Oktober 2022

Computeralgebra Rundbrief

> Ausgabe 71

- Tagung der Fachgruppe 2023 in Hannover
- Symmetric embeddings of the icosahedron
- Algebra in probabilistic reasoning
- Automated reasoning in the class

www.fachgruppe-computeralgebra.de

Maple™ Calculator

Verfügbar in Deutsch!

Warum warten?

Versuchen Sie es doch mal!

Available on the App Store | GET IT ON Google Play

www.maplesoft.com/calculator

Möchten Sie unsere Maple Calculator app ausprobieren, haben aber kein Maple?

Holen Sie sich eine 15-tägige Testlizenz und schauen Sie, was die Calculator app in Verbindung mit Maple leisten kann!

www.maplesoft.com/trial

Der leistungsstärkste Taschenrechner mit Lösungsschritten (und er ist kostenlos!)

Der Maple Taschenrechner ist ein kostenloser, leistungsstarker Kalkulator und ein vielseitiges Lernwerkzeug, das Ihnen Ergebnisse, 2D- und 3D-Diagramme und sogar Schritt-für-Schritt-Lösungen liefert! Ob Sie nun einfache Berechnungen durchführen oder an mathematischen Problemen auf Universitätsniveau arbeiten, der Maple Taschenrechner kann alles.

- Klicken Sie einfach, um Ihre Formeln einzugeben
- Für alle Zweige der Mathematik
- Schritt-für-Schritt-Lösungen erhalten
- Graphische Probleme und Ergebnisse
- Erhalten Sie Antworten, auch wenn Sie offline sind
- Validieren handschriftlicher Lösungen in Maple Learn
- Vermeiden Sie Transkriptionsfehler bei der Verwendung von Maple Desktop
- Spielen Sie Sumzle, wenn Sie eine Pause brauchen

Maplesoft
Mathematik • Modellierung • Simulation
A Cybernet Group Company

www.maplesoft.com | info@maplesoft.com • Tel: +49 241 980 977 30

© 2022 Maplesoft, ein Geschäftsbereich von Waterloo Maple Inc. Bei Maplesoft, Maple, Maple Learn und Maple Calculator handelt es sich jeweils um Warenzeichen von Waterloo Maple Inc. Alle anderen Warenzeichen sind Eigentum ihrer jeweiligen Inhaber.

Computeralgebra-Rundbrief

Nr. 71 Oktober 2022

Inhaltsverzeichnis

Inhalt . . . 3

Impressum . . . 4

Mitteilungen der Sprecher . . . 5

Kandidatinnen und Kandidaten für die Fachgruppenleitung . . . 6

Tagungen der Fachgruppe . . . 8

Themen und Anwendungen . . . 10
Symmetric embeddings of the icosahedron (D. Robertz) . . . 10
Algebra in probabilistic reasoning (T. Boege) . . . 15

Computeralgebra in der Hochschule . . . 21
Automated Reasoning in the Class (I. Drămnesc, E. Ábráham, T. Jebelean, G. Kusper, S. Stratulat) . . . 21

Computeralgebra und Industrie . . . 27
Local Inversion of maps: Black Box Cryptanalysis (V. Sule) . . . 27

Berichte über Arbeitsgruppen . . . 34
Explizite Methoden in Zahlentheorie und Algebra an der Universität Siegen (T. Hofmann) . . . 34

Publikationen über Computeralgebra . . . 35

Promotionen in der Computeralgebra . . . 36

Berichte von Konferenzen . . . 37

Hinweise auf Konferenzen . . . 38

Fachgruppenleitung Computeralgebra 2020–2023 . . . 39

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,

so schnell geht ein Wahlperiode zu Ende! Es ist schon wieder Zeit für die Vorstellung der Kandidatinnen und Kandidaten für die Fachgruppenleitung der nächsten drei Jahre, die gleich nach diesem Text auf den folgenden Seiten zu finden ist. Nach der erfolgreichen Erprobung der Online-Wahl in der letzten Wahlperiode wird die Wahl auch diesmal online stattfinden. Eine Mail mit den konkreten Wahlinformationen sollte Ihnen jetzt, wenn Sie den Rundbrief in Händen halten, bereits zugegangen sein. Den Wahlvorstand bilden dieses Mal Florian Heß und Eva Zerz, die als Vertreter der DMV bzw. GAMM nicht zu den gewählten Mitgliedern gehören. Wir hoffen, dass sich der bei der letzten Wahl sichtbare Trend zu einer hohen Wahlbeteiligung in diesem Jahr fortsetzt.

Die kurzfristige pandemiebedingte Umstellung der Computeralgebratagung im vergangenen März in ein Online-Format haben wir alle bedauert. Gerade bei einer Tagung für den wissenschaftlichen Nachwuchs fehlt in so einem Format der ungezwungene informelle Austausch bei den Kaffeepausen. Deshalb unternehmen wir bereits im Mai 2023 einen neuen Anlauf, eine Computeralgebratagung in Präsenz zu organisieren, und nehmen damit den ursprünglichen und durch die Pandemie gestörten zweijährigen Rhythmus wieder auf. Die Tagungsankündigung finden Sie auf Seite 8.

Der vorliegende Rundbrief ist recht umfangreich, so dass wir Sie nicht allzu lange mit diesen Mitteilungen aufhalten wollen. Einer der enthaltenen Artikel wirft einen etwas ungewöhnlichen Blick auf das Ikosaeder, siehe Seite 10. Weitere widmen sich algebraischen Aspekten der Entscheidungsfindung (Seite 15), und beleuchten einen Aspekt der Kryptanalyse (Seite 27). In der Rubrik Computeralgebra an der Hochschule schließlich findet sich ein Betrag zur Vermittlung von Themen des Automated Reasoning.

Wir wünschen Ihnen eine angenehme und anregende Lektüre.

Anne Frühbis-Krüger *Gregor Kemper*

Kandidatinnen und Kandidaten für die Fachgruppenleitung

Dipl. Math. Xenia Bogomolec, selbstständige IT-Fachkraft in verschiedenen industriellen Bereichen, hauptsächlich IT-Security. Diplom in algorithmischer kommutativer Algebra. Breites Netzwerk zwischen Industrie und Wissenschaft. Seit 2018 als Fachexpertin Industrie in der Fachgruppenleitung.
https://quant-x-sec.com

Prof. Dr. Michael Cuntz, Professor für Diskrete Mathematik an der Leibniz Universität Hannover. Arbeitsgebiete: Computeralgebra, Arrangements von Hyperebenen, Spiegelungsgruppen, Quantengruppen und Tensorkategorien. Entwickler von zahlreichen Computerbeweisen und Experimenten für die Forschung, insbesondere mit Magma, GAP und Sage.
https://www.iazd.uni-hannover.de/cuntz.html

Prof. Dr. Claus Fieker, Professor für konstruktive Zahlentheorie und Computeralgebra an der TU Kaiserslautern. Arbeitsgebiete: Computeralgebra, konstruktive Zahlentheorie, Klassenkörpertheorie, Darstellungstheorie und Galois Theorie. Mitentwickler von Magma (elf Jahre in Sydney), Kant/Kash und Singular.
http://www.mathematik.uni-kl.de/~fieker

Prof. Dr. Anne Frühbis-Krüger, Professorin für Computeralgebra und Arithmetische/Algebraische Geometrie an der Universität Oldenburg. Arbeitsgebiete: Algorithmische Singularitätentheorie, Algorithmische Algebraische und Arithmetische Geometrie, seit 1996 Mitarbeit an der Entwicklung des Computeralgebrasystems Singular und inzwischen auch OSCAR.
https://uol.de/anne-fruehbis-krueger

Dr. Thomas Hahn, Wissenschaftler am Max-Planck-Institut für Physik, München. In der Fachgruppenleitung seit 2002 als Fachexperte Physik, Autor der Computeralgebra-Softwarepakete FeynArts und FormCalc für Rechnungen im Bereich der Teilchenphysik.
https://wwwth.mpp.mpg.de/members/hahn

Prof. Dr. Tommy Hofmann, Juniorprofessor für Algorithmische Algebra am Department Mathematik der Universität Siegen. Arbeitsgebiete: Computeralgebra, algorithmische Zahlentheorie und arithmetische Gruppen. Mitentwickler der Computeralgebrasysteme OSCAR und Hecke.
http://algebra.mathematik.uni-siegen.de/hofmann

Prof. Dr. Max Horn, Professor für algorithmische Algebra an der TU Kaiserslautern. Arbeitsgebiete: Computeralgebra (insbesondere Gruppentheorie) algebraische Lie-Theorie, Kac-Moody-Gruppen und Gebäude. Mitentwickler und Teil der Projektleitung der Computeralgebrasysteme GAP und Oscar.
https://www.quendi.de/math

Prof. Dr. Gregor Kemper, Professor für algorithmische Algebra an der TU München. Arbeitsgebiete: Invariantentheorie, algorithmische kommutative Algebra, Computeralgebra. Autor von Software-Paketen für Invariantentheorie in Maple und Magma.
https://www.math.cit.tum.de/algebra/kemper

Prof. Dr. Jürgen Klüners, Professor für Computeralgebra und Zahlentheorie an der Universität Paderborn. Arbeitsgebiete: Computeralgebra, Galois- und Zahlentheorie. Mitentwickler der Computeralgebrasysteme Kant/Kash und Magma sowie einer Datenbank für Zahlkörper.
https://math.uni-paderborn.de/ag/ca/

Prof. Dr. Martin Kreuzer, Universitätsprofessor, Lehrstuhl für Symbolic Computation, Fakultät für Informatik und Mathematik, Universität Passau. Arbeitsgebiete: Computeralgebra, insbesondere Gröbnerbasen und Randbasen, industrielle Anwendungen der Computeralgebra, algebraische Kryptographie, algebraische Geometrie. Leiter des Entwicklerteams des Computeralgebrapakets ApCoCoA.
http://staff.fim.uni-passau.de/~kreuzer

Dr. Fabian Reimers, wissenschaftlicher Mitarbeiter am Lehrstuhl für algorithmische Algebra, Technische Universität München. Arbeitsgebiet: Invariantentheorie. Seit 2017 als Redakteur des Computeralgebra-Rundbriefs in der Fachgruppenleitung.
https://www.math.cit.tum.de/algebra/reimers

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

Fachgruppe Computeralgebra

Tagung der Fachgruppe Computeralgebra

Hannover

31.05.–02.06.2023

https://www.fachgruppe-computeralgebra.de/hannover-2023

Schon zum 10. Mal richtete 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)
- **Mima Stanojkovski*** (RWTH Aachen)
- **Ulrich Thiel** (TU Kaiserslautern)

Welfenschloss Hannover, Foto: M.Cuntz.

* to be confirmed

Website: `http://www.fachgruppe-computeralgebra.de/hannover-2023`

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. April 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)

Symmetric embeddings of the icosahedron

Daniel Robertz (RWTH Aachen)

daniel.robertz@rwth-aachen.de

Introduction

Among the well-known Platonic solids is the icosahedron whose 20 faces are equilateral triangles of the same congruence type. At each of the twelve vertices five triangles meet and form what we call an umbrella. This convex polyhedron is only one geometric incarnation of an abstract icosahedron that admits variations of the Platonic solid as follows. Prescribing the incidence structure and the Euclidean geometry of its flat faces leaves – as it turns out – infinitely many possibilities of realizing the icosahedron as a surface made up of equilateral triangles in three-dimensional Euclidean space $\mathbb{R}^3$. Most of these polyhedra are self-intersecting.

Some non-convex embeddings can be obtained from the convex one as follows. For any vertex V of the regular icosahedron the five neighbouring vertices lie in a plane. Orthogonal reflection of V through the plane yields a "dented icosahedron". Figure 5 shows an icosahedron that can be obtained from the regular one by denting at two vertices at combinatorial distance 3.

In recent work [7] by Brakhage, Niemeyer, Plesken, Strzelczyk and the author of this note, the embeddings of the icosahedron admitting non-trivial symmetry have been classified. (The restriction to non-trivial symmetry was decided upon after it had turned out that the cases with trivial symmetry would be too numerous to deal with in the same approach.) Here we summarize our results and give a few details on how the classification was obtained. For more details we refer to the paper [7] as well as its accompanying web page.

There is a vast literature on polyhedra, cf., e.g., [2], [10], [15], [16], to cite just a few of the most prominent ones. The problem addressed here can also be understood as the problem of realizing a given graph in $\mathbb{R}^3$ with prescribed edge lengths and is linked with questions about rigidity, cf., e.g., [14]. Even closer related to the topic of this note, e.g., the existence of icosahedra all of whose faces are congruent to one scalene triangle was shown in [12].

Simplicial surfaces

The context in which said classification has been achieved is a more general study of surfaces that are composed of triangles, which we call *simplicial surfaces*. Motivated by a question posed by an architect at RWTH Aachen University what would be the types of surfaces one could build from a collection of congruent triangles, a systematic mathematical investigation has been launched. A first monograph on the subject is in preparation [13].

Simplicial surfaces are now studied from different points of view:

- considered as combinatorial objects, simplicial surfaces can be generated and classified by group-theoretical methods;
- considered as (abstract) geometrical objects, simplicial surfaces can be studied as manifolds with singularities;
- embeddings of simplicial surfaces in Euclidean space form a rich source of concrete mathematical objects of interest in algebraic geometry and related fields as well as for applications in the sciences and engineering.

Embeddings of an infinite family of simplicial surfaces with dihedral symmetry, called double $2n$-gons, have been studied in [8].

Quite a few results on combinatorial aspects have already been obtained, which are beyond the scope of this note. As a byproduct a package `SimplicialSurfaces` [1] for GAP [11] has been developed by Markus Baumeister, Alice C. Niemeyer and further collaborators. For instance, it allows to generate all (combinatorial) simplicial surfaces for a given number of faces (up to isomorphism), possibly taking further given conditions on the incidence structure into account. It also provides routines for classifying edge colorings of these surfaces, which is relevant for embeddings realizing a single congruence type of triangle.

The topic of this note is an intriguing case study concerning the third aspect of our research program listed above. It turned out that, in the order of increasing complexity of simplicial surfaces, the icosahedron is a first challenging case to analyze.

Figure 1: *An icosahedron with symmetry group* ${C_2}^2$ *from the first row of Table 2*

The icosahedron

We may use the automorphism group of the (combinatorial) icosahedron to give a succinct definition of its incidence structure, involving the vertices numbered from 1 to 12, edges represented by unordered pairs of vertices, and faces represented as 3-element sets of vertices.

We choose the permutations of twelve points

$$\begin{aligned} a &:= (1,2)(3,4)(5,7)(6,8)(9,11)(10,12),\\ b &:= (1,10)(3,9)(2,12)(4,11)(5,6)(7,8),\\ c &:= (1,7)(2,3)(4,11)(5,12)(6,8)(9,10),\\ d &:= (1,12)(3,9)(2,10)(4,11)(5,7)(6,8) \end{aligned}$$

as a generating set for the combinatorial automorphism group $A \cong C_2 \times A_5$ of the icosahedron. The center of A is generated by the involution d, whose effect on the (embedded) regular icosahedron with barycenter $(0,0,0)$ would be the point reflection. According to the above choice, the combinatorial description of the icosahedron is endowed with a labeling: the orbit of 1 under A consists of the vertices 1, 2, ..., 12; the orbit of $\{1,2\}$ is the set of 30 edges; and the orbit of $\{1,2,3\}$ is formed by the 20 faces. Thus d interchanges combinatorially opposite vertices.

Symmetric embeddings

Our task is to classify the embeddings of the icosahedron in $\mathbb{R}^3$ realizing the edges as line segments of length 1, with 12 distinct vertices, and admitting a non-trivial symmetry group. Such an embedding is given by a map $p : \{1,\ldots,12\} \to \mathbb{R}^3$ such that the Euclidean distance between $p(i)$ and $p(j)$ is equal to 1 for all edges $\{i,j\}$ of the icosahedron, and we require that such a map be injective. We usually arrange $p(1),\ldots,p(12)$ as columns of a *coordinate matrix* $M \in \mathbb{R}^{3\times 12}$.

We are thus interested, in principle, in finding the real solutions of a system of 30 multivariate quadratic equations in $12 \times 3 = 36$ unknowns (which, however, do not incorporate all the above conditions, e.g., vertex-faithfulness, symmetry). Since this system of polynomial equations is very difficult to solve, our approach had to be refined.

Figure 2: *Convex icosahedron with symmetry group* $C_2 \times A_5$ *from the second row of Table 2*

Of course, the group $E(3) \cong \mathbb{R}^3 \rtimes O(3,\mathbb{R})$ of Euclidean isometries acts on the set of embeddings, and we factor out this action as follows. Translations allow to move the barycenter of the (equally weighted) points $p(1)$, ..., $p(12)$, i.e. the columns of M, to the origin. Now, by considering the *Gram matrix* $G := M^{tr}M$ we also factor out the action of the orthogonal group $O(3,\mathbb{R})$. Note that a Gram matrix G is a real symmetric 12×12 matrix that is positive semidefinite of rank at most 3.

As a result for our classification task, the isometry classes of (embedded) icosahedra are given by the equivalence classes of Gram matrices that are defined with respect to conjugacy by permutation matrices.

The group A acts on the set of Gram matrices G by simultaneous row and column permutations, and the stabilizer of G in A is the *symmetry group* of the (embedded) icosahedron.

Theorem 1 The subgroups U of A with more than one element that arise as symmetry group of an icosahedron fall into 10 conjugacy classes of subgroups of A. For each of these conjugacy classes the number of (isometry classes of) icosahedra with symmetry group in that class is listed in Table 1.

