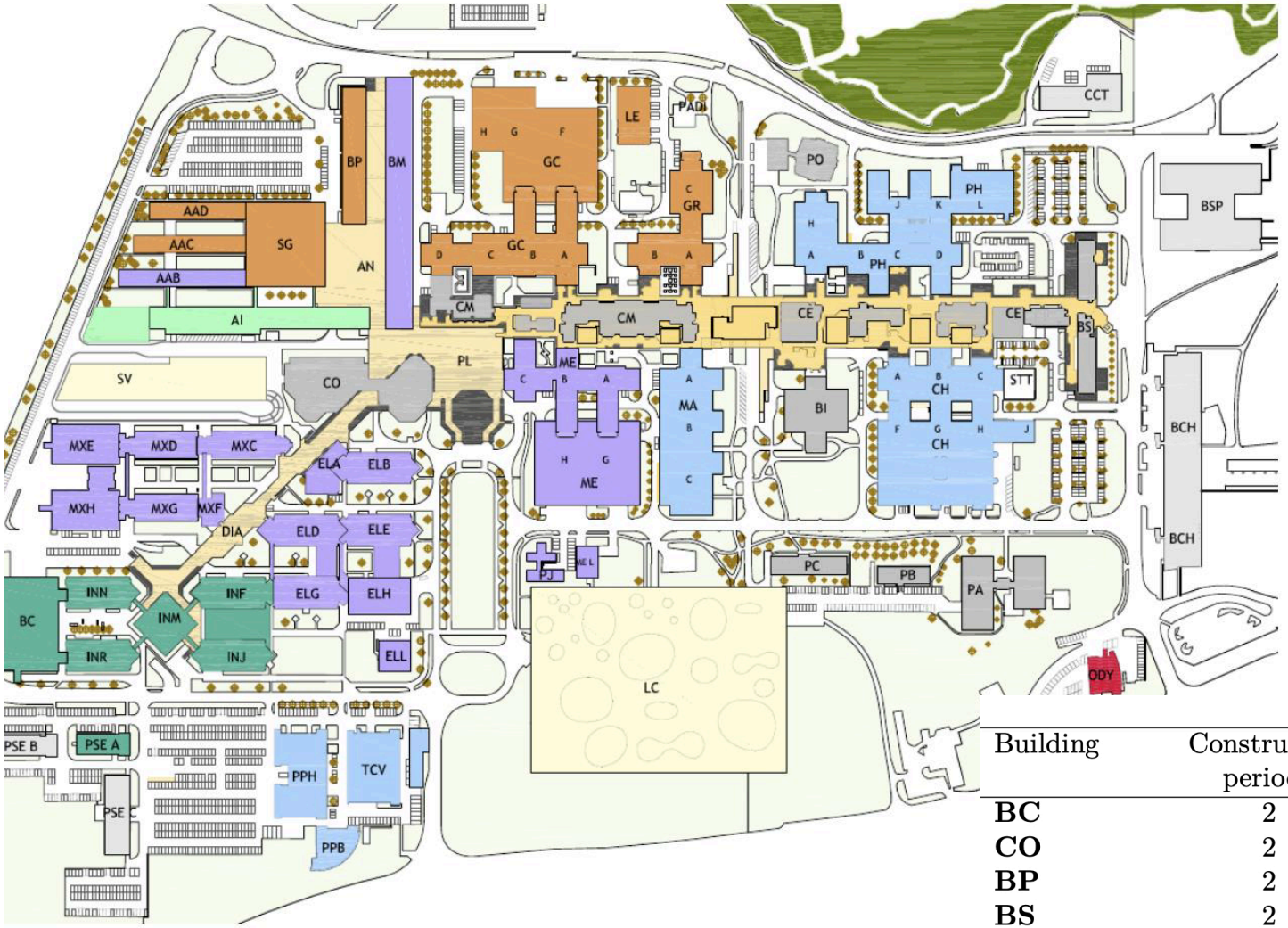
Solving a set of non linear equations

MOES
François Marechal

IPESE
Industrial Process
and Energy Systems
Engineering

Part1 : defining the building needs

Buildings stock



Heating needs are defined by :

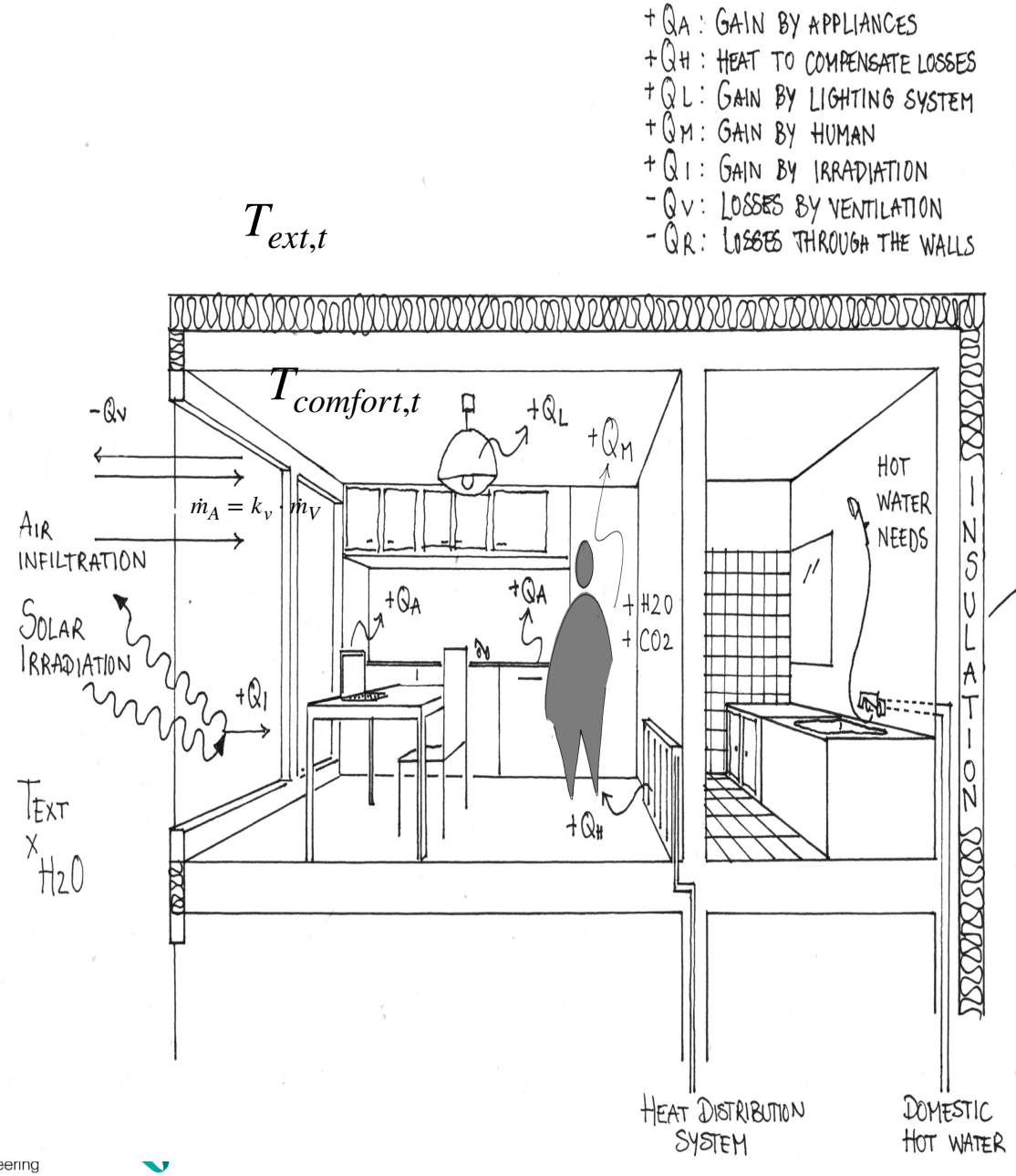
- $\dot{Q}_{b,t} \quad \forall b \in \{1..n_b\}$ [kW]
  heat required by building b
  at time t of the year
- $T^{supply}_{b,t} \quad \forall b \in \{1..n_b\}$ [°C]
  Minimum temperature
  of the heat supply of building b
  at time t of the year

Available :

Table 1.1: EPFL Buildings

Building	Construction period[a]	Heated surface A_{th} [m^2]	Annual heat demand Q_{th} [kWh]	Annual electricity demand Q_{el} [kWh]
BC	2	17480	418,491	1,603,596
CO	2	11901	477,008	943,653
BP	2	10442	457,861	691,031
BS	2	10267	509,183	350,860
TCV	2	6095	318,209	2,067,675

What do we know ?



