

März 2023

Computeralgebra Rundbrief

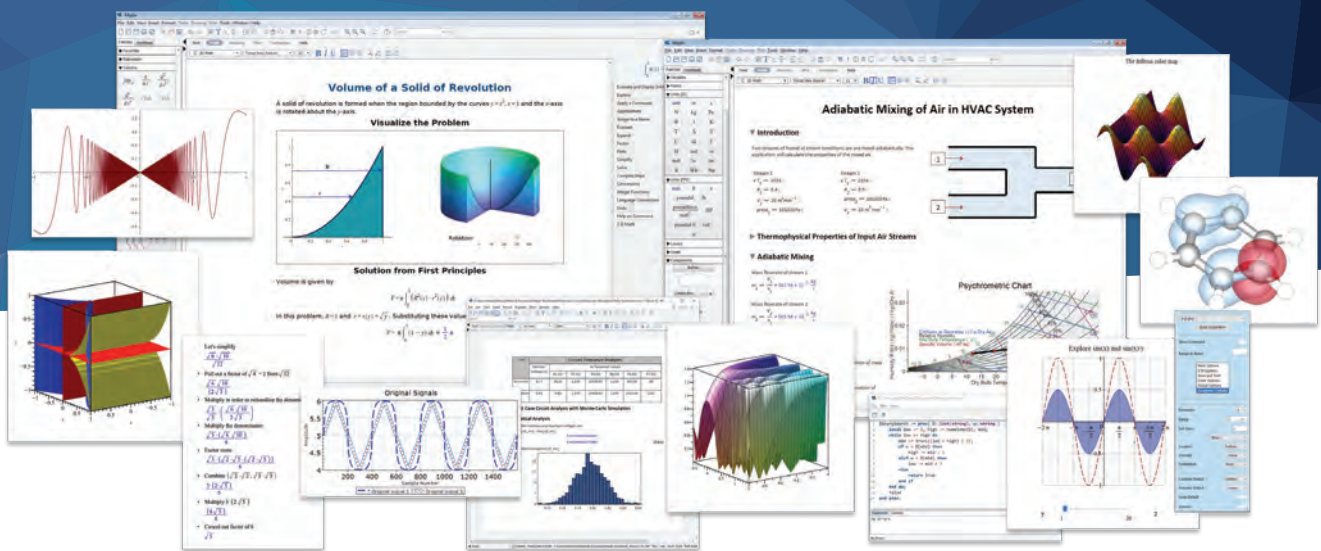
> Ausgabe 72

- ▶ Tagung der Fachgruppe 2023 in Hannover
- ▶ Hands-on Tropical Geometry
- ▶ OSCAR: An introduction
- ▶ Toric Geometry in OSCAR



Mehr leisten mit Maple 2023!

Lösen Sie mehr Probleme noch leichter mit Maple 2023



Neues Maple 2023 jetzt verfügbar!

Die leistungstärkste und umfassendste Umgebung zum Erforschen, Visualisieren und Lösen selbst der schwierigsten mathematischen Probleme ist jetzt noch besser geworden!

Probieren Sie Maple kostenlos für 15 Tage ohne Verpflichtungen
www.maplesoft.com/CAR2023



Inhaltsverzeichnis

Inhalt	3
Impressum	4
Mitteilungen der Sprecher	5
Tagungen der Fachgruppe	7
Themen und Anwendungen	10
<i>Hands-on Tropical Geometry</i> (H. Gangl, Y. Ren, Z. Urbancic)	10
Neues über Systeme	16
<i>OSCAR: An introduction</i> (M. Horn)	16
<i>Toric Geometry in OSCAR</i> (M. Bies, L. Kastner)	20
Computeralgebra und Industrie	26
<i>Entropie und Kryptographie</i> (K. Oeztas, C. Tobias, M. Firoozi)	26
Publikationen über Computeralgebra	30
Besprechungen zu Büchern der Computeralgebra	31
<i>Gregory V. Bard: Sage for Undergraduates, 2. Auflage</i> (M. Kreuzer)	31
Promotionen in der Computeralgebra	32
Hinweise auf Konferenzen	33
Fachgruppenleitung Computeralgebra 2023–2026	36

Impressum

Der Computeralgebra-Rundbrief wird herausgegeben von der Fachgruppe Computeralgebra der GI in Kooperation mit der DMV und der GAMM
(verantwortlicher Redakteur: Dr. Fabian Reimers car@mathematik.de)

Der Computeralgebra-Rundbrief erscheint halbjährlich, Redaktionsschluss 15.02. und 15.09. ISSN 0933-5994. Mitglieder der Fachgruppe Computeralgebra erhalten je ein Exemplar dieses Rundbriefs im Rahmen ihrer Mitgliedschaft. Fachgruppe Computeralgebra im Internet:
<https://www.fachgruppe-computeralgebra.de>.

Konferenzankündigungen, Mitteilungen, einzurichtende Links, Manuskripte und Anzeigenwünsche bitte an den verantwortlichen Redakteur.

GI (Gesellschaft für
Informatik e.V.)
Wissenschaftszentrum
Ahrstr. 45
53175 Bonn
Telefon 0228-302-145
Telefax 0228-302-167
bonn@gi.de
<https://gi.de>



DMV (Deutsche Mathematiker-
Vereinigung e.V.)
Mohrenstraße 39
10117 Berlin
Telefon 030-20377-306
Telefax 030-20377-307
dmv@wias-berlin.de
<https://www.mathematik.de>



GAMM (Gesellschaft für Angewandte
Mathematik und Mechanik e.V.)
Technische Universität Dresden
Institut für Statik und Dynamik der
Tragwerke
01062 Dresden
Telefon 0351-463-33448
Telefax 0351-463-37086
GAMM@mailbox.tu-dresden.de
<https://www.gamm-ev.de>



Mitteilungen der Sprecher

Liebe Mitglieder der Fachgruppe Computeralgebra,

wie sicherlich kaum jemandem entgangen ist, fand Ende letzten Jahres turnusgemäß die Wahl der Fachgruppenleitung statt. Nicht zuletzt durch das elektronische Wahlverfahren, das nach dem Testlauf 2019 nun zum Standard wurde, konnten die Wahlleiter Florian Heß und Eva Zerz auch diesmal eine erfreulich hohe Wahlbeteiligung vermelden: 134 Mitglieder gaben ihre Stimme ab, wobei jeder bis zu 9 Personen auswählen konnte. Dabei ergab sich folgendes Ergebnis:

Prof. Dr. Anne Frühbis-Krüger	Oldenburg	111
Prof. Dr. Gregor Kemper	München	104
Prof. Dr. Michael Cuntz	Hannover	102
Prof. Dr. Claus Fieker	Kaiserslautern	101
Xenia Bogomolec	Hannover	95
Dr. Fabian Reimers	München	89
Prof. Dr. Tommy Hofmann	Siegen	88
Dr. Thomas Hahn	München	86
Prof. Dr. Martin Kreuzer	Passau	82
Prof. Dr. Max Horn	Kaiserslautern	81
Prof. Dr. Jürgen Klüners	Paderborn	80

Die ersten 9 Kandidaten sind damit direkt gewählt. Wir danken allen Kandidaten für ihre Bereitschaft, sich zur Wahl zu stellen, den Wahlleitern Florian Heß und Eva Zerz für die Durchführung und Auswertung der Wahl sowie Thomas Hahn für die technische Unterstützung des elektronischen Verfahrens. Jürgen Klüners, der viele Jahre lang der Fachgruppenleitung angehörte und nun ausscheidet, möchten wir für sein Engagement und die geleistete Arbeit danken.

Neben den gewählten Mitgliedern gehören der Fachgruppe noch drei Vertreter der Muttergesellschaften sowie in der konstituierenden Sitzung am 27. Februar berufene Fachexperten an. Für die nächste Wahlperiode sind das:

Prof. Dr. Erika Abraham	Aachen	Vertreterin der GI
Prof. Dr. Florian Heß	Oldenburg	Vertreter der DMV
Prof. Dr. Eva Zerz	Aachen	Vertreterin der GAMM
Prof. Dr. Max Horn	Kaiserslautern	Fachexperte SFB

Die Position des Fachexperten Schule und Didaktik wird erst nachträglich besetzt werden, da uns bis jetzt noch keine Zusage der angesprochenen Person vorliegt. In der konstituierenden Sitzung erfolgte ebenfalls die Wahl des Sprecherteams für diese Wahlperiode. Da Gregor Kemper nach drei Wahlperioden das Sprecherteam auf eigenen Wunsch verläßt, gab es hier einen personellen Wechsel: Während Anne Frühbis-Krüger weiterhin die Aufgaben der Sprecherin wahrnehmen wird, ist Michael Cuntz zum neuen stellvertretenden Sprecher gewählt worden.

Blicken wir nun auf die kommenden drei Jahre, so erwartet uns als erstes die Tagung der Fachgruppe in Hannover in der Pfingstwoche, eine ausführliche Ankündigung findet sich auf Seite 7. Wir hoffen, dass sie diesmal wieder in gewohntem Präsenzformat stattfinden kann, das ja beim letzten Mal durch Corona im letzten Moment ausgebremst worden war. Ebenso hoffen wir, dass unsere Workshopförderung nun wieder Interessenten findet; die Ausschreibung dazu befindet sich im Kasten auf Seite 35.

Ganz neu wird es ab 2024 einen Promotionspreis der Fachgruppe geben, der jährlich eine herausragende Promotion in Computeralgebra auszeichnen wird und mit 500,- Euro dotiert ist. Vorschlagbar sind Promotionen, die an einer wissenschaftlichen Einrichtung im deutschsprachigen Raum bzw. unter Betreuung eines Mitglieds einer solchen Einrichtung angefertigt wurden. Sowohl ein Vorschlag durch den Betreuer als auch ein Selbstvorschlag sind möglich, die Promotion sollte im Abschnitt Promotionen des Rundbriefs vermeldet sein. Die preisgekrönte Arbeit erhält dann in der folgenden Ausgabe des Rundbriefs die Möglichkeit einer detaillierteren Vorstellung der Ergebnisse in Form eines kurzen Artikels.

Nach derart vielen Ankündigungen und Verlautbarungen in eigener Sache wird es jetzt aber höchste Zeit, Sie nicht länger von der Lektüre des Heftes abzuhalten, das Sie auf Seite 10 diesmal in die tropische Geometrie mit POLYMAKE entführen will. Gleich zwei Artikel der Rubrik 'Neues über Systeme' behandeln ganz unterschiedliche Aspekte des im Rahmen des SFB/TRR 195 entstehenden Computeralgebra-Systems OSCAR, einmal eine etwas gruppentheoriellastige Einführung ab Seite 16 und dann ein Einblick in die Funktionalität für torische Geometrie ab Seite 20.

Wir wünschen Ihnen eine kurzweilige und anregende Lektüre.

Anne Frühbis-Krüger

Michael Cuntz

Tagung der Fachgruppe Computeralgebra

Hannover

31.05.–02.06.2023

<https://konferenz.uni-hannover.de/event/83>

Schon zum 10. Mal richtet die Fachgruppe in der Pfingstwoche 2023 eine Computeralgebra-Tagung aus. Nachdem und gerade weil pandemiebedingt bei der vorigen Tagung in München nicht einmal die Verschiebung um ein Jahr von 2021 auf 2022 eine Durchführung in Präsenz ermöglichte, liegt diesmal zwischen der vergangenen und der kommenden nur etwas mehr als ein Jahr. Wie schon in München sie am Mittwoch Mittag beginnen und am Freitag Mittag enden. Angesichts der zeitlichen Lage planen wir eine reine Präsenztagung.



Ganz in der Tradition der früheren Tagungen werden auch diesmal mehrere Hauptvortragende Übersichtsvorträge über wichtige Themen aus Computeralgebra und über Computeralgebrasysteme halten, während in den anderen Vorträgen dem wissenschaftliche Nachwuchs Gelegenheit gegeben wird, seine Ergebnisse vorzustellen. Deutsch und Englisch sind dabei wie immer gleichberechtigte Konferenzsprachen. Für den besten Nachwuchs-Vortrag vergibt die Fachgruppe auch dieses Mal wieder einen mit 500,- Euro dotierten Preis.

Hierzu sind Nachwuchswissenschaftler und -wissenschaftlerinnen (**Promovendi, Post-Docs**) aufgefordert, sich bis zum **16.04.2023** mit einem **Vortrag anzumelden**.

Als Hauptvortragende konnten wir folgende Wissenschaftlerinnen und Wissenschaftler gewinnen:

- **Timo Keller** (Rijksuniversiteit Groningen)
- **Marta Panizzut** (TU Berlin)
- **Raman Sanyal**
- **Mima Stanojkovski** (RWTH Aachen)
- **Ulrich Thiel** (TU Kaiserslautern)



Welfenschloss Hannover, Foto: M.Cuntz.

Website: <https://konferenz.uni-hannover.de/event/83>

Termin und Ort: Die Tagung findet in der Zeit vom 31. Mai – 02. Juni 2023 im Hauptgebäude der Leibniz Universität Hannover statt. Sie wird am 31. Mai 2023 um circa 13:00 Uhr eröffnet (Anreisetag) und endet am 02. Juni 2023 um circa 12:30 Uhr (Abreisetag).

Anmeldung: Die Anmeldung eines Vortrags ist bis 16. April 2023 möglich. Die Anmeldung ohne Vortrag ist bis 10. Mai 2023 möglich. Details zur Anmeldung finden Sie auf der Website der Tagung.

Konferenzgebühren: Jedes Nichtmitglied der Fachgruppe entrichtet vor Ort einen Unkostenbeitrag in Höhe von 20 € für die Kaffeepausen, alternativ kann man vor Ort zum Jahresbeitrag von 9 € Mitglied der Fachgruppe werden.

Nachwuchspreis: Die Fachgruppe Computeralgebra vergibt für den besten Vortrag eines Nachwuchswissenschaftlers wieder einen mit 500 € dotierten Nachwuchspreis. Verbunden mit dem Geldpreis ist die Einladung auf der nächsten Tagung der Fachgruppe einen Hauptvortrag zu halten.

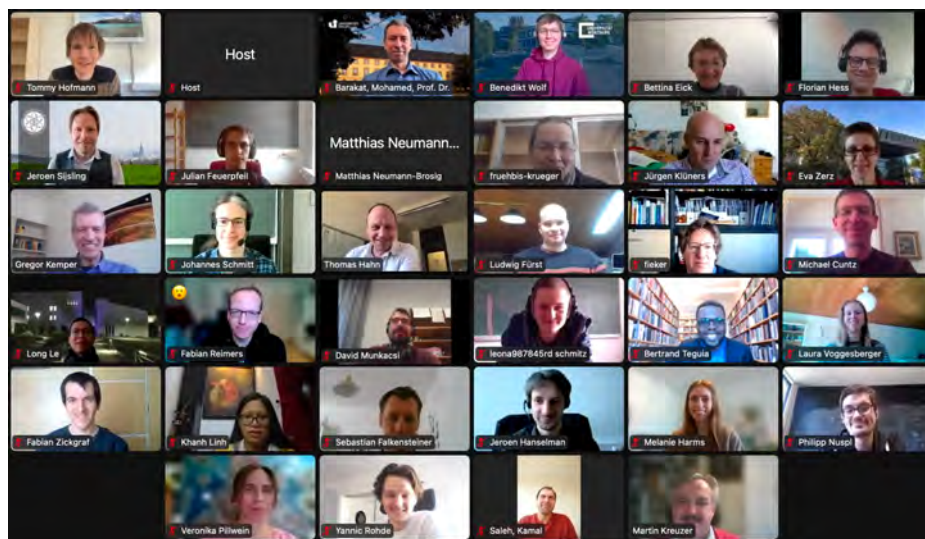


Foto der letzten Tagung 2022 in München (leider online)



Dissertationspreis der Fachgruppe Computeralgebra:

Die Fachgruppe Computeralgebra möchte herausragende Dissertationen im Themenbereich der Computeralgebra durch die Vergabe eines Dissertationspreises würdigen. Die Ausschreibung erfolgt zum ersten Mal im Jahr 2024 und danach jährlich, jeweils mit der Einreichungsfrist 1. April.

