Rezolvarea sistemelor de ecuaţii algebrice neliniare

Prof.dr.ing. Gabriela Ciuprina

Universitatea "Politehnica" Bucureşti, Facultatea de Inginerie Electrică

Suport didactic pentru disciplina *Algoritmi Numerici*, 2017-2018

Notes

Cuprins

Notes

Formularea problemei

Se dau $f_k : \mathbb{R}^n \to \mathbb{R}$, continue, $k = 1, \ldots, n$.
Se cer x_k pentru care

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

Notes

Formularea problemei

Se dă $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$, continuă.
Se cere $\mathbf{x} \in \mathbb{R}^n$ pentru care

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \tag{1}$$

unde

$$\mathbf{F} = [f_1, f_2, \ldots, f_n]^T \in \mathbb{R}^n$$

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$$

Exemplu care conduce la un astfel de sistem: analiza circuitelor rezistive neliniare.

Notes

Dificultăţi

- Nu există o metodă simplă de a încadra soluţia astfel încât să obţinem o metodă garantat convergentă ca în cazul 1D. Metodele de încadrare nu se pot generaliza.
- Efortul de calcul creşte rapid odată cu dimensiunea sistemului.

Notes

Iteraţii simple

Formularea problemei ca o problemă de punct fix

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{x} = \mathbf{G}(\mathbf{x}) \tag{2}$$

unde

$$\mathbf{G}(\mathbf{x}) = \mathbf{x} + \mathbf{C}\mathbf{F}(\mathbf{x}) \tag{3}$$

şi $\mathbf{C} \in \mathbb{R}^{n \times n}$

Iteraţii de punct fix

$$\mathbf{x}^{(k+1)} = \mathbf{G}(\mathbf{x}^{(k)}) \tag{4}$$

Notes

Iteraţii simple

O condiţie suficientă de convergenţă este:

- În 1D $|g'(x^*)| < 1$ unde x^* este soluţia.
- În nD

$$\rho(\mathbf{G}'(\mathbf{x}^*)) < 1$$

unde $\mathbf{G}'$ este matricea Jacobian.

şi iniţializarea este suficient de apropiată de soluţie
Deoarece $\rho(\mathbf{A}) \leq \|\mathbf{A}\| \Rightarrow$ condiţie suficientă de convergenţă:

$$\|\mathbf{G}'(\mathbf{x}^*)\| < 1 \Leftrightarrow \|\mathbf{I} + \mathbf{C}\mathbf{F}'(\mathbf{x})\| < 1$$

unde $\mathbf{F}'$ este matricea Jacobian.

Notes

Iteraţii simple

Matricea Jacobian

$$\mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \stackrel{not}{=} \mathbf{J}_F(\mathbf{x}) \qquad (5)$$

Notes

Newton - ideea

Convergenţa este cu atât mai rapid convergentă cu cât $\|\mathbf{I} + \mathbf{C}\mathbf{F}'(\mathbf{x})\|$ este mai mică.

$$\Rightarrow$$

Viteza maximă de convergentă pentru

$$\|\mathbf{I} + \mathbf{C}\mathbf{F}'(\mathbf{x})\| = 0$$

$$\mathbf{C} = -(\mathbf{F}'(\mathbf{x}))^{-1} \tag{6}$$

Iteraţii Newton:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{F}'(\mathbf{x}^{(k)}))^{-1}\mathbf{F}(\mathbf{x}^{(k)}) \tag{7}$$

Eşec dacă se întâlneşte o matrice Jacobian singulară.

