1. Zeige, dass die Funktion $f : V \to W$ genau dann affin ist, wenn es eine lineare Funktion $l : V \to W$ und ein $a \in W$ gibt, sodass $f(v) = l(v) + a$.

2. Gib eine allgemeine multiaffine Abbildung $a : \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ an.

   1. Welche Voraussetzung müssen die Koeffizienten von a erfüllen, damit a multilinear ist?

   2. Welche Voraussetzung müssen die Koeffizienten von a erfüllen, damit a symmetrisch ist?

   3. Wann sind die elementarsymmetrischen Polynome S_k multilinear?

3. Bestimme

   1. $\#\{I : I \subseteq \{1, \ldots, n\}, \#I = k\}$ durch kombinatorische Überlegungen.

   2. $\#\{I : I \subseteq \{1, \ldots, n\}\}$ mit Hilfe des binomischen Lehrsatzes und dem Resultat aus (a).

   3. $$\sum_{k=0}^{n} (-1)^k \binom{n}{k}.$$

4. Gib zu dem Polynom $f(x) = 2x^3 - 3x^2 + 5x + 1$ die zugehörige Polarform F an. Gib weiters die Werte $F(0, 0, 0)$, $F(0, 0, 1)$ und $F(0, 1, 1)$ an.

5. Zerlege das Polynom $p(t) = t^4 - 2t^3 + 2$ in Bernsteinpolynome.

6. Zeige, dass das Maximum des Bernsteinpolynoms $B_i^n(t)$ mit $i \in \mathbb{N}_0, n \in \mathbb{N}, i \leq n$ im Intervall $[0, 1]$ an der Stelle $\frac{i}{n}$ liegt.

7. Zeige, dass die Linearkombination von Polarformen von Polynomen die Polarform der Linearkombination der Polynome ist.

8. Installiere PYTHON und eine Entwicklungsumgebung auf deinem Laptop/Computer.

Für die Beispiele 9 *bis* 12 *könnt ihr die Vorlage* Uebungsbeispiele.txt *verwenden. Ändert nach dem Herunterladen die Dateiendung auf* Uebungsbeispiele.py.

9. Schreibe in PYTHON eine Funktion, die Kugelkoordinaten in Kartesische Koordinaten umrechnet und ausgibt.

10. Schreibe in PYTHON eine Funktion `mysqrt(x, y)`, die eine Quadratwurzel aus der komplexen Zahl $x + iy$ berechnet, ohne komplexe Zahlen in PYTHON zu verwenden.

11. Schreibe in PYTHON eine Funktion `outside(x, y, z)`, die feststellt, ob der Punkt (x, y, z) außerhalb der Einheitskugel liegt, oder nicht.

12. Schreibe in PYTHON eine Funktion `inset(x)`, die der folgenden mathematischen Definition entspricht:

$$\texttt{inset(x)} = f(x) = \begin{cases} 1 & x \in \bigcup_{k\in\mathbb{Z}}[2k, 2k+1) \\ -1 & x \in \bigcup_{k\in\mathbb{Z}}[2k+1, 2k+2) \end{cases}$$

wobei $x \in \mathbb{R}$.

13. Schreibe ein Pythonskript, das die Summe aller ungeraden Zahlen von 1 bis n berechnet ohne dabei eine Schleife zu verwenden.

14. Die Fibnoacci Zahlen sind definiert als

$$F_1 = F_2 = 1$$
$$F_n = F_{n-1} + F_{n-1}$$

Schreibe eine Funktion `fibo(n)`, die die n-te Fibnoacci-Zahl zurück gibt.

15. Schreibe eine Funktion `fibo_sum(n)`, die die Summe der ersten n Fibnoacci-Zahlen für $n \geq 2$ zurück gibt.

16. Zeige: Für $b_i^j(t)$ im Algorithmus von deCasteljau gilt

$$b_i^j(t) = F(\underbrace{u,\ldots,u}_{n-i-j},\underbrace{t,\ldots,t}_{j},\underbrace{v,\ldots,v}_{i}).$$

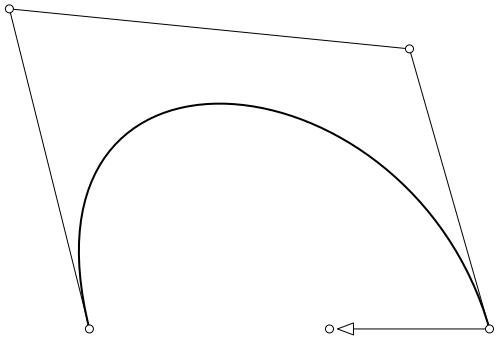
17. Konstruiere in *Geogebra* jeweils eine Bézierkurve vom Grad 2, 3, 4.

18. Finde eine Parameterdarstellung einer Parabel durch die Punkte $(0,0)$, $(1,0)$, $(0,2)$ und $(1,4)$.

19. Finde eine Parameterdarstellung einer Parabel durch die Punkte $(0,0)$, $(1,0)$, $(0,2)$ und $(2,4)$. Es darf PYTHON zur Hilfe genommen werden.

20. Die durch die drei Punkte $b_0 = (3,9)$, $b_1 = (0,0)$, $b_2 = (15,0)$ definierte Bézierkurve ist eine Parabel. Bestimme von dieser Parabel Achse und Scheitel.

21.



Gegeben ist eine kubische Bézierkurve $b^3(t)$ samt Kontrollpolygon. Unterteile $b^3(t)$ mit Hilfe der Unterteilungseigenschaft im Punkt $b^3(\frac{1}{2})$ in zwei Bézierkurven, eine linke c und eine rechte Bézierkurve d.
Modifiziere die rechte Bézierkurve d, indem der letzte Kontrollpunkt an die angegebene Stelle verschoben wird.

