The Sage package comb_walks

for Walks in the Quarter Plane

Alin Bostan   Frédéric Chyzak
Antonio Jiménez-Pastor   Pierre Lairez

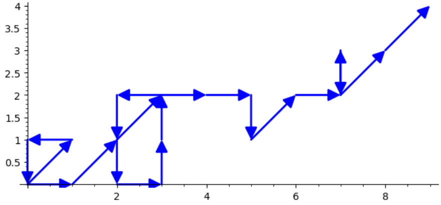
Der Wissenschaftsfonds.
Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

ISSAC (July 2020)

Outline

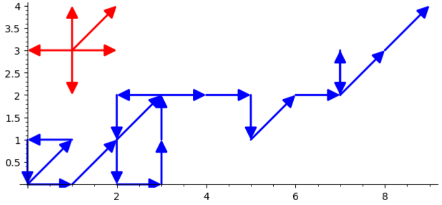
Walks in the Quarter Plane

Walks in the Quarter Plane



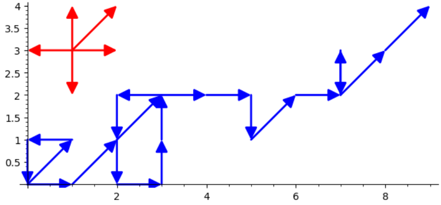
- Never cross the X and Y axis.

Walks in the Quarter Plane



- Never cross the X and Y axis.
- Steps taken from a fixed set of valid steps.

Walks in the Quarter Plane



- Never cross the X and Y axis.
- Steps taken from a fixed set of valid steps.
- Length of the walk = Number of steps

Combinatorial problem

A counting problem

- Given a set of valid steps $\mathcal{S}$, how many walks are there that start from $(0, 0)$ and end at (i, j) after n steps taken from $\mathcal{S}$?
- We call that number $q_{i,j,n}$.

Combinatorial problem

A counting problem

- Given a set of valid steps $\mathcal{S}$, how many walks are there that start from $(0, 0)$ and end at (i, j) after n steps taken from $\mathcal{S}$?
- We call that number $q_{i,j,n}$.

$$Q(x, y, t) = \sum_{i,j,n \in \mathbb{N}} q_{i,j,n} x^i y^j t^n.$$

A classification problem

Given a set $\mathcal{S}$, what algebraic or differential properties does $Q(x, y, t)$ have?

A classification problem

Given a set $\mathcal{S}$, what algebraic or differential properties does $Q(x, y, t)$ have?

- **Rational**: there are $N(x, y, t), D(x, y, t) \in \mathbb{Q}[x, y, t]$ such that
$$Q(x, y, t) = \frac{N(x, y, t)}{D(x, y, t)}.$$
- **Algebraic**: there is $P(Z) \in \mathbb{Q}(x, y, t)[Z]$ such that $P(Q(x, y, t)) = 0$.
- **D-finite**: $Q(x, y, t)$ satisfies linear differential equations w.r.t. ∂_x, ∂_y and ∂_t with coefficients in $\mathbb{Q}[x, y, t]$.
- **D-algebraic**: $Q(x, y, t)$ satisfies non-linear differential equations w.r.t. ∂_x, ∂_y or ∂_t with coefficients in $\mathbb{Q}[x, y, t]$.
- **D-transcendental**: any other case.

The kernel equation

Kernel polynomial

Fixed set $\mathcal{S}$ of valid steps:

$$K(x, y, t) = xy \left(1 - t \sum_{(a,b)\in\mathcal{S}} x^a y^b\right).$$

The kernel equation

Kernel polynomial

Fixed set $\mathcal{S}$ of valid steps:

$$K(x,y,t) = xy\left(1 - t\sum_{(a,b)\in\mathcal{S}} x^a y^b\right).$$

Functional equation

$$Q(x,y,t)K(x,y,t) = xy - Q(0,0,t)K(0,0,t) + Q(x,0,t)K(x,0,t) + Q(0,y,t)K(0,y,t).$$

Tools used for classification

- Sections of $Q(x,y,t)K(x,y,t)$: study

$$F_1(x,t) = Q(x,0,t)K(x,0,t),\ \ F_2(y,t) = Q(0,y,t)K(0,y,t).$$

- Algebraic geometry: study the curve in $\mathbb{P} \times \mathbb{P}$:

$$E_t = \left\{ (x_0 : x_1, y_0 : y_1) \ : \ K\left(\frac{x_0}{x_1}, \frac{y_0}{y_1}, t\right) = 0 \right\}.$$

