Examples of ZetaFunctions Final

```
reset()
```

```
load(DIR + '/ZetaFunctions59bis3.sage')
```

```
ZetaFunctions?
```

File: /home/srj/sage-5.9-linux-64bit-ubuntu_10.04.4_lts-x86_64-Linux/local/lib/python2.7/site-packages/sage/all_notebook.py

Type: <type 'type'>

Definition: ZetaFunctions([noargspec])

Docstring:

Class `Zetafunctions` gets an integral multivariate polynomial as argument for calculate their associated (local) Igusa and Topological zeta functions. This class allows to get information about the associated Newton's polyhedron, their faces, the associated cones,...

This class is composed by a integer multivariate polynomial with non-constant term and his associated Newton's polyhedron.

Methods in ZetaFunctions:

- give_info_facets(self)
- give_info_newton(self, faces = False, cones = False)
- newton_plot(self)
- cones_plot(self)
- give_expected_pole_info(self,d = 1, local = False, weights = None)
- igusa_zeta(self, p = None, dict_Ntau = {}, local = False, weights = None, info = False, check = 'ideals')
- topological_zeta(self, d = 1, local = False, weights = None, info = False, check = 'ideals')
- monodromy_zeta(self, weights = None, char = False, info = False)

EXAMPLES:

```
sage: R.<x,y,z> = QQ[]
sage: zex = ZetaFunctions(x^2 + y*z)
sage: zex.give_info_newton()
Newton's polyhedron of x^2 + y*z:
        support points = [(2, 0, 0), (0, 1, 1)]
        vertices = [(0, 1, 1), (2, 0, 0)]
        number of proper faces = 13
        Facet 1: x >= 0
        Facet 2: y >= 0
        Facet 3: z >= 0
        Facet 4: x + 2*z - 2 >= 0
        Facet 5: x + 2*y - 2 >= 0
sage: zex.topological_zeta()
(s + 3)/((s + 1)*(2*s + 3))
sage: zex.give_expected_pole_info()
The candidate poles of the (local) topological zeta function (with d =
1) of x^2 + y*z in function of s are:

-3/2 with expected order: 2
The responsible face of maximal dimension is ``tau_0`` = minimal face
who intersects with the diagonal of ambient space:
        tau6: dim 1,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = []
        generators of cone = [(1, 0, 2), (1, 2, 0)], partition into simplicial
        cones = [[(1, 0, 2), (1, 2, 0)]]

-1 with expected order: 1
(If all Vol(tau) are 0, where tau runs through the selected faces that
are no vertices, then the expected order of -1 is 0).
```

Note: All these examples are extracted of the original Maple's work of K. Hoornaert and D. Loots: "Computer program written in Maple for the calculation of Igusa local zeta function", http://www.wis.kuleuven.ac.be/algebra/kathleen.htm, 2000.

Examples for the **Igusa Zeta Function**

Example 1: $x^2 - y^2 + z^3$

```
R.<x,y,z> = QQ[]
zex3 = ZetaFunctions(x^2 - y^2 + z^3)
```

```
zex3.give_expected_pole_info?
```

File: /tmp/tmpcl0nxa/<string>

Type: <type 'instancemethod'>

Definition: zex3.give_expected_pole_info(d=1, local=False, weights=None)

Docstring:

Prints information about the candidate real poles for the topological zeta function of `f` relative to orders and responsible faces of highest dimension.

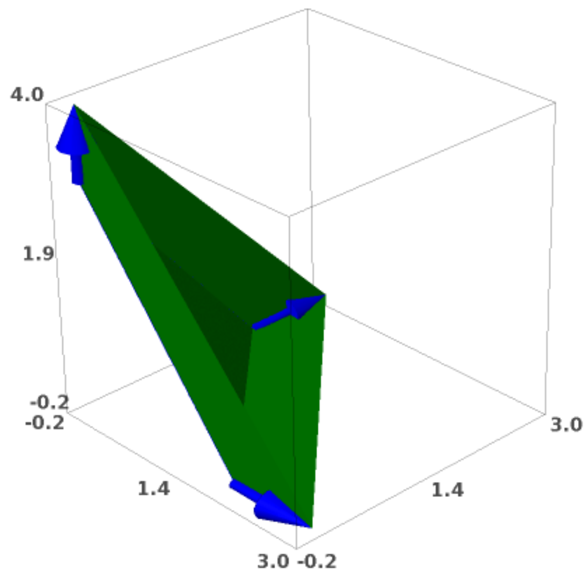
- `local = True` calculates the local (at the origin) topological Zeta function.
- `weights` – a list of weights for the volume form.

```
zex3.give_info_newton(faces = True)
```

```
Newton's polyhedron of z^3 + x^2 - y^2:
        support points = [(0, 0, 3), (2, 0, 0), (0, 2, 0)]
        vertices = [(0, 0, 3), (0, 2, 0), (2, 0, 0)]
        number of proper faces = 13
        Facet 1: y >= 0
        Facet 2: z >= 0
        Facet 3: x >= 0
        Facet 4: 3*x + 3*y + 2*z - 6 >= 0
Information about faces:
tau0: dim 0,  vertices = [(0, 0, 3)],  rays = []

tau1: dim 0,  vertices = [(0, 2, 0)],  rays = []

tau2: dim 0,  vertices = [(2, 0, 0)],  rays = []

tau3: dim 1,  vertices = [(0, 0, 3)],  rays = [(0, 0, 1)]

tau4: dim 1,  vertices = [(0, 0, 3), (0, 2, 0)],  rays = []

tau5: dim 1,  vertices = [(0, 2, 0)],  rays = [(0, 1, 0)]

tau6: dim 1,  vertices = [(0, 0, 3), (2, 0, 0)],  rays = []

tau7: dim 1,  vertices = [(0, 2, 0), (2, 0, 0)],  rays = []

tau8: dim 1,  vertices = [(2, 0, 0)],  rays = [(1, 0, 0)]

tau9: dim 2,  vertices = [(0, 0, 3), (2, 0, 0)],  rays = [(0, 0, 1),
(1, 0, 0)]

tau10: dim 2,  vertices = [(0, 2, 0), (2, 0, 0)],  rays = [(0, 1,
0), (1, 0, 0)]

tau11: dim 2,  vertices = [(0, 0, 3), (0, 2, 0)],  rays = [(0, 0,
1), (0, 1, 0)]

tau12: dim 2,  vertices = [(0, 0, 3), (0, 2, 0), (2, 0, 0)],  rays =
[]
```

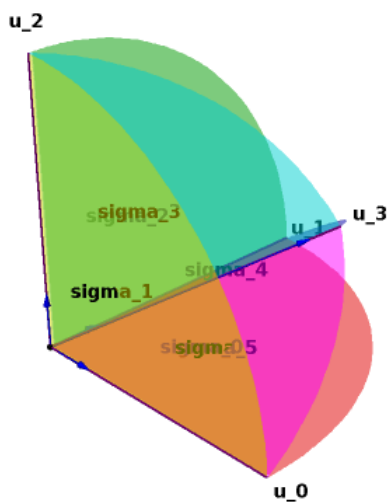
```
zex3.newton_plot()
```

Sleeping... Make Interactive




```
zex3.cones_plot()
```

Sleeping... Make Interactive



$p = 3$:

```
zex3.igusa_zeta(3)
```

```
2*(3^(2*s + 4) - 3^(s + 1) + 2)*3^(2*s)/((3^(s + 1) - 1)*(3^(3*s +
4) - 1))
```

