Prof. Dr. Stefan Funken
M.Sc. Andreas Bantle
Dipl.-Math. oec. Klaus Stolle

Universität Ulm
Institut für Numerische Mathematik
Sommersemester 2015

Numerische Analysis - Matlab-Blatt 6
Lösung

(Besprechung in den MATLAB-Tutorien in KW 27/28)

Hinweise:
Siehe MATLAB-Blatt 1/2.

Aufgabe 10 *(Addition, Multiplikation und Differentiation von Polynomen)* (9 Punkte)

Seien $p, q : \mathbb{R} \to \mathbb{R}$ Polynome mit $p(x) = \sum_{k=0}^{m} a_k x^k$ und $q(x) = \sum_{k=0}^{n} b_k x^k$, sowie $a_m \neq 0$ und $b_n \neq 0$.

a) Schreiben Sie drei Funktionen

(i) `c = add(a,b)`

(ii) `c = mult(a,b)`

(iii) `c = derivative(a)`

die zu den Spaltenvektoren $a = (a_0, \ldots, a_m)^T$ und $b = (b_0, \ldots, b_n)^T$ jeweils die Koeffizienten des Polynoms $p + q$, $p \cdot q$ und der Ableitung p' als Spaltenvektor `c` ausgeben sollen. Dabei soll der führende Koeffizient, also die letzte Koordinate des Koeffizientenvektors, ungleich Null sein. Das Nullpolynom soll durch den „leeren Vektor“ `[]` dargestellt werden.

```
function b=add(a,b)
na=size(a,1);
nb=size(b,1);
if isempty(a)
elseif isempty(b)
    b=a;
elseif na<nb % Addition
    b(1:na,1)=b(1:na,1)+a;
else % Addition
    a(1:nb,1)=a(1:nb,1)+b;
    % geg Abschneiden der Nullen am Ende von c
    k=na;
    while k~=0 && a(k,1)==0
        k=k-1;
    end
    if k==0
        b=[];
    else
        b=a(1:k,1);
    end
end
```

```
function c=mult(a,b)

na=size(a,1);
nb=size(b,1);

% Berechnung der Groesse des resultierenden Polynoms
n=na+nb-1;

```

```
if na*nb~=0
    c=zeros(n,1);
    % sukzessive Multiplikation von a_k*x^k auf b
    for i=1:na
        c(i:nb+i-1)=c(i:nb+i-1)+a(i)*b;
    end
else
    c=[];
end
```

```
function a=derivative(a)

n=size(a,1);
if n==0 || n==1 % Falls das Polynom konstant ist
    a=[];
else
    % sonst konstantes Glied abschneiden
    a(1)=[];
    % mit entsprechendem Faktor multiplizieren
    a=(1:n-1)'.*a;
end
```

b) Schreiben Sie ein Testskript `testPoly.m`, welches die Funktionen an den Beispielen

(i) $p(x) = 0\,,$ $q(x) = 4x^2 - 3$
(ii) $p(x) = 2x^2 - 3x\,,$ $q(x) = -2x^2 + 3x$

testet.

```
clear all;
close all;
clc;

%example 1
a = [];
b = [-3; 0; 4];

c_add  = add(a,b)
c_mult = mult(a,b)
c_der  = derivative(a)



%example 1
a = [0; -3; 2];
b = [0; 3; -2];

c_add  = add(a,b)
c_mult = mult(a,b)
c_der  = derivative(a)
```

Aufgabe 11 *(Momente und Skalarprodukt)* (9 Punkte)

Sei $\omega : [r, s] \to \mathbb{R}$ eine positive, integrierbare Funktion. Ein Skalarprodukt $(\cdot, \cdot)_\omega$ über den Raum der Polynome $\mathbb{P}$ mit

$$(p, q)_\omega := \int_r^s p(x)q(x)\omega(x)\,dx$$

ist eindeutig durch seine Momente

$$\mu_n = \int_r^s x^n\omega(x)\,dx \qquad (n \in \mathbb{N}_0)$$

bestimmt.

a) Schreiben Sie eine Funktion

```
mu = myMoment(n,name)
```

welche für die Eingabeparameter $n \in \mathbb{N}_0$ und `name` als String die entsprechende Momente $\mu_0, ..., \mu_n$ berechnet. Dabei gilt für die Momente folgendes