Automorphism group $U \leq A \cong C_2 \times A_5$	Number of icosahedra
$C_2 \times A_5$	2
$C_2 \times D_{10}$	4
$C_2 \times D_6$	2
D_{10} $(\not\leq A_5)$	3
D_6 $(\not\leq A_5)$	2
${C_2}^2$ $(\ni d)$	1
${C_2}^2$ $(\not\ni d, \not\leq A_5)$	5
${C_2}^2$ $(\leq A_5)$	1
C_2 $(\leq A_5)$	5
C_2 $(\not\ni d, \not\leq A_5)$	10
C_2 $(= \langle d \rangle)$	∞

Table 1: *Equivalence classes of symmetric embeddings of the icosahedron*

[1] There is a discrete subset of points on the curve corresponding to degenerate icosahedra, in the sense that the set of vertices is mapped to fewer than 12 points.

The last row of Table 1 corresponds to an algebraic curve, almost all[1] of whose (closed) points correspond to icosahedra with (at least) the specified symmetry. Hence, there is a continuous family of such icosahedra, which can be understood as a *flexible icosahedron*, i.e., an icosahedron admitting motions preserving all edge lengths, but which are not rigid motions.

By the following famous theorem, there is only one convex icosahedron and it is necessarily rigid.

Theorem 2 (Cauchy's Rigidity Theorem, 1813, [9]) If two closed *convex* polyhedra are combinatorially equivalent and corresponding pairs of faces of these two polyhedra are congruent, then the two polyhedra are congruent (i.e., there exists a rigid motion transforming one to the other).

Figures 1–7 show visualizations of some of the icosahedra with non-trivial symmetry listed in Table 2, the details of which we explain in the next section.

For more visualizations and, in particular, animations of the flexible icosahedron found in the last row of Table 1 we refer to the web page accompanying our paper [7].

Figure 3: *Non-convex icosahedron with symmetry group $C_2 \times A_5$ from the third row of Table 2*

Classification

Since we focused on icosahedra with non-trivial symmetry group, we went through a list of representatives of conjugacy classes of minimal subgroups U of A, namely

$$\langle abc \rangle \cong C_3, \qquad \langle ac \rangle \cong C_5, \qquad \langle a \rangle \cong C_2,$$
$$\langle d \rangle \cong C_2, \qquad \langle ad \rangle \cong C_2,$$

and we determined the embeddings of the icosahedron whose symmetry groups contain U. We arrived at computationally tractable tasks thanks to the following consequence of the fact that any two coordinate matrices associated with the same Gram matrix G differ multiplicatively by an orthogonal matrix.

Lemma Let G be a Gram matrix of an icosahedron with symmetry group U. Then there exist a faithful orthogonal representation $\delta : U \to O(3, \mathbb{R})$ and a coordinate matrix $M \in \mathbb{R}^{3\times 12}$ such that

$$\delta(g)\, M = M\, \pi(g) \quad \text{for all } g \in U \qquad (1)$$

and $G = M^{tr}\, M$, where $\pi : A \to \mathrm{GL}(12, \mathbb{R})$ is the representation of A by permutation matrices.

We were able to rule out the minimal subgroups of odd order and therefore had to investigate the three representatives U isomorphic to C_2, whose generators can be mapped to one of the following three matrices by δ:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

Hence, we had to consider nine pairs of U and δ. In each case the imposed symmetry condition allows via (1) to drastically reduce the number of unknowns in the system of polynomial equations for the entries of M. We computed a primary decomposition of the corresponding ideal I in the appropriate polynomial ring R. In all but one case the Krull dimension of I turned out to be zero, in just one case the Krull dimension of I was one.

For a maximal ideal m arising in the minimal primary decomposition of a zero-dimensional ideal $I \trianglelefteq R$ as above we refer to the image modulo m of $G = M^{tr}\, M$ with reduced number of unknowns as *formal Gram matrix*. Several checks had to be performed to rule out the following scenarios:

- the field R/m may have no real embedding;
- the resulting formal Gram matrix may not be positive semi-definite of rank at most 3;
- U may be a proper subgroup of the symmetry group of the resulting formal Gram matrix.

Table 2 records the field degree $d_G := [R/m : \mathbb{Q}]$, the number $r_{1,G}$ of real embeddings of R/m, the number r_G of those real embeddings of R/m defining a positive semidefinite formal Gram matrix, and the number $r_{f,G}$ the number of those that moreover yield 12 pairwise distinct points $p(1), \ldots, p(12)$.

The rows in Table 2 are arranged in blocks following the order in which we investigated the nine cases. Three of the nine cases did not contribute to that list, and one case led to the ideal of Krull dimension 1 to be discussed further in the next section. Since the symmetry group for the resulting Gram matrix can strictly contain the chosen minimal subgroup U, duplicate formal Gram matrices arising from different cases had to be eliminated.

For computing in R/m we chose a primitive element, preferably the trace of the Gram matrix, if possible. However, it is not always the case that the trace generates R/m. Table 2 lists the minimal polynomial of the trace of G over Q in its last column.

Finally, in the second column of Table 2 in certain cases we indicate the trace value of the single generator, or the trace values of the two generators above a horizontal line and the trace value of their product below the horizontal line, where $+$, $-$, $\bullet$ stands for 1, -1, -3, respectively.

$S:=\mathrm{Stab}_A$	$\mathrm{Syl}_2(S)$		d_G	$r_{1,G}$	r_G	$r_{f,G}$	Trace relation
${C_2}^2$	$\langle a,d\rangle$	$\overset{\bullet-}{+}$	8	4	2	1	$\lambda^4-\frac{76}{3}\lambda^3+238\lambda^2-\frac{4964}{5}\lambda+\frac{23767}{15}$
$C_2\times A_5$	$\langle a,b,d\rangle$		2	2	2	2	$\lambda^2-15\lambda+45$
$C_2\times D_{10}$	$\langle a,d\rangle$	$\overset{-\bullet}{+}$	2	2	2	2	$\lambda^2-15\lambda+\frac{269}{5}$
${C_2}^2$	$\langle a,bd\rangle$	$\overset{-+}{+}$	2	2	2	2	$\lambda^2-\frac{71}{5}\lambda+\frac{10561}{225}$
$C_2\times D_{10}$	$\langle a,d\rangle$	$\overset{-+}{+}$	4	2	2	2	$\lambda^4-18\lambda^3+\frac{583}{5}\lambda^2-\frac{1658}{5}\lambda+\frac{9101}{25}$
$C_2\times D_6$	$\langle a,d\rangle$	$\overset{-+}{+}$	4	4	2	2	$\lambda^4-26\lambda^3+243\lambda^2-970\lambda+1397$
${C_2}^2$	$\langle a,bd\rangle$	$\overset{-+}{+}$	24	10	6	3	$\lambda^{12}-\frac{5179\cdot 2^2}{3^2\cdot 5^2}\lambda^{11}\pm\cdots$
${C_2}^2$	$\langle a,b\rangle$	$\overset{--}{-}$	30	18	6	1	$\lambda^5-\frac{117}{2}\lambda^4\pm\cdots$
C_2	$\langle a\rangle$	$-$	172	48	20	5	$\lambda^{43}-\frac{73\cdot 7\cdot 11\cdot 461687}{2^2\cdot 3^3\cdot 5^2\cdot 29\cdot 79}\lambda^{42}\pm\cdots$
D_{10}	$\langle ad\rangle$	$+$	2	2	2	2	$\lambda^2-\frac{44}{3}\lambda+\frac{2131}{45}$
D_6	$\langle ad\rangle$	$+$	2	2	2	2	$\lambda^2-\frac{68}{5}\lambda+\frac{1111}{25}$
C_2	$\langle ad\rangle$	$+$	36	12	8	4	$\lambda^{18}-\frac{1106}{9}\lambda^{17}\pm\cdots$
C_2	$\langle ad\rangle$	$+$	168	40	24	6	$\lambda^{42}-\frac{2\cdot 719\cdot 1223}{3^3\cdot 5\cdot 43}\lambda^{41}\pm\cdots$
D_{10}	$\langle ad\rangle$	$+$	4	2	2	1	$\lambda^2-\frac{26}{3}\lambda+\frac{149}{9}$

Table 2: *List of all formal Gram matrices with non-trivial symmetry*

Figure 4: *An icosahedron with symmetry group $C_2\times D_{10}$ from the third row of Table 2*

A curve of icosahedra

If $U=\langle d\rangle$ and $\delta(d)$ is equal to the first orthogonal matrix listed earlier, the ideal I is of Krull dimension 1. In this case computational resources did not allow to compute a primary decomposition of I. We managed to prove the existence of a flexible icosahedron in this case by recognizing the corresponding curve as integral curve of a polynomial vector field. Knowing the coordinates of some point on the curve in terms of algebraic numbers from another case of the classification as well as the tangent vectors to the curve, we can solve (at least approximately) the corresponding initial value problem. Moreover, standard theorems on ordinary differential equations yield the existence of $\varepsilon>0$ and a non-constant real analytic map $\Phi:[0,\varepsilon)\to\mathbb{R}^{3\times 12}$ such that $\Phi(t)$ is the coordinate matrix of a $\langle d\rangle$-invariant icosahedron for all by finitely many $t\in[0,\varepsilon)$. Since the map $t\mapsto\Phi(t)^{tr}\,\Phi(t)$ is not constant and only finitely many isometry types of icosahedra correspond to a Gram matrix $\Phi(t)^{tr}\,\Phi(t)$, it follows that there are infinitely many pairwise inequivalent $\langle d\rangle$-invariant icosahedra.

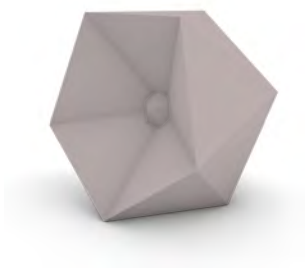


Figure 5: *An icosahedron with symmetry group $C_2\times D_{10}$ from the third row of Table 2*

Figure 6: *An icosahedron with symmetry group $C_2\times D_6$ from the sixth row of Table 2*

Computational aspects

In a first approach to solving the system of multivariate quadratic equations expressing that the 30 edges of the embedded icosahedron have length 1 we applied the software Bertini [3] implementing homotopy continuation of

solutions to systems of polynomial equations. It gave a very good impression on how many solutions to expect for our problem, but we were not able to convert the results obtained into exact solutions nor to make progress in the classification task with that information.

For the new successful attempt with exact arithmetic we used Magma [6] for computing primary decompositions of polynomial ideals and the Maple package Involutive [4] and GINV [5] for further processing of the prime ideals.

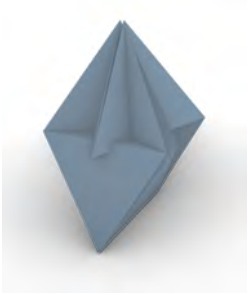


Figure 7: *An icosahedron with symmetry group* $C_2{}^2$ *from the seventh row of Table 2*

Acknowledgements

The author would like to express his gratitude to Tom Görtzen for the preparation of the graphics using Rhino®.

References

[1] R. Akpanya, M. Baumeister, T. Görtzen, A.C. Niemeyer, M. Weiß. The GAP package SimplicialSurfaces, 2022. `https://github.com/markusbaumeister/simplicial-surfaces`.

[2] A.D. Alexandrov. *Convex polyhedra*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2005. Translated from the 1950 Russian edition by N. S. Dairbekov, S. S. Kutateladze and A. B. Sossinsky, with comments and bibliography by V. A. Zalgaller and appendices by L. A. Shor and Yu. A. Volkov.

[3] D.J. Bates, J.D. Hauenstein, A.J. Sommese, C.W. Wampler. Bertini: Software for numerical algebraic geometry. Available at `bertini.nd.edu` with permanent doi: `dx.doi.org/10.7274/R0H41PB5`.

[4] Y.A. Blinkov, C.F. Cid, V.P. Gerdt, W. Plesken, D. Robertz. The MAPLE Package Janet: I. Polynomial Systems. In V. Ganzha, E. Mayr, E. Vorozhtsov, editors, *Proceedings of the 6th International Workshop on Computer Algebra in Scientific Computing, Sep. 20–26, 2003, Passau (Germany)*, pages 31–40, 2003. Available from `http://algebra.data.rwth-aachen.de/software/Janet`.

[5] Y.A. Blinkov, V.P. Gerdt. The specialized computer algebra system GINV. *Programmirovanie*, 34(2):67–80, 2008. `http://invo.jinr.ru`.

[6] W. Bosma, J. Cannon, C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3–4):235–265, 1997.

[7] K.-H. Brakhage, A.C. Niemeyer, W. Plesken, D. Robertz, A. Strzelczyk. The icosahedra of edge length 1. *J. Algebra*, 545:4–26, 2020. Accompanying web page: `http://algebra.data.rwth-aachen.de/Icosahedra/visualplusdata.html`.

[8] K.-H. Brakhage, A.C. Niemeyer, W. Plesken, A. Strzelczyk. Simplicial surfaces controlled by one triangle. *J. Geom. Graph.*, 21(2):141–152, 2017.

[9] A.-L. Cauchy. Sur les polygones et les polyèdres: second mémoire. *Journal de l'École polytechnique, XVIe cahier*, t. IX:87, 1813.

[10] H.S.M. Coxeter. *Regular polytopes*. Dover Publications, Inc., New York, third edition, 1973.

[11] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.12.0*, 2022. `https://www.gap-system.org`.

[12] E.N. Miller. Icosahedra constructed from congruent triangles. *Discrete Comput. Geom.*, 24(2–3):437–451, 2000. The Branko Grünbaum birthday issue.

[13] A.C. Niemeyer, W. Plesken, D. Robertz. *Simplicial Surfaces of Congruent Triangles*. Monograph in preparation.

[14] B. Schulze. Symmetry as a sufficient condition for a finite flex. *SIAM J. Discrete Math.*, 24(4):1291–1312, 2010.

[15] E. Steinitz and H. Rademacher. *Vorlesungen über die Theorie der Polyeder unter Einschluss der Elemente der Topologie*. Grundlehren der Mathematischen Wissenschaften, No. 41. Springer-Verlag, Berlin-New York, 1976. Reprint of the 1934 edition.

[16] G.M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995.

Algebra in probabilistic reasoning

Tobias Boege (Aalto University, Finland)

post@taboege.de

Probabilistic reasoning meets synthetic geometry

Reasoning about relevance

Probabilistic reasoning is an approach, based on probability theory, to tasks of *decision making under uncertainty*. Random variables are used to model our uncertainty about the factors that define what a good decision is. Decision making (certain or uncertain) is a vast field with different paradigms. For example, one may seek to make the globally least bad decision given what is known certainly about the state of the random variables. In probably approximately correct learning, on the other hand, one tries to keep the errors small, most of the time.

A part of probabilistic reasoning is concerned with *conditional independence*. This is a ternary relation among jointly distributed random variables which gives information of the following sort: "Given that factor C is observed, factor A does not influence factor B". This relation is denoted by $[A \perp\!\!\!\perp B \mid C]$. We can also chose to condition on multiple observed variables or none at all and recover the familiar notion of stochastic independence $[A \perp\!\!\!\perp B]$. Conditional independence (in the following abbreviated to CI) is an extension of stochastic independence which can accommodate a priori knowledge that we may have about the outcome of a subsystem C of random variables, such as when C is *controlled* in a random experiment. CI reveals essential combinatorial information that can guide decision making when only incomplete data about the state of a system is available.

As a first exercise in conditional independence, note what this relation specializes to when $A = B$. Suppose the outcome C is known and $[A \perp\!\!\!\perp A \mid C]$ holds. Then the outcome of A has no bearing on itself after we observe C. This is absurd, *unless* C reveals everything there is to know about A, i.e., A takes only a single value that depends on the value C takes. This is known as a *functional dependence* (FD) and it implies the existence of a function f such that $A = f(C)$ as random variables.

CI and FD provide basic qualitative information about dependencies among the observations made in, say, a random experiment in the sciences. But they play a role in other disciplines that deal with the representation or processing of *information*. For example, a database in relational algebra may be seen as a (large) sample from an unknown discrete probability distribution. The designer of a database will usually anticipate CI and FD relations in the data (e.g., your zip code functionally determines your city). The purpose of various *normal forms* for relational databases is to eliminate undesirable dependencies because they increase the risk of inconsistencies in the data after updates. Instead, the database must be factored into multiple "tables" according to the normal form. In spirit, what these normal forms demand is similar to the factorization of a rank-1 matrix into an outer product

$$M = ab^{\mathsf{T}}.$$

If M is the probability matrix of the joint distribution of discrete random variables A and B, then a and b are uniquely determined up to a scalar as the row and column sums of M and they correspond to the marginal distributions of A and B. (With more than two random variables, we factor a rank-1 tensor.) This representation of M as ab^{T} makes maintaining the independence of A and B in the joint distribution automatic under updates to the marginal distributions a and b, and it saves space!

Synthetic statistics

The reasoning task attached to CI is that of *conditional independence inference*: given a boolean formula φ whose variables are CI statements and a family of probability distributions, decide if φ is true for every distribution in the family. By writing boolean formulas in conjunctive normal form, one can restrict this investigation to disjunctive clauses written in implication form, such as:

$$[A \perp\!\!\!\perp C \mid B] \wedge [A \perp\!\!\!\perp B] \Rightarrow [A \perp\!\!\!\perp C].$$

This formula is one half of the *semigraphoid property* and it holds for all random vectors [16, Appendix A.7].

