Python for Mathematical Visualization

A Four-Dimensional Case Study

David Dumas
University of Illinois at Chicago

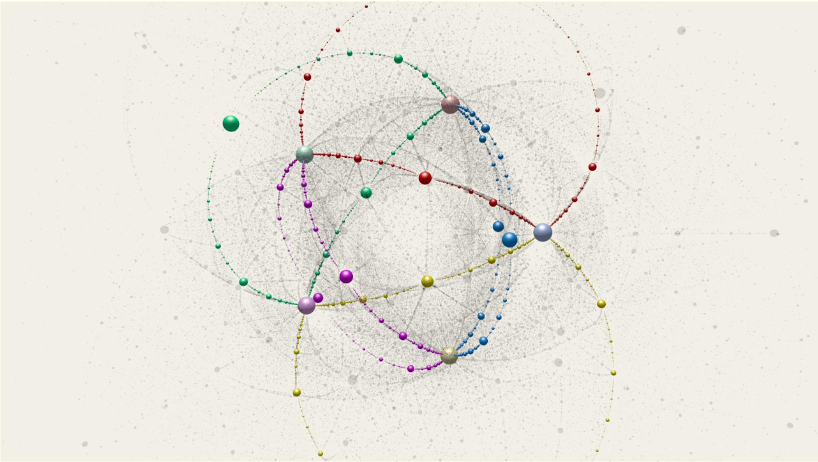
May 20, 2017

Goal

Describe a mathematical visualization project that uses Python.

Highlight characteristics of Python that make it suitable for this application.