For p arbitrary, without information over the faces:

```
zex3.igusa_zeta()
```

```
(N_Gamma - N_tau10 - N_tau11 - N_tau12 + N_tau4 + N_tau6 + N_tau7 -
N_tau9 + p^(7*s + 12) - p^(7*s + 11) - p^(7*s + 9) + p^(7*s + 8) -
p^(6*s + 11) + p^(6*s + 10) + p^(6*s + 8) - p^(6*s + 7) + p^(5*s +
```

```
9) - p^(5*s + 8) - p^(5*s + 7) + p^(5*s + 6) - p^(4*s + 8) +
2*p^(4*s + 7) - p^(4*s + 6) + p^(3*s + 5) - 2*p^(3*s + 4) + p^(3*s +
3) - p^(2*s + 5) + p^(2*s + 4) + p^(2*s + 3) - p^(2*s + 2) -
p^s*N_Gamma + p^s*N_tau10 + p^s*N_tau11 + p^s*N_tau12 - p^s*N_tau4 -
p^s*N_tau6 - p^s*N_tau7 + p^s*N_tau9 - N_Gamma*p - N_Gamma*p^(7*s +
9) + N_Gamma*p^(7*s + 8) + N_Gamma*p^(6*s + 9) - N_Gamma*p^(6*s + 8)
+ N_Gamma*p^(s + 1) - N_tau10*p^(7*s + 8) + N_tau10*p^(6*s + 8) -
N_tau11*p^(7*s + 8) + N_tau11*p^(6*s + 8) + N_tau12*p - N_tau12*p^(s
+ 1) - N_tau7*p^(5*s + 6) + N_tau7*p^(4*s + 6) - N_tau7*p^(3*s + 3)
+ N_tau7*p^(2*s + 3) - N_tau9*p^(7*s + 8) + N_tau9*p^(6*s +
8))/((p^(s + 1) - 1)*(p^(3*s + 4) - 1)*(p^(3*s + 4) + 1)*(p -
1)*p^2)
```

For p arbitrary, with the number of solutions over the faces:

```
dNtau3 = { x^2-y^2+z^3 : (p-1)*(p-3), -y^2+z^3 : (p-1)^2, x^2+z^3 : (p-1)^2, x^2-y^2 : 2*(p-1)^2 }
```

```
zex3.igusa_zeta(dict_Ntau = dNtau3)
```

```
(p - 1)*(p + p^(2*s + 4) - p^(s + 1) - 1)*p^(2*s)/((p^(s + 1) -
1)*(p^(3*s + 4) - 1))
```

Example 4: $(x-y)^2+z$

```
zex4 = ZetaFunctions((x - y)^2 + z)
```

$p=7$:

```
zex4.igusa_zeta(7)
```

```
The formula for Igusa Zeta function is not valid:
The polynomial is degenerated at least with respect to the face tau
= {dim 1,  vertices = [(0, 2, 0), (2, 0, 0)],  rays = []} over
GF(7)!
NaN
```

For an arbitrary p:

```
zex4.igusa_zeta()
```

```
The formula for Igusa Zeta function is not valid:
The polynomial is degenerated at least with respect to the face tau
= {dim 1,  vertices = [(0, 2, 0), (2, 0, 0)],  rays = []} over the
complex numbers!
NaN
```

