

JSXGraph -- SageMath

Cascimodot 2024
LA 40E JOURNÉE DU PROJET

E. Eveno - C. Piatecki - Y. Stroppa

Sommaire

- Présentation de la problématique
- Une solution
 - visuel
 - calcul
- Problèmes rencontrés
- Perspectives

Problématique

Construire des graphes/représentations de qualité de haut niveau

Contexte pédagogique

Contexte Recherche

Les rendre interactifs -- amélioration de la compréhension

Disposer de solveur de résolutions analytique et/ou numérique

Disposer de moyen de calculs adaptés aux besoins dans un contexte Web

Développer une communauté autour de ces pratiques

Vulgariser les moyens et s'ouvrir sur d'autres

Partager les approches et expériences (communautaire)

Usage plus large que le notebook (dans un contexte web)

JSXGraph -- javascript

<https://jsxgraph.uni-bayreuth.de/wp/index.html> -- v1.9.1

Dynamic Mathematics with JavaScript

Avantages :

==> librairie js : portable

==> contexte javascript : Accessibilité à toutes les fonctionnalités du langage.

Objet et événementiel

==> permet d'enrichir la qualité visuel des représentations graphiques

==> simplifie leur utilisation ; simplifie l'écriture du code résultant.

==> dispose d'objet scientifiques : functiongraph, curve

==> gère le 2D et 3D

==> fonctions de calculs

Inconvénients :

==> ne gère pas encore le responsive

JessieCode : bibliothèque complémentaire de fonctions mathématiques

JSXGraph en action

Simplicité des dépendances entre les objets ou comment rendre les objets dépendants ?

Quels sont les objets à notre disposition sur lesquels attendre et gérer une interaction utilisateur :