Eingereicht werden können deutsch- oder englischsprachige Dissertationen, die innerhalb von 12 Monaten vor der Einreichungsfrist verteidigt und veröffentlicht wurden. Zugelassen sind Dissertationen aus dem deutschsprachigen Raum, mit einem betreuenden Institut aus Deutschland, Österreich oder der Schweiz. Das Thema der Dissertation soll einen klaren Bezug zur Computeralgebra (Theorie, Algorithmen oder Implementierung) aufweisen.

Einreichungen können entweder als Eigenbewerbung oder als Nominierung durch die wissenschaftlichen Betreuerinnen und Betreuer per E-Mail an die Fachgruppe Computeralgebra

`ca-promotionspreis@mathematik.de`

erfolgen.

Einzureichen sind in elektronischer Form die Dissertation, eine Kurzfassung (max. 1/2 Seite), akademischer Werdegang mit Publikationsliste und optional ein Empfehlungsschreiben.

Der Dissertationspreis ist mit 500 Euro dotiert. Die Kurzfassungen aller eingereichten Dissertationen werden im Rundbrief der Fachgruppe Computeralgebra veröffentlicht.



Hands-on Tropical Geometry

H. Gangl, Durham University

Y. Ren, Durham University

Z. Urbancic, Durham University

herbert.gangl@durham.ac.uk,

yue.ren2@durham.ac.uk,

ziva.urbancic@durham.ac.uk



Introduction

Tropical varieties are piecewise linear structures that arise in many areas inside and outside mathematics. They describe the loci of indifference prices in product mix auctions [1], spaces of phylogenetic trees [11, 15], or images of solutions of polynomial systems under component-wise valuation [9]. Geometrically, tropical varieties are strongly tied to their classical algebro-geometric counterparts, they are strikingly similar in some aspects yet also fascinatingly different in others. However, while there has been a significant effort in modelling algebraic surfaces and curves - from Schilling's white plaster casts popularized by Klein in the 19th century [5, 14] to 3D-printed models of today [10] [3] - nothing comparable exists for tropical varieties despite the simplicity of their polyhedral nature.

This article aims to address this gap by presenting a comprehensive guide on how to create 3D-printable models of tropical surfaces, tropical curves, and combinations thereof. We will make use of the following open-source software, which are freely available on their respective websites for all platforms:

POLYMAKE [4] (<https://polymake.org>) for creating mathematically accurate tropical models.

OPENSCAD [12] (<https://openscad.org>) for solidifying the tropical models.

We will only focus on the creation of 3D models in OPENSCAD, which in turn is capable of exporting to a variety of file formats common in 3D printing such as STL. Exporting color is also supported by 3rd party scripts such as COLORSCAD. The actual 3D printing process depends significantly on the printer, the material, and the associated printing software used.

All scripts and templates in this report can be found in the official POLYMAKE repository and will be part of the official POLYMAKE distribution from version 4.6 onward. The notebook is also publicly available under

https://polymake.org/doku.php/user_guide/tutorials/master/hands.on.tropical.geometry

Summary

This article assumes some familiarity with the basic objects in tropical geometry. If any terminology is unknown, refer to [9] and [7].

The starting point for creating a 3D-model is one or more polyhedral complexes in \mathbb{R}^3 . In POLYMAKE, objects of type `fan::PolyhedralComplex` can always be constructed manually by specifying their POINTS and INPUT_POLYTOPES. In special cases, which are covered in the upcoming sections, they can also be constructed via some shortcuts.

Given polyhedral complexes in POLYMAKE, the 3D-model can then be created in four steps:

1. Construct a bounding box in POLYMAKE. This can be done automatically using our script (which draws a box with a prescribed margin around all vertices) or manually by specifying any bounded polytope (which need not be a cuboid). Use “`->VISUAL`” to verify that the bounded box cuts off the polyhedral complex as desired.
2. Export the model from POLYMAKE to OPENSCAD using our script.
3. Adjust the thicknesses in OPENSCAD. Use the preview window to verify that the thickness is as desired.
4. Export the model from OPENSCAD to any 3D-printable file format.

In the following sections, we will specifically discuss the cases of

- a single tropical surface,
- a single tropical curve,
- a tropical surface containing a tropical curve.

The methods used can however be combined to cover arbitrary arrangements of surfaces and curves.

Modelling Tropical Surfaces

For an example of producing a 3D-printable model of a tropical surface see the file

3d_printing_template_surface.pl.

To run the example, simply copy its contents into any POLYMAKE session. In this section, we briefly explain some of its contents. Note that some variables are re-named in this article due to spacing.

Constructing a tropical surface in POLYMAKE

An easy way to construct a tropical surface in \mathbb{R}^3 of type `fan::PolyhedralComplex` in POLYMAKE is by constructing and converting a `tropical::Hypersurface`. This can be done either by specifying a tropical polynomial as a string, or by specifying an exponent matrix and a coefficient vector:

```
# Tropical quadratic surface via polynomial:
$f = toTropicalPolynomial("min(1+2*w,1+2*x,1+2*y,
                             1+2*z,w+x,w+y,w+z,
                             x+y,x+z,y+z)");

$T = new tropical::Hypersurface<Min>(POLYNOMIAL=>
                                     $f);

# Tropical quadratic surface via exponent matrix
# and coefficient vector:
$m = [[2,0,0,0], [1,1,0,0], [1,0,1,0], [1,0,0,1],
      [0,1,1,0], [0,1,0,1], [0,0,1,1], [0,2,0,0],
      [0,0,2,0], [0,0,0,2]];
$c = [1,0,0,0,0,0,0,1,1,1];
$T = new tropical::Hypersurface<Min>(MONOMIALS=>$m
                                     ,
                                     COEFFICIENTS=>$c);

# Converting Hypersurface to PolyhedralComplex
$T = new fan::PolyhedralComplex(
    VERTICES=>$T->VERTICES->minor(All,~[1]),
    MAXIMAL_POLYTOPES=>$T->MAXIMAL_POLYTOPES);

# Visualization:
$T->VISUAL;
```

Important: Since POLYMAKE uses homogeneous coordinates, i.e., it identifies polyhedra in \mathbb{R}^3 with cones in \mathbb{R}^4 , tropical polynomials need to be tetravariate and homogeneous instead of trivariate and inhomogeneous.

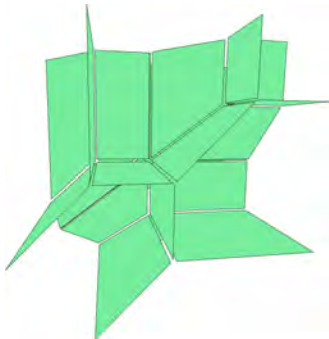


Figure 1: Tropical quadratic surface in POLYMAKE.

Bounding a tropical surface in POLYMAKE and exporting it to OPENSCAD

Constructing a bounding box can be done using the command `generateBoundingBox` or by specifying a custom `polytope`, which need not be a cuboid:

```
# Constructing bounding box automatically:
$bBox = generateBoundingBox($T);
# Constructing bounding box manually:
$bBox = scale(cube(3),2);
```

Note: `generateBoundingBox($T)` draws a box around all vertices of `$T` with a margin of 1, `generateBoundingBox($T,3,4,5)` draws a box around all vertices of `$T` with margins 3, 4, 5 in x -, y -, and z -direction, respectively.

The command `intersectWithBoundingBox` intersects the `fan::PolyhedralComplex` with the bounding box, `->VISUAL` visualizes their intersection:

```
# Constructing bounded PolyhedralComplex:
$Tbounded = intersectWithBoundingBox($T,$bBox);
$Tbounded->VISUAL;
```

The command `generateSCADFileForSurface` exports the tropical surface to OPENSCAD. It requires a bounded polyhedral complex and a filename. If the file already exists, it will be overwritten:

```
$filename = "foo.scad";
generateSCADFileForSurface($Tbounded,$filename);
```

Solidifying a tropical surface in OPENSCAD and exporting it for 3D-printing

To solidify the surface into a three-dimensional model, open the exported file in OPENSCAD. See Figure 2 for a preview of the model.

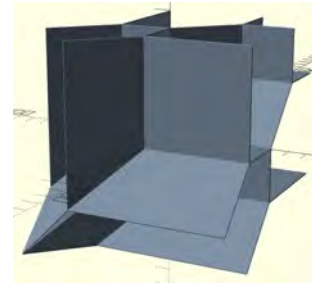


Figure 2: Solidified tropical quadratic surface in OPENSCAD.

The top of the file contains parameters which control the thickness of the model amongst other things:

```
colorSurface = "SlateGray"; // color of surface
scalingFactor = 1; // global scaling factor
thicknessSurface = 0.05; // thickness of surface
```

- `colorSurface` is the color of the surface in the render. It can be specified either by an HTML color name or by RGB value.
- `scalingFactor` is a factor by which the polyhedral complex is scaled. It can be used to scale the model to a desired size.
- `thicknessSurface` controls the thickness of the surface. OPENSCAD solidifies the surface by replacing every vertex with a ball with specified thickness and taking convex hulls.

Important: In order to 3D-print the tropical surface, every face needs to have a minimal thickness. As a rule of thumb for printing with PLA, we recommend a minimal thickness of 2mm for a print of size (10cm)³.

Tropical Curves

For an example of producing a 3D-printable model of a tropical curve see the file

3d_printing_template_curve.pl.

To run the example, simply copy its contents into any POLYMAKE session. In this section, we briefly explain some of its contents. Note that some variables are renamed due to spacing.

Constructing a tropical curve in POLYMAKE

An easy way to construct a tropical curve in \mathbb{R}^3 of type `fan::PolyhedralComplex` in POLYMAKE is by intersecting two tropical surfaces. Note however that not all tropical curves can be constructed this way.

```
# Tropical sextic curve that is the intersection
# of a tropical quadratic and cubic surface
$Mq = [[2,0,0,0], [0,2,0,0], [0,0,2,0], [0,0,0,2],
       [1,1,0,0], [1,0,1,0], [1,0,0,1],
       [0,1,1,0], [0,1,0,1], [0,0,1,1]];
$Cq = [1, -1/4, -2/4, -3/4, -3/4,
       -4/4, -5/4, 2/4, 0, -2/4];
$Tq = new tropical::Hypersurface<Min>(<
    MONOMIALS=>$Mq,
    COEFFICIENTS=>$Cq);
$Mc = [[3,0,0,0], [0,3,0,0], [0,0,3,0], [0,0,0,3],
       [1,1,1,0], [1,1,0,1], [1,0,1,1], [0,1,1,1],
       [2,1,0,0], [2,0,1,0], [2,0,0,1],
       [1,2,0,0], [1,0,2,0], [1,0,0,2],
       [0,2,1,0], [0,2,0,1], [0,1,2,0],
       [0,1,0,2], [0,0,2,1], [0,0,1,2]];
$Cc = [3,3,3,3,0,0,0,0,1,1,1,1,1,1,1,1,1,1];
$Tc = new tropical::Hypersurface<Min>(<
    MONOMIALS=>$Mc,
    COEFFICIENTS=>$Cc);
$Ts = tropical::intersect($Tq,$Tc);
# Converting Cycle to PolyhedralComplex
$Ts = new fan::PolyhedralComplex(
    VERTICES=>$Ts->VERTICES->minor(All,~[1]),
    MAXIMAL_POLYTOPES=>$Ts->MAXIMAL_POLYTOPES);
```

Important: `intersect` is the intersection of tropical cycles. In terms of polyhedral complexes, it therefore computes the stable intersection, not the set-theoretic intersection.

Alternatively and in particular for tropical curves which are not realizable, they can also be manually constructed as a one-dimensional polyhedral complex by specifying vertices and maximal polyhedra:

```
# constructing a tropical genus-2 cubic curve
$V = [[1,-1,-1,0], [1,-4,-3,0], [1,-8,-5,0],
      [1,-9,-6,0], [1,-9,-7,0], [1,-8,-7,0],
      [1,-10,-8,0], [1,1,1,0], [1,4,4,1],
      [1,8,8,1], [1,9,9,2], [1,9,9,3],
      [1,8,8,3], [1,10,10,2], [0,-1,0,0],
      [0,0,-1,0], [0,0,0,-1], [0,1,1,1]];
$E = [[0,1], [1,2], [2,3], [3,4], [4,5],
      [5,1], [4,6], [0,15], [5,15], [6,15],
      [6,14], [3,14], [2,14], [0,7], [7,8],
      [8,9], [9,10], [10,11], [11,12], [12,8],
      [10,13], [7,16], [9,16], [13,16], [13,17],
      [11,17], [12,17]];
$Tc = new fan::PolyhedralComplex(VERTICES=>$V,
    MAXIMAL_POLYTOPES=>$E);
```

Framing a tropical curve in POLYMAKE and exporting it to OPENSCAD

Constructing a bounding box can be done using the command `generateBoundingBox` as before:

```
$bBox = generateBoundingBox($Ts);
$Tbounded = intersectWithBoundingBox($Ts,$bBox);
```

For the sake of stability, we recommend constructing a frame for the tropical curve. This is best done using a tropical surface containing it:

```
$Tq = new fan::PolyhedralComplex(
    VERTICES=>$Tq->VERTICES->minor(All,~[1]),
    MAXIMAL_POLYTOPES=>$Tq->MAXIMAL_POLYTOPES);
$Tframe = intersectWithBoundingBoxForFraming($Tq,
    $bBox);
```

Exporting the frame and the tropical curve to OPENSCAD can be done using the command `generateSCADFileForCurve`. It requires two bounded polyhedral complexes and a filename. If the file already exists, it will be overwritten:

```
$filename = "foo.scad";
generateSCADFileForCurve($Tframe,$Ts,$filename);
```

Solidifying a tropical curve in OPENSCAD and exporting it for 3D-printing

To solidify the curve into a three-dimensional model, open the exported file in OPENSCAD. See Figures 3 and 4 for a preview of the model.

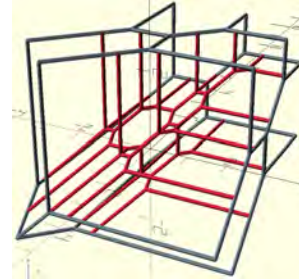


Figure 3: Framed and solidified tropical sextic curve in OPENSCAD.

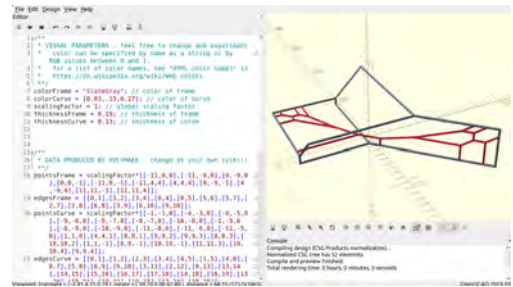


Figure 4: Framed and solidified genus-2 tropical cubic curve in OPENSCAD.

