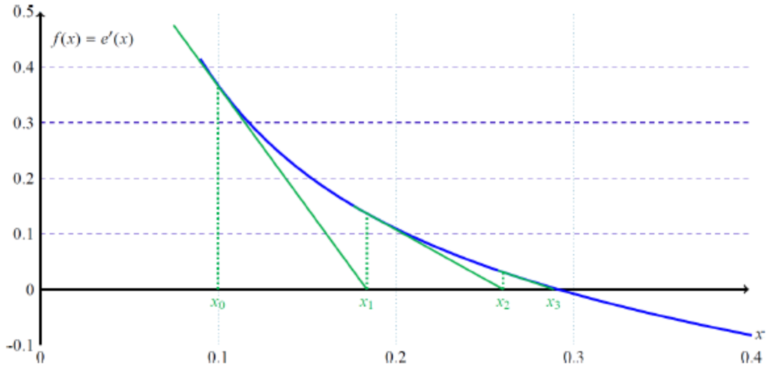
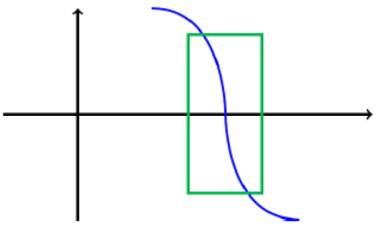
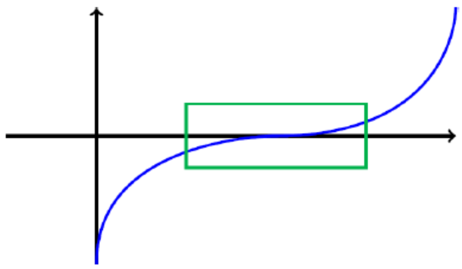
- <u>Dichotomie - Newton</u> –

PROBLÈMES STATIONNAIRES À UNE DIMENSION DU TYPE $f(x) = 0$





Fort gradient



Faible gradient

Résolution de problèmes stationnaires à 1 dimension du type f(x) = 0 *(Newton et Dichotomie)*

1. Introduction

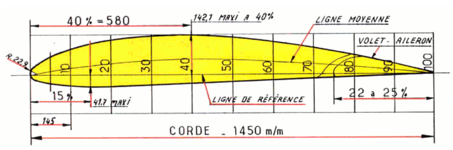
1.1. Objectif

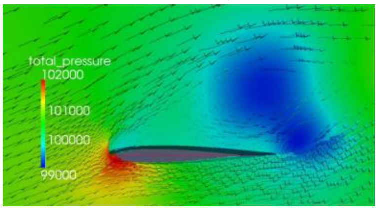
L'objectif de ce cours est de traiter des problèmes stationnaires (constant au cours du temps), linéaires ou non, conduisant à la **résolution approchée d'une équation algébrique du type f(x)=0**.

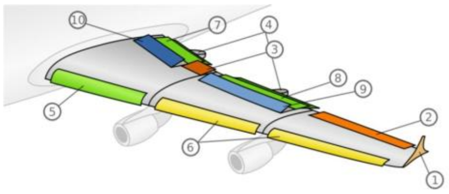
1.2. Support de l'étude

Comme support de cours, nous prenons un profil d'aile d'avion dont la demi-épaisseur $e(x)$ est approchée par l'équation :

$$e(x) = 0,15 \times \left(3,7 \times \sqrt{x} - 3,4 \times x - 0,3 \times x^4\right), \forall x \in [0, 1]$$

