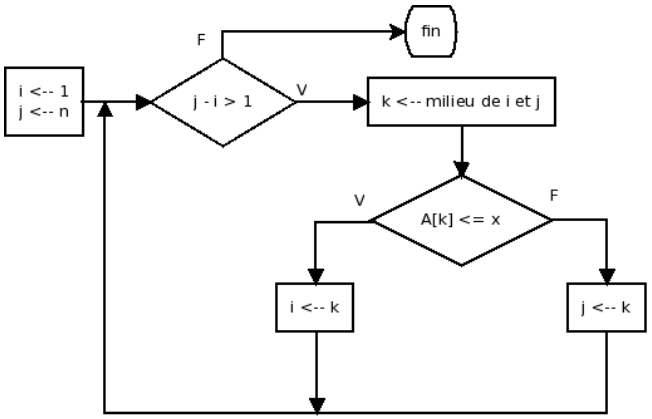

FIG. 1 – Recherche par dichotomie dans un tableau

1. Recherche par dichotomie dans un tableau.

On se donne un tableau linéaire de nom `A` indexé de 1 à `n` dont les valeurs sont strictement croissantes. Soit `x` un nombre supérieur ou égal à la plus petite valeur de `A` et strictement plus petit que sa plus grande valeur. On veut former une procédure de nom `dich_t` prenant `x` comme paramètre et renvoyant un entier i entre 0 et `n-1` tel que $A[i] \leq x < A[i+1]$
Le diagramme conventionnel de l'algorithme résolvant ce problème est présenté en figure 1. Une implémentation possible en syntaxe Maple est

```
dich_t:=proc(x)
  global A,n; #on peut se passer de cette déclaration
  local i,j,k;
  i:=1; j:=n;
  while j-i>1 do
    k := floor((i+j)/2);
    if A[k]<= x then
      i :=k;
    else
      j:=k;
    fi;
  od;
  return i;
end proc;
```

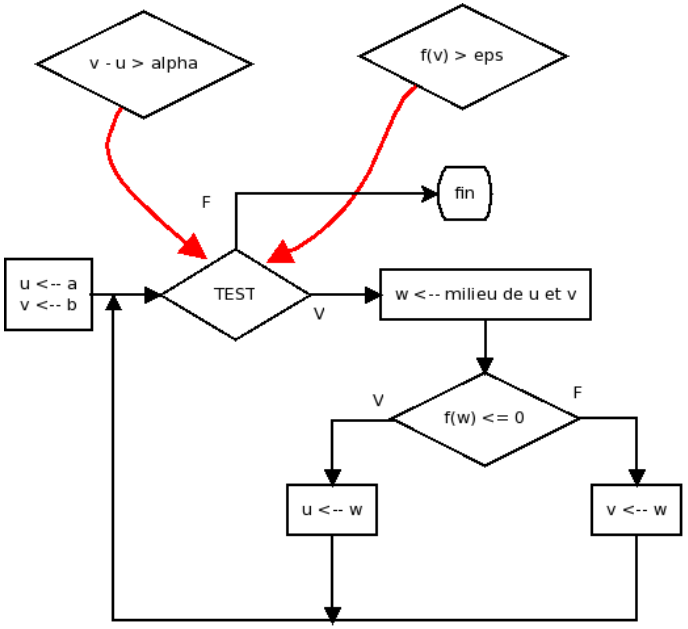

FIG. 2 – Résolution dichotomique d'une équation

Le code suivant fabrique des tableaux croissants aléatoires et appelle la procédure pour vérification

```
n:=10; A:=array(1..n,[]):
accMax:=2;
A[1]:=rand(1..accMax)():
for i from 2 to n do
  A[i] := A[i-1]+rand(1..accMax)();
od:
print(A);
x:=rand(A[1]..A[n])();
dich_t(x);
```

2. Résolution numérique d'une équation par dichotomie.

La problématique est la même mais dans un contexte *numérique* avec des objets du type `float`. Ils ne constituent qu'une partie de l'ensemble des nombres

décimaux. Tout calcul de ce type comporte donc des erreurs d'arrondi et aucun test d'égalité n'est réellement sigificatif dans un tel contexte.
Du point de vue mathématique, on se donne deux réels a et b et une fonction f continue et strictement croissante dans $[a, b]$ telle que $f(a) < 0$ et $f(b) > 0$. D'après le théorème des valeurs intermédiaires, il existe un réel c tel que $f(c) = 0$, ce c est unique à cause de la stricte croissance.
On ne peut évidemment espérer trouver une expression exacte de c mais il est facile d'obtenir des valeurs approchées. Le diagramme présenté en figure 2 est très semblable au précédent. Il est à noter que plusieurs tests sont possibles pour arréter la boucle. Dans le diagramme `alpha` et `eps` sont des noms de petits nombre représentant la précision désirée.
Il faut adapter le nombre de décimales utilisées par le calculateur à la précision souhaitée. Ce nombre est désigné par `Digits`. Il est modifiable à volonté. Un test avec `alpha` égal à 10^{-10} et `Digits` égal à 10 (par défaut) peut conduire à une erreur d'exécution. Les erreurs d'arrondi peuvent empêcher la convergence mathématique de se réaliser. Le code suivant implémente une procédure et l'appelle pour chercher x tel que $cos(x) = x$.

```
 res_dich:=proc()
  #global f,a,b;
  local u,v,w;
  u:=a;v:=b;
  while v-u > eps do
    w := (u+v)/2;
    if f(w)<= 0 then
      u := w;
    else
      v := w;
    fi;
  od;
  return u;
end proc;
f := x -> x - cos(x);
a:= 0.;
b:= evalf(Pi/2);
eps := 10^(-12);
Digits:=13:
res_dich();
```

Une feuille de calcul est disponible.