As before, the top of the file contains parameters which control the thickness of the model amongst other things:

```
colorFrame = "SlateGray"; // color of frame
colorCurve = [0.83,.15,0.27]; // color of curve
scalingFactor = 1; // global scaling factor
thicknessFrame = 0.05; // thickness of frame
thicknessCurve = 0.05; // thickness of curve
```


- `colorFrame` and `colorCurve` are the colors of the surface in the render. They can be specified either by an HTML color name or by RGB value.
- `scalingFactor` is a factor by which the polyhedral complex is scaled. It can be used to scale the model to the desired size.
- `thicknessFrame` and `thicknessCurve` control the thickness of the frame and curve. OPENSCAD solidifies both by replacing every vertex with a ball with diameter equal to the specified radius and taking convex hulls.

Modelling Tropical Curves on Tropical Surfaces

For an example of producing a 3D-printable model of a tropical curve on a tropical surface see the file

`3d.printing.template.surface.and.curve.pl`.

To run the example, simply copy its contents into any POLYMAKE session. In this section, we briefly explain some of its contents. Note that some variables are renamed due to spacing.

Constructing a curve and surface in POLYMAKE

The previous section showed an easy way to construct a tropical curve lying on a tropical surface in POLYMAKE: start with the surface and construct the curve by intersecting it with another surface. However, as mentioned before, this may not always be possible or easy to do. One option that always works is to construct the curve manually by specifying its vertices and edges:

```
# constructing tropical plane via polynomial
$l=toTropicalPolynomial("max(y,z,w)",qw(w x y z));
$Tl = new tropical::Hypersurface<Max>(
    POLYNOMIAL=>$l);
$Tl = new fan::PolyhedralComplex(
    POINTS=>$Tl->VERTICES->minor(All,~[1]),
    INPUT_POLYTOPES=>$Tl->MAXIMAL_POLYTOPES,
    INPUT_LINEALITY=>$Tl->LINEALITY_SPACE->
        minor(All,~[1]));

# constructing tropical curve manually:
$V = [[1,0,0,0],[1,-8,0,0],[1,-8,-6,0],
      [1,12,0,0],[0,0,-1,0],[1,-6,-6,0],
      [1,-11,-3,0],[1,-11,-6,0],[0,-1,0,0],
      [1,-12,-7,0],[1,4,0,-2],[1,10,0,-2],
      [1,8,0,-6],[0,0,0,-1],[0,1,1,1],
      [1,-4,6,6],[1,-4,5,5],[1,-3,5,5],
      [1,-5,4,4],[1,-3,3,3],[1,-5,3,3]];
$E = [[3,4],[0,5],[2,5],[4,5],[1,6],[6,7],
      [6,8],[2,7],[7,9],[8,9],[4,9],[10,11],
      [3,11],[11,12],[0,10],[10,12],[12,13],
      [1,13],[14,15],[8,15],[15,16],[14,17],
      [16,17],[16,18],[17,19],[8,18],[18,20],
      [0,19],[19,20],[1,20],[3,14]];
$Tq = new fan::PolyhedralComplex(POINTS=>$V,
    INPUT_POLYTOPES=>$E);
```

Bounding a tropical surface in POLYMAKE and exporting it to OPENSCAD

Constructing a bounding box is easiest done using the command `generateBoundingBox` as before:

```
$bBox = generateBoundingBox($Tq);
$TlBounded = intersectWithBoundingBox($Tl,$bBox);
$TqBounded = intersectWithBoundingBox($Tq,$bBox);
```

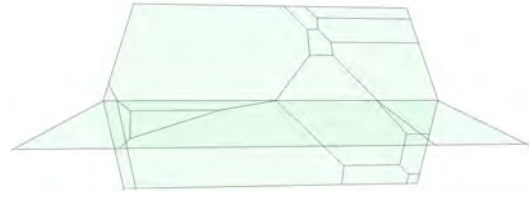


Figure 5: Tropical plane with a tropical quartic curve in POLYMAKE.

Exporting the tropical surface and curve to OPENSCAD can be done using the command `generateSCADFileForSurfaceAndCurve`. It requires two bounded polyhedral complexes and a file-name. If the file already exists, it will be overwritten:

```
$filename = "foo.scad";
generateSCADFileForSurfaceAndCurve($TlBounded,
    $TqBounded,$filename);
```

Solidifying a tropical surface with a tropical curve in OPENSCAD and exporting them for 3D-printing

To solidify the tropical surface with a tropical curve into a three-dimensional model, open the exported file in OPENSCAD. See Figure 6 for a preview of the model.

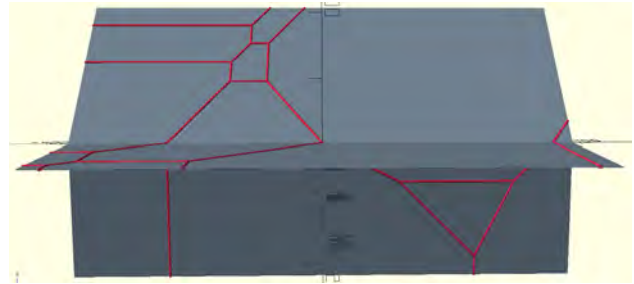


Figure 6: Solidified tropical plane with a tropical quartic curve in OPENSCAD.

The top of the file contains parameters which control the thickness of the model amongst other things:

```
colorSurface = "SlateGray"; // color of surface
colorCurve = [0.83,.15,0.27]; // color of curve
scalingFactor = 1; // global scaling factor
thicknessSurface = 0.01; // thickness of surface
thicknessCurve = 0.1; // thickness of curve
```

- `colorSurface` and `colorCurve` are the colors of the surface and curve in the render. It can be specified either by an HTML color name or by RGB value.
- `scalingFactor` is a factor by which the polyhedral complex is scaled. It can be used to scale the model to the desired size.
- `thicknessSurface` and `thicknessCurve` control the thickness of the surface and of the curve, respectively. OPENSCAD solidifies the surface by replacing every vertex with a ball with diameter equal to the specified radius and taking convex hulls.

Gallery



Figure 7: A tropical cubic surface from [13] with 5 of 27 tropical lines drawn (one per motif).

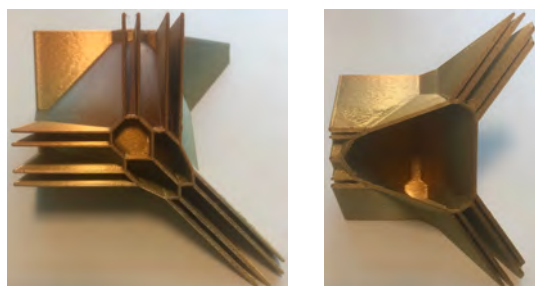


Figure 8: A tropical quartic surface from [2], cut in half to reveal the inscribed $K3$ polytope.



Figure 9: A tropical sextic curve from [6], the white frame traces the quadric surface containing it.

References

- [1] Elizabeth Baldwin, Paul Klemperer: *Understanding Preferences: “Demand Types”, and the Existence of Equilibrium With Indivisibilities*. *Econometrica* **87**, No. 3, 867–932 (2019).
- [2] Gabriele Balletti, Marta Panizzut, Bernd Sturmfels: *K3 polytopes and their quartic surfaces*. *Adv. Geom.* 21, No. 1, 85–98 (2021).
- [3] *bertini-real: a software for real algebraic sets*. Publicly available at <http://bertinireal.com>.
- [4] Ewgenij Gawrilow, Michael Joswig: *polymake: a framework for analyzing convex polytopes*. In: *Polytopes—combinatorics and computation* (Oberwolfach, 1997), 43–73, DMV Sem., 29, Birkhäuser, Basel, 2000.
- [5] Stefan Halverscheid, Oliver Labs: *Felix Klein’s Mathematical Heritage Seen Through 3D Models*. In: *The Legacy of Felix Klein*, 131–152, Springer International Publishing, 2019.
- [6] Corey Harris, Yoav Len: *Tritangent planes to space sextics: the algebraic and tropical stories*. Smith, Gregory G. (ed.) et al., *Combinatorial algebraic geometry. Selected papers from the 2016 apprenticeship program, Ottawa, Canada, July–December 2016*. Fields Institute Communications 80, 47–63 (2017).
- [7] Michael Joswig: *Essentials of tropical combinatorics*. Graduate Studies in Mathematics **219**. Providence, RI: American Mathematical Society (2021).
- [8] jschobben: *colorscad*. <https://github.com/jschobben/colorscad> (visited February 2022).
- [9] Diane Maclagan, Bernd Sturmfels: *Introduction to tropical geometry*. Graduate Studies in Mathematics **161**. Providence, RI: American Mathematical Society (2015).
- [10] MO-Labs: *The premium designer of 3D-printed and laser-in-glass mathematical models with a context in the history of mathematics*. <https://math-sculpture.com/> (visited December 2021).
- [11] Anthea Monod, Bo Lin, Ruriko Yoshida, Qiwen Kang: *Tropical Geometry of Phylogenetic Tree Space: A Statistical Perspective* arXiv:1805.12400 (2018).
- [12] OpenSCAD: *The Programmers Solid 3D CAD Modeller*. <https://openscad.org/>.
- [13] Marta Panizzut, Magnus Dehli Vigeland: *Tropical lines on cubic surfaces*. *SIAM J. Discrete Math.* 36, No. 1, 383–410 (2022).
- [14] Martin Schilling: *Catalog mathematischer Modelle*. Martin Schilling Verlag, Leipzig, 1911. Gallery publicly available at <http://www.universitaetssammlungen.de/publikation/5290>.
- [15] David Speyer, Bernd Sturmfels: *The tropical Grassmannian*. *Adv. Geom.* **4**, No. 3, 389–411 (2004).



Antrag auf Mitgliedschaft in der Fachgruppe Computeralgebra

Die Fachgruppe Computeralgebra sieht es als ihre Aufgabe an, Lehre, Forschung, Entwicklung, Anwendungen, Informationsaustausch und Zusammenarbeit auf dem Gebiet der Computeralgebra in Deutschland zu fördern.

Eine Mitgliedschaft in der Fachgruppe Computeralgebra gibt es bereits ab 7,50 € pro Jahr (für Mitglieder von DMV, GI oder GAMM; ansonsten 9 €).

Vorteile einer Mitgliedschaft:

- Sie fördern durch Ihren Beitrag die Workshops, Seminare, Tagungen und andere Aktivitäten auf dem Gebiet der Computeralgebra, die die Fachgruppe organisiert und unterstützt.
- Sie erhalten zweimal im Jahr den Computeralgebra-Rundbrief mit vielen interessanten Informationen rund um die Computeralgebra frei Haus.
- Sie verleihen unserer Stimme an Gewicht, die wir aktiv in Diskussionen um die Stellung der Computeralgebra in der Ausbildung in Schule und Hochschule einbringen.

Wir würden uns sehr über Ihre Unterstützung freuen. Die Mitgliedschaft in der Fachgruppe steht allen offen. Weiter Informationen zur Mitgliedschaft und einen Aufnahmeantrag finden Sie auf unserer Webseite unter folgender Adresse, oder scannen Sie einfach den QR-Code.

<https://fachgruppe-computeralgebra.de/aufnahmeantrag>



OSCAR: An introduction

Max Horn (RPTU Kaiserslautern-Landau)

mhorn@rptu.de



What is OSCAR?

OSCAR is a new **Open Source Computer Algebra Research** system written in `Julia`, developed as central software project of the SFB-TRR 195 *Symbolic Tools in Mathematics and their Application* funded by the German Research Foundation (DFG). OSCAR is a collaborative effort with many contributors, and currently lead by Wolfram Decker, Claus Fieker, Michael Joswig and myself. The full source code is available at

<https://www.oscar-system.org/>

Cornerstones

OSCAR is built on *four cornerstones*, each providing state of the art capabilities in different domains:

- **ANTIC** (Nemo and Hecke): number theory
- **GAP**: group and representation theory
- **Polymake**: polyhedral and tropical geometry
- **Singular**: commutative and non-commutative algebra, algebraic geometry

These are not merely software packages used by OSCAR, rather they are *integral components* whose development is interlocked with that of OSCAR.

OSCAR relies on further components such as `Arb`, `Calcium`, `FLINT`, `msolve` and more. Yet it is more than just the sum of its components: it fuses them together into a comprehensive computer algebra system with a consistent user interface driven by mathematics only. The intention is to provide users with high-level objects closely paralleling what you might find in a text book. As an example, you can work with schemes and varieties without touching Gröbner bases.

Julia

The cornerstones are tied together by a ton of new code written in the `Julia` [1] programming language.

`Julia` is relatively young, first appearing in the public in 2012. Originally developed at MIT for fast numerical mathematics, it enjoys a rapidly growing user base in many mathematical disciplines. Thanks to a just-in-time compiler (JIT) it can be used to produce highly efficient code that is competitive with pure C code, while at the same time being a high-level safe programming language. This addresses the “two language problem”: instead of writing most code in one (high-level) language (e.g. Python, Perl), but performance critical parts in another (e.g. C, C++, Fortran), all code can be written in the same language. Its excellent support for calling into existing C/C++ code was critical for developing the bridges to `GAP`, `Polymake` and `Singular` (`ANTIC` is written in `Julia` and thus needs no interfacing). `Julia` has many other compelling features, such as a REPL (read-eval-print loop, i.e., one can use it interactively with a prompt, multi-line editing, persistent history, tab completion, etc.), its package manager, or its support for multiple dispatch. There are also high quality interfaces to Python, R and other systems should one need to call code in these systems.

One downside of `Julia`’s JIT compiler is that the first time a function is executed, there can be a significant delay while OSCAR is being compiled. However, the `Julia` developers have made great strides towards improving this in the past few years, and we are confident this will improve further.

Installing

To install OSCAR, the following steps are required:

1. Install a C/C++ compiler.
2. Install `Julia` version 1.6 or later¹.
3. Start `Julia` and execute the following:

```
julia> using Pkg
julia> Pkg.add("Oscar")
```

¹available from <https://julialang.org/>; avoid installing it via `apt-get` or similar package managers

On Windows, you have to perform these steps in the *Windows Subsystem for Linux* (WSL).² Detailed installation instructions for all platforms are available at

<https://oscar-system.org/install/>

Now, you can use OSCAR in a Julia session:

```
julia> using Oscar
[ ... a banner is printed ... ]
```

The OSCAR team is working on providing a website on which one can try out OSCAR without installation.

Documentation and a warning

Below I will give a few short examples that showcase how OSCAR allows combining the cornerstones. However I will only scratch the surface of what the system can do, and the selection here is doubtlessly biased by my own background as a group theorist. To get a better impression of the existing capabilities and how to use them, please take a look at the reference manual at

<https://docs.oscar-system.org/>

One more caveat: The following examples were done using the current development version of OSCAR, which will eventually become version 0.12.0. Hopefully that version will have been released by the time you are reading this. Be advised that some examples might fail with older OSCAR versions.

A quick tour across OSCAR

We start our tour by creating a matrix representation of the dihedral group of order 12 over a number field.

```
F, t = quadratic_field(3)
a = diagonal_matrix(F, ([1, -1, -1]))
b = matrix(F, [1//2 -t//2 0
               t//2 1//2 0
               0      0 1])
G = matrix_group(a, b)
```

The field F in this example is provided by ANTIC, but matrix groups are handled by GAP in the background. OSCAR takes care of all necessary wrapping and conversions. Consequently we can forget about the nitty-gritty implementation details and focus on what we really want to do instead. For instance, we can verify that the group G as defined above is indeed a dihedral group of order 12.

```
julia> order(G)
12

julia> describe(G)
"D12"
```

At this point, we may wish to understand the group action a bit better by visualizing it. Here we take the orbit of a vector in F^3 and visualize its convex hull with some help by Polymake.

