In [1]:
```
import numpy as np
import cmath
import math
import matplotlib.pyplot as plt
import csv
import random
random.seed(123)
```

Lab Book 1

In [2]:
```
z= 1+1j

print(z.real)
print(z.imag)
```

```
1.0
1.0
```

Lab Book 2

In [3]:
```
print(cmath.exp(1j * math.pi))

# check ten random numbers and return the result
for i in range(0,10):
    r = random.random()
    theta = random.random()
    left = r * cmath.exp(1j * theta)
    right = r * (math.cos(theta) + 1j * cmath.sin(theta))
    print((left - right) == 0)
```

```
(-1+1.2246467991473532e-16j)
True
True
True
True
True
True
True
True
True
True
```

Lab Book 3

In [4]:
```
a = np.linspace(1,10,10)
print(a)
a[1:len(a)-1] = 0
print(a)
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
[ 1.  0.  0.  0.  0.  0.  0.  0.  0. 10.]
```

Lab Book 4

Numpy will not be able to deal with such computation as it is meaningless and undefined in mathmatics and python will return error when excuting it. This is not a sensible choice because the dot product between two matrics A(n*m) and B(i*j) A@B can only be operated when m=i.

Lab Book 5

In [5]:
```
def decompositionVector(x, y):
    v1 = y.dot(x) / np.linalg.norm(y) ** 2 * y
    v2 = x - v1
    return v1, v2
x = np.array([2.0, -3.0])
y = np.array([1.0, 1.0])
v1,v2 = decompositionVector(x,y)
print(v1,v2)
```

```
[-0.5 -0.5] [ 2.5 -2.5]
```

Lab Book 6

In [6]:
```
c = np.linspace(1,100,100)
c = c.reshape(10,10)
csquare = c[0:2,0:2]
print(c)
print(np.linalg.det(csquare))
```

```
[[  1.   2.   3.   4.   5.   6.   7.   8.   9.  10.]
 [ 11.  12.  13.  14.  15.  16.  17.  18.  19.  20.]
 [ 21.  22.  23.  24.  25.  26.  27.  28.  29.  30.]
 [ 31.  32.  33.  34.  35.  36.  37.  38.  39.  40.]
 [ 41.  42.  43.  44.  45.  46.  47.  48.  49.  50.]
 [ 51.  52.  53.  54.  55.  56.  57.  58.  59.  60.]
 [ 61.  62.  63.  64.  65.  66.  67.  68.  69.  70.]
 [ 71.  72.  73.  74.  75.  76.  77.  78.  79.  80.]
 [ 81.  82.  83.  84.  85.  86.  87.  88.  89.  90.]
 [ 91.  92.  93.  94.  95.  96.  97.  98.  99. 100.]]
-10.000000000000002
```

Lab Book 7a

In [7]:
```
def trace(A):
    temp = np.ones(A.shape[0])
    diagMatrix = np.diag(temp)
    return np.sum(diagMatrix * A)
print(trace(csquare))
```

```
13.0
```

Lab Book 7b

In [8]:
```
A = c[0:4,0:5]
print(A)
left = trace(A.T @ A)
right = np.sum(A ** 2)
print(left - right)
```

```
[[ 1.  2.  3.  4.  5.]
 [11. 12. 13. 14. 15.]
 [21. 22. 23. 24. 25.]
 [31. 32. 33. 34. 35.]]
0.0
```

Lab Book 8

Lab Book 7c

$A \in \mathbb{R}^{m,n}$ $\quad m\{\underbrace{\begin{bmatrix} & & \end{bmatrix}}_{n}$ $\quad n\{\underbrace{\begin{bmatrix} & \end{bmatrix}}_{m} A^T$

$$A^T \cdot A = \begin{bmatrix} A_{11} & \cdots & A_{m1} \\ \vdots & A_{22} & \\ \vdots & & \ddots \\ A_{1n} & & A_{mn} \end{bmatrix} \cdot \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & A_{22} & \ddots \\ A_{m1} & & A_{mn} \end{bmatrix}$$

identity trace $(A^T A) = \sum_1^m A_{i1}^2 + \sum_1^m A_{i2}^2 + \cdots + \sum_1^m A_{in}^2$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2$$