Tools used for classification

- Sections of $Q(x,y,t)K(x,y,t)$: study

$$F_1(x,t) = Q(x,0,t)K(x,0,t), \; F_2(y,t) = Q(0,y,t)K(0,y,t).$$

- Algebraic geometry: study the curve in $\mathbb{P} \times \mathbb{P}$:

$$E_t = \left\{ (x_0 : x_1, y_0 : y_1) \; : \; K\left(\frac{x_0}{x_1}, \frac{y_0}{y_1}, t\right) = 0 \right\}.$$

1. Case of a curve of genus 0: rational parametrization.
2. Case of a curve of genus 1: elliptic curve.

Tools used for classification

- Sections of $Q(x,y,t)K(x,y,t)$: study

$$F_1(x,t) = Q(x,0,t)K(x,0,t), \; F_2(y,t) = Q(0,y,t)K(0,y,t).$$

- Algebraic geometry: study the curve in $\mathbb{P} \times \mathbb{P}$:

$$E_t = \left\{ (x_0 : x_1, y_0 : y_1) \; : \; K\left(\frac{x_0}{x_1}, \frac{y_0}{y_1}, t\right) = 0 \right\}.$$

1. Case of a curve of genus 0: rational parametrization.
2. **Case of a curve of genus 1: elliptic curve.**

How to get and use comb_walks

Git repository

Code hosted in GitLab Inria

https://gitlab.inria.fr/discretewalks/comb_walks

Git repository

Code hosted in GitLab Inria

`https://gitlab.inria.fr/discretewalks/comb_walks`

Installation via *pip*

Recommended to install all the dependencies:

```
sage -pip [--user] install git+https://gitlab.inria.fr/discretewalks/comb_walks.git
```

Documentation and Demo

Documentation available online

`https://discretewalks.gitlabpages.inria.fr/comb_walks/docs/`

Documentation and Demo

Documentation available online

https://discretewalks.gitlabpages.inria.fr/comb_walks/docs/

Easy accessible demo using Binder

Documentation and Demo

jupyter comb_walks_demo (unsaved changes)

Join this repo's Video Chat | Visit repo | Copy Binder lin

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | SageMath 9.0

Run | Markdown

Walks in the Quarter Plane in Sage

The package comb_walks

1. Introduction to walks in the plane

A *walk* in the lattice $\mathbb{Z}^2$ is a sequence of points $(P_0, \ldots, P_n)$. We say that P_0 is the *origin* of the walk and P_n is its *ending point*, that n is its length and that all the differences $s_i = P_i - P_{i-1} \in \mathbb{Z}^2$ for $1 \leq n$ are its *steps*.

Given a set $S \subset \mathbb{Z}^2$ whose elements we call *valid steps*, we can consider walks in the lattice plane that only have steps in such subset, i.e., walks where $s_i \in S$ for all $1 \leq i \leq n$. We call the resulting set of walks a *model* of walks.

The following combinatorial problem was considered:

- **Walks on the plane:** given a set $S \subset \{-1, 0, 1\}^2 \setminus \{(0,0)\}$ (which are called *small steps*), how many walks are there from the origin $(0,0)$ to the point (i,j) using n steps taken from S?

To study this problem, we collect the numbers $g_{i,j,n}$ in a generating series:

$$G(x,y,t) = \sum_{i,j\in\mathbb{Z}} \sum_{n\in\mathbb{N}} g_{i,j,n} x^i y^j t^n.$$

The classification of different models according to the algebraic and differential nature of the function $G(x,y,t)$ is done by using the recurrence relation

$$g_{i,j,n} = \sum_{(a,b)\in S} g_{(i-a),(j-b),(n-1)},$$

which holds for all different models. Moreover, for each $(a,b) \in \mathbb{Z}^2$ we have

$$\begin{aligned} tx^a y^b G(x,y,t) &= \textstyle\sum_{i,j\in\mathbb{Z}} \sum_{n\in\mathbb{N}} g_{i,j,n} x^{i+a} y^{j+b} t^{n+1} \\ &= \textstyle\sum_{i,j\in\mathbb{Z}} \sum_{n\geq 1} g_{(i-a),(j-b),(n+1)} x^i y^j t^n. \end{aligned}$$