The PML Visualization Project



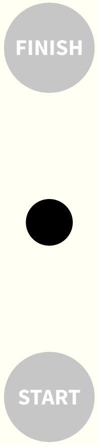
Joint work with François Guéritaud

http://dumas.io/PML

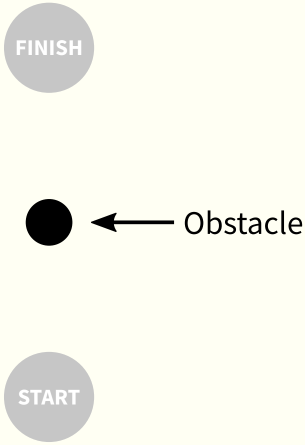
Steps

- Enumerate the objects
- Calculate coordinates
- Render an image

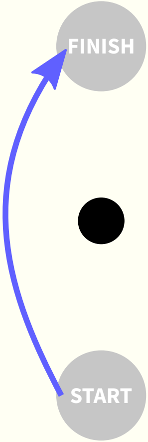
Topological enumeration



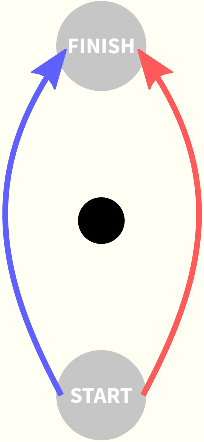
Topological enumeration



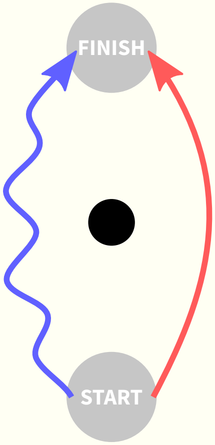
Topological enumeration



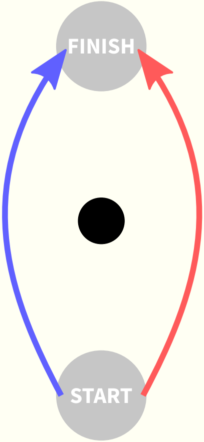
Topological enumeration



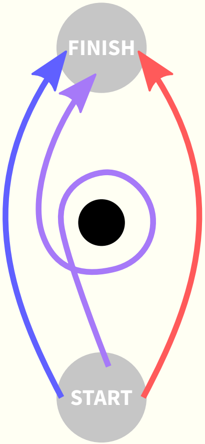
Topological enumeration



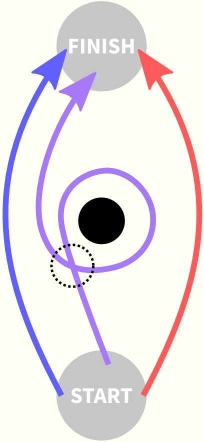
Topological enumeration



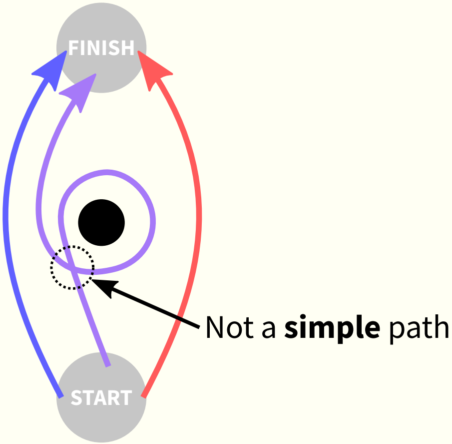
Topological enumeration



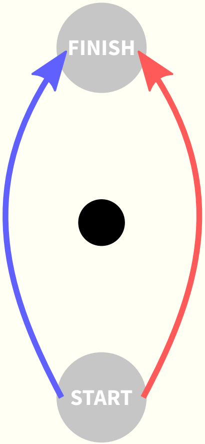
Topological enumeration



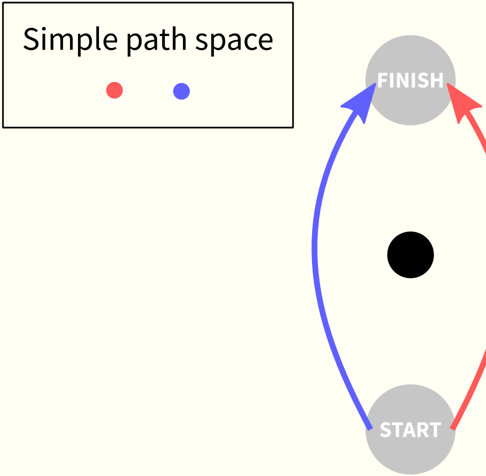
Topological enumeration



Topological enumeration

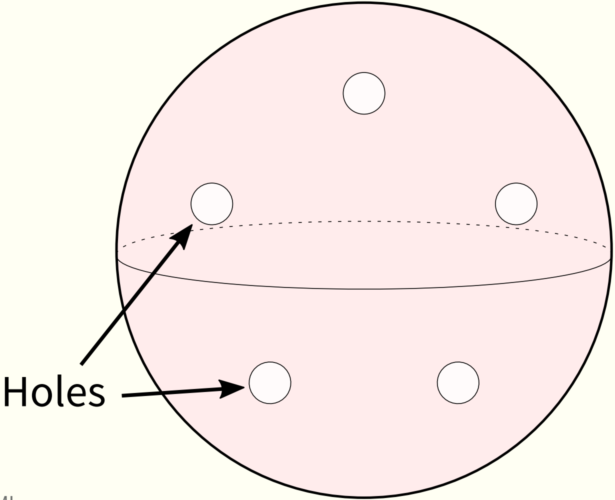


Topological enumeration



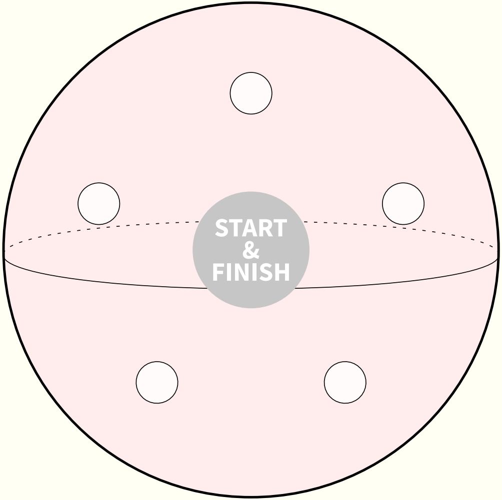
Topological enumeration

Simple loops on the five-holed sphere



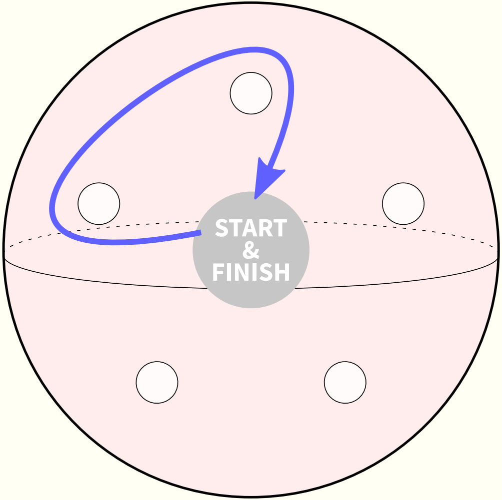
Topological enumeration

Simple loops on the five-holed sphere



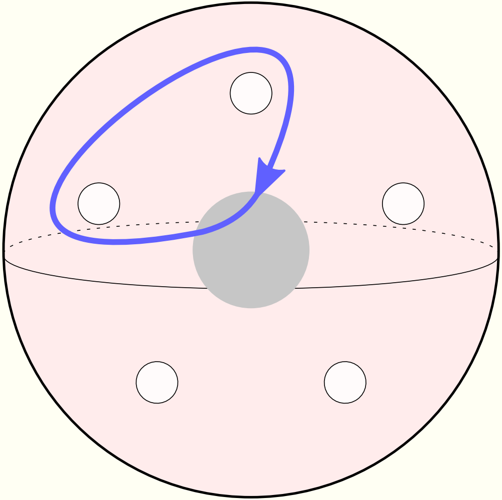
Topological enumeration

Simple loops on the five-holed sphere



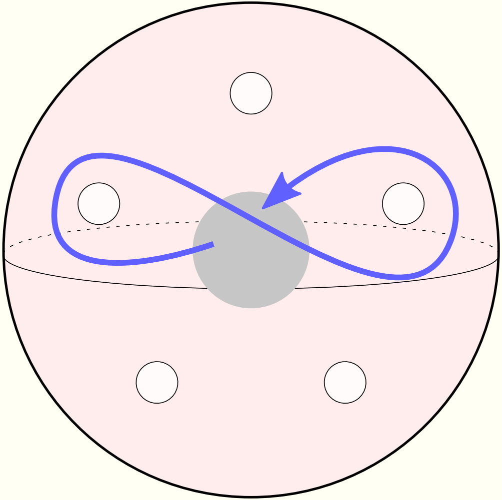
Topological enumeration

Simple loops on the five-holed sphere



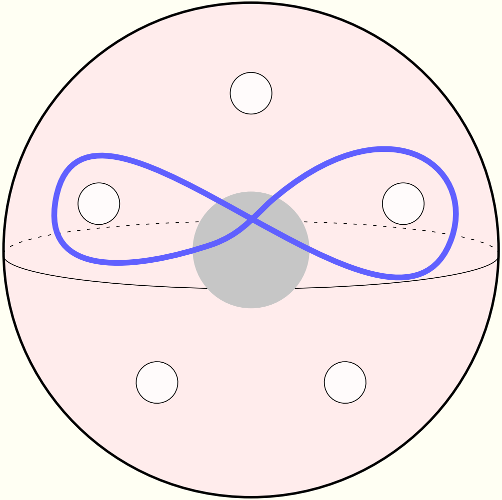
Topological enumeration

Simple loops on the five-holed sphere



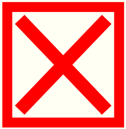
Topological enumeration

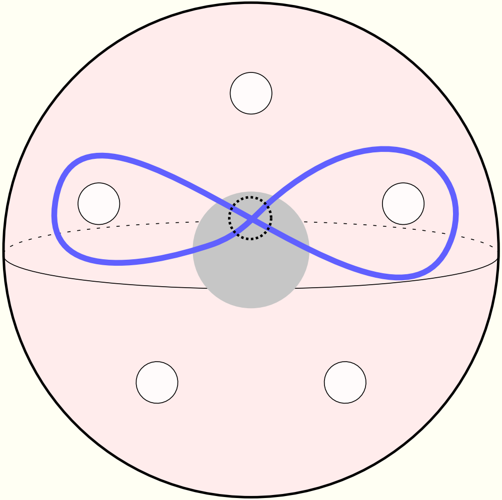
Simple loops on the five-holed sphere



Topological enumeration

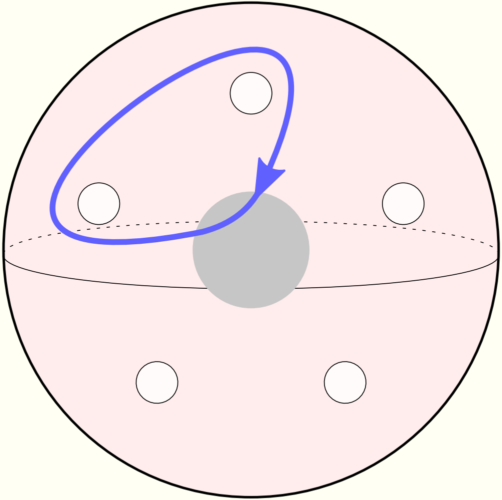
Simple loops on the five-holed sphere





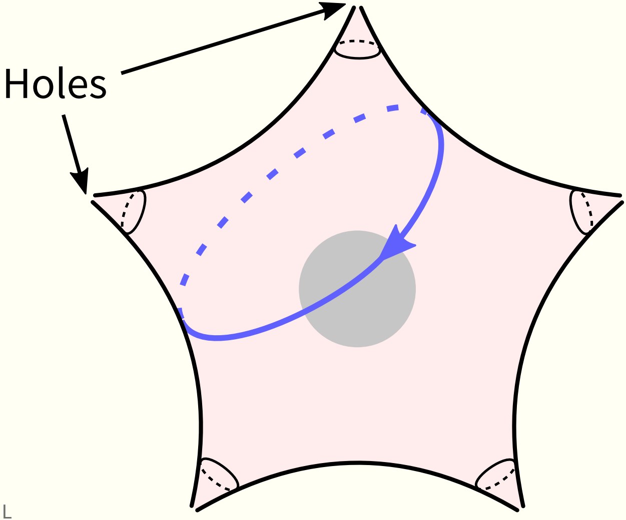
Topological enumeration

Simple loops on the five-holed sphere



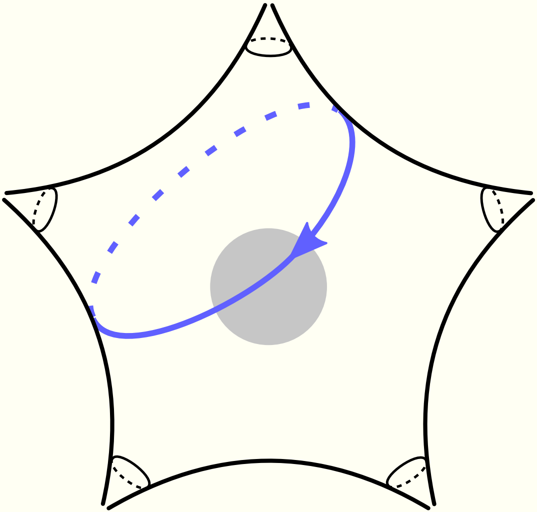
Topological enumeration

Simple loops on the five-holed sphere



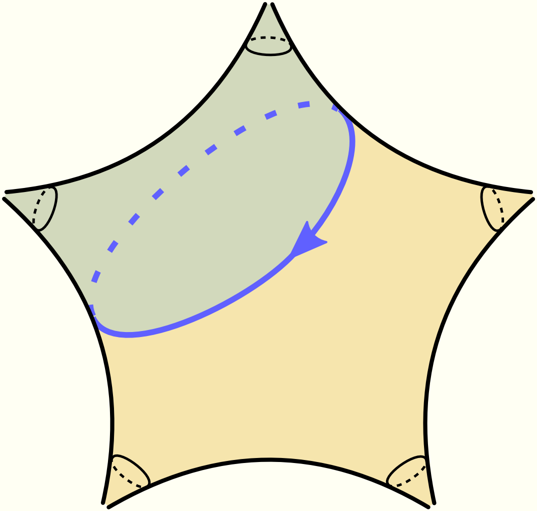
Topological enumeration

Simple loops on the five-holed sphere



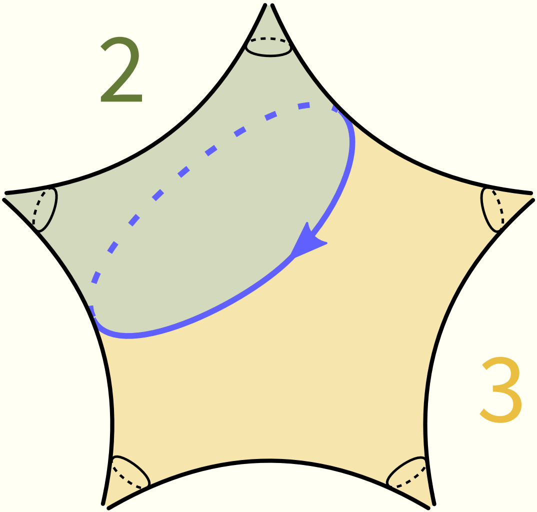
Topological enumeration

Simple loops on the five-holed sphere



Topological enumeration

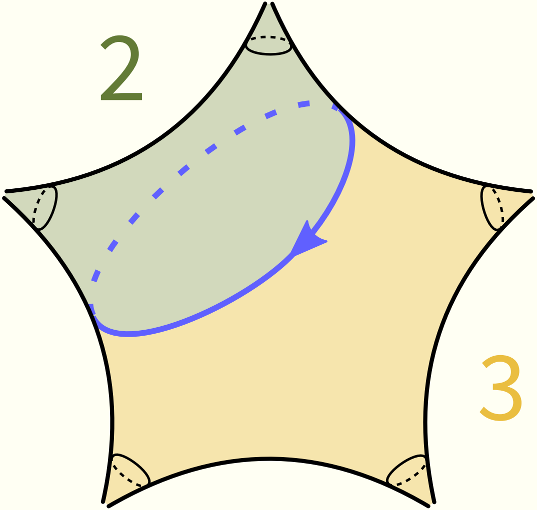
Simple loops on the five-holed sphere



dumas.io/PML

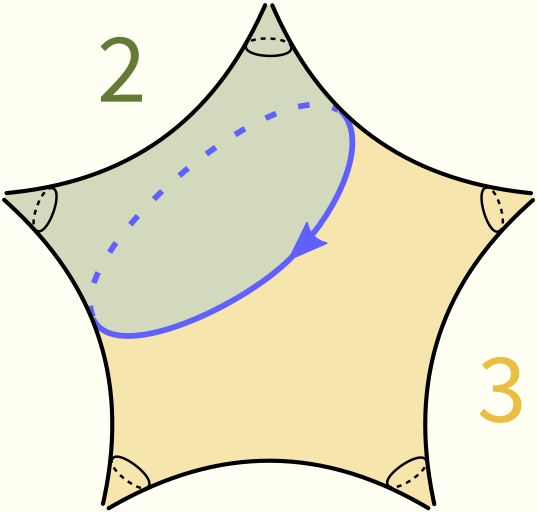
Topological enumeration

Nonperipheral simple closed curves on the five-holed sphere



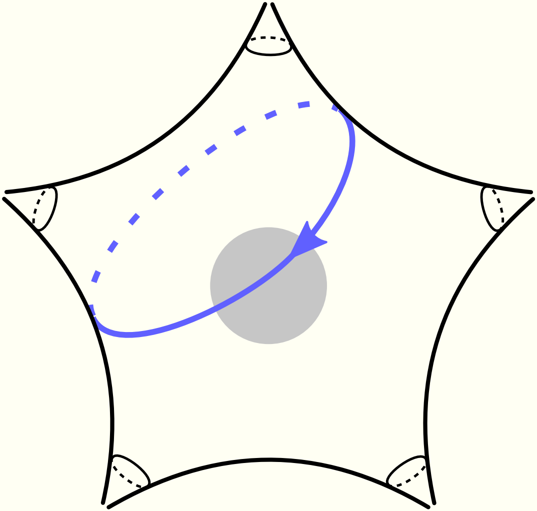
Topological enumeration

Nonperipheral simple closed curves on the five-holed sphere
(NSCC)



Topological enumeration

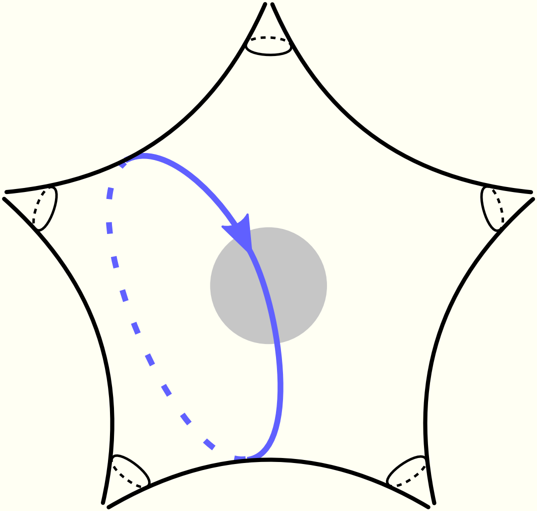
Nonperipheral simple closed curves on the five-holed sphere
(NSCC)



Topological enumeration

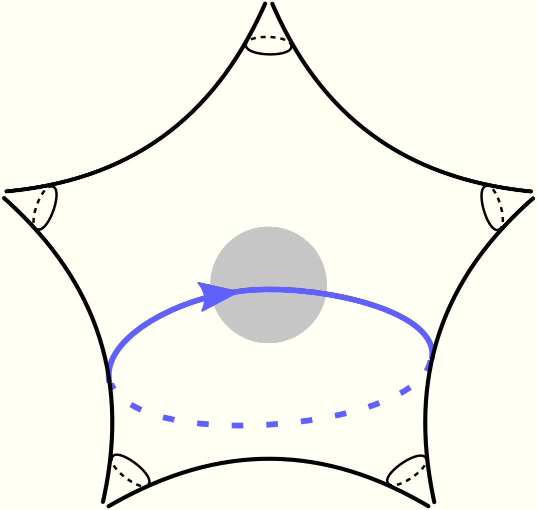
Nonperipheral simple closed curves on the five-holed sphere

(NSCC)



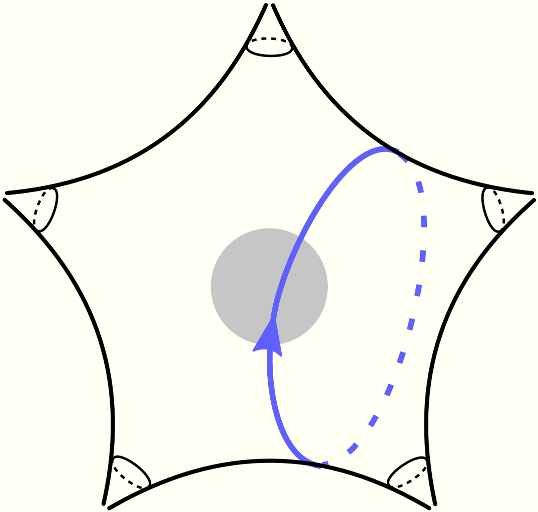
Topological enumeration

Nonperipheral simple closed curves on the five-holed sphere
(NSCC)



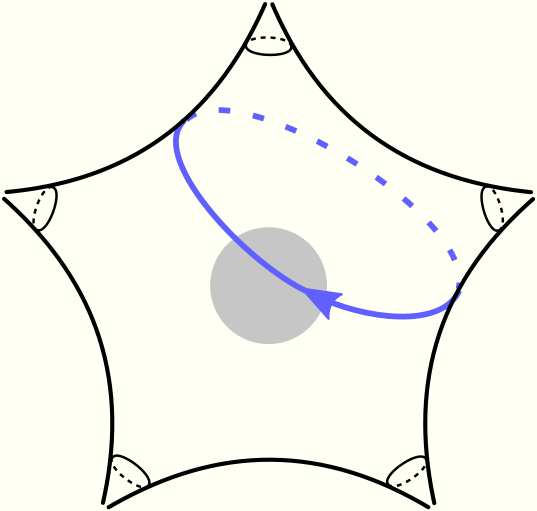
Topological enumeration

Nonperipheral simple closed curves on the five-holed sphere
(NSCC)



Topological enumeration

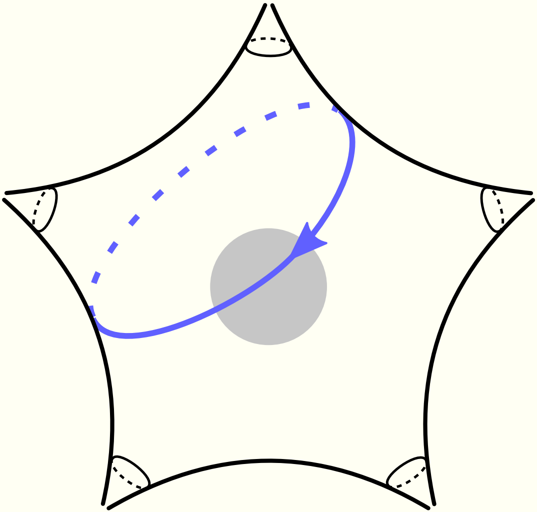
Nonperipheral simple closed curves on the five-holed sphere
(NSCC)