Its meaning should be intuitively clear: suppose that knowing B makes A and C independent, but also that B has no influence on A; then A and C must be independent even without knowledge of B. One should be cautious, however, of leaping to "intuitively clear" conclusions in CI inference because the laws of probability theory sometimes seem to defy intuition. The implications

$$[A \perp\!\!\!\perp B] \Rightarrow [A \perp\!\!\!\perp B \mid C] \qquad (\mathscr{X})$$
$$[A \perp\!\!\!\perp B \mid C] \Rightarrow [A \perp\!\!\!\perp B] \qquad (\mathscr{Z})$$
$$[A \perp\!\!\!\perp B] \wedge [A \perp\!\!\!\perp C] \Rightarrow [A \perp\!\!\!\perp (B, C)] \qquad (\mathscr{E})$$

are all wrong. The reader is invited to construct random experiments which falsify them.

The semigraphoid property allows us to deduce with absolute confidence from some CI assumptions other CI consequences, no matter what the underlying probability distribution is. A valid implication is also called a *CI axiom* or *inference rule*. More restrictions on the distributions under consideration make more valid inference rules available for reasoning. In this article, we will consider multivariate normal distributions. This class has many favorable properties: (1) relatively few parameters are needed to specify a distribution, (2) they include classes of popular graphical models and (3) conditional independence has an *algebraic* reformulation. Whenever possible we wish to use algebra in reasoning and benefit from the exactness of symbolic methods.

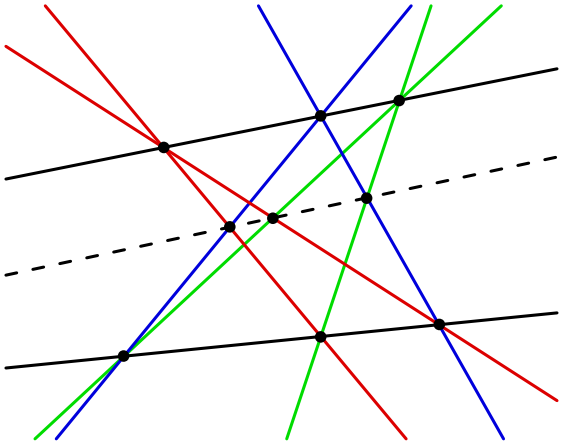


Figure 1: *Pappus's theorem in the projective plane.*

In this article, I want to explain a different point of view on probabilistic reasoning, in particular CI inference. It is motivated by similarities to synthetic geometry [6] which describes geometric objects in relations of "special position" to each other. Figure 1 illustrates Pappus's theorem in the projective plane over a field. This is an inference rule in synthetic geometry stating that: if all points on the solid lines are collinear, then the points on the dashed line must also be collinear. Instead of geometric objects, in probabilistic reasoning we describe random variables and their "special position" in relation to each other. Special position in this case is conditional independence and we wish to obtain rules of reasoning such as Pappus's theorem in this setting — which we may call *synthetic statistics*.

The geometry of CI inference

Algebraic statistics of Gaussian random vectors

We suppose a finite *ground set* N of size n indexing the entries of a random vector $X = (X_i : i \in N)$. Instead of referring to random variables, subsequently we refer to their indices. It is customary not to distinguish between an element $i \in N$ and a singleton subset $\{ i \} \subseteq N$: both are usually denoted by i. Moreover, the symbol for set union $I \cup K$ for subsets of N is usually omitted. Hence, an expression such as iK means the subset $\{ i \} \cup K \subseteq N$. Denote by $\mathrm{Sym}_N(\mathbb{K})$ the affine space of $N \times N$ symmetric matrices over a field $\mathbb{K} \subseteq \mathbb{R}$ and by $\mathrm{PD}_N(\mathbb{K})$ the semialgebraic subset of positive definite matrices. Recall that this is a full-dimensional, open convex cone and that its boundary is the hypersurface of singular positive semidefinite matrices.

A regular multivariate normal ("Gaussian") distribution is determined by its mean $\mu \in \mathbb{R}^N$ and its covariance matrix $\Sigma \in \mathrm{PD}_N(\mathbb{R})$. Regularity refers to the covariance matrix Σ which in general only needs to be positive semidefinite. While an algebraic theory of conditional independence can be developed even in the degenerate case, this is more involved and we stick to the regular Gaussians here.

The density of a Gaussian random vector X with respect to the standard Lebesgue measure on $\mathbb{R}^N$ is a proper transcendental function depending on μ and Σ:

$$\frac{1}{\sqrt{(2\pi)^n \det \Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu) \right).$$

A surprising but basic fact of algebraic statistics is that the conditional independence relation among the components of X depends only on certain polynomial expressions in the covariance matrix.

Definition 3 A *principal minor* of Σ is a subdeterminant $\Sigma[K] := \det \Sigma_{K,K}$ for some $K \subseteq N$. An *almost-principal minor* is a subdeterminant of the form $\Sigma[ij|K] := \det \Sigma_{iK,jK}$ for $ijK \subseteq N$ and $i, j \notin K$ distinct, where iK indexes rows and jK columns.

Sign convention It is advantageous for the general CI theory to view the set N and hence the rows and columns of our matrices as unordered. To still get a well-defined sign for its determinant, it only matters which row and column labels $r, c \in N$ are paired together in the k^{th} position from the top-left corner of the matrix. For principal minors $\Sigma[K]$ and almost-principal minors $\Sigma[ij|K]$ we establish the following convention: in the principal submatrix with respect to $K \subseteq N$, pair each $k \in K$ with itself, and in the almost-principal submatrix additionally pair row i with column j.

Lemma 4 A symmetric matrix $\Sigma \in \mathrm{Sym}_N(\mathbb{R})$ is positive definite if and only if $\Sigma[K] > 0$ for all $K \subseteq N$. If Σ is the positive definite covariance matrix of the Gaussian random vector X, then the conditional independence $[X_i \perp\!\!\!\perp X_j \mid X_K]$ holds if and only if $\Sigma[ij|K] = 0$.

The crucial ingredient to prove this is in [18, Proposition 4.1.9]. In particular, the CI relation does not depend on the mean μ and we may identify Gaussian distributions with their positive definite covariance matrices.

Definition 5 A *Gaussian CI model* is a subset of $\mathrm{PD}_N(\mathbb{R})$ which is given by vanishing and non-vanishing constraints on almost-principal minors (referred to as the *independence* and the *dependence assumptions* of the model, respectively).

Remark 6 It should be noted that in geometry it is (linear) *dependence* of vectors which corresponds to a Zariski-closed condition, whereas in statistics it is the (conditional) *independence* which is closed.

Gaussian conditional independence models are subsets of the PD cone which are cut out by **very special** classes of polynomial constraints. They are all determinantal and, up to the symmetric group on N acting on the coordinates of $\mathrm{Sym}_N(\mathbb{R})$, there is precisely one principal and one almost-principal minor of each degree.

Conditional independence inference

Consider the general implication formula φ:

$$\bigwedge_{p=1}^{s} [i_p \perp\!\!\!\perp j_p \mid K_p] \;\Rightarrow\; \bigvee_{q=1}^{t} [x_q \perp\!\!\!\perp y_q \mid Z_q]$$

involving CI statements over a fixed ground set N. This formula is a valid inference rule for Gaussians if and only if every $\Sigma \in \mathrm{PD}_N(\mathbb{R})$ which satisfies $\Sigma[i_p j_p | K_p] = 0$ for *all* $p \in [s]$ also satisfies $\Sigma[x_q y_q | Z_q] = 0$ for *at least one* $q \in [t]$. We associate to φ a CI model $\mathcal{M}(\varphi)$ which is defined by the independence assumptions $[i_p \perp\!\!\!\perp j_p \mid K_p]$, $p \in [s]$, and the dependence assumptions $\neg[x_q \perp\!\!\!\perp y_q \mid Z_q]$, $q \in [t]$. This is the set of *counterexamples* to φ; the implication is valid if and only if $\mathcal{M}(\varphi) = \emptyset$. Conversely, every CI model is the set of counterexamples to a suitable CI implication formula.

Hence, the CI inference problem is equivalent to the problem of checking if a system of independence and dependence assumptions is consistent, which in turn reduces to checking the feasibility of a semialgebraic set which is defined by integer polynomials — the CI assumptions as well as positive definiteness.

Example 7 The *weak transitivity* property of Gaussians over $N = ijk$ states that

$$[i \perp\!\!\!\perp j] \wedge [i \perp\!\!\!\perp j \mid k] \Rightarrow [i \perp\!\!\!\perp k] \vee [j \perp\!\!\!\perp k].$$

To prove this rule algebraically, we first determine the ideal generated by the assumptions:

$$\begin{aligned} \Sigma[ij|] &= \sigma_{ij}, \\ \Sigma[ij|k] &= \sigma_{ij}\sigma_{kk} - \sigma_{ik}\sigma_{jk}. \end{aligned}$$

These vanishing conditions give rise to a hyperplane and a quadratic hypersurface in the space of 3×3 correlation matrices, pictured in Figure 2. The equation $\sigma_{ik}\sigma_{jk} = 0$ holds on the intersection, which is the union of two line segments: one for each of the possible conclusions $[i \perp\!\!\!\perp k] \vee [j \perp\!\!\!\perp k]$ of weak transitivity reasoning. The line segments intersect in the identity matrix which satisfies both conclusions.

It was observed by Matúš [10] that fundamental inference rules for Gaussians, including a generalization of Example 7 to arbitrary conditioning sets, follow from a single determinantal identity:

Matúš's identity Let R be a commutative ring with unity and $\Sigma \in \mathrm{Sym}_N(R)$. The following identity holds for all $ijkL \subseteq N$:

$$\begin{aligned} \Sigma[kL] \cdot \Sigma[ij|L] = \\ \Sigma[L] \cdot \Sigma[ij|kL] + \Sigma[ik|L] \cdot \Sigma[jk|L]. \end{aligned} \tag{2}$$

Remark 8 Matúš's formulation of this result in [10] adds a sign to one of the terms of the identity, depending on the relative ordering of i, j and k. This sign is fixed to $+1$ by our Sign convention.

Remark 9 Moreover, Matúš proves this identity over the complex numbers, but the extension to any commutative ring is standard. It suffices even to prove the identity only for real positive-definite matrices. Since then the identity is known to hold on a full-dimensional (and therefore dense) subset of the irreducible affine space $\mathrm{Sym}_N(\mathbb{C})$, it holds for all symmetric matrices over $\mathbb{C}$. But then evaluating the linear combination of determinants in (2) must result in the zero polynomial in $\mathbb{Z}[\Sigma]$ and thus in every commutative ring with unity.

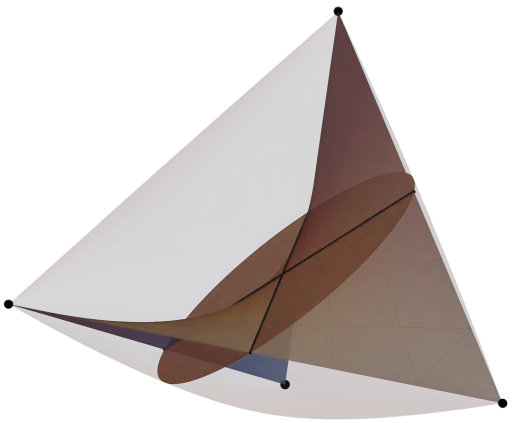


Figure 2: *Geometric proof of weak transitivity.*

The general weak transitivity property

$$[i \perp\!\!\!\perp j \mid L] \wedge [i \perp\!\!\!\perp j \mid kL] \Rightarrow [i \perp\!\!\!\perp k \mid L] \vee [j \perp\!\!\!\perp k \mid L]$$

is a simple consequence of this relation and the algebraic definition of conditional independence in Lemma 4.

This motivates the investigation of general determinantal identities which can be formulated in terms of principal and almost-principal minors only. More precisely, we consider the polynomial ring $\mathcal{R}_N$ whose variables are formal brackets $[K]$ and $[ij|K]$ over the ground set N. (The empty principal minor bracket $[\emptyset]$ acts as a homogenization variable.) Let $\mathcal{J}_N$ be the homogeneous ideal given by the kernel of the evaluation map sending an element of $\mathcal{R}_N$ into the coordinate ring $\mathbb{C}[\Sigma]$ of $\mathrm{Sym}_N(\mathbb{C})$ by evaluating brackets $[K] \mapsto \Sigma[K]$ and $[ij|K] \mapsto \Sigma[ij|K]$. The generators of the quadratic part of $\mathcal{J}_N$ have been found in [5].

In some way, the Matús identity, which belongs to this generating set, acts like the three-term Grassmann–Plücker relations in geometry: it is sufficient to prove

the *gaussoid axioms*, the most basic inference rules for Gaussian CI relations. The gaussoid axioms satisfy two important completeness results which justify viewing the Matúš identity as fundamental (even though it alone does not generate the quadratic part of $\mathcal{J}_N$).

Proposition 10 (Folklore) The gaussoid axioms generate (by logical implication) all true inferences for three Gaussian random variables.

Proposition 11 ([3]) The gaussoid axioms generate (by logical implication) all true inferences, having at most two assumptions, among any number of Gaussian random variables.

The ideal $\mathcal{J}_N$ encodes the particular combinatorial flavor of principal and almost-principal minors of symmetric matrices that distinguishes the algebraic geometry of Gaussian conditional independence from that of point configurations and special position which is instead derived from the maximal minors of rectangular matrices. The following conjecture about $\mathcal{J}_N$ is still open. Its analogue in synthetic geometry is an established theorem stating that the Grassmann–Plücker relations generate the vanishing ideal of the Grassmannian.

Conjecture 12 ([5]) The ideal $\mathcal{J}_N$ is generated by its quadratic part.

Algebraic certificates and reproducibility

There is no finite axiomatization of all inference rules which are valid for n Gaussian random variables as n grows. The notion of axiomatization can be made precise by introducing *minors*. In analogy to graph and matroid theory, minors are the "natural subconfigurations" of a collection of random variables. In graph theory, one obtains minors by deleting and contracting edges. In probability theory, we take marginaland conditional distributions. Non-axiomatizability results have been achieved independently by Šimeček [15] and Sullivant [18].

Nevertheless, one may be interested in finding all valid inference rules for small numbers of random variables, so that larger systems can be partially reasoned about, one couple of variables at a time. The case of three Gaussians is covered by the gaussoid axioms. The four-variate case was solved by Lněnička and Matúš in [9] and the five-variate case is still wide open.

This classification task was posed as Challenge 1 in [5]. Determining the realizability status of the 254 826 candidate gaussoids computed there is equivalent to finding all valid inference rules on five Gaussian random variables. A proper solution to such a large-scale classification task is a FAIR database (cf. [8]) which includes not only the classification itself but also *machine-checkable proofs* of its correctness. The existence of these proofs is guaranteed by theorems in real algebra. I want to close this exposition by discussing what they are and the obstacles currently faced when computing them in practice.

Algebraic numbers and final polynomials

Let us return to Pappus's theorem for a moment. Call the three points on the upper, the lower and the middle line in Figure 1 `a`, `b`, `c`, then `d`, `e`, `f` and then `g`, `h`, `i`, respectively, from left to right. For brevity denote below the determinant of the 3×3 matrix those columns are the homogeneous coordinates of the points labeled `p`, `q`, `r` by the bracket `[pqr]`. One strikingly mechanic way of proving Pappus's theorem is presented in the following snippet of `Macaulay2` code:

```
-- Homogeneous coordinates for a ... i.
R = QQ[a_1..a_3, b_1..b_3, c_1..c_3,
       d_1..d_3, e_1..e_3, f_1..f_3,
       g_1..g_3, h_1..h_3, i_1..i_3];

-- Bracket is an abbreviation for
-- the 3x3 determinant of (p q r).
br = (p,q,r) -> det matrix(
  apply({p,q,r}, x -> apply({1,2,3},
    i -> value(toString(x)|"_"|i)
  )));

-- The ideal of collinearity assumptions
-- for Pappus's theorem.
papp = ideal(
  br(a,b,c),br(d,e,f),br(a,e,g),br(a,f,h),
  br(b,d,g),br(b,f,i),br(c,d,h),br(c,e,i)
);

-- The conclusion [ghi] and non-degeneracy
-- assumptions.
G = {
  br(g,h,i),br(a,d,i),br(a,b,d),br(a,c,i),
  br(a,d,e),br(a,g,i),br(a,d,h),br(a,f,i),
  br(d,e,i),br(a,d,f),br(d,e,i),br(a,d,f),
  br(d,h,i),br(a,c,d),br(b,d,i),br(a,d,g),
  br(a,b,i),br(d,f,i),br(a,e,i),br(c,d,i)
};

fold((x,y) -> (x*y) % papp, G) --> 0
```

This computation producing a zero at the end proves the existence of polynomials $h_1, \ldots, h_8$ with rational coefficients such that

$$\begin{aligned}[\mathtt{ghi}] \cdot \prod_{i=2}^{20} g_i = \\ h_1[\mathtt{abc}] + h_2[\mathtt{def}] + h_3[\mathtt{aeg}] + h_4[\mathtt{afh}] + \\ h_5[\mathtt{bdg}] + h_6[\mathtt{bfi}] + h_7[\mathtt{cdh}] + h_8[\mathtt{cei}],\end{aligned} \tag{3}$$

where the polynomials g_i are the elements of the list `G` in the above code. The first polynomial in `G` is the desired conclusion that `g`, `h` and `i` are collinear. All other polynomials in `G` are non-zero because they correspond to genericity condition for the point and line configuration in Figure 1. This implies Pappus's theorem because it shows that `[ghi]` vanishes, under the non-degeneracy conditions, whenever the assumptions of Pappus's theorem are satisfied. The crucial list `G` for this proof is extracted from the fourth proof of Pappus's theorem in [13, Section 1.3].

