labcalc_eser_04

November 12, 2025

1 Esercitazione 04: Stima di radice quadrata $\sqrt{a}$ con il metodo babilonese

La successione ricorsiva $r_{n+1} = \frac{1}{2}(r_n + a/r_n)$ converge al valore di $\sqrt{a}$

La scelta del termine r_0 e‘ ininfluente.

Invece la successione $y_{n+1} = \frac{3}{2}y_n - \frac{a}{2}y_n^3$ converge a $1/\sqrt{a}$. La scelta di y_0 e‘ molto importante per evitare divergenze e tipicamente viene stimata con un paio di iterazione della succesione r_n ponendo quindi $y_0 = 1/r_2$.

2 Programma in python

il codice in python non va compilato ma interpretato al momento di esecuzione dall'interprete.

Noi useremo come interprete `python3`.

Il seguente codice viene salvato in un file con il nome `convergenza.py` e viene eseguito dal terminale con il comando

```
python3 convergenza.py
```

---

programma `convergenza.py`

```
# carica moduli pyplot, numpy e matematica di python
import matplotlib.pyplot as plt
import numpy as np
import math as m

plt.title("Calcolo della radice quadrata con il metodo babilonese")

# carica dati dal file
# iter e` un array con i dati nella prima colonna (iterazione)
# stima e` un array con i dati della seconda colonna (stima di pigreco)
iter, stima = np.loadtxt('dati.txt', unpack=True)

# crea grafica di stima in funzione di iter
plt.plot( iter, stima, 'x-', label='stima con il metodo babilonese')
```

```
# specifica limite inferiore superiore per asse x e y con numeri opportuni in base ai vostri d
# per esempio se state stimando la radice di a = 15 con circa 5 iterazioni
plt.xlim(0, 10)
plt.ylim(1., 1000.)
plt.yscale('log')

# aggiungi legenda per gli assi
plt.xlabel('numero iterazione')
plt.ylabel('stima radice a')

# linea orizzontale rossa al valore di sqrt(a) usando la libreria matematica
plt.axhline( y=m.sqrt(15), color = 'red', linestyle='--')

# nome del file in cui salvare il grafico
plt.savefig("radice.png")

# mostra grafico
plt.show()
```

vediamo ora il significato del codice python usato

3 File di dati

suppniamo di girare il programma con i seguenti parametri

```
Mac:Esercitazioni rahatlou$ gcc -o /tmp/app babilon-radice.c
Mac:Esercitazioni rahatlou$ /tmp/app
Calcolo radice quadrata di a con precisione eps.
Precisione eps in [0.000001,0.100000]: 0.000002
Numero di cui calcolare la radice a>0 : 3.4
Primo termine della successione r0!=0: 2.1
Metodo babilonese
#iter    rn
  1   1.8595238095
  2   1.8439744528
  3   1.8439088926
  4   1.8439088915
Stima radice quadrata di 3.400000: 1.8439088915
Numero totale di iterazioni:   4
|rn - sqrt(a)|: 0
Metodo alternativo
#iter    1/yn
  1  419.5023264024
  2  279.6700186892
  3  186.4493807618
  4  124.3036396427
  5  82.8751718470
  6  55.2592328595
```

```
 7  36.8531665783
 8  24.5892966223
 9  16.4236491279
10  10.9952976485
11  7.3995649461
12  5.0373095903
13  3.5152107653
14  2.5801172243
15  2.0729995502
16  1.8770279539
17  1.8447653082
18  1.8439094875
19  1.8439088915
Stima radice quadrata di 3.400000: 1.8439088915
Numero totale di iterazioni:  19
|1/yn - sqrt(a)| = 2.8888e-13
```

Salviamo i risultati per la successione y_n in un file di dati chiamato `babilon.txt`

4 caricamento di moduli pyplot, numpy e matematica di python

in python al posto delle `librerie` si usano i `moduli` - `math`: libreria matematica - `numpy`: modulo per calcolo vettoriale ottimizzato - `matplotlib.pyplot`: ricca libreria di funzioni per creare grafici ed istogrammi

e' uso comune in python dare nomi corti (massimo 2 o 3 lettere) quando uno importa una libreria.