Topological enumeration

Nonperipheral simple closed curves on the five-holed sphere

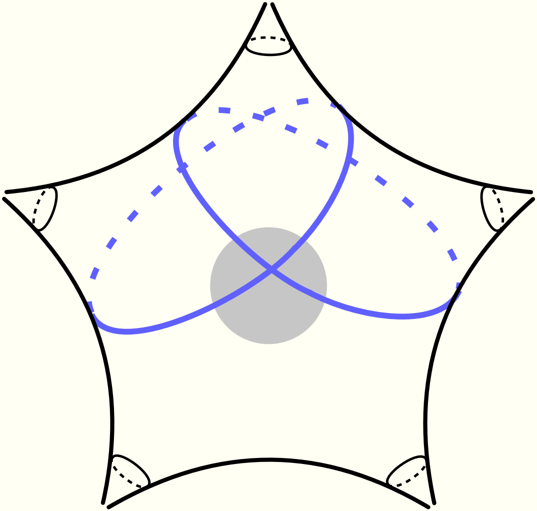
(NSCC)



Topological enumeration

Nonperipheral simple closed curves on the five-holed sphere

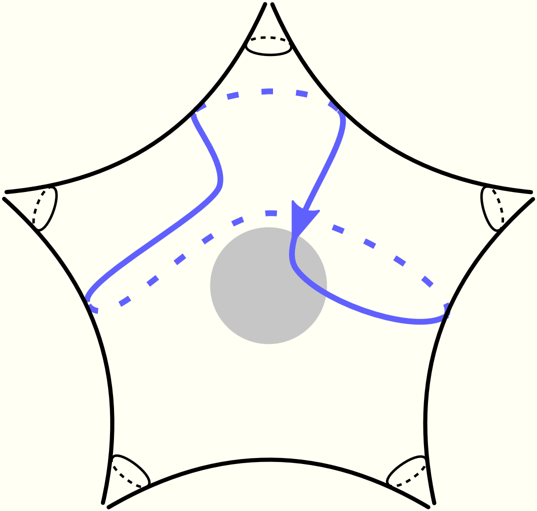
(NSCC)



Topological enumeration

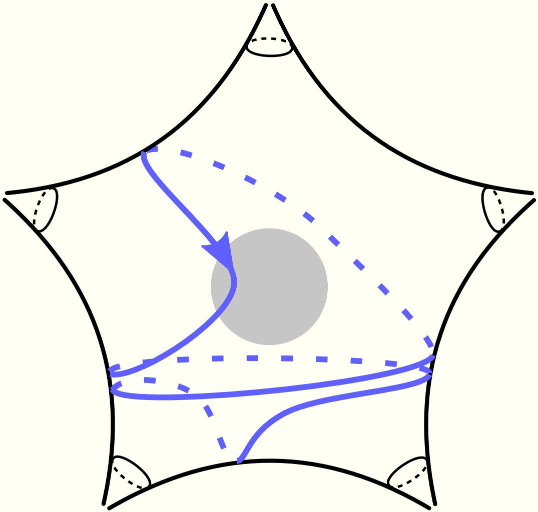
Nonperipheral simple closed curves on the five-holed sphere

(NSCC)

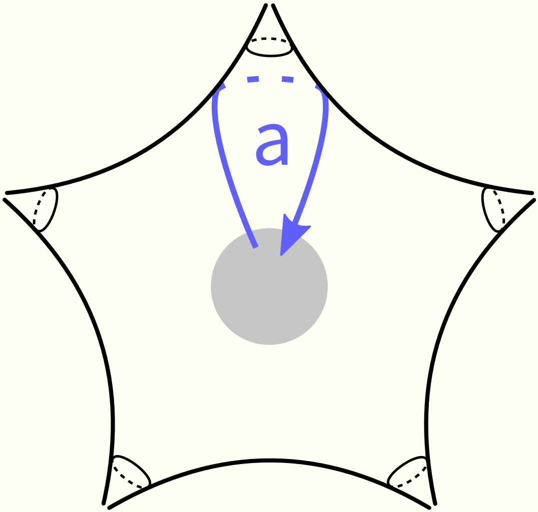


Topological enumeration

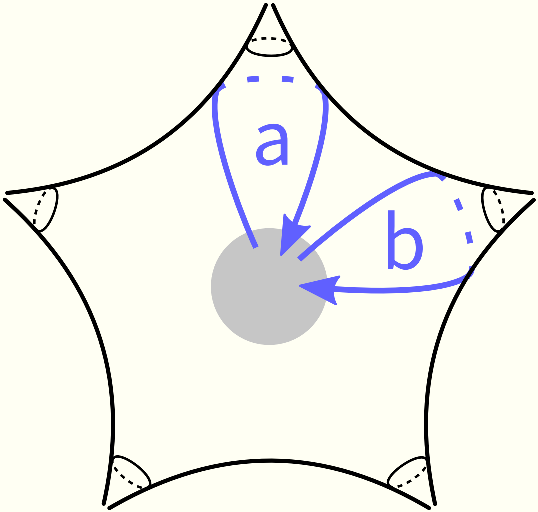
Nonperipheral simple closed curves on the five-holed sphere
(NSCC)



