

*restart;*

## ▼ Aufgabe 1: Matrizen

*with(LinearAlgebra) :*

$$A := \begin{bmatrix} 1 & 4 & 7 \\ 8 & 2 & 5 \\ 6 & 9 & 3 \end{bmatrix};$$

$$\begin{bmatrix} 1 & 4 & 7 \\ 8 & 2 & 5 \\ 6 & 9 & 3 \end{bmatrix} \quad (1.1)$$

*Determinant(A);*

$$405 \quad (1.2)$$

*A<sup>-1</sup>;*

$$\begin{bmatrix} -\frac{13}{135} & \frac{17}{135} & \frac{2}{135} \\ \frac{2}{135} & -\frac{13}{135} & \frac{17}{135} \\ \frac{4}{27} & \frac{1}{27} & -\frac{2}{27} \end{bmatrix} \quad (1.3)$$

*MatrixInverse(A);*

$$\begin{bmatrix} -\frac{13}{135} & \frac{17}{135} & \frac{2}{135} \\ \frac{2}{135} & -\frac{13}{135} & \frac{17}{135} \\ \frac{4}{27} & \frac{1}{27} & -\frac{2}{27} \end{bmatrix} \quad (1.4)$$

$$B := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix};$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad (1.5)$$

*Determinant(B);*

$$0 \quad (1.6)$$

Da die Determinante 0 ist, ist die Matrix nicht invertierbar. Maple kommentiert dies so:

*MatrixInverse(B);*

Error, (in MatrixInverse) singular matrix

restart;

## ▼ Aufgabe 2: Prozeduren

*with(numtheory) :*

*myproc := proc(n)*

**local** *i, sum;*

*sum := 0;*

**for** *i from 1 to n do*

**if** *nops(factorset(i)) ≥ 4 then*

*sum := sum + i;*

**end if;**

**end do;**

**return** *sum;*

**end proc;**

**proc(n)**

**local** *i, sum;*

*sum := 0;*

**for** *i to n do*

**if**  $4 \leq \text{nops}(\text{numtheory:-factorset}(i))$  **then** *sum := sum + i* **end if**

**end do;**

**return** *sum*

**end proc**

*myproc(4321);*

(2.1)

724245

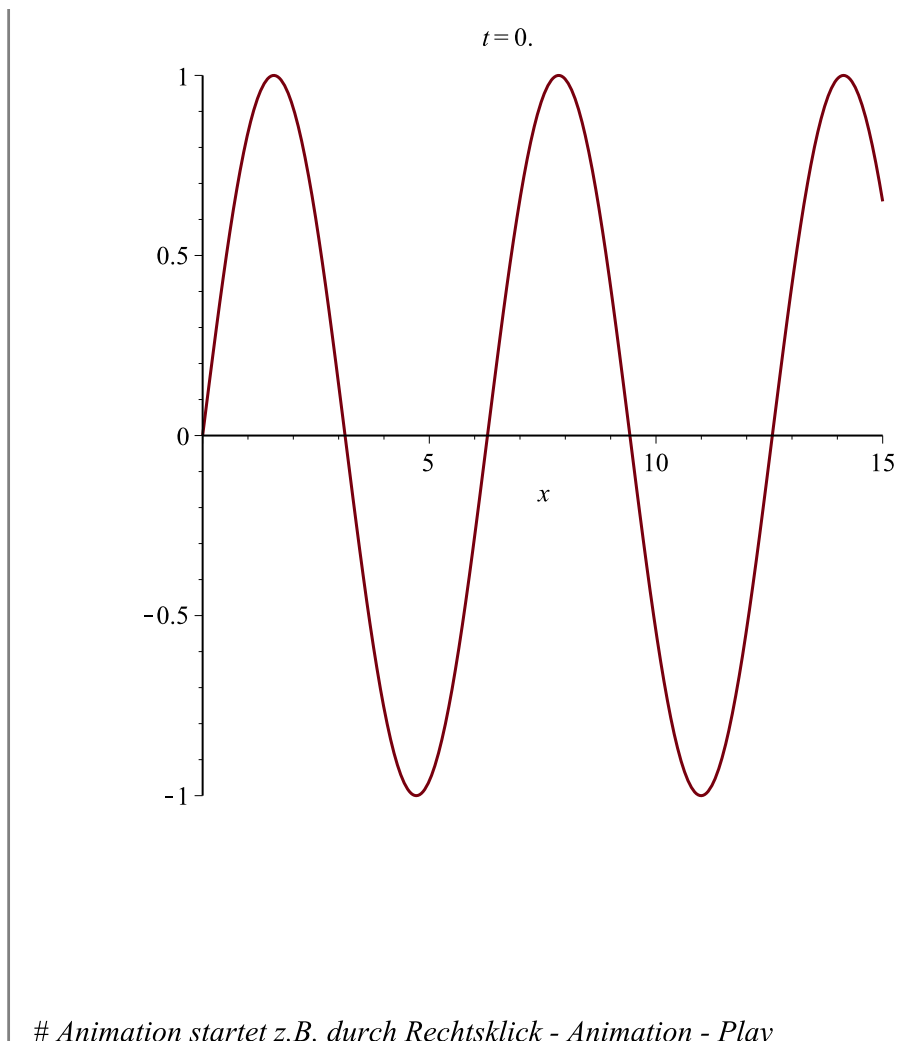
(2.2)

