DEUX EXEMPLES DE CONSTRUCTION D'UN ALGORITHME POUR EXPLORER DES SUITES NUMÉRIQUES ET CONJECTURER CERTAINES DE LEURS PROPRIÉTÉS

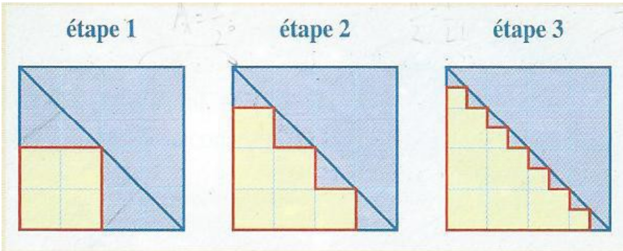
Hubert Raymondaud, LEGTA Louis Giraud à Carpentras-Serres

PREMIER EXEMPLE : ÇA FAIT PENSER À UNE FRACTALE MAIS ÇA N'EN EST PAS UNE

I.A – PRÉSENTATION

Cet exemple est tiré d'un exercice du manuel Terracher 1re S 2001 "Séquence 1", page 204 (fac-similé de l'énoncé original en dernière page ce document)

Dans un carré de côté 4 carreaux (l'unité est le côté d'un carreau), on considère une construction en n étapes, selon la description suivante :



On construit des carrés s'inscrivant dans les triangles isocèles formés en traçant la diagonale du carré.

À chaque étape, les nouveaux points sur la diagonale sont les milieux des segments découpés sur cette diagonales à l'étape précédente. Ces segments constituent les hypoténuses des nouveaux triangles ainsi formés.

On note Sn l'aire du polygone colorée en jaune, à l'étape n, et Pn le périmètre de ce polygone.

On aurait tendance à penser fractale : une surface de mesure finie pour un périmètre de mesure infinie ?

Un algorithme nous aiderait à conjecturer l'évolution de ces deux grandeurs. Mais comme il ne coule pas de source, il faut faire un petit travail de prospection pour déterminer une stratégie d'élaboration de l'algorithme. Il va de soit que l'algorithme présenté n'a nullement la prétention d'être le “meilleur”. Et d'ailleurs selon quels critères juge-t-on qu'un algorithme est meilleur qu'un autre ? Vaste question !

Je vais procéder en trois phases :

- Dans une première phase je vais déterminer l'algorithme et les commandes permettant de tracer le triangle de base puis les carrés des étapes successives. Avec **R** les carrés sont tracés avec la fonction `polygon(X, Y)` où X représente les abscisses des quatre sommets des carrés et Y leurs ordonnées. Il faut donc, pour chaque étape trouver les valeurs de X et de Y, et conjecturer leur relation de récurrence.
- Dans une deuxième phase, je détermine l'algorithme et les commandes permettant de placer les milieux des segments successifs sur la diagonale du carré initial de coté 4 unités. Chacun de ces points constituera le sommet supérieur droit de chaque carré en contact avec la diagonale.
- Dans une troisième phase, on rassemble les deux algorithmes précédents pour tracer la série des carrés correspondant à chaque étape, quel que soit le nombre d'étapes. Tous les carrés correspondant à un étape sont de la même couleur. Il faut être attentif au fait qu'il ne faut pas recouvrir les carrés déjà placés, il faut donc bien calculer et placer ses pas … Les séries de r couleurs sont générées par la fonction `rainbow(r)`.

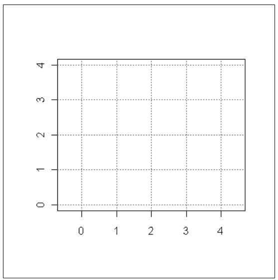
I.B – PREMIÈRE PHASE

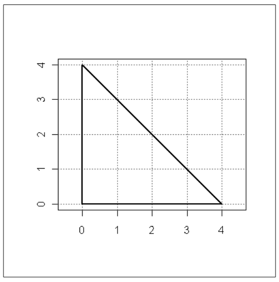
On commence par quelques lignes de commandes, pour faire les premières étapes, et voir comment on pourra généraliser. Personnellement il est rare que je trouve l'algorithme général du premier coup...

