Primes_part01

May 5, 2019

This documents shows how the results of the first intersections looks like.

```
In [1]: from sage.repl.ipython_kernel.install import SageKernelSpec
        #SageKernelSpec.identifier()
```

```
In [2]: # var('x1','x2','z12','D1','D2')
        n_not_12(x1,x2,z12,D1,D2) = (2*x1 + 1)*(2*x2 + 1)*z12 + (-D2*(2*x1 + 1)*x2 + D1*(2*x2
        n_not_12
```

```
Out[2]: (x1, x2, z12, D1, D2) |--> D1*x1*(2*x2 + 1) - D2*(2*x1 + 1)*x2 + (2*x1 + 1)*(2*x2 + 1)
```

```
In [3]: x12 = ((2*x1 + 1)*(2*x2 + 1) - 1)*(1/2)
        x12
```

```
Out[3]: 1/2*(2*x1 + 1)*(2*x2 + 1) - 1/2
```

```
In [4]: D12 = (n_not_12 - (2*x1 + 1)*(2*x2 + 1)*z12)*(-1)
        D12
```

```
Out[4]: (x1, x2, z12, D1, D2) |--> -D1*x1*(2*x2 + 1) + D2*(2*x1 + 1)*x2 + 2*D1*(x1 - x2)
```

```
In [5]: var('x3','z123','D3')
        n_not_123 = n_not_12(x12,x3,z123,D12,D3)
        n_not_123
```

```
Out[5]: -D3*(2*x1 + 1)*(2*x2 + 1)*x3 + (2*x1 + 1)*(2*x2 + 1)*(2*x3 + 1)*z123 - 1/2*(D1*x1*(2*x
```

```
In [6]: # Notation: Be z_12 = z2
        var('x_1','x_2','z_2','D_1','D_2')
        n_not_12(x_1,x_2,z_12,D_1,D_2) = (2*x_1 + 1)*(2*x_2 + 1)*z_2 + (-D_2*(2*x_1 + 1)*x_2 +

        @cached_function
        def n_not(n,f,x1,x2,z,D1,D2):
            if n == 2:
                print(f)
            else:
                n_not_12 = f

                x_1new = ((2*x1 + 1)*(2*x2 + 1) - 1)*(1/2)
                D_1new = (f - (2*x1 + 1)*(2*x2 + 1)*z)*(-1)

                #print(x_1new)

                x_new = var('x_%d' % n)
                z_new = var('z_%d' % n)
                D_new = var('D_%d' % n)

                x_2new = x_new
                z_new = z_new
                D_2new = D_new

                n_not(n-1,n_not_12(x_1new,x_2new,z_new,D_1new,D_2new),x_1new,x_2new,z_new,D_1ne

        result = n_not(3,n_not_12,x_1,x_2,z_2,D_1,D_2)
```

```
-D_3*(2*x_1 + 1)*(2*x_2 + 1)*x_3 + (2*x_1 + 1)*(2*x_2 + 1)*(2*x_3 + 1)*z_2 - 1/2*(D_1*x_1*(2*x_
```

```
In [7]: result_2 = D_1*x_1*(2*x_2 + 1) - D_2*(2*x_1 + 1)*x_2 + (2*x_1 + 1)*(2*x_2 + 1)*z_2 - 2
        result_2

Out[7]: D_1*x_1*(2*x_2 + 1) - D_2*(2*x_1 + 1)*x_2 + (2*x_1 + 1)*(2*x_2 + 1)*z_2 - 2*D_1*(x_1 -

In [8]: result_2.full_simplify()

Out[8]: -D_1*x_1 + (2*(D_1 - D_2)*x_1 + 2*D_1 - D_2)*x_2 + (2*(2*x_1 + 1)*x_2 + 2*x_1 + 1)*z_2

In [9]: result_3 = (2*D_1*x_1*(2*x_2 + 1) - 2*D_2*(2*x_1 + 1)*x_2 - 4*D_1*(x_1 - x_2) - 1)*D_3
        result_3

Out[9]: (2*D_1*x_1*(2*x_2 + 1) - 2*D_2*(2*x_1 + 1)*x_2 - 4*D_1*(x_1 - x_2) - 1)*D_3*(2*z_3 + 1

In [10]: result_3.full_simplify()

Out[10]: -2*D_1*D_3*x_1 + 2*D_1*x_1^2 - 2*(4*(D_1 - D_2)*x_1^2 + 2*(3*D_1 - 2*D_2)*x_1 + 2*D_1
```