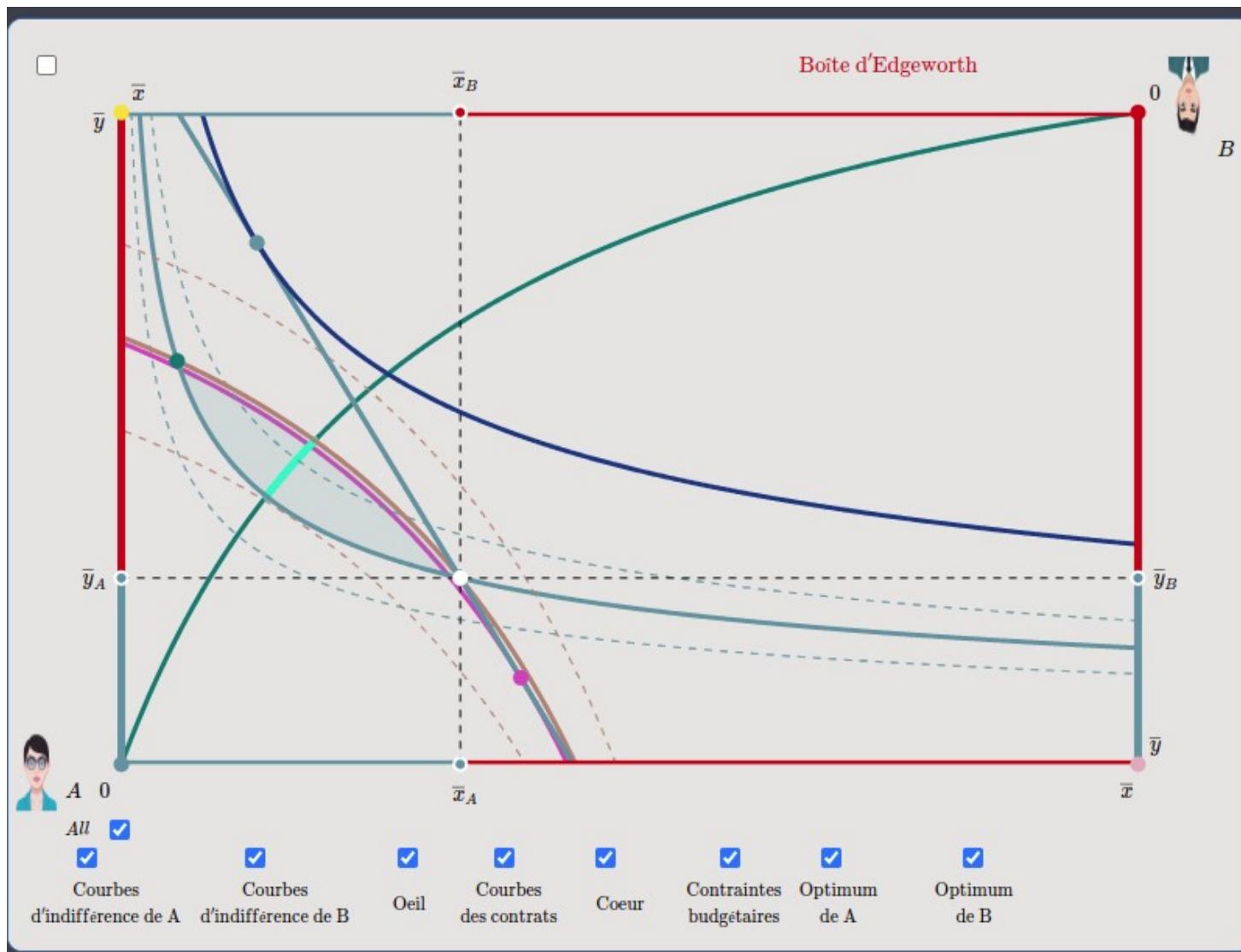
Slider

Glider

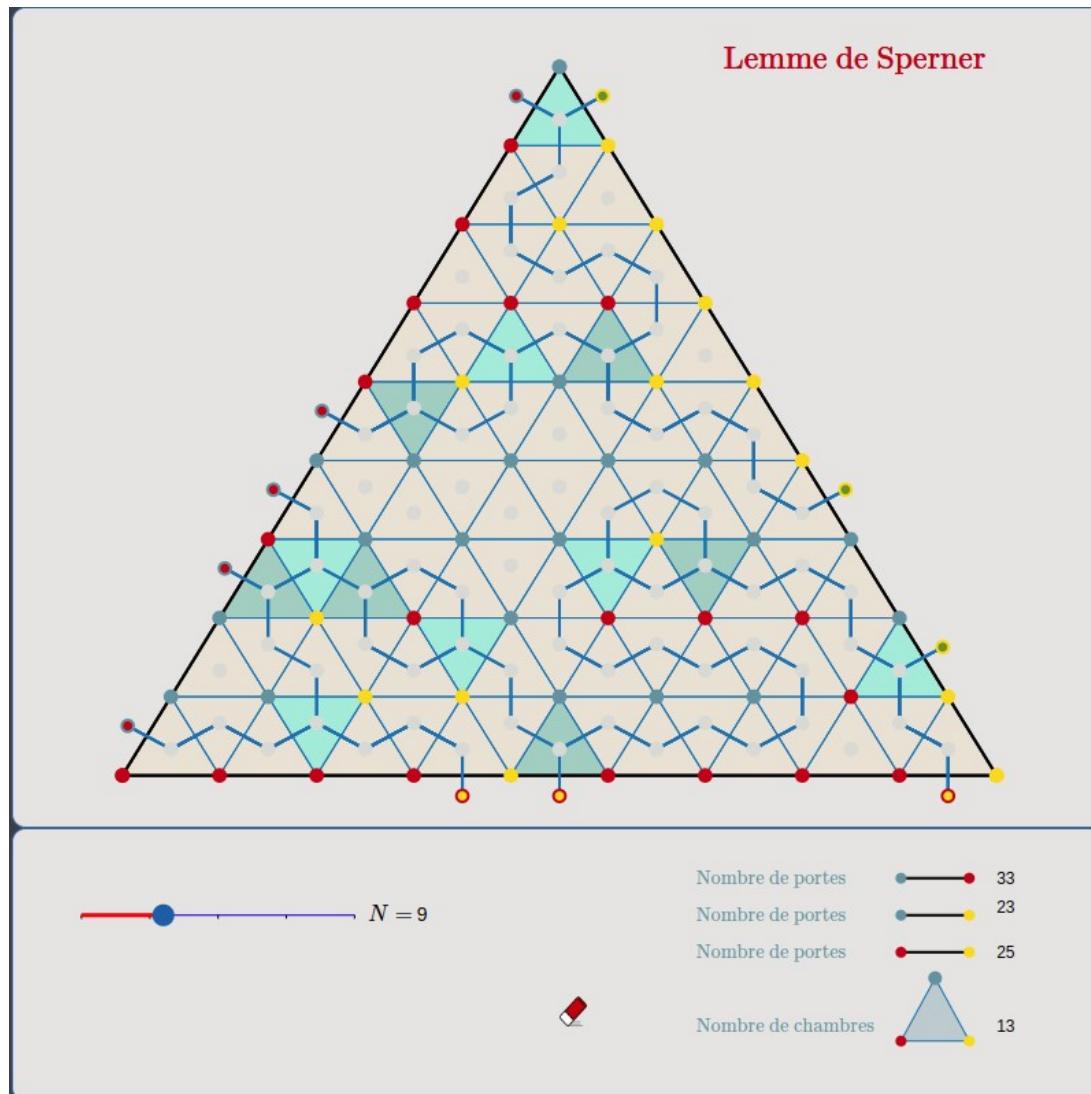
Comment relier les éléments entre eux pour modifier le visuel ?

