Teil I

Aufgabe 1

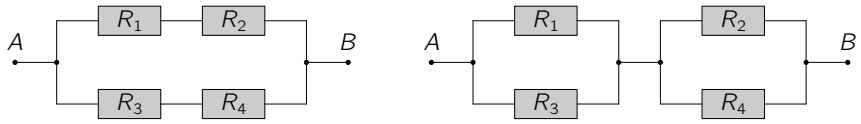
Die Sinusfunktion kann durch Taylorpolynome approximiert werden:

$$\sin x \approx S(x,n) := \sum_{j=0}^{n} (-1)^j \frac{x^{2j+1}}{(2j+1)!}$$

Schreiben Sie eine Funktion `S(x,n)`, welche $S(x,n)$ berechnet. Benutzen Sie Ihre Funktion, um $\sin x$ zusammen mit $S(x,n)$, $n = 1,2,3,6,12$ auf dem Intervall $[0,4\pi]$ zu plotten.

Aufgabe 2

Es seien $R_1,\ldots,R_4 \geq 0$ Ohmsche Widerstände, die in zwei Varianten wie folgt zusammengeschaltet sind:



Schreiben Sie zwei Funktionen von jeweils vier reellen Argumenten, die den Gesamtwiderstand zwischen A und B entsprechend den beiden Schaltungsvarianten zurückgeben. Die Funktionen sollen auch die Grenzfälle $R_i = 0$ korrekt behandeln, im Fall von $R_i < 0$ soll eine Fehlermeldung ausgegeben werden. Testen Sie mit Ihren Funktionen, in welcher Situation der Gesamtwiderstand größer ist. (Läßt sich das beweisen?)

Aufgabe 3

Aus der Analysisvorlesung ist $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$ bekannt, wobei die rechtsstehende unendliche Reihe für alle $x \in \mathbb{R}$ konvergiert. Überzeugen Sie sich durch eine Implementierung in einem Python-Programm, daß man über diese Reihe nicht die Werte von $\cos x$ für große x, etwa $x = 31.4159265 \approx 10\pi$ hinreichend genau berechnen kann, da Auslöschungseffekte auftreten können. Berechnen Sie dazu die $\cos(k\pi)$, $k = 1,2,\ldots,20$ mittels der Reihe und geben Sie die Werte zusammen mit k und dem exakten Wert in einer Tabelle aus.

Aufgabe 4

Schreiben Sie ein Programm, das in einer Endlosschleife Zahlen abfragt und für diese die Primfaktorenzerlegung ausgibt, z.B. in der Form 200 = 2 * 2 * 2 * 5 * 5. Bei Eingabe von 0 soll das Programm beendet werden.

Aufgabe 5

Ein beliebiges Dreieck kann durch die Koordinaten (x_1,y_1), (x_2,y_2),(x_3,y_3) seiner drei Eckpunkte gegeben werden (in mathematisch positiver Orientierung). Schreiben Sie eine Funktion `flaeche(eckpunkte)`, die den Flächeninhalt des Dreiecks zurückgibt, wobei das Argument `eckpunkte` eine verschachtelte Liste der Eckpunkte ist, z.B. `[[0,0], [1,0], [0,2]]`.

Aufgabe 6

Ein überaus beliebter Mathematikprofessor bietet vier Studenten eine zusätzliche Konsultation zur Numerik an. Leider können sich die Studenten (Anton, Berta, Claus und Doris) nicht einigen, wer hingeht und wer nicht (eigentlich wollen alle nicht). In einer gemeinsamen Diskussion kann man sich jedoch auf die folgenden Grundsätze verständigen:

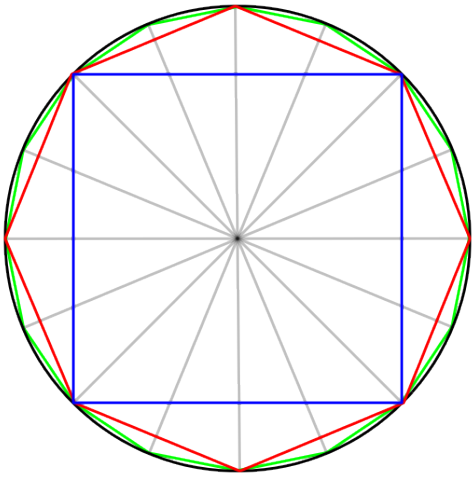
a) Mindestens ein Student geht zur Konsultation.

