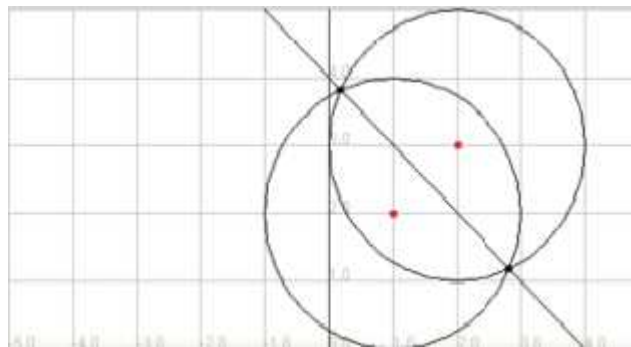


Arbitrary Algebras using GAALOP

May 8th, 2019

Dr.-Ing. Dietmar Hildenbrand

TU Darmstadt, Germany



Example: Conic Algebra GAC

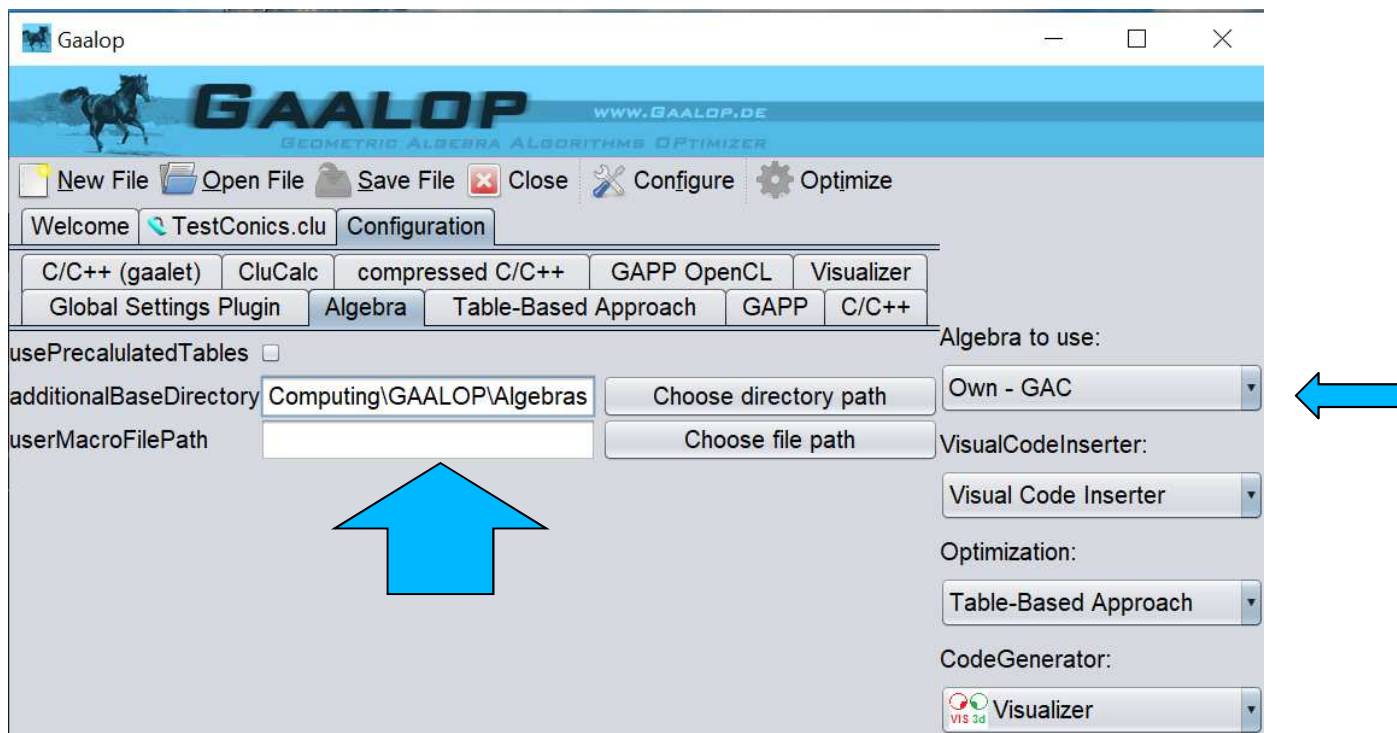
- 8-dim GA for the computation with conics
 - Reference
[Geometric Algebra for Conics,
Jaroslav Hrdina, Ales Navrat and Petr Vasik,
Advances in Applied Clifford Algebras, 2018]
-

