# Aufgabe 1 Beispiel 5 Arcustangens

**Choi Hyungi und Matkovich Sebastian**

bei Winfried Auzinger und Ewa Weinmüller

**Zusammenfassung**—Es wird ein Programm zur Berechnung der Arcustangensfunktion $\arctan(x)$ entwickelt. Die Annäherung erfolgt durch die Taylorreihe um $x_0 = 0$, die für $|x| \leq 1$ konvergiert. Die notwendigen Taylorreihenglieder zur Erreichung einer Genauigkeit von mindestens 100 eps werden berechnet und grafisch dargestellt.
Relative Fehler bei konkreten $x$-Werten werden untersucht und mit dem matlabeigenen Arcustangens verglichen, wobei die Genauigkeit bei natürlicher und umgekehrter Summationsreihenfolge betrachtet wird. Für $x$-Werte nahe 1 wird eine alternative, schneller konvergierende Reihe verwendet und getestet. Schließlich wird ein Vergleich mit Lagrangeinterpolation mit äquidistanten Stützstellen und mit duch Chebyshevpolynome definierte Stützstellen hergestellt. Verwendet werden ein HP Envy x360 und ein Razer Blade Stealth 13 2020.

## Inhaltsverzeichnis



## 1. Parameters

### 1.1. Beispiel 5.1

Es ist sinnvoll für den Bereich -1 bis 1 die Approximation zu berechnen. Da der Wertebereich um 0 antisymmetrisch ist, kann er auf das Intervall 0 bis 1 eingeschränkt werden. Außerhalb dieses Bereichs nähert sich der Funktionswert $\frac{\pi}{2}$ an. Die trivialen Funktionswerte sind für das Argument 0: 0, für das Argument 1: $\frac{\pi}{4}$ und für das Argument $-1$: $-\frac{\pi}{4}$. Für die Entwicklungen haben die folgende Reihe verwendet:

$$x = 1 - 2^{-k} \quad , k = 1 \ldots, 19$$

### 1.2. Beispiel 5.2

Die Anzahl nötiger Reihenglieder für eine gewünschte Genauigkeit lassen sich mit dem Term abschätzen, der entsteht, wenn in der Taylorreihenentwicklung des Arcustangens der Summenindex n, bis zu dem Summiert wird, um 1 erhöht wird. Dann entsteht folgender Term: $\frac{x^{2n+3}}{2n+3}$. Mit der Division durch $\arctan(x)$ erhalten wir eine Abschätzung für den relativen Fehler. Wenn wir diesen relativen Fehler gleich 100eps setzen, können wir numerisch nach n auflösen. Wenn der Betrag von x kleiner als 1 ist, konvergiert dieser Fehler und 100eps können erreicht werden. Genau bei 1 müssten in etwa $10^{14}$ Werte Summiert werden, da bei uns 100eps in etwa $10^{-14}$ ist. Bei Werten im Betrag kleiner 1 könnte auch eine Umformung folgendermaßen geschehen:

$$\frac{\frac{x^{2n+3}}{2n+3}}{\arctan(x)} \leq 100eps \quad \Big| \cdot \arctan(x)$$

$$\frac{x^{2n+3}}{2n+3} \leq 100eps \arctan(x) \quad \Big| \frac{d}{dx}$$

$$x^{2n+2} \leq \frac{100eps}{1+x^2} \quad \Big| \ln$$

$$(2n+2)\ln(x) \leq \ln\left(\frac{100eps}{1+x^2}\right) \quad \Big| \frac{1}{\ln(x)}$$

$$(2n+2) \leq \frac{\ln\left(\frac{100eps}{1+x^2}\right)}{\ln(x)} \quad \Big| \frac{1}{2} \quad \Big| -1$$

$$n \leq \frac{\ln\left(\frac{100eps}{1+x^2}\right)}{2\ln(x)} - 1$$

Dieser Ausdruck liefert eine schnellere Näherung an die Anzahl benötigter Reihenglieder um die Genauigkeit von 100eps zu erreichen. Vorallem wächst die Rechenzeit der numerische Methode exponential je näher wir bei eins sind im Vergleich zur analytische Methode.

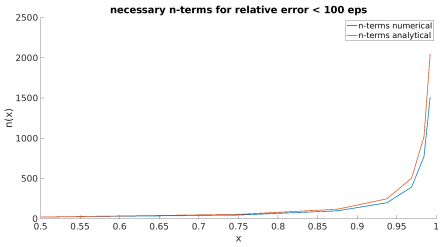


**Abbildung 1.** Anzahl benötigter Taylorreiheterme bis nahe 1.

Bei genau 1 verhaltet sich die analytische Methode asymptotisch während die numerische im Bereich von $10^{14}$ ist.

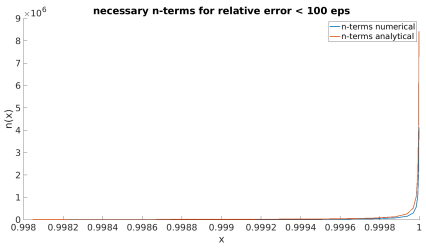


**Abbildung 2.** Anzahl benötigter Taylorreiheterme bis nahe 1.

## 2. Taylorentwicklung um 0

### 2.1. Beispiel 5.3

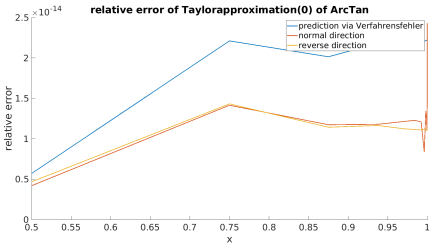


**Abbildung 3.** relativer Fehler bei Entwicklung bis nahe 1.

Die relative Genauigkeit wird für $|x| < 1$ erreicht, jedoch müssen immer mehr Werte aufsummiert werden, damit diese erreicht wird.

### 2.2. Beispiel 5.4

Wie wir an der Abschätzung erkennen können müssen für Argumente, die sich im Betrag 1 annähern, immer mehr Werte summiert werden, um die geforderte Genauigkeit zu erreichen. Bei Summation in umgekehrter Reihenfolge sind die zu summierenden Werte eher in gleicher Größenordnung und damit kann die Fehlerquelle, dass kleine Zahlen zu großen dazuaddiert, diese Zahl nicht mehr erhöhen, weil die Größenordnungen zu weit auseinander liegen, weitestgehend ausgeschlossen werden. Deshalb wird die Genauigkeit der umgekehrten Reihenfolge im Bereich 1, wo die n werte exponential ansteigen, deutlich höher.

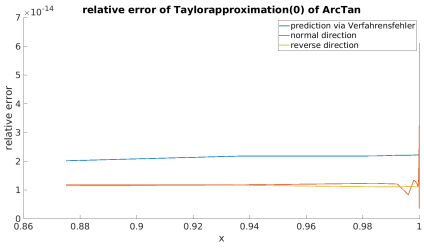


**Abbildung 4.** relativer Fehler bei Entwicklung nahe 1.

Zusätzlich können wir bei niedrigen x-Werten von Abb.3 auch beobachten, dass bei niedrigen n Werten zufällige Rechenfehler auftauchen, so dass das Kommutativgesetz nicht erhalten bleibt.

## 3. Taylorentwicklung um 1

### 3.1. Beispiel 5.5

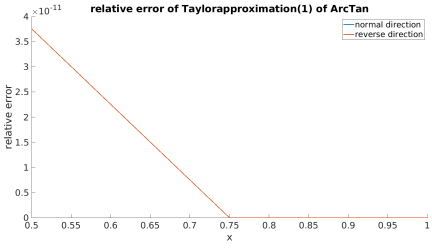


**Abbildung 5.** relativer Fehler bei Entwicklung um 1.

Hier sieht man, dass Reihenentwickung des Arcustanges um 1 nur für x werten gößer als ca. 0,75 sinnvoll ist. Und in der folgenden Abbildung können wir auch sehen dass die relative Genauigkeit der umgekehrter Summationsreihenfolge generell um Größenordnungen besser als die normale.

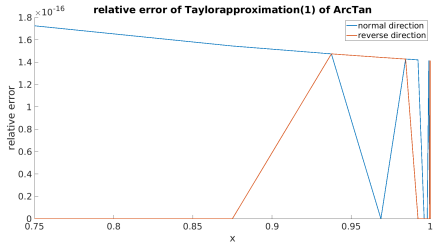


**Abbildung 6.** relativer Fehler bei Entwicklung zwischen 0.75 und 1.

Und im Vergleich zu der Taylorentwicklung um 0 ist die Taylorentwicklung um 1 ab 0.75 besonders hervorrangend.

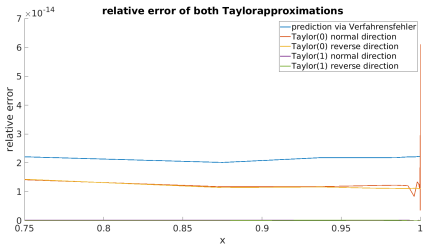


**Abbildung 7.** relativer Fehler beider Taylorentwicklungen zwischen 0.75 und 1.

## 4. selektive Taylorentwicklung

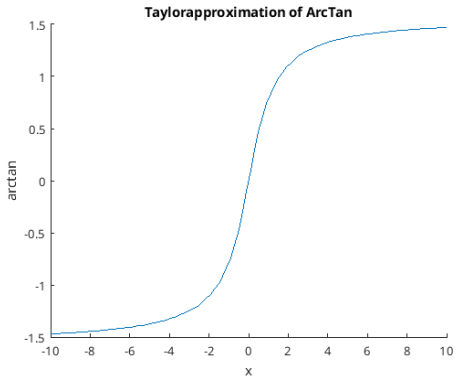


**Abbildung 8.** Taylorapproximation von arctan.

### 4.1. Beispiel 5.6, n numerisch

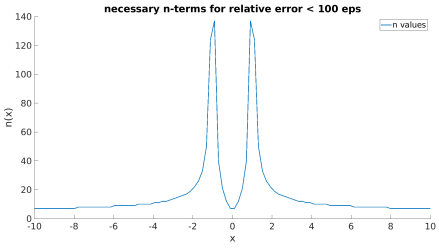


**Abbildung 9.** Anzahl benötigter Reihenglieder.

  

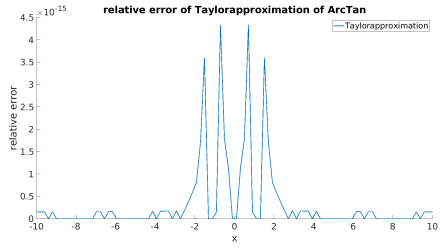


**Abbildung 10.** Relativer Fehler mit Taylorreihenentwicklung.

### 4.2. Beispiel 5.6, n analytisch

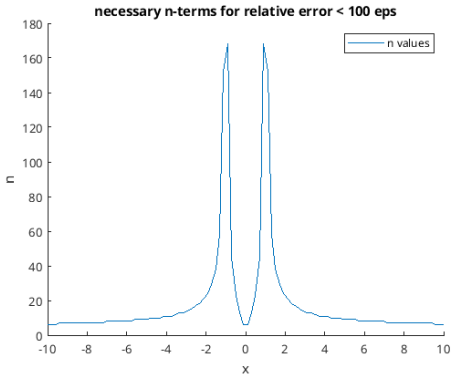


**Abbildung 11.** Anzahl benötigter Reihenglieder.

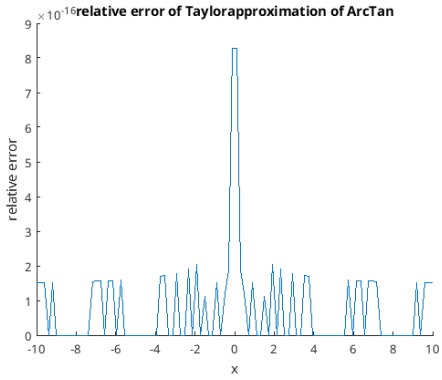


**Abbildung 12.** Relativer Fehler mit Taylorreihenentwicklung.

## 5. Lagrange Interpolation

### 5.1. Beispiel 5.7

Die relative Genauigkeit bei Lagrange Interpolation mit äquidistanter Stützstellen zwischen 0 und 1, ist bei 20 Stützstellen maximal. Die extreme Ungenauigkeit an die Grenzwerten kommt durch die Unstetigkeit der Stützstellen in diesem Bereich zustande.

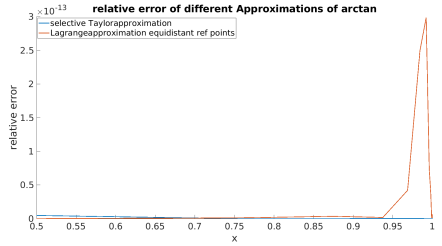


**Abbildung 13.** Lagrange Interpolation mit äquidistante Stützstellen.

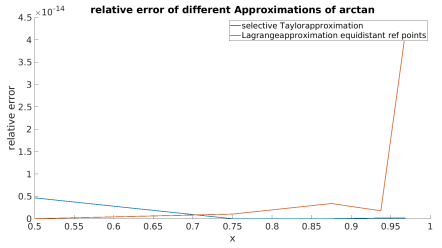


**Abbildung 14.** Lagrange Interpolation mit äquidistante Stützstellen bis 0.95.

Bei der Lagrange Interpolation mit Chebyshev Knoten zwischen [-1, 1] sehen ebenfalls einen extremen Anstieg an den Grenzwerten aus dem selben Grund. Allerdings wächst bei Chebyshew Knoten die relative Genauigkeit mit der Anzahl der Stützstellen, falls man den Bereich um die 1 vernachlässigt. Um den gesamten Bereich unter 100 eps zu halten ist die Anzahl von 36 Stützstellen optimal.

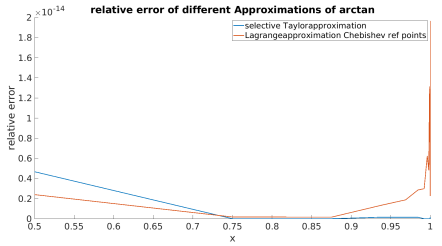


**Abbildung 15.** Lagrange Interpolation mit Chebyshev Knoten.

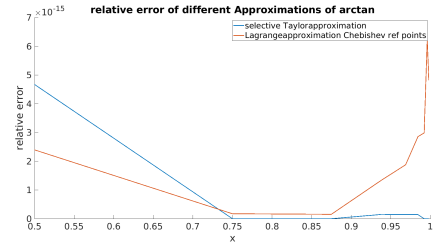


**Abbildung 16.** Lagrange Interpolation mit Chebyshev Knoten bis nahe 1.

In den folgenden Graphen kann man beobachten das die Lagrange Interpolationen generell eine höhere Genauigkeit aufweisen als die Taylorentwicklung. Die höhere Genauigkeit der Taylorreihe für $x > 0.75$ kommt dadurch zustande weil die Taylorentwicklung um 1 mit so vielen Gliedern errechnet wird, dass ihre Genauigkeit, wie in Abb.7 erkennbar, in eine anderen Größen Ordnung ist.

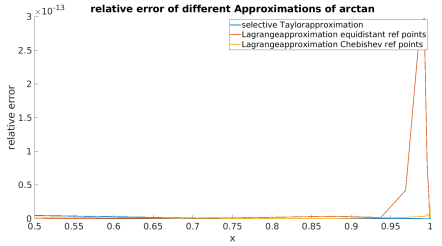


**Abbildung 17.** relativer Fehler von verschiedener Approximationen.

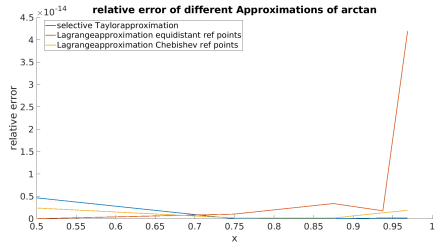


**Abbildung 18.** relativer Fehler von verschiedener Approximationen bis nahe 1.

## 6. Quellcode

### 6.1. Mains

```
clear variables;
close all;
%{
Task:
focused primarily on Analysing Approximations in the interval [0, 1]
%}

%------------------------------
% x Values
%------------------------------
% Parameters
xpts = 19;
xmin = 0;
xmax = 1;

xs = xvalues(xmax, xpts);
%xs = linspace(xmin, xmax, xpts);

%------------------------------
% minimum n terms for the Taylorapproximation via Verfahrensfehler
% to keep relative Error below 100 eps
%------------------------------
[ns_num, ers_num] = n_numerical(xs); % numerically
[ns_ana, ers_ana] = n_analytical(xs); % analytically

ns = ns_num;        % n values
ers_ns = ers_num;   % relative error according to Verfahrensfehler
disp('ns ok');

% plot n-terms
figure;
    idx_strt = 1;
    idx_end = fix(length(xs)*0.4); % 1 for fullrange; 0.4 for relevant range

    hold on;
    plot(xs(idx_strt:idx_end), ns_num(idx_strt:idx_end), 'DisplayName', 'n-terms numerical');
    plot(xs(idx_strt:idx_end), ns_ana(idx_strt:idx_end), 'DisplayName', 'n-terms analytical');
    title('necessary n-terms for relative error < 100 eps');
    xlabel('x');
    ylabel('n');
    legend;
    hold off;

%------------------------------
% Taylorsequence around 0untitled
%------------------------------
[arctans_taylor0, ers_taylor0] = taylor0_arctan(xs, ns);                  % Taylorapprox(0) normal direction
[rev_arctans_taylor0, rev_ers_taylor0] = reversetaylor0_arctan(xs, ns);  % Taylorapprox(0) reverse direction

%plot relative Error
figure
    idx_strt = 1;
    idx_end = fix(length(xs)*0.7);

    hold on;
    plot(xs(idx_strt:idx_end), ers_ns(idx_strt:idx_end), 'DisplayName', 'prediction via Verfahrensfehler');
    plot(xs(idx_strt:idx_end), ers_taylor0(idx_strt:idx_end), 'DisplayName', 'normal direction');
    plot(xs(idx_strt:idx_end), rev_ers_taylor0(idx_strt:idx_end), 'DisplayName', 'reverse direction');

    title('relative error of Taylorapproximation (0) of ArcTan')
    xlabel('x');
    ylabel('relative error');
    legend;
    hold off;

%------------------------------
% Taylorapprox around 1
%------------------------------
[arctans_taylor1, ers_taylor1] = taylor1_arctan(xs, ns);                  % Taylorapprox(1) normal order
[rev_arctans_taylor1, rev_ers_taylor1] = reversetaylor1_arctan(xs, ns);   % Taylorapprox(1) reverse order

%plot relative Error to arctan(x)
figure
    idx_strt = fix(length(xs)*0.15); %fix(length(xs)*0.15)  f  r  relevante range
    idx_end = length(xs);

    hold on;
    %plot(xs(idx_strt:idx_end), ers_ns(idx_strt:idx_end), 'DisplayName', 'prediction via Verfahrensfehler');
    plot(xs(idx_strt:idx_end), ers_taylor1(idx_strt:idx_end), 'DisplayName', 'normal direction')
    plot(xs(idx_strt:idx_end), rev_ers_taylor1(idx_strt:idx_end), 'DisplayName', 'reverse direction')

    title('relative error of Taylorapproximation (1) of ArcTan');
    xlabel('x');
    ylabel('relative error');
    legend;
    hold off;

%------------------------------
% Compare Taylorapprox(0) and Taylorapprox(1)
%------------------------------
figure
    idx_strt = fix(length(xs)*0.15);
    idx_end = length(xs);

    hold on;
    plot(xs(idx_strt:idx_end), ers_ns(idx_strt:idx_end), 'DisplayName', 'prediction via Verfahrensfehler');
    plot(xs(idx_strt:idx_end), ers_taylor0(idx_strt:idx_end), 'DisplayName', 'Taylor(0) normal direction');
    plot(xs(idx_strt:idx_end), rev_ers_taylor0(idx_strt:idx_end), 'DisplayName', 'Taylor(0) reverse direction');
    plot(xs(idx_strt:idx_end), ers_taylor1(idx_strt:idx_end), 'DisplayName', 'Taylor(1) normal direction')
```

  

```
    plot(xs(idx_strt:idx_end), rev_ers_taylor1(
    idx_strt:idx_end), 'DisplayName', 'Taylor(1)
     reverse direction')

    title('relative error of both
    Taylorapproximations');
    xlabel('x');
    ylabel('relative error');
    legend;
    hold off;

%-----------------------------
% selective Taylorapprox
%-----------------------------
[arctans_taylor,ers_taylor] = fulltaylor_arctan(
    xs, ns);    % selective Taylorapprox(0 or 1)
    , reverse order

%-----------------------------
% Lagrangeapprox
%-----------------------------
refmin = 0;
refmax = 1;
refpts1 = 20;   % max approximation at 20
refs1 = linspace(refmin, refmax, refpts1);
refpts2 = 36;   % min 36 to be comparable to
    taylor
refs2 = chebishevnodes(refpts2);

[arctans4, ers4] = lagrangeapprox(xs, refs1);
[arctans5, ers5] = lagrangeapprox(xs, refs2);

figure
    idx_strt = 1;
    idx_end = fix(length(xs)*0.3); % 0.5 for rel

    hold on;
    plot(xs(idx_strt:idx_end), ers_taylor(
    idx_strt:idx_end), "DisplayName", 'selective
     Taylorapproximation')
    plot(xs(idx_strt:idx_end), ers4(idx_strt:
    idx_end), "DisplayName", '
    Lagrangeapproximation equidistant ref points
    ')
    plot(xs(idx_strt:idx_end), ers5(idx_strt:
    idx_end), "DisplayName", '
    Lagrangeapproximation Chebishev ref points')

    title('relative error of different
    Approximations of arctan');
    xlabel('x');
    ylabel('relative error');
    legend;
    hold off;

figure
    idx_strt = 1;
    idx_end = length(xs);

    hold on;
    plot(xs(idx_strt:idx_end), arctans_taylor(
    idx_strt:idx_end), "DisplayName", 'selective
     Taylorapproximation')
    plot(xs(idx_strt:idx_end), arctans4(idx_strt
    :idx_end), "DisplayName", '
    Lagrangeapproximation equidistant ref points
    ')
    plot(xs(idx_strt:idx_end), arctans5(idx_strt
    :idx_end), "DisplayName", '
    Lagrangeapproximation Chebishev ref points')

    title('arctan of different Approximations');
    xlabel('x');
    ylabel('arctan(x)');
    legend;
    hold off;
```

**Code 1.** main01.

```
clear variables;
close all;

%-----------------------------
% x Values
%-----------------------------
% Parameters
xpts = 100;
xmin = -10;
xmax = 10;

%xs = xvalues(xmax, kmax);
xs = linspace(xmin, xmax, xpts);

%-----------------------------
% minimum n Values to each x
%-----------------------------
[ns_num, ers_num] = n_numerical(xs);
[ns_ana, ers_ana] = n_analytical(xs);

ns = ns_ana;        % n values
ers_ns = ers_ana;   % errors via
    Verfahrensfehler
ns = ns + 1;
disp('ns, ok')

% plot necessary n-terms
figure;
hold on;
plot(xs, ns, 'DisplayName', 'n values');
title('necessary n-terms for relative error <
    100 eps');
xlabel('x');
ylabel('n');
legend;
hold off;

%-----------------------------
% Taylorapproximation of arctan
%-----------------------------
[arctans, ers] = fulltaylor_arctan(xs, ns);

figure
    idx_strt = 1;
    idx_end = length(xs);

    hold on;
    plot(xs(idx_strt:idx_end), arctans(idx_strt:
    idx_end))

    title('Taylorapproximation of ArcTan');
    xlabel('x');
    ylabel('arctan');
    hold off;

figure
    idx_strt = 1;
    idx_end = length(xs);

    hold on;
    plot(xs(idx_strt:idx_end), ers(idx_strt:
    idx_end), 'DisplayName','Taylorapproximation
    ')
    plot(xs(idx_strt:idx_end), ers_ns(idx_strt:
    idx_end), 'DisplayName','Verfahrensfehler')

    title('relative error of Taylorapproximation
     of ArcTan');
    xlabel('x');
    ylabel('relative error');
    legend;
    hold off;
```

**Code 2.** main02.

### 6.2. Anzahl der Taylorterme

```
function xs = xvalues(xmax, kmax)
    xs = zeros(kmax, 1);
```

```
    for k = 1 : kmax
        x = xmax - 2^(-k);
        xs(k) = x;
    end
end
```

**Code 3.** x werte

```
function [ns, ers] = n_analytical(xs)
%{
Parameter:
    array of x values
Todo:
    calculate n values analytically
Return:
    [array, array]
    1st array: n-values to each x
    2nd array: relative error to each x from
    Verfahrensfehler
%}
    vmax = length(xs);
    ns = zeros(vmax,1);      % array of n values
    ers = zeros(vmax,1);     % array of relative
    errors
    for k = 1 : vmax
        o = xs(k);
        if abs(o) <= 1
            x = abs(o);
        else
            x = abs(1/o);
        end
        comp = atan(x);
        n = log(100*eps/(1+x^2)) / (2*log(x)) -
    1;
        er = (x^(2*n+3)/(2*n+3))/comp;

        ers(k) = abs(er);
        ns(k) = fix(n);
    end
end
```

**Code 4.** n analytisch.

```
function [ns, ers] = n_numerical(xs)
%{
Parameter:
    array of x values
Todo:
    determine n values numerically
Return:
    [array, array]
    1st array: n-values to each x
    2nd array: relative error to each x from
    Verfahrensfehler
%}
    vmax = length(xs);
    ns = zeros(vmax, 1);     % array of n values
    ers = zeros(vmax, 1);    % array of relative
    errors
    for k = 1 : vmax
        er = 1;
        o = xs(k);
        if abs(o)<=1
            x = abs(o);
        else
            x = abs(1/o);
        end
        n = -1;
        comp = atan(x);
        if comp == 0
            n = 0;
        else
            while er > 100*eps
                n = n+1;
                er = (x^(2*n+3)/(2*n+3))/comp;
                er = abs(er);
            end
        end
        ns(k) = n;
        ers(k) = er;
    end
end
```

**Code 5.** n numerisch.

### 6.3. Taylorreihe um 0

```
function termN = nterm_taylor0_arctan(x, n)
%{
Parameter:
    x: x value
    n: order of term
Return:
    n-th term of the Taylorsequence around 0
%}
    term = 2*n + 1;
    termN = (-1)^n * x^term/term;
end
```

**Code 6.** n-ter glied von Taylorreihe um 0.

```
function [arctans, ers] = taylor0_arctan(xs, ns)
%{
Parameter:
    xs: array of x values
    ns: array of n values
Todo:
    1.) approximate arctan(x) via Taylorsequence
    (0) in normal order
    2.) calculate relative error of that
    approximation
Return:
    [array01, array02]
    array01: array of arctan(x) to each x
    array02: array of relative errors to each x
%}
    arctans = zeros(length(xs),1);  % array of
    arctan(x)
    ers = zeros(length(xs), 1);     % arrax of
    relative errors
    for k = 1 : length(xs)
        x = xs(k);
        arctan = 0;
        comp = atan(x);
        for i = 0 : ns(k)
            arctan = arctan +
    nterm_taylor0_arctan(x,i);
        end
        arctans(k) = arctan;
        ers(k) = abs((arctan - comp)/comp);
    end
end
```

**Code 7.** Taylorentwicklung um 0 in normal.

```
function [arctans, ers] = reversetaylor0_arctan(
    xs, ns)
   %{
Parameter:
    xs: array of x values
    ns: array of n values
Todo:
    1.) approximate arctan(x) via Taylorsequence
    (0) in reverse order
    2.) calculate relative error of that
    approximation
Return:
    [array01, array02]
    array01: array of arctan(x) to each x
    array02: array of relative errors to each x
%}
    arctans = zeros(length(xs),1);  % array of
    arctan(x)
```

```
    ers = zeros(length(xs), 1);      % array of
    relative errors
    for k = 1 : length(xs)
        x = xs(k);
        arctan = 0;
        comp = atan(x);
        for i = ns(k) : (-1) : 0
            arctan = arctan +
    nterm_taylor0_arctan(x,i);
        end
        arctans(k) = arctan;
        ers(k) = abs((arctan - comp)/comp);
    end
end
```

**Code 8.** Taylorentwicklung um 0 in umgekehrt.

### 6.4. Taylorreihe um 1

```
function arctan = nterm_taylor1_arctan(delta, k)
%{
Parameter:
    delta: delta between x and 1
    k: order of term
Return:
    k-th term of the Taylorsequence around 1
%}
    arctan = 2^(-k/2)*delta^k*sin(3*k*pi/4)/k;
end
```

**Code 9.** n-ter glied von Taylorreihe um 1

```
function [arctans, ers] = taylor1_arctan(xs, ns)
%{
Parameter:
    xs: array of x values
    ns: array of n values
Todo:
    1.) approximate arctan(x) via Taylorsequence
    (1) in normal order
    2.) calculate relative error of that
    approximation
Return:
    [array01, array02]
    array01: array of arctan(x) to each x
    array02: array of relative errors to each x
%}
    arctans = zeros(length(xs),1);  % array of
    arctan(x)
    ers = zeros(length(xs), 1);      % array of
    relative errors
    for k = 1 : length(xs)
        x = xs(k);
        delta = x - 1;
        arctan = pi/4;
        comp = atan(x);
        for i = 1 : ns(k)
            arctan = arctan +
    nterm_taylor1_arctan(delta,i);
        end
        arctans(k) = arctan;
        ers(k) = abs((arctan - comp)/comp);
    end
end
```

**Code 10.** Taylorentwicklung um 1 normal

```
function [arctans, ers] = reversetaylor1_arctan(
    xs, ns)
%{
Parameter:
    xs: array of x values
    ns: array of n values
Todo:
    1.) approximate arctan(x) via Taylorsequence
    (1) in reverse order
    2.) calculate relative error of that
    approximation
Return:
    [array01, array02]
    array01: array of arctan(x) to each x
    array02: array of relative errors to each x
%}
    arctans = zeros(length(xs),1);  % array of
    arctan(x)
    ers = zeros(length(xs), 1);      % array of
    relative errors
    for k = 1 : length(xs)
        x = xs(k);
        delta = x - 1;
        arctan = 0;
        comp = atan(x);
        for i = ns(k) : (-1) : 1
            arctan = arctan +
    nterm_taylor1_arctan(delta,i);
        end
        arctan = arctan + pi/4;
        arctans(k) = arctan;
        ers(k) = abs((arctan - comp)/comp);
    end
end
```

**Code 11.** Taylorentwicklung um 1 umgekehrt.

### 6.5. selective Taylor

```
function [arctans, ers] = fulltaylor_arctan(xs,
    ns)
    % selective taylorapproximation of arctan
    arctans = zeros(length(xs),1);
    ers = zeros(length(xs), 1);

    for k = 1 : length(xs)
        % selective x value
        o = xs(k);
        if abs(o) <= 1
            x = abs(o);
        else
            x = abs(1/o);
        end

        % Taylorapproximation
        arctan = 0;
        comp = atan(o);
        if(x<0.75)
            for i = ns(k) : (-1) : 0
                arctan = arctan +
    nterm_taylor0_arctan(x,i);
            end
        else
            delta = x - 1;
            for i = ns(k) : (-1) : 1
                arctan = arctan +
    nterm_taylor1_arctan(delta,i);
            end
            arctan = arctan + pi/4;
        end

        % range adjustments
        if abs(o) > 1
            arctan = pi/2 - arctan;
        end
        % sign adjustments
        if o < 0
            arctan = arctan * (-1);
        end

        % return
        arctans(k) = arctan;
        ers(k) = abs((arctan - comp)/comp);
    end
end
```

**Code 12.** selective Taylorentwicklung.

### 6.6. Lagrange Interpolation

```
function xs = chebishevnodes(n)
    xs = zeros(n, 1);
    for i = 0 : (n-1)
        x = cos(i/(n-1)*pi);
        xs(i+1) = x;
    end
end
```

**Code 13.** Chebishev Knoten.

### 6.7. Lagrange Interpolation

```
function f = lagrangepolynom(xs)
    syms x;
    n = length(xs);
    ys = zeros(n,1);
    for i = 1 : n
        ys(i) = atan(xs(i));
    end
    Ls = sym(zeros(1,n));
    for i = 1 : n
        Ls(i) = 1;
        for j = [1:i-1, i+1:n]
            Ls(i) = Ls(i) * (x-xs(j))/(xs(i)-xs(
    j));
        end
    end
    disp("Ls, ok")
    p = Ls*ys;
    p = matlabFunction(p);
    f = p;
end
```

**Code 14.** Lagrangepolynom.

### 6.8. Lagrange Interpolation

```
function [arctans, ers] = lagrangeapprox(xs,
    refs)
    max = length(xs);
    arctans = zeros(max,1);
    ers = zeros(max,1);
    p = lagrangepolynom(refs);

    for i = 1 : max
        x = xs(i);
        arctan = p(x);
        comp = atan(x);
        er = abs(arctan - comp)/comp;

        arctans(i) = arctan;
        ers(i) = er;
    end
end
```

**Code 15.** Lagrangeapproximation.

```
    \RequirePackage[
  backend=biber,
  style=ieee,
  sorting=ynt
    ]{biblatex}
```

**Code 16.** References style.

  