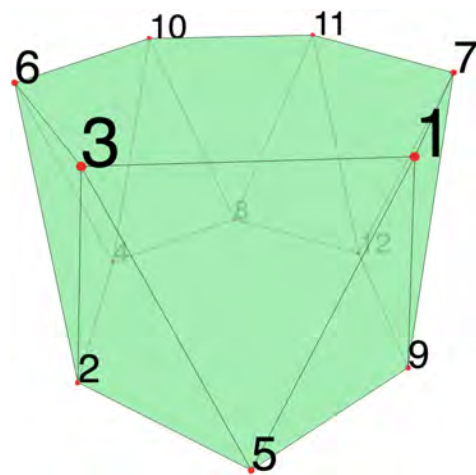
```
julia> o = orbit(G, *, F.([1,1,1]));

julia> vert = matrix(F, collect(o));

julia> ph = convex_hull(vert)
Polyhedron in ambient dimension 3

julia> visualize(ph)
```

This opens a web browser window with an interactive rendering of the polyhedron. It looks like this:



We can then compute the full combinatorial automorphism group of this polytope, given by its action on the vertices. Naturally it contains (an isomorphic copy of) our starting group G , but it got a bit bigger (the vertex labels are shifted by 1 here).

```
julia> aut = automorphism_group(ph;
                                action=:on_vertices);

julia> small_generating_set(aut)
3-element Vector{PermGroupElem}:
 (1, 3, 6, 10, 11, 7) (2, 4, 8, 12, 9, 5)
 (2, 12) (3, 7) (4, 8) (5, 9) (6, 11)
 (1, 2) (3, 5) (4, 7) (6, 9) (8, 11) (10, 12)

julia> describe(aut)
"D24"
```

Naturally we can also study other actions, e.g. the one induced naturally on the polynomial ring $F[x, y, z]$.

²see <https://learn.microsoft.com/en-us/windows/wsl/install>

```
julia> R, (x,y,z) = F["x", "y", "z"];

julia> f = x^2 + 3y;

julia> f^gen(G,1)
x^2 - 3*y

julia> o = orbit(G, ^, f); sum(o)
3*x^2 + 3*y^2
```

Thanks to Singular, the full power of computational commutative algebra is available at our fingertips. As with everything else, we are just scratching the tip of the iceberg with the following example: we compute the ideal generated by the orbit (the `collect` is necessary because the orbit is just an iterator). Then we can quotient that ideal out and determine the Krull dimension of the quotient. Again, OSCAR seamlessly translates data for us as necessary in the background.

```
julia> I = ideal(collect(o));

julia> A, proj_hom = quo(R, I);

julia> dim(A)
1
```

A classical topic that brings together group theory and commutative algebra is invariant theory. OSCAR has extensive support for this. Here is a quick example.

```
julia> IR = invariant_ring(G);

julia> fundamental_invariants(IR)
4-element Vector:
 x3^2
 x1^2 + x2^2
 x1^6 + 5*x1^4*x2^2 + 5//3*x1^2*x2^4
 + 11//9*x2^6
 x1^5*x2*x3 - 10//3*x1^3*x2^3*x3
 + x1*x2^5*x3
```

Each of the cornerstones is a powerful tool on its own, but of course there are many cross-cutting applications that really need the expertise of multiple or all of them. The invariant theory example above is just one such instance. Another is described in the article “Toric Geometry in OSCAR” on page 20. Yet another is our state of the art implementation of Galois group computations over number fields, function fields and more.

```
R, x = QQ["x"];
f = x^6 - 366x^4 - 878x^3 + 4329x^2
    + 14874x + 10471;
g, ctx = galois_group(f);
```

The result is a permutation group g together with a context object how precisely it acts as Galois group. We can of course treat g like any other group, but we can also e.g. obtain a p -adic approximation of the roots.

OSCAR automatically picks a suitable prime for this, here $p = 13$. It is also possible to obtain complex approximations for teaching, but for practical applications the p -adic approach is superior.

```
julia> describe(g)
"C3 x S3"

julia> roots(ctx, 5)[1]
13^0 + 2*13^1 + 4*13^2 + 2*13^3
+ 7*13^4 + O(13^5)
```

Deep dive: the low-level interfaces

As mentioned, if necessary users can directly access functionality of the cornerstones. Let me emphasize that this is intended as a last resort. To use these low-level interfaces, you usually need some expertise in the corresponding cornerstone, and things become more tedious. If at all possible, we prefer to provide a “nice” high-level interface. So if something is missing, please let us know (see the final section to learn how).

But realistically, GAP, Polymake and Singular together with their many extension packages provide a vast reservoir in capabilities in many highly specialized domains, and wrapping them all with all their intricate details will take approximately forever. Thus it is crucial that there is an escape hatch that allows users to reach these capabilities if they need to.

For GAP you can evaluate arbitrary GAP code using `GAP.evalstr`. E.g. we do not (yet!) have Lie algebras in OSCAR, but we can get them from GAP:

```
julia> L = GAP.evalstr("""
    SimpleLieAlgebra("A", 2,
    Rationals)""")
GAP: <Lie algebra of dimension 10 ...>
```

This approach makes it cumbersome to pass around data. But direct access to GAP variables and functions is possible.

```
julia> R = GAP.Globals.RootSystem(L)
GAP: <root system of rank 2>
```

In principle, all packages of the GAP distribution are available. However, those packages that require compilation require some extra care. Please consult the `GAP.jl` documentation for more details.³ We will address this limitation in a future release.

³see <https://oscar-system.github.io/GAP.jl/stable/packages>

```
julia> GAP.Packages.load("sla")
true

julia> GAP.Globals.
      WeylGroupAsPermGroup(R)
GAP: Group([ (1,4) (2,3) (5,6),
              (1,3) (2,5) (4,6) ])
```

Lastly, die-hard GAP fans can even get a fully functional GAP prompt from within OSCAR, which of course also provides full access to all Julia objects.

```
julia> GAP.prompt()

gap> IsLieNilpotent(Julia.L);
false
gap> quit; # return to Julia
```

Similarly, there is a Polymake prompt that can be reached by pressing the `$` key at the start of a Julia prompt; exit it by pressing the backspace key.

```
julia> Polymake.Shell.j = 42
42

common > print($j);
42
common > $c = polytope::cube(3);

julia> C = Polymake.Shell.c
name: c
type: Polytope<Rational>
description: cube of dimension 3
[...]
```

By the way, with the Polymake prompt we can conveniently reproduce the examples from the article “Hands-on Tropical Geometry” on page 10.

Programmatic access from Julia is also possible.

```
julia> Polymake.polytope.dim(C)
3

julia> C.F_VECTOR
pm::Vector{pm::Integer}
8 12 6
```

Note that in this particular example, it would be better to use OSCAR’s high-level `cube` and `dim` methods.

Finally, for Singular we do not offer access to a REPL, but one can call into it as needed, and e.g. invoke library functions. For example, `Singular.LibDeRham.deRhamCohomology` is a wrapper for the function `deRhamCohomology` from `deRham.lib`. However the details for this interface are a bit more complicated and would require a lengthier discussion, so we omit them here – suffice it to say that

access *is* possible.

That completes our quick tour. Once more let me recommend the article “Toric Geometry in OSCAR” on page 20 for a proper mathematical example. There have also been prior articles in the Rundbrief on some OSCAR components [3, 4] which are still quite relevant. Finally, we are currently preparing a book about OSCAR [2].

We want you!

We are eager to hear from you if you are using OSCAR or are generally interested in it. We constantly strive to improve OSCAR, and as such look forward to bug reports, feature request, and code contributions. Users of all experience levels are welcome.

If you are experimenting with OSCAR and have questions, I heartily recommend our Slack chat, where many people participate and provide answers. You can join via

<https://oscar-system.org/slack>

We also have a mailing list, a GitHub discussion forum and more, see

<https://oscar-system.org/community/>

for details. It also explains how to best report issues (spoiler: via our issue tracker). And if you prefer personal communication, don’t hesitate to reach out to me personally and e.g. email me at mhorn@rptu.de.

Last but not least, you can get the latest source code from our Git repository at

<https://github.com/oscar-system/Oscar.jl>

Acknowledgement

This work was supported by the SFB-TRR 195 *Symbolic Tools in Mathematics and their Application* of the German Research Foundation (DFG).

References

- [1] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, *Julia: A fresh approach to numerical computing*, SIAM review **59** (2017), no. 1, 65–98.
- [2] W. Decker, C. Eder, C. Fieker, M. Horn, M. Joswig, *The OSCAR book*, Springer, 2024.
- [3] C. Fieker, C. Sircana, T. Hofmann, *Hecke: A Number Theory Package*, Computeralgebra Rundbrief **66** (2020), 12–14.
- [4] W. Hart, *ANTIC: Algebraic Number Theory in C*, Computeralgebra Rundbrief **56** (2015), 10–11.

Toric Geometry in OSCAR

Martin Bies (RPTU Kaiserslautern-Landau)

Lars Kastner (TU Berlin)

bies@mathematik.uni-kl.de

kastner@math.tu-berlin.de



Why toric geometry in OSCAR?

Toric geometry – an arena for mathematical theories

Among the fields of algebraic geometry, the field of toric geometry is particularly well understood and algorithmic. Among others, the cohomology ring, the Chow ring, topological intersection numbers as well as cohomologies of coherent sheaves can be obtained with computer algorithms [10]. Therefore, toric varieties provide a useful platform for testing mathematical theories.

To put it briefly, toric varieties are characterized by having an algebraic torus $(\mathbb{C}^*)^r$ as a dense and open subset. This is why they are called *toric*. While the realm of toric varieties is more constrained compared to that of general schemes/varieties, the toric universe still provides a significant degree of versatility. As an example, many Calabi-Yau manifolds can be constructed as complete intersections in toric varieties [19, 20]. This includes many K3 surfaces [19]. More recently, starting from such K3 surfaces, researchers have discovered in the framework of F-theory – a non-perturbative regime of string theory – the largest currently-known class of globally consistent Standard Model solutions without chiral exotics and gauge coupling unification [11].

For all these reasons, there is a high demand for computer implementations of toric geometry. Some examples of computer algebra systems that support toric geometry are [17, 25].

OSCAR – a melting pot

We present a computer implementation for toric varieties in the computer algebra system OSCAR [14, 21]. The funding for OSCAR is provided by the SFB-TRR 195 *Symbolic Tools in Mathematics and their Application* of the German Research Foundation (DFG). The main architectural feature of OSCAR is that its four fundamental tools *Antic* (Hecke, Nemo), *GAP*, *Polymake* and *Singular* are *integral components*, rather than external software that can be used. For more information, the interested reader can consult the article “OSCAR: An introduction” on page 16 of this Rundbrief or the OSCAR homepage:

<https://www.oscar-system.org>

¹Interested readers may also explore the actual OSCAR code on GitHub.

By leveraging *Polymake*, we can carry out polyhedral geometry operations, such as handling cones and fans, and utilize cutting-edge algorithms for triangulations [18]. This provides a reliable backbone for toric geometry in OSCAR. The Cox ring as well as the Chow ring of toric varieties are polynomial rings. Closed subvarieties of toric varieties correspond to homogeneous polynomial in the Cox ring [10]. This functionality is provided by the software *Singular*. Additionally, tools from group and number theory are essential in toric geometry. Such tasks are executed with *Antic* (Hecke, Nemo) and *GAP*. To sum up, toric geometry benefits greatly from the combination of *Antic* (Hecke, Nemo), *GAP*, *Polymake* and *Singular*.

Julia – a modern programming language

Julia [5] is a high-performance programming language designed for numerical and scientific computing. The growing ecosystem of *Julia* packages ensures its continued viability for scientific computing and data analysis. OSCAR is written in *Julia*. This implies that the performance of OSCAR should be comparable or even better than many other implementations.

Overview

Our goal with OSCAR is to create a computer algebra system that is both user-friendly and convenient. To assist users with toric geometry, we offer a tutorial.¹

<https://www.oscar-system.org/tutorials/>.

The toric implementation in OSCAR are conceptually based on [10]. This is a fundamental guiding principle within OSCAR: Implementations are conceptually grounded in a few carefully selected publications.

The relationship between toric geometry and polyhedral geometry is crucial for any toric geometry implementation. We illustrate this connection for affine toric varieties.

In OSCAR, the toric implementations focus on the lattice $N = \mathbb{Z}^n$, where $n \in \mathbb{Z}_{\geq 0}$ is a suitable integer. Let M be the dual lattice of N . We then consider a rational polyhedral cone $\sigma \subseteq N \otimes_{\mathbb{R}} \mathbb{R} \cong \mathbb{R}^n$. To this cone, we associate the semigroup $S_{\sigma} = \sigma^{\vee} \cap M$. The corresponding affine toric variety U_{σ} is given by [10]:

$$U_{\sigma} = \text{Spec}(\mathbb{C}[S_{\sigma}]) = \text{Spec}(\mathbb{C}[\sigma^{\vee} \cap M]) . \quad (1)$$

As an example, consider

$$\sigma = \text{Span}_{\mathbb{Z}_{\geq 0}} \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right). \quad (2)$$

We create U_σ in OSCAR:

```
o = positive_hull([1 0; 0 1])
U = affine_normal_toric_variety(o)
```

Many properties of U_σ are encoded in σ . For instance, U_σ is smooth if and only if σ can be generated by a subset of a basis of the lattice N . An interactive check in OSCAR can determine whether U_σ is smooth:

```
julia> hilbert_basis(o)
2-element SubObjectIterator{PointVector{ZZRingElem}}:
 [1, 0]
 [0, 1]

julia> is_smooth(U)
true
```

Similarly, the dimension of U_σ matches that of σ :

```
julia> dim(o) == dim(U)
true
```

Below is an instance of a non-smooth affine toric variety that can be created using OSCAR:

```
o2=positive_hull([-1 1; 0 1; 1 1])
U2=affine_normal_toric_variety(o2)
```

We verify interactively that U_2 is not smooth:

```
julia> hilbert_basis(o2)
3-element SubObjectIterator{PointVector{ZZRingElem}}:
 [-1, 1]
 [0, 1]
 [1, 1]

julia> is_smooth(U2)
false
```

Notice the appearance of the generator $[0, 1]$ in the Hilbert basis of σ_2 . Its appearance signifies that U_2 is not smooth. Alternatively, we can inspect U_2 as subvariety of the affine space. To this end, we compute the toric ideal:

```
julia> toric_ideal(U2)
ideal(-x1*x2 + x3^2)
```

This means, that in the affine space \mathbb{A}^3 with coordinates (x_1, x_2, x_3) , it holds $U_2 \cong V(-x_1x_2 + x_3^2)$. Consequently, U_2 is singular.

Much more can be said about the interplay between polyhedral and toric geometry. For example, there exists a connection between normal toric varieties and rational polyhedral fans. Indeed, in OSCAR, one can create a general normal toric variety based on a rational polyhedral fan. Moreover, OSCAR offers specialized constructors for several well-known toric varieties, including `projective_space`, `hirzebruch_surface`, `del_pezzo_surface`, and `cyclic_quotient_singularity`. For further information, interested readers may wish to consult [10].