The polynomial $\prod_{i=1}^{20} g_i$ and the linear combinators $h_1, \ldots, h_8$ represent a self-contained proof of Pappus's theorem. This proof is big and relatively hard to find but

verifying it is a matter of *multiplying out* both sides of (3) and comparing coefficients. This can be done in any computer algebra system off-the-shelf.

This high standard of verifiability is, fortunately, a theorem which extends far beyond Pappus:

Positivstellensatz The system $\{ f_i = 0, g_j \geq 0, h_k \neq 0 \}$ defined by finite collections of polynomials $f_i, g_j, h_k \in \mathbb{Z}[x_1, \ldots, x_n]$ has no solution if and only if $0 \in \mathcal{I} + \mathcal{P} + \mathcal{U}^2$, where $\mathcal{I}$ is the ideal generated by the f_i, $\mathcal{P}$ is the cone generated by the g_j and $\mathcal{U}$ is the multiplicative monoid generated by the h_k in $\mathbb{Z}[x_1, \ldots, x_n]$.

This version of the Positivstellensatz is proved in [2, Proposition 4.4.1]. The condition $0 \in \mathcal{I} + \mathcal{P} + \mathcal{U}^2$ implies the existence of an integer polynomial $f \in \mathcal{I} \cap (\mathcal{P} + \mathcal{U}^2)$. This *final polynomial* (cf. [6, Section 4.2]) must be simultaneously zero and positive on every point satisfying the polynomial system. It therefore serves as an obvious proof of the emptiness of the semialgebraic set. The coefficients witnessing that $f \in \mathcal{I}$ and $f \in \mathcal{P} + \mathcal{U}^2$ constitute the algebraic certificate.

In the context of probabilistic reasoning, we can now certify when the model of counterexamples $\mathcal{M}(\varphi)$ of an inference formula φ is empty. Hence, we obtain proofs for the *validity* of true inference rules. On the other hand, if an inference formula is wrong, there must be a counterexample. A famous theorem in model theory implies that this counterexample can be chosen algebraic; see [2, Proposition 5.2.3].

Tarski's transfer principle The semialgebraic set defined by $\{ f_i = 0, g_j \geq 0, h_k \neq 0 \}$ is non-empty over some real-closed field if and only if it is non-empty over every real-closed field, in particular the real closure of $\mathbb{Q}$.

The counterexample is a symmetric matrix $\Sigma \in \mathrm{Sym}_N(\mathbb{K})$ whose entries come from some finite real extension $\mathbb{K}$ of $\mathbb{Q}$. By the Primitive element theorem $\mathbb{K} = \mathbb{Q}(\alpha)$ and all entries of Σ may be represented exactly on a computer as polynomials in the minimal polynomial of α. This allows again verification of a claim about the *invalidity* of an inference formula by off-the-shelf computer algebra systems. In summary:

Alternatives in Gaussian CI inference If φ is a true inference rule for Gaussians, there exists a final polynomial proof for it with integer coefficients. Otherwise there exists a counterexample to φ with real algebraic coordinates.

The Matúš identity is a final polynomial for weak transitivity (and all other gaussoid axioms). For four and more Gaussian random variables, there exist inference rules which are valid but do not follow from the gaussoid axioms. To be precise, there are five of them up to symmetry, for example [9, Lemma 10 (20)]

$$[i \perp\!\!\!\perp j \mid k] \wedge [i \perp\!\!\!\perp k \mid l] \wedge [i \perp\!\!\!\perp l \mid j] \Rightarrow [i \perp\!\!\!\perp j]. \quad (4)$$

This inference rule is valid for 4×4 positive definite matrices. From its proof one can extract the following fact: the bracket polynomial

$$[ij|\emptyset] \cdot \Big([jk][jl|\emptyset]^2[kl|\emptyset]^2 + [j][k]^2[l][jl] + [j][k][kl][jl|\emptyset]^2 \Big) \quad (5)$$

is in the ideal generated by the assumptions of (4). But this polynomial splits into the desired conclusion $[ij|\emptyset]$ and another factor which is a sum of non-negative polynomials at least one of which is positive as a product of principal minors. Hence, $[ij|\emptyset]$ must vanish whenever the assumptions are satisfied by a positive definite matrix.

Notably, counterexamples to (4) exist when positive definiteness is relaxed to allow (indefinite) matrices all of whose principal minors do not vanish. This condition of *principal regularity* is a natural substitute for positive definiteness over algebraically closed fields. An example is this matrix:

$$\begin{array}{c} \\ i \\ j \\ k \\ l \end{array} \begin{array}{c} \begin{array}{cccc} i & j & k & l \end{array} \\ \begin{pmatrix} 1 & 4 & -2 & -8 \\ 4 & 1 & -2 & -2 \\ -2 & -2 & 1 & 1/4 \\ -8 & -2 & 1/4 & 1 \end{pmatrix} \end{array}.$$

This shows that positive definiteness is a crucial feature of our reasoning task. In particular, we may not find all valid inference rules by working in an algebraically closed field and basing all computations on ideals only.

Computation and hardness

The problem of deciding whether a proposed inference formula φ is valid for all Gaussian distributions over ground set N reduces the problem of checking if the model of counterexamples $\mathcal{M}(\varphi)$ is empty. This is a problem in the *existential theory of the reals* (ETR) and the associated complexity class of all problems which reduce in polynomial time to ETR is known as $\exists\mathbb{R}$; cf. [14]. This is a fundamental complexity class in computational geometry, polynomial optimization and statistics. We have thus an upper bound on the complexity of the Gaussian CI inference problem.

Unfortunately this upper bound is attained in the complexity-theoretic sense due to a universality result for Gaussian CI models. Similar universality theorems have been known for the realizability of rank-3 matroids [6], 4-polytopes [12] or Nash equilibria of 3-person games [7]. See [4, Chapter 5] for a more thorough discussion.

Theorem 13 ([4]) The Gaussian CI inference problem is co-$\exists\mathbb{R}$-complete.

This theorem is proved by encoding synthetic geometry in the real projective plane in conditional independence and dependence constraints. Even though CI models are at first glance very special semialgebraic sets, they possess no structure which makes them in general easier to work with than arbitrary semialgebraic sets.

There exist implementations of quantifier elimination over the real numbers via cylindrical algebraic decomposition in computer algebra systems such as Wolfram

`Mathematica`. These methods, when they terminate, decide the emptiness of a semialgebraic set and in the inhabited case they return a real algebraic number certifying this. Software for finding final polynomials does not seem to be readily available but there are recent advances in computing real radical ideals [1].

An even more fundamental obstacle in the tabulation of all valid inference rules among five Gaussian random variables is finding reasonable candidate implications. The following problem should enable an experimental and data-driven approach:

Problem 14 Develop (numerical) software for sampling positive definite points uniformly from varieties inside $\mathrm{Sym}_N(\mathbb{R})$.

Sampling allows to check if a proposed inference formula has any obvious counterexamples. It can also aid in testing candidates for final polynomials, such as the left-hand side in (3) or the expression (5), by evaluating these candidates on sufficiently many samples and checking whether they vanish. Since CI equations are continuous, small numerical errors can be tolerated.

References

[1] Lorenzo Baldi and Bernard Mourrain: Computing Real Radicals by Moment Optimization. In *Proceedings of the 2021 International Symposium on Symbolic and Algebraic Computation (ISSAC'21)*, pages 43–50, Association for Computing Machinery (ACM), 2021.

[2] Jacek Bochnak, Michel Coste, and Marie-Françoise Roy: *Real algebraic geometry*. Ergebnisse der Mathematik und ihrer Grenzgebiete, 3. Folge, vol. 36, Springer, 1998.

[3] Tobias Boege: Gaussoids are two-antecedental approximations of Gaussian conditional independence structures. *Ann. Math. Artif. Intell.*, 90:645–673, 2022.

[4] Tobias Boege: *The Gaussian conditional independence inference problem*. Ph.D. thesis, OvGU Magdeburg, 2022.

[5] Tobias Boege, Alessio D'Alì, Thomas Kahle and Bernd Sturmfels: The Geometry of Gaussoids. *Found. Comput. Math.*, 19(4):775–812, 2019.

[6] Jürgen Bokowski and Bernd Sturmfels: *Computational synthetic geometry*. Lecture Notes in Mathematics, vol. 1355, Springer, 1989.

[7] Ruchira S. Datta: Universality of Nash equilibria. *Math. Oper. Res.*, 28(3):424–432, 2003.

[8] Claudia Fevola and Christiane Görgen: The Mathematical Research-Data Repository MathRepo. *Der Computeralgebra-Rundbrief Nr. 70*, 2022.

[9] Radim Lněnička and František Matúš: On Gaussian conditional independence structures. *Kybernetika*, 43(3):327–342, 2007.

[10] František Matúš: Conditional independences in gaussian vectors and rings of polynomials. In Gabriele Kern-Isberner, Wilhelm Rödder, and Friedhelm Kulmann, editors: *Conditionals, Information, and Inference*, pages 152–161, Springer, 2005.

[11] Judea Pearl: *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[12] Jürgen Richter-Gebert: *Realization spaces of polytopes*. Lecture Notes in Mathematics, vol. 1643, Springer, 1997.

[13] Jürgen Richter-Gebert: *Perspectives on projective geometry. A guided tour through real and complex geometry*. Springer, 2011.

[14] Marcus Schaefer and Daniel Štefankovič: Fixed points, Nash equilibria, and the existential theory of the reals. *Theory Comput. Syst.*, 60(2):172–193, 2017.

[15] Petr Šimeček: Classes of Gaussian, discrete and binary representable independence models have no finite characterization. In *Proceedings of Prague Stochastics*, vol. 400, pages 622–631, 2006.

[16] Milan Studený: *Probabilistic Conditional Independence Structures*. Information Science and Statistics, Springer, 2005.

[17] Seth Sullivant: Gaussian conditional independence relations have no finite complete characterization. *J. Pure Appl. Algebra*, 213(8):1502–1506, 2009.

[18] Seth Sullivant: *Algebraic Statistics*. Graduate Studies in Mathematics, vol. 194, American Mathematical Society (AMS), 2018.

Automated Reasoning in the Class

I. Drămnesc (West University of Timişoara, Romania)
E. Ábrahám (RWTH Aachen, Germany)
T. Jebelean (Johannes Kepler University of Linz, Austria)
G. Kusper (Eszterhazy Catholic University of Eger, Hungary)
S. Stratulat (Univ. de Lorraine, CNRS, LORIA, Metz, France)

isabela.dramnesc@e-uvt.ro
abraham@cs.rwth-aachen.de
tudor.jebelean@jku.at
gkusper@aries.ektf.hu
sorin.stratulat@univ-lorraine.fr

Introduction

Automated Reasoning is a field of Computer Science concerned with the automation of deduction processes in Mathematics. For this purpose, algorithms for logical deduction (Automated Theorem Proving) have been developed. Theorem provers allow to prove mathematical statements interactively by computer programs, SAT solvers are available for checking the satisfiability of propositional logic formulae, and SMT (Satisfiability Modulo Theories) solvers that combine logical procedures with methods from Computer Algebra (e.g. for the manipulation of polynomial expressions) can be used to check the satisfiability of Boolean combinations of constraints from certain theories.

Automated Reasoning finds application in numerous areas, for example for the verification of computer hardware and software, and in general of complex systems, as well as semantical information storage and retrieval. Therefore it is crucial for computer specialists to understand the theoretical basis and familiarize themselves with the underlying algorithms, in order to be able to develop Automated Reasoning tools or to use them for solving their problems. Moreover, this knowledge is also necessary for the development of semantically aware Internet repositories and tools to access them. In practice, we currently encounter numerous situations in which design or implementation errors in complex systems lead to undesired results, ranging from small nuisances to important loss of value and even to fatalities. In our opinion, an important cause for such errors is the lack of systematic use of formal modeling and verification tools, and one important way of changing this situation is the improvement of the education of students and of the academic staff in fields related to Computational Logic.

As the field of Automated Reasoning is relatively new, there is yet not enough teaching material that sufficiently addresses mathematically involved concepts for the education of students with diverse background, and in particular material that helps to increase the motivation of both the students and the teachers and supports efficient interaction.

The project *ARC (Automated Reasoning in the Class)* aimed at improving the teaching of subjects related to Automated Reasoning by producing teaching material and tools that support the activities in the class and by training academic staff on how to use them. The project was funded in the frame of the *Erasmus+* programme of the European Union as a partnership between 5 universities from Romania (coordinator), Austria, France, Germany, and Hungary, running from 2019 to 2022.

The main activities of the project have been: production of a book and related tools (available on the project home page [1]), 5 modules for training of teaching staff, a summer school for students, and an international symposium for disseminating the project results. The book *"Computational Logic: A Practical Approach"* describes the main models from Mathematical Logic and the most important algorithms from Automated Reasoning, with corresponding illustrative exercises. The tools are various programs that illustrate the main methods and can be used to support the teaching based on the various sec-

tions of the book. The training of the academic staff for using the tools and the book took place in 5 one–week modules with the subjects: *Mathematica and Theorema* (Linz, Austria), *Satisfiability Module Theories (SMT) Solving* (Aachen, Germany), *Problem Based Learning* (Timişoara, Romania), *SAT Solving* (Eger, Hungary), and *Coq* (Metz, France).

As an illustration of our approach we describe in more detail the teaching process of the DPLL algorithm [2, 3] for checking the satisfiability of propositional logic formulas. The material and the tools for the other algorithms described in our book are freely available on the home page of the project [1].

Teaching the DPLL Algorithm

Problem-based Learning

To motivate the application of logics and automated reasoning, we developed material to illustrate how certain problems can be encoded logically; later on, these logical encodings can also be used to illustrate the execution of automated reasoning algorithms and the application of relevant tools.

In this article we focus on *propositional logic* [4], whose formulas connect propositions (Boolean variables) by the unary operation of negation ($\neg$) and the binary operators of conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$) etc. A *literal* is a proposition or its negation; a *clause* is a disjunction of literals; a propositional logic formula in *conjunctive normal form* (*CNF*) is a conjunction of clauses. Below we give two examples how to encode problems in propositional logic.

Example: *The pigeon hole problem*. Let $n \in \mathbb{N}\setminus\{0\}$. The *pigeon hole problem* is the problem to decide whether $n+1$ pigeons fit into n holes, if no two pigeons fit into one hole. This problem is an example which can be easily solved by humans but which often challenges automated reasoning tools. We can encode this problem in propositional logic as follows, where $x_{i,j}$ stands for pigeon i being in hole j ($1 \le i \le n+1$, $1 \le j \le n$):

$$\left(\bigwedge_{i=1}^{n+1}\left(\bigvee_{j=1}^{n} x_{i,j}\right)\right) \wedge \left(\bigwedge_{j=1}^{n}\bigwedge_{i1=1}^{n}\bigwedge_{i2=i1+1}^{n+1}\left(\neg x_{i1,j} \vee \neg x_{i2,j}\right)\right)$$

Above, the left operand of the outmost conjunction encodes that each pigeon is in at least one hole, and the right operand encodes for each pair of pigeons that they cannot be in the same hole. Note that we break the symmetry for the pigeon pairs in the second block, ordering the pairs by increasing indices. Furthermore, we require only that each pigeon is in at least one hole; one could also encode the fact that each pigeon is in at most one hole, but since we only want to decide the existence of a solution we know that if there is a solution in which a pigeon uses more than one hole than there is also a solution in which it uses exactly one hole (the others being empty). On this example we can illustrate the relevance of how we encode problems.

Example: *Sudoku Light*. Assume an $n \times n$ square grid for some $n \in \mathbb{N} \setminus \{0\}$. Each square in the grid is either empty or it contains a natural number from $\{1, \ldots, n\}$. We want to fill each empty square in the grid with numbers such that each row and each column contain exactly one occurrence of each number from $\{1, \ldots, n\}$. This problem can be encoded by the propositional logic formula

$$\varphi_{init} \wedge \varphi_{squares} \wedge \varphi_{rows} \wedge \varphi_{columns}$$

using propositions $g_{i,j,k}$ for $i, j, k \in \{1, ..., n\}$ to encode that the square in row i and column j contains the number k; φ_{init} is a conjunction stating for each initially non-empty field in row i and column j with number k the truth of $g_{i,j,k}$; and the following sub-formulas:
Each square has at most one number:

$$\varphi_{squares} := \bigwedge_{i=1}^{n}\bigwedge_{j=1}^{n}\bigwedge_{k1=1}^{n-1}\bigwedge_{k2=k1+1}^{n}\left(\neg g_{i,j,k1} \vee \neg g_{i,j,k2}\right)$$

Each row contains each number:

$$\varphi_{rows} := \bigwedge_{i=1}^{n}\bigwedge_{k=1}^{n}\left(\bigvee_{j=1}^{n} g_{i,j,k}\right)$$

Each column contains each number:

$$\varphi_{columns} := \bigwedge_{j=1}^{n}\bigwedge_{k=1}^{n}\left(\bigvee_{i=1}^{n} g_{i,j,k}\right)$$

Teaching the DPLL Algorithm with Mathematica

The Davis–Putnam–Logemann–Lovelace (DPLL) algorithm [2, 3] solves SAT problems in conjunctive normal form by investigating on a search tree a certain subset of the possible assignments to the propositional variables and by unit propagation.

A *unit* clause consists of a single literal – note that such a clause determines the truth assignment of the corresponding variable, since each of the clauses of a CNF formula need to be satisfied in order to satisfy the whole formula. Unit propagation uses this information to simplify the set of clauses: all instances of the opposite of this literal (whose truth value is *false*) are removed from the corresponding clauses (*unit resolution*) and all clauses containing an instance of this literal (whose truth value is *true*) are deleted (*unit subsumption*). (The unit clause itself is also removed by this rule, but the corresponding assignment is kept in case we want to actually find the solution[s].) Note that unit resolution may also produce new units – in this case they are also propagated (*Boolean constraint propagation*) – as well as an empty clause or an empty set of clauses.

