NUMERIK

Projekt 1

Name des Tutors: Johann Faschingleitner

Name der Autoren:

Amanda Schöfl

Matrikelnummer: e0271473

Christoph Mayer

Matrikelnummer: e01425430

Aufgabe 3

3.1 Aufgabenstellung

Basierend auf einer 1D Quadratur kann durch Bildung des Tensorproduktes eine 2D Quadratur erzeugt werden. Sei dazu $f : \mathbb{R}^2 \to \mathbb{R}$ eine auf dem Einheitsqudrat $Q := [0,1] \times [0,1]$ integrierbare Funktion, dann gilt (mittels Fubini)

$$\begin{aligned} \int_{\hat{Q}} f(x,y)d(x,y) &= \int_0^1 \Big(\int_0^1 f(x,y)dx \Big) dy &= \\ &= \int_0^1 \Big(\sum_{i=0}^{n} \alpha_i f(x_i,y) \Big) dy &= \\ &= \sum_{i=0}^{n} \alpha_i \int_0^1 f(x_i,y)dy &= \\ &= \sum_{i=0}^{n} \alpha_i \Big(\sum_{j=0}^{n} \alpha_j f(x_i,y_i) \Big) &= \sum_{i,j=0}^{n} \alpha_i \alpha_j f(x_i,y_j), \end{aligned} \tag{0.1}$$

wobei $\{\alpha_i : i = 0, ...n\}$ und $\{x_i, y_j : i, j = 0, ...n\}$ die 1D Quadraturgewichte und -knoten sind. Somit kann man mit den bereits bekannte 1D Quadraturen auch Integrale am Einheitsquadrat numerisch berechnen. Implementieren Sie mit Hilfe des zur Verfügung gestellten Programms **gauss(n)** eine Funktion **quadInt(f,n)** die $\int_{\hat{Q}} f(x,y)d(x,y)$ berechnet. Für welche n wird ein Polynom $p \in \prod_2^k := \text{span}\{x^i y^j : 0 \leq i, j \leq k\}$ exakt integriert?

3.1 Durchführung

Die folgende Implementierung der Funktion **quadInt**, hat die Inputparameter

- `f` ein Function Handle einer Funktion mit oben beschriebenen Eigenschaften,
- `n` der Grad der Gaußquadratur.

Sie gibt den approximierten Wert des Integrals der übergebenen Funktion auf $\hat{Q}$ zurück.

```
function ret_val = quadInt(f,n)
    [nodes, weights] = gauss(n);
    tmp = 0;
    for i = 1:n+1
        for j = 1:n+1
            tmp = tmp+weights(i)*weights(j)*f(nodes(i),nodes(j));
        end
    end
```

```
    ret_val = tmp;
end
```

Widmen wir uns jetzt noch der Frage für welche n ein Polynom $p \in \prod_2^k := \text{span}\{x^i y^j : 0 \leq i, j \leq k\}$ exakt integriert wird. Wir wissen, dass mittels 1D Quadraturen vom Grad n Polynome vom Grad $2n+1$ exakt numerisch berechnet werden können. Sei die in 0.1 vorkommende Funktion f nun unser Polynom p. Da zunächst nach x integriert wird, wirken die in p vorkommenden y bloß als reelle Koeffizienten. Somit kann p genau dann numerisch exakt nach x integriert werden, wenn das Polynom

$$x \mapsto p_y(x) := p(x, y)$$

für feste y höchstens vom Grad $2n+1$ ist. Analoges gilt für die Integration nach y. Damit kann p genau für $k \leq 2n+1$ exakt integriert werden.

3.2 Aufgabenstellung

Da zum Beispiel bei der Finite-Element-Methode eine Quadratur auf Dreiecken benötigt wird, verwendet man gerne die *Duffy-Transformation* um die Quadratur auf dem Einheitsquadrat $\hat{\text{Q}}$ auf das Referenzdreieck $\hat{\text{T}}$ mit den Eckpunkten $(0,0), (1,0)$ und $(0,1)$ zu transformieren. Die *Duffy-Transformation* ist definiert als:

$$\Psi = \begin{cases} \hat{Q} \to \hat{T} \\ (s,t) \mapsto (s, (1-s)t) \end{cases} \tag{0.2}$$

Implementieren Sie mithilfe der *Duffy-Transformation* die Funktion **trigInt(f,n)**, sodass Sie Integrale auf $\hat{\text{T}}$ berechnen können. *Hinweis*: Verwenden Sie dazu die Substitutionsregel Satz 7.34 (Transformationssatz für Integrale) aus dem Analysis-Skript von Professor Engl.