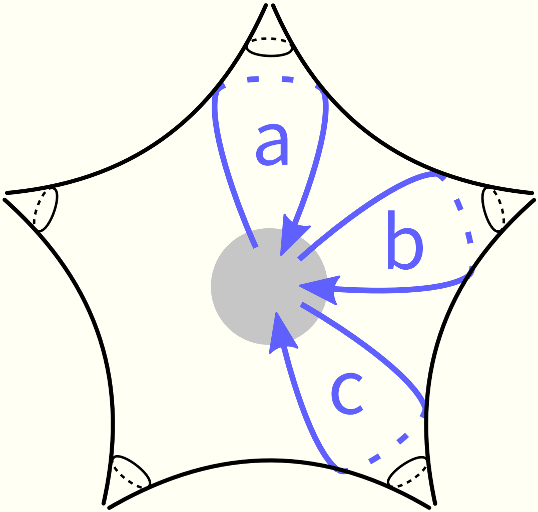
Generators



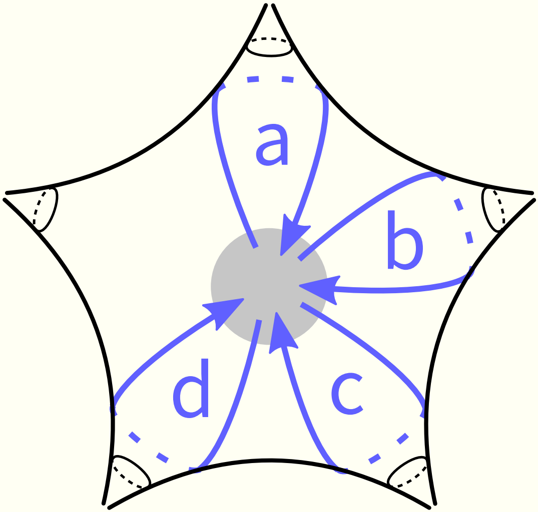
Generators



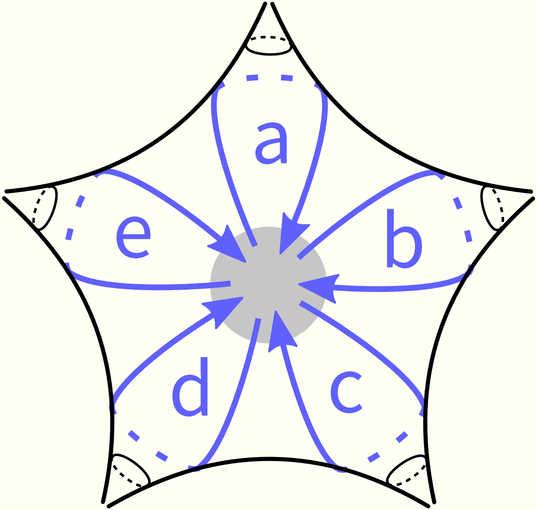
Generators



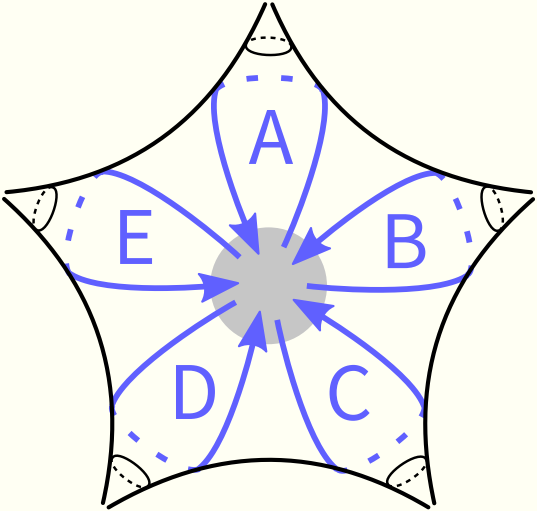
Generators



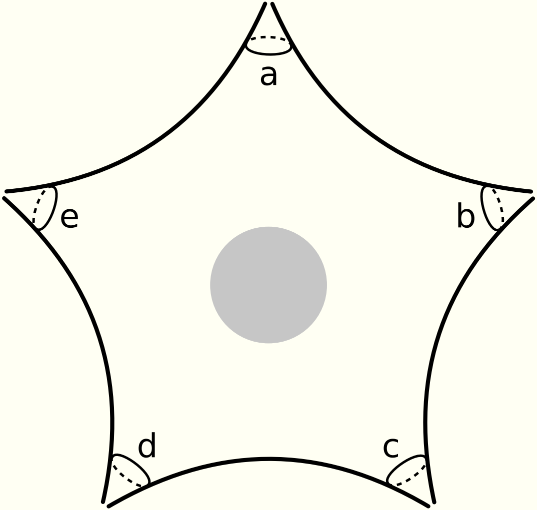
Generators



Generators

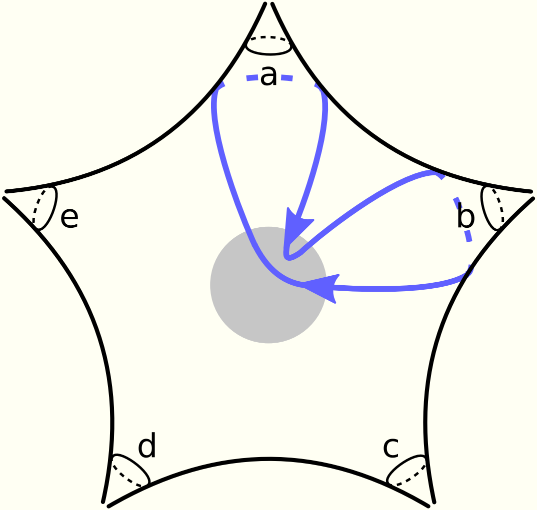


Generators



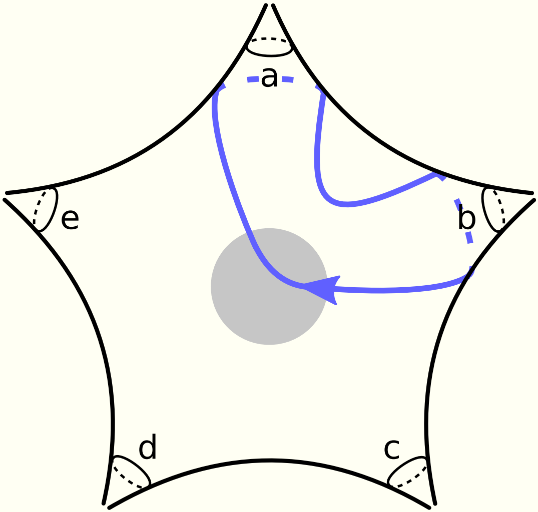
Encoding curves in strings

ab



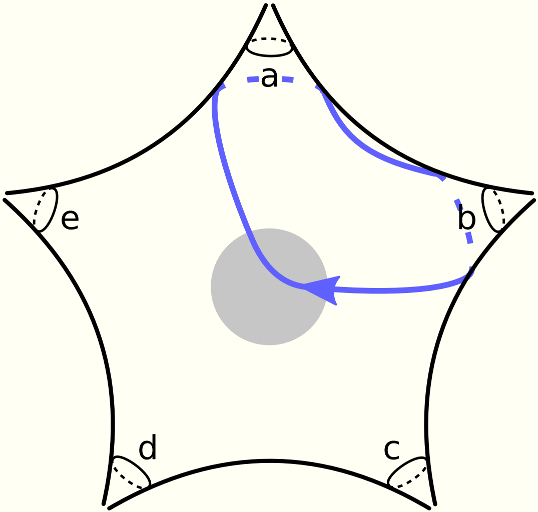
Encoding curves in strings

ab



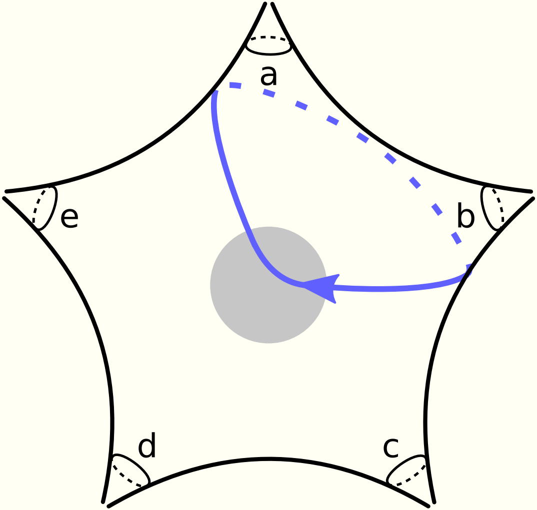
Encoding curves in strings

ab



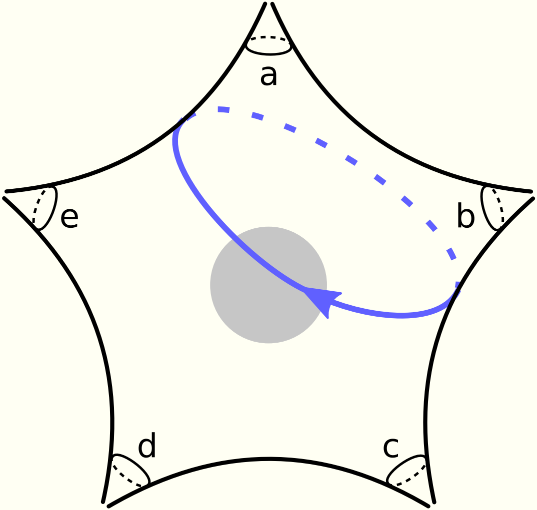
Encoding curves in strings

ab



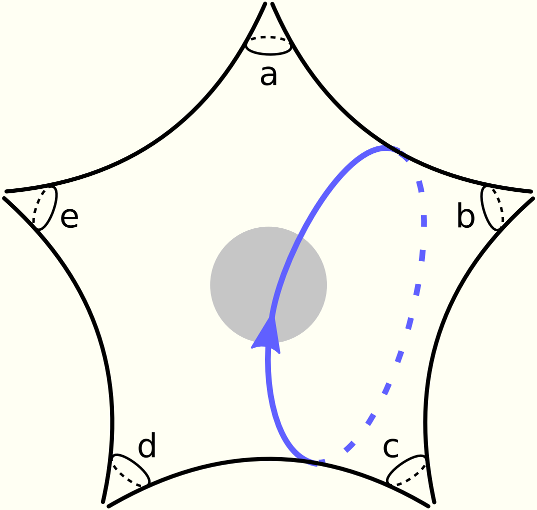
Encoding curves in strings

ab



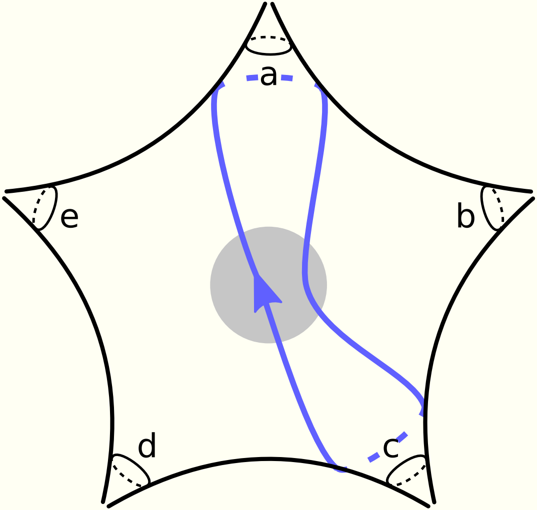
Encoding curves in strings

bc

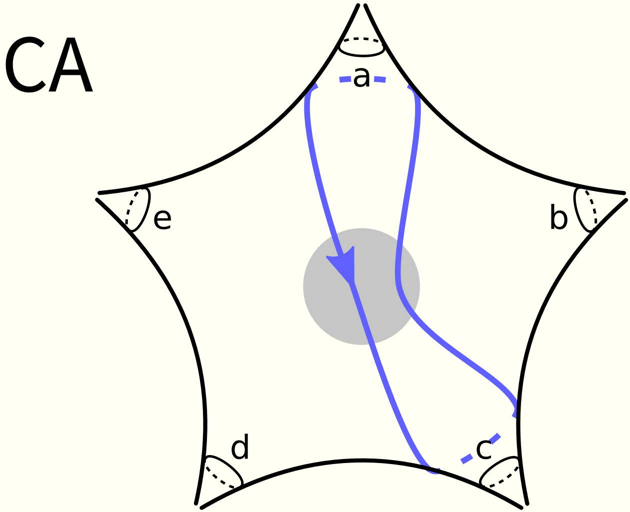


Encoding curves in strings

ac

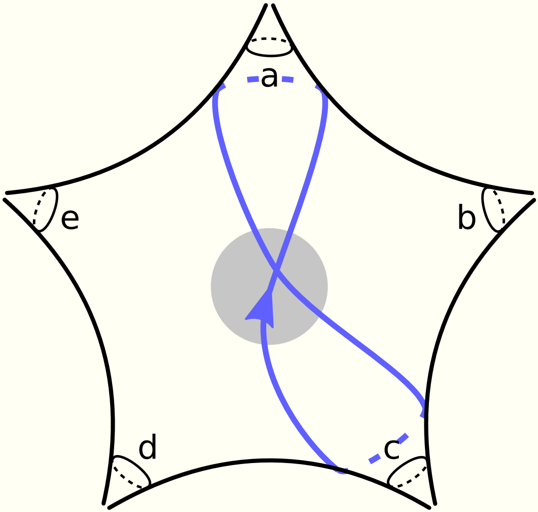


Encoding curves in strings



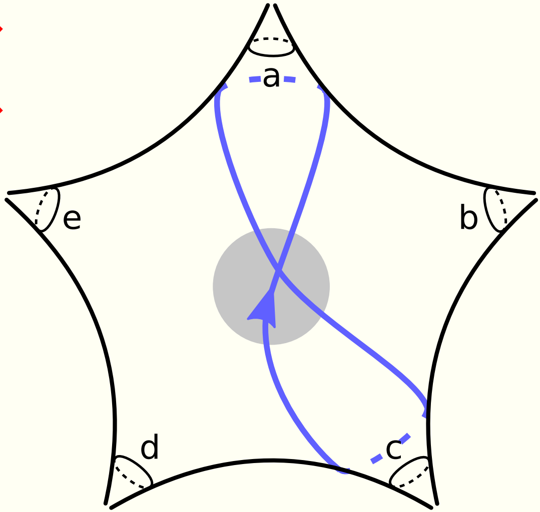
Encoding curves in strings

aC



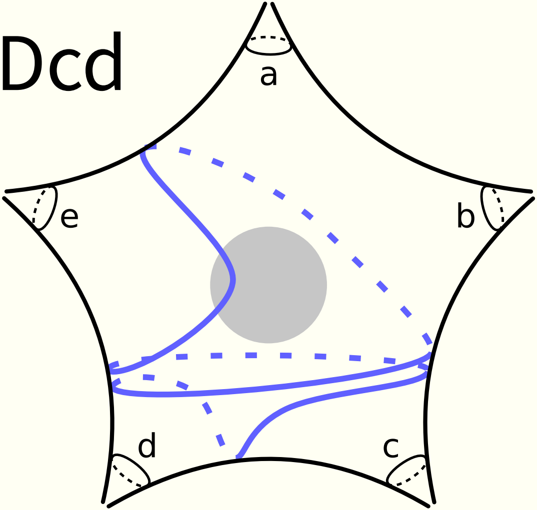
Encoding curves in strings

~~ac~~



Encoding curves in strings

abDcd



Naïve generator

All words in alphabet `'abcdeABCDE'`.

```
from itertools import product

N = 5  # Word length

for w in product('abcdeABCDE',repeat=N):
    print(''.join(w))
```

Naïve generator

Simple words in alphabet `'abcdeABCDE'`.

```
from itertools import product

N = 5  # Word length

for w in product('abcdeABCDE',repeat=N):
    if is_simple_curve(w):
        print(''.join(w))
```

Naïve generator

Simple words in alphabet `'abcdeABCDE'`.

```
from itertools import product

N = 5  # Word length

for w in product('abcdeABCDE',repeat=N):
    if is_simple_curve(w):
        print(''.join(w))

# TODO: Implement is_simple_curve()
```

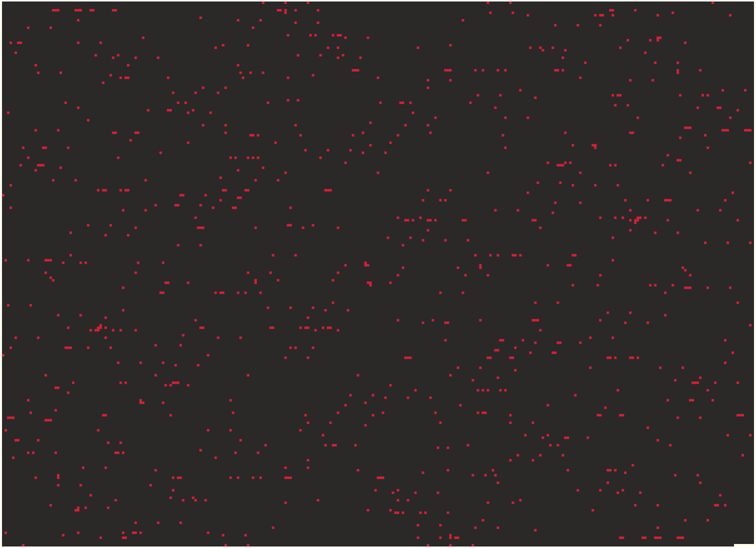
Exponential haystack

```
aa, ab, ac, ad, ae, aB, aC, aD, aE, ba,
bb, bc, bd, be, bA, bC, bD, bE, ca, cb,
cc, cd, ce, cA, cB, cD, cE, da, db, dc,
dd, de, dA, dB, dC, dE, ea, eb, ec, ed,
ee, eA, eB, eC, eD, Ab, Ac, Ad, Ae, AA,
AB, AC, AD, AE, Ba, Bc, Bd, Be, BA, BB,
BC, BD, BE, Ca, Cb, Cd, Ce, CA, CB, CC,
CD, CE, Da, Db, Dc, De, DA, DB, DC, DD,
DE, Ea, Eb, Ec, Ed, EA, EB, EC, ED, EE
```