22. Zeige, dass ein Kreis nicht als Bézierkurve darstellbar ist.

23. Erzeuge in PYTHON eine Funktion `bez(t, p)`, die als Eingabe $t \in \mathbb{R}$ und ein Array $p = (b_0,\ldots,b_n)$ übernimmt und den Kurvenpunkt $b^n(t)$ der Bézierkurve zu den Kontrollpunkten $b_0,\ldots,b_n$ zurückgibt. Dabei soll der Punkt mit der Formel für die Parameterdarstellung berechnet werden. Teste die Funktion!

24. Implementiere in PYTHON eine Funktion `casteljau(t, p)`, die als Eingabe $t \in \mathbb{R}$ und ein Array $p = (b_0,\ldots,b_n)$ übernimmt und den Kurvenpunkt $b^n(t)$ der Bézierkurve zu den Kontrollpunkten $b_0,\ldots,b_n$ zurückgibt. Dabei soll der Kurvenpunkt über den Algorithmus von de Casteljau bestimmt werden. Teste die Funktion!

25. Gegeben sei die Parabel mit Gleichung $y = 2(x-1)^2 + 1$. Stelle die Parabel als Bézierkurve für $x \in [-1,1]$ dar.

26. Gegeben sei eine Polynomfunktion $f : \mathbb{R} \to \mathbb{R}$ mit $f(t) = t^3 - t^2 + 3t - 1$. Finde die Kontrollpunkte um den Graph $\{(t, f(t)) \in \mathbb{R}^2 | t \in [0,1]\}$ als Bézierkurve in $\mathbb{R}^2$ darstellen zu können.

27. Gegeben sei eine Polynomfunktion $f : \mathbb{R} \to \mathbb{R}$ vom Grad n. Finde die Kontrollpunkte um den Graph $\{(t, f(t)) \in \mathbb{R}^2 | t \in [0, 1]\}$ als Bézierkurve in $\mathbb{R}^2$ darstellen zu können.

28. Betrachte die Bézierkurve zu den Kontrollpunkten $b_0 = (-2, 0)$, $b_1 = (-2, 3)$, $b_2 = (2, 3)$, $b_3 = (2, 0)$ sowie den Kreisring $\sqrt{2} \leq \|x\| \leq 3$. Zeige, dass die Bézierkurve im Intervall $[0, 1]$ im Inneren des Kreisringes liegt.

29. Zeige, dass es keine Bezierkurve zu dem Graph $f(t) = (t, \mathrm{e}^t)$ mit $t \in [0, 1]$ geben kann.

30. Zeige, dass für die elementarsymmetrischen Funktionen gilt

$$S_k(X_1, \ldots, X_{n-1}, a) = S_k(X_1, \ldots, X_{n-1}) + aS_{k-1}(X_1, \ldots, X_{n-1}).$$

31. Finde eine Menge $M \subset \mathbb{R}^2$, sodass es keine Punkte $p_1, \ldots, p_n \in M$ gibt mit

$$\mathrm{conv}(M) = \mathrm{conv}(\{p_1, \ldots, p_n\}).$$

32. Zeige: Aus $A \subseteq B \subseteq \mathbb{R}^n$ folgt $\mathrm{conv}(A) \subseteq \mathrm{conv}(B)$.

33. Zeige, dass eine affine Abbildung eine konvexe Menge auf eine konvexe Menge abbildet.

34. Zeige, dass wenn das affine Bild einer Menge M konvex ist, muss die Menge M selbst nicht konvex sein.

35. Schreibe eine Funktion `element(p1,p2,p3,q)` in PYTHON, die bestimmt, ob der Punkt q in der konvexen Hülle von $p1$, $p2$ und $p3$ liegt.

36. Zeige, dass für die Funktionen N_k^i aus der Vorlesung gilt

$$N_i^{n-(r-1)} = (1 - \alpha_{i+1}^r) N_{i+1}^{n-r} + \alpha_i^r N_i^{n-r}.$$

37. Sei $m = 4, n = 2$ und der Knotenvektor $T = (0, 0, 0, 1, 2, 3, 3, 3)$. Skizziere die B-Spline Funktionen $N_4^0(t)$, $N_4^1(t)$, $N_4^2(t)$.

38. Konstruiere eine Bézierkurve mit einem singulären Punkt.

39. Zeige, dass der Grundriss einer räumlichen Bézierkurve eine ebene Bézierkurve ist.

40. Gegeben sei der Knotenvektor $T = (0, 0, 1, 2, 3, 4, 5, 5)$ und $n = 2$. Konstruiere zu geeigneten Kontrollpunkten die quadratische B-Splinekurve.

Die *Reliefperspektive* ist definiert im reellen projektiven Raum $P^3 = P(\mathbb{R}^4)$. Sei dazu $z\mathbb{R} \in P^3$ ein Punkt, $\mathbb{R}a \subset P^3$ eine Hyperebene und die Funktion

$$\begin{aligned} f : \mathbb{R}^4 &\longrightarrow \mathbb{R}^4 \\ x &\longmapsto f(x) = x + (c-1)\frac{\langle x, a\rangle}{\langle z, a\rangle} z \end{aligned}$$

gegeben, wobei $c \in (0, 1)$. Die Reliefperspektive ist dann gegeben durch

$$\begin{aligned} \kappa : P^3 &\longrightarrow P^3 \\ x\mathbb{R} &\longmapsto \kappa(x\mathbb{R}) = f(x)\mathbb{R}. \end{aligned}$$

41. Zeige, dass f bijektiv und linear ist.

42. Zeige, dass das *Zentrum* $z\mathbb{R}$ und die Punkte der *Fixpunktebene* $\mathbb{R}a$ fest bleiben.

43. Implementiere in Python eine Funktion, die als Eingabe das Zentrum, die Hyperebene, $c \in (0, 1)$ und einen eigentlichen Punkt $x \in \mathbb{R}^3$ übernimmt und als Ausgabe $\kappa(x\mathbb{R})$ zurückgibt, wenn dies kein Fernpunkt ist.