restart;

## ▼ Aufgabe 3: Animation

*with(plots) :*

*animate(plot, [sin(x + t), x = 0 .. 15], t = 0 .. 4 \* pi);*



# Animation startet z.B. durch Rechtsklick - Animation - Play  
restart;

## ▼ Aufgabe 4: Fibonacci-Zahlen

▼ a)

```
fibrek := proc(n :: nonnegint)
  if n ≤ 1 then return n; end if;
  return fibrek(n - 1) + fibrek(n - 2);
end proc;
```

```
proc(n::nonnegint)
```

(4.1.1)

```
  if n ≤ 1 then return n end if; return fibrek(n - 1) + fibrek(n - 2)
```

```
end proc
```

```
fibrek(30);
```

832040

(4.1.2)

**b)**

```

fibit := proc(n :: nonnegint)
  local f1, f2, tmp, i;
  if n ≤ 1 then return n; end if;
  f1 := 0;
  f2 := 1;
  for i from 2 to n do
    tmp := f1 + f2;
    f1 := f2;
    f2 := tmp;
  end do;
  return f2;
end proc;
proc(n::nonnegint) (4.2.1)
  local f1, f2, tmp, i;
  if n ≤ 1 then return n end if;
  f1 := 0;
  f2 := 1;
  for i from 2 to n do tmp := f1 + f2; f1 := f2; f2 := tmp end do;
  return f2

```

**end proc**

```

fibit(100000)
2597406934722172416615503402127591541488048538651769658472477070395253454\
351127368626555677283671674[...20699 digits...]
9259130435572321635660895603514383883939018953166274355609970015699780
289236362349895374653428746875

```

**c)**

```

fibmat := proc(n :: nonnegint)
  local F;
  
$$F := \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1};$$

  return F[1, 1];
end proc;
proc(n::nonnegint) (4.3.1)
  local F;
  F := Matrix(2, 2, {(1, 1) = 1, (1, 2) = 1, (2, 1) = 1, (2, 2) = 0})^(n - 1);
  return F[1, 1]

```

**end proc**

```

fibmat(10000000); (4.3.2)
...Integer too large for display...

```

**d)**

```
fibmb := proc(n :: nonnegint)
  forget(simplify);
  # Simplify merkt sich manchmal Werte von vorher. Diese sollten für die Laufzeitmessung
  entfernt werden.
  return simplify(  $\frac{1}{\sqrt{5}} \left( \left( \frac{1 + \sqrt{5}}{2} \right)^n - \left( \frac{1 - \sqrt{5}}{2} \right)^n \right)$  );
```

