Allgemeine Informationen: Dieses Aufgabenblatt enthält schriftliche und/oder Programmieraufgaben. Bitte kombinieren Sie alle Lösungen zu den schriftlichen Aufgaben zu einem einzelnen PDF Dokument, welches Sie nach folgendem Schema benennen:`{lastname}-written.pdf`. Sie können Ihre Lösungen auch scannen oder fotografieren. Achten Sie in diesem Fall auf die Lesbarkeit. Es werden JPEG/PNG Bilddateien akzeptiert welche wie folgt benannt werden müssen:`{exercisenumber}-{lastname}-written.{jpeg/png}`. Stellen Sie sicher, dass alle Rechenschritte nachvollziehbar sind und kombinieren Sie nicht zu viele kleine Schritte zu einem einzelnen. Die Programmieraufgaben müssen in *Julia* gelöst sein und Ihr Quellcode sollte nach folgendem Schema benannt sein: `{exercisenumber}-{lastname}.jl`.

(1) (2.5 Punkte) Gegeben ist die folgende Funktion:

$$f(x) = 10 - x^2 - 8e^{-x^2}.$$

Verwenden Sie das Julia Template **1-finitediff.jl** für alle Implementierungen in dieser Aufgabe.

a) (0.5 Punkte) Ermitteln Sie die analytische Ableitung $f'(x)$ per Hand.

b) (0.5 Punkte) Finite Differenzen können verwendet werden, um die Ableitungen einer Funktion numerisch zu approximieren. Es gibt verschieden Finite Differenzen Methoden. Drei grundlegende Methoden sind die *Vorwärts*, *Rückwärts* und *Zentrale Differenz*:

$$\begin{aligned}\text{Vorwärts: } D_+ [f(x)] =& \frac{f(x+h) - f(x)}{h},\\ \text{Rückwärts: } D_- [f(x)] =& \frac{f(x) - f(x-h)}{h},\\ \text{Zentral: } D_c [f(x)] =& \frac{f(x+\frac{h}{2}) - f(x-\frac{h}{2})}{h}.\end{aligned}$$

Berechnen Sie den Wert der ersten Ableitung $f'(x)$ bei $x = 0$. Verwenden Sie dann die drei Finite-Differenzen Methoden mit $h = 0.1$, um die Ableitung numerisch zu approximieren. Werten Sie für die numerische Approximation zuerst $f(x)$ bei -0.1, -0.05, 0.0, 0.05, 0.1 aus und notieren Sie die resultierenden Werte.