Our knowledge :

The model of the building envelop heat balance

$$\dot{Q}_{A,t}[kW] = \dot{E}_{A,t}$$
$$\dot{Q}_{L,t}[kW] = \dot{E}_{L,t}$$
$$\dot{Q}_{M,t}[kW] = \dot{q}_{cap} \cdot \sum_{cap} cap_t$$
$$\rightarrow \dot{Q}_{gain,t}$$

Gains

$$\dot{Q}_{I,t}[kW] = \dot{Irr}_t \cdot k_{sun}$$
$$\dot{Q}_{V,t}[kW] = \dot{m}_V \cdot cp_{air} \cdot (T_{comfort,t} - T_{ext,t})$$
$$\dot{Q}_{R,t}[kW] = \sum_{wall} k_{wall} \cdot S_{wall} \cdot (T_{comfort,t} - T_{ext,t})$$

Losses

$$\dot{Q}_{V,t} + \dot{Q}_{R,t} = k_{th}(T_{comfort,t} - T_{ext,t})$$

Heat balance :

what is the load to maintain the comfort temperature ?

$$\dot{Q}_{H,t}[kW] = max(0, \dot{Q}_{V,t} + \dot{Q}_{R,t} - (\dot{Q}_{A,t} + \dot{Q}_{L,t} + \dot{Q}_{M,t} + \dot{Q}_{I,t}))$$

State the problem to be solved : what needs to be calculated

$$Q_H(t_0, t_{end}) = \int_{t_0}^{t_{end}} \dot{Q}_{H,t} \cdot dt = \int_{t_0}^{t_{end}} max(0, k_{th} \cdot (T_{comfort,t} - T_{ext,t}) - k_{sun} \cdot Irr_t - \dot{Q}_{gain,t}) \cdot dt$$

Calculate k_{sun} and k_{th} if

$$Q^1 = Q_H(January, December)$$

and

$$Q^2 = Q_H(February, March)$$

$$Q_H(January, December)(k_{sun}, k_{th}) - Q^1 = 0$$

$$Q_H(February, March)(k_{sun}, k_{th}) - Q^2 = 0$$

2 equations and 2 unknowns : k_{sun}, k_{th}

Activate your knowledge :

Solve a set of equations

$F(X,\pi)=0$

Numerical methods for solving a set of non linear equations

When f(x,π)=0 can not be transformed into x=$\phi(\pi)$ by mathematical manipulation

- Different problem formulations
  - 1 variable & 1 equation
    - f(x,π) = 0 : implicit equation
    - x=f(x,π) : explicit equation
  - N Equations & N Variables
    - F(X,π) = 0 : implicit equations
    - X = F(X,π) : explicit equations

Solving f(x)= 0 : Newton-Raphson method

find x such that f(x) = 0

TAYLOR development to approximate any function :

$$f(x) = f(x^0) + (x - x^0) \cdot f'(x^0) + \frac{(x - x^0)^2}{2!} f''(x^0) + \cdots$$

Irst order limitation (approximate with a straight line) near x^0

$$f(x^0) + (x^* - x^0) \,.f'(x^0) \cong f(x^*) = 0$$

$$x^* = x^0 - \frac{f(x^0)}{f'(x^0)}$$

Solving f(x)= 0

- Newton Raphson algorithm
  - Take $n = 0$ and x^0 = initial guess of x^*.
  - Repeat (Iterations) :

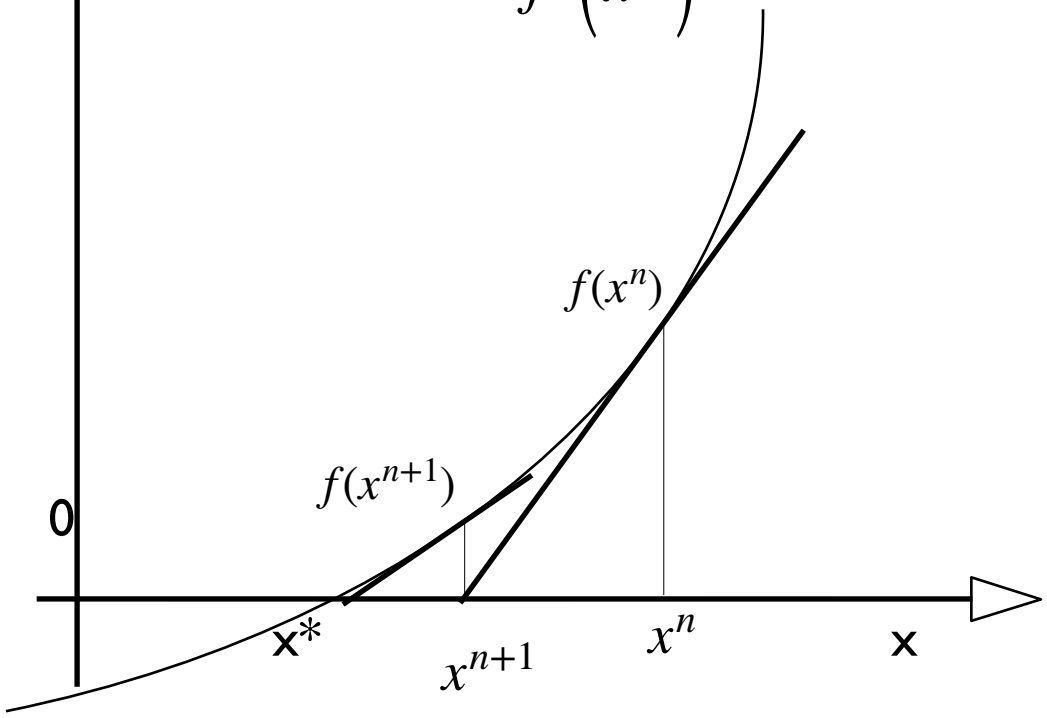
$$x^{n+1} = x^n - \frac{f(x^n)}{f'(x^n)}$$

until $f(x^{n+1})$ sufficiently close to 0

geometrical interpretation

- tangent approximation
- intersection with f=0

$$x^{n+1} = x^n - \frac{f\left(x^n\right)}{f'\left(x^n\right)}$$



Numerical calculation of derivatives

Forward calculation

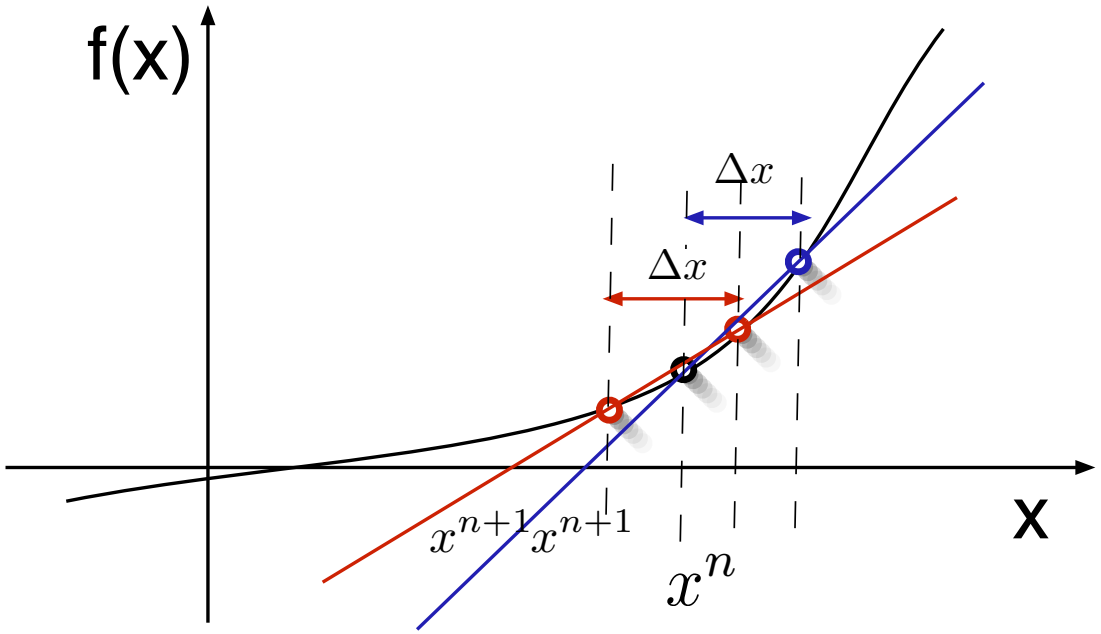
$$f'(x^n) = \left[\frac{\delta f(x)}{\delta x}\right]_{x^n} = \frac{f(x^n) - f(x^n + \Delta x)}{\Delta x}$$