Now, if we sum the left-hand sides of the previous equation over all $(a,b) \in S$, we can simplify the right-hand side using the recurrence equation. This leads to the following equation:

$$G(x,y,t) = \frac{1}{1 - tS(x,y)},$$

where $S(x,y)$ is the so-called *step polynomial* of the model S and it is defined as follows:

What can we do with comb_walks

Main user data structure

WalkModel

- Each instance of the class represents a model of walks.
- The constructor inputs the set of valid steps.

```
In [2]: WalkModel((1,0),(0,1),(-1,-1))

Out[2]: Walk Model with steps: ((1, 0), 1), ((0, 1), 1), ((-1, -1), 1)
```

Main user data structure

WalkModel

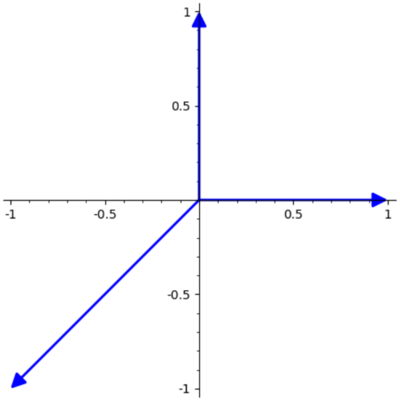
- Each instance of the class represents a model of walks.
- The constructor inputs the set of valid steps.
- Possible to use letters for the steps: N, S, E, W, etc.

```
In [3]: WalkModel(N,E,SW)

Out[3]: Walk Model with steps: ((0, 1), 1), ((1, 0), 1), ((-1, -1), 1)
```

Plotting methods

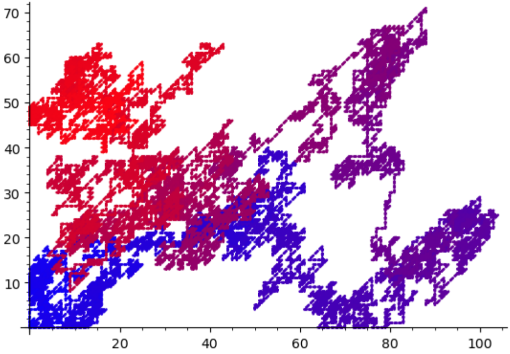
```
In [6]: plot(WalkModel(N,E,SW))
Out[6]:
```



Plotting methods

In [12]:
```
# A long example of the model
m.plot_random_walk(10000, arrowsize=0.001)
```

Out[12]:



Methods for geometric objects

Kernel function

The function K in $\mathbb{P}^2$:

```
In [13]: WalkModel(N,NE,W,SW).kernel()
```

Out[13]: $(-t)\,x^2y^2 + (-t)\,xy^2z + xyz^2 + (-t)\,yz^3 + (-t)\,z^4$

Methods for geometric objects

Kernel function

The function K in $\mathbb{P} \times \mathbb{P}$:

```
In [14]: WalkModel(N,NE,W,SW).kernel('P')
```

Out[14]: $(-t)\, x_0^2 y_0^2 + (-t)\, x_0 x_1 y_0^2 + x_0 x_1 y_0 y_1 + (-t)\, x_1^2 y_0 y_1 + (-t)\, x_1^2 y_1^2$

Methods for geometric objects

Kernel function

The function K in $\mathbb{P} \times \mathbb{P}$:

The automorphism τ

Map that changes first the y coordinate and then the x coordinate:

```
In [20]: WalkModel(N,NE,W,SW).tau('P')

Out[20]: Scheme endomorphism of Closed subscheme of Product of projective spaces P^1 x P^1 over Fraction Field of Univariate Polynomial
         Ring in t over Rational Field defined by:
           (-t)*x0^2*y0^2 + (-t)*x0*x1*y0^2 + x0*x1*y0*y1 + (-t)*x1^2*y0*y1 + (-t)*x1^2*y1^2
           Defn: Defined by sending (x0 : x1 , y0 : y1) to
                 (-x0^2*y0 - x0*x1*y0 + t*x0*x1*y1 + t*x1^2*y1 : (-t)*x1^2*y1 , -x1^2*y1 : -x0^2*y0 - x0*x1*y0).
```