Example 5: x^2+yz+z^2

```
zex5 = ZetaFunctions(x^2 + y*z + z^2)
```

For $p=3$ mod 4, we can give the number of solutions over the faces:

```
dNtau5 = { x^2+y*z+z^2 : (p-1)^2, y*z+z^2 : (p-1)^2,  x^2+y*z : (p-1)^2, x^2+z^2 : 0 }
zex5.igusa_zeta(dict_Ntau = dNtau5, info = True)
```

```
Gamma: total polyhedron
L_gamma = -((p - 1)^2*(p^s - 1)*p/(p^(s + 1) - 1) - (p - 1)^3)/p^3


tau0: dim 0,  vertices = [(0, 0, 2)],  rays = []
generators of cone = [(0, 1, 0), (1, 0, 0), (1, 1, 1)], partition
into simplicial cones = [[(0, 1, 0), (1, 0, 0), (1, 1, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = 1/((p^(2*s + 3) - 1)*(p -
```

```
1)^2)

tau1: dim 0,  vertices = [(0, 1, 1)],  rays = []
generators of cone = [(1, 0, 0), (1, 0, 2), (1, 1, 1)], partition
into simplicial cones = [[(1, 0, 0), (1, 0, 2), (1, 1, 1)]]
multiplicities = [2], integral points = [[(0, 0, 0), (1, 0, 1)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = (p^(s + 2) + 1)/((p^(2*s
+ 3) - 1)^2*(p - 1))

tau2: dim 0,  vertices = [(2, 0, 0)],  rays = []
generators of cone = [(0, 1, 0), (0, 0, 1), (1, 0, 2), (1, 1, 1)],
partition into simplicial cones = [[(1, 0, 2), (0, 1, 0)], [(1, 0,
2), (0, 0, 1), (0, 1, 0)], [(1, 0, 2), (1, 1, 1), (0, 1, 0)]]
multiplicities = [1, 1, 1], integral points = [[(0, 0, 0)], [(0, 0,
0)], [(0, 0, 0)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = 1/((p^(2*s + 3) - 1)*(p -
1)) + 1/((p^(2*s + 3) - 1)*(p - 1)^2) + 1/((p^(2*s + 3) - 1)^2*(p -
1))

tau3: dim 1,  vertices = [(0, 0, 2)],  rays = [(0, 0, 1)]
generators of cone = [(0, 1, 0), (1, 0, 0)], partition into
simplicial cones = [[(0, 1, 0), (1, 0, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = (p - 1)^(-2)

tau4: dim 1,  vertices = [(0, 0, 2), (0, 1, 1)],  rays = []
generators of cone = [(1, 0, 0), (1, 1, 1)], partition into
simplicial cones = [[(1, 0, 0), (1, 1, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = (p - 1)^2, L_tau = (p - 1)^2*(p^(s + 2) - 2*p^(s + 1) +
1)/((p^(s + 1) - 1)*p^3) , S_tau = 1/((p^(2*s + 3) - 1)*(p - 1))

tau5: dim 1,  vertices = [(0, 1, 1)],  rays = [(0, 1, 0)]
generators of cone = [(1, 0, 0), (1, 0, 2)], partition into
simplicial cones = [[(1, 0, 0), (1, 0, 2)]]
multiplicities = [2], integral points = [[(0, 0, 0), (1, 0, 1)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = (p^(s + 2) + 1)/((p^(2*s
+ 3) - 1)*(p - 1))

tau6: dim 1,  vertices = [(0, 0, 2), (2, 0, 0)],  rays = []
generators of cone = [(0, 1, 0), (1, 1, 1)], partition into
simplicial cones = [[(0, 1, 0), (1, 1, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = 1/((p^(2*s + 3) - 1)*(p -
1))

tau7: dim 1,  vertices = [(2, 0, 0)],  rays = [(0, 1, 0)]
generators of cone = [(0, 0, 1), (1, 0, 2)], partition into
simplicial cones = [[(0, 0, 1), (1, 0, 2)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = 1/((p^(2*s + 3) - 1)*(p -
1))

tau8: dim 1,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = []
generators of cone = [(1, 0, 2), (1, 1, 1)], partition into
simplicial cones = [[(1, 0, 2), (1, 1, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = (p - 1)^2, L_tau = (p - 1)^2*(p^(s + 2) - 2*p^(s + 1) +
1)/((p^(s + 1) - 1)*p^3) , S_tau = (p^(2*s + 3) - 1)^(-2)
```

```
tau9: dim 1,  vertices = [(2, 0, 0)],  rays = [(1, 0, 0)]
generators of cone = [(0, 1, 0), (0, 0, 1)], partition into
simplicial cones = [[(0, 1, 0), (0, 0, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = (p - 1)^(-2)


tau10: dim 2,  vertices = [(0, 0, 2), (2, 0, 0)],  rays = [(0, 0,
1), (1, 0, 0)]
generators of cone = [(0, 1, 0)], partition into simplicial cones =
[[(0, 1, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = 1/(p - 1)


tau11: dim 2,  vertices = [(0, 0, 2), (0, 1, 1)],  rays = [(0, 0,
1), (0, 1, 0)]
generators of cone = [(1, 0, 0)], partition into simplicial cones =
[[(1, 0, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = (p - 1)^2, L_tau = (p - 1)^2*(p^(s + 2) - 2*p^(s + 1) +
1)/((p^(s + 1) - 1)*p^3) , S_tau = 1/(p - 1)


tau12: dim 2,  vertices = [(2, 0, 0)],  rays = [(0, 1, 0), (1, 0,
0)]
generators of cone = [(0, 0, 1)], partition into simplicial cones =
[[(0, 0, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = 0, L_tau = (p - 1)^3/p^3 , S_tau = 1/(p - 1)


tau13: dim 2,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = [(0, 1,
0)]
generators of cone = [(1, 0, 2)], partition into simplicial cones =
[[(1, 0, 2)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = (p - 1)^2, L_tau = (p - 1)^2*(p^(s + 2) - 2*p^(s + 1) +
1)/((p^(s + 1) - 1)*p^3) , S_tau = 1/(p^(2*s + 3) - 1)


tau14: dim 2,  vertices = [(0, 0, 2), (0, 1, 1), (2, 0, 0)],  rays =
[]
generators of cone = [(1, 1, 1)], partition into simplicial cones =
[[(1, 1, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
N_tau = (p - 1)^2, L_tau = (p - 1)^2*(p^(s + 2) - 2*p^(s + 1) +
1)/((p^(s + 1) - 1)*p^3) , S_tau = 1/(p^(2*s + 3) - 1)


(p^(s + 3) - 1)*(p - 1)*p^(2*s)/((p^(s + 1) - 1)*(p^(2*s + 3) - 1))
```

For $p = 1 \bmod 4$:

```
dNtau5bis = { x^2+y*z+z^2 : (p-1)*(p-3), y*z+z^2 : (p-1)^2,  x^2+y*z : (p-1)^2, x^2+z^2 : 2*(p-1)^2 }
zex5.igusa_zeta(dict_Ntau = dNtau5bis)
```

```
(p^(s + 3) - 1)*(p - 1)*p^(2*s)/((p^(s + 1) - 1)*(p^(2*s + 3) - 1))
```

Example 6: $x^2z + y^2z + u^3$

```
S.<x,y,z,u> = QQ[]
zex6 = ZetaFunctions(x^2*z + y^2*z + u^3)
```

For $p = 1 \bmod 4$, with the number of solutions over the faces:

```
dNtau6 = { x^2*z+y^2*z+u^3 : (p-1)^2*(p-3), x^2*z+u^3 : (p-1)^3, y^2*z + u^3: (p-1)^3,  x^2*z+y^2*z
: 2*(p-1)^3}
zex6.igusa_zeta(dict_Ntau = dNtau6)
```

```
(p - 1)*(p^(4*s + 8) - 3*p^(3*s + 5) + 2*p^(3*s + 4) + 3*p^(2*s + 5)
- 6*p^(2*s + 4) + 3*p^(2*s + 3) + 2*p^(s + 4) - 3*p^(s + 3) +
1)*p^(3*s)/((p^(s + 1) - 1)*(p^(3*s + 4) - 1)^2)
```

Local for $p = 1 \bmod 4$, with the number of solutions over the faces:

```
zex6.igusa_zeta(local = True, dict_Ntau = dNtau6)
```

```
(p - 1)*(p^(4*s + 8) - 3*p^(3*s + 5) + 2*p^(3*s + 4) + 3*p^(2*s + 5)
- 6*p^(2*s + 4) + 3*p^(2*s + 3) + 2*p^(s + 4) - 3*p^(s + 3) +
1)/((p^(s + 1) - 1)*(p^(3*s + 4) - 1)^2*p^4)
```