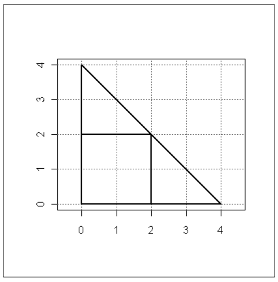
`# LIGNES DE COMMANDES`
`# Des carrés en noir et blanc`
`plot(0:4, 0:4, type = "n", asp = 1,`
`     xlab = "", ylab = "")`
`  grid(col = "grey40")`
	Phase 1-1 : On trace quelques point "invisibles" (type = "n") pour disposer d'une portion de plan avec repère orthonormé (asp = 1), quadrillée : grid(col = "grey40").

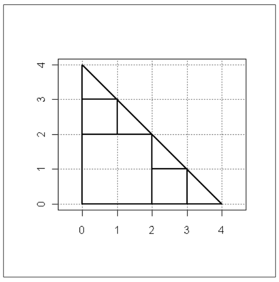
`  polygon(c(0, 0, 4), c(0, 4, 0), lwd = 2)`	Phase 1-2 : Ensuite on trace le triangle dans lequel s'inscriront les carrés.
`  polygon(c(0, 0, 2, 2), c(0, 2, 2, 0), lwd = 2)`	Phase 1-3 : Ensuite c'est le premier carré de l'étape 1
`  polygon(c(0, 0, 1, 1), c(2, 3, 3, 2), lwd = 2)` `  polygon(c(2, 2, 3, 3), c(0, 1, 1, 0), lwd = 2)`	Phase 1-4 : Puis les deux carrés de l'étape 2.

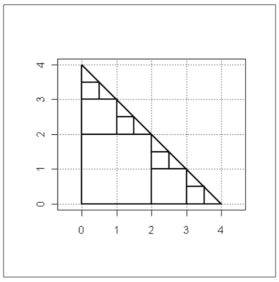
`  polygon(c(0, 0, .5, .5), c(3, 3.5, 3.5, 3), lwd = 2)`
`  polygon(c(1, 1, 1.5, 1.5), c(2, 2.5, 2.5, 2), lwd = 2)`
`  polygon(c(2, 2, 2.5, 2.5), c(1, 1.5, 1.5, 1), lwd = 2)`
`  polygon(c(3, 3, 3.5, 3.5), c(0, .5, .5, 0), lwd = 2)`
	Phase 1-5 : Puis les 4 carrés de l'étape 3.


Phase 1-1


Phase 1-2


Phase 1-3


Phase 1-4


Phase 1-5

Il s'agit de mettre à jour une stratégie dans le tracé des carrés qui permettra d'élaborer l'algorithme général.

On remarque qu'on peut construire tous les carrés à partir de leur coin haut droit. Il suffit donc de repérer tous ces coins et la longueur du coté correspondant pour tracer les carrés. C'est ce que l'on va faire à la phase suivante.

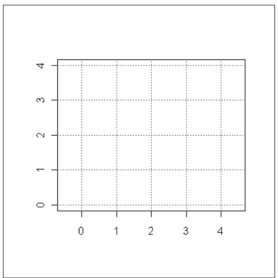
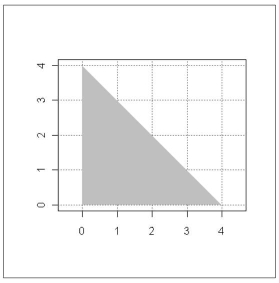
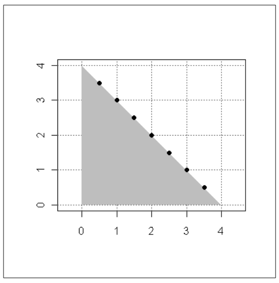
I.C – DEUXIÈME PHASE

On va placer les points sur la diagonale ce qui permettra d'organiser le placement des carrés lors des n étapes. En fonction du nombre d'étapes r, on calcule le nombre de segments de diagonales ($nbI = 2^r$) et le nombre de carrés à tracer (nbI – 1) donc le nombre de points sur la diagonale. Reste à générer les abscisses et les ordonnées de tous ces points. La longueur du côté d'un carré de l'étape i (pas) sera $4 / 2^i$ va nous y aider.