+ 1 function evaluation

Central value

$$f'(x^n) = \left[\frac{\delta f(x)}{\delta x}\right]_{x^n} = \frac{f(x^n - \frac{\Delta x}{2}) - f(x^n + \frac{\Delta x}{2})}{\Delta x}$$

+ 2 function evaluations



Convergence criteria

- Choose a small value well chosen value

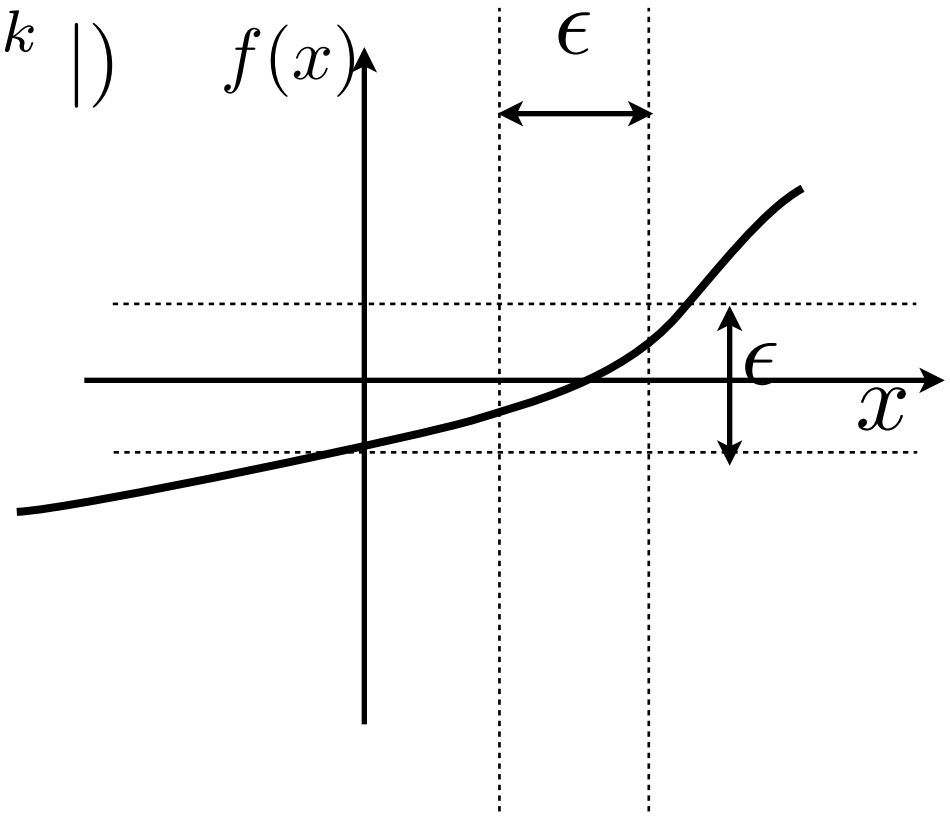
$$\epsilon = 1.0E - 06$$

- Test variable variations

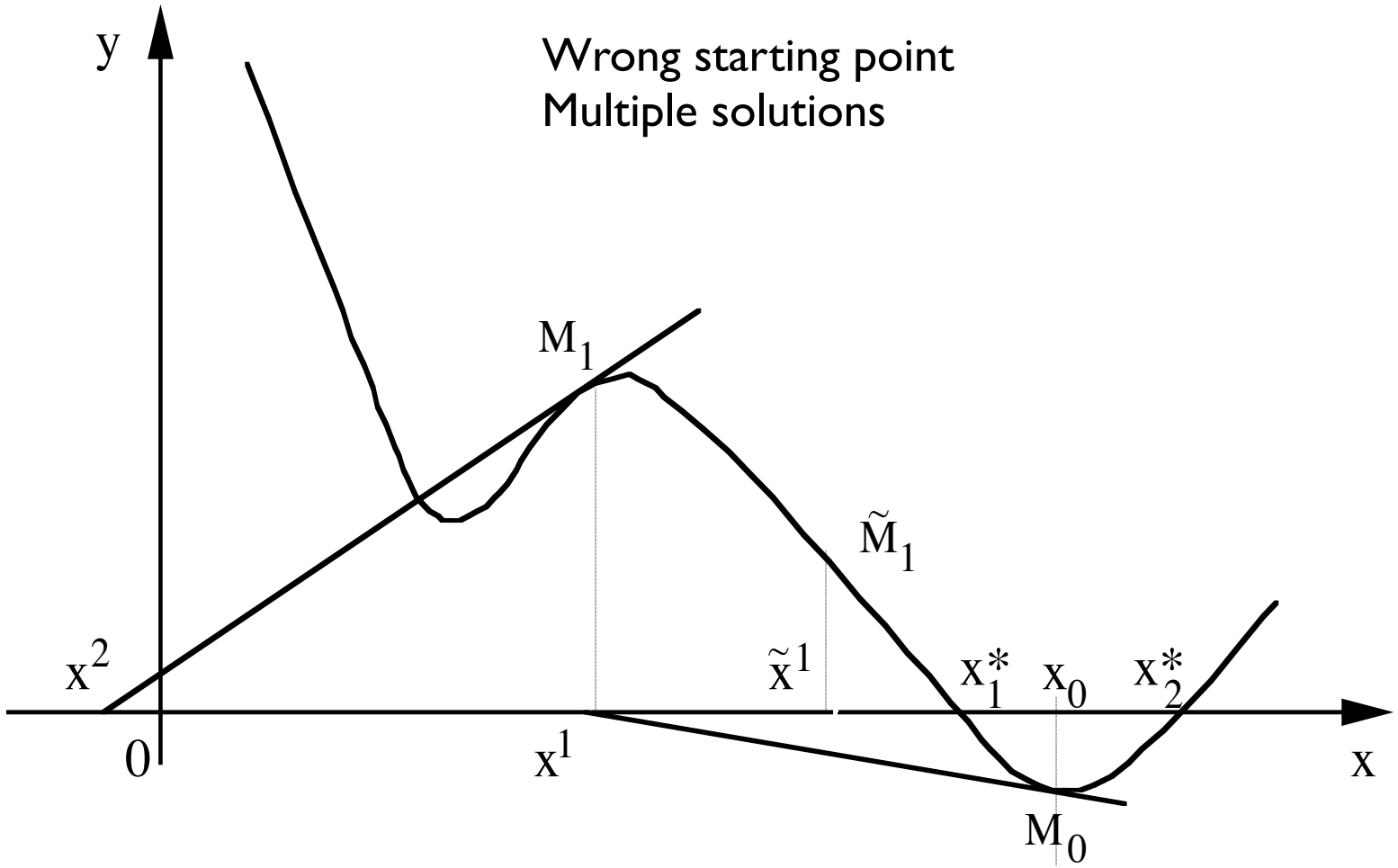
$$\mid x^{k+1} - x^k \mid \leq \epsilon * (1 + \mid x^k \mid)$$

- Test function variation

$$\mid f(x^k) \mid \leq \epsilon$$



Newton-Raphson : possible problems



Increase robustness of solving procedure

- When you suspect divergence :
  i.e. $|f(x^{n+1})| > |f(x^n)|$ and $sign(f(x^{n+1})) = sign(f(x^n))$

- Use relaxation factor 0 < q < 1

$$x^{n+1} = x^n \cdot q + (x^n - \frac{f(x^n)}{f'(x^n)}) \cdot (1 - q)$$

  - note : q=0 => full step
- Choice of q is critical ?
  - convergence speed wrt robustness
    - q >> small steps and lots of iterations
    - q << Newton step : direct convergence if linear problem

Newton-Raphson : drawbacks

- Requires initial point
- Requires derivatives calculation
- divergence possible
- no step if tangent = 0

More efficient method

- 2nd order development (quadratic) to accelerate convergence

  - $f(x) = 0 \approx c + b(x - x^0) + a(x - x^0)^2$

  - $x - x_0 = \dfrac{-b \pm \sqrt[2]{b^2 - 4ac}}{2a}$

  - a, b, c obtained if we know 3 values of f(x)