Notes

Algoritm

Nu se implementează formula

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{F}'(\mathbf{x}^{(k)}))^{-1}\mathbf{F}(\mathbf{x}^{(k)}) \tag{8}$$

Dacă notăm $\mathbf{z}$ corecţia:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{z} \tag{9}$$

atunci

$$\mathbf{z} = -(\mathbf{F}'(\mathbf{x}^{(k)}))^{-1}\mathbf{F}(\mathbf{x}^{(k)})$$

$$(\mathbf{F}'(\mathbf{x}^{(k)}))\mathbf{z} = -\mathbf{F}(\mathbf{x}^{(k)}) \tag{10}$$

La fiecare iteraţie neliniară

1. se calculează corecţia prin rezolvarea sist. algebric liniar (10);
2. se actualizează soluţia cu (9).

Notes

Convergenţa

Pătratică
Funcţia de iteraţie $\mathbf{G}(\mathbf{x}) = \mathbf{x} - \mathbf{J}_F^{-1}(x)\mathbf{F}(\mathbf{x})$
$\mathbf{J}_G(x^*) = \mathbf{0}$
Taylor:

$$\mathbf{G}(\mathbf{x}) = \mathbf{G}(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^T \mathbf{J}_G(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^T \mathbf{H}_G(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*)/2$$

O demonstraţie riguroasă găsiţi aici

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \leq M\|(\mathbf{x}^{(k-1)} - x^*)\|^2 \qquad (11)$$

Notes

Convergenţa

Matricea Hessian

$$\mathbf{F}''(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1^2} & \frac{\partial^2 f_1}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f_1}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f_2}{\partial x_2 \partial x_1} & \frac{\partial^2 f_2}{\partial x_2^2} & \cdots & \frac{\partial^2 f_2}{\partial x_2 \partial x_n} \\ \vdots & & & \\ \frac{\partial^2 f_n}{\partial x_n \partial x_1} & \frac{\partial^2 f_n}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f_n}{\partial x_n^2} \end{bmatrix} \overset{not}{=} \mathbf{H}_F(\mathbf{x}) \qquad (12)$$

Notes

Efort de calcul

La fiecare iteraţie:

- Evaluarea Jacobianului - n^2 evaluări de funcţii scalare dacă problema este densă (fiecare componentă a funcţiei depinde de toate componentele variabilei);
- Rezolvarea unui sistem de ecuaţii algebrice liniare - n^3 dacă matricea este plină.

Foarte costisitor!

Notes

Variante inspirate din metodele 1D

Scop: reducerea efortului de calcul pe iteraţie.
Dar: convergenţă nu va mai fi pătratică ⇒ compromis.

- Newton-Kantorovich (tangente paralele)

$$(\mathbf{F}'(\mathbf{x}^{(0)}))\mathbf{z} = -\mathbf{F}(\mathbf{x}^{(k)}) \quad (13)$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{z} \quad (14)$$

  Sistemul de rezolvat are întotdeauna aceeaşi matrice a coeficienţilor ⇒ este eficientă folosirea factorizării.
- Secante - derivatele parţiale din formula Jacobianului se calculează numeric.

$$\frac{\partial f_i}{\partial x_j} = \frac{f_i(x_1^{(k-1)}, x_2^{(k-1)}, \ldots, x_j^{(k)}, \ldots, x_n^{(k-1)}) - f_i(x_1^{(k-1)}, x_2^{(k-1)}, \ldots, x_j^{(k-1)}, \ldots, x_n^{(k-1)})}{x_j^{(k)} - x_j^{(k-1)}}$$

Notes

Convergenţă locală/globală

Metodă locală

O metoda iterativă este locală - dacă ea converge doar dacă iniţializarea este suficient de aproape de soluţie (*în interiorul razei de convergenţă.*)

- Metoda Newton şi variantele ei sunt locale.
- Dacă iniţializarea este foarte proastă, atunci metodele locale pot fi modificate pentru a îmbunătăţi convergenţa lor globală [Martinez]. Exemple de astfel de metode:
  - Newton cu amortizare;
  - quasi-Newton (Broyden);
  - regiuni de încredere;
  - descompunere (separare/decuplare/segregare).