Local for $p = 3 \bmod 4$, with the number of solutions over the faces::

```
dNtau6bis = { x^2*z+y^2*z+u^3 : (p-1)^3, x^2*z+u^3 : (p-1)^3, y^2*z + u^3: (p-1)^3,  x^2*z+y^2*z : 0}
zex6.igusa_zeta(local = True, dict_Ntau = dNtau6bis, info = True)
```

```
tau0: dim 0,  vertices = [(0, 0, 0, 3)],  rays = []
generators of cone = [(0, 1, 0, 0), (0, 0, 3, 1), (1, 0, 0, 0), (0,
0, 1, 0), (3, 3, 0, 2)], partition into simplicial cones = [[(1, 0,
0, 0), (0, 0, 3, 1), (0, 1, 0, 0)], [(0, 1, 0, 0), (1, 0, 0, 0), (0,
0, 3, 1), (0, 0, 1, 0)], [(3, 3, 0, 2), (1, 0, 0, 0), (0, 0, 3, 1),
(0, 1, 0, 0)]]
multiplicities = [1, 1, 6], integral points = [[(0, 0, 0, 0)], [(0,
0, 0, 0)], [(0, 0, 0, 0), (3, 3, 1, 2), (2, 2, 2, 2), (2, 2, 0, 1),
(1, 1, 1, 1), (1, 1, 2, 1)]]
N_tau = 0, L_tau = (p - 1)^4/p^4 , S_tau = (p^(6*s + 9) + p^(6*s +
8) + 2*p^(3*s + 5) + p^(3*s + 4) + 1)/((p^(3*s + 4) - 1)*(p^(6*s +
8) - 1)*(p - 1)^2) + 1/((p^(3*s + 4) - 1)*(p - 1)^2) + 1/((p^(3*s +
4) - 1)*(p - 1)^3)


tau1: dim 0,  vertices = [(0, 2, 1, 0)],  rays = []
generators of cone = [(0, 0, 0, 1), (0, 0, 3, 1), (1, 0, 0, 0), (3,
3, 0, 2)], partition into simplicial cones = [[(0, 0, 0, 1), (0, 0,
3, 1), (1, 0, 0, 0), (3, 3, 0, 2)]]
multiplicities = [9], integral points = [[(0, 0, 0, 0), (1, 1, 0,
1), (2, 2, 0, 2), (0, 0, 1, 1), (1, 1, 1, 1), (2, 2, 1, 2), (0, 0,
2, 1), (1, 1, 2, 2), (2, 2, 2, 2)]]
N_tau = 0, L_tau = (p - 1)^4/p^4 , S_tau = (p^(6*s + 8) + p^(5*s +
7) + 2*p^(4*s + 6) + p^(3*s + 4) + 2*p^(2*s + 3) + p^(s + 2) +
1)/((p^(3*s + 4) - 1)*(p^(6*s + 8) - 1)*(p - 1)^2)


tau2: dim 0,  vertices = [(2, 0, 1, 0)],  rays = []
generators of cone = [(0, 1, 0, 0), (0, 0, 0, 1), (0, 0, 3, 1), (3,
3, 0, 2)], partition into simplicial cones = [[(0, 1, 0, 0), (0, 0,
0, 1), (0, 0, 3, 1), (3, 3, 0, 2)]]
multiplicities = [9], integral points = [[(0, 0, 0, 0), (1, 1, 0,
1), (2, 2, 0, 2), (0, 0, 1, 1), (1, 1, 1, 1), (2, 2, 1, 2), (0, 0,
2, 1), (1, 1, 2, 2), (2, 2, 2, 2)]]
N_tau = 0, L_tau = (p - 1)^4/p^4 , S_tau = (p^(6*s + 8) + p^(5*s +
7) + 2*p^(4*s + 6) + p^(3*s + 4) + 2*p^(2*s + 3) + p^(s + 2) +
1)/((p^(3*s + 4) - 1)*(p^(6*s + 8) - 1)*(p - 1)^2)


tau11: dim 1,  vertices = [(0, 0, 0, 3), (0, 2, 1, 0)],  rays = []
generators of cone = [(0, 0, 3, 1), (1, 0, 0, 0), (3, 3, 0, 2)],
partition into simplicial cones = [[(0, 0, 3, 1), (1, 0, 0, 0), (3,
3, 0, 2)]]
multiplicities = [3], integral points = [[(0, 0, 0, 0), (1, 1, 1,
1), (2, 2, 2, 2)]]
N_tau = (p - 1)^3, L_tau = (p - 1)^3*(p^(s + 2) - 2*p^(s + 1) +
1)/((p^(s + 1) - 1)*p^4) , S_tau = (p^(6*s + 8) + p^(3*s + 4) +
1)/((p^(3*s + 4) - 1)*(p^(6*s + 8) - 1)*(p - 1))
```

```
tau17: dim 1,  vertices = [(0, 0, 0, 3), (2, 0, 1, 0)],  rays = []
generators of cone = [(0, 1, 0, 0), (0, 0, 3, 1), (3, 3, 0, 2)],
partition into simplicial cones = [[(0, 1, 0, 0), (0, 0, 3, 1), (3,
3, 0, 2)]]
multiplicities = [3], integral points = [[(0, 0, 0, 0), (1, 1, 1,
1), (2, 2, 2, 2)]]
N_tau = (p - 1)^3, L_tau = (p - 1)^3*(p^(s + 2) - 2*p^(s + 1) +
1)/((p^(s + 1) - 1)*p^4) , S_tau = (p^(6*s + 8) + p^(3*s + 4) +
1)/((p^(3*s + 4) - 1)*(p^(6*s + 8) - 1)*(p - 1))


tau20: dim 1,  vertices = [(0, 2, 1, 0), (2, 0, 1, 0)],  rays = []
generators of cone = [(0, 0, 0, 1), (0, 0, 3, 1), (3, 3, 0, 2)],
partition into simplicial cones = [[(0, 0, 0, 1), (0, 0, 3, 1), (3,
3, 0, 2)]]
multiplicities = [9], integral points = [[(0, 0, 0, 0), (0, 0, 1,
1), (0, 0, 2, 1), (1, 1, 0, 1), (1, 1, 1, 1), (1, 1, 2, 2), (2, 2,
0, 2), (2, 2, 1, 2), (2, 2, 2, 2)]]
N_tau = 0, L_tau = (p - 1)^4/p^4 , S_tau = (p^(6*s + 8) + p^(5*s +
7) + 2*p^(4*s + 6) + p^(3*s + 4) + 2*p^(2*s + 3) + p^(s + 2) +
1)/((p^(3*s + 4) - 1)*(p^(6*s + 8) - 1)*(p - 1))


tau22: dim 2,  vertices = [(0, 0, 0, 3), (0, 2, 1, 0), (2, 0, 1,
0)],  rays = []
generators of cone = [(0, 0, 3, 1), (3, 3, 0, 2)], partition into
simplicial cones = [[(0, 0, 3, 1), (3, 3, 0, 2)]]
multiplicities = [3], integral points = [[(0, 0, 0, 0), (1, 1, 1,
1), (2, 2, 2, 2)]]
N_tau = (p - 1)^3, L_tau = (p - 1)^3*(p^(s + 2) - 2*p^(s + 1) +
1)/((p^(s + 1) - 1)*p^4) , S_tau = (p^(6*s + 8) + p^(3*s + 4) +
1)/((p^(3*s + 4) - 1)*(p^(6*s + 8) - 1))


(p^(s + 2) + 1)*(p - 1)*(p^(3*s + 6) - p^(2*s + 4) - p^(2*s + 3) +
p^(s + 3) + p^(s + 2) - 1)/((p^(s + 1) - 1)*(p^(3*s + 4) -
1)*(p^(3*s + 4) + 1)*p^4)
```