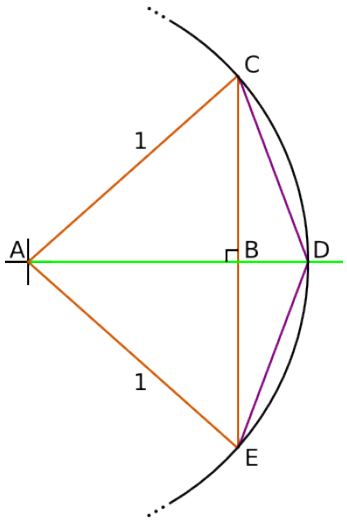
b) Anton geht auf keinen Fall zusammen mit Doris.
c) Wenn Berta geht, dann geht Claus mit.
d) Wenn Anton und Claus gehen, dann bleibt Berta zu Hause.
e) Wenn Anton zu Hause bleibt, dann geht entweder(!) Doris oder Claus.
Schreiben Sie ein Programm, das alle Gruppierungen ermittelt und auf dem Bildschrim ausgibt, in denen die Studenten zur Konsultation gehen könnten. Hinweis: Erstellen Sie alle denkbaren Kombinationen und prüfen Sie jede darauf, ob alle obigen Grundsätze erfüllt sind.

Aufgabe 7
Programmieren Sie das Spiel „Zahlenraten“: Der Computer generiert eine ganze Zufallszahl zwischen 1 und 100, die der Spieler mit maximal 6 Versuchen erraten muß. Es wird bei jedem Versuch mitgeteilt, ob die eingegebene Zahl zu groß oder zu klein war. Hinweis: Eine ganze Zufallszahl kann mit der Funktion `randint(m, n)` aus dem Modul `random` erzeugt werden.

Teil II

Aufgabe 8
Eine untere Schranke für 2π, den Umfang des Einheitskreises, erhält man durch die Summe der Seitenlängen eines dem Einheitskreis eingeschriebenen regelmäßigen n-Ecks. Die Abbildung links zeigt, wie man beginnend mit einem Viereck der Seiteblänge $s_4 = \sqrt{2}$ die Eckenzahl stets verdoppelt.





DIe Abbildung rechts zeigt die Geometrie der Eckenverdoppelung.
Mit $|\overline{CE}| = s_n$, $|\overline{CD}| = |\overline{DE}| = s_{2n}$, $|\overline{AB}| = a$, $|\overline{BD}| = 1 - a$, liefert Pythagoras für die Dreiecke ABC und BDC jeweils

$$a^2 + \left(\frac{s_n}{2}\right)^2 = 1 \tag{1}$$

$$(1-a)^2 + \left(\frac{s_n}{2}\right)^2 = s_{2n}^2 \tag{2}$$

Elimination von a liefert die Rekursion

$$s_{2n} = \sqrt{2 - \sqrt{4 - s_n^2}} \qquad \text{mit Startwert} \qquad s_4 = \sqrt{2} \tag{3}$$

für die Länge s_n *einer* Seite des eingeschriebenen regelmäßigen n-Ecks. Die Folge $(n\, s_n)$ konvergiert monoton von unten gegen den Grenzwert 2π:

$$n\, s_n \uparrow 2\pi \tag{4}$$

Als relativen Fehler kann man definieren

$$\epsilon_n = \left| \frac{n\, s_n - 2\pi}{2\pi} \right| \tag{5}$$

Aufgabenstellung

1. Überzeugen Sie sich davon, dass die Gleichung

$$s_{2n} = \frac{s_n}{\sqrt{2 + \sqrt{4 - s_n^2}}} \tag{6}$$

   mathematisch äquivalent zu Gleichung (3) ist.