`# LIGNES DE COMMANDES`
`# Intégrer la construction des carrés dans des boucles`
`# La suite des milieux détermine la position des carrés`
`# Quelques essais préliminaires`
 	
`r <- 3 ; nbI <- 2^r ; pas <- 4 / nbI`
	Phase 2-1 : On commence par un essai avec 3 étapes, n = 3. Le nombre de segments de diagonales nbI est de 2^r. La longueur du coté des carrés associés à chaque segment est de pas = 4/nbI.

`plot(0:4, 0:4, type = "n", asp = 1,`
`     xlab = "", ylab = "")`
`grid(col = "grey40")`
	Phase 2-2 : On trace quelques point "invisibles" (type = "n") pour disposer d'une portion de plan avec repère orthonormé (asp = 1), quadrillée : grid(col = "grey40")

`polygon(c(0, 0, 4), c(0, 4, 0), border = NA, col = "grey")`	Phase 2-3 : Ensuite on trace le triangle en couleur, dans lequel s'inscriront les carrés.
`i <- 1:(nbI - 1)`	Phase 2-4 : On génère la suite du nombre de points sur la diagonale, correspondant à chaque étape
`xi <- 0 + pas * i` `yi <- 4 - pas * i`	Phase 2-5 : Calcul des abscisses et des ordonnées des coins supérieurs droits de chaque carré, qui se trouvent sur la diagonale..
`points(xi, yi, pch = 19)`	Phase 2-6 : Placement des 2^r points sur la diagonale.

 Phase 2-2	 Phase 2-3	[1] 1 2 3 4 5 6 7 Phase 2-4	[1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 [1] 3.5 3.0 2.5 2.0 1.5 1.0 0.5 Phase 2-5	 Phase 2-6

Il ne reste plus qu'à assembler ces deux phases pour obtenir le tracer de tous les carrés.

I.D – TROISIÈME PHASE

C'est la phase de création de l'algorithme qui va positionner les carrés correspondant à chaque étape et faire les calculs d'aire et de périmètre. Mi est le coin haut droit de chaque carré, Ai, Bi, Ci sont les autres coins, dans le sens trigonométrique.

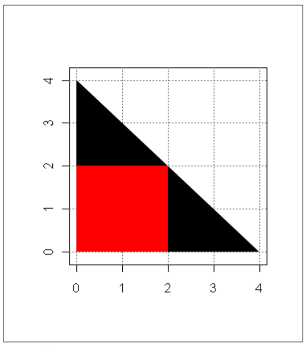
```
# Fonction R produisant calculs et graphiques pour un nombre
# d'étapes quelconque
tricar1 <- function(r = 4){
   plot(0:4, 0:4, type = "n", asp = 1,
        xlab = "", ylab = "")
      grid(col = "grey40")
      polygon(c(0, 0, 4), c(0, 4, 0), border = NA, col = "black")
         Sys.sleep(2)
      aires <- 0
      for (pr in 1:r) {
         pas <- 4 / 2^pr ; sub <- (1:(2^pr - 1))[seq(1, 2^pr - 1, 2)]
         for (i in sub) {
            if (r < 8) {
               xMi <- pas * i ; yMi <- 4 - pas * i
               xAi <- xMi - pas ; yAi <- yMi
               xBi <- xMi - pas ; yBi <- yMi - pas
               xCi <- xMi ; yCi <- yMi - pas
               polygon(c(xMi, xAi, xBi, xCi), c(yMi, yAi, yBi, yCi),
                       col = rainbow(r)[pr], border = NA)
                  Sys.sleep(.5)
            }
            aires <- aires + pas * pas
         }
      }
   if (r >= 8) {
      polygon(c(0, 0, 4), c(0, 4, 0), border = NA, col = "green")
   }
   perim <- (2^r - 2) * pas * 2 + 2 * pas + (4 - pas) * 2
   cat("Aire de tous les carrés :", aires, "\n\n")
   cat("Périmètre de la réunion des carrés :", perim, "\n\n")
}

tricar(r = 4)
Aire de tous les carrés : 7.5
Périmètre de la réunion des carrés : 15

tricar1(r = 15)
Aire de tous les carrés : 7.999756
Périmètre de la réunion des carrés : 15.99951
```

Tracé du repère, du quadrillage et du triangle de base.
Sys.sleep(2) permet de temporiser l'affichage des carrés successifs.