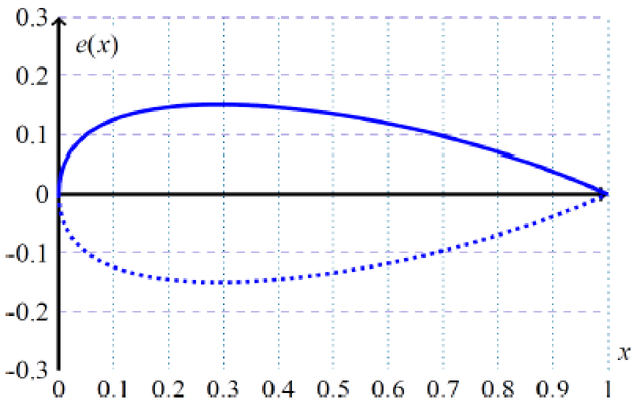


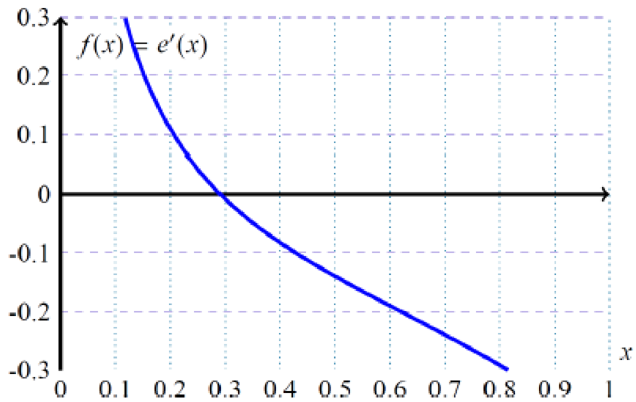




Rechercher la valeur de x pour laquelle le profil est le plus épais, revient à résoudre l'équation :

$$f(x) = 0,15 \times \left(\frac{3,7}{2.\sqrt{x}} - 3,4 - 1,2 \times x^3\right) = 0$$

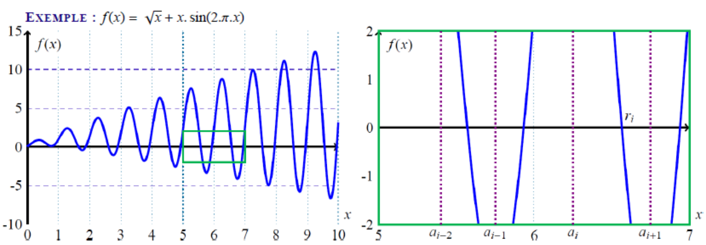




1.3. Notion de convergence

La plupart des méthodes de recherche de racines d'une fonction, se comportent bien **si une seule racine *r*** (valeur pour laquelle la fonction s'annule) **est présente dans l'intervalle d'étude.**
Les algorithmes que l'on va découvrir plus loin ne marche que pour le cas de racine unique.

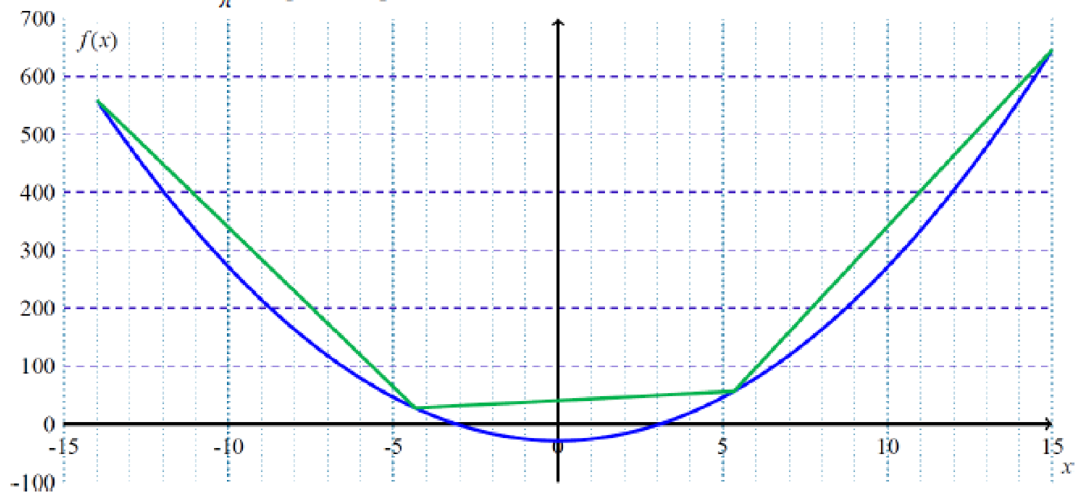
Si racine multiple, il faut mettre en place une méthode numérique (pas au programme) permettant de localiser l'intervalle ou se trouve chacune d'elle en calcul préalable.