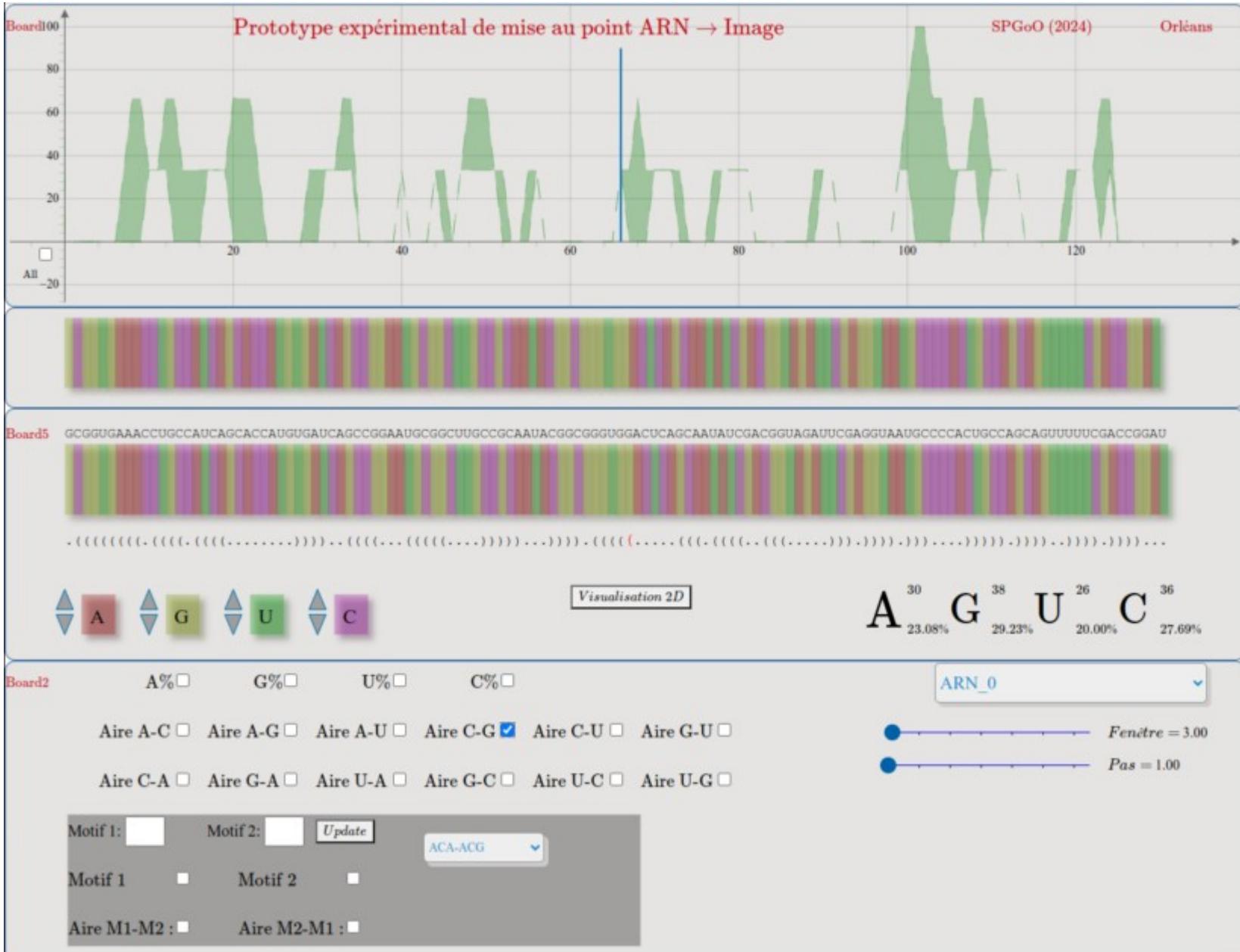
==> définir dans les propriétés des objets à relier des valeurs fournies par des fonctions

