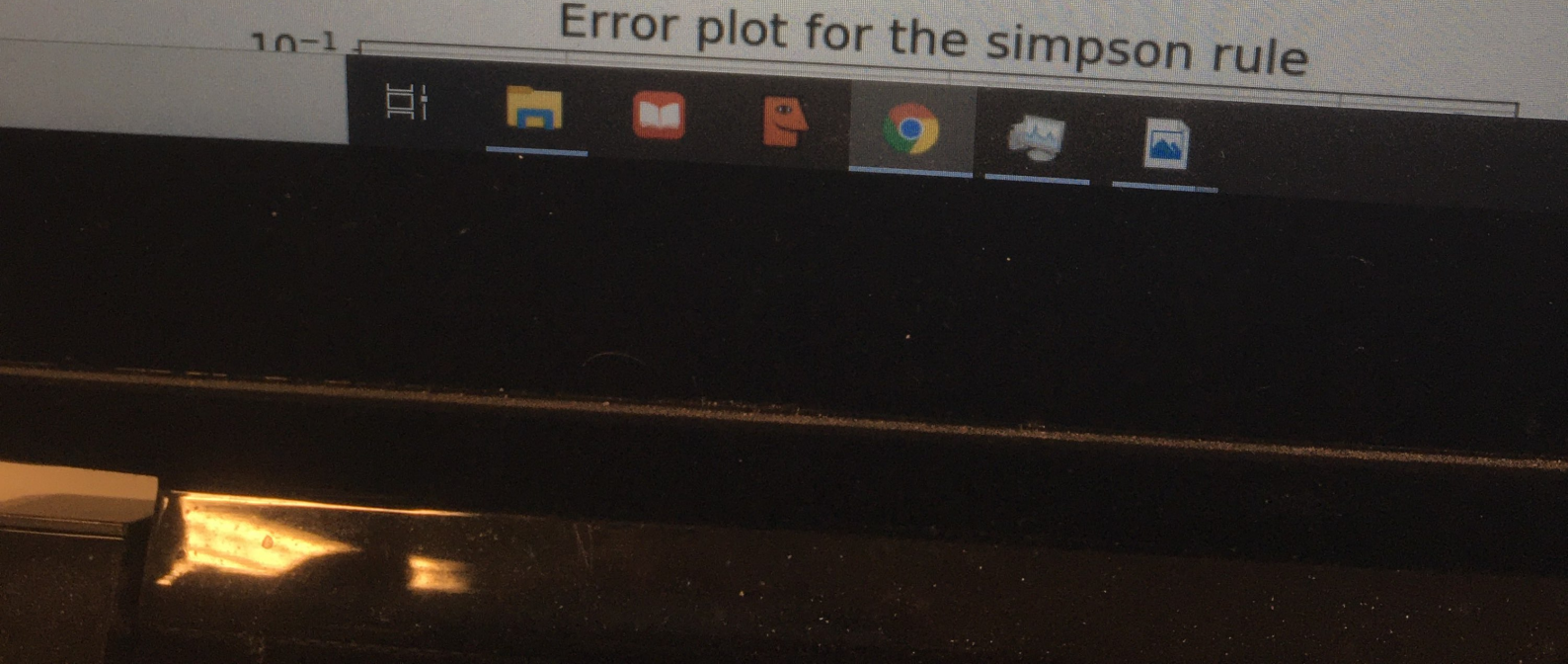
```
clf()
loglog(steps, errors, 'o-')
xlabel('h')
ylabel('e(h)')
title('Error plot for the simpson rule')
grid('True')
```

