METODE DIRECTE PARALELE DE REZOLVARE A SISTEMELOR DE ECUAŢII LINIARE

Problema: rezolvarea sistemului $Ax = b$, unde A este o matrice $m \times n$, b este un vector m-dimensional, x este vectorul n-dimensional necunoscut.

---

0. Metode secvenţiale acceptate:

- calcul pe bază de determinanţi (pt. sisteme de dimeniuni mici),
- eliminare Gaussiană,
- factorizare (LU, QR),
- metode directe pentru sisteme cu matrice de formă specială (exemplu: substituţie regresivă pentru sisteme triunghiulare).

---

1. Metode paralele de eliminare pentru matrice dense

A. *Tehnica Gauss* (secvenţială), în doi paşi:

1. reduce sistemul la forma tridiagonală ("reducere înainte")
2. se face o substituţie regresivă ("substituţie înapoi").

Exemplu: se consideră sistemul

$$\begin{cases} 2x_1 + 2x_2 + 4x_3 + 2x_4 = 76 \\ x_1 + 3x_2 + 2x_3 + x_4 = 52 \\ 3x_1 + 2x_2 + 2x_3 + x_4 = 57 \\ x_1 + x_2 + 3x_3 + 3x_4 = 64 \end{cases}$$

Pas 1: împarte prima ec. cu coef. x_1 şi scade din celelalte ec. * cu un coef. a.î. $\rightarrow 0$ pe coloana x_1 ...

Etapa 1:	Etapa 2:
$\begin{cases} x_1 + x_2 + 2x_3 + x_4 = 38 \\ 2x_2 + 0x_3 + 0x_4 = 14 \\ -x_2 - 4x_3 - 2x_4 = -57 \\ 0x_2 + x_3 + 2x_4 = 26 \end{cases}$	$\begin{cases} x_1 + x_2 + 2x_3 + x_4 = 38 \\ x_2 + 0x_3 + 0x_4 = 7 \\ -4x_3 - 2x_4 = -50 \\ x_3 + 2x_4 = 26 \end{cases}$
Etapa 3:	**Etapa 4:**
$\begin{cases} x_1 + x_2 + 2x_3 + x_4 = 38 \\ x_2 + 0x_3 + 0x_4 = 7 \\ -4x_3 - 2x_4 = -50 \\ x_3 + 2x_4 = 26 \end{cases}$	$\begin{cases} x_1 + x_2 + 2x_3 + x_4 = 38 \\ x_2 + 0x_3 + 0x_4 = 7 \\ x_3 + 1/2x_4 = 25/2 \\ 3/2x_4 = 27/2 \end{cases}$

Pas 2: substituţie regresivă

Etapa 1:	Etapa 2:
$x_4 = 9$	$x_3 = 25/2 - 1/2x_4 = 8$
Etapa 3:	**Etapa 4:**
$x_2 = 7 - 0x_3 - 0x_4 = 7$	$x_1 = 38 - x_2 - 2x_3 - x_4 = 6$

Pivotare= alegerea liniei ,,pivot" pe baza determinării max. în modul a elementelor de pe o coloană

Idee paralelism: liniile matricei sunt împărţite între procesoare şi procesate concurent + linia pivot este sticată într-o memorie globală.

Problemă: distribuţie inegală dacă 1 linie $\rightarrow$ 1 procesor

Soluţii:

1. procesor i conţine linia $i + pq$ unde p=nr. proc, q a.i. distribuţie ,,egală" pe proc.
   Ex: $m = n = 10$, $p = 3 \rightarrow$ proc 1. deţine linile 1, 4, 7, 10, proc. 2: 2, 5, 8, proc. 3: 3, 6, 9
2. scheme de eliminare:
   Ex: două elemente marcate cu i dacă pot fi eliminate simultan la pasul i:

$$\begin{array}{cccccccc}
* & & & & & & & \\
1 & * & & & & & & \\
2 & 3 & * & & & & & \\
3 & 4 & 5 & * & & & & \\
4 & 5 & 6 & 7 & * & & & \\
5 & 6 & 7 & 8 & 9 & * & & \\
6 & 7 & 8 & 9 & 10 & 11* & & \\
7 & 8 & 9 & 10 & 11 & 12 & 13* &
\end{array}$$

Operaţii pt. $m = n$:

1. algoritm secvenţial: $\frac{1}{3}n^3 + \mathcal{O}(n^2)$.
2. algoritm paralel: $\mathcal{O}(n)$ multiplicări (în paralel) pentru triangularizare + $\mathcal{O}(n)$ multiplicări (in paralel) pentru substituţie

. . . . . .

B. <u>*Tehnica Gauss-Jordan*</u> (secvenţială), într-un pas:

1. reduce sistemul la forma diagonală

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{pmatrix} \xrightarrow{a_{11}\neq 0} \begin{pmatrix} 1 & a_{12}^{(1)} & \dots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} & b_n^{(1)} \end{pmatrix} \xrightarrow{a_{22}^{(1)}\neq 0}$$

$$\begin{pmatrix} 1 & 0 & \dots & a_{1n}^{(2)} & b_1^{(2)} \\ 0 & 1 & \dots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & a_{nn}^{(2)} & b_n^{(2)} \end{pmatrix} \xrightarrow{a_{33}^{(2)}\neq 0} \dots \xrightarrow{a_{nn}^{(n-1)}\neq 0} \begin{pmatrix} 1 & 0 & \dots & 1 & b_1^{(n)} \\ 0 & 1 & \dots & 1 & b_2^{(n)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & b_n^{(n)} \end{pmatrix} \qquad \begin{array}{c} x_1 = b_1^{(n)} \\ x_2 = b_2^{(n)} \\ \vdots \\ x_n = b_n^{(n)} \end{array}$$

Exemplu: pt. sistem de mai sus

Etapa 1:	Etapa 2:
$\begin{cases} x_1 & +x_2 & +2x_3 & +x_4 & = 38 \\ & +2x_2 & +0x_3 & +0x_4 & = 14 \\ & -x_2 & -4x_3 & -2x_4 & = -57 \\ & +0x_2 & +x_3 & +2x_4 & = 26 \end{cases}$	$\begin{cases} x_1 & & +2x_3 & +x_4 & = 31 \\ & +x_2 & +0x_3 & +0x_4 & = 7 \\ & & -4x_3 & -2x_4 & = -50 \\ & & +x_3 & +2x_4 & = 26 \end{cases}$
Etapa 3:	**Etapa 4:**
$\begin{cases} x_1 & & & +0x_4 & = 6 \\ & +x_2 & & +0x_4 & = 7 \\ & & -4x_3 & -2x_4 & = -50 \\ & & & +2x_4 & = 26 \end{cases}$	$\begin{cases} x_1 & & & & = 38 \\ & +x_2 & & & = 7 \\ & & x_3 & & = 8 \\ & & & 3/2x_4 & = 27/2 \end{cases}$

Idee paralelism: un pas paralel constă din anihilarea simultană a coeficienţilor de pe fiecare linie alta decât linia pivot (considerat $\mathcal{O}(1)$ atunci întreg alg. necesită $\mathcal{O}(n)$ paşi)

Operaţii pt. $m = n$:

1. algoritm secvenţial: $\frac{1}{2}n^3 + \mathcal{O}(n^2) >$ decât alg. secv. Gauss
2. algoritm paralel: $\mathcal{O}(n) <$ decât alg. paralel Gauss

Concluzie:

1. agoritmii secvenţiali ,,optimali" nu sunt întotdeauna ,,optimali" şi în implementarea paralelă
2. întreaga teorie asupra optimalităţii algoritmilor dintr-o clasă dată trebuie reconsiderată în cazul implementării paralele
3. algoritmii optimali pt. implementări paralele pot fi algoritmi noi sau vechi, ineficienţi în cazul secvenţial.

Obs: pt. stabilitate numerică necesară o formă de pivotare.

---

2. Metode paralele de descompunere pentru matrice dense

A. *Descompunerea LU* (secvenţială), în trei paşi:

1. factorizare $A = LU$, L triunghiular inferioară, U superior triunghiulară şi $l_{ii} = 1$, $|l_{ij}| < 1$
2. rezolvă $Ly = b$ (,,pas înainte")
3. rezolvă $Ux = y$ (,,pas înapoi").

Variantă cu permutare: $PA = LU$, $Ly = Pb$ şi $Ux = y$.

Ex: în cazul sistemului de mai sus

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1 & 0 & 0 \\ 3/2 & -1/2 & 1 & 0 \\ 1/2 & 0 & -1/4 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 2 & 4 & 2 & 76 \\ 0 & 2 & 0 & 0 & 14 \\ 0 & 0 & -4 & -2 & -50 \\ 0 & 0 & 0 & 3/2 & 27/2 \end{pmatrix}$$

Alg. secvenţial:

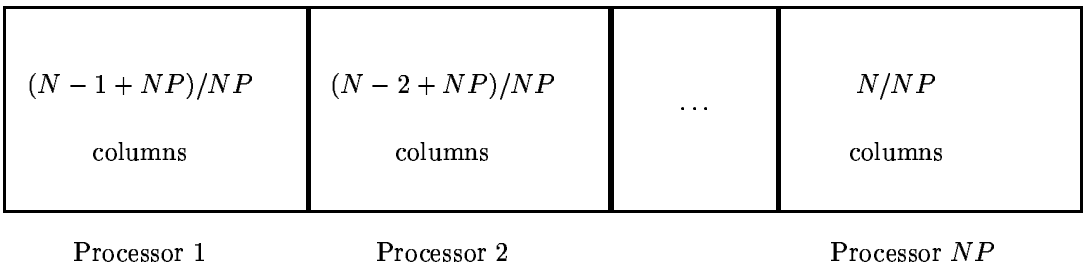
for $j = \overline{1,n}$
  for $k = \overline{1,j-1}$
    următorul element în coloana lui U : $a_{k_j} \to u_{kj} = a_{kj}/l_{kk}$
    actualizează coloana j a lui A,
    coloana provizorie j a lui L este calculată în $\underline{a}_{\cdot j}$:
      $a_{\cdot j} \to a_{\cdot j} - \underline{l}_{\cdot k} u_{kj}$,
[Determină $p \in \{j, j+1, \ldots, n\}$: $|a_{pj}| = \max_{j \le i \le n} |a_{ij}|$ (selectare pivot)
interschimbare linii: $a_{j\cdot} \leftrightarrow a_{p\cdot}$]
actualizează coloana lui L: $a_{\cdot j} \to l_{\cdot j} = \underline{a}_{\cdot j}/(u_{jj})$

Obs: similar cu eliminarea gaussiană.

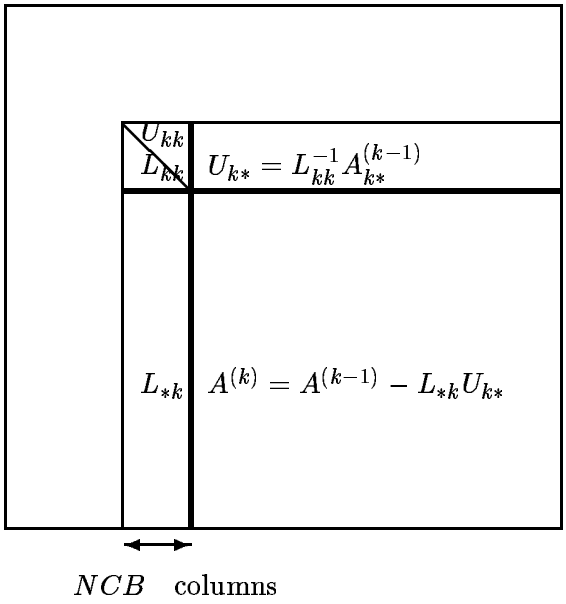
Idee parallelism: în factorizarea LU coloanele matricei A sunt distribuite pe proc.
(a) distribuţie 1 coloană la 1 procesor (ineficient ca şi în cazul eliminării gaussiene)
(b) distribuţie pe blocuri
(c) distribuţie în manieră ciclică
+ când un procesor face o descompunere celelalte aplică transformarea la col. lor.

Distribuţie pe blocuri:

Fie $n = N$ coloane, $p = NP$ procesoare şi NCB=nr. coloane în bloc.
A este distribuit astfel:



Etape:
  fie $A^{(0)} = A$, $NB = \lceil N/NCB \rceil$;
  pentru $k = 1, \ldots, NB$ execută
  (a) calculează L_{kk}, U_{kk}, şi L_{kk} prin factorizare LU locală utilizând elementele lui $A^{(k-1)}$;
  (b) calculează L_{kk}^{-1};
  (c) calculează $U_{k*} = L_{kk}^{-1} A_{kk}^{(k-1)}$;
  (d) calculează $A^{(k)} = A^{(k-1)} - L_{*k} U_{k*}$.



Distribuţie în manieră ciclică:

... de tip $\{\mu, \mu + p, \mu + 2p, \ldots, \mu + (l-1)p \leq n\}$.

Fiecare procesor p efectuează în paşii $k = 1, \ldots, NB$ (NB=nr. blocuri):

Eu stochez blocul la pasul k?
dacă da
(a) calculează L_{kk}, U_{kk}, L_{*k} and L_{kk}^{-1}.
(b) transmite L_{*k} şi L_{kk}^{-1} la celelalte procesoare
(c) calculează porţiunea din U_{k*} şi $A^{(k)}$
altfel
(a) primeşte L_{*k} şi L_{kk}^{-1} trimis de procesor care efectuează ramura DA;
(b) calculează porţiunea din U_{k*} şi $A^{(k)}$.

Strategie divide-and-conquer

Alg. 1: pp. la pasul i:

1. se cunoaşte factoriz. minorului principal de dim. $(i-1)\omega$, $A_{i-1} = L_{i-1}U_{i-1}$.
2. se calculează următoarele ω linii ale lui L şi ω coloane ale lui U şi se calculează

$$A_i = \begin{pmatrix} A_{i-1} & C \\ B^T & H \end{pmatrix} = \begin{pmatrix} L_{i-1} & 0 \\ M^T & L_2 \end{pmatrix} \begin{pmatrix} U_{i-1} & G \\ 0 & U_2 \end{pmatrix}.$$

Etape:

(a) rezolvă în G: $C \leftarrow L_{i-1}G = C$.
(b) rezolvă în M: $B \leftarrow U_{i-1}^T M = B$.
(c) $H \leftarrow H - M^T G$.
(d) factorizează $H \leftarrow L_2 U_2 = H$.

Alg. 2: pp. la pasul i:

1. se cunosc primele $\varepsilon = (i-1)\omega$ coloane ale lui L şi ε linii ale lui U
2. se aplică transformarea matricei $A^i \in \mathbb{R}^{(n-\varepsilon)\times(n-\varepsilon)}$ din colţ dreapta jos A
3. se produc următoarele ω coloane şi linii ale lui L şi U şi se calculează A^{i+1} (generalizare a eliminării gausiene) din:

$$A^i = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & A^{i+1} \end{pmatrix}$$

Etape:

(a) factorizează $A_{11} \leftarrow L_{11}U_{11} = A_{11}$.
(b) rezolvă în L_{21} : $A_{21} \leftarrow U_{11}^T L_{21}^T = A_{21}^T$.
(c) rezolvă în U_{12} : $A_{12} \leftarrow L_{11}U_{12} = A_{12}$.
(d) $A^{i+1} \leftarrow A_{22} - L_{21}U_{12}$.

. . . . . .

B. Descompunerea LQ (secvenţială), în doi paşi:

1. factorizare $A = QR$, Q ortogonală ($Q^TQ = I$), R superior triunghiulară
2. rezolvă $Rx = Q^Tb$

Idee: se fac zero (anihilează) elementele lui A de sub diagonală prin utilizarea unor „rotaţii", adică matrice similare următoarei:

$$P = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

care nu distrug zerourile anterioare.

Alg. secvenţial bazat pe rotaţii Givens:

For $r = \overline{1,n}$
  calculează $Q_r = \prod_{i=0}^{m-r-1} Q_{m-i,r}$,
  $A_r = \left(\prod_{i=0}^{r-1} Q_{r-i}\right) A$

unde $Q_{p,r}$, $p = \overline{p+1,m}$ = rotaţia Givens între liniile p şi $p-1$ pt. a face 0 elementul (p,r) a lui $\left(\prod_{i=1}^{p-r-1} Q_{p-i,r}\right) A_r$.

Atunci $R = A_n$, $Q^T = Q_n \cdots Q_1$, şi $A = QR$.

Rotaţii Givens:

$$P_{k,l}^{(i,j)} = \begin{cases} \delta_{kl}, & \text{if } \{i,j\} \cap \{k,l\} = \emptyset \\ \cos(\theta), & \text{if } k = l = i \text{ or } k = l = j \\ \sin(\theta), & \text{if } k = i < l = j \text{ or } k = j < l = i \\ -\sin(\theta), & \text{if } k = j > l = i \text{ or } k = i > l = j \end{cases}, \quad i \neq j$$

?θ : Se cere modificarea numai a liniilor i şi $i-1$ la eliminarea unui (i,j). adică se schimbă numai

$$P \begin{pmatrix} a_{i-1\,1} & a_{i-1\,2} & \cdots & a_{i-1\,j} & \cdots a_{i-1\,n} \\ a_{i\,1} & a_{i\,2} & \cdots & a_{i\,j} & \cdots a_{i\,n} \end{pmatrix}.$$

Atunci $-\sin\theta a_{i-1\,j} + \cos\theta a_{ij} = 0$ pt. ca val a_{ij} să devină $\rightarrow$

$$\sin\theta = \frac{a_{ij}}{\sqrt{a_{ij}^2 + a_{i-1j}^2}}, \qquad \cos\theta = \frac{a_{i1}j}{\sqrt{a_{ij}^2 + a_{i-1j}^2}}.$$

Idee paralelism: mai multe rotaţii sunt efectuate simultan.
Ex: dacă $n = m = 8$:

$$\begin{array}{cccccccc}
x & & & & & & & \\
3 & x & & & & & & \\
2 & 5 & x & & & & & \\
2 & 4 & 7 & x & & & & \\
1 & 3 & 6 & 8 & x & & & \\
1 & 3 & 5 & 7 & 9 & x & & \\
1 & 2 & 4 & 6 & 8 & 10 & x & \\
1 & 2 & 3 & 5 & 7 & 9 & 11 & x
\end{array}$$

(un întreg r apare unde zerourile sunt create la pasul r)
Ordinea de anihilare: elementele mulţimii

$$S = \{(p,q) |\ 1 \le q < p \le m,\ q \le n\},$$

pt. $m \ge n$, se numerotează astfel $(p_k, q_k),\ k = \overline{1, mn - \frac{1}{2}n^2 - \frac{1}{2}n}$, în ordinea

$$(m,1),\ (m-1,1), \ldots,\ (2,1);\ (m,2),\ (m-1,2), \ldots,\ (3,2), \ldots,$$

$$(m,n),\ (m-1,n), \ldots,\ (n+1,n).$$

Se partiţionează S în submulţimi $S_r,\ r = \overline{1, m+n-2}$ în ordinea

$$\{(m,1)\},\ \{(m-1,1)\},\ \{(m,2),(m-2,1)\},\ \{(m-1,2),(m-3,1)\},$$

$$\{(m,3),(m-2,2),(m-4,1)\}, \ldots$$

a.î. $S_r = \{(p,q) |\ 1 \le q < p \le m,\ q \le n,\ m + 2q = p + r + 1\}$.
Rotaţiile corespunzătoare unui S_r sunt disjuncte! şi pot fi efectuate simultan.

Alte: reflecţii Householder pt. anihilare componente vector.

. . . . . .

C. <u>*Descompunerea WZ*</u> (algoritm paralel!) în trei paşi:

1. factorizare $A = WZ$, unde

$$W = \begin{pmatrix}
1 & & & 0 & & & 0 \\
w_{21} & 1 & & & & 0 & w_{2n} \\
\vdots & & \ddots & & & & \vdots \\
\vdots & & & 1 & & & \vdots \\
\vdots & & & & \ddots & & \vdots \\
w_{n-11} & 0 & & & & 1 & w_{n-1n} \\
0 & & & 0 & & & 1
\end{pmatrix},$$

$$Z = \begin{pmatrix} z_{11} & & & & & & z_{1n} \\ 0 & z_{22} & & & & z_{2n-1} & 0 \\ \vdots & & \ddots & & & & \vdots \\ \vdots & & & z_{ii} & & & \vdots \\ \vdots & & & & \ddots & & \vdots \\ 0 & & & & & \ddots & 0 \\ z_{n1} & & & & & & z_{nn} \end{pmatrix}$$

2. rezolvă $Wp = b$ („pas înainte")
3. rezolvă $Zx = p$ („pas înapoi").