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import math as m
```

5 File di dati

abbiamo un file di dati **babilon.txt** che contiene due valori per ciascuna riga - numero di iterazione - valore di $1/y_n$ (stima di $\sqrt{a}$)

```
1  419.5023264024
2  279.6700186892
3  186.4493807618
4  124.3036396427
5  82.8751718470
6  55.2592328595
...
```

6 Lettura dati dal file

usando la funzione `loadtxt()` del modulo `numpy` possiamo creare due array di dati

- `iter`: numero di iterazioni

- `stima`: $1/y_n$

NB: lavorando sul vostro computer dalla riga di comando se il file `babilon.txt` si trova nella stessa cartella dove si trova anche il codice python, non serve specificare la path

[2]:
```
iter, stima = np.loadtxt('babilon.txt', unpack=True)
```

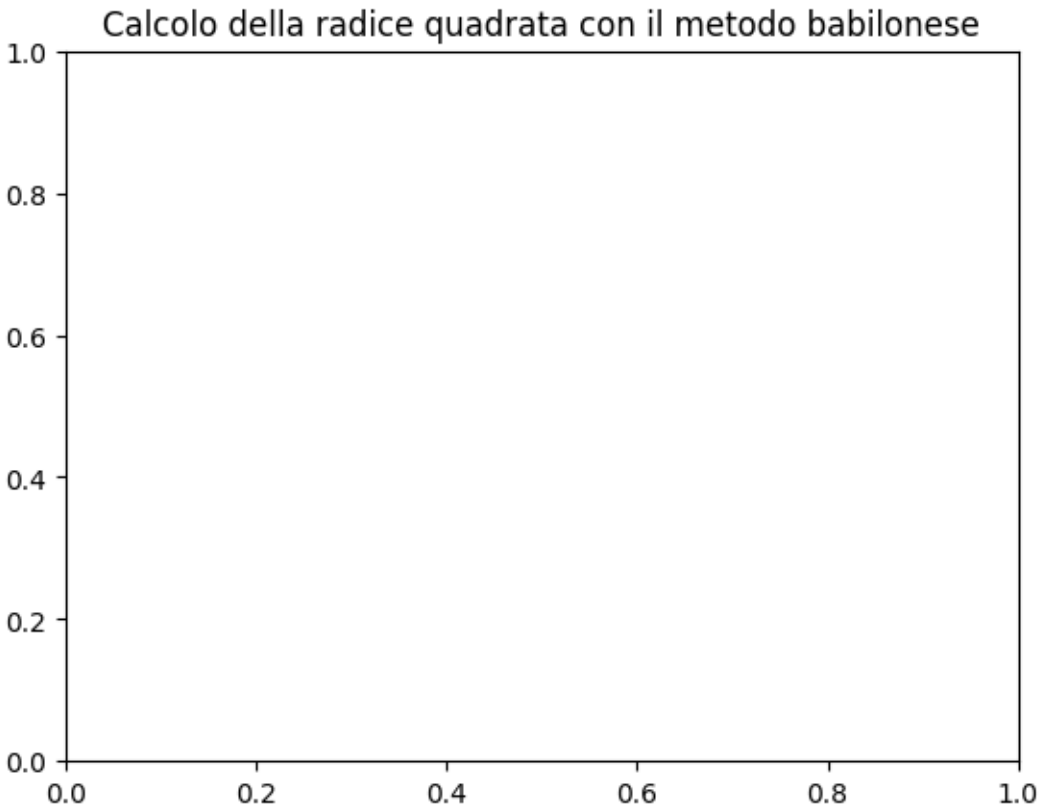
7 Titolo del grafico

per prima cosa diamo un titolo al grafico che vogliamo creare usando la funzione `title()` del modulo `plt` caricato in precedenza. Questo si fa chiamando `plt.title()`.

Il titolo appare in alto sulla figura.

[3]:
```
plt.title("Calcolo della radice quadrata con il metodo babilonese")
```

[3]:
```
Text(0.5, 1.0, 'Calcolo della radice quadrata con il metodo babilonese')
```



8 Grafica di `stima` in funzione di `iter`

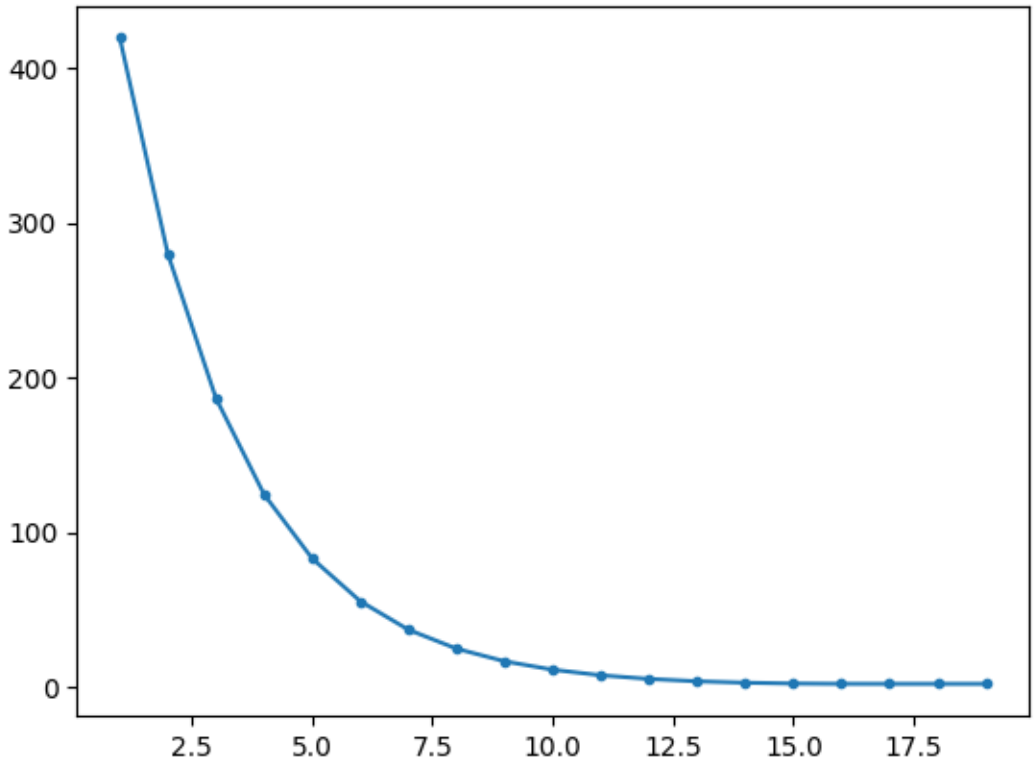
La funzione `plt.plot()` ha i seguenti argomenti (e altri argomenti opzionali qui non usati)

- `iter`: valori sulll'asse x