Examples for the **Topological Zeta Function**

Example 10: $x^2 + yz$

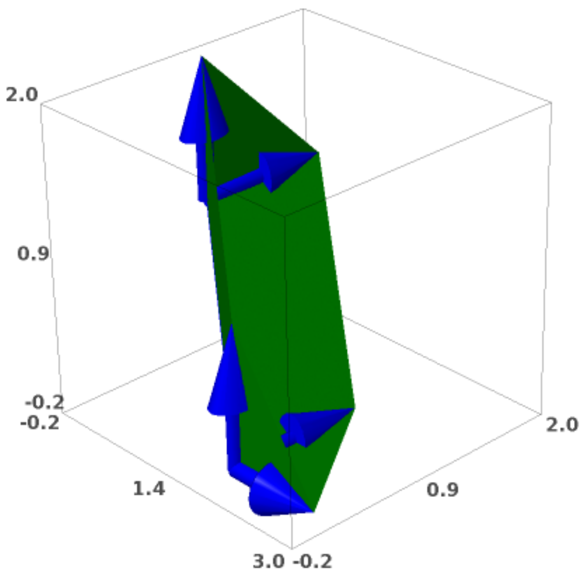
```
R.<x,y,z> = QQ[]
zex10 = ZetaFunctions(R(x^2 + y*z))
```

```
zex10.give_info_newton()
```

```
Newton's polyhedron of x^2 + y*z:
        support points = [(2, 0, 0), (0, 1, 1)]
        vertices = [(0, 1, 1), (2, 0, 0)]
        number of proper faces = 13
        Facet 1: x >= 0
        Facet 2: y >= 0
        Facet 3: z >= 0
        Facet 4: x + 2*z - 2 >= 0
        Facet 5: x + 2*y - 2 >= 0
```

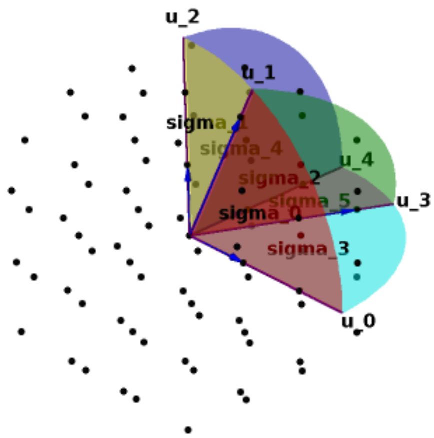
```
zex10.newton_plot()
```

Sleeping... Make Interactive




```
zex10.cones_plot()
```

Sleeping... Make Interactive




```
zex10.give_expected_pole_info()
```

```
The candidate poles of the (local) topological zeta function (with d
= 1) of x^2 + y*z in function of s are:

-3/2 with expected order: 2
The responsible face of maximal dimension is ``tau_0`` = minimal
face who intersects with the diagonal of ambient space:
         tau6: dim 1,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = []
        generators of cone = [(1, 0, 2), (1, 2, 0)], partition into
simplicial cones = [[(1, 0, 2), (1, 2, 0)]]

-1 with expected order: 1
(If all Vol(tau) are 0, where tau runs through the selected faces
that are no vertices, then the expected order of -1 is 0).
```

```
zex10.topological_zeta(info = True)
```

```
Gamma: total polyhedron
J_gamma = 1 , dim_Gamma!*Vol(Gamma) = 0


tau0: dim 0,  vertices = [(0, 1, 1)],  rays = []
generators of cone = [(1, 0, 0), (1, 0, 2), (1, 2, 0)], partition
into simplicial cones = [[(1, 0, 0), (1, 0, 2), (1, 2, 0)]]
multiplicities = [4], integral points = [[(0, 0, 0), (1, 1, 0), (1,
0, 1), (1, 1, 1)]]
J_tau = 4/(2*s + 3)^2 , dim_tau!*Vol(tau) = 1


tau1: dim 0,  vertices = [(2, 0, 0)],  rays = []
generators of cone = [(0, 1, 0), (0, 0, 1), (1, 0, 2), (1, 2, 0)],
partition into simplicial cones = [[(1, 0, 2), (0, 1, 0)], [(1, 0,
2), (0, 0, 1), (0, 1, 0)], [(1, 0, 2), (1, 2, 0), (0, 1, 0)]]
multiplicities = [1, 1, 2], integral points = [[(0, 0, 0)], [(0, 0,
0)], [(0, 0, 0), (1, 1, 1)]]
J_tau = (2*s + 5)/(2*s + 3)^2 , dim_tau!*Vol(tau) = 1


tau2: dim 1,  vertices = [(0, 1, 1)],  rays = [(0, 0, 1)]
generators of cone = [(1, 0, 0), (1, 2, 0)], partition into
simplicial cones = [[(1, 0, 0), (1, 2, 0)]]
multiplicities = [2], integral points = [[(0, 0, 0), (1, 1, 0)]]
J_tau = 2/(2*s + 3) , dim_tau!*Vol(tau) = 0


tau3: dim 1,  vertices = [(0, 1, 1)],  rays = [(0, 1, 0)]
generators of cone = [(1, 0, 0), (1, 0, 2)], partition into
simplicial cones = [[(1, 0, 0), (1, 0, 2)]]
multiplicities = [2], integral points = [[(0, 0, 0), (1, 0, 1)]]
J_tau = 2/(2*s + 3) , dim_tau!*Vol(tau) = 0


tau4: dim 1,  vertices = [(2, 0, 0)],  rays = [(0, 0, 1)]
generators of cone = [(0, 1, 0), (1, 2, 0)], partition into
simplicial cones = [[(0, 1, 0), (1, 2, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1/(2*s + 3) , dim_tau!*Vol(tau) = 0


tau5: dim 1,  vertices = [(2, 0, 0)],  rays = [(0, 1, 0)]
generators of cone = [(0, 0, 1), (1, 0, 2)], partition into
simplicial cones = [[(0, 0, 1), (1, 0, 2)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1/(2*s + 3) , dim_tau!*Vol(tau) = 0


tau6: dim 1,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = []
generators of cone = [(1, 0, 2), (1, 2, 0)], partition into
simplicial cones = [[(1, 0, 2), (1, 2, 0)]]
multiplicities = [2], integral points = [[(0, 0, 0), (1, 1, 1)]]
J_tau = 2/(2*s + 3)^2 , dim_tau!*Vol(tau) = 1


tau7: dim 1,  vertices = [(2, 0, 0)],  rays = [(1, 0, 0)]
generators of cone = [(0, 1, 0), (0, 0, 1)], partition into
simplicial cones = [[(0, 1, 0), (0, 0, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1 , dim_tau!*Vol(tau) = 0


tau8: dim 2,  vertices = [(0, 1, 1)],  rays = [(0, 0, 1), (0, 1, 0)]
generators of cone = [(1, 0, 0)], partition into simplicial cones =
[[(1, 0, 0)]]
```

```
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1 , dim_tau!*Vol(tau) = 0


tau9: dim 2,  vertices = [(2, 0, 0)],  rays = [(0, 0, 1), (1, 0, 0)]
generators of cone = [(0, 1, 0)], partition into simplicial cones =
[[(0, 1, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1 , dim_tau!*Vol(tau) = 0


tau10: dim 2,  vertices = [(2, 0, 0)],  rays = [(0, 1, 0), (1, 0,
0)]
generators of cone = [(0, 0, 1)], partition into simplicial cones =
[[(0, 0, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1 , dim_tau!*Vol(tau) = 0


tau11: dim 2,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = [(0, 1,
0)]
generators of cone = [(1, 0, 2)], partition into simplicial cones =
[[(1, 0, 2)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1/(2*s + 3) , dim_tau!*Vol(tau) = 0


tau12: dim 2,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = [(0, 0,
1)]
generators of cone = [(1, 2, 0)], partition into simplicial cones =
[[(1, 2, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1/(2*s + 3) , dim_tau!*Vol(tau) = 0


(s + 3)/((s + 1)*(2*s + 3))
```

