A.D. 1308
unipg
DIPARTIMENTO
DI FISICA E GEOLOGIA
DIPARTIMENTO DI ECCELLENZA
MUR 2023/2027

METODI COMPUTAZIONALI PER LA FISICA

Equazioni e Minimizzazione

S. Germani - stefano.germani@unipg.it

SOMMARIO

- Equazioni Lineari
- Equazioni Non Lineari
  - Metodo del Rilassamento
  - Metodo della Bisezione
  - Metodo della Secante
- Minimizzazione
  - Regressione Lineare
  - Metodo di Gauss Newton
  - SciPy optimize
  - pyROOT
  - iminuit

EQUAZIONI LINEARI

Esistono metodi per risolvere sistemi di equazioni lineari in maniera automatica

$$A\,x = v$$

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix},$$

$$x = \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix}$$

$$v = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Per una breve trattazione matematica del metodo utilizzabile (fattorizzazione LU) vedere il Notebook relativo a questa lezione o il libro consigliato.

NUMPY / SCIPY LINALG

$$A\,x = v$$

```
# Matrice A corrispondente ai coefficienti delle equazioni
A = np.array([ [ 2, 2, 3,-1],
               [ 3, 4,-1, 1],
               [ 1, 3, 4, 2],
               [-1,-2, 1, 3] ], float)

# Vettore v corrsipondente
v = np.array( [ -2, 3, 2, 7], float)
```

NUMPY / SCIPY LINALG

$$A\,x = v$$

```python
# Matrice A corrispondente ai coefficienti delle equazioni
A = np.array([ [ 2, 2, 3,-1],
               [ 3, 4,-1, 1],
               [ 1, 3, 4, 2],
               [-1,-2, 1, 3] ], float)

# Vettore v corrsipondente
v = np.array( [ -2, 3, 2, 7], float)
```

Il metodo della fattorizzazione LU è molto usato per la risoluzione di sistemi lineari ed è implementato sia in *numpy* che in *scipy*

NUMPY / SCIPY LINALG

$$A\,x = v$$

```python
# Matrice A corrispondente ai coefficienti delle equazioni
A = np.array([ [ 2, 2, 3,-1],
               [ 3, 4,-1, 1],
               [ 1, 3, 4, 2],
               [-1,-2, 1, 3] ], float)

# Vettore v corrsipondente
v = np.array( [ -2, 3, 2, 7], float)
```

Il metodo della fattorizzazione LU è molto usato per la risoluzione di sistemi lineari ed è implementato sia in *numpy* che in *scipy*

NUMPY

```python
# Risolvo il sistema di equazioni tramine numpy.linalg.solve
xs = np.linalg.solve(A, v)
print('Soluzione numpy', xs)
```

```
Soluzione numpy [ 1.5  -1.   -0.25  2.25]
```

SCIPY

```python
from scipy import linalg

# Risolvo il sistema di equazioni tramine scipy.linalg.solve
xsp = linalg.solve(A, v)
print('Soluzione scipy', xsp)
```

```
Soluzione scipy [ 1.5  -1.   -0.25  2.25]
```

EQUAZIONI NON LINEARI – METODO DEL RILASSAMENTO

$$x = f(x)$$

- Si ipotizza un valore di partenza per calcolare la funzione e poi usare il risultato come nuovo input
- Si continua ad usare il risultato come input in maniera iterativa.
- In alcuni casi il risultato converge ad un valore costante che corrisponde alla soluzione dell'equazione.

EQUAZIONI NON LINEARI – METODO DEL RILASSAMENTO

$$x = 2 - e^{-x}$$

```
def myfun(x):
    """
    myfun(x)
    return 2-e^-x
    """
    return 2-math.exp(-x)
```

EQUAZIONI NON LINEARI – METODO DEL RILASSAMENTO

$$x = 2 - e^{-x}$$

```python
def myfun(x):
    """
    myfun(x)
    return 2-e^-x
    """
    return 2-math.exp(-x)
```

```python
# parto dal valore 1000 e applico iterativamente il metodo del rilassamento
xx = 1000
for i in range(30):
    xx = myfun(xx)
    print(xx)
```

EQUAZIONI NON LINEARI – METODO DEL RILASSAMENTO

$$x = 2 - e^{-x}$$

```
def myfun(x):
    """
    myfun(x)
    return 2-e^-x
    """
    return 2-math.exp(-x)
```

```
# parto dal valore 1000 e applico iterativamente il metodo del rilassamento
xx = 1000
for i in range(30):
    xx = myfun(xx)
    print(xx)
```

```
2.0
1.8646647167633872
1.8450518473052135
1.8419828720850022
1.8414971765224537
1.8414201737059899
1.8414079621425745
1.8414060254740223
1.8414057183297619
1.8414056696184309
1.8414056618930899
1.8414056606678946
1.8414056604735856
1.841405660442769
1.841405660437882
1.8414056604371067
1.8414056604369837
1.8414056604369642
1.841405660436961
1.8414056604369606
1.8414056604369606
1.8414056604369606
1.8414056604369606
1.8414056604369606
1.8414056604369606
```

EQUAZIONI NON LINEARI – METODO DEL RILASSAMENTO

Giustificazione Matematica Metodo del Rilassamento:

$$x_s = f(x_s)$$

EQUAZIONI NON LINEARI – METODO DEL RILASSAMENTO

Giustificazione Matematica Metodo del Rilassamento:

$$x_s = f(x_s)$$

Sviluppo in serie di Taylor: $f(x) = f(x_s) + (x - x_s)f'(x_s) + \ldots$

EQUAZIONI NON LINEARI – METODO DEL RILASSAMENTO

Giustificazione Matematica Metodo del Rilassamento:

$$x_s = f(x_s)$$

Sviluppo in serie di Taylor: $$f(x) = f(x_s) + (x - x_s)f'(x_s) + \ldots$$

$$x_1 = f(x_0) = f(x_s) + (x_0 - x_s)f'(x_s),$$

$$x_1 = x_s + (x_0 - x_s)f'(x_s)$$

EQUAZIONI NON LINEARI – METODO DEL RILASSAMENTO

Giustificazione Matematica Metodo del Rilassamento:

$$x_s = f(x_s)$$

Sviluppo in serie di Taylor:

$$f(x) = f(x_s) + (x - x_s)f'(x_s) + \dots$$

$$x_1 = f(x_0) = f(x_s) + (x_0 - x_s)f'(x_s),$$

$$x_1 = x_s + (x_0 - x_s)f'(x_s)$$

$$x_1 - x_s = (x_0 - x_s)f'(x_s)$$

EQUAZIONI NON LINEARI – METODO DEL RILASSAMENTO

Giustificazione Matematica Metodo del Rilassamento:

$$x_s = f(x_s)$$

Sviluppo in serie di Taylor:

$$f(x) = f(x_s) + (x - x_s)f'(x_s) + \dots$$

$$x_1 = f(x_0) = f(x_s) + (x_0 - x_s)f'(x_s),$$

$$x_1 = x_s + (x_0 - x_s)f'(x_s)$$

$$x_1 - x_s = (x_0 - x_s)f'(x_s)$$

$|x_1 - x_s| < |x_0 - x_s|$ se: $|f'(x_s)| < 1$

Se il modulo della derivata è < 1 mi avvicino progressivamente alla soluzione

EQUAZIONI NON LINEARI – METODO DELLA BISEZIONE

$$f(x) = 0$$

ESEMPIO

$$f(x) = 2\cos\tfrac{4+x}{10} + \tfrac{1}{2}$$

$$2\cos\frac{4+x}{10} + \frac{1}{2} = 0.$$

EQUAZIONI NON LINEARI – METODO DELLA BISEZIONE

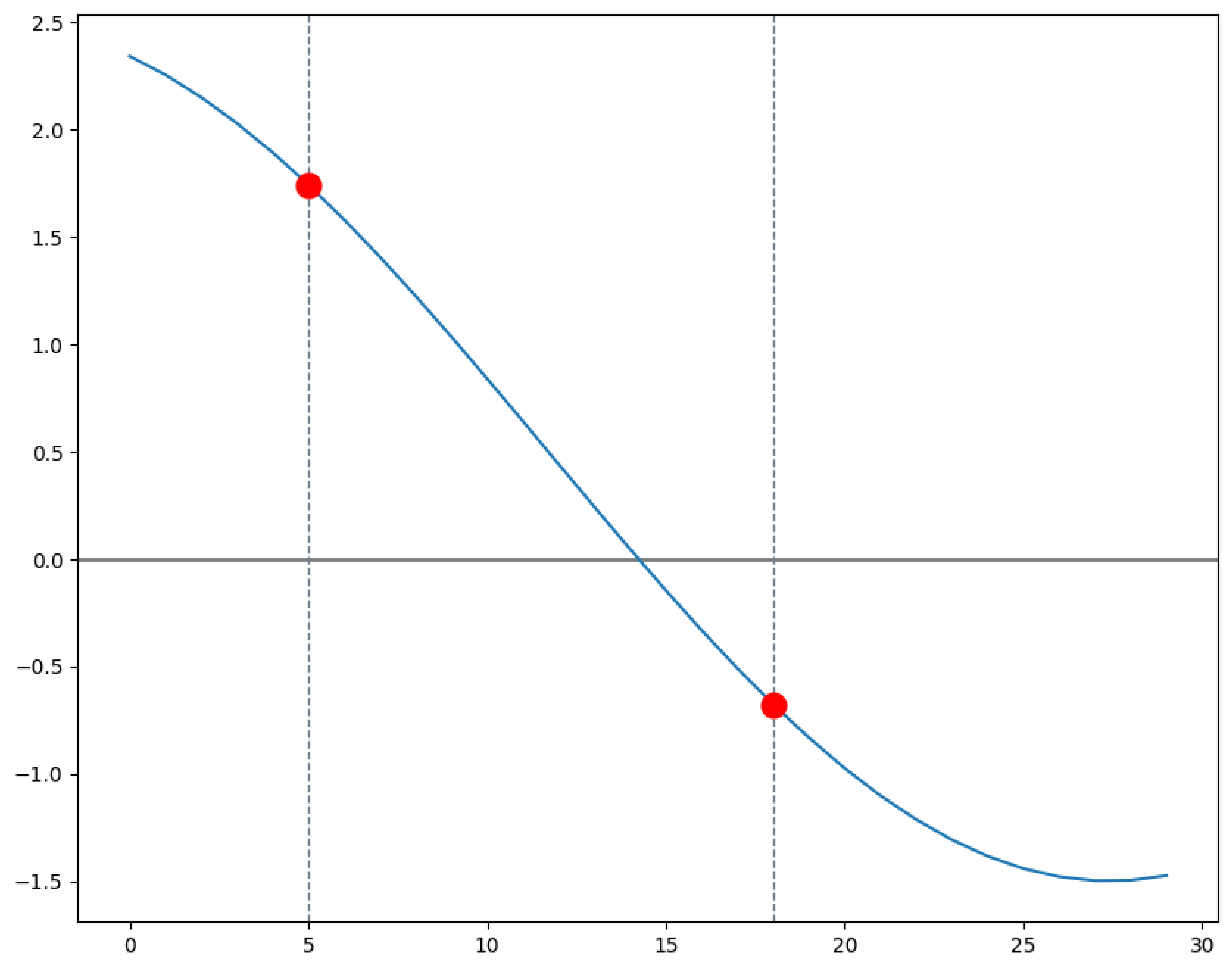
$$f(x) = 0$$

ESEMPIO

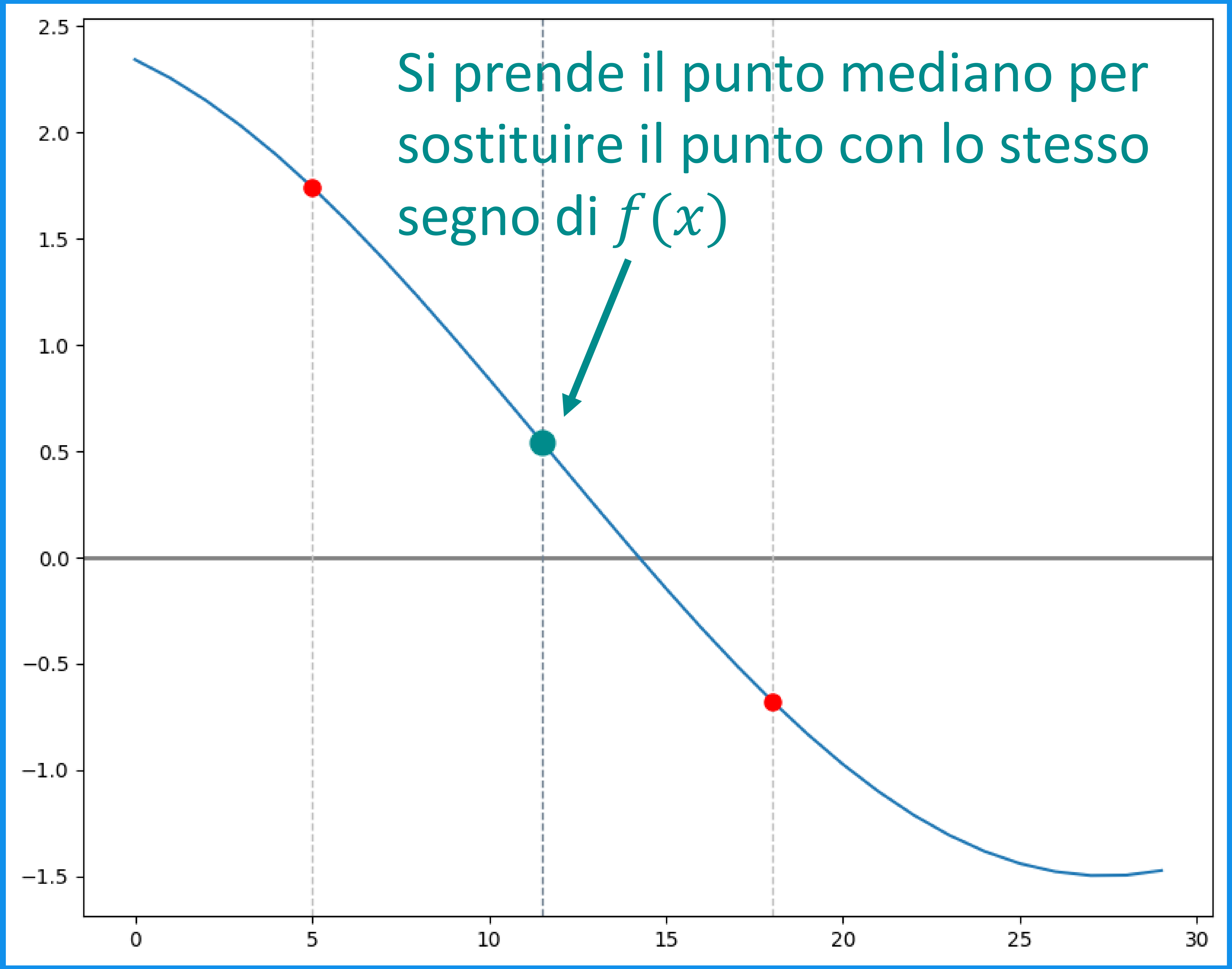
$$f(x) = 2\cos\frac{4+x}{10} + \frac{1}{2}$$

$$2\cos\frac{4+x}{10} + \frac{1}{2} = 0.$$

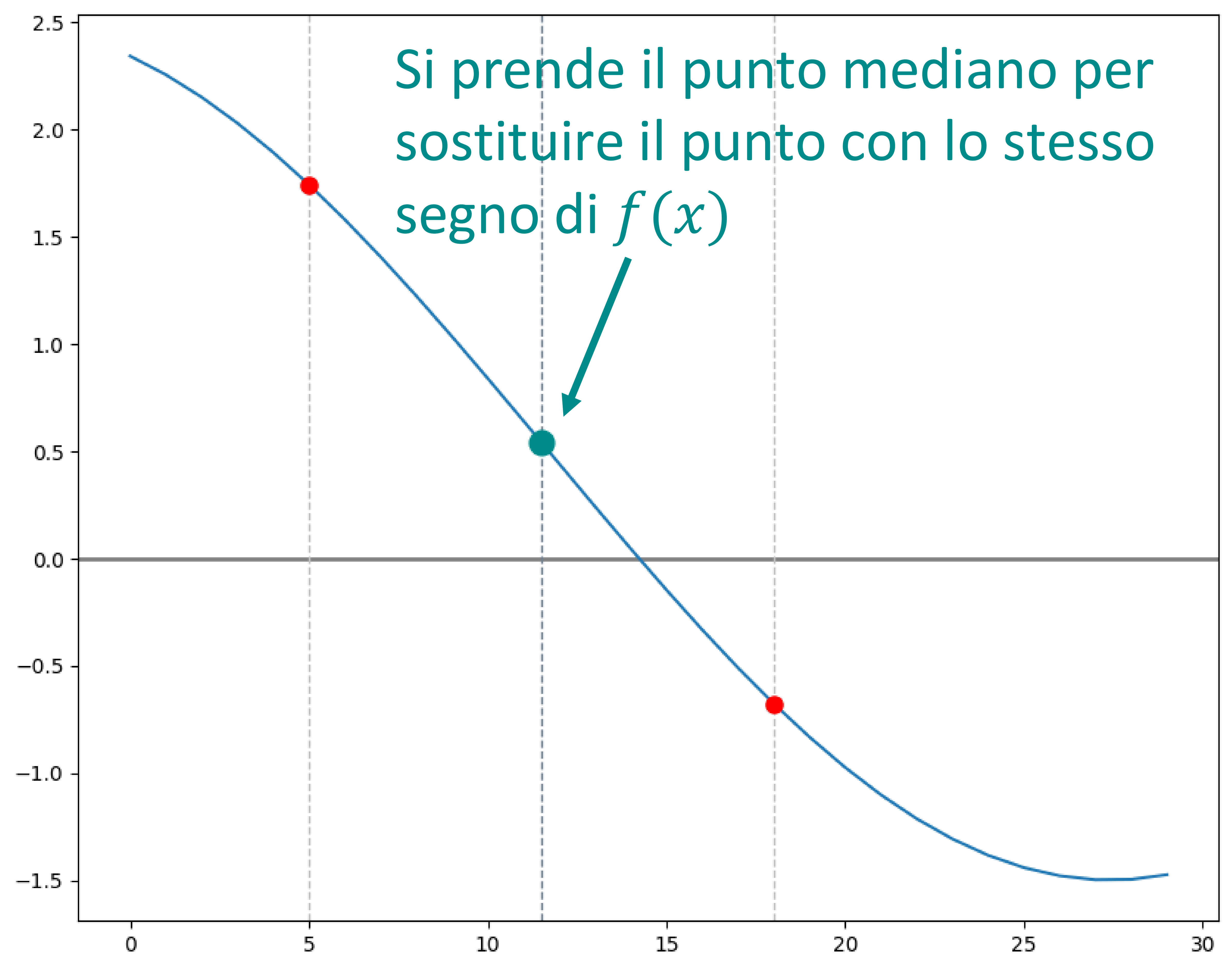
Si parte scegliendo due punti di partenza (x_0, x_1) per cui la funzione $f(x)$ ha segno opposto

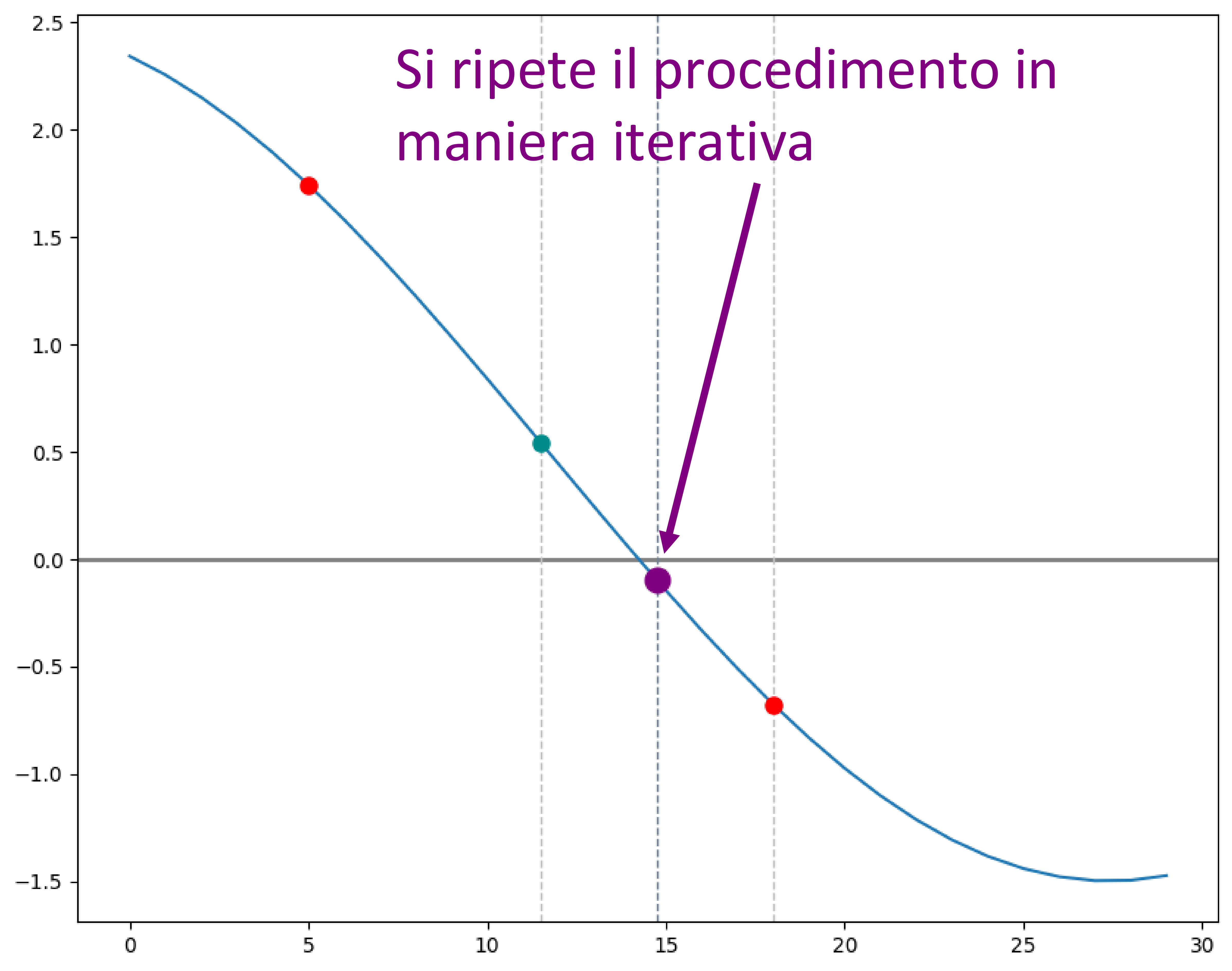


EQUAZIONI NON LINEARI – METODO DELLA BISEZIONE

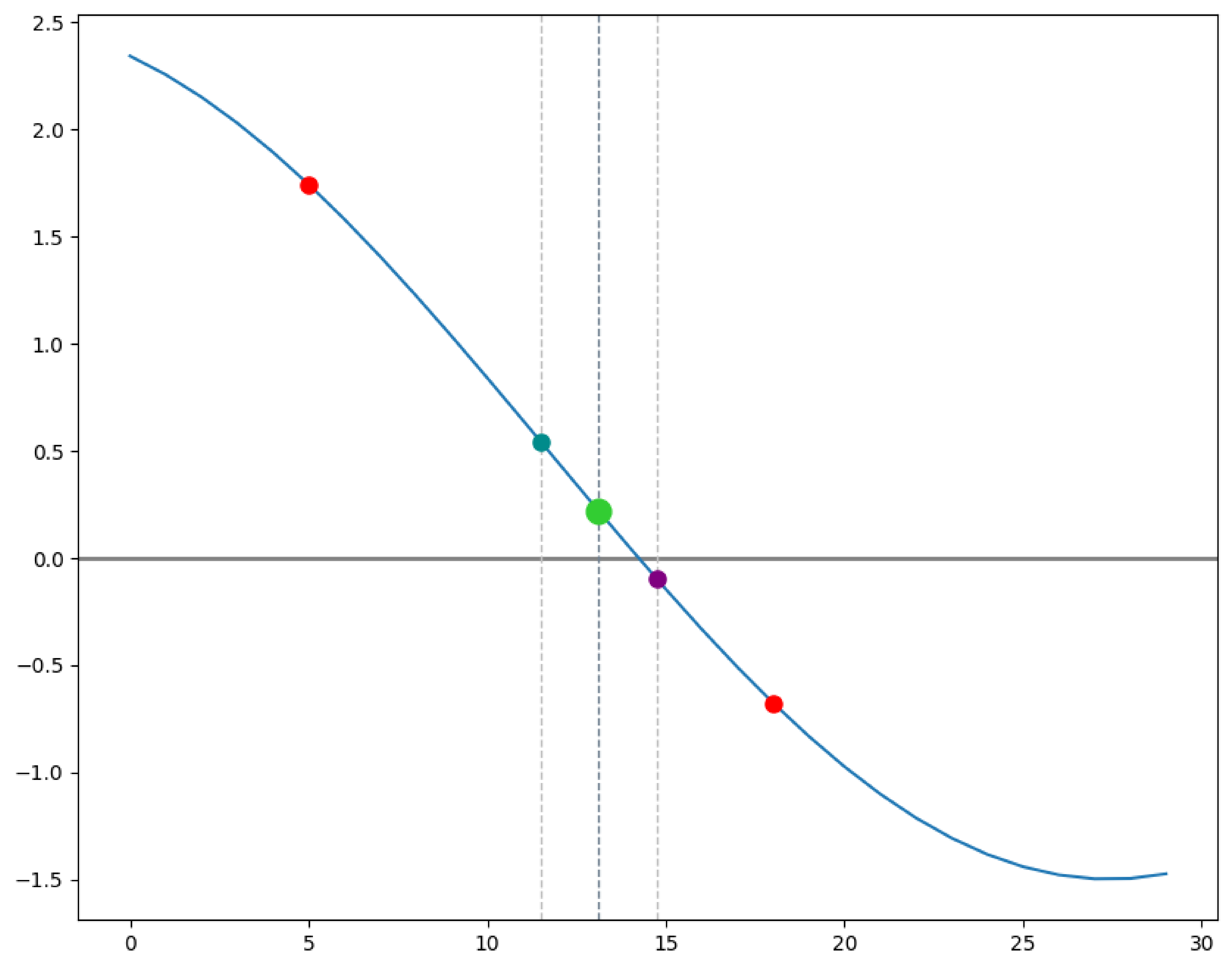


EQUAZIONI NON LINEARI – METODO DELLA BISEZIONE

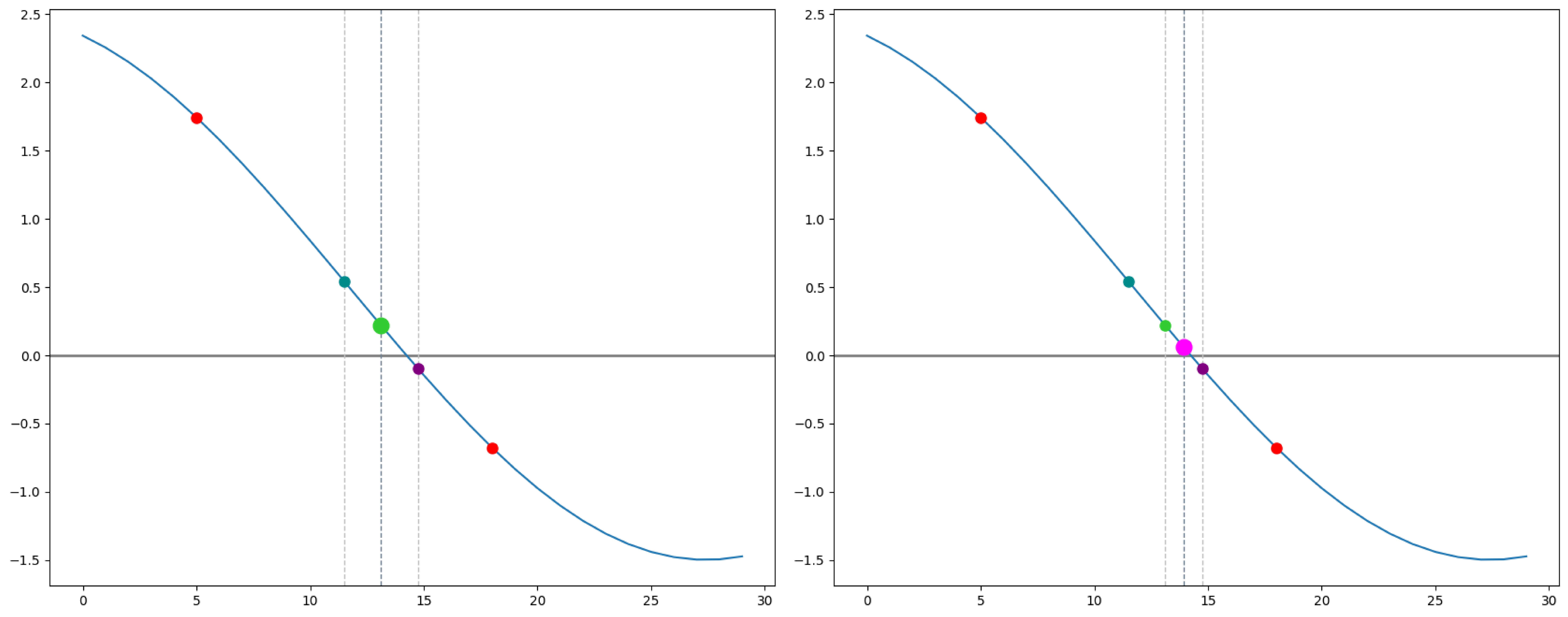




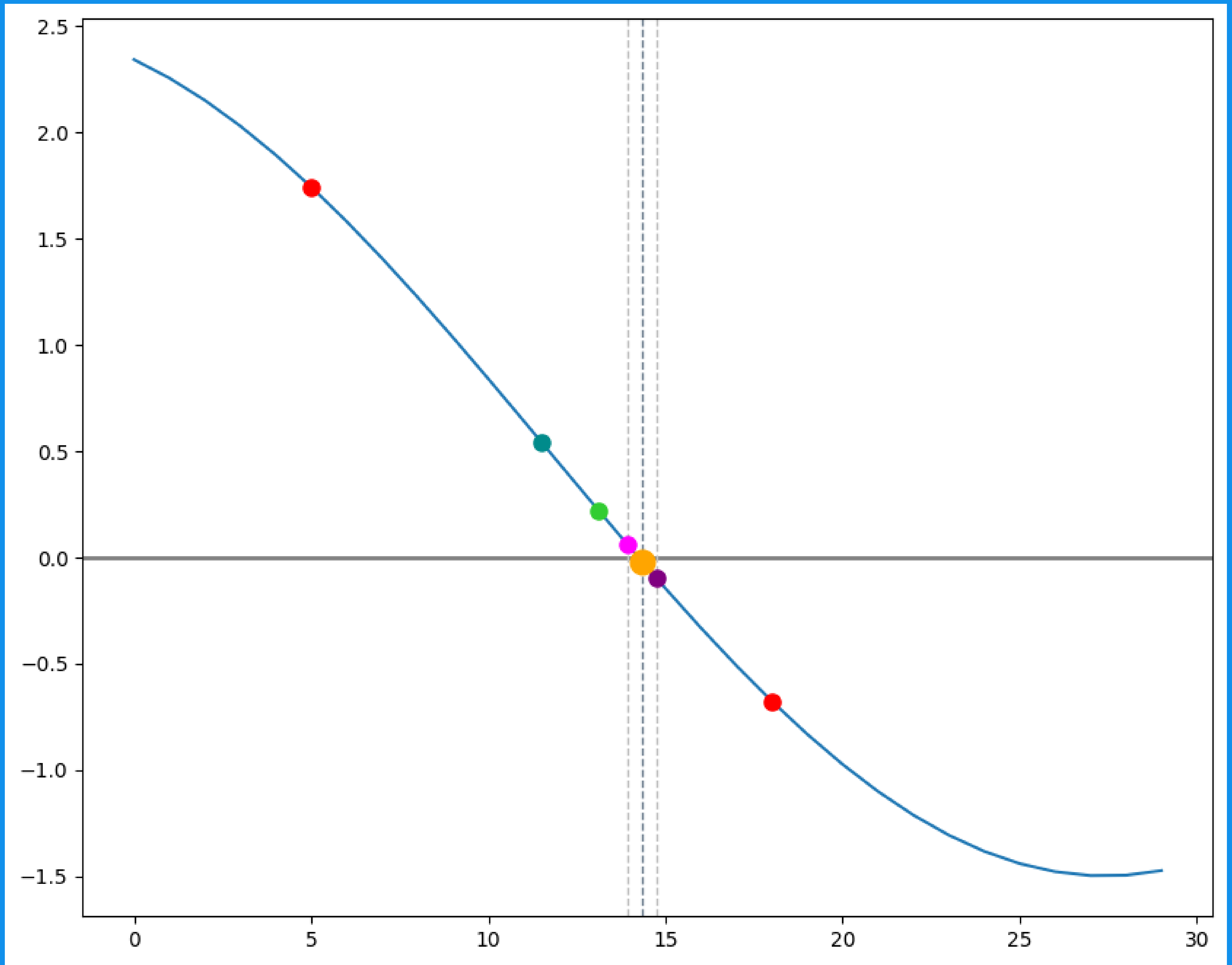
EQUAZIONI NON LINEARI – METODO DELLA BISEZIONE



EQUAZIONI NON LINEARI – METODO DELLA BISEZIONE



EQUAZIONI NON LINEARI – METODO DELLA SECANTE



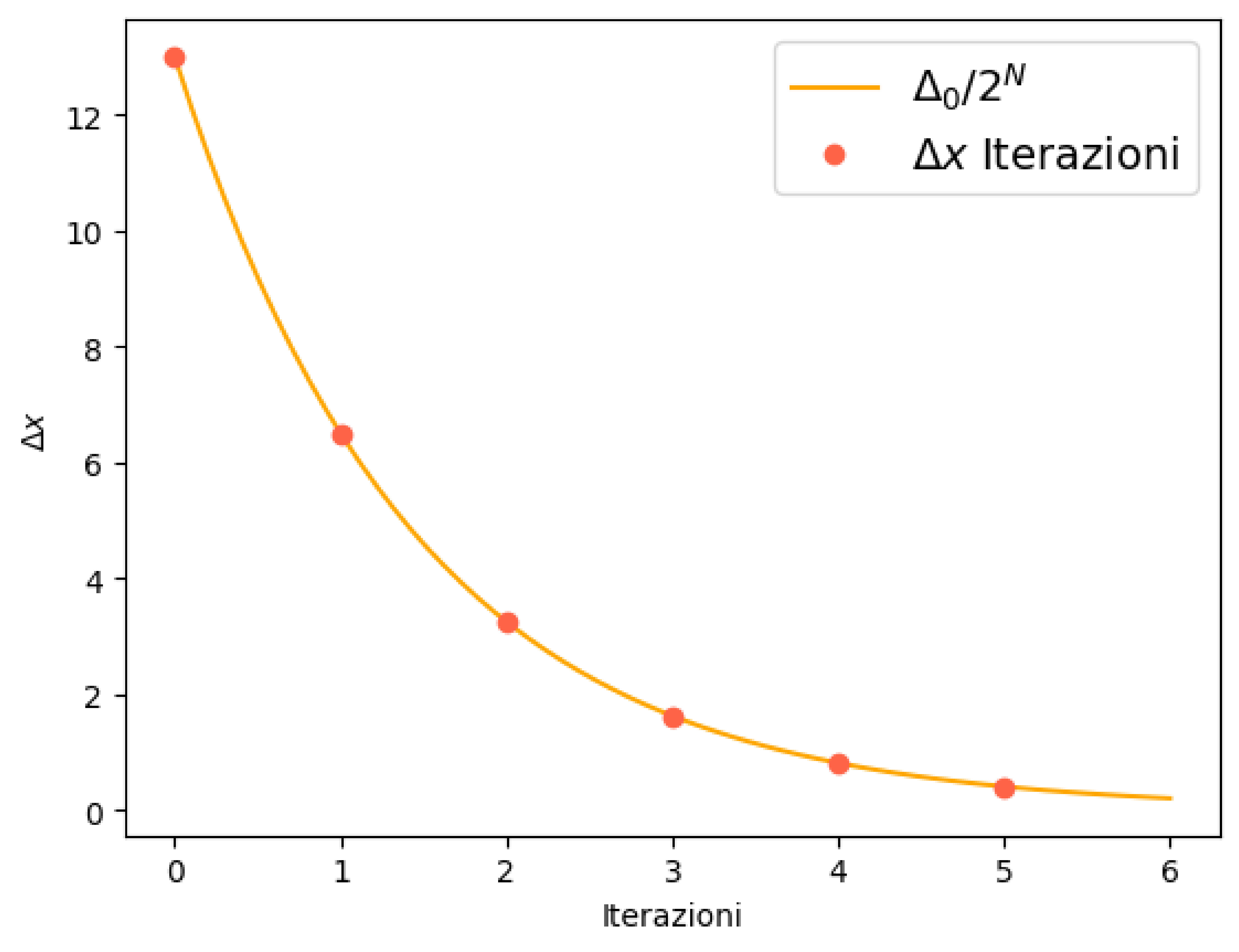
Il Processo iterativo viene interrotto in base alla
Condizione di Arresto

$$|x_N - x_{N-1}| < \epsilon$$

EQUAZIONI NON LINEARI – METODO DELLA BISEZIONE - CONVERGENZA

$$\Delta_0 = |x_1 - x_0|$$

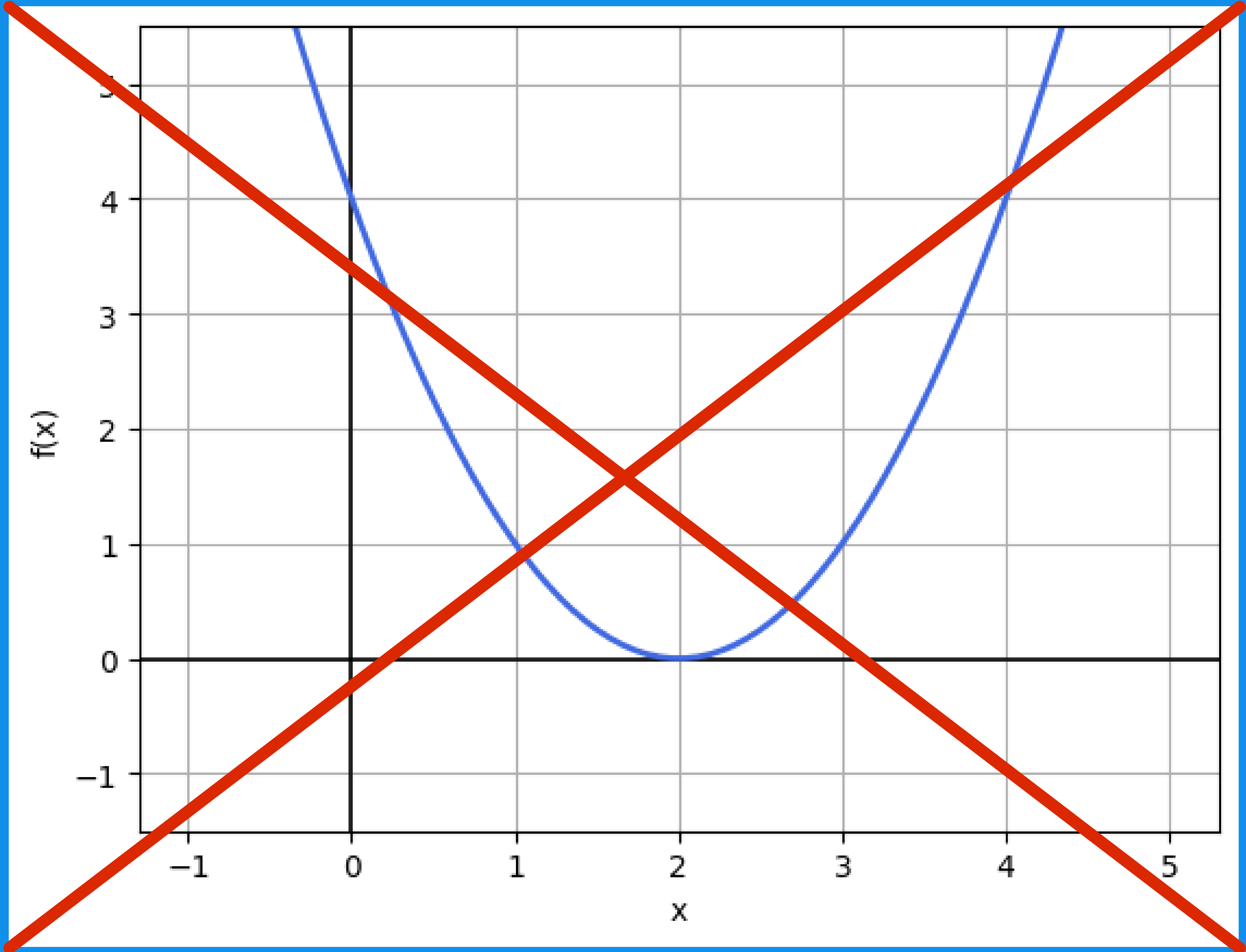
$$\Delta x = \frac{\Delta_0}{2^N}$$



$$N = log_2 \frac{\Delta_0}{\epsilon}.$$

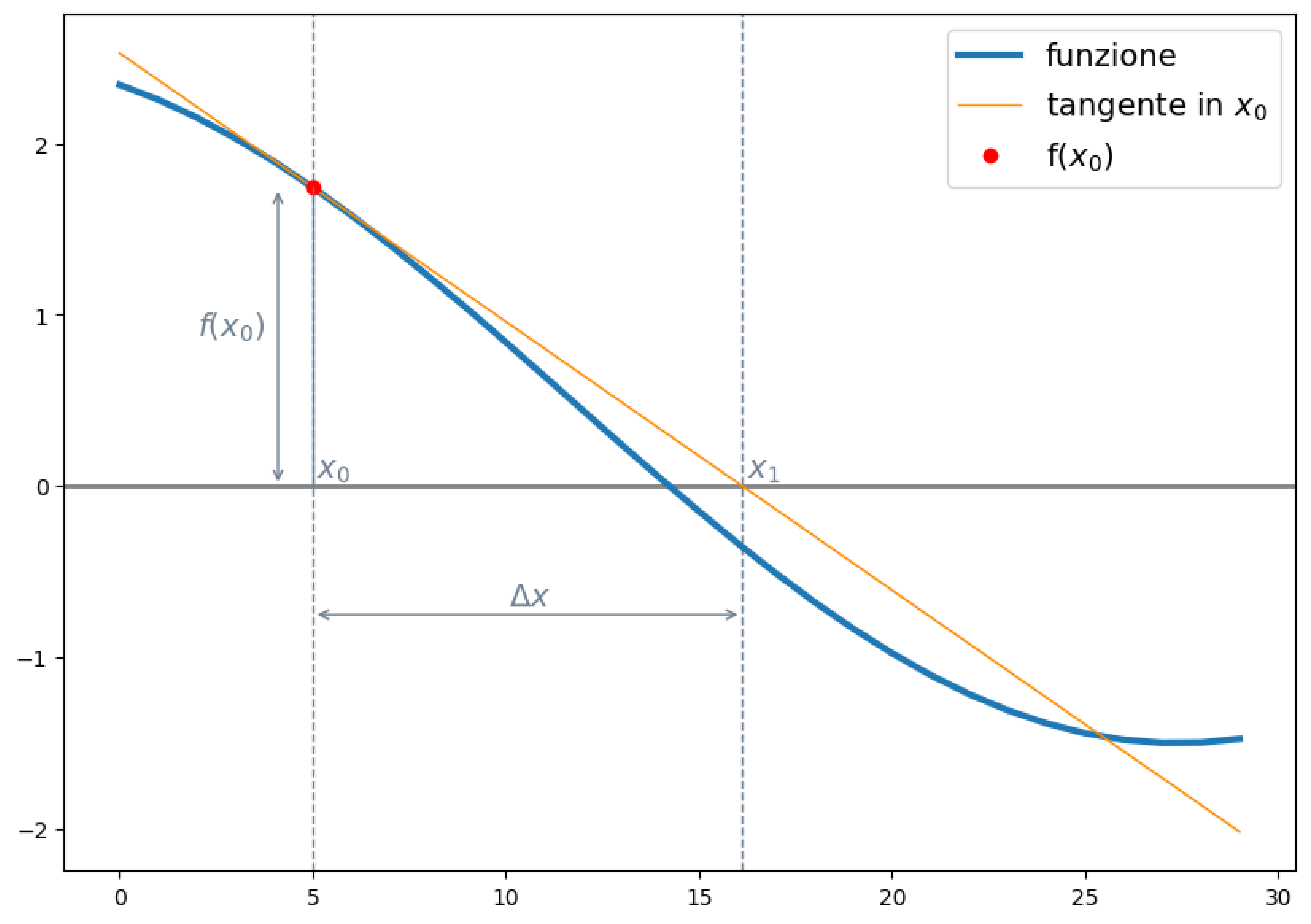
EQUAZIONI NON LINEARI – METODO DELLA BISEZIONE

Il metodo della bisezione può fallire per funzioni con un numero pari di radici e per funzioni con radici multiple coincidenti che non passano da valori negativi a positivi o viceversa



EQUAZIONI NON LINEARI – METODO DI NEWTON

$$f(x) = 0$$

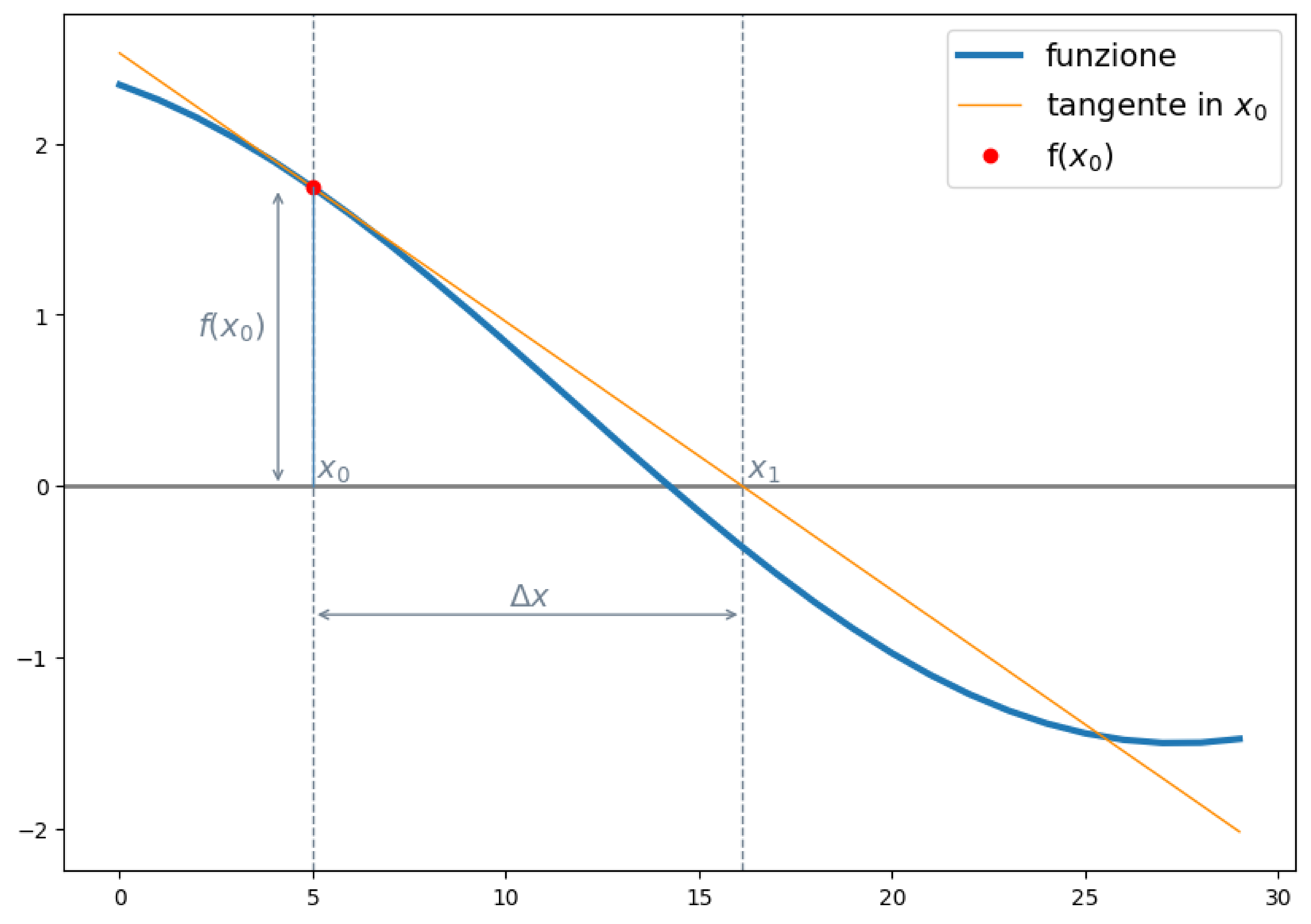


EQUAZIONI NON LINEARI – METODO DI NEWTON

$$f(x) = 0$$

Si parte dalla costatazione che:

$$f'(x_0) = \frac{f(x_0)}{\Delta x}$$



EQUAZIONI NON LINEARI – METODO DI NEWTON

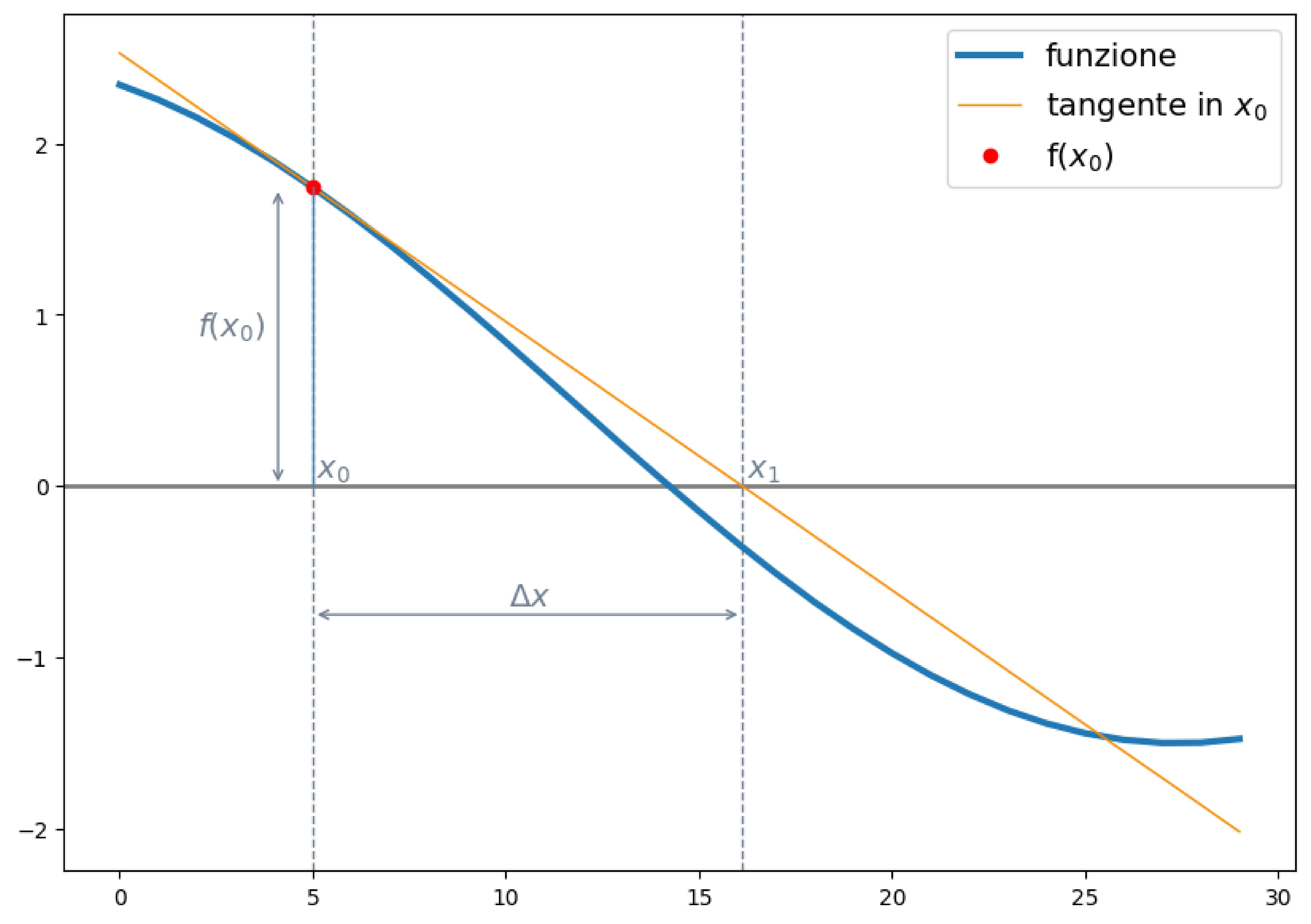
$$f(x) = 0$$

Si parte dalla costatazione che:

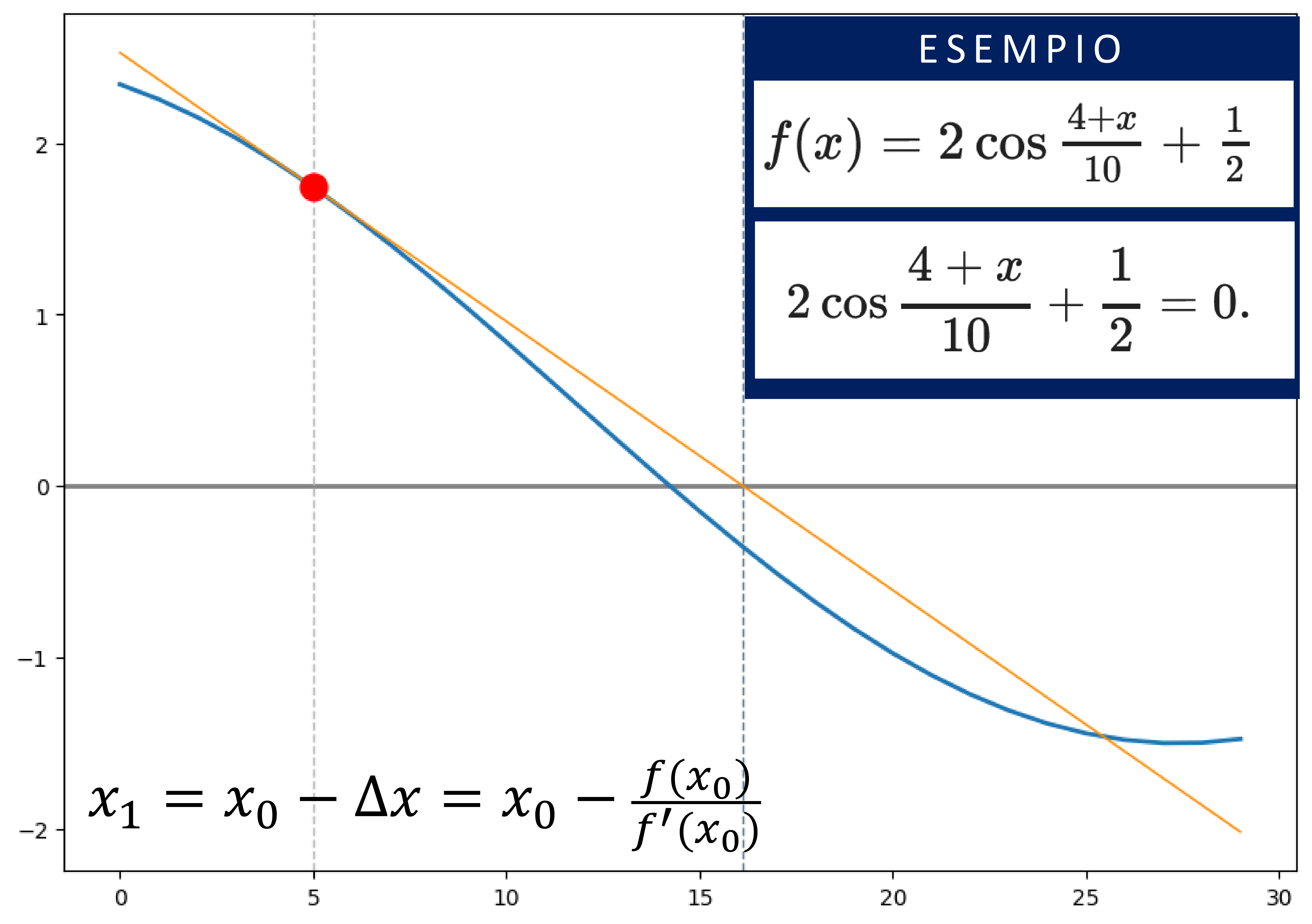
$$f'(x_0) = \frac{f(x_0)}{\Delta x}$$

Da cui ricaviamo il punto x_1:

$$x_1 = x_0 - \Delta x = x - \frac{f(x)}{f'(x)}$$



EQUAZIONI NON LINEARI – METODO DI NEWTON



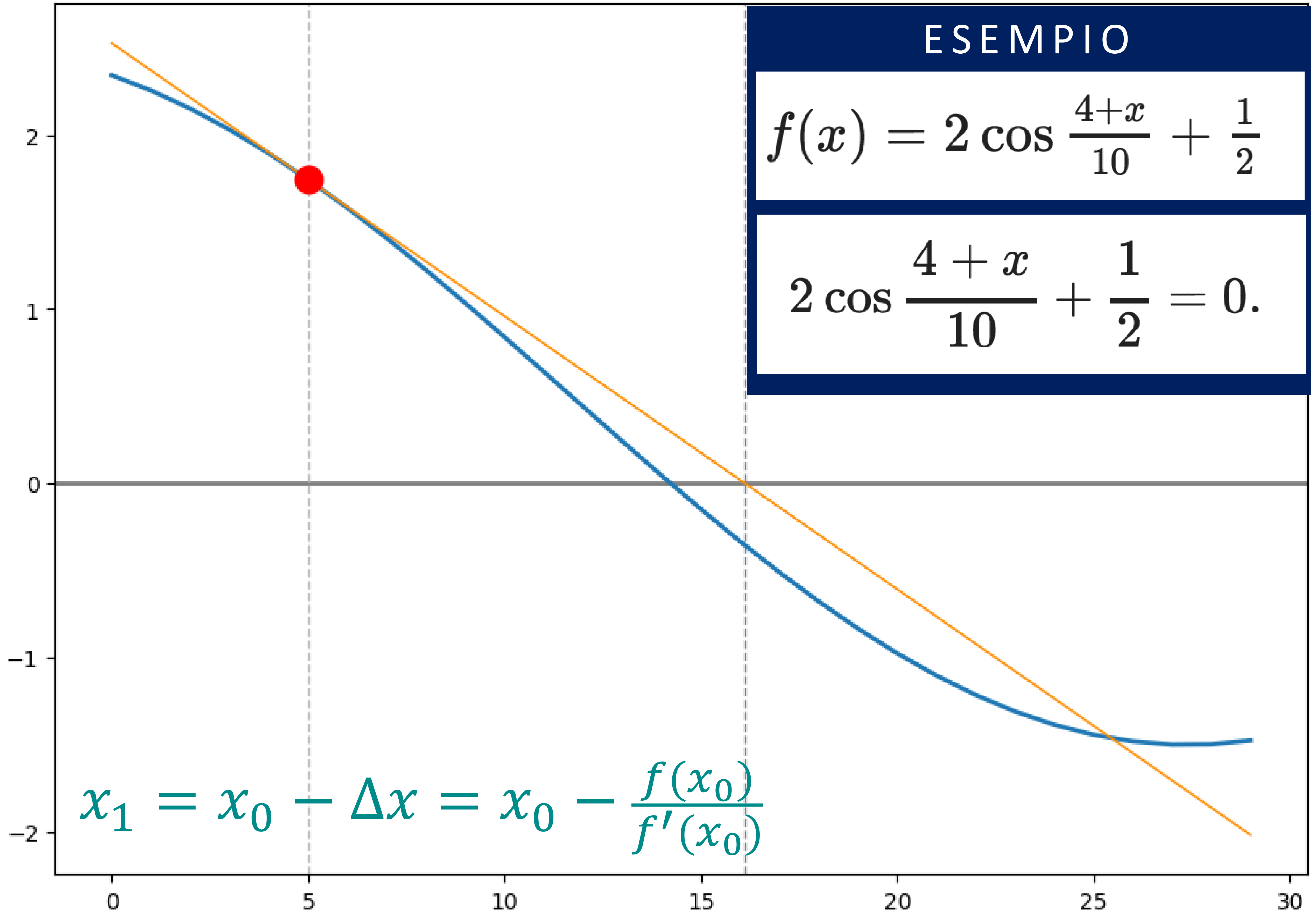
ESEMPIO

$$f(x) = 2\cos\frac{4+x}{10} + \frac{1}{2}$$

$$2\cos\frac{4+x}{10} + \frac{1}{2} = 0.$$

$$x_1 = x_0 - \Delta x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

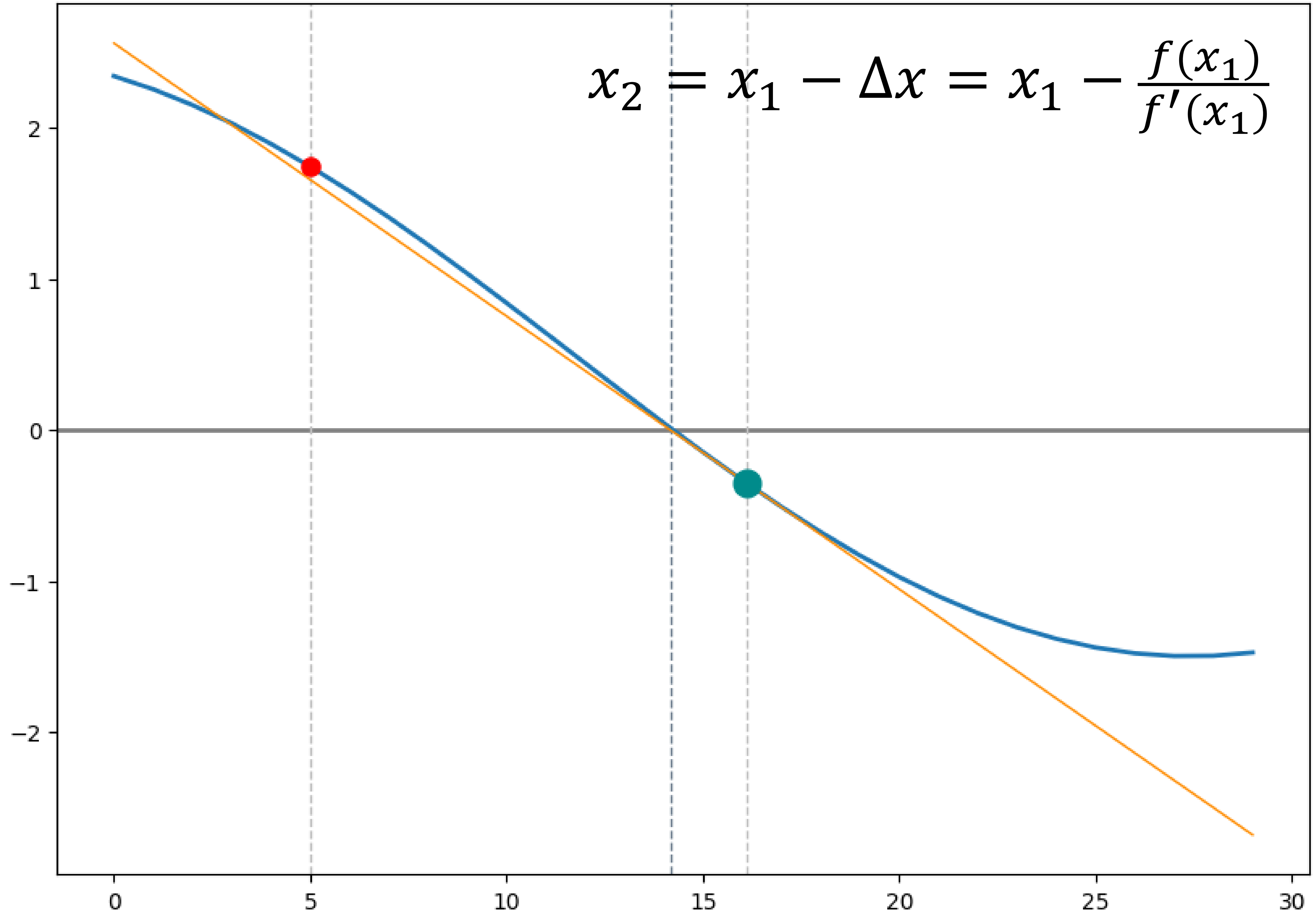
EQUAZIONI NON LINEARI – METODO DI NEWTON



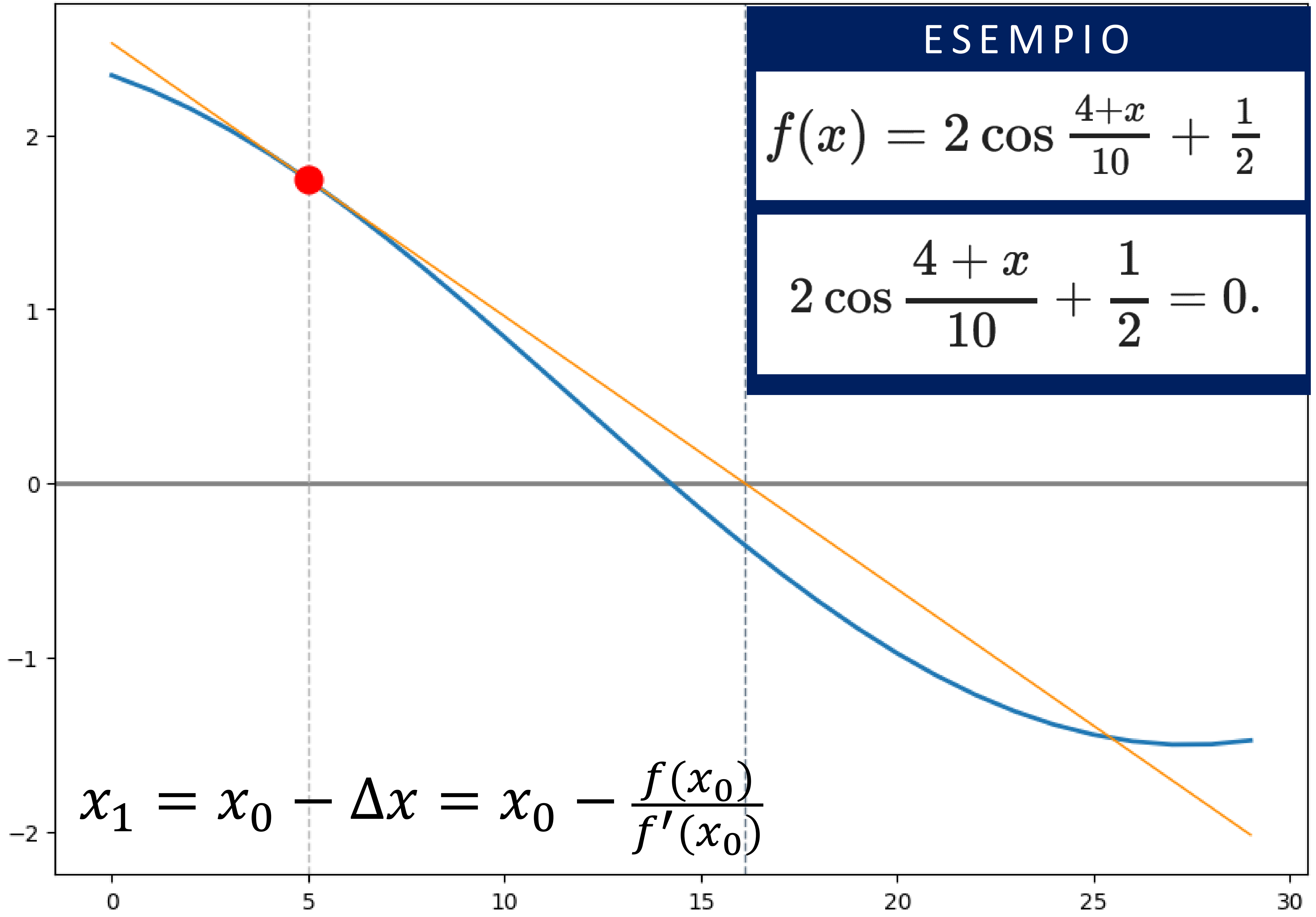
ESEMPIO

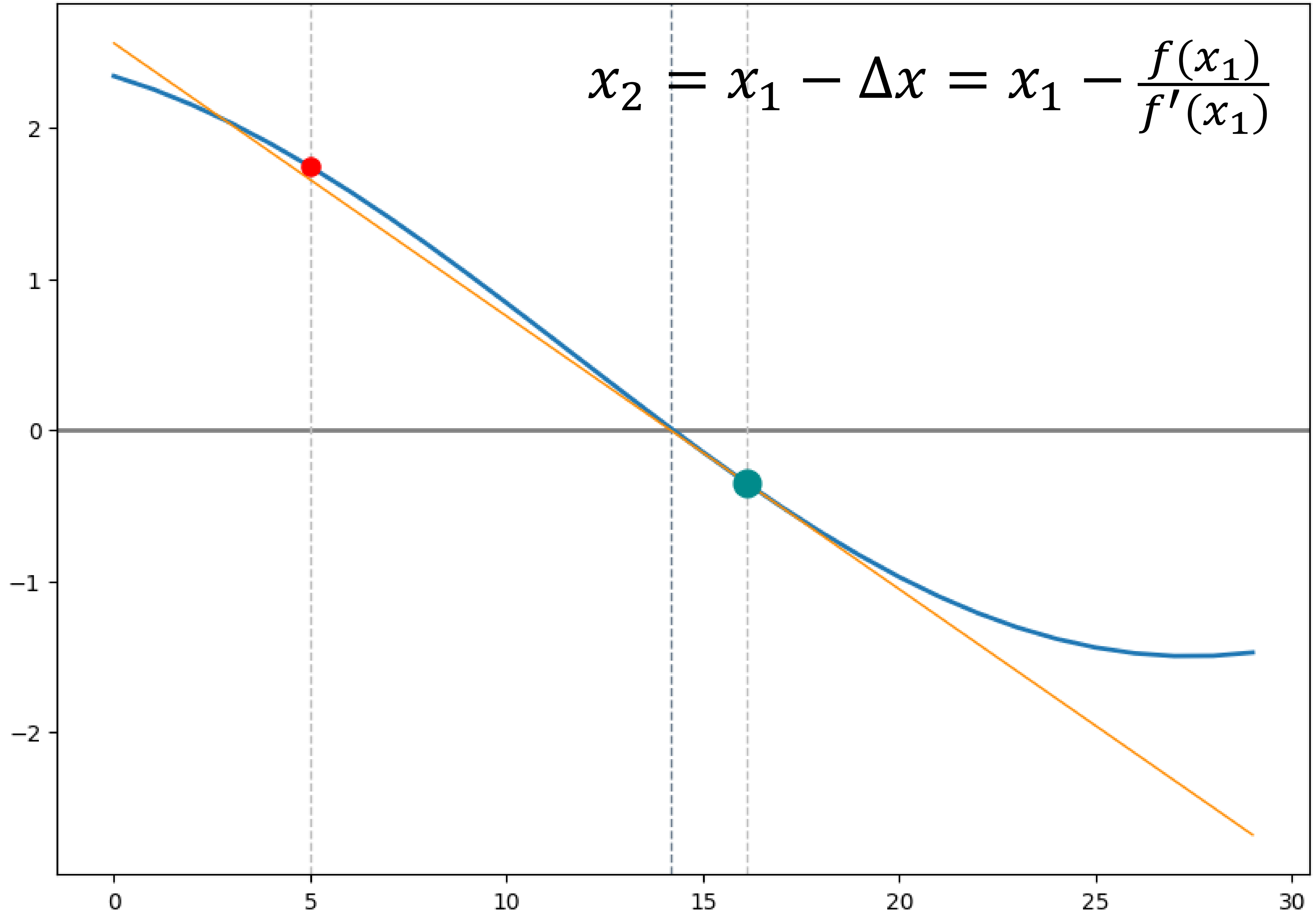
$$f(x) = 2\cos\frac{4+x}{10} + \frac{1}{2}$$

$$2\cos\frac{4+x}{10} + \frac{1}{2} = 0.$$



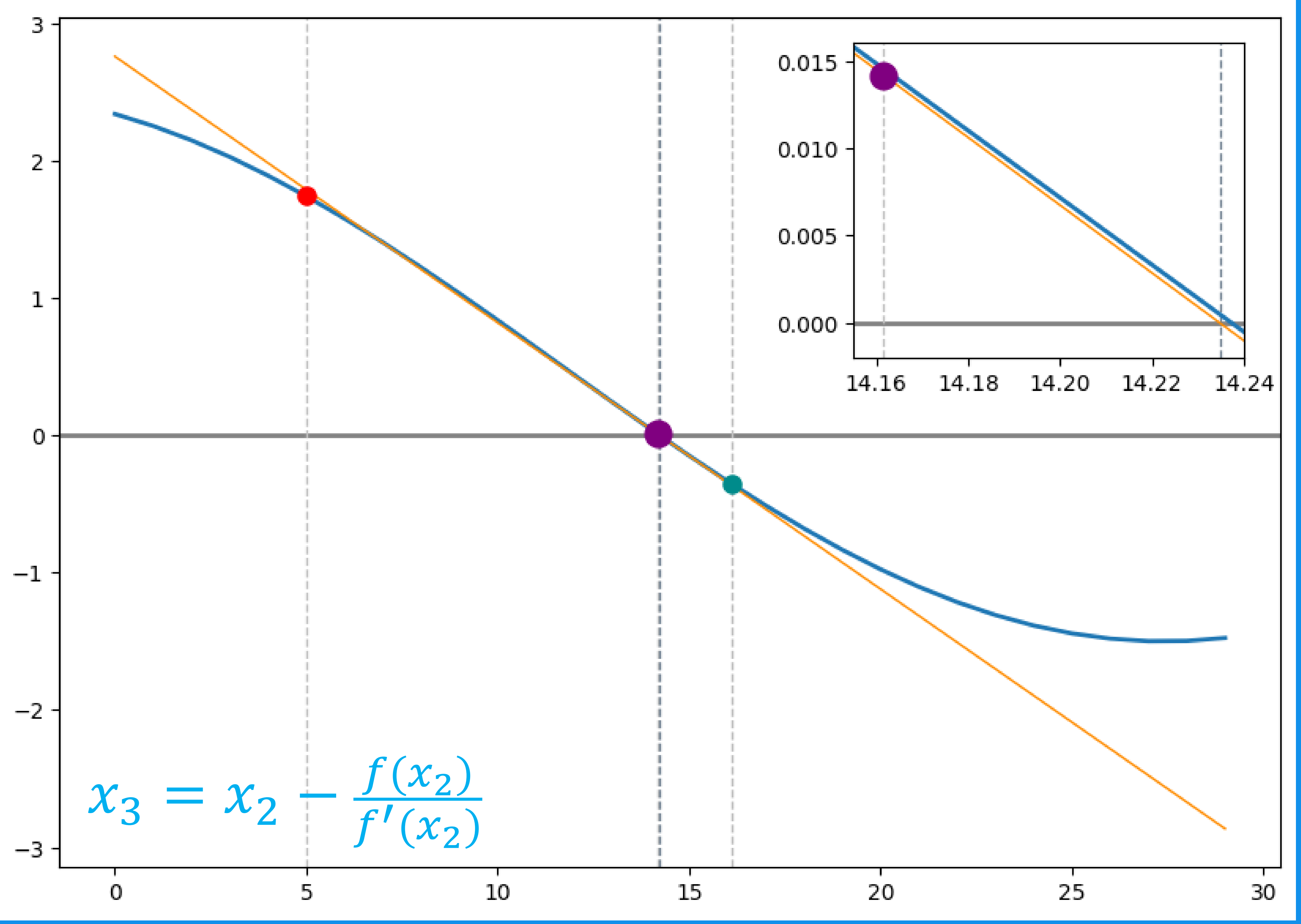
EQUAZIONI NON LINEARI – METODO DI NEWTON



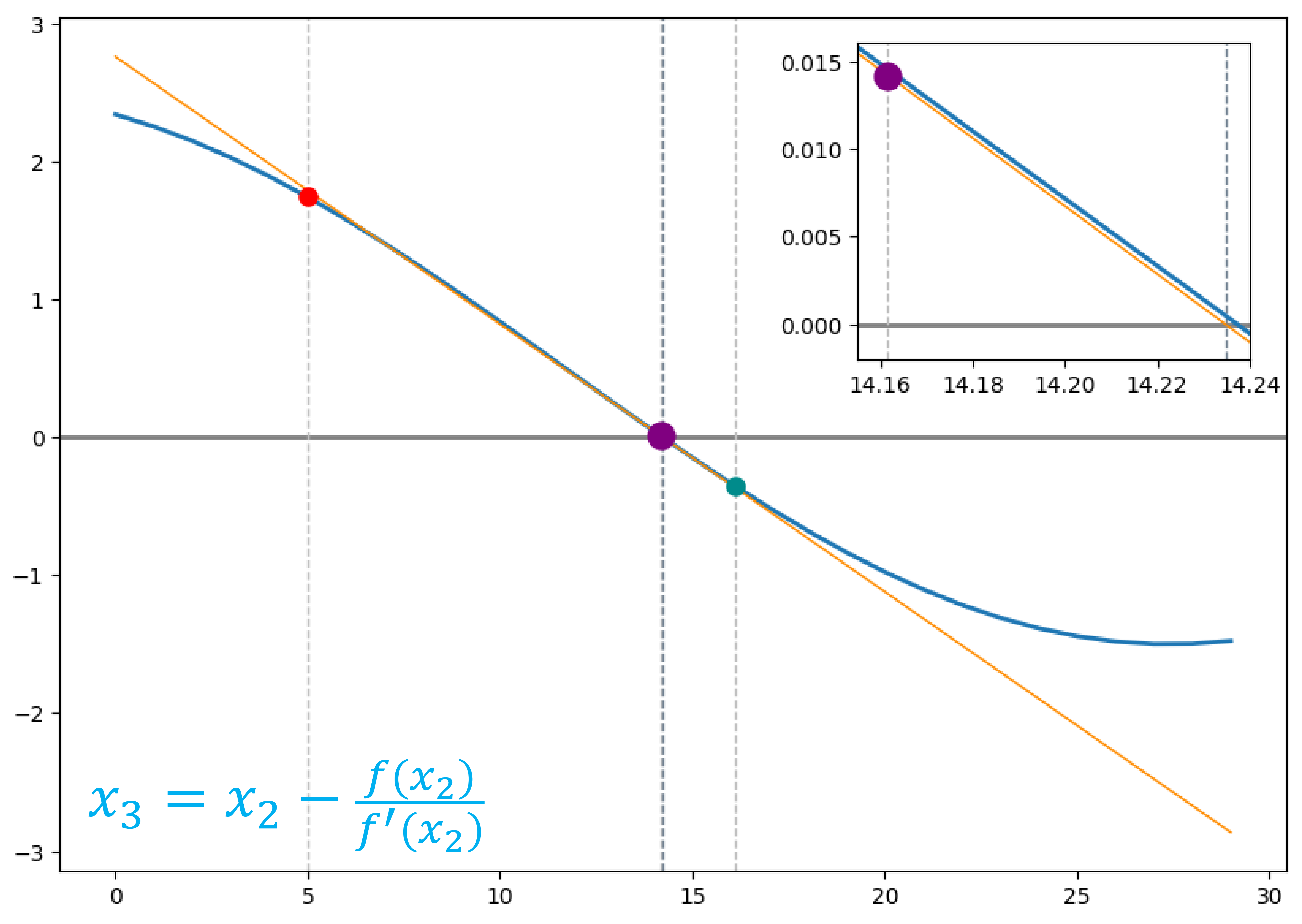


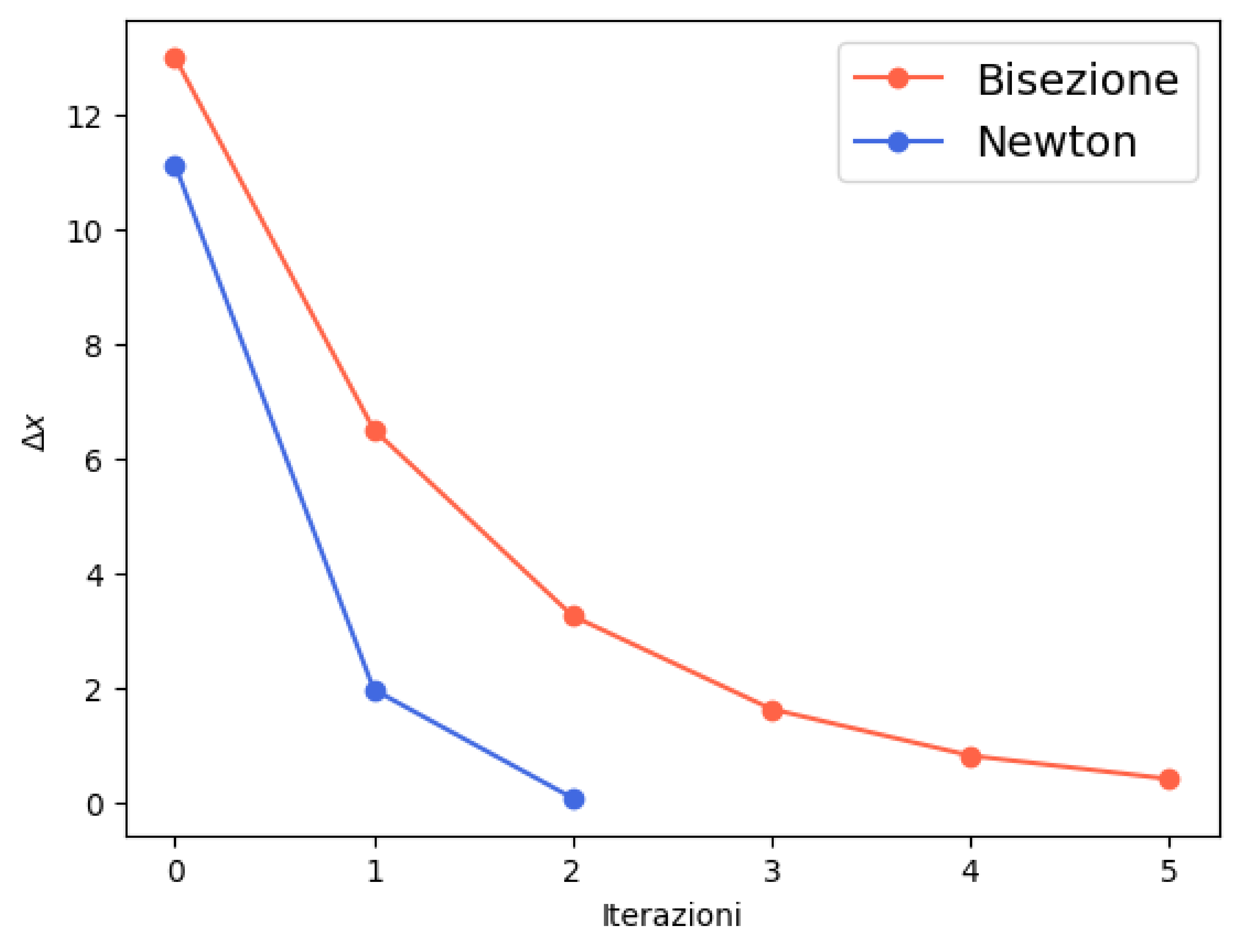
Conoscendo $f'(x)$ si può trovare la radice di $f(x)$ con l'approssimazione desiderata reiterando il calcolo per x_i successivi.

EQUAZIONI NON LINEARI – METODO DI NEWTON

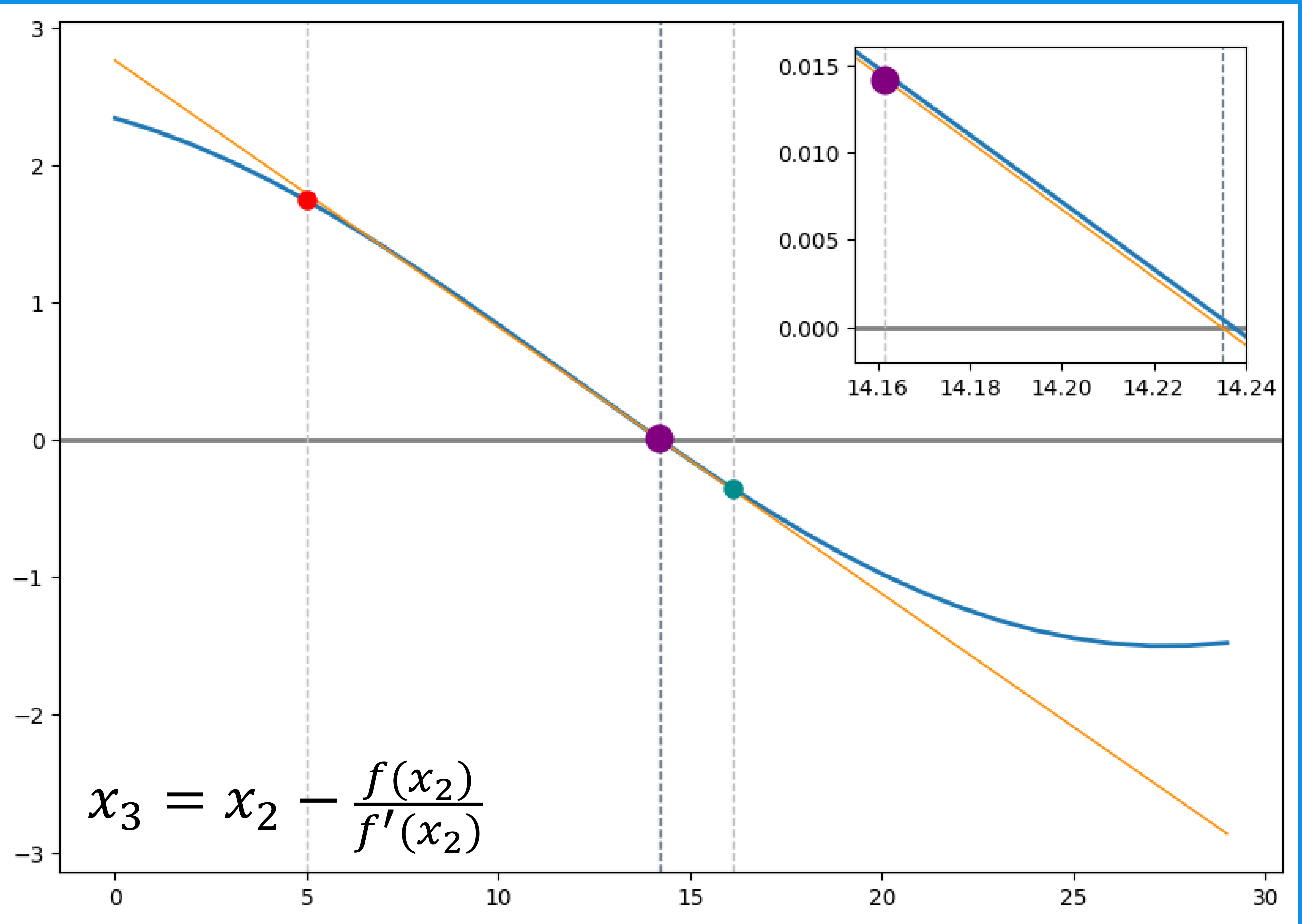


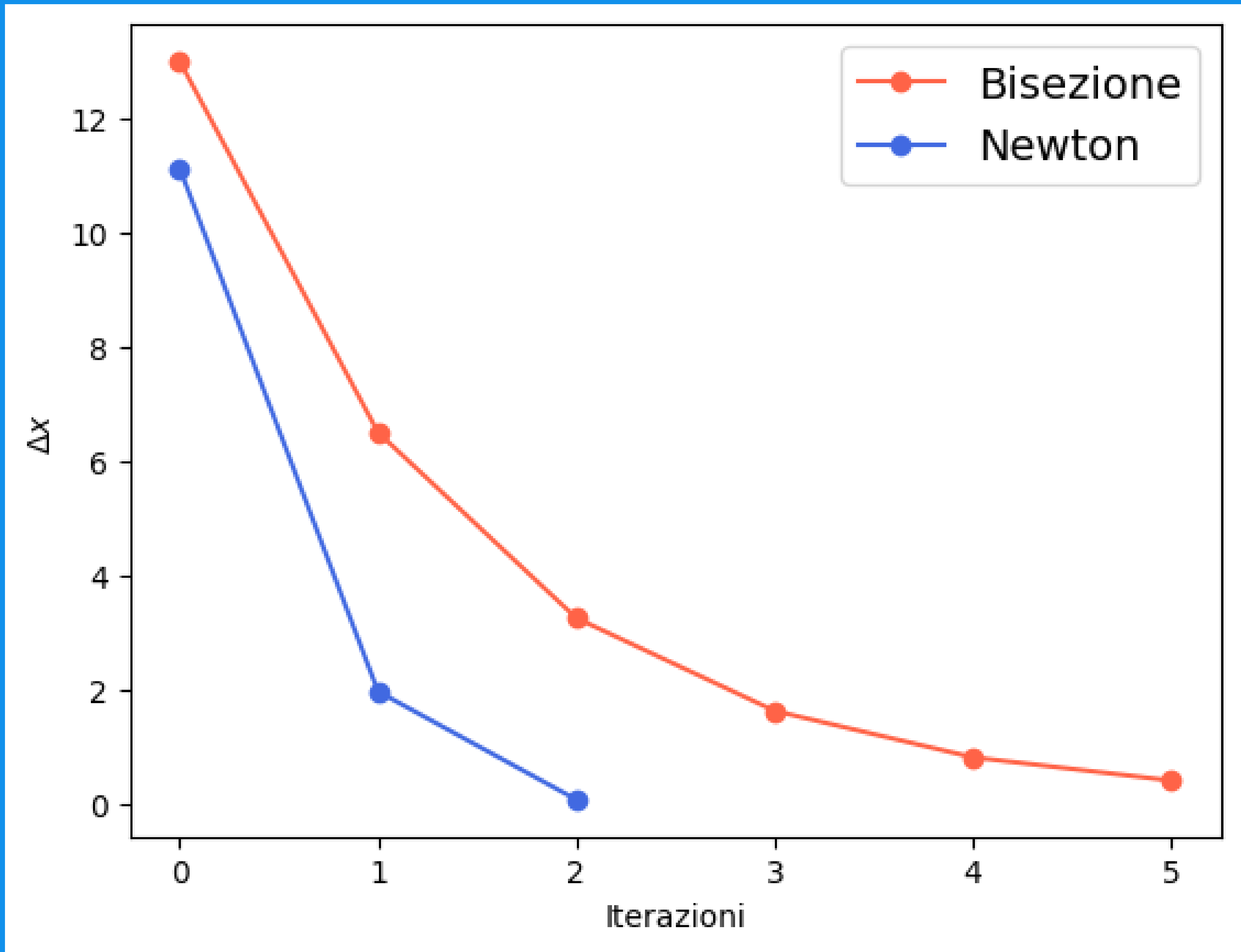
EQUAZIONI NON LINEARI – METODO DI NEWTON





EQUAZIONI NON LINEARI – METODO DI NEWTON





Necessario conoscere analiticamente la derivata $f'(x)$

Non applicabile per serie di punti (x_i, y_i)

EQUAZIONI NON LINEARI – METODO DELLA SECANTE

Il metodo della Secante applica lo stesso ragionamento del metodo di Newton aggirando il problema della conoscenza di $f'(x)$

EQUAZIONI NON LINEARI – METODO DELLA SECANTE

Il metodo della Secante applica lo stesso ragionamento del metodo di Newton aggirando il problema della conoscenza di $f'(x)$

Partendo dai punti iniziali x_0 e x_1 possiamo approssimare la derivata in x_1 come:

$$f'(x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

EQUAZIONI NON LINEARI – METODO DELLA SECANTE

Il metodo della Secante applica lo stesso ragionamento del metodo di Newton aggirando il problema della conoscenza di $f'(x)$

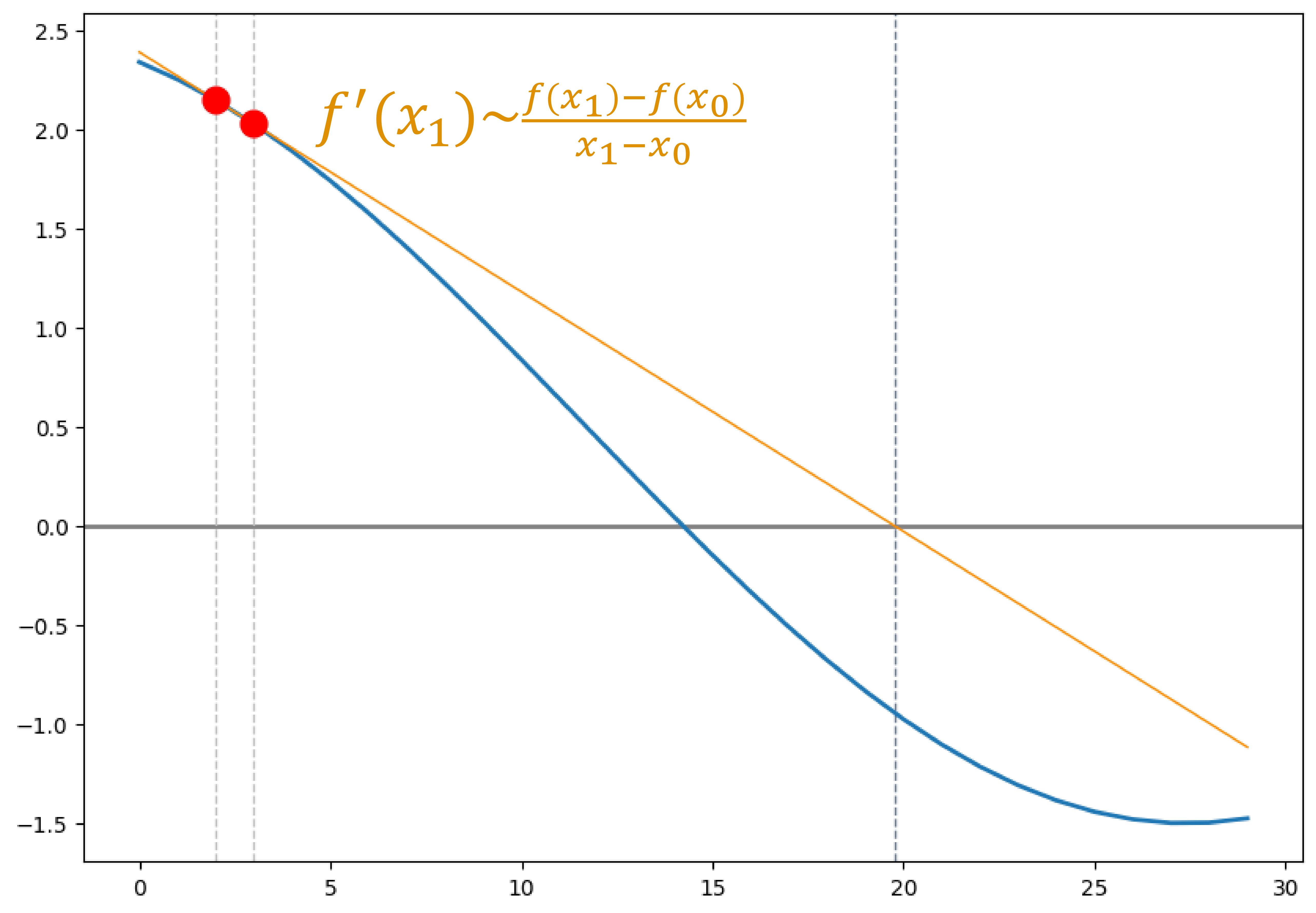
Partendo dai punti iniziali x_0 e x_1 possiamo approssimare la derivata in x_1 come:

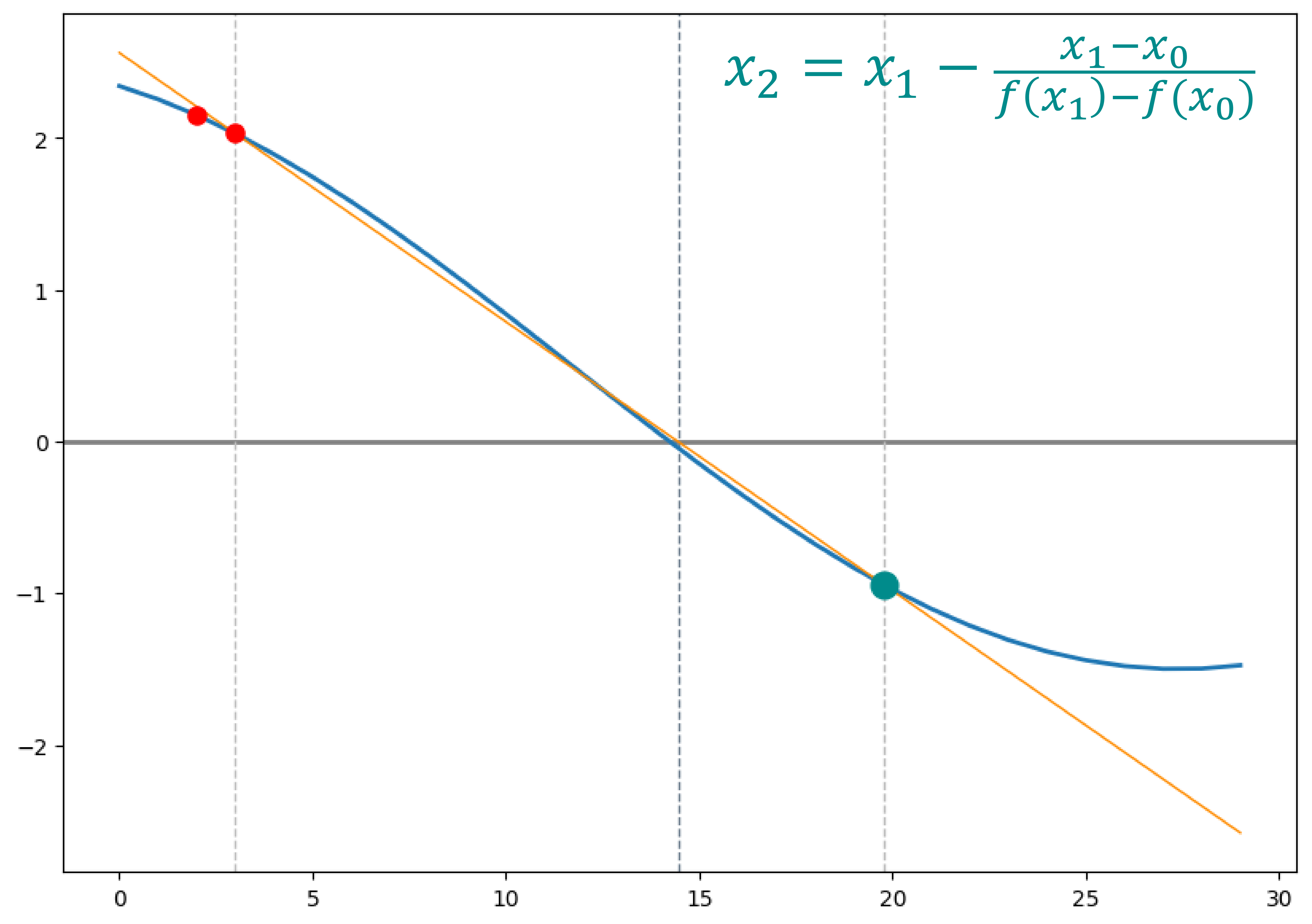
$$f'(x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

Ricaviamo il punto successivo x_2 come:

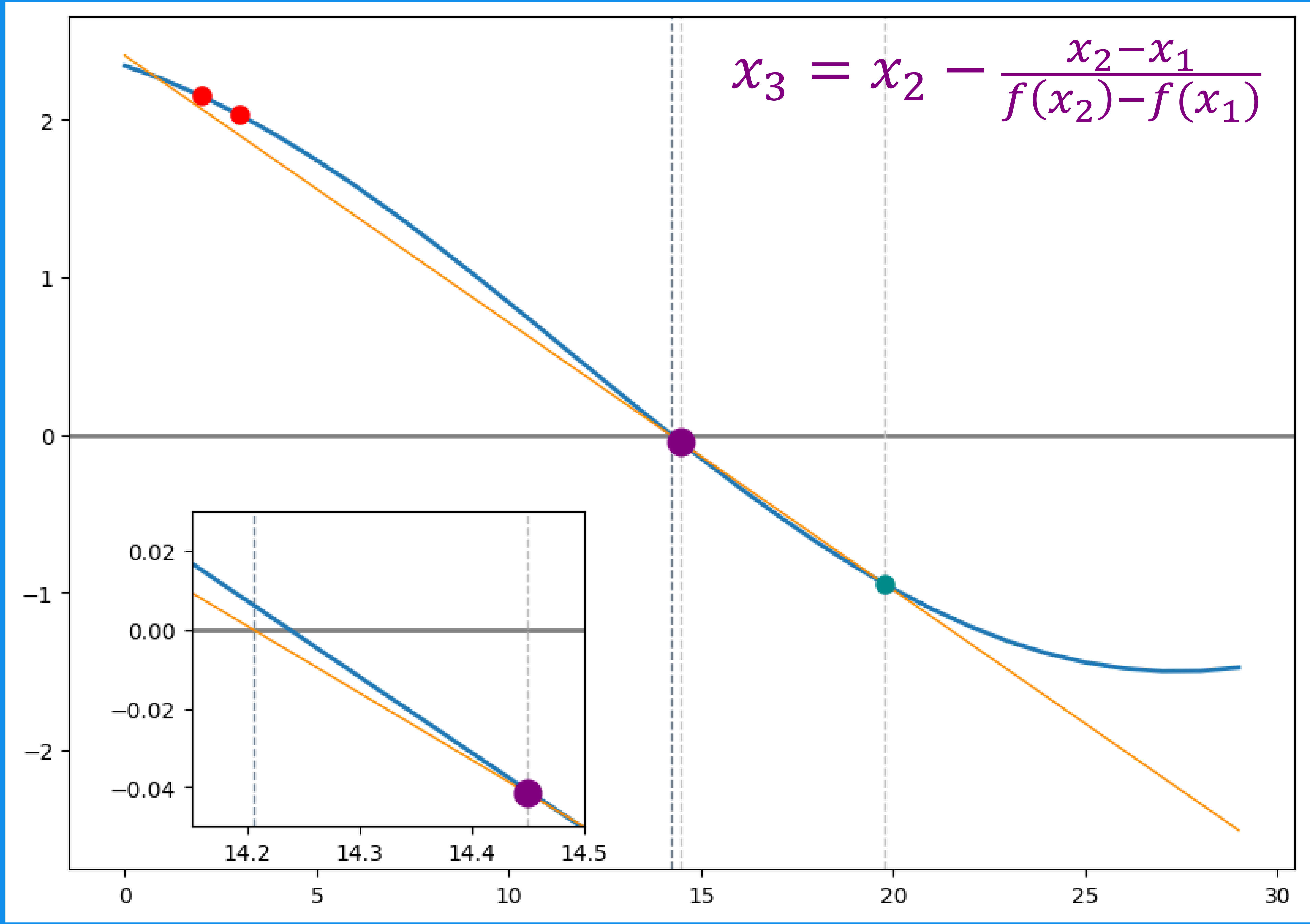
$$x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

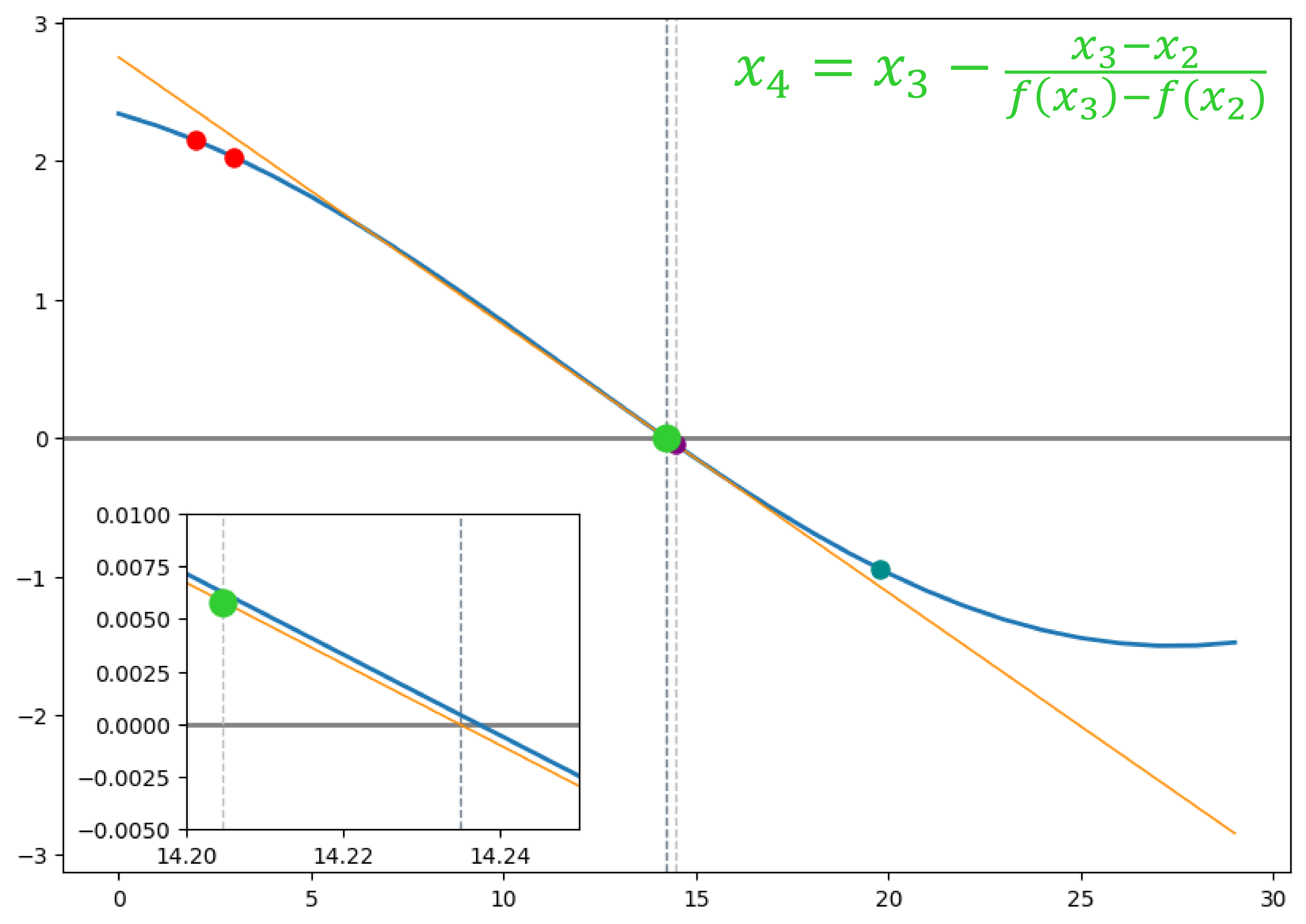
EQUAZIONI NON LINEARI – METODO DELLA SECANTE





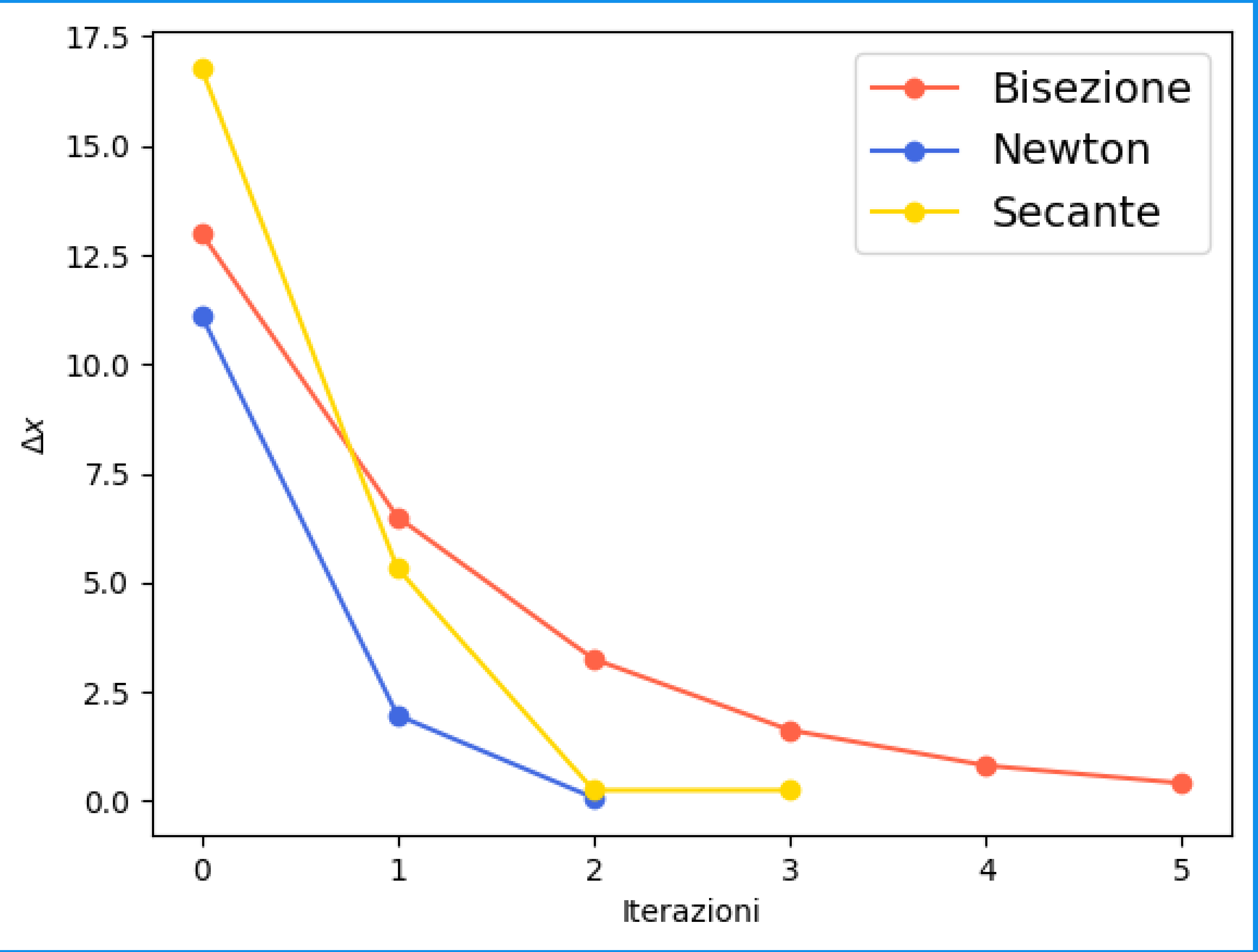
EQUAZIONI NON LINEARI – METODO DELLA SECANTE





EQUAZIONI NON LINEARI – METODO DELLA SECANTE

La rapidità di convergenza del metodo della Secante è paragonabile a quella del metodo di Newton



EQUAZIONI NON LINEARI - SOMMARIO

- Metodo del Rilassamento: $x = f(x)$

  Utilizzabile solo se $|f'(x_s)| < 1$

- Metodo della Bisezione $f(x) = 0$

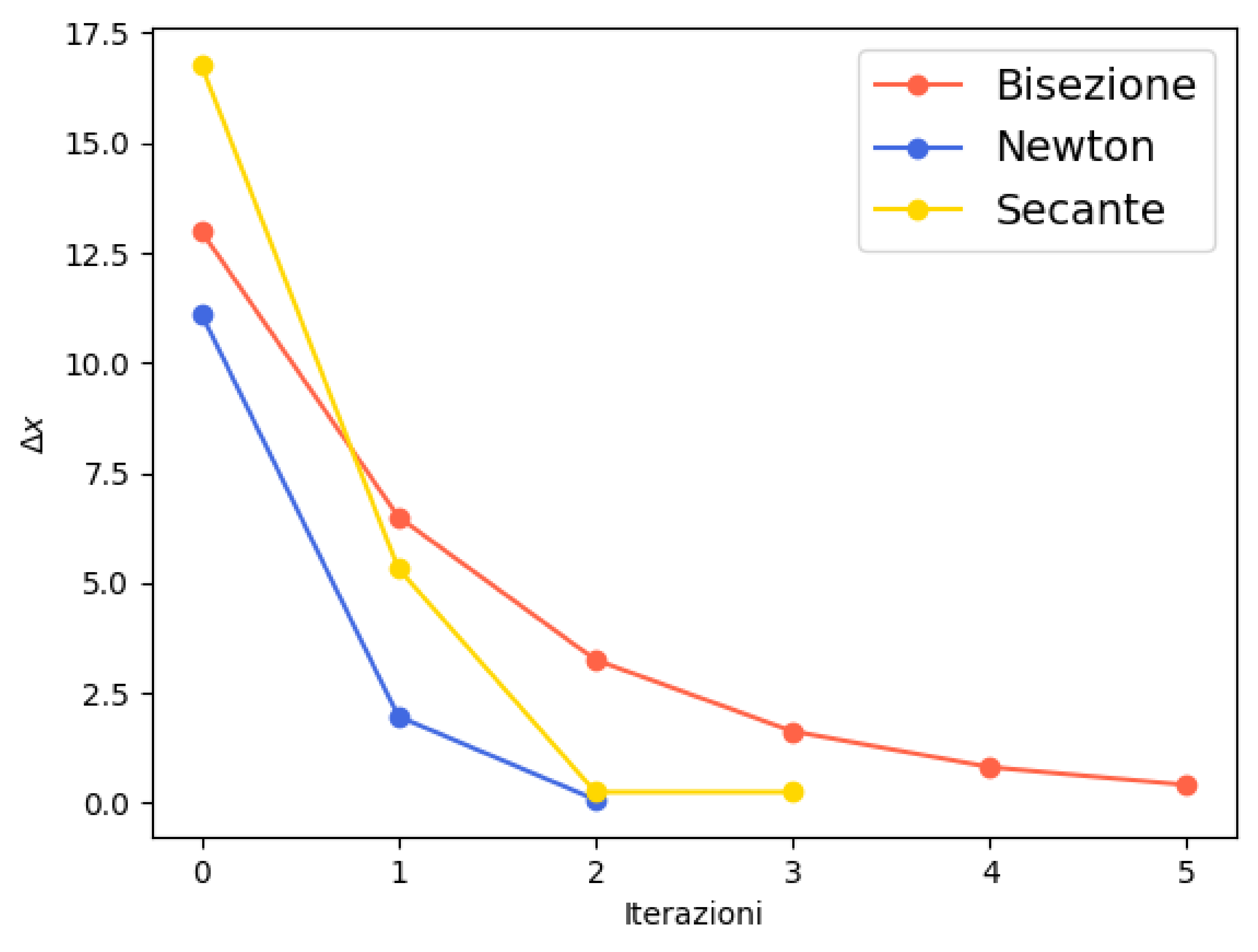
  Necessario individuare due punti iniziali di segno opposto

- Metodo di Newton $f(x) = 0$

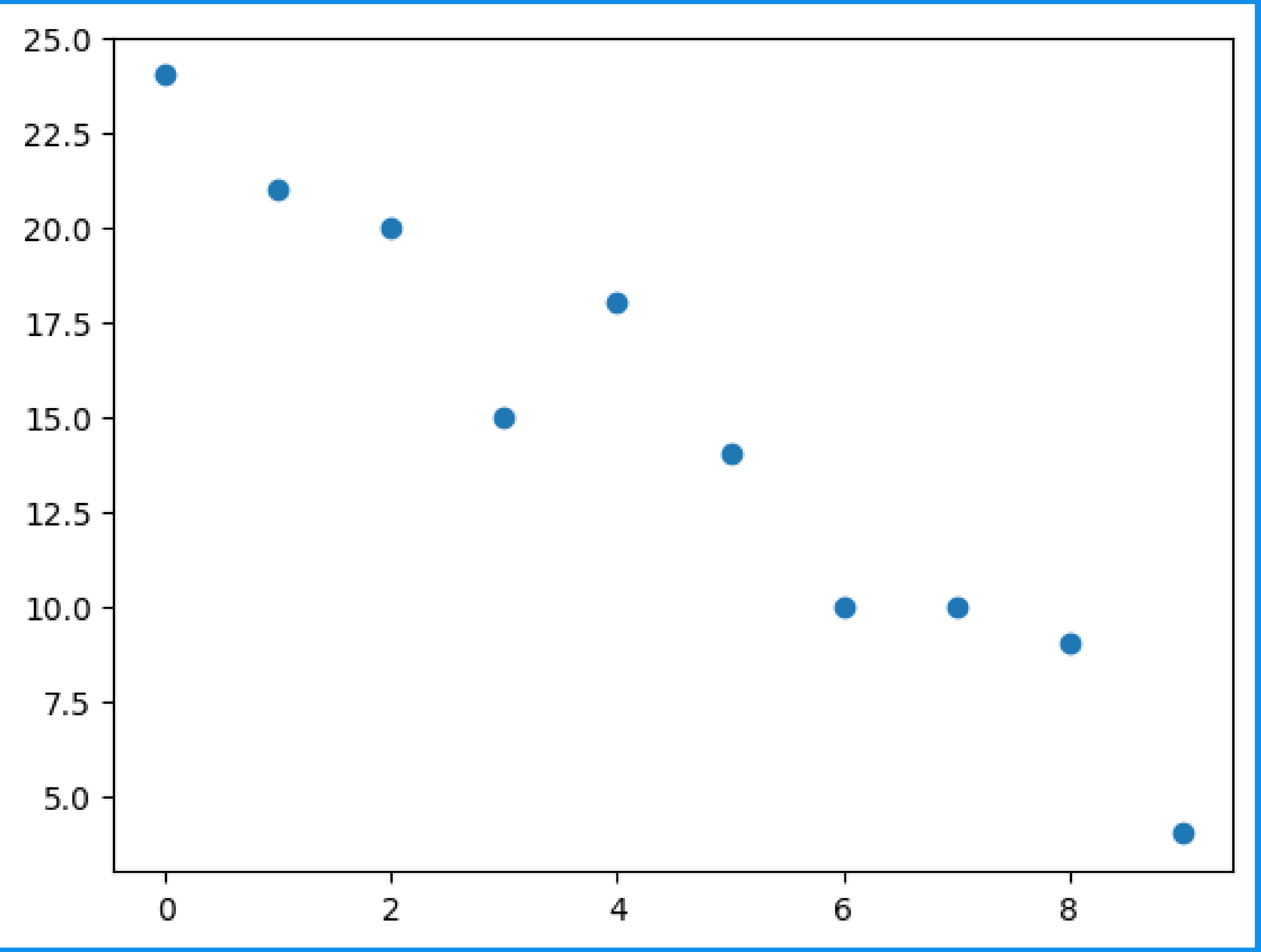
  Necessario conoscere $f'(x)$

- Metodo della Secante

  Estende metodo di Newton per casi in cui la funzione analitica $f(x)$ non è nota



MINIMIZZAZIONE – REGRESSIONE LINEARE



Assumiamo di voler trovare i parametri a e b della funzione $F(x) = ax + b$ che meglio descrivono i punti.

MINIMIZZAZIONE – REGRESSIONE LINEARE

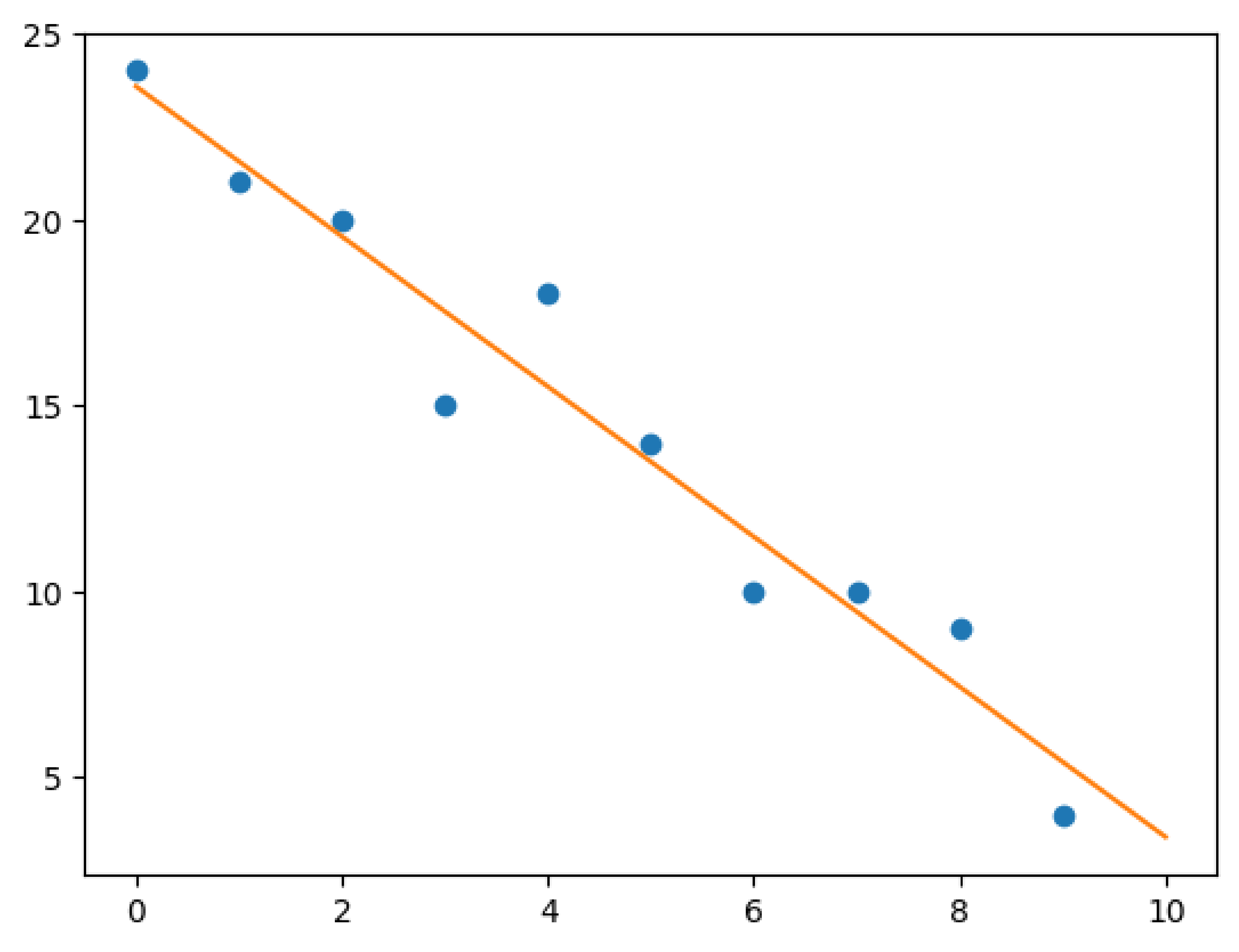
$$S(a,b) = \sum_i \left(y_i - a x_i - b\right)^2.$$

MINIMIZZAZIONE – REGRESSIONE LINEARE

$$S(a,b) = \sum_i \left(y_i - a x_i - b\right)^2.$$

$$\frac{\partial S}{\partial a} = \sum_{i=1}^{n} -2(y_i - a x_i - b)\, x_i \;= 2\left(a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i y_i\right) = 0$$

$$\frac{\partial S}{\partial b} = \sum_{i=1}^{n} -2(y_i - a x_i - b) \;= 2\left(a\sum_{i=1}^{n} x_i + nb - \sum_{i=1}^{n} y_i\right) = 0$$



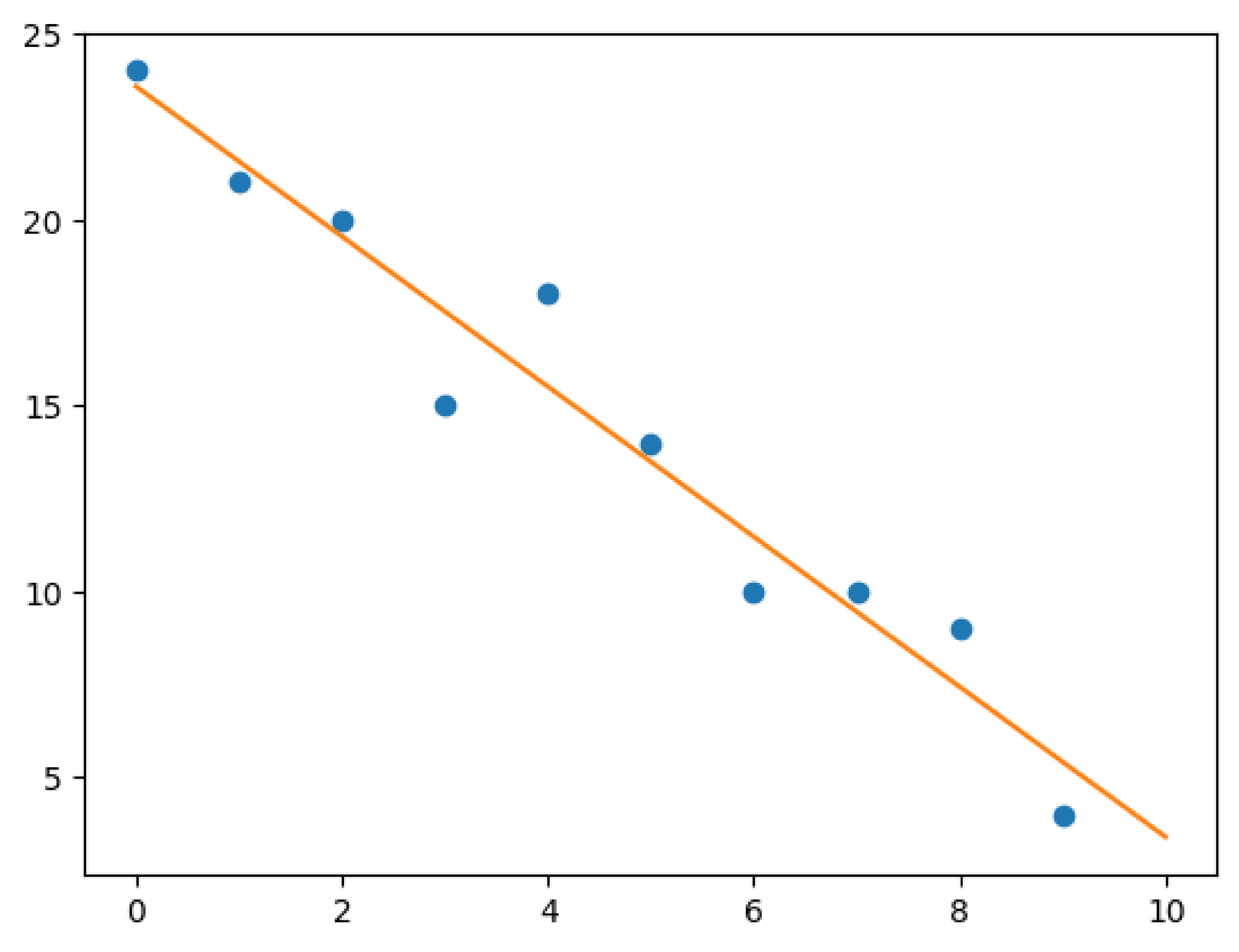
MINIMIZZAZIONE – REGRESSIONE LINEARE

$$S(a,b) = \sum_i (y_i - ax_i - b)^2.$$

$$\frac{\partial S}{\partial a} = \sum_{i=1}^{n} -2(y_i - ax_i - b)\, x_i \;= 2\left(a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i y_i\right) = 0$$

$$\frac{\partial S}{\partial b} = \sum_{i=1}^{n} -2(y_i - ax_i - b) \;= 2\left(a\sum_{i=1}^{n} x_i + nb - \sum_{i=1}^{n} y_i\right) = 0$$

$$\chi^2(a,b) = \sum_{i=1}^{n} \left(\frac{y_i - ax_i - b}{\sigma_i}\right)^2.$$



MINIMIZZAZIONE

Uscendo dal caso specifico delle funzioni lineari, il minimo o massimo di una funzione può essere individuato nel punto in cui la derivata si annulla

$$f'(x) = 0$$

che corrisponde a trovare la radice di $f'(x)$

MINIMIZZAZIONE –METODI DI GAUSS-NEWTON

Il metodo di Gauss-Newton consiste nell'applicare il metodo di Newton alla funzione derivata $f'(x)$

Newton

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

Gauss - Newton

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}.$$

MINIMIZZAZIONE – DISCESA DEL GRADIENTE

Metodo di Gauss-Newton

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}.$$

Nel caso in cui non sia possibile calcolare la derivata seconda della funzione di interesse il parametro $\frac{1}{f''(x)}$ può essere rimpiazzata con una costane (γ) positiva per la ricerca dei minimi, negativa per i massimi:

$$x_{n+1} = x_n - \gamma f'(x_n).$$

Come ordine di grandezza γ dovrebbe essere simile alla derivata seconda di interesse

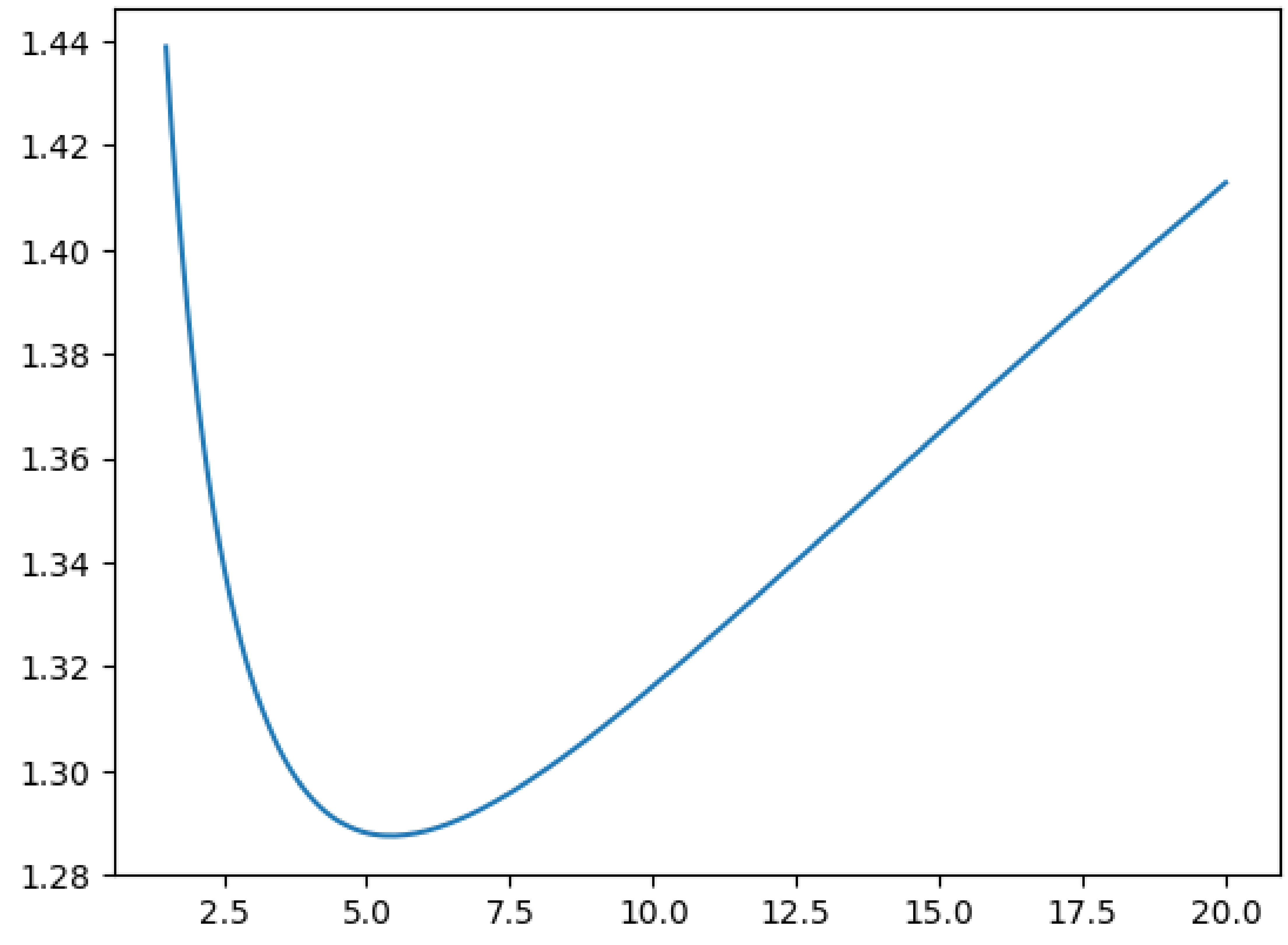
MINIMIZZAZIONE – DISCESA DEL GRADIENTE + SECANTE

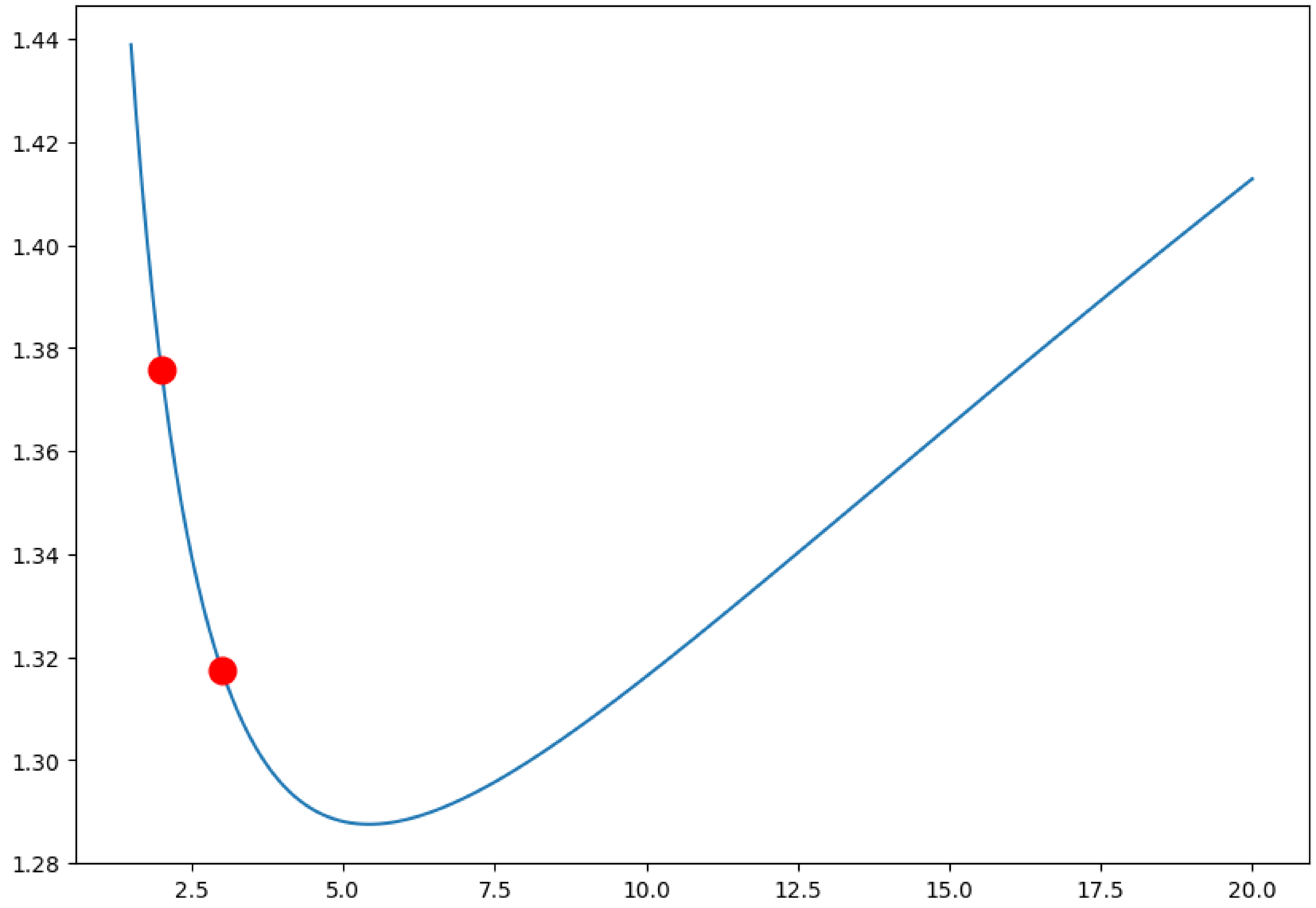
Nel caso in cui si abbiano solo dei valori e non fosse possibile calcolare o conoscere alcuna derivata si può utilizzare un metodo che combina quello della Secante per le equazioni e quello della Discesa del Gradiente approssimando la derivata prima in maniera numerica:

$$f'(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

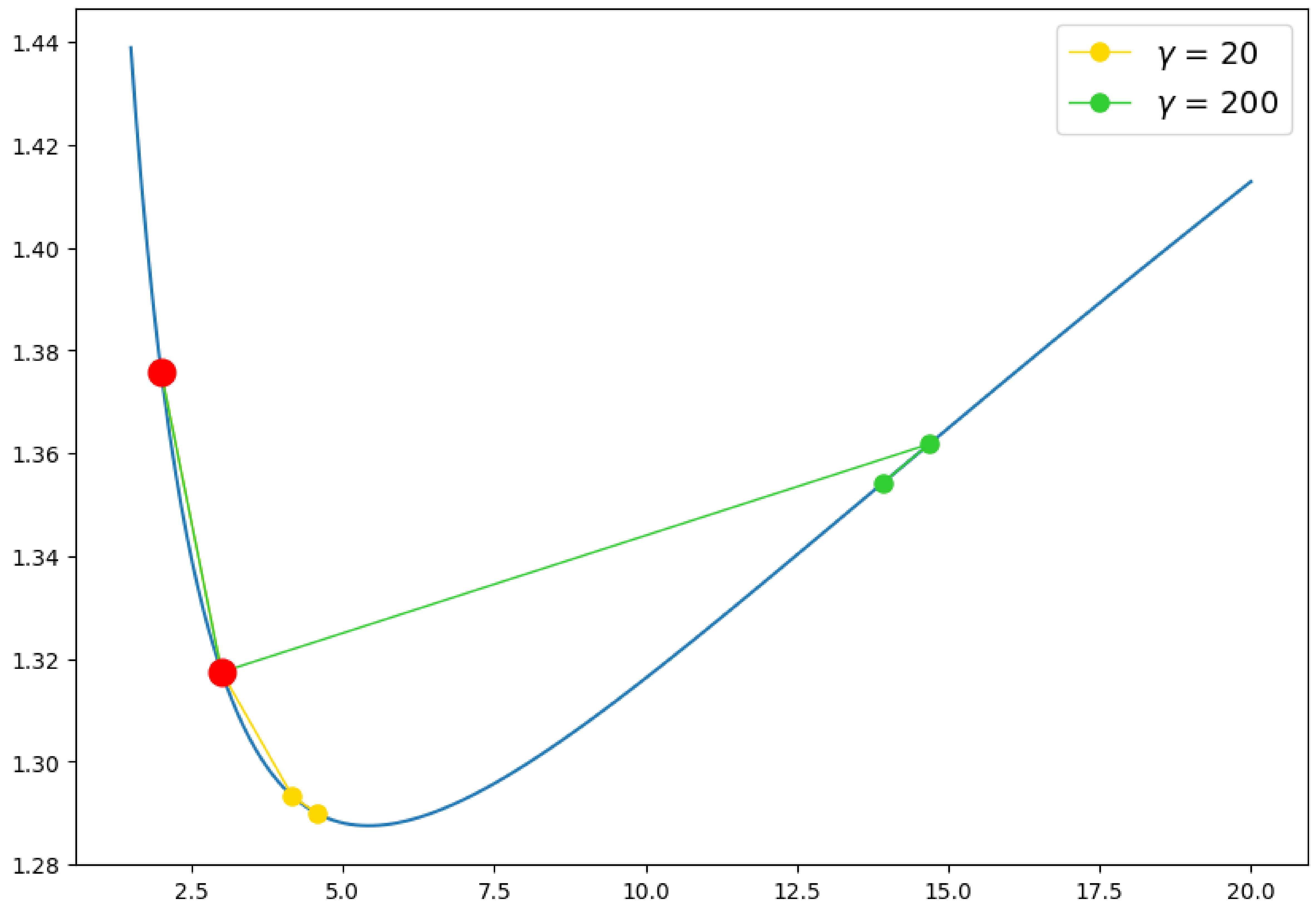
$$x_{n+1} = x_n - \gamma \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

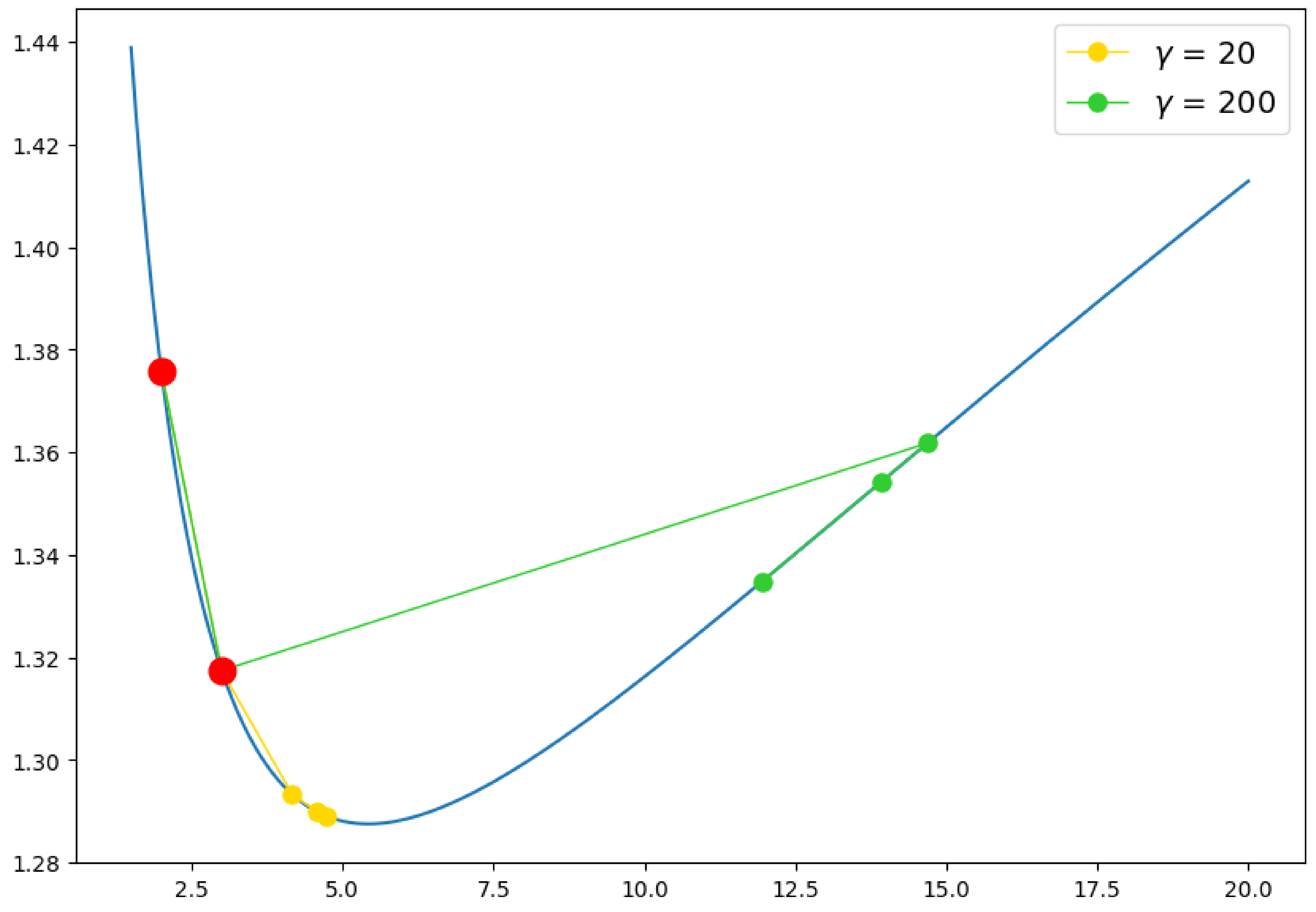
MINIMIZZAZIONE – DISCESA DEL GRADIENTE + SECANTE



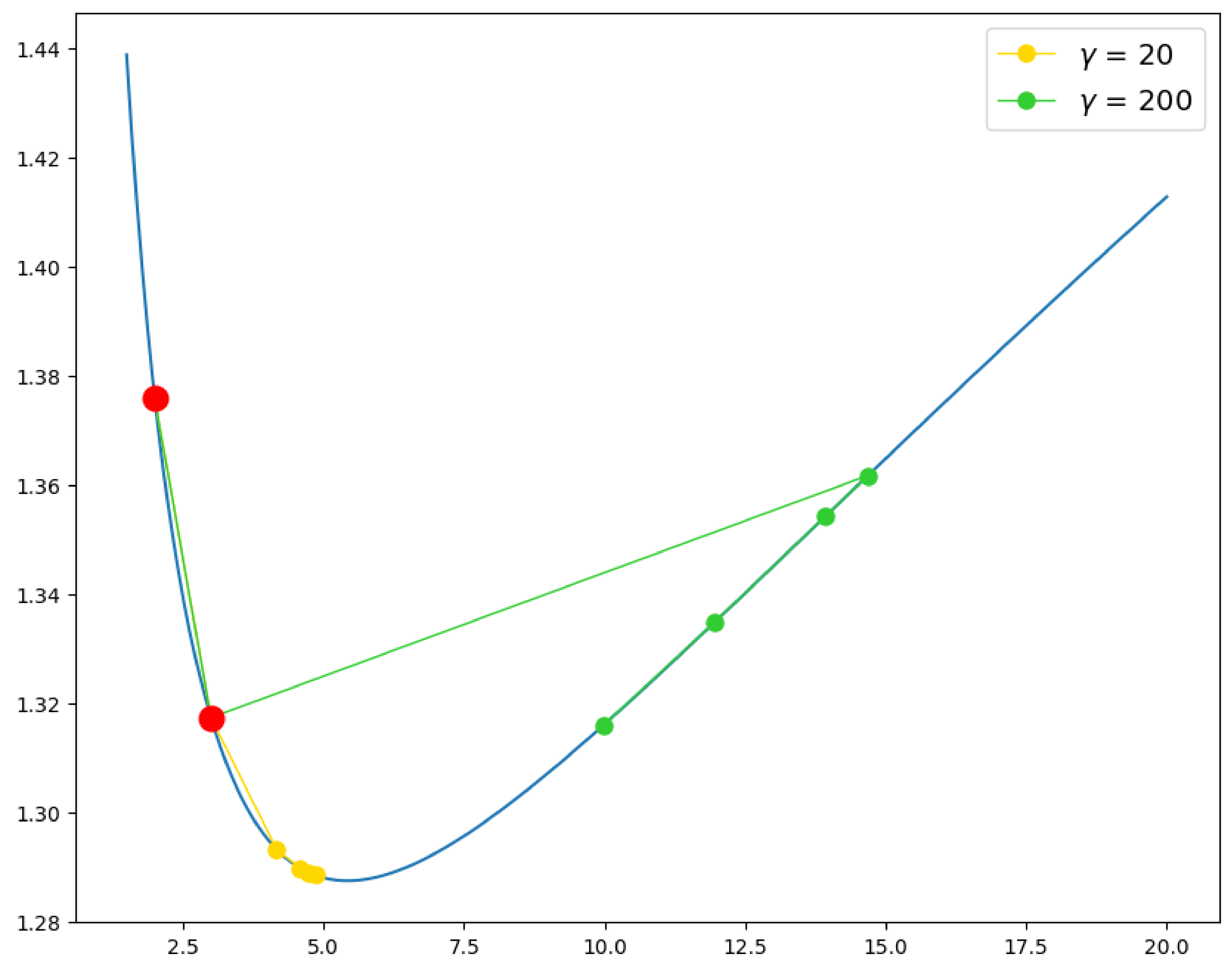


MINIMIZZAZIONE – DISCESA DEL GRADIENTE + SECANTE





MINIMIZZAZIONE – DISCESA DEL GRADIENTE + SECANTE



MINIMIZZAZIONE

- Metodo di Gauss-Newton

  Applica metodo di Newton alla derivata $f'(x) = 0$

- Discesa del Gradiente

  Estende il metodo di Gauss-Newton nel caso in cui $f''(x)$ non sia disponibile

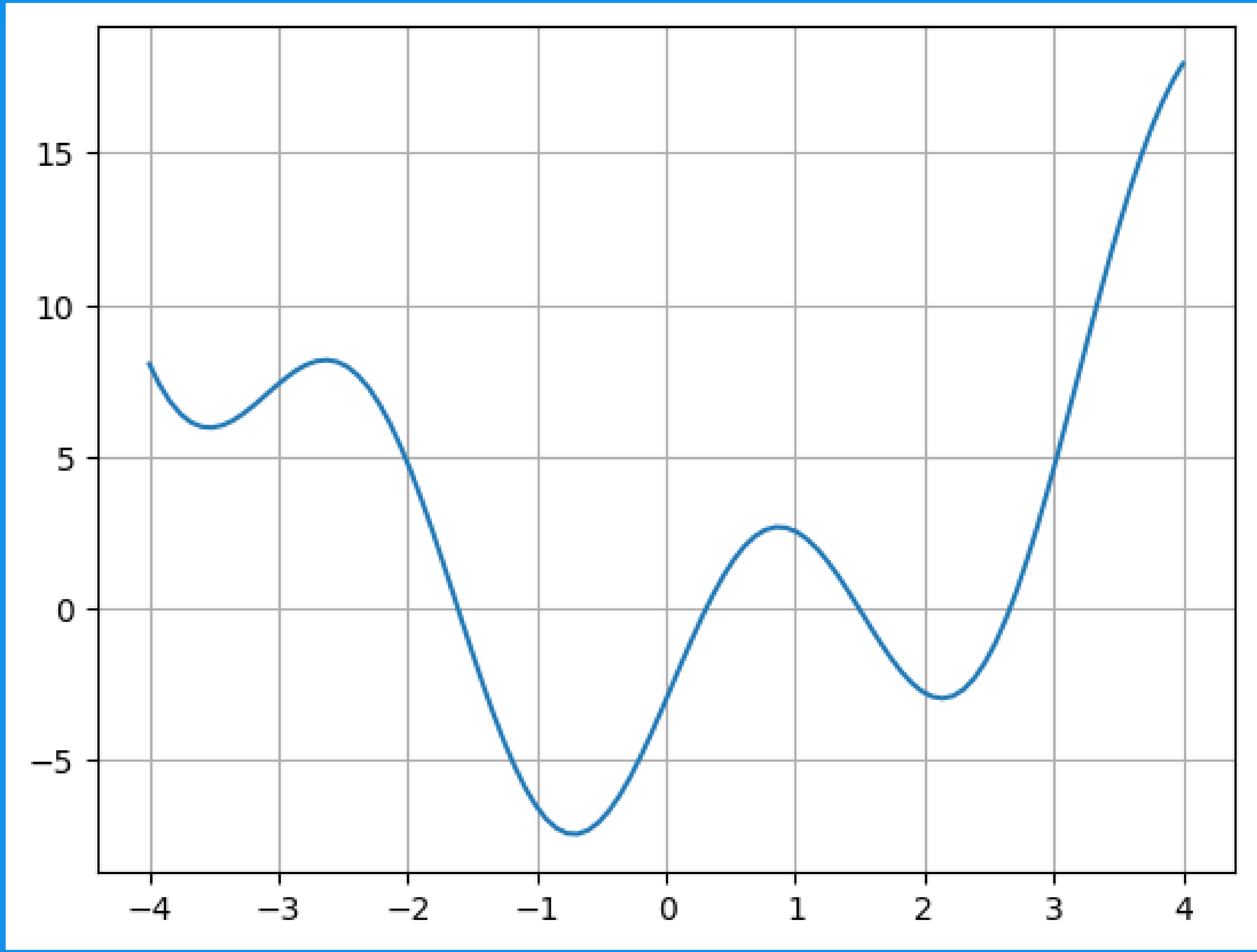
- Discesa del Gradiente + Metodo della Secante

  Combina il metodo della Discesa del Gradiente con il metodo della Secante per casi in cui la funzione analitica $f(x)$ non è nota