Name	r	s	$\omega(x)$	μ_n
'legendre'	-1	1	1	n gerade: $\frac{2}{n+1}$, n ungerade: 0
'log'	0	1	$-\log x$	$\frac{1}{(n+1)^2}$
'tscheb'	-1	1	$\frac{1}{\sqrt{1-x^2}}$	n gerade: $\pi \cdot \frac{1 \cdot 3 \cdot 5 \cdots n-1}{2 \cdot 4 \cdot 6 \cdots n}$, n ungerade: 0
'laguerre'	0	∞	e^{-x}	$n!$
'hermite'	$-\infty$	∞	e^{-x^2}	n gerade: $\Gamma\left(\frac{n+1}{2}\right)$, n ungerade: 0

```
function mu = myMoment(n,name)

mu = zeros(n+1,1);

for i=0:n
    if strcmp(name,'legendre')
        if mod(i,2) == 0
            mu(i+1) = 2/(i+1);
        end
    elseif strcmp(name,'log')
        mu(i+1) = 1/(i+1)^2;
    elseif strcmp(name,'tscheb')
        if mod(i,2) == 0
            if i == 0
                mu(i+1) = pi;
            else
                v1 = 1:2:i-1;
                v2 = 2:2:i;
                mu(i+1) = pi*prod(v1)/prod(v2);
            end
        end
    elseif strcmp(name,'laguerre')
        mu(i+1) = factorial(i);
    elseif strcmp(name,'hermite')
        if mod(i,2) == 0
            mu(i+1) = gamma((i+1)/2);
        end
    end
end
```

b) Schreiben Sie eine Funktion

```
s = scalar(a,b,name)
```

welche für die Eingabeparameter

- `a`: der Koeffizientenvektor eines Polynoms $p \in \mathbb{P}$
- `b`: der Koeffizientenvektor eines Polynoms $q \in \mathbb{P}$
- `name`: String, der die Gewichtsfunktion festlegt

das Skalarprodukt $(p, q)_\omega$ berechnet. *Hinweis: Verwenden Sie* `myMoment` *und* `mult`.

```
function s = scalar(a,b,name)

c = mult(a,b);
n = length(c);
m = myMoment(n-1,name);
s = c'*m;
```

c) Schreiben Sie ein Testskript `testMomentScalar.m`, welches beide Funktionen ausreichend und sinnvoll für die von Ihnen gewählten Beispiele testet. Beachten Sie hierbei, dass Sie alle Varianten testen sollen. Aufrufe könnten beispielsweise wie folgt aussehen

```
myMoment(5,'tscheb') = [3.1416; 0; 1.5708; 0; 1.1781; 0]
scalar([1;0;-8;0;8],[0;5;0;-20;0;16],'tscheb') = 0
```

```
clear all;
close all;
clc;

myMoment(8,'legendre')
myMoment(5,'log')
myMoment(6,'tscheb')
myMoment(9,'laguerre')
myMoment(8,'hermite')

a = [1;0;-8;0;8];
b = [0;5;0;-20;0;16];

s = scalar(a,b,'tscheb')
```

Aufgabe 12 *(Determinant Representation of Orthogonal Polynomials)* (12 Punkte)

A positiv weight function ω on an interval determines a system of orthogonal polynomials P_n uniquely, apart from a constant factor. Using the moments of the weight function, i.e.

$$\mu_n = \int_r^s x^n \omega(x)\, dx,$$

we find a determinant representation formula for a point wise evaluation of such orthogonal polynomials. The representation is ($P_0(z) = 1$) and

$$P_n(z) = \frac{\det M}{\det T} \qquad n = 1, ..$$

with

$$M = \begin{pmatrix} \mu_0 & \mu_1 & \mu_2 & \dots & \mu_n \\ \mu_1 & \mu_2 & \mu_3 & \dots & \mu_{n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ \mu_{n-1} & \mu_n & \mu_{n+1} & \dots & \mu_{2n-1} \\ 1 & z & z^2 & \dots & z^n \end{pmatrix} \in \mathbb{C}^{(n+1)\times(n+1)} \quad \text{and} \quad T = \begin{pmatrix} \mu_0 & \mu_1 & \mu_2 & \dots & \mu_{n-1} \\ \mu_1 & \mu_2 & \mu_3 & \dots & \mu_n \\ \vdots & \vdots & \vdots & & \vdots \\ \mu_{n-1} & \mu_n & \mu_{n+1} & \dots & \mu_{2n-2} \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