When there is no unit clause in the current set, then the algorithm proceeds by *branching*: it chooses one of the variables and it produces two branches for the possible truth assignments and propagates these. A heuristics decides which branch to process first.

The search on a branch finishes in two possible ways. If unit resolution produces the empty clause (a *contradiction* is found), then the formula is not satisfied on the

current branch of the search tree; the algorithm backtracks by moving up along the current path to the most recent branching point with a yet unexplored child and continues with processing that child. If unit subsumption empties the current clause set, then the formula is satisfied for the current assignment of the variables. In case we are only interested in satisfiability, then the search can stop at the first such satisfying situation.

Some variants of DPLL also use the *pure literal rule*: if a literal occurs with only one sign in the problem, then all the corresponding clauses can be removed without changing the satisfiability of the problem.

The formula is unsatisfiable if contradiction occurs on all branches. Otherwise, each satisfiable branch produces one or more solutions (the variables that have not been assigned can have any value).

Example: *DPLL solving*. Fig.1 shows the search tree for the set of clauses listed in node 0.

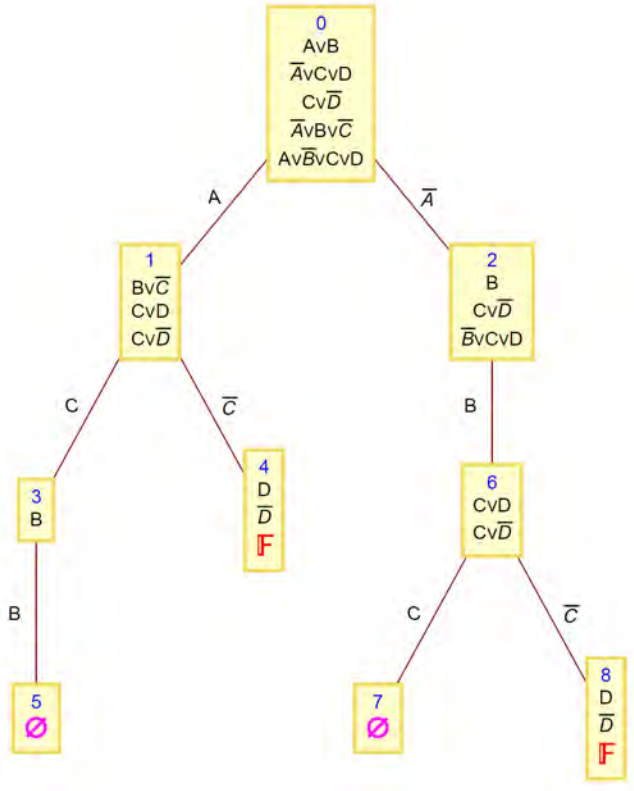


Figure 1: *Example of DPLL solving.*

Because there is no unit clause in the original set, the first step is a branching one, using the variable A and producing the two new sets 1 and 2. (One can use any of the other variables and then the sets would be different.) On the branch corresponding to A the first and the last clause are removed, while the second and the fourth clause loose the negation of A. On the branch corresponding to the negation of A the second and the third clause are removed, while the first and the last clause loose A. On both branches the third clause remains unchanged.

Set 1 again does not contain any unit, thus branching is applied to C producing sets 3 and 4. In set 3 the second and the third clause are removed and the first clause looses B. In set 4 the first clause is removed and the last two clauses loose the negation of C.

Set 3 becomes empty either by unit propagation or by the pure literal rule, thus the assignment of true to A, B and C satisfies the formula. On can check that this assignment satisfies the original set by observing that each of the original clauses contains at least one of these three variables. The two possible assignments to D give two satisfying full assignments.

Set 4 gives a contradiction, thus we can infer no solution here.

Set 2 has pure literal C, thus true assignment to this satisfies the last two clauses. If we use the pure literal rule then we can eliminate them, and then we have again pure literal B that finishes the search. This gives the solution assignments that have negative A and positive B and C, with D of any value. However note that by using the pure literal rule we could ignore some of the solutions, because in principle also the negative C may be part of some solutions (however this is not the case here).

If we do not use the pure literal rule, then unit B is propagated and we obtain the set 6, in which branching is applied on C and we obtain a satisfying assignment on set 7 and contradiction on set 8.

The Chaff implementation

Chaff [5] is an implementation of the DPLL algorithm that is *lazy*: it avoids some unnecessary operations. For instance when a clause is deleted, all previous updates of this clause by unit resolution are useless.

This implementation does not use recursion, but it explores the search tree in an explicit way. However the amount of backtracking data is very small: only the assignments to the variables and the branching variables need a stack. This is because the information about the watched literals (details follow) does not need to be backtracked: the algorithm runs correctly no matter which are the watched literals.

In order to avoid some unnecessary operations one uses the concept of *watched literals*. Namely, at the beginning, two literals from every clause are setup to be watched, and a clause is updated (*visited*) only when one of the watched literals must be removed. Otherwise, in the process of unit propagation the clauses are neither removed and the literals are also not deleted, but of course the necessary information (a clause or a literal should have been deleted) is used by just checking the current assignment of the corresponding variables. This method allows to avoid many operations, however the fact that a clause becomes unit is detected immediately as explained below.

We describe here a particular version of Chaff, having all the essential features. At the beginning the watched literals are established in an arbitrary way, and for each variable we instantiate a record containing:

- the truth assignment (initially unassigned);
- two lists of clauses on which the variable occurs as positive, respectively as negative literal, each

composed of two sublists (occurrence as the first, respectively the second watched literal).

Additionally the algorithm uses a *backtrack stack* for the branching variables, a *variable stack* for currently assigned variables, and a *unit queue* for the units that have been found but not yet propagated. All these are initially empty.

The main loop has three kinds of steps:

- Branching: If the unit queue is empty:
  - if all variables are assigned we have a solution, store it and go to backtracking.
  - otherwise choose an unassigned variable, assign it true, put it in the backtrack stack and in the variable stack, and go to unit propagation for the positive literal of this variable.
- Continue unit propagation: If the unit queue is not empty, choose one of the variables and propagate the literal corresponding to its current assignment.
- Backtracking:
  - If the backtrack stack is empty then stop, the search tree is exhausted.
  - Otherwise pop the variables from the variable stack and unassign them one by one, until the same variable occurs in the backtrack stack. Remove the variable from the backtrack stack, assign the variable to false, and propagate the negative literal of this variable.

Unit propagation of a literal consists in scanning the lists of clauses corresponding to the opposite of this literal and visiting each of them.

Visiting a clause consists in the following:

- If the other watched literal is assigned true, end the visit (the clause should have been removed).
- Otherwise, scan the list of the literals that are not watched:
  - If the scan ends then either there are non watched literals, or they should all have been removed: this clause is a unit - namely the other watched literal, try to assign the corresponding value to the variable.
    - If the variable has no value: assign the value found and put the variable in the unit queue.
    - If the variable already has the same value: end the visitation.
    - If the variable already has the opposite value: contradiction, end the visitation, end the unit propagation, and go to backtracking in the main loop.
  - If the scanned literal has no value: in this clause switch this literal with the watched literal that triggered the visitation and update the list of clauses that contain watched literals in the records corresponding the two variables involved, then end the visitation.
  - If the scanned literal is positive: end the visitation (the clause should have been removed).
  - If the scanned literal is negative: continue to scan (the literal should have been deleted).

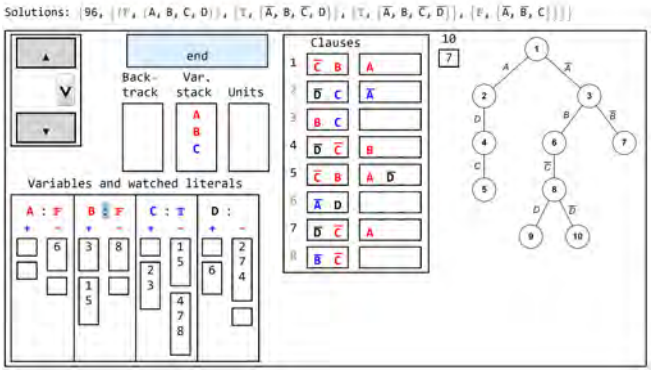


Figure 2: *Demo of the Chaff algorithm.*

We developed a Mathematica implementation of the DPLL algorithm; our implementation can generate and display the corresponding search trees, as show in Figure 1. Furthermore, in Mathematica we developed another interactive simulation tool for the Chaff algorithm, whose functionalities we illustrate next on an example.

Example: *Chaff*. Fig.2 presents the screen shot at the end of the interactive animation demonstrating the Chaff algorithm on the set of clauses shown in the table "Clauses". The box on the LHS of each clause contains the watched literals. The buttons on the LHS upper corner can be used for navigating forward and backward through the demo. The upper box indicates the state of the main loop. The search tree is displayed on the RHS only for a better understanding by the user, it is not actually necessary for the running of the algorithm. Over the box we see the final result: after 96 elementary steps we have two contradictions (nodes 5 and 7) and two satisfying assignments (nodes 9 and 10).

Teaching: After presenting the main structure of the algorithm as above, we show the students several examples as the previous ones and explain how the algorithm is applied. The pictures are produced dynamically using our tools implemented in Mathematica, and the students can themselves experiment interactively using different examples.

Exercise Generation for the DPLL Algorithm

Interaction during lectures enable the students' active participation and increases the effectiveness of learning. We made especially good experience with *bonus questions*. These are posed during the lectures (in average one

question per 45 minutes lecture). The teacher first poses a small *example* problem and gives the students some fixed time (3-5 minutes) to try to solve it. After that, the teacher presents the solution and gives the students the opportunity to ask questions. Successively, each student gets an *individual* problem to solve within a fixed time window (3-8 minutes). These individual problems should differ from the example problem just in certain parameters, such that the solutions for the individual problems can be achieved through the same steps as for the example problem. If time permits, the students should be given the possibility to discuss (face-to-face or online) or even to check each other's solutions. Solutions can be submitted on an online learning platform like Moodle. For each correct solution, the students can earn *bonus points* for the final exam. According to the short time frame per task, the posed problems should be small and cover a central concept.

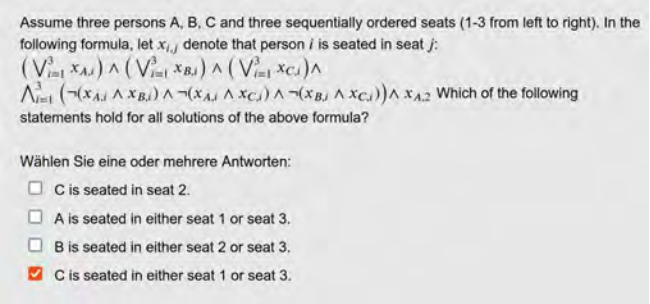
Besides bonus questions, similar problems are needed to design weekly exercise sheets, intermediate tests, and written exams. Also here, it is advantageous to have *parametric* problems, whose instances share a common solution scheme and a comparable solution effort. Optimally, also the problem sizes should be *scalable*, as the students have more time to solve the exercise sheets in home work than the bonus questions during the lectures.

The definition of such parametric problems and the automated generation of their instances is highly challenging, but once implemented, also highly rewarding. Individual problem instances assigned to the students have a strong motivating effect, they harden cheating and they can be evaluated automatically, which is very important when teaching large classes.

However, for topics related to computer algebra and symbolic computation, it is hard to come up with a clear measure that defines the quality of such parametric problems. We developed a catalogue of quality criteria, covering aspects of (i) which problems to choose, (ii), how to formulate the problems, (iii) which questions to ask and (iv) how to give feedback. We do not discuss the above catalogue here in detail, but present some aspects on a few examples.

In our project, we developed 36 such parametric problems for different topics covered in an elective lecture on *Satisfiability Checking* taught at RWTH Aachen University by Ábrahám, with 200–550 registered students in the previous years. In the following we present three of these problems that are related to the Chaff algorithm.

Example: *Propositional logic*. The following problem instance should check the understanding of the syntax and semantics of propositional logic, as well as aspects of encoding real-world problems in propositional logic.



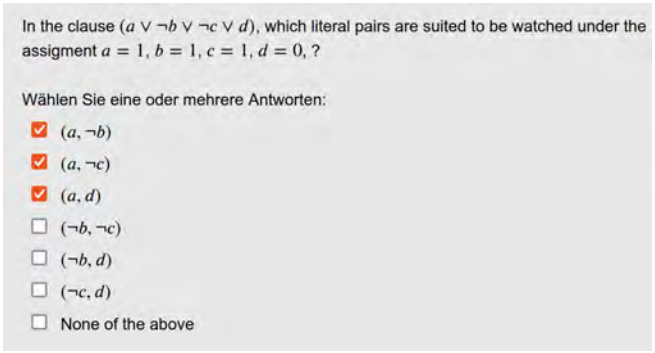
In the problem statement, we could ask e.g. to encode by a propositional logic formula the problem to seat three people in three sequentially ordered seats under certain side conditions. However, this would require a free textual answer, which we want to avoid for several reasons: it would require to fix the input syntax; reading this input syntax specification takes valuable time; it might be inconvenient to answer on cell phones; it is unclear how to evaluate incorrect syntax; the answer would not be unique, such that students might be uncertain which answer to give, furthermore some of the encodings might be easier to find than others, and the time needed to type different solutions might vary.

Instead, we present a propositional logic formula and assign meanings to its propositions, and ask to select all correct answers from a list (single/multiple choice). Though this form is more easily guessable, having a list with at least 4 choices (and thus at least 4 possible answers for single choice and 16 for multiple choice) lowers the probability of lucky guessing. If possible, these answer choices should cover regular but also corner cases, highlight important points in definitions etc.

To achieve a parametric problem, we defined 3 complex and 27 easy sub-formulas related to the seating problem, and further 32 statements formalized in natural language as well as in propositional logic. To generate a problem instance, we randomly select 2 complex and 1 easy sub-formulas and build their conjunction. Then we use a SAT solver to assert this formula and classify all 32 statements in tautologies and non-tautologies. Having done this, we randomly select four times one of the two classes and a statement from that class as choices. We assure to select both classes at least once and that there will be at least three statements in both classes. By randomly selecting first the tautology class and then an instance from that class, we assure that each possible answer is equally probable, making the guessing harder (as typically there will be less tautologies than non-tautologies under the statements).

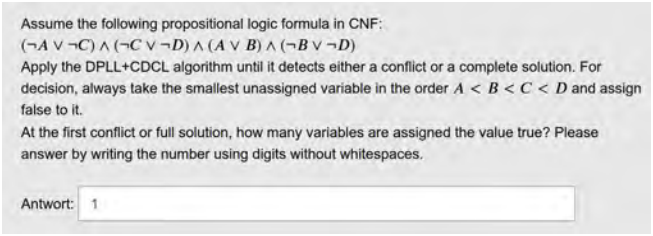
Example: *Watched literals for DPLL*. With the next parametric problem we train the understanding of the watched-literals scheme. We fix four propositions (we use a, b, c, d, but the naming could be also chosen randomly from a given set of options). For each problem instance, we generate a clause with four literals, each literal being one of the propositions with a random sign, yielding $2^4 = 16$ clause variants. Next we generate an assignment of *true*, *false* or *unassigned* to each of the propositions, offering $3^4 = 81$ possible (partial) assignments. The students should decide which literal pairs are

suited to be watched together for the given clause under the given (partial) assignment. Again, to avoid free-text answers, we use the multiple-choice format. To reduce the probability of successful guessing, we include all 7 possible literal pairs (up to ordering) as options. Note the additional option *None of the above*, to exclude the possibility that points will be awarded for not answering the question.



Example: *Boolean constraint propagation*. This task addresses the combined mechanisms of branching and propagation in the DPLL algorithm (excluding backtracking). We again fix how many propositions we want to use (here: 4) and fix their names (here: A, B, C, D), along with static orderings for propositions (here: alphabetic) as well as for the Boolean values (here: $false < true$). A further parameter is the number (here: 4) and maximal size (here: 4) of the clauses. To create a problem instance, we randomly generate the different clauses: for each clause, we randomly decide for each proposition whether it is included, and for each included proposition we select a random sign. We assure that no clause is empty, that the clauses are pairwise different and that each proposition appears at least once. We simulate the DPLL algorithm until either a conflict is detected or until it terminates without reaching any conflict. To be comparable, we process only those instances that reach a full solution after making a number of propagated assignments from a given interval (here: at least two), and dismiss all other instances (i.e. we repeatedly generate new random instances until the above condition is met).

We now need to fix a question to be asked, which provides a good indication whether the students successfully applied the DPLL algorithm, without any complicated input syntax (e.g. providing the final assignment) and without offering good chances to be guessed correctly (e.g. the value of a given proposition in the final assignment). In this example we ask for the number of true propositions in the final assignment, the possible answers being any number between 0 and 4. Using more propositions would further reduce the probability for successful guesses.



Further DPLL-related Topics

Further aspects of automated reasoning for propositional logic that we could not cover in this article but are covered by the ARC material include the conflict-driven clause learning (CDCL) [5] approach (that most state-of-the-art SAT solvers implement), parallel SAT solving techniques and other algorithms not based on the DPLL algorithm. For hands-on exercises, the standard DIMACS [6] input format for SAT solvers is introduced.