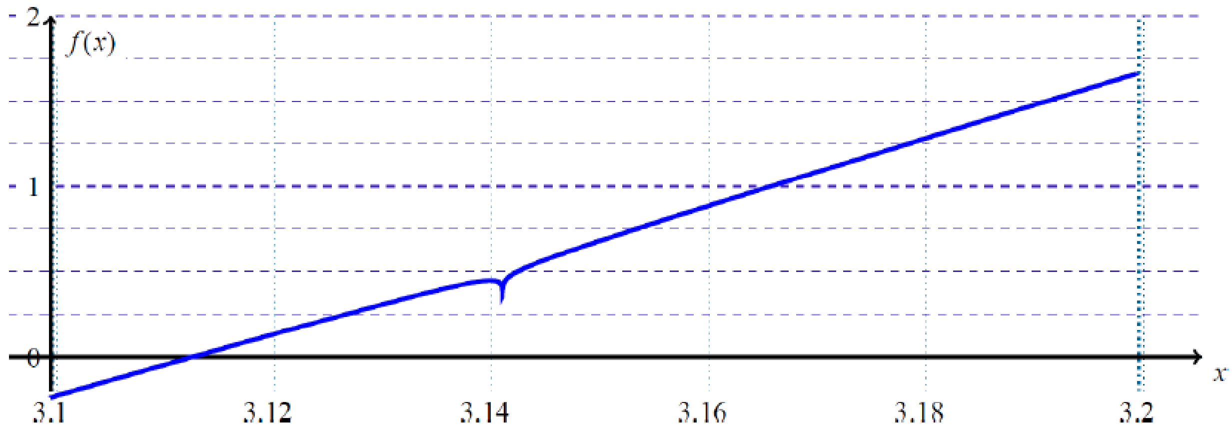
Résolution de problèmes stationnaires à 1 dimension du type f(x) = 0 *(Newton et Dichotomie)*

Les représentations graphiques sont utiles parfois pour avoir **une idée des lieux de convergence**. Cela permet d'obtenir à vu d'oeil un premier intervalle ou un premier candidat pour démarrer une méthode numérique. Cependant, il convient aussi de se méfier des représentations graphiques. Pour cela, il est préférable d'observer attentivement l'expression de la fonction et son domaine de définition.

EXEMPLE : $f(x) = 3.x^3 + \frac{1}{\pi^4}.\ln\left[(\pi - x)^2\right] - 29$



Courbes tracées entre -14 et 15. La courbe verte n'ayant que 4 points, on ne constate pas d'intersection avec l'axe des abscisses.



Et avec le zoom sur l'intervalle [3, 1; 3, 2], il semble encore qu'il n'y ait qu'une racine sur l'intervalle. Or $\lim_{x\to\pi} f(x) = -\infty$.

Par continuité sur les intervalle [3, 14; π[et [π; 3, 15], la fonction f admet **donc 3 racines dans l'intervalle [3, 1; 3, 2].**

Problème : en maths, le théorème de la valeur intermédiaire donne un zéro mais on ne sait pas lequel …

Bilan :

Les méthodes numériques **ne permettent pas d'obtenir une réponse formelle à un problème**. En revanche, elles permettent **d'approcher la solution** d'un problème avec **une précision** fonction du type de stockage des nombres *(cf premier semestre)* et du calcul numérique nécessaire pour obtenir cette solution *(cf la sensibilité de certains calculs aux erreurs d'arrondis)*.

Pour une **erreur normée en abscisse =précision de calcul** doit être définie et se répercutera en coût (temps exécution sur l'algorithme). Une tolérance de 10^{-8} est raisonnable.
En revanche, chercher à obtenir 10^{-16} est a priori non nécessaire et impossible.