Calcul factorizare:

Pas 1

(a) $z_{1j} = a_{1j},\ z_{nj} = a_{nj},\ j = \overline{1,n}$

(b) prima şi ultima coloană a lui W se onţin din cele $n-2$ sisteme 2×2 (în paralel)

$$\begin{cases} z_{11}w_{i1} + z_{n1}w_{in} = a_{i1} \\ z_{1n}w_{i1} + z_{nn}w_{in} = a_{in} \end{cases}$$

(c) se actualizează $A \leftarrow A - WE_1Z_1^T - W_nZ_n^T$ unde W_1 and W_n coloane W, Z_1^T şi Z_n^T linii ale lui Z.

Pas k $(k = \overline{1, \left\lfloor \frac{1}{2}(n+1) \right\rfloor})$:

(a)

$$z_{hj} = a_{hj} - \left(\sum_{l=1}^{k-1} + \sum_{l=n-k+2}^{n} \right) w_{hl}z_{lj},\ h = k, n-k+1,\ j = \overline{k, n-k+1}$$

(b) rezolvă pt. $i = \overline{k+1, n-k}$ ($n-2k$ sisteme în paralel)

$$z_{kh}w_{ik} + z_{n-k+1\,h}w_{i\,n-k+1} = a_{ih} - \left(\sum_{l=1}^{k-1} + \sum_{l=n-k+2}^{n} \right) w_{il}z_{lh}.$$

Calcul $Wp = b$:

$$\begin{pmatrix} 1 & & & 0 & & & 0 \\ w_{21} & 1 & & & & 0 & w_{2n} \\ \vdots & & \ddots & & & & \vdots \\ \vdots & & & 1 & & & \vdots \\ \vdots & & & & \ddots & & \vdots \\ w_{n-11} & 0 & & & & 1 & w_{n-1n} \\ 0 & & & 0 & & & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \\ \vdots \\ \\ p_{n-1} \\ p_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \\ \vdots \\ \\ b_{n-1} \\ b_n \end{pmatrix}$$

Pas 1: calcul p_1 şi p_n first

Pas i: calcul

$$p_i = b_i^{(i)}, \quad p_{n-i+1} = b_{n-i+2}^{(i)}$$

unde

$$b_j^{(1)} = b_j \text{ and } b_j^{(i+1)} = b_j^{(i)} - w_{ij}p_i - w_{j\,n-i+1}p_{n-i+1}, \quad j = \overline{i+1, n-i}$$

Calcul $Zx = p$ similar.

---

3. Metode paralele pentru sisteme tridiagonale

$$\begin{pmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & c_n & a_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}.$$

A. *Descompunere LU*

Linia i a lui L: $(0, \ldots, 0, l_i, 1, 0, \ldots, 0)$; linia i a lui U: $(0, \ldots, 0, u_i, b_i, 0, \ldots, 0)$ unde $u_i = a_i - c_i b_{i-1}/u_{i-1}$

Algoritm vectorial:

$$t_i = c_i b_{i-1} \text{ (operaţie vectorială – în paralel)}$$
$$d_1 = 1/a_1$$
$$d_i = 1/(a_i - t_i d_{i-1}),$$
$$l_i = c_i d_{i-1} \text{ (operaţie vectorială – în paralel)}$$

Dublare recursivă:

$$q_0 = 1,\ q_1 = b_1,\ q_i = b_i q_{i-1} - c_i a_{i-1} q_{i-2},\ i = \overline{2, n}.$$
$$u_i = q_i/q_{i-1}, \quad i = \overline{1, n}.$$
$$Q_i = \begin{pmatrix} q_i \\ q_{i-1} \end{pmatrix} = \begin{pmatrix} b_i & -c_i a_{i-1} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} q_{i-1} \\ q_{i-2} \end{pmatrix} = G_i Q_{i-1} = \left(\prod_{j=2}^{i} G_j \right) Q_1 = S_i Q_1$$

Calcul paralel S_i prin dublare recursivă.

. . . . . .

B. *Reducere ciclică* (reducere par-impar)

Idee: se elimină variabilele impare din ecuaţiile numerotate cu nr. par. Pp. $R(2i)$ linia $2i$ a matricei A. Atunci:

$$R(2i) - (c_{2i}/a_{2i-1}) * R(2i-1) - (b_{2i}/a_{2i+1}) * R(2i+1)$$

$$c_{i-1}x_{i-2} + a_{i-1}x_{i-1} + b_{i-1}x_i = d_{i-1}, \quad (1)$$
$$c_i x_{i-1} + a_i x_i + b_i x_{i+1} = d_i, \quad (2)$$
$$c_{i+1}x_i + a_{i+1}x_{i+1} + b_{i+1}x_{i+2} = d_{i+1}, \quad (3)$$

Scădere (2) din (1) şi (3) multiplicate a.î. se obţin 0 la coef. x_{i-1} şi x_{i+1}:

$$c_i^{(1)}x_{i-2} + a_i^{(1)}x_i + b_i^{(1)}x_{i+2} = d_i^{(1)}.$$

Acelaşi procedeu:

$$c_i^{(2)}x_{i-4} + a_i^{(2)}x_i + b_i^{(2)}x_{i+4} = d_i^{(2)},$$

Se consideră $x_i = b_i = c_i = y_i = 0$ şi $a_i = 1$ când $i < 0,\ i > n$.
După $k = \lceil \log_2 n \rceil$ paşi $a_i^{(k)}x_i = d_i^{(k)}$.

. . . . . .

C. *Divide-and-conquer*

Algoritm 1. $m = n = pq$:
1. subdivizare în p subprobleme
2. rezolvarea în paralel a celor p subprobleme de dimensiune q

Ex: $n = 9\ p = q = 3$

$$\begin{pmatrix} b_1 & c_1 & & & & & & & \\ a_2 & b_2 & c_2 & & & & & & \\ & a_3 & b_3 & c_3 & & & & & \\ & & a_4 & b_4 & c_4 & & & & \\ & & & a_5 & b_5 & c_5 & & & \\ & & & & a_6 & b_6 & c_6 & & \\ & & & & & a_7 & b_7 & c_7 & \\ & & & & & & a_8 & b_8 & c_8 \\ & & & & & & & a_9 & b_9 \end{pmatrix} \rightarrow \begin{pmatrix} b_1 & c_1 & & & & & & & \\ & b_2' & c_2' & & & & & & \\ & & b_3' & c_3' & & & & & \\ & & a_4 & b_4 & c_4 & & & & \\ & & a_5' & & b_5' & c_5' & & & \\ & & a_6' & & & b_6' & c_6' & & \\ & & & & & a_7 & b_7 & c_7 & \\ & & & & & a_8' & & b_8' & c_8' \\ & & & & & a_9' & & & b_9' \end{pmatrix} \rightarrow$$

$$\rightarrow \begin{pmatrix} b"_1 & & c"_1 & & & & & & \\ & b_2' & c_2' & & & & & & \\ & & b"_3 & & & c"_3 & & & \\ & & a"_4 & b"_4 & & c"_4 & & & \\ & & a_5' & & b_5' & c_5' & & & \\ & & a"_6 & & & b"_6 & & & c"_6 \\ & & & & & a"_7 & b"_7 & & c"_7 \\ & & & & & a_8' & & b_8' & c_8' \\ & & & & & a_9' & & & b_9' \end{pmatrix}$$

Transformările se aplică şi asupra vectorului termenilor liberi.
Sisteme: în (x_1, x_4, x_7), (x_2, x_5, x_8), (x_3, x_6, x_9), de exemplu:

$$\begin{pmatrix} b"_3 & c"_3 & \\ a"_6 & b"_6 & c"_6 \\ a'_9 & b'_9 & \end{pmatrix} \begin{pmatrix} x_3 \\ x_6 \\ x_9 \end{pmatrix} = \begin{pmatrix} d"_3 \\ d"_6 \\ d"_9 \end{pmatrix}.$$

---

4. Metode paralele pentru sisteme triunghiulare

Sistem liniar recurent $R < n, m >$ de ordin m pt. n ecuaţii:

$$R < n, m >: \qquad x_k = \begin{cases} 0, & \text{if } k \le 0 \\ b_k + \sum_{j=k-m}^{k-1} a_{kj} x_j, & \text{if } 1 \le k \le m \end{cases} \quad m \le n-1$$

adică $x = Ax + b$ cu A triunghiular strict inferior.
Un sistem triunghiular $\overline{A}x = \overline{b}$ poate fi scris $(I - A)x = b$, unde $a_{ij} = \delta_{ij} - \overline{a}_{ij}/a_{ii}$ and $b_i = \overline{b}_i/\overline{a}_{ii},\ i, j = \overline{1, n}$. Atunci $x = Ax + b$.

A. *Algoritmul de schimbare a coloanei*

Pas 1 evaluează în paralel $b_i^{(1)} = b_i + a_{i1}x_1$, pt. $i = \overline{2, n}$, unde $x_1 = b_1$ este cunoscut.
Pas 2 evaluează în paralel $b_i^{(2)} = b_i^{(1)} + a_{i2}x_2$, for $i = \overline{3, n}$, unde x_1 şi x_2 sunt cunoscute
$\vdots$
Pas k evaluează în paralel $b_i^{(k)} = b_i^{(k-1)} + a_{ik}x_k$, for $i = \overline{k+1, n}$, unde $x_1, \ldots x_k$ sunt cunoscute.
$\vdots$

Un sistem $n \times n$ necesită $n - 1$ paşi şi $n - 1$ proc. la primul pas şi mai puţine după.

Operaţii: $\mathcal{O}(n)$ paşi pt. rezolvare sistem triunghiular $n \times n$ utilizând $\mathcal{O}(n)$ proc.

. . . . . .

B. *Algoritmul produsului recurent* (dublare recursivă)

$x = (I - A)^{-1}b$, unde A este triunghiular strict inferioară.
Obs:

$$(I - A)^{-1} = \prod_{i=1}^{n-1} M_{n-i}$$

$$M_i = \begin{pmatrix} 1 & & & 0 & \\ & \ddots & & & \\ & & 1 & & \\ & & a_{i+1\,i} & 1 & \\ 0 & & \vdots & & \ddots \\ & & a_{ni} & & 0 \quad 1 \end{pmatrix}.$$

Ex:

$$\begin{cases} x_1 = b_1 \\ x_2 = b_2 + a_{21}x_1 \\ x_3 = b_3 + a_{31}x_1 + a_{32}x_2 \\ x_4 = b_4 + a_{42}x_2 + a_{43}x_3 \end{cases}$$

Atunci

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -a_{21} & 1 & 0 & 0 \\ -a_{31} & -a_{32} & 1 & 0 \\ 0 & -a_{42} & -a_{43} & 1 \end{pmatrix}^{-1} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & a_{43} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & a_{32} & 1 & 0 \\ 0 & a_{42} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ a_{21} & 1 & 0 & 0 \\ a_{31} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}.$$

(5 paşi în paralel)

$\Pi_{i=1}^{n-1} M_{n-i}$ poate fi calculat în $\mathcal{O}((\log_2 n)^2)$ paşi prin tehnica dublării recursive.

Ex: $n = 8$: $(M_8(M_7(M_6(M_5(M_4(M_3(M_2 f)))))))$ prin $(((N_7N_6)(N_5N_4))((N_3N_2)(N_1 f)))$.

Operaţii: $\mathcal{O}((\log_2 n)^2)$ paşi utilizând $\mathcal{O}(n^3)$ procesoare.

. . . . . .

C. *Tehnica partiţionării* (divide-and-conquer)
Obs: dacă

$$A = \begin{pmatrix} A_1 & 0 \\ A_2 & A_3 \end{pmatrix},$$

cu A_1 şi A_2 inferior triunghiulare , A_2 densă, atunci

$$A^{-1} = \begin{pmatrix} A_1^{-1} & 0 \\ -A_3^{-1}A_2A_1^{-1} & A_3^{-1} \end{pmatrix}.$$

Etape:
(a) inversare simultană A_1 şi A_3 şi rezolvă $A_3Y = A_2$ în Y
(b) multiplică Y şi A_1^{-1}.
Operaţii: $\mathcal{O}((\log_2 n)^2)$ paşi cu $\mathcal{O}(n^2)$ procesoare.

---

5. Metode paralele pentru sisteme cu matrice bandă sau rare

- eliminare Gaussiană sau Gauss-Jordan
- partiţionare pe blocuri
- reducere Givens
- descompunere LU

METODE ITERATIVE PARALELE DE REZOLVARE A SISTEMELOR DE ECUAŢII LINIARE

Problema: rezolvarea sistemului $Ax = b$, unde A este o matrice $m \times n$, b este un vector m-dimensional, x este vectorul n-dimensional necunoscut.

---

0. Metode secvenţiale acceptate:

1. *Metode de relaxare*
   Idee: $A = G - H,\ Ax^* = b \quad \Leftrightarrow \quad Gx^* = b + Hx^*$
   Metoda de relexare=met. iteraţiilor simple: $Gx^{(k+1)} = b + Hx^{(k)}$ dacă $\exists G^{-1}$
   Eroare: $v^{(k)} = x^* - x^{(k)}$ satisface $v^{(k+1)} = G^{-1}Hv^{(k)} = (G^{-1}H)^{k+1}v^{(0)}$.
   Matricea iteraţiei:$M = G^{-1}H$
   Factor de convergenţă a metodei – raza spectrală a lui M: $\rho(G^{-1}H)$ (val. proprie maximă în modul)

$$\text{if } \rho(G^{-1}H) < 1 \text{ then } \forall v^{(0)} \in \mathbb{R}^m : \quad \lim_{k\to\infty} (G^{-1}H)^k v^{(0)} = 0$$

   Caz particular: $A = D - L - U$, $D = diag(D)$, L strict inferior triunghiulară, U strict superior triunghiulară:

$$d_{ij} = \begin{cases} a_{ij} & i = j \\ 0, & \text{altfel} \end{cases} \qquad l_{ij} = \begin{cases} -a_{ij} & i < j \\ 0, & \text{altfel} \end{cases} \qquad u_{ij} = \begin{cases} -a_{ij} & i > j \\ 0 & \text{altfel} \end{cases}$$

$$Ax = b \qquad \Leftrightarrow \qquad Dx = (L + U)x + b \text{ sau } (D - L)x = Ux + b$$

   (a) Metoda Jacobi (J): $A = G_J - H_J,\ G_J = D,\ H_J = -(L + U)$

$$Dx_J^{(k+1)} = (L + U)x_J^{(k)} + b \ \Leftrightarrow$$

$$x_J^{(k+1)} = D^{-1}(D - A)x_J^{(k)} + D^{-1}b = x_J^{(k)} - D^{-1}(Ax_J^{(k)} - b)$$

   Condiţia de convergenţă: raza spectrală $\rho(D^{-1}(L + U) < 1$ adică A este strict diagonal dominant

   (b) Metoda Gauss-Seidel (GS) (mai rapidă decât J): $A = G_{GS} - H_{GS},\ G_{GS} = D + L,\ H_{GS} = -U$.

$$x_{GS}^{(k+1)} = (D - L)^{-1}Ux_{GS}^{(k)} + (D - L)^{-1}b = x_{GS}^{(k)} - (D - L)^{-1}(Ax_{GS}^{(k)} - b)$$

   Condiţia de convergenţă îndeplinită dacă A=matrice simetrică pozitiv-definită

   (c) Metoda suprarelaxării succesive (SOR): îmbunătăţirea cond. converg. GS prin $A = G_{SOR} - H_{SOR},\ G_{SOR} = (D + \omega L)/\omega,\ H_{SOR} = ((1 - \omega)D - \omega U)/\omega$:

$$x_{SOR}^{(k+1)} = (1 - \omega)x_{SOR}^{(k)} + \omega x_{GS}^{(k+1)}, \quad \omega \in [0, 2] \text{ factor de supra-relaxare}$$

$$Dd_1^{(k)} = D(x^{(k+1)} - x^{(k)}) = Lx^{(k+1)} + Ux^{(k)} - Dx^{(k)} + b$$
$$d^{(k)} = \omega d_1^{(k)}$$
$$x^{(k+1)} - x^{(k)} = \omega D^{-1}(Lx^{(k+1)} + Ux^{(k)} - Dx^{(k)} + b.$$

$$x^{(k+1)} = \omega D^{-1}Lx^{(k+1)} + [(1-\omega)I + \omega D^{-1}U]x^{(k)} + \omega D^{-1}b$$
$$(D + \omega L)x^{(k+1)} = \omega b + ((1-\omega)D - \omega U)x^{(k)}.$$

Condiţia de convergenţă: $0 < \omega < 2$

Optim ω minimizează raza spectrală a matricei de conv. SOR:
$\omega_b = 2/(1 + \sqrt{1 - \rho^2(B)}),\ B = I - D^{-1}A$

(d) Metoda JOR: îmbunătăţirea cond. converg. met. J:

$$x^{(k+1)} - x^{(k)} = \omega D^{-1}[(L+U)x^{(k)} + b - Dx^{(k)}]$$

$$x^{(k+1)} = [\omega D^{-1}(L+U) + (1-\omega)]x^{(k)} + \omega D^{-1}b,\ k \geq 0$$

cu ω factor de sub-relaxare

Optim ω: fie $B = I - D^{-1}A$ şi $m(B)$, $M(B)$ val. proprie minimă/maximă a lui B ($\rho(B) = \max(|m(B)|, |M(B)|)$)

$$m(B_\omega) = \omega m(B) + 1 - \omega,\ M(B_\omega) = \omega M(B) + 1 - \omega$$

ω=optim când $m(B_\omega) = -M(B_\omega)$ adică $\omega = 2/(2 - m(B) - M(B))$.

(e) Metoda SSOR (symmetric succesive over relaxation method) alternează paşi SOR „înainte" şi „înapoi": $A = G_{SSOR} - H_{SSOR}$, $G_{SSOR} = \frac{1}{\omega(2-\omega)}(D + \omega L)D^{-1}(D + \omega U)$, $H_{SSOR} = \frac{1}{\omega(2-\omega)}((1-\omega)D - \omega L)D^{-1}((1-\omega)D - \omega U)$.

$$(D + \omega L)x^{(k+1/2)} = \omega b + ((1-\omega)D - \omega U)x^{(k)}$$
$$(D + \omega U)x^{(k+1)} = \omega b + ((1-\omega)D - \omega L)x^{(k+1/2)}$$

Condiţia de convergenţă îndeplinită dacă A=simetrică şi pozitiv definită şi $0 < \omega < 2$.

2. *Metode de tip gradient* – ex: metoda gradientului conjugat

Idee: fie $\mathcal{F}(x) = \frac{1}{2}x^TAx - x^Tb,\ A \in \mathbb{R}^{m\times m}$ simetrică şi pozitiv-definită

$$?x^* :\ \mathcal{F}(x^*) = \min_{x\in\mathbb{R}^n} \mathcal{F}(x) \quad \Leftrightarrow Ax = b$$

$$\forall k \geq 0, \quad x_{k+1} = x_k + \frac{p_k^Tp_k}{p_k^TAp_k}p_k.$$

? p_k. Ex:

(a) metoda pasului descendent: $p_k = r_k = b - Ax_k$

(b) vectori A-conjugaţi: $p_k^TAp_j = 0$, if $k \neq j$

$$x_{k+1} = x_k + \varepsilon_k p_k,\ p_{k+1} = \lambda_k(Ap_k - \mu_k p_k - \nu p_{k-1})$$

$$\varepsilon_k = \frac{p_k^T r_0}{p_k^T A p_k}, \quad \mu_k = (Ap_k)^T Ap_k/(p_k^T Ap_k), \quad \nu = (Ap_k)^T Ap_{k-1}/(p_{k-1}^T Ap_{k-1}),$$

$\lambda_k =$ factor de scalare

Convergenţă: funcţie de nr. de condiţie a lui A (mic $\rightarrow$ converg.rapidă)

Obs:

- îmbunătăţire convergenţă prin rezolvare problemă nouă în care $A \rightarrow M^{-1}A$ – metode cu precondiţionare (ex: $M = LL^T$ unde L din factrizare $A = LU$)
- terminare în număr finit de paşi

---

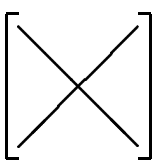
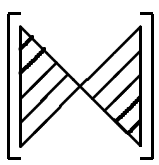
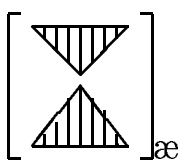
1. Metode paralele:

1. *paralelizare* J, JOR, GS, SOR, SSOR (obs: prin natura lor secvenţiale!):
   (a) *distribuirea* efort de calcul *per componente* ale vectorului iter. Jacobi
   (b) *metoda alb-negru* asociată cu GS, SOR, SSOR:
   - se colorează componentele vectorului x alternativ în alb-negru;
   - componentele negre sunt primele actaulizate, utilizând componentele albe anterioare
   - componentele albe sunt actualizate folosind ultimele valori negre
   - comp. negre sunt actualizate la un nivel de iteraţie, cele albe la următorul nivel de iterare
   - după o iteraţie, negrele sunt la nivel 1, albele la nivel 2
   - paralelizare prin asignare grupuri de albe, respectiv negre, la proc. diferite
   - iterare până când are loc condiţia de acurateţe: $|x_j^{(k+1)} - x_j^{(k)}| < \varepsilon, \ \forall j$
2. *noi metode de relaxare* – ex: QDP (quadrant - diagonal partitioning) or QI (Quadrant Interlocking) similară SOR: $A = X - W - Z$

$$x_{ij} = \begin{cases} 0, & \text{if } i \neq j \text{ and } i \neq n - j + 1 \\ a_{ij}, & \text{otherwise} \end{cases}$$

$$w_{ij} = \begin{cases} 0, & \text{if } i \leq j \leq n - i + 1 \text{ or } n - i + 1 \leq j \leq i \\ -a_{ij}, & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 0, & \text{if } j \leq i \leq n - j + 1 \text{ or } n - j + 1 \leq i \leq j \\ -a_{ij}, & \text{otherwise} \end{cases}$$

$X =$  $W =$  $Z =$ 

(a) Jacobi QDP: $Xx^{(k+1)} = (W + Z)x^{(k)} + b$. Convergenţă: dacă A este ireductibil şi diagonal dominant.