Conclusion

In this article we reported on teaching materials developed in the *ARC (Automated Reasoning in the Class)* project. These materials can be used in lectures, seminars and practical courses, as introductory material to prepare for B.Sc., M.Sc. and PhD projects, as well as in the context of training schools. We hope that the community will find these materials useful. We aim to further maintain and extend this collection in the future and are grateful for feedback and external contributions.

Acknowledgements. This work is co-funded by the Erasmus+ Programme of the European Union, project ARC: Automated Reasoning in the Class, 2019-1-RO01-KA203-063943.

References

[1] `https://arc.info.uvt.ro/`

[2] Davis, M., Hilary, P.: A computing procedure for quantification theory. J of ACM **7**(3) (1960) 201–215

[3] Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Commun. ACM **5**(7) (Jul 1962) 394–397

[4] Biere, A., Heule, M., van Maaren, H., Walsh, T.: Handbook of Satisfiability. Volume 185 of Frontiers in Artificial Intelligence and Applications. IOS Press (2009)

[5] Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: Proc. of the 38th Annual Design Automation Conference, Association for Computing Machinery (2001) 530–535

[6] `http://www.domagoj-babic.com/uploads/ResearchProjects/Spear/dimacs-cnf.pdf`

Local Inversion of maps: Black Box Cryptanalysis

Virendra Sule,
Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai 400076, India
vrs@ee.iitb.ac.in

viren.sule@gmail.com

Abstract

This paper is a short summary of the theory of local inversion announced in [6] and results announced in a previous paper on a new universal method for cryptanalysis which uses a Black Box linear algebra approach to computation of local inversion of nonlinear maps in finite fields. It is shown that one local inverse x of the map equation $y = F(x)$ can be computed by using the minimal polynomial of the sequence $y(k)$ defined by iterates (or recursion) $y(k+1) = F(y(k))$ with $y(0) = y$ when the sequence is periodic. This is the only solution in the periodic orbit of the map F. Furthermore, when the degree of the minimal polynomial is of polynomial order in the number of bits of the input of F (called low complexity case), the solution can be computed in polynomial time. The method of computation only uses the forward computations $F(y)$ for given y which is why this is called a Black Box approach.

An application of this approach is then shown for cryptanalysis of several maps arising in cryptographic primitives. It is shown how in the low-complexity cases maps defined by block and stream ciphers can be inverted to find the symmetric key under known-plaintext attack. Then it is shown how the RSA map can be inverted to find the plaintext as well as an equivalent private key to break the RSA algorithm without factoring the modulus. Finally, it is shown that the discrete log computation in finite field and elliptic curves can be formulated as a local inversion problem and the low-complexity cases can be solved in polynomial time.

Introduction

If $F : \mathbb{F}^n \to \mathbb{F}^n$ is a map acting in a cartesian space $\mathbb{F}^n$ over a finite field $\mathbb{F}$, then the *Local Inversion Problem* of F at a given y in $\mathbb{F}^n$ is the problem of computing all x in $\mathbb{F}^n$ such that

$$y = F(x) \tag{1}$$

Such a problem arises in most situations of cryptanalysis of symmetric and public key primitives such as the problem of key recovery (under a known-plaintext attack) in symmetric-key algorithms, plaintext recovery in RSA without factoring the modulus, private-key recovery in RSA without factoring the modulus, or finding discrete logarithms in finite fields and elliptic curves. However, in many such cases, no algebraic or Boolean (in case of $\mathbb{F}$ being $\mathbb{F}_2$) model of Eq. (1) may be available or, if available, is unsuitable for algorithmic solution due to a large number of unknowns in the equations. The map F in such cases can only be practically used as a black box: an algorithm or a hardware machine or a computer code for efficient forward computation of $F(x)$ for a given x.

It is hence suggested that the approach to the solution of cryptanalysis problems using only forward computations by F be called a *Black Box approach* analogous to the well-known Black Box Linear Algebra for solving linear systems of equations. In this paper, we develop such an approach and determine the required conditions for the local inversion to be computable in polynomial time in order $O(n^k)$. Although no assumption such as F being invertible (or a permutation) is made, practically and most often, computing one solution x of a local inversion is a significant achievement even if there may be more solutions. This approach is therefore a universal method to formulate diverse problems of cryptanalysis such as the key recovery of symmetric encryption algorithms, the breaking of RSA without factoring the modulus, and the computation of the discrete logarithm in finite fields and elliptic curves over finite fields as local inversion problems.

Local inversion using the Black Box approach under low complexity data

It is known in the literature that the algebraic approach to solving Eq. (1) poses a very challenging computational problem. Either the algebraic equations expressing the relation (1) involve too many (latent) variables and equations for which most approaches fail to scale up, or building the algebraic (or Boolean model) of (1) with the minimum number of variables by symbolic elimination is yet another challenging computation. Hence, it is worthwhile considering the Black Box approach in which F is

not specified algebraically but the output $y = F(x)$ can be computed for any input x efficiently. It is shown in [6] that the existence of a linear recurrence relation in the periodic sequence $y, F(y), F^{(2)}(y), \ldots$ with *Linear Complexity*[1] (LC) of polynomial size $O(n^k)$ is sufficient to solve one solution x (or a rational root) in polynomial time[2]. This sequence is generated recursively by the map F and the output y at an input x, and is different from the output sequence of a cryptosystem such as a stream cipher. Hence the LC of this recursive sequence is different from that of the LC of the output sequence (often called the LC of the cryptosystem). Cryptosystems are designed with the purpose of having large LC of output sequences. However, even if the output sequences have large LC, the recursive sequence above can have low LC since it also depends on the input (plaintext or IV) and the cipher design does not necessarily consider the constraint of a high LC of this recursive sequence. This result has an important consequence to applications in cryptanalysis.

It turns out that the cryptanalysis of symmetric-key algorithms, RSA cryptanalysis of inverting ciphertext to plaintext, as well as breaking RSA encryption by the same public key from chosen ciphertext attack (CCA) and solutions of Discrete Logarithms in finite fields and Elliptic Curves (ECDLP) are all local inversion problems. Hence, the low LC of respective sequences is a sufficient condition for solving them in polynomial time. Local inversion is also a new method and can be used as a universal attack for solving all of these cryptanalysis problems.

This article is a very short summary of the local inversion method announced in [6] which the reader is expected to refer to for details and proofs of results. The inversion of maps for cryptanalysis has been studied in previous literature by the Time–Memory Tradeoff (TMTO) attack [1, 2, 7]. However, TMTO does not consider finite fields' structure on the domain of the maps. TMTO is also referred to as Rainbow attack [8], which is governed by the square-root bound on the number of points in the image of F, which is of exponential size in the number of unknowns. Hence, this attack is not feasible for realistic sizes of domains (or number of unknowns) of maps F (or Eq. (1)).

Theory of local inversion of maps

It is stated above that local inversion of a map F at y is equivalent to solving the roots of the system of (polynomial) Eq. (1) if such a system is available. Hence, this appears to be a problem solvable by computer algebra, by building the algebraic system of equations corresponding to the map F. Alternatively, we observe that the solutions of (1) generate trajectories of the dynamical system

$$y(k+1) = F(y(k)), y(k) \in \mathbb{F}^n, k = 0, 1, 2, \ldots \quad (2)$$

which, in turn, are in one-to-one correspondence with the recurring sequences

$$S(F, y) = \{y, F(y), F^{(2)}(y), \ldots\}. \quad (3)$$

Thus, the theory behind the local inversion of maps relates three different mathematical domains, maps F defined by algebraic equations in finite fields, dynamical systems defined by maps F, and the ensemble of sequences generated by maps F with respect to points y in their range. The complexity of solving the problem of local inversion depends on the distribution of linear complexities of sequences $S(F, y)$ where y is taken over the range of F. The shift of focus from solving an algebraic system model of $y = F(x)$ to a Black Box model of F generating sequence $S(F, y)$, gives a practically feasible solution to the local inversion problem under the condition of low LC of the sequence $S(F, y)$.

Role of Linear Complexity (LC)

In the literature, LC was proposed as a complexity measure for sequences in finite fields. Such a complexity measure was used for qualifying the complexity of output sequences of maps defined by block ciphers for counter inputs (in counter mode of operation), or for outputs of stream ciphers [3]. The LC of several specially constructed sequences is also studied in Number Theory [4]. Low-LC m of such sequences enables the Berlekamp–Massey attack which allows for prediction of the complete sequence over the whole period from $2m$ terms. In the local inversion approach, however, the sequence $S(F, y)$ defined recursively by F and y is completely different from a sequence of an output stream of the cipher for counter input or output of stream cipher for a fixed IV. The LC of output sequences is used for modeling the sequence as an output sequence of LFSRs. But such modeling does not solve the symmetric key recovery problem. Recursive sequences such as $S(F, y)$ have not been studied previously for known cipher algorithms or general maps from the objective of inversion of maps. The application of the concepts of LC and minimal recurrence for local inversion is a fresh idea proposed in [6] and presents a Black Box approach for the inversion of non-linear maps.

A practical constraint in solving the problem

Due to the finiteness of the field $\mathbb{F}$, the recurring sequences $S(F, y)$ are quasi-periodic for any y. However, even when the sequence is periodic, the full sequence $S(F, y)$ is never practically available for computation since the period may be of exponential order in number of variables n or the field size $n = \log |\mathbb{F}|$. Hence, only a small number of terms in the sequence $S(F, y)$ of order $O(n^3)$ are available for solving x given y. Therefore, the data of a limited number of terms of $S(F, y)$ may be insufficient to compute any solution x or may

[1] i.e. the length of the linear recurrence relation

[2] Polynomial time of a computation: the computation is achievable in $O(n^k)$ steps, equivalently in $O((\log q)^k)$ steps in the field $\mathbb{F}_q$

even compute a false-positive solution which needs to be eliminated by verifying whether $y = F(x)$.

Linear Complexity of a sequence

A given sequence $\widehat{s} = \{s_0, s_1, s_2, \ldots\}$ over a finite field $\mathbb{F}$ is always ultimately periodic due to the finiteness of $\mathbb{F}$, i.e. there exist numbers $r \geq 0$ and $N > 0$ such that

$$s_{(k)} = s_{(k+N)} \text{ for } k \geq r. \tag{4}$$

The smallest r is called the pre-period of $\widehat{s}$ and the smallest N is called period. The sequence is called periodic of period N when $r = 0$. A polynomial can be associated with the periodic sequences which satisfy the relation (4) by defining a one-step right-shift operator on the sequence,

$$X(\widehat{s}) = \{s_{(k+1)}, k = 0, 1, 2, \ldots\}$$

and scalar multiplication of $\widehat{s}$ by constants. Then the relation (4) for periodic sequences can be expressed as

$$s_k = X^N(s_k) \text{ for } k \geq 0.$$

which is equivalent to

$$(X^N - 1)(\widehat{s}) = 0.$$

In general, if

$$p(X) = X^d - \sum_{i=0}^{d-1} \alpha_i X^i$$

is any polynomial in $\mathbb{F}[X]$ such that with the definition of X being a shift operator as defined above,

$$p(X)(\widehat{s}) = 0 \tag{5}$$

then $p(X)$ is called a characteristic polynomial of $\widehat{s}$. Thus, $(X^N - 1)$ is a characteristic polynomial. A characteristic polynomial of smallest degree is unique and is called the *minimal polynomial* of $\widehat{s}$. The degree of the minimal polynomial is called the *Linear Complexity* (LC) of the sequence. Now, getting back to the sequence $S(F, y)$ defined in Eq. (3), let $p(X)$ be a characteristic polynomial of $S(F, y)$. Then the relation (5) signifies a *linear recurrence relation* of degree d satisfied by $S(F, y)$,

$$F^{(d+j)}(y) - \sum_{i=0}^{d-1} \alpha_i F^{(i+j)}(y) = 0 \tag{6}$$

for $j = 0, 1, 2, \ldots$ If $S(F, y)$ is periodic of period N, $(X^N - 1)$ is always a characteristic polynomial. But there is a possibility of a lower-degree characteristic polynomial satisfied by $S(F, y)$. Hence, there exists a minimal polynomial for $S(F, y)$ which is a characteristic polynomial of least degree and is unique because the polynomial is monic.

Solution of the local inversion problem

First consider the theoretical situation in which there are no limitations on the number of terms within the sequence $S(F, y)$ being available for inversion. In the first theorem below, we show that computation of the minimal polynomial solves the local inversion problem when the sequence $S(F, y)$ is periodic. Let the minimal polynomial of a periodic $S(F, y)$ be denoted as

$$f(x) = X^m - \sum_{i=0}^{m-1} \alpha_i X^i. \tag{7}$$

Theorem 1 Let the sequence $S(F, y)$ be periodic of period N and $f(x)$ be its minimal polynomial. Then

1. $f(0) \neq 0$.
2. N is the order of $f(x)$ over $\mathbb{F}$.
3. The solution of the local inverse of $y = F(x)$ in the periodic $S(F, y)$ is

$$x = \frac{1}{\alpha_0} \left[F^{(m-1)}(y) - \sum_{i=1}^{m-1} \alpha_i F^{(i-1)}(y) \right]. \tag{8}$$

Proof 1 Let $f(x)$ as denoted in Eq. (7) be the minimal polynomial of $S(F, y)$. Since the period of the sequence is N, $f(x)|(X^N - 1)$ which implies $\alpha_0 = f(0) \neq 0$ and since N is the smallest number such that the recurrence

$$F^{(N)}(y) = y$$

holds, N is the order of $f(x)$. As $f(x)$ is also a characteristic polynomial of $S(F, y)$,

$$f(x)(S(F, y)) = 0$$

which is equivalent to

$$F^{(m)}(y) - \sum_{i=0}^{m-1} \alpha_i F^i(y) = 0$$

Let x be the solution of $y = F(x)$ in the same periodic orbit of the dynamical system (2). Then $S(F, x)$ is the sequence $S(F, y)$ shifted right by one index due to which the sequence satisfies the same recurrence relation (6) and the relation

$$f(x)(S(F, x)) = 0$$

which is equivalent to

$$F^{(m)}(x) - \sum_{i=0}^{m-1} \alpha_i F^i(x) = 0$$

w.r.t. the minimal polynomial. Substituting for $y = F(x)$ and solving for x from the above equation as $\alpha_0 \neq 0$ gives the solution as stated.

It is important to observe that the solution x of the local inverse is obtained in terms of a linear combination of the sequence $\{y, F(y), F^{(2)}(y), \ldots, F^{(m-1)}(y)\}$.

An incomplete algorithm

We now come to the practically most relevant problem of local inversion of a map F at y. The practical constraint to be faced is that the sequence $S(F, y)$ is not available for the full period because the period is exponential in n (or close to the size of the field when $n = 1$). However, the partial sequence is available up to M terms where M is of polynomial size $O(n^k)$ for some k as the sequence,

$$\{y, F(y), \ldots, F^{(M-1)}(y)\}. \tag{9}$$

In this case, the recurrence relation (6) is satisfied only up to the degree $m = \lfloor M/2 \rfloor$ since there is no further data of the sequence available beyond M terms. Hence, the largest degree of the minimal polynomial is m. As shown in Section 2.6 of [6], the minimal polynomial of the limited sequence up to M terms is obtained from the unique solution $\widehat{\alpha}$ of the linear system

$$H(k)\widehat{\alpha} = h(k+1) \tag{10}$$

where $H(k)$ is the Hankel matrix defined as

$$H(k) = \begin{bmatrix} y & F(y) & \cdots & F^{(k-1)}(y) \\ F(y) & F^{(2)}(y) & \cdots & F^{(k)}(y) \\ \vdots & \vdots & \ddots & \vdots \\ F^{(k-1)}(y) & F^{(k)}(y) & \cdots & F^{(2k-1)}(y) \end{bmatrix}$$

of largest rank for $k \leq \lfloor M/2 \rfloor$. The vector of coefficients of the minimal polynomial of degree k is denoted $\widehat{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_{(k-1)})^T$, and the vector on the r.h.s. of Eq. (10), $h(k+1)$, is the last column of $H(k+1)$ after dropping the bottom entry.

The Incomplete Algorithm 1 is useful to decide whether the local inverse can be computed from the given data. Since the given data is the partial sequence of M terms, which is of polynomial size, the algorithm computes the inverse in polynomial time when the data is sufficient to compute the inverse. The algorithm also checks for false-positive cases of solutions and, if no solution is found, returns a string stating that the data is insufficient.

We can thus conclude:

Theorem 2 Let M be of polynomial order $O(n^k)$. If the sequence (9) has a minimal polynomial satisfying the recurrence relation (6) of degree $m \leq \lfloor M/2 \rfloor$ and the solution x computed in (8) satisfies $y = F(x)$, then the local inverse is computable in polynomial time.

In practice, the polynomial bound $O(n^k)$ is limited to $k = 1, 2, 3$ for realistic cases.

Algorithm 1 Incomplete algorithm to find the unique solution of $y = F(x)$ given y

```
procedure LocalInversion(given M terms of S(F, y))
    Input M of O(n^k) and the sequence (9).
    repeat
        m ← ⌊M/2⌋
        if m = rank H(m) = rank H(m + 1) then
            // A unique minimal polynomial of degree m exists
            Compute min. polynomial and solution x as in (8).
            if x satisfies y = F(x) then
                return x
            end if
        else if rank H(m + 1) > m then
            return "Insufficient data to compute the
              local inverse."
        else   // rank H(m) < m
            M ← M − 1
        end if
    until M = 2
    return "Insufficient data to compute the local inverse."
end procedure
```

Embedding maps

The aforementioned algorithm for local inversion is applicable to maps $F : \mathbb{F}^n \to \mathbb{F}^n$ where the number of bits n of the input and the output are the same. In many practical situations of cryptanalysis, however, the maps arise as $F : \mathbb{F}^n \to \mathbb{F}^m$ where $m > n$. Such a map is called an embedding map. The solution of local inverse $y = F(x)$ for such a map can be found using the theory developed above for regular maps with $m = n$ as explained in the following.

