Sistemi Non Lineari

Obiettivi di apprendimento

- Strategia di risoluzione di un’equazione non-lineare
- Metodo auto-coerente
- Metodo dicotomico
- Metodo di Newton
- Metodo di quasi-Newton per risolvere sistemi di equazioni non-lineari
- Utilizzo del modulo `scipy.optimize` per la soluzione numerica di sistemi non-lineari con Python.

Esempi


Calcolo degli zeri di una funzione
SISTEMI NON LINEARI IN CHIMICA
Bilanci simultanei di materia ed energia in processi industriali
Determinazione dello stato stazionario di un sistema cinetico
Studio di equilibri multipli in soluzione

Un’equazione non lineare verrà in genere scritta in questo modo

$$f(x) = 0$$

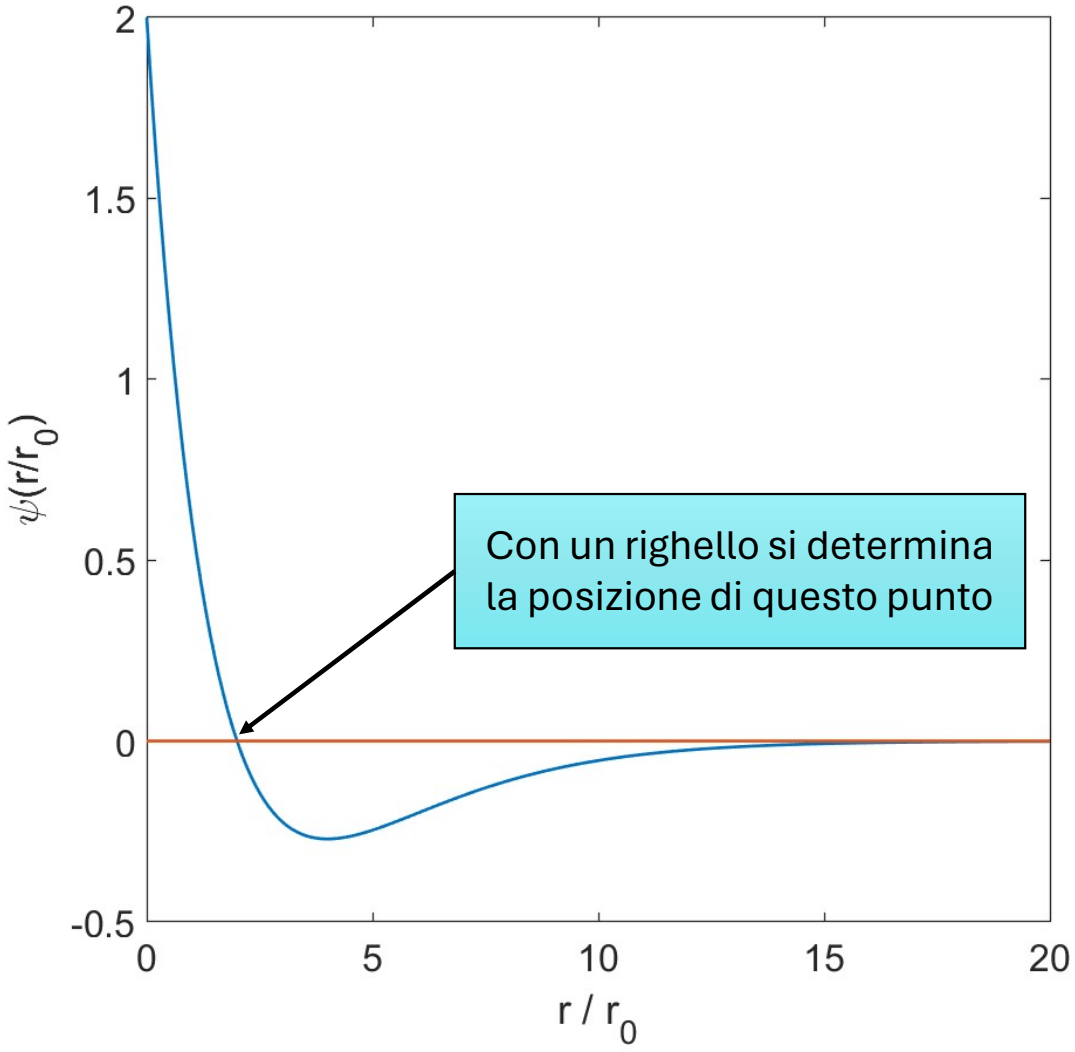
Qualche esempio:

$pV_m - RT = 0$	Equazione dei gas in cui pressione e volume sono variabili
$(2 - r)^2 e^{-r} - a = 0$	Densità di probabilità radiale di un orbitale 2s pari ad a
$K_{\mathrm{eq}} - e^{-\Delta_{\mathrm{r}} G^{\ominus}/RT} = 0$	Costante di equilibrio in funzione della temperatura

Risoluzione grafica

Calcolare i nodi radiali dell'orbitale 2s

$\psi(r) \propto (2 - r)e^{-r/2} = 0$



La ricerca delle soluzioni di un'equazione non lineare scritta in questo modo

$$f(x) = 0$$

si riconduce al problema di calcolare le radici della funzione f.

Nella soluzione del problema, si deve tener conto che:

1. le radici possono essere più di una; ad esempio, se $f(x)$ è un polinomio di grado n in x, allora esistono n radici
2. le radici possono essere numeri complessi
3. non esiste, in genere, una soluzione analitica, per cui le radici saranno calcolate entro una dato errore

Guess iniziale: in genere la radice viene cercata a partire da un dato un punto del dominio scelto in maniera tale che permetta di raggiungere quella soluzione.

Sebbene sembri un controsenso, in genere in problemi in cui la soluzione ha un significato fisico, sono le sue proprietà a guidare la scelta del *guess* iniziale. Altrimenti, si deve esplorare tutto il dominio.

In problemi chimici, in cui le incognite di equazioni non lineari sono quantità osservabili (concentrazioni, velocità di reazione, volumi, tempi, …) è ovvio che la soluzione deve essere cercata tra quelle reali.

Un'ulteriore restrizione sul sottodominio in cui cercare la soluzione deriva dal segno atteso della soluzione.

Infine, in problemi di natura chimica, è spesso possibile avere una stima iniziale, anche se molto grezza, della soluzione. Ad esempio, se il problema è quello di calcolare il volume di un gas, date temperatura e pressione, secondo il modello di Van der Waals, un ottimo *guess* iniziale sarà l'intorno del volume calcolato come se il gas fosse ideale.

Criterio di convergenza: nella ricerca della soluzione viene prodotto, iterativamente, un numero candidato a essere radice. Allo step k-esimo del ciclo si deve, quindi, controllare che x_k sia la soluzione. Sembrerebbe naturale scegliere di controllare se $f(x_k) = 0$.

Il problema, però, è che non è detto a priori che la radice sia rappresentabile nel computer e/o che non vi siano errori di arrotondamento. Di conseguenza, la condizione $f(x_k) = 0$ potrebbe non presentarsi mai. In genere il criterio di convergenza usato è $|\boldsymbol{f(x_k)}| < \boldsymbol{\epsilon}$, dove ϵ è un numero piccolo a piacere, che determina la precisione con cui si conoscerà la soluzione.

Spesso occorre determinare quando il calcolo **non** sta andando a convergenza. In questo caso si può controllare la variazione sulla soluzione. Se $|x_k - x_{k-1}| < \epsilon$, ma $|f(x_k)| > \epsilon$, allora il calcolo non sta convergendo e conviene usare un *guess* iniziale differente, o aumentare ϵ.

