Une conjecture avec geogebra

Groupe mathématiques dynamiques – Irem de Lyon

Résumé

Usage de l'outil geogebra pour émettre une conjecture sur une expression explicite d'une suite définie par récurrence.

Utilisation possible du tableur, du grapheur, du calcul formel, de python.

La partie démonstration nécessite la partie expérimentale (il est difficile pour un lycéen d'émettre une hypothèse de récurrence correcte sans cette partie expérimentale)

1 Fiche prof

1. Niveau : terminale S.
2. Connaissances : second degré, récurrence.
3. Technicité :
   - du point de vue des mathématiques, il s'agit d'un exercice de début d'année de terminale, la partie expérimentale donne des formules explicites, l'élève n'a plus qu'à vérifier par récurrence ces formules sans autre technique que le calcul algébrique élémentaire.
   - du point de vue logiciel : la technicité peut être assez avancée (mais reste simple si l'on choisit la méthode tableur, les autres techniques présentées ici étant essentiellement à destination de l'enseignant). En donnant beaucoup de détails sur l'usage du logiciel pour la première suite et en demandant aux élèves de refaire l'ensemble de la démarche pour la seconde suite, on a ainsi une situation qui permet en début d'année de prendre en main le logiciel, ce qui pourra dans la suite de l'année permettre des situations laissant un peu plus d'autonomie à l'élève dans la partie expérimentale.
4. Objectifs. L'un des objectifs est ici de lutter contre deux tendances des élèves suite à une partie expérimentale :
   - Première tendance : croire que la partie expérimentale constitue une démonstration. . .Lorsqu'on demande ici pour le devoir de démontrer les formules conjecturées, on force l'élève à revenir sur le statut de ce qui a été obtenu (remarque : 3 copies rendues sur une classe de 35 se contentaient de détailler la résolution du système mis en place dans la méthode faisant intervenir le calcul formel, l'objectif est donc clairement raté pour ces élèves !)
   - Deuxième tendance : refuser de se servir de la partie expérimentale pour raisonner et chercher coûte que coûte à résoudre l'exercice par des méthodes « directes ». Ici, surtout en début d'année, cette démarche mène la plupart des élèves à une impasse, alors que chercher à prouver par récurrence directement les formules obtenues dans la partie expérimentale est accessible (le DM, sur une classe de 35 élèves, a donné majoritairement de bonnes copies). Cette deuxième tendance est peut être liée à la précédente : l'élève pense que la partie expérimentale prouve, mais comme le prof demande de prouver. . .Cette deuxième tendance peut être aussi la nôtre (enseignants), qui allons chercher à appliquer des méthodes expertes en sautant directement à la deuxième partie (démontrer) sans passer par la partie expérimentale, il nous faut parfois lutter contre des "réflexes mathématiques" pour construire des situations de classe où le raisonnement s'appuie fortement sur la partie expérimentale.

2 Le problème

On considère deux suites (u_n) et (v_n) définies par :

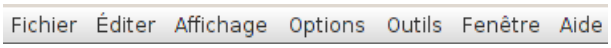
$$\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N}\ u_{n+1} = v_n + n - 1 \end{cases} \qquad \begin{cases} v_0 = -1 \\ \forall n \in \mathbb{N}\ v_{n+1} = -u_n + 2v_n + n \end{cases}$$

Émettre une conjecture sur une formule explicite de u_n et v_n en fonction de n à l'aide du logiciel geogebra. Puis démontrer.

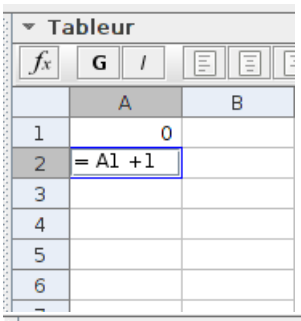
3 Partie expérimentale

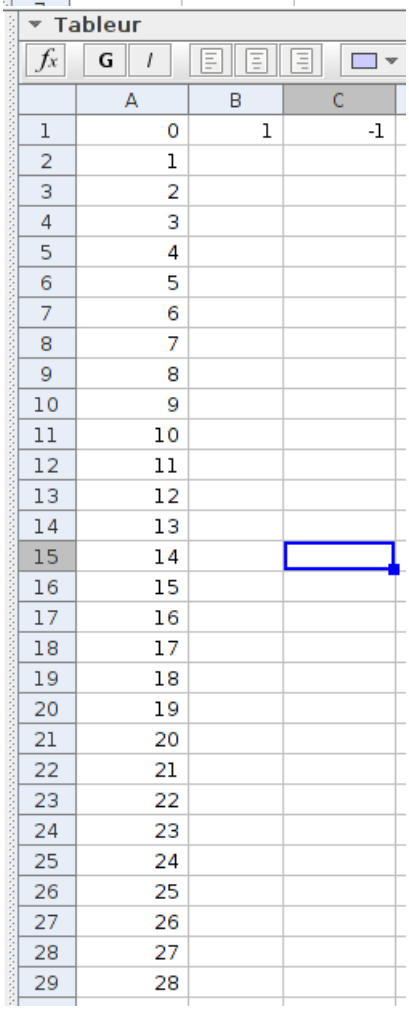
3.1 Calcul des premiers termes de la suite et affichage dans le tableur

3.1.1 Calcul des premiers termes de la suite et affichage dans le tableur : par le tableur.

Dans le menu « affichage » 

cocher « tableur ».
On entre en colonne A les valeurs des premiers indices, ainsi que les valeurs de u_0 et de v_0 en cellules B1 et C1.