a) Write a function

```
Pz = evalOrtho(z,n,name)
```

which calculates $P_n(z)$ for $n \in \mathbb{N}_0$ corresponding to the weight function defined through `name`.

```
function Pz = evalOrtho(z,n,name)


m = myMoment(2*n+2,name);

M = zeros(n+1,n+1);
T = zeros(n+1,n+1);

M(end,1) = 1;
for k = 1:n
    M(k,:) = m(k:k+n)';
    if k > 1
        M(end,k) = z^(k-1);
    end
end
```

```
M(end,end) = z^n;

T = M(1:end-1,1:end-1);

Pz = det(M)/det(T);
%Pz = sqrt(2/(2*n+1))*det(M)
```

b) Write a test script `plotOrtho.m`, which generates six plots

(i) $P_n(z)$ for $z \in [-1,1]$ with `name = 'legendre'` and `n=4,5,6`

(ii) $P_n(z)$ for $z \in [0,1]$ with `name = 'log'` and `n=4,5,6`

(iii) $P_n(z)$ for $z \in [-1,1]$ with `name = 'tscheb'` and `n=4,5,6`

(iv) $P_n(z)$ for $z \in [0,5]$ with `name = 'laguerre'` and `n=4,5,6`

(v) $P_n(z)$ for $z \in [-2,2]$ with `name = 'hermite'` and `n=4,5,6`

(vi) $\log|P_n(z)|$ for $z \in \mathbb{C}$ with $\mathrm{Re}(z)$, $\mathrm{Im}(z) \in [-4,4]$, `name = 'legendre'` and `n=5`

```
clear all;
close all;
clc;

%% generate first plot: 'legendre'
x = linspace(-1,1,256);

for n=4:6
    for k = 1:length(x)
        fx(n-3,k) = evalOrtho(x(k),n,'legendre');
    end
end

figure(1)
plot(x,fx(1,:),x,fx(2,:),x,fx(3,:))
title('name = legendre')
legend('P_4(z)','P_5(z)','P_6(z)')

%% generate second plot: 'log'
x = linspace(0,1,256);

for n=4:6
    for k = 1:length(x)
        fx(n-3,k) = evalOrtho(x(k),n,'log');
    end
end

figure(2)
plot(x,fx(1,:),x,fx(2,:),x,fx(3,:))
title('name = log')
legend('P_4(z)','P_5(z)','P_6(z)')

%% generate third plot: 'tscheb'
x = linspace(-1,1,256);

for n=4:6
    for k = 1:length(x)
        fx(n-3,k) = evalOrtho(x(k),n,'tscheb');
    end
end

figure(3)
plot(x,fx(1,:),x,fx(2,:),x,fx(3,:))
title('name = tscheb')
legend('P_4(z)','P_5(z)','P_6(z)')

%% generate fourth plot: 'laguerre'
x = linspace(0,5,256);

```

```
for n=4:6
    for k = 1:length(x)
        fx(n-3,k) = evalOrtho(x(k),n,'laguerre');
    end
end

figure(4)
plot(x,fx(1,:),x,fx(2,:),x,fx(3,:))
title('name = laguerre')
legend('P_4(z)','P_5(z)','P_6(z)')

%% generate fourth plot: 'hermite'
x = linspace(-2,2,256);

for n=4:6
    for k = 1:length(x)
        fx(n-3,k) = evalOrtho(x(k),n,'hermite');
    end
end

figure(5)
plot(x,fx(1,:),x,fx(2,:),x,fx(3,:))
title('name = hermite')
legend('P_4(z)','P_5(z)','P_6(z)')

%% generate last plot: 'legendre' for complex argument
sx = linspace(-4,4);
sy = linspace(-4,4);
[X,Y] = meshgrid(sx,sy);

fz = zeros(length(sy),length(sx));

for nx = 1:length(sx)
    for ny = 1:length(sy)
        fz(nx,end-ny+1) = evalOrtho(sx(nx)+i*sy(ny),5,'legendre');
    end
end

figure(6)
surf(X,Y,log(abs(fz)))
title('Plot of log(abs(P_5(z))) with name = legendre on the complex plane' )
```

