

Projekt P5 zu Mathe am Computer, WS 2002/03

"Eine ganz besondere Spirale"

von Oleg Gudi und Sebastian Precker

5. Februar 2003

```
> restart:
with(linalg):
with(LinearAlgebra):
with(plots):
with(plottools):
Warning, the protected names norm and trace have been redefined and unprotected
Warning, the assigned name GramSchmidt now has a global binding
Warning, the name changecoords has been redefined
Warning, the name arrow has been redefined
```

(i) Bestimmen der Rekursionsformel.

Die ersten zwei Werte werden vordefiniert:

```
a[0]:=0;
```

```
a[1]:=1;
```

Die n-te Kantenlänge berechnet sich dann wie folgt:

```
a[n]:=a[n-1]+a[n-2];
```

$$a_0 := 0$$

$$a_1 := 1$$

$$a_n := a_{n-1} + a_{n-2}$$

(ii) Prozedur zur Berechnung der n-ten Kantenlänge:

```
a:=proc(n)
local b,i:
b[0]:=0:
b[1]:=1:
for i from 2 to n do
b[i]:=b[i-1]+b[i-2];
end do:
return b[n]:
end proc:
```

```
> a(20);
```

6765

Andere Möglichkeit:

```
aa:=proc(n)
local b,i:
b[0]:=0:
b[1]:=1:
if n>1 then;
b[n]:=procname(n-1)+procname(n-2);
end if:
return b[n]:
end proc:
> aa(20);
```

Nun wollen wir prüfen, welche der beiden Möglichkeiten schneller ist:

```
> st:=time():
a(2000):
time()-st;
st:=time():
aa(20):
time()-st;
```

0.040

1.862

Wir sehen, dass die erste Variante viel schneller ist. Die zweite Prozedur ist zudem für Zahlen größer 25 extrem langsam!!

Und jetzt noch einige Fibonacci Zahlen:

```
> for i from 1 to 15 do
a(i);
od;
```

1

1

2

3

5

8

13

21

34

55

89

144

233

377

610

(iii) Verhältnis $a(n+1)/a(n)$ für wachsende n :

```
Verhaeltnis:=proc(n);
evalf(a(n+1)/a(n));
end proc;
```

```
> Verhaeltnis(100);
```

1.618033989

(iv) Bestimmen der Spirale:

```
Spirale:=proc(n)
local Y,X,y,x,j,i,k,L;
```

Erster Halbkreis mit Radius r und Nullpunkt $(0,0)$

```
y[1]:=(r,x)->sqrt(r^2-(x)^2):
```

Hier berechnen wir die einzelnen Viertelkreise (insgesamt $n+1$, weil wir für weitere Berechnungen eine zusätzliche Angabe brauchen):

```
for i from 2 to n+1 do
Verschiebung des Nullpunktes auf der x-Achse
X:=add(a(2*j)*(-1)^(j-1),j=1..(i-1)/2): #
X=a(2)-a(4)+a(6)-a(8)+.....a(n)
Verschiebung des Nullpunktes auf der y-Achse
Y:=-add(a(2*j-1)*(-1)^(j-1),j=1..i/2): #
Y=-a(1)+a(3)-a(5)+a(7)-.....a(n)
```

Es müssen jeweils zwei Viertelkreise von oben und unten gezeichnet werden. Dies wird über ein Vorzeichen bestimmt, was hiermit berechnet werden soll: Zufällig ist das Vorzeichen immer negativ, wenn der Nullpunkt links vom Startpunkt liegt !

```
if X>0 then k:=-1;
else k:=1;
end if:
```

i -ter Halbkreis mit Radius r und um (X,Y) verschobenen Nullpunkt

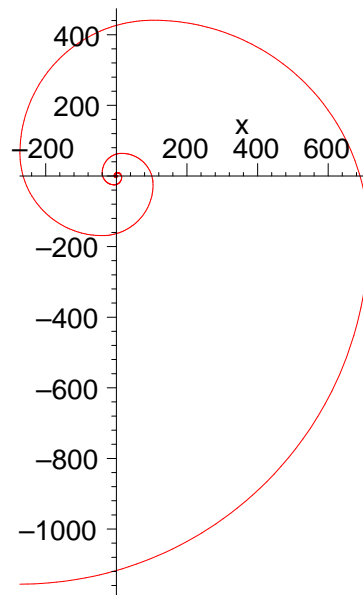
```
y[i]:=k*y[1](r,x+X)+Y;
y[i]:=unapply(y[i],(r,x));
end do:
```