44. Implementiere in Python eine Klasse `myC`, die den Realteil und Imaginärteil einer Komplexen Zahl beinhaltet. Erstelle in dieser Klasse eine Methode, die den Absolutbetrag zurück gibt.

45. Erweitere in Python die Klasse `myC`, sodass man komplexe Zahlen addieren und subtrahieren kann, indem du die Methoden `__add__(self,other)` und `__sub__(self,other)` implementierst. Erweitere die Klasse außerdem um eine Methode, die den Winkel zur positiven reellen Achse zurück gibt.

46. Erweitere in Python die Klasse `myC`, sodass man komplexe Zahlen mit komplexen Zahlen und mit floats multiplizieren kann, indem du die Methoden `__mul__(self,other)` und `__rmul__(self,const)` implentierst. Erweitere deine Klasse außerdem um eine Methode `inv(self)`, die die komplexe Zahl invertiert.

47. Implementiere in Python eine Klasse `myVec`, die die x, y, z Koordinaten eines dreidimensionalen Vektors speichern kann. Als Standardwerte soll der Nullvektor verwendet werden. Füge der Klasse `myVec` die Memberfunktion `len` hinzu, die die Länge des Vektors zurückgibt.

48. Implementiere die skalare Multiplikation und das Skalarprodukt für die Klasse `myVec` aus Beispiel 47.

49. Implementiere in PYTHON eine Klasse `BeziKurve`, die die Punkte des Kontrollpolygons beinhaltet. Schreibe eine Methode für diese Klasse, die den Wert der Bézierkurve zum Parameter t zurück gibt.

50. Erweitere die Klasse `BeziKurve` aus Beispiel 49 mit einer Methode `add(self,point)`, die einen Punkt zum Kontrollpolygon hinzufügt.

51. Erweitere die Klasse `BeziKurve` aus Beispiel 49 mit einer Methode `plot(self)`, die die Bézierkurve plottet.

52. Erweitere die Klasse `BeziKurve` aus Beispiel 49 mit einer Methode `aff`, die eine affine Transformation des Kontrollpolygons durchführt.

53. Illustriere in PYTHON eine Bézierkurve vom Grad 5. Illustriere in derselben Grafik NURBS Kurven mit den gleichen Kontrollpunkten zu verschiedenen Gewichten.

54. Illustriere in PYTHON einen ganzen Kreis durch mehrere NURBS Kurven.

55. Konstruiere mit Zirkel und Lineal in Grund- und Aufriss eine Tensorprodukt Bézierfläche vom Grad $1,1$ mit $b_{0,0} = (6,0,5)$ $b_{1,0} = (12,3,2)$ $b_{0,1} = (3,12,2)$ $b_{1,1} = (9,15,8)$. Maße in cm.

56. Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine stetig differenzierbare Funktion. Zeige, dass aus $\nabla f(a) = 0$ nicht notwendigerweise folgt, dass a ein Extremum von f ist.

57. Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine stetige Funktion. Zeige, dass f nicht notwendigerweise ein globales Maximum und ein globales Minimum besitzen muss.

58. Bestimme zum Punkt $p = (a, \frac{1}{2})$ jenen Punkt auf der Kurve $\{(x,y) \in \mathbb{R}^2 \mid x^2 - y = 0\}$ mit dem kürzesten Abstand zu p.

59. Welche der folgenden Mengen ist ein Vektorraum?

$$\left\{\boldsymbol{x} \in \mathbb{R}^2 : \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} \cdot \boldsymbol{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right\}, \quad \{(x,y,z)^T \in R^3 : x \leq y \leq z\}, \quad \{(x,y,z)^T \in \mathbb{R}^3 : x^2 + y^2 + z^2 = 0\}$$

60. Welche der folgenden Mengen ist ein Vektorraum? Begründe deine Aussage! (Hier steht $D[0,1]$ für den Raum der einmal stetig differenzierbaren Funktionen auf dem Intervall $[0,1]$ und $C[0,1]$ für die stetigen Funktionen auf $[0,1]$. Beides sind Vektorräume. Man muss also nur das Unterraumkriterium nachprüfen.)

$$\{f \in D[0,1] : f'(x) = 0\}, \quad \{f \in D[0,1] : f'(0) = \pi\}, \quad \left\{f \in C[0,1] : \int_0^1 f(x)\,\mathrm{dx} = f(0) \cdot f(1)\right\}$$

61. Berechne den Gradienten $\nabla(x^T M x)$, wobei $x \in \mathbb{R}^n$ und $M \in \mathbb{R}^{n \times n}$. Verwende dazu:

(a) den Gradient der Funktion $(x^T b)$, wobei $b \in \mathbb{R}^n$,

(b) die Produktregel: $\nabla(x^T M x) = \nabla_y(y^T M x) + \nabla_y(x^T M y)$.

62. Lege eine Ausgleichsgerade $y = kx + d$ durch die 2D Punktwolke mit Punkten (x_i, y_i) in der Datei,

`NoisyData1.txt`

die die Summe der quadrierten Abstände $\sum \|kx_i + d - y_i\|^2$ minimiert. Visualisiere die Daten mit der Ausgleichsgeraden.