Chord method : approximating the derivative

$$f(x) - f\left(x^k\right) = \frac{f\left(x^k\right) - f\left(x^{k-1}\right)}{x^k - x^{k-1}} \cdot \left(x - x^k\right)$$

$$x^{k+1} = x^k - \psi^{-1} \cdot f\left(x^k\right)$$

$$\psi = \frac{f\left(x^k\right) - f\left(x^{k-1}\right)}{x^k - x^{k-1}}$$

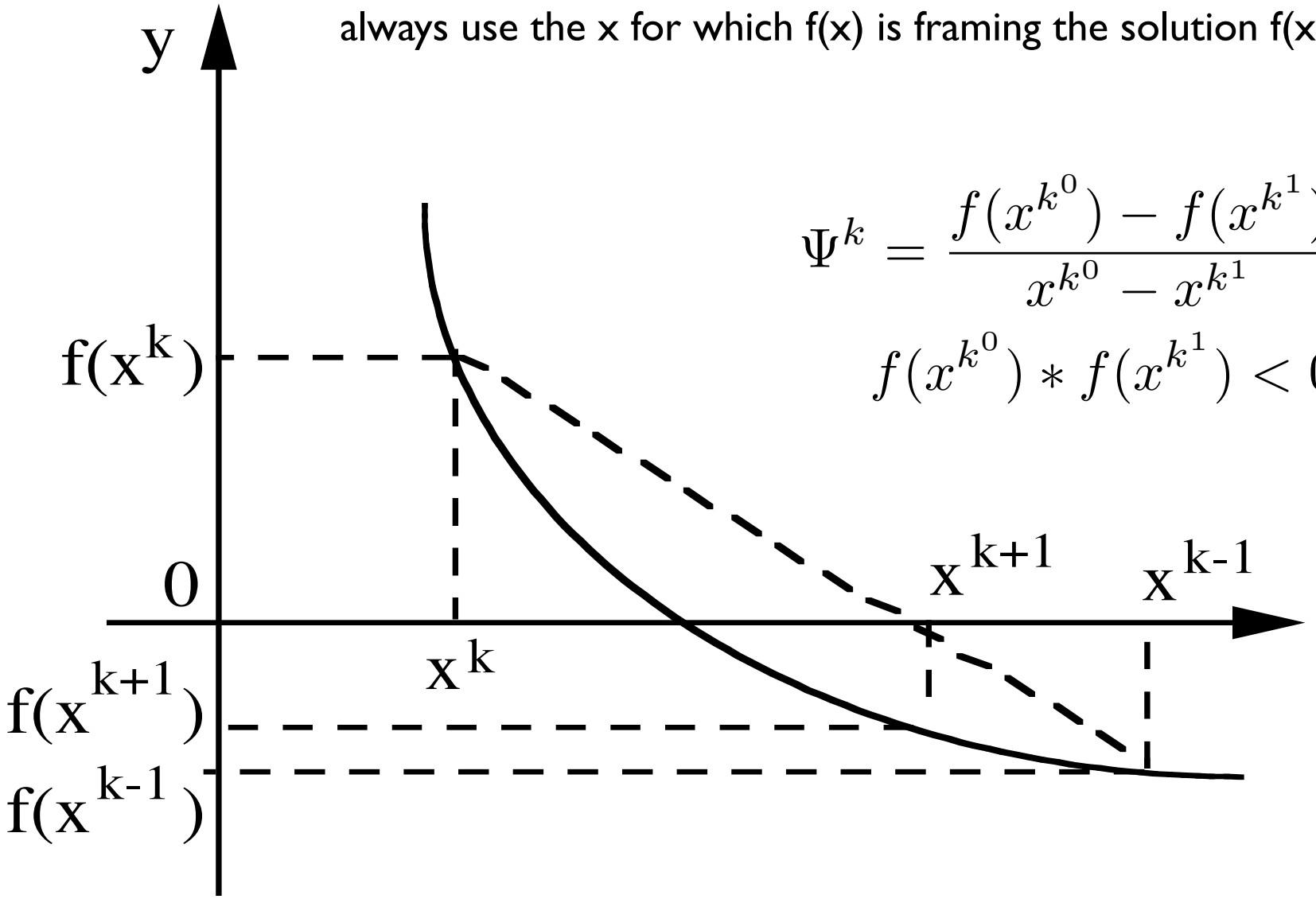
approximation of the derivative by the iteration step

Chord method : solving f(x)=0

always use the x for which f(x) is framing the solution f(x)=0,

$$\Psi^k = \frac{f(x^{k^0}) - f(x^{k^1})}{x^{k^0} - x^{k^1}}$$

$$f(x^{k^0}) * f(x^{k^1}) < 0$$



Chord method

- Pro :
  - No derivative needed
  - Similar speed as Newton-Raphson.
- Cons :
  - Slow near the solution
  - Initial point like in newton method

Solving explicit equations : x=f(x)

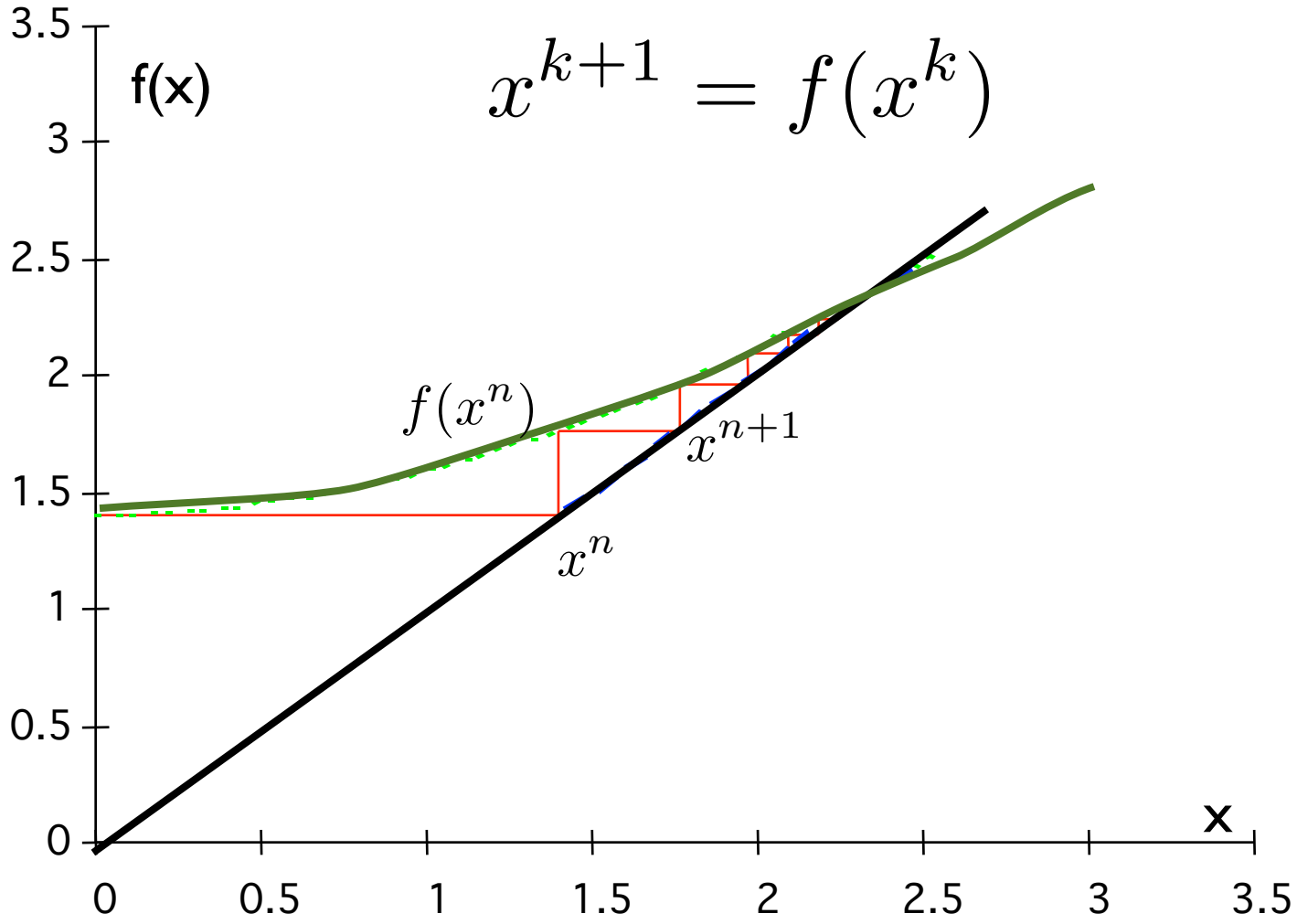
- may be transformed into implicit form
  - $x = f(x) \Rightarrow \varphi(x) = x - f(x) = 0$