executed in 2.06s, finished 09:46:23 2019-11-05

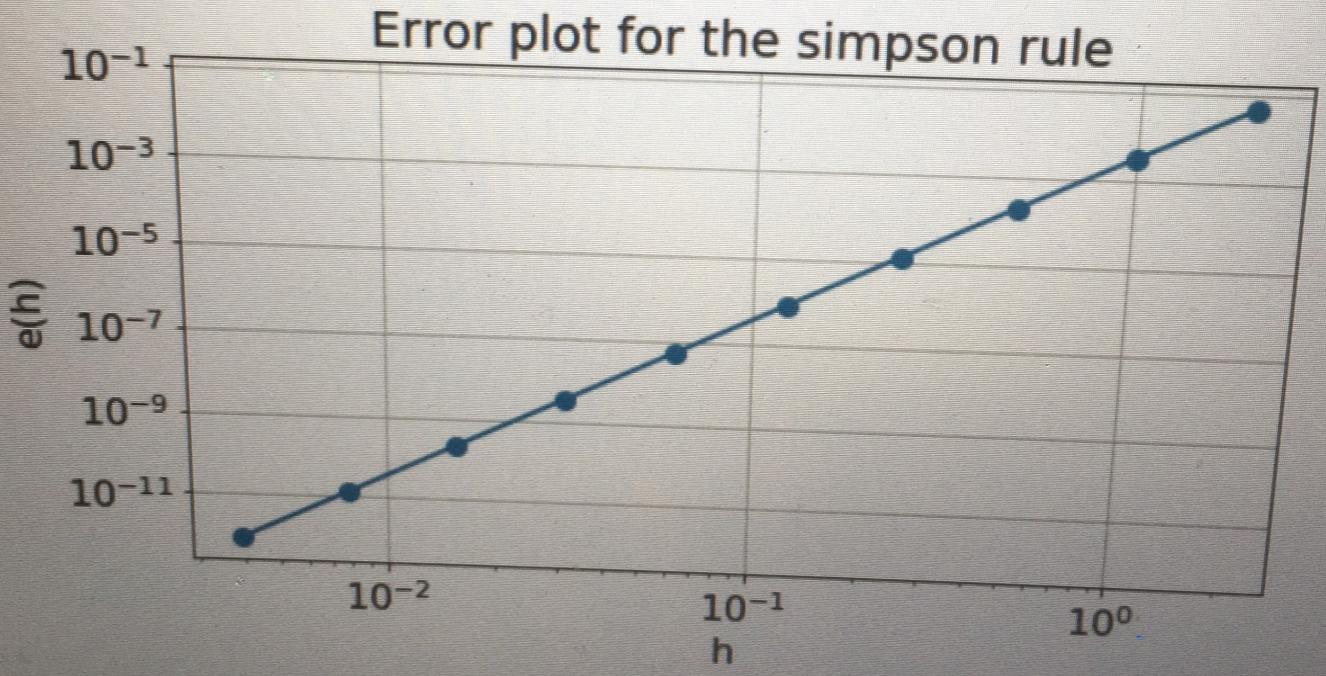
```
1] a)
h = 2.00e+00,    S(h) = 0.78346746,    e(h) = 4.77e-02
h = 1.00e+00,    S(h) = 0.73913060,    e(h) = 3.37e-03
h = 5.00e-01,    S(h) = 0.73597650,    e(h) = 2.18e-04
h = 2.50e-01,    S(h) = 0.73577259,    e(h) = 1.37e-05
h = 1.25e-01,    S(h) = 0.73575974,    e(h) = 8.59e-07
h = 6.25e-02,    S(h) = 0.73575894,    e(h) = 5.37e-08
h = 3.12e-02,    S(h) = 0.73575889,    e(h) = 3.36e-09
h = 1.56e-02,    S(h) = 0.73575888,    e(h) = 2.10e-10
h = 7.81e-03,    S(h) = 0.73575888,    e(h) = 1.31e-11
h = 3.91e-03,    S(h) = 0.73575888,    e(h) = 8.19e-13

 The order p and the error constant C
 h = 1.00e+00,  p = 3.95,  C = 0.0034
 h = 5.00e-01,  p = 3.99,  C = 0.0035
 h = 2.50e-01,  p = 4.00,  C = 0.0035
 h = 1.25e-01,  p = 4.00,  C = 0.0035
 h = 6.25e-02,  p = 4.00,  C = 0.0035
 h = 3.12e-02,  p = 4.00,  C = 0.0035
 h = 1.56e-02,  p = 4.00,  C = 0.0035
 h = 7.81e-03,  p = 4.00,  C = 0.0035
```



```
h = 1.25e-01,   S(h) = 0.73575974,   e(h) = 8.59e-07
h = 6.25e-02,   S(h) = 0.73575894,   e(h) = 5.37e-08
h = 3.12e-02,   S(h) = 0.73575889,   e(h) = 3.36e-09
h = 1.56e-02,   S(h) = 0.73575888,   e(h) = 2.10e-10
h = 7.81e-03,   S(h) = 0.73575888,   e(h) = 1.31e-11
h = 3.91e-03,   S(h) = 0.73575888,   e(h) = 8.19e-13
```

```
The order p and the error constant C
h = 1.00e+00,   p = 3.95,   C = 0.0034
h = 5.00e-01,   p = 3.99,   C = 0.0035
h = 2.50e-01,   p = 4.00,   C = 0.0035
h = 1.25e-01,   p = 4.00,   C = 0.0035
h = 6.25e-02,   p = 4.00,   C = 0.0035
h = 3.12e-02,   p = 4.00,   C = 0.0035
h = 1.56e-02,   p = 4.00,   C = 0.0035
h = 7.81e-03,   p = 4.00,   C = 0.0035
```