Anleitung:

1. Ihr importiert die Punkte aus der Datei:

   (a) Speichert die Datei im gleichen Ordner, wie euren PYTHON-Code

(b) Legt eine leere Liste an: `vertices = []`

(c) Öffnet die Datei mit: `Import = open('NoisyData1.txt','r')`

(d) Lasst eine for-Schleife über `Import` laufen und fügt die Koordinaten jeder Zeile zu eurer Liste hinzu. (Genauso wie im Beispiel aus den Unterlagen...)

2. Jetzt müsst ihr noch aus der Liste mit *Strings* einen `np.array` mit *floats* machen. Dazu verwendet ihr den Befehl `vertices = np.array(vertices).astype(float)`.
3. Jetzt könnt ihr euch die Matrix A und den Vektor b aus der Vorlesung erzeugen. Wenn ihr das habt könnt ihr A^TA und A^Tb berechnen durch `A.transpose() @ A` und `A.transpose() @ b`.
4. Ein lineares LGS könnt ihr in PYTHON einfach lösen mit `np.linalg.solve(Matrix,RechteSeite)`.

63. Lege eine quadratische Kurve $y = ax^2 + bx + c$ durch die 2D Punktwolke mit Punkten (x_i, y_i) in der Datei,

`NoisyData2.txt`

die die Summe der quadrierten Abstände $\sum \|ax_i^2 + bx_i + c - y_i\|^2$ minimiert. Visualisiere die Daten mit der quadratischen Kurve.

64. Berechne den Normalvektor der Fläche $x^2 + y^2 - z = 1$ im Punkt $(2, 2, z)$.

65. Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine stetig differenzierbare Funktion. Die Ableitung in Richtung $\boldsymbol{a}$ an der Stelle $\boldsymbol{x}$ (wir schreiben dafür $\nabla_{\boldsymbol{a}} f$) kann berechnet werden als

$$\nabla_{\boldsymbol{a}} f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) \cdot \boldsymbol{a}.$$

Maximiere den Betrag der Richtungsableitung unter der Nebenbedingung $\|\boldsymbol{a}\|^2 = 1$.

66. Reinhold geht in dem Funktionsgebirge $f(x, y) = (x-2)^4 + (y+2)^2$ spazieren. Als er sich im Punkt $(4, -3)$ befindet, verliert er seinen Ball. In welche Richtung rollt der Ball? Wo findet Reinhold seinen Ball wieder?

67. Sei $g : \mathbb{R}^2 \to \mathbb{R}$ eine stetig differenzierbare Funktion und $M = \{x \in \mathbb{R}^2 : g(x) = c\}$ nicht leer. Beweise, dass für alle $x \in M$ der Gradient $\nabla g(x)$ normal auf M steht. Ein Beweis könnte in etwa so aussehen:

Der Gradient erfüllt die Bedingung der linearen Approximation der Funktion g. Das heißt

$$\lim_{x \to x_0} \left(\frac{g(x) - \big(\nabla g(x_0) \cdot (x - x_0) + g(x_0)\big)}{\|x - x_0\|} \right) = 0.$$

(Das darf einfach verwendet werden.)

Zu jedem Tangentialvektor t am Punkt x_0 kann man eine Folge $(x_n) \subseteq M$ angeben, die gegen x_0 konvergiert, sodass

$$t = \lim_{n \to \infty} \frac{x_n - x_0}{\|x_n - x_0\|}.$$

Diese beiden Tatsachen müssen jetzt geschickt kombiniert werden um zu zeigen, dass für alle Tangentialvektoren t gilt, dass $\nabla g(x_0) \cdot t = 0$.

68. Lege eine Ausgleichsgerade durch die Punkte `NoisyData1.txt`, die die quadrierten Normalabstände zur Punktwolke minimiert.

69. Finde eine Ebene, die die Normalabstände zur Punktwolke `NoisyEbene.txt` minimiert.

70. Bestimme das Paraboloid

$$f(x, y) = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2,$$

das die quadrierten Abstände in z-Richtung

$$\sum_{i=1}^{n} \|f(x_i, y_i) - z_i\|^2$$

zur Punktwolke `NoisyParaboloid.txt` minimiert.

71. Verwende PYTHON um auf der Fläche $\Phi : x_1^2 + 3x_2^2 + 2x_3^2 + 2x_1x_2 - 1 = 0$ einen Punkt zu finden, der die Funktion $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + x_3^2 - 2x_2x_3$ minimiert.

72. Verwende PYTHON um auf der Fläche $\Phi : x_1^2 - 3x_2^2 + 2x_3^2 + 2x_1x_2 - 1 = 0$ einen Punkt zu finden, der die Funktion $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + x_3^2 - 2x_2x_3$ minimiert.

73. Approximiere in PYTHON die $2D$ Daten `NoisyCircle.txt` mit einem Kreis. Visualisiere dein Ergebnis. Wie wird bei dieser Aufgabe der Abstand zum Kreis geschätzt? Welche Annahmen müssen gelten damit die Berechnung des Abstands so funktioniert? Wie genau sieht das Minimierungsproblem aus?