Solution of local inverse for embedding maps

Consider a projection Π from $\mathbb{F}^m \to \mathbb{F}^n$. For a vector y in $\mathbb{F}^m$, $\Pi(y)$ denotes a vector of some of the n components of y. If x is a local inverse of $y = F(x)$ for any such projection, $\Pi(y) = (\Pi \circ F)(x)$. Therefore, x is also a local inverse of $y_1 = \Pi(y)$ under the map $F_1 = \Pi \circ F$. Since F_1 is a standard map in $\mathbb{F}^n$, we can utilize the recurrence sequence generation using F_1 and the minimal polynomial to find a possible local inverse of $y_1 = F_1(x)$. Then such a solution is verified with all other maps to verify whether x satisfies $\Pi_k(y) = F_k(x)$ for all other projections Π_k of $\mathbb{F}^m$ to $\mathbb{F}^n$. A solution is rejected if it fails to satisfy the equation for any k. Hence, we can define the LC of an embedding map to be the smallest LC of the recurrences defined by $y_k = F_k(x)$ over all projections $\Pi_k : \mathbb{F}^m \to \mathbb{F}^n$. In fact, it can be observed that a restricted set of projections on the following subsets of coordinates of y are sufficient to compute and verify the local inverse x,

$$\begin{aligned} \Pi_1(y) &= \{y^1, \ldots, y^n\} \\ \Pi_2(y) &= \{y^2 \ldots, y^{(n+1)}\} \\ &\vdots \\ \Pi_{(m-n+1)} &= \{y^{(m-n+1)}, \ldots, y^m\} \end{aligned} \tag{11}$$

The computational effort depends on the minimal polynomial of recurrences $y_k = F_k(x)$ at the first choice

of k which chooses a projection. Hence, an embedding map can still be locally inverted in polynomial time if there is an index k such that the recurrence defined by $y_k = F_k(x)$ has a LC of polynomial order in number of the variables, and the solution is verified by all other projections.

Typical maps in cryptanalysis

In this final section, we will outline some of the typical maps arising in cryptography which can be considered for local inversion to show that the local inversion approach is a uniform methodology for cryptanalysis. It must be understood, however, that the local inversion is a theoretical methodology – its practical feasibility depends on how small the LC of the local inversion problem as shown in Theorem 2 actually is.

Symmetric encryption algorithms

The two types of algorithms are block ciphers and stream ciphers.

Block ciphers: An algorithm is described by $C = E(K, P)$ where P, C are the blocks of plaintext and ciphertext, respectively, while K is the symmetric key block. In the cases of Known-Plaintext Attack (KPA), the pair (P, C) of blocks is known to the attacker. Thus, the local inversion problem is $y = F(x)$ where $y = C$, $x = K$ and $F(x) = E(x, P)$. The map F is from l bits of K to block length in bits of C. In the chosen plaintext attack, P is chosen such that the inversion problem is easier. For local inversion, P can be chosen such that the LC of $F(x)$ is smallest or polynomially bounded. Such an input may not be easy to compute, though.

Stream ciphers: The algorithm is described by a dynamic system with a state update map $x(k+1) = F(x(k))$ and an output map $y(k) = f(x(k))$. The initial condition $x(0) = (K, IV)$ consists of the symmetric key K (with bit length l) and an initializing vector IV. Hence, the local inversion problem is defined by $y = \widehat{F}(x)$ where y is the vector of output stream $(y(k_0), y(k_0+1), \ldots, y(k_0+l-1))$ while

$$\widehat{F}(x) = ((F^*)^{(k_0+i)} f(x, IV)),$$
$$\text{for } i = 0, 1, 2, \ldots, (k_0 + l - 1)$$

The map $\widehat{F}$ can be turned into an embedding by collecting more samples of the output stream.

For both types the maps for key recovery are available in black-box form with l bits of input x (key length) and output y on which the inversion algorithm can be applied after choosing M bounded by a polynomial order $O(l^k)$.

RSA cryptanalysis

RSA has public keys $n = pq$ where p, q are odd primes and an exponent e such that $\gcd(e, \phi(n)) = 1$. The private keys are p, q, d where $ed = 1 \bmod \phi(n)$. Two local inversion problems can be defined as follows:

Decryption of ciphertext: For the message $m \in [0, n-1]$ the ciphertext is $c = m^e \bmod n$. Let l be the bit-length of n. For an l-bit number x let (x) denote the bit-string in the binary expansion of x and $[x]$ denote the operation of recovering the number x from the binary string (x). Then the map $F : \mathbb{F}_2^l \to \mathbb{F}_2^l$ is defined by $y = (c)$ and $F(x) = ([x]^e \bmod n)$. The sequence $S(F, y)$ in $[0, n-1]$ is

$$\{c, c^e \bmod n, c^{e^2} \bmod n, c^{e^3} \bmod n \ldots\}$$

while the sequence $S(F, y)$ in $\mathbb{F}_2^l$ is

$$\{(c), ([c]^e \bmod n), ([c]^{e^2} \bmod n), ([c]^{e^3} \bmod n), \ldots\}.$$

It is well-known that the sequence $c^{e^k} \bmod n$ is periodic since e is co-prime to $\phi(n)$. The LC is obtained from the sequence of binary vectors. The message m is obtained by solving the local inverse $(c) = F(x)$, $m = [x]$. This way of solving for m shows how RSA ciphertext can be decrypted by local inversion without factoring n. In cases where c causes a low LC of the above sequence, the encryption can be broken in polynomial time as shown by Theorem 2.

Chosen Ciphertext Attack: In this attack, a ciphertext c is chosen by the attacker and the decrypted plaintext m is sought as a verification. The relation is $m = c^d \bmod n$. Hence, the local inverse problem is defined by $y = m$ and $F(x) = c^x \bmod n$. Since d belongs to the ring $\mathbb{Z}_{\phi(n)}$, the number of bits of x is equal to $\phi(n)$. Although $\phi(n)$ is not available to the attacker, we can choose the number of bits to be n. The sequence $S(F, y)$ is

$$\{m, c^m \bmod n, c^{c^m \bmod n} \bmod n, \ldots\}$$

while the map $F : \mathbb{F}_2^l \to \mathbb{F}_2^l$ is defined by $(m) = (c^{[x]} \bmod n)$ and the sequence of binary vectors is

$$\{(m), (c^{[m]} \bmod n), (c^{[c^{[m]} \bmod n]} \bmod n), \ldots\}.$$

Note that the computation modulo n automatically restricts the exponent's modulo $\phi(n)$, hence the sequence generated is correct for the local inversion of the map F. The periodicity of the sequences above is explained in [6]. The sequence is converted to a sequence of binary expansions to compute the LC and the local inverse x. The inverse x is converted back to a number in $[0, n-1]$. Now, an important observation to be noted in this inversion is that the inverse of $m = F(x)$ is x which is not necessarily d but satisfies $ex = 1 \bmod \phi(n)$. Therefore, although the private key d is not exactly recovered, the inverse allows for the decryption of any other ciphertext $\widetilde{c}$ corresponding to the plaintext $\widetilde{m}$ since we have

$$(\widetilde{c}^x \bmod n)^e \bmod n = \widetilde{c}^{ex} \bmod n = \widetilde{c},$$

hence $\widetilde{m} = \widetilde{c}^x \bmod n$. This shows that local inversion by CCA on RSA breaks RSA equivalent to computing the private d key without factoring the modulus n.

Discrete logarithm in finite fields and elliptic curves

The Discrete Logarithm Problem (DLP) has been the first major tool for development of Public Key Cryptography, same as the Diffie–Hellman key-exchange scheme (DH). Later on, the DH scheme was upgraded to elliptic curves, leading to Elliptic Curve Cryptography (ECC). The DLP on Elliptic Curves (ECDLP) has not been found to have an algorithm for solution better than exponential complexity. What we show in this section is that the solution of both of these problems can be addressed by local inversion. Thus, it turns out that at least in cases where the map F and given data y meet the conditions so that the sequence $S(F, y)$ has low LC, the DLPs can be solved efficiently. The situation in case of ECDLP is very complicated because the map F turns out to be an embedding.

DLP on finite fields

In a prime field $\mathbb{F}_p$ for a large prime p, the exponent function with a base a in $\mathbb{F}_p^*$, $\phi : [1, p-1] \to \mathbb{F}_p, x \mapsto a^x \bmod p$ is actually defined as a map from $\mathbb{F}_p^*$ to $\mathbb{F}_p^*$. Taking the binary expansion of numbers in $[1, p-1]$, where l is the bit-length of p, this function expresses a map operation

$$\begin{aligned} F: \quad & \mathbb{F}_2^l \to \mathbb{F}_2^l \\ & (x) \mapsto (a^{[x]} \bmod p). \end{aligned}$$

Here (x) denotes the binary string corresponding to a number $x \in [1, p-1]$ and $[x]$ the reconstruction of the number x from the binary string. Let b be a given element in $\mathbb{F}_p^*$ for a primitive element a. Then the equation for local inversion is $F((x)) = (b)$. The map is available for black box computation. The sequence of binary vectors $S(F, (b))$ is

$$\{(b), (a^{[b]} \bmod p), (a^{a^{[b]} \bmod p} \bmod p), \ldots\}. \qquad (12)$$

As defined by the map iterations $y(0) = (b)$, $y(k+1) = F(y(k))$ for F defined as above on $\mathbb{F}_2^l$. The reader is referred to [6] to see details justifying the periodicity of the sequence.

The solution of the DLP can thus be found as local inversion of this map F at the value (b). If the LC of the above sequence given up to polynomial number of terms $O(l^k)$ is small, then, as described in Theorem 2, the DLP can be solved in polynomial time using Algorithm 1. For the solution of DLP over general field $\mathbb{F}_q$ using local inversion, the reader is referred to [6].

DLP on elliptic curves

In the discrete log problem on an elliptic curve E over $\mathbb{F}_q$, there are given points P and $Q = [m]P$ in E where m is the integral multiplier. It is required to solve for the multiplier m. Define the map

$$F_P : m \mapsto [m]P$$

then the local inversion of $Q = F_P(m)$ solves the ECDLP. However, the map F needs to be expressed in the standard form as before. We can do this by defining the map in $\mathbb{F}_2^l$ for an appropriate l.

Formulation as local inversion

The multiplier m is less than the order of the cyclic group $n = \langle P \rangle$ in E. Sometimes the group E itself has prime order $n = \sharp E$, hence the order of $\langle P \rangle$ is n. To fix the number of bits l in m we consider estimates of the order of E. The well-known bound on the order of E is

$$\sharp E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}$$

Assuming $q > 4$ we have $\sharp E \leq 2q$. On the other hand, the point Q in E has two co-ordinates in $\mathbb{F}_q$. Thus, the bit length of $\sharp E \leq 1 + \log(2q) = 2 + \log q = l$ while the bit length of two co-ordinates of a point taken together is $1 + \log(2q) = l$. Consider the map F_P defining the scalar multiplication of P in E

$$\begin{aligned} F_P: \quad & \mathbb{Z}_n \to E \\ & m \mapsto [m]P \end{aligned}$$

In the binary co-ordinate expansion on both sides, this mapping is

$$F_P((m)) = ((Q_x), (Q_y)) \qquad (13)$$

where (m) denotes the coefficients in the binary expansion of m and (Q_x), (Q_y) are coefficients in the binary expansions of co-ordinates of $Q = [m]P$. We shall denote the binary expansion of the co-ordinate pair of Q as (Q).

Formulation of F_P as a map over the binary field

In order to utilize the previous theory of inversion on the map (13), it is necessary to express it as a map in the cartesian spaces of $\mathbb{F}_2$ and verify that it is an embedding.

Let $r = 1 + \log n$ where n is the order of $\langle P \rangle$. Then F_P in (13) represents a map $F_P : \mathbb{F}_2^r \to \mathbb{F}_2^l$ where $r < l$. Thus, F_P is an embedding and it is required to apply the theory of local inversion of embeddings to solve the embedding equation for the local inversion of $F_P(m) = Q$ from the binary representation in (13). The application of Algorithm 1 requires that n the order of P is known. Let $t = l - r$, then the projection equations 11 give $(t+1)$ standard equations

$$\Pi_i \circ F_P(x) = \Pi_i((Q)), i = 1, 2, \ldots (t+1)$$

denoted by $F_i = \Pi_i \circ F_P$ and $y(i) = \Pi_i((Q))$ where Π_i are a projection on r components of y. The details of this projection are described in [6]. Following Theorem 2, we now have:

Theorem 3 If for any of the indices i, $1 \leq i \leq (t+1)$ the projection equation $F_i(x) = y(i)$ has a periodic recurrence sequence $S(F_i, y(i))$ with a LC $m \leq \lfloor M/2 \rfloor$ where M is of polynomial order $O(l^k)$ and the local inverse x satisfies all other projection equations, then the ECDLP is solved in polynomial time by Algorithm 1.

Conclusions

This brief paper summarizes the ideas on a new method which can be referred to as a Black Box approach to cryptanalysis by Local Inversion of maps in finite fields. The methodology is universal in the sense that it is applicable to diverse problems of cryptanalysis of both symmetric as well as public-key cryptography. The most important conclusion arising from this method is that, since the recursive sequences associated with the inversion problems identified by the method have not been studied in the past for their LC for most of the maps F in cryptography, urgent work should be carried out to determine density of points y in the image of F for which the sequences $S(F, y)$ have low LC. These densities should be useful for deciding the grading of security of parameters of cryptographic primitives and should be included as a standard for the design of ciphers.

The algorithm proposed for local inversion always works in polynomial time in the number of unknown variables n of the map F, i.e. when it discovers a correct solution the solution is computed in polynomial time. Another practical advantage of the algorithm is that it does not require an algebraic model of the map equation $y = F(x)$ but only uses forward computation at specified arguments x. Hence this approach is called black box cryptanalysis.

References

[1] Martin Hellman: *A cryptanalytic time-memory trade-off. IEEE Trans. on Information Theory, 26(4), pp.401-406*, 1980.

[2] Howard M. Hays: *Distributed Time Memory Trade-off Attacks on Ciphers.* `http://eprint.iacr.org/2018/123`.

[3] Rainer A. Rueppel: *Analysis and Design of Stream Ciphers.* Springer Verlag, Berlin, Heidelberg, 1986. ISBN 9783540168706.

[4] Harald Niederreiter: *Linear complexity and related complexity measures for sequences. Indocrypt 2003, LNCS 2904, pp.1-17*, Springer Verlag, 2003.

[5] Lawrence Washington: *Elliptic curves, Number Theory and Cryptography.* Chapman and Hall/CRC Press, 2003.

[6] Virendra Sule: *Local inversion of maps: A new attack of Symmetric Encryption, RSA and ECDLP.* `https://arxiv.org/abs/2202.06584v2`, March, 2022.

[7] Mark Stamp and Richard M. Low: *Applied Cryptanalysis.* Wiley-Interscience, 2007.

[8] Rainbow table, Wikipedia: `https://en.wikipedia.org/wiki/Rainbow_table`.

Explizite Methoden in Zahlentheorie und Algebra an der Universität Siegen

Die seit Ende 2021 existierende und unter der Leitung von Tommy Hofmann (Juniorprofessor) stehende Gruppe ist in die Fachgruppe für Algebra, Diskrete Optimierung, Geometrie & Zahlentheorie am Department Mathematik der Universität Siegen eingebettet; letzterer gehören zudem die Professoren Mohamed Barakat, Jörg Jahnel und Rob van Stee an. Thematischer Schwerpunkt sind algorithmische Fragestellungen und Anwendungen im Bereich der Zahlentheorie und Algebra. Neben der Entwicklung neuartiger und Verbesserung bestehender Algorithmen beinhaltet dies insbesondere auch die Untersuchung und numerische Verifikation schwieriger Vermutungen aus der Zahlentheorie. Konkrete Forschungsvorhaben befassen sich unter anderem mit den folgenden Themen:

1. Hauptidealproblem und S-Klassengruppenberechnung. Dies sind Probleme aus der klassischen algebraischen Zahlentheorie, bei deren Lösung wir durch Ausnutzen zusätzliche Struktur von normalen Zahlkörpern sowohl thereotische als auch praktische Fortschritte erzielen. Hier verfolgen wir auch die Verbindung zur gitterbasierten Post-Quanten-Kryptographie.

2. Inverse Galoistheorie und Zahlköperaufzählung. Hauptziel ist die effiziente Konstruktionen von Zahlkörpern mit vorgegebenen Eigenschaften (Galoisgruppen oder Verzweigung) und statistische Fragestellungen, die sich damit untersuchen lassen; etwa die Verteilung von p-adischen Regulatoren.

3. Nichtkommutative Zahlentheorie. Nichtkommutative endlich-dimensionale Algebren wie etwa Gruppenalgebren und deren ganzzahlige Strukturen spielen eine zentrale Rolle in der Galoismodultheorie. Die Algorithmisierung dieser Theorie ist eines unserer Ziele, insbesondere im Hinblick auf die numerische Verifikation von Vermutungen, beispielsweise der äquivarianten Tamagawa-Zahl-Vermutung.