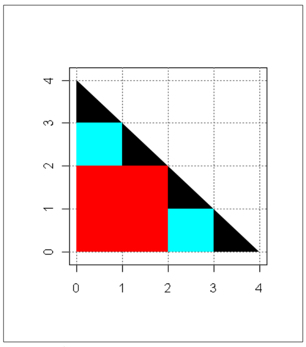
Boucle pour les r étapes
Calcul de la longueur du coté des carrés et du nombre de carrés, à chaque étape.
On ne fait le graphique que lorsque r est inférieur à 8.
Auquel cas on calcule les coordonnées des 4 coins de chaque carré que l'on trace en couleur. rainbow(r) génère automatiquement r couleurs.
Calcul et cumul de l'aire des carrés.

Lorsque n >= 8, on ne trace qu'un triangle vert.
Calcul du périmètre et affichage des résultats.

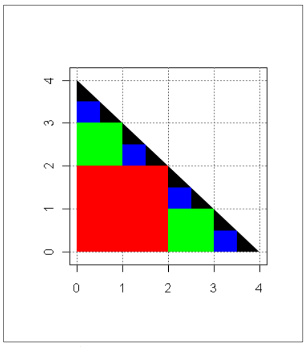
Il semble bien que ces suites tendent vers une limite …



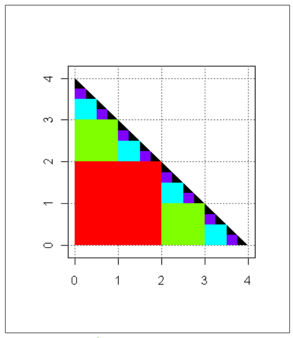
Une étape



Deux étapes



Trois étapes



Quatre étapes

C'est l'occasion de travailler sur les suites des coordonnées des quatre coins de chaque carré tracé, sur la suite des longueur de leurs côtés, en plus de la suite des périmètres et des aires totales étudiées dans l'exercice original.

L'aspect ludique du tracé de la figure est un élément motivant pour les élèves.

Maintenant que l'on a conjecturé l'existence de limites, tant pour l'aire (8 ?) que pour le périmètre (16 ?), il ne reste plus qu'à passer aux mathématiques, avec l'exercice figurant en dernière page.

DEUXIÈME EXEMPLE :
SUITES DE SYRACUSE, DU CHAOS DANS L'R

II.A – PRÉSENTATION

Je prolonge l'exemple développé par Stéphan Manganelli avec LARP, pour montrer comment on peut découvrir et illustrer graphiquement différents aspects des suites de Syracuse $\{u_0$ entier ; $u_{n+1} = u_n / 2$ si u_n est pair ; sinon $u_{n+1} = 3 \times u_n + 1\}$.

Je présente trois algorithmes illustrant :

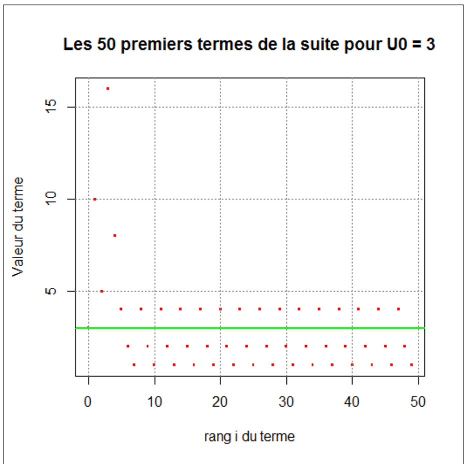
- La périodicité de la suite à partir d'un certain rang.
- La “longueur du vol”, c'est à dire le rang à partir duquel la suite devient périodique, et la “hauteur du vol”, c'est à dire la plus grande valeur atteinte lors de ce “vol”.
- La distribution des “longueurs du vol”, pour des valeurs initiales de u de 1 à N.