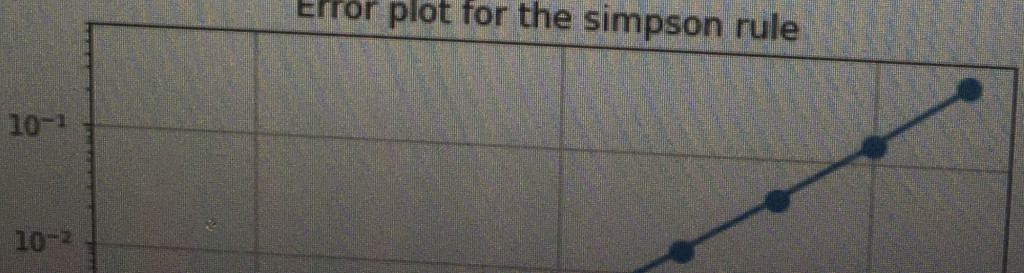

```
[8]:  print("1] b)")
```

```
title('Error plot for the simpson rule')
grid('True')
```

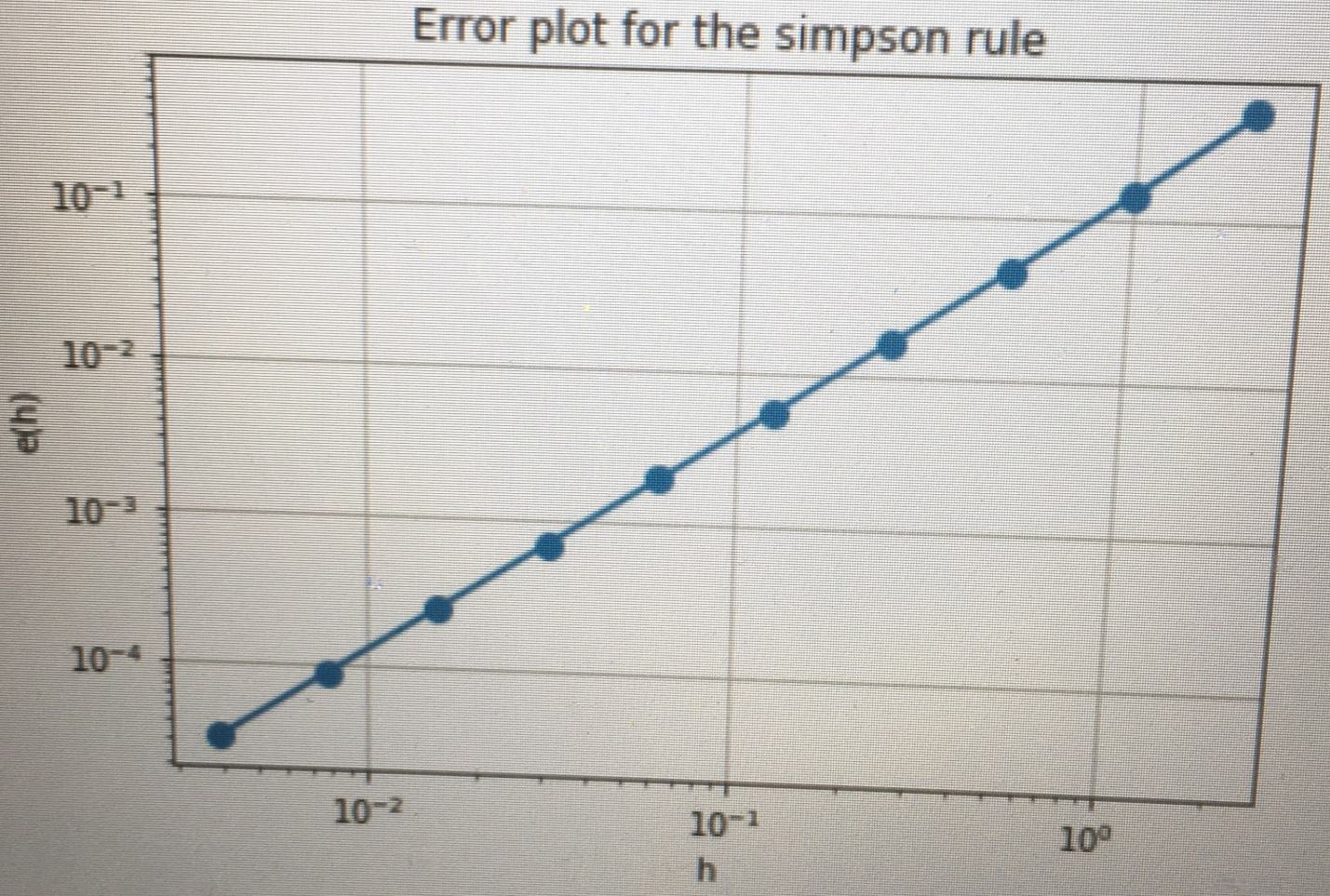
executed in 2.11s, finished 09:36:06 2019-11-05

```
1] b)
h = 2.00e+00,   S(h) = 1.33333333,   e(h) = 4.42e-01
h = 1.00e+00,   S(h) = 1.63540364,   e(h) = 1.40e-01
h = 5.00e-01,   S(h) = 1.72886020,   e(h) = 4.66e-02
h = 2.50e-01,   S(h) = 1.75945841,   e(h) = 1.60e-02
h = 1.25e-01,   S(h) = 1.76989907,   e(h) = 5.60e-03
h = 6.25e-02,   S(h) = 1.77353123,   e(h) = 1.97e-03
h = 3.12e-02,   S(h) = 1.77480571,   e(h) = 6.94e-04
h = 1.56e-02,   S(h) = 1.77525467,   e(h) = 2.45e-04
h = 7.81e-03,   S(h) = 1.77541312,   e(h) = 8.66e-05
h = 3.91e-03,   S(h) = 1.77546909,   e(h) = 3.06e-05

The order p and the error constant C
h = 1.00e+00,  p = 1.59,  C = 0.1401