2. Berechnen Sie s_n und daraus Näherungen für 2π und deren Fehler ϵ_n für $n = 2^k, \quad k = 3, ..., 35$ mit Hilfe der Rekursion Gl. (3) und der Rekursion Gl. (6)

3. Plotten Sie die relativen Fehler ϵ_{2^k} beider Rekursionen als Funktion von k in einem Plot mit logarithmischer y-Achse. Analysieren Sie das entstehende Bild. Vergleichen Sie die Stabilität der Algorithmen Gl. (3) und Gl. (6) und finden Sie den Grund für den Unterschied.

Aufgabe 9

Nützliches für die Biergartensaison: Ein quadratischer Tisch mit vier Beinen in den Ecken kann auch auf unebenen Boden stets so gedreht werden, daß er nicht kippelt. In etwas vereinfachter mathematischer Formulierung: Zu einer gegebenen Raumkurve $[0, 2\pi] \ni \alpha \mapsto P(\alpha) = \big(\cos\alpha, \sin\alpha, f(\alpha)\big)$ mit einer 2π-periodischen Funktion f ist ein Winkel α gesucht, so daß die vier Punkte $P(\alpha)$, $P(\alpha + \frac{\pi}{2})$, $P(\alpha + \pi)$, $P(\alpha + \frac{3\pi}{2})$ in einer Ebene liegen. Schreiben Sie eine Funktion in Python, um zu einer gegebenen Funktion f einen solchen Winkel numerisch zu bestimmen. Verwenden Sie zur numerischen Nullstellensuche eine einfache Bisektion. Geben Sie am Ende den gefundenen Winkel α in Grad aus. Testen Sie Ihre Funktion mit $f(\alpha) = \sin\alpha + 0.5 \cdot \cos 2\alpha$. (Literatur: N. Herrmann: Mathematik ist wirklich überall.)

Aufgabe 10

Schreiben Sie in Python zwei Funktionsroutinen, um die Darstellung des Interpolationspolynoms in der Form von Newton zu implementieren. Die erste Funktion soll als Argumente Listen der gegebenen Wertepaare $(x_i, y_i), i = 0, \ldots, n$ und den Grad n des Polynoms haben und eine Liste der gesuchten Koeffizienten $a_0, \ldots, a_n$ in der Darstellung

$$p(x) = a_0 + a_1(x - x_0) + \ldots + a_n(x - x_0)(x - x_1) \ldots (x - x_{n-1})$$

zurückgeben. Die zweite Funktion soll die Auswertung des Newtonschen Interpolationspolynoms in Analogie zum Horner-Schema an einer Stelle x ermöglichen, wenn die Koeffizienten $a_0, \ldots, a_n$ bereits berechnet sind.

Von einer monoton wachsenden Funktion $f = f(x)$ seien die folgenden Funktionswerte gegeben:

x	2.800	2.805	2.810	2.815	2.820	2.825	2.830
$f(x)$	8.1919	8.2333	8.2749	8.3166	8.3586	8.4008	8.4432

x	2.835	2.840	2.845	2.850	2.855	2.860	2.865	2.870
$f(x)$	8.4858	8.5287	8.5717	8.6150	8.6584	8.7021	8.7460	8.7902

Stellen Sie mit Hilfe der inversen Interpolation zur Funktion f (d.h. mit vertauschten Rollen von Stützstellen und Funktionswerten) eine Wertetabelle der Umkehrfunktion im Intervall $[8.20, 8.75]$ mit der Schrittweite 0.05 auf. Benutzen Sie zur Berechnung Ihre Funktionenroutinen zur Darstellung des Interpolationspolynoms in der Form von Newton.