Methods for geometric objects

The function $b_i := \tau(F_i) - F_i$

```
In [23]: WalkModel(N,NE,W,SW).b(1)
```

Out[23]:

$$\frac{\left(\frac{-1}{t}\right) x^2 y + \left(\frac{-1}{t}\right) xy + 2x + 1}{-x^2 y - xy}$$

```
In [24]: WalkModel(N,NE,W,SW).b(2)
```

Out[24]:

$$\frac{x^2 y^2 + xy^2 - 1}{-xy - y}$$

Telescoping functions over the curve

Poles of rational functions on the kernel curve

```
In [28]: model = WalkModel(NE,S,W)
         model.poles(model.b(2))

Out[28]: [(0 : 1 , 1 : 0), (1 : 0 , 0 : 1), (0 : 1 , 0 : 1)]
```

Telescoping functions over the curve

Poles of rational functions on the kernel curve

```
In [28]: model = WalkModel(NE,S,W)
         model.poles(model.b(2))

Out[28]: [(0 : 1 , 1 : 0), (1 : 0 , 0 : 1), (0 : 1 , 0 : 1)]
```

Telescoping functions over the curve

Compute L and g such that $L \cdot b = \tau(g) - g$:

```
In [31]: L,g = model.telescoping(model.b(2)(x=x0/x1,y=y0/y1))
         print(L) # It is (0,1)
         tau = pullback(model.tau('P'))
         # Checking the equation on the curve
         simplify_rational_variety(model.derivative(model.b(2))(x=x0/x1,y=y0/y1) - (tau(g)-g), model.curve('P')) == 0

         0 1

Out[31]: True
```

Telescoping functions over the curve

Poles of rational functions on the kernel curve

```
In [28]: model = WalkModel(NE,S,W)
         model.poles(model.b(2))

Out[28]: [(0 : 1 , 1 : 0), (1 : 0 , 0 : 1), (0 : 1 , 0 : 1)]
```

Telescoping functions over the curve

Compute L and g such that $L \cdot b = \tau(g) - g$. In this case:

$$\delta b_2 = (\tau - 1)\left(t \frac{x_0^2 y_0 + x_1^2 y_1}{x_0 x_1 y_0} \right).$$

Built-in models

- `FiniteGroup`: all models where $ord(\tau) < \infty$.
- `EllipticC`: models with infinite group and elliptic kernel.
- `NonEllipticC`: other models.

Built-in models

- `FiniteGroup`: all models where $ord(\tau) < \infty$.
- `EllipticC`: models with infinite group and elliptic kernel.
- `NonEllipticC`: other models.

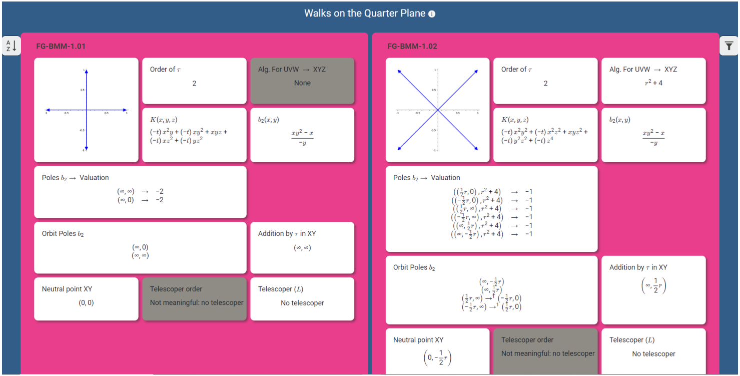
Exhaustive website

The functionality of the package has been applied to all those models. The results can be found in the webpage