Monotonic convergence

- Substitution method

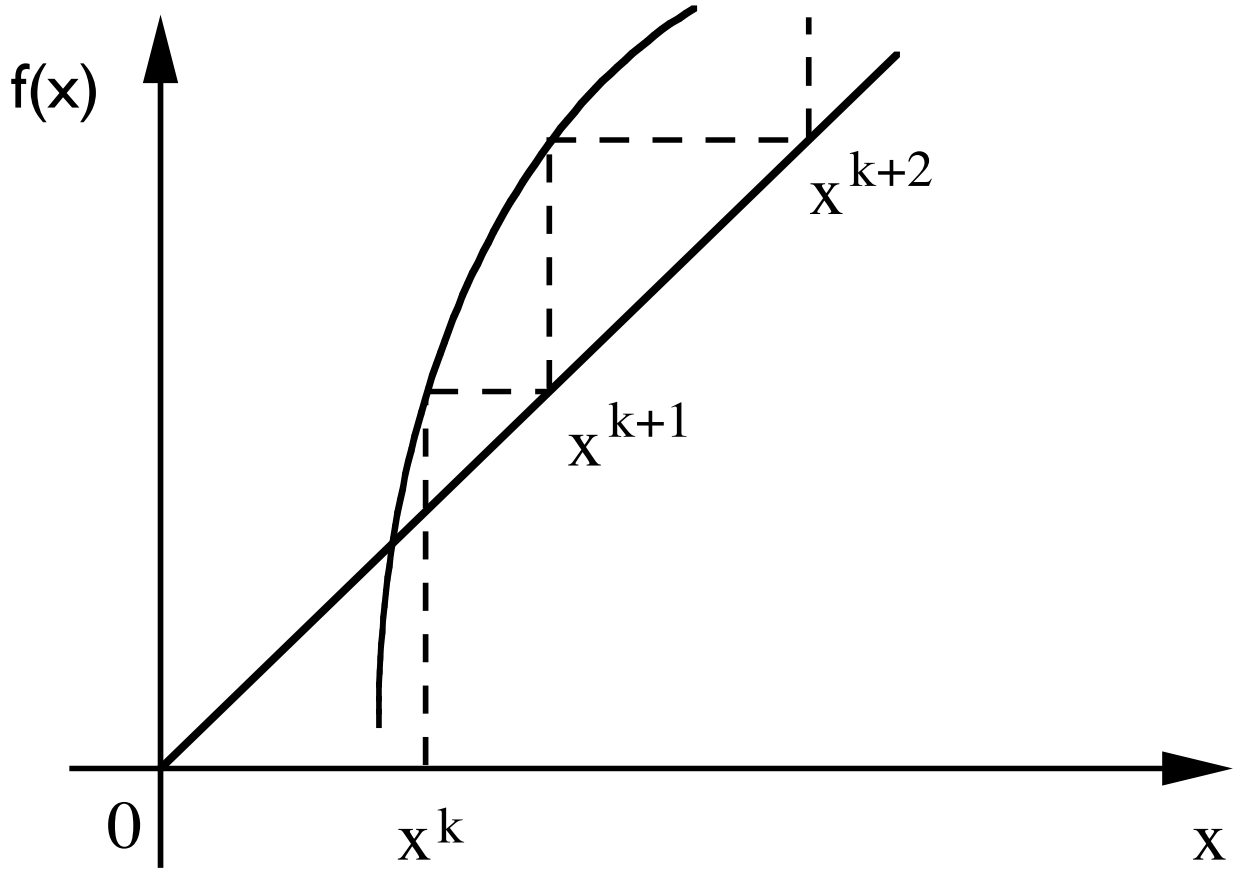
Solving x=f(x) : Explicit Equation

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

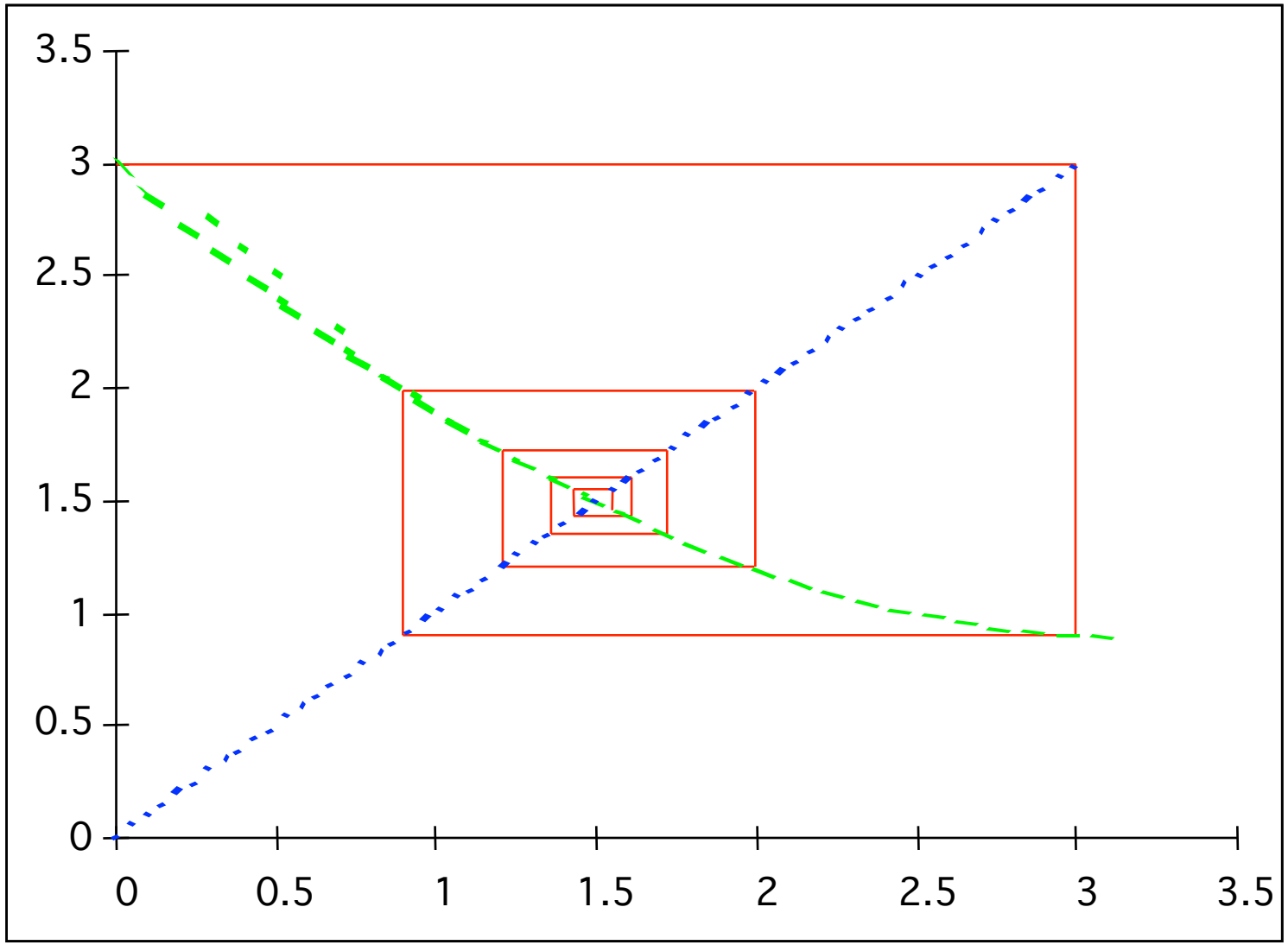


x=f(x) : Substitution may diverge

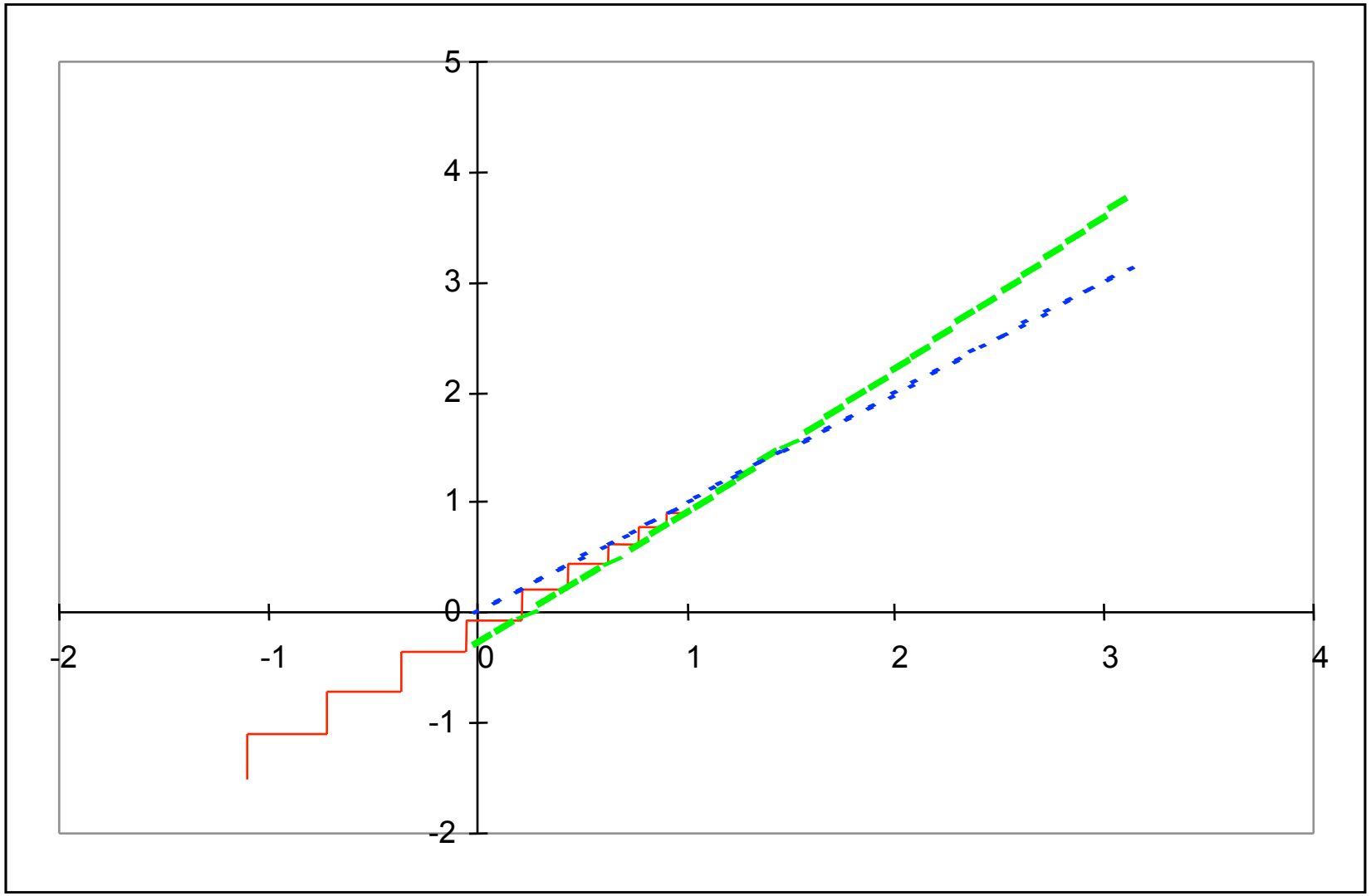
- Substitution : x = f(x) May diverge



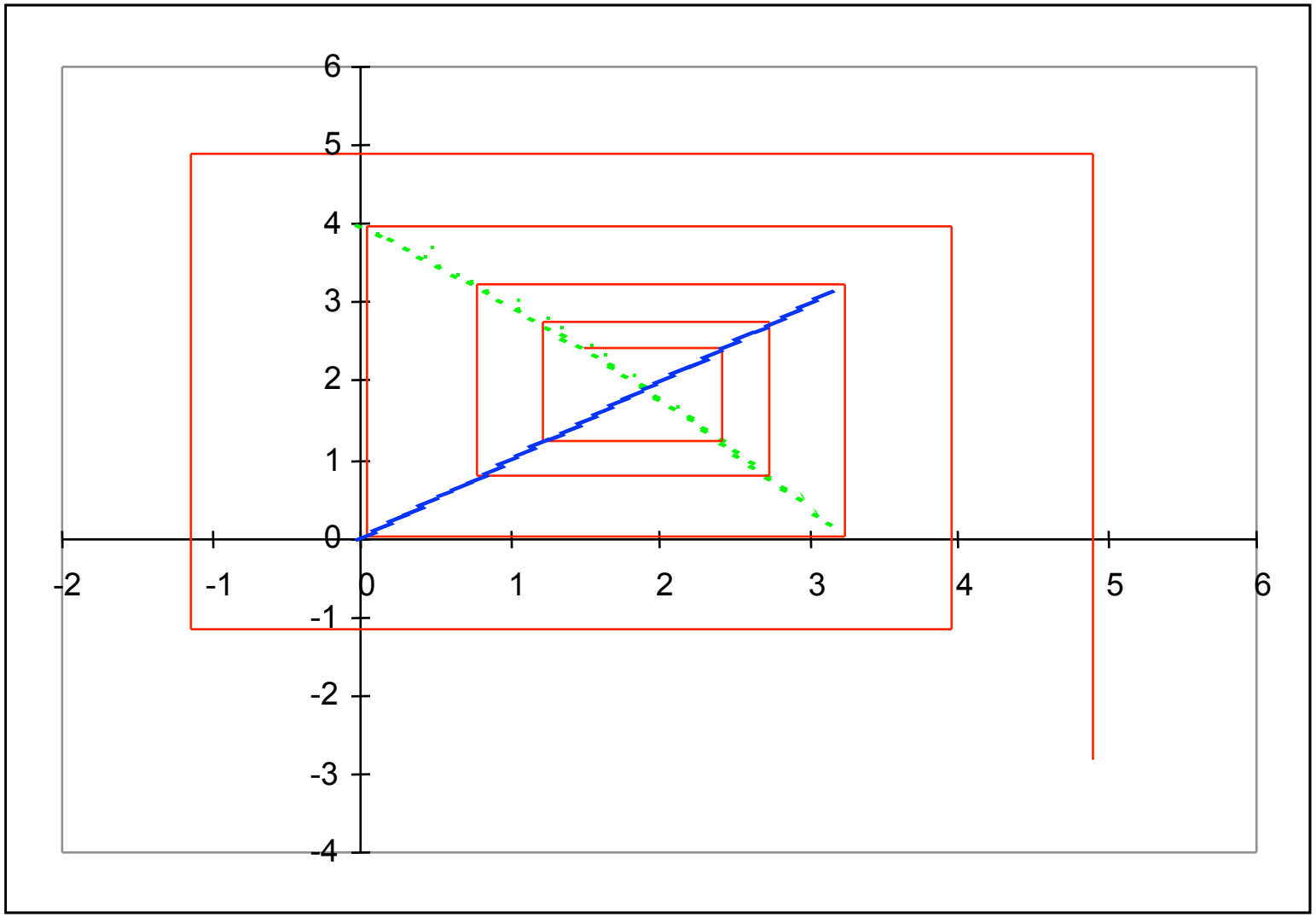
Oscillating Convergence



Monotonic Divergence



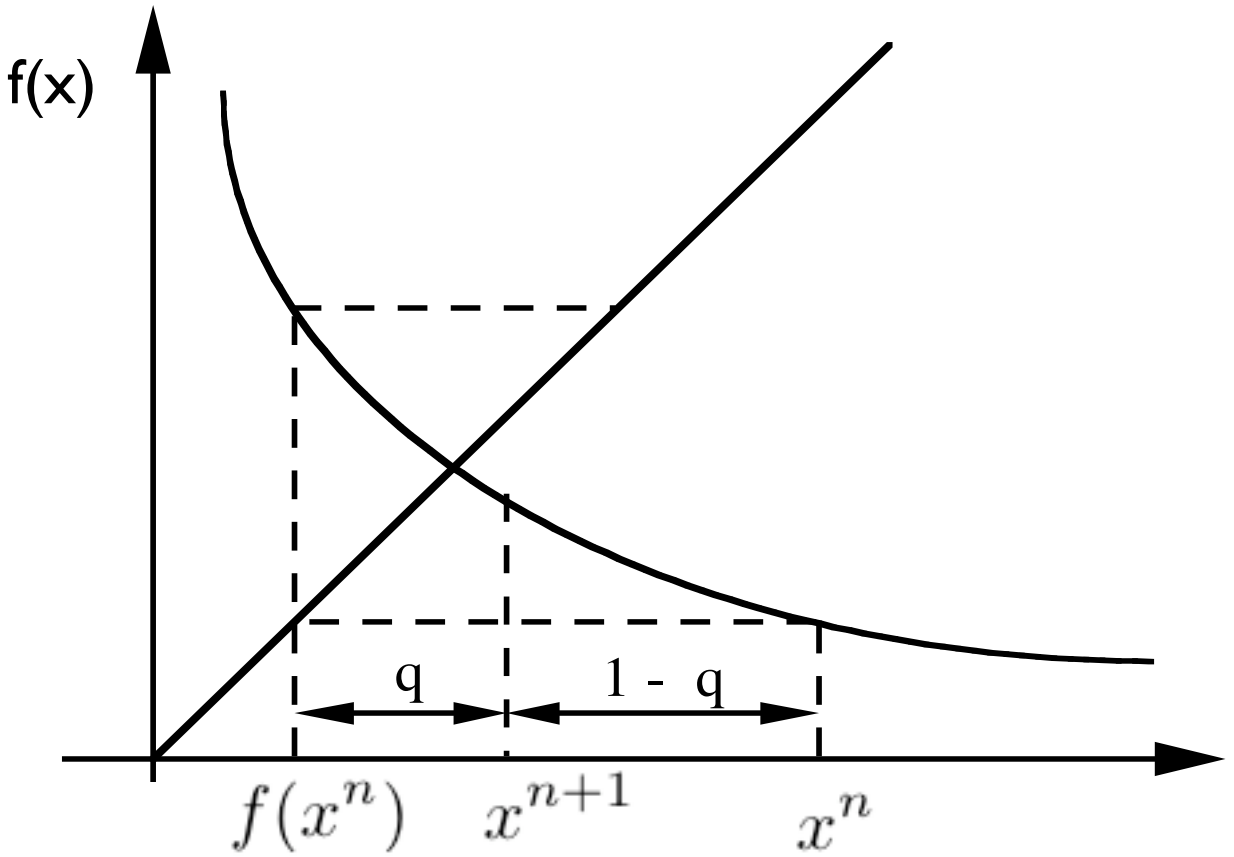
Oscilating divergence



Solution : Apply relaxation

$$x^{n+1} = x^n \cdot q + f(x^n) \cdot (1 - q)$$

Full substitution if $q = 0$

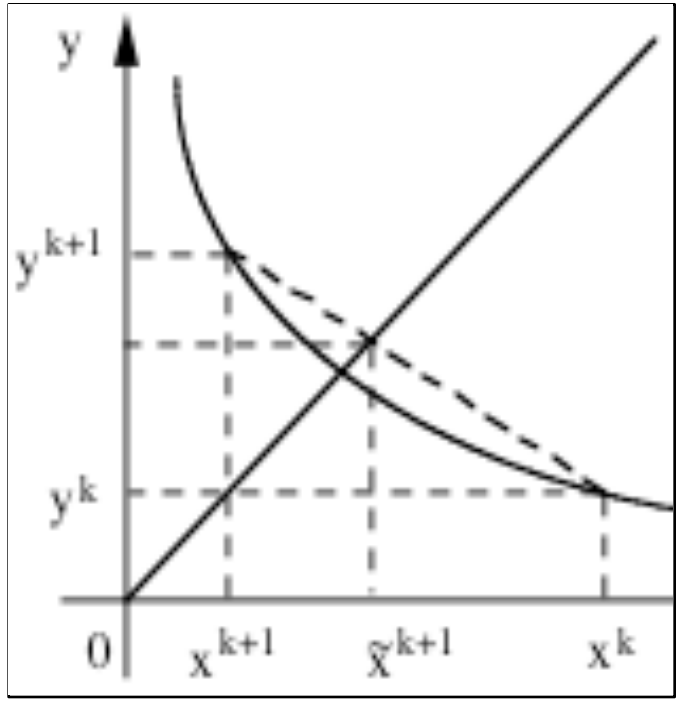


Wegstein method

- System to be solved : intersection of two curves

$$y - y^k = \frac{y^{k+1} - y^k}{x^{k+1} - \tilde{x}^k}\left(x - \tilde{x}^k\right) = \psi\left(x - \tilde{x}^k\right)$$

$$y = x$$



- ψ = Chord slope
- Therefore
-
- relaxation of substitution

$$\tilde{x}^{k+1} = \frac{1}{1-\psi}\left(x^{k+1} - \psi\,\tilde{x}^k\right)$$

$$\tilde{x}^{k+1} = q\,\tilde{x}^k + (1-q)\,x^{k+1} \qquad q = \frac{-\psi}{1-\psi} \quad et \quad 1-q = \frac{1}{1-\psi}$$

- Increased speed of convergence