Puis les formules de récurrence :

Tableur

	A	B	C
1	0	1	-1
2	1	= C1 + A1 -1	
3	2		
4	3		
5	4		

Tableur

	A	B	C
1	0	1	-1
2	1	-2	=- B1 +2* C1 + A1

On sélectionne alors les deux cellules B2, C2 et on tire vers le bas.

Tableur

	A	B	C
1	0	1	-1
2	1	-2	-3

Tableur

	A	B	C
1	0	1	-1
2	1	-2	-3
3	2	-3	-3
4	3	-2	-1
5	4	1	3
6	5	6	9
7	6	13	17
8	7	22	27
9	8	33	39
10	9	46	53
11	10	61	69
12	11	78	87
13	12	97	107
14	13	118	129
15	14	141	153
16	15	166	179
17	16	193	207
18	17	222	237
19	18	253	269
20	19	286	303
21	20	321	339
22	21	358	377
23	22	397	417
24	23	438	459
25	24	481	503
26	25	526	549
27	26	573	597
28	27	622	647
29	28	673	699

3.1.2 Calcul des premiers termes de la suite et affichage dans le tableur : avec python.

Un script python (version geogebra ... ?) permet également de remplir les colonnes. Ce qui permet un travail sur l'algorithmique.

Pour cela, dans l'onglet « interactive » de la fenêtre python, on peut entrer le code suivant :

Script 1 – PythonGG

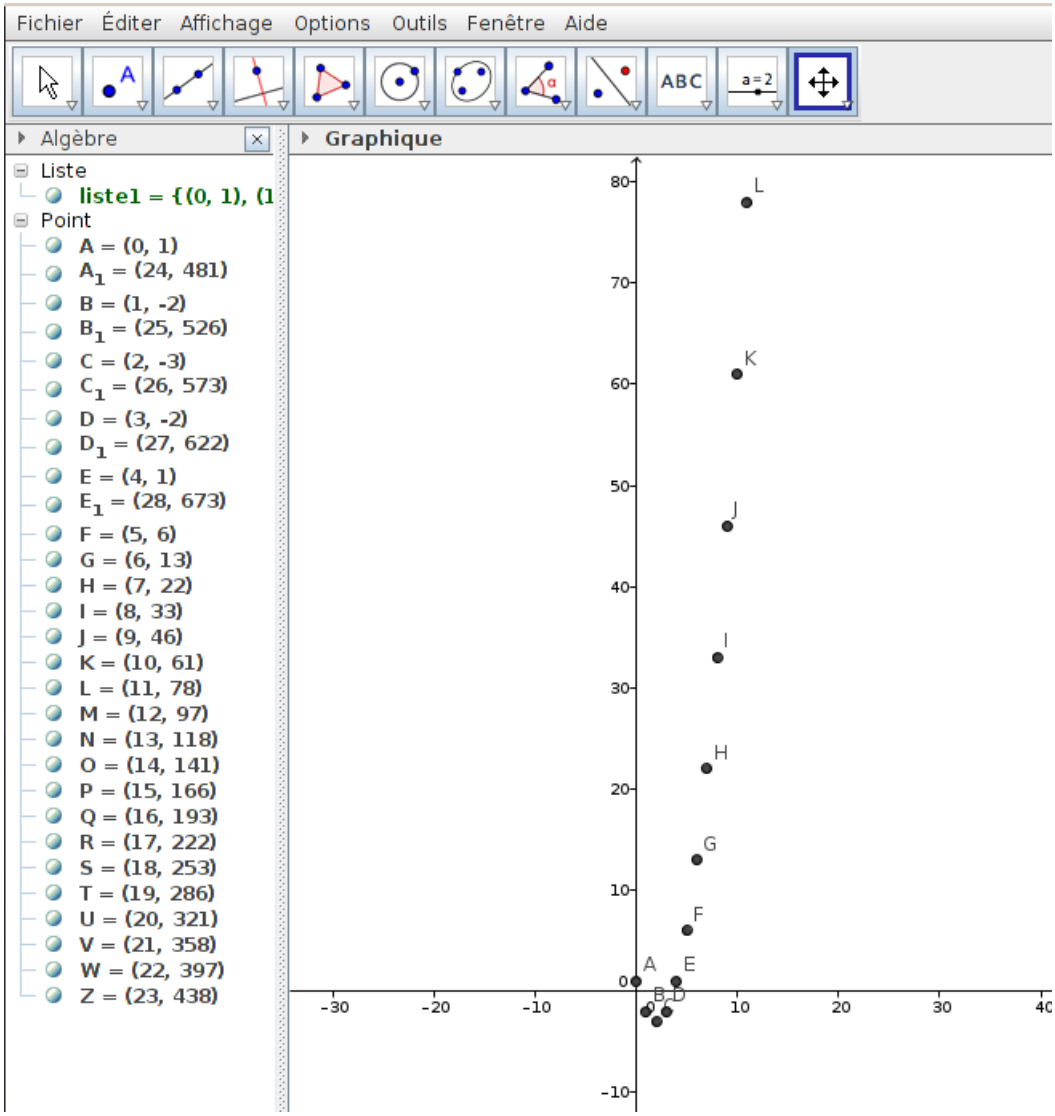
```
def u(n):
        if n>0 : return v(n-1)+n-2
        else : return 1
def v(n):
        if n>0 : return -u(n-1)+2*v(n-1)+n-1
        else : return -1

for j in range(1,22):
        $['A',j]=j-1
        $['B',j]=u(j-1)
        $['C',j]=v(j-1)
```