Résolution de problèmes stationnaires à 1 dimension du type f(x) = 0 *(Newton et Dichotomie)*

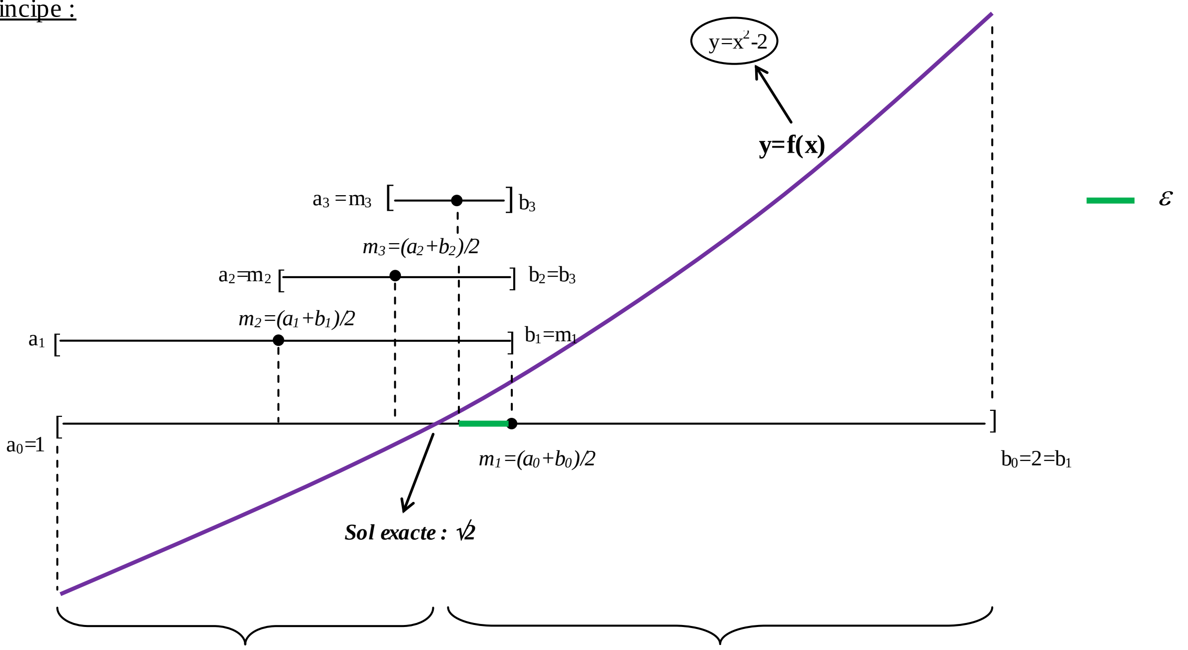
Il y a 3 types d'équations que l'on souhaite savoir résoudre en **simulation numérique** :

- **F(x)=0 : 2 algorithmes au programme (Dichotomie et Newton)**
- *Résolution numérique d'équations différentielles (1^er et 2^nd ordre)*
- *Résolution numérique de systèmes linéaires*

2. Algorithme de Dichotomie

But : créer une **suite** qui converge vers le zéro.
Hypothèse : on suppose que la fonction est continue et qu'elle est croissante ou décroissante et change de signe.
si la fonction s'annule plusieurs fois *(cf convergence)*, on demande quel zéro nous intéresse

Principe :



Algorithme :

Entrées : *f, a, b, ε* # f : fonction dont on cherche le zéro sur [a,b]
a, b : bornes de l'intervalle de recherche
ε **: précision = test d'arrêt** (ou nb d'itérations) = écart entre dernier b-a appelé

Tant que $|b-a| > \varepsilon$ faire

$m \leftarrow (a+b)/2$
si f(a).f(b) < 0 alors
$b \leftarrow m$
sinon
$a \leftarrow m$