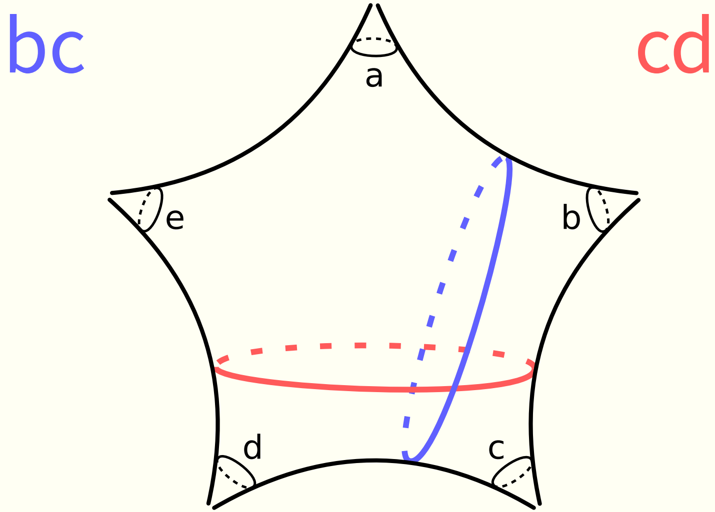
Exponential haystack

aaa, aab, aac, aad, aae, aaB, aaC, aaD, aaE, aba, abb, abc, abd, abe, abA, abC, abD, abE, aca, acb, acc, acd, ace, acA, acB,
acD, acE, ada, adb, adc, add, ade, adA, adB, adC, adE, aea, aeb, aec, aed, aee, aeA, aeB, aeC, aeD, aBa, aBc, aBd, aBe, aBA,
aBB, aBC, aBD, aBE, aCa, aCb, aCd, aCe, aCA, aCB, aCC, aCD, aCE, aDa, aDb, aDc, aDe, aDA, aDB, aDC, aDD, aDE, aEa, aEb, aEc,
aEd, aEA, aEB, aEC, aED, aEE, baa, bab, bac, bad, bae, baB, baC, baD, baE, bba, bbb, bbc, bbd, bbe, bbA, bbC, bbD, bbE, bca,
bcb, bcc, bcd, bce, bcA, bcB, bcD, bcE, bda, bdb, bdc, bdd, bde, bdA, bdB, bdC, bdE, bea, beb, bec, bed, bee, beA, beB, beC,
beD, bAb, bAc, bAd, bAe, bAA, bAB, bAC, bAD, bAE, bCa, bCb, bCd, bCe, bCA, bCB, bCC, bCD, bCE, bDa, bDb, bDc, bDe, bDA, bDB,
bDC, bDD, bDE, bEa, bEb, bEc, bEd, bEA, bEB, bEC, bED, bEE, caa, cab, cac, cad, cae, caB, caC, caD, caE, cba, cbb, cbc, cbd,
cbe, cbA, cbC, cbD, cbE, cca, ccb, ccc, ccd, cce, ccA, ccB, ccD, ccE, cda, cdb, cdc, cdd, cde, cdA, cdB, cdC, cdE, cea, ceb,
cec, ced, cee, ceA, ceB, ceC, ceD, cAb, cAc, cAd, cAe, cAA, cAB, cAC, cAD, cAE, cBa, cBc, cBd, cBe, cBA, cBB, cBC, cBD, cBE,
cDa, cDb, cDc, cDe, cDA, cDB, cDC, cDD, cDE, cEa, cEb, cEc, cEd, cEA, cEB, cEC, cED, cEE, daa, dab, dac, dad, dae, daB, daC,
daD, daE, dba, dbb, dbc, dbd, dbe, dbA, dbC, dbD, dbE, dca, dcb, dcc, dcd, dce, dcA, dcB, dcD, dcE, dda, ddb, ddc, ddd, dde,
ddA, ddB, ddC, ddE, dea, deb, dec, ded, dee, deA, deB, deC, deD, dAb, dAc, dAd, dAe, dAA, dAB, dAC, dAD, dAE, dBa, dBc, dBd,
dBe, dBA, dBB, dBC, dBD, dBE, dCa, dCb, dCd, dCe, dCA, dCB, dCC, dCD, dCE, dEa, dEb, dEc, dEd, dEA, dEB, dEC, dED, dEE, eaa,
eab, eac, ead, eae, eaB, eaC, eaD, eaE, eba, ebb, ebc, ebd, ebe, ebA, ebC, ebD, ebE, eca, ecb, ecc, ecd, ece, ecA, ecB, ecD,
ecE, eda, edb, edc, edd, ede, edA, edB, edC, edE, eea, eeb, eec, eed, eee, eeA, eeB, eeC, eeD, eAb, eAc, eAd, eAe, eAA, eAB,
eAC, eAD, eAE, eBa, eBc, eBd, eBe, eBA, eBB, eBC, eBD, eBE, eCa, eCb, eCd, eCe, eCA, eCB, eCC, eCD, eCE, eDa, eDb, eDc, eDe,
eDA, eDB, eDC, eDD, eDE, Aba, Abb, Abc, Abd, Abe, AbA, AbC, AbD, AbE, Aca, Acb, Acc, Acd, Ace, AcA, AcB, AcD, AcE, Ada, Adb,
Adc, Add, Ade, AdA, AdB, AdC, AdE, Aea, Aeb, Aec, Aed, Aee, AeA, AeB, AeC, AeD, AAb, AAc, AAd, AAe, AAA, AAB, AAC, AAD, AAE,
ABa, ABc, ABd, ABe, ABA, ABB, ABC, ABD, ABE, ACa, ACb, ACd, ACe, ACA, ACB, ACC, ACD, ACE, ADa, ADb, ADc, ADe, ADA, ADB, ADC,
ADD, ADE, AEa, AEb, AEc, AEd, AEA, AEB, AEC, AED, AEE, Baa, Bab, Bac, Bad, Bae, BaB, BaC, BaD, BaE, Bca, Bcb, Bcc, Bcd, Bce,
BcA, BcB, BcD, BcE, Bda, Bdb, Bdc, Bdd, Bde, BdA, BdB, BdC, BdE, Bea, Beb, Bec, Bed, Bee, BeA, BeB, BeC, BeD, BAb, BAc, BAd,
BAe, BAA, BAB, BAC, BAD, BAE, BBa, BBc, BBd, BBe, BBA, BBB, BBC, BBD, BBE, BCa, BCb, BCd, BCe, BCA, BCB, BCC, BCD, BCE, BDa,
BDb, BDc, BDe, BDA, BDB, BDC, BDD, BDE, BEa, BEb, BEc, BEd, BEA, BEB, BEC, BED, BEE, Caa, Cab, Cac, Cad, Cae, CaB, CaC, CaD,
CaE, Cba, Cbb, Cbc, Cbd, Cbe, CbA, CbC, CbD, CbE, Cda, Cdb, Cdc, Cdd, Cde, CdA, CdB, CdC, CdE, Cea, Ceb, Cec, Ced, Cee, CeA,
CeB, CeC, CeD, CAb, CAc, CAd, CAe, CAA, CAB, CAC, CAD, CAE, CBa, CBc, CBd, CBe, CBA, CBB, CBC, CBD, CBE, CCa, CCb, CCd, CCe,
CCA, CCB, CCC, CCD, CCE, CDa, CDb, CDc, CDe, CDA, CDB, CDC, CDD, CDE, CEa, CEb, CEc, CEd, CEA, CEB, CEC, CED, CEE, Daa, Dab,
Dac, Dad, Dae, DaB, DaC, DaD, DaE, Dba, Dbb, Dbc, Dbd, Dbe, DbA, DbC, DbD, DbE, Dca, Dcb, Dcc, Dcd, Dce, DcA, DcB, DcD, DcE,
Dea, Deb, Dec, Ded, Dee, DeA, DeB, DeC, DeD, DAb, DAc, DAd, DAe, DAA, DAB, DAC, DAD, DAE, DBa, DBc, DBd, DBe, DBA, DBB, DBC,
DBD, DBE, DCa, DCb, DCd, DCe, DCA, DCB, DCC, DCD, DCE, DDa, DDb, DDc, DDe, DDA, DDB, DDC, DDD, DDE, DEa, DEb, DEc, DEd, DEA,
DEB, DEC, DED, DEE, Eaa, Eab, Eac, Ead, Eae, EaB, EaC, EaD, EaE, Eba, Ebb, Ebc, Ebd, Ebe, EbA, EbC, EbD, EbE, Eca, Ecb, Ecc,
Ecd, Ece, EcA, EcB, EcD, EcE, Eda, Edb, Edc, Edd, Ede, EdA, EdB, EdC, EdE, EAb, EAc, EAd, EAe, EAA, EAB, EAC, EAD, EAE, EBa,
EBc, EBd, EBe, EBA, EBB, EBC, EBD, EBE, ECa, ECb, ECd, ECe, ECA, ECB, ECC, ECD, ECE, EDa, EDb, EDc, EDe, EDA, EDB, EDC, EDD,
EDE, EEa, EEb, EEc, EEd, EEA, EEB, EEC, EED, EEE

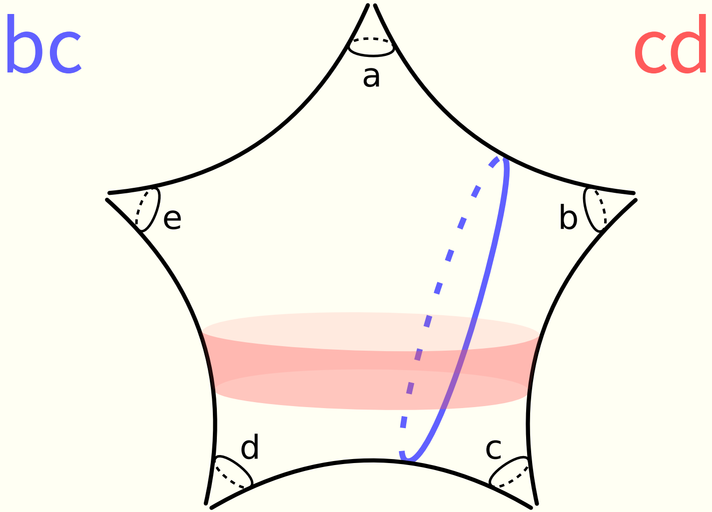
Exponential haystack



Twisting



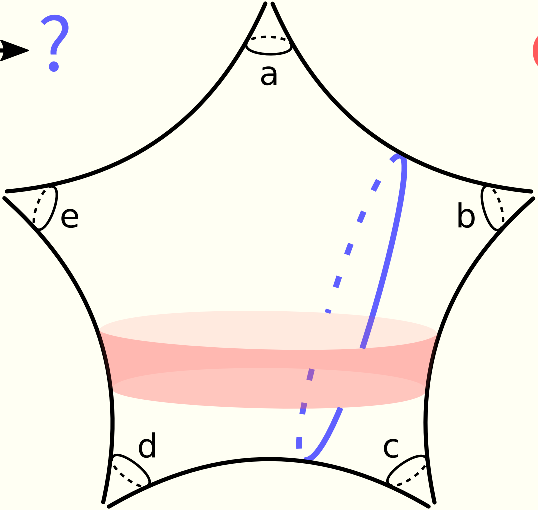
Twisting



Twisting

bc → ?

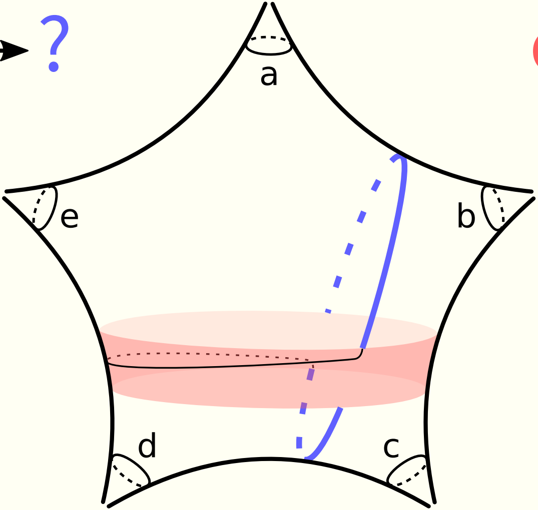
cd



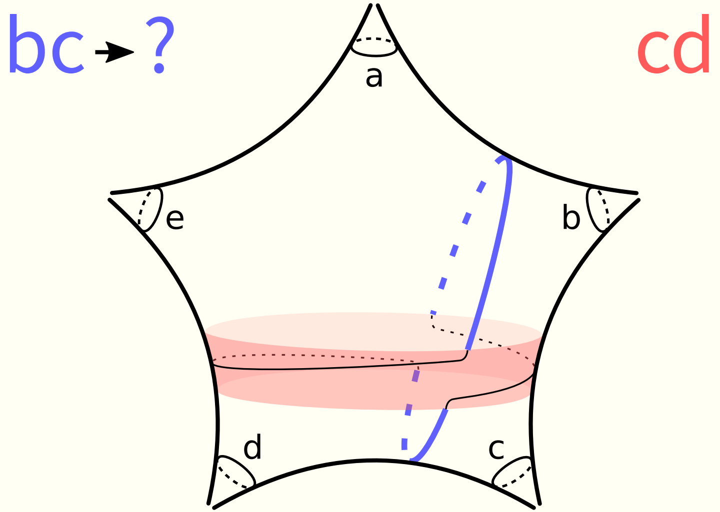
Twisting

bc → ?

cd



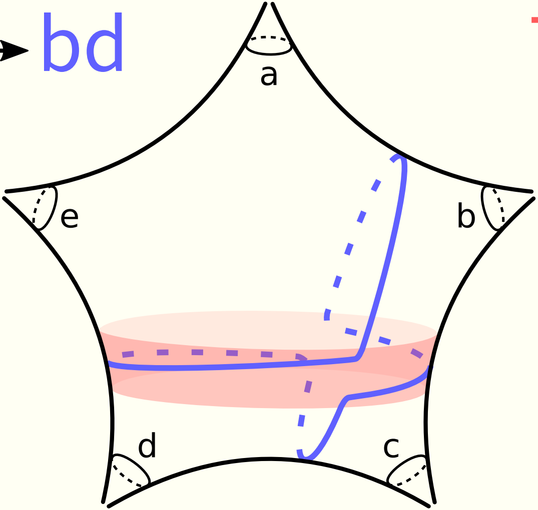
Twisting



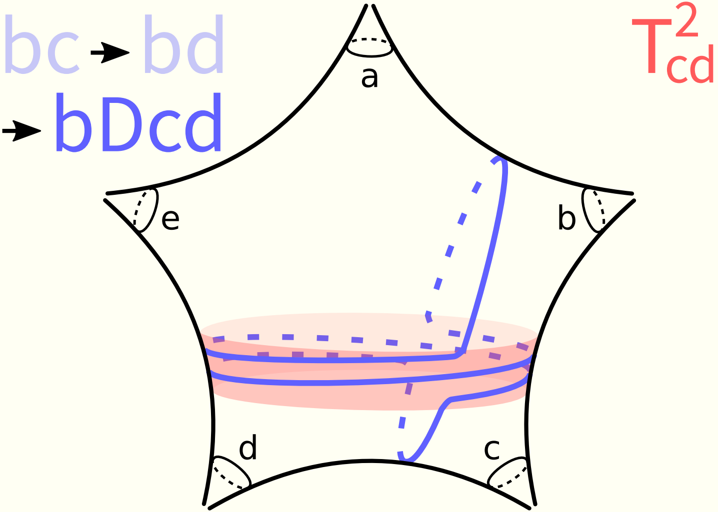
Twisting

bc → bd

T_{cd}



Twisting



Twisting is search & replace

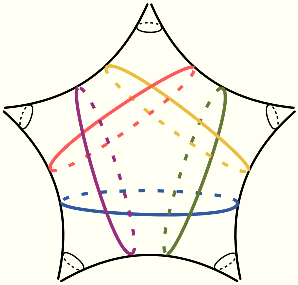
T_{cd} is equivalent to substituting:

$$
\begin{aligned}
\mathrm{a} &\to \mathrm{a} \\
\mathrm{b} &\to \mathrm{b} \\
\mathrm{c} &\to \mathrm{d} \\
\mathrm{d} &\to \mathrm{Dcd} \\
\mathrm{e} &\to \mathrm{e}
\end{aligned}
$$

Twisting is search & replace

		T_{ab}	T_{bc}	T_{cd}	T_{de}	T_{ea}
a	$\rightarrow$	b	a	a	a	Aea
b	$\rightarrow$	Bab	c	b	b	b
c	$\rightarrow$	c	Cbc	d	c	c
d	$\rightarrow$	d	d	Dcd	e	d
e	$\rightarrow$	e	e	e	Ede	a

Twists generate everything

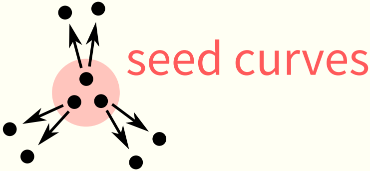


Fact: Any nonperipheral simple closed curve can be transformed to any other one by applying twists T_{ab}, T_{bc}, T_{cd}, T_{de}, T_{ea} and their inverses.