h = 5.00e-01,  p = 1.54,  C = 0.1356
h = 2.50e-01,  p = 1.52,  C = 0.1316
h = 1.25e-01,  p = 1.51,  C = 0.1290
h = 6.25e-02,  p = 1.50,  C = 0.1274
h = 3.12e-02,  p = 1.50,  C = 0.1265
h = 1.56e-02,  p = 1.50,  C = 0.1260
h = 7.81e-03,  p = 1.50,  C = 0.1257
```



```
n = 5.00e-01,  p = 1.54,  C = 0.1356
n = 2.50e-01,  p = 1.52,  C = 0.1316
n = 1.25e-01,  p = 1.51,  C = 0.1290
h = 6.25e-02,  p = 1.50,  C = 0.1274
h = 3.12e-02,  p = 1.50,  C = 0.1265
h = 1.56e-02,  p = 1.50,  C = 0.1260
h = 7.81e-03,  p = 1.50,  C = 0.1257
```




```
:
def newton(f, df, x0, tol=1.e-8, max_iter=30):
    # Solve f(x)=0 by Newtons method
    # The output of each iteration is printed
    # Input:
    #   f, df:   The function f and its derivate f'.
    #   x0:  Initial values
    #   tol: The tolerance
    # Output:
    #   The root and the number of iterations
    x = x0
```