- More robust

Solving a set of equations F(X)=0 : N Variables

- **Newton-Raphson n dimensions**
  - Solve F(X) = 0 (Array)

    f1(x) = f1(x1,x2,...,xn) = 0

    f2(x) = f2(x1,x2,...,xn) = 0

    . . . .

    fn(x) = fn(x1,x2,...,xn) = 0

  - Initial guess is given
  - X0 = t[x... x]

Newton Raphson : N dimension

- Taylor development

-

$$f_i(x) \approx f_i(x^0) + \left(\frac{\delta f(x)}{\delta x}\right)_{x^0} \cdot (x - x^0) = 0 \qquad i = I, n$$

Matrix representation

Jacobian matrix
matrix of derivatives

function evaluated for x0

$$\begin{vmatrix} \frac{\delta f_1}{\delta x_1} & \cdots & \frac{\delta f_1}{\delta x_j} & \cdots & \frac{\delta f_1}{\delta x_n} \\ \cdots & & \cdots & & \cdots \\ \frac{\delta f_i}{\delta x_1} & \cdots & \frac{\delta f_i}{\delta x_j} & \cdots & \frac{\delta f_i}{\delta x_n} \\ \cdots & & \cdots & & \cdots \\ \frac{\delta f_n}{\delta x_1} & \cdots & \frac{\delta f_n}{\delta x_j} & \cdots & \frac{\delta f_n}{\delta x_n} \end{vmatrix} \cdot \begin{vmatrix} \Delta x_1 \\ \cdots \\ \Delta x_j \\ \\ \Delta x_n \end{vmatrix} + \begin{vmatrix} f_1(x^0) \\ \cdots \\ f_i(x^0) \\ \cdots \\ f_n(x^0) \end{vmatrix} = \begin{vmatrix} f_1(x) \\ \cdots \\ f_i(x) \\ \cdots \\ f_n(x) \end{vmatrix}$$