II.B – À LA DÉCOUVERTE DE PLUSIEURS SUITES

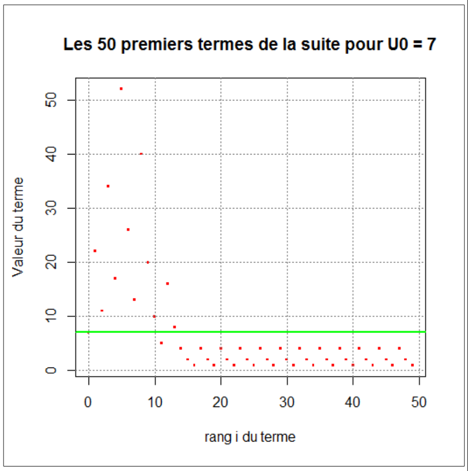
L'algorithme doit générer les N premiers termes de la suite en fonction d'une valeur de u_0 et en faire la représentation graphique.

```
# Cette fonction calcule, en fonction de U0, les N premiers
# termes de la suite et en fait la représentation graphique
syracN_B = function(U0 = 3, N = 50){
   if (U0 <= 0 | trunc(U0) != U0) {
      cat("U0 doit être entier naturel différent de 0") ; stop()
   }
   u <- U0 ; suiteN <- U0
   for (i in 1:(N - 1)) {
      if (u %% 2 == 0) {u <- u / 2} else {u <- 3 * u + 1}
      suiteN <- c(suiteN, u)
   }
# Affichage des résultats et des graphiques
   cat("Les N premiers termes de la suite sont :",
        suiteN, "\n")
   plot(0:(N - 1), suiteN,
         main = paste("Les", N,
            "premiers termes de la suite pour U0 =", U0),
         xlab = "rang i du terme", ylab = "Valeur du terme",
         pch = ".", cex = 4, col = "red")
      grid(col = "grey50")
      abline(h = U0, col = "green", lwd = 2)
}
```

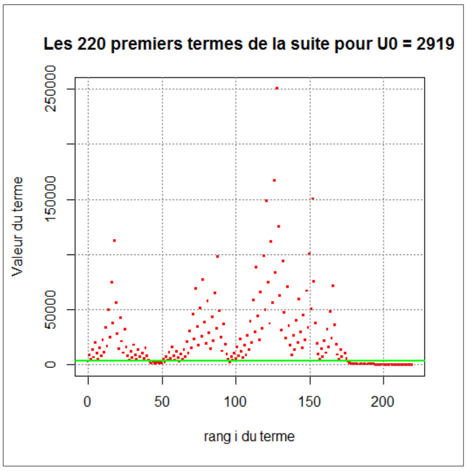
```
U0 est une valeur entière
N est le nombre de termes
calculés

suiteN est le vecteur des N
valeurs calculées.

Affichage

et graphique des N premières
valeurs calculées

La valeur de U0 est marquée sur
le graphique par une ligne
verte.
```



`syracN_B()`



`syracN_B(U0 = 7)`



`syracN_B(U0 = 2919, N = 220)`

Quelques observations intéressantes : Quel est le nombre de termes avant d'arriver à la phase périodique ? Quelle est la valeur maximale atteinte ? Peut-on trouver des algorithmes pour déterminer ces valeurs ?

II.C – EXPLORATION DES LONGUEURS ET DES HAUTEURS DE VOL

L'algorithme suivant permet de calculer le nombre k de termes (longueur du vol) pour atteindre la valeur 1, après laquelle la suite est périodique et la valeur maximale atteinte (ligne bleue) lors de ce vol.

```
# Cette fonction calcule, en fonction de U0, la longueur
# de vol et la hauteur de vol et en fait la représentation
# graphique
syracLH_C = function(U0 = 3){
   if (U0 <= 0 | trunc(U0) != U0) {
      cat("U0 doit être entier naturel différent de 0")
      stop()
   }
   u <- U0 ; suiteK <- U0 ; k <- 0 ; maxU <- U0
   while (u != 1) {
      if (u %% 2 == 0) {u <- u / 2} else {u <- 3 * u + 1}
      suiteK <- c(suiteK, u)
      k <- k + 1
      if (u > maxU) {maxU <- u}
   }
# Affichage des résultats et des graphiques
   cat("Les", k + 1, "premiers termes de la suite sont :",
        suiteK, "\n")
   plot(0:k, suiteK,
         main = paste("Pour U0 =", U0, ",
            La longueur du vol est :", k,
             "\nLa hauteur du vol est :", maxU),
         xlab = "rang i du terme",
         ylab = "Valeur du terme",
         pch = ".", cex = 5, col = "red")
      grid(col = "grey50")
      abline(h = U0, col = "green", lwd = 2)
      abline(h = maxU, col = "blue")
}
```

U0 est représenté par un segment vert.