Twist-based generator

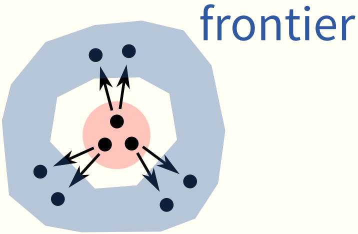
Twist-based generator

twists $\rightrightarrows$



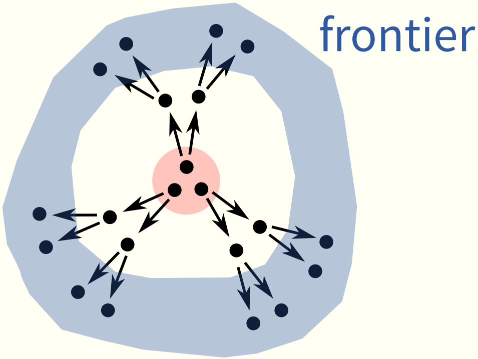
Twist-based generator

twists $\rightrightarrows$

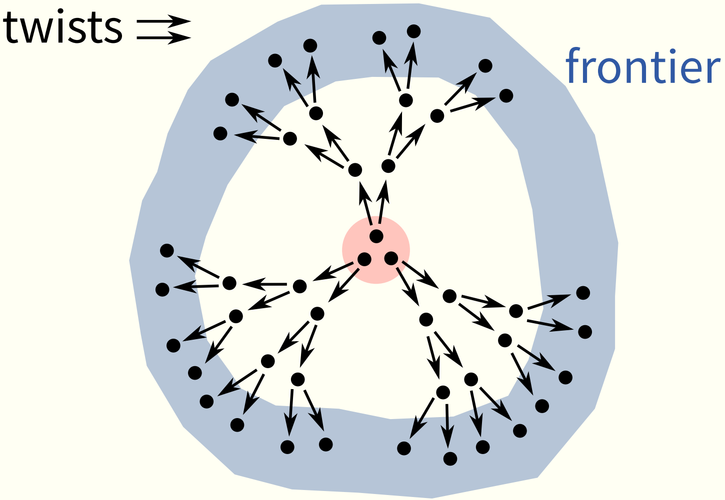


Twist-based generator

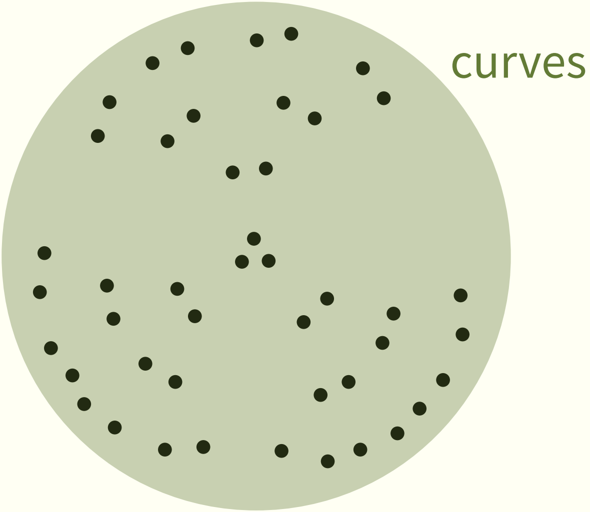
twists ⇉



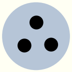
Twist-based generator



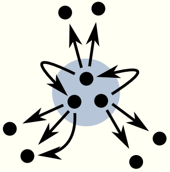
Twist-based generator



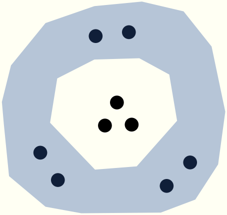
Twist-based generator



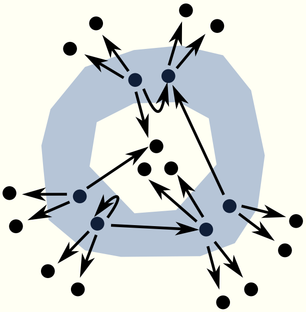
Twist-based generator



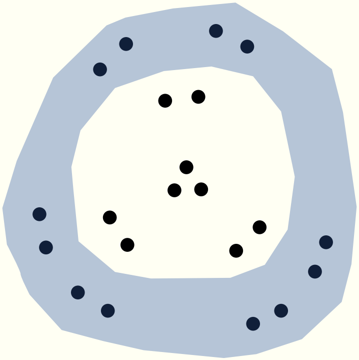
Twist-based generator



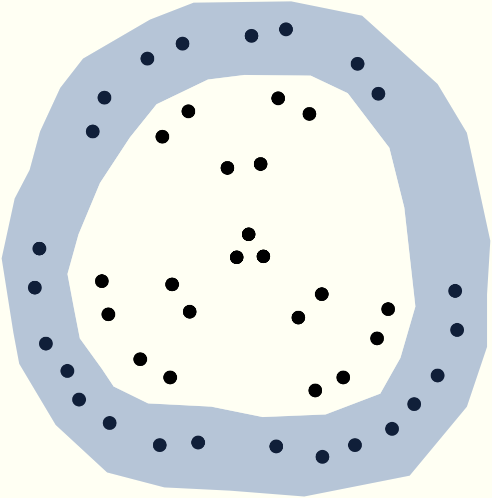
Twist-based generator



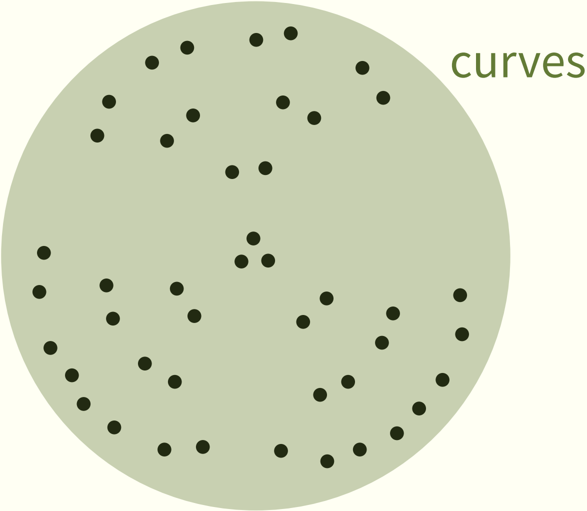
Twist-based generator



Twist-based generator



Twist-based generator



Twist-based generator

Python's set type stores collections of distinct elements and supports efficient boolean operations.

```
depth = 5

twists = { Tab, Tab_inv, Tbc, Tbc_inv } # etc
curves = set()
frontier = {'ab', 'bc', 'cd', 'de', 'ea'}

for _ in range(depth):
    latest = \
        { T(x) for x in frontier for T in twists }
    frontier = latest.difference(curves)
    curves.update(frontier)
```

Coordinates

How to visualize the entire set of NSCCs?

Assign coordinates.

Coordinates

How to visualize the entire set of NSCCs?
Assign coordinates.

In 1986, W. P. Thurston described a way to associate a 4-dimensional vector to each NSCC on the five-holed sphere.

Coordinates

How to visualize the entire set of NSCCs?
Assign coordinates.

In 1986, W. P. Thurston described a way to associate a 4-dimensional vector to each NSCC on the five-holed sphere.

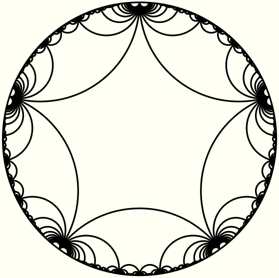
The resulting cloud of points densely fills a 3-dimensional hypersurface in 4-space.
This hypersurface is *PML*.

Length & dlength

Curve x $\rightarrow$ matrix ϱ(x) $\rightarrow$ length L(x)

$+$

Tiling $\nearrow$

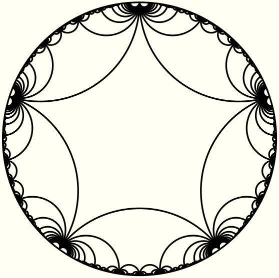


Length & dlength

Curve x $\to$ matrix ϱ(x) $\to$ length L(x)

+

Tiling



$$\varrho(a) = \begin{pmatrix} 1 & 5.50 \\ 0 & 1 \end{pmatrix}$$

$$\varrho(b) = \begin{pmatrix} 3.62 & 3.60 \\ -1.90 & -1.62 \end{pmatrix}$$

$$\vdots$$

$$\varrho(abcB) = \varrho(a)\varrho(b)\varrho(c)\varrho(b)^{-1} = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$$

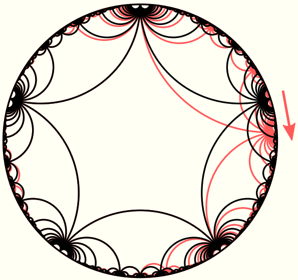
$$L(abcB) = 2\,\text{arccosh}\left(\tfrac{1}{2}(p+s)\right) = 6.467$$

Length & dlength

Curve x $\to$ matrix $\varrho(\text{x})$ $\to$ length L(x)

+

ΔTiling$_1$ $\nearrow$



$$\varrho(a) = \begin{pmatrix} 1 & 5.50 \\ 0 & 1 \end{pmatrix} + \Delta\varrho(a)_1$$

$$\varrho(b) = \begin{pmatrix} 3.62 & 3.60 \\ -1.90 & -1.62 \end{pmatrix} + \Delta\varrho(b)_1$$

$$\vdots$$

$$\varrho(abcB) = \varrho(a)\varrho(b)\varrho(c)\varrho(b)^{-1} = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$$

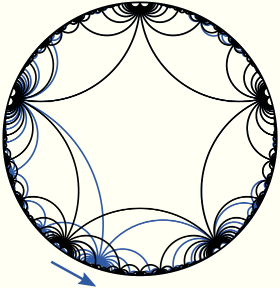
$$L(abcB) = 2\,\text{arccosh}\left(\tfrac{1}{2}(p+s)\right) = 6.467 + \Delta L_1$$

Length & dlength

Curve x $\to$ matrix $\varrho(x)$ $\to$ length L(x)

+

ΔTiling$_2$



$$\varrho(a) = \begin{pmatrix} 1 & 5.50 \\ 0 & 1 \end{pmatrix} + \Delta\varrho(a)_2$$

$$\varrho(b) = \begin{pmatrix} 3.62 & 3.60 \\ -1.90 & -1.62 \end{pmatrix} + \Delta\varrho(b)_2$$

$$\vdots$$

$$\varrho(abcB) = \varrho(a)\varrho(b)\varrho(c)\varrho(b)^{-1}$$

$$= \begin{pmatrix} p & q \\ r & s \end{pmatrix}$$

$$L(abcB) = 2 \operatorname{arccosh}\left(\tfrac{1}{2}(p+s)\right) = 6.467 + \Delta L_2$$

Computing matrices

Numpy's matrix algebra + recursive splitting to compute $\rho(x)$:

```
import numpy as np

def representation(gen_matrices):
    def _rho(x):
        if len(x) == 0:
            # identity matrix
            return np.eye(2)
        elif len(x) == 1:
            # generator matrix
            return gen_matrices[x]
        else:
            N = len(x)
            return np.dot(_rho(x[:N//2]),_rho(x[N//2:]))
    return _rho

rho = representation( {'a': np.array( [[0,1],[1,1]] )} ) # etc

print(rho('aaaaa'))  # -> [[3 5], [5 8]]
```

Length and dlength

```
eps = 0.0001
rho0 = representation( {'a': np.array( [[0,1],[1,1]] )} )
rho1 = representation( {'a': np.array( [[0,1],[1+eps,1]] )} )

L0 = 2.0*np.arccosh(0.5*np.trace(rho0('aaaaa')))
L1 = 2.0*np.arccosh(0.5*np.trace(rho1('aaaaa')))

dL = (L1 - L0) / eps
print(dL)
```

Four calculations like this give the four coordinates of the vector *dL* associated to a curve.