(b) Gauss-Seidel QDP: $(X-W)x^{(k+1)} = Zx^{(k)} + b$. Convergenţă: dacă A este ireductibil şi slab diagonal dominant.

(c) tip suprarelexare Jacobi şi SOR: JOQI şi SOQI

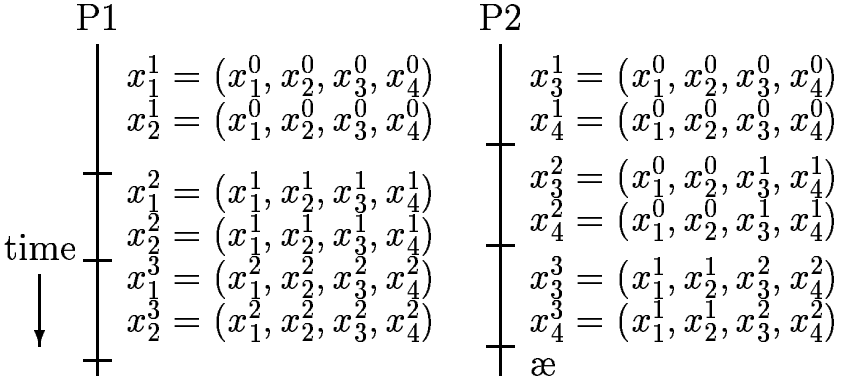
$$x^{(k+1)} = [\omega X^{-1}(W+Z) + (1-\omega)I]x^{(k)} + d,\ k = 0, 1, \ldots$$
$$x^{(k+1)} = (X - \omega W)^{-1}[\omega Z + (1-\omega)X]x^{(k)} + d,\ k = 0, 1, \ldots$$

Condiţii de convergenţă:

- A ireductibil diagonal dominată + $0 < \omega \leq 1 \Rightarrow$ SOQI converge
- A şi X simetrice şi pozitiv definite + $0 < \omega < 2 \Rightarrow$ SOQI converge
- JOQI converge $\Leftrightarrow$ $2\omega^{-1}X - A$ pozitiv definite

Obs: paralelism natural metode iterative cu blocuri 2×2 – soluţiile sistemelor cu matricele X respectiv $X - \omega W$ se det. rezolvând $n/2$ sisteme tip 2×2, care pot fi det. în paralel.

3. *scheme de relaxare asincronă*:
   - orice evaluare utilizează valorile componentelor care sunt disponibile la începutul calculului pe un procesor;
   - fiecare procesor deţien un set independent de componente, acelaşi la fiecare iteraţie
   - probleme: convergenţă iteraţie



Două procesoare care lucrează simultan, asincron, la rezolvarea unui sistem liniar de 4 ecuaţii

4. *scheme de relaxare haotică*: procesele aleg în mod dinamic componentele care vor fi evaluate şi valorile anterioare necesare calculului.
5. *metode gradient* – ex: căutare paralelă, prin start de la un număr de iteraţii iniţiale diferite (figura)

   Fie $p_i^{k,0} = r_i^{k-1} = b - Ax_i^{k-1}$ $(1 \leq i \leq p)$, $\pi_i^k = Ap_i^k$

   Algoritm:

   (a)if $k > 1$ then

   for $i \in P$ do

   $p_i^k = r_i^{k-1}$

   for $j \in P$ do $p_i^k = p_i^k - \frac{<r_i^{k-1}, \pi_j^{k-1}>}{<p_j^{k-1}, \pi_j^{k-1}>} p_j^{k-1}$.

(b)for $i \in P$ do $\pi_i^k = Ap_i^k$

(c)for $i = 1$ to p do

for $j \in \{1, \ldots, i-1\}$ do

$c = \frac{<\pi_j^k, p_i^k>}{<\pi_j^k, p_j^k>}$

$p_i^k = p_i^k - cp_j^k$

$\pi_i^k = \pi_i^k - c\pi_j^k$.

(d)for $i \in P$ do $x_i^k = x_i^{k-1}$, $r_i^k = r_i^{k-1}$

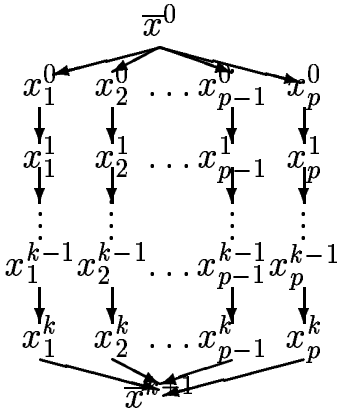
for $j \in P$ do

$c = \frac{<p_j^k, r_i^{k-1}>}{<\pi_j^k, p_j^k>}$

$x_i^k = x_i^k + cp_j^k$

$r_i^k = r_i^k - c\pi_j^k$.

în `for` $i \in P$ `do statement(`i`)` : `statement(`i`)` pot fi efectuate în paralel $i \in P$.



---

2. Metode paralele pentru sisteme tridiagonale:

Sistem liniar tridiagonal: $Ax = d$ cu $d = (d_1, \ldots, d_n) \in \mathbb{R}^n$,

$$A = \begin{pmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & a_{n-1} & b_{n-1} \\ & & & c_{n-1} & a_n \end{pmatrix}$$

1. *reducere ciclică*: vezi curs anterior descompunere LU pt. caz tridiag – calcul U înlocuit cu iteraţii

$$u_i^{(k+1)} = a_i - c_i b_{i-1}/u_{i-1}^{(k)}, \quad k = \overline{1, m},$$

Convergenţă: dependentă de gradul de dominanţă a diagonalei.

2. metode de *relaxare în bloc*: fie $A_p = (c_{i_p}, a_{i_p}, b_{i_p})$, $i_p \in \overline{M_p, M_{p+1} - 1}$ (corespunzătoare $M_1 = 1$, $M_i \leq M_{i+1}$, $M_p = n$)
Coef. b_{M_p} and c_{M_p} cuplează blocurile A_{p-1} şi A_p.

Fie $A = G - H$ cu

$$G = \begin{pmatrix} A_1 & 0 & & & \\ 0 & A_2 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & A_{p-2} & 0 \\ & & & 0 & A_{p-1} \end{pmatrix}$$

şi H conţine numai coef de cuplare b_{M_p} şi c_{M_p}
Metoda de relaxare:

$$Gx^{(k+1)} = b + Hx^{(k)}, \qquad x^{(0)} = G^{-1}b$$

Paralelism: rezolvare p siteme tridiagonale local pe procesoare diferite (fără comunicaţii) + multiplicare H cu un vector (comunicaţii!).

3. paralelizare prin *permutare* în caz $n = 2^k$ şi SOR:

$$\begin{aligned} a_1x_1^{(k+1)} &= (1-\omega)a_1x_1^{(k)} - \omega b_1x_2^{(k)} + \omega d_1 \\ a_ix_i^{(k+1)} &= (1-\omega)a_ix_i^{(k)} - \omega c_ix_{i-1}^{(k+1)} - \omega b_ix_{i+1}^{(k)} + \omega d_i, \ i = \overline{2, n-1} \\ a_nx_n^{(k+1)} &= (1-\omega)a_nx_n^{(k)} - \omega c_nx_{n-1}^{(k+1)} + \omega d_n \end{aligned}$$

nu permite paralelizare!
Fie P matrice de permutare a.î: $PAP^Ty = \overline{A}y = \overline{d}$,

$$y = (x_1, x_3, \ldots, x_{2^k-1}, x_2, \ldots, x_{2^k})^T, \ \overline{d} = (d_1, d_3, \ldots, d_{2^k-1}, d_1, \ldots, d_{2^k})^T$$

$$\overline{A} = \begin{pmatrix} a_1 & & & & & & b_1 & & & \\ & a_3 & & & & & c_3 & b_3 & & \\ & & \ddots & & & & & \ddots & \ddots & \\ & & & & a_{2^k-1} & & & & c_{2^k-1} & b_{2^k-1} \\ c_2 & b_2 & & & & & a_2 & & & \\ & c_4 & b_4 & & & & & a_4 & & \\ & & \ddots & \ddots & & & & & \ddots & \\ & & & & b_{2^k-2} & & & & & \\ & & & & c_{2^k} & & & & & a_{2^k} \end{pmatrix}.$$

SOR aplicată la $\overline{A}y = \overline{d} = z$:

$$\begin{aligned} a_1y_1^{(k+1)} &= (1-\omega)a_1y_1^{(k)} + \omega z_1 \\ a_{2i-1}y_i^{(k+1)} &= (1-\omega)a_{2i-1}y_i^{(k)} - \omega c_{2i-1}y_{n/2+i-1}^{(k)} - \omega b_{2i-1}y_{n/2+i}^{(k)} + \omega z_i, \ i = \overline{2, n/2} \\ a_{2i}y_{n/2+i}^{(k+1)} &= (1-\omega)a_{2i}y_{n/2+i}^{(k)} - \omega c_{2i}y_i^{(k+1)} - \omega b_{2i}y_{i+1}^{(k+1)} + \omega z_{n/2+i}, \ i = \overline{1, n/2-1} \\ a_ny_n^{(k+1)} &= (1-\omega)a_ny_n^{(k)} - \omega c_ny_{n/2}^{(k+1)} + \omega z_n. \end{aligned}$$

Obs: dacă $y_1^{(k)}, \ldots, y_n^{(k)}$ cunoscute $\Rightarrow$ $y_1^{(k+1)}, \ldots, y_{n/2}^{(k+1)}$ pot fi calc. în paralel, apoi $y_{n/2+1}^{(k+1)}, \ldots, y_n^{(k+1)}$ pot fi calc. în paralel.

METODE ITERATIVE PARALELE DE REZOLVARE A SISTEMELOR DE ECUAŢII NELINIARE

Problema: rezolvarea sistemului $F(x) = 0$, unde $F : R^n \to R^n$.

---

0. Metode secvenţiale acceptate: metoda secantei (caz $n = 1$) metoda Newton, metode de tip gradient conjugat.

---

1. Metode paralele pentru cazul $n = 1$

A. Generalizarea metodei bisecţiei

Idee: creşterea numărului de evaluări de funcţie.

Pasul i:

1. alegerea a k puncte distincte în intervalul curent
2. evaluarea în paralel a valorilor funcţiei în punctele ales
3. determinarea unui nou interval de aproximare (valorile între care semnul funcţiei se schimbă)
4. dacă lungimea noului interval este mai mare decât nivelul de eroare admis se reiau paşii de mai sus

Obs: dacă valorile alese sunt valori echidistante (generare metoda bisecţiei), cu cât numărul acestora este mai mare cu atât lungimea noului interval de aproximare la fiecare pas este mai mică, deci alg. este mai rapid.

. . . . . .

B. Metoda Newton asincronă

Iteraţii Newton: $x_{k+1} = x_k - f_2/f_1$, $f_2 = f(x_k)$, $f_1 = f'(x_k)$. Idei:

1. se consideră disponibile 2 procesoare
2. variabilele ce trebuie actualizate sunt x, f_1, f_2
3. pp. f_1 actualizat de un procesor, x, f_2 de al doilea
4. procesoarele utilizează valorile curente ale variabilelor globale fară a aşetpta terminarea pasului de către celălalt procesor.

Obs: formula este modificată:

$$x_{k+1} = xk - f(x_k)/f'(x_j), \qquad j \leq k$$

şi se schimbă ordinul de convergenţă a metodei.

Ex: linie orizontală = comunicare, timp pe verticală

Proc. 1	Memorie	Proc.2
$f(x_1)$	$x_1, f(x_0)$	$x_2 = x_1 - f(x_1)/f'(x_0)$
	$x_2, f(x_0)$	$x_3 = x_2 - f(x_2)/f'(x_0)$
$f(x_3)$	$x_3, f(x_0)$	$x_4 = x_3 - f(x_3)/f'(x_0)$
	$x_4, f(x_1)$	$x_5 = x_4 - f(x_4)/f'(x_1)$
	$x_5, f(x_1)$	$x_6 = x_5 - f(x_5)/f'(x_1)$

---

2. Metode paralele pentru cazul multidimensional $n > 1$

A. Multiplexarea extrapolării Lagrange

Obs: generalizare metodă bisecţie sau secantă pt. cazul multidimensional

Idei:

(a) producerea unei secvenţe de vectori a căror componente converg spre rădăcină.
(b) la un pas primul procesor utilizează cele mai noi m aproximaţii şi determină nouă aproximaţie utilizând L_{m+1}, polinomul de interpolare Lagrange a lui $f(x)$ în cele m aproximaţii şi extrapolează L_{m+1} la 0.
(c) simultan procesorul 2 utilizează cele mai curente $m+1$ aproximaţii şi determină L_{m+2}, extrapolează şi determină noua aproximaţie, etc.
(d) astfel r procesoare produc r noi aproximaţii ale rădăcinii

Fie $x_n = (x_n^1, \ldots, x_n^2)$ = vector cu r aproximaţii ale rădăcinii lui $f(x)$.

Atunci x_{n+1} det. astfel:

1. se alege $m \geq 2$;
2. se det. r polinoame de interpolare Lagrange $L_{m+j}(x)$ de grade $m+j-2$, $j = \overline{1,r}$;
3. pt. fiecare j, $j = \overline{1,r}$, $L_{m+j}(x)$ interpolează punctele $(x_\alpha^\beta, f(x_\alpha^\beta))$ unde $\beta = r - \gamma + \lfloor\gamma\rfloor - r\delta(0, \gamma/r - \lfloor\gamma\rfloor/r$, $\alpha = n\lfloor(\gamma-1)/r\rfloor$, pt. fiecare $\gamma = \overline{1, m+j-1}$, unde δ =simbolul Kronecker.

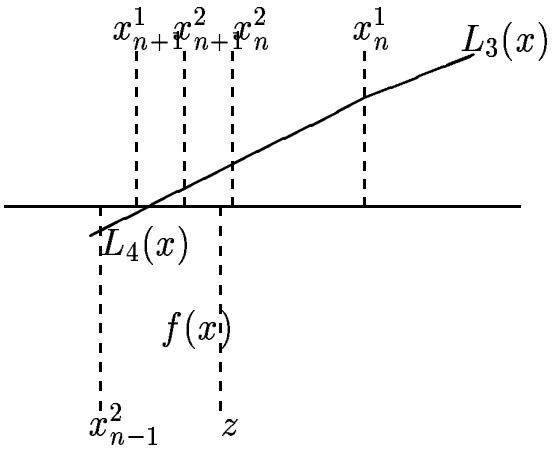
Notaţie: x_{n+1}^j, $j = \overline{1,r}$ rădăcina lui $L_{m+j}(x)$ şi F_m funcţie de $\lfloor(m-1)/r\rfloor + 2$ variabile vector pt. care $x_{n+1} = F_m(x_n, x_{n-1}, \ldots, x_{n-1-\lfloor(m-1)/r\rfloor})$.

Caz particular: $r = 2$, $m = 2$, $F_2 = F_2(x_n, x_{n-1})$ unde

$$F_2 = \Big(\frac{x_n^1 f_n^2 - x_n^2 f_n^1}{f_n^2 - f_n^1}, \frac{f_n^1 f_n^2 x_{n-1}^2}{(f_{n-1}^2 - f_n^1)(f_{n-1}^2 - f_n^2)} + \frac{f_{n-1}^2 f_n^2 x_n^1}{(f_n^1 - f_{n-1}^2)(f_n^1 - f_n^2)} + \frac{f_{n-1}^2 f_n^1 x_n^2}{(f_n^2 - f_{n-1}^2)(f_n^2 - f_n^1)}.\Big)$$

Obs:

(a) noile aprox. x_{n+1}^1 şi x_{n+1}^2 sunt obţinute din 3 aprox. x_{n-1}^2, x_n^1, x_n^2.
(b) x_{n+1}^1 este det. din coarda asociată cu x_n^1 şi x_n^2 iar x_{n+1}^2 este det. din parabole asociată cu x_{n-1}^2, x_n^2 şi x_n^2.
(c) viteza alg. paralel logaritmică în nr. de procesoare



. . . . . .

B. Multiplexarea interpolării Hermite

Obs:

(a) pol. de interpolare Hermite $H(x)$ a lui $f(x)$ este parametrizat prin perechile (x_i, b_i), $i = \overline{1,m}$, a.î. $H^{(j)}(x) = f^{(j)}(x_j)$, $j = \overline{0, b_i - 1}$, $i = \overline{1,m}$.
(b) dacă $N + 1 = \Sigma_{i=1}^m b_i$, $\exists!$ pol. Hermite de grad $\leq N$
(c) pol. Hermite corespunzător datelor (x_i, b_i), $i = \overline{1,m}$ poate fi utilizat pt. extrap. la 0 pt. a găsi o nouă iteraţie x_{m+1}

Caz $b_j = jb$ – alg paralel cu m procesoare:

1. proc. k ($k = \overline{1,m}$) efectuează b evaluări $f^{kb-i}(x_{n-i+1})$, $i = \overline{1,b}$, unde x_n este ceamai nouă aprox. a rădăcinii
2. x_{n+1} este calc. prin extrapol. pol. Hermite la zero
3. x_{n-j+1}, $i = \overline{1,b}$ sunt actualizate prin $x_{n-i+1} = x_{n-i+2}$, $i = \overline{1,b}$.

Ex.: $m = 2$, $b = 1$,

- primul procesor evaluează $f(x_n)$
- proc. secundar eval. $f'(x_{n-1})$
- pol. Hermite corespunzător perechilor $(x_n, 1)$ şi $(x_{n-1}, 2)$ este construit
- x_{n+1} este det. prin extrapol.

Obs: algoritmul cu $b = 2$ şi $m = 2$ utilizează tot 2 proc. (primul eval. $f(x_n)$ şi $f'(x_n)$, secundar eval. $f''(x_{n-1})$ şi $f'''(x_{n-1})$).

. . . . . .

C. Metode de tip Newton

1. metoda Newton discretă (convergenţă pătratică)

$$x^{(k+1)} = x^{(k)} - J(x^{(k)})^{-1} F(x^{(k)}), \quad F = (f_1, \ldots, f_n), \quad J(x) = (\partial f_i / \partial x_j)_{1 \leq i,j \leq n}$$

Eval. J complexă $\to J_k \approx J(x_k)$:

$$J_k c_j = [F(x^{(k)}) + \varepsilon_k c_j) - F(x^{(k)})]/\varepsilon_k, \quad 1 \le j \le n \qquad (c_j)_k = \begin{cases} 1, & k = j \\ 0, & \text{altfel} \end{cases},$$

unde$0 < \varepsilon \le \|F(x^{(k)})\|$.

Idee: evaluarea Jacobianului de-a lungul a mai multe iteraţii. Variante:

(a) actualizare treptată coloane (alg. sincron)

(b) reevaluare după un nr. fix de iteraţii (calcul în paralel cu iteraţiile, alg. sincron)

(c) reevaluare după un număr arbitrar de iteraţii (calcul în paralel cu iteraţiile, alg. asincron)

2. metoda Jacobi-Newton (convergenţă superliniară)

$$x_i^{(k+1)} = x_i^{(k)} - J_i^{-1} F(x_1^{(k)}, \ldots, x_i^{(k)}, \ldots, x_n^{(k)}) \quad i = 1, \ldots, n$$

$$J_i = \frac{\partial f_i(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i, x_{i+1}^{(k)}, \ldots, x_n^{(k)})}{\partial x_i}\Big|_{x_i = x_i^{(k)}}$$

Obs: fiecare componentă a vectorului iterativ poate fi calculat independent cunoscând valorile anterioare.