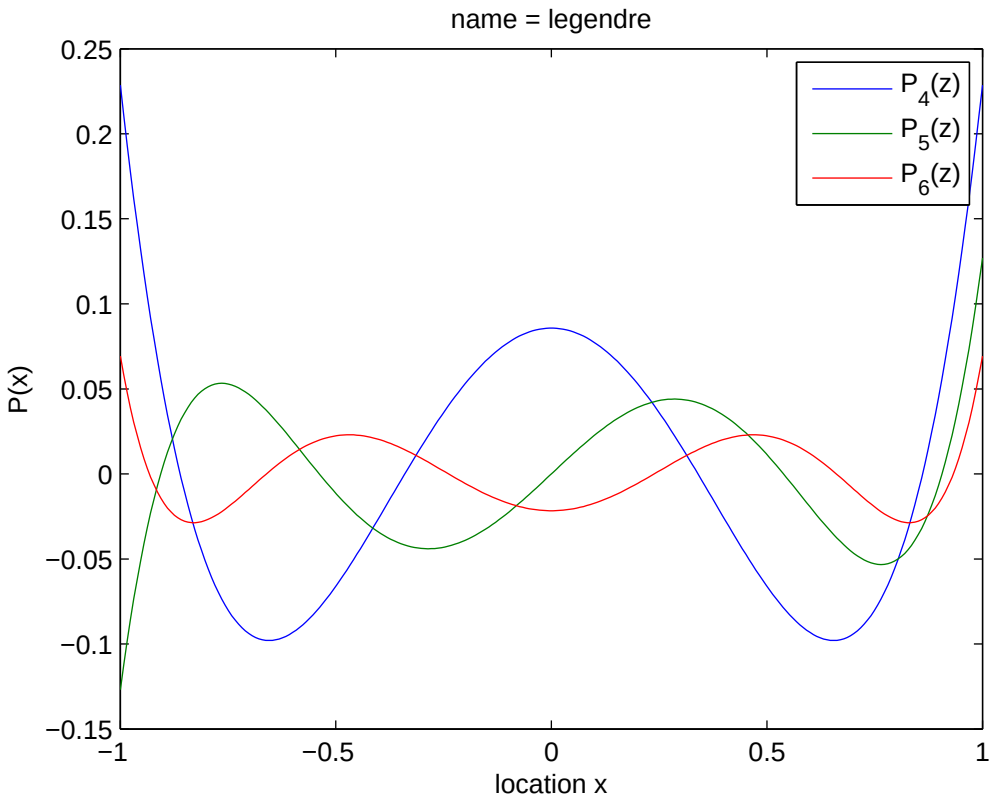

Abbildung 1: Functional plot of legendre polynomials

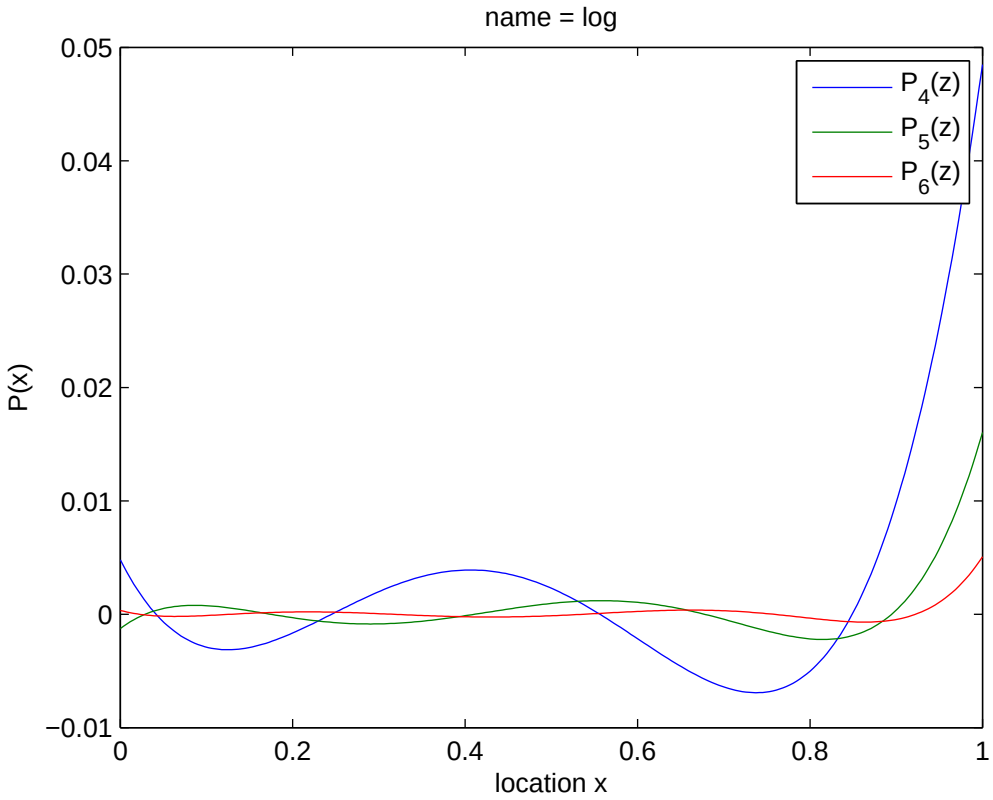

Abbildung 2: Functional plot of the logarithmic polynomials