3.2 Création d'une représentation graphique.

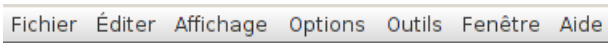
3.2.1 Création d'une représentation graphique : par le tableur

On sélectionne les cellules A1 à A29 et les cellules B1 à B29, puis clic droit sur la sélection, et dans le menu qui s'affiche, on choisit créer/liste de points. On obtient une représentation graphique de la suite (u_n).



Remarque. Pour le travail analogue sur la suite v, on sera amené à selectionner des cellules de colonnes ne se trouvant pas côte à côte : il suffit pour cela de laisser le bouton ctrl enfoncé lors de la sélection des cellules à la souris.

3.2.2 Création d'une représentation graphique : avec python.

Dans le menu « affichage » 

cocher « python » (avec les versions de geogebra l'intégrant).

Dans l'onglet « interactive » de la fenêtre python, entrer le code suivant (ligne par ligne) dans la petite fenêtre de saisie en bas de l'écran :

Script 2 – PythonGG

```
xcoord=range(0,20)
def u(n):
        if n==0:return 1
        else  : return v(n-1)+n-2
def v(n):
        if n==0: return -1
        else  : return -u(n-1)+2*v(n-1)+n-1
termesu=[u(j) for j in xcoord]
termesv=[v(j) for j in xcoord]
```

```
pointsu = [Point(x, y) for x, y in zip(xcoord,termesu)]
```

On peut vouloir éviter une définition récursive des deux suites pour un travail sur les boucles :

Script 3 – PythonGG

```
xcoord=range(0,20)
def uv(n) :
        u,v=1,-1
        for j in range(n) :
                u,v=v+j-1,-u+2*v+j
        return u,v
termesu=[uv(j)[0] for j in xcoord]
termesv=[uv(j)[1] for j in xcoord]
pointsu=[Point(x,y) for x,y in zip(xcoord,termesu)]
```

3.2.3 Création d'une représentation graphique : par macros.

1. On crée les points $U_0 = (0\ ;\ 1)$ et $V_0 = (0\ ;\ -1)$.

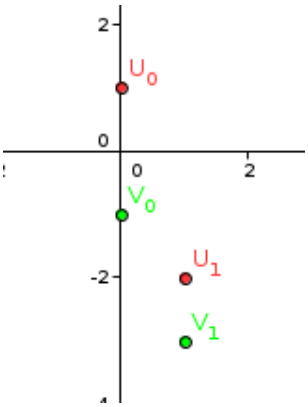
   Saisie: V_0=(0,-1)

   On crée ensuite les points U_1 et V_1 :

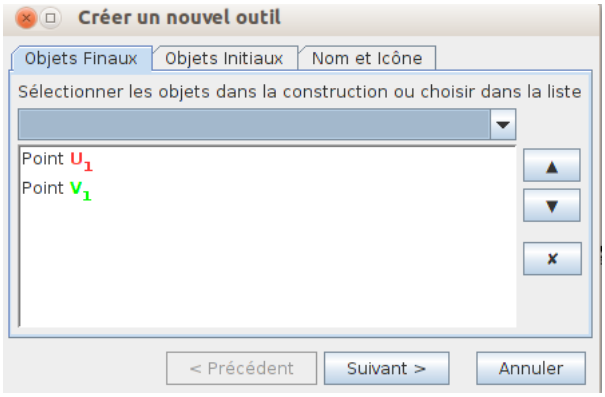
   Saisie: U_1=(x(U_0)+1 , y(V_0)+x(U_0)-1)

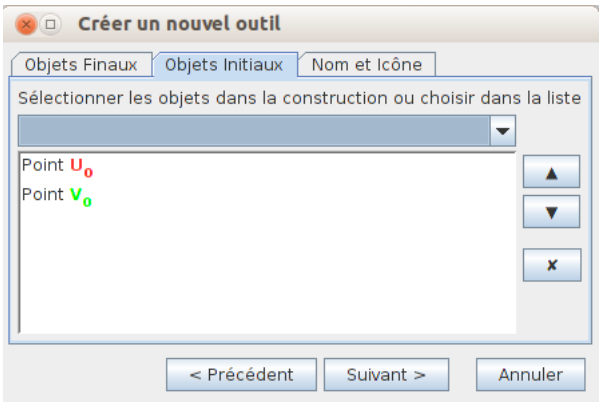
   Saisie: V_1=(x(V_0)+1 , -y(U_0)+2*y(V_0)+ x(V_0))