3. metoda Gauss-Seidel-Newton modificată a.î să se aplic. în paralel + convergenţă pătratică:

$$x_i^{(k+1)} = x_i^{(k)} - J_i^{-1} F(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i^{(k)}, x_{i+1}^{(k)} \ldots, x_n^{(k)}) \quad i = 1, \ldots, n$$

cu J_i de mai sus; se aplică în forma:

$$x_i^{(k,0)}) = x_i^{(k)}$$

$$x_i^{(k,l+1)} = x_i^{(k,l)} - J_i^{-1} F(x_1^{(k+1,l)}, \ldots, x_{i-1}^{(k+1,l)}, x_i^{(k,l)}, x_{i+1}^{(k,l)} \ldots, x_n^{(k,l)}), \quad l = 1, \ldots, m-1$$

$$x_i^{(k+1)} = x_i^{(k,m)}.$$

. . . . . .

D. Reducerea problemei la un sistem liniar

Ex: se consideră iteraţiile Newton

$$x^{(k+1)} = x^{(k)} - [f'(x^{(k)})]^{-1} f(x^k)$$

care pot fi scrise

$$f'(x^{(k)})(x^{(k+1)} - x^{(k)}) = -f(x^{(k)})$$

şi sistemului liniar i se aplică un alg. paralel.

3. Metode pentru funcţii speciale

Ecuaţii polinomiale

Obs: alg. secvenţiali conţin un grad de paralelism (Durant-Kerner sau Ehrlich)

Ex: alg. Durant-Kerner

Problema: se caută o descompunere a polinomului P: $P(z) = \Pi_{k=1}^{n}(z - z_i^*)$.

1. Fie $z_1,\ z_2, \ldots,\ z_n$ val. aprox. a $z_1^*,\ z_2^*, \ldots,\ z_n^*$,
2. Fie $Q(z) = \Pi_{z=1}^{n}(z - z_i)$. Atunci $P'(z) \approx Q'(z)$.
3. Iteraţii Newton modificate:

$$\begin{aligned} z_1^{(k+1)} &= z_1^k - P(z_1^{(k)})/Q'(z_1^{(k)}) \\ &\vdots \\ z_n^{(k+1)} &= z_n^k - P(z_n^{(k)})/Q'(z_n^{(k)}) \end{aligned}$$

Obs: cele n eval. polinomiale + n aprox. noi pot fi calculate independent pe un calculator paralel.

METODE PARALELE
DE DETERMINARE A VALORILOR PROPRII ŞI OPTIMIZARE

A. DETERMINAREA VALORILOR PROPRII

Problema: $?(\lambda, x):\ Ax = \lambda x,\ x \neq 0$, unde A matrice simetrică $n \times n$
λ = val.prop. A, x = vector propriu A
Obs:

- valorile proprii sunt rădăcinile $\det(A - \lambda I) = 0$, notate $\lambda_1, \ldots, \lambda_n$, de multiplicitate m_λ
- A e simetrică $\Rightarrow$ valori proprii reale.

.....

1. Metoda planelor de rotaţie

Idee: matricea A este redusă la forma diagonală care conţine val. proprii ale lui A prin aplicarea unor rotaţii; elementul $a_{pq}^{(k)}$ (cel mai mare în $|\cdot|$) anihilat printr-o rotaţie în planele (p, q) de unghi $\alpha_{pq}^{(k)}$,

$$A^{(1)} = A, \quad A^{(k+1)} = J^{(k)} A^{(k)} (J^{(k)})^T, \quad k = 1, 2, \ldots$$

$$J_{ij}^{(k)} = \delta_{ij},\ i, j \neq p, q, \quad J_{(p,q)}^{(k)} = \sin \alpha_{pq}^{(k)} = -J_{(q,p)}^{(k)}, \quad J_{(p,p)}^{(k)} = \cos \alpha_{pq}^{(k)} = J_{(q,q)}^{(k)}.$$

$$|a_{pq}^{(k)}| = \max_{i \neq j} |a_{ij}^{(k)}|.$$

$$\alpha_{pq}^{(k)} = \frac{1}{2} \tan^{-1} \left(\frac{2a_{pq}^{(k)}}{a_{pp}^{(k)} - a_{qq}^{(k)}} \right), \quad -\frac{1}{4}\pi < \alpha_{pq}^{(k)} < \frac{1}{4}\pi.$$

$$\lim_{k \to \infty} A^{(k)} = D$$

Aplicare simultană a rotaţiilor:

Obs: fiecare rotaţie R_{pq} afectează liniile p şi q, coloanele p şi q of A
R_{pq} şi R_{rs} sunt disjuncte dacă indicii $p,\ q,\ r$, şi s sunt toate distincte
Ex: scheme ciclice Jacobi scheme (rată convergenţă pătratică) – fiecare etapă anihilare a $N = \frac{1}{2} n(n-1)$ perechi din $((p, q) | 1 \leq p < q \leq n)$.

1. metoda linie-ciclică

$$(p_0, q_0) = (1, 2),$$

$$(p_{k+1}, q_{k+1}) = \begin{cases} (p_k, q_k + 1), & \text{dacă } p_k < n-1, \text{ and } q_k < n \\ (p_k + 1, p_k + 2), & \text{dacă } p_k < n-1, \text{ şi } q_k = n \\ (1, 2), & \text{dacă } p_k = n-1, \text{ şi } q_k = n. \end{cases}$$

2. metoda coloană-ciclică

$$(p_0, q_0) = (1, 2),\ (p_{k+1}, q_{k+1}) = \begin{cases} (p_k + 1, q_k), & \text{dacă } p_k < q_k - 1, \text{ şi } q_k \leq n \\ (1, q_k + 1), & \text{dacă } p_k = q_k - 1, \text{ şi } q_k < n \\ (1, 2), & \text{dacă } p_k = n-1, \text{ şi } q_k = n. \end{cases}$$

Obs: schemele ciclice de mai sus sunt indicate numai în cazul secvenţial deoarece prechi consecutive nu sunt disjuncte.

? set de rotaţii $R_i = \{R_{p,q}\}$: $\{p_i, q_i\} \cap \{p_j, q_j\} = \emptyset$, $i \neq j$ care pot fi efectuate simultan.

Obs: max. nr. rotaţii simultane: $\lfloor \frac{1}{2}n \rfloor$, nr. de paşi: $2\lceil \frac{1}{2}n \rceil - 1$ paraleli în loc $\frac{1}{2}n(n-1)$ în caz paralel.

Ex:

1. schema Sameh

$$e(n) = \begin{cases} n, & \text{if } n \text{ este par} \\ n-1, & \text{if } n \text{ impar} \end{cases} \qquad o(n) = \begin{cases} n-1, & \text{if } n \text{ par} \\ n, & \text{if } n \text{ impar} \end{cases}$$

$$Z = \{(p,q) |\ 1 \leq p < q \leq n\},$$

$Z_k \subseteq Z$, $k = \overline{1, o(n)}$ a.î. $\{Z_1, \ldots, Z_{o(n)}\}$ partiţionare Z şi $|Z_k| = \dfrac{1}{2}e(n)$, $k = \overline{1, o(n)}$

Anihilare simultană a $\frac{1}{2}e(n)$ elemente a lui $A^{(k)}$ indexate după elem. Z_k,

$$A^{(1)} = A, \qquad A^{(k+1)} = J^{(k)} A^{(k)} (J^{(k)})^T, \quad k = \overline{1, o(n)}$$

$$J^{(k)} = -J^{(k)} = \sin\theta_{pq}, \ J^{(k)}(p,p) = J^{(k)}(q,q) = \cos\theta_{pq},$$

$$J^{(k)}(i,j) = \delta_{ij}, \text{ dacă } \{i,j\} \not\subseteq \{p,q\} \quad (p,q) \in Z_k$$

$$m = \frac{1}{2}\lfloor \frac{1}{2}(n+1) \rfloor$$

Z_k cu $k = \overline{1, 2m-1}$ definit astfel:

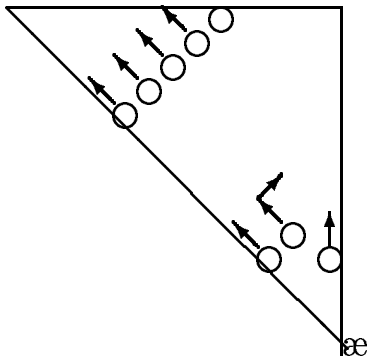
(a) for $k = \overline{1, m-1}$, $q = \overline{m-k+1, n-k}$

$$p = \begin{cases} 2m-2k+1-q, & \text{if } m-k+1 \leq q \leq 2m-2k, \\ 4m-2k-q, & \text{if } 2m-2k < q < 2m-k, \\ n, & \text{if } 2m-k \leq q \leq n-k. \end{cases}$$

(b) for $k = \overline{m, 2m-1}$, $q = \overline{4m-n, 3m-k-1}$

$$p = \begin{cases} n, & \text{if } 4m-n \leq q \leq 2m-k, \\ 4m-2k-q, & \text{if } 2m-2k < q < 4m-2k, \\ 6m-2k-1-q, & \text{if } 4m-2k \leq q \leq 3m-k-1. \end{cases}$$

Figura: poziţia elementelor din pasul $(k+1)$ deduse din pasul k:



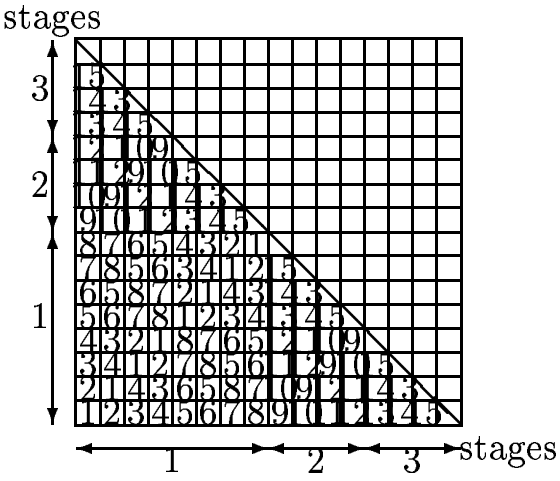
2. scheme mobile de generare a rotaţiilor:
   (a)

Column	1	2	3	4	5	6	7	8	9	Rotations
Initial	·←	(2)←	·←	(4)←	·←	(6)←	·←	(8)←	·	
	↓								↑	
	(1)→	·→	(3)→	·→	(5)→	·→	(7)→	·→	(9)	
Stage 1	(2)←	·←	(4)←	·←	(6)←	·←	(8)←	·←	(9)	(1,2),(3,4),(5,6),(7,8)
	↓								↑	
	·→	(1)→	·→	(3)→	·→	(5)→	·→	(7)→	·	
Stage 2	·←	(4)←	·←	(6)←	·←	(8)←	·←	(9)←	·	(1,4),(3,6),(5,8),(7,9)
	↓								↑	
	(2)→	·→	(1)→	·→	(3)→	·→	(5)→	·→	(7)	
Stage 3	(4)←	·←	(6)←	·←	(8)←	·←	(9)←	·←	(7)	(2,4),(1,6),(3,8),(5,9)
	↓								↑	
	·→	(2)→	·→	(1)→	·→	(3)→	·→	(5)→	·	
⋮ Stage 9										⋮

   (b)

$k\backslash i$	1 2 3 4 5 6 7 8
1	1 2 3 4 5 6 7 8
2	1 2 3 4 5 6 7 8
3	2 3 4 5 6 7 8 1
4	2 3 4 5 6 7 8 1
⋮	

3. algoritm bloc recursiv – ex. anihilare



. . . . . .

Alte metode:

1. scheme ciclise pentru matrice bidiagonale
2. metoda Householder (alt tip de rotaţii)
3. metoda bisecţiei (determinare intervale pt. val. proprii)
4. metoda puterii, de determinare a valorii proprii dominante

B. OPTIMIZARE

Probl. (fără constrâng.): det. extreme $\min_{x\in D} f(x),\ \max_{x\in D} f(x),\ D \subset R^m$.

Metode secvenţiale:

1. utilizarea derivatei
2. metode pt. cazul unidimensional:
   (a) divizarea intervalului de căutare în 3 părti egale şi reducerea acestuia la fiecare pas cu 1/3 din interval (2 evaluări de funcţie per pas)
   (b) divizarea intervalului în părţi inegale: cea mai rapidă este metoda secţiunii de aur care se bazează pe o diviziune construită pornind de la şirul lui Fibonacci (1 eval. funcţie per pas)
3. metode de tip gradient pt. cazul multidimensional
4. metode non-gradient, pe bază de direcţii de căutare/minimizare

Idee metode paralele:

1. pt. cazul unidimensional, mai multe evaluări de funcţii simultan
2. pt. cazul multidimensional, căutare în mai multe direcţii simultan

. . . . . .

Cazul unidimensional. Metoda Kiefer a secţiunii de aur generalizată

Pp. lungimea interval de căutare L satisface $G_r(k) \le L \le G_{r+1}(k)$ unde

$$G_r(k) = \tfrac{k+1}{2}(G_{r-1}(k) + G_{r-2}(k)),\ G_0(k) = 1,\ G_1(k) = \tfrac{k+1}{2},\ k\text{impar},$$
$$G_r(k) = \tfrac{k+1}{2} G_{r-1}(k),\ G_0(k) = 1,\ k \text{ par}.$$

şi funcţia $f : [0, L] \to \mathbb{R}$ este unimodală adică pct. de maxim $\tilde{x} \in [0, L]$ a.i.

$$\text{if } x < y \le \tilde{x} \text{ then } f(x) < f(y) \text{if } \tilde{x} \le x < y \text{ then } f(x) > f(y).$$

Atunci $r + 1$ paşi a $k + 1$-upluri de evaluări de funcţii (paşi paraleli) sunt suficiente pt. a localiza pct. maxim în intervalul unitate.

Pas 1. Dacă

$$L = \begin{cases} \theta G_r(k), & 0 < \theta < 1,\ k \text{ impar} \\ G_r(k) - \varepsilon, & G_r(k) - \varepsilon,\ 0 < \varepsilon < 1,\ k \text{ par} \end{cases}.$$

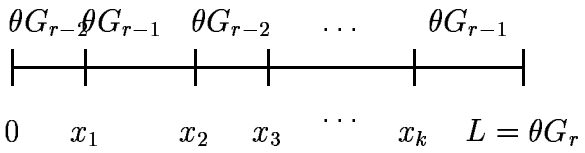
atunci cele k puncte în care f este evaluată, $x_1,\ x_2,\ \ldots,\ x_k$ sunt

$$x_{2t} = \begin{cases} \theta t(G_{r-2}(k) + G_{r-1}(k)), & t = \overline{1, (k-1)/2},\ k \text{ impar} \\ tG_{r-1}(k), & t = \overline{1, k/2},\ k \text{ par}. \end{cases}$$

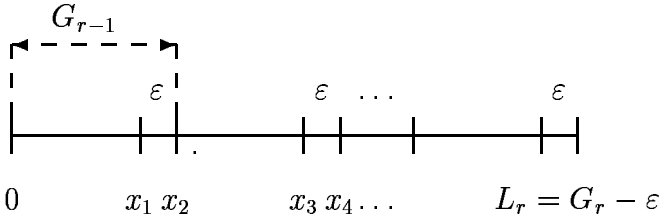
$$x_{2t+1} = \begin{cases} \theta((t+1)G_{r-2}(k) + tG_{r-1}(k)), & t = \overline{0, (k-1)/2},\ k \text{ impar} \\ x_{2t} - \varepsilon, & t = \overline{1, k/2},\ k \text{ par}. \end{cases}$$

(descompun $[0, L]$ în $k + 1$ subintervale adiacente de lungimne alternative

$$\begin{cases} \theta G_{r-1}(k), & k \text{ impar} \\ G_{r-1}(k) - \varepsilon, & k \text{ par}, \end{cases} \qquad \begin{cases} \theta G_{r-2}(k), & k \text{ impar} \\ \varepsilon, & k \text{ par}. \end{cases}$$



k odd

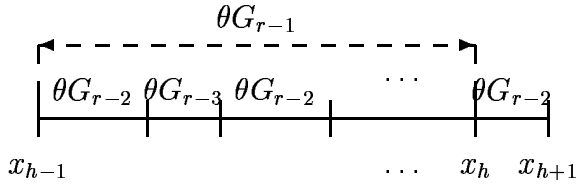


k even

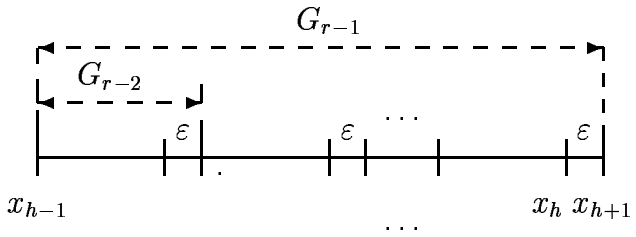
Obs: dacă x_h corespunde la cea mai mare valoare f atunci val. max. căutată se află în subintervalul $[x_{h-1}, x_{h+1}]$.
Lungime interval: $\theta G_{r-1}(k)+\theta G_{r-2}(k)$, dacă k este impar, $G_{r-1}(k)$ dacă k este par.
Pas 2. Eval. f în $x_{h-1}+y_1, \ldots, x_{h-1}+y_k$

$$y_{2t} = \begin{cases} \theta t(G_{r-3}(k)+G_{r-2}(k)), & t = \overline{1,(k-1)/2},\ k \text{ impar} \\ tG_{r-2}(k), & t = \overline{1,k/2},\ k \text{ par.} \end{cases}$$

$$y_{2t+1} = \begin{cases} \theta((t+1)G_{r-2}(k)+tG_{r-3}(k)), & t = \overline{0,(k-1)/2},\ k \text{ impar} \\ y_{2t}-\varepsilon, & t = \overline{1,k/2},\ k \text{ par.} \end{cases}$$



k odd

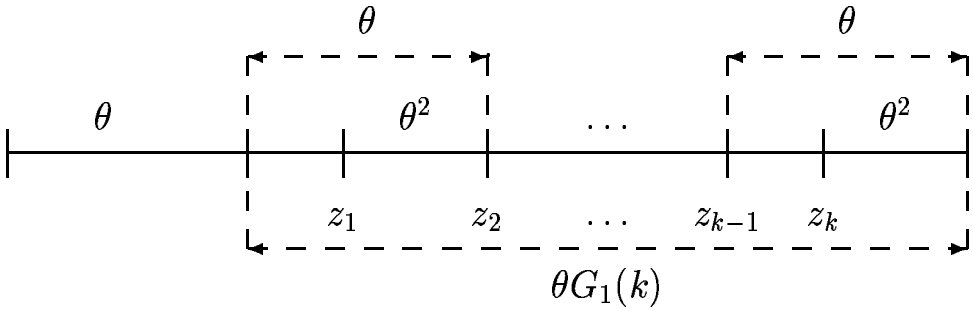


k even

$\vdots$

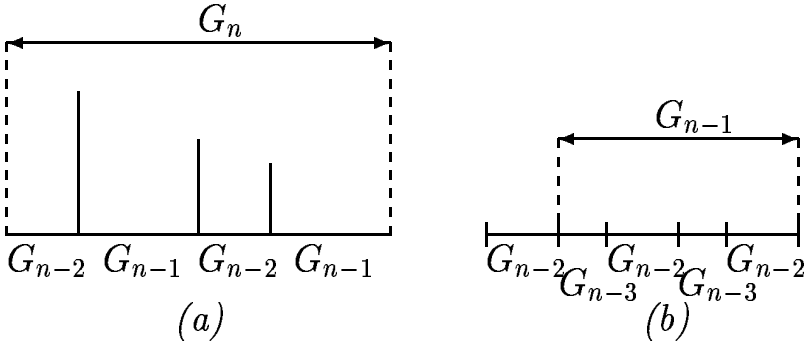
Pas $r-1$ dacă k este impar, r dacă k este par. Pct. $(z_1, \ldots, z_k)$ where

$$z_{2t} = a+(t+1)\theta,\ t = \overline{1,(k-1)/2},\ z_{2t+1} = a+(t+1)\theta+\theta(1-\theta),\ t = \overline{0,(k-1)/2}.$$



Lungime a 2 subintervale consecutive: $\theta+\theta(1-\theta) < 1$ şi $\theta < 1$.

Ex: $k = 3$ şi $L = G_n(3)$



. . . . . .