Quelques exemples



Quelques exemples





Board4

Liste des actions

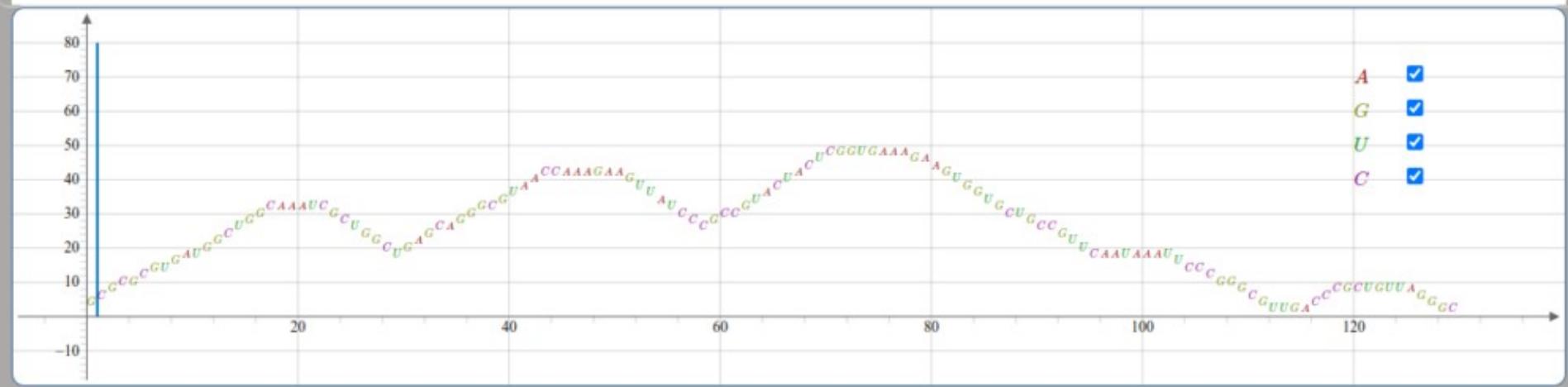
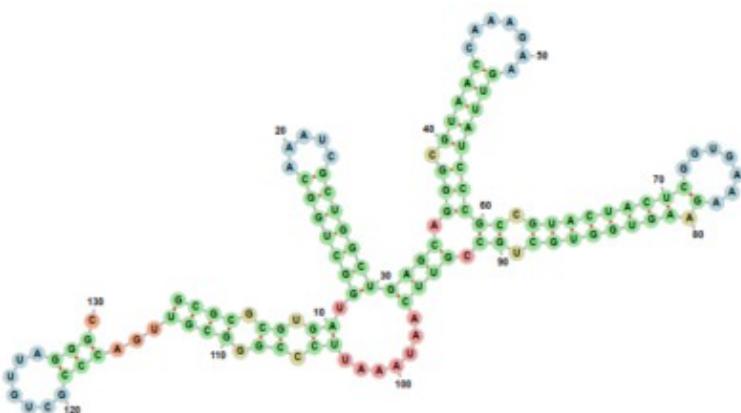


Caractéristiques de l'ARN

pA : 23.08%
 pAA : 3.85%
 pUA : 3.08%
 pGA : 6.92%
 pCA : 8.46%
 pU : 20.00%
 pAU : 7.69%
 pUU : 3.08%
 pGU : 4.62%
 pCU : 3.08%
 pG : 29.23%
 pAG : 5.38%
 pUG : 6.92%
 pGG : 6.92%
 pCG : 8.46%
 pC : 27.69%
 pAC : 5.38%
 pUC : 4.62%
 pGC : 10.00%
 pCC : 6.92%