In [9]:

```
A1 = np.array([[2.0,3.0],[3.0,4.5]])
b1 = np.array([3.5, 6.5])
x1 = np.linalg.solve(A1,b1)
print(x1)
A2 = np.array([[1.0,1.0],[1.0,1.0]])
b2 = np.array([3.5, 6.5])
x2 = np.linalg.solve(A2,b2)
print(x2)
```

```
[ 7.50599938e+15 -5.00399959e+15]

---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-9-c1e2b0f8d845> in <module>
      5 A2 = np.array([[1.0,1.0],[1.0,1.0]])
      6 b2 = np.array([3.5, 6.5])
----> 7 x2 = np.linalg.solve(A2,b2)
      8 print(x2)

<__array_function__ internals> in solve(*args, **kwargs)

~/opt/anaconda3/lib/python3.7/site-packages/numpy/linalg/linalg.py in solve(a, b)
    397     signature = 'DD->D' if isComplexType(t) else 'dd->d'
    398     extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
--> 399     r = gufunc(a, b, signature=signature, extobj=extobj)
    400 
    401     return wrap(r.astype(result_t, copy=False))

~/opt/anaconda3/lib/python3.7/site-packages/numpy/linalg/linalg.py in _raise_linalgerror_singular(err, flag)
     95 
     96 def _raise_linalgerror_singular(err, flag):
---> 97     raise LinAlgError("Singular matrix")
     98 
     99 def _raise_linalgerror_nonposdef(err, flag):

LinAlgError: Singular matrix
```

For some certain singular matrix python will return error of singular matrix and for most cases python will return an extremly small value which also means the given matrix is singular.

Lab Book 9

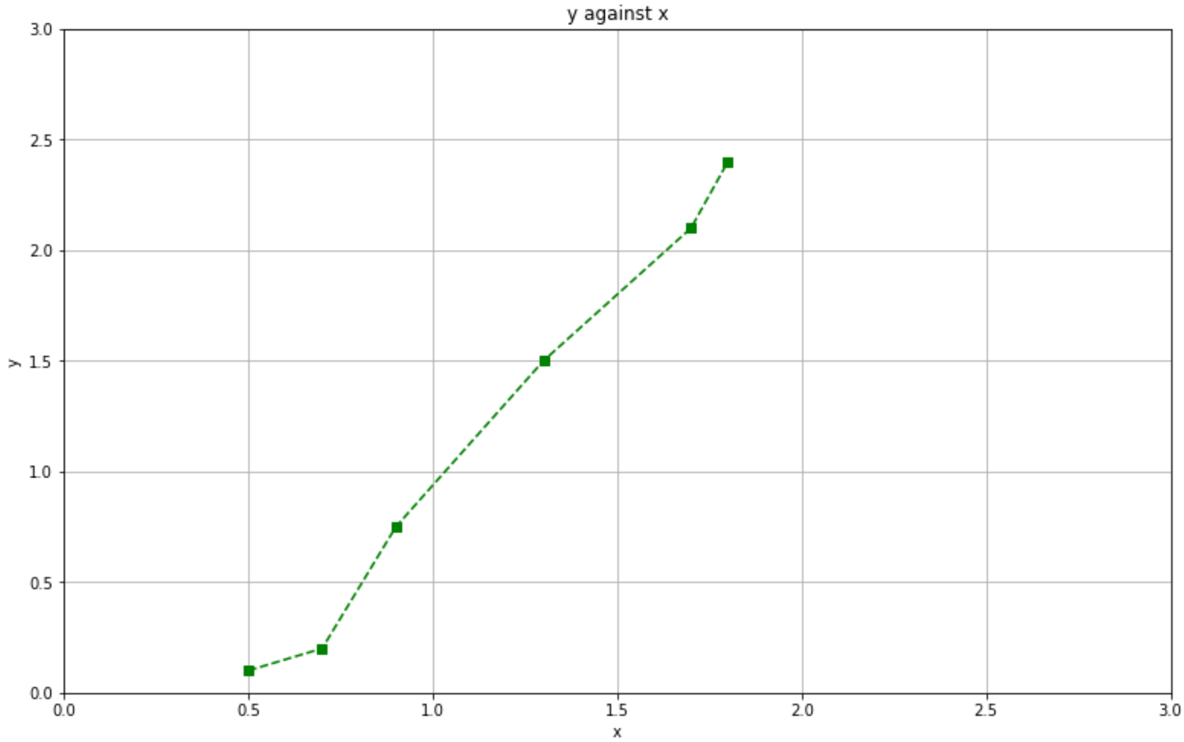
In [10]:
```
def geneHilbert(size):
    A = np.zeros([size, size])
    for i in range(size):
        for j in range(size):
            A[i][j] = 1 / (i + j + 1)
    return A
A = geneHilbert(15)
b = A[:,0]
x1 = np.linalg.solve(A,b)
x2 = np.linalg.inv(A) @ b
print(x1)
print(x2)
print(x1 - x2)
```

```
[ 1.  0.  0.  0.  0.  0. -0.  0. -0.  0. -0. -0.  0. -0.  0.]
[-2.16378957e-01  1.47670870e+01 -5.12857056e+01  4.34057617e+01
  1.24613281e+02 -3.44265625e+02  3.32843750e+02 -1.07750000e+02
 -5.71875000e+01  6.15000000e+01 -1.80000000e+01  1.00000000e+00
  0.00000000e+00  0.00000000e+00 -3.12500000e-02]
[ 1.21637896e+00 -1.47670870e+01  5.12857056e+01 -4.34057617e+01
 -1.24613281e+02  3.44265625e+02 -3.32843750e+02  1.07750000e+02
  5.71875000e+01 -6.15000000e+01  1.80000000e+01 -1.00000000e+00
  0.00000000e+00 -0.00000000e+00  3.12500000e-02]
```