Notes

Metoda Newton cu amortizare

Se calculează corecţia $\mathbf{z}$ la fiecare iteraţie, dar aproximaţia nouă se calculează ca

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}$$

α_k este un parametru scalar care se alege a.î. aproximaţia $\mathbf{x}^{(k+1)}$ să fie mai bună decât aproximaţia $\mathbf{x}^{(k)}$.

Notes

Metoda Newton cu amortizare

Exemplu:
α_k se alege a.î. $\|\mathbf{f}(\mathbf{x}^{(k)})\|_2$ să scadă suficient de mult la fiecare iteraţie.

sau

α_k se alege a.î. $h(\alpha) = \|\mathbf{f}(\mathbf{x}^{(k)}) + \alpha \mathbf{z}\|_2$ să fie minimă

- Există o legătură între rezolvarea sistemelor neliniare şi tehnicile de minimizare (optimizare).
- Când aproximările au devenit suficient de aproape de soluţie, coeficientul de amortizare trebuie să devină 1.

Notes

Metoda regiunii de încredere

Este o metodă mai complicată dar care face metoda de tip Newton să fie mai robustă.
Ideea:

- Se estimeaza razei unei "regiuni de încredere" în care aproximarea seriei Taylor pe care se bazează metoda Newton să fie suficient de precisă.
- Corecţia soluţiei se ajustează în funcţie de raza acestei regiuni de încredere.
- În apropierea soluţiei, regiunile de încredere sunt suficient de mari a.î sunt permise corecţii totale Newton.

Şi această metodă foloseşte tehnici de optimizare[1]

[1] Detalii ale acestor algoritmi vor fi prezentate în capitolul de optimizare.

Notes

Metoda regiunii de încredere

Exemple de algoritmi din această categorie:

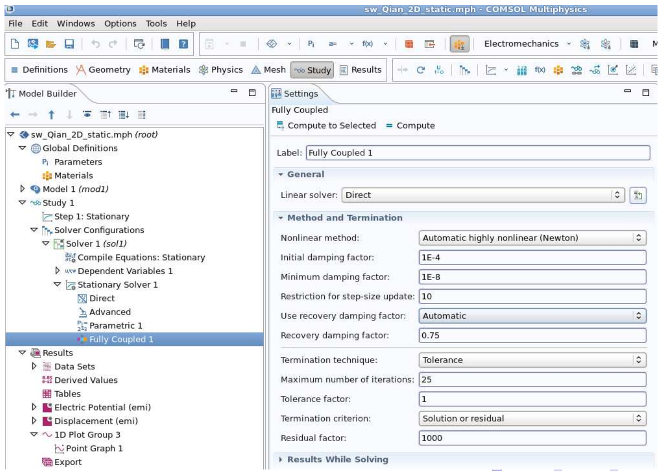
- Levenberg - Marquardt;
- *double dogleg*;
- etc.

Astfel de algoritmi sunt folosiţi de exemplu în:

- funcţia matlab `fsolve`;
- COMSOL
- etc

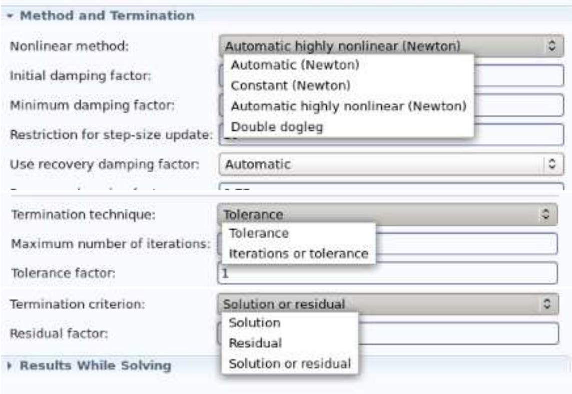
Notes

Fiţi atenţi la astfel de informaţii (capturi din COMSOL)