WORK SHEET 9 ANF

3] a) $f(x) = e^x + x^2 - x - 4 = 0$, we shall prove that f has only one zero "r" in the interval $[1,2]$

$x=1$: $e^1 + 1 - 1 - 4 = e - 4 < 0$, $e^1 + 2 - 1 > 0$ $\Rightarrow$ only one zero

$x=2$: $e^2 + 4 - 2 - 4 = e^2 - 2 > 0$, $e^2 + 4 - 2 > 0 > f'(1)$

An approximate r from the plot is $1.3 = x_0$

$f'(x) = e^x + 2x - 1$

newtons method is defined as:

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

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1.3 - \frac{0.051297}{5.269297} \approx 1.288747$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \approx 1.288678$$

$x_3 \approx 1.288678$

$x_4 \approx$ —"—

c, we know from a) that $r \approx 1.28868$

$g_i(x) = \ln(4 + x - x^2)$ $g_i'(x) = \frac{-2x+1}{4+x-x^2}$

$|g_i'(r)| = \left|\frac{-2r+1}{4+r-r^2}\right| \approx 0.43478 < 0$

the iterations will converge if x_0 is close enough to r

$g_{ii}(x) = \sqrt{-e^x + x + 4}$, $g_{ii}'(x) = \frac{-e^x + 1}{2\sqrt{-e^x + x + 4}}$

$|g_{ii}'(r)| \approx 1.01965$, $|g_{ii}'(r)| > 1$ and the iterations will not converge

Home | Assignment 9

tack.it.ntnu.no/user/289a3be5-4f3f-4806-8b8f-06d797998702/notebooks/Assignment%209.ipynb

Jobb | Sjakk | Basket-ting | NTNU H19 | Google Kalender | Blackboard | NTNU(annet) | Keyboa

Last Checkpoint: forrige onsdag kl. 08:39 (unsaved changes)

Cell Kernel Navigate Widgets Help

Run | Code | nbdiff

```
        fx = f(x)
        if abs(fx) < tol:              # Accept the solution
            break
        x = x - fx/df(x)               # Newton-iteration
        print('k ={:3d}, x = {:18.15f}, f(x) = {:10.3e}'.format(k+1, x, f(x)))
    return x, k+1
```

executed in 15ms, finished 15:53:09 2019-11-01

In [4]:

```
# 3] a)

def der(x):                            # The derivative f'
    return exp(x) + 2*x - 1

x0 = 1.3                               # Starting value
x3, nit = newton(t, der, x0, tol=1.e-14, max_iter=30)  # Apply Newton
print('\n\nResult:\nx={}, number of iterations={}'.format(x3, nit))
```

executed in 7ms, finished 15:53:11 2019-11-01

```
k =  0, x =  1.300000000000000, f(x) =  5.930e-02
k =  1, x =  1.288746758560088, f(x) =  3.581e-04
k =  2, x =  1.288677969382023, f(x) =  1.332e-08
k =  3, x =  1.288677966823868, f(x) = -4.441e-16


Result:
x=1.2886779668238684, number of iterations=4
```

In [5]:

```
def fixpoint(g, x0, tol=1.e-8, max_iter=30):
    # Solve x=g(x) by fixed point iterations
    # The output of each iteration is printed
    # Input:
    #   g:   The function g(x)
    #   x0:  Initial values
    #   tol: The tolerance
    # Output:
    #   The root and the number of iterations
```