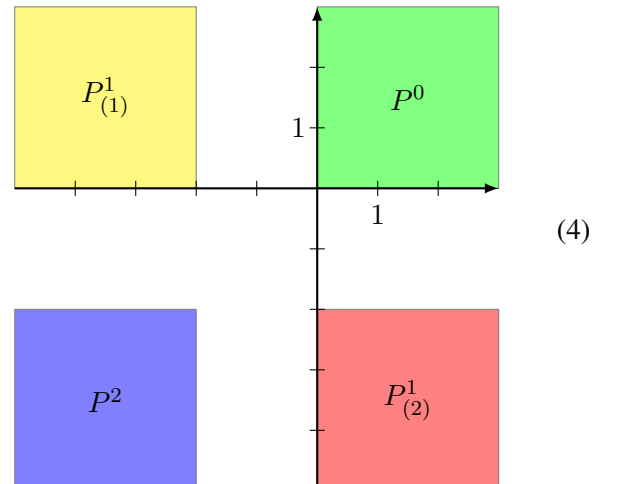
Notable capabilities

Vanishing sets of line bundle cohomology

Support for torus invariant divisors, divisor classes and line bundles is available in OSCAR. The `cohomCalc` algorithm [1, 9] is employed to infer dimensions of line bundle cohomologies on any smooth and complete, as well as any simplicial and projective toric variety X_Σ . The set $V^i(X_\Sigma)$ of all line bundles on X_Σ with vanishing i -th sheaf cohomology can be derived [6]:

$$V^i(X_\Sigma) = \text{Pic}(X_\Sigma) - \bigcup_{m=1}^l L_{(m)}^i, \quad (3)$$

where $L_{(m)}^i$ is the set of lattice points in a certain polyhedron $P_{(m)}^i$. For $\mathbb{P}^1 \times \mathbb{P}^1$, it holds $\text{Pic}(\mathbb{P}^1 \times \mathbb{P}^1) = \mathbb{Z}^2$ and that the vanishing sets can be represented as follows:



Specifically,

$$\begin{aligned}
V^0(\mathbb{P}^1 \times \mathbb{P}^1) &= \mathbb{Z}^2 - (P^0 \cap \mathbb{Z}^2), \\
V^1(\mathbb{P}^1 \times \mathbb{P}^1) &= \mathbb{Z}^2 - (P_{(1)}^1 \cup P_{(2)}^1) \cap \mathbb{Z}^2, \\
V^2(\mathbb{P}^1 \times \mathbb{P}^1) &= \mathbb{Z}^2 - (P^2 \cap \mathbb{Z}^2), \\
P^0 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \text{Span}_{\mathbb{Z}_{\geq 0}} \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right), \\
P_{(1)}^1 &= -\begin{bmatrix} 2 \\ 0 \end{bmatrix} + \text{Span}_{\mathbb{Z}_{\geq 0}} \left(\begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right), \\
P_{(2)}^1 &= -\begin{bmatrix} 0 \\ 2 \end{bmatrix} + \text{Span}_{\mathbb{Z}_{\geq 0}} \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right), \\
P^2 &= -\begin{bmatrix} 2 \\ 2 \end{bmatrix} - \text{Span}_{\mathbb{Z}_{\geq 0}} \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right).
\end{aligned} \tag{5}$$

With the following lines, we can replicate these results in OSCAR:

```

P1 = projective_space(
    NormalToricVariety, 1)
v0, v1, v2 = vanishing_sets(P1*P1)
ph0 = polyhedra(v0)[1]
ph11, ph12 = polyhedra(v1)
ph2 = polyhedra(v2)[1]

```

With OSCAR, we can investigate the polyhedra interactively. For example, we can find inequalities for P^0 and P^2 as follows:

```

julia> print_constraints(ph0)
-x1 ≤ 0
-x2 ≤ 0

julia> print_constraints(ph2)
x2 ≤ -2
x1 ≤ -2

```

Indeed, from eq. (5) we see that L^0, L^2 can be expressed as follows:

$$L^0 = \{(x_1, x_2) \in \mathbb{Z}^2 \mid x_1, x_2 \geq 0\}, \tag{6}$$

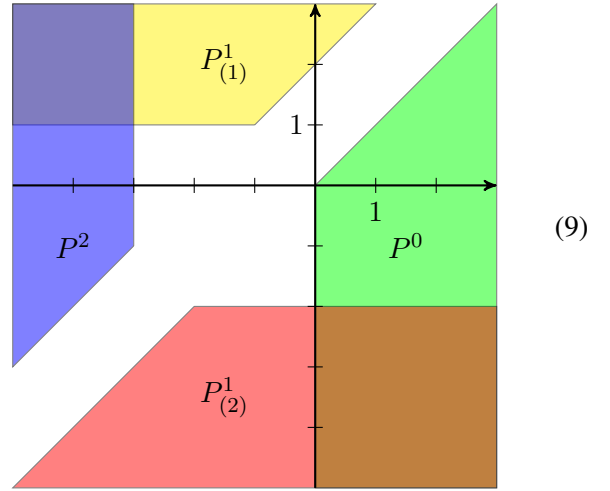
$$L^2 = \{(x_1, x_2) \in \mathbb{Z}^2 \mid x_1, x_2 \leq -2\}. \tag{7}$$

It is not too hard to repeat this exercise for $L_{(1)}^1, L_{(2)}^1$.

As a more interesting example, consider the del Pezzo surface dP_1 with \mathbb{Z}^2 -graded Cox ring:

	x_1	x_2	x_3	e_1
H	1	1	1	
$-E_1$	1	1		-1

For this grading, we visualize the vanishing sets:



The interested reader might find it entertaining to “see” Serre duality in eq. (4) and eq. (9).

We emphasize that the vanishing sets can be determined algorithmically for any smooth and complete, as well as any simplicial and projective toric variety X_Σ . However, our ability to visualize the vanishing sets reduces drastically once the polyhedra are of dimension 4 or higher. This happens for instance for $\mathbb{P}^1 \times \mathbb{P}^1 \times dP_1$. Still, the vanishing sets can be derived in OSCAR:

```

P1 = projective_space(
    NormalToricVariety, 1)
dP1 = del_pezzo_surface(1)
v0, v1, v2, v3, v4
    = vanishing_sets(P1*P1*dP1)

```

Intersection theory

Loosely speaking, intersection theory provides an answer to the question “At how many points do two algebraic cycles intersect?”. A caveat arises whenever the algebraic cycles in question are “similar/the same”. This leads to the notion of *rational equivalence* and the observation, that sometimes the number of intersection points can be negative. To demonstrate this somewhat exotic idea in a concrete setting, we focus on the del Pezzo surface dP_1 with \mathbb{Z}^2 -graded Cox ring as in eq. (8). Next, consider the following algebraic cycles:

$$H = V(x_1) + V(e_1), \quad E_1 = V(e_1). \tag{10}$$

Strictly speaking, we want to consider the rational equivalence classes of these algebraic cycles. For ease of notation, we do not introduce new symbols. The intersection numbers among H and E_1 are as follows:

$$H^2 = 1, \quad H \cdot E_1 = 0, \quad E_1 \cdot E_1 = -1. \tag{11}$$

The following code computes this result in OSCAR:


```
julia> dP1 = del_pezzo_surface(1);

julia> intersection_form(dP1)
Dict{MPolyRingElem, QQFieldElem}
with 10 entries:
  x1*x3 => 1
  e1^2  => -1
  x2*x3 => 1
  x3^2  => 1
  x1*x2 => 0
  x3*e1 => 0
  x2^2  => 0
  x1*e1 => 1
  x2*e1 => 1
  x1^2  => 0
```

Certainly, we can create the rational equivalence classes of H and E_1 in OSCAR:

```
x1, x2, x3, e1
      = gens(chow_ring(dP1))
E1 =
rational_equivalence_class(dP1, e1)
H = E1 +
rational_equivalence_class(dP1, x1)
```

With this, we can explicitly and interactively verify in OSCAR how these algebraic cycles intersect:

```
julia> H*H
Rational equivalence class on a
normal toric variety represented
by V(x2,x3)

julia> H*E1
Trivial rational equivalence class
on a normal toric variety

julia> E1*E1
Rational equivalence class on a
normal toric variety represented
by -1V(x2,x3)
```

In the last computation, notice the appearance of -1 . To understand its meaning, we must understand how the intersection points of E_1 with itself are computed. The theory tells us that we should use different, yet rationally equivalent, algebraic cycles which intersect “nicely”. The technical term for this is to move the algebraic cycles in *general position* [15].

In toric varieties, rational equivalences are captured by the ideal of linear relations:

```
julia> ideal_of_linear_
relations(dP1)
ideal(x1 - x3 + e1, x2 - x3 + e1)
```

Let \sim denote rational equivalence. Then it holds:

$$V(x_1) - V(x_3) + V(e_1) \sim 0, \quad (12)$$

$$V(x_2) - V(x_3) + V(e_1) \sim 0. \quad (13)$$

Hence $V(e_1) \sim V(x_3) - V(x_1)$ and it follows that

$$E_1 \cdot E_1 \sim V(e_1) \cdot [V(x_3) - V(x_1)] \quad (14)$$

$$= V(e_1, x_3) - V(e_1, x_1). \quad (15)$$

Next, let us look at the Stanley-Reisner ideal of dP_1 :

```
julia> stanley_reisner_ideal(dP1)
ideal(x1*x2, x3*e1)
```

From this ideal we learn that

$$\{p \in dP_1 \mid x_3 = e_1 = 0\} = \emptyset. \quad (16)$$

Consequently, we find

$$E_1 \cdot E_1 \sim -V(e_1, x_1). \quad (17)$$

It is not too hard to verify that $V(e_1, x_1) \sim V(x_2, x_3)$. This finally aligns our investigations with the result computed by OSCAR. We do hope that this example illustrates the origin of negative intersection numbers.

The collection of rational equivalence classes of algebraic cycles enjoys a ring structure where the multiplication corresponds to the intersection of the algebraic cycles. This ring is known as the *Chow ring* and can be computed for any complete, simplicial toric variety [10]. For example, the Chow ring of dP_1 can be computed in OSCAR as follows:

```
julia> chow_ring(dP1)
Quotient of Multivariate
Polynomial Ring in x1, x2, x3, e1
over Rational Field by ideal
(x1-x3+e1, x2-x3+e1, x1*x2, x3*e1)
```

It has been noted more recently that the completeness assumption can be dropped [23].² Indeed, OSCAR is capable of computing the Chow ring for simplicial toric varieties that are not complete. As an example, we create a non-complete, yet simplicial toric variety v from its rays r and (maximal) cones c :

```
r = [[1,0], [0,1], [-1,-1]]
c = [[1], [2], [3]]
v = normal_toric_variety(r, c)
```

We verify that v is not complete but simplicial:

```
julia> is_complete(v)
false

julia> is_simplicial(v)
true
```

We can also compute the Chow ring interactively:

²See also [16] for the significance of this observation for the interplay between matroids and toric varieties.

```
julia> chow_ring(v)
Quotient of Multivariate
Polynomial Ring in x1, x2, x3
over Rational Field by ideal
(x1-x3, x2-x3, x1*x2, x1*x3, x2*x3)
```

Triangulations

As explained in [19, 20], reflexive polytopes Δ° (and their polar duals) can be used to classify Calabi-Yau hypersurfaces in toric spaces. The ambient toric spaces can be found from fine regular star triangulations (FRST) of Δ° (see e.g. [12] for background). For an example, consider the square with vertices at $(\pm 1, \pm 1)$. The informed reader will notice immediately that this configuration has a unique FRST corresponding to $\mathbb{P}^1 \times \mathbb{P}^1$.

```
P = convex_hull(
    [1 1; -1 1; 1 -1; -1 -1])
X = NormalToricVarieties\
    FromStarTriangulations(P)
```

Certainly, we can verify that X consists only of a single variety. Furthermore, we compute the Stanley-Reisner and the irrelevant ideal of this toric variety, to provide evidence that this variety is indeed just $\mathbb{P}^1 \times \mathbb{P}^1$:

```
julia> length(X)
1

julia> irrelevant_ideal(X[1])
ideal(x3*x4, x2*x4, x1*x3, x1*x2)

julia> stanley_reisner_ideal(X[1])
ideal(x1*x4, x2*x3)
```

A much more involved example is included in the tutorial (<https://www.oscar-system.org/tutorials/>). This example is computationally demanding and its code was optimized for performance. We propose to use this example for benchmarking purposes. Note also that this code was used in a recent string theory application [7].

Outlook

OSCAR is a relatively new software and still under heavy development. This is an opportunity for young developers – we truly appreciate contributions. While this means that things are changing within OSCAR, the interface for toric varieties is already rather mature. In recent times, this interface has remained stable. For users (of the toric functionality) this is great news, as you need not be afraid of changes to the interface that might break your workflow. We strongly encourage users to try out and enjoy the existing toric functionality.

There are plans to significantly extend the toric functionality for coherent sheaves. In the realm of smooth

and complete toric varieties, coherent sheaves are equivalent to certain classes of finitely presented graded modules [10]. This equivalence can be utilized to compute sheaf cohomologies of coherent sheaves. In fact, a relevant algorithm for this purpose was proposed in [6].³ There are plans to incorporate this functionality into OSCAR in the future. It would also be advantageous to explore specialized algorithms, e.g. based on [2–4, 22], for cohomologies of vector bundles.

In view of applications in the field of F-theory – a specialized domain of string theory – initial discussions have taken place to assess the possibility of incorporating `FTheoryTools` [8] into OSCAR. `FTheoryTools` is primarily focused on computing resolution for singular elliptic fibrations. Such computations pose a significant arithmetic challenge in F-theory. The goal is to make this task as convenient as possible for researchers in this area. There are overlaps with some of the schemes technology that is currently being actively developed in OSCAR. For instance, OSCAR has basic support for toric schemes in experimental stages.

A more specialized task in F-theory involves constructing solutions that replicate the particle physics observed in modern accelerator experiments. Recently, numerous promising solutions known as the *Quadrillion F-theory Standard Models* (F-theory QSMs) were identified in [11]. These solutions are based on the geometry of toric K3 surfaces via [19]. Consequently, toric technology is critical to constructing and exploring these solutions in the future. In fact, many F-theory constructions are based on toric geometry (see [26] and references therein). It would be interesting to provide user-friendly and convenient tools in OSCAR for toric F-theory constructions.

Cosmological investigations within string theory led to the development of the software *CYTools* [13]. This software focuses on high-performance triangulations of the 4-dimensional reflexive polytopes in [20]. Such triangulation tasks matter in many explicit realizations of Calabi-Yau manifolds. For these reasons, [13] is a very interesting software package. We expect that its capabilities can be boosted by using `mptopcom` [18] or the latest version of `TOPCOM` [24]. This task is reserved for future work.

Acknowledgement

M. B. and L. K. express their gratitude and appreciation for the support provided by the OSCAR team, led by Claus Fieker, Max Horn, Michael Joswig, and Wolfram Decker. M. B. acknowledges financial support from the Forschungsinitiative des Landes Rheinland-Pfalz through the project *SymbTools – Symbolic Tools in Mathematics and their Application*. L. K. is thankful for the funding received from *MaRDI – Mathematical research initiative* of the German Research Foundation (DFG). This work was supported by the SFB-TRR 195 *Symbolic Tools in Mathematics and their Application* of the German Research Foundation (DFG).

³This algorithm is available at https://github.com/homalg-project/ToricVarieties_project.