Notes

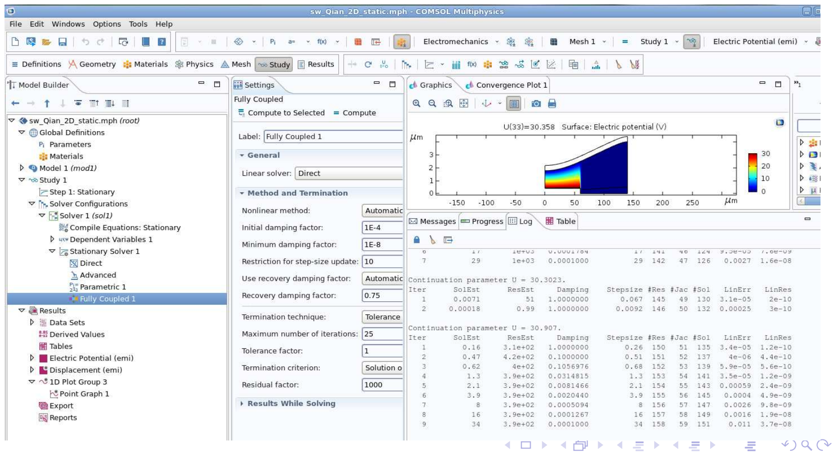
Fiţi atenţi la astfel de informaţii (capturi din COMSOL)



Notes

Formularea problemei
Iteraţii simple
Newton
Metode care aproximează Jacobianul
Metode robuste de tip Newton
Metode de descompunere
Damped Newton
Trust region

Fiţi atenţi la astfel de informaţii (capturi din COMSOL)



Notes

Ideea

Până acum componentele Jacobianului au fost evaluate în acelaşi punct.

$$\begin{aligned} f_1(x_1, x_2) &= 0, \\ f_2(x_1, x_2) &= 0 \end{aligned}$$

Newton - aplicat întregului sistem

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x_1^{(k)}, x_2^{(k)}) & \frac{\partial f_1}{\partial x_2}(x_1^{(k)}, x_2^{(k)}) \\ \frac{\partial f_2}{\partial x_1}(x_1^{(k)}, x_2^{(k)}) & \frac{\partial f_2}{\partial x_2}(x_1^{(k)}, x_2^{(k)}) \end{bmatrix}^{-1} \begin{bmatrix} f_1(x_1^{(k)}, x_2^{(k)}) \\ f_2(x_1^{(k)}, x_2^{(k)}) \end{bmatrix} \quad (15)$$

Notes

Ideea: Jacobi-Newton

Iteraţii - ca la Jacobi; Newton - aplicat pe grupuri de ecuaţii.

- se rezolvă $f_1(x_1, x_2^{(k)}) = 0 \Rightarrow x_1^{(k+1)}$;
- se rezolvă $f_2(x_1^{(k)}, x_2) = 0 \Rightarrow x_2^{(k+1)}$;

Deplasări simultane:

$$x_1^{(j+1)} = x_1^{(j)} - f_1'^{-1}(x_1^{(j)}, x_2^{(k)}) f_1(x_1^{(j)}, x_2^{(k)}) \to x_1^{(k+1)}$$

$$x_2^{(j+1)} = x_2^{(j)} - f_2'^{-1}(x_1^{(k)}, x_2^{(j)}) f_2(x_1^{(k)}, x_2^{(j)}) \to x_2^{(k+1)}$$

Condiţie: să existe inversele.

Notes

Ideea: Jacobi-Newton

Aproape echivalent cu

$$\begin{bmatrix} x_1^{(j+1)} \\ x_2^{(j+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \end{bmatrix} - \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x_1^{(j)}, x_2^{(k)}) & 0 \\ 0 & \frac{\partial f_2}{\partial x_2}(x_1^{(k)}, x_2^{(j)}) \end{bmatrix}^{-1} \begin{bmatrix} f_1(x_1^{(j)}, x_2^{(k)}) \\ f_2(x_1^{(k)}, x_2^{(j)}) \end{bmatrix} \quad (16)$$

Termenii care indică cuplajul $\partial f_1/\partial x_2$, $\partial f_2/\partial x_1$ sunt neglijaţi.