Fenêtre de visualisation 2D x

[ViennaRNA Web Services](#)



Board0

Prototype expérimental de mise au point ARN → Image

Mise au point

SPGoO (2024)

Size iso Size Centrée de longueur
 webp jpeg png svg

100



P_R10_2749::NC_000913.3:3281563-3281905(+)--->(342)



P_R10_2762::NC_000913.3:3308249-3308372(+)--->(123)



M_R10_3500::NC_000913.3:4352137-4352384(-)--->(247)



M_R10_3631::NC_000913.3:4503571-4503983(-)--->(412)



M_R10_3637::NC_000913.3:4505394-4506172(-)--->(778)

SageMath

<https://www.sagemath.org/> : 10.3

- Environnement de calcul
- Plusieurs langages accessibles
 - python
 - Julia
 - sage
 - R Stats
 - maxima
 - octave
 - gap
 -

Autres exemples

- Programmation en nombre entiers

Zone calcul

Problème 6-6

Code pour calculer la solution

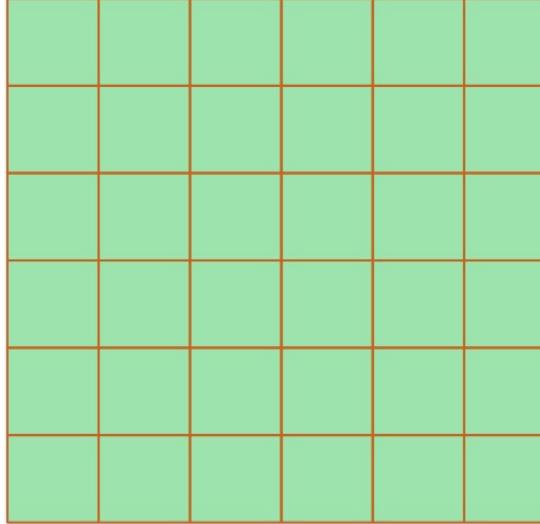


```
1 p = MixedIntegerLinearProgram(maximization=False, solver="GLPK")
2 x = p.new_variable(integer=True, nonnegative=True)
3 y = p.new_variable(integer=True, nonnegative=True)
4 p.add_constraint(x[0]+x[1]+x[6]-2*y[0]==1);
5 p.add_constraint(x[1]+x[0]+x[2]+x[7]-2*y[1]==1);
6 p.add_constraint(x[2]+x[1]+x[3]+x[8]-2*y[2]==1);
7 p.add_constraint(x[3]+x[2]+x[4]+x[9]-2*y[3]==1);
8 p.add_constraint(x[4]+x[3]+x[5]+x[10]-2*y[4]==1);
9 p.add_constraint(x[5]+x[4]+x[11]-2*y[5]==1);
10 p.add_constraint(x[6]+x[7]+x[0]+x[12]-2*y[6]==1);
11 p.add_constraint(x[7]+x[6]+x[8]+x[1]+x[13]-2*y[7]==1);
12 p.add_constraint(x[8]+x[7]+x[9]+x[2]+x[14]-2*y[8]==1);
13 p.add_constraint(x[9]+x[8]+x[10]+x[3]+x[15]-2*y[9]==1);
14 p.add_constraint(x[10]+x[9]+x[11]+x[4]+x[16]-2*y[10]==1);
15 p.add_constraint(x[11]+x[10]+x[5]+x[17]-2*y[11]==1);
16 p.add_constraint(x[12]+x[13]+x[6]+x[18]-2*y[12]==1);
```

Sol(L, C) :

A vous de jouer

Fiver



Cliquer sur une cellule pour changer de couleur
Objectif : toutes les cellules de couleur jaunes

N (Lignes)

M (Colonnes)