Aufgabe 11

Schreiben Sie in Python ein Programm, um die Approximationseigenschaften der Interpolationspolynome am Beispiel der Runge-Funktion $f(x) = 1/(1 + 25x^2)$ im Intervall $[-1, 1]$ zu veranschaulichen:

1. Schreiben Sie eine Funktion, die als Argumente einen Vektor mit Stützstellen x_i, einen Vektor mit Funktionswerten $y_i = f(x_i)$ und einen Wert x übergeben bekommt und die mit Hilfe des Aitken-Neville-Algorithmus den Wert des Interpolationspolynoms an der Stelle x berechnet und zurückgibt.

2. Verwenden Sie diesen Algorithmus, um die Runge-Funktion $f(x) = \frac{1}{1+25x^2}$ auf dem Intervall $[-1, 1]$ mit $N = 4, 8, 12, 16$ äquidistanten Stützstellen (also mit einem Polynom 3., 7., 11. bzw. 15. Grades) zu approximieren. Plotten Sie dazu jeweils die Runge-Funktion und das Polynom mit je 200 Datenpunkten. Verwenden Sie zum Plotten des Polynoms Ihre Aitken-Neville-Funktion.

   Was beobachten Sie?

Aufgabe 12

Schreiben Sie in Python eine Funktionsroutine, um das Gaußsche Eliminierungsverfahren ohne Pivotisierung bei Gleichungssystemen mit strikt diagonaldominanten Tridiagonalmatrizen als Koeffizientenmatrizen zu implementieren. Die Funktion sollte als Argumente die Anzahl der Gleichungen sowie Listen für die Haupt- und Nebendiagonalelemente und die rechten Seiten haben und sie sollte eine Liste mit den Lösungen zurückgeben. Testen Sie Ihre Funktion am Beispiel einer Tridiagonalmatrix der Größe $n = 1000$, bei der alle Hauptdiagonalelemente gleich 2 und alle Nebendiagonalelemente gleich 1/2 sind sowie der rechten Seite $y_k = \sqrt{k}$, $k = 1, \ldots, 1000$. Berechnen Sie den Wert von x_{500}!

Aufgabe 13

Implementieren Sie die Berechnung von verallgemeinerten natürlichen kubischen Interpolationssplines. Schreiben Sie dazu zunächst eine Funktionsroutine, um die Berechnung der Koeffizienten der kubischen Polynome in den Teilintervallen durchzuführen. Die Funktion sollte als Argumente also Listen mit den gegebenen Wertepaaren, die zwei Randbedingungen und die Anzahl der Knoten besitzen und Listen mit den gesuchten koeffizienten zurückgeben. Benutzen Sie günstigerweise Ihre Funktion aus Aufgabe 12 zur Lösung des Gleichungssystems. Schreiben Sie dann eine weitere Funktionsroutine, um die Interpolationssplines an einer Stelle x auszuwerten, wobei neben den Funktionswerten auch die Ableitungen bis zur Ordnung 3 berechnet werden sollen. Testen Sie Ihre Routinen am Beispiel der sin-Funktion im Intervall $[0, 1]$ mit äquidistanten Stützstellen $0.0, 0.1, \ldots, 1.0$. Berechnen Sie dann die Werte der natürlichen kubischen Splineinterpolation an den Stellen $0.05, 0.15, \ldots, 0.95$. Vergleichen Sie die Fehler mit dem Fehler, der sich bei Polynominterpolation ergibt (Plots!).

Aufgabe 14

Implementieren Sie in Python in einer Funktionroutine das Verfahren der Richardson-Extrapolation zur numerischen Differentiation. Testen Sie das Verfahren der Richardson-Extrapolation am Beipiel der Berechnung der zweiten Ableitung der Funktion $f(x) = \log x$ an der Stelle $x = 1$. Verwenden Sie für das Verfahren $B(h)$ den Differenzenausdruck

$$B(h) = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}$$

als Näherung für $f''(x)$. Starten Sie mit der Grundschrittweite $h = 0.5$ und benutzen Sie die relative Fehlerschranke epsilon $= 10^{-8}$. Wieviele Iterationen sind notwendig?