<u>*Cazul multidimensional. Metoda Powell*</u>

Secvenţial:

- se dă un pct. p şi m direcţii $v^1, \ldots, v^m$.
- pornind la p se fac m minimizări univariate în secvenţă în direcţiile $v^1, \ldots, v^m$
- minimizarea în direcţia v^j porneşte din pct. furnizat de minimizarea furnizată de v^{j-1}, $j = \overline{2, m}$
- această procedură produce o traiectorie diagonală care se termină într-un pct, fie q
- minimizare finală pornind de la q în direcţia $v^{m+1} = q - p$; rezultă un pct. r
- ciclul se reia cu vectori actualizaţi $r \to p$, $v^{i+1} \to v^i$, $i = \overline{1, m}$
- pt. $f(x) = x^T Ax + bx + c$, cu A matrice $m \times m$ simetrică alg. se termină în max. m cicluri şi m^2 minimizări în m^2 paşi

Paralel:

- U set de m vcetori unitari linear independenţi $u^1, \ldots, u^m$
- la fiecare ciclu se selectează un vector din U în ordine ciclică
- se dă un pct. p şi $m - 1$ vectori $v^2, \ldots, v^{m-1}$
- se selectează un vector din U , fie v^m.
- pornind de la p, se construieşte o linie poligonală cu ajutorul vectorilor $v^1, \ldots, v^m$
- din fiecare vârf a liniei poligonale (fără p) se efectuează m minimizări univariate simultane în direcţia v^1, producându-se deplasamentul $\alpha_i v^1$, $i = \overline{1, m}$ de la vârful $p + \Sigma_{j \le i} v^j$, $i = \overline{1, m}$ a liniei poligonale
- se actualizează p şi $v^1, \ldots, v^{m-1}$: $p + v^1 + \alpha_1 v^1 \to p$, $v^{i+1} + (\alpha_{i+1} - \alpha_i) v^1 \to v^i$, $i = \overline{1, m-1}$
- se reia ciclul.
- în caz f cuadratic, m^2 minimizări în m paşi paraleli

METODE PARALELE PENTRU ECUAŢII DIFERENŢIALE ORDINARE - PARTEA I

Problema (cu valori iniţiale IVP pt. ecuaţii diferenţiale ordinare ODE): $\begin{cases} y'(t) = f(t, y(t)), \\ y(t_0) = y_0 \end{cases}$
unde $t \in [t_0, t_0 + T]$, $f : [t_0, t_0 + T] \times \mathbb{R}^n \to \mathbb{R}^n$.

De ce coduri paralele?

1. cazul $n \gg 1$ (de exemplu, dacă sistemul este obţinut din ecuaţii ciu derivate parţiale dependente de timp)
2. soluţiile sunt necesare în timp real;
3. cazul $T \gg 1$;
4. integrări repetate;
5. evaluarea funcţiilor este costisitoare;
6. sistemul este rigid (metodele explicite rapide nu pot fi utilizate).

Stare-de-fapt:

(a) nr. algoritmilor paraleli pt. ODE $\ll$ nr. alg. paraleli pt. sisteme liniare sau ec. cu derivate parţiale;
(b) majoritatea metodelor nu au fost testate în implementare;
(c) măsurătorile de eficienţă nu trebuie efectuate relativ la cea mai apropiată metodă secvenţială, ci cea optimală (nu există reţeta perfectă! precum în cazul ec. liniare);
(d) cercetările privind alg. paraleli pt. ODE au fost efectuate cu preponderenţă după 1990.
Raţiune: integrarea numerică a IVP prin metode cu diferenţe este de natură secvenţială.

Cum? Utilizând <u>metode multivalori</u> şi tehnici paralele bazate pe

- paralelism în sistem;
- parallelism în metodă;
- parallelism în timp.

<u>*O metodă serială „bună”*</u>

Pp. că

(a) intrevalul de integrare este divizat: $t_0 < t_1 < \cdots < t_N = t_0 + T$
(b) metoda numerică oferă o aproximare: $y_n \approx y(t_i)$.

Stabilitate:

Metoda numerică este stabilă dacă pt. fiecare ODE stabilă, există K şi h_0 a.î.

$$\|y_n - \overline{y}_n\| \leq K\|y_0 - \overline{y}_0\|, \ \forall h : \ 0 < h < h_0,$$

unde y_n sunt calculate în secvenţă $y_0 \to y_1 \to \ldots \to y_n$ utilizând h, iar $\overline{y}_n$ în $\overline{y}_0 \to \overline{y}_1 \to \ldots \to \overline{y}_n$ utilizând h.

Regiunea de stabilitate absolută:

$$A = \{h\lambda | \text{ aplicând metoda la: } y' = \lambda y,\ y(t_0) = y_0,$$
$$\text{cu pasul constant } h > 0, \text{ se obţin valorile aproximative}$$
$$y_n \text{ pt. care } y_n \to 0,\ \text{când } n \to \infty\}$$

Acurateţe: Fie

(a) metoda unipas $y_{n+1} = y_n + h\Phi(t_n, y_n, h)$

(b) $z_{n+1} := y(t_n) + h\Phi(t_n, y(t_n), h)$.

Atunci

(a) $TE = y(t_{n+1}) - z_{n+1}$ este eroarea locală de discretizare;

(b) $y(t_{n+1}) - y_{n+1}$ este eroarea globală la t_{n+1};

Dacă $TE = Ch^{p+1}\varphi(t_n, t(t_n)) + \mathcal{O}(h^{p+2})$, atunci p=ordin de acurateţe.

Condiţie de consistenţă: $p \geq 1$.

Acurateţe=consistenţă + constantă de eroare mică C.

Convergenţă: Fie $y_h(t)$ fcţ. de interpolare a valorilor aprox,calculate cu pas h. Metoda este convergentă dacă pt. un IVP dat

$$\lim_{h \to 0} \|y(t) - y_h(t)\| = 0, \quad \forall t \in [t_0, t_0 + T].$$

Convergenţă = Stabilitate + Consistenţă.

Paralelism în sistem

Fie metoda multi-pas explicită:

$$\begin{aligned} y_j^{(i+1)} &= g(y_j^{(i)}, y_j^{(i-1)}, \ldots, y_j^{(i-p)}, f_j^{(i-1)}, \ldots, f_j^{(i-q)}, h),\ j = \overline{1,n}, \\ f_j^{(i)} &= f_j(t_i, y_1^{(i)}, \ldots, y_n^{(i)}),\ t_i = t_0 + ih. \end{aligned}$$

1. procesare paralelă la *nivel de sarcină majoră*: met.de segmentare a ec. (ES) $\{1, \ldots, n\} = \cup_{l=1}^{m} J_l$ (figura)
2. procesare paralelă la *nivel de sarcină medie*: partiţionarea evaluării funcţiei sistemului.
3. procesare paralelă la *nivel de sarcină minoră*: partiţionarea evaluării funcţiei sistemului la nivel de operator de bază (de ex., utlizând un alg. paralel pt. ec. liniare recurente)

Avantaje: utilizarea unei metode numerice arbitrare

Dezavantaje:

(a) grad scăzut de paralelism;

(b) viteza alg. paralel depinde de sistemul de ODE-uri (dependent de utilizator).

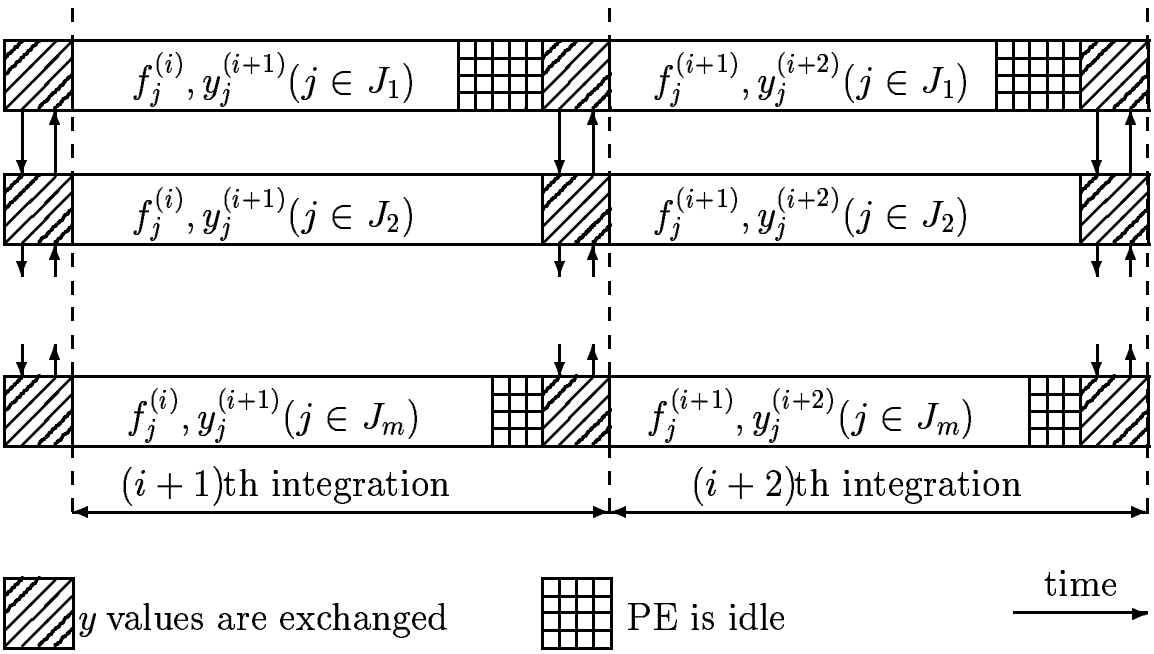
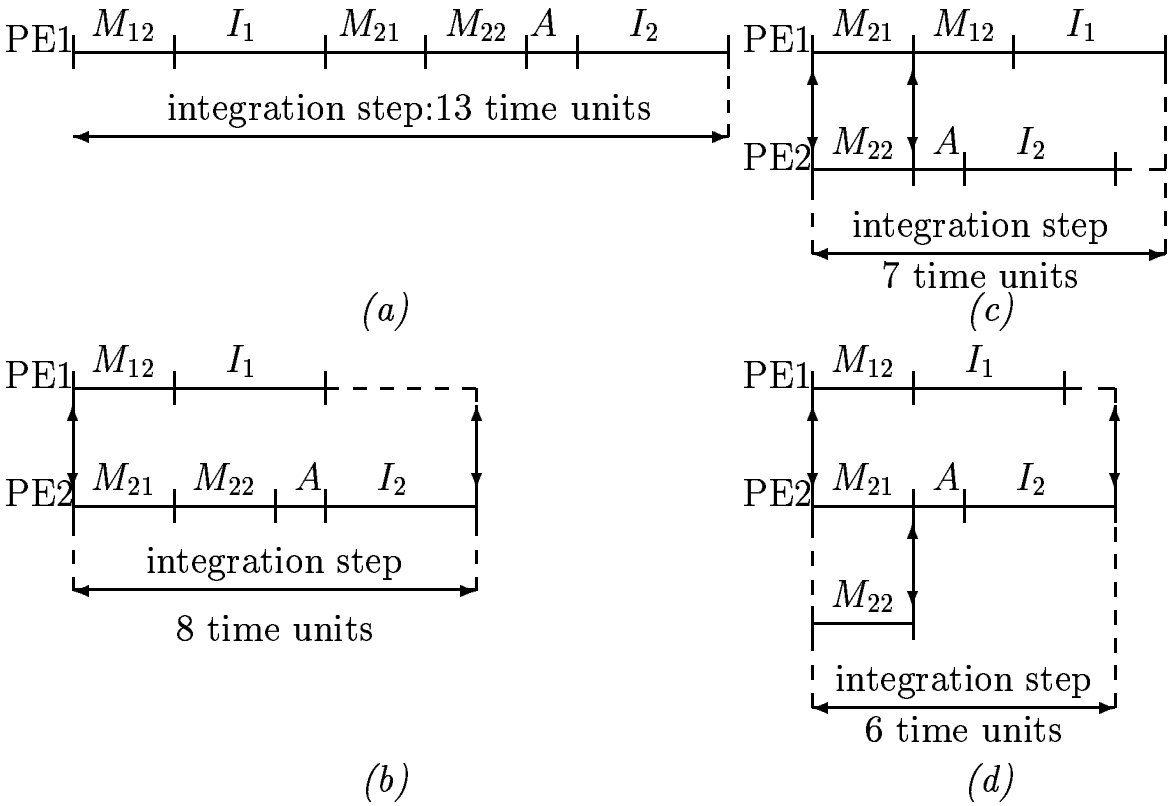


Diagrama timpului pt. $y_1' = a_{12}y_2,\ y_2' = a_{21}y_1 + a_{22}y_2$ unde M =multiplicare, A =adunare I = integrare cu metoda Euler $y_{n+1} = y_n + hf(t_n, y_n))$



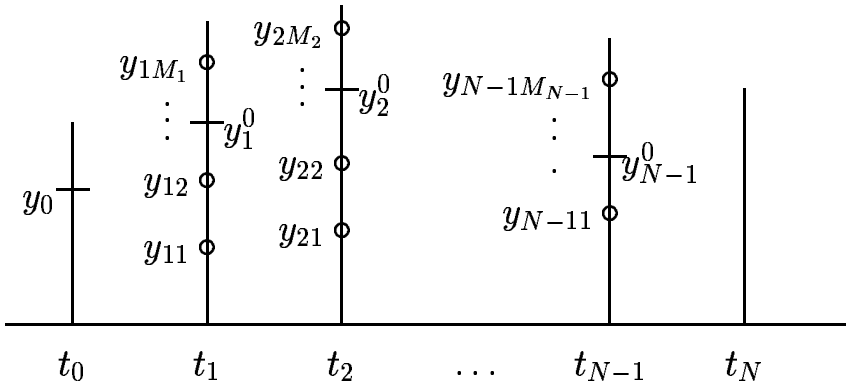
(a) procesare serială (b) procesare paralelă la nivel de sarcină majoră (c) la nivel de sarcină medie (d) la nivel de sarcină minoră

Paralelism în metodă

1. Metoda Nivergelt

Se face o predicţie y_i^0 a soluţiei $y(t_i),\ i = \overline{1, N}$.

Pt. fiecare t_i: sunt alese $y_{ij} \in \mathcal{V}(y_i^0),\ j = \overline{1, M_i}$.



Se rezolvă simultan cu aceeaşi metodă numerică IVPurile:

$$\begin{aligned} y\prime &= f(t,y),\ y(t_0) = y_0,\ a \le t \le t_1, \\ y\prime &= f(t,y),\ y(t_i) = y_{ij},\ t_i \le t \le t_{i+1}, \quad j = \overline{1, M_i},\ i = \overline{1, N-1}. \end{aligned}$$

Aproximaţia finală u_i la $y(t_i)$, $i = \overline{1, N}$:

(a) u_1 este valoarea unică la t_1;

(b) u_i este determinat prin interpolare $Y(t_i, y)$ la $y = u_{i-1}$ utilizând perechile $(y_{i-1\,j}, Y(t_i, y_{i-1\,j}))$, $j = \overline{1, M_{i-1}},\ i = \overline{3, N}$.

Avantaje: viteza alg. par. teroretică este un factor de N; metoda a iscat o discuţie asupra clasei de metode numerice paralele.

Dezavantaje:

(a) este necesar un număr excesiv de procesoare $(1 + \sum_{i=1}^{N-1} M_i)$;

(b) nu este o metodă practică.

. . . . . .

<u>*2. Tehnica de înclinare a frontului de calcul*</u>

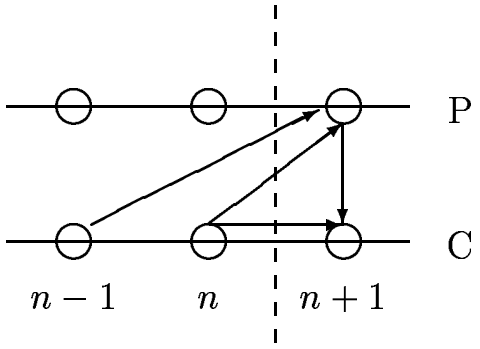
(a) metode *predictor-corector* (PC);
Ex:

- metoda serială a trapezului $y_{n+1} = y_n + \frac{h}{2}(f_n + f_{n+1})$ poate fi aplicată . . . în mod PC serial

$$\begin{aligned} y_{n+1}^P &= y_n^C + \frac{h}{2}\left[3f_n^C - f_{n-1}^C\right], \\ y_{n+1}^C &= y_n^C + \frac{h}{2}\left[f_{n+1}^P + f_n^C\right]. \end{aligned}$$

Secvenţa de calcule: $y_{n+1}^P \to f_{n+1}^P \to y_{n+1}^C \to f_{n+1}^C \to$

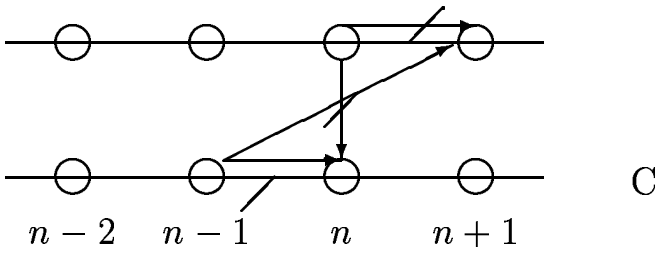
Frontul de calcul a metodei seriale:



... în mod PC paralel

$$\begin{aligned} y_{n+1}^P &= y_{n-1}^C + 2hf_n^P, \\ y_n^C &= y_{n-1}^C + \frac{h}{2}(f_n^P + f_{n-1}^C). \end{aligned}$$

Secvenţa de calcule: $\rightarrow y_{n+1}^P \rightarrow f_{n+1}^P \rightarrow, \rightarrow y_n^C \rightarrow f_n^C \rightarrow$
Frontul de calcul a metodei paralele:

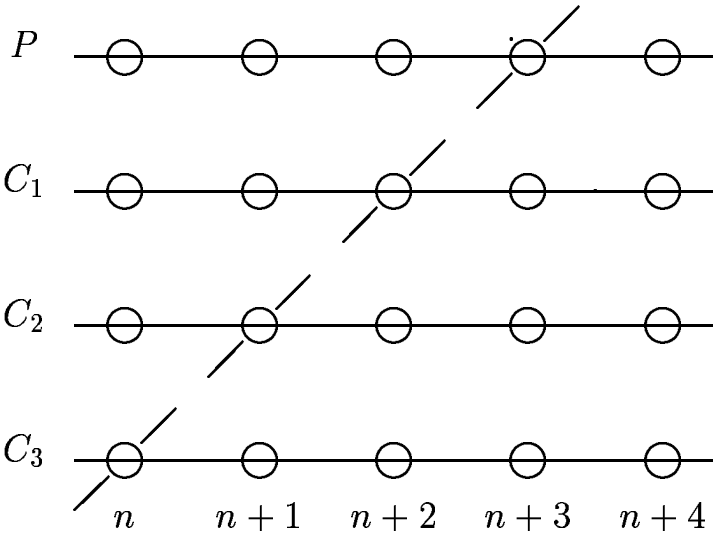


- metode noi pentru un număr date de procesoare:
  (i) metode PC în bloc; de ex:

$$\begin{cases} y_{2n+1}^P = y_{2n-2} + 4hf_{2n}^P, \\ y_{2n+1}^P = y_{2n-2} + \frac{3h}{2}(f_{2n}^P + f_{2n-1}^p) \\ y_{2n} = y_{2n-3} - \frac{h}{2}(3f_{2n}^P - 9f_{2n-1}^P), \\ y_{2n-1} = y_{2n-3} + 2hf_{2n-2}^P \end{cases}$$

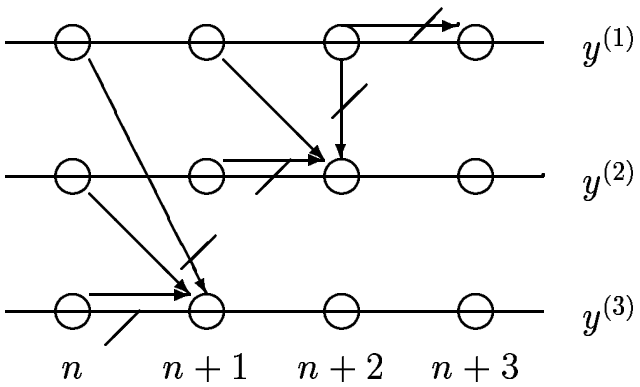
  (ii) corecţii concurente:

$$\begin{aligned} P: &\quad y_{n+4}^{(0)} = y_n^{(3)} + h\left(\tfrac{8}{3}f_{n+3}^{(0)} - \tfrac{4}{3}f_{n+2}^{(1)} + \tfrac{8}{3}f_{n+1}^{(2)}\right), \\ C_1: &\quad y_{n+3}^{(1)} = y_n^{(3)} + h\left(\tfrac{3}{8}f_{n+3}^{(0)} + \tfrac{9}{8}f_{n+2}^{(1)} + \tfrac{9}{8}f_{n+1}^{(2)} + \tfrac{3}{8}f_n^{(3)}\right), \\ C_2: &\quad y_{n+2}^{(2)} = y_n^{(3)} + h\left(\tfrac{1}{3}f_{n+2}^{(1)} + \tfrac{4}{3}f_{n+1}^{(2)} + \tfrac{1}{3}f_n^{(3)}\right), \\ C_3: &\quad y_{n+1}^{(3)} = y_n^{(3)} + h\left(\tfrac{9}{24}f_{n+1}^{(2)} + \tfrac{19}{24}f_n^{(3)} - \tfrac{5}{24}f_{n-1}^{(3)} + \tfrac{1}{24}f_{n-2}^{(3)}\right). \end{aligned}$$



(b) metode *Runge-Kutta*. Ex:

$$
\begin{aligned}
k_{1,n+1} &= hf(x_{n+2}, y^{(1)}_{n+2}),\\
y^{(1)}_{n+3} &= y^{(1)}_{n+2} + k_{1,n+2}\\
k_{2,n+1} &= hf(x_{n+1} + \alpha h, y^{(1)}_{n+1} + \alpha k_{1,n+1}),\\
y^{(2)}_{n+2} &= y^{(1)}_{n+1} + \left(1 - \frac{1}{2\alpha}\right) k_{1,n+1} + \frac{1}{2\alpha} k_{2,n+1}\\
k_{3,n+1} &= hf\left(x_n + \alpha_1 h, y^{(2)}_n + \left(\alpha_1 - \frac{1}{6\alpha}\right) k_{1,n} + \frac{1}{6\alpha} k_{2,n}\right),\\
y^{(3)}_{n+1} &= y^{(3)}_n + \left(\frac{2\alpha_1 - 1}{2\alpha}\right)(k_{1,n} - k_{2,n}) + k_{3,n}
\end{aligned}
$$



(c) metode *bloc*. Ex: o schemă paralela uni-pas implicită

$$
\begin{aligned}
y_{n+1} &= y_n + \frac{h}{720}(251f_n + 646f_{n+1} - 264f_{n+2} + 106f_{n+3} - 19f_{n+4}),\\
y_{n+2} &= y_n + \frac{h}{90}(29f_n + 124f_{n+1} + 24f_{n+2} + 4f_{n+3} - f_{n+4}),\\
y_{n+3} &= y_n + \frac{3h}{80}(9f_n + 34f_{n+1} + 24f_{n+2} + 14f_{n+3} - f_{n+4}),\\
y_{n+4} &= y_n + \frac{2h}{45}(7f_n + 32f_{n+1} + 12f_{n+2} + 32f_{n+3} + 2f_{n+4}),
\end{aligned}
$$

Avantaje: uşor de implementat.

