# Gruppentheorie mit GAP

GAP ist ein Computeralgebra-System, welches sich insbesondere für gruppentheoretische Berechnungen eignet. Der Name GAP steht für Groups, Algorithms, Programming, womit der Zweck der Software bereits charakterisiert ist.

Dieser Artikel verfolgt die Absicht, GAP kurz vorzustellen und anhand einiger Beispiele dessen Nutzen anzudeuten, etwa im Finden von Inspirationen, Ersparen von mechanischen Rechnen, Lösungenentwurf und in der Ergebniskontrolle. Er soll weder Lehrbuch noch Einführung ersetzen, vielmehr werden am Artikelende Verweise auf Web-Ressourcen und Dokumentationen gegeben.

Vorausgesetzt werden Kenntnisse der ↗Gruppentheorie↗, später auch etwas Wissen aus der ↗Darstellungstheorie↗ endlicher Gruppen.

## Inhalt:



## 1 Über GAP

1986 an der RWTH Aachen enstanden, wird GAP seit 1997 an der St. Andrews University in Schottland gepflegt und weiterentwickelt. GAP ist freie Software und wurde unter die ↗GNU General Public License↗ gestellt. Der Quelltext ist ebenfalls frei verfügbar. Lauffähige Versionen existieren für Unix/Linux, Windows und Macintosh Computer.

Die Installation ist nicht schwierig und wird auf der ↗GAP-Homepage↗ beschrieben, wo Software und Quelltext zum Download bereitgestellt werden.

Installation und Start können sich auf verschiedenen Betriebssystemen unterscheiden, deswegen wird hier nicht näher darauf eingegangen.

GAP beinhaltet eine interpretierte Pascal-ähnliche Programmiersprache, eine Vielzahl algebraischer Algorithmen sowie Datenbibliotheken, welche beispielsweise tausende Gruppen konstruieren und Möglichkeiten zu deren Manipulation bereitstellen.

Im weiteren wird stets Bezug auf die Version GAP4 genommen.

# 2 Benutzung und erste Kommandos

In diesem Text wird die Benutzung von GAP nur skizziert und Kommandos sowie Syntax werden nur soweit erklärt, wie es für die Vorstellung und das Verständnis der nachfolgenden Beispiele nötig ist. Wer sich daraufhin für GAP interessiert, findet auf der GAP-Homepage eine Fülle von einführendem und dokumentierendem Material.

Nach dem Starten von GAP sieht man das GAP4-Banner sowie Informationen über geladene Komponenten und Pakete. Die Zeichenkette

```
gap>
```

symbolisiert den Eingabe-Prompt und steht am Beginn der Eingabezeile.

Eingaben werden mit Semikolon abgeschlossen, ein doppeltes Semikolon unterdrückt die Ausgabe der Berechnung bzw. des Befehls. Stets empfehlenswert ist das Einstellen einer automatischen Aufzeichnung der GAP-Sitzung, mit

```
gap> LogTo("name.txt");
```

sagt man GAP, daß alle Ein- und Ausgaben in der Datei name.txt protokolliert werden sollen - sehr praktisch für späteres Sichten der Rechnungen und Ergebnisse. Durch

```
gap> LogTo();
```

ist dies wieder abstellbar. Die Sitzung kann man beenden mittels:

```
gap> quit;
```

Gearbeitet wird textbasiert und interaktiv, d.h. man gibt einen Ausdruck ein, GAP berechnet das Ergebnis und zeigt es an. Beispielsweise eine Berechnung

von größtem gemeinsamen Teiler oder Binomialkoeffizienten:

```
gap> Gcd(12,66);
6
gap> Binomial(9,3);
84
```

Die Groß-/Kleinschreibung der Eingaben ist relevant. GAP besitzt eine interaktive Hilfefunktion, welche mit vorangestelltem Fragezeichen aufgerufen wird:

```
gap> ?Group;
gap> ?Stabilizer;
gap> ?help;
```

Neben direkten Berechnungen kann man in GAP auch umfangreiches archiviertes Wissen abrufen. Zum Beispiel, wenn man wie in ↗jenem Forumbeitrag↗ überlegt, wie Gruppen der Ordnung 8 aussehen können, dafür gibt es einen Abfrage-Befehl:

```
gap> SmallGroupsInformation(8);
 There are 5 groups of order 8.
     1 is of type c8.
     2 is of type 2x4.
     3 is of type D8.
     4 is of type Q8.
     5 is of type 2^3.

  The groups whose order factorises in at most 3 primes
  have been classified by O. Hoelder. This classification is
  used in the SmallGroups library.

  This size belongs to layer 1 of the SmallGroups library.
  IdSmallGroup is available for this size.
```