Tant que u différent de 1, on calcule le terme suivant de la suite.

On compte les rangs

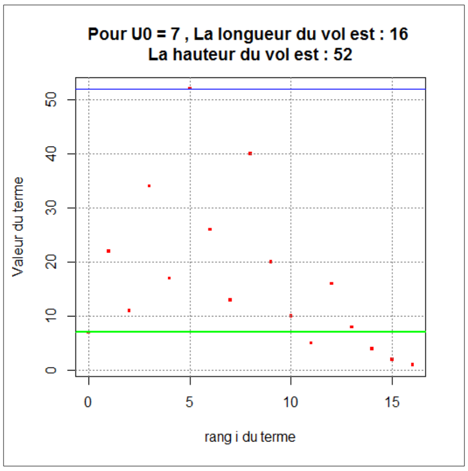
Calcul du maximum (segment bleu).

Affichage des valeurs des termes de la suite

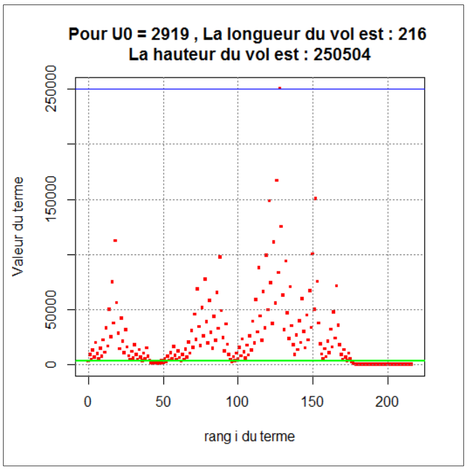
Représentation graphique

La valeur initiale U0 est marquée en vert, la valeur maximale en bleu

syracLH_C()



syracLH_C(U0 = 7)



syracLH_C(2919)

Les valeurs successives des termes de la suite varient de façon “erratique”, il en est de même pour la longueur du vol. La forme des nuages de points obtenus peut prendre des aspects singuliers.

Comment explorer ces valeurs “erratiques” ? J'ai donc eu l'idée d'utiliser les outils de la statistique descriptive (tableau des effectifs des longueurs de vols par exemple) sur des suites de valeurs obtenues par les algorithmes précédents.

II.D – DES STATISTIQUES DESCRIPTIVES POUR DÉCRIRE UNE SUITE !

L'algorithme calcule la suite des longueurs des vols correspondant à une suite de valeurs de U_0 comprises entre deux entiers. Cette suite de valeurs est ensuite décrite par un nuage de points et par un tableau des effectifs établissant une distribution observée des longueurs de vols.

```
# Cette fonction calcule, pour un intervalle de valeurs de
# U0, les durées de vol, k. Graphique de k en fonction de U0
# On constate que des valeurs de k se répètent. D'où l'idée
# de déterminer la distribution des valeurs de k, pour un
# intervalle de valeurs de U0 donné et d'en faire un
# diagramme en barre...
syracDL_D = function(U0Inf = 1, U0Sup = 100){
   vectn <- NULL
   for (i in U0Inf:U0Sup) {
      k <- 0 ; u <- i
      while (u != 1) {
         if (u %% 2 == 0) {u <- u / 2} else {u <- 3 * u + 1}
         k <- k + 1
      }
      vectn <- c(vectn, k)
   }
   effec <- table(vectn)
# *********** Affichage des résultats et des graphiques ****
   layout(c(1, 2), heights = c(2, 1))
   plot(vectn, U0Inf:U0Sup,
      main = paste("Durée k du vol en fonction de U0", "de\n",
         U0Inf, "à", U0Sup),
      ylab = "valeurs de U0",
      xlab = "Durée k du vol",
      pch = ".", cex = 3)
   plot(effec,
      main = paste("Diagramme en barre de la distribution",
        "des valeurs de k", "\npour des valeurs de U0 entre",
         U0Inf, "et", U0Sup),
      xlab = "Durée k du vol",
      ylab = "Effectifs")
}
```

On peut choisir un intervalle quelconque de valeurs de U0

Tant que u différent de 1, on calcule le terme suivant de la suite.