Zone résultat

ex : <https://jed.educandgames.org/cdsFiver/jeu.html>

Autres exemples

- Programmation en nombre entiers

Zone calcul

Jeu B

Code pour calculer la solution

```
p = MixedIntegerLinearProgram(maximization=False, solver="GLPK")
x = p.new_variable(integer=True, nonnegative=True)
y = p.new_variable(integer=True, nonnegative=True)
p.add_constraint(x[0][0][1]+x[0][0][2]*x[0][0][3]+x[0][0][4]*x[0][0][5])
p.add_constraint(x[0][1][1]+x[0][1][2]*x[0][1][3]+x[0][1][4]*x[0][1][5])
p.add_constraint(x[0][2][1]+x[0][2][2]*x[0][2][3]+x[0][2][4]*x[0][2][5])
p.add_constraint(x[0][3][1]+x[0][3][2]*x[0][3][3]+x[0][3][4]*x[0][3][5])
p.add_constraint(x[0][4][1]+x[0][4][2]*x[0][4][3]+x[0][4][4]*x[0][4][5])
p.add_constraint(x[0][5][1]+x[0][5][2]*x[0][5][3]+x[0][5][4]*x[0][5][5])
p.add_constraint(x[0][6][1]+x[0][6][2]*x[0][6][3]+x[0][6][4]*x[0][6][5])
p.add_constraint(x[0][7][1]+x[0][7][2]*x[0][7][3]+x[0][7][4]*x[0][7][5])
p.add_constraint(x[1][0][1]+x[1][0][2]*x[1][0][3]+x[1][0][4]*x[1][0][5])
p.add_constraint(x[1][1][1]+x[1][1][2]*x[1][1][3]+x[1][1][4]*x[1][1][5])
p.add_constraint(x[1][2][1]+x[1][2][2]*x[1][2][3]+x[1][2][4]*x[1][2][5])
p.add_constraint(x[1][3][1]+x[1][3][2]*x[1][3][3]+x[1][3][4]*x[1][3][5])
p.add_constraint(x[1][4][1]+x[1][4][2]*x[1][4][3]+x[1][4][4]*x[1][4][5])
```

Sol(L, C) :