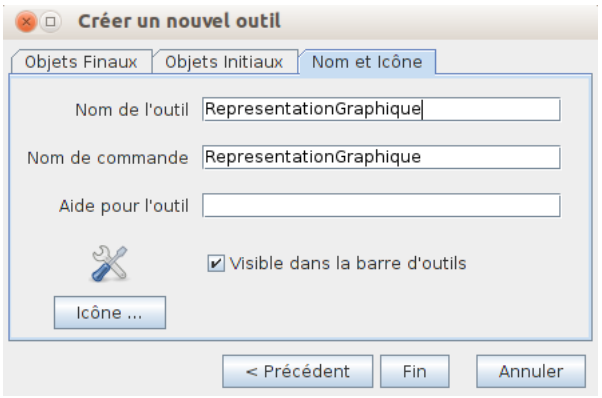
   Avec un clic droit sur les points, propriétés puis onglet couleur, on met les points de la représentation de la suite (u_n) en rouge et ceux de la suite (v_n) en vert.



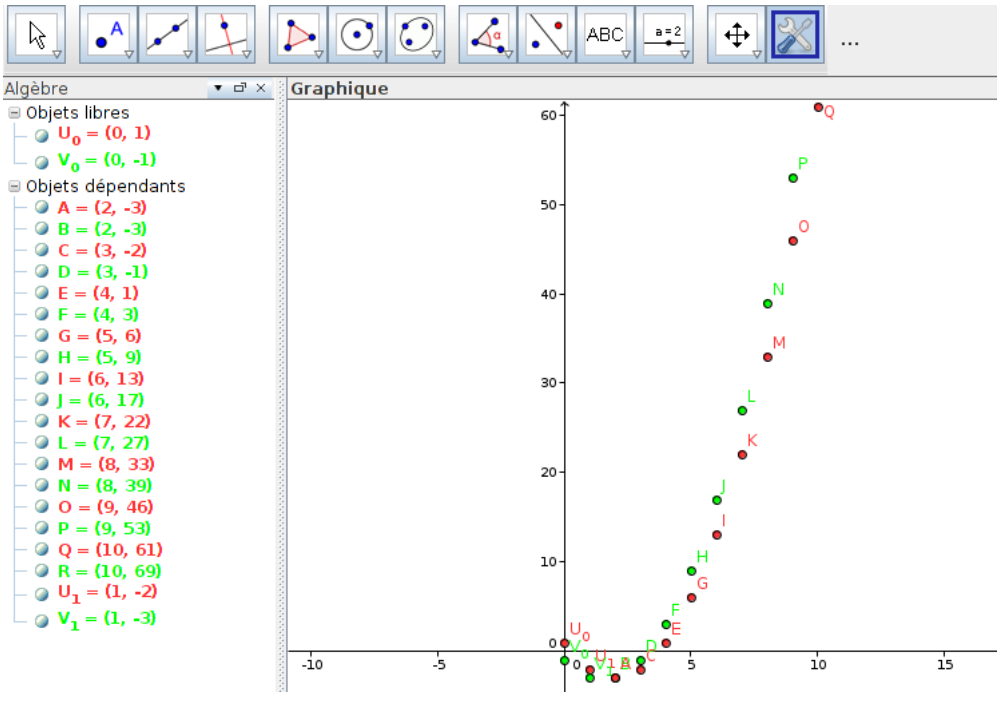
   Dans la barre des menus, on sélectionne alors outils/créer un nouvel outil.







On peut ensuite sélectionner l'outil RepresentationGraphique ainsi crée en cliquant sur la nouvelle icône apparue dans la barre des icônes. On sélectionne ensuite dans la fenêtre graphique ou dans la fenêtre algèbre le point U_1 puis le point V_1. Les points correspondants à U_2 et V_2 sont alors créés (avec les couleurs rouge et verte). Et ainsi de suite.



3.2.4 Création d'une représentation graphique : par commandes script en ligne de saisie.

On peut procéder de la façon suivante :

1. On définit le point $W_0 = (u_0\ ;\ v_0) = (1\ ;\ -1)$.

   Saisie: `W_{0} = (1, -1)`

2. On définit ensuite les points W_n par la relation de récurrence :

$$\begin{pmatrix} u_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} u_n \\ v_n \end{pmatrix} + \begin{pmatrix} n-1 \\ n \end{pmatrix}$$

   ce que l'on peut faire à l'aide de la commande suivante :

   Saisie: `Exécute[ Séquence["W_{"+(i+1)+"}={{0, 1}, {-1, 2}} * W_{"+i+"}+("+i+"-1,"+i+")",i,0,20]]`

3. On peut ensuite faire disparaitre de l'écran graphique ces points W (maintenir la touche maj enfoncée pour les sélectionner tous en même temps puis clic droit, on décoche "Afficher l'objet".)