- `stima`: valore sull'asse y per ciascun valore sull'asse x
- 'o-': mette una una pallina per ciascun dato e connette i dati con un trattino
- `label`: legenda per quel che si trova nel grafico

Potete porvare altri stili per il grafico: `'-.'`, `'.-'`, `'o-'`, `'o'`, `'x'`, `'+'`, `'x-'`

[4]:
```
plt.plot( iter, stima, '.-')
```

[4]:
```
[<matplotlib.lines.Line2D at 0x10bd28b20>]
```

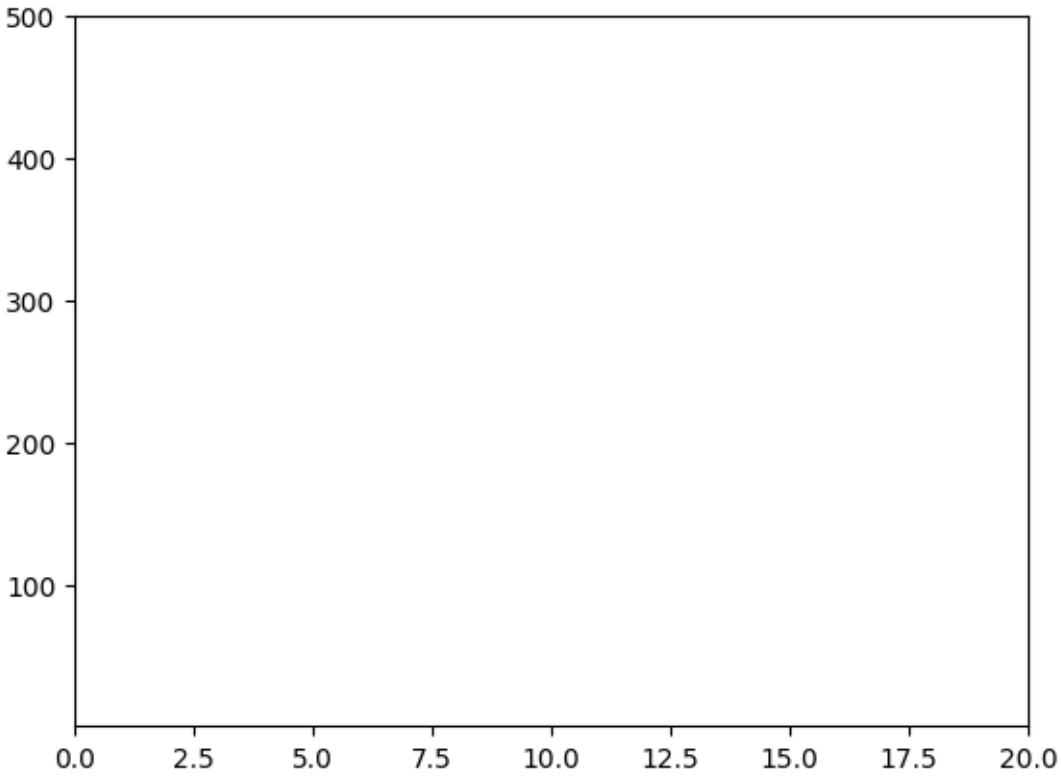


9 Minimo e massimo per asse x e y

- asse x: tra [0,20] dato che abbiamo 19 iterazioni
- asse y: tra [1, 500] dati i valori nel file

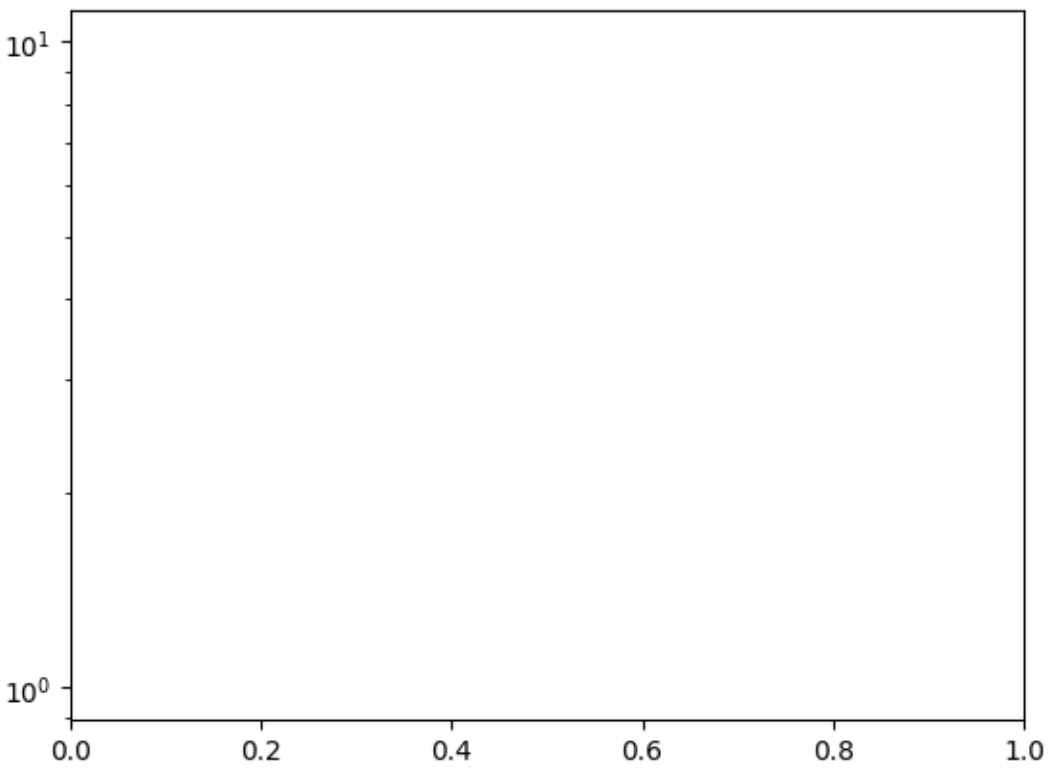
[5]:
```
plt.xlim(0, 20)
plt.ylim(1., 500.)
```

[5]:
```
(1.0, 500.0)
```