Dezavantaje:

(a) characteristici de stabilitate slabe,
(b) grad redus de acurateţe,
(c) grad redus de paralelism,
(d) viteza alg. paralel depinde într-o anumită măsură de sistemul ODE.

.....

3. Metode de extrapolare

Pp.

(a) $h_1 > h_2 > h_3 > \cdots$,
(b) $n_i = T/h_i$,
(c) o metodă numerică de ordin p care furnizează $T_{i,0} = y_{n_i}$ utilizând h_i,
(d) polinomul de interpolare

$$P(h) = e_0 + e_p h^p + e_{p+1} h^{p+1} + \cdots + e_{p+k-1} h^{p+k-1},$$

$$P(h_i) = T_{i,0},\ i = j-k, \ldots, j$$

Se extrapolează $T_{j,k} = P(0)$.

Tabelul de extrapolare

$$\begin{array}{cccc} T_{j,0} & & & \\ & T_{j-1,1} & & \\ \vdots & \vdots & \ldots & T_{j,k} \\ & T_{j-k+1,1} & & \\ T_{j-k,0} & & & \\ \text{ordin } p & \text{ordin } p+1 & \ldots & \text{ordin } p+k \end{array}$$

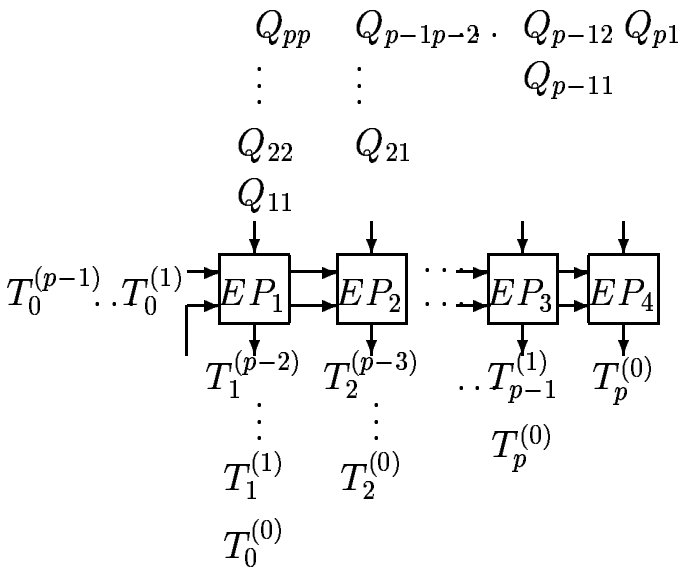
Ex: regula rombului pt. $p = 2$, metoda Euler, $h_i = h_{i-1}/2$, $Q_{ij} = h_j^2/h_{i+1}^2$:

$$T_{-1}^i = 0, \quad T_k^{(i)} = T_{k-1}^{(i+1)} + \frac{T_{k-1}^{(i+1)} - T_{k-1}^{(i)}}{\left(\dfrac{h_i}{h_{i+k}}\right)^p \left(1 - \dfrac{T_{k-1}^{(i+1)} - T_{k-1}^{(i)}}{T_{k-1}^{(i+1)} - T_{k-2}^{(i+1)}}\right) - 1}.$$

Avantaje: paralelism intrinsec, stabilitate înaltă şi proprietăţi bune de acurateţe.

Dezavantaje: structură specială pt. reţea de interconectare procesoare.

Stuctura unei matrice sistolice pt. generarea tabloului de extrapolare:



. . . . . .

4. Tehnica digrafului pentru metode Runge-Kutta

Metodă Runge-Kutta:

$$\begin{aligned} y_{n+1} &= y_n + h\sum_{l=1}^{m} b_l k_l, \\ k_i &= f(t_n + hc_i, y_n + h\sum_{l=1}^{n} a_{il}k_l),\ i = \overline{1,m} \end{aligned}$$

Tabloul Butcher:

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1m} \\ \vdots & \vdots & & \vdots \\ c_m & a_{m1} & \dots & a_{mm} \\ \hline & b_1 & \dots & b_m \end{array}$$

Digraful (graf direct) $G = (V, E)$ a matricei A:

(a) $V = \{1, \dots, m\}$

(b) $(i, j) \in E$ if $i, j \in V$ şi $a_{ij} \neq 0$.

Metodă de tip I:

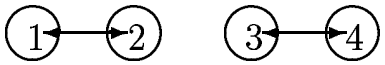
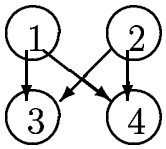
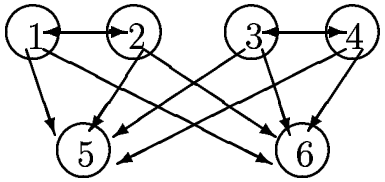
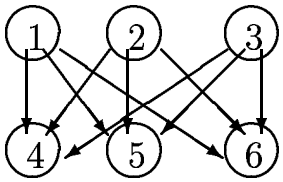
$$\frac{\begin{array}{c|c} c_1 & A_1 \\ \hline \rho_1 & b_1^T \end{array}}{} + \dots + \frac{\begin{array}{c|c} c_\nu & A_\nu \\ \hline \rho_\nu & b_\nu^T \end{array}}{}$$

$$y_{n+1} = \rho_1 y_{n+1}^{(1)} + \dots + \rho_\nu y_{n+1}^{(\nu)}.$$

1. Vârfurile $v_1, \dots, v_s$, unde $s \geq 2$ formează un s-ciclu dacă $v_i \neq v_{i+1},\ i = \overline{1,s}$: $\{v_i, v_{i+1}; i = \overline{1,s}\} \subseteq E$ or $\{(v_{i+1}, v_i) : i = \overline{1,s}\} \subseteq E$, where $v_{s+1} = v_1$.
2. Fie σ lungimea celul mai lung ciclu din digraf.
3. *partiţionare pe procesoare*: se pp. că mulţimea V este partţionată în ν submulţimi disjuncte nevide: $\{1, \dots, m\} = \cup_{i=1}^{\nu} \mathcal{P}_i,\ \mathcal{P}_1, \dots, \mathcal{P}_\nu \neq \emptyset,\ \mathcal{P}_i \cap \mathcal{P}_j,\ \forall i \neq j$

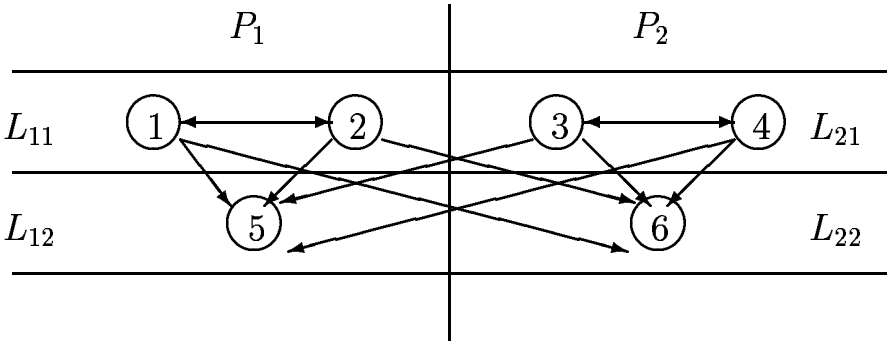
4. *partiţionare pe nivele*: fiecare $\mathcal{P}_l$, $\mathcal{P}_l = \cup_{i=1}^{\mu} \mathcal{L}_{li}$, $i = \overline{1,\mu}$ este partiţionat, unde $\mathcal{L}_{li} \cap \mathcal{L}_{lj} = \emptyset$ pt. toţi $i \neq j$ ($\mathcal{L}_{li}$ paote fi$\emptyset$, dar pt. anumiţi $r \in \{1,\ldots,\nu\}$, $L_{rj} \neq \emptyset$, $j = \overline{1,n}$ pt. anumit $\mu \geq 1$). Aceasta este o partiţionare pe nivele dacă are loc următoarea condiţie:
   (a) pt. fiecare $l = \overline{1,\nu}$ şi $j = \overline{1,\mu}$, toate vârfurile din $\mathcal{L}_{lj}$ aparţin unui ciclu.
   (b) dacă $a_{ij} \neq 0$ atunci fie că exist;ă $p \in \{1,\ldots,\mu\}$ şi $r,s \in \{1,\ldots,\mu\}$ such that $r \leq s$, $i \in \mathcal{L}_{ps}$, $j \in \mathcal{L}_{pr}$ sau există $q,t \in \{1,\ldots,\nu\}$ and $r,s \in \{1,\ldots,\mu\}$ a.î. $q \neq t$, $r < s$, $i \in \mathcal{L}_{ts}$, $j \in \mathcal{L}_{qr}$.

Ex:

Method	RK matrix	Digraph
(a)	xxoo xxoo ooxx ooxx	
(b)	xooo oxoo xxxo xxox	
(c)	xxoooo xxoooo ooxxoo ooxxoo xxxxxo xxxxox	
(d)	xooooo oxoooo ooxooo xxxxoo xxxoxo xxxoox	

O combinaţie de partiţionare pe procesoare şi nivele a unei metode σ-implicite este numită $\{\nu,\mu,\sigma\}$ *implementare.*

Ex: o implementare $\{2,2,2\}$ a metodei (c)



Avantaje:

(a) costul calculelor a unei implementări seriale poate fi redus.

(b) se prezervă proprietăţile de stabilitate şi acurateţe ale metodei seriale.

Dezavantaje: grad scăzut de paralelism..

METODE PARALELE PENTRU ECUAŢII DIFERENŢIALE ORDINARE - PARTEA II

Cum? Utilizând metode multivalori şi tehnici paralele bazate pe

- paralelism în sistem;
- parallelism în metodă;
- parallelism în timp.

Paralelism în metodă

1. *Metoda Nivergelt*
2. *Tehnica de înclinare a frontului de calcul*
3. *Metode de extrapolare*
4. *Tehnica digrafului pentru metode Runge-Kutta*

. . . . . .

5. Transformarea ecuaţiilor etapelor din scheme Runge-Kutta

Metoda RK:

$$\begin{aligned} y_{n+1} &= y_n + h(b^T \otimes I)F(t_n, Y), \\ Y &= e \otimes y_n + h(A \otimes I)F(t_n, Y), \end{aligned}$$

unde
$Y = (Y_1, \ldots, Y_q)^T \in \mathbb{R}^{qn}$,
$F(t, Y) = (f(t + c_1 h, Y_1), \ldots, f(t + c_q h, Y_q))^T$
Transformare:

$$\begin{aligned} y_{n+1} &= y_n + d^T Z, \\ Z &= h(A \otimes I)F(t_n, e \otimes y_n + Z), \end{aligned}$$

unde $Z = Y - e \otimes y_n$ şi $d^T = b^T A^{-1}$.
Se utilizează iteraţiile Newton simplificate:

$$\begin{aligned} Z^{(k+1)} &= Z^{(k)} + \Delta Z^{(k)}, \quad k = 0, 1, \ldots \\ (I - hA \otimes J)\Delta Z^{(k)} &= -Z^{(k)} + h(A \otimes I)F(t_n, e \otimes y_n + Z^{(k)}) \end{aligned}$$

cu $Z^{(0)} = e \otimes f(t_n, y_n)$.
Dacă $\Lambda = T^{-1}A^{-1}T$ şi $W^{(k)} = (T^{-1} \otimes I)Z^{(k)}$,

$$\begin{aligned} W^{(k+1)} &= W^{(k)} + \Delta W^{(k)}, \quad k = 0, 1, \ldots \\ (h^{-1}\Lambda \otimes I - I \otimes J)\Delta W^{(k)} &= -h^{-1}(\Lambda \otimes I)W^{(k)} + \\ &\quad + (T^{-1} \otimes I)F(t_n, e \otimes y_n + (T \otimes I)W^{(k)}) \end{aligned}$$

Caz 1: Λ este o matrice diagonală: $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_q)$.
q sisteme liniare de dimensiune n:

$$\begin{aligned} w_i^{(k+1)} &= w_i^{(k)} + \Delta w_i^{(k)}, \quad k = 0, 1, \ldots \\ (h^{-1}\lambda_i I - J)\Delta w_i^{(k)} &= -h^{-1}\lambda_i w_i^{(k)} + G_i^{(k)}, \end{aligned}$$

unde $i = 1, 2, \ldots, q$, $(w_1^{(k)}, \ldots, w_q^{(k)})^T = W^{(k)}$,
$G^{(k)} = (T^{-1} \otimes I) F(t_n, e \otimes y_n + (T \otimes I) W^{(k)})$.
Ex:

- metode implicit-diagonale DIRK (A inferior triunghiular) , $\lambda_1 = \ldots = \lambda_q$

$$\begin{array}{c|cc} c_1 & c_1 & 0 \\ \frac{1}{2} + \frac{1}{12}\left(\frac{1}{2} - c_1\right)^{-1} & \frac{1}{6}\left(\frac{1}{2} - c_1\right)^{-1} & \left(\frac{1}{6} - \frac{1}{2}c_1\right)\left(\frac{1}{2} - c_1\right)^{-1} \\ \hline & \frac{1}{12}\left(c_1^2 - c_1 + \frac{1}{3}\right)^{-1} & \left(\frac{1}{2} - c_1\right)^2\left(c_1^2 - c_1 + \frac{1}{3}\right) \end{array}$$

  cu $y(t_n) - y_n = \mathcal{O}(h^3)$.
- metode multi-implicit-diagonale MIRK ($\lambda_i \neq \lambda_j$, for $i \neq j$),

$$\begin{array}{c|cc} \tau_1 & \tau_1\left(\tau_2 - \frac{1}{2}\tau_1\right)/(\tau_2 - \tau_1) & -\frac{1}{2}\tau_1^2/(\tau_2 - \tau_1) \\ \tau_2 & \frac{1}{2}\tau_2^2/(\tau_2 - \tau_1) & \tau_2\left(\frac{1}{2}\tau_2 - \tau_1\right)/(\tau_2 - \tau_1) \\ \hline & \left(\tau_2 - \frac{1}{2}\right)/(\tau_2 - \tau_1) & \left(\frac{1}{2}\tau_2 - \tau_1\right)/(\tau_2 - \tau_1) \end{array}$$

  unde $\tau_1 = \lambda_1 + \lambda_2 - \sqrt{\lambda_1^2 + \lambda_2^2}$, $\tau_2 = \lambda_1 + \lambda_2 + \sqrt{\lambda_1^2 + \lambda_2^2}$, $\lambda_2 = (\lambda_1/2 - 1/6)/(\lambda_1 - 1/2)$, $\lambda_1 > 0$, $y(t_n) - y_n = \mathcal{O}(h^3)$.

Caz 2: Λ este o matrice bloc-diagonală. Pp. q este par. Se vor rezolva $q/2$ sisteme liniare de dimensiune $2n$:

$$\begin{aligned} \begin{pmatrix} w_{2i}^{(k+1)} \\ w_{2i+1}^{(k+1)} \end{pmatrix} &= \begin{pmatrix} w_{2i}^{(k)} \\ w_{2i+1}^{(k)} \end{pmatrix} + \begin{pmatrix} \Delta w_{2i}^{(k)} \\ \Delta w_{2i+1}^{(k)} \end{pmatrix}, \qquad k = 0, 1, \ldots \\ \begin{pmatrix} h^{-1}\mathrm{Re}\,\lambda_i I - J & -h^{-1}\mathrm{Im}\,\lambda_i I \\ h^{-1}\mathrm{Im}\,\lambda_i I & h^{-1}\mathrm{Re}\,\lambda_i I - J \end{pmatrix} \begin{pmatrix} \Delta w_{2i}^{(k)} \\ \Delta w_{2i+1}^{(k)} \end{pmatrix} &= \\ = -h^{-1} \begin{pmatrix} \mathrm{Re}\,\lambda_i I & -\mathrm{Im}\,\lambda_i I \\ \mathrm{Im}\,\lambda_i I & \mathrm{Re}\,\lambda_i I \end{pmatrix} \begin{pmatrix} w_{2i}^{(k)} \\ w_{2i+1}^{(k)} \end{pmatrix} &+ \begin{pmatrix} G_{2i}^{(k)} \\ G_{2i+1}^{(k)} \end{pmatrix}, \end{aligned}$$

unde $i = 1, \ldots, q/2$, $(w_1^{(k)}, \ldots, w_q^{(k)})^T = W^{(k)}$,
$G^{(k)} = (T^{-1} \otimes I) F(t_n, e \otimes y_n + (T \otimes I) W^{(k)})$.
Ex.: metode Gauss, Radau IA, Radau IIA şi Lobatto IIIC cu q par, şi ordin $2q$, $2q-1$, $2q-1$ şi $2q-2$. Pt. Lobatto IIIC cu $q = 4$ etape

$$\begin{array}{c|cccc} 0 & 1/12 & -\sqrt{5}/12 & \sqrt{5}/12 & -1/2 \\ (5-\sqrt{5})/10 & 1/12 & 1/4 & (10-7\sqrt{5})/60 & \sqrt{5}/60 \\ (5+\sqrt{5})/10 & 1/12 & (10+7\sqrt{5})/60 & 1/4 & -\sqrt{5}/60 \\ 1 & 1/12 & 5/12 & 5/12 & 1/12 \\ \hline & 1/12 & 5/12 & 5/12 & 1/12 \end{array}$$

matricea Λ este

$$\Lambda = \begin{pmatrix} 2.22098 & -4.16039 & 0 & 0 \\ 4.16039 & 2.22098 & 0 & 0 \\ 0 & 0 & 3.77902 & -1.38018 \\ 0 & 0 & 1.38018 & 3.77902 \end{pmatrix}$$

Avantaje: stabilitate şi acurateţe specifice metodelor RK.

Dezavantaje:

(a) tehnica nu poate fi aplicată la orice schemă;

(b) grad mic de paralelism.

.....

6. Metode PIRK

PIRK = metodă diagonal-iterativă tip Runfge-Kutta paralelă.

metoda RK:

$$Y - hAf(et_n + ch, Y) = ey_n + haf(t_n, y_n)$$
$$y_{n+1} = y_n = hb_0 f(t_n, y_n) + hb^T f(et_n + ch, Y)$$

Proces iterativ:

$$\begin{aligned} Y^{(1)} - hDf(et_n + ch), Y^{(1)}) &= ey_n + ahf(t_n, y_n) + \\ &+hAf(et_n + c^*h, Y^{(0)}) - hDf(et_n + c^*h, Y^{(0)}) \\ Y^{(j+1)} - hDf(et_n + ch), Y^{(j+1)}) &= ey_n + ahf(t_n, y_n) + \\ &+hAf(et_n + ch, Y^{(j)}) - hDf(et_n + ch, Y^{(j)}), \end{aligned}$$

unde

$j = 0, \ldots, m-1$,

c^* depind de formula predictor $Y^{(0)}$,

D =diag(D), şi

$y_{n+1} = y_n + hb_0 f(t_n, y_n) + hb^T f(et_n + ch, Y^{(m)})$

$Y^{(0)} = ey_n + hEf(et_n, ey_n) + hBf(et_n + c^*h, Y^{(0)})$

(de exemplu $E = 0,\ B = D$)

D este ales astfel înc at:

1. $D = 0$: este metodă iterativă explicită pentru probleme nerigide;
2. condiţii de stabilitate înalte pentru a prescrie numărul de iteraţii m;
3. D =diag(A): iteraţiide tip Jacobi neliniare cu o convergenţă slabă;
4. min$\sigma(A - D)$ pentru reducerea erorii în componentele nerigide;
5. termen de eroare cel mai mic din eroarea de trunchiere este minimizat;
6. varianta funcţiei de stabilitate: convergenţă rapidă a funcţiei de stabilitate a metodei iterative la metoda RK de bază (min$\sigma(I - D^{-1}A)$);

Avantaje: înalt grad de paralelism în exploatarea metodelor RK.