3.2 Durchführung

Transformationssatz für Integrale:
Es sei $\Omega \in \mathbb{R}$ eine offene Menge und $\Phi : \Omega \to \Phi(\Omega) \in \mathbb{R}^d$ ein Diffeomorphismus. Dann ist die Funktion f auf $\Phi(\Omega)$ genau dann integrierbar, wenn die Funktion $x \mapsto f(\Phi(x))|\det(D\Phi(x))|$ auf Ω integrierbar ist. In diesem Fall gilt:

$$\int_{\Phi(\Omega)} f(y)dy = \int_{\Omega} f(\Phi(x))|\det(D\Phi(x))|dx.$$

Dabei ist $D\Phi(x)$ die Jacobi-Matrix und $\det(D\Phi(x))$ die Funktionaldeterminante von Φ, also $Df(\Phi) = \left(\frac{\partial f_i}{\partial x_j}(x)\right)_{i,j=1,\ldots,n}$.

In unserem Fall sei $\Omega = (0,1) \times (0,1)$. Dies entspricht der Menge $\hat{Q}$ ohne Rand, welcher jedoch eine Nullmenge bezüglich des zweidimensionalen Lebesguemaß darstellt und somit

$\int_{(0,1)\times(0,1)} f(x)dx = \int_{\hat{Q}} f(x)dx$ gilt. Analog ist das Integral einer Funktion über $\hat{T}$ ident mit jenem über $\hat{T}$ ohne den Rand $\partial\hat{T}$. Für Φ wählen wir die Abbildung Ψ. Diese ist offenbar unendlich oft stetig differenzierbar, hat auf $\hat{T}\backslash\partial\hat{T}$ mit

$$\Psi^{-1} := \begin{cases} \hat{T}\backslash\partial\hat{T} & \to \hat{Q}\backslash\partial\hat{Q} \\ (x,y) & \mapsto (x, \frac{y}{(1-x)}) \end{cases}$$

eine überall stetig differenzierbare Umkehrabbildung. Ψ eingeschränkt auf $\hat{Q}\backslash\partial\hat{Q}$ ist folglich bijektiv und erfüllt damit alle Voraussetzungen für einen Diffeomorphismus. Die Funktionaldeterminante von Ψ ergibt sich gemäß

$$\det \Psi = |\det \begin{pmatrix} 1 & -t \\ 0 & (1-s) \end{pmatrix}| = |(1-s)|.$$

Nach dem Transformationssatz gilt folglich

$$\int_{\hat{T}} f(x,y)d(x,y) = \int_{\hat{Q}} f(s,(1-s)t)|(1-s)|d(t,s).$$

Das Rechte Integral lässt sich nun analog zur vorherigen Aufgabe berechnen.

Die folgende Implementierung der Funktion **trigInt**, hat die Inputparameter

- `f` ein Function Handle einer Funktion mit oben beschriebenen Eigenschaften auf dem Dreieck $\hat{T}$,
- `n` der Grad der Gaußquadratur.

Sie gibt den approximierten Wert des Integrals der Funktion f auf dem Gebiet $\hat{T}$ zurück.

```
--
function ret_val = quadInt(f,n)
    [nodes, weights] = gauss(n);
    function ret = f_n(s,t)
        x = s;
        y = t*(1-s);
        ret = abs(1-s)*f(x,y);
    end
    tmp = 0;
    for i = 1:n+1
        for j = 1:n+1
            tmp = tmp+weights(i)*weights(j)*f_n(nodes(i),nodes(j));
        end
    end
    ret_val = tmp;
end
```

3.3 Aufgabenstellung

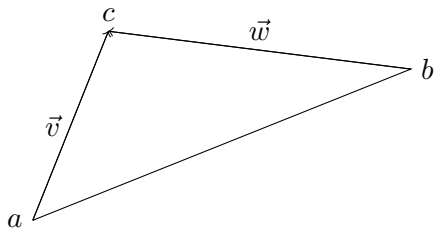
Erweitern Sie unter Verwendung einer affinen Transformation $\Phi_T : \hat{T} \to T$ Ihre Implementierung von **duffyInt**, um auch Integralen auf beliebigen Dreiecken T berechnen zu können. Für welche n wird ein Polynom $p \in \prod_2^k := \text{span}\{x^i y^j : 0 \leq i + j \leq k\}$ exakt integriert?

Hinweis: Sie können die Implementation Ihrer Transformation testen, indem Sie das Integral über $f \equiv 1$ numerisch berechnen und mit der analytisch bestimmten Fläche des Dreiecks T vergleichen.

3.3 Durchführung

Anstatt einer affinen Transformation von $\Phi_{\hat{T}} : \hat{T} \to T$ verwenden wir eine affine Transformation $\Phi : \hat{Q} \mapsto T$.

Sei nun T ein beliebiges Dreieck mit den Eckpunkten $a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$.