Le rang ainsi trouvé est ajouté au vecteur vectn

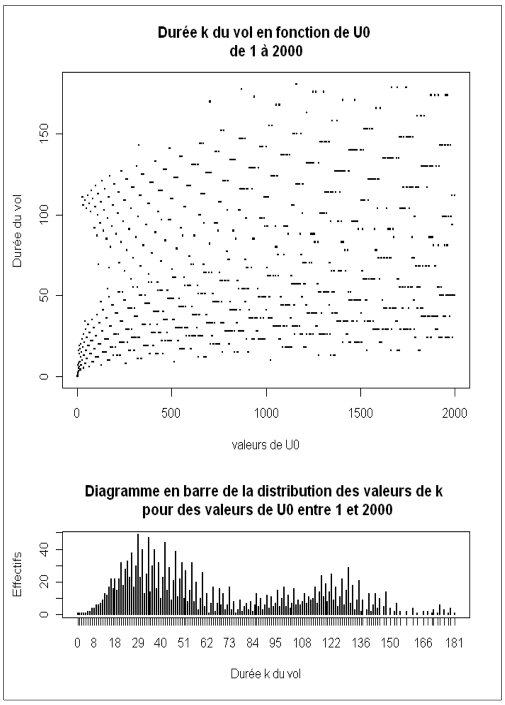
Calcul du tableau des effectifs de la série des durées de vols.

On partage la fenêtre graphique

Graphique en nuage de points des durées de vol en fonction de U0.

Et diagramme en bâtons des effectifs des durées de vol.

On remarquera que la fonction plot adapte le graphique produit en fonction du type de données entrées.




```
syracDL_D(1, 2000)
```

Ce premier graphique en nuage de points montre que certaines durées de vol se répètent quand U0 varie. D'où l'idée d'établir la distributions d'une série de valeurs de durées de vols.

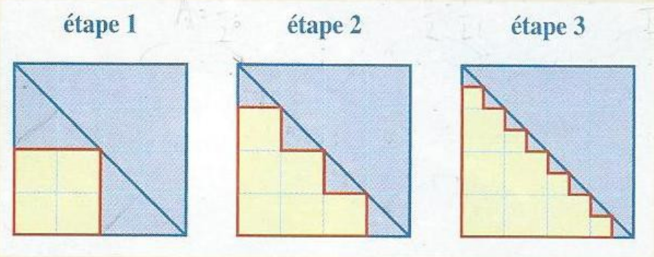
C'est ce qui est représenté dans ce graphique en diagramme en barres.

À partir des suites de valeurs qui nous paraissent chaotiques, ces descriptions nous permettent d'entrevoir des structures, certaines “régularités”. Un premier pas vers l'exploration des valeurs de ces suites.

Séquence 1 Aire et périmètre d'un domaine

La situation à l'étude

Dans un carré de côté 4, on considère une construction en n étapes, selon la description suivante :



À chaque étape, les nouveaux points sur la diagonale sont les milieux des segments découpés sur cette diagonale à l'étape précédente.

L'unité est le côté d'un carreau.

On note S_n l'aire du polygone colorée en jaune, à l'étape n, et P_n le périmètre de ce polygone.

1 La suite $(S_n)_{n \geqslant 1}$

a) Conjecturer le comportement de S_n pour les grandes valeurs de n, à l'aide du dessin.

b) Calculer S_1, S_2 et S_3, puis montrer que, pour $n \geqslant 1$:

$$S_{n+1} = S_n + \frac{1}{2^{n-2}}.$$

c) En déduire les égalités :

$$S_n = 4 + 2 + \ldots + \frac{1}{2^{n-3}} = 8 - \frac{1}{2^{n-3}}.$$

d) Montrer que, pour $n \geqslant 13$, S_n est situé dans l'intervalle $[7{,}999 ; 8]$.

2 La suite $(P_n)_{n \geqslant 1}$

a) Conjecturer le comportement de P_n pour les grandes valeurs de n.

b) Calculer P_1, P_2 et P_3, puis montrer que :

$$P_n = 8 + 4 + 2 + \ldots + \frac{1}{2^{n-4}} = 16 - \frac{1}{2^{n-4}}.$$

c) Déterminer un rang n_0 à partir duquel P_n est situé dans l'intervalle $[15{,}999 ; 16]$.