Dezavantaj: metoda este echivalentă cu una DIRK (adică se impun anumite restricţii asupra acurateţii metodei)

. . . . . .

7. Metode DIMSIM

Metode general liniare:

$$\begin{aligned} Y_i &= h\sum_{j=1}^{s} c_{ij}^{11} f(Y_j) + \sum_{j=1}^{r} c_{ij}^{12} y_j^{[n-1]}, \quad i = \overline{1,s} \\ y_i^{[n]} &= h\sum_{j=1}^{s} c_{ij}^{21} f(Y_j) + \sum_{j=1}^{r} c_{ij}^{22} y_j^{[n-1]}, \quad i = \overline{1,r} \end{aligned}$$

reprezentare prin matricea

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

DIMSIM (metoda de integrare multi-etape diagonal implicite): C_{11} inferior triunghiular.

	C_{11}	rigid vs. nerigid	paralel vs. serial
Tip1	strict inferior triunghiular	nerigid	serial
Tip 2	diag(C_{11})=λI	rigid	serial
Tip 3	$C_{11} = 0$	rigid	paralel
Tip 4	C_{11} =diag(C_{11})=λI	rigid	paralel

Avantaje: potenţial intern de paralelism pt. o clasă mare de metode.

Dezavantaje: număr mic de articole referitoare a proprietătile unor asemenea metode.

. . . . . .

8. Integrarea unor sisteme ODE speciale

Fie sistemul ODE liniar:

$$y'(t) = A(t)y(t) + b(t)$$

(a) utilizând o metodă unipas (ex: schemă RK) se obţine o relaţie recurentă de forma:

$$y_{n+1} = R_{n+1}y_n + b_n, \; n = \overline{0, N-1}$$

(b) dacă $Q(t) \equiv Q$ se poate utiliza tehnica de descompunere (vezi metoda WR):

$$\frac{d}{dt}y^{(k+1)} + My^{(k+1)}(t) = Ny^{(k)}(t) + b(t)$$

unde $N - M = Q$.

(c) utilizând o metodă multipas liniară se obţine un sistem liniar care se poate rezolva prin metode paralele.

Avantaje: grad înalt de paralelism.
Dezavantaje: paralelim nespecific pentru ODEuri.

Paralelism în timp

Ex: Metoda relaxării în valuri (WR)

Metoda Picard: se dă $y^{(0)}(t) : [t_0, t_0 + T]$ o soluţie aproximativă, şi se caută $y^{(1)}(t),\ y^{(2)}(t), \ldots$ a.î.

$$\frac{d}{dt}y^{(k+1)}(t) = f(t, y^{(k)}(t),\ y^{(k+1)}(t_0) = y_0.$$

Metode general continue în timp:

$$\begin{aligned}
\frac{d}{dt}z^{(k+1)}(t) &= G(t, z^{(k+1)}(t), y^{(k+1)}(t), y^{(k)}(t)),\ z^{(k+1)}(t_0) = y_0, \\
y^{(0)}(t_0) &= y_0, \\
y^{(k+1)}(t) &= g(t, z^{(k+1)}(t), y^{(k)}(t)),
\end{aligned}$$

unde $G(t, y, y, y) = f(t, y),\ g(t, y, y) = f(t, y)$ (splitting functions) decouple the ODE system into independent subsystems.

Ex.:

- WR-Jacobi

$$\frac{d}{dt}y_i^{(k+1)} = f_i(t, y_1^{(k)}, \ldots, y_{i-1}^{(k)}, y_i^{(k+1)}, y_{i+1}^{(k)}, \ldots, y_n^{(k)}),\ i = \overline{1, n}$$

- WR-Gauss-Seidel
- WR-SOR
- WR-Newton

$$\frac{d}{dt}y^{(k+1)} = f(t, y^{(k)}) + f_y(t, y^{(k)})(y^{(k+1)} - y^{(k)})$$

Metode de integrare: metode RK continue. Ex: se utilizează metoda Heun

$$\begin{aligned}
Y_{n+1}^1 &= y_n \\
Y_{n+1}^2 &= y_n + h_{n+1}f(t_n, Y_{n+1}^1) \\
y_{n+1} &= y_n + \frac{1}{2}h_{n+1}[f(t_n, Y_{n+1}^1 + f(t_{n+1}, Y_{n+1}^2)]
\end{aligned}$$

şi interpolarea liniară

$$y_{n+\theta} = y_n + \frac{1}{2}h_{n+1}[f(t_n, Y_{n+1}^1 + f(t_{n+1}, Y_{n+1}^2)]$$

se obţine metoda WR-Jacobi-Heun.

Avantaje: grad înalt de paralelism.

Dezavantaje: convergenţă înceată la soluţia exactă (grad mic de acurateţe).

METODE PARALELE CU DIFERENŢE PENTRU ECUAŢII CU DERIVATE PARŢIALE

0. Problema de test:

Ecuaţia Poisson pe domeniul $\Omega \subset \mathbb{R}^2$

$$\forall (x,y) \in \Omega : -\Delta u = f(x,y).$$

Δ=operator pt. fcţ. de 2 ori dif. $u(x,y)$

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}.$$

Pp. $\Omega = (a,b) \times (c,d)$.

Cond.frontieră: $\begin{cases} \text{Dirichlet: } u(a,y) = g(y),\ \forall y \in (c,d) \\ \text{Neumann: } \frac{\partial u}{\partial y}(c,d) = g(x),\ \forall x \in (a,b),\ \frac{\partial y}{\partial x}(b,y) = g(y),\ \forall y \in (c,d). \\ \text{periodice: } u(a,y) = u(b,y),\ \forall y \in (c,d) \text{ express periodicity in } x. \\ \text{combinare condiţii} \end{cases}$

Problema clasică: $\begin{cases} -\Delta u = f(x,y), & \forall (x,y) \in \Omega \\ u(x,y) = g(x,y), & \forall (x,y) \in \partial\Omega. \end{cases}$

discretizată pe grila $M \times N$ cu pct. $(x_i, y_j),\ \forall (i,j) \in \{0,\dots,M\} \times \{0,\dots,N\}$.

$$\Delta u(x,y) = \frac{u(x-h,y) + u(x+h,y) + u(x,y-h) + u(x,y+h) - 4u(x,y)}{h^2} + \mathcal{O}(h^2),$$

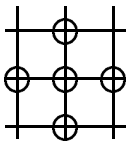
Fie $u(x_i,y_j) \approx u_{ij}$ şi $f_{ij} = f(x_i,y_j)$

Formularea problemei Poisson discrete

$$(4u_{ij} - u_{i-1\,j} - u_{i+1\,j} - u_{i\,j-1} - u_{i\,j+1})/h^2 = f_{ij}.$$

($(M-1)(N-1)$ ecuaţii, valorile u_{ij} din interior grilă sunt necunoscute),

Discretizarea de mai sus - cu 5 puncte:



Rescriere: $U_{(j-1)(M-1)+i-1} = u_{ij}$.

$$(4U_l - U_{l-1} - U_{l+1} - U_{l-M+1} - U_{l+M-1})/h^2 = f_{ij},$$

cu $l = (j-1)(M-1) + i - 1$. Fie $b_{(j-1)(M-1)+i-1} = f_{ij}$+termeni frontieră.

Atunci

$$AU = f - Bg$$

formulare matriceală problema Poisson discretă.

Obs: metode de rezolvare ţin seama că A este rară.

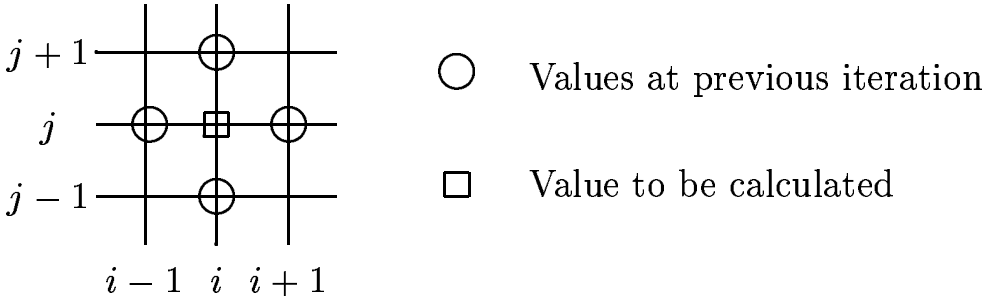
1. Metoda Jacobi punctuală

Metoda de relaxare Jacobi:

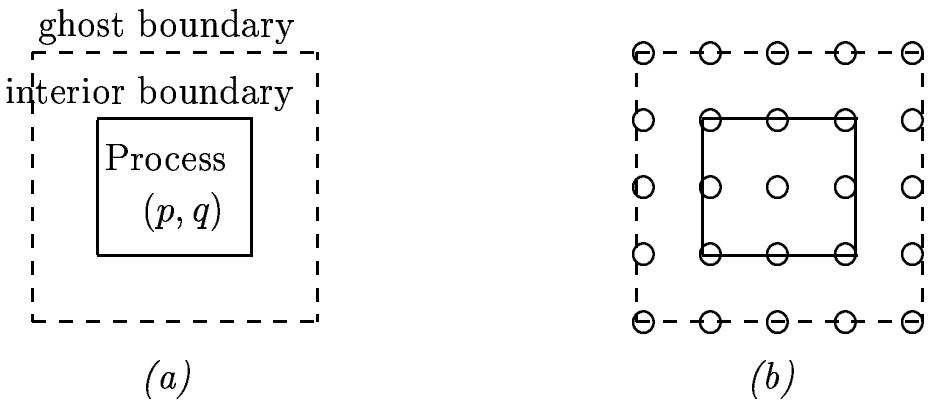
$$u_{ij}^{(p+1)} = \frac{1}{4}(u_{i-1\,j}^{(p)} + u_{i+1\,j}^{(p)} + u_{i\,j-1}^{(p)} + u_{i\,j+1}^{(p)} - q_{ij}).$$

Stop: $|u_{ij}^{(p+1)} - u_{ij}^{(p)}| < \varepsilon \ \forall i, j$

Obs: calculele se pot efectua în paralel!

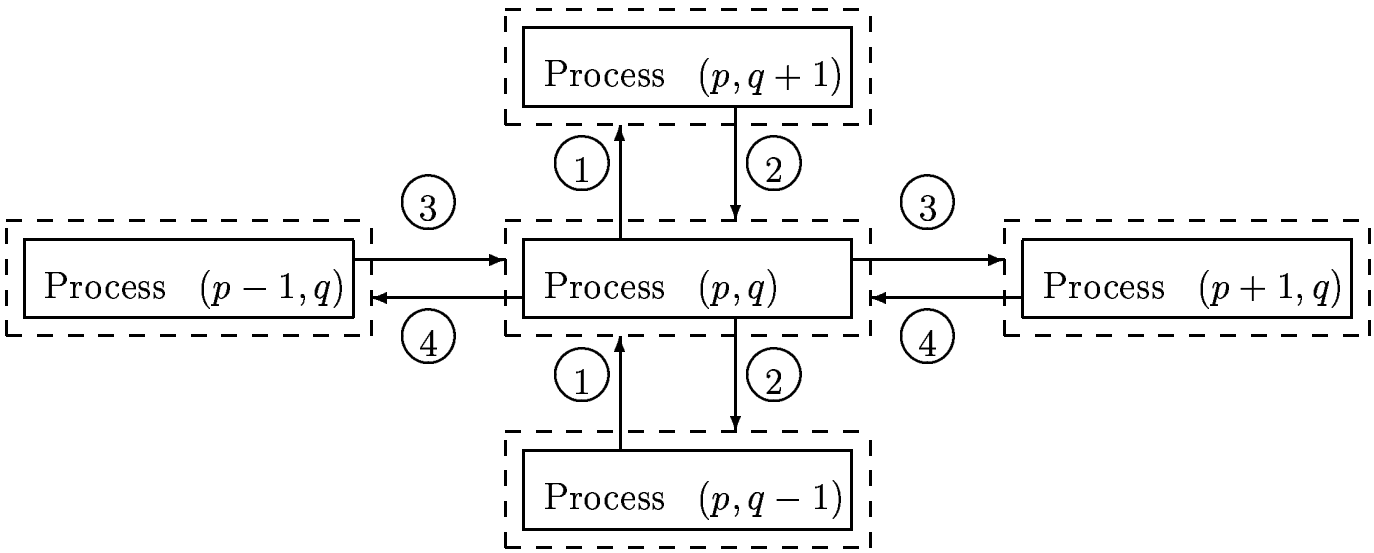


Variantă: mai multe pct. sunt alocate unui procesor.



Fiecare pas are 2 faze: relaxare locală şi schimb la frontiera-fantomă.

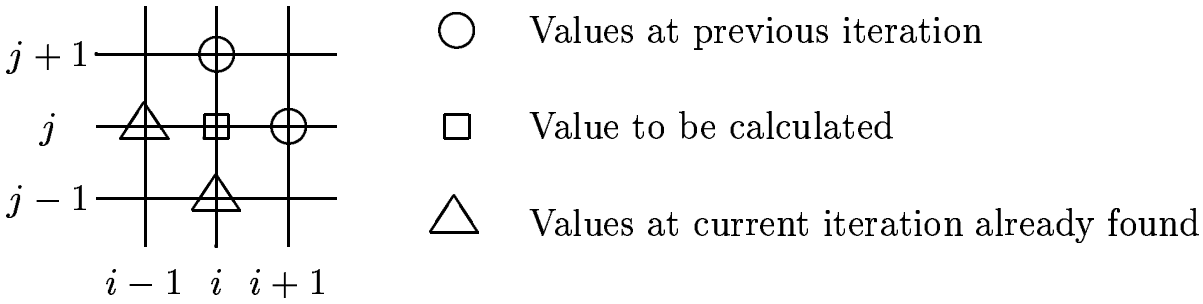
Schimbul la frontiere se face în etape:



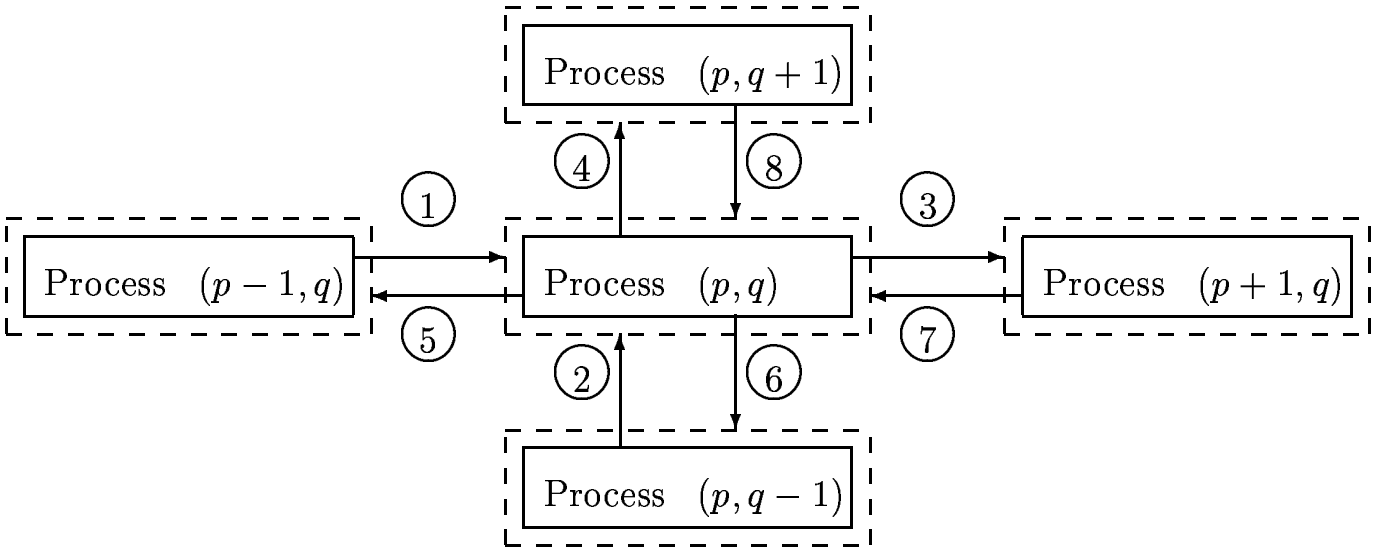
2. Metoda punctuală Gauss-Seidel

Iteraţii:

$$u_{ij}^{(p+1)} = \frac{1}{4}(u_{i\,j-1}^{(p+1)} + u_{i-1\,j}^{(p+1)} + u_{i\,j+1}^{(p)} + u_{i+1\,j}^{(p)} - q_{ij}).$$



Schimbul la frontiere se face astfel:



Factor de convergenţă:

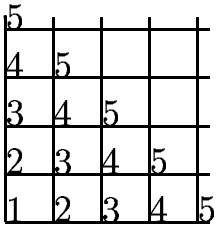
$$\rho_J = 1 - \frac{h^2}{2} + \mathcal{O}(h^4), \quad \rho_{GS} = 1 - h^2 + \mathcal{O}(h^4)$$

cu h interspaţii grilă

Obs:

1. Proces (0,0) activ, celelalte aşteaptă; apoi procesele (1,0) şi (0,1) devin active etc. (în valuri).
2. Odată ce primul val lucrează la iteraţia k propagates, procesele anterioare pot lucra la iteraţia $k+1$.

.



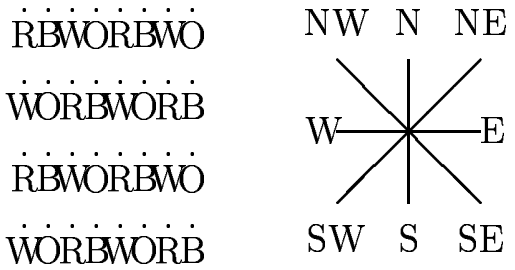
3. Metoda tablei de şah (ordonare roşu-negru sau alb-negru)

Idee:

1. metoda GS în loc de valori numai pe 2 nivele
2. pct. negre actaulizate primele, celelalte după.

$$
\begin{array}{ccccc}
 & B_{11} & R_9 & B_{12} & \\
B_8 & R_7 & B_9 & R_8 & B_{10} \\
R_4 & B_6 & R_5 & B_7 & R_6 \\
B_3 & R_2 & B_4 & B_7 & B_5 \\
 & B_1 & R_1 & B_2 &
\end{array}
$$

Variantă: multicolorare pt. alte tipuri de aprox. derivate; ex: ordonare pe 3 culori pt. scheme cu 9-puncte



(a) *(b)*

Alte metode: metode cu grile adaptive, grile rare, metoda SOR (supra-relaxare GS cu ordonare roşu-negru etc.

.....

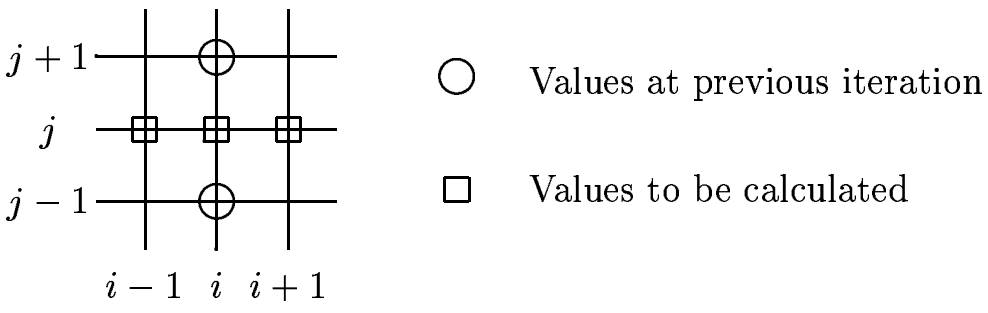
4. Metode de relaxare în linie

Idee: o întreagă linie de ct. este relaxată simultan; ex: metoda Jacobi

$$u_{i-1\,j}^{(p+1)} - 4u_{ij}^{(p+1)} + u_{i+1\,j}^{(p+1)} = q_{ij} - u_{i\,j+1}^{(p)} - u_{i\,j-1}^{(p)}.$$

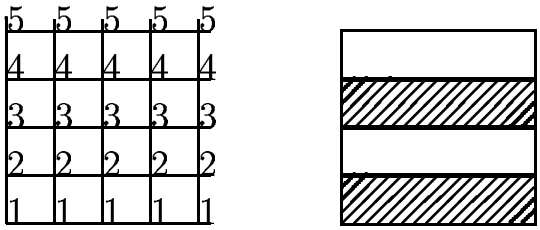
(la un pas, linia j în direcţia x este relaxată)

Obs: un sisttem de ec. liniare cu cu matrice tridiagonală $T = \text{tridiag}\,(-1, 4, -1)$ trebuie rezolvată la fiecare pas (ex: în paralel cu metoda de reducere ciclică par-impar)



Metodele Gauss-Seidel şi SOR: similare.
Metoda pe 2 nivele → metoda ,,zebrei"

.