74. Registrierung mit bekannten Korrespondenzen: Seien P, Q zwei Polygone mit

$P = ((0,0),(0,1),(-1,1),(-1,-1),(1,-1),(1,0))$
$Q = ((0,0),(0,1),(-2,1),(-2,-1),(2,-1),(2,0))$

Finde die affine Abbildung $\alpha(x) = Rx + a$ mit $R \in \mathbb{R}^{2\times 2}$, $a \in \mathbb{R}^2$, sodass

$$\sum_{i=1}^{6} \|\alpha(p_i) - q_i\|^2 \to \min.$$

75. Betrachte $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ und $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ in $\mathbb{R}^2$, mit

$$\mathbf{x}_k := \begin{pmatrix} \cos(k2\pi/N) \\ \sin(k2\pi/N) \end{pmatrix}$$

und

$$\mathbf{y}_k := 5\cos(k2\pi/N)\begin{pmatrix} 1 \\ 2 \end{pmatrix} + 2\sin(k2\pi/N)\begin{pmatrix} -3 \\ 1 \end{pmatrix} + \begin{pmatrix} r_k \\ \tilde{r}_k \end{pmatrix},$$

wobei $r_k, \tilde{r}_k$ Zufallszahlen aus dem Intervall $[-0.3, 0.3)$ sind. Finde die Matrix $R \in \mathbb{R}^{2\times 2}$ und den Vektor $\mathbf{a} \in \mathbb{R}^2$, der das affine Registrierungsproblem

$$\sum_{k=1}^{N} \|\mathbf{x}'_k - \mathbf{y}_k\|^2 \to min,$$

mit $\mathbf{x}'_k = \mathbf{a} + R\ \mathbf{x_k}$ löst.

76. Sei $(p_1, \ldots, p_N)$ ein Polygon mit $p_i \in \mathbb{R}^2$ und sei $q \in \mathbb{R}^n$. Finde und implementiere einen Algorithmus der einen Punkt des Polygons (gegeben als Liste seiner Eckpunkte) zurückgibt, der am nächsten bei q liegt.

77. Verwende den Algorithmus, aus Beispiel 76 um einen iterativen Algorithmus zu implementieren, der das affine Registrierungsproblem für zwei Polygone ohne Korrespondenzen löst. Illustriere den Algorithmus mit einem geeigneten Beispiel.

78. Schreibe einen iterativen Algorithmus, der eine robuste Regressionsgerade durch die Punktwolke `NoisyDataWithOutliers.txt` legt.
Was ist die Idee hinter einer robusten Regression? Welche Eigenschaften sollte eine Gewichtsfunktion haben? Was ist eine sinnvolle Abbruchsbedingung bei einer robusten Regression? Gib k und d der Regressionsgerade durch die ursprüngliche Punktwolke (also nicht die in den Nullpunkt verschobene) an.

79. Schreibe eine Klasse `Triangle`, die von `Mesh` erbt und als Initiation ein Dreieck anlegt.

80. Schreibe eine Methode `translate(Vector)` für die Klasse `Mesh`, die das ganze Mesh um den Vektor `Vector` verschiebt.

81. Schreibe eine Methode `add_point(self, point, bd_edge)` für die Klasse `Mesh`, die ein Dreieck bestehend aus der Randkante und dem neuen Punkt zum Mesh hinzufügt. Der Algorithmus soll auch überprüfen, ob es sich beim Input tatsächlich um eine Randkante handelt.

82. Erkläre die Methode `plot()` der Klasse `Mesh` in der Datei `HalfedgeDataStructure.txt`. (Vergesst nach dem Herunterladen nicht die Dateiendung auf `.py` zu ändern.)

83. Berechne die Gauß-Krümmung und Mittlere-Krümmung der parametrisierten Fläche:

$$f : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(u, v) \mapsto (v\cos(u), v\sin(u), u).$$

Ist die Parametrisierung konjugiert?

84. Für die Helix-Kurve $\gamma(t) := (\cos(t), sin(t), t), \quad t \in \mathbb{R}$, definieren wir die Tangentialfläche

$$f : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(t, s) \mapsto \gamma(t) + s\gamma'(t)$$

1. Für welche Punkte $(t, s) \in \mathbb{R}^2$ is f regulär?

2. Zeige, dass für jeden Punkt (t, s) eine Gerade existiert, die in der Fläche enthalten ist.

3. Zeige, dass f die 1-Parameter Familie von Ebenen

$$E_t := \{(x, y, z) \in \mathbb{R}^3 | \, x\sin(t) - y\cos(t) + z - t = 0\}$$

einhüllt.

85. Finde eine Parametrisierung für die Fläche, die von der 1-Parameter Familie von Ebenen E_t eingehüllt wird und fertigt eine Skizze der einhüllenden Fläche.

1. $E_t := \{(x, y, z) \in \mathbb{R}^3 | \, x\cos(t) + y\sin(t) - 1 = 0\}$.

2. $E_t := \{(x, y, z) \in \mathbb{R}^3 | \, x\cos(\alpha)\cos(t) + y\cos(\alpha)\sin(t) + z\sin(\alpha) - 1 = 0\}$, für ein festes $\alpha. \in (0, \frac{\pi}{2})$.

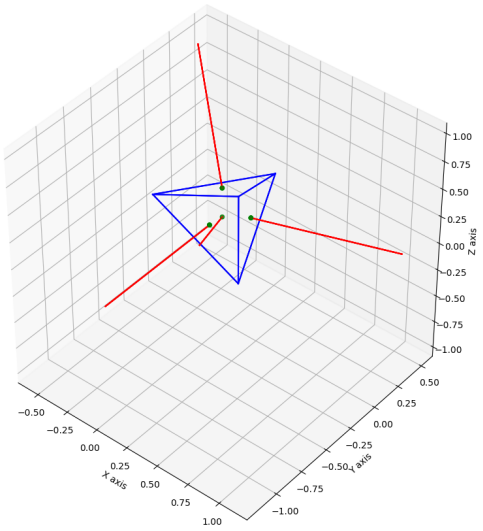
3. An welchen Punkten sind die Flächen regulär?