Arbitrary Algebras using GAALOP

- No fixed limit for the dimension of the algebra
 - Note: the multivector of an n-dimensional GA is 2^n dimensional!
 - Support of
 - predefined multiplication tables or
 - on-the-fly computation
 - 2d/3d-Visualization
 - Integration based on 2 files
 - Definition.csv
 - Macros.clu
 - In a specific directory for the own algebras (to be configured in GAALOP)
-

Integration of Conic Algebra

Add Directory GAC in the additionalBaseDirectory!



and include two specific files for the algebra (definition.csv and macros.clu)

Definition.csv

5 lines in order to define basis vectors ...

1,e1,e2,e0p,e0m,e0c,einfp,einfm,einfc

ep1=0.5*einfp-e0p,em1=0.5*einfp+e0p,ep2=0.5*einfm-
e0m,em2=0.5*einfm+e0m,ep3=0.5*einfc-e0c,em3=0.5*einfc+e0c

1,e1,e2,ep1,ep2,ep3,em1,em2,em3

e1=1,e2=1,ep1=1,ep2=1,ep3=1,em1=-1,em2=-1,em3=-1

e0p=0.5*em1-0.5*ep1,einfp=em1+ep1,e0m=0.5*em2-
0.5*ep2,einfm=em2+ep2,e0c=0.5*em3-0.5*ep3,einfc=em3+ep3

Macros.clu

For basic functionality ...

```
createPoint = { cz = _P(3); e0p + _P(1)*e1 + _P(2)*e2 + 0.5*(_P(1)*_P(1) +  
_P(2)*_P(2))*einfp + 0.5*(_P(1)*_P(1) - _P(2)*_P(2))*einfm + _P(1)*_P(2)*einfc}
```

```
// Define the dual operator * that dualizes A as (*A).
```

```
Dual = { _P(1)/(e0p^e0m^e0c^e1^e2^einfp^einfm^einfc)}
```

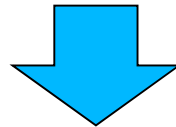
```
...
```

Integration of Conic Algebra

2. Geometric Algebra for Conics

The idea of C. Perwass is to generalize the concept of (two-dimensional) conformal geometric algebra $\mathbb{G}_{3,1}$. In the usual basis \bar{n}, e_1, e_2, n , embedding of a plane in $\mathbb{G}_{3,1}$ is given by

$$(x, y) \mapsto \bar{n} + xe_1 + ye_2 + \frac{1}{2}(x^2 + y^2)n.$$



$$C(x, y) = \bar{n}_+ + xe_1 + ye_2 + \frac{1}{2}(x^2 + y^2)n_+ + \frac{1}{2}(x^2 - y^2)n_- + xyn_\times. \quad (2)$$

Integration of Conic Algebra

$\bar{n}_+, \bar{n}_-, \bar{n}_\times, e_1, e_2, n_+, n_-, n_\times.$
e0p e0m e0c e1 e2 einfpm einfcm

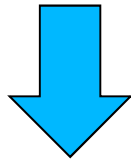
Integration of Conic Algebra

$\bar{n}_+, \bar{n}_-, \bar{n}_\times, e_1, e_2, n_+, n_-, n_\times.$
e0p e0m e0c e1 e2 einfpm einfcm einfcp

Integration of Conic Algebra

Point

$$C(x, y) = \bar{n}_+ + xe_1 + ye_2 + \frac{1}{2}(x^2 + y^2)n_+ + \frac{1}{2}(x^2 - y^2)n_- + xyn_x. \quad (2)$$

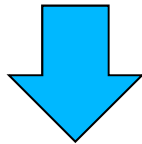


```
createPoint = {e0p + _P(1)*e1 + _P(2)*e2 + 0.5*(_P(1)*_P(1) +  
_P(2)*_P(2))*einfp + 0.5*(_P(1)*_P(1) - _P(2)*_P(2))*einfm +  
_P(1)*_P(2)*einfc + _P(3)*0}
```