Length and dlength

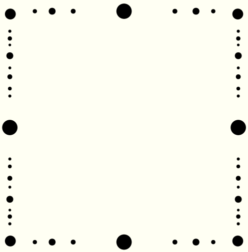
```
eps = 0.0001
rho0 = representation( {'a': np.array( [[0,1],[1,1]] )} )
rho1 = representation( {'a': np.array( [[0,1],[1+eps,1]] )} )

L0 = 2.0*np.arccosh(0.5*np.trace(rho0('aaaaa')))
L1 = 2.0*np.arccosh(0.5*np.trace(rho1('aaaaa')))

dL = (L1 - L0) / eps
print(dL)
```

Four calculations like this give the four coordinates of the vector *dL* associated to a curve.

Next problem: 4D space is difficult to visualize!

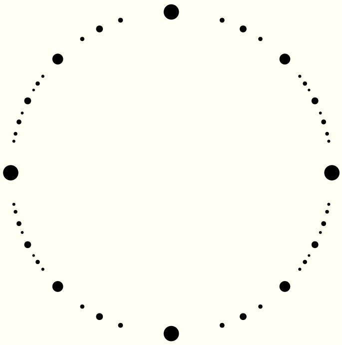
Stereographic projection

“Open up” the surface in 4-space and flatten to 3-space.



Stereographic projection

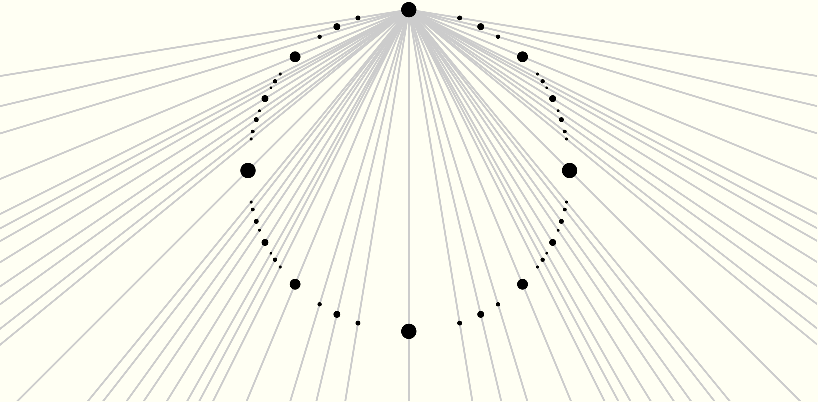
"Open up" the surface in 4-space and flatten to 3-space.



$$(p_1, p_2, p_3, p_4) \longrightarrow \left(\frac{p_1}{1 - p_4}, \frac{p_2}{1 - p_4}, \frac{p_3}{1 - p_4} \right)$$

Stereographic projection

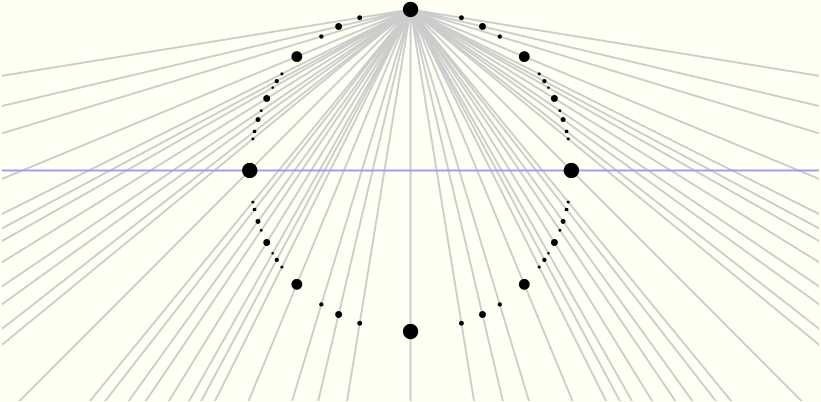
“Open up” the surface in 4-space and flatten to 3-space.



$$(p_1, p_2, p_3, p_4) \longrightarrow \left(\frac{p_1}{1-p_4}, \frac{p_2}{1-p_4}, \frac{p_3}{1-p_4} \right)$$

Stereographic projection

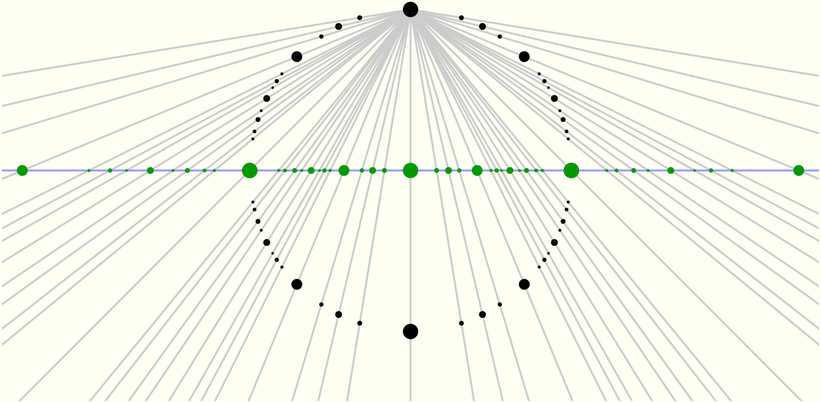
“Open up” the surface in 4-space and flatten to 3-space.



$$(p_1, p_2, p_3, p_4) \longrightarrow \left(\frac{p_1}{1-p_4}, \frac{p_2}{1-p_4}, \frac{p_3}{1-p_4}\right)$$

Stereographic projection

"Open up" the surface in 4-space and flatten to 3-space.



$$(p_1, p_2, p_3, p_4) \longrightarrow \left(\frac{p_1}{1-p_4}, \frac{p_2}{1-p_4}, \frac{p_3}{1-p_4} \right)$$

Stereographic projection

"Open up" the surface in 4-space and flatten to 3-space.

$$(p_1, p_2, p_3, p_4) \longrightarrow \left(\frac{p_1}{1-p_4}, \frac{p_2}{1-p_4}, \frac{p_3}{1-p_4} \right)$$

Rendering

Projecting each dL to a 3-vector (x, y, z), we obtain data like

```
word      L         x         y        z
aDBdbd   19.195   0.630   0.729   0.695
aDbcBd   16.790   0.596   0.470   1.133
bcbCBd   14.664   0.177   0.544   1.242
```

with $> 10^6$ rows.

To make short curves more prominent, but all curves visible, we draw a sphere centered at (x, y, z) of radius $1/L$ for each curve.

Rendering

Projecting each dL to a 3-vector (x, y, z), we obtain data like

```
word    L       x      y      z
aDBdbd  19.195  0.630  0.729  0.695
aDbcBd  16.790  0.596  0.470  1.133
bcbCBd  14.664  0.177  0.544  1.242
```

with $> 10^6$ rows.

To make short curves more prominent, but all curves visible, we draw a sphere centered at (x, y, z) of radius $1/L$ for each curve.

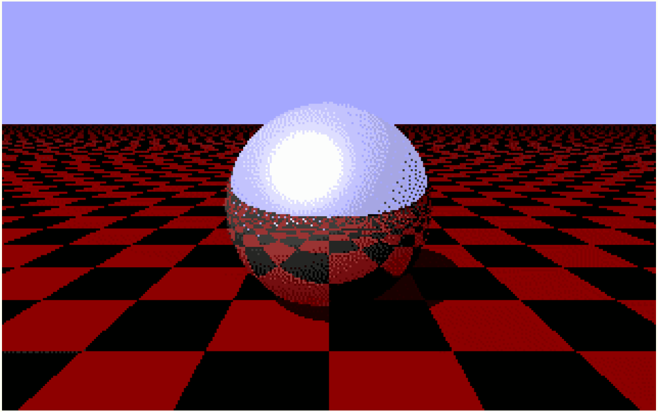
Easy way to draw millions of spheres if real-time rendering is not essential?

Rendering

Party like it's 1992

Rendering

Party like it's 1992



with ray-tracing!

POV-Ray

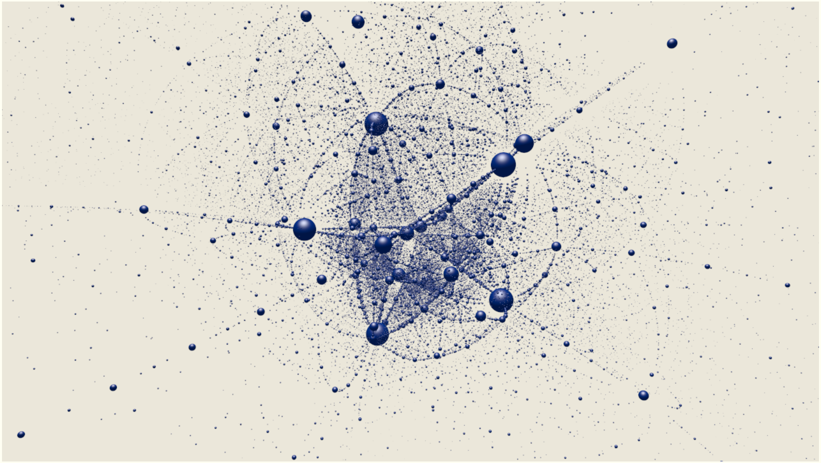
POV-Ray is an open-source ray tracer that uses a scene description language (SDL) with syntax reminiscent of C.

We generate SDL sphere primitives from the *dL* data using:

```
outfile.write('sphere { <%f, %f, %f>, %f }\n'
↪  % (x,y,z,1.0/L))
```

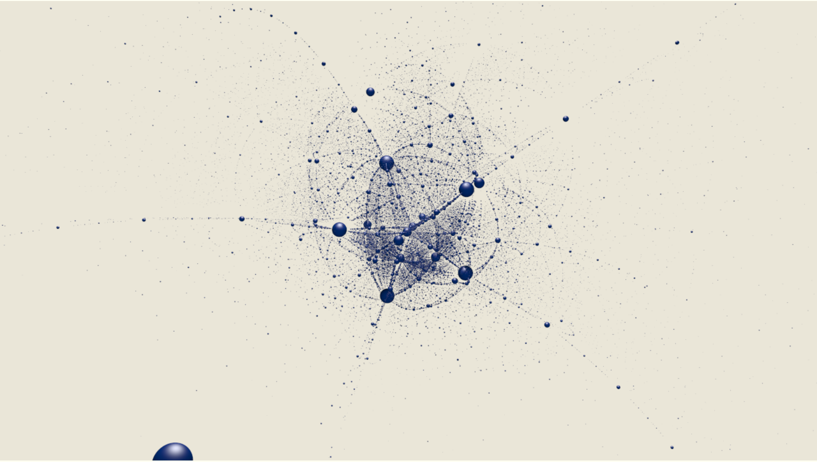
Appending some camera and lighting setup boilerplate gives a full scene file for rendering with POV-Ray.

Fractal dust



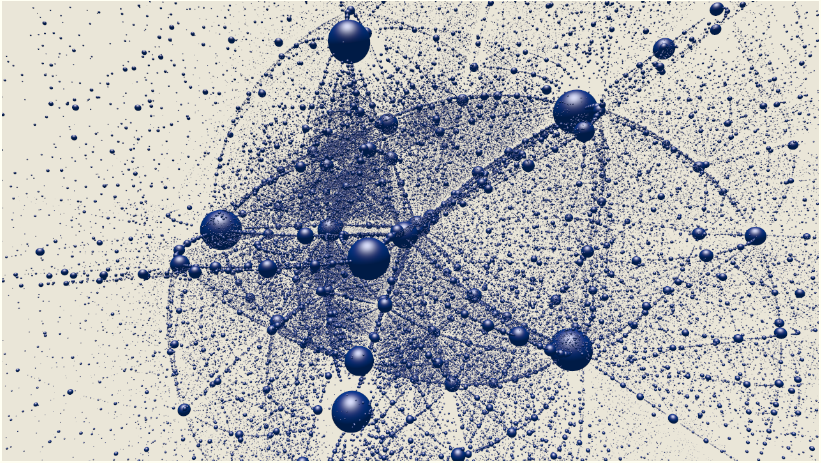
pmls05-001

Rotating the pole (wide angle)