86. Schreibe in Python eine Methode für die Klasse Faces, welche die Anzahl der Randkanten eines Faces berechnet und eine weitere Methode die testet ob ein Face eben ist.

87. Schreibt in Python eine Methode in der Klasse Faces, die testet ob ein Face eben ist und für jedes ebene Face den Schwerpunkt berechnet. Erweitert die Plotmethode in der Klasse Mesh, so dass die Schwerpunkte ebener Faces visualisiert werden können. Testet eure Methode an den Obj-Dateien Tetrahedron und Icosahedron.

88. Erweitere die Klasse Faces, so dass jedes Objekt Face einen weiteren Parameter 'normal' hat. Dieser soll den Standartwert np.zeros(3) haben. Schreibe weiter eine Methode die testet ob ein Face eben ist, und für jedes ebene Face die Normale berechnet und dem Face zuweist. Schreibt nun eine Methode in der Klasse Mesh, welche die Normalen aller Faces eines Meshes als Liste ausgibt.

89. Erweitert die Plotmethode in der Klasse Mesh, so dass die Normalen ebener Faces visualisiert werden können. Die Normalen sollen an die Schwerpunkten der Faces geklebt werden. Testet eure Methode an den Obj-Dateien Tetrahedron und Sphere. Das Ergebenis sollte ungefähr so aussehen:



90. Schreibt eine Methode in der Klasse Halfedge, welche den Winkel zwischen einer Kante und der ihr folgenden Kante berechnet.

91. Beweist folgenden Satz aus der Vorlesung:
Sei $g : U \subset \mathbb{R}^2 \to \mathbb{R}^+$ eine glatte Funktion. Es existieren zwei Funktionen α, β mit $g(u, v) = \frac{\alpha(u)}{\beta(v)}$ genau dann, wenn $\frac{\partial^2}{\partial u \partial v}(\log(g(u, v))) = 0$.

92. Seien $c, d : I \subset \mathbb{R} \to \mathbb{R}^3$ zwei glatte Kurven. Wir definieren die parametrisierte Fläche

$$\begin{aligned} f : I^2 &\to \mathbb{R}^3 \\ (s, t) &\mapsto c(s) + d(t) \end{aligned}$$

1. Welche Bedingungen müssen die Kurven c und d erfüllen, damit die Fläche f regulär ist?

2. Angenommen f ist regulär, welche der folgenden Eigenschaften hat die parametrisierte Fläche f?

   - f is ein konjugiertes Netz
   - f is ein Königsnetz
   - f ist isotherm

93. Sei $c : I \subset \mathbb{R} \to \mathbb{R}^3$ eine glatte Kurve mit $\|c'(t)\| = 1$ und $c''(t) \neq 0$ für alle $t \in I$.

1. Zeige: $\langle c'(t), c''(t) \rangle = 0$ für alle $t \in I$.

2. Wir definieren die Tangentenfläche von c:

$$\begin{aligned} f : \mathbb{R} \times (0, \infty) &\to \mathbb{R}^3 \\ (t, s) &\mapsto c(t) + sc'(t) \end{aligned}$$

   Welche der folgenden Eigenschaften hat die parametrisierte Fläche f?

   - f is ein konjugiertes Netz
   - f is ein Königsnetz
   - f isotherm

94. Sei $r : (0, \infty) \to \mathbb{R}$ eine glatte Funktion, wir betrachten die parametrisiere Fläche:

$$f(u, v) := \begin{pmatrix} u \sin(v) \\ u \cos(v) \\ r(u) \end{pmatrix}, \qquad (u, v) \in (0, \infty) \times [0, 2\pi).$$

- Zeigt, dass f eine Krümmungslinien-Parametrisierung ist.
- Ist f isotherm?

95. Schreibt eine Methode in der Klasse Vertices, welche jeder Ecke die Summe der Innenwinkel zwischen allen ausgehenden Kanten zuordnet, die in einer Facette liegen. Siehe Abbildung 1:

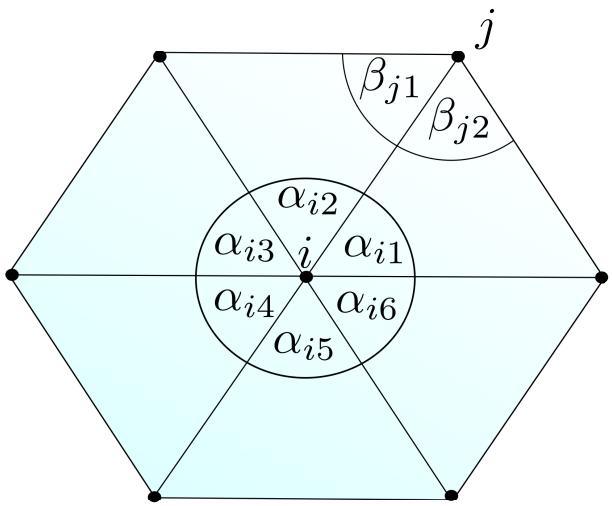


Abbildung 1: Für die innere Ecke i summiere alle 6 Winkel zwischen den Kanten auf. Für die Randecke j summiere nur die beiden Winkel auf, welche in der Fläche liegen.

96. Für diskrete Flächen mit ebenen Facetten kann man die Gaußkrümmung an den Ecken wie folgt definieren:

$$K(v) := 2\pi - \sum_i \alpha_i,$$

wobei die α_i's die Innenwinkel an der Ecke v bezeichnen.(Siehe Abbildung 1). Schreibt eine Methode in der Klasse Mesh, welche die Gaußkrümmung des Meshes als Liste ausgibt und eine Methode, welche die Gaußkrümmungen aller Ecken der Fläche aufsummiert. Testet eure Methode an den Obj Dateien: torus, sphere, sphere2 und bunny fine.