References

- [1] cohomCalg package – High-performance line bundle cohomology computation based on [9]. <https://github.com/BenjaminJurke/cohomCalg>, 2010.
- [2] K. Altmann, J. Buczyński, L. Kastner, and A.-L. Winz. Immaculate line bundles on toric varieties, 2018.
- [3] K. Altmann and D. Ploog. Displaying the cohomology of toric line bundles, 2019.
- [4] K. Altmann and F. Witt. The structure of exceptional sequences on toric varieties of Picard rank two, 2021.
- [5] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.
- [6] M. Bies. *Cohomologies of coherent sheaves and massless spectra in F-theory*. PhD thesis, Heidelberg U., 2 2018.
- [7] M. Bies, M. Cvetič, R. Donagi, and M. Ong. Brill-Noether-general limit root bundles: absence of vector-like exotics in F-theory Standard Models. *JHEP*, 11:4, 2022.
- [8] M. Bies and A. P. Turner. FTheoryTools – Julia tools for F-Theory compactifications. <https://github.com/Julia-meets-String-Theory/FTheoryTools.jl>, 2022-2023.
- [9] R. Blumenhagen, B. Jurke, T. Rahn, and H. Roschy. Cohomology of Line Bundles: A Computational Algorithm. *J. Math. Phys.*, 51:103525, 2010.
- [10] D. A. Cox, J. B. Little, and H. K. Schenck. *Toric Varieties*. Graduate studies in mathematics. American Mathematical Soc., 2011.
- [11] M. Cvetič, J. Halverson, L. Lin, M. Liu, and J. Tian. Quadrillion F -Theory Compactifications with the Exact Chiral Spectrum of the Standard Model. *Phys. Rev. Lett.*, 123(10):101601, 2019.
- [12] J. A. De Loera, J. Rambau, and F. Santos. *Triangulations*. Graduate studies in mathematics. Springer, Heidelberg Dordrecht London New York, 2010.
- [13] M. Demirtas, A. Rios-Tascon, and L. McAllister. CYTools: A Software Package for Analyzing Calabi-Yau Manifolds, 2022.
- [14] Decker, W. and Eder, C. and Fieker, C. and Horn, M. and Joswig, M., editor. *The OSCAR book*. 2024.
- [15] D. Eisenbud and J. Harris. *3264 and All That: A Second Course in Algebraic Geometry*. Cambridge University Press, 2016.
- [16] E. M. Feichtner and S. Yuzvinsky. Chow rings of toric varieties defined by atomic lattices. *Inventiones Mathematicae*, 155(3):515–536, mar 2004.
- [17] D. R. Grayson and M. E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at <http://www.math.uiuc.edu/Macaulay2/>, 2023.
- [18] C. Jordan, M. Joswig, and L. Kastner. Parallel Enumeration of Triangulations. *The Electronic Journal of Combinatorics*, 2018. <https://polymake.org/doku.php/mptopcom>.
- [19] M. Kreuzer and H. Skarke. Classification of reflexive polyhedra in three-dimensions. *Adv. Theor. Math. Phys.*, 2:853–871, 1998.
- [20] M. Kreuzer and H. Skarke. Complete classification of reflexive polyhedra in four-dimensions. *Adv. Theor. Math. Phys.*, 4:1209–1230, 2000.
- [21] OSCAR – Open Source Computer Algebra Research system, Version 0.12.0-DEV. <https://www.oscar-system.org>, 2023.
- [22] S. Payne. Moduli of toric vector bundles. *Compositio Mathematica*, 144(5):1199–1213, Sept 2008.
- [23] C. Pegel. Chow rings of toric varieties. Master’s thesis, University of Bremen, Faculty of Mathematics, Sept 2014. Refereed by Prof. Dr. Eva Maria Feichtner and Dr. Emanuele Delucchi.
- [24] J. Rambau. TOPCOM: Triangulations of Point Configurations and Oriented Matroids. *Proceedings of the International Congress of Mathematical Software*, 2002.
- [25] SageMath, the Sage Mathematics Software System (Version 9.8). <https://www.sagemath.org>, 2023.
- [26] T. Weigand. F-theory. *PoS, TASI2017:016*, 2018.

Entropie und Kryptographie^a

Kemal Öztas, Quant-X Security & Coding GmbH
Christian Tobias, MTG AG
Mehrzaad Firoozi, Fraunhofer IPMS

kemal.oeztas@quant-x-sec.com
christian.tobias@mtg.de
mehrzaad.firoozi@ipms.fraunhofer.de



^aDer Artikel wurde nach dem Druck des Rundbriefs aktualisiert. Dies ist die aktualisierte, online veröffentlichte Version.

Mit der freundlichen Unterstützung von Prof. Dr. J. Krämer,

Fakultät für Informatik und Data Science, Universität Regensburg und Xenia Bogomolec, CEO von Quant-X Security & Coding GmbH in Hannover.

Einführung

In diesem Artikel wird die Rolle von Zufallswerten im Kontext von Vertraulichkeit, Integrität und Authentizität von Daten in heutigen digitalen Kommunikationssystemen dargestellt. Vertraulichkeit, Integrität und Authentizität werden mit kryptographischen Algorithmen realisiert. Kryptographische Algorithmen, deren Sicherheit auf mathematischen Problemen beruhen, sind ein Teilgebiet der Computeralgebra. Diese machen den Großteil der Verschlüsselungen in heutigen digitalen Systemen aus. Valide Schlüssel für jede kryptographische Methode sind Zufallswerte (symmetrische Schlüssel) oder beruhen zumindest auf solchen (asymmetrische Schlüssel). Diese Zufallswerte wiederum sind Ausgaben als Bitfolgen von (physikalischen) Quellen mit hoher Entropie. Die Entropie der genutzten Quelle bestimmt die Unvorhersagbarkeit der verschlüsselten Daten. Ohne die Nutzung einer physikalischen Quelle mit ausreichender Entropie ist jegliche algorithmische Komplexität in Kryptosystemen wertlos. Die Qualität einer Entropiequelle kann jedoch niemals bewiesen, sondern nur nach physikalischen Eigenschaften und mit statistischen Methoden zur Überprüfung des Outputs abgeschätzt werden.

Sicherheitsniveau eines Kryptosystems

Der Begriff des Sicherheitsniveaus eines Kryptosystems ist in der heutigen Zeit für Organisationen aus mehreren Gründen wichtig. Zum einen ist es global ein Kriterium dafür, welche Algorithmen noch als valide, und welche als veraltet betrachtet werden. Bei einem erfolgreichen Angriff auf Daten, die mit veralteten Algorithmen verschlüsselt wurden, kann die verantwortliche Stelle unter Umständen von regulatorischen Einrichtungen zur Rechenschaft gezogen werden. Innerhalb der als valide klassifizierten Kryptosysteme hat eine Organisation je-

weils die Möglichkeit, das Sicherheitsniveau aufgrund der Kritikalität der Daten und Performanz der verarbeitenden Systeme zu wählen. Z. B. werden für mobile Systeme oft kleinere Schlüssel und ein damit einhergehendes niedrigeres Sicherheitsniveau gewählt.

Nun zum mathematischen Aspekt: Das Sicherheitsniveau eines Kryptosystems entspricht dem binären Logarithmus des Rechenaufwands um den Klartext mit der effizientesten bekannten Methode zu einem verschlüsselten Text zu finden (z.B. 128 bits bei 2^{128} notwendigen Operationen zur Entschlüsselung eines verschlüsselten Texts). Die effizienteste Methode ist im idealen Fall ein Brute Force Angriff, d. h. das Durchtesten aller möglichen Schlüssel. Es kann aber auch eine neue Methode sein, um das dem Kryptosystem zugrundeliegende mathematische Problem zu lösen.

Algorithmus	Schlüssellänge	Sicherheitsniveau (klassisch)	Sicherheitsniveau (Quantum)	Quantenangriff
RSA-1024	1024 bits	80 bits	broken	Shor
RSA-2048	2048 bits	112 bits	broken	Shor
ECC-256	256 bits	128 bits	broken	Shor
ECC-521	521 bits	256 bits	broken	Shor
AES-128	128 bits	128 bits	64 bits	Grover
AES-256	256 bits	256 bits	128 bits	Grover

Quantenalgorithmen wie Shor und Grover bieten solche effizientere Methoden, um bisherige Verschlüsselungen zu brechen und damit diesen Aufwand zu reduzieren, sofern geeignete Quantenprozessoren zur Verfügung stehen. Im Fall des Shor Algorithmus gelten Primzahlfaktorisation (RSA), die Berechnung des Diskreten Logarithmus (Diffie-Hellmann und Elliptische Kurven oder eine Kombination davon) bezüglich des Sicherheitsniveaus gegen Quantencomputer als gebrochen (engl. *broken*), da dadurch ein Angriff in Polynomialzeit, viel mächtiger als ein Brute-Force-Angriff in Exponentialzeit, vorgestellt wird. Der Klassifizierung von

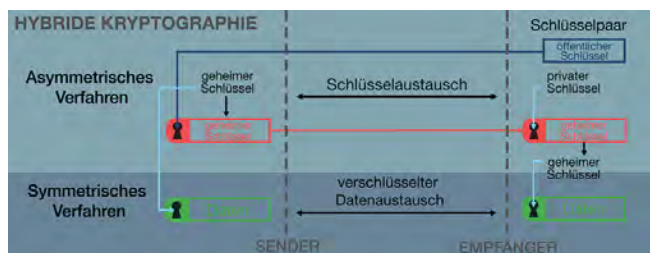
(engl. *broken*) liegt ein Sicherheitsniveau von 30 – 60 bit zugrunde.

Kryptographische Schlüssel und Schlüsselraum

Kryptographische Schlüssel sind Bitfolgen, die von dem kryptographischen Algorithmus verwendet werden, um Klartext in verschlüsselten Text umzuwandeln und umgekehrt. Der verschlüsselte Text soll so zufällig wie möglich aussehen. Der Schlüsselraum ist die Menge aller möglichen Schlüssel eines Kryptosystems.

Es gibt zwei Arten von kryptographischen Verfahren, symmetrische und asymmetrische. Bei der symmetrischen Kryptographie gibt es nur geheime Schlüssel, die sowohl zum Verschlüsseln als auch zum Entschlüsseln verwendet werden. Solche Schlüssel müssen zu jeder Zeit geheim gehalten werden. Bei asymmetrischen Verfahren gibt es private und öffentliche Schlüssel. Mit öffentlichen Schlüsseln können Daten nur verschlüsselt, aber nicht entschlüsselt werden. Daher müssen diese nicht geheim gehalten werden.

Kerckhoffs Prinzip besagt, dass die Sicherheit eines Kryptosystems nicht von der Geheimhaltung des Algorithmus abhängen darf, sondern nur von der Geheimhaltung des privaten bzw. geheimen Schlüssels. Dieses Prinzip schafft zum einen die Möglichkeit, Kryptosysteme in globalen Standardisierungsverfahren zu validieren. Zum anderen haben internationale Experten die Möglichkeit, Schwachstellen zu finden und entsprechende Kryptosysteme als veraltet zu deklarieren. Beides erhöht die Informationssicherheit für Nutzer auf der ganzen Welt. Die Geheimhaltung der privaten Schlüssel hängt wiederum direkt von der Qualität der genutzten Entropiequellen ab. Bei einer minderwertigen Entropiequelle ist der resultierende Schlüsselraum kleiner und die geheimen Schlüssel können einfacher erraten werden. Es reicht also nicht, gute kryptographische Softwarebibliotheken zu verwenden. Es spielt eine große Rolle, auf welchem System sie im Einsatz sind.



Bei einem hybriden kryptographischen Verfahren werden geheime Schlüssel für die Datenverschlüsselung mit einem symmetrischen Verfahren und private Schlüssel für den Schlüsselaustausch durch ein asymmetrisches Verfahren eingesetzt. Die Vorteile von hybriden Kryptosystemen sind vielfältig. Da asymmetrische Verfahren etwa 10.000 mal langsamer sind als symmetrische Verschlüsselungen, verschlüsselt man nur einen temporären, sogenannten *Session Key* asymmetrisch. Dieser wird dann genutzt, um die in der Regel viel längeren Nachrichten während eines limitierten

Zeitraums symmetrisch zu verschlüsseln. In der Folge kann man große verschlüsselte Datenmengen effizient mit Partnern austauschen. Da die Session Keys nur für kurze Zeiträume gültig sind, führt ein gehackter Session Key nur zur Offenlegung der entsprechenden Daten. Ein Beispiel für ein hybrides Protokoll ist *TLS*, womit unser globaler Webtraffic verschlüsselt wird. Webtraffic beinhaltet heutzutage auch viele Chats und E-Mails. Letztere werden meist zumindest von E-Mailserver zu E-Mailserver und von E-Mailserver zu einem Desktop-client mit TLS verschlüsselt. Dies bedeutet jedoch, dass die involvierten E-Mail Provider mitlesen können. Wenn man also eine E-Mail von Google Mail zu GMX versendet, können beide Provider mitlesen, außer, man nutzt eine zusätzliche Verschlüsselung wie z. B. PGP.

Entropie und Schlüsselraum

Wir haben eben erwähnt, dass die genutzte Entropiequelle den tatsächlichen Schlüsselraum eines Kryptosystems bestimmt. Bei AES-256 besteht der Schlüsselraum beispielsweise aus $2^{256} \approx 1.1 \times 10^{77}$ Schlüsseln. Als erstes Zufälligkeitskriterium von einigen Entropie-Testsuiten wird erwartet, dass die resultierenden Bitfolgen statistisch ungefähr aus gleich vielen Nullen und Einsen bestehen [1]. Bei Entropiequellen, die dieses Kriterium nicht erfüllen, verfügen auch die mit dieser Quelle erzeugten Schlüssel über ein Bias, das bei der Suche nach dem Schlüssel ausgenutzt werden kann. Damit wäre es ggf. möglich, Schlüssel effizienter als mit der Brute-Force Methode zu ermitteln. Als Resultat schwächt ein solcher mangelhafter Zufallszahlengenerator die Sicherheit des Gesamtsystems.

Das obige Zufälligkeitskriterium dient dabei nur als Beispiel. Andere Schwächen des Zufallszahlengenerators (wie etwa periodische Bitfolgen oder ähnliche Muster) können ebenfalls zu Schlüsseln führen, die nicht mehr gleichverteilt sind und deren Vorhersagbarkeit erhöhen.

Mask Attacks (Maskenangriffe) nutzen beim Angriff auf Passwörter aus, dass Menschen gerne Muster oder Zeichen in Passwörtern verwenden, die sie sich gut merken können. Dazu gehören zum Beispiel Wörter oder Jahreszahlen. Dadurch gibt es (wie im Falle von schlechten Zufallszahlengeneratoren) Passwörter, die wahrscheinlicher sind als andere. Anstatt also alle möglichen Passwörter durchzuprobieren (wie bei Brute-Force-Angriffen) wird bei Mask Attacks versucht, die Menge der im Angriff probierten Passwörter auf einen überschaubaren Teil der möglichen Passwörter zu reduzieren, die mit einer höheren Wahrscheinlichkeit von menschlichen Nutzern gewählt werden.

Entropie und Passwörter

Der Begriff Entropie wird auch als Maß für die Zufälligkeit von Passwörtern verwendet. Analog zu kryptographischen Schlüsseln gibt sie an, wie schwer es ist, ein Passwort mittels Durchprobieren zu erraten (Brute-Force Angriff). Passwörter sind meist weniger zufällig, als ihre Nutzer glauben:

- Ein 8-stelliges Passwort bestehend aus Klein- und Großbuchstaben und Sonderzeichen hat eine maximale Entropie von ca. 52 Bits. Um 120 Bits an Entropie erreichen zu können, müsste das Passwort etwa 18 Zeichen haben.
- Wird ein Passwort zum Schutz eines kryptographischen Schlüssels verwendet oder der Schlüssel aus dem Passwort abgeleitet, so muss das Passwort mindestens über so viel Entropie verfügen, dass es der Stärke des betreffenden Schlüssels entspricht.

Erzeugung von Zufallszahlen

Zufällige Bitfolgen können durch verschiedene Konstruktionen erzeugt werden.

