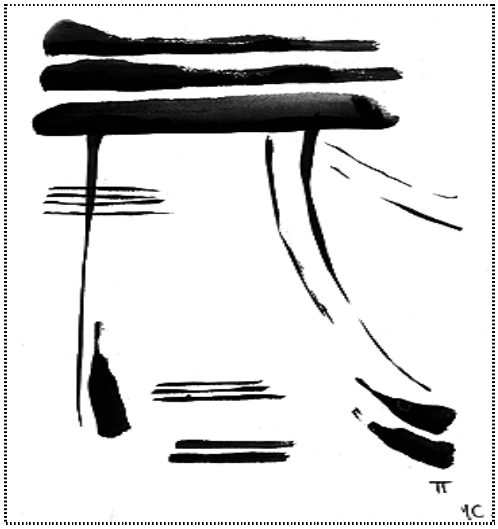
Ausgewählte Algorithmen zur Berechnung von Pi



Belegarbeit zur Lehrveranstaltung Algorithmierung / Programmierung
Vorgelegt von Mirko Hans, Christian-Weise-Gymnasium Zittau
Matrikelnummer:2753736

Inhaltsverzeichnis

1. Aufgabenstellung

Im Rahmen der vorliegenden Belegarbeit sollen exemplarisch einige ausgewählte Berechnungsmethoden der Kreiszahl Pi vorgestellt werden. Anlass für die Idee zu dieser Belegarbeit ist ein Projekt zur Zahl Pi das im Rahmen der Projektwoche am Christian-Weise-Gymnasium in Zittau durchgeführt werden soll.

Beim Studium der Literatur [1] – [3] sowie [5] - [7] stellt man zunächst erst einmal überraschenderweise fest, dass es sehr viele verschiedene Berechnungsmöglichkeiten für die Zahl Pi gibt. Natürlich können nicht alle hier in dieser Arbeit vorgestellt werden. Deshalb habe ich exemplarisch einige Methoden ausgewählt. Die Auswahl erfolgte vor allem unter dem Gesichtspunkt, dass die Berechnungsverfahren mit Mitteln der Schulmathematik für Schüler größerer Klassenstufen nachvollziehbar sein müssen. Dies trifft für die ersten drei Verfahren (Monte-Carlo-Methode, Verfahren nach Gregory, Verfahren nach Cusanus) zu. Die letzten beiden Verfahren (AGM-Methode nach Gauß und Arcustangens-Methode nach Machin) wurden ausgewählt um auch zwei moderne schnell konvergierende Algorithmen vorzustellen.

In dieser Belegarbeit zur Lehrveranstaltung Algorithmierung und Programmierung sollen vor allem informatische Aspekte im Vordergrund stehen und nicht die Mathematik. Deshalb wird immer nur kurz auf die den Berechnungsverfahren zu Grunde liegenden wesentlichen mathematischen Gedanken eingegangen. Zur exakten mathematischen Begründung der Rechenverfahren verweise ich auf die entsprechende Fachliteratur.

2. Theoretischer Hintergrund

In diesem Kapitel möchte ich nun, wie schon angekündigt, einige wesentliche Grundgedanken der ausgewählten Rechenmethoden vorstellen.

2.1. Die Monte-Carlo-Methode

Abb. 1: Comte de Buffon

Interessanterweise ist es möglich Pi mit Hilfe von stochastischen Verfahren zu bestimmen. Ausgangspunkt hierfür sind Untersuchungen des französischen Gelehrten Comte de Buffon (1707 – 1788), der den Wert der Zahl Pi empirisch mit Hilfe eines Zufallsexperimentes bestimmte (Abb. 1). Dazu denkt man sich eine Ebene überdeckt von einer Parallelenschar mit dem Abstand d. Auf diese Parallelen wird willkürlich eine Nadel mit der Länge a (a < d) geworfen (Abb. 2). Es interessiert die Wahrscheinlichkeit mit der die Nadel beim Wurf eine der Parallelen schneidet. Mit Hilfe einiger elementarmathematischer Überlegungen lässt sich eine Formel zur Berechnung von Pi ableiten [4, S. 386 ff.]. Wird dieser Versuch genügend oft durchgeführt, erhält man ganz vernünftige Näherungswerte für Pi.
Der Nachteil dieser Methode besteht darin, dass sie ziemlich unpraktisch ist. Wer möchte schon gern mehrere tausend Mal eine Nadel auf ein Blatt Papier werfen? Einmal mehr würde so die Meinung bestätigt, dass Mathematiker offensichtlich doch wohl nur bedauernswerte „Spinner“ sind, die den Blick für die Realität offenbar schon längst verloren haben. Zumindest sollte man dafür sorgen, dass man bei der Durchführung des Experimentes nicht beobachtet wird.

Hier kann uns einmal mehr der Computer helfen, der geduldig ohne zu murren auch monotone Aufgaben erledigt. Natürlich kann ein Computer keine Nadeln werfen. Also muss man den Versuch entsprechend simulieren. Dazu verwendet man Zufallszahlen, die intern im Rechner erzeugt werden. Eine solche Simulation eines Zufallsversuches mit Hilfe von Zufallszahlen nennt man in der Mathematik Monte-Carlo-Methode.

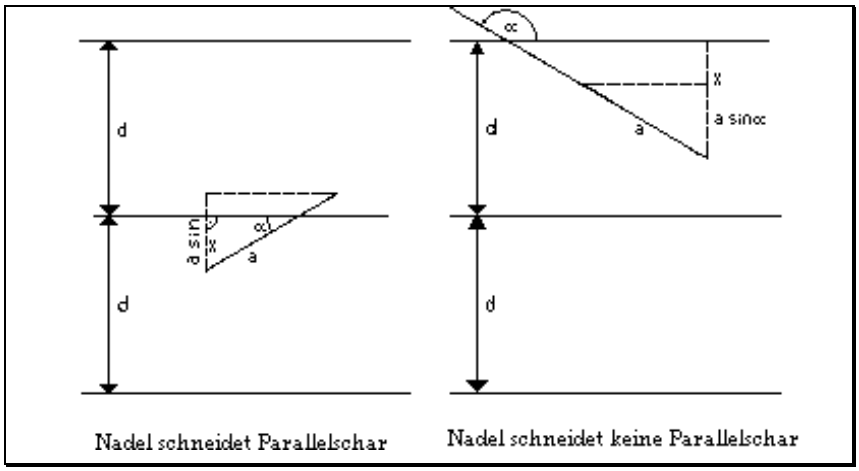


Abb. 2: Prinzip des Nadelversuchs von Buffon

Die Vorgehensweise soll nun anhand von Abb. 3 beschrieben werden. Ausgangspunkt ist ein Quadrat der Länge Eins, dem ein Viertelkreis einbeschrieben wird (Das Verfahren funktioniert auch bei Verwendung eines Vollkreises!). Über das Quadrat wird nun ein „Zufallsregen“ von Punkten verstreut und man zählt die Punkte die innerhalb des Viertelkreises liegen. Aus dem ermittelten zahlenmäßigen Verhältnis der Punkte innerhalb und außerhalb des Viertelkreises kann unter Zuhilfenahme der Flächenformeln von Kreis und Quadrat folgende Gleichung hergeleitet werden ([1], S.38 f.):

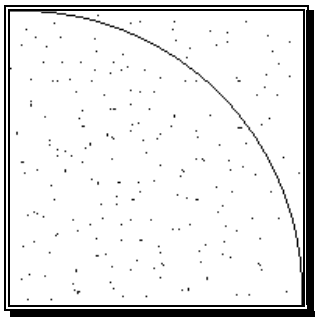


Abb. 3: Monte-Carlo-Verfahren

$$\pi = \frac{4 \cdot \text{Trefferanzahl im Kreis}}{\text{Gesamtanzahl der Versuche}}$$

Es lässt sich leicht nachrechnen, dass diese Gleichung auch bei Verwendung eines Vollkreises gilt. Je mehr zufällige Punkte verstreut werden, um so genauer wird der ermittelte Wert für Pi. Allerdings ist das hier beschriebene Verfahren lediglich von prinzipiellem Interesse und nicht für eine ernsthafte Bestimmung der Zahl Pi geeignet, da auch bei einer sehr hohen Versuchsanzahl nur wenige Stellen von Pi exakt ermittelt werden.

2.2. Methode von Archimedes

Abb. 4: Archimedes von Syrakus

Von Archimedes von Syrakus (287 – 212 v Chr.) stammt die „Exhaustionsmethode“ zur Berechnung unbekannter Flächeninhalte. Dabei wird die gesuchte Fläche einer ebenen Figur durch den Flächeninhalt bekannter Figuren, z.B. von Polygonen, „ausgeschöpft“. Der Figur werden dabei regelmäßige Vielecke ein- bzw. umbeschrieben. Damit erhält man eine obere und eine untere Schranke für den gesuchten Flächeninhalt. Durch eine weitere Verfeinerung , d.h. durch einer Erhöhung der Eckenanzahl der Vielecke , fällt der Unterschied zwischen diesen Näherungen immer geringer aus.
Dieses Verfahren wendete Archimedes auch für die Kreisberechnung an. Dazu berechnete er die Umfänge von regelmäßigen Vielecken, die er in und um einen Kreis legte. Im Laufe der Mathematikgeschichte erfuhr dieses Verfahren eine Reihe von Modifikationen und Verbesse-

rungen. Noch bis zum Beginn der Neuzeit blieb diese Verfahren die wichtigste Methode zur Berechnung von Pi. Archimedes selbst führte die Berechnung eigenhändig bis zum 96-Eck aus. Als Grenzen für Pi erhielt er dabei $3\frac{10}{71} < \pi < 3\frac{1}{7}$.

Je nachdem, ob man bei der schrittweisen Verfeinerung den Radius oder den Umfang verändert bzw. konstant hält unterscheidet man zwei prinzipielle Verfahren.

2.2.1. Die Methode von Gregory

Abb. 5: James Gregory

Bei diesem ersten der Idee von Archimedes folgenden Verfahren werden einem Kreis mit festem Radius Vielecke um- bzw. einbeschrieben, deren Umfänge sich bei der schrittweisen Verfeinerung dem Umfang des Kreises jeweils von oben bzw. unten nähern. Bei bekanntem Durchmesser des Kreises kann dann gemäß der Gleichung $\pi = \frac{u}{d}$ ein Näherungswert für Pi berechnet werden. Dies ist sozusagen die klassische Methode. Die Herleitung der Iterationsgleichungen mit Mitteln der Elementarmathematik ist recht mühselig. Hier möchte ich auf die Literatur (z.B.: [1], S. 164 ff.; [6], S. 3 ff.; oder [7], S. 6 ff.) verweisen.

2.2.2. Die Methode von Cusanus

Abb. 6: Nikolaus von Kues

Im Gegensatz zur klassischen Polygonmethode wählte der deutsche Kardinal und Philosoph Nikolaus von Kues einen anderen Ansatz. Er wählte ein Polygon festen Umfangs (u = 2) und berechnete den Radius des Inkreises sowie des Umkreises. Bei Erhöhung der Anzahl der Ecken des Polygons nähern sich die Radien einander an. Aus den Radien lassen sich wiederum Näherungswerte für Pi berechnen. Zur Herleitung der Iterationsgleichungen verweise ich auf [5], S. 10 ff.

2.3. Die Arcustangensmethode nach Machin

Abb. 7: John Machin

Nachdem Newton und Leibniz zu Beginn des 17. Jahrhunderts die Differenzial- und Integralrechnung entwickelt hatten, ergaben sich völlig neue Möglichkeiten der Berechnung von Pi. Mit Hilfe von unendlichen Reihen konnte man Pi nun wesentlich schneller auf deutlich mehr Stellen berechnen als bisher. Am meisten verwendet wurde die Arcustangens-Reihe. Ausgangspunkt der Überlegungen ist, dass für die Tangens-Funktion die Gleichung

$\tan\frac{\pi}{4} = 1 \quad \text{also} \quad \frac{\pi}{4} = \arctan 1 \quad \text{und damit} \quad \pi = 4 \cdot \arctan 1$ gilt. Wenn man die Arcustangens-Funktion in eine Reihe entwickelt erhält man somit eine Berechnungsmöglichkeit für Pi. Die Idee für dieses Verfahren geht auf Gregory und Leibniz zurück. Allerdings hatte ihr Verfah-

ren den Nachteil, dass die von ihnen entwickelten Reihen nur langsam konvergierten und die Berechnung von Pi somit ziemlich zeitaufwändig war. Leonhard Euler hatte die Idee das Konvergenzverhalten dadurch zu verbessern, dass man in der Gleichung mehrere Arcustangens-Werte verwendet. Ausgehend von dieser Überlegung entwickelte John Machin im Jahre 1706 die im Programm verwendete Formel: $\frac{\pi}{4} = 4 \cdot \arctan\frac{1}{5} - \arctan\frac{1}{239}$. Damit berechnete er den Wert von Pi auf 100 Nachkommastellen. Im Laufe der Zeit entwickelte man noch viele derartige Reihen zur Berechnung von Pi, z.T. mit noch deutlich besserem Konvergenzverhalten.

2.4. Die AGM-Methode nach Gauß

Abb. 8: Carl Friedrich Gauß

Besonders interessant ist die letzte im Programm vorgestellte Berechnungsmethode. Der große deutsche Mathematiker Carl Friedrich Gauß entwickelte sie um das Jahr 1800 herum. Danach geriet sie jedoch in Vergessenheit und wurde erst im Jahr 1976 erneut gefunden. Die AGM-Methode ist eine der schnellsten modernen Berechnungsmethoden für Pi. Die ihr zugrunde liegende Formel lautet:

$$\pi = \frac{2\ AGM^2(1, \frac{1}{\sqrt{2}})}{\frac{1}{2} - \sum_{j=1}^{\infty} 2^j c_j^2}$$

Das Herzstück dieser Formel ist das arithmetisch – geometrische Mittel (AGM), das Gauß bereits im zarten Alter von 14 Jahren fand. Das AGM ist eine Kombination aus dem arithmetischen Mittel (a + b) / 2 und dem geometrischen Mittel $\sqrt{a \cdot b}$. Die Rechenvorschrift für das AGM ist iterativ:

Initialisierung:

$a_0 = a$

$b_0 = b$

Iterationsvorschrift: (k = 0, 1, 2, ...)

$$a_{k+1} := \frac{a_k + b_k}{2}$$

$$b_{k+1} := \sqrt{a_k \cdot b_k}$$

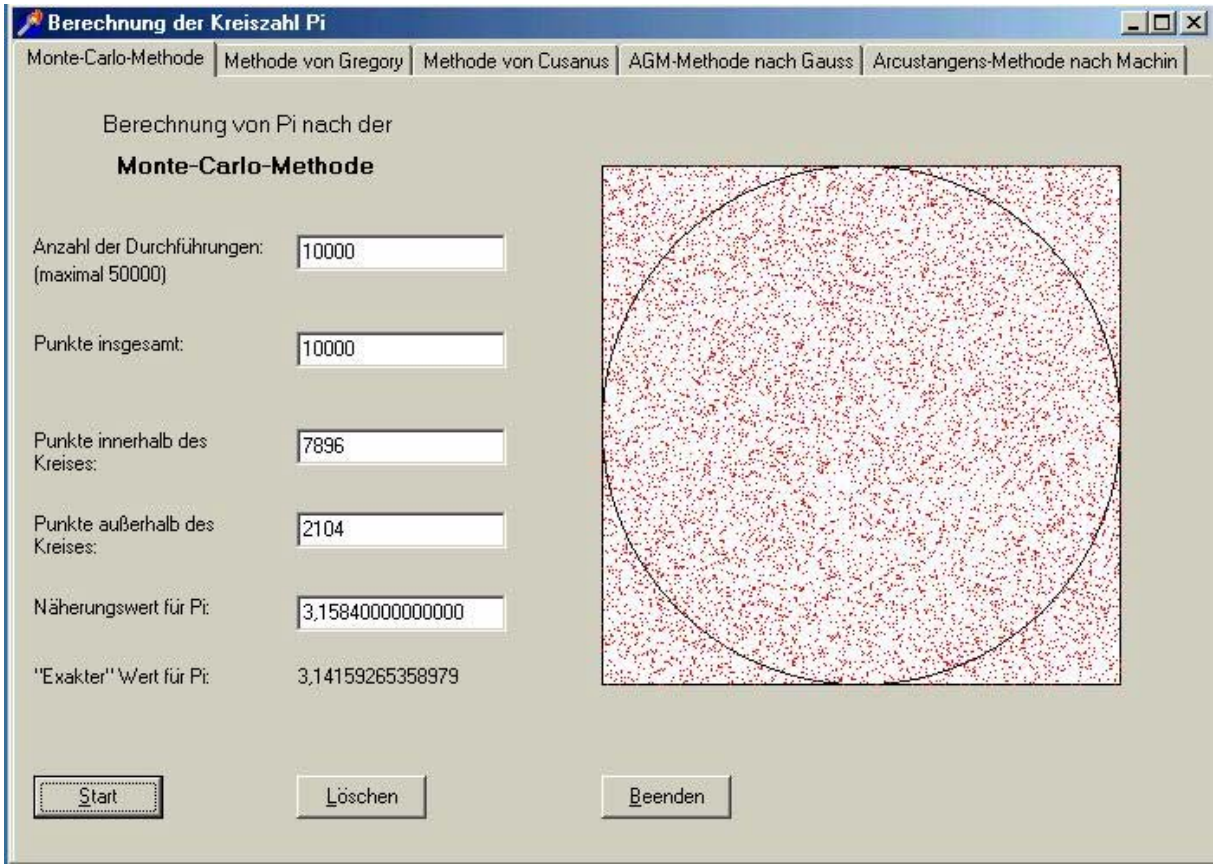
Beide Folgen a_k und b_k konvergieren quadratisch gegen den selben Wert AGM(a, b). Dieses ungemein schnelle Konvergenzverhalten überträgt sich auf die oben dargestellte Formel zur Berechnung von Pi, so dass bereits die ersten drei Iterationsschritte 19 genaue Stellen von Pi liefern.

Für Mathematiker ist sicherlich interessant, dass sich hinter der Pi-Formel von Gauß eine sehr anspruchsvolle mathematische Theorie verbirgt. Die Formel ging aus Untersuchungen von Gauß zu den lemniskatischen Funktionen und den elliptischen Integralen hervor. Genaueres hierzu siehe [1], S. 87 ff.

3. Erläuterung der verwendeten Unterprogramme

3.1. Monte – Carlo – Methode

Hinweise zur Bedienung der Programmoberfläche:



Im linken Teil des Fensters erfolgt zunächst die Eingabe der Anzahl der Durchführungen. Ausgegeben werden die insgesamt gesetzten Punkte, die Punkte innerhalb und die Punkte außerhalb des Kreises. Mit Hilfe der in Kapitel 2.1 angegebenen Gleichung erfolgt dann die Berechnung von Pi. Die bei der Berechnung erzielte Genauigkeit kann mit Hilfe des in Delphi implementierten Wertes für Pi verglichen werden. Das einzelne Setzen der Punkte wird in der Grafik im rechten Teil des Fensters veranschaulicht. Die Buttons *Start*, *Löschen* und *Beenden* steuern das Programm.

Hinweise zum Quelltext:

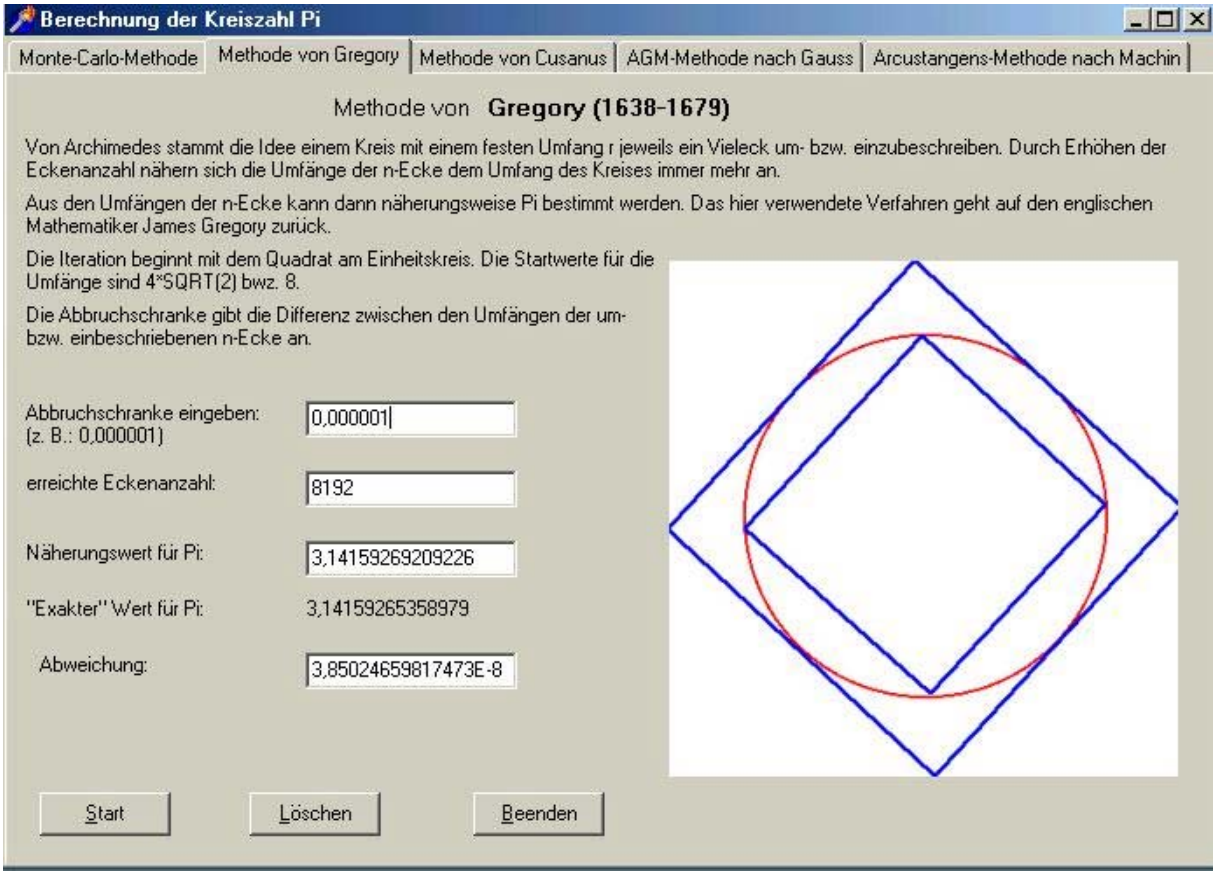
```
procedure TForm1.Button1Click(Sender: TObject);
var
  i, treffer, x, y, n: integer;
  Pi_wert: double;
begin
  n := StrToInt(Edit1.Text);          //Anzahl der Versuche einlesen
  if (n < 0) or (n > 50000) then      //unzulässige Eingabe abfangen
  begin
```

```
    ShowMessage('Unzulässige Versuchsanzahl!');
    Edit1.Clear;
    Exit;
  end  {if}
  else begin
    treffer := 0;                  // Zähler initialisieren
    Screen.Cursor := crHourGlass;      //Mauszeiger in Stundenglas verwandeln
    for i := 1 to n do
    begin
      x := Random(300);             //Zufallskoordinaten ermitteln
      y := Random(300);
      Image1.Canvas.Pixels[x,y] := clRed;    //Punkte einzeichnen
      if Power(x,2) + Power(y,2) <= Power(300,2) then
        inc(treffer);               //Zähler inkrementieren
    end;  {for i := 1 to n do}
    Pi_wert := 4 * treffer / n;     //Näherungswert für Pi berechnen
    //Ausgabe
    Edit2.Visible := True;
    Edit2.Text := IntToStr(n); //Gesamtanzahl der Punkte
    Edit3.Visible := True;
    Edit3.Text := IntToStr(treffer);     //Anzahl der Punkte im Kreis
    Edit4.Visible := True;
    Edit4.Text := IntToStr(n - treffer);//Anzahl der Punkte außerhalb des Kreises
    Edit5.Visible := True;
    Edit5.Text :=  FloatToStrF(Pi_wert,ffFixed,11,14);        //Näherungswert für Pi
  end; {else}
  Screen.Cursor := crDefault; //Mauszeiger zurücksetzen
end;
```

Bereits im Kapitel [2.1] wurde die Gleichung zur Berechnung des Näherungswertes von Pi mathematisch plausibel gemacht. Entscheidend dafür ob ein zufällig ermittelter Punkt innerhalb oder außerhalb des Kreises liegt, ist die Verwendung des Satzes von Pythagoras: Power(x,2) + Power(y,2) <= Power(300,2). Ist die Bedingung erfüllt, so liegt der ermittelte Punkt innerhalb des Kreises oder auf der Kreislinie und der entsprechende Zähler für die Treffer wird erhöht. Bei Nichterfüllung der Bedingung liegt der Punkt außerhalb des Kreises.
Die Initialisierung des Zufallszahlengenerators erfolgt automatisch beim Start des Programms in der Prozedur FormCreate.

3.2. Die Methode von Gregory

Hinweise zur Bedienung der Programmoberfläche:



Die entscheidende Idee der Methode von Gregory ist das Ein – bzw. Umbeschreiben von n – Ecken bezüglich eines vorgegebenen Kreises. Die im ersten Feld einzugebende Abbruchschranke bestimmt die Genauigkeit der Berechnung. Solange die Differenz der Umfänge der um – bzw. einbeschriebenen Kreise den hier angegebenen Wert überschreitet wird die Berechnung weiter fortgesetzt.
Ausgegeben werden die bei der Berechnung erreichte Eckenanzahl, sowie der berechnete Näherungswert für Pi und seine Abweichung vom Vergleichswert.

Hinweise zum Quelltext

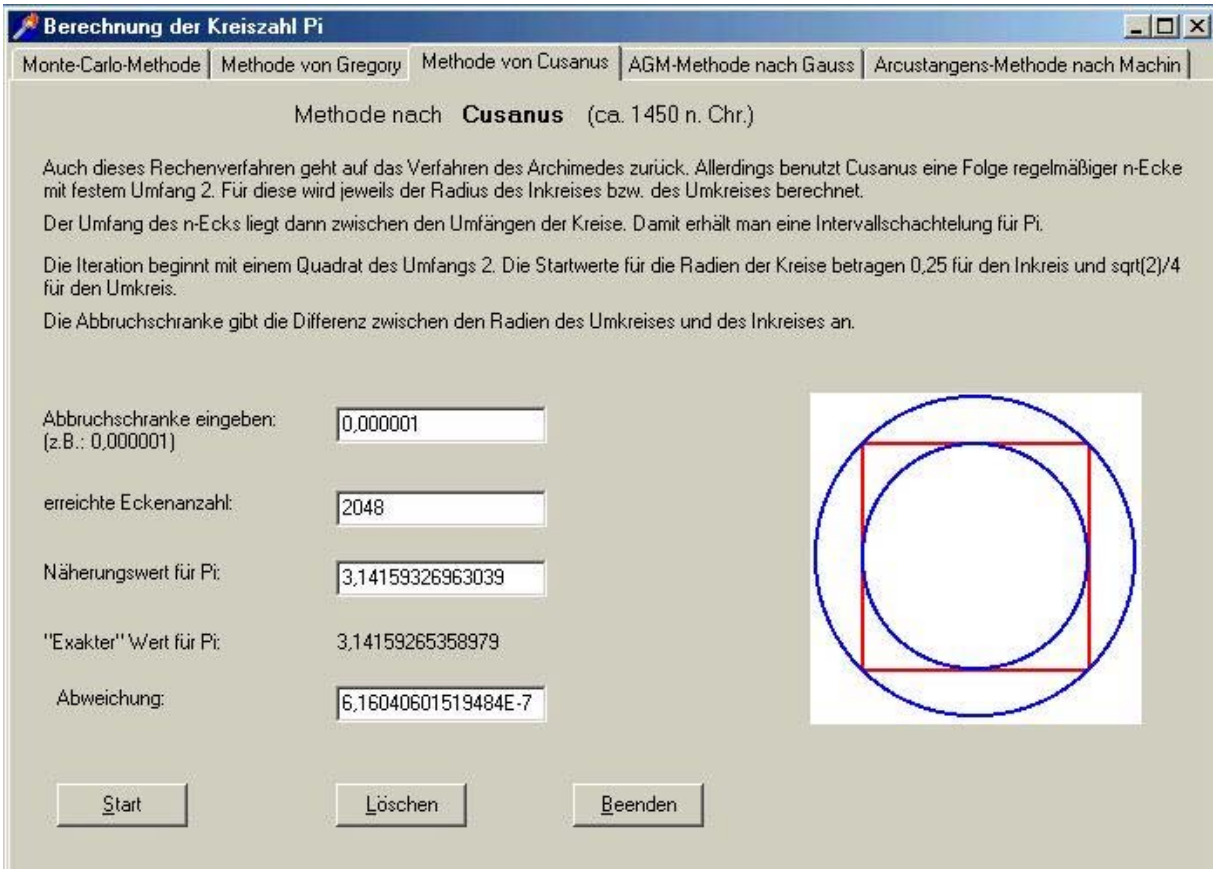
```
procedure TForm1.Button4Click(Sender: TObject);
var
  u, v, abbruch, Pi_u, Pi_v, MW_Pi, abweichg: extended;
  n: integer;
begin
  Screen.Cursor := crHourGlass;       //Mauszeiger in Stundenglas verwandeln
  Edit6.SetFocus;
  abbruch := StrToFloat(Edit6.Text);          //Abbruchbedingung einlesen
  //Startwerte
  n := 4;
  u := 4 * sqrt(2);
```

```
  v := 8;
  while ABS(v - u) > abbruch do       //Abbruchbedingung
  begin
    //Iteration
    v := 2 * u * v / (u + v);    //Umfang des umbeschriebenen n-Ecks
    u := sqrt(u * v);            //Umfang des einbeschriebenen n-Ecks
    n := 2 * n;                  //Eckenanzahl verdoppeln
  end; {while ABS(v - u) > abbruch do}
  //Pi berechnen
  Pi_u := u / 2;                        //untere Näherung für Pi
  Pi_v := v / 2;                        //obere Näherung für Pi
  MW_Pi := (Pi_u + Pi_v) / 2;           //Mittelwert der Näherung
  abweichg := ABS(Pi - MW_Pi);          //Abweichung vom Vergleichswert
  //Ausgabe
  Edit7.Visible := True;
  Edit7.Text := IntToStr(n);            //erreichte Eckenanzahl
  Edit8.Visible := True;
  Edit8.Text := FloatToStr(MW_Pi);          //Näherungswert für Pi
  Edit9.Visible := True;
  Edit9.Text := FloatToStr(abweichg);       //Abweichung vom Vergleichswert
  Screen.Cursor := crDefault;               //Mauszeiger zurücksetzen
end;
```

Die Iterationsgleichungen sowie die Startwerte wurden der in Kapitel 2.2.1 erwähnten Literatur entnommen. Die Berechnung von Pi erfolgt auch hier mit Hilfe der Formel $u = \pi * d$ ermittelt. Die Berechnung erfolgt am Einheitskreis mit dem Durchmesser $d = 2$. Daraus resultiert die Gleichung zur Berechnung von Pi : $\pi = u / 2$.
Zur Verbesserung der errechneten Näherung wird nach erfolgter Iteration der Mittelwert der unteren und der oberen Näherung für Pi gebildet.

3.3. Methode von Cusanus

Hinweise zur Bedienung der Programmoberfläche



Die Intention der von Cusanus verwendeten Methode zur Berechnung von Pi ist dem soeben beschriebenen Verfahren von Gregory sehr ähnlich. Im Unterschied zu Gregory variiert Cusanus die Radien der Umkreise und Inkreise bei konstantem Umfang 2 der n – Ecke. Der Aufbau der Programmoberfläche ist damit identisch zur im vorhergehenden Kapitel beschriebenen Oberfläche.

Hinweise zum Quelltext

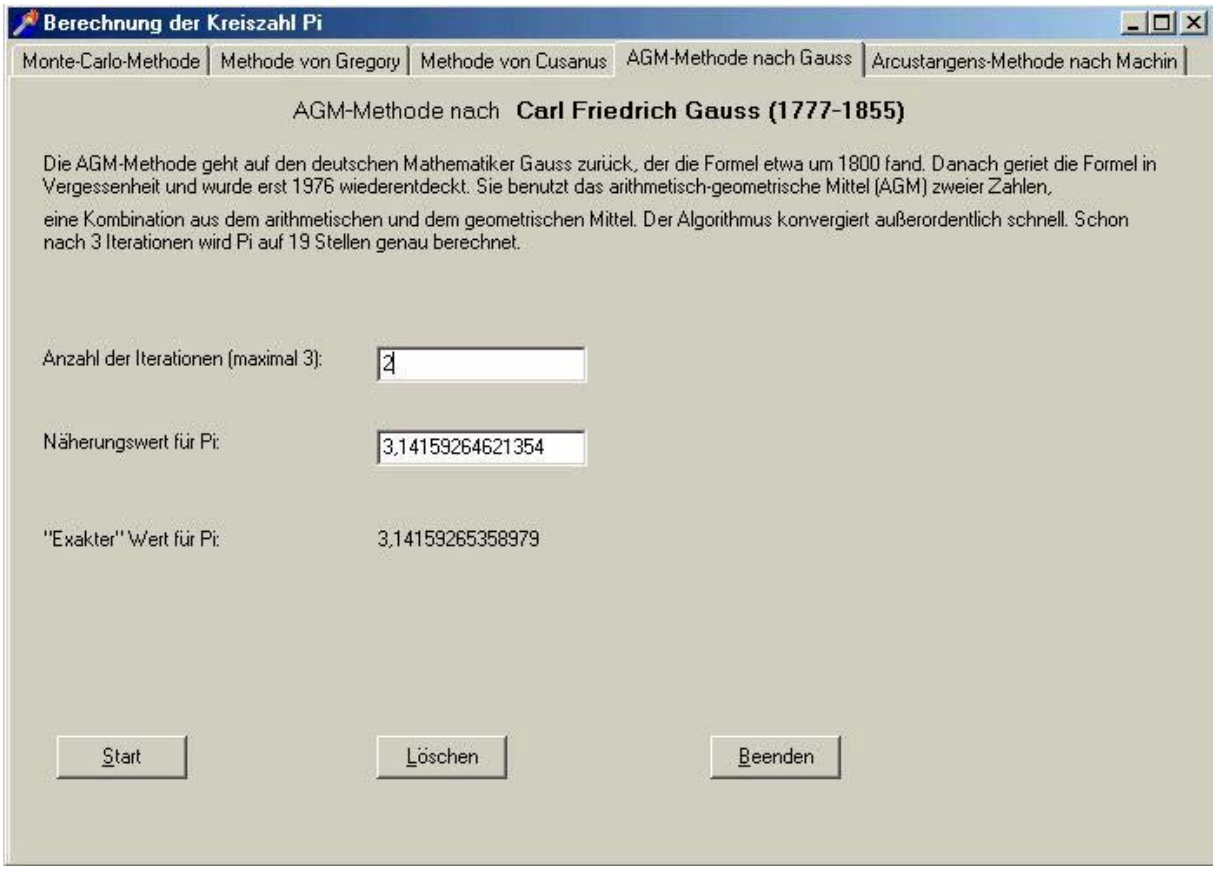
```
procedure TForm1.Button7Click(Sender: TObject);
var
  abbruch, inkr, umkr, Pi_inkr, Pi_umkr, MW_Pi: extended;
  n: integer;
begin
  Edit10.SetFocus;
  abbruch := StrToFloat(Edit10.Text);        //Abbruchschranke einlesen
  //Startwerte initialisieren
  n := 4;
  inkr := 1/4;
  umkr := sqrt(2)/4;
  Screen.Cursor := crHourGlass;              //Mauszeiger in Stundenglas verwandeln
  while (umkr - inkr) > abbruch do           //Abbruchbedingung
```

```
  begin
    //Iteration
    inkr := (umkr + inkr) / 2;       //Radius des Inkreises berechnen
    umkr := sqrt(umkr * inkr);       //Radius des Umkreises berechnen
    n := n * 2;                      //Eckenzahl verdoppeln
  end; {while (umkr - inkr) > abbruch do}
  //Pi berechnen
  Pi_inkr := 1 / inkr;
  Pi_umkr := 1 / umkr;
  MW_Pi := (Pi_inkr + Pi_umkr) / 2;
  //Ausgabe
  Edit11.Visible := True;
  Edit11.Text := IntToStr(n);                    //erreichte Eckenanzahl
  Edit12.Visible := True;
  Edit12.Text := FloatToStr(MW_Pi);              //Näherungswert für Pi
  Edit13.Visible := True;
  Edit13.Text := FloatToStr(ABS(MW_Pi - Pi));    //Abweichung zum Vergleichswert
  Screen.Cursor := crDefault;                    //Zurücksetzen des Mauszeigers
end;
```

Die Iterationsgleichungen wurden wiederum der in 2.2.2 angegebenen Literatur entnommen. Auch hier wird nach erfolgter Iteration noch einmal der Mittelwert der errechneten oberen und unteren Schranken für Pi berechnet. Dies soll die Genauigkeit der Näherungslösung noch einmal verbessern.

3.4. Die Methode von Gauß

Hinweise zur Bedienung der Oberfläche



Vom Benutzer ist lediglich die Anzahl der Iterationsschritte einzugeben. Ausgegeben wird der errechnete Näherungswert für Pi.

Hinweise zum Quelltext

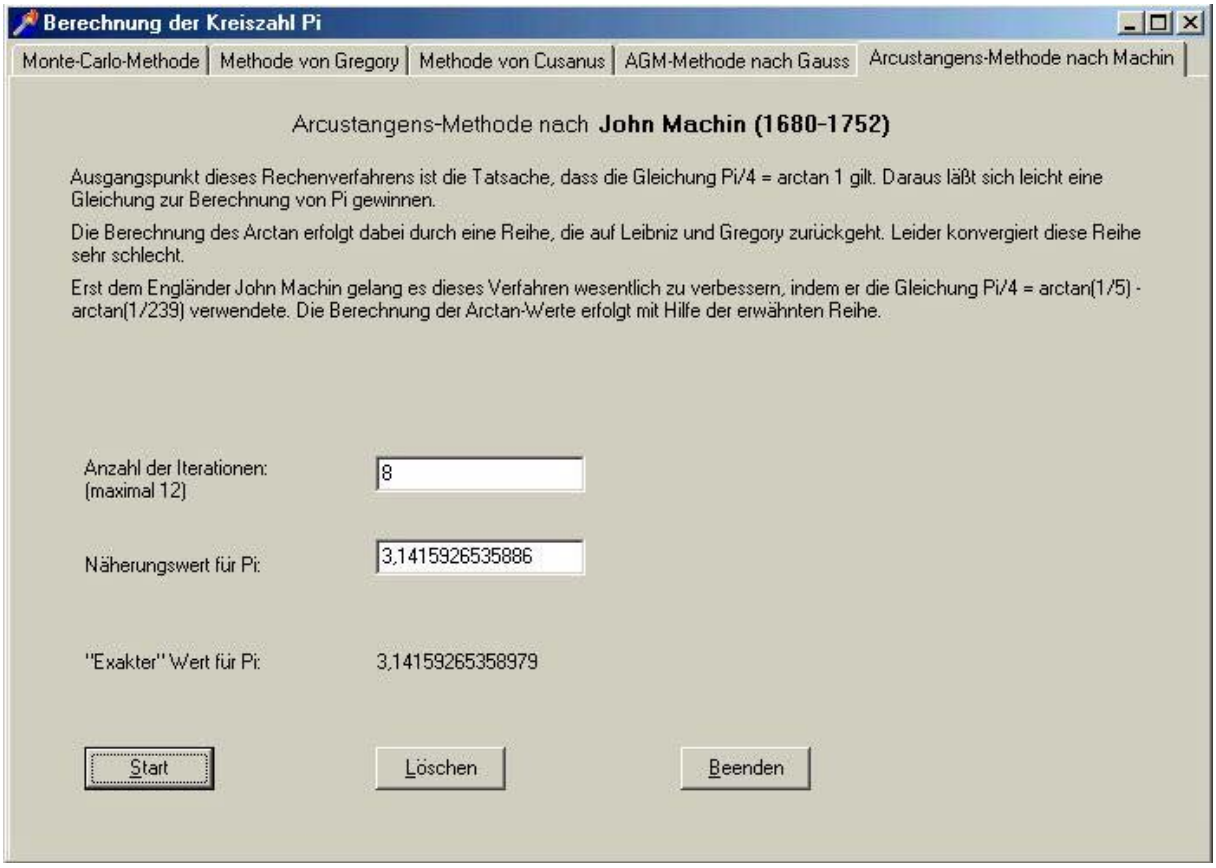
```
procedure TForm1.Button10Click(Sender: TObject);
var
  a, b, s, hilf, t, Pi_wert: extended;
  i, n: integer;
begin
  Edit14.SetFocus;
  n := StrToInt(Edit14.Text);
  if (n < 1) or (n > 3) then
  begin
    //Unzulässige Versuchsanzahl abfangen
    ShowMessage('Unzulässige Versuchsanzahl!');
    Exit;
  end {if}
  else begin
    //Startwerte initialisieren
    a := 1;
```

```
    b := 1 / sqrt(2);
    s := 1 / 2;
    for i := 1 to n do
    begin
      //Iteration
      hilf := a;
      a := (a + b) * 0.5;
      b := sqrt(hilf * b);
      t := Power((a - hilf),2);
      if t = 0 then exit;
      t := t * Power(2,i);
      s := s - t;
    end; {for i := 1 to n do}
    Pi_wert := Power((a + b),2) / (2 * s);
    //Ausgabe
    Edit15.Visible := True;
    Edit15.Text := FloatToStr(Pi_wert);
  end; {else}
end;
```

Die Methode von Gauß ist eine der schnellsten modernen Methoden zur Berechnung von Pi. Sie beruht auf dem bereits in Kapitel 2.4 beschriebenen arithmetisch – geometrischen Mittel. Der Algorithmus besitzt eine quadratische Konvergenz. So sind z.B. nur 36 Iterationsschritte notwendig, um Pi auf 68,7 Milliarden Stellen zu berechnen. Hier liegt allerdings auch das Problem des Algorithmus. Für große Stellenzahlen müssen die Variablen so groß wie das Ergebnis dimensioniert werden. Dafür ist ein wahrhaft gigantischer Speicheraufwand zu betreiben, denn nur spezielle Großrechner realisieren können. Außerdem sind Operationen mit sehr großen Zahlen erforderlich, für die besondere „Langzahl – Bibliotheken“ benötigt werden [1, S. 91 f.]. Aus diesem Grund wurde die Anzahl der Iterationsschritte für das hier vorliegende PC – Programm auf drei begrenzt.
Die Zeilen a := (a + b) * 0.5 und b := sqrt(hilf * b) dienen der Berechnung des AGM. Die nächsten Zeilen t := Power((a - hilf),2), t := t * Power(2,i) und s := s – t berechnen den Nenner der in Kapitel 2.4 angegebenen Gleichung. Die Zeile Pi_wert := Power((a + b),2) / (2 * s) berechnet schließlich den gewünschten Näherungswert für Pi.

3.5. Die Methode von Machin

Hinweise zur Bedienung der Oberfläche



Vom Benutzer sind wiederum nur die Anzahl der Iterationen einzugeben. Ausgegeben wird der berechnete Näherungswert für Pi.

Hinweise zum Quelltext

```
procedure TForm1.Button13Click(Sender: TObject);
const
  x1 = 1/5;         //Argumente des arctan
  x2 = 1/239;
var
  nenner1, nenner2 : Longint;
  a1, a2 : extended;      //Werte der arctan
  zaehler1, zaehler2, Pi_wert : extended;
  anz, i : integer;
begin
  //Anzahl der Iterationen
  anz := StrToInt(Edit17.Text);
  if (anz < 0) or (anz > 12) then
  begin
    ShowMessage('Unzulässige Versuchsanzahl!');
    Exit;
```

```
  end
  else begin
    //Startwerte initialisieren
    nenner1 := 1;
    nenner2 := 1;
    a1 := 0;
    a2 := 0;
    zaehler1 := x1;
    zaehler2 := x2;

    for i := 1 to anz do begin
      //Iteration
      a1 := a1 + zaehler1/nenner1;
      zaehler1 := -zaehler1*x1*x1;
      Inc(nenner1, 2);

      a2 := a2 + zaehler2/nenner2;
      zaehler2 := -zaehler2*x2*x2;
      Inc(nenner2, 2);
    end;  {for i := 1 to anz do begin}
    Pi_wert := 4*(a1*4-a2);
    //Ausgabe
    Edit16.Visible := True;
    Edit16.Text := FloatToStr(Pi_wert);
  end; {else}
end;
```

Der Algorithmus verwendet zur Berechnung von Pi eine Reihenentwicklung der Arcustangensfunktion (s. Kapitel 2.3). Nach der Initialisierung der Startwerte für die Reihenentwicklung werden zunächst die einzelnen Glieder der Reihe separat berechnet.

Die Zeilen a1 := a1 + zaehler1/nenner1 und zaehler1 := -zaehler1*x1*x1 berechnen $\arctan\frac{1}{5}$.

Die Berechnung von $\arctan\frac{1}{239}$ erfolgt in den Zeilen a2 := a2 + zaehler2/nenner2 und zaehler2 := -zaehler2*x2*x2. Der Näherungswert für Pi gemäß der Gleichung $\frac{\pi}{4} = 4 \cdot \arctan\frac{1}{5} - \arctan\frac{1}{239}$ erfolgt in der Zeile Pi_wert := 4*(a1*4-a2).

4. Hinweise auf Probleme und Erweiterungsmöglichkeiten des Programms

Aufgabe des Programms ist es, exemplarisch einige Berechnungsmöglichkeiten der Zahl Pi aufzuzeigen. Natürlich gibt es noch eine ganze Reihe weiterer Berechnungsmöglichkeiten, die hier nicht realisiert worden sind. Eine mögliche Erweiterung des Programms bestünde also darin, noch weitere Varianten der Berechnung von Pi zu implementieren.
Für den Benutzer wäre es außerdem sicherlich hilfreich, wenn eine Hilfe zur Bedienung und auch zum mathematischen Hintergrund des jeweils gewählten Algorithmus zur Verfügung stünde. Dort könnte man, ähnlich wie in dieser Dokumentation, ganz kurz und knapp wichtige Informationen an den Benutzer weitergeben. Allerdings hätte ich mich in die Programmierung

von Hilfen erst von Grund auf einarbeiten müssen. Dafür reichte leider die mir zur Verfügung stehende Zeit nicht mehr.
Ein Problem des vorliegenden Programms besteht darin, dass die Zahl der berechneten genauen Stellen von Pi, gemessen an den heute vorliegenden bekannten Stellenzahlen ziemlich gering erscheint. Zum Berechnen von Stellenzahlen in Größenordnungen von mehreren tausend oder noch mehr Stellen, sind jedoch spezielle Datentypen bzw. Großrechner erforderlich. Allerdings war es auch nicht die Zielsetzung dieses Programms auf „Weltrekordjagd“ in der Berechnung von Pi zu gehen.

5. Literaturverzeichnis

[1] Arndt, J., Haenel, C., *Pi – Algorithmen, Computer, Arithmetik*, Springer Verlag, Berlin, Heidelberg, 2000

[2] Lergenmüller, A, Schmidt, G., *Computer* Zusatzband, Sekundarstufe I, Ernst Klett Schulbuchverlag, Stuttgart, 1990

[3] Kreutzer, K.-H., Wiedemann, A., *Informatik*, Addita für die 9. und 10. Jahrgangsstufe der mathematisch – naturwissenschaftlichen Gymnasien, Ehrenwirth Verlag, München, 1994

[4] Barth, F., Haller, R., *Stochastik*, Leistungskurs, Ehrenwirth Verlag München, 1992

[5] Morandell, W., *Die Magie der Zahl Pi,* Online – Dokument bei `http://www.geocities.com/CollegePark/Hall/4866/`

[6] Tappe, S., *Die Geschichte von Pi*, Seminar Computeralgebra zur Berechnung von Pi, `http://www-math.uni-paderborn.de/~aggathen/vorl/2000ss/sem/`

[7] Altiparmak, D., *Methoden zur Berechnung der Kreiszahl Pi*, Bodensee – Gymnasium Lindau, 1999, Online – Dokument bei `http://www.astro.univie.ac.at/~wasi/PI/misc/altip/facharbeit.htm`

[8] Ebner, M., Klawun, C., *Go To Pascal mit Delphi 4*, Addison – Wesley – Longman Verlag, 1998