https://discretewalks.gitlabpages.inria.fr/comb_walks/

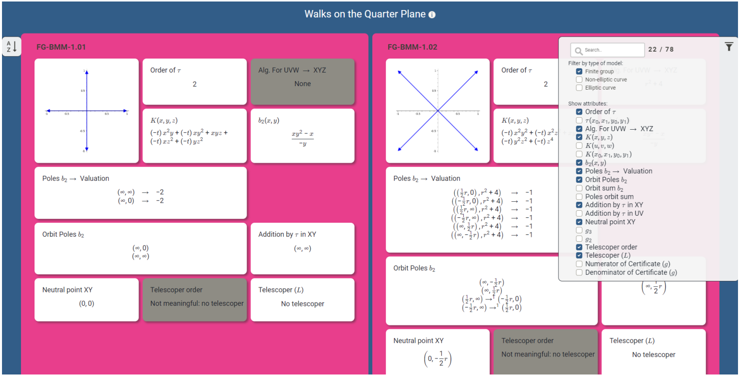
The website

https://discretewalks.gitlabpages.inria.fr/comb_walks/



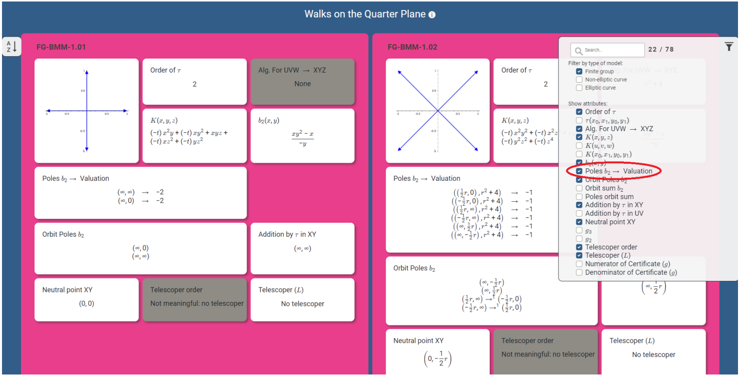
The website

https://discretewalks.gitlabpages.inria.fr/comb_walks/



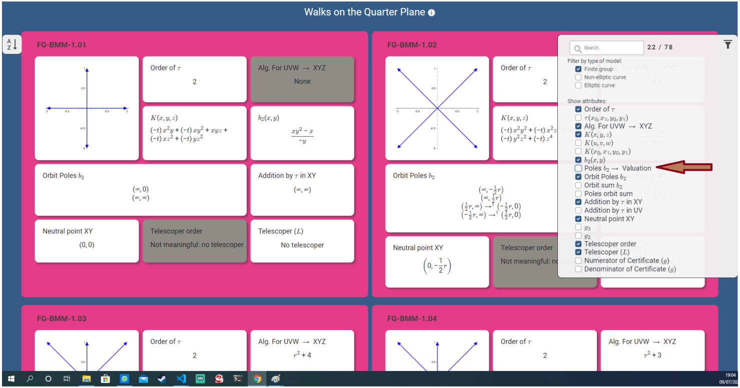
The website

https://discretewalks.gitlabpages.inria.fr/comb_walks/



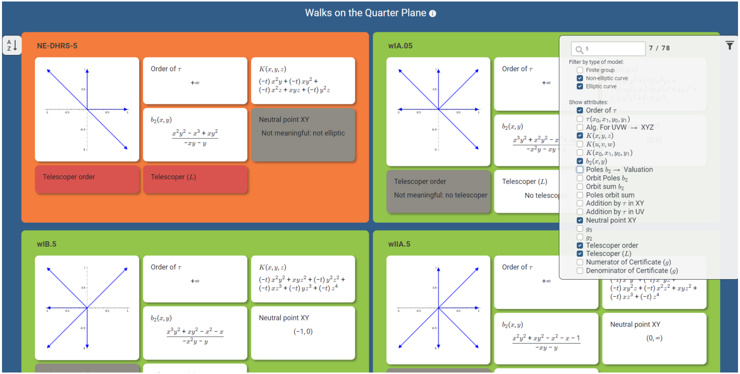
The website

https://discretewalks.gitlabpages.inria.fr/comb_walks/



The website

https://discretewalks.gitlabpages.inria.fr/comb_walks/



Conclusions

Conclusions

Achievements

- A framework to classify models of walks in the quarter plane
- Telescoping method over elliptic kernel functions
- Visualization of the results over classical examples

Future work

- Develop a constructive method for obtaining algebraic and differential equations
- Study the performance and complexity of the current code
- Adapt the current implementation to wider use

Thank you!

Results and documentation:

- https://discretewalks.gitlabpages.inria.fr/comb_walks/
- https://discretewalks.gitlabpages.inria.fr/comb_walks/docs/

GitLab repository:

- https://gitlab.inria.fr/discretewalks/comb_walks