Wir erkennen $\mathbb{Z}_8, \mathbb{Z}_2 \times \mathbb{Z}_4$, die Diedergruppe mit $8$ Elementen, die Quaternionengruppe $\mathbb{Q}_8$ sowie $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$. Im angegebenen Forumthread gibt es einen Verweis auf eine Herleitung dieser Klassifikation.

„Small“ ist übrigens relativ - wir können Parameter weit größer als 2000 angeben.

Die meisten Kommandos und Ausdrücke sprechen für sich, und ihr Sinn ist deutlich, wenn man die Englische Sprache beherrscht, daher werde ich forthin die Beispielkommandos nur kommentieren, falls dies notwendig oder nützlich ist.

# 3 Gruppentheoretische Berechnungen

Gruppen können durch Angabe der Erzeuger festgelegt werden:

```
gap> D4:=Group((1,3),(1,2,3,4));
Group([ (1,3), (1,2,3,4) ])
gap> Elements(D4);
[ (), (2,4), (1,2)(3,4), (1,2,3,4), (1,3),
  (1,3)(2,4), (1,4,3,2), (1,4)(2,3) ]
gap> Size(d4);
8
gap> Centre(D4);
Group([ (1,3)(2,4) ])
gap> Size(last);
2
```

last bezeichnet die Variable, welche das letzte Resultat enthält.

Permutations- und Matrixgruppen kann man durch Erzeuger definieren, oder beliebige Gruppen durch Präsentationen (Erzeuger mit Relationen). Gebräuchliche Gruppen kann man einfacher direkt „anfassen“. Ich nehme einen ↗Forumbeitrag zu Normalteilern von S4↗ zum Anlaß, nebst Blick wie dort auf die Automorphismengruppe der „Kleinschen“:

```
gap> S4:=SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap> NormalSubgroups(S4);
[ Group(()), Group([ (1,4)(2,3), (1,3)(2,4) ]),
  Group([ (2,4,3), (1,4)(2,3), (1,3)(2,4) ]),
  Sym( [ 1 .. 4 ] ) ]
gap> V4:=Subgroup(S4,[(1,2)(3,4),(1,3)(2,4)]);
Group([ (1,2)(3,4), (1,3)(2,4) ])
gap> IsomorphismGroups(V4,DirectProduct(CyclicGroup(2),
     CyclicGroup(2)));
[ f1, f2 ] -> [ f1, f2 ]
gap> IsomorphismGroups(AutomorphismGroup(V4),
     FactorGroup(S4,V4));
CompositionMapping( [ (1,2), (1,2), (1,3), () ] ->
[ f1*f2^2, f1*f2^2, f1*f2, <identity> of ... ],
  <action isomorphism> )
```

Wie wir gerade sahen, läßt sich direkt auf Isomorphie prüfen. Für nicht isomorphe Gruppen erhält man als Ausgabe

```
fail
```

und im positiven Fall wird der Isomorphismus ausgerechnet. Man kann sich also zwecks Nachweis einer Isomorphie selbige ausgeben lassen und durch Nachrechnen bestätigen. Steht man umgekehrt vor der Aufgabe, zu zeigen, daß zwei Gruppen nicht isomorph sind, hilft ein „fail“ nicht viel weiter. Man benötigt ein strukturelles Argument.

Erörtern wir das am Beispiel: ist $A_4$ isomorph zu $D_6$? Die Diedergruppe, auch dihedrale Gruppe genannt, mit der Ordnung $2n$ wird in der Literatur mal mit $D_{2n}$ und manchmal mit $D_n$ benannt, da sollte man achtgeben. In GAP wird die erstere Bezeichnungsweise verwendet.

```
gap> Size(AlternatingGroup(4));
12
gap> Size(DihedralGroup(12));
12
gap> IsomorphismGroups(AlternatingGroup(4),DihedralGroup(12));
fail
gap> List(Elements(AlternatingGroup(4)),Order);
[ 1, 3, 3, 2, 3, 3, 3, 3, 2, 3, 3, 2 ]
gap> List(Elements(DihedralGroup(12)),Order);
[ 1, 2, 6, 3, 2, 2, 2, 3, 2, 2, 6, 2 ]
```

Hierbei benutzten wir den sehr praktischen List-Befehl, womit man eine Funktion jeweils auf sämtliche Objekte einer Liste anwenden kann.