Aufgabe 15

Die Lösungen der Gleichung

$$x^5 + 3x + 4 = e^{\cos x} + 8x^2$$

sollen im Intervall $[-1, 2]$ mit Hilfe von Fixpunktiterationen bestimmt werden. Fertigen Sie Plots der Funktionsverläufe der linken bzw. rechten Seite der Gleichung an. Die drei Funktionen

$$\Phi_1(x) = \frac{e^{\cos x} - 4}{x^4 - 8x + 3}, \quad \Phi_2(x) = \frac{e^{\cos x} - 3x - 4}{x^4 - 8x}, \quad \Phi_3(x) = \frac{e^{\cos x} - 3x - 4}{x^4 + 2x^3 + 4x^2} + 2$$

definieren Fixpunktiterationen $x_{k+1} = \Phi_i(x_k)$. Zeigen Sie, daß die Fixpunkte Lösungen der obigen Gleichung sind.

Implementieren Sie mittels Python eine Funktion `fixIter(F,x0,epsilon,maxit)`, die eine Fixpunktiteration $x_{k+1} = F(x_k)$, $k = 0, 1, \ldots$ mit dem Startwert x_0 bis zur Genauigkeit `epsilon` oder bis `maxit` Iterationen ausführt. Rückgabewerte sollen der gefundene Fixpunkt, die Anzahl der benötigten Iterationen um die gewünschte absolute Genauigkeit $|x_{i+1} - x_i| <$ `epsilon` zu erreichen und ein Konvergenzflag (Fixpunktiteration konvergiert oder nicht) sein.

Benutzen Sie Ihre Funktion für Φ_1 mit Startwert $x_0 = 0$, für Φ_2 mit Startwert $x_0 = 0.5$ und für Φ_3 mit Startwert $x_0 = 1$. Verwenden Sie als Abbruchkriterien `epsilon` $= 10^{-12}$ und `maxit` $= 100$.

Aufgabe 16

Gesucht ist eine Lösung $x = (x_1, x_2) \in \mathbb{R}^2$ des nichtlinearen Gleichungssystems

$$\begin{aligned} 2x_1 - \frac{\sin x_1}{4} - x_2 &= 1, \\ -x_1 + 2x_2 - \sin x_2 &= 1 \end{aligned}$$

als Fixpunkt der Abbildung

$$\phi : \mathbb{R}^2 \to \mathbb{R}^2, \qquad \phi(x) = \frac{1}{2}\left(1 + \frac{\sin x_1}{4} + x_2, 1 + x_1 + \sin x_2\right)$$

Schreiben Sie ein Programm, welches mit dem Startwert $x^0 := (0, 0)$ die gesuchte Lösung des nichtlinearen Gleichungssystem näherungsweise mit der Fixpunktiteration $x^{(k+1)} := \phi(x^{(k)})$, $k = 0, 1, \ldots$ berechnet. Rechnen Sie so lange, bis $\|x^{(k+1)} - x^{(k)}\|^2 \leq$ `epsilon`, maximal aber `maxit` Schritte. `epsilon` $= 10^{-8}$ und `maxit` $= 200$ seien als Konstante zu vereinbaren.

Aufgabe 17

Aus den Ergebnissen über die asymptotische Entwicklung der summierten (zusammengesetzten) Trapezformel $T_n(h)$ folgt bei Anwendung auf eine periodische Funktion $f \in C^{2k+1}$ und Integration über ein Periodenintervall die Fehlerordnung

$$\int_a^b f(x)\,dx = T_n(h) + O(h^{2k+2}), \qquad h = (b - a)/n.$$

Also ist bei glatten Funktionen eine sehr schnelle Konvergenz zu erwarten, wenn über das gesamte Periodenintervall integriert wird. Überzeugen Sie sich davon am Beispiel der 2π-periodischen Funktion $f(x) = 1/(1 + 0.5 \sin x)$! Berechnen Sie dazu die absoluten Fehler der l-fach zusammengesetzten Trapez-Regel mit $l = 1, \ldots, 30$ bei den Integralen $\int_0^6 f(x)\,dx$ und $\int_0^{2\pi} f(x)\,dx$. Zeichnen Sie die beiden Fehlerkuren.