. . . . . .

5. Metode bloc iterative

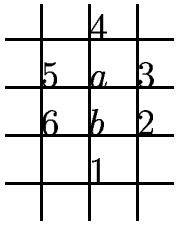
Metoda GS punctuală:

$$x_i = \frac{b_i - \sum_{j \neq i} a_{ij} x_j}{a_{ii}}, \quad i = \overline{1,n}$$

Metoda GS în bloc:

$$\sum_{j \in \text{block}_l} a_{ij} x_j = b_i - \sum_{j \notin \text{block}_l} a_{ij} x_j, \qquad i \in \text{block}_l,\ l \geq 1.$$

Caz particular: relaxarea zebra pt. blocuri egale cu linii
Ex:



Din

$$u_a = \frac{1}{4}(u_b + u_3 + u_4 + u_5 - q_a),$$
$$u_b = \frac{1}{4}(u_1 + u_2 + u_a + u_6 - q_b),$$

se obţine

$$u_a = \frac{1}{15}[4(u_3 + u_4 + u_5 - q_a) + u_1 + u_2 + u_6 - q_b],$$
$$u_b = \frac{1}{15}[4(u_1 + u_2 + u_6 - q_b) + u_3 + u_4 + u_5 - q_a].$$

Astfel se obţin ec. indepenedente care pot fi evaluate simultan

$$u_{ij} = \frac{1}{15}[4(u_{i-1\,j} + u_{i\,j+1} + u_{i\,j-1} - q_{i\,j}) + u_{i+1\,j+1} + u_{i+2\,j} + u_{i+1\,j-1} - q_{i+1\,j}],$$

$$u_{i+1\,j} = \frac{1}{15}[4(u_{i+1\,j+1} + u_{i+2\,j} + u_{i+1\,j-1} - q_{i+1\,j}) + u_{i-1\,j} + u_{i\,j+1} + u_{i\,j-1} - q_{i\,j}].$$

.....

6. Metoda direcţiilor alternante

Metoda ADI: $A = H + V$ şi procesul iterativ

$$(a) \qquad (H + \alpha_k I)x^{(k+1/2)} = (\alpha_k I - V)x^{(k)} + b,$$

$$(b) \qquad (V + \alpha_k I)x^{(k+1)} = (\alpha_k I - H)x^{(k+1/2)} + b,$$

Obs:

1. (a) consistă în soluţ. a N sisteme tridiagonale de mărime N coresp. linii orizontale,
2. (b) tot N sisteme tridiagonale coresp. liniilor verticale
3. sistemele pot fi rezolvate în paralel.

.....

7. Probleme complexe

Ex:

$$au_{xx} + bu_{yy} + cu_{zz} = f, \quad 0 \leq x, y, x \leq 1$$

cu a, b, c funcţii x, y şi z.

Metoda cu diferenţe cu 7-puncte şi $h = \Delta x = \Delta y = \Delta z$

$$2(a_{ijk} + b_{ijk} + c_{ijk})u_{ijk} - a_{ijk}(u_{i+1\,jk} + u_{i-1\,jk}) -$$

$$-b_{ijk}(u_{i\,j+1\,k} + u_{i\,j-1\,k}) - c_{ijk}(u_{uj\,k+1} + u_{ij\,k-1}) = h^2 f_{ijk}.$$

METODELE MULTIGRILĂ, ELEMENTULUI FINIT, DESCOMPUNEREA DOMENIULUI, PARALELE PENTRU ECUAŢII CU DERIVATE PARŢIALE

Problema Poisson discretizată: $?U: \ LU = F$

---

1. Metode multigrile

Idee:

1. Fie $G^i, \ i = 1, \ldots, m$ o secvenţă de grile ce ocoepră domeniul de interes a.î. pasul grilei G^{i-1} este $2h_i$ unde h_i este pasul lui G^i
2. G^m este grila pe care se cere soluţia, G^1 este grila cea mai rară.
3. Soluţia aprox. u^m, a ec. $L^h U^h = F^h$ este obţinută prin ietarţii ale unei metode de relaxare pe G^m.
4. Pr. $i = m, \ldots, 2$
   (a) Restul $F^h + L^h u^i = f^i$ este transferat pe grila G^{i-1} printr-un proces numit injecţie;
   (b) O corecţie a sol. u^i este calculată pe grila $G^{(i-1)}$.
   (c) Soluţia pe grila G^i va fi corectată utilizând informaţie interpolată din grilele G^{i-1} cu $i = \overline{2, m}$.

Ex. 1: problema Poisson uni-dimensională:

$$-\frac{\mathrm{d}^2 u}{\mathrm{d}\, x^2} = f(x), \ \forall x \in (0, \pi), \quad \begin{cases} u(0) = g_0, \\ u(\pi) = g_\pi \end{cases},$$

discretizată pe grilă cu pas $h = \pi / M$

$$-u_i + 2u_i - u_{i+1} = h^2 f_i, \quad i = \overline{1, M-1}, \qquad \begin{cases} u_0 = g_0 \\ u_M = g_\pi. \end{cases}$$

în formularea matriceală

$$A_h u_h = b_h,$$

cu $A_h = \text{tridiag}\,(-1/h^2, 2/h^2, -1/h^2)$, şi

$$b_h = \begin{pmatrix} f_1 + (u_0/h^2) \\ f_2 \\ \vdots \\ f_{M-2} \\ f_{M-1} + (u_M/h^2) \end{pmatrix}.$$

Metoda JOR:

$$u_m^{(t+1)} = \frac{\omega}{2}(f_m h^2 + u_{m+1}^{(t)} + u_{m-1}^{(t)}) + (1-\omega) u_m^{(t)}, \quad m = \overline{1, M-1}.$$

corespunzătoare $G_h - H_h = A_h$, şi convergenţa det. de $R_h = G_h^{-1} H_h$. Eroarea

$$v_h^{(t+1)} = R_h v_h^{(t)}, \quad v_h^{(t)} = u_h^* - u_h^{(t)}$$

şi u_h^* soluţia exactă.
Deci:

- se dă o estimare u_h^0 pt. soluţia pe grila fină,
- se aplică K_1 [paşi de relaxare pt. a obţine u_h^1
- vectorii $v_h^0 = u_h^* - u_h^0$, $v_h^1 = u_h^* - u_h^1$, satisfac $v_h^1 = R_h^{K_1} v_h^0$;
- se estimează v_h^1, pt. a îmbunătăţi estimarea u_h^1 astfel:
  (a) restul asociat cu u_h^1, $d_h^1 = b_h - A_h u_h^1$ este calculat şi legat de v_h^1: $A_h v_h^1 = d_h^1$.
  (b) v_h^1 se reprezintă prin interpolare v_H^1: $v_h^1 \approx I_H^h v_H^1$.
  (c) sistemul aprox. pt. v_H^1: $A_h I_H^h v_H^1 \approx d_h^1$.
  (d) sistemul este transferat pe grila mai rară: $(I_h^H A_h I_H^h) v_H^1 \approx I_h^H d_h^1$.
  (e) operatorii sunt aleşi a.î $I_h^H A_h I_H^h \approx A_H$.
  (f) v_H^1 este definit prin $A_H v_H^1 = I_h^H d_h^1$
  (g) v_h^1 se determină prin interpolare din v_H^1.
- $u_h^1 + v_h^1 = u_h^2$ nouă estimare

Formal:

$$d_h^1 = b_h - A_h u_h^1,\ d_H^1 = I_h^H d_h^1,\ v_H^1 = A_H d_H^1,\ v_h^1 = I_H^h V_H^1,\ u_h^2 = u_h^1 + v_h^1$$

sau în formă matriceală

$$u_h^2 = u_h^1 + I_H^h D_H^{-1} I_h^H (b_h - A_h u_h^1).$$

Operator I_H^h : $w_k^h = \begin{cases} (w_n^H + w_{n+1}^H)/2, & \text{if } k = 2n+1 \\ w_n^H, & \text{if } k = 2n \end{cases}$, $\quad n = \overline{0, N-1}.$

Operator I_h^H : $\quad w_n^H = (w_{2n-1}^h + 2w_{2n}^h + w_{2n+1}^h)/4, \quad n = \overline{1, N-1}.$

Ex. 2: metoda cu 2-grile pt. problema Poisson bi-dimensională

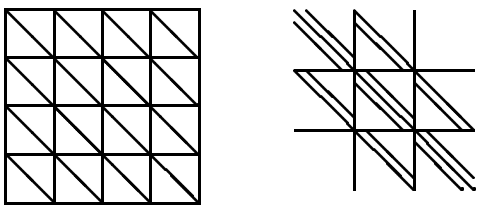
Pe $\Omega \subset \mathbb{R}^2$,

$$\sum_{i,j=1}^{2} a_{ij} \frac{\partial^2}{\partial x_i \partial y_j} u + \sum_{i=1}^{2} a_i \frac{\partial}{\partial x_i} u + a_0 u = f,$$

cu condiţiile de frontieră $\partial\Omega = \Gamma_n \cup \Gamma_s$,

$$\frac{\partial}{\partial n} u + \alpha \frac{\partial}{\partial s} u + \beta u = \gamma \text{ on } \Gamma_n, \qquad u = g \text{ on } \Gamma_s.$$

Caz: Ω=dreptunghi $\rightarrow A_h u_h = f_h$ cu A regular 7-diagonal

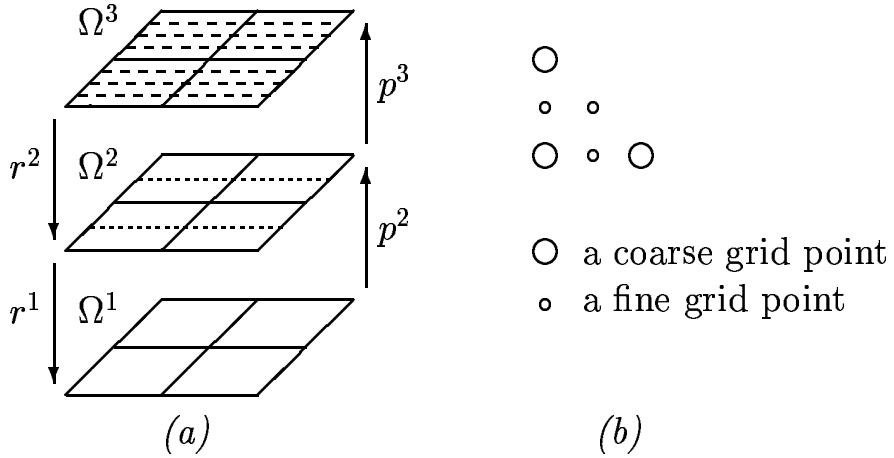


(a) *(b)*

Grile: $\Omega^k,\ k = \overline{1,l}$ obtained by

$$\Omega^k = \{(x_1, x_2) |\ x_i = x_0 + jh_i^k,\ j = \overline{0, 2^k}\} \quad h_i^{k-1} = 2h_i^k,\ k = \overline{l, 2} \quad i \geq 1$$

Operatori de prelungire şi restricţie:



(a) *(b)*

Ex: restricţie

$$\begin{pmatrix} 1/2 & 1/2 & \\ 1/2 & 1 & 1/2 \\ & 1/2 & 1/2 \end{pmatrix}.$$

Met. cu 2 nivele:

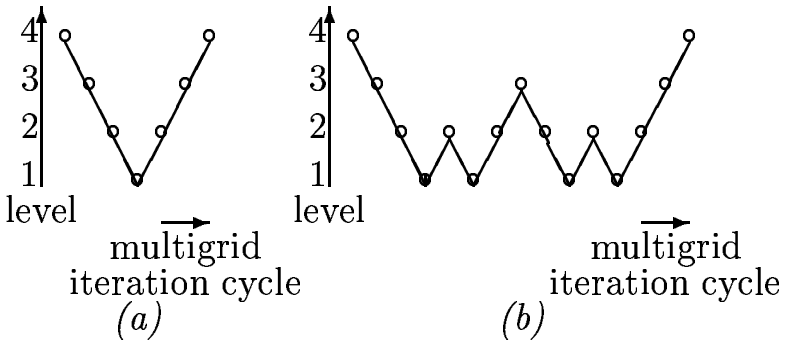
- relaxare cu met. GS.
- eroarea este transferată pe grila mai rară prin op. restricţie.
- sistemul rezultat este rezolvat direct sau printr-o metodă de relaxare.
- corecţia este transferată pe grila mai fină prin op. prelungire si adăugat aprox. existente.
- se aplică o metodă de relaxare.

Ex 3: metoda multi-grilă pt. problema Poisson bi-dimensională

proc mgsc($level, A, u, f$)
  if $level = 1$ then solve $A^1u^1 = f^1$ (solve directly)
  else
    do p times relax($A^{level}, u^{level}, f^{level}$) (prerelax)
    $f^{level-1} \leftarrow r^{level-1}(f^{level} - A^{level}u^{level})$ (restrict the residual)
    (coarse grid-correction: recursive application of mgcs)
    $u^{level-1} \leftarrow 0$
    do σ times mgcs($level - 1, A^{level-1}, u^{level-1}, f^{level-1}$)
    $u^{level} \leftarrow u^{level} + p^{level}u^{level-1}$ (add prolongation of the correction)
    do q times relax($A^{level}, u^{level}, f^{level}$) (post relaxation).

($p,\ \sigma$,q) definesc strategia (nr. de prelexări, corecţii şi post-relaxări)

Ex: $l = 4,\ \sigma = 1 \rightarrow$ V-ciclu; $l = 4,\ \sigma = 2 \rightarrow$ W-ciclu.



Implementare paralelă

1. metodele de relaxare intrinseci pot fi paralelizate (ex: ordonare roşu-negru)
2. varianta 1: grila fină este distribuită între procesoare
3. varianta 2: grile distribuite între procesoare (efort nebalansat)
4. varianta 3: mai multe grile rare pentru fiecare grilă fină.
5. varianta 4: asincron – toate nivelele sunt activate simultan dar fiecare comunică ocazional cu cele învecinate.

---

2. Metoda elementului finit

Problema: ? $u \in H_0^1(\Omega)$:

$$a(u,v) =< f, v >, \quad \text{for } v \in H_0^1(\Omega)$$

$$a(u,v) = \int_\Omega \sum_{i,j=1,2} a_{ij}(x) \frac{\partial}{\partial x_i} \frac{\partial}{\partial x_j} v(x) dx,$$

$$< w, v >= \int_\Omega w(x) v(x) dx,$$

? u a.i.

$$\int_\Omega \nabla u(x,y) \cdot \nabla v(x,y) dx = \int_\Omega f(x,y) v(x,y) dx,\ \forall v$$

$$\nabla u := i\frac{\partial u}{\partial x} + j\frac{\partial u}{\partial y}, \quad \nabla v := i\frac{\partial v}{\partial x} + j\frac{\partial v}{\partial y}.$$

Ω divizat într-un set de elemente.
(ex: caz bidimensional, triunghiuri, patrulatere etc)
Se consideră un subspaţiu finit dimensional generat de polinoame Ω^h.
? $u^h \approx u$

$$\int_{\Omega^h} \nabla u^h(x) \cdot \nabla v^h(x) dx = \int_{\Omega^h} f(x) v^h(x) dx.$$

Se aleg

$$u^h(x,y) = \sum_{j=1}^{n} \alpha_j \Phi_j(x,y), \quad v^h(x,y) = \sum_{i=1} 6n\beta_i \Phi_i(x,y)$$

atunci

$$\sum_{i,j} \beta_i \alpha_j \int_{\Omega^h} \cdot \nabla \Phi_j(x) \nabla \Phi_i(x) dx = \sum_i \beta_i \int_{\Omega^h} f(x) \Phi_i(x) dx.$$

şi se def.

$$k_{ij} = \int \Omega^h \nabla \Phi_j(x) \cdot \nabla \Phi_i(x) dx, \quad f_i = \int_{\Omega^h} f(x) \Phi_i(x) dx$$

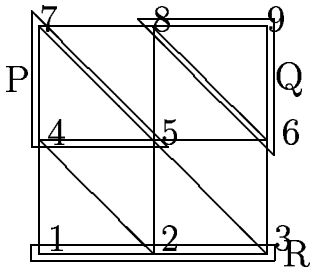
atunci

$$\sum_{ij} \beta_i (\alpha_j k_{ij} - f_i) = 0$$

adică trebuie rezolvat

$$\sum_j k_{ij} \alpha_j = f_i.$$

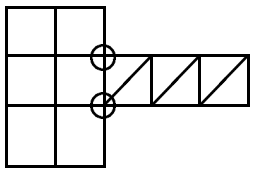
Exploatare paralelism: calcularea unor sisteme liniare în paralel, apoi asamblarea lor şi soluţionarea sistemului mare.
Ex: distribuţia calculului pe 3 procesoare



---

3. Metoda descompunerii domeniului

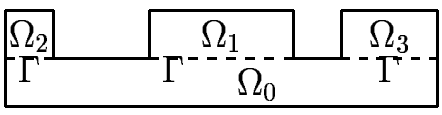
Ex: domeniu descompus în 2 subdomenii (cu cercuri elemente de legătură)



Se numerotează pct. a.î. cele de interfaţă sunt ultimele.
Sistem liniar de rezolvat are matricea

$$\begin{pmatrix} A_1 & & C_1 \\ & A_2 & C_2 \\ D_1 & D_2 & B \end{pmatrix}$$

Idee: se fac presupuneri asupra pct. de legătură, se rezolvă (în paralel) sistemele subdomeniilor, se actualizează pct. de legătură, se reia procesul.
Ex. 2:



Metoda Schwarz aplicată la problema

$$F(u,x,y)=\frac{\partial u}{\partial x}-\varepsilon\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=0,\ x,y\in[0,1]\times[0,1],$$

$$u(0,y)=u(x,1)=1,\ u(x,0)=0,\frac{\partial u}{\partial x}(1,y)=0.$$

$\Omega=\Omega_1\cup\Omega_2\cup\Omega_3$, cu $\begin{cases}\Omega_1=[0,1]\times[b,1],\\ \Omega_2=[0,r]\times[0,t],\ b<t,\\ \Omega_3=[l,1]\times[0,t],\ l<r.\end{cases}$

Fie $u_1^{(0)}=u_2^{(0)}=u_3^{(0)}=1$.

Se definesc şirurile $\{u^{(i)}\}$, $\{u_2^{(i)}\}$, $\{u_3^{(i)}\}$:

(a) $u_2^{(i)}$ rezolvă ec. de mai sus pe Ω_2 cu

$$\begin{cases}u_2^{(i)}=u_1^{(i)} & \text{on } [0,r]\times\{t\}\\ u_2^{(i)}=1 & \text{on } \{0\}\times[0,t]\\ u_2^{(i)}=0 & \text{on } [0,r]\times\{0\}\\ \frac{\partial}{\partial x}u_2^{(i)}=0 & \text{on } \{r\}\times[0,t]\end{cases}$$

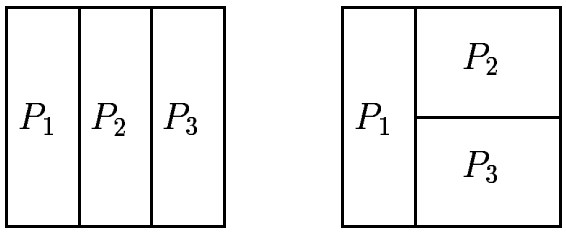
(b) $u_3^{(i)}$ rezolvă ec. pe Ω_3 cu

$$\begin{cases}u_3^{(i)}=u_2^{(i)} & \text{on } \{l\}\times[0,t]\\ u_3^{(i)}=u_1^{(i)} & \text{on } [l,1]\times\{t\}\\ u_3^{(i)}=0 & \text{on } [l,1]\times\{0,t\}\end{cases}$$

(c) $u_1^{(i)}$ rezolvă ecuaţia pe Ω_1 cu

$$\begin{cases}u_1^{(i)}=u_2^{(i)} & \text{on } [0,r]\times\{b\}\\ u_1^{(i)}=u_3^{(i)} & \text{on } [r,1]\times\{b\}\end{cases}$$

Posibilităţi de partiţionare a unui dreptunghi rectangular, între procesoare:



(a) (b)