4. Arithmetische Gruppen. Für arithmetische Gruppen, zum Beispiel $\mathrm{GL}_n(\mathbf{Z})$, sind zahlreiche algorithmische Fragestellungen bekanntermaßen entscheidbar. Ergebnisse hinsichtlich praktikabler Algorithmen oder Komplexitätsaussagen sind hingegen kaum bekannt. Mit einem Hauptaugenmerk auf dem Konjugationsproblem und dessen Anwendungen ist es eines unserer Ziele diese Lücken zu schließen (für ausgewählte Familien von Beispielen).

5. HECKE und OSCAR. Die Implementierung der von uns entwickelten Algorithmen erfolgt mit wenigen Ausnahmen in der Programmiersprache JULIA. In diesem Rahmen findet bei uns ein wichtiger Teil der Entwicklung des Softwarepakets HECKE statt, welches die ganze Bandbreite klassischer algorithmischer algebraischer Zahlentheorie abbildet und auch moderne Algorithmen aus unseren Themenbereichen bereitstellt. Für Fragestellungen, welche die gesamte Breite algorithmischer Werkzeuge brauchen, wird OSCAR eingesetzt und weiterentwickelt.

Tommy Hofmann (Universität Siegen)

Publikationen über Computeralgebra

Journal of Computational Algebra

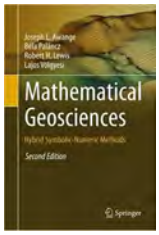
In diesem Jahr wurde im Bereich der Computer-Algebra ein neues Open-Access-Journal gegründet. Weitere Informationen befinden sich auf der Homepage

`https://www.journals.elsevier.com/journal-of-computational-algebra`

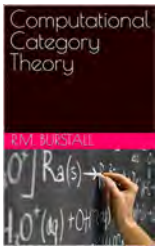
Im Zentrum liegen algebraische und diskrete Strukturen. Hierbei sollen sowohl die Mathematik als auch die Algorithmen zu neuen Ergebnissen beitragen. Bei Einreichungen bis zum 31.12.2022 wird auf die Artikel-Gebühr (APC) verzichtet.

Jürgen Klüners (Universität Paderborn)

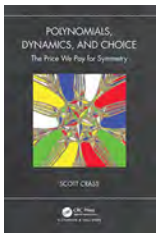
Neuerscheinungen:



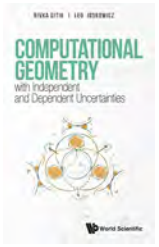
Joseph Awange et al.,
Mathematical Geosciences: Hybrid Symbolic-Numeric Methods,
2nd Edition, Springer-Verlag,
Nov. 2022, 744 Seiten,
ISBN 978-3030924942



R.M. Burstall,
Computational Category Theory,
Kindle Edition, Sep. 2022,
Prentice Hall,
318 Seiten,
ISBN 978-0131627369

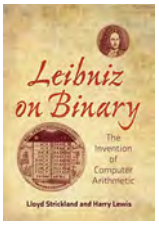


Scott Crass,
Polynomials, Dynamics and Choice: The Price We Pay for Symmetry,
CRC Press, Aug. 2022,
190 Seiten,
ISBN 978-0367564933



Rivka Gitik, Leo Joskowicz,
Computational Geometry with Independent and Dependent Uncertainties,
World Scientific, Sep. 2022,
160 Seiten,
ISBN 978-9811253836

C.G. Raab, M.F. Singer (Hrsg.),
Integration in Finite Terms: Fundamental Sources,
Texts and Monographs
in Symbolic Computation,
Springer, Juni 2022,
312 Seiten,
ISBN 978-3030987664



L. Strickland, H.R. Lewis,
Leibniz on Binary: The Invention of Computer Arithmetic,
The MIT Press, Oct. 2022,
248 Seiten,
ISBN 978-0262544344

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 Jürgen Klüners oder Martin Kreuzer (`klueners@math.uni-paderborn.de`, `martin.kreuzer@uni-passau.de`) auf.

Promotionen in der Computeralgebra

Dominik Bernhardt: Constructive aspects of wreath products and quasiprimitive permutation groups
Betreuer: Alice Niemeyer (Aachen)
Weiterer Gutachter: Max Horn (Kaiserslautern)
Juni 2022

Abstract: In 1993, Praeger introduced quasiprimitive permutation groups. A finite group acting on a finite set is said to act quasiprimitively if every non-trivial normal subgroup acts transitively. In a theorem similar to the famous O'Nan-Scott-theorem for primitive groups, Praeger classified quasiprimitive groups by dividing them into several mutually exclusive classes. Quasiprimitive permutation groups feature prominently in both the theory of permutation groups and in the study of symmetry groups of incidence structures. Yet, in contrast to arbitrary transitive or primitive permutation groups, no complete database of quasiprimitive groups up to a certain degree is available.

A main result of this thesis is the construction of a database of all quasiprimitive but imprimitive permutation groups - called quimp groups - of degree at most 4095. Together with the database of primitive groups of degree at most 4095 constructed with contributions by many authors, a database of all quasiprimitive permutation groups of degree at most 4095 is now available.

Praeger and Baddeley refined the original classification and divided quasiprimitive groups into eight mutually exclusive classes. To construct all quimp groups of degree at most 4095, we first observe that three of the eight types of quasiprimitive groups, namely groups of HA-, HS- and HC-type, are always primitive and that the minimal permutation degree of a quimp group of SD- or CD-type exceeds our degree bound. For the remaining three types of quasiprimitive groups, AS-type, PA-type and TW-type, we present structure theory and algorithms to construct such groups of a given degree. Our results are available in the GAP-package QuimpGrp which accompanies this thesis.

Many (quasi-)primitive groups arise as subgroups of wreath products. The second main result of this thesis is a constructive description of the conjugacy classes and centralisers and the solution to the conjugacy problem for wreath products where the base group is any group and the top group acts faithfully on a finite set. Our approach is inspired by ideas originally developed by Ore, Specht, James and Kerber and our theory is presented in a way that is close to the implementation. Our results are implemented in a GAP-package by Rober.

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.

Fachgruppe Computeralgebra

Berichte von Konferenzen

ISSAC 2022

Lille, Frankreich, 04.07. – 07.07.2022

`www.issac-conference.org/2022`

Endlich fand die ISSAC wieder in Präsenz statt, das graduelle Ende der Pandemie war also auch hier zu spüren. So konnten die Teilnehmer die für ein wissenschaftliches Ereignis so wichtigen informellen Treffen und Gespräche bei Kaffeepausen, Mahlzeiten, Gängen zwischen Gebäuden und Pausen zwischen Vorträgen in vollen Zügen genießen und davon profitieren. Dass dann nicht nur Mathematik sondern auch Zugverspätungen Thema sein können, gehört dazu. Allerdings galt das „in Präsenz“ mit der Einschränkung, dass Teilnehmer, die aus triftigen Gründen nicht anreisen konnten, auch online partizipieren konnten. De facto betraf das die Teilnehmer aus China, denn ihnen war insbesondere die Rückreise aufgrund des Festhaltens der dortigen Regierung an der Null-Covid Politik so gut wie unmöglich.

Deshalb mussten die Organisatoren trotz weitestgehendem Präsenzbetrieb die gesamte Infrastruktur für eine Hybridkonferenz zur Verfügung stellen, inklusive dem Herumreichen von Saalmikrophonen während der Fragerunden nach Vorträgen. All dies hat bemerkenswert gut funktioniert, was zahlreichen Helfern zu verdanken ist, aber maßgeblich auch Marc Moreno Maza als General Chair und François Lemaire als Local Arrangements Chair. Das Programmkomitee wurde mit ruhiger Hand von Lihong Zhi geleitet und wählte, nach Einholung zahlreicher Gutachten, aus den Einreichungen 53 Vorträge aus, die auf der ISSAC präsentiert werden durften. Neben diesen — jeweils in zwei Sektionen abgehaltenen — eingereichten Vorträgen gab es drei eingeladene Hauptvorträge und, als Auftakt zur Konferenz, drei Tutorials. Außerdem wurden 17 Poster und fünf Software-Demos präsentiert, die auch einen Auswahlprozess durchlaufen hatten. Für weitere Details, Titel und Autoren sei auf die Homepage der Konferenz verwiesen.

Vielen Lesern wird bekannt sein, dass die Fachgruppe Computeralgebra regelmäßig einen Beitrag für die ISSAC leistet, indem wir zwei Preise vergeben: den Distinguished Poster Award und den Distinguished Software Demonstration Award. Genauer gesagt: Die Fachgruppe stellt das Preisgeld von je 250 Euro, aber die Auswahl der Preisträger wird sinnvoller- und dankenswerterweise von den jeweils zuständigen Komitees geleistet. So war es auch in diesem Jahr, mit folgendem Ergebnis: Der Preis für das beste Poster ging an:

Preisträger: Antonio Jiménez-Pastor und Gleb Pogudin.
Titel: Computing exact nonlinear reductions of dynamical models.

Als beste Software-Demo wurde ausgezeichnet:

Preisträger: Philipp Nuspl.
Titel: C-finite and C^2-finite Sequences in SageMath.

Vielleicht passt an diese Stelle die Information, dass unsere Sprecherin, Anne Frühbis-Krüger, als „organizational member“ in das Steering Committee der ISSAC gewählt wurde.

Zugleich liefert dies eine Überleitung zum ISSAC Business Meeting, bei dem ein Tagesordnungspunkt wieder die Wahl eines neuen Mitglieds „at-large“ des Steering Committees war. (Der Bericherstatter erwägt ernsthaft, zu gegebener Zeit ein kleines ISSAC-Glossar zu erstellen.) Hier fiel die Wahl unter drei Kandidaten auf den schon oben erwähnten Gleb Pogudin (École Polytechnique, Frankreich). Für den Austragungsort der ISSAC 2024 lag diesmal nur eine Bewerbung vor, nämlich von der North Carolina State University. Aber die nächste Auflage der ISSAC ist vom 24. bis 27. Juli 2023 an der arktischen Universität in Tromsø, Norwegen, geplant.

Gregor Kemper (München)

Geometry meets Combinatorics

Bielefeld, 05.09. – 09.09.2022

`www.math.uni-bielefeld.de/geocomb`

Wie im Titel schon angedeutet war dieser Workshop – organisiert von Martina Juhnke-Kubitzke, Lukas Kühne, Raman Sanyal und Christopher Voll – ein Treffen verschiedener Communities. Darunter waren die Gebiete der Matroide, Polytope, Arrangements von Hyperebenen und Coxeter-Gruppen vertreten. Diese einzigartige Kombination führte dazu, dass knapp 100 TeilnehmerInnen (hauptsächlich aus Deutschland, bis auf wenige Ausnahmen ansonsten aus Europa) angemeldet waren, was für Tagungen aus der Kombinatorik in Deutschland beachtlich ist.

Im Fokus waren bei dieser Tagung teilweise Themen, die wegen der diesjährigen Vergabe der Fields-Medaille an June Huh aktuell sehr beliebt sind. Aber auch die Computeralgebra war in vielen Vorträgen präsent, etwa im Vortrag von Günter Ziegler, in dem er die Community aufforderte, Beispiele und Gegenbeispiele zu seinen Lieblingsfragen zu berechnen.

Michael Cuntz (Hannover)

(Geometry meets Combinatorics in Bielefeld, September 2022)

Hinweise auf Konferenzen

Nikolaus School on Computational Geometry

Kaiserslautern, 28.11.– 01.12.2022

www.mathematik.uni-kl.de/boehm/computationalgeometryschool

The school is aimed at Ph.D. students, post-docs, young researchers and anyone who is interested. It will feature mini courses by Gavin Brown (Univeristy of Warwick, tbc), Alexander M Kasprzyk (University of Nottingham), Vladimir Lazic (Universitaet des Saarlandes), and Frank-Olaf Schreyer (Universitaet des Saarlandes).
The school aims at interlinking the development of the next generation open source computer algebra system OSCAR to new topics from computational algebraic geometry, and to bring geometers in touch with OSCAR.
There will be the possibility for early career participants to present a lightning talk.

Nikolaus conference 2022

Aachen, 09.12. – 10.12.2022

www.math.rwth-aachen.de/Nikolaus2022

This is the announcement of the next Nikolaus conference at Lehrstuhl für Algebra und Zahlentheorie in Aachen! The dates are December 9-10, 2022.

Details and registration are here: https://www.math.rwth-aachen.de/Nikolaus2022.

CoCoA - School and Conference on Computational Commutative Algebra

Hue, Vietnam, 19.03. – 25.03.2023

cocoa.dhsphue.edu.vn

Die internationale Doktorandenschule und Tagung COCOA 2020 ist aufgrund der Pandemie mehrfach verschoben wurden und findet jetzt im März 2023 in Hue (Vietnam) statt. Weitere Informationen finden Sie demnächst auf der Website.

Computeralgebra-Tagung der Fachgruppe

Hannover, 31.05. – 02.06.2023

www.fachgruppe-computeralgebra.de

Schon zum 10. Mal richtete 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 wird sie am Mittwoch Mittag beginnen und am Freitag Mittag enden. Angesichts der zeitlichen Lage planen wir eine reine Präsenztagung.

Siehe ausführliche Ankündigung auf Seite 8.

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.

Registration and submission of Abstracts will be open by 14th of November 2022.

ACA 2023

Warschau, Polen, Juli 2023

math.unm.edu/aca.html

Traditionally, ACA (Applications of Computer Algebra) 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

Tromsö, 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, poster sessions, software demonstrations and vendor exhibits with a center-piece of contributed research papers.

All areas of computer algebra and symbolic computation are of interest at ISSAC 2023.

DMV-Jahrestagung 2023

Ilmenau, 25.09. – 28.09.2023

www.mathematik.de/dmv/jahrestagungen

Die DMV-Jahrestagung 2023 wird vom 25.9.-28.09.2023 in Ilmenau stattfinden. Weitere Informationen demnächst.

Fachgruppenleitung Computeralgebra 2020–2023

Sprecherin:
Prof. Dr. Anne Frühbis-Krüger
Carl-von Ossietzky Universität Oldenburg
Institut für Mathematik
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 Ábrahám
Fachgruppe Informatik
RWTH Aachen University
Ahornstr. 55, 52056 Aachen
0241 80-21242, -22243 (Fax)
abraham@cs.rwth-aachen.de
https://ths.rwth-aachen.de/people/erika-abraham/

Fachreferent CA an der Hochschule:
Prof. Dr. Michael Cuntz
Leibniz Universität Hannover
Institut für Algebra, Zahlentheorie und Diskrete Math.
Welfengarten 1, 30167 Hannover
0511 762-4252
cuntz@math.uni-hannover.de
https://www.iazd.uni-hannover.de/de/cuntz

Fachexperte 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:
Prof. Dr. Max Horn
Fachbereich Mathematik
Technische Universität Kaiserslautern
Gottlieb-Daimler-Straße, 67663 Kaiserslautern
0631 205-2730, -4427 (Fax)
horn@mathematik.uni-kl.de
https://www.quendi.de/de/mathe

Fachreferent Themen, Anwendungen und Publikationen:
Prof. Dr. Martin Kreuzer
Fakultät für Informatik und Mathematik
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
Lehrstuhl für Algebra und Zahlentheorie
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. Gregor Kemper
Zentrum Mathematik – M11
Technische Universität München
Boltzmannstr. 3, 85748 Garching
089 289-17454, -17457 (Fax)
kemper@ma.tum.de
https://www.math.cit.tum.de/algebra/kemper

Fachreferentin Industrie:
Xenia Bogomolec
Coding Services Hannover
Engelbosteler Damm 15, 30167 Hannover
0173 3031816
indigomind@protonmail.ch
https://quant-x-sec.com

Fachreferent CA-Systeme und -Bibliotheken:
Prof. Dr. Claus Fieker
Fachbereich Mathematik
Technische Universität Kaiserslautern
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

Fachreferent Themen, Anwendungen und Publikationen:
Prof. Dr. Jürgen Klüners
Mathematisches Institut der Universität Paderborn
Warburger Str. 100, 33098 Paderborn
05251 60-2646, -3516 (Fax)
klueners@math.uni-paderborn.de
https://math.uni-paderborn.de/ag/ca/

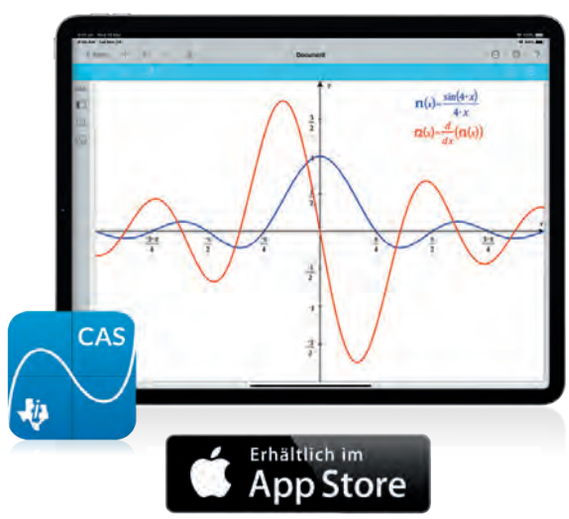
Fachreferent Redaktion Rundbrief:
Dr. Fabian Reimers
Zentrum Mathematik – M11
Technische Universität München
Boltzmannstr. 3, 85748 Garching
089 289-17474
reimers@ma.tum.de
https://www.math.cit.tum.de/algebra/reimers

TI-Nspire™ CX CAS Technologie

Kostenloses UPDATE

Einfach, schnell, sicher: Der **neue** Prüfungsmodus

Die neue Version 6.0 der TI-Nspire™ CAS App für iPad® verfügt über einen integrierten Prüfungsmodus, den Sie mithilfe eines praktischen Testcodes einrichten können. Aktuelle Anwender können kostenlos auf die neue Version updaten.



www.education.ti.com/de

Texas Instruments