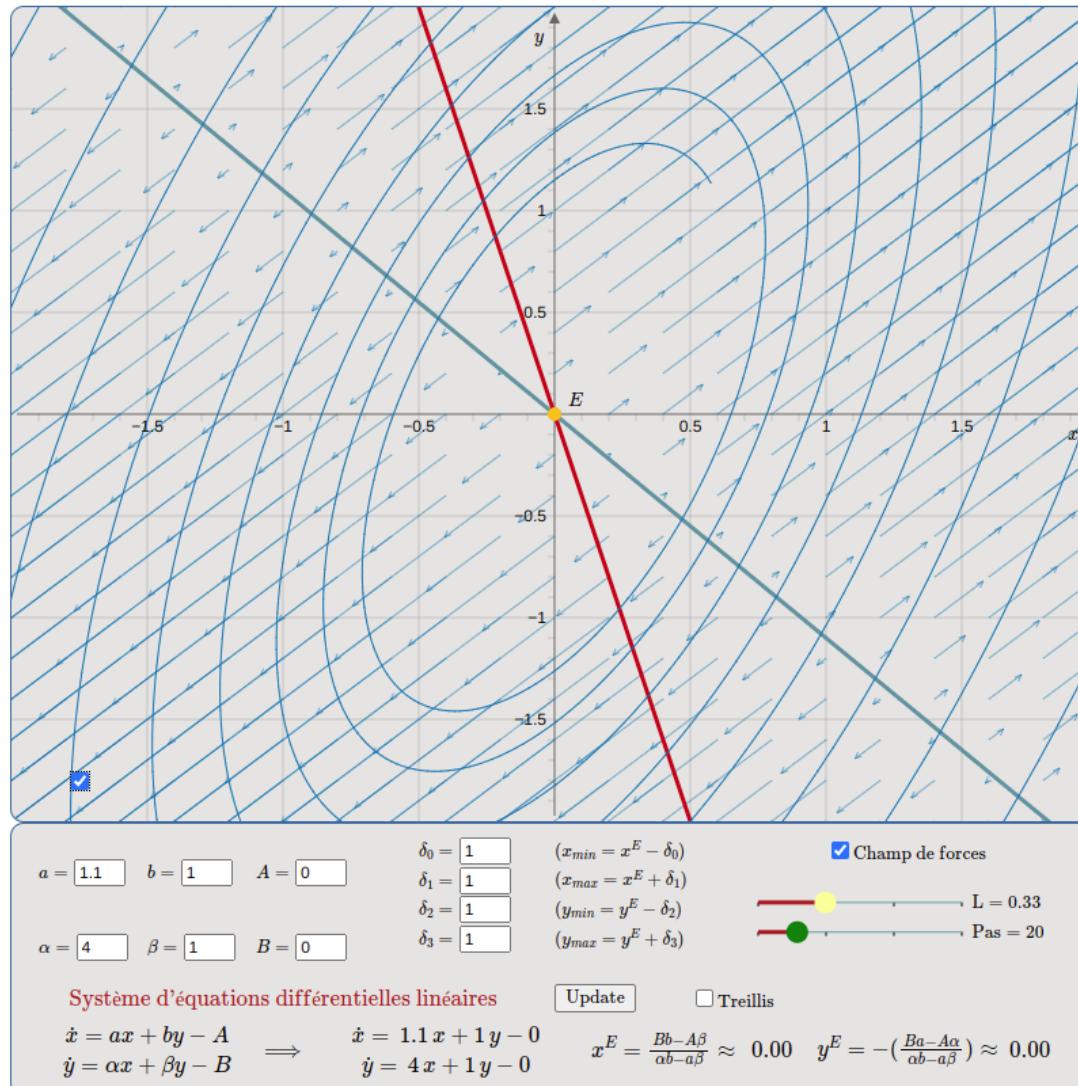
A vous de jouer

Monodot

Zone résultat

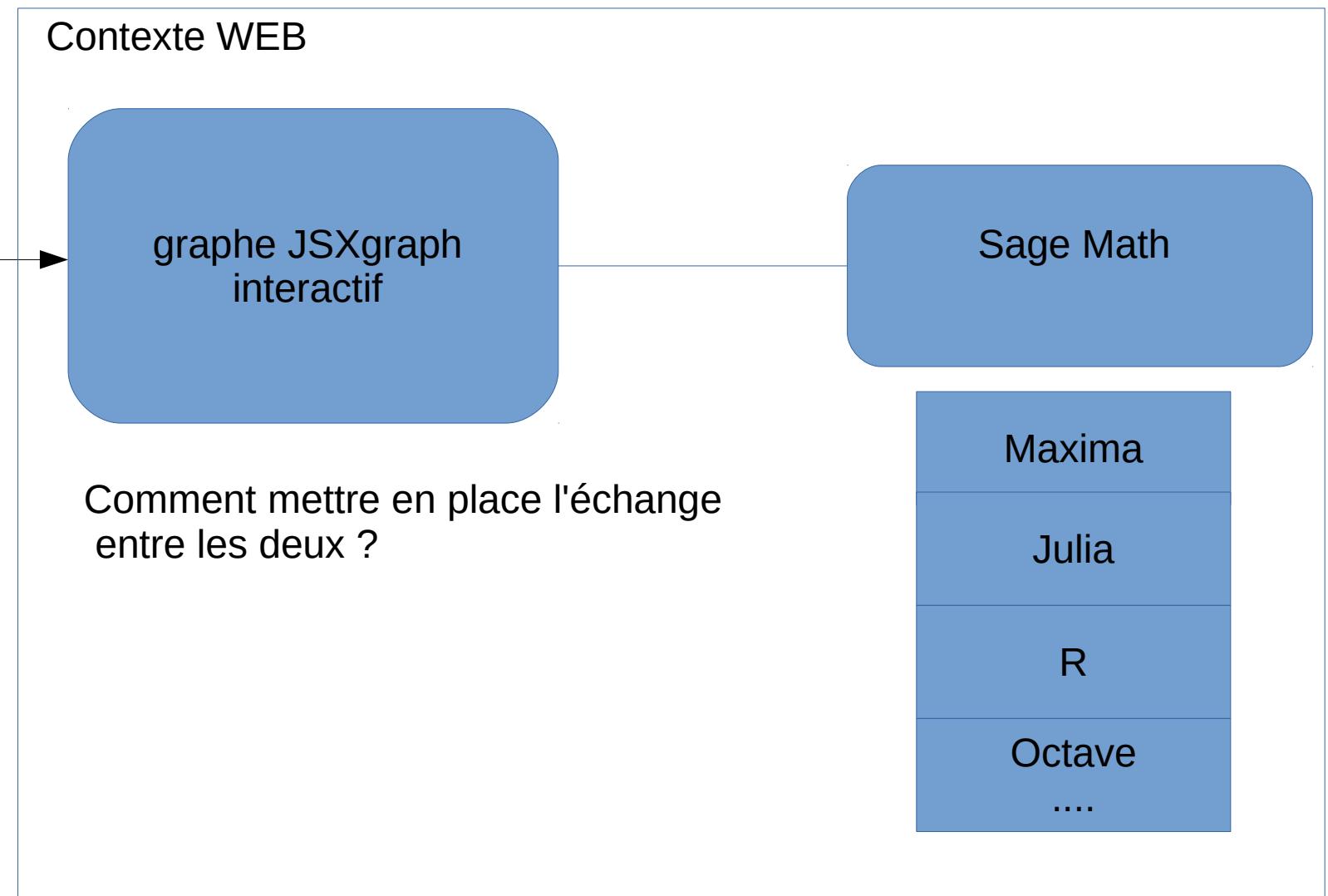
<https://jed.educandgames.org/cdsMonodot/jeu.html>