97. Schreibe eine Methode in der Klasse Halfedge, welche jeder Kante e die Zahl $H(e) := \frac{1}{2} l(e)\Theta(e)$ zu ordnet, dabei bezeichnet $l(e)$ die Länge der Kante e und $\Theta(e)$ den Winkel zwischen den Normalen, der an die Kante e grenzenden Facetten.

98. Schreibt eine Methode in der Klasse Mesh, welche die Fläche an einer Kugel mit Radius R und Mittelpunkt M spiegelt.

99. Wir betrachten ein diskretes Viereckne tz:

$$f : \mathbb{Z}^2 \to \mathbb{R}^3$$
$$(i, j) \mapsto f(i, j).$$

Zeigt, dass die partiellen Ableitungen vertauschen, d.h.:

$$\triangle_{ij} f = \triangle_{ji} f$$

100. Zwei ebene konvexe Vierecke (A, B, C, D) und (A^*, B^*, C^*, D^*) sind dual zu einander, wenn ihre Kanten parallel und ihre Diagonalen antiparallel sind. Zeigt dass es zu jedem ebene konvexen Viereck ein duales Viereck existiert und dieses bis auf Verschiebungen und Skalierungen eindeutig ist.

101. Seien (A, B, C, D) und (A^*, B^*, C^*, D^*) zwei ebene konvexe Vierecke mit parallelen Kanten. Zeigt:

$$AC \,||\, B^*D^* \Leftrightarrow BD \,||\, A^*C^*$$

Hinweis: Ein möglicher Lösungsweg verwendet Aufgabe 100.

102. Wir betrachten die Taylorentwicklung zweiten Grades eines glatten parametrisierten Flächenstücks

$$f : U \subset \mathbb{R}^2 \to \mathbb{R}^3$$
$$(s, t) \mapsto f(s, t),$$

am Entwicklungspunkt $(0, 0)$:

$$T(f)(s, t) := f(0, 0) + s\, f_s(0, 0) + t\, f_t(0, 0) + \frac{1}{2}(s^2\, f_{ss} + 2st\, f_{st} + t^2\, f_{tt}).$$

Sei $\epsilon > 0$ und TE der Tetraeder mit Eckpunkten

$$f_1 := T(f)(\epsilon, \epsilon), f_2 := T(f)(-\epsilon, \epsilon), f_3 := T(f)(-\epsilon, -\epsilon), f_4 := T(f)(\epsilon, -\epsilon).$$

Das Volumen von TE kann mit den Kanten $v_{ij} := f_i - f_j$, wie folgt berechnet werden:

$$\mathrm{Vol}(TE) = \frac{1}{6} |\det(v_{13}, v_{12}, v_{14})| = \frac{1}{6} |\langle v_{13}, v_{12} \times v_{14} \rangle|.$$

Zeigt: $\mathrm{Vol}(TE) = 0$, genau dann wenn f ein konjugiertes Netz ist.

103. Schreibt in der Klasse Faces Methoden, die erst testen ob die Facette ein ebenes Viereck ist, und im Erfolgsfall die Diagonalen und ihren Schnittpunkt M berechnen. Schreibt weiter in der Klasse Mesh Methoden welche die Diagonalen bzw. ihre Schnittpunkte als Liste ausgibt.

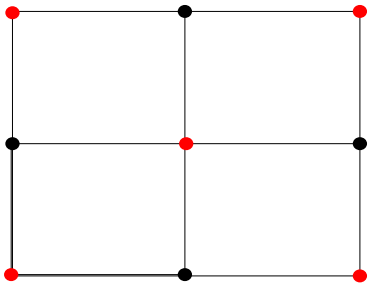
104. Schreibt in der Klasse Halfedges eine Methoden, die falls die Facette der Kante ein ebenes Viereck ist, den Abstand der Ecke, auf die die Kante zeigt, zum Schnittpunkt M der Diagonalen berechnet. Schreibt weiter in der Klasse Mesh eine Methode welche die Abstände als Liste ausgibt.

105. Erweitert die Plotmethode in der Klasse Mesh so, dass ihr die Diagonalen und ihre Schnittpunkte für ebene Viereckne tze plotten könnt. Fügt der Methode außerdem einen weiteren Parameter hinzu, der entscheidet ob die Kanten des Meshes gezeichnet werden oder nicht.

106. Sei $\overrightarrow{E}$ die Menge der orientierten Kanten des Viereckснetzes $\mathbb{Z}^2$. Ein Punkt $(n, m) \in \mathbb{Z}^2$ heißt rot falls $|n| + |m|$ gerade ist und schwarz falls $|n| + |m|$ ungerade ist. (Siehe Abbildung)

Wir definieren $q_1 : \overrightarrow{E} \to \mathbb{R}$ durch $q_1(e) = 3$ falls e von einem schwarzen zu einem roten Punkt verläuft und $q_1(e) = \frac{1}{3}$ falls e von einem roten zu einem schwarzen Punkt verläuft.

- Zeigt, dass q_1 eine exakte multiplikative 1-Form ist.
- Finde $\nu : \mathbb{Z}^2 \to \mathbb{R}^*$ so dass für jede gerichtete Kante e, die die Punkte x und y verbindet, gilt: $q_1(e) = \frac{\nu(y)}{\nu(x)}$.