**end proc;****proc(n::nonnegint)****(4.4.1)**

forget(simplify);

return simplify(((1/2 + 1/2 \* sqrt(5))^n - (1/2 - 1/2 \* sqrt(5))^n) / sqrt(5))

**end proc**

fibmb(25000);

```
2195438355517303012780791914841720922849015222302155773145178112730662303\ (4.4.2)
998294388326731046697794553205344778951126287924932107427759864934238\
982127116700031702303758195833103609129478390489446109048353175364789\
602348407758024937078368622211028096142835259244470859742950247959173\
097297162352647751262119867748993044912978879730256945445887101598561\
852896240814094692721968792533798209860951746361393836607992705067710\
042620749500787001244417521987828274486196345409730699935248646448002\
099486172016340837622177625412488128183565823839681117147554784050557\
728049729076469785258091835237832245198138984091205393427086714923971\
764032960455754157636842553889454001433914636754491428389597134631113\
060743894372627869820078095826993694733293044620322404164708462810601\
696905333714162025084123290258033847470987363286392269159470707335216\
234289173174621091516132795704202799168694074538860973938568721305434\
399866802769125320916703397809559673570359525801786556002107827441453\
986276775162513506228590570474327975563079481094603466177112962126117\
440542434702915102799877664360166812897585744093962656052912423886113\
100461230142784146023711777420526315229638008014989857335491622206553\
432801243509562482038767520086997146161082906654202446422946113709406\
918144698019603262487504401240322964205220021141356622967635738223415\
333791990897008621304124750742321036677825587496906743896332364366496\
138093122610395787935469738420787025802872527884297599935502454201570\
481621997508955573315041880985203796467537567522299713622634363391966\
951781139362015429474824593874336723409381728045838357307674036232976\
172964849804217509930088307381163714216347319672854286456833768480422\
693677608036778208888135806796161585831624637343362732327552974747569\
724035372797962506356732207678790344000415753644840243085840693705241\
496451702203474493711294782696421566073432559966567557353875841052667\
413155101970619194483251743981974098614050625487854359625151165141527\
022212109918027829575378201609757302583895364211351191963400047276378\
533876526444961997217464476816410341144513254799193440795873864549893\
```

106400983697904854625181640857277908280593079260922362597888507736945\  
158693268017681586045153250828445893153468310287805833119925409176384\  
779999179792543986768911251349899390459042481716624531287108224344630\  
388197851287107167310452300346365099746836330994730194732786455171203\  
553487997916567305621096005424046632826716255667276502869412549646415\  
087607598856062761032427516753856297480620846982922680440240789492674\  
800445072429006460252405902849978143949425879220662652649726478991348\  
849854976304998029625242180637878759057422604712651884180250536037794\  
686571716606891131924474199080777930909758098280349365022887770898121\  
294753703481574971955358741672207272659743385198090569815228970245669\  
408793977060275665914868800969209970442233031577010661825049476736044\  
769511024515279382249991897708260566735104329753279423017150422323200\  
261866375280818749856127144617214292070663053115051282686314062201595\  
686102285790452946433691910492502818098570070153735144337300871484364\  
989278622751621727080556237093670929054410334777645340214480030955613\  
169949108956207793714391764043504478324684917048583211198571305351984\  
858555731389992350443489458909934833942635089060913400909364442872524\  
135291934193160585778048761174844361381092189738506769594370757275592\  
131123138842797434774319441540922683325975401508191959577960053363977\  
407385689775204437763705345505871292901513722292518398645659537543312\  
281182869158706110315382183568215769477878679942998963081136045667638\  
060830058727162470289959669677108108863765886742011827777581089029856\  
376004266672070861254545663470736499793929043099117149825960207676425\  
109713593539624486901362762428578241313625026256976310596644786430900\  
805963201447064965159771754255927850256208874776182880436842315814581\  
195562340152242738932326124075996925869465986679126689538451357896574\  
430422403711592312334013691251782657924188862255729419086564383995069\  
175344258078907672485813282267142001466216006029276047348238306148848\  
843706894507173399089691520187261450663342561130679145645767176324776\  
708996884391782767088219127017224191468212141892708210269459532105888\  
561207362657330038050947566167536307819378822094601515004814199498936\  
258842796604552254255354840694472110919968001610782651306076273608351\  
020303008762173540012271277316120641743739717009907499823530931284994\  
068770213296645065744771565606171201417636597504870788348501866718868\  
415899916828323425083040065614906939886267325245707777527768770068072\  
158655280053325719471720693717268927702119474727043191024388701482373\  
934201540393905573578152388165175003644951606293684230307204786518611\  
840108390350444254434049835408764985099935050603560609186566017381404\  
478493856109286561840266560776111588440944204249355145580824513414497\  
979461286816997500031522311225531430018197880147344268828233277914619\  
455101841152334171285111860680623717698430410850791543363963511516997\  
722924613542669609128047742355009178297021340699996115304962265387488\

```
661470671539218492570038835218943289376111409469663775103390510034337\
398717853672753636270306303086899557631317919804843678735586795606845\
593106221969533896433396191345178217444964347900106990325352689919643\
0613467544838316226807341377036587231777596875
```