Wir bemerken sofort, daß es in $D_6$ Elemente der Ordnung $6$ gibt, in $A_4$ dagegen nicht, was wir als Beweisargument nutzen können.

Bisher erledigten wir Rechnungen, welche wir auch rasch ohne Computerhilfe durchführen konnten. Nutzen wir nun die Kapazitäten von GAP und gehen wir einmal größere Gruppen an!

Beispielsweise wurde im Forum eine ↗Frage↗ zum Nachweis gestellt, daß alle Gruppen der Ordnung 2002 einen Normalteiler vom Index $2$ haben.

Wir verwenden GAP, um das Ziel der Aufgabenstellung zu prüfen. Wir lassen uns alle Gruppen mit Elementanzahl 2002 geben:

```
gap> G2002:=AllSmallGroups(2002);;
gap> Size(G2002);
8
```

G2002 enthält jetzt eine Liste von 8 polyzyklischen Gruppen mit jeweils 4 Generatoren. Wir ermitteln die jeweiligen Normalteiler und deren Ordnungen:

```
gap> N2002:=List(G2002,NormalSubgroups);;
gap> List(N2002,i->List(i,Size));
[ [ 1, 2, 7, 14, 11, 22, 77, 154, 13, 26, 91, 182, 143,
    286, 1001, 2002 ],
  [ 1, 7, 11, 77, 13, 26, 91, 182, 143, 286, 1001, 2002 ],
  [ 1, 7, 11, 22, 77, 154, 13, 91, 143, 286, 1001, 2002 ],
  [ 1, 7, 11, 77, 13, 91, 143, 286, 1001, 2002 ],
  [ 1, 7, 14, 11, 77, 154, 13, 91, 182, 143, 1001, 2002 ],
  [ 1, 7, 11, 77, 13, 91, 182, 143, 1001, 2002 ],
  [ 1, 7, 11, 77, 154, 13, 91, 143, 1001, 2002 ],
  [ 1, 7, 11, 77, 13, 91, 143, 1001, 2002 ] ]
```

Wie man erkennt, enthält jede der Gruppen von Ordnung 2002 eine normale Untergruppe der Ordnung 1001, also mit Index 2.

In einem weiteren ↗Beitrag im Forum↗ wurde nach den Isomorphietypen einer Gruppe mit 2002 Elementen gefragt. Setzen wir unsere Untersuchungen dahingehend fort, neben den Überlegungen in ↗jenem Forumthread↗. Daß es genau 8 Typen gibt, wissen wir schon, sehen wir sie uns nun genauer an:

```
gap> List(G2002,IsAbelian);
[ true, false, false, false, false, false, false, false ]
gap> List(G2002,IsCyclic);
[ true, false, false, false, false, false, false, false ]
```

Ein erstes, nicht wirklich überraschendes Ergebnis ist ablesbar: G2002[1] ist die zyklische Gruppe $\mathbb{Z}_{2002}$.

Nehmen wir uns die 8. Gruppe vor. Wir prüfen die Erzeuger der Gruppe und ihre Relationen:

```
gap> gen:=MinimalGeneratingSet(G2002[8]);
[ f1, f2*f3*f4 ]
gap> Order(gen[1]);
2
```

```
gap> Order(gen[2]);
1001
gap> gen[2]*gen[1]*gen[2]*gen[1];
<identity> of ...
```

Die 8. Gruppe wird von 2 Elementen der Ordnungen 2 und 1001 erzeugt, letztere Relation verschafft uns die Gewißheit, daß es sich bei dieser Gruppe um die dihedrale Gruppe der Ordnung 2002 handelt.