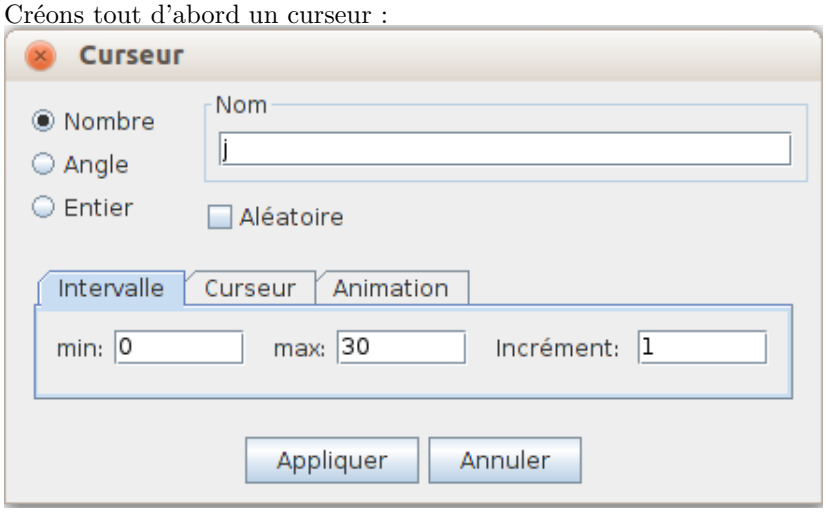
4. Il nous reste à générer la représentation graphique des suites (u_n) et (v_n) :

   Saisie: `Exécute[ Séquence["U_{"+(i)+"}=( "+i+",x(W_{"+i+"}))",i,0,20]]`

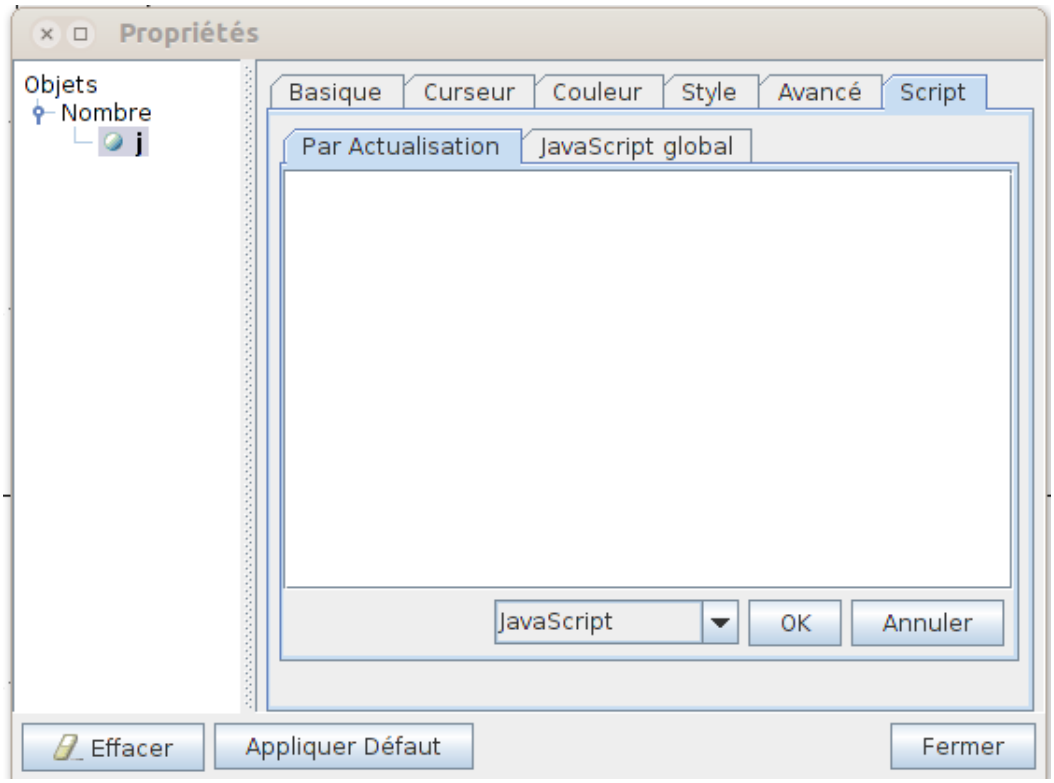
   Saisie: `Exécute[ Séquence["V_{"+(i)+"}=( "+i+",y(W_{"+i+"}))",i,0,20]]`

3.2.5 Création d'une représentation graphique : par script javascript.

Créons tout d'abord un curseur :



Nous allons associer un script à ce curseur : clic droit sur le curseur/ propriétés. Dans l'onglet script/ par actualisation on sélectionne le javascript.



On tape le script suivant :

Script 4 – JavaScript

```
with(ggbApplet)
{

        function u(n)
        {
                if (n>0)
                {return v(n-1)+n-2;}
                else
                {return 1;}
        }

        function v(n)
        {
                if (n>0)
                {return -u(n-1)+2*v(n-1)+n-1;}
                else
                {return -1;}
        }

var j=getValue('j');

for(var i =0;i<j;i++)
    evalCommand("U_{"+i+"}=("+i+","+u(i)+")");

}
```

On valide par OK et on ferme.