- Ein *PRNG* (*Pseudo random number generator*) erzeugt als Softwarekomponente deterministisch pseudozufällige Bitfolgen aus gegebenem Startwert (Seed), und bietet eine hohe Outputrate an. Ein PRNG benötigt eine Bitfolge von hoher Zufälligkeit, um einen Zufallswert hoher Qualität zu generieren. Insofern ist ein PRNG allein keine valide Quelle für kryptographische Geheimnisse.
- Ein *TRNG* (*True random number generator*) erzeugt Bitfolgen auf Basis physikalischer Phänomene oder nicht-deterministischem Nutzer- und Hardware-Verhalten.
- Ein *QRNG* (*Quantum random number generator*) ist Ausprägung eines physikalischen TRNGs, wobei zur Erzeugung von zufälligen Bitfolgen quantenmechanische Effekte verwendet werden.

Zufälligkeits-Tests

Die Zufälligkeit von Bitfolgen kann durch statistische Hypothesentests überprüft werden, wodurch beim Aufkommen statistischer Auffälligkeiten die Zufälligkeitshypothese verworfen wird. Eine Reihe solcher Tests wird im NIST SP 800-22 Test Suite präsentiert [1]. Beispielsweise wird bei dem sogenannten Monobittest das Verhältnis von Einsen und Nullen getestet, bei dem Runs Test wird die Anzahl der ununterbrochenen Folgen identischer Bits gezählt.

Die mathematische Entwicklung neuer Testalgorithmen ermöglicht es, Bitfolgen auf weitere Muster zu testen und schlechte Zufallsquellen auszusortieren. Ein interessanter Fall stellt der lineare Kongruenzgenerator RANDU dar,

$$X_0 \text{ ungerade, } X_1 = (65539X_0) \bmod 2^{31},$$

wobei die ausgegebenen Bits, z.B. als 3-Tupeln dargestellt, sich auf parallelen Ebenen im dreidimensionalen

Raum sammeln [2]. Der sogenannte Spektraltest liefert durch die Distanz dieser Ebenen eine Aussage über die Qualität von linearen Kongruenzgeneratoren. Es wurde anschließend festgestellt, dass dieser früher oft verwendete Kongruenzgenerator RANDU an den meisten Tests in drei Dimensionen scheitert [3].

Während mehrere frequenzbasierte Tests in der NIST Test Suite den zentralen Grenzwertsatz berücksichtigen, entwickelte Yongge Wang sogenannte LIL Tests, die auf das Gesetz des iterierten Logarithmus basieren:

$$\limsup_{n \rightarrow \infty} \frac{|\sum_{k=0}^n X_k|}{\sqrt{2n \log \log n}} = 1 \quad \text{fast sicher,}$$

wobei $(X_k)_{k \in \mathbb{N}}$ eine Folge von Zufallsvariablen mit Erwartungswert 0 und Varianz 1 ist. Durch die LIL Tests können noch weitere Bitfolgen festgestellt werden, die die NIST Tests bestehen, sich jedoch von der Gleichverteilung unterscheiden [4].

Beispiele von Schwachstellen

Es gab in der Vergangenheit wiederholt Schwachstellen in Kryptosystemen wegen der Nutzung von minderwertigen Entropiequellen. Einige Beispiele sind:

- *Fehlende Überprüfung von Zufallswerten in WebCrypto (2022)*: Die Änderungen der Entropiequelle von WebCrypto mit unzureichender Überprüfung der Zufallszahlen führte zu Schwachstellen in Node.js.¹
- *Unsichere AES-Schlüsselerzeugung für Zugangssysteme (2021)*: Es wurde festgestellt, dass mit eingeschränkter Entropie generierte AES-Schlüssel vorhersagbar sind, wodurch Angreifer NFC-Tags klonen und sich unbefugten Zugriff zu gesicherten Bereichen verschaffen können.²
- *Wiederholtes Verwenden von Zufallszahlen auf virtuellen Maschinen (2017)*: Eine akademische Studie untersucht eine neue Art und Weise, wie Zufallszahlengeneratoren aufgrund der Wiederverwendung von Snapshots virtueller Maschinen fehlschlagen und vorhersagbar werden.³



Ergebnisse des Projekts (Quant-ID), ein Konsortialprojekt zur Integration von QRNG und Post-Quanten Kryptographie in Identity Access Management Protokolle für kritische Infrastrukturen, gefördert

mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 16KISQ108K.

¹<https://github.com/advisories/GHSA-p36x-w6hr-88jp>

²<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-34600>

³<https://www.ndss-symposium.org/wp-content/uploads/2017/09/rist.pdf>

Literatur

- [1] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Special Publication 800-22 Revision 1a, NIST, 2010.
- [2] George Marsaglia, *Random Numbers Fall Mainly in the Planes*, PNAS USA, Vol. 61, No. 1 (Sep. 15, 1968), pp. 25-28
- [3] Donald E. Knuth, *The Art of Computer Programming*, 3. Edition. 23. Printing. Volume 2: Seminumerical Algorithms. Addison-Wesley, Boston MA u. a. 2008, ISBN 978-0-201-89684-8, S. 93ff.
- [4] Yongge Wang, *On the Design of LIL Tests for (Pseudo) Random Generators and Some Experimental Results*, <https://webpages.charlotte.edu/yonwang/>

Publikationen über Computeralgebra

Subject: Special issue “Computational Algebra”

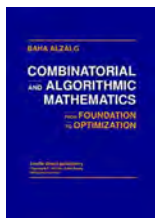
Dear Colleagues,

the Springer Journal *Beiträge zur Algebra und Geometrie* (Contributions to Algebra and Geometry) publishes high quality articles reflecting the interplay between algebra and geometry. Computational Algebra fits very well into both, the aims and scope of our journal and the renewal process of our editorial board. This is why we take the opportunity to design a special issue “Computational Algebra”. We invite researchers to submit their articles for this special issue by end of June 2023.

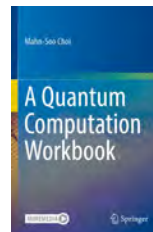
We hope for many excellent contributions. In behalf of the managing editors

Gabriele Nebe

Neuerscheinungen:



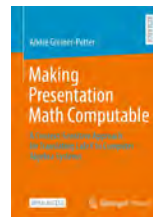
Baha Alzalg,
Combinatorial and Algorithmic Mathematics: From Foundation to Optimization,
Kindle Direct Publ.,
Sep. 2022, 543 Seiten,
ISBN 979-8353925729



Mahn-Soo Choi,
A Quantum Computation Handbook,
Taschenbuch, März 2023,
Springer Verlag,
446 Seiten,
ISBN 978-3030912161



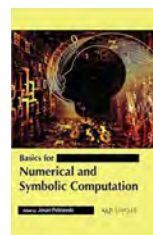
Murat Durmus,
Eine Einführung in die 42 am häufigsten angewandten Machine Learning Algorithmen,
Indep. Publ., Jan. 2023,
198 Seiten,
ISBN 979-8375634890



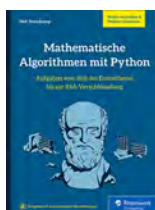
André Greiner-Petter,
Making Presentation Math Computable: Translating LaTeX to Computer Algebra Systems,
Springer Vieweg, Jan. 2023,
215 Seiten,
ISBN 978-3658404727



Andreas Öchsner, Resam Makvandi,
Numerische technische Optimierung: Anwendung des CAS Maxima,
Springer Vieweg, Jan. 2023,
249 Seiten,
ISBN 978-3031150142



Jovan Pehcevski,
Basics for Numerical and Symbolic Computation
Arcler Press, Dec. 2022,
277 Seiten,
ISBN 978-1774694459



Veit Steinkamp,
Mathem. Algorithmen mit Python: Aufgaben vom Sieb des Eratosthenes bis zur RSA Verschlüsselung,
Rheinwerk Computing, Mai 2022
512 Seiten,
ISBN 978-3836285742

Die Rubrik Publikationen ist nicht allein auf eine Liste von Neuerscheinungen und Neuauflagen beschränkt. Sie lebt vor allem von fundierten Rezensionen von Fachgruppenmitgliedern für Fachgruppenmitglieder, die wir an dieser Stelle gerne abdrucken. Sollte eines der oben genannten Bücher, insbesondere eine der Neuerscheinungen, Ihr Interesse geweckt haben, und Sie möchten dieses für den Computeralgebra-Rundbrief besprechen, nehmen Sie bitte Kontakt zu Martin Kreuzer (martin.kreuzer@uni-passau.de) auf.

Gregory V. Bard

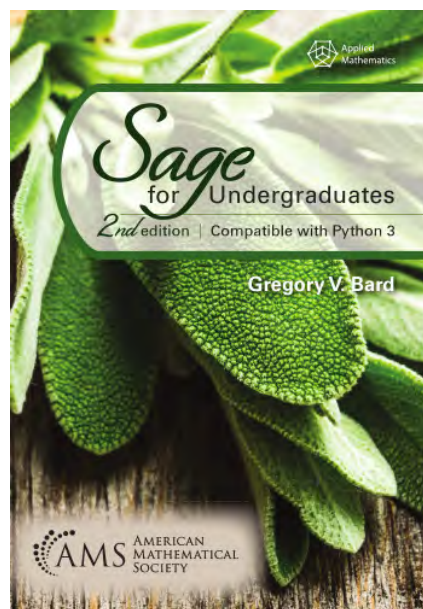
Sage for Undergraduates, 2. Auflage

Obwohl die erste Auflage dieses Buchs aus dem Jahr 2014 stammt und somit noch nicht sehr alt ist, war eine zweite Auflage unumgänglich, denn mit dem Jahreswechsel 2019/2020 ist die Programmiersprache Python 2, auf der die ältere Version des Computeralgebrasystems Sage beruhte, durch die nicht rückwärtskompatible Version Python 3 ersetzt worden, so dass ein Großteil der alten Sage Programme nicht mehr funktionierte. In mühevoller Detailarbeit hat der Autor daher seine an Bachelorstudierende gerichtete Einführung in dieses System für die neue Version aktualisiert und ergänzt.

Das erste Kapitel enthält dabei die essentiellen Grundlagen zur Verwendung von Sage, die jeder Anwender kennen sollte. Im zweiten Kapitel werden verschiedene kleine Projekte vorgestellt, wie man mit diesem Hilfsmittel Probleme aus diversen Anwendungsbereichen lösen kann, wie z.B. der Kryptographie, der Physik oder der industriellen Optimierung. Das dritte Kapitel hat die Grafikfähigkeiten von Sage zum Thema: Funktionsgraphen, Kurven plotten, Konturlinien, etc. Das 3D Plotten und farbige Darstellungen sind jedoch in ein Online Tutorium zum Buch ausgelagert, da sie sich in Buchform nicht so gut darstellen lassen.

Der zentrale Teil des Buchs ist Kapitel 4, in dem die fortgeschrittenen Funktionen von Sage zur Sprache kommen. Multivariable Funktionen und Gleichungen, Vektoren und Matrizen, Zahlentheorie, Ableitungen und Limites, Taylorpolynome, lineare Optimierung, ja sogar Differentialgleichungen, Laplace-Transformationen und Kettenbrüche: so ziemlich alle mathematischen Objekte, denen Bachelorstudierende üblicherweise begegnen, werden symbolisch behandelt. Alle Berechnungen werden anhand ausführlich erläuterter Beispiele vorgenommen.

Die letzten beiden Kapitel sind danach mehr in Tutorienform geschrieben und sollten sequentiell gelesen werden. Kapitel 5 hat dabei die Sage Programmierung in Python 3 zum Thema und in Kapitel 6 dreht sich alles um die Verwendung von Sage zur Generierung interaktiver Webseiten. Drei Anhänge zur Fehlersuche, zur SageMathCloud und zu den Unterschieden zwischen Python 2 und Python 3 runden das Buch ab.



Gregory Bards Werk bietet einen ausgezeichneten Einstieg in das Sage System mit vielen konkreten Hilfen, Tipps und vorgeführten Beispielen. Es ist etwas schade, dass sein Schwerpunkt dabei etwas mehr auf der Analysis zu liegen scheint und die klassischen Themen der Computeralgebra wie Polynomideale, Gröbner-Basen oder algebraische Gleichungssysteme nicht oder nur am Rande vorkommen. Dabei bietet Sage für diese Bereiche Schnittstellen zu mächtigen Systemen wie Gap 3, Macaulay 2 oder Singular an. Für einen Einstieg in das Sage System, der für Bachelorstudierende und nicht auf die Computeralgebra spezialisierte Mathematiker geeignet ist, stellt das Buch jedoch eine hervorragende Basis dar und kann uneingeschränkt empfohlen werden. Die Tatsache, dass es in Englisch geschrieben ist, sollte für diesen Leserkreis auch kein Problem darstellen. Der AMS Listenpreis von 59\$ ist recht stolz, erniedrigt sich aber für AMS Mitglieder und während der gelegentlichen Sonderangebote.

Martin Kreuzer (Passau)

Alheydis Geiger: Tropical Geometric Counting Problems
Betreuerin: Hannah Markwig (Tübingen)

Weitere Gutachter: Bernd Sturmfels (Berkeley, MPI MiS Leipzig), Thorsten Theobald (Frankfurt)

Juli 2022

Abstract: A degeneration of algebraic counting problems to tropical geometry produces new concepts and computational approaches. The challenge is to find methods to lift the results in the degeneration back to the original problem. This thesis contributes towards the fundamental research in this area by investigating two geometric counting problems via tropical geometry. The first part of this thesis closes a research gap on the recovery of the classical count of 4, 8, 16 or 28 real bitangents to smooth quartic curves by Plücker and Zeuthen from tropical geometry. Building on previous research results, we develop methods to produce an understanding of the global lifting of tropical bitangents over certain real closed fields. Moreover, we establish a computational tool in polymake for the investigation of tropical bitangents and their lifting behavior, and present results of analyses using this tool. The second part of this thesis explores the question of tropically counting binodal surfaces. We exploit tropical floor plans, a recent enumerative tool from tropical geometry. We prove that tropical floor plans recover the algebraic counts of plane curves, while for surfaces the current technique is not sufficient to asymptotically recover the second order term. To improve the approach for surfaces, we provide a classification of the smallest examples of polytopes that can appear as Newton polytopes of a binodal surface together with instructions how to use them for tropical counting. This investigation is of computational nature and is aided by functions written for the computer algebra system OSCAR. Furthermore, we extend the definition of tropical floor plans that contains the newly found cases. Additionally, we show that these smallest cases contribute to the third order term of the asymptotic count.

Birte Johansson: The inductive McKay–Navarro condition for finite groups of Lie type

Betreuer: Gunter Malle (Kaiserslautern)

Weitere Gutachter: Mandi A. Schaeffer Fry (Denver)

November 2022

Abstract: In the representation theory of finite groups, the so-called local-global conjectures assert a relation between the representation theory of a finite group and one of its local subgroups. The McKay–Navarro conjecture claims that the action of a set of Galois automorphisms on certain ordinary characters of the local and global group is equivariant. Navarro, Späth, and Vallejo reduced the conjecture to a problem about simple groups in 2019 and stated an inductive condition that has to be verified for all finite simple groups.

In the first part of this thesis, we give an introduction to the character theory of finite groups and state the McKay–Navarro conjecture and its inductive condition. Furthermore, we recall the definition of finite groups of Lie type and present

results regarding their structure and their representation theory.

In the second part, we verify the inductive McKay–Navarro condition for various families of finite groups of Lie type. In defining characteristic, most groups have already been considered by Ruhstorfer. We show that the inductive condition also holds for the groups with exceptional graph automorphisms, the Suzuki and Ree groups, the groups $B_n(2)$ for $n \geq 2$, as well as for the simple groups of Lie type with non-generic Schur multiplier in their defining characteristic. This completes the verification of the inductive McKay–Navarro condition in defining characteristic.

