SORBONNE UNIVERSITÉ

NUMERICAL ALGORITHMS

Practical 5

Méline TROCHON
Anatole VERCELLONI

Teacher : Stef Graillat

Janvier 2023

Table des matières

1 Exercise 3 **2**

2 Exercise 4 **2**

3 Exercise 5 **2**

1 Exercise 3

```
function x_min = newton(f, x0, tol, it_max)
    val = x0;
    fprim = diff(f);
    for it = 1:it_max
        prec = val;
        val = val - eval(f(val)/fprim(val));
        disp(it)
        abs(prec-val)
        if abs(prec - val) < tol
            break
        end
    end
    x_min = val;
end
```

We defined the function $f(x) = x^3 - cos(x)$ and with $x_0 = 1$ we found that the root was 0.8, the speed of this algorithm is quadratic.

2 Exercise 4

```
function res = newton_padic(f, x0, p, k)
    df = diff(f);
    x = x0;
    for it = 1:k
        [~, f_inv, ~] = gcd(eval(df(x)), p);
        x = mod(x - eval(f(x)) * mod(f_inv,p), p^(it+1))
    end
    res = x;
end
```

We testes our algorithm for $x^2 - 2$ and get for k = 1,.., 4 and get respectivly 1à, 108, 2166, 4567

3 Exercise 5

1. Let consider the function $f(X) = A - X^{-1}$, $A, X \in \mathbb{R}^{n\times n}$
First of all, we want to show that f is differentiable and compute it.
Let pick $H \in \mathbb{R}^{n\times n}$, we have :

$$\begin{aligned}
f(X+H) &= A - (X+H)^{-1} \\
&= A - [(X+H)^{-1}XX^{-1}] \\
&= A - [(XX^{-1}(X+H)^{-1}] \\
&= A - [(X(I_n + X^{-1}H)^{-1})] \\
&= A - [(I_n + X^{-1}H)^{-1}X^{-1}] \\
&= A - [X^{-1} - X^{-1}HX^{-1} + X^{-2}H^2X^{-1} + o(\|H^2\|)](1) \\
&= f(X) + X^{-1}HX^{-1} + o(\|H\|)
\end{aligned}$$

(1)Taylor expension
We find an expression, $Df(X)(H) = X^{-1}HX^{-1}$ if :
(i) $L(H) : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is linear in H
. $\quad H \mapsto X^{-1}HX^{-1}$
(ii) $X^{-2}H^2X^{-1} + o(\|H^2\|) = o(\|H\|)$

We will assume (i) and (ii)

In this case, we have for the Newton's method :

$$
\begin{aligned}
X_{n+1} &= X_n - Df(X_n)^{-1} f(X_n) \\
&= X_n - (X_n f(X_n) X_n) \\
&= X_n - X_n A X_n + X_n \\
&= 2X_n - X_n A X_n
\end{aligned}
$$

which is the exepted formula

```
function res = newton_it(A, tol, it_max)
    X = A'./trace(A' * A);
    for it=1:it_max
        X = 2 .* X - X * A * X
        if norm(eye - X * A, 2) < tol
            break
        end
    end
    res = X;
    return
end
```