Example 11: $x^2 + yz$

$d = 2$:

```
zex11 = zex10
```

```
zex11.give_expected_pole_info(d = 2)
```

```
-3/2 with expected order: 2
The responsible face(s) of maximal dimension is/are:
        tau6: dim 1,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = []
       generators of cone = [(1, 0, 2), (1, 2, 0)], partition into
simplicial cones = [[(1, 0, 2), (1, 2, 0)]]
```

```
zex11.topological_zeta(d = 2, info = True)
```

```
tau1: dim 0,  vertices = [(2, 0, 0)],  rays = []
generators of cone = [(0, 1, 0), (0, 0, 1), (1, 0, 2), (1, 2, 0)],
partition into simplicial cones = [[(1, 0, 2), (0, 1, 0)], [(1, 0,
2), (0, 0, 1), (0, 1, 0)], [(1, 0, 2), (1, 2, 0), (0, 1, 0)]]
multiplicities = [1, 1, 2], integral points = [[(0, 0, 0)], [(0, 0,
0)], [(0, 0, 0), (1, 1, 1)]]
J_tau = (2*s + 5)/(2*s + 3)^2 , dim_tau!*Vol(tau) = 1


tau4: dim 1,  vertices = [(2, 0, 0)],  rays = [(0, 0, 1)]
generators of cone = [(0, 1, 0), (1, 2, 0)], partition into
simplicial cones = [[(0, 1, 0), (1, 2, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1/(2*s + 3) , dim_tau!*Vol(tau) = 0
```

```
tau5: dim 1,  vertices = [(2, 0, 0)],  rays = [(0, 1, 0)]
generators of cone = [(0, 0, 1), (1, 0, 2)], partition into
simplicial cones = [[(0, 0, 1), (1, 0, 2)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1/(2*s + 3) , dim_tau!*Vol(tau) = 0


tau6: dim 1,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = []
generators of cone = [(1, 0, 2), (1, 2, 0)], partition into
simplicial cones = [[(1, 0, 2), (1, 2, 0)]]
multiplicities = [2], integral points = [[(0, 0, 0), (1, 1, 1)]]
J_tau = 2/(2*s + 3)^2 , dim_tau!*Vol(tau) = 1


tau7: dim 1,  vertices = [(2, 0, 0)],  rays = [(1, 0, 0)]
generators of cone = [(0, 1, 0), (0, 0, 1)], partition into
simplicial cones = [[(0, 1, 0), (0, 0, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1 , dim_tau!*Vol(tau) = 0


tau8: dim 2,  vertices = [(0, 1, 1)],  rays = [(0, 0, 1), (0, 1, 0)]
generators of cone = [(1, 0, 0)], partition into simplicial cones =
[[(1, 0, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1 , dim_tau!*Vol(tau) = 0


tau9: dim 2,  vertices = [(2, 0, 0)],  rays = [(0, 0, 1), (1, 0, 0)]
generators of cone = [(0, 1, 0)], partition into simplicial cones =
[[(0, 1, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1 , dim_tau!*Vol(tau) = 0


tau10: dim 2,  vertices = [(2, 0, 0)],  rays = [(0, 1, 0), (1, 0,
0)]
generators of cone = [(0, 0, 1)], partition into simplicial cones =
[[(0, 0, 1)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1 , dim_tau!*Vol(tau) = 0


tau11: dim 2,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = [(0, 1,
0)]
generators of cone = [(1, 0, 2)], partition into simplicial cones =
[[(1, 0, 2)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1/(2*s + 3) , dim_tau!*Vol(tau) = 0


tau12: dim 2,  vertices = [(0, 1, 1), (2, 0, 0)],  rays = [(0, 0,
1)]
generators of cone = [(1, 2, 0)], partition into simplicial cones =
[[(1, 2, 0)]]
multiplicities = [1], integral points = [[(0, 0, 0)]]
J_tau = 1/(2*s + 3) , dim_tau!*Vol(tau) = 0


1/(2*s + 3)
```

Example 12: $x^2y^2z + xyz^2$

$d = 2$:

```
zex12 = ZetaFunctions(R(x^2*y^2*z + x*y*z^2))
```

```
zex12.give_expected_pole_info(d = 2)
```

```
There will be no poles for the (local) topological zeta function
(with d = 2) of x^2*y^2*z + x*y*z^2.
```

```
zex12.topological_zeta(d = 2)
```

```
0
```

Example 14: $xyz + uvw + xyw + zuv$

```
S2.<x,y,z,u,v,w> = QQ[]
zex14 = ZetaFunctions(x*y*z + u*v*w + x*y*w + z*u*v)
```

```
zex14.give_info_newton()
```

```
Newton's polyhedron of x*y*z + z*u*v + x*y*w + u*v*w:
        support points = [(1, 1, 1, 0, 0, 0), (0, 0, 1, 1, 1, 0), (1, 1, 0,
0, 0, 1), (0, 0, 0, 1, 1, 1)]
        vertices = [(0, 0, 0, 1, 1, 1), (0, 0, 1, 1, 1, 0), (1, 1, 0, 0, 0,
1), (1, 1, 1, 0, 0, 0)]
        number of proper faces = 203
        Facet 1: z + w - 1 >= 0
        Facet 2: w >= 0
        Facet 3: u >= 0
        Facet 4: v >= 0
        Facet 5: x + v - 1 >= 0
        Facet 6: x + u - 1 >= 0
        Facet 7: y + u - 1 >= 0
        Facet 8: y + v - 1 >= 0
        Facet 9: y >= 0
        Facet 10: x >= 0
        Facet 11: z >= 0
```

```
zex14.give_expected_pole_info()
```

```
The candidate poles of the (local) topological zeta function (with d
= 1) of x*y*z + z*u*v + x*y*w + u*v*w in function of s are:

-2 with expected order: 4
The responsible face of maximal dimension is ``tau_0`` = minimal
face who intersects with the diagonal of ambient space:
         tau178: dim 2,  vertices = [(0, 0, 0, 1, 1, 1), (0, 0, 1, 1, 1,
0), (1, 1, 0, 0, 0, 1), (1, 1, 1, 0, 0, 0)],  rays = []
        generators of cone = [(0, 0, 1, 0, 0, 1), (1, 0, 0, 0, 1, 0), (1,
0, 0, 1, 0, 0), (0, 1, 0, 1, 0, 0), (0, 1, 0, 0, 1, 0)], partition
into simplicial cones = [[(0, 0, 1, 0, 0, 1), (1, 0, 0, 0, 1, 0),
(0, 1, 0, 1, 0, 0)], [(0, 0, 1, 0, 0, 1), (1, 0, 0, 0, 1, 0), (1, 0,
0, 1, 0, 0), (0, 1, 0, 1, 0, 0)], [(0, 0, 1, 0, 0, 1), (0, 1, 0, 0,
1, 0), (0, 1, 0, 1, 0, 0), (1, 0, 0, 0, 1, 0)]]

-1 with expected order: 1
(If all Vol(tau) are 0, where tau runs through the selected faces
that are no vertices, then the expected order of -1 is 0).
```