Integration of Conic Algebra

Ellipse

$$\begin{aligned}E_I &= (a^2 + b^2)\bar{n}_+ + (a^2 - b^2)\bar{n}_- - a^2b^2n_+, \\H_I &= (a^2 + b^2)\bar{n}_+ + (a^2 - b^2)\bar{n}_- + a^2b^2n_+, \\P_I &= \bar{n}_+ + \bar{n}_- + pe_2.\end{aligned}$$



$$\text{Ellipse} = \{ (_P(1)*_P(1) + _P(2)*_P(2))*e_0p - (_P(1)*_P(1) - _P(2)*_P(2))*e_0m - (_P(1)*_P(1)*_P(2)*_P(2))*e_{infp} \}$$

Gaalop Visualization

The screenshot displays the Gaalop software interface, which is used for visualizing mathematical models. The interface is divided into several sections:

- Settings Panel (Left):** Contains various configuration options:
 - Automatic Rendering
 - point size: 0.2
 - epsilon: 1E-4
 - cubeEdgeLength: 5
 - density: 1
 - max_n: 50
- Visible Objects Panel (Left):** Lists objects to be rendered with checkboxes:
 - H
 - E
 - P
 - C
- Main Editor (Center):** Displays a code file named "TestConics.clu*" with the following content:

```
// GAC G(5,3) test file
// maybe uncomment one block at a time to test

// Test basic curves

//:Red::PNT = createPoint(1,0,0);
:Blue::E = Ellipse(1,2);
:Green::C = Circle(0,0,2);
:Cyan::P = Parabola(1);
:White::H = Hyperbola(1,2);

// Test some GAC versors
// Note, rotor is slow; expect to wait some.
/*
:White::TXE = TranslatorX(3)*E*~TranslatorX(3);
:Cyan::TXYE = TranslatorY(3)*TXE*~TranslatorY(3);
:Magenta::TXYRE = Rotor(45)*TXYE*~Rotor(45);
*/
```
- Visualization Window (Right):** Shows a 3D rendering of the defined curves. The scene features a grid of points forming a paraboloid-like shape, with several colored curves overlaid: a blue ellipse, a green circle, a cyan parabola, and a white hyperbola. The background is a dark blue sky with white clouds.

GAALOP -> Implicit Functions

```
P=createPoint(x,y,z);  
S=e0-0.5*r*r*einfl;  
?I=S.P;
```



GAALOP -> Implicit Functions

```
P=createPoint(x,y,z);  
S=e0-0.5*r*r*ein;f;  
?I=S.P;
```

```
I[0] = -z * z - y * y - x * x + r * r; // 1.0
```



GAALOP -> Implicit Functions

```
P=createPoint(x,y,z);  
S=e0-0.5*r*r*einf;  
?I=S.P;
```

```
I[0] = -z * z - y * y - x * x + r * r; // 1.0
```

Sphere around the origin with radius r

GAALOP -> arbitrary visualizations

```
P=createPoint(x,y,z);  
S=e0-0.5*r*r*einr;  
?I=S.P;
```

```
I[0] = -z * z - y * y - x * x + r * r; // 1.0
```

How can this be directly visualized in Python?

GAALOP -> arbitrary visualizations

SymPy 1.3 documentation » SymPy Modules Reference »

[previous](#)



Table of Contents

- Plotting Module
 - Introduction
 - Plot Class
 - Plotting Function
- Reference
 - Series Classes
- Pyglet Plotting Module
 - Plot Window Controls
 - Coordinate Modes
 - Specifying Intervals for

Plotting Module

Introduction

The plotting module allows you to make 2-dimensional and 3-dimensional plots. Presently the plots are rendered using `matplotlib` as a backend. It is also possible to plot 2-dimensional plots using a `TextBackend` if you don't have `matplotlib`.

The plotting module has the following functions:

- `plot`: Plots 2D line plots.
- `plot_parametric`: Plots 2D parametric plots.
- `plot_implicit`: Plots 2D implicit and region plots.
- `plot3d`: Plots 3D plots of functions in two variables.
- `plot3d_parametric_line`: Plots 3D line plots, defined by a parameter.
- `plot3d_parametric_surface`: Plots 3D parametric surface plots.

Thanks a lot