Run | Code | nbdiff

In [7]:

```
# 3] b)

def i(x):
    return log(4 + x + x**2)

def ii(x):
    return sqrt(-exp(x) + x + 4)

def iii(x):
    return exp(x) + x**2 - 4

print('log(4 + x + x**2): (kun denne konvergerer)')
x4, nit = fixpoint(i, x0=1.5, tol=1.e-6, max_iter=10)
print('\nResultat:\nx = {}, antall iterasjoner={}\n\nsqrt(-exp(x) + x + 4):'.form

x5, nit = fixpoint(ii, x0=1.5, tol=1.e-6, max_iter=10)
print('\n\nResultat:\nx = {}, antall iterasjoner={}\n\nexp(x) + x**2 - 4:'.format(

x6, nit = fixpoint(iii, x0=1.5, tol=1.e-6, max_iter=10)
print('\n\nResultat:\nx = {}, antall iterasjoner={}'.format(x6, nit))
```

executed in 104ms, finished 15:55:30 2019-11-01

```
log(4 + x + x**2): (kun denne konvergerer)
k =  0, x =   1.5000000000
k =  1, x =   2.0476928434
k =  2, x =   2.3263737678
k =  3, x =   2.4628645542
k =  4, x =   2.5280113463
k =  5, x =   2.5586876951
k =  6, x =   2.5730373624
k =  7, x =   2.5797287280
k =  8, x =   2.5828443623
k =  9, x =   2.5842940641
k = 10, x =   2.5849683923
```

```
Resultat:
x = 2.58496839225596, antall iterasjoner=10

sqrt(-exp(x) + x + 4):
k =  0, x =   1.5000000000
k =  1, x =   1.0091139329
k =  2, x =   1.5053054929
k =  3, x =   0.9998878264
k =  4, x =   1.5105995169
k =  5, x =   0.9905322092
k =  6, x =   1.5158710552
k =  7, x =   0.9810634053
k =  8, x =   1.5211088914
k =  9, x =   0.9714992449
k = 10, x =   1.5263017049


Resultat:
x = 1.5263017049440268, antall iterasjoner=10

exp(x) + x**2 - 4:
k =  0, x =   1.5000000000
k =  1, x =   2.7316890703
k =  2, x =  18.8209324059
k =  3, x = 149220368.2729534507
k =  4, x =            inf
k =  5, x =            inf
k =  6, x =            inf
k =  7, x =            inf
k =  8, x =            inf
k =  9, x =            inf
k = 10, x =            inf


Resultat:
x = inf, antall iterasjoner=10
```

WORK SHEET 9 ANF

3] a) $f(x) = e^x + x^2 - x - 4 = 0$, we shall prove that f has only one zero "r" in the interval $[1,2]$

$x=1$: $e^1 + 1 - 1 - 4 = e - 4 < 0$, $e^1 + 2 - 1 > 0$ $\Rightarrow$ only one zero

$x=2$: $e^2 + 4 - 2 - 4 = e^2 - 2 > 0$, $e^2 + 4 - 2 > 0 > f'(1)$

An approximate r from the plot is $1.3 = x_0$

$f'(x) = e^x + 2x - 1$

newtons method is defined as:

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

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1.3 - \frac{0.051297}{5.269297} \approx 1.288747$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \approx 1.288678$$

$x_3 \approx 1.288678$

$x_4 \approx$ —"—

c, we know from a) that $r \approx 1.28868$

$g_i(x) = \ln(4 + x - x^2)$ $g_i'(x) = \frac{-2x+1}{4+x-x^2}$

$|g_i'(r)| = \left|\frac{-2r+1}{4+r-r^2}\right| \approx 0.43478 < 0$

the iterations will converge if x_0 is close enough to r

$g_{ii}(x) = \sqrt{-e^x + x + 4}$, $g_{ii}'(x) = \frac{-e^x + 1}{2\sqrt{-e^x + x + 4}}$