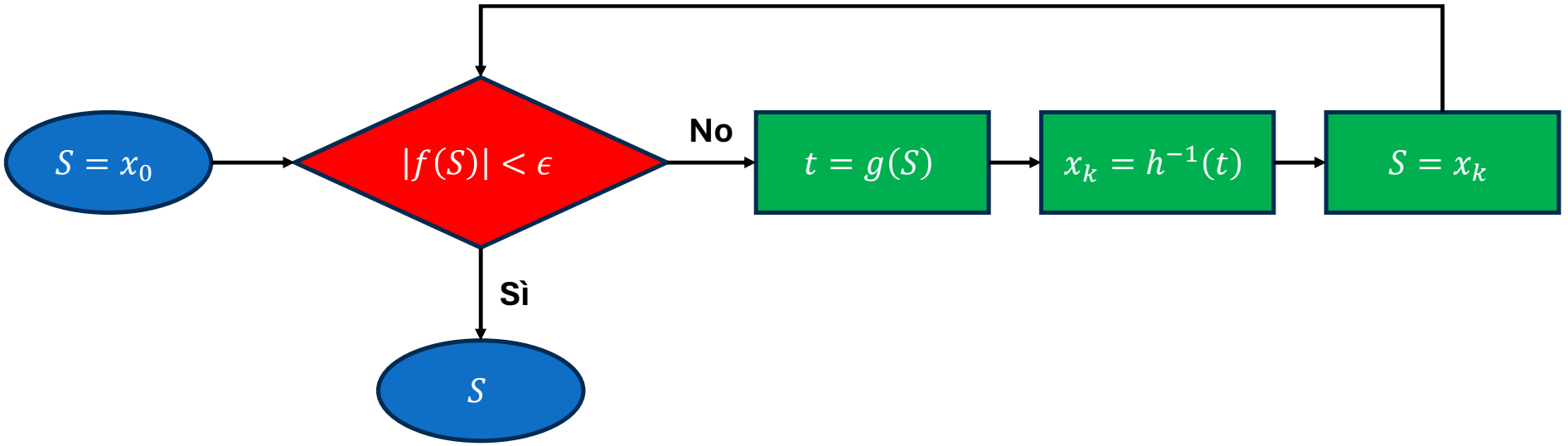
In genere viene stabilito anche un **numero massimo di step del ciclo**, per evitare che il calcolo continui indefinitamente in caso vi sia una continua oscillazione tra due o più valori che non sono riconosciuti come soluzione secondo il criterio di convergenza. Il numero massimo di step viene impostato ad un valore in cui, generalmente, ci si aspetta che la soluzione arrivi a convergenza. Questo significa che se si supera tale numero di step, è probabile che il *guess* iniziale non sia molto buono.

Ciclo auto-coerente

Se si può partizionare la funzione in questo modo: $f(x) = h(x) - g(x)$ in modo tale che:

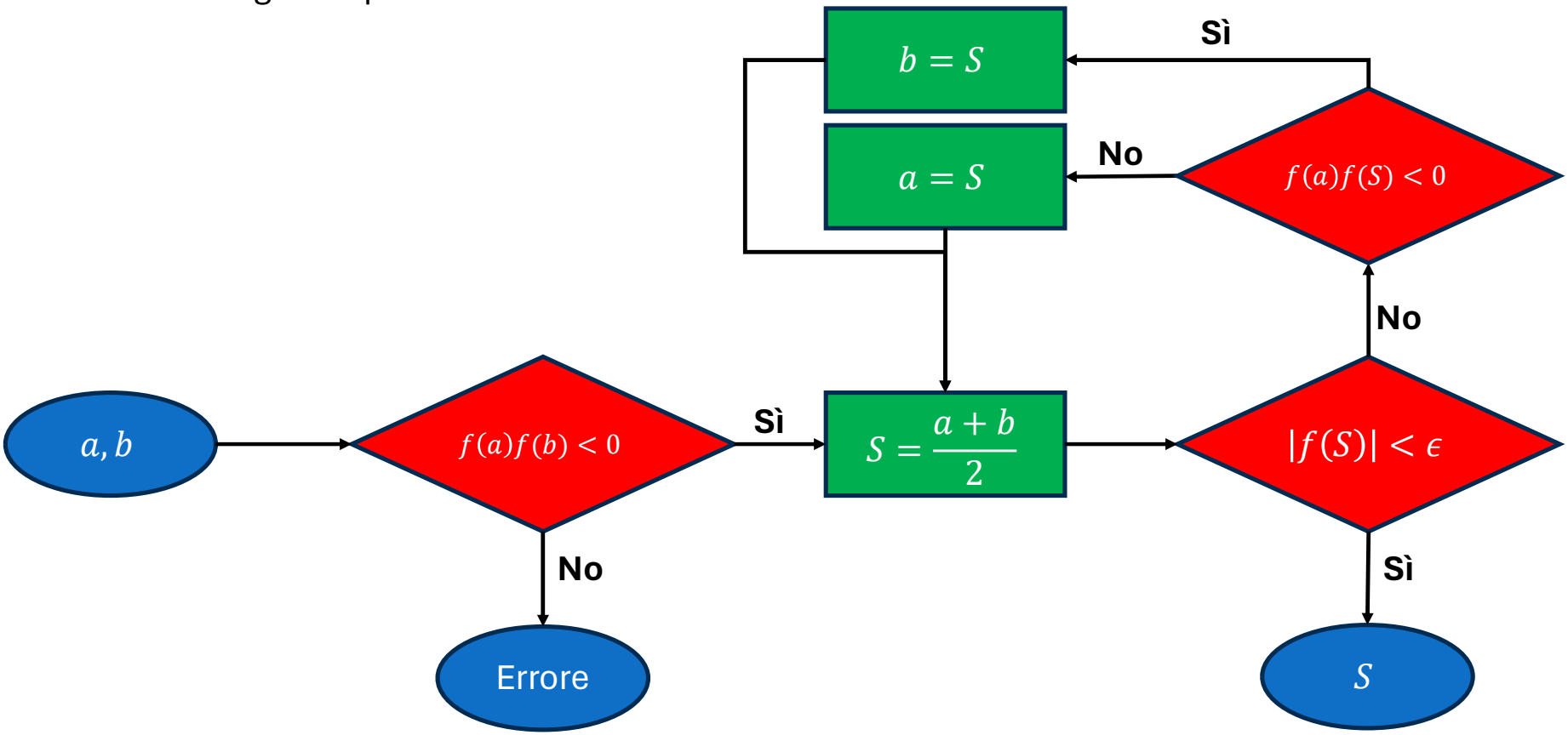
- $h(x)$ sia invertibile e sia facile calcolare $x = h^{-1}(x)$
- $|dg/dx| < 1$ in un intorno della soluzione