per apprezzare le piccole variazioni su un intervallo cosi‘ grande, usiamo la scala logaritma per l’asse y

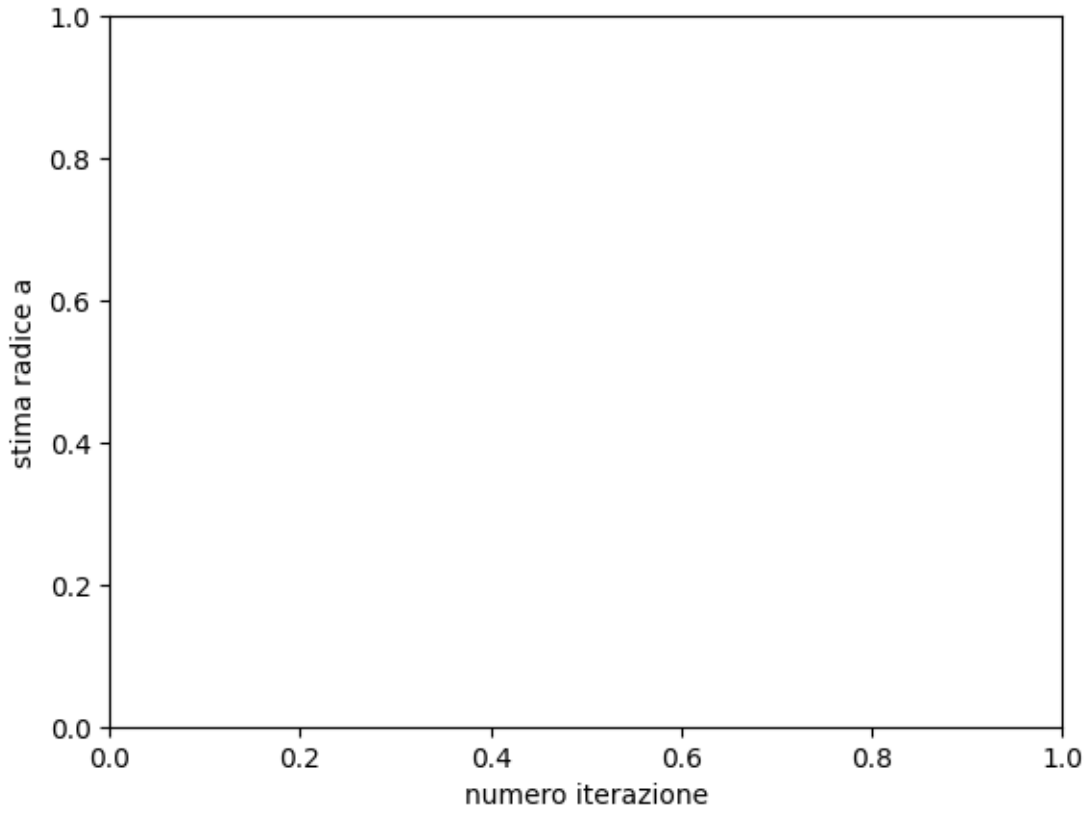
[6]:
```
plt.yscale('log')
```



10 Titolo per asse x e y

[7]:
```
plt.xlabel('numero iterazione')
plt.ylabel('stima radice a')
```

[7]:
```
Text(0, 0.5, 'stima radice a')
```



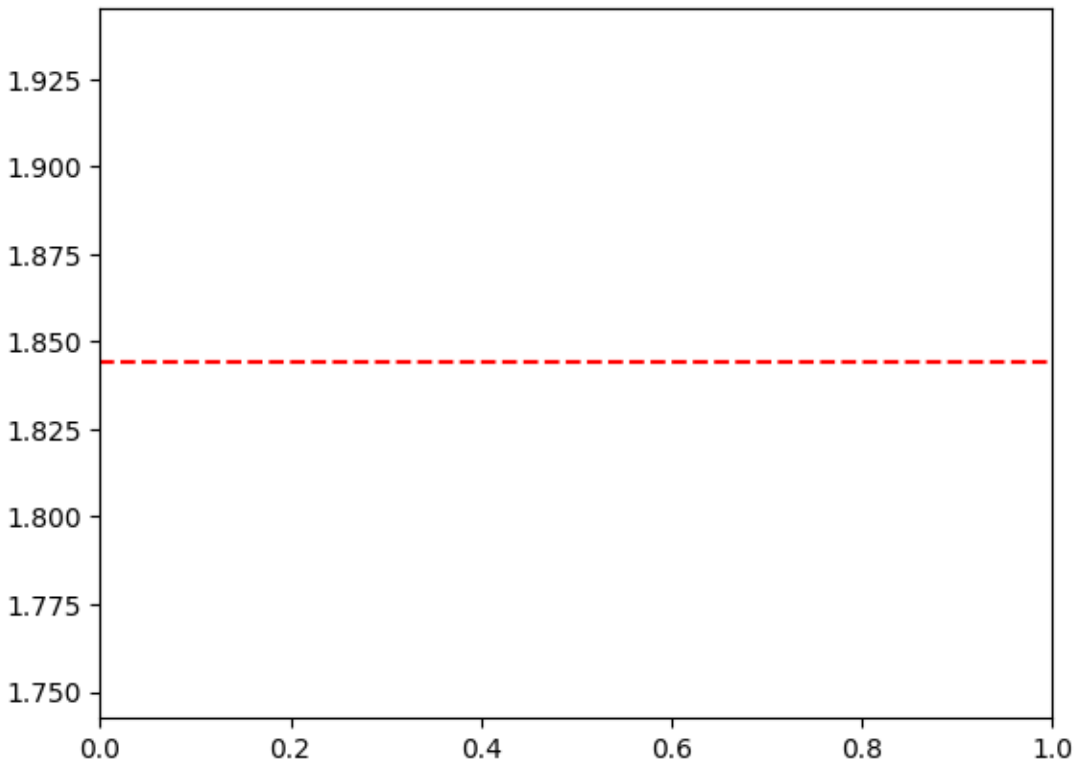
11 Linea orizzontale

disegniamo una linea orizzonatale per indicare il valore vero di $\sqrt{a}$ usando la funzione `m.sqrt()` della libreria matematica con $a = 3.4$.

Gli argomenti della funzione sono: - `y`: valore dell'ordinata y dove tracciare la retta - `color`: colore rosso per la retta - `linestyle`: stile della retta `'--'` (tratteggiata) oppure `'-.'` o altri stili combinando i simboli - `label`: legenda per quel che si trova nel grafico