Il n'y a plus qu'à modifier le curseur pour créer les j premiers points de la représentation graphique de la suite (u_n).

On peut vouloir également faire disparaître les points d'indice dépassant la valeur de j. Pour cela, le script suivant

les créé tous puis rend visibles ou invisibles les points suivant la valeur donnée au curseur j dans la feuille de graphique geogebra :

Script 5 – JavaScript

```
with(ggbApplet)
{

        function u(n)
        {
                if (n>0)
                {return v(n-1)+n-2;}
                else
                {return 1;}
        }

        function v(n)
        {
                if (n>0)
                {return -u(n-1)+2*v(n-1)+n-1;}
                else
                {return -1;}
        }

var j=getValue('j');



for(var i =0;i<30;i++)
{evalCommand("U_{"+i+"}=("+i+","+u(i)+")");}

for(var i=0;i<j;i++)
{evalCommand("SoitVisibleDansVue[U_{"+i+"},1,true]");}

for(var i=j;i<30;i++)
{evalCommand("SoitVisibleDansVue[U_{"+i+"},1,false]");}

}
```

Si l'on préfère éviter le with(ggbApplet) :

Script 6 – JavaScript

```
function u(n)
{
        if (n>0)
                return v(n-1)+n-2;
        else
                return 1;
}

function v(n)
{
        if (n>0)
                return -u(n-1)+2*v(n-1)+n-1;
        else
                return -1;
}

var j=ggbApplet.getValue('j');

```

```
19
20
21 for(var i =0;i<30;i++)
22 ggbApplet.evalCommand("U_{"+i+"}=("+i+","+u(i)+")");
23
24 for(var i=0;i<j;i++)
25 ggbApplet.evalCommand("SoitVisibleDansVue[U_{"+i+"},1,true]");
26
27 for(var i=j;i<30;i++)
28 ggbApplet.evalCommand("SoitVisibleDansVue[U_{"+i+"},1,false]");
```

Dans le script suivant, plutôt que de rendre invisibles les points dans la partie graphique, on les détruit réellement (ils disparaissent donc également de la fenêtre d'algèbre) :

Script 7 – JavaScript

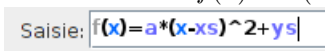
```
1  function u(n)
2  {
3      if (n>0) return v(n-1)+n-2;
4      else  return 1;
5  }
6
7  function v(n)
8  {
9      if (n>0)  return -u(n-1)+2*v(n-1)+n-1;
10     else  return -1;
11 }
12
13 var j=ggbApplet.getValue('j');
14
15 for(var i =0;i<j;i++)
16 ggbApplet.evalCommand("U_{"+i+"}=("+i+","+u(i)+")");
17
18 for(var i=j; i<100; i++)
19 {
20     var objet="U_{"+i+"}";
21     if (ggbApplet.isDefined(objet))
22         ggbApplet.evalCommand("Delete("+objet+")");
23 }
```

3.3 Démarche pour une conjecture

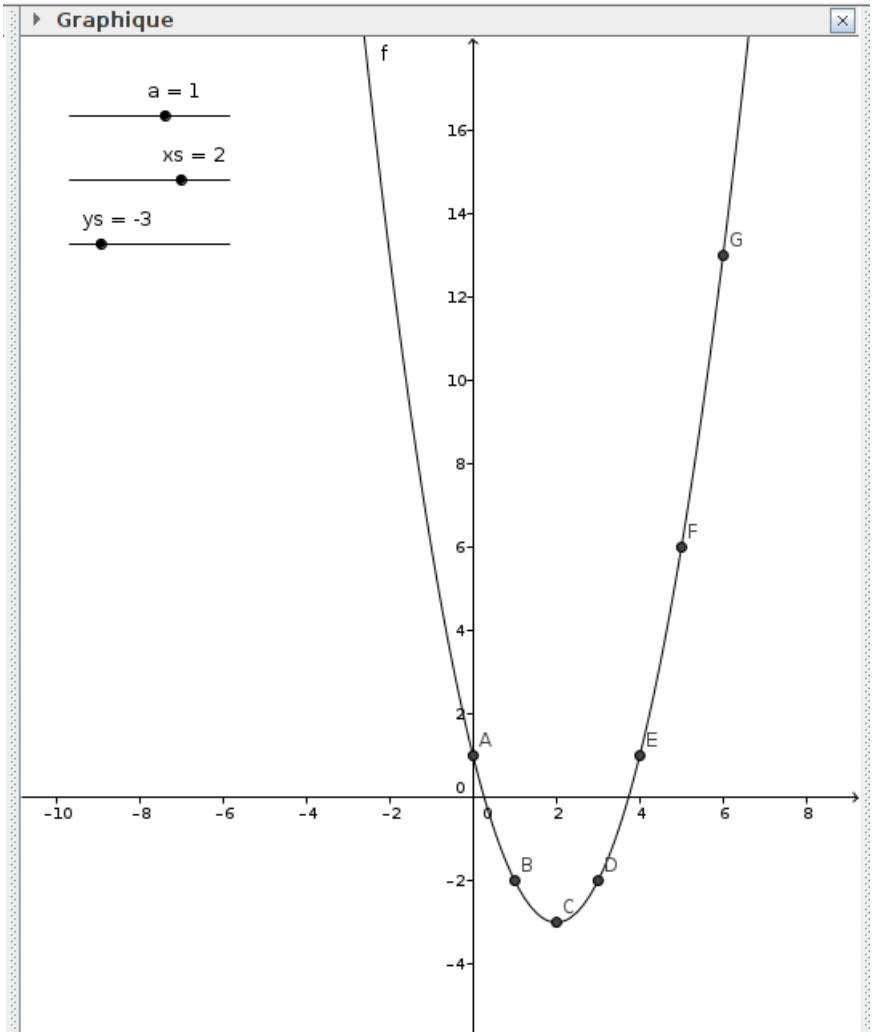
3.3.1 Première démarche pour une conjecture : à l'aide de curseurs.