Die Fläche des Dreiecks ist nun genau die Konvexe Hülle der Eckpunkte. Um diese zu bilden, betrachten wir zunächst die Vektoren

$$v = \begin{pmatrix} (1-s)a_1 + sc_1 \\ (1-s)a_2 + sc_2 \end{pmatrix} \text{ und } w = \begin{pmatrix} (1-s)b_1 + sc_1 \\ (1-s)b_2 + sc_2 \end{pmatrix}.$$

Aus einer Kombination $(1-t)v + tw$ dieser Vektoren ergibt sich nun eine Parametrisierung Φ des Dreiecks gemäß

$$\Phi = \begin{cases} \hat{Q} \to T \\ (s,t) \mapsto \begin{pmatrix} a_1 + s(c_1 - a_1) + t(b_1 - a_1) + st(a_1 - b_1) \\ a_2 + s(c_2 - a_2) + t(b_2 - a_2) + st(a_2 - b_2) \end{pmatrix} \end{cases}.$$

Setzt man nun die Eckpunkte aus Aufgabe 3.2 ein, also

$$a = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, c = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

ergibt sich genau die Duffy-Transformation

$$(s,t) \mapsto \begin{pmatrix} t - st \\ s \end{pmatrix} = \begin{pmatrix} t(1-s) \\ s \end{pmatrix}. \tag{0.3}$$

Die Implementierung der Funktion erfolgt nun analog zur vorherigen Aufgabe. Die Funktion **duffyInt**, hat die Inputparameter

- `f` ein Function Handle einer Funktion mit oben beschriebenen Eigenschaften,
- `n` der Grad der Gaußquadratur.
- a_1, a_2 : Koordinaten des Punktes a
- b_1, b_2 : Koordinaten des Punktes b
- c_1, c_2 : Koordinaten des Punktes c

Sie gibt den approximierten Wert des Integrals nach der Transformation auf ein beliebiges Dreieck mit den Eckpunkten (a,b,c) zurück.

```
function ret_val = duffyInt(f,n,a1,a2,b1,b2,c1,c2)
    [nodes, weights] = gauss(n);
    function ret = f_n(s,t)
        x = a1*s*t-b1*s*t-a1*s-a1*t+b1*t+c1*s+a1;
        y = a2*s*t-b2*s*t-a2*s-a2*t+b2*t+c2*s+a2;
        ret = abs(-a1*b2*s+a1*c2*s+a2*b1*s-a2*c1*s-b1*c2*s+b2*c1*s+a1*b2...
            -a1*c2-a2*b1+a2*c1+b1*c2-b2*c1)*f(x,y);
    end
    tmp = 0;
    for i = 1:n+1
        for j = 1:n+1
            tmp = tmp+weights(i)*weights(j)*f_n(nodes(i),nodes(j));
        end
    end
    ret_val = tmp;
end
```

Widmen wir uns nun noch der Frage wann ein Polynom $p \in \prod_2^k := \text{span}\{x^i y^j : 0 \leq i+j \leq k\}$ exakt integriert werden kann.Da p eine Linearkombination von Polynomen der Gestalt $(x,y) \mapsto x^i y^j$ ist, reicht es das Polynom $q(x,y) = x^i y^j$ zu betrachten. Unter der Transformation Φ wird q zu

$$q(\Phi(x), \Phi(y)) = (\alpha_1 + s\alpha_2 + t\alpha_3 + st\alpha_4)^i (\beta_1 + s\beta_2 + t\beta_3 + st\beta_4)^j.$$

Durch Ausmultiplizieren wird ersichtlich, dass es sich dabei um ein Polynom vom Grad $i+j$ in s und t handelt. Das heißt der Term mit den höchsten Potenzen von s und t ist von der Gestalt $cs^i t^i s^j t^j = cs^{(i+j)} t^{(i+j)}$ mit einer reellen Konstanten c. Für die Integration über T wird dieser Ausdruck nun noch mit dem Betrag der Funktionaldeterminante multipliziert. Darin kommen s und t jeweils linear vor. Der Summand mit den höchsten Potenzen ist also von Gestalt $\tilde{c}st$ mit einer reellen Konstanten $\tilde{c}$. Insgesamt ergibt sich, für den Integranden von

$$\int_{\hat{Q}} p(\Phi(x), \Phi(y)) \det |d\Phi| d(s,t)$$

ein Polynom in s und t, welches jeweils vom Grad $i+j+1$ ist. Auf Grund vorher bereits angestellter Überlegungen, wissen wir, dass dies genau für $i+j+1 \leq 2n+1$ mittels einer 2D-Quadratur vom Grad n exakt numerisch integriert werden kann. Insgesamt gilt also, dass für $k \leq 2n$ eine exakte numerische Integration gegeben ist.

3.4 Aufgabenstellung

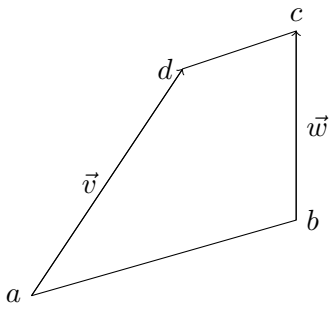
Erweitern Sie Ihre Implementierung von **quadInt(f,n)** zur Berechnung von Integralen auf beliebigen konvexen Vierecken $Q = \text{conv}\,(a1, a2), (b1, b2), (c1, c2), (d1, d2)$. Verwenden Sie dazu eine eine Transformation der Form:

$$\Psi_{\hat{Q}} = \begin{cases} \hat{Q} \to Q \\ (s,t) \mapsto \begin{pmatrix} \alpha_1 + \alpha_2 s + \alpha_3 t + \alpha_4 st \\ \beta_1 + \beta_2 s + \beta_3 t + \beta_4 st \end{pmatrix} \end{cases}$$

mit den Konstanten α_i,$\beta_i \in \mathbb{R}$. Überlegen Sie sich, warum nicht-konvexe Vierecke unzulässig sind.

3.4 Durchführung

Sei Q nun ein beliebiges konvexes Viereck mit den Eckpunkten $a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$, $b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$, $c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$, $d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$.



Die Fläche des Vierecks geht nun aus einer zu Aufgabe 3.3 analogen Konvexkombination $\vec{v}(1-t) + \vec{w}t$ aus den Vektoren $\vec{v} = a(1-s) + ds$ und $\vec{w} = b(1-s) + cs$ hervor. Aus der Tatsache, dass die Transformation aus der Angabe äquivalent zu einer Konvexkombination der Eckpunkte des Vierecks ist, geht auch hervor, dass nicht konvexe Vielecke damit nicht abgebildet werden können.

Nun ergeben sich die Konstanten α_i, β_i entweder durch das Ausmultiplizieren der obigen Konvexkombination oder dem Lösen eines Gleichungssystems, wenn man davon ausgeht dass die Eckpunkte von $\hat{Q}$ jeweils auf die Eckpunkte von Q abgebildet werden.

Nun kann die Funktion **quadInt2(f)** analog zur Funktion duffyInt(f) aus der Aufgabe 3.2 implementiert werden. Sie hat die Inputparamenter

- `f` ein Function Handle einer Funktion mit oben beschriebenen Eigenschaften,
- `n` der Grad der Gaußquadratur,
- a_1, a_2 Koordinaten des Punktes a
- b_1, b_2 Koordinaten des Punktes b
- c_1, c_2 Koordinaten des Punktes c
- d_1, d_2 Koordinaten des Punktes d.

Zurückgegeben wird der approximierte Wert des Integrals der Funktion f über das Viereck Q.

```
function ret_val = quadInt2(f,n,a1,a2,b1,b2,c1,c2,d1,d2)
    [nodes, weights] = gauss(n);
    function ret = f_n(s,t)
        x = (d1-b1+a1)*s*t+(c1-a1)*s+(b1-a1)*t+a1;
        y = (d2-b2+a2)*s*t+(c2-a2)*s+(b2-a2)*t+a2;
        ret = abs(a1*b2*s-a1*c2*s-a1*d2*s+a1*d2*t-a2*b1*s+a2*c1*s+a2*d1*s ...
            -a2*d1*t+b1*c2*s-b1*d2*t-b2*c1*s+b2*d1*t+c1*d2*s-c2*d1*s-a1*b2 ...
            +a1*c2+a2*b1-a2*c1-b1*c2+b2*c1)*f(x,y);
    end
    tmp = 0;
    for i = 1:n+1
        for j = 1:n+1
            tmp = tmp+weights(i)*weights(j)*f_n(nodes(i),nodes(j));
        end
    end
    ret_val = tmp;
end
```

3.5 Aufgabenstellung

Testen Sie ihre Quadraturen für $\int_G f(x,y)\,d(x,y)$:

- $f(x,y) = x^7 + 3x^4y^4 + 3x^2y + 7y^6, G = \text{conv}\{(0,0),(0,5,-0,5),(1,1)\}$
  Stimmt das benötigte n um das Polynom exakt zu integrieren mit den theoretischen Überlegen zusammen? Wenn nicht, wieso?
- $f(x,y) = \sin(50x)\sin(50y), G = \text{conv}\{(1,0),(10.2),(3,4)(-1,1)\})$
- $f(x,y) = \sqrt{\frac{9y}{(1-x^2)}}, G = \hat{Q}$

- $f(x,y) = \begin{cases} 1, x^2 + y^2 \leq 1 \\ 0, sonst \end{cases}, G = \mathbb{R}^2$

  Hinweis: Approximieren Sie den Kreis durch ein n-Eck.

Verwenden Sie die letzten beiden Beispiele um π zu approximieren. Nehmen Sie als Referenzwert bei allen Integralen den analytisch bestimmten Integralwert, sofern dies möglich ist und stellen Sie die Fehler grafisch dar.

3.5 Durchführung

Für die Berechnung der Integrale der ersten beiden Funktionen benutzen wir Maple, welches in der Lage ist analytisch zu integrieren. Für die Abschätzung der Konvergenz des Fehlers suchen wir jeweils eine Funktion aus $O(g(n))$ welche ein ähnliches Konvergenzverhalten zeigt. Weiters ist nur der Absolutbetrag des Fehlers dargestellt.

Sei zunächst $f_1(x,y) = x^7 + 3x^4y^4 + 3x^2y + 7y^6$ und $G_1 = \text{conv}\{(0,0), (0,5, -0,5), (1,1)\}$. Nun benutzen wir die vorher gefundene Transformation und berechnen analytisch das Integral. Führt man folgenden Code aus, erhält man den, bis auf 10 Kommastellen genauen, Wert von 0.2877808780.

```
v1 := c1·s + (1 − s)·a1;
v2 := c2·s + (1 − s)·a2;
w1 := c1·s + (1 − s)·b1;
w2 := c2·s + (1 − s)·b2;
d1 := simplify(t·w1 + (1 − t)·v1);
d2 := simplify(t·w2 + (1 − t)·v2);
with(LinearAlgebra);
md := Matrix([[diff(d1, t), diff(d1, s)], [diff(d2, t), diff(d2, s)]]);
det := simplify(Determinant(md));
a1 := 0;
a2 := 0;
b1 := 0.5;
b2 := −0.5;
c1 := 1;
c2 := 1;
simplify(int(int((d1^7 + 3·d1^4·d2^4 + 3·d1^2·d2 + 7·d2^6)·det, s = 0..1), t = 0..1))
```

Anschließend benutzen wir die vorher implementierte Funktion **duffyInt** um das Integral numerisch zu berechnen und schätzen den Fehler ab.

```
% calculate numerical integration
f=@(x,y) x^7+3*x^4*y^4+3*x^2*y+7*y^6;
n_max = 10;
integral = zeros(1,n_max+1);
for n = 1:n_max+1
    integral(n) = duffyInt(f,n-1,0,0,0.5,-0.5,1,1);
end

% calculate error and reference function O
error = abs(integral-0.2877808780)
O = error;
for n=1:4
    O(n+1) = O(n)/(2^((n-1)^2.4));
end
for n=5:n_max
    O(n+1) = 0;
end

% plot error and reference function O
hold off
loglog(0:n_max,abs(error))
xlabel('n')
ylabel('Fehler')
hold on
loglog(0:n_max, O)
legend('error','O(2^{-n(n-1)^{2.4}})')
```

Aus dem Plot des Fehlers in Abbildung 1 geht hervor, dass der numerische Wert des Integrals ab $n = 4$ mit dem analytischen Wert (bis zur berechneten Kommastelle) überein stimmt. Das ist insofern bemerkenswert, da $f \in \text{span}\{x^i y^j : 0 \leq i + j \leq 13\}$. Jedoch ergibt sich unter der Transformation als Integrand nur ein Polynom (in s und t) vom Grad 9, welches wie bereits vorher bemerkt mit einer 2D Quadratur vom Grad 4 exakt berechnet werden kann.

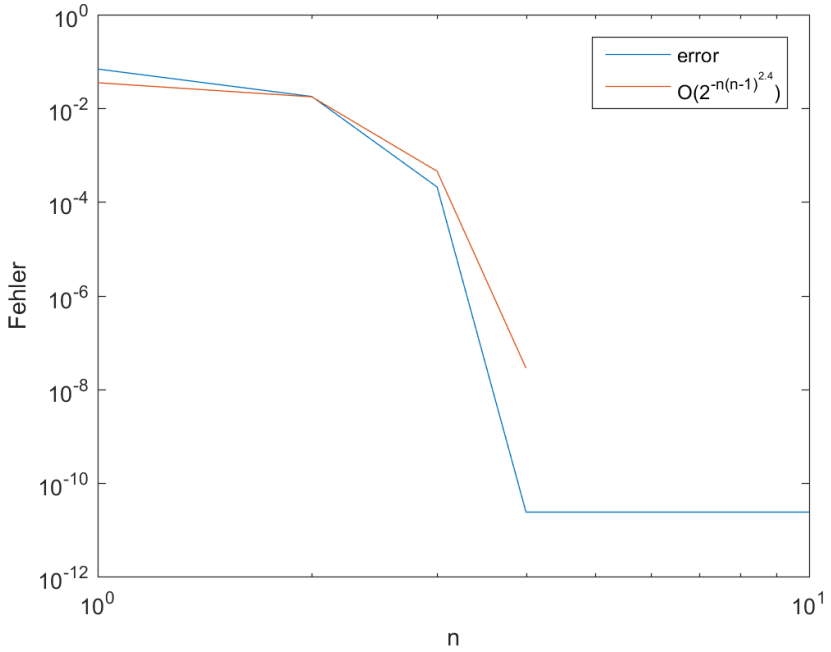


Abbildung 1: Darstellung des Fehlers der numerischen Integration von f_1 über G_1 in einem log-log Plot

Wir betrachten als nächstes die Funktion $f_2(x, y) = \sin(50x)\sin(50y)$ auf dem Gebiet $G_2 = \text{conv}\{(1, 0), (10.2), (3, 4)(-1, 1)\})$. Analog zu vorher berechnen wir auch hier mit Hilfe von Mapel analytisch das Integral.

```
alpha1 := a1;
alpha2 := c1 − a1;
alpha3 := b1 − a1;
alpha4 := d1 − b1 + a1;
beta1 := a2;
beta2 := c2 − a2;
beta3 := b2 − a2;
beta4 := d2 − b2 + a2;
with(LinearAlgebra);
q1 := alpha1 + alpha2·s + alpha3·t + alpha4·s·t;
q2 := beta1 + beta2·s + beta3·t + beta4·s·t;
mq := Matrix([[diff(q1, s), diff(q1, t)], [diff(q2, s), diff(q2, t)]]);
det := simplify(Determinant(mq));
a1 := 1;
a2 := 0;
b1 := 10;
b2 := 2;
c1 := 3;
c2 := 4;
d1 := −1;
d2 := 1;
simplify(int(int((sin(50·s)·sin(50 t)·det, s = 0 ..1 ), t = 0 ..1))
```

Das Ausführen des obigen Codeausschnitts ergibt den Term $(-8/625+(37/2500)*\cos(50)+(27/125000)*\sin(50)-(27/125000)*\cos(50)*\sin(50)-(1/500)*\cos(50)^2)$ was in etwa dem Wert von 0.0003 entspricht.

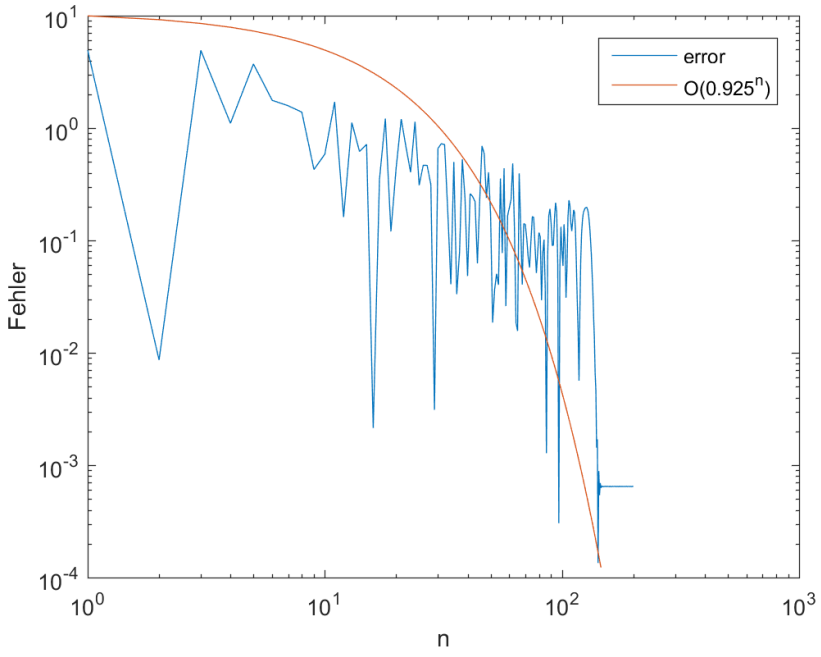


Abbildung 2: Darstellung des Fehlers der numerischen Integration von f_2 über G_2 in einem log-log Plot

Anschließend verwenden wir die Funktion **quadInt2**, um den numerischen Wert des Integrals zu berechnen. Da es sich diesmal nicht um ein Polynom handelt, nähert sich der numerische Wert des Integrals zwar immer mehr dem analytischen Wert an, erreicht diesen jedoch nie. Ab einem Grad von $n = 147$ entspricht der numerische Wert bis auf vier Nachkommastellen dem analytischen. Es zeigt sich weiters, dass der Trend der Konvergenz einer Funktion aus $O(0.95^n)$ ähnelt. Dieser Zusammenhang ist in Abbildung 2 dargestellt.

```
% calculate numerical integration
f=@(x,y) sin(50*x)*sin(50*y);
n_max = 200;
integral = zeros(1,n_max+1);
for n = 1:n_max+1
    integral(n) = quadInt2(f,n-1,1,0,10,2,3,4,-1,1);
end

% calculate error and reference function O
error = abs(integral-(-8/625+(37/2500)*cos(50)+(27/125000)*sin(50)-...
    (27/125000)*cos(50)*sin(50)-(1/500)*cos(50)^2));
O = error;
for n=1:147
    O(n+1) = O(n)*0.925;
end
for n=147:n_max
    O(n+1) = 0;
end

% plot error and reference function O
hold off
loglog(0:n_max,error)
hold on
xlabel('n')
ylabel('Fehler')
loglog(0:n_max, O)
legend('error','O(0.925^n)')
```

Für die Funktion $f_3(x,y) = \sqrt{\frac{9y}{(1-x^2)}}$ wissen wir bereits, dass ihr Integral über $G_3 = \hat{Q}$ gegen π konvergiert. Zur Berechnung des numerischen Integralwerts verwenden wir diesmal die Funktion **quadInt**.

```
% calculate numerical integration
f=@(x,y) sqrt(9*y/(1-x^2));
n_max = 600;
integral = zeros(1,n_max);
for n = 1:n_max
    integral(n) = quadInt(f,n);
end

% calculate error and reference function O
error = abs(integral-pi);
O = error;
for n=1:n_max
    O(n) = n^(-0.85)*O(1)
end

% plot error and reference function O
hold off
loglog(1:n_max,error)
hold on
loglog(1:n_max,O)
xlabel('n')
ylabel('Fehler')
legend('error','O(n^{-0.85})')
```

Aus Abbildung 3 wird ersichtlich, dass der numerische Wert nur sehr langsam gegen den tatsächlichen konvergiert. Der Fehler verhält sich dabei etwa wie eine Funktion aus $O(n^\alpha)$ mit einem $\alpha \in (-1,-0.5)$.

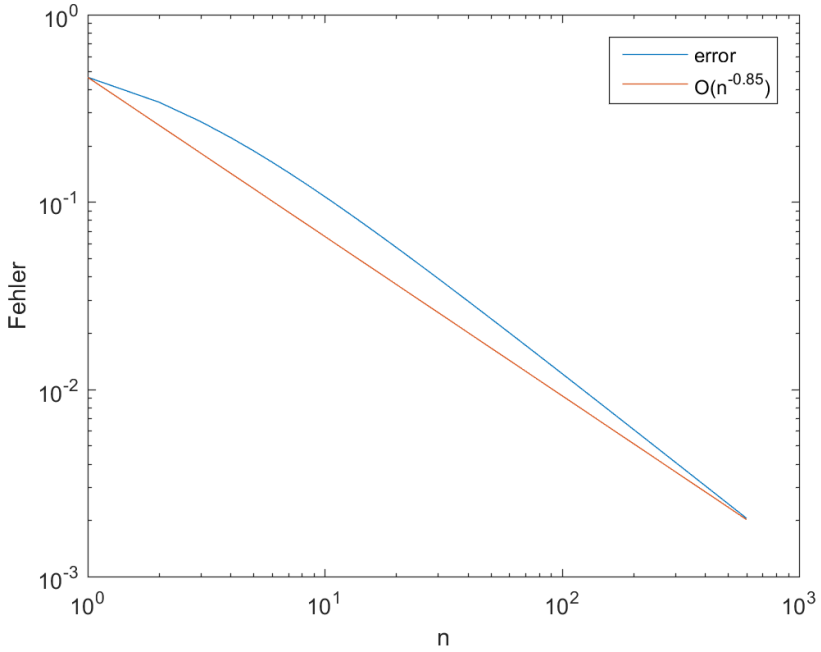
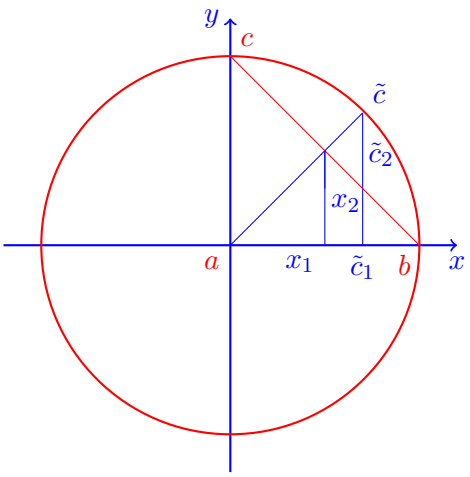


Abbildung 3: Darstellung des Fehlers der numerischen Integration von f_3 über G_3 in einem log-log Plot

Um die Fläche des Einheitskreises numerisch zu berechnen, approximieren wir diesen mit jeweils 2^n Dreiecken, benutzen eine Gaußquadratur vom Grad 0, um das Integral der konstanten Einsfunktion über einem der Dreiecke und multiplizieren dieses Ergebnis mit 2^n. Im folgenden wird kurz eine Möglichkeit skizziert wie man rekursiv auf die richtigen Eckpunkte eines der Dreiecke kommt.
Betrachten wir dazu folgende Skizze. Wir gehen von dem Dreieck, welches von den Punkten a, b und c aufgespannt wird aus und wollen uns den Punkt $\tilde{c} = \begin{pmatrix} \tilde{c_1} \\ \tilde{c_2} \end{pmatrix}$ berechnen.



Durch den Satz von Pythagoras ergibt sich

$$x = (\frac{a_1 + b_1}{2}, \frac{a_2 + b_2}{2}).$$

Sei nun γ der Winkel zwischen den Vektoren $\tilde{c} - a$ und $b - a$. Dann ist $\tilde{c_1} = \cos(\gamma)$ und $\tilde{c_2} = \sin(\gamma)$. Damit folgt

$$tan\gamma = \frac{x_2}{x_1} = \frac{\sin\gamma}{\cos\gamma} = \frac{\tilde{(c_2)}}{\tilde{(c_1)}} \tag{0.4}$$

Wir wissen aus

$$\sin(x)^2 + \cos(x)^2 = 1,$$

dass

$$\tilde{c_1}^2 + \tilde{c_2}^2 = 1.$$

Durch Lösen dieses Gleichungssystems ergibt sich

$$\tilde{c_2} = \frac{c_2}{c_1 + 1}\sqrt{\frac{1}{(1 + \frac{c_2^2}{(c_1+1)^2}}} \text{ und } \tilde{c_1} = \sqrt{\frac{1}{(1 + \frac{c_2^2}{(c_1+1)^2}}}.$$

Dieses Verfahren wird nun iterativ angewandt um beliebig kleine Dreiecke zu bekommen. Nun kann die Funktion **duffyInt** verwendet werden, um die Fläche des Einheitskreises zu approximieren.

```
% calculate numerical integration
f=@(x,y) 1;
n_max = 7; % 2^n_max ... maximale Anzahl der Ecken
integral = zeros(1,n_max-1);
c1=0;
c2=1;
for n = 2:n_max
    integral(n-1) = 2^n*duffyInt(f,0,0,0,1,0,c1,c2);
    c1_tmp=sqrt(1/(1+(c2^2/(c1+1)^2)));
    c2=c2/(c1+1)*sqrt(1/(1+(c2^2/(c1+1)^2)));
    c1 = c1_tmp;
end

% calculate error and reference function O
error = abs(integral-pi);
O = error;
for n=2:n_max-1
    O(n) = (2^n)^(-2)*O(1)
end

% plot error and reference function O
x_ax = [2,4,8,16,32,64];
hold off
loglog(x_ax,error)
hold on
loglog(x_ax,O)
xlabel('Anzahl an Ecken zur Approximation')
ylabel('Fehler')
legend('error','O(n^{-2})')
```

Aus Abbildung 4 wird ersichtlich, dass dieses Verfahren π zu berechnen wesentlich schneller zum Ziel führt. Der Fehler entwickelt sich mit steigender Anzahl der Ecken wie eine Funktion aus $O(n^{-2})$ ein weitere Vorteil dieser Variante ist, dass nur Polynome nullten Grades berechnet werden müssen.

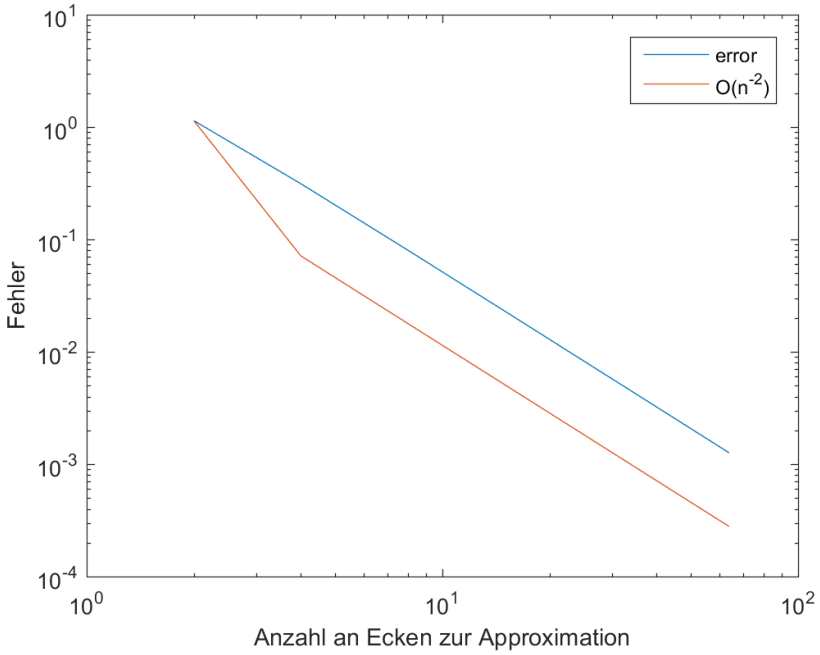


Abbildung 4: Darstellung des Fehlers der numerischen Berechnung der Fläche des Einheitskreises über Approximation mittels Dreiecken