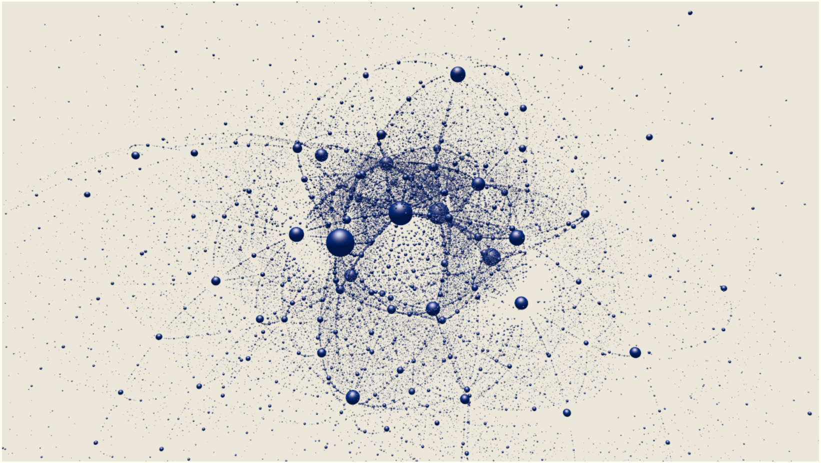
pmls05-010

Rotating the pole (close-up)



pmls05-020

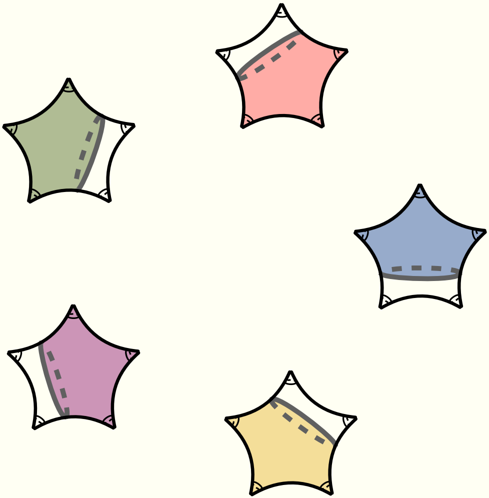
Clifford flow

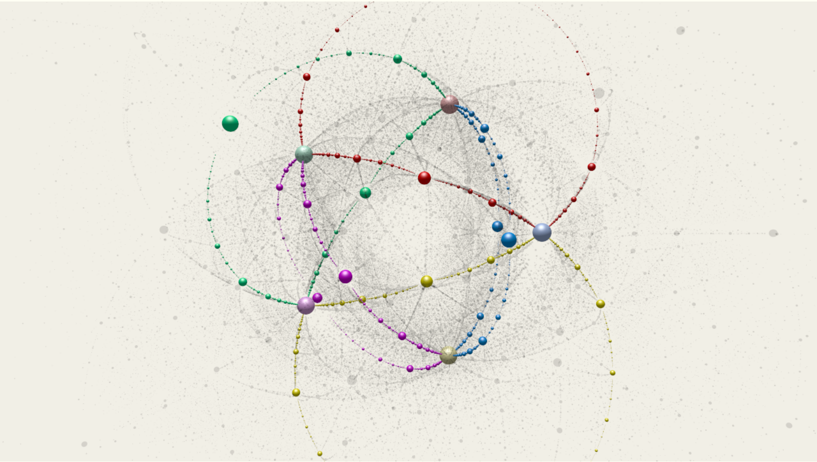


pmls05-030

Rings

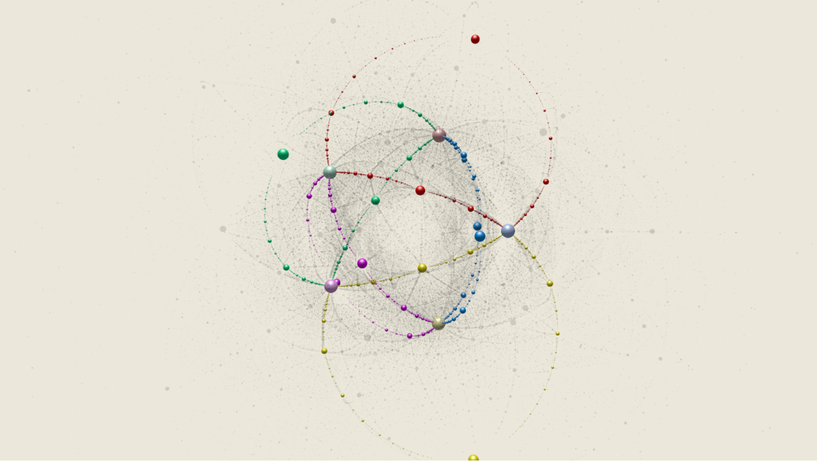
Rings





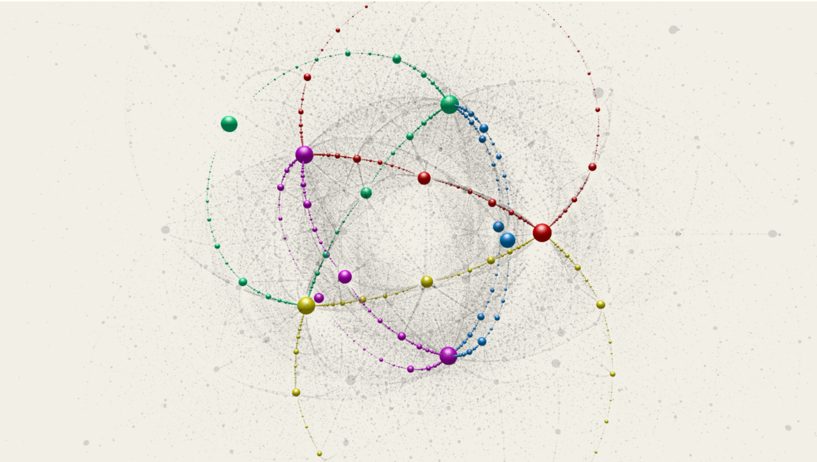
pmls05-071

Rotating the pole



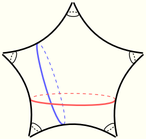
pmls05-041

Rotating the pole II

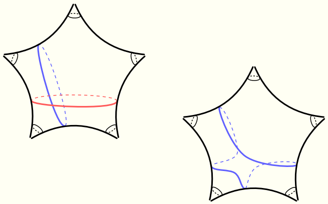


pmls05-061

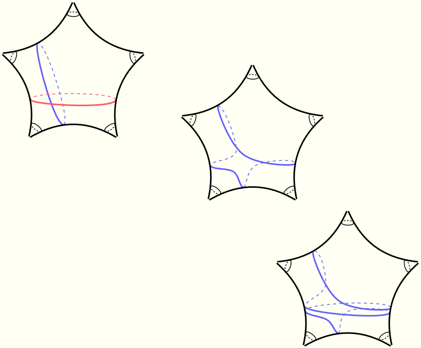
Twists



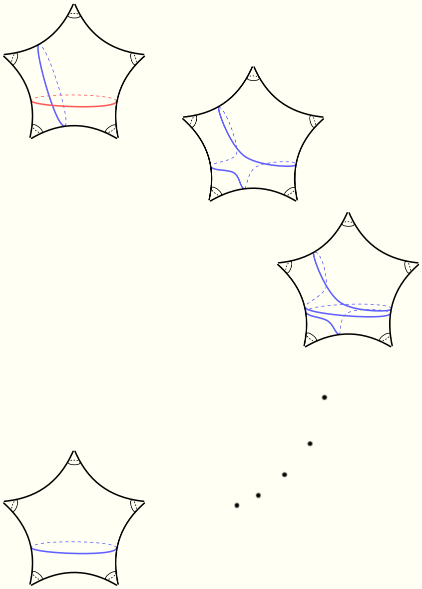
Twists



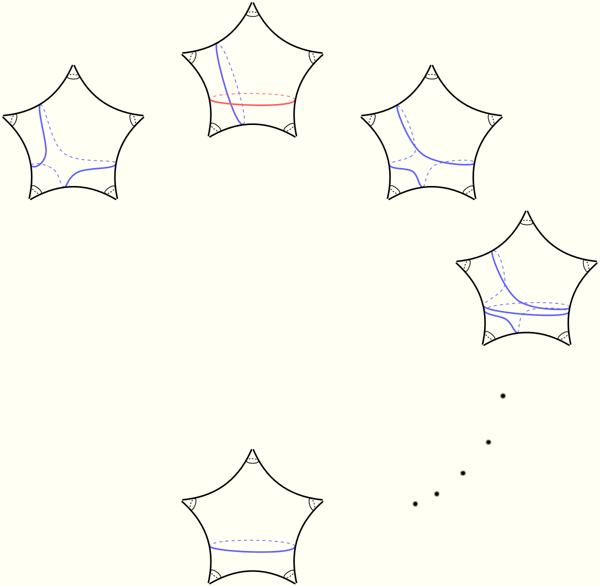
Twists



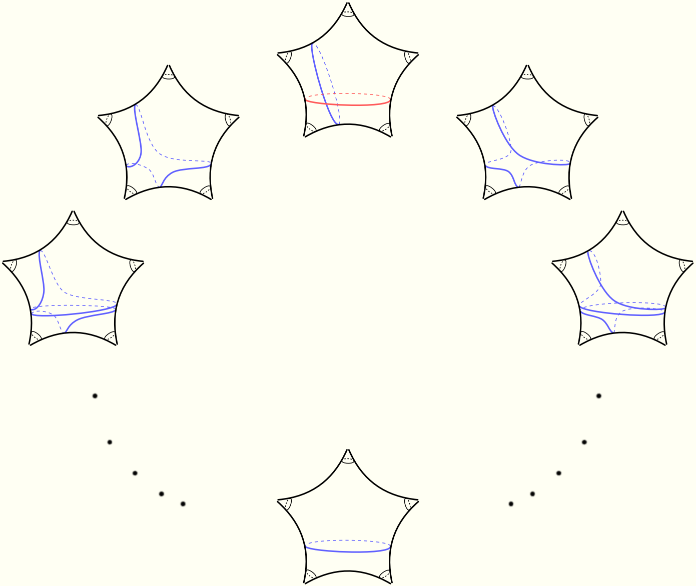
Twists

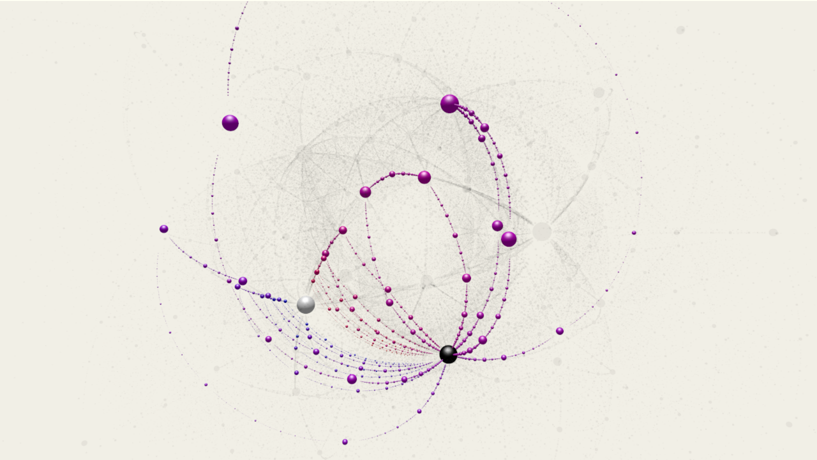


Twists



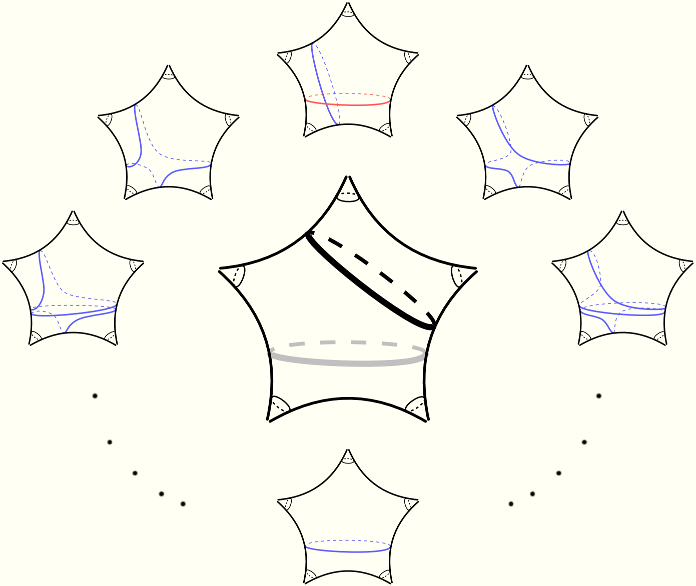
Twists



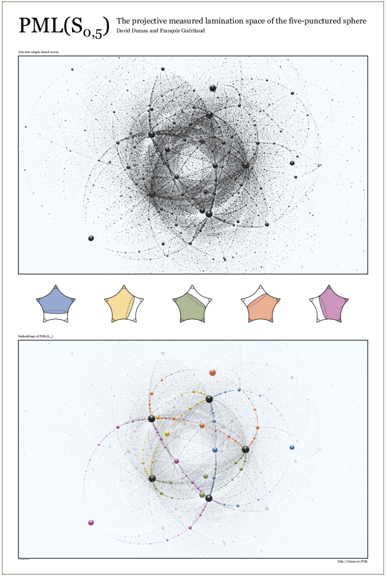


pmls05-081

Twists



Rings poster



dumas.io/PML

David Dumas

david@dumas.io

http://dumas.io

github.com/daviddumas