On conjecture, au vu de la représentation graphique obtenue, une expression de la forme $u_n = an^2 + bn + c$.
Pour estimer les coefficients, on définit en ligne de saisie trois variables : a=1, xs=1, ys=1.

Puis la fonction $f(x) = a\,(x - x_s)^2 + y_s$.



Un clic droit dans la fenêtre d'algèbre sur a, xs, ys permet de les afficher en tant que curseurs. En agissant sur ces curseurs, on peut alors ajuster la courbe représentative de f de façon à la superposer au mieux aux points de la représentation de u et ainsi émettre une conjecture sur les valeurs des coefficients a, b, c.

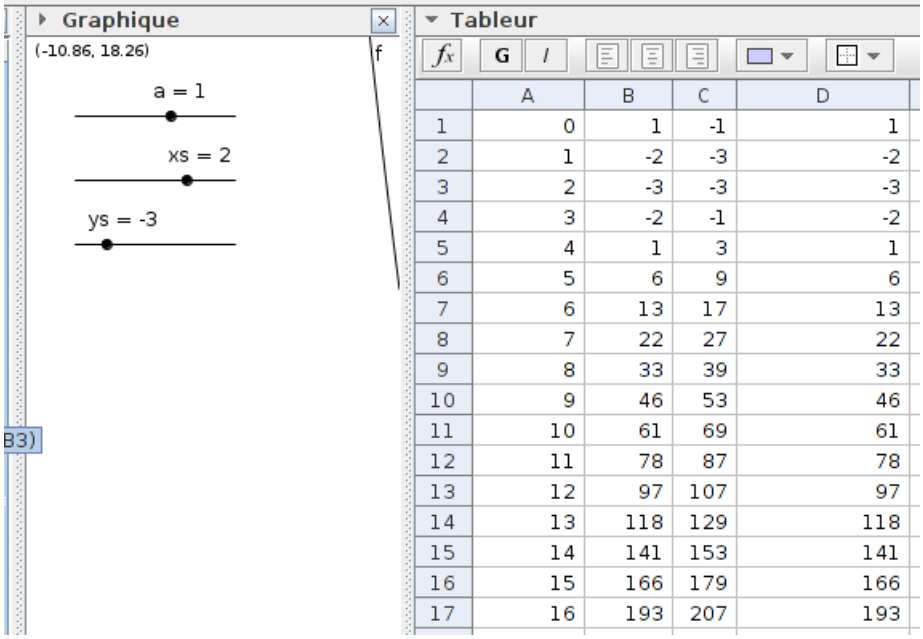


3.3.2 Renforcer la conjecture : retour au tableur.

On peut entrer en cellule D1 du tableur la formule =f(A1) puis tirer cette formule vers le bas afin de comparer les valeurs obtenues avec celles de la colonne B.

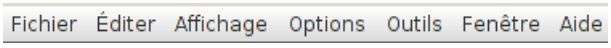
Tableur

	A	B	C	D
1	0	1	-1	=f(A1)
2	1	-2	-3	

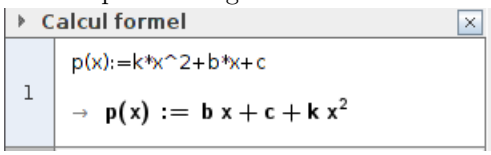


	A	B	C	D
1	0	1	-1	1
2	1	-2	-3	-2
3	2	-3	-3	-3
4	3	-2	-1	-2
5	4	1	3	1
6	5	6	9	6
7	6	13	17	13
8	7	22	27	22
9	8	33	39	33
10	9	46	53	46
11	10	61	69	61
12	11	78	87	78
13	12	97	107	97
14	13	118	129	118
15	14	141	153	141
16	15	166	179	166
17	16	193	207	193

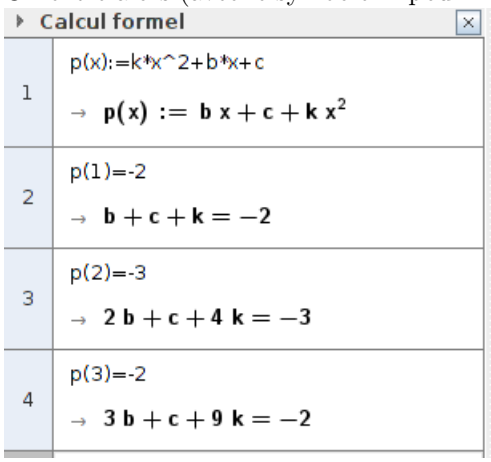
3.3.3 Seconde démarche pour une conjecture : avec le calcul formel.