107. Sei $\overrightarrow{E}$ die Menge der orientierten Kanten des Viereckснetzes $\mathbb{Z}^2$. Für benachbarte Punkte $(m_i, n_i), (m_j, n_j) \in \mathbb{Z}^2$ sei $e_{ij} \in \overrightarrow{E}$ die orientierte Kante von (m_i, n_i) nach (m_j, n_j). Wir definieren $q_2 : \overrightarrow{E} \to \mathbb{R}^3$ durch

$$q_2(e_{ij}) = \begin{cases} (1,0,0) & , \text{falls } n_j - n_i = 1 \\ (-1,0,0) & , \text{falls } n_j - n_i = -1 \\ (0,1,0) & , \text{falls } m_j - m_i = 1 \\ (0,-1,0) & , \text{falls } m_j - m_i = -1 \end{cases}$$

- Zeigt, dass q_2 eine exakte additive 1-Form ist.
- Finde $f : \mathbb{Z}^2 \to \mathbb{R}^3$ so dass für jede gerichtete Kante e, die die Punkte x und y verbindet, gilt: $q_2(e) = f(y) - f(x)$.

108. Schreibt ein Methode in der Klasse Face, die, falls die Facette ein Viereck ist, den Ecken des Vierecks neue Koordinaten zuweist, so dass das alte und neue Viereck dual zueinander sind. Testet eure Methode an Vierecken, welche ihr in der Klasse Quad erzeugt.

109. Schreibt ein Methode in der Klasse Quad, welche testet ob zwei Vierecke dual zueinander sind und eine Methode in der Klasse Mesh, welche testet ob zwei Viereckнetze Koenigsdual zu einander sind.

110. Verändert in PYTHON in der Klasse Mesh die Methode Koenigsdual2, welche die Koenigsduale des Netzes berechnet, so dass sie die Breitensuche verwendet.

111. Erzeugt mit dem Applet 'prescribed mean curvature' drei Flächen und speichert sie als Viereckснetzt (OBJ datei). Löscht die leeren Faces (es gibt 6061 Ecken) in den OBJ Dateien und lest diese in Phython ein. Dort sollt ihr die Koenigsdulaen der Netze berechnen und wieder als OBJ Datei speichern. Plottet nun die Ursprungsflächen und ihre Dualen in Rhino oder einem ähnlichem Programm. Was fällt euch auf?

112. Wir betrachten das Viereckснetz $\mathbb{Z}^2$ mit Ecken V orientierten Kanten E und Facetten F. Die Vekorräume der diskreten additiven $\mathbb{R}^3$-wertigen Formen sind definiert als:

$$\begin{aligned}\Omega^0(\mathbb{Z}^2,\mathbb{R}^3) &:= \{f : V \to \mathbb{R}^3\},\\ \Omega^1(\mathbb{Z}^2,\mathbb{R}^3) &:= \{\omega : E \to \mathbb{R}^3 | \, \omega(e) = -\omega(-e)\},\\ \Omega^2(\mathbb{Z}^2,\mathbb{R}^3) &:= \{\sigma : F \to \mathbb{R}^3\}.\end{aligned}$$

Auch für additive Formen kann man äußere Ableitungen definieren:

$$\begin{aligned} d_1 : \Omega^0(\mathbb{Z}^2,\mathbb{R}^3) &\to \Omega^1(\mathbb{Z}^2,\mathbb{R}^3)\\ f &\mapsto \omega\\ \omega(e_{ij}) &:= f(j) - f(i)\end{aligned}$$

$$\begin{aligned} d_2 : \Omega^1(\mathbb{Z}^2,\mathbb{R}^3) &\to \Omega^2(\mathbb{Z}^2,\mathbb{R}^3)\\ \omega &\mapsto \sigma\\ \sigma(\phi) &:= \sum_{e\in\phi} \omega(e).\end{aligned}$$

Zeigt dass $d_2 \circ d_1 = 0$ gilt.

113. Sei Möb(2) die Menge der Möbiustranformation auf $\hat{\mathbb{C}}$. Zeigt, dass Möb(2) eine Gruppe bezüglich der Hintereinanderausführung ist. Das heißt es gilt:

- $\forall\, T_1, T_2 \in \text{Möb}(2) \Rightarrow T_1 \circ T_2 \in \text{Möb}(2)$
- $Id \in \text{Möb}(2)$
- $T \in \text{Möb}(2) \Rightarrow T^{-1} \in \text{Möb}(2)$

114. Seine $p_1, p_2, p_3 \in \mathbb{C}$. Zeigt, dass die Abbildung

$$\begin{aligned} T : \hat{\mathbb{C}} &\to \hat{\mathbb{C}}\\ z &\mapsto DV(z, p_1, p_2, p_3),\end{aligned}$$

eine Möbiustransformation ist, welche die Punkte p_1, p_2, p_3 auf $0, 1, \infty$ abbildet.

115. Seien $p_1, p_2, p_3, q_1, q_2, q_3 \in \hat{\mathbb{C}}$ 6 Punkte. Zeigt, es gibt genau eine orientierungserhaltene Möbiustranformation $T_1(z) = \frac{az+b}{cz+d}$ mit $T_1(p_i) = q_i$.
Hinweis: Ein möglicher Lösungsweg verwendet die beiden vorhergegangen Aufgaben.

116. Zeigt, dass eine orientierungserhaltene Möbiustranformation, welche mehr als zwei Fixpunkte hat, die Identität ist.

117. In $\hat{\mathbb{C}}$ fassen wir Kreise und Geraden zu verallgemeinerten Kreisen zusammen. Zeigt, dass Möbiustransformationen in $\hat{\mathbb{C}}$ verallgemeinerte Kreise auf verallgemeinerte Kreise abbilden.