## Laufzeitvergleich

**$n = 32$**

$$t := \text{time}() : \text{fibrek}(32) : \text{time}() - t; \quad 11.560 \quad (4.5.1.1)$$

$$t := \text{time}() : \text{fibit}(32) : \text{time}() - t; \quad 0. \quad (4.5.1.2)$$

$$t := \text{time}() : \text{fibmat}(32) : \text{time}() - t; \quad 0. \quad (4.5.1.3)$$

$$t := \text{time}() : \text{fibmb}(32) : \text{time}() - t; \quad 0.016 \quad (4.5.1.4)$$

**$n = 25.000$**

$$t := \text{time}() : \text{fibit}(25000) : \text{time}() - t; \quad 0.140 \quad (4.5.2.1)$$

$$t := \text{time}() : \text{fibmat}(25000) : \text{time}() - t; \quad 0. \quad (4.5.2.2)$$

$$t := \text{time}() : \text{fibmb}(25000) : \text{time}() - t; \quad 30.420 \quad (4.5.2.3)$$

**$n = 250.000$**

$$t := \text{time}() : \text{fibit}(250000) : \text{time}() - t; \quad 11.934$$

$$t := \text{time}() : \text{fibmat}(250000) : \text{time}() - t; \quad 0.031 \quad (4.5.3.2)$$

**$n=100.000.000$**

$$t := \text{time}() : \text{fibmat}(100000000) : \text{time}() - t; \quad 42.885 \quad (4.5.4.1)$$

*restart;*

## Aufgabe 5: Ikosaeder

with(*combinat*) :  
 with(*ArrayTools*) :  
 with(*LinearAlgebra*) :  
 with(*geom3d*) :

$$\tau := \frac{1 + \sqrt{5}}{2} :$$

Tetraeder in homogenen Koordinaten als Zeilenvektoren einer Matrix.

$C := \langle \langle 0|\tau|1|1 \rangle, \langle 0|\tau|-1|1 \rangle, \langle 0|-\tau|1|1 \rangle, \langle 0|-\tau|-1|1 \rangle, \langle 1|0|\tau|1 \rangle, \langle 1|0|-\tau|1 \rangle, \langle -1|0|\tau|1 \rangle, \langle -1|0|-\tau|1 \rangle, \langle \tau|1|0|1 \rangle, \langle \tau|-1|0|1 \rangle, \langle -\tau|1|0|1 \rangle, \langle -\tau|-1|0|1 \rangle \rangle :$

## Facettenbestimmung

Facettenbestimmung mittels Chirotop (Stoff z.B. von LA2 bei J. Bokowski, SS2007, vgl. Übung 4, Aufgabe H7)

*getFaces* := **proc**(*A*)

**local** *S*,  $\chi$ , *F*, *v*, *zero*, *pl*, *mi*, *i*, *j*, *m* :

*S* := *choose*(*Size*(*A*, 1), 4) :