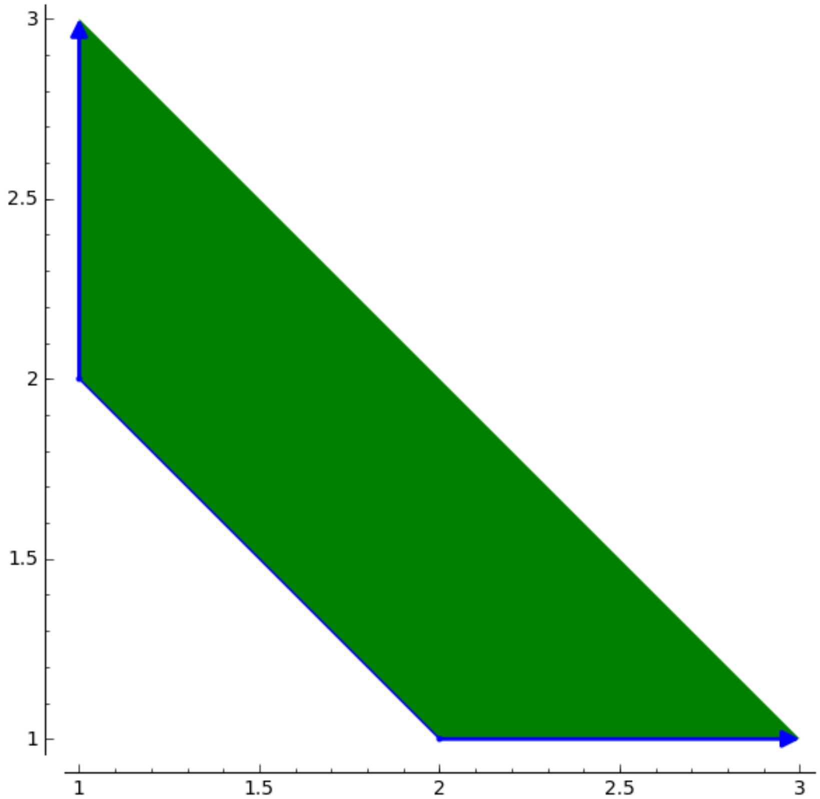
```
zex14.topological_zeta()
```

```
The formula for Topological Zeta function is not valid:
The polynomial is degenerated at least with respect to the face tau
= {dim 2,  vertices = [(0, 0, 0, 1, 1, 1), (0, 0, 1, 1, 1, 0), (1,
1, 0, 0, 0, 1), (1, 1, 1, 0, 0, 0)],  rays = []} over the complex
numbers!
NaN
```

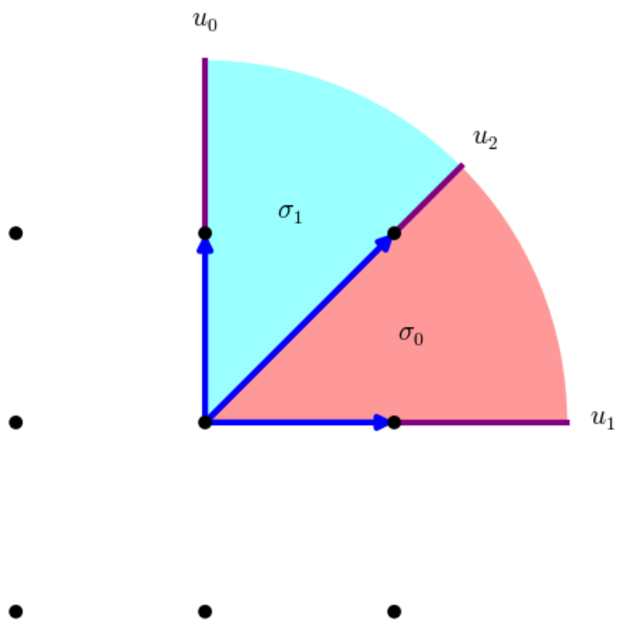
Example 15: $xy^3 + xy^2 + x^2y$

```
R2.<x,y> = QQ[]
zex15 = ZetaFunctions(x*y^3 + x*y^2 + x^2*y)
```

```
zex15.newton_plot()
```




```
zex15.cones_plot()
```



Local:

```
zex15.give_expected_pole_info(local = True)
```

```
The candidate poles of the (local) topological zeta function (with d
= 1) of x*y^3 + x^2*y + x*y^2 in function of s are:

-2/3 with expected order: 1
The responsible face of maximal dimension is ``tau_0`` = minimal
face who intersects with the diagonal of ambient space:
         tau4: dim 1,  vertices = [(1, 2), (2, 1)],  rays = []
        generators of cone = [(1, 1)], partition into simplicial cones =
[[(1, 1)]]

-1 with expected order: 1
The responsible face(s) of maximal dimension is/are:
         tau1: dim 0,  vertices = [(2, 1)],  rays = []
        generators of cone = [(0, 1), (1, 1)], partition into simplicial
cones = [[(0, 1), (1, 1)]]

         tau0: dim 0,  vertices = [(1, 2)],  rays = []
        generators of cone = [(1, 0), (1, 1)], partition into simplicial
cones = [[(1, 0), (1, 1)]]
```

```
zex15.topological_zeta(local = True, info = True)
```

```
tau0: dim 0,  vertices = [(1, 2)],  rays = []
generators of cone = [(1, 0), (1, 1)], partition into simplicial
cones = [[(1, 0), (1, 1)]]
multiplicities = [1], integral points = [[(0, 0)]]
J_tau = 1/((s + 1)*(3*s + 2)) , dim_tau!*Vol(tau) = 1


tau1: dim 0,  vertices = [(2, 1)],  rays = []
generators of cone = [(0, 1), (1, 1)], partition into simplicial
cones = [[(0, 1), (1, 1)]]
multiplicities = [1], integral points = [[(0, 0)]]
J_tau = 1/((s + 1)*(3*s + 2)) , dim_tau!*Vol(tau) = 1
```

```
tau4: dim 1,  vertices = [(1, 2), (2, 1)],  rays = []
generators of cone = [(1, 1)], partition into simplicial cones =
[[(1, 1)]]
multiplicities = [1], integral points = [[(0, 0)]]
J_tau = 1/(3*s + 2) , dim_tau!*Vol(tau) = 1


-(s - 2)/((s + 1)*(3*s + 2))
```