Hier wird der Intervall $x=x(\text{unten})..x(\text{oben})$ berechnet, in dem der jeweilige Halbkreis gezeichnet werden soll:

```
solve(y[0](a(1),x)=y[1](a(2),x)):= -1: # der Anfang wird einfach
definiert
for j from 0 to n-1 do
x[j+1]=Schnittpunkt von y[j] mit y[j+1] ..... Schnittpunkt von y[j+1] mit y[j+2]
x[j+1]:=solve(y[j](a(j+1),x)=y[j+1](a(j+2),x))..solve(y[j+1](a(j+2)
),x)=y[j+2](a(j+3),x));
end do:
```

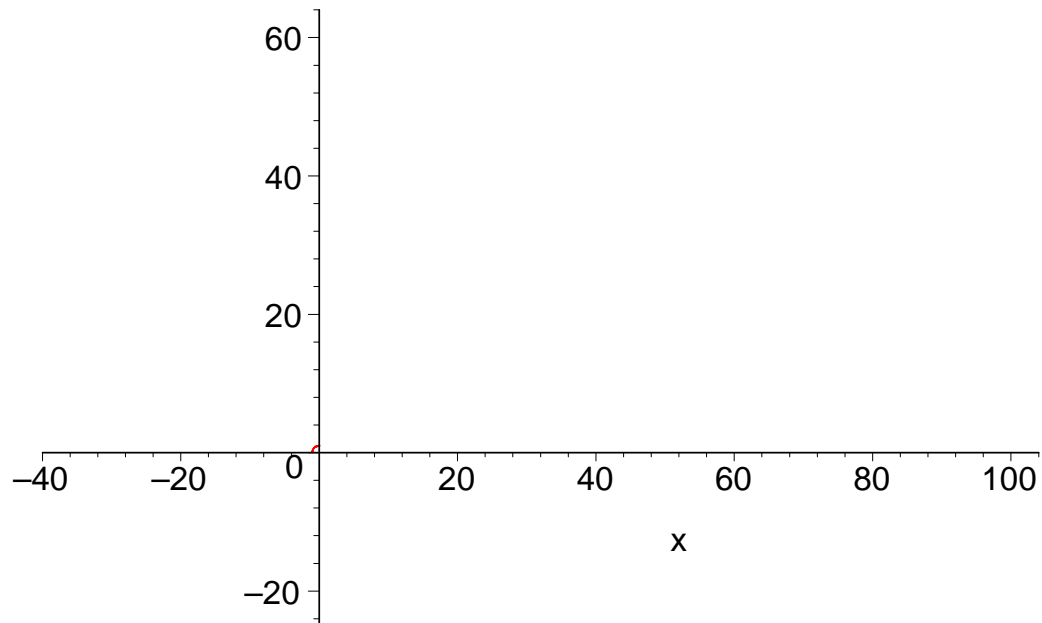
Plotten einzelner Viertelkreise und Darstellung in einem gemeinsamen Diagramm

```
for i from 1 to n do
L[i]:=plot(y[i](a(i+1),x),x=x[i],scaling=constrained);
end do:
L:= [seq(L[i],i=1..n)]:
display(L);
end proc:
> Spirale(15);
```



Und nun noch als Film:

```
> M:=NULL:  
  for i from 1 to 10 do  
    M:=M,Spirale(i):  
  end do:  
display([M], insequence=true,scaling=constrained);
```



(v) Als Nächstes schreiben wir die obige Rekursionsformel um:

Zur Erinnerung:

> $a[n]=a[n-1]+a[n-2];$

$$a_n = a_{n-1} + a_{n-2}$$

Wir definieren einen Vektor:

$b[n] := \langle a[n-1], a[n] \rangle;$

$$b_n := \begin{bmatrix} a_{n-1} \\ a_n \end{bmatrix}$$

Dann ist der nächste Vektor:

> $b[n+1] := \langle a[n], a[n+1] \rangle;$

$$b_{n+1} := \begin{bmatrix} a_n \\ a_{n+1} \end{bmatrix}$$

[oder

> `b[n+1]:=<a[n],a[n-1]+a[n]>;`

$$b_{n+1} := \begin{bmatrix} a_n \\ a_{n-1} + a_n \end{bmatrix}$$

[Dann sehen wir, dass folgendes gilt:

> `b[n+1]=A.b[n];`

$$\begin{bmatrix} a_n \\ a_{n-1} + a_n \end{bmatrix} = A \cdot \begin{bmatrix} a_{n-1} \\ a_n \end{bmatrix}$$

[mit:

> `A:=<<0,1>|<1,1>>;`

$$A := \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

[Probe:

> `b[n+1]=A.b[n];`

$$\begin{bmatrix} a_n \\ a_{n-1} + a_n \end{bmatrix} = \begin{bmatrix} a_n \\ a_{n-1} + a_n \end{bmatrix}$$

[**Verfeinerungen:**

> `J,Q:=JordanForm(A,output=['J','Q']);`

$$J, Q := \begin{bmatrix} \frac{1}{2} + \frac{\sqrt{5}}{2} & 0 \\ 0 & \frac{1}{2} - \frac{\sqrt{5}}{2} \end{bmatrix}, \begin{bmatrix} \frac{(-1+\sqrt{5})\sqrt{5}}{10} & \frac{(1+\sqrt{5})\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & -\frac{\sqrt{5}}{5} \end{bmatrix}$$

[> `#?JordanForm`

Aus der Hilfedatei erfahren wir, dass Die Hauptdiagonale von J die Eigenwerte von A enthält:

> `Eigenvalues(A);`

$$\begin{bmatrix} \frac{1}{2} + \frac{\sqrt{5}}{2} \\ \frac{1}{2} - \frac{\sqrt{5}}{2} \end{bmatrix}$$

> `evalf(Eigenvalues(A));`

$$\begin{bmatrix} 1.618033988 \\ -0.6180339880 \end{bmatrix}$$

> `Verhaeltnis(100);`

$$1.618033989$$

[Ausserdem erfahren wir, dass es gilt:

> `J=evala(Q^(-1) . A . Q);`

$$\begin{bmatrix} \frac{1}{2} + \frac{\sqrt{5}}{2} & 0 \\ 0 & \frac{1}{2} - \frac{\sqrt{5}}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{\sqrt{5}}{2} & 0 \\ 0 & \frac{1}{2} - \frac{\sqrt{5}}{2} \end{bmatrix}$$

Die obere Gleichung stellen wir nach A um, und erhalten:

$$A = \text{evala}(Q \cdot J \cdot Q^{-1});$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

und folglich muss gelten:

$$> A^n = (Q \cdot J \cdot Q^{-1})^n;$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n =$$

$$\left[\frac{\left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)(-1 + \sqrt{5})\sqrt{5}}{10} + \frac{\left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)(1 + \sqrt{5})\sqrt{5}}{10}, \right.$$

$$\left. \frac{\left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^2(-1 + \sqrt{5})\sqrt{5}}{10} + \frac{\left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^2(1 + \sqrt{5})\sqrt{5}}{10} \right]$$

$$\left[\frac{\left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)\sqrt{5}}{5} - \frac{\left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)\sqrt{5}}{5}, \frac{\left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^2\sqrt{5}}{5} - \frac{\left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^2\sqrt{5}}{5} \right]^n$$

oder

$$> A^n = Q \cdot J^n \cdot Q^{-1};$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n = \begin{bmatrix} \frac{(-1 + \sqrt{5})\sqrt{5}}{10} & \frac{(1 + \sqrt{5})\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & -\frac{\sqrt{5}}{5} \end{bmatrix} \cdot \left(\begin{bmatrix} \frac{1}{2} + \frac{\sqrt{5}}{2} & 0 \\ 0 & \frac{1}{2} - \frac{\sqrt{5}}{2} \end{bmatrix} \right)^n \cdot \begin{bmatrix} 1 & \frac{1}{2} + \frac{\sqrt{5}}{2} \\ 1 & \frac{1}{2} - \frac{\sqrt{5}}{2} \end{bmatrix}$$

Zur Berechnung von J^n :

$$> J^2;$$

$$J^3;$$

$$J^{10};$$

$$\begin{bmatrix} \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^2 & 0 \\ 0 & \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^2 \end{bmatrix}$$

$$\begin{bmatrix} \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^3 & 0 \\ 0 & \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^3 \end{bmatrix}$$

$$\begin{bmatrix} \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^{10} & 0 \\ 0 & \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^{10} \end{bmatrix}$$

Für $J^n = J_n$ gilt entsprechend:

> $J_n := \langle\langle J[1,1]^n, 0 \rangle \mid \langle 0, J[2,2]^n \rangle \rangle;$

$$J_n := \begin{bmatrix} \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^n & 0 \\ 0 & \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^n \end{bmatrix}$$

$A^n = A^n$

> $A_n := Q \cdot J_n \cdot Q^{-1};$

$A_n :=$

$$\left[\frac{(-1 + \sqrt{5})\sqrt{5} \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^n}{10} + \frac{(1 + \sqrt{5})\sqrt{5} \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^n}{10}, \right.$$

$$\left. \frac{(-1 + \sqrt{5})\sqrt{5} \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^n \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)}{10} + \frac{(1 + \sqrt{5})\sqrt{5} \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^n \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)}{10} \right]$$

$$\left[\frac{\sqrt{5} \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^n}{5} - \frac{\sqrt{5} \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^n}{5}, \frac{\sqrt{5} \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^n \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)}{5} - \frac{\sqrt{5} \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^n \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)}{5} \right]$$

> $\text{evalf}(\%);$

$[0.2763932021 \cdot 1.618033988^n + 0.7236067975 \cdot (-0.6180339880)^n,$
 $0.4472135950 \cdot 1.618033988^n - 0.4472135948 \cdot (-0.6180339880)^n]$
 $[0.4472135954 \cdot 1.618033988^n - 0.4472135954 \cdot (-0.6180339880)^n,$
 $0.7236067972 \cdot 1.618033988^n + 0.2763932018 \cdot (-0.6180339880)^n]$

Wir betrachten einige Potenzen der Matrix A und vergleichen sie mit den jeweiligen Fibonacci-Zahlen:

> $A^1, \quad a(1);$

$A^2, \quad a(2);$

```
A^3, a(3);
A^5, a(5);
A^10, a(10);
```

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, 1$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}, 1$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, 2$$

$$\begin{bmatrix} 3 & 5 \\ 5 & 8 \end{bmatrix}, 5$$

$$\begin{bmatrix} 34 & 55 \\ 55 & 89 \end{bmatrix}, 55$$

Wir erkennen, dass die Nebendiagonale die von uns gesuchte Zahl enthält:
Also gilt für die n-te Fibonacci-Zahl: $F(n)$ =linke untere Stelle der Matrix A^n =rechte obere Stelle der Matrix A^n

Wir schreiben:

```
> F[n]=simplify(An[1,2]);
F[n]:=simplify(An[2,1]);
```

$$F(n) = \frac{\sqrt{5} \left(\frac{1}{2} + \frac{\sqrt{5}}{2} \right)^n}{5} - \frac{\sqrt{5} \left(\frac{1}{2} - \frac{\sqrt{5}}{2} \right)^n}{5}$$

$$F(n) := \frac{\sqrt{5} \left(\frac{1}{2} + \frac{\sqrt{5}}{2} \right)^n}{5} - \frac{\sqrt{5} \left(\frac{1}{2} - \frac{\sqrt{5}}{2} \right)^n}{5}$$

Nun packen wir alles in eine Prozedur:

```
> a2:=proc(n)
local F;
F :=
1/5*5^(1/2)*(1/2+1/2*5^(1/2))^n-1/5*5^(1/2)*(1/2-1/2*5
^(1/2))^n;
return evalf(F);
end proc;
```

```
> a2(9);
```

33.99999985

```
> st:=time();
a(2000);
time()-st;
```

```
st:=time():
a2(20000000):
time()-st;
```

0.040

0.

Wir können aber auch die ganze obere Matrixrechnung in eine Prozedur packen:

```
a3:=proc(n)
local JJ,QQ,An,A;
A:=<<0,1>|<1,1>>;
JJ,QQ:=JordanForm(A , output=['J','Q']);
An:=QQ . <<JJ[1,1]^n,0>|<0,J[2,2]^n>> . QQ^(-1);
return evalf(An[2,1]):
end proc:
```

```
> a3(9);
```

33.99999985

```
> st:=time():
a(20000):
time()-st;
st:=time():
a2(20000):
time()-st;
st:=time():
a3(20000000):
time()-st;
```

3.575

0.

0.200

```
>
```

Wir sehen, dass die dritte Prozedur auch um einiges schneller ist als die erste.

Um unsere Prozedur Spirale() zu verbessern, könnte man überall die Prozedur a() durch a2() ersetzen und somit Spirale() schneller machen!!