allora la soluzione, S, può essere calcolata a partire da un *guess* iniziale x_0 tramite questo ciclo



Metodo dicotomico

Supponiamo di aver individuato un intervallo $[a, b]$ del dominio della funzione in cui la stessa passi per lo zero una sola volta. Quindi, la soluzione S per cui $|f(S)| < \epsilon$ è nell'intervallo, ossia $S \in]a, b[$. Il metodo dicotomico è il metodo più robusto per trovare S con certezza e si basa sul concetto di *divide & conquer*: si divide l'intervallo in due sotto-intervalli. Si deduce in quale dei due continua a essere contenuta S e si procede dividendo ancora. Il metodo converge sempre.



Metodo di Newton

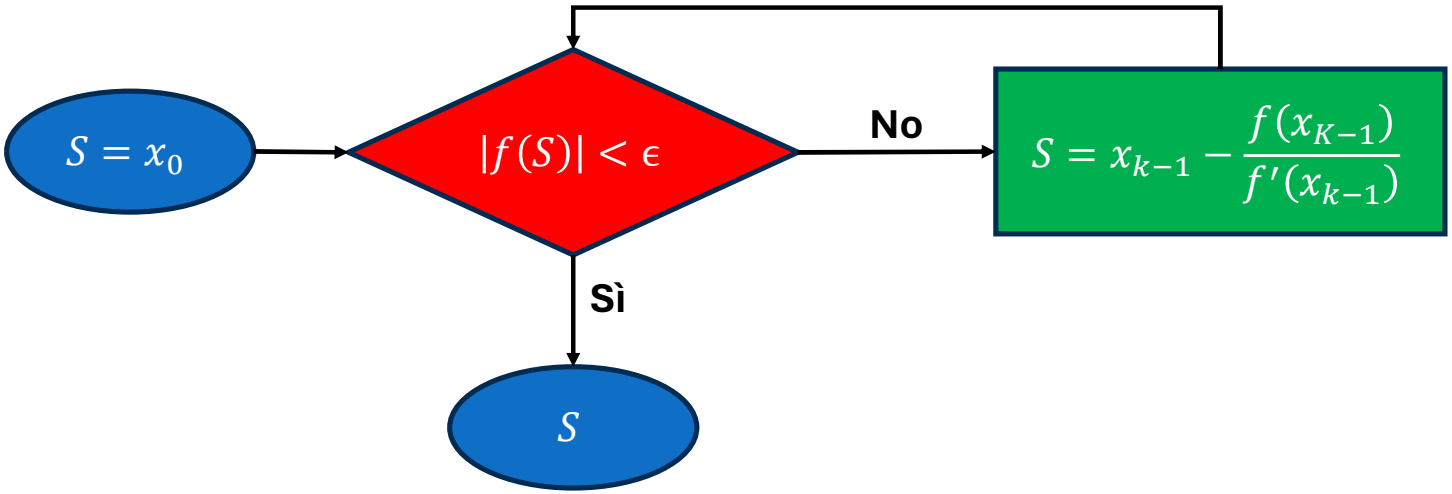
Supponiamo che la soluzione giaccia nell'intervallo $[a, b]$ del dominio della funzione, che in tale intervallo la funzione sia continua e derivabile; inoltre, la derivata della funzione attorno alla soluzione sia ben diversa da 0.

Sia x_k l'approssimazione della soluzione allo step k. Il prossimo passo sarà quello di muoversi nella direzione di variazione della funzione di una quantità δ_k sufficiente piccola tale per cui

$$f(x_{k+1}) = f(x_k + \delta_k) \approx f(x_k) + f'(x_k)\delta_k$$

Se x_{k+1} fosse la soluzione, allora

$$\delta_k = -f(x_k)/f'(x_k)$$



Metodo della *regula falsi*

Il metodo di Newton implica di conoscere l'espressione analitica per la derivata $f'(x)$.

Se tale forma analitica non è ottenibile: espressione troppo complicata, oppure banalmente la funzione non è nota a priori, ci sono due possibili strade.

Se la funzione da derivare non è eccessivamente complessa, si può utilizzare un algoritmo di derivazione automatica (AD – *automatic differentiation*) che, tramite la regola della catena delle derivate, permette di calcolare in maniera esatta la derivata di una qualsiasi funzione in un dato punto.

In alternativa, la derivata viene calcolata in maniera numerica. Nel metodo della *regula falsi*, si usa lo schema all'indietro per la stima numerica della derivata. Quindi

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

La nuova stima della soluzione si calcola come

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = \approx x_k - \frac{f(x_k)}{f(x_k) - f(x_{k-1})}(x_k - x_{k-1})$$

Sistemi di equazioni non lineari

La soluzione di un sistema di equazioni non lineari si trova generalizzando il metodo di Newton.

Esempio. Una soluzione acquosa viene preparata a 298.15 K sciogliendo 0.01 mol di nitrato d'ammonio in 1 L d'acqua. La costante basica per l'ammoniaca è $1.8 \cdot 10^{-5}$. Qual è la composizione della soluzione all'equilibrio?

Incognite: $[\mathrm{NH_4^+}]$, $[\mathrm{NO_3^-}]$, $[\mathrm{NH_3}]$, $[\mathrm{H_3O^+}]$, $[\mathrm{OH^-}]$. Si assume il sale completamente dissociato.

Per risolvere il problema si imposta il sistema in 5 equazioni:

$$\begin{cases} [\mathrm{NO_3^-}] = 0.01 \\ [\mathrm{NH_4^+}] + [\mathrm{NH_3}] = 0.01 \\ [\mathrm{H_3O^+}] + [\mathrm{NH_4^+}] = [\mathrm{NO_3^-}] + [\mathrm{OH^-}] \\ [\mathrm{H_3O^+}][\mathrm{OH^-}] = 10^{-1} \\ \dfrac{[\mathrm{NH_4^+}][\mathrm{OH^-}]}{[\mathrm{NH_3}]} = 1.8 \cdot 10^{-5} \end{cases}$$