Sortie : m

Valeur approchée de f(x)=0 sur [a,b] à ε près

Compteur évite boucle infinie

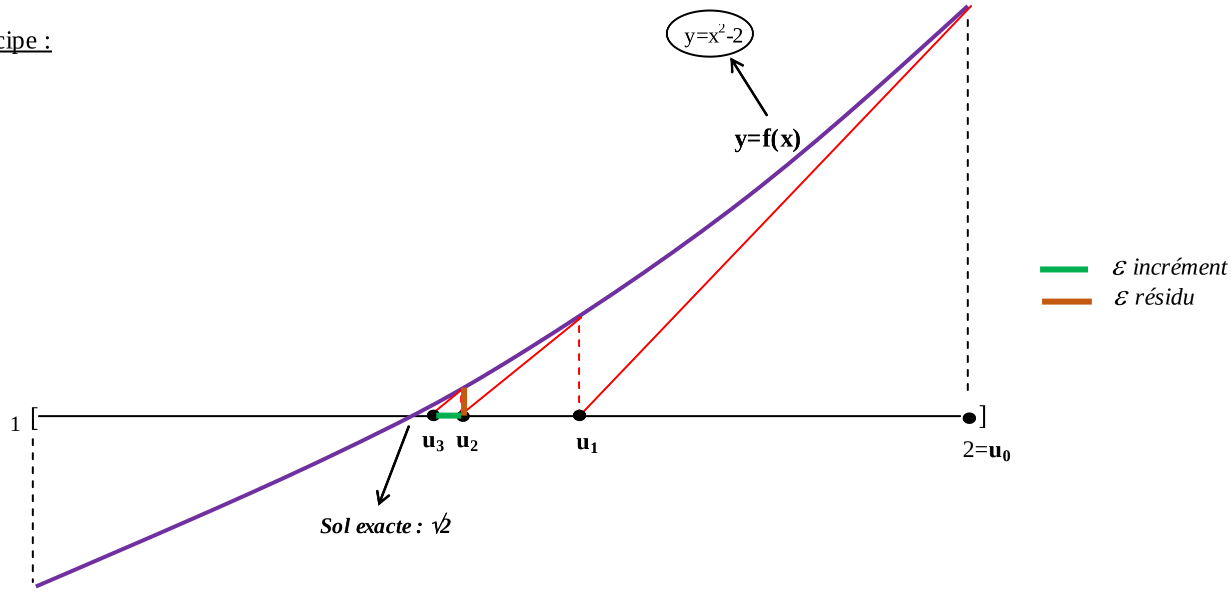
```
def dichotomie (f,a,b,eps):
    c=0
    while abs(b-a)> eps and c<10000:
        m=(a+b)/2
        c=c+1
        if f(a)*f(m)<0:
            b=m
        else:
            a=m
    return [c,m]
```

Résolution de problèmes stationnaires à 1 dimension du type f(x) = 0 *(Newton et Dichotomie)*

3. Algorithme de Newton (Raphson)

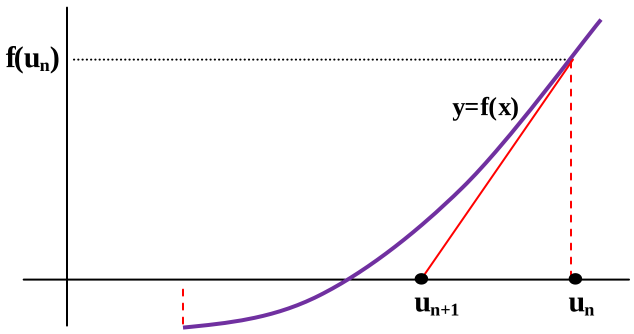
Idée : on remplace la courbe par **sa tangente.**
Problème : mais quel point de départ prendre ? **u0** → étude mathématique aussi…

Principe :



Req : cela **semble converger très vite**

Il faut exprimer u_{n+1} en fonction de u_n



Il faut donc trouver une relation de récurrence entre u_{n+1} et u_n :

Equation de la tangente à Cf en $x=u_n$: $\mathbf{y = f'(u_n)(x-u_n) + f(u_n)}$

On a u_{n+1} est donné (vérifie l'équation) par : $f'(u_n)(u_{n+1}-u_n) + f(u_n) = 0$

(u_n) est une suite récurrente définie par $u_{n+1}=g(u_n)$ avec $g(x) = x-f(x)/f'(x)$	donc :	$\mathbf{u_{n+1} = u_n - f(u_n)/f'(u_n)}$ *avec $f'(un) \neq 0$*

Formule de base de l'algo de Newton

*attention : **il faut savoir calculer f'** !! (en TP, on étudiera des fonctions simples)*

