```
import math
import numpy as np
from scipy import linalg
import matplotlib.pyplot as plt
import itertools
```

```
# writing the 1st order richardson iteration
def richardson(A, alpha, b):
  x = np.zeros(A.shape[1])
  C = (np.identity(A.shape[0]) - alpha*A)
  # sys = A@x
  while np.linalg.norm(b-A@x,2) > .000001:
    x = C@x + alpha*b

  print(x)
```

```
# testing ricahrdson on small scale system with known solution, should produce
# vector [0,1]

# works for alpha > .1, less than 1
A = np.array([[1,1], [0,2]])
b = np.array([1,2])


# w, v = np.linalg.eig(A)
# alpha = (2 / w[-1] + w[0]) should minimize norm(I-alpha*A) to acheive fastest
# convergence according to cs.yale.edu lec15
```

```
richardson(A,.5, b)
```

```
[9.53674316e-07 1.00000000e+00]
```

```
# testing ricahrdson on small scale system with known solution, should produce
# vector [3,1,1]
A = np.array([[2,-4,-1], [1,-3,1], [3,-5,-3]])
print(A)
alpha= 0
print(np.linalg.norm(np.identity(A.shape[0]) - alpha*A))

# this system will not converge under this richardson since norm(I-alpha*A) >1
# for all alpha, even though known solution, 1st order always
# converges if this quantity is less than 1.

# w, v = np.linalg.eig(A)

b = np.array([1,1,1])
print(b)

print(A@np.array([3,1,1]))
# the above two lines should be the same
# richardson(A,.5, b)
```

```
[[ 2 -4 -1]
 [ 1 -3  1]
 [ 3 -5 -3]]
1.7320508075688772
[1 1 1]
[1 1 1]
```

✓ 0s completed at 2:26 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.