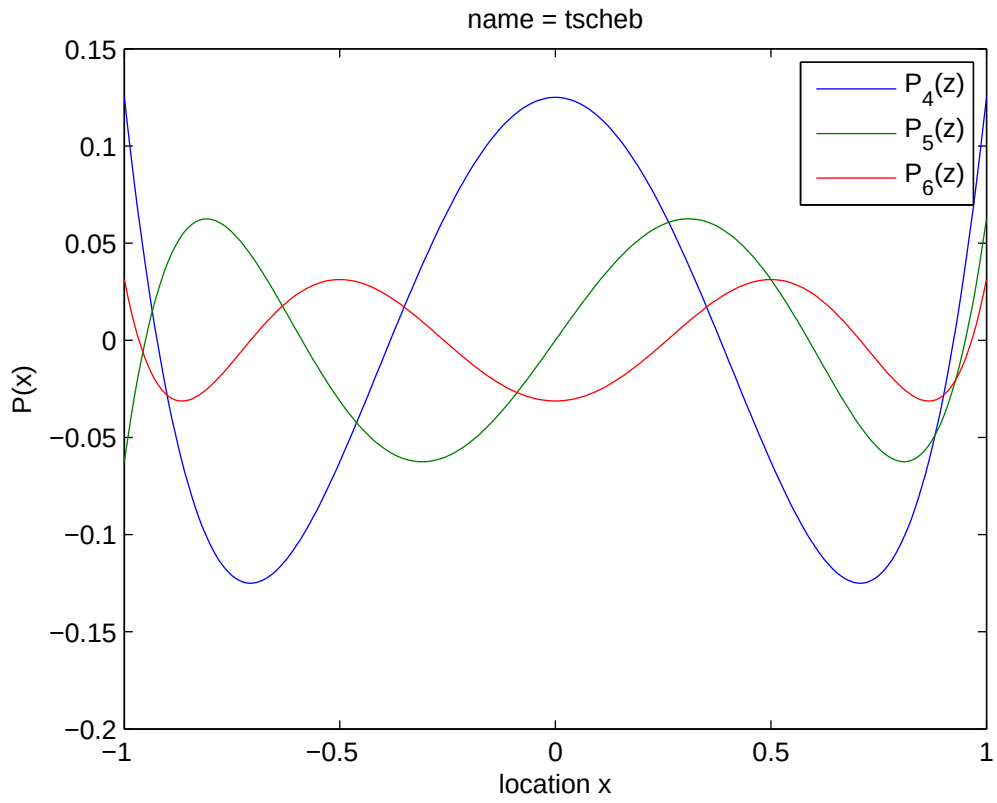

Abbildung 3: Functional plot of the tschebyscheff polynomials

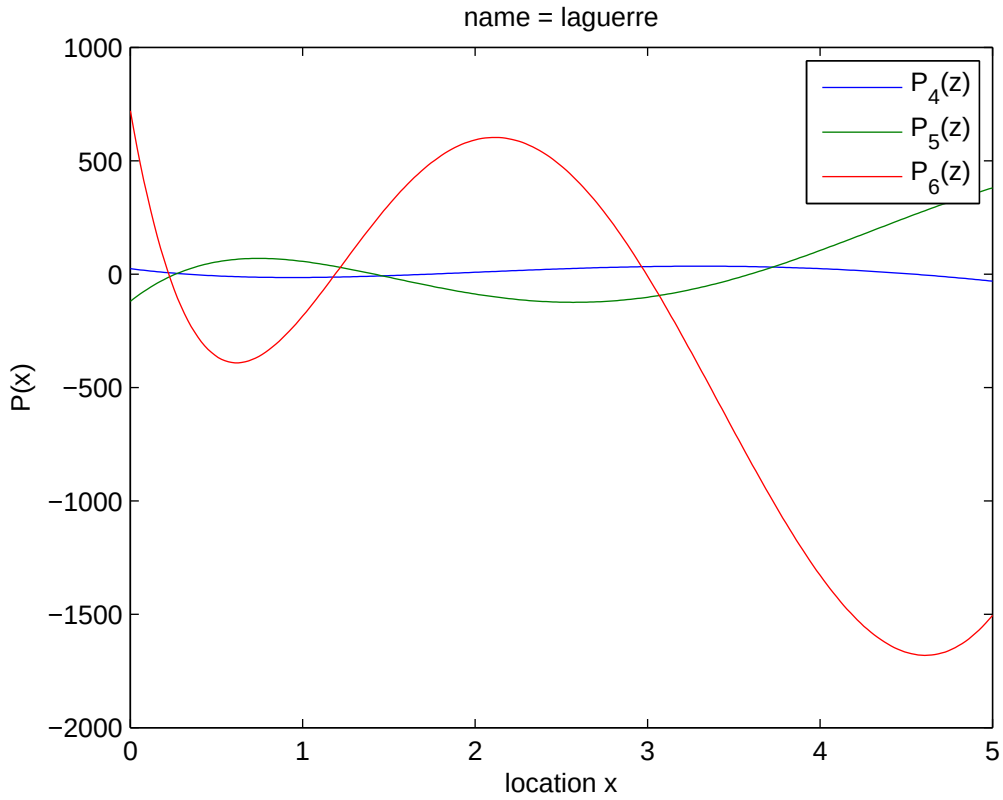

Abbildung 4: Functional plot of the laguerre polynomials

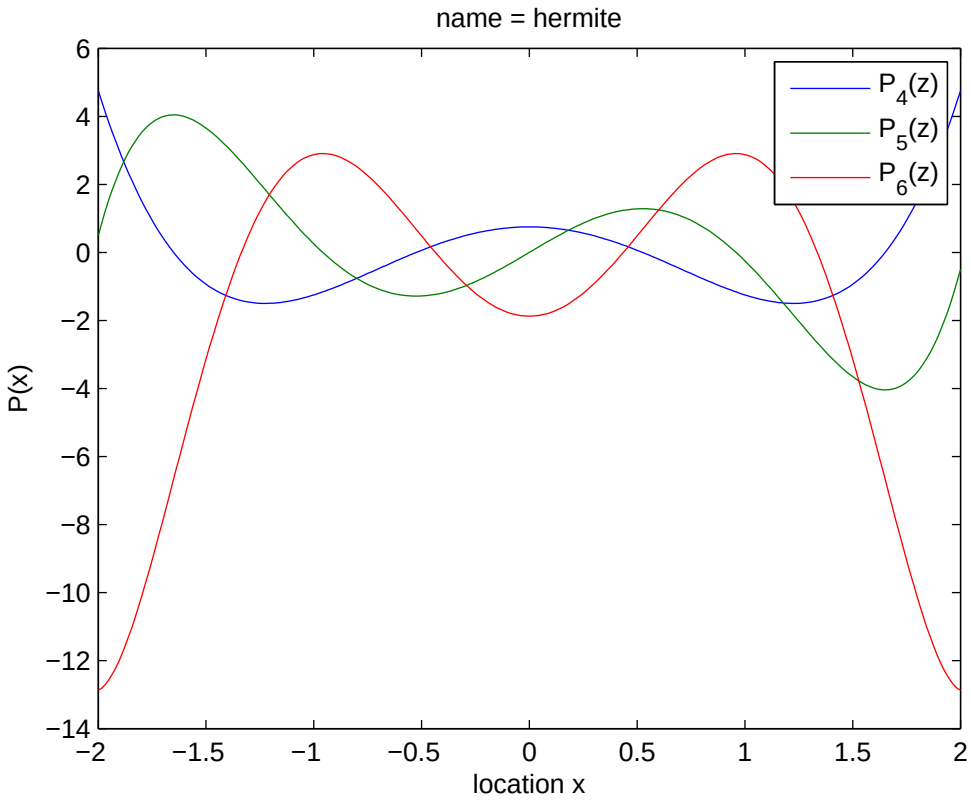

Abbildung 5: Functional plot of the hermite polynomials

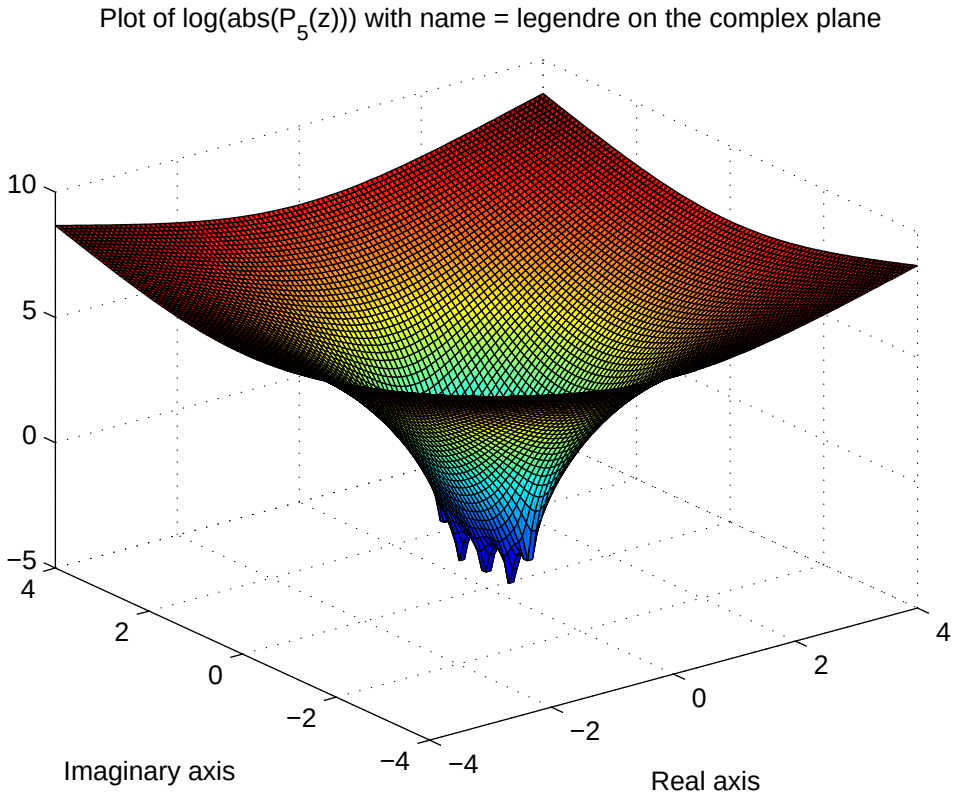

Abbildung 6: Functional plot of the tschebyscheff polynomials

Mehr Informationen zur Vorlesung und den Übungen finden Sie auf

`http://www.uni-ulm.de/mawi/mawi-numerik/lehre/sose15/numana0.html`