$|g_{ii}'(r)| \approx 1.01965$, $|g_{ii}'(r)| > 1$ and the iterations will not converge

WORK SHEET 9 ANF

3] c) cont. $r = 1.28868$

$g_{iii}(x) = e^x + x^2 - 4$, $g_{iii}'(x) = e^x + 2x$

$|g_{iii}'(r)| = |e^r + 2r| \simeq 6.20535$, $|g_{iii}'(r)| > 1$, no convergence

4] a) for $g(x)$ we have $g(r) = r$ where r is a fixed point

applying $g^{-1}(x)$ on both sides we get $g^{-1}(g(r)) = g^{-1}(r)$

$\Leftrightarrow r = g^{-1}(r) \Leftrightarrow g^{-1}(r) = r$, which we wanted to show

b) we want to show that if $|g'(r)| > 1$, then $|(g^{-1})'(r)| < 1$

by definition, we have that $(g^{-1})'(r) = \frac{1}{g'(r)}$, $|g'(r)| > 1$

α

which gives $|(g^{-1})'(r)| = \left|\frac{1}{g'(r)}\right| = \frac{1}{\alpha}$ where $\alpha > 1$

giving $\frac{1}{\alpha} < 1$ which we wanted to prove

$||$

$|(g^{-1})'(r)| < 1$

WORK SHEET 9 ANF

5] a) $g(x) = x^2 + y^2 = 4$, $h(x) = xy = 1$ $\Rightarrow f(x) = \begin{bmatrix} x^2+y^2-4 \\ xy-1 \end{bmatrix} = 0$ $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$

$$J(x) = \begin{bmatrix} \frac{\partial g}{\partial x}(x,y) & \frac{\partial g}{\partial y}(x,y) \\ \frac{\partial h}{\partial x}(x,y) & \frac{\partial h}{\partial y}(x,y) \end{bmatrix} = \begin{bmatrix} 2x & 2y \\ y & x \end{bmatrix}$$

we have the general solution: $\begin{bmatrix} 2x_k & 2y_k \\ y_k & x_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix} = -f(x_k) \Leftrightarrow J(x_k)\Delta_k = -f(x_k)$

$k=0 \Rightarrow \begin{matrix} x_0 = 2 \\ y_0 = 0 \end{matrix} \Rightarrow \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta y_0 \end{bmatrix} = -\begin{bmatrix} 0 \\ -1 \end{bmatrix} \Rightarrow \begin{bmatrix} \Delta x_0 \\ \Delta y_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1/2 \end{bmatrix}$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} \Delta x_0 \\ \Delta y_0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1/2 \end{bmatrix}$$ ↙ jupyter. correct

$k=1 \Rightarrow \begin{matrix} x_1 = 2 \\ y_1 = 1/2 \end{matrix} \Rightarrow \begin{bmatrix} 4 & 1 \\ 1/2 & 2 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta y_1 \end{bmatrix} = \begin{bmatrix} 1/4 \\ 0 \end{bmatrix}$

using gaussian:

$$\begin{matrix} R1 \\ R2 \end{matrix} \left[\begin{array}{cc|c} 4 & 1 & 1/4 \\ 1/2 & 2 & 0 \end{array}\right] \overset{R2}{\underset{R1}{\sim}} \left[\begin{array}{cc|c} 1/2 & 2 & 0 \\ 4 & 1 & 1/4 \end{array}\right]$$

$$R1 \sim R1 - 8 \cdot R2 \left[\begin{array}{cc|c} 1/2 & 2 & 0 \\ 0 & -15 & 1/4 \end{array}\right] \quad R1 \sim R1 \div (-15) \left[\begin{array}{cc|c} 1/2 & 2 & 0 \\ 0 & 1 & -1/60 \end{array}\right]$$

$$R2 \sim 2 \cdot R2 \left[\begin{array}{cc|c} 1 & 4 & 0 \\ 0 & 1 & -1/60 \end{array}\right] \quad R2 \sim R2 - 4 \cdot R1 \left[\begin{array}{cc|c} 1 & 0 & 1/15 \\ 0 & 1 & -1/60 \end{array}\right]$$