Die verbleibenden Gruppen kann man ähnlich untersuchen, wir beschränken uns hier auf die 4. Gruppe:

```
gap> gen:=GeneratorsOfGroup(G2002[4]);
[ f1, f2, f3, f4 ]
gap> MinimalGeneratingSet(G2002[4]);
[ f1*f2, f3*f4 ]
gap> Order(gen[1]);
2
gap> Order(gen[2]);
7
gap> Order(gen[3]*gen[4]);
143
gap> gen[3]*gen[4]*gen[1]*gen[3]*gen[4];
f1
```

Der erste Erzeuger bestimmt gemeinsam mit dem dritten und vierten die $D_{143}$ (als semidirektes Produkt), der zweite Erzeuger die $\mathbb{Z}_7$, wir erhalten die 4. Gruppe als isomorph zu $D_{143} \times \mathbb{Z}_7$. Wir kontrollieren die Isomorphie mit GAP:

```
gap> IsomorphismGroups(DirectProduct(DihedralGroup(286),
     CyclicGroup(7)),G2002[4]);
[ f1*f2^6*f3^9*f4^4, f2^3*f3^3 ] -> [ f1*f2, f3*f4 ]
```

Wir haben also Isomorphie. Was man selbstverständlich auch durch eigene Überlegungen nachweisen sollte.

Umfassendere Kenntnis der GAP-Features gestattet noch effektiveres Forschen, demonstriert an dem Einzeiler

```
gap> List(G2002,StructureDescription);
[ "C2002", "C77 x D26", "C91 x D22", "C7 x D286",
  "C143 x D14", "C11 x D182", "C13 x D154", "D2002" ]
```

Es gibt enorm viele Funktionen für das Rechnen mit Gruppen: das Bestimmen von Ordnung, Konjugationsklassen, Ableitungen, Sylow-Untergruppen, Normalteilern, Automorphismen, Charakteren uvm. Weiterhin für Homomorphismen zwischen Gruppen, Operationen von Gruppen (Stabilisatoren, Bahnen, ...), direkte/semidirekte Produkte, ... haufenweise Spielzeug.

# 4 Anwendung in der Darstellungstheorie

In der Darstellungstheorie kann GAP seine Stärken gut demonstrieren. Man sehe sich einmal ↗Janas Artikelserie↗ über die Darstellungstheorie endlicher Gruppen an, wie rechenintensiv es vor allem im 3. Teil wird.

Bei der Arbeit mit Darstellungen von Gruppen sind Charaktere ein wichtiges Hilfsmittel. Für GAP gibt es Bibliotheken sowohl von vorberechneten Charaktertafeln als auch mit Algorithmen für deren Bestimmung.

In ↗Darstellungstheorie: Teil 2↗ kann man nachlesen, daß Charaktere von Darstellungen auf Konjugationsklassen der zugehörigen Gruppe konstant sind, und daß es insbesondere exakt soviele irreduzible Charaktere wie Konjugationsklassen der Gruppe gibt. Die Untersuchung von Darstellungen erfordert somit auch Wissen über die Konjugationsklassen der betreffenden Gruppe.

In ↗Darstellungstheorie: Teil 3↗ werden die Darstellungen der Symmetrischen Gruppe $S_6$ untersucht. Wieviele irreduzible Darstellungen gibt es? Fragen wir kurz GAP:

```
gap> S6:=SymmetricGroup(6);;
gap> Size(ConjugacyClasses(S6));
11
```

Wie man sich auch rasch überlegt, ist die Anzahl der Konjugationsklassen der Symmetrischen Gruppen gleich der Anzahl möglicher Zykelstrukturen, da Permutationen mit gleicher Zykelstruktur konjugiert sind, also gleich der Partitionszahl der Anzahl der permutierten Elemente. Doch die Bestimmung der Konjugationsklassen kann bei anderen Gruppen schwieriger sein.
Schauen wir mal auf die oben untersuchte 4. Gruppe der Ordnung 2002:

```
gap> Size(ConjugacyClasses(G2002[4]));
511
```

Oha, da mußte der Computer schon ein bißchen rechnen. Mit der Untersuchung des letzten Abschnitts hätten wir es auch über

```
gap> Size(ConjugacyClasses(DirectProduct(DihedralGroup(286),
     CyclicGroup(7))));
 511
```

bestimmen können. Für die Diedergruppe gibt es ja Formeln für die Anzahl der Konjugationsklassen, die auch ↗hier im Forum↗ besprochen wurden, für die Diedergruppe mit 286 Elementen erhalten wir 73. Theoretisch ist klar, daß die Zahl der Konjugationsklassen eines direkten Produkts gleich dem Produkt der entsprechenden Anzahlen der beteiligten Gruppen ist, und daß die Konjugationsklassen abelscher Gruppen nur aus je einem Element bestehen, somit muß die anfänglich gesuchte Anzahl der Konjugationsklassen der 4. Gruppe der Ordnung 2002 ja gleich $73 * 7 = 511$ sein. Aber warum sich nicht mit GAP kontrollieren.

