Dichotomie

On souhaite résoudre l'équation $f(x) = 0$ dans l'intervalle $[a, b]$ où $f(x)$ est une fonction monotone sur l'intervalle $[a, b]$ et telle que $f(a) \times f(b) < 0$.

`f(x)` étant définie dans une autre fonction, l'algorithme suivant renvoie l'intervalle de largeur `eps` contenant la solution.

```
def dichotomie(a, b, eps):
    while b-a > eps:
        m = (a+b)/2
        if f(a)*f(m) <= 0:
            b = m
        else:
            a = m
    return a,b
```

Euler

On souhaite résoudre l'équation différentielle :

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,t) \quad \text{avec } y(t_0) = y_0$$

`f(y,t)` étant définie dans une autre fonction (souvent, elle ne dépend pas explicitement de `t`), pour un pas de temps `dt` et un nombre de points à calculer `n`, l'algorithme suivant renvoie deux listes : les instants t_k successifs dans la liste `t` et les valeurs approchées de $y(t_k)$ dans la liste `y`.

```
def euler(f, t0, y0, dt, n):
    y = [y0]
    t = [t0]
    for k in range(n):
        t.append(t[-1] + dt)
        y.append(y[-1] + dt*f(y[-1], t[-1]))
    return t, y
```

Monte-Carlo

On souhaite déterminer l'incertitude par propagation d'une grandeur z définie par une fonction de deux variables x et y dont on a estimé (par une méthode de type B) la demi-étendue de l'intervalle dans lequel leur valeur se trouve de façon quasi-certaine :

$$z = f(x,y) \quad \text{où } x \in [x_m - \Delta_x, x_m + \Delta_x] \text{ et } y \in [y_m - \Delta_y, y_m + \Delta_y]$$

`f(x,y)` étant définie dans une autre fonction, l'algorithme suivant renvoie une liste `z` de `n` valeurs simulées. On peut ensuite en extraire la valeur moyenne et l'écart-type qui est l'incertitude-type sur z.

```
def montecarlo(x_m, Delta_x, y_m, Delta_y, n):
    z = []
    for k in range(n):
        x_k = x_m + dx * np.random.uniform(-1,1)
        y_k = y_m + dy * np.random.uniform(-1,1)
        z.append(f(x_k, y_k))
    return z
```