Dans le menu « affichage » 

cocher « calcul formel ».
Dans la première ligne de la fenêtre de calcul formel, entrer p(x)=k*x^2+b*x+c.

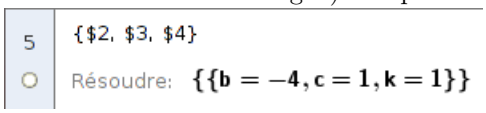


On lit dans le tableur $u_1 = -2$, $u_2 = -3$, $u_3 = -2$.
On entre alors (avec le symbole = pour introduire des équations) les lignes suivantes :



En maintenant la touche ctrl enfoncée, on sélectionne les lignes de ces équations (en cliquant sur la case bleue contenant le numéro de la ligne). On peut alors relâcher la touche ctrl et cliquer sur le bouton 

et on obtient :



ce qui confirme la conjecture précédente.

3.3.4 Troisième démarche pour une conjecture : observation...

On peut également évidemment pousser les élèves à se remémorer leurs connaissances du second degré.
Par exemple :
L'observation des résultats dans le tableur montre que l'on dispose de points $B(1;-2)$ et $D(3;-2)$ symétriques par rapport à la droite d'équation $x=2$. Le point $C(2;-3)$ est alors nécessairement le sommet si la courbe est bien une parabole.
La fonction trinôme recherchée doit donc avoir une expression de la forme $f(x)=a\,(x-2)^2-3$.

3.3.5 Quatrième démarche pour une conjecture : avec python.

Faisons le pari que les coefficients de la parabole sont des entiers.
On peut automatiser une recherche de la bonne expression par un algorithme (à entrer dans l'onglet « interactive ») :

Script 8 – PythonGG

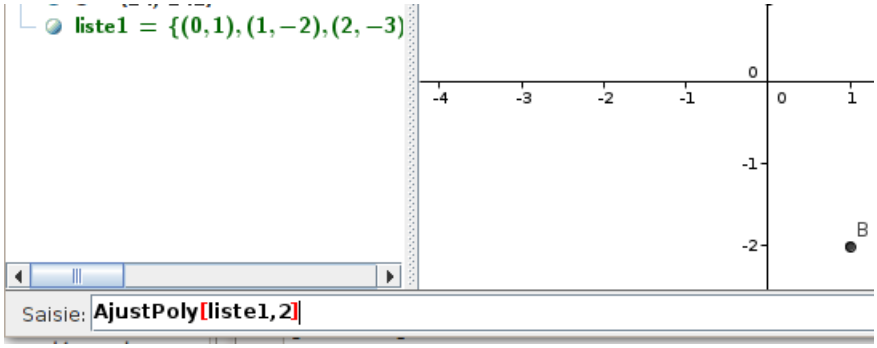
```
def f(x,a,b,c):
        return a*x**2+b*x+c

for a in range(-10,10):
        for b in range(-10,10):
                for c in range(-10,10):
                        if f(1,a,b,c)==-2 and f(2,a,b,c)==-3 and f(3,a,b,c)==-2:
                                def g(x):return f(x,a,b,c)
                                $g=Function(g)
```

La fonction $g\colon x\longmapsto x^2-4x+1$ est créée, ainsi que sa représentation graphique.

3.3.6 Cinquième démarche pour une conjecture : avec AjustPoly.

AjustPoly[<Liste Points>,<Degré>] : calcule une régression polynomiale de degré n.
Après avoir créé la liste de points à partir du tableur (sélection des colonnes A et B, clic droit, créer/ liste de points), on utilise AjustPoly et la liste créée :



La fonction $x\longmapsto x^2-4x+1$ est immédiatement créée.

3.4 Conjecture pour la suite (v_n)

Un avantage de cette situation : même si pour la première suite, on est amené à donner tout le détail technique permettant de réaliser la conjecture, les élèves doivent immédiatement remettre en oeuvre la démarche pour la seconde suite, ce qui évite en partie qu'il ne reste de la séance qu'une impression de presse-bouton.
La conjecture obtenue pour cette seconde suite : $v_n=n^2-3n-1$.

4 Démonstration

On établit par récurrence la propriété suivante :

$$P_n : u_n = n^2 - 4n + 1 \text{ et } v_n = n^2 - 3n - 1$$

5 Autre approche

On peut simplifier le travail à faire en demandant aux élèves une conjecture, puis une preuve, d'une expression explicite pour la suite (w_n) définie par $w_n = v_n - u_n$ pour $n \in \mathbb{N}$ (suite arithmétique).