Kehren wir zur $S_6$ zurück. Wir wollen sämtliche irreduzible Charaktere bestimmen. Im ↗3. Darstellungstheorie-Artikel↗ wird dies ausführlich demonstriert.

Wie wir in jenem Artikel sehen, brauchen wir die Ordnungen der Konjugationsklassen und der Zentralisatoren von Repräsentanten. Bestimmen wir zunächst ein Repräsentantensystem, die Größen der zugehörigen Zentralisatoren und die Klassengrößen:

```
gap> List(ConjugacyClasses(S6),i->Elements(i)[1]);
[ (), (5,6), (3,4)(5,6), (1,2)(3,4)(5,6), (4,5,6),
  (2,3)(4,5,6), (1,2,3)(4,5,6), (3,4,5,6), (1,2)(3,4,5,6),
  (2,3,4,5,6), (1,2,3,4,5,6) ]
gap> List(last,i->Size(Centralizer(S6,i)));
[ 720, 48, 16, 48, 18, 6, 18, 8, 8, 5, 6 ]
gap> List(ConjugacyClasses(S6),Size);
[ 1, 15, 45, 15, 40, 120, 40, 90, 90, 144, 120 ]
gap> List(last,i->Size(S6)/i);
[ 720, 48, 16, 48, 18, 6, 18, 8, 8, 5, 6 ]
```

Letzteres soll nur kurz zeigen, daß wir uns das Rechnen mit den Zentralisatoren auch hätten sparen können, wenn wir an den Satz von Lagrange denken, und daran, daß die Ordnung des Zentralisators gleich dem Index der zugehörigen Konjugationsklasse ist...

Vonnöten ist weiterhin theoretisches Vorwissen über lineare Charaktere, Permutationscharakter und Tensorprodukte, sowie das Rechnen mit den Orthogonalitätsrelationen. Wenn wir das alles durchgezogen haben, können wir uns

mit GAP kontrollieren:

```
gap> c:=CharacterTable(S6);
CharacterTable( Sym( [ 1 .. 6 ] ) )
gap> Display(c);
CT1
      2  4  4  4  4  1  1  1  3  3  .  1
      3  2  1  .  1  2  1  2  .  .  .  1
      5  1  .  .  .  .  .  .  .  .  1  .

         1a 2a 2b 2c 3a 6a 3b 4a 4b 5a 6b
      2P 1a 1a 1a 1a 3a 3a 3b 2b 2b 5a 3b
      3P 1a 2a 2b 2c 1a 2a 1a 4a 4b 5a 2c
      5P 1a 2a 2b 2c 3a 6a 3b 4a 4b 1a 6b

X.1    1 -1  1 -1  1 -1  1 -1  1  1 -1
X.2    5 -3  1  1  2  .  -1 -1 -1  .  1
X.3    9 -3  1 -3  .  .  .  1  1 -1  .
X.4    5 -1  1  3 -1 -1  2  1 -1  .  .
X.5   10 -2 -2  2  1  1  1  .  .  . -1
X.6   16  .  .  . -2  . -2  .  .  1  .
X.7    5  1  1 -3 -1  1  2 -1 -1  .  .
X.8   10  2 -2 -2  1 -1  1  .  .  .  1
X.9    9  3  1  3  .  .  . -1  1 -1  .
X.10   5  3  1 -1  2  . -1  1 -1  . -1
X.11   1  1  1  1  1  1  1  1  1  1  1
```

zeigt uns die gesamte Charaktertafel an. Auf irreduzible Charactere können wir zugreifen mit Irr:

```
gap> Irr(c)[2];
Character( CharacterTable( Sym( [ 1 .. 6 ] ) ),
[ 5, -3, 1, 1, 2, 0, -1, -1, -1, 0, 1 ] )
```

Bestimmen wir einmal manuell den Permutationscharakter und konstruieren aus ihm einen irreduziblen:

```
gap> p:=Character(c,[6,4,2,0,3,1,0,2,0,1,0]);
Character( CharacterTable( Sym( [ 1 .. 6 ] ) ),
[ 6, 4, 2, 0, 3, 1, 0, 2, 0, 1, 0 ] )
```

```
gap> ScalarProduct(p,p);
    2
    gap> q:=p-TrivialCharacter(S6);
    VirtualCharacter( CharacterTable( Sym( [ 1 .. 6 ] ) ),
    [ 5, 3, 1, -1, 2, 0, -1, 1, -1, 0, -1 ] )
    gap> ScalarProduct(q,q);
    1
```

Nach Irreduzibilitätskriterium ist also p reduzibel, q irreduzibel und entspricht X.10.