syst equa diff premier degré



https://spgoo.educandgames.org/?page_id=423

Liens



Mécanique de Sage

- Différents besoins :
 - Librairie jquery
 - Librairie embedded_sagecell.js
- cellule input
- cellule output

On construit en utilisant l'instruction de sagecell.makeSagecell la cellule input en lui rajoutant un bouton de lancement et on complète la cellule output en lui rajoutant une structure de données sagecell_session

```
> sagecell
<- {modes: {...}, kernels: Array(0), templates: {...}, al
  lLanguages: Array(10), makeSagecell: f, ...} i
  ▶ allLanguages: (10) ['sage', 'gap', 'gp', 'html',
  ▶ deleteSagecell: f (e)
  ▶ kernels: []
  ▶ makeSagecell: f (e)
  ▶ makeSinglecell: f (e)
  ▶ modes: {sage: 'python', python: 'python', html: '
  ▶ moveInputForm: f (e)
  quietMode: false
  ▶ restoreInputForm: f (e)
  ▶ templates: {minimal: {...}, restricted: {...}}
  ▶ _initPromise: {promise: Promise, state: 'fulfilled'}
  ▶ _makeSagecell: f (e)
  ▶ [[Prototype]]: Object
```

→

```
1 var('X Y t')
2 a0=1.1;b0=1;A0=0;a1=4;b1=1;A1=0;
3 X = function('X')(t)
4 Y = function('Y')(t)
5 de1=(diff(X,t)-a0*X+b0*Y +A0 == 0)
6 de2=(diff(Y,t)-a1*X+b1*Y +A1 == 0)
7 m=0.4;n=0.4
8 sol=desolve_system([de1,de2],[X,Y],[0,m,n],t)
9 print(sol)
```

Evaluate

Exemple de mise en oeuvre

Il faut paramétriser la construction de la cellule sage de façon automatique et transparente, pour ce faire il faut introduire au niveau du document html un script et l'ajouter dans body.

```
const script = document.createElement('script');
script.src ='https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.1/jquery.min.js';
script.async = true;
script.onload = () => {
    sagecell.makeSagecell({
        editor: "codemirror_READONLY",autoeval: true,
        "inputLocation": ".sage2","outputLocation": ".sageRes_2",
        linked: true});
};
script.onerror = () => {
    console.log('Error occurred while loading script');
};
document.body.appendChild(script);
```

Exécution du script de fabrication de la cellule Cell et lancement de la résolution de façon automatique.

Comment se passe l'appel ?

- Et surtout vers où ?
 - https://sagecell.sagemath.org/kernel/session_ID
- Premier pas :
 - Démarrage du kernel --> kernel.js
 - Ouverture d'une connexion sur le service distant
 - Création d'une session
 - Passage des paramètres de calculs --> message
 - Récupération des résultats --> message
 - Fermeture connexion

—

et du côté serveur

- Procédure d'installation d'une instance serveur
 - <https://github.com/sagemath/sagecell>
- Démonstration locale