$$^{t}\underline{\underline{J}}\,\underline{\Delta x} + \underline{F}\left(\underline{x}^{0}\right) = \underline{F}\left(\underline{x}\right)$$

Newton -Raphson (N dimensions)

$$ {}^{t}\underline{\underline{J}}\,\underline{\Delta x} + \underline{F}\left(\underline{x}^{0}\right) = \underline{F}\left(\underline{x}\right) $$

$$ \underline{F}\left(\underline{X}\right) = 0 $$

$$ \underline{x}^{1} = \underline{x}^{0} - {}^{t}\underline{\underline{J}}_{\underline{x}^{0}}^{-1}\,\underline{F}\left(\underline{x}^{0}\right) $$

Iteration n :

$$ \underline{x}^{n+1} = \underline{x}^{n} - {}^{t}\underline{\underline{J}}_{\underline{x}^{n}}^{-1}\,\underline{F}\left(\underline{x}^{n}\right) $$

Convergence criteria

- At iteration k
  - Variables variations

$$| x^{k+1} - x^k | \leq \epsilon * (1 + | x^k |)$$

  - Functions variation

$$| f(x^k) | \leq \epsilon$$

Scaling variables and equations

- Variables

Original problem

$$\mid f^{k+1} \mid \leq \epsilon$$
$$\mid x^{k+1} - x^{k} \mid \leq \epsilon * (1 + \mid x^{k} \mid)$$

