Santiago Passos – Alejandro Rojas

Verification of methods for Numerical Analysis

Incremental Search:

```
def incremental_search_verif():
    tol = 1e-7
    N = 100
    print(incremental_search(f_x, 0.5, -3, N, 1))

In [6]: incremental_search_verif()
(-2.5, -2.0)
```

Bisection:

```
def bisection_verif():
    tol = 1e-7
    N = 100
    bisection(f_x, 0, 1, N, tol)

Iteration     |    x     |    f(x)     |    Error
    0      0.5000000      -0.2931087      -
    1      0.5000000      -0.2931087      0.1747123
    2      0.7500000      -0.1183964      0.0815787
    3      0.8750000      -0.0368177      0.0374516
    4      0.9375000      0.0006339      0.0184062
    5      0.9062500      -0.0177723      0.0092857
    6      0.9218750      -0.0084866      0.0045812
    7      0.9296875      -0.0039054      0.0022749
    8      0.9335938      -0.0016304      0.0011335
    9      0.9355469      -0.0004969      0.0005658
    10      0.9365234      0.0000688      0.0002828
    11      0.9360352      -0.0002140      0.0001414
    12      0.9362793      -0.0000726      0.0000707
    13      0.9364014      -0.0000019      0.0000353
    14      0.9364624      0.0000335      0.0000177
    15      0.9364319      0.0000158      0.0000088
    16      0.9364166      0.0000070      0.0000044
    17      0.9364090      0.0000026      0.0000022
    18      0.9364052      0.0000003      0.0000011
    19      0.9364033      -0.0000008      0.0000006
    20      0.9364042      -0.0000002      0.0000003
    21      0.9364047      0.0000001      0.0000001
0.9364045858383179
```

```
def regula_falsi_verif():
    tol = 1e-7
    N = 100
    regula_falsi(f_x, 0, 1, N, tol)
```

```
Iteration     |     x     |     f(x)     |     Error
    0       0.9339404       -0.0014291       -
    1       0.9339404       -0.0014291       0.0014878
    2       0.9365061       0.0000588       0.0000587
0.9364045811008 69
```

Newton Raphson:

```
def newton_verif():
    tol = 1e-7
    N = 100
    x_0 = 0.5
    print(newton_raphson(f_x, f_x_d, x_0, N, tol))
```

```
Iteration     |     x     |     f(x)     |     Error
    0       0.5000000       0.9283920       -
    1       0.9283920       0.9363667       0.0079748
    2       0.9363667       0.9364046       0.0000378
    3       0.9364046       0.9364046       0.0000000
4
0.9364045808795621
```

Fixed Point:

```
def fixed_point_verif():
    x_0 = -0.5
    tol = 1e-7
    N = 100
    print(fixed_point(g_x, x_0, N, tol))
```

```
Iteration     |     x     |     f(x)     |     Error
    0       -0.5000000       -0.2931087       -
    1       -0.2931087       -0.4198215       0.1267128
    2       -0.4198215       -0.3463045       0.0735170
    3       -0.3463045       -0.3909585       0.0446539
    4       -0.3909585       -0.3644050       0.0265534
    5       -0.3644050       -0.3804263       0.0160213
    6       -0.3804263       -0.3708368       0.0095895
    7       -0.3708368       -0.3766056       0.0057689
    8       -0.3766056       -0.3731454       0.0034602
    9       -0.3731454       -0.3752246       0.0020792
    10       -0.3752246       -0.3739766       0.0012481
    11       -0.3739766       -0.3747262       0.0007496
    12       -0.3747262       -0.3742761       0.0004501
    13       -0.3742761       -0.3745464       0.0002703
    14       -0.3745464       -0.3743841       0.0001623
    15       -0.3743841       -0.3744816       0.0000975
    16       -0.3744816       -0.3744231       0.0000585
    17       -0.3744231       -0.3744582       0.0000351
    18       -0.3744582       -0.3744371       0.0000211
    19       -0.3744371       -0.3744498       0.0000127
    20       -0.3744498       -0.3744422       0.0000076
    21       -0.3744422       -0.3744467       0.0000046
    22       -0.3744467       -0.3744440       0.0000027
    23       -0.3744440       -0.3744456       0.0000016
    24       -0.3744456       -0.3744447       0.0000010
    25       -0.3744447       -0.3744452       0.0000006
    26       -0.3744452       -0.3744449       0.0000004
    27       -0.3744449       -0.3744451       0.0000002
    28       -0.3744451       -0.3744450       0.0000001
    29       -0.3744450       -0.3744451       0.0000001
-0.37444505296105535
```

a

Secant:

```
def secant_verif():
    tol = 1e-7
    N = 100
    print(secant(f_x, 0.5, 1, N, tol))
```

```
In [68]: secant_verif()
Iteration |     x     |     f(x)     |     Error
    0       0.5000000      -0.2931087       -
    1       1.0000000      -0.2931087       -
    2       0.9461662      0.0056194      0.0538338
    3       0.9359966      -0.0002363      0.0101696
    4       0.9364070      0.0000014      0.0004104
    5       0.9364046      0.0000000      0.0000024
    6       0.9364046      -0.0000000      0.0000000
0.9364045808795615
```

Multiple Roots:

```
def multiple_roots_verif():
    tol = 1e-7
    N = 100
    print(multiple_roots(h_x, h_x_d, h_x_d2, 1, N, tol))
```

```
In [56]: multiple_roots_verif()
Iteration |     x     |     f(x)     |     Error
    0       1.0000000      0.7182818         -
    1      -0.2342106      0.0254058      1.2342106
    2      -0.0084583      0.0000357      0.2257523
    3      -0.0000119      0.0000000      0.0084464
    4      -0.0000000      0.0000000      0.0000119
    5      -0.0000000      0.0000000      0.0000000
-4.218590698935789e-11
```

Matrix:

```
In [9]: print(A)
[[ 2.  -1.   0.   3. ]
 [ 1.   0.5  3.   8. ]
 [ 0.  13.  -2.  11. ]
 [14.   5.  -2.   3. ]]

In [11]: print(b)
[[1.]
 [1.]
 [1.]
 [1.]]
```

Gaussian Simple:

```
In [33]: sol = gauss_simple(A,b)
    ...: print(sol)
[[ 0.03849518810148722]
 [-0.18022747156605434]
 [-0.30971128608923887]
 [ 0.24759405074365706]]
```

Gauss Partial:

```
In [34]: sol = gauss_partial(A,b)
    ...: print(sol)
[[ 0.03849518810148722]
 [-0.18022747156605426]
 [-0.3097112860892389 ]
 [ 0.24759405074365706]]
```

Gauss Total:

```
In [37]: sol = gauss_total(A,b)
    ...: print(sol)
[[ 0.03849518810148732]
 [-0.18022747156605423]
 [-0.3097112860892388 ]
 [ 0.24759405074365703]]
```