[8]:
```
a = 3.4
plt.axhline( y=m.sqrt(a), color = 'red', linestyle='--', label='sqrt di lib␣
 ↪mat')
```

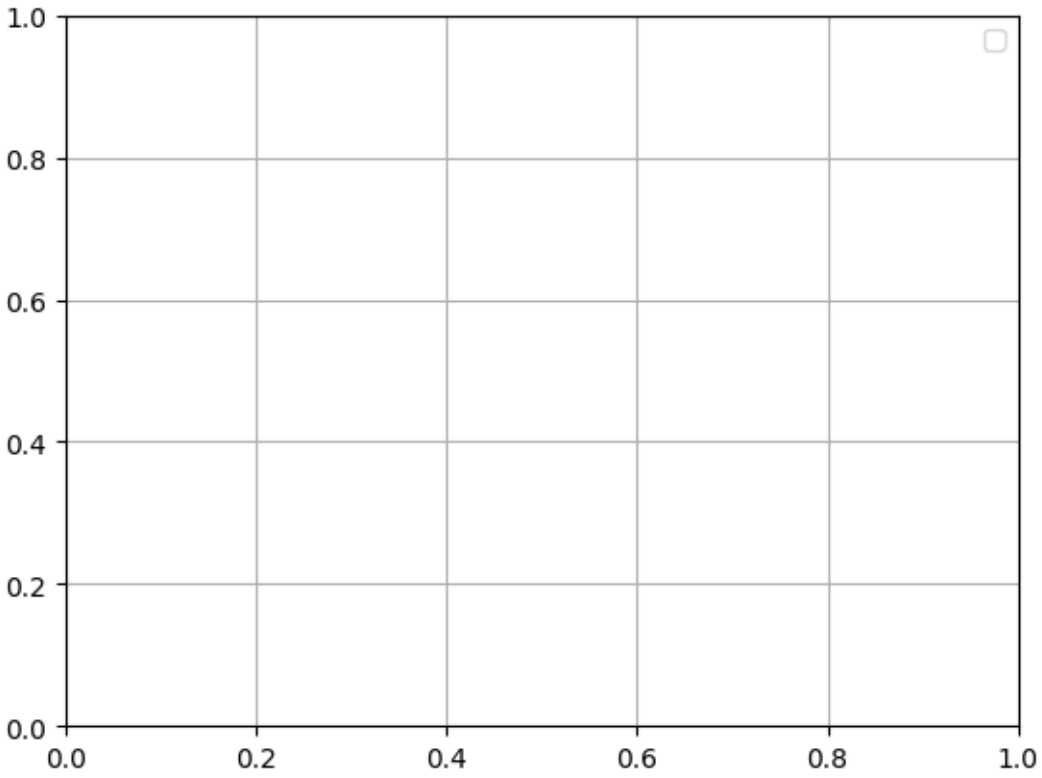
[8]:
```
<matplotlib.lines.Line2D at 0x10c03f1c0>
```



12 Legenda e griglia sul grafico

[9]:
```
plt.legend()
plt.grid()
```

```
/var/folders/wr/vnt8sxxx60g698kqqpt7p4180000gn/T/ipykernel_79299/1741845256.py:1
: UserWarning: No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is called with no
argument.
  plt.legend()
```



13 Mostrare la figura

la funzione `plt.show()` apre la finestra grafica quando eseguite il programma dal terminale con `python3 convergenza.py`

[10]:
```
plt.show()
```

14 salvataggio del grafico su file

`plt.savefig()` salva il contenuto della finestra grafica sul disco in un file chiamato `radice.png`

[11]:
```
plt.savefig("radice.png")
```

```
<Figure size 640x480 with 0 Axes>
```

15 Creazione del grafico con tutte le specifiche

la seguente cella equivale a quel che ottenete girando il codice python dal terminale con il comando

`python3 convergenza.py`

[12]:
```
from os import minor
a = 3.4
plt.plot( iter, stima, '.-', label='stima con il metodo babilonese')
plt.axhline( y=m.sqrt(a), color = 'red', linestyle='--', label='sqrt di lib␣
 ↪mat')

plt.legend()
plt.grid()

plt.xlim(0, 20)
plt.ylim(1., 500.)

plt.xlabel('numero iterazione')
plt.ylabel('stima radice a')

plt.yscale('log')
```