$\chi := [\text{seq}(\text{signum}(\text{Determinant}(A[S[i]])), i = 1 .. \text{binomial}(\text{Size}(A, 1), 4))]$  :

*F* := *NULL* :

**for** *i* **in** *choose*(*Size*(*A*, 1), 3) **do**

*zero* := *false* :

*pl* := *true* :

*mi* := *true* :

**for** *j* **from** 1 **to** *Size*( $\chi$ , 2) **do**

**if** {*op*(*i*)} **subset** {*op*(*S*[*j*])} **then**

*v* := *op*( {*op*(*S*[*j*]) } \ {*op*(*i*) } ) :

**if** *S*[*j*, 1] = *v* **or** *S*[*j*, 3] = *v* **then** *m* := -1 : **else** *m* := 1 : **fi**:

**if** *m* ·  $\chi$ [*j*] > 0 **then** *pl* := *false* :

**elif** *m* ·  $\chi$ [*j*] < 0 **then** *mi* := *false* :

**else** *zero* := *true* :

**fi**:

**fi**:

**od**:

**if** *pl* **or** *mi* **then**

*F* := *F*, *i* :

**if** *zero*

**then** *print*("Mehr als 4 Punkte in Facetten-Ebene gefunden?!? Das Ergebnis wird nicht stimmen...") : **fi**:

**fi**:

**od**:

**return** [*F*] :

**end proc**:

## Tetraederzerlegung



```

getTetras :=proc(A, name)
local F, T, P, i, j, t :
F := getFaces(A) :
point(p0, 0, 0, 0) :
P := NULL :
for i from 1 to Size(A, 1) do
  P := P, point(cat('p', name, i), A[i, 1], A[i, 2], A[i, 3]) :
od:
P := [P] :
j := 0 :
T := NULL :
for i in F do
  j := j + 1 :
  T := T, tetrahedron(cat('t', name, j), [p0, op(P[i])]) :
od:
return [T] :
end proc:

```

▼ a)

```
T := getTetras(C,'normal') :
```

(5.3.1)

```
V := 0 :
for t in T do
  V := V + volume(t) :
od:
simplify(V);
```

$$10 + \frac{10}{3} \sqrt{5}$$

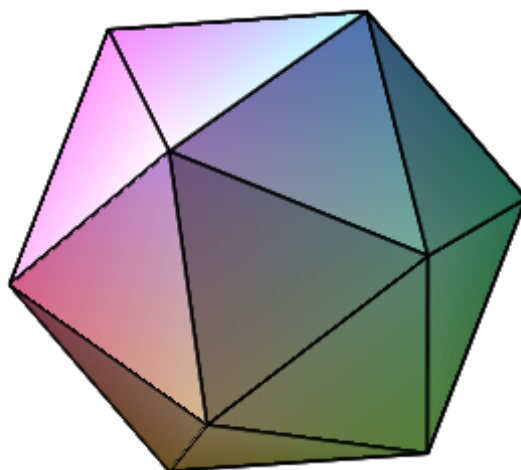
(5.3.2)

```
evalf(V);
```

$$17.45355992$$

(5.3.3)

```
draw(T);
```



▼ **b) bis d)**

$E := \langle \langle 0|0.1|0|0 \rangle, \langle 0.03|0.1|-0.07|0 \rangle, \langle 0|0|-0.1|0 \rangle, \langle 0|0|0|0 \rangle, \langle 0|0|0|0 \rangle, \langle 0.13|0.07|0|0 \rangle, \langle 0|0|0.23|0 \rangle, \langle -0.17|0|0|0 \rangle, \langle 0|0|0|0 \rangle, \langle -0.1|0.1|0|0 \rangle, \langle 0.1|-0.03|0|0 \rangle, \langle 0|0|0|0 \rangle \rangle :$

$T := \text{getTetras}(C + E, \text{'pert'}) :$

$V := 0 :$

**for**  $t$  **in**  $T$  **do**

$V := V + \text{volume}(t) :$

**od:**

$\text{evalf}(V);$

18.07014099

(5.4.1)

$\text{draw}(T);$