Notes

Ideea: Gauss-Seidel-Newton

Iteraţii - ca la Gauss-Seidel; Newton - aplicat pe grupuri de ecuaţii.

- se rezolvă $f_1(x_1, x_2^{(k)}) = 0 \Rightarrow x_1^{(k+1)}$;
- se rezolvă $f_2(x_1^{(k+1)}, x_2) = 0 \Rightarrow x_2^{(k+1)}$;

Deplasări succesive:

$$x_1^{(j+1)} = x_1^{(j)} - f_1'^{-1}(x_1^{(j)}, x_2^{(k)}) f_1(x_1^{(j)}, x_2^{(k)}) \to x_1^{(k+1)}$$

$$x_2^{(j+1)} = x_2^{(j)} - f_2'^{-1}(x_1^{(k+1)}, x_2^{(j)}) f_2(x_1^{(k+1)}, x_2^{(j)}) \to x_2^{(k+1)}$$

Condiţie: să existe inversele.

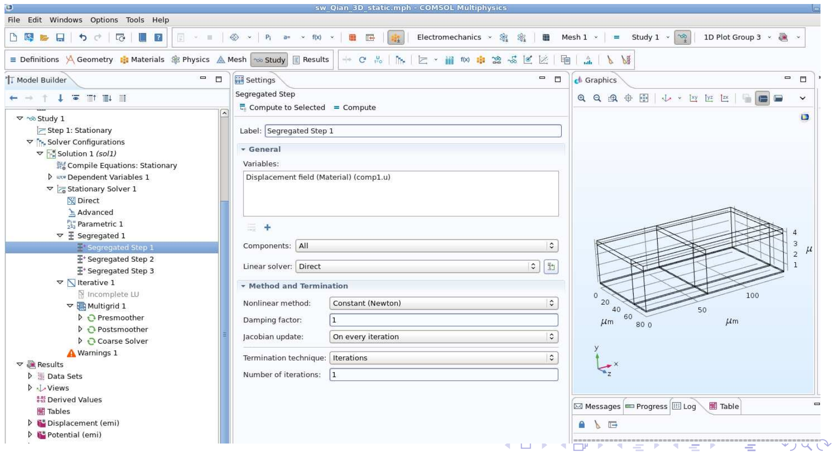
Notes

Aplicaţii tipice

- Această abordare este mai ales utilă în probleme multifizice, în care separarea sistemului în grupuri de ecuaţii se face pe considerente fizice.
- Fiecare grup de ecuaţii provine din formulările matematice ale unor probleme foarte bine definite şi cunoscute, cu operatori matematici foarte bine studiaţi.
- Rezolvarea pe componente (cuplaj slab) poate fi mai robustă decât rezolvarea simultană (cuplaj tare).