MINIMIZZAZIONE – SCIPY

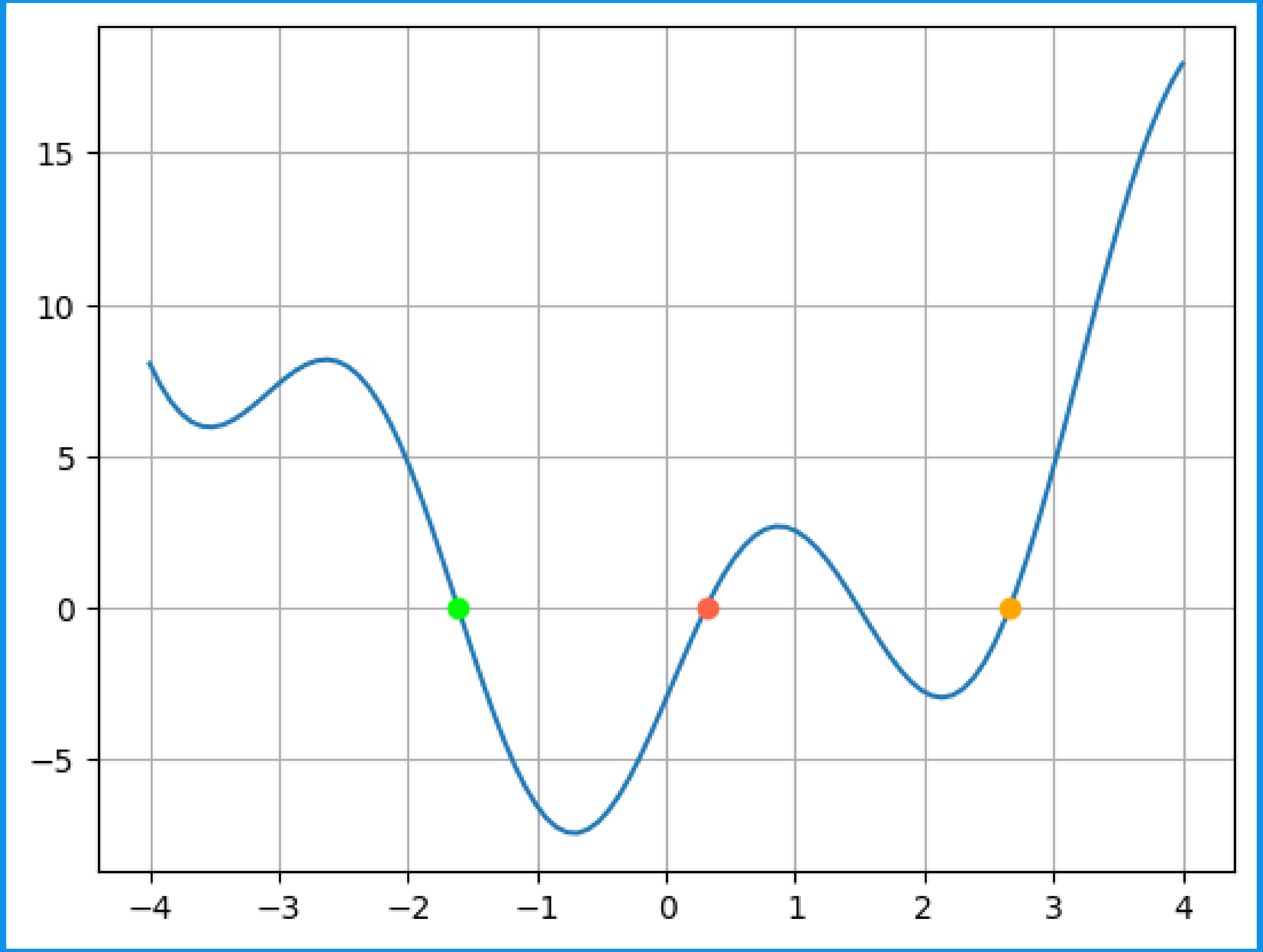
```
from scipy import optimize
```

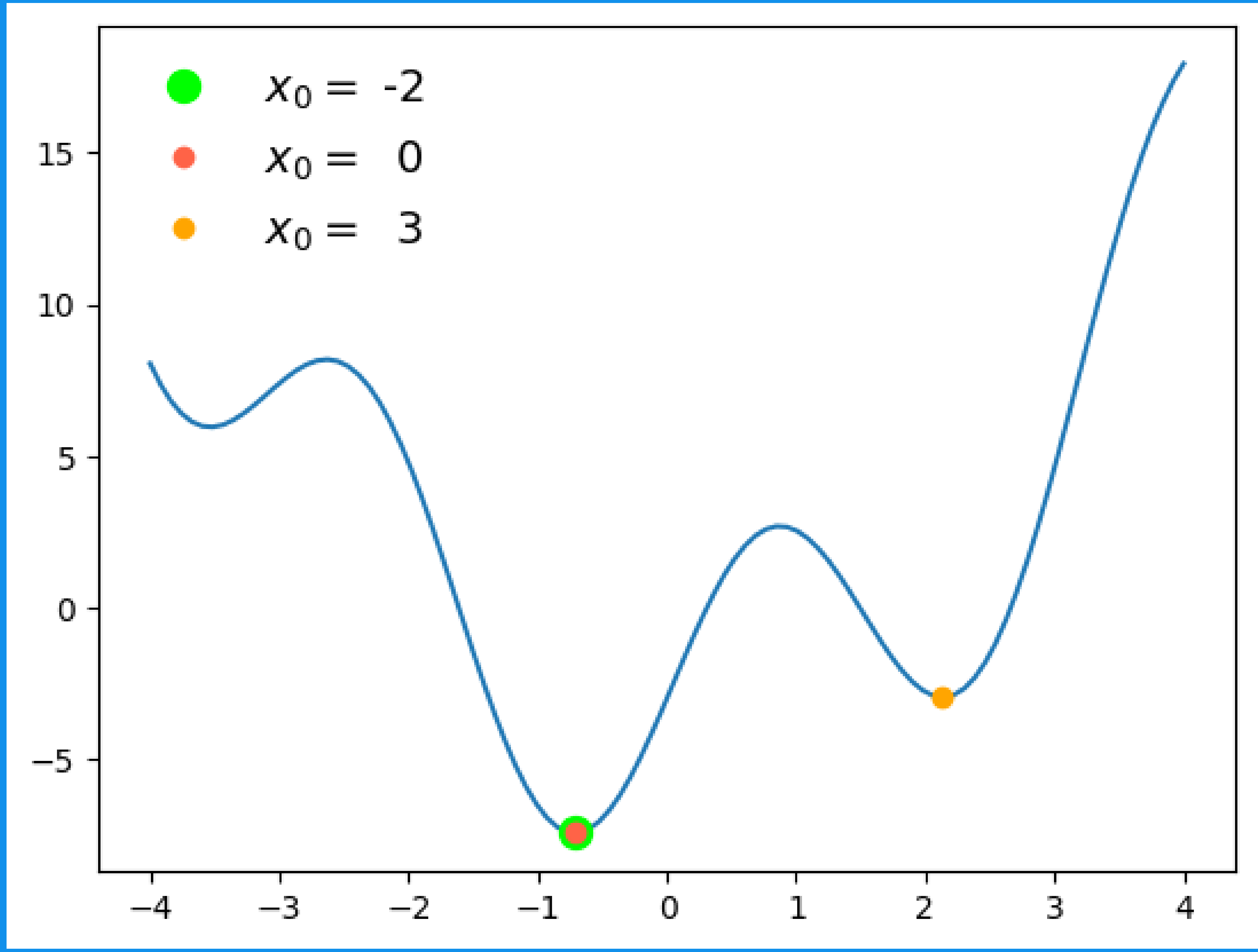
```
# Funzione non lineare
def fp4(x, p ):
    """
    fp4(x, p )
    return p0 x^2 + p1 sin(p2 x) x + p3
    """
    return p[0]*x**2 + p[1]*np.sin(p[2]*x) + p[3]
```



MINIMIZZAZIONE – SCIPY

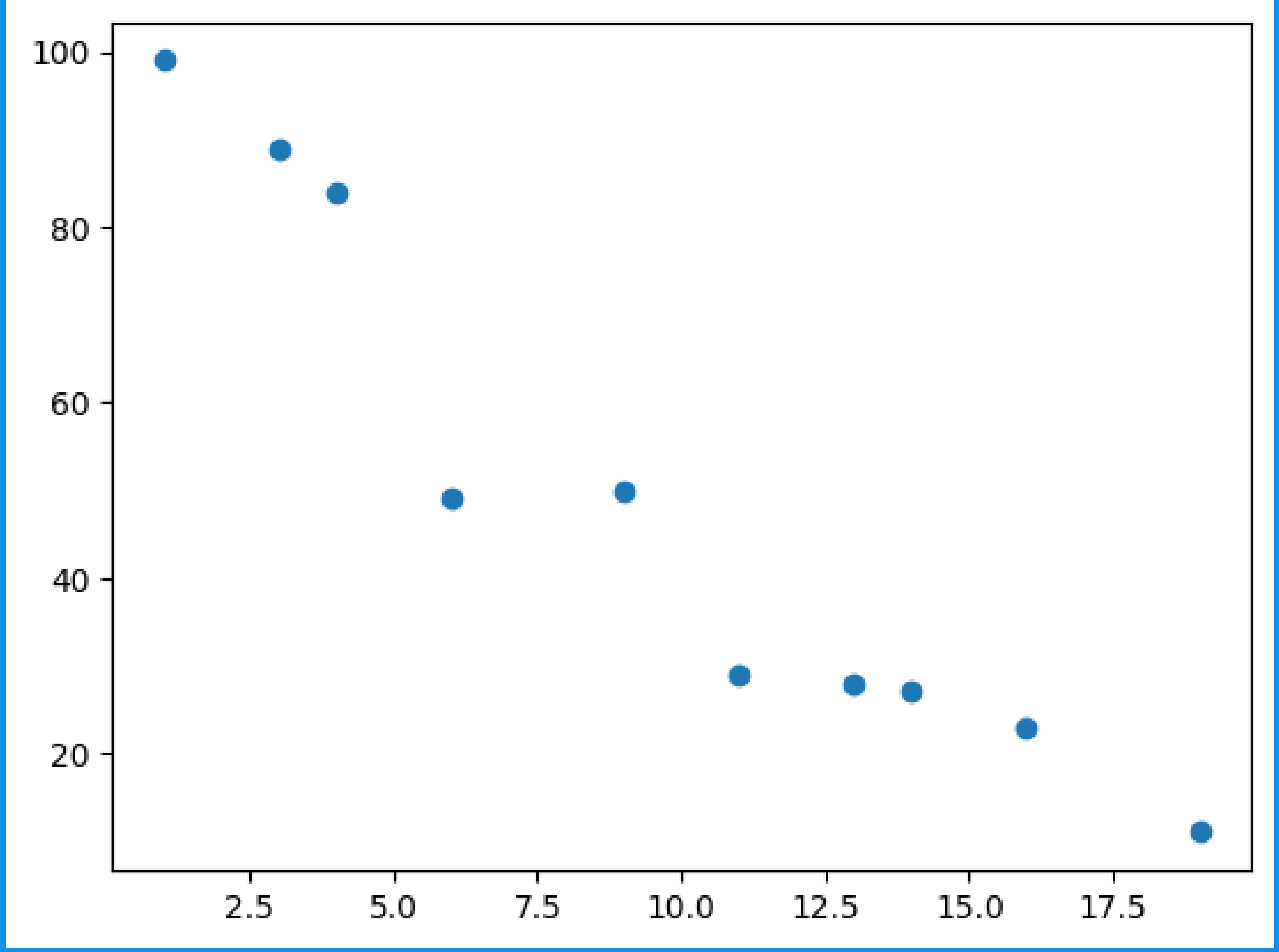
```
rmin_m2 = optimize.minimize(fp4, x0=-2, args=ppar4, method="L-BFGS-B")
rmin_0 = optimize.minimize(fp4, x0=0, args=ppar4)
rmin_p3 = optimize.minimize(fp4, x0=3, args=ppar4)
```





MINIMIZZAZIONE – SCIPY

```python
from scipy import optimize
```

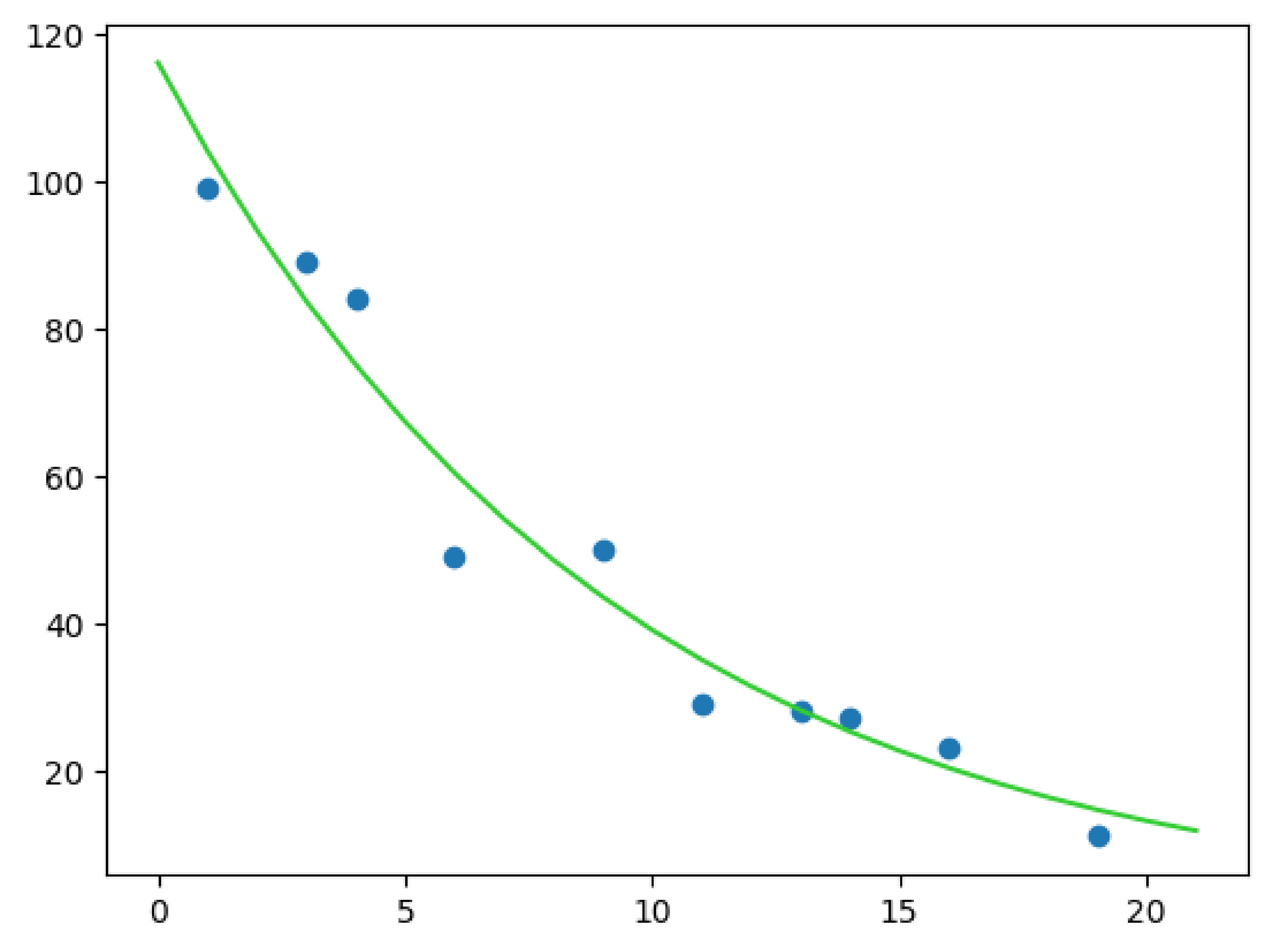


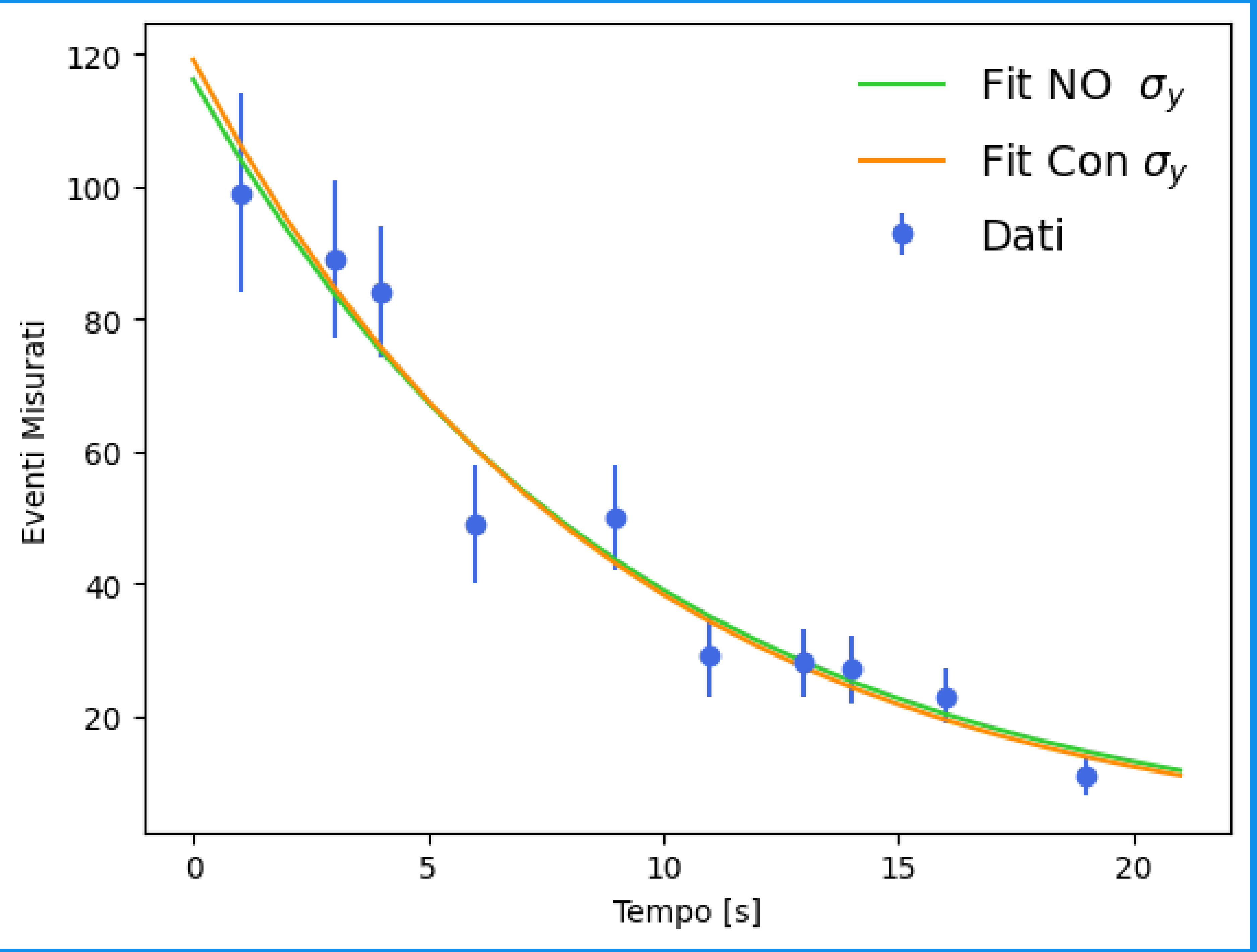

```python
def fexp(x, A, tau):
    """
    Fnzione esponenziale f(x) = A*e^(-x/tau)
    A   : valore della funzionea t=0
    tau : costante di tempo
    """
    return A*np.exp(-x/tau)
```

MINIMIZZAZIONE – SCIPY

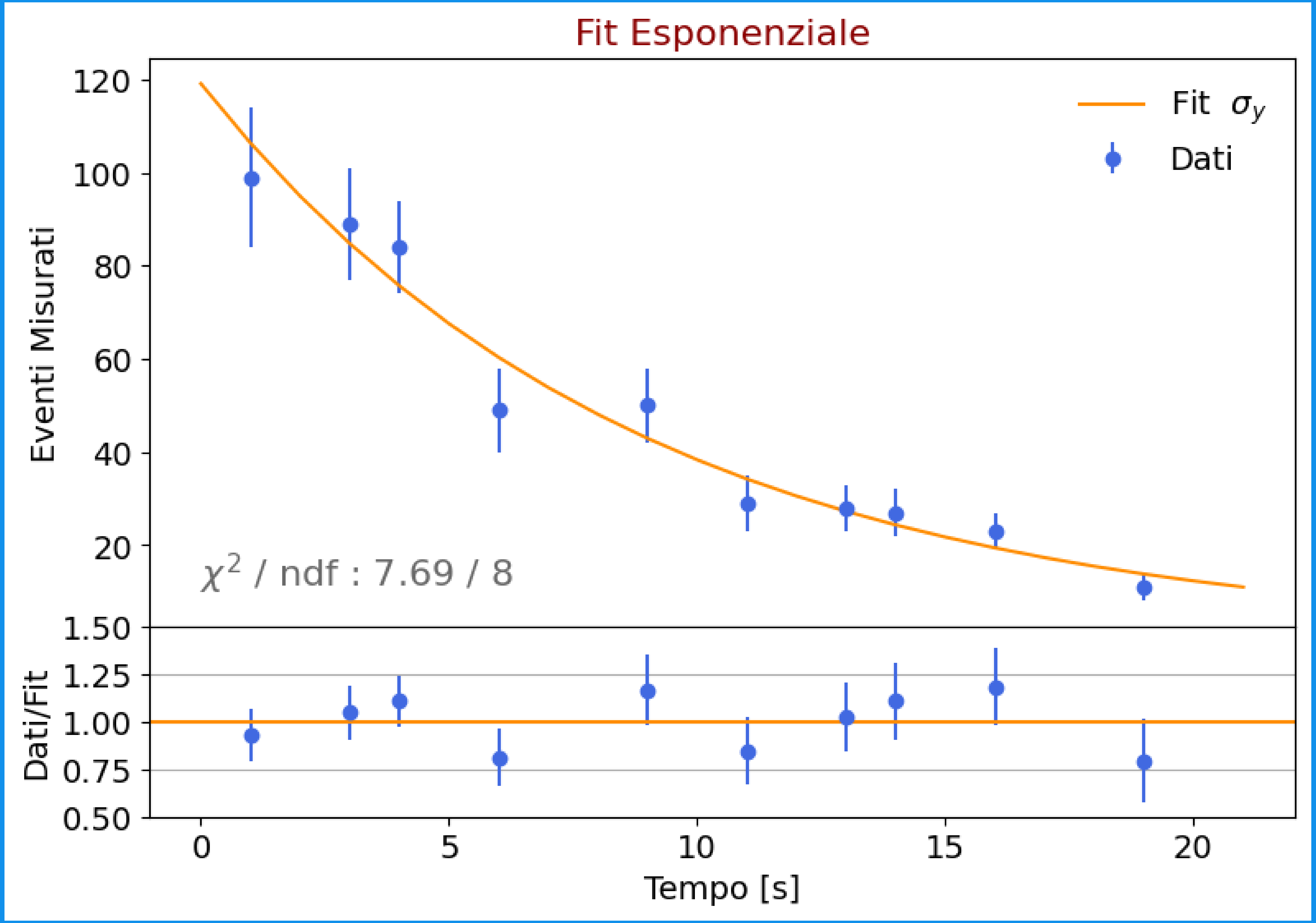
```
# Fit per trovare parametri
pstart = np.array([10, 1])
params, params_covariance = optimize.curve_fit(fexp, xdata, ydata, p0=[pstart])
```

```
params [116.14394111   9.17144608]
params_cov [[42.91414238 -4.09111438]
 [-4.09111438  0.7003829 ]]
errori params [6.55088867 0.83688882]
```





MINIMIZZAZIONE – SCIPY



MINIMIZZAZIONE – SCIPY

Per esempi con il codice per l’utilizzo dei pacchetti di minimizzazione vedere il notebook della Lezione L07:

https://github.com/s-germani/metodi-computazionali-fisica-2025/blob/main/notebooks/lezioni/L07_Equazioni_Minimizzazione.ipynb