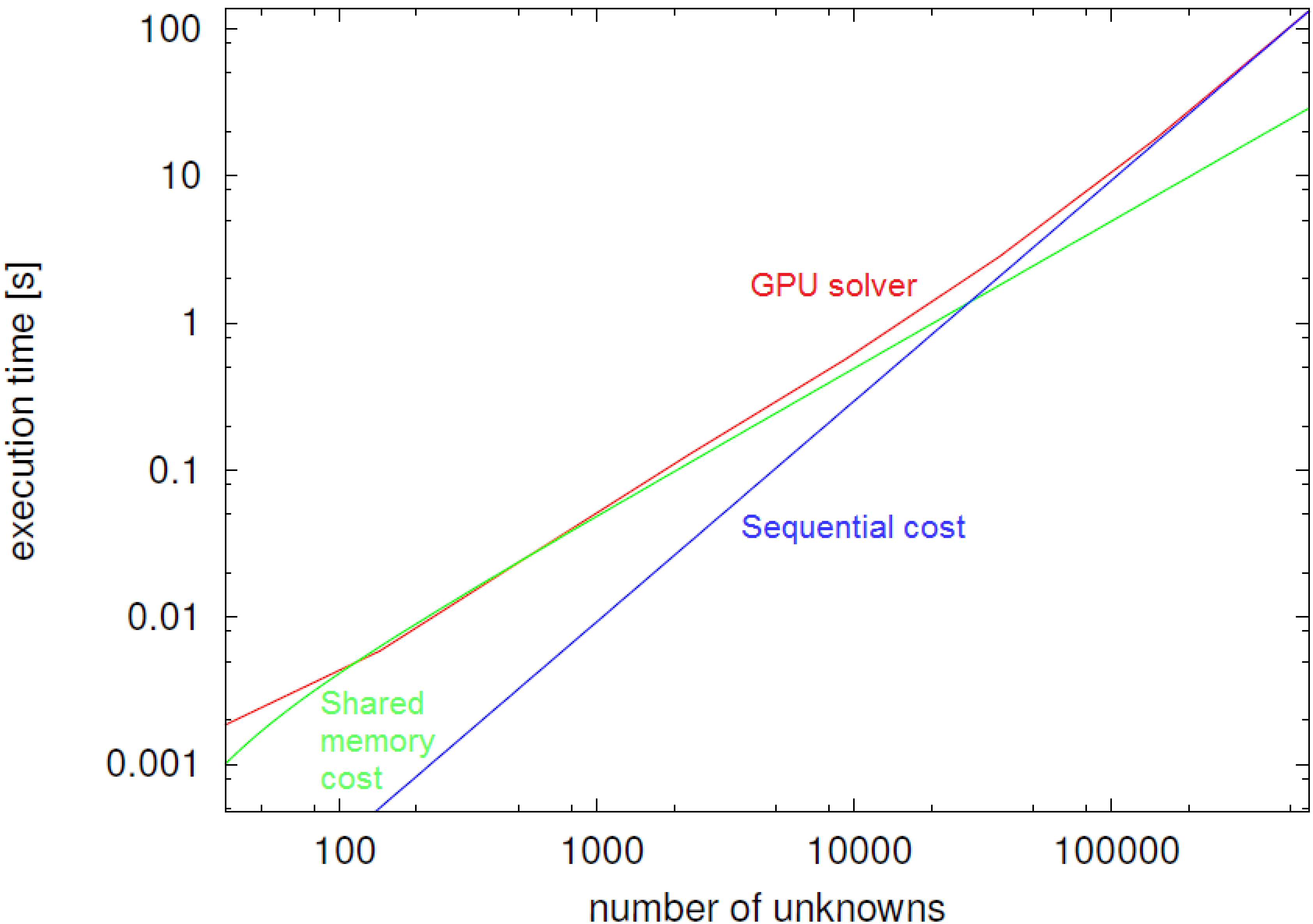
1D QINTIC B-SPLINES



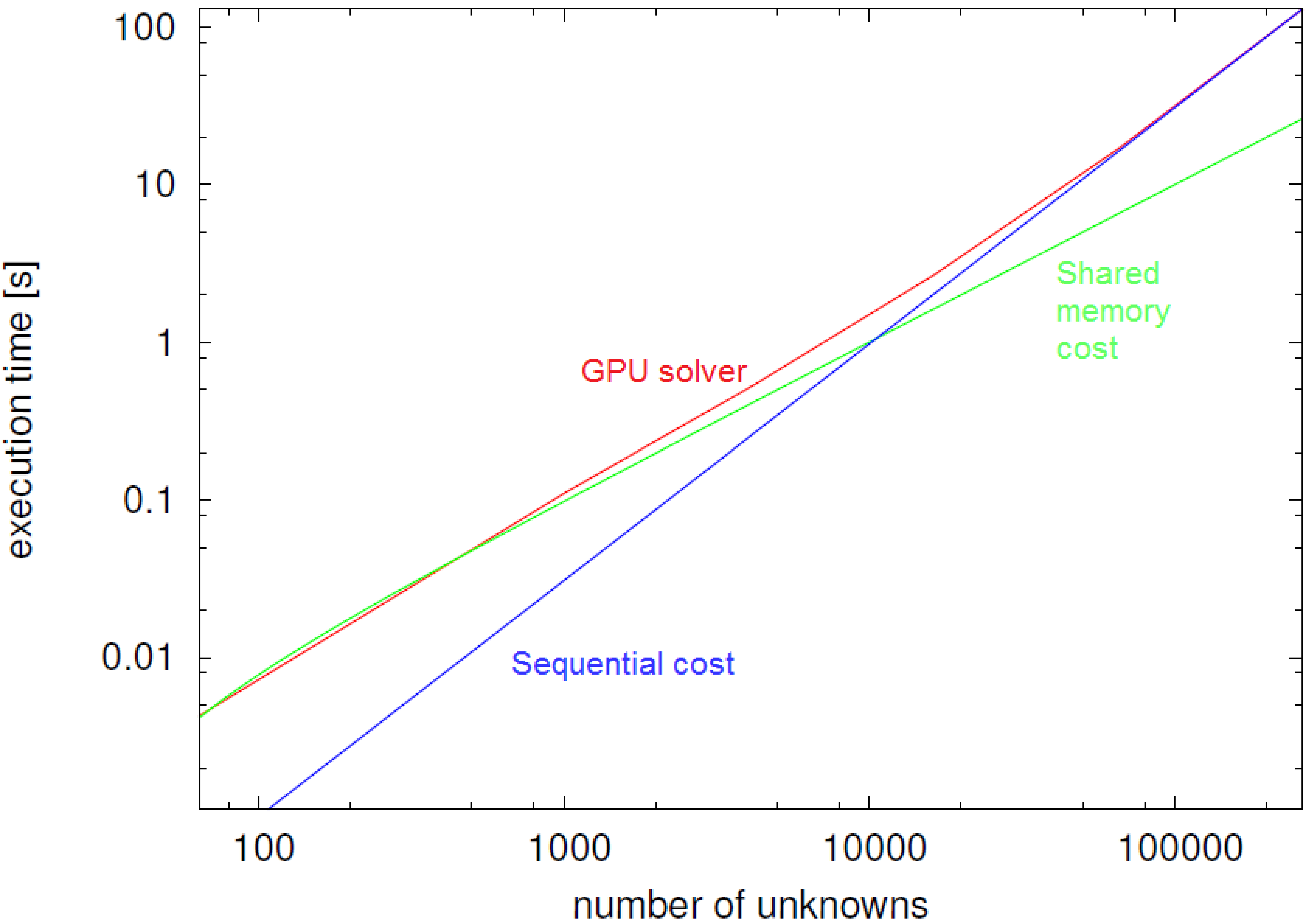
2D LINEAR B-SPLINES



2D QUADRATIC B-SPLINES



2D CUBIC B-SPLINES



PAPERS

Multi-frontal solver for IGA discretizations in GPUs

K. Kuznik, M. Paszynski, V. Calo, D.Pardo (2013) Multi-frontal solver for IGA discretizations in GPUs, submitted to **Computers and Mathematics with Applications**

Graph grammar based 2D isogeometric FEM solver:

K. Kuznik, M. Paszynski, V. Calo (2012) Graph Grammar-Based Multi-Frontal Parallel Direct Solver for Two-Dimensional Isogeometric Analysis, ***Procedia Computer Science*****,** 9, p. 1454-1463

Computational costs for 1D/2D/3D sequential isogeometric FEM:

N. Collier, D. Pardo, L. Dalcin, M. Paszynski, V.Calo; (2012) The cost of continuity: a study of performance of isogeometric finite elements using direct solvers, ***Computer Methods in Applied Mechanics and Engineering*****,** 213-216, p. 353-361

Graph grammar based 2D FEM solver for distributed memory linux cluster:

M. Paszyński, R. Schaefer (2010) Graph grammar-driven parallel partial differential equation solver ***Concurrency & Computations, Practise & Experience*** 22 (9) p.1063-1097

Graph grammar based 3D FEM solver for distributed memory linux cluster:

M. Paszyński, D. Pardo, A. Paszynska (2010) Parallel multi-frontal solver for p adaptive finite element modeling of multi-physics computational problems, ***Journal of Computational Science*** 1 (1) p.48-54