Scaling

$$F(X) = 0 \rightarrow \Phi(\Xi) = 0$$
$$\Xi = Scale_x \cdot X$$
$$\Phi = Scale_f \cdot F(X)$$

All equations and variables are compared with the same reference value

New problem

$$\mid \chi_j^{k+1} - \chi_j^{k} \mid \leq \epsilon * (1 + \mid \chi_j^{k} \mid) \quad \forall j$$
$$\mid \phi_j^{k+1}(\chi) - \phi_j^{k}(\chi) \mid \leq \epsilon \quad \forall j$$

Newton-Raphson n dimensions ;Drawbacks

- Initial guess
- Matrix inversion + derivatives
- Matrix inversion at each iteration n*n derivatives,
- Relaxation may be applied :

$$\underline{\tilde{x}}^{n+1} = \underline{\tilde{x}}^{n} - {}^{t}\underline{\underline{J}}_{x^n}^{-1}\,\underline{F}\left(\underline{\tilde{x}}^{n}\right).(1-q)$$

Calculating derivatives

- Forward numerical estimation

$$\frac{\partial f}{\partial x_i} = \frac{f(X + \Delta x_i) - f(X)}{\Delta x_i} \quad \forall i = 1, \ldots, n_X + 1$$

- Computation time cost !
  - central derivatives avoided

- Numerical noise (is the derivative a derivative ?)

$$\Delta f(X) = |F(X)_0 - F(X)_{n_X+1}|$$

Solving X=F(X) n dimensions

- $f(x) = x - \psi(x) = 0$

- Newton?
  - $x^{k+1} = x^k - \{{}^tJ\,[f(x^k)]\}^{-1}\, f(x^k)$

- How to estimate the Jacobian matrix ?
  - $x^{k+1} = x^k - \{E - {}^tJ\,[\psi(x^k)]\}^{-1}\,[x^k - \psi(x^k)]$

quasi Newton/ Rubin Method

- Taylor development

$$\psi_1(\underline{x}) = \psi_1(\underline{x}^k) + \left(\frac{\delta\psi_1}{\delta x_1}\right)^k .(x_1 - x_1^k) + \ldots + \left(\frac{\delta\psi_1}{\delta x_n}\right)^k .(x_n - x_n^k)$$

$$\ldots$$

$$\psi_j(\underline{x}) = \psi_j(\underline{x}^k) + \left(\frac{\delta\psi_j}{\delta x_1}\right)^k .(x_1 - x_1^k) + \ldots + \left(\frac{\delta\psi_j}{\delta x_n}\right)^k .(x_n - x_n^k)$$

$$\ldots$$

$$\psi_n(\underline{x}) = \psi_n(\underline{x}^k) + \left(\frac{\delta\psi_n}{\delta x_1}\right)^k .(x_1 - x_1^k) + \ldots + \left(\frac{\delta\psi_n}{\delta x_n}\right)^k .(x_n - x_n^k)$$

Rubin method

- select n+1 initial points (by substitution for example)

  x0 , x1, x2, ..., xk,

- For each iteration store x and ψ (x)

Soit les matrices $\underline{\underline{C}}$ et $\underline{\underline{D}}$ (n+1 lignes et n colonnes)

$$\underline{\underline{C}} = \begin{matrix} x_1^0 & \dots & x_j^0 & \dots & x_n^0 \\ x_1^1 & \dots & x_j^1 & \dots & x_n^1 \\ \dots & & \dots & & \dots \\ x_1^k & \dots & x_j^k & \dots & x_n^k \\ \dots & & \dots & & \dots \\ x_1^n & \dots & x_j^n & \dots & x_n^n \end{matrix}$$

$$\underline{\underline{D}} = \begin{matrix} \psi_1(\underline{x}^0) & \dots & \psi_j(\underline{x}^0) & \dots & \psi_n(\underline{x}^0) \\ \psi_1(\underline{x}^1) & \dots & \psi_j(\underline{x}^1) & \dots & \psi_n(\underline{x}^1) \\ \dots & & \dots & & \dots \\ \psi_1(\underline{x}^k) & \dots & \psi_j(\underline{x}^k) & \dots & \psi_n(\underline{x}^k) \\ \dots & & \dots & & \dots \\ \psi_1(\underline{x}^n) & \dots & \psi_j(\underline{x}^n) & \dots & \psi_n(\underline{x}^n) \end{matrix}$$

After n iterations
we have n equations
=> J is estimated

Rubin cont.

- From C and D,

$$\underline{\underline{A}} = \begin{bmatrix} x_1^1 - x_1^0 & \cdots & x_j^1 - x_j^0 & \cdots & x_n^1 - x_n^0 \\ \vdots & \ddots & \vdots & & \vdots \\ x_1^k - x_1^0 & \cdots & x_j^k - x_j^0 & \cdots & x_n^k - x_n^0 \\ \vdots & & \vdots & \ddots & \vdots \\ x_1^n - x_1^0 & \cdots & x_j^n - x_j^0 & \cdots & x_n^n - x_n^0 \end{bmatrix}$$

$$\underline{\underline{B}} = \begin{bmatrix} \psi_1(\underline{x}^1) - \psi_1(\underline{x}^0) & \cdots & \psi_j(\underline{x}^1) - \psi_j(\underline{x}^0) & \cdots & \psi_n(\underline{x}^1) - \psi_n(\underline{x}^0) \\ \vdots & \ddots & \vdots & & \vdots \\ \psi_1(\underline{x}^k) - \psi_1(\underline{x}^0) & \cdots & \psi_j(\underline{x}^k) - \psi_j(\underline{x}^0) & \cdots & \psi_n(\underline{x}^k) - \psi_n(\underline{x}^0) \\ \vdots & & \vdots & \ddots & \vdots \\ \psi_1(\underline{x}^n) - \psi_1(\underline{x}^0) & \cdots & \psi_j(\underline{x}^n) - \psi_j(\underline{x}^0) & \cdots & \psi_n(\underline{x}^n) - \psi_n(\underline{x}^0) \end{bmatrix}$$

Rubin (in practice)

System to be solved

$$\underline{\psi}(\underline{x}) - \underline{\psi}\left(\underline{x}^k\right) = \left\{ {}^t\underline{J}\left[\underline{\psi}\left(\underline{x}^k\right)\right]\right\}\underline{\Delta x}^k$$

$$ {}^t\underline{B} = {}^t\underline{J}\ {}^t\underline{A}$$

and

$$ou \quad {}^t\underline{B}\ {}^t\underline{A}^{-1} = {}^t\underline{J}$$

$$d'où \quad \underline{J} = \underline{A}^{-1}\ \underline{B}$$

After n iterations J can be evaluated and Newton-Raphson can be applied to calculate the new point.

$$\underline{\tilde{x}}^{n+1} = \underline{x}^n - \left\{\underline{E} - {}^t\left(\underline{A}^{-1}\ \underline{B}\right)\right\}^{-1}\left[\underline{x}^n - \underline{\psi}\left(\underline{x}^n\right)\right]$$

From this point the **J** matrix can be updated at each step.
eliminate the worst point in the list and then reevaluate **J**

Updating the matrix

- As calculating Jacobian matrix is expensive, the idea is to preserve the validity of the inverted matrix for several steps : i.e.

$$\underline{\tilde{\mathrm{x}}}^{n+1} = \underline{\mathrm{x}}^{n} - \{\underline{\mathrm{E}} - (\underline{\mathrm{A}}^{k_n})^{-1}\underline{\mathrm{B}}^{k_n}\}^{-1}(\underline{\mathrm{x}}^{n} - \psi(\underline{\mathrm{x}}^{n}))$$

  - k_n is the evaluation k of the Jacobian matrix used at iteration n
  - the validity of the step is obtained by comparing the x obtained and the expected value (is the difference decreasing ?)

Conclusions : Solving non linear equations

- Iterative procedure when no mathematical formulation is found
  - Does not always converge !
  - Bounds on variables + safe guards
- Initialisation point : use ranges of x
  - bad initialisation point means long calculations
- Direction (derivative) + Step length (relaxation)
  - Derivatives : defines direction
    - Numerical calculations => length of Δx
    - Numerical calculation cost + noise
  - Matrix inversion : defines the direction
    - Try to reuse direction for several steps
  - Relaxation : if divergence observed
- Test of convergence
  - on both equations and variables
  - one criteria for several equations+ variables => scaling