Sistemi Non-Lineari

May 15, 2024

1 Introduzione

Quando in un sistema una o più equazioni sono non-lineari almeno in una delle variabili, allora si parla di sistemi di equazioni non-lineari. In genere la soluzione di equazioni e di sistemi di equazioni non linerari si imposta riscrivendo tutte le funzioni nella forma:

$$f_j(x_1, x_2, ...) = 0 \tag{1}$$

con j che va da 1 al numero di equazioni.

Raccogliendo tutte le variabili in un array $\mathbf{x} = [x_1\, x_2\, ...]^{\text{tr}}$, il probelma è quello di cercare il vettore (o i vettori) che siano "radici" delle equazioni del sistema, ossia gli zeri comuni delle funzioni.

Nel caso di una equazione in una variabile, il problema in genere può essere risolto facilmente e in maniera anche accurata. In casi multidimensionali e/o con più equazioni, la risoluzione diventa più complessa.

Esempio 1. Studio di equilibri multipli

In un sistema reattivo in cui siano presenti due o più processi stechiometrici indipendenti la condizione di equilibrio viene determinata dai vincoli imposti delle costanti di equilibrio e dai bilanci di materia, di carica e stechiometrico. Tali vincoli si esprimono come un set di equazioni non-lineari accoppiate delle concentrazioni delle specie chimiche del sistema.

Esempio 2. Studio dello stato stazionario

Dato uno schema cinetico, o un set di equazioni alle derivate parziali che esprimono la dipendenza temporale di un sistema, si può determinare lo stato stazionario di tale sistema imponendo che tutte le derivate temporali si annullino e risolvendo il sistema di equazioni non lineari.

Esempio 3. Bilanci di materia e energia in processi industriali

Quando si deve impostare un bilancio di materia e di energia all'interno di un processo industriale in stato stazionario è comune imbattersi in un sistema di equazioni non lineari. Lo stesso vale in problemi di ottimizzazione dei processi.

Esempio 4. Calcolo degli zeri di una funzione

Il calcolo dei nodi di un orbitale atomico o molecolare, il calcolo degli estremi di una funzione, il calcolo del valore della variabile per cui una sua funzione ha un valore specifico sono tutti problemi di calcolo delle radici di una funzione non-lineare.

La soluzione di un'equazione nonlineare tramite quadrature (ossia con un numero finito di operazioni elementari e radicali) è in genere impossibile. Si conoscono formule di quadratura solo per polinomi

fino al grado quinto. In caso di polinomi di ordine superiore, oppure in caso di funzioni differenti, in cui appaiono anche esponenziali e logaritmi, si deve procedere alla ricerca di soluzioni *numeriche* al problema.

2 Funzioni di una variabile

La soluzione di equazioni di una variabile si basa in genere su metodi iterativi, nei quali partendo da una stima (*guess*) iniziale della soluzione, questa viene affinata fino a raggiungere la precisione desiderata (se si riesce).

2.1 Ciclo auto-coerente

Nel caso di funzioni scrivibili come $f(x) = g(x) - h(x)$, con $h(x)$ una funzione per la quale si conosce l'inversa, e quindi si può calcolare $x = h^{-1}(x)$, allora si può impostare la ricerca della soluzione di $f(x) = 0$ attraverso un ciclo auto-coerente di questo tipo. Sia $x_1 = a$ un *guess* iniziale per la soluzione, allora si procede in questo modo:

1. $x_1 = a$
2. $x_2 = h^{-1}(g(x_1))$
3. torna allo step 2

Quindi, se con $k = 1, 2, ...$ si conta il numero di iterazioni, allora la $k+1$-esima stima della soluzione sarà data da $x_{k+1} = h^{-1}(g(x_k))$ sotto la condizione che $x_1 = a$. Se la funzione $g(x)$ ha una derivata minore di 1 (in valore assoluto) in un intorno della soluzione $x = x^*$ e il *guess* iniziale, a viene scelto in tale intorno, allora la successione di operazioni convergerà a x^*. Come in tutti i metodi iterattivi, la soluzione che si trova dipende dal *guess* iniziale, che quindi deve essere fatto con attenzione.

Questo algoritmo tende spesso a fallire, per cui servono altri approcci.

Esempio. Qual è il volume molare dell'anidride carbonica a 1 atm e 293.15 K se lo stato del gas viene descritto dal modello di Van der Waals, con $a = 3.64 \cdot 10^{-6}$ m6 bar / mol, e $b = 4.267 \cdot 10^{-5}$ m3 / mol?