The np.linald.colve will return an exact value while by calculating A-1b will return float point numbers and those numbers have bigger error with the exact value as the size grow up. This is because when calculating the inverse of A, as the entries in A is too small, there will be error when saving these values. When size getting bigger, the values in Hilbert Matrix become smaller and lead to bigger error.

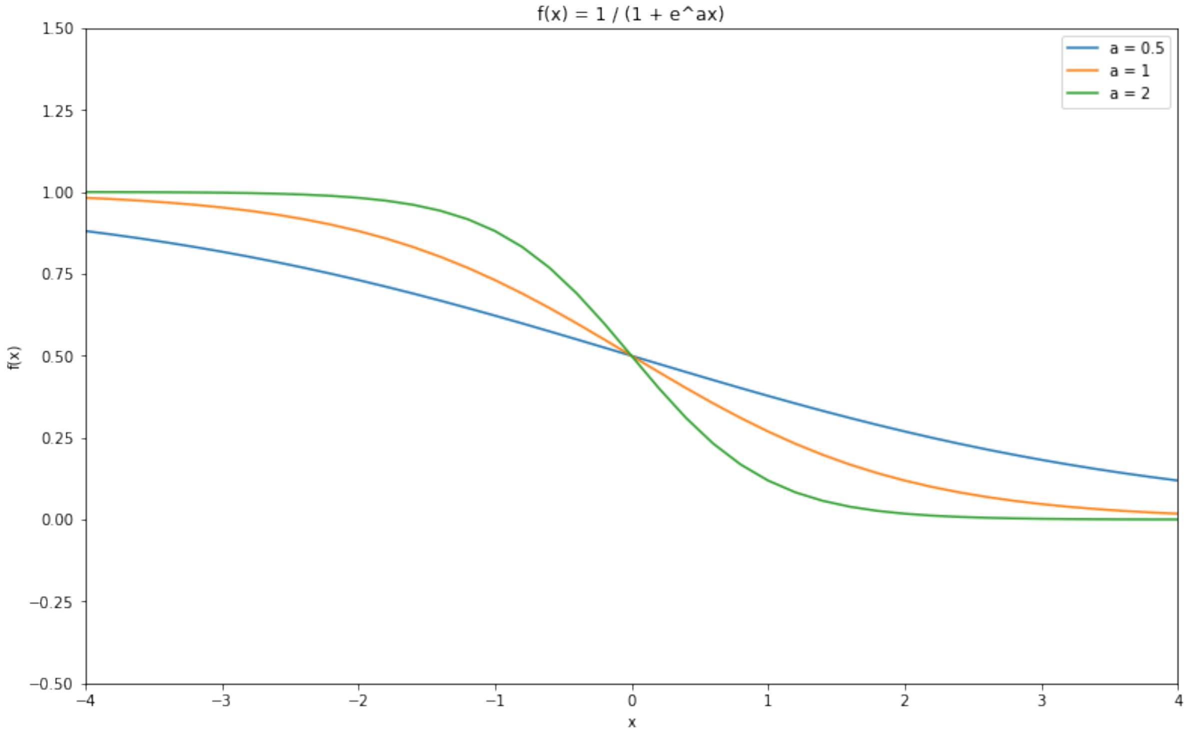
Lab Book 10 & 11

In [11]:
```
x = np.array([0.5, 0.7, 0.9, 1.3, 1.7, 1.8])
y = np.array([0.1, 0.2, 0.75, 1.5, 2.1, 2.4])
plt.figure(figsize = (13,8))
plt.clf()
plt.plot(x, y, color='g', linestyle='--', linewidth='1.5', mark

plt.title('y against x')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.xlim(0,3)
plt.ylim(0,3)

plt.show()
```



Lab Book 12

In [12]:
```
plt.figure(figsize = (13,8))
plt.clf()
an = np.array([0.5, 1, 2])
x = np.linspace(-4, 4, 41)
for a in an:
    fx = 1 / (1 + np.exp(a * x))
    plt.plot(x, fx)
plt.legend(['a = 0.5', 'a = 1', 'a = 2'])
plt.title('f(x) = 1 / (1 + e^ax)')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.xlim(-4, 4)
plt.ylim(-0.5, 1.5)
plt.show()
```



In [ ]:
```

```