Résolution de problèmes stationnaires à 1 dimension du type f(x) = 0 *(Newton et Dichotomie)*

3.1. Algorithme avec **contrôle du résidu**:

Les itérations s'arrêtent dès que $|\mathbf{f(u_n)}| < \varepsilon$

Entrées : *f, df,* u_0, ε

\# f : fonction dont on cherche le zéro sur [a,b]
\# df: dérivée de f
\# u_0: C.I *(donnée par étude math. préalable)*
\# **ε : précision = test d'arrêt** sur $f(u_n)$

$u \leftarrow u_0$

Tant que $|f(u)| > \varepsilon$ faire

$u \leftarrow u - f(u)/df(u)$

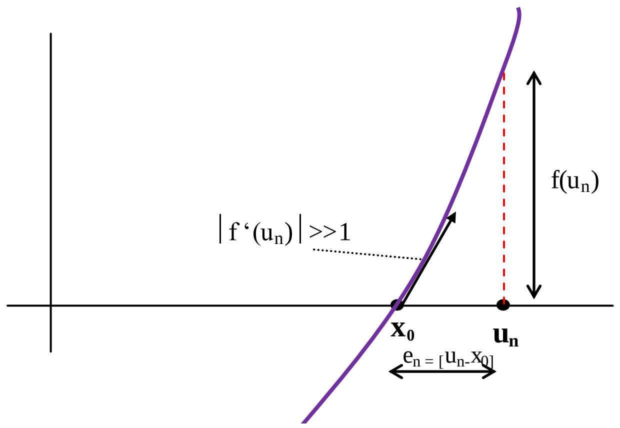
Sortie : u

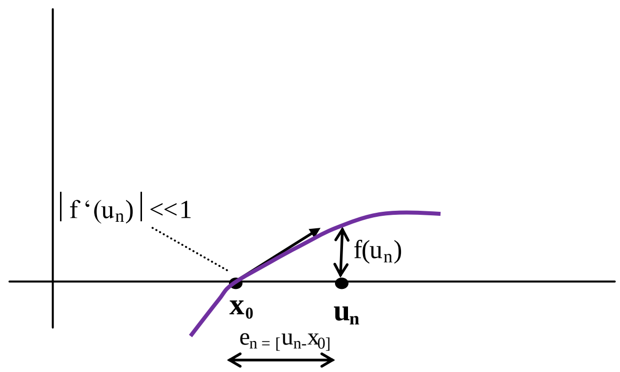
Attention si df(u)=0

Attention flottants et arrondis

```python
def newton_residu(f,df,u0,eps):
    u = u0
    c = 0
    while abs(f(u)) > eps:
        u = u - f(u)/df(u)
        c=c+1
    return [c,u]
```

Mais il y a 2 situations où le contrôle du résidu est un **mauvais critère d'arrêt :**





e_n est petit mais $|f(u_n)|$ est très grand. Pente forte en f(x)=0

→ *risque de faire trop itérations*

e_n grand mais pourtant $|f(u_n)|$ petit.

→ *risque d'être trop loin de la solution*

3.2. <u>Algorithme avec **contrôle de l'incrément**</u>

Les itérations s'arrêtent dès que $|\mathbf{u_{n+1}}\text{-}\mathbf{u_n})| < \varepsilon$

<u>Entrées</u> : f, df, u_0, ε

u $\leftarrow$ u_0
up $\leftarrow$ u -2ε # juste pour rentrer ds boucle (aurait pu être autre valeur)

<u>Tant que</u> $|$u-$u_p|$ > ε faire

  u $\leftarrow$ u_p

  u $\leftarrow$ u – f(u)/df(u)

<u>Sortie</u> : u

```
def newton_increment(f,df,u0,eps):

    u = u0
    up = u - 2*eps
    c = 0
    while abs(u-up) > eps:
        up = u
        u = u - f(u)/df(u)
        c += 1
    return [c,u]
```

Remarque : $/u_{n+1}\text{-}u_n) / \rightarrow 0$ est une condition nécessaire mais pas suffisante pour $/u_n - x_0) / \rightarrow 0$