L'equazione di Van der Waals riscritta come equazione nel volume molare, V_m, è

$$V_m^3 - \frac{RT + pb}{p} V_m^2 + \frac{a}{p} V_m - \frac{ab}{p} = 0 \tag{2}$$

Che può essere riscritta, tra gli alti modi, come

$$h(V_m) = V_m^3 = \frac{RT + pb}{p} V_m^2 - \frac{a}{p} V_m + \frac{ab}{p} = g(V_m) \tag{3}$$

Il codice che segue risolve l'equazione con il metodo auto-coerente, partendo dal volume molare valutato con l'equazione dei gas ideali quale *guess* iniziale.

[1]:
```
import numpy as np
```

```
T = 293.15 # K
p = 101325 # Pa
R = 8.314 # J / (mol K)

a = 3.64e-6 * 1.0e5 # m^6 Pa / mol
b = 4.267e-5 # m^3 / mol

# Guess iniziale: volume molare gas ideale
V1 = R * T / p
print('Guess iniziale: ',str(V1*1000),' L/mol\n')

# Per evitare cicli infiniti, si impone un numero massimo di iterazioni
nmax = 200   # max iterations

# Come condizione di convergenza si utilizzerà la differenza relativa
# tra la soluione allo step i e quella allo step i-1. Tale differenza
# relativa deve essere sotto la tolleranza che segue
tol = 1.0e-8

itol = 0
for i in range (0, nmax):
  g = ((R * T + p * b) / p) * V1 * V1 - (a / p) * V1 + (a * b) / p
  V2 = g**(1./3.)
  dV = (V2 - V1) / V2
  if (np.abs(dV) <= tol):
    itol = 1
    break
  V1 = V2

print('Volume molare stimato: ',str(V2*1e3),' L/mol\n')
print('Condizione di uscita [0: nmax raggiunto, non convergenza | 1:␣
 ↪convergenza entro tol]: ', str(itol))
```

```
Guess iniziale:  24.053778435726624  L/mol

Volume molare stimato:  23.946699626679163  L/mol

Condizione di uscita [0: nmax raggiunto, non convergenza | 1: convergenza entro
tol]:  1
```

2.2 Metodo dicotomico

Il metodo dicotomico (o metodo di bisezione) è un metodo molto robusto, per quanto poco efficiente in termini di tempo di calcolo. L'approccio si basa sull'ipotesi di conoscere un intervallo $[a, b]$ entro il quale sia contenuta una (e una sola) radice. Sia $x^* \in [a, b]$ tale la radice (ossia, $f(x^*) = 0$), allora deve essere vero che $f(a)f(b) < 0$.

Il metodo dicotomico implementa questa successione di operazioni: 1. siano dati $x_-^{(1)} = a$ e $x_+^{(1)} =$

b 2. si definisca $\bar{x}_1 = \dfrac{x_-^{(1)} + x_+^{(1)}}{2}$ 3. si scelgano i nuovi punti $x_-^{(2)}, x_+^{(2)}$ secondo la regola: se $f(x_-^{(1)})f(\bar{x}_1) < 0$ allora $x_-^{(2)} = x_-^{(1)}$ e $x_+^{(2)} = \bar{x}_1$, altrimenti se $f(\bar{x}_1)f(x_+^{(1)}) < 0$ allora $x_-^{(2)} = \bar{x}_1$ e $x_+^{(2)} = x_+^{(1)}$ 4. si torni allo step 2

La successione $\bar{x}_1$, $\bar{x}_2$, ... converge *sempre* alla radice x^*. La soluzione si trova quando $|f(\bar{x}_k)| < \epsilon$, dove ϵ è la tolleranza scelta per la soluzione numerica dell'equazione.

Esercizio. Si risolva il problema della CO_2 di sopra, con il metodo dicotomico.

[2]:
```
import numpy as np

# Funzione che calcola l'espressione per l'equazione di stato del gas
# di van der waals
def vdw(V, T, p, a, b, R):
  f = V**3 - ((R * T + p * b) / p) * V**2 + (a / p) * V - (a * b) / p
  return f


T = 293.15 # K
p = 101325 # Pa
R = 8.314 # J / (mol K)

a = 3.64e-6 * 1.0e5 # m^6 Pa / mol
b = 4.267e-5 # m^3 / mol

V1 = R * T / p

# Cerca Vm1 e Vp1 attorno a V1 finché f(Vm1)*f(Vp1) < 0
deltaV = 0.1 * V1
Vm1 = V1 - deltaV
Vp1 = V1 + deltaV
fm1 = vdw(Vm1, T, p, a, b, R)
fp1 = vdw(Vp1, T, p, a, b, R)
while (fm1 * fp1 > 0):
  Vm1 = Vm1 - deltaV
  Vp1 = Vp1 + deltaV
  fm1 = vdw(Vm1, T, p, a, b, R)
  fp1 = vdw(Vp1, T, p, a, b, R)

print('Intervallo entro cui si cercherà la soluzione: [',str(Vm1*1000),', ',␣
 ↪str(Vp1*1000),'] L/mol\n')

# Numero massimo di iterazioni
nmax = 200

# Tolleranza per la convergenza sulla funzione
```

```
tol = 1.0e-8 # tolleranza

itol = 0
for i in range (0, nmax):
  V2 = 0.5 * (Vm1 + Vp1)
  f2 = vdw(V2, T, p, a, b, R)
  if (np.abs(f2) <= tol):
    itol = 1
    break
  elif (fm1 * f2 < 0):
    Vp1 = V2
    fp1 = f2
  else:
    Vm1 = V2
    fm1 = f2

print('Volume molare stimato: ',str(V2*1e3),' L/mol\n')
print('Condizione di uscita [0: nmax raggiunto, non convergenza | 1:␣
 ↪convergenza entro tol]: ', str(itol))
```

```
Intervallo entro cui si cercherà la soluzione: [ 21.64840059215396 ,
26.459156279299286 ] L/mol

Volume molare stimato:  23.94102634930915  L/mol

Condizione di uscita [0: nmax raggiunto, non convergenza | 1: convergenza entro
tol]:  1
```

2.3 Metodi di Newton e *regula falsi*

Questi metodi si basano sulla generazione di successive approssimazioni della soluzione. Data la stima x_k, si genera la nuova approssimazione come $x_{k+1} = x_k + \delta_k$, ipotizzando che $f(x_k + \delta_k) = 0$. Se δ_k è sufficientemente piccolo da poter scrivere:

$$f(x_k + \delta_k) = f(x_k) + \delta_k f'(x_k) = 0 \tag{4}$$

con $f'(x_k) = (df(x)/dx)_{|x=x_k}$, allora la scelta per l'incremento è $\delta_k = -f(x_k)/f'(x_k)$.

L'algortimo, quindi, è il seguente: 1. sia dato un *guess* iniziale $x_1 = a$ 2. si definisce $x_2 = x_1 - \dfrac{f(x_1)}{f'(x_1)}$

3. si torna allo step 2

Tale ciclo continua finché $|f(x_k)| < \epsilon$, con ϵ la tolleranza scelta per determinare la convergenza della soluzione.

Il metodo di Newton, se converge, ha una convergenza di tipo *quadratico*, quindi più rapido rispetto al metodo dicotomico. D'altra parte, a differenza del metodo dicotomico, non è assicurata la convergenza. In particolare, tolto il caso di un *guess* iniziale mal definito, il metodo fallisce se la

derivata è vicina a zero o se la funzione non è continua nell'intervallo compreso tra il *guess* iniziale e la radice.

Il calcolo con il metodo di Newton necessita la valutazione della derivata della funzione. Se questa non è nota a priori, bisogna calcolare la derivata in maniera numerica. La *regula falsi* (o metodo della secante variabile) prevede che la devivata sia approssimata secondo il metodo *backward*, ossia $f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$. Tale metodo converge in maniera superlineare, ossia più rapidamente del metodo dicotomico, ma meno rapidamente di quello di Newton.

Esempio. Risolvere il problema della CO_2 di sopra, usando il metodo di Newton.

[3]:
```
# Calcolo della derivata della funzione di cui trovare la soluzione
def dvdw(V, T, p, a, b, R):
  df = 3*V**2 - ((R * T + p * b) / p) * 2 * V + (a / p)
  return df


T = 293.15 # K
p = 101325 # Pa
R = 8.314 # J / (mol K)

a = 3.64e-6 * 1.0e5 # m^6 Pa / mol
b = 4.267e-5 # m^3 / mol

# Guess iniziale
V1 = R * T / p

nmax = 200
tol = 1.0e-8

itol = 0
for i in range (0, nmax):
  V1 = V1 - vdw(V1, T, p, a, b, R) / dvdw(V1, T, p, a, b, R)
  if (np.abs(vdw(V1, T, p, a, b, R)) <= tol):
    itol = 1
    break

print('Volume molare stimato: ',str(V1*1e3),' L/mol\n')
print('Condizione di uscita [0: nmax raggiunto, non convergenza | 1:␣
 ↪convergenza entro tol]: ', str(itol))
```

```
Volume molare stimato:  23.947647087911687  L/mol

Condizione di uscita [0: nmax raggiunto, non convergenza | 1: convergenza entro
tol]:  1
```

3 Funzioni di più variabili

La risoluzione di un sistema di equazioni non-lineare ha una complessità molto maggiore rispetto alla risoluzione di una funzione in una variabile. Si cita qui una estensione in più dimensioni della *regula falsi*. In particolare, dato un sistema di n equazioni non-lineari in n variabili, il metodo *quasi-Newton* prevede di costruire la sequenza di approssimazioni successive della soluzione come

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J}^{-1}(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k) \tag{5}$$

dove $\mathbf{f}(\mathbf{x}_k)$ è una matrice colonna con tutti i valori delle funzioni nel punto $\mathbf{x}_k$, mentre $\mathbf{J}$ è la matrice Jacobiana

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_2} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{6}$$

Il calcolo della matrice Jacobiana richiede di conoscere tutte le derivate parziali di tutte le funzioni rispetto a tutte le variabili. Se tali derivate non sono note, la matrice Jacobiana viene approssimata, ad esempio con metodi alle differenze finite.

4 Implementazione in Python

Il modulo `optimize` di `scipy` implementa diversi metodi di risoluzione di equazioni non lineari, tra i quali `fsolve`, `root`, e `newton_krylov`.

Esercizio. Utilizzo di `scipy.optimize.fsolve` per calcolare la concentrazione delle specie all'equilibrio in una soluzione acquosa di un acido debole monoprotico, HA, a 298.15 K, e pressione standard, date la concentrazione analitica C_0 e la costante acida k_a.

N.B.: tra le specie all'equilibrio c'è anche $\mathrm{OH^-}$.

[10]:
```
from scipy.optimize import fsolve
import numpy as np

# Implementazione delle equazioni del sistema
# (costanti di equilibrio, bilanci di materia, bilancio di carica)
# x1 = HA, x2 = A-, x3 = H3O+, x4 = OH-
def equations(vars, args):
    x1, x2, x3, x4 = vars
    eq1 = x1 + x2 - args[0]
    eq2 = x2 * x3 - x1 * args[1]
    eq3 = x3 - x2 - x4
    eq4 = x3 * x4 - args[2]
```

```
    return [eq1, eq2, eq3, eq4]

# Concentrazione dell'acido
C0 = 1.0e-3

# Costante acida
ka = 1.0e-5

# Termini noti nel sistema (che dipendono da C0 e/o ka)
p1 = C0
p2 = ka
p3 = 1.0e-14
args = [p1, p2, p3]

# Guess iniziale
initial_guess = [C0, C0, C0, 1.0e-14/C0]

# Calcola la soluzione con fsolve
solution = fsolve(equations, initial_guess, args, epsfcn=1.0e-15)

print("Soluzioe:", solution)
print("pH = ", -np.log10(solution[2]))
print("Soluzione approssimata = ",-np.log10(np.sqrt(C0*ka)))
```

```
Soluzioe: [9.04875128e-04 9.51248720e-05 9.51249772e-05 1.05124851e-10]
pH =  4.021705434498702
Soluzione approssimata =  4.0
```

5 Approfondimento: Ottimizzazione

I problemi di ottimizzazione sono quelli in cui si devono determinare i migliori parametri di una data funzione affinché si raggiunga uno specifico obiettivo, che viene formulato nella ricerca di un estremo, o nella ricerca di uno zero.

Alcuni esempi:

- trovare i parametri della retta che meglio interpola la misura dell'inverso della costante cinetica contro l'inverso della temperatura in una reazione chimica
- trovare i valori dei flussi in entrata e uscita a un processo industriale per minimizzare la perdita di una certa sostanza
- determinare i coefficienti dell'espansione della funzione d'onda di stato fondamentale della molecola in esame su orbitali atomici
- determinare le condizioni di temperatura e pH che massimizzano la resa di una certa reazione chimica

In tutti questi esempi, si può definire una funzione $f(\mathbf{x}|\mathbf{p})$ che dipende da un certo numero di variabili, $\mathbf{x}$, ed è parametrica in un certo numero di parametri, $\mathbf{p}$.

Ad esempio, un problema ai minimi quadrati si riscontra quando i parametri devono essere tali che dato un insieme di N misure sperimentali $f_j^{\exp}$, si minimizza la funzione

$$\chi^2 = \frac{1}{2}\sum_{j=1}^{N}\left(f_j^{\exp} - f(\mathbf{x}_j|\mathbf{p})\right)^2 = \sum_{j=1}^{N} F_j(\mathbf{p})^2 \tag{7}$$

con $F_j(\mathbf{p}) = f_j^{\exp} - f(\mathbf{x}_j|\mathbf{p})$. Tale problema si chiama: problema agli scarti quadratici non lineari (*nonlinear least squares*).

Minimizzare χ^2 significa cercare i valori dei parametri per cui il gradiente è nullo: $\mathbf{J}^{tr}\mathbf{F} = \mathbf{0}$, con $\mathbf{F} = [F_1, F_2, ...]^{\text{tr}}$ e $J_{i,j} = \partial F_i/\partial p_j$. Si tratta di un sistema di equazioni nonlineari, che può essere risolto con il metodo di Newton descritto sopra.

Si noti che, in questo caso, l'applicazione del metodo di Newton implica dover calcolare non solo lo Jacobino, ma anche l'Hessiano di χ^2.