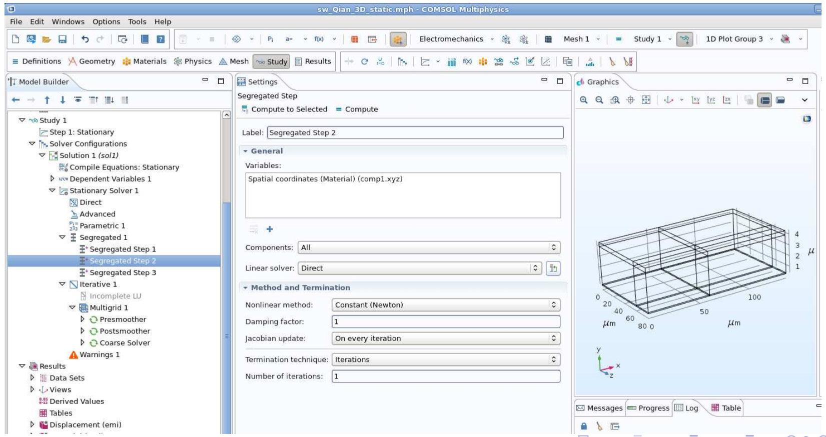
Notes

Rezolvarea pe componente (cuplaj slab) în COMSOL



Notes

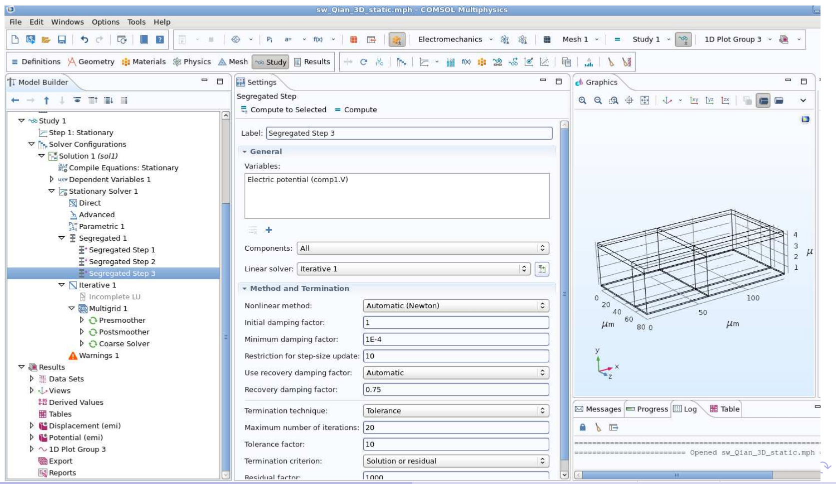
Rezolvarea pe componente (cuplaj slab) în COMSOL



Notes

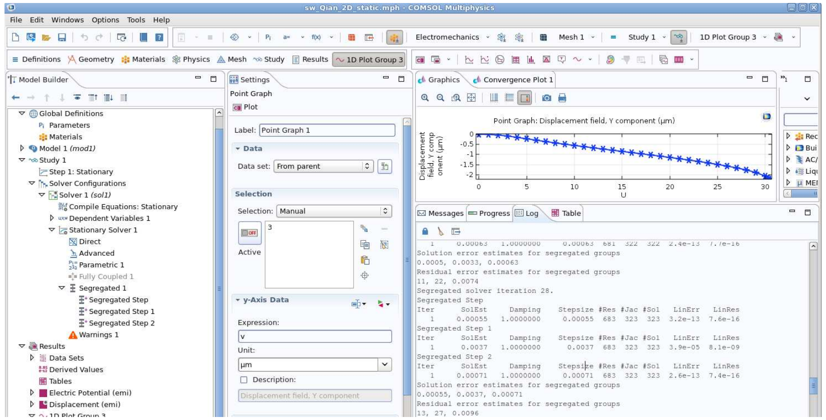
Formularea problemei
Iterații simple
Newton
Metode care aproximează Jacobianul
Metode robuste de tip Newton
Metode de descompunere

Rezolvarea pe componente (cuplaj slab) în COMSOL



Notes

Rezolvarea pe componente (cuplaj slab) în COMSOL



Notes

Referinţe

- [Ioan98] D. Ioan et al., *Metode numerice în ingineria electrică*, Ed. Matrix Rom, Bucuresti, 1998. (Capitolul 16)
- [Cheney] Ward Cheney and David Kincaid, *Numerical Mathematics and Computing*, Brooks/Cole publishing Company,2000.
- [Heath] Michael Heath, *Scientific computing. An Introductory Survey*, McGraw Hill 2002 (capitolul 5 din carte) şi alte resurse de la

  disponibilă la http://heath.cs.illinois.edu/scicomp/
- [Martinez] Jose Mario Martinez, *Algorithms for Solving Nolinear Systems of Equations*, 1994, disponibil la http://www.ime.unicamp.br/ martinez/nato.pdf

Notes