Sistemi di equazioni non lineari

Eliminiamo la prima equazione, e chiamiamo le incognite rimanenti: $x_1 = [\mathrm{NH_4^+}]$, $x_2 = [\mathrm{NH_3}]$, $x_3 = [\mathrm{H_3O^+}]$, $x_4 = [\mathrm{OH^-}]$. Inoltre, sistemiamo le equazioni in maniera tale da avere $f_j(\mathbf{x}) = 0$. Il sistema diventa

$$\begin{cases} x_1 + x_2 - 0.01 = 0 \\ x_3 + x_1 - x_4 - 0.01 = 0 \\ x_3 x_4 - 10^{-14} = 0 \\ x_1 x_4 - 1.8 \cdot 10^{-5} x_2 = 0 \end{cases}$$

Raccogliendo le quattro equazioni in un vettore $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x})\ \ f_2(\mathbf{x})\ \ f_3(\mathbf{x})\ \ f_4(\mathbf{x})\ ]^{\mathrm{tr}}$, la versione generalizzata a più dimensioni del metodo di Newton è

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J}^{-1}(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)$$

Dove $\mathbf{J}(\mathbf{x})$ è la matrice Jacobiana:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_2} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Sistemi di equazioni non lineari

Nell'esempio che si sta trattando

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} 1 & 1 & 0 & x_4 \\ 1 & 0 & 0 & -1.8 \cdot 10^{-5} \\ 0 & 1 & x_4 & 0 \\ 0 & -1 & x_3 & x_1 \end{bmatrix}$$

Dato un *guess* iniziale $\mathbf{x}_0 = \left[x_1^{(0)} \; x_2^{(0)} \; x_3^{(0)} \; x_4^{(0)}\right]^{\mathrm{tr}}$ si può risolvere il problema. Ad esempio, si può usare il metodo di Gauss-Jordan per calcolare l'inversa di $\mathbf{J}(\mathbf{x})$ e propagare la soluzione di Newton finché per tutte le equazioni $\left|f_j(\mathbf{x})\right| < \epsilon$.

Algoritmo:

1. Dato $\mathbf{x}_0$ costruisco $\mathbf{J}(\mathbf{x}_0)$
2. Costruisco la matrice associata $\mathbf{A}(\mathbf{x}_0) = (\mathbf{J}(\mathbf{x}_0)|\mathbf{I})$
3. Calcolo l'inversa con il metodo di Gauss-Jordan: $\left(\mathbf{I}|\mathbf{J}^{-1}(\mathbf{x}_0)\right)$
4. Aggiorno la soluzione con il metodo di Newton: $\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{J}^{-1}(\mathbf{x}_0)\mathbf{f}(\mathbf{x}_0)$
5. Se $\left|f_j(\mathbf{x}_1)\right| < \epsilon$, allora $\mathbf{x}_1$ è la soluzione, altrimenti assegno $\mathbf{x}_0 = \mathbf{x}_1$ e torno al punto 2.

Sistemi di equazioni non lineari

E se non sapessi come calcolare la matrice Jacobiana?

Similmente al metodo della *regula falsi*, la matrice Jacobiana viene calcolata in maniera numerica. In questo caso si usa, quindi, una *approssimazione* di **J**. Questo metodo viene detto: **quasi-Newton**.

Un tipico esempio è l'utilizzo di approcci alle differenze finite per il calcolo delle derivate parziali.

Python time

Il modulo `optimize` di `scipy` ha il metodo `fsolve` per risolvere sistemi di equazioni non lineari.

```
s = scipy.optimize.fsolve(f, x0)
```

assegna alla variabile `s` la soluzione del sistema data una funzione `f`, che implementa il calcolo delle equazioni del sistema, e dato il *guess* iniziale `x0`.

Altri metodi sono `root` e `newton_Krylov`. Utilizzano diversi modi per approssimare la matrice Jacobiana.

Team work

Costituire gruppi di lavoro di 4 – 5 persone per risolvere gli esercizi presenti nel Jupyter Notebook:

1. Calcolo del volume molare della CO_2 date temperatura e pressione, e dati i parametri di Van der Waals usando il ciclo auto-coerente.
2. Calcolo del volume molare della CO_2 date temperatura e pressione, e dati i parametri di Van der Waals usando il metodo dicotomico.
3. Calcolo del volume molare della CO_2 date temperatura e pressione, e dati i parametri di Van der Waals usando il metodo di Newton.
4. Calcolo del pH di una soluzione di acido debole monoprotico usando il metodo `scipy.optimize.fsolve`.