Rechnen mit Charakteren erfordert oft Summierungen von Ausdrücken über Gruppenelemente (Orthogonalitätsrelationen), was GAP durch u.a. Listenoperationen erleichtert, wie man z.B. bei der obigen Berechnung des skalaren Produkts von Charakteren sehen kann:

$$\langle p, q\rangle = \frac{1}{|G|}\sum_{g\in G} p(g)\overline{q(g)},$$

ScalarProduct(p,q) entspricht hier abkürzend

```
gap> Sum([1 .. Size(ConjugacyClasses(c))],i->SizesConjugacy
     Classes(c)[i]*p[i]*ComplexConjugate(q[i]))/Size(S6);
1
```

Im Dokument: ↗Character Tables in GAP↗ wird erläutert, wie man Charaktertafeln durch Charakter-Arithmetik unter Hilfe von GAP berechnen kann, wenn sie nicht bereits in der GAP-Bibliothek nachschlagbar wären.

# 5 Schlußwort

GAP bietet noch viel mehr Möglichkeiten, als hier auch nur angedeutet werden konnten. Es läßt sich mit weiteren Strukturen wie Ringen, Körpern, Vektorräumen, Moduln, ... arbeiten, zudem gibt es viele spezielle Lösungen und Erweiterungen. Ich empfehle das Stöbern auf der ↗Homepage der GAP-Group↗.

Dieser Artikel sollte anhand eines kleinen Einblicks in die GAP-Anwendung aufzeigen, daß man intuitiv und ohne großen Lernaufwand bereits Gewinn aus Algebra-Software schöpfen kann. Dokumentation nachlesen muß man anfangs trotzdem, doch nicht unbedingt gleich Programmieren lernen.

Ich würde mich freuen, wenn mancher Leser hierdurch zur Nutzung der Möglichkeiten der Computeralgebra angeregt wird, speziell den Vorteil freier akademischer Software erkennt.

Die Mathematik mag heute so fortgeschritten und komplex oder gar schwierig erscheinen, doch in den letzten Jahren wurden uns Möglichkeiten in die Hand gegeben, welche Mathematiker früher nicht besaßen: wie Computeralgebra-Systeme, weltweite Recherchemöglichkeiten durch Internet und Suchmaschinen, Erleichterung von Publikationen durch LaTeX, effiziente Kommunikationsmöglichkeiten und zugängliche Wissens-Archive, wie auch ↗Matroids Matheplanet↗ demonstriert.

↗Stefan$_K$↗

# 6 Referenzen

**↗The GAP Group↗, GAP - Groups, Algorithms, Programming**
(↗http://www.gap-system.org↗) Homepage der GAP-Group, enthält: ↗Manuals↗, ↗Tutorials↗, ↗Lehrmaterial↗ und vieles mehr

**↗Character Tables in GAP↗**
Begleitmaterial zu einem Kurs in Darstellungstheorie von Alexander Hulpke

**↗The GAP Character Table Library↗**
Informationen, Dokumentation, Download, Anwendungsbeispiele. Von Thomas Breuer

**↗Stephan Rosebrock: „Geometrische Gruppentheorie“↗**
Ein Einstieg mit dem Computer. Vieweg, 2004

---

**Gruppentheorie auf dem Matheplaneten:**
↗Artikel↗, ↗Links↗, ↗Buchbesprechungen↗, ↗Forum↗

**Darstellungstheorie auf dem Matheplaneten:**
↗Artikel↗, ↗Links↗, ↗Buchbesprechungen↗

---

**↗http://gruppentheorie.de/mit/GAP/↗**
Dieser Artikel im pdf-Format

---

**↗Matroids Matheplanet↗**
↗http://matheplanet.com↗