Example 19: $x_1x_2x_3^2x_4 + x_1x_2^2x_3x_4 + x_1^2x_2x_3x_4^2$

```
T.<x_1,x_2,x_3,x_4> = QQ[]
zex19 = ZetaFunctions(x_1*x_2*x_3^2*x_4 + x_1*x_2^2*x_3*x_4 + x_1^2*x_2*x_3*x_4^2)
```

```
zex19.give_info_newton()
```

```
Newton's polyhedron of x_1^2*x_2*x_3*x_4^2 + x_1*x_2^2*x_3*x_4 +
x_1*x_2*x_3^2*x_4:
        support points = [(2, 1, 1, 2), (1, 2, 1, 1), (1, 1, 2, 1)]
        vertices = [(1, 1, 2, 1), (1, 2, 1, 1), (2, 1, 1, 2)]
        number of proper faces = 33
        Facet 1: x_2 - 1 >= 0
        Facet 2: x_3 - 1 >= 0
        Facet 3: x_1 - 1 >= 0
        Facet 4: x_4 - 1 >= 0
        Facet 5: x_2 + x_3 + x_4 - 4 >= 0
        Facet 6: x_1 + x_2 + x_3 - 4 >= 0
```

```
zex19.give_expected_pole_info()
```

```
The candidate poles of the (local) topological zeta function (with d
= 1) of x_1^2*x_2*x_3*x_4^2 + x_1*x_2^2*x_3*x_4 + x_1*x_2*x_3^2*x_4
in function of s are:

-3/4 with expected order: 2
The responsible face of maximal dimension is ``tau_0`` = minimal
face who intersects with the diagonal of ambient space:
         tau26: dim 2,  vertices = [(1, 1, 2, 1), (1, 2, 1, 1), (2, 1, 1,
2)],  rays = []
        generators of cone = [(0, 1, 1, 1), (1, 1, 1, 0)], partition into
simplicial cones = [[(0, 1, 1, 1), (1, 1, 1, 0)]]

-1 with expected order: 3
The responsible face(s) of maximal dimension is/are:
         tau5: dim 1,  vertices = [(1, 1, 2, 1)],  rays = [(0, 0, 1, 0)]
        generators of cone = [(0, 1, 0, 0), (1, 0, 0, 0), (0, 0, 0, 1)],
partition into simplicial cones = [[(0, 1, 0, 0), (1, 0, 0, 0), (0,
0, 0, 1)]]

         tau9: dim 1,  vertices = [(1, 2, 1, 1)],  rays = [(0, 1, 0, 0)]
        generators of cone = [(0, 0, 1, 0), (1, 0, 0, 0), (0, 0, 0, 1)],
partition into simplicial cones = [[(0, 0, 1, 0), (1, 0, 0, 0), (0,
0, 0, 1)]]
```

```
zex19.topological_zeta()
```

```
(s^3 - 5*s^2 + 6*s + 9)/((s + 1)^3*(4*s + 3)^2)
```

Example 21: $x_1^2 + x_2^3x_4^3 + x_3^3x_5^3$

```
T2.<x_1,x_2,x_3,x_4,x_5> = QQ[]
zex21 = ZetaFunctions(x_1^2 + x_2^3*x_4^3 + x_3^3*x_5^3)
```

```
zex21.give_info_newton()
```

```
Newton's polyhedron of x_2^3*x_4^3 + x_3^3*x_5^3 + x_1^2:
```

```
        support points = [(0, 3, 0, 3, 0), (0, 0, 3, 0, 3), (2, 0, 0, 0,
0)]
        vertices = [(0, 0, 3, 0, 3), (0, 3, 0, 3, 0), (2, 0, 0, 0, 0)]
        number of proper faces = 85
        Facet 1: x_2 >= 0
        Facet 2: x_4 >= 0
        Facet 3: x_3 >= 0
        Facet 4: x_5 >= 0
        Facet 5: x_1 >= 0
        Facet 6: 3*x_1 + 2*x_3 + 2*x_4 - 6 >= 0
        Facet 7: 3*x_1 + 2*x_4 + 2*x_5 - 6 >= 0
        Facet 8: 3*x_1 + 2*x_2 + 2*x_5 - 6 >= 0
        Facet 9: 3*x_1 + 2*x_2 + 2*x_3 - 6 >= 0
```

```
zex21.give_expected_pole_info()
```

```
The candidate poles of the (local) topological zeta function (with d
= 1) of x_2^3*x_4^3 + x_3^3*x_5^3 + x_1^2 in function of s are:

-7/6 with expected order: 3
The responsible face of maximal dimension is ``tau_0`` = minimal
face who intersects with the diagonal of ambient space:
         tau57: dim 2,  vertices = [(0, 0, 3, 0, 3), (0, 3, 0, 3, 0), (2,
0, 0, 0, 0)],  rays = []
        generators of cone = [(3, 0, 2, 2, 0), (3, 0, 0, 2, 2), (3, 2, 0,
0, 2), (3, 2, 2, 0, 0)], partition into simplicial cones = [[(3, 0,
2, 2, 0), (3, 2, 0, 0, 2)], [(3, 0, 0, 2, 2), (3, 2, 0, 0, 2), (3,
0, 2, 2, 0)], [(3, 2, 0, 0, 2), (3, 2, 2, 0, 0), (3, 0, 2, 2, 0)]]

-1 with expected order: 1
(If all Vol(tau) are 0, where tau runs through the selected faces
that are no vertices, then the expected order of -1 is 0).
```

```
zex21.topological_zeta()
```

```
(108*s^3 + 456*s^2 + 647*s + 343)/((s + 1)*(6*s + 7)^3)
```

Examples for the **Monodromy Zeta Function at the origin**

```
zexmon1 = ZetaFunctions(R2(y^7+x^2*y^5+x^5*y^3))
zexmon1.monodromy_zeta(char = True)
```

```
The characteristic polynomial of the monodromy is (T - 1)^3*(T^6 +
T^5 + T^4 + T^3 + T^2 + T + 1)*(T^18 + T^17 + T^16 + T^15 + T^14 +
T^13 + T^12 + T^11 + T^10 + T^9 + T^8 + T^7 + T^6 + T^5 + T^4 + T^3
+ T^2 + T + 1)

1/((t^7 - 1)*(t^19 - 1))
```

```
zexmon2 = ZetaFunctions(R(x*y + z^3))
zexmon2.monodromy_zeta(char = True)
```

```
The characteristic polynomial of the monodromy is T^2 + T + 1

-t^3 + 1
```

```
zexmon3 = ZetaFunctions(R((3*x+5*z)*(x+2*z)+y^3))
zexmon3.monodromy_zeta(char = True)
```

```
The characteristic polynomial of the monodromy is T^2 + T + 1

-t^3 + 1
```

```
zexmon4 = ZetaFunctions(R(x*(y+x)+x^2*z+z^3))
zexmon4.monodromy_zeta(char = True)
```

```
The characteristic polynomial of the monodromy is T^2 + T + 1

-t^3 + 1
```

```
zexmon4 = ZetaFunctions(R(x*y*(x+y)+z^4))
zexmon4.monodromy_zeta(char = True)
```

```
The characteristic polynomial of the monodromy is (T + 1)^2*(T^2 +
1)^2*(T^2 - T + 1)*(T^4 - T^2 + 1)
```

```
-(t^4 - 1)*(t^12 - 1)/(t^3 - 1)
```