We further consider the Suzuki and Ree groups and verify the inductive condition for all primes. On the way, we show that there exists a Galois-equivariant Jordan decomposition for their irreducible characters. Moreover, we consider some families of groups of Lie type that do not admit a generic choice of a local subgroup. We show that the inductive condition is satisfied for the prime $\ell = 3$ and the groups $\mathrm{PSL}_3(q)$ with $q \equiv 4, 7 \pmod 9$, $\mathrm{PSU}_3(q)$ with $q \equiv 2, 5 \pmod 9$, and $G_2(q)$ with $q \equiv 2, 4, 5, 7 \pmod 9$. Further, we verify the inductive condition for the prime $\ell = 2$ and the groups $G_2(3^f)$ for $f \geq 1$, ${}^3D_4(q)$, and ${}^2E_6(q)$ where q is an odd prime power.

Laura Voggesberger: Nilpotent Pieces in Lie Algebras of Exceptional Type in Bad Characteristic

Betreuer: Gunter Malle (Kaiserslautern)

Weitere Gutachter: Meinolf Geck (Stuttgart)

Dezember 2022

Abstract: In group theory, a big and important family of infinite groups is given by the algebraic groups. These groups and their structures are already well-understood. In representation theory, the study of the unipotent variety in algebraic groups — and by extension the study of the nilpotent variety in the associated Lie algebra — is of particular interest.

Let G be a connected reductive algebraic group over an algebraically closed field k , and let $\mathrm{Lie}(G)$ be its associated Lie algebra. By now, the orbits in the nilpotent and unipotent variety under the action of G are completely known and can be found for example in a book of Liebeck and Seitz. There exists, however, no uniform description of these orbits that holds in both good and bad characteristic. With this in mind, Lusztig defined a partition of the unipotent variety of G in 2011. Equivalently, one can consider certain subsets of the nilpotent variety of $\mathrm{Lie}(G)$ called the *nilpotent pieces*. This approach appears in the same paper by Lusztig in which he explicitly determines the nilpotent pieces for simple algebraic groups of classical type. The nilpotent pieces for the exceptional groups of type G_2, F_4, E_6, E_7 , and E_8 in bad characteristic have not yet been determined. This thesis gives an introduction to the definition of the nilpotent pieces and presents a solution to this problem for groups of type G_2, F_4, E_6 , and partly for E_7 , relying heavily on computational work.

CATS 2023

Tuscaloosa, Alabama, USA, 15.04. – 16.04.2023

cats-math.github.io

The first edition of CATS (Commutative Algebra in The South) will be held in April 15-16, 2023, in Tuscaloosa, Alabama. Invited Speakers are Rankeya Datta (University of Missouri), Florian Enescu (Georgia State University), Tàì Hà (Tulane University), Patricia Klein (Texas A & M), Paolo Mantero (University of Arkansas), Frank Moore (Wake Forest University) and Christopher Manon (University of Kentucky).

GAMM - 93rd Annual Meeting

Dresden, 30.05. – 02.06.2023

jahrestagung.gamm-ev.de

The GAMM Annual Meeting 2023 will be hosted by Technische Universität Dresden.

It will take place from 30th of May until 2nd of June in the historical City of Dresden.

Submission of Abstracts will be open by 1st of December 2022.

Computeralgebra-Tagung der Fachgruppe

Hannover, 31.05. – 02.06.2023

konferenz.uni-hannover.de/event/83

Die 10. Computeralgebra-Tagung der Fachgruppe findet vom 31. Mai bis 2. Juni 2023 an der Fakultät für Mathematik und Physik der Leibniz Universität Hannover statt. Die Organisatoren sind Prof. Anne Fröhbis-Krüger und Prof. Michael Cuntz. Es wird wieder ein Nachwuchspreis verliehen. Siehe ausführliche Ankündigung der Tagung auf Seite 7.

28 National School on Algebra

Bukarest, Rumänien, 24.06. – 28.06.2023

sites.google.com/view/ronsa/the-28th-national-school-on-algebra

The 28 National School on Algebra will take place between 24-28 June 2023 at the University of Bucharest and is a satellite event of the Tenth Congress of the Romanian Mathematicians.

The aim of the school is to present recent directions of research in commutative algebra highlighting connections with geometry and combinatorics. There will be series of lectures delivered by the invited speakers: Klaus Altmann, Yairon Cid-Ruiz (Some lectures on multidegrees), Alexandru Constantinescu (Polyhedra, Lattice Structures, and Extensions of Semigroups), Alexandra Seceleanu (Symbolic powers in algebra and geometry), and also time for contributed talks by the participants.

Conference on Rings and Factorizations

Graz, Österreich, 10.07. – 14.07.2023

imsc.uni-graz.at/rings2023

From July 10th to 14th, 2023 the conference Rings and Factorizations 2023 will take place at the Institute of Mathematics and Scientific Computing at the University of Graz in Graz, Austria.

The conference continues a series of ring theory conferences in Graz from 2012, 2014, 2016, 2018, and 2021. It will include Special Sessions dedicated to Prof. Matej Brešar and Prof. Sophie Frisch on the occasion of their 60th birthdays.

ACA 2023

Warschau, Polen, 17.07. – 21.07.2023

iit.sggw.edu.pl

[/instytut-informatyki-technicznej/aca2023](https://instytut-informatyki-technicznej/aca2023)

The ACA conference series is devoted to promoting all kinds of computer algebra applications, and encouraging the interaction of developers of computer algebra systems and packages with researchers and users (including scientists, engineers, educators, and mathematicians).

Traditionally, ACA is organized in scientific sessions covering all foundational aspects and applications of computer algebra. Details about the planned special sessions will become available soon as these become accepted.

ISSAC 2023

Tromsø, Norwegen, 24.07. – 27.07.2023

www.issac-conference.org/2023

The International Symposium on Symbolic and Algebraic Computation (ISSAC) is the premier conference for research in symbolic computation and computer algebra. ISSAC 2023 will be the 48th meeting in the series, which started in 1966 and has been held annually since 1981. The conference presents a range of invited speakers, tutorials, short communications, software demonstrations and vendor exhibits with a center-piece of contributed research papers.

SC² Workshop 2023

Tromsø, Norwegen, 28.07.2023

www.sc-square.org/CSA/workshop8.html

The 8th International Workshop on Satisfiability Checking and Symbolic Computation will be held on July 28, 2023, at The Arctic University of Norway (UiT) in Tromsø. It will be collocated with ISSAC 2023.

IC-MSQUARE-2023

Belgrad, Serbien, 28.08. – 31.08.2023

www.icmsquare.net

The 12th International Conference on Mathematical Modeling in Physical Sciences is to be held at Belgrade, Serbia during August 28-31, 2023. The conference aims to promote the knowledge and the development of high-quality research in mathematical fields that have to do with the applications of other scientific fields and the modern technological trends that appear in them, these fields being those of Physics, Chemistry, Biology, Medicine, Economics, Sociology, Environmental sciences etc..

CASC 2023

Havanna, Kuba, 28.08. – 01.09.2023

www.casc-conference.org

The tools of Scientific Computing play an important role in the natural sciences and engineering. Computer Algebra Systems and the underlying algorithms for Symbolic Computation play an increasingly important role within Scientific Computation. The CASC workshop series has been running for over two decades to explore the interaction of these topics, their implementation, and their application.

SYNASC 2023

Nancy, Frankreich, 11.09. – 14.09.2023

synasc.ro/2023

SYNASC aims to stimulate the interaction among multiple communities focusing on defining, optimizing and executing complex algorithms in several application areas. The focus of the conference then ranges from symbolic and numeric computation to formal methods applied to programming, artificial intelligence, distributed computing and computing theory. The interplay between these areas, in fact, is essential in the current scenario where economy and society demand for the development of complex, data intensive, trustable and high performant computational systems.

Recent Trends in Computer Algebra

Paris, Frankreich, 18.09. – 11.12.2023

indico.math.cnrs.fr/category/588

This is a thematic trimester program at the Institut Henri Poincaré, Paris.

The conferences and courses of this program will also be accessible online. Main scientific event:

September 18th to December 11th, 2023: Recent Trends in Computer Algebra (IHP, Paris). Workshops:

September 25th to 29th: Fundamental Algorithms and Algorithmic Complexity (IHP, Paris) October 16th to 20th: Geometry of Polynomial System Solving, Optimization and Topology (IHP, Paris) December 4th to 8th: Computer Algebra for Functional Equations in Combinatorics and Physics (IHP, Paris)

DMV-Jahrestagung 2023

Ilmenau, 25.09. – 28.09.2023

www.tu-ilmenau.de/dmv2023

Die DMV-Jahrestagung findet reihum im Bundesgebiet an Universitäten statt, die sich auf die Ausrichtung beworben haben. Veranstaltet wird sie zur Förderung wissenschaftlicher Kontakte und beinhaltet eine Mitgliederversammlung, Mini-Symposien, einen Lehrentag, eine Studierendenkonferenz und ein kulturelles Rahmenprogramm.

Jahrestagung des SFB-TRR 195

Saarbrücken, 25.09. – 28.09.2023

www.computeralgebra.de/sfb

The third annual meeting of the SFB-TRR 195 has been scheduled for 2023.

It will take place from September 25-28, 2023 at Saarland University in Saarbrücken.

Further information will be provided as soon as possible.

GI-Jahrestagung 2023

Berlin, 26.09 – 29.09.2023

informatik2023.gi.de

Zum INFORMATIK FESTIVAL 2023, der offiziellen Jahrestagung der Gesellschaft für Informatik e.V. (GI), präsentieren wir uns dieses Jahr im neuen Gewand und bewegen uns einen Schritt Richtung Zukunft - ganz im Sinne des diesjährigen Leitthemas „Designing Futures: Zukünfte gestalten“. Die Gestaltung digitaler Technologien und informatischer Systeme sowie deren Gestaltungskraft für gesellschaftliche, kulturelle, soziale und wirtschaftliche Prozesse stehen im Fokus des INFORMATIK FESTIVALS 2023.

WICA II

CIRM Trento, Italien, 16.10. – 20.10.2023

mathstat.dal.ca/~faridi/WICAII

The workshop Women in Commutative Algebra II (WICA II) will take place at CIRM in Trento, Italy on October 16-20, 2023. This workshop will be entirely dedicated to working on research topics in commutative algebra in a collaborative environment.

Please see the WICA II website for a list of research topics and group leaders in 2023, as well as a link to the application form. If you are interested in participating and you have received or will receive your PhD before September 2023, please apply by May 1, 2023.



Workshop-Förderung der Fachgruppe:

Sie veranstalten einen Workshop zu einem Thema aus dem Bereich der Computeralgebra und könnten mit einer kleinen finanziellen Unterstützung den Workshop deutlich interessanter oder effektiver gestalten? Die Fachgruppe Computeralgebra unterstützt Workshops mit bis zu 1000,- Euro.

Anträge können mit einer kurzen Beschreibung des Workshops (ca. 1 DIN A4 Seite; kurze Beschreibung des Gebiets, Thema des Workshops, Zielgruppe, Budget-Planung) und einer Darstellung, inwiefern diese Förderung einen deutlich erkennbaren Beitrag zum Gelingen des Workshops und zur Nachwuchsförderung liefert, an die Sprecherin der Fachgruppe gerichtet werden:

anne.fruehbis-krueger@uni-oldenburg.de,

bitte „**Workshop-Förderung**“ im Betreff angeben.



Fachgruppenleitung Computeralgebra 2023–2026

**Sprecherin:**

Prof. Dr. Anne Frühbis-Krüger
Carl-von Ossietzky Universität Oldenburg
Carl-von-Ossietzky-Straße 11, 26129 Oldenburg
0441 798-3233
anne.fruehbis-krueger@uni-oldenburg.de
<https://uol.de/anne-fruehbis-krueger>

**Vertreterin der GI:**

Prof. Dr. Erika Abraham
RWTH Aachen
Ahornstr. 55, 52056 Aachen
0241 80-21242, -22243 (Fax)
abraham@cs.rwth-aachen.de
<https://ths.rwth-aachen.de/people/erika-abraham/>

**Fachreferent CA-Systeme und -Bibliotheken:**

Prof. Dr. Claus Fieker
RPTU Kaiserslautern-Landau
Gottlieb-Daimler-Straße, 67663 Kaiserslautern
0631 205-2392, -4427 (Fax)
fieker@mathematik.uni-kl.de
<https://www.mathematik.uni-kl.de/~fieker>

**Vertreter der DMV:**

Prof. Dr. Florian Heß
Carl-von Ossietzky Universität Oldenburg
Institut für Mathematik, 26111 Oldenburg
0441 798-2906, -3004 (Fax)
florian.hess@uni-oldenburg.de
<https://uol.de/florian-hess>

**Fachexperte SFB-TRR 195:**

Prof. Dr. Max Horn
RPTU Kaiserslautern-Landau
Gottlieb-Daimler-Straße, 67663 Kaiserslautern
0631 205-2730, -4427 (Fax)
mhorn@rptu.de
<https://www.quendi.de/de/mathe>

**Fachreferent Publikationen:**

Prof. Dr. Martin Kreuzer
Universität Passau
Innstr. 33, 94030 Passau
0851 509-3120, -3122 (Fax)
martin.kreuzer@uni-passau.de
<https://staff.fim.uni-passau.de/kreuzer/>

**Vertreterin der GAMM:**

Prof. Dr. Eva Zerz
RWTH Aachen
Pontdriesch 14/16, 52062 Aachen
0241 80-94544, -92108 (Fax)
eva.zerz@math.rwth-aachen.de
<https://www.math.rwth-aachen.de/~Eva.Zerz/>

**Stellvertretender Sprecher:**

Prof. Dr. Michael Cuntz
Leibniz Universität Hannover
Welfengarten 1, 30167 Hannover
0511 762-4252
cuntz@math.uni-hannover.de
<https://www.iazd.uni-hannover.de/de/cuntz>

**Fachreferentin Industrie:**

Xenia Bogomolec
Quant-X Security & Coding
Engelbosteler Damm 15, 30167 Hannover
0173 3031816
xb@quant-x-sec.com
<https://quant-x-sec.com>

**Fachreferent Physik:**

Dr. Thomas Hahn
Max-Planck-Institut für Physik
Föhringer Ring 6, 80805 München
089 32354-300, -304 (Fax)
hahn@feynarts.de
<https://wwwth.mpp.mpg.de/members/hahn>

**Fachreferent CA-Systeme und -Bibliotheken:**

Jun.-Prof. Dr. Tommy Hofmann
Universität Siegen
Walter-Flex-Straße 3, 57072 Siegen
0271-740-2868
tommy.hofmann@uni-siegen.de
<https://www.thofma.com/>

**Fachreferent Themen und Anwendungen:**

Prof. Dr. Gregor Kemper
Technische Universität München
Boltzmannstr. 3, 85748 Garching
089 289-17454, -17457 (Fax)
kemper@ma.tum.de
<https://www.math.cit.tum.de/algebra/kemper>

**Fachreferent Redaktion Rundbrief:**

Dr. Fabian Reimers
Technische Universität München
Boltzmannstr. 3, 85748 Garching
089 289-17474
reimers@ma.tum.de
<https://www.math.cit.tum.de/algebra/reimers>

TI-Nspire™ CX CAS Technologie

Gut vorbereitet in die Prüfung

Mit wenigen Handgriffen haben Sie bei allen TI-Nspire™ CX CAS Produkten den integrierten Prüfungsmodus aktiviert, einfach via Testcode. Ob am Taschenrechner, am iPad®, oder am PC.

Für umfassende Sicherheit bei allen Tests.

Sehen Sie sich kurze Demo-Videos zum Prüfungsmodus an:



SCAN ME



www.education.ti.com/de