WORK SHEET 9 ANF

5] a) cont.

$$\left[\begin{array}{cc|c} 1 & 0 & 1/15 \\ 0 & 1 & -1/60 \end{array}\right] \Rightarrow \begin{bmatrix} \Delta x_1 \\ \Delta y_1 \end{bmatrix} = \begin{bmatrix} 1/15 \\ -1/60 \end{bmatrix}$$

jupyter, correct

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} \Delta x_1 \\ \Delta y_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1/2 \end{bmatrix} + \begin{bmatrix} 1/15 \\ -1/60 \end{bmatrix} = \begin{bmatrix} 31/15 \\ 29/60 \end{bmatrix} \approx \begin{bmatrix} 2.066667 \\ 0.483333 \end{bmatrix}$$

after two iterations

```
    return y

# The Jacobian
def jac(x):
    J = array([[2*x[0], 2*x[1]],
               [x[1],    x[0]]])
    return J

x0 = array([2.0, 0.0])          # Starting values
max_iter = 20

print("5] b):")
x, nit = newton_system(f, jac, x0, tol = 1.e-12, max_iter =

if nit == max_iter:
    printf('Warning: Convergence har not been achieved')
```

executed in 72ms, finished 17:26:45 2019-11-01

```
5] b):
k =  0, x =  [ 2.  0.]
k =  1, x =  [ 2.   0.5]
k =  2, x =  [ 2.06666667  0.48333333]
k =  3, x =  [ 2.05693613  0.48614666]
k =  4, x =  [ 2.05831082  0.48583507]
k =  5, x =  [ 2.05815443  0.48587219]
k =  6, x =  [ 2.05817298  0.48586781]
k =  7, x =  [ 2.0581708   0.48586833]
k =  8, x =  [ 2.05817105  0.48586827]
k =  9, x =  [ 2.05817102  0.48586827]
k = 10, x =  [ 2.05817103  0.48586827]
k = 11, x =  [ 2.05817103  0.48586827]
k = 12, x =  [ 2.05817103  0.48586827]
k = 13, x =  [ 2.05817103  0.48586827]
k = 14, x =  [ 2.05817103  0.48586827]
```

n [17]: # 5] c)

```
x, nit = newton_system(f, jac, x0, tol = 1.e-12, max

if nit == max_iter:
    printf('Warning: Convergence har not been achieve

print("The method still converges")
```

executed in 63ms, finished 17:34:41 2019-11-01

```
5] c):
k =  0, x =  [ 2.  0.]
k =  1, x =  [ 1.5  0.5]
k =  2, x =  [ 1.4375  0.6875]
k =  3, x =  [ 1.61565564  0.6104473 ]
k =  4, x =  [ 1.52610006  0.65278079]
k =  5, x =  [ 1.56375336  0.63915904]
k =  6, x =  [ 1.5494657   0.64532689]
k =  7, x =  [ 1.55552858  0.64285867]
k =  8, x =  [ 1.55304215  0.64389587]
k =  9, x =  [ 1.55407623  0.64346879]
k = 10, x =  [ 1.55364862  0.64364613]
k = 11, x =  [ 1.55382587  0.64357275]
k = 12, x =  [ 1.55375247  0.64360316]
k = 13, x =  [ 1.55378288  0.64359056]
k = 14, x =  [ 1.55377029  0.64359578]
k = 15, x =  [ 1.5537755   0.64359362]
k = 16, x =  [ 1.55377334  0.64359452]
k = 17, x =  [ 1.55377424  0.64359414]
k = 18, x =  [ 1.55377387  0.6435943 ]
k = 19, x =  [ 1.55377402  0.64359423]
k = 20, x =  [ 1.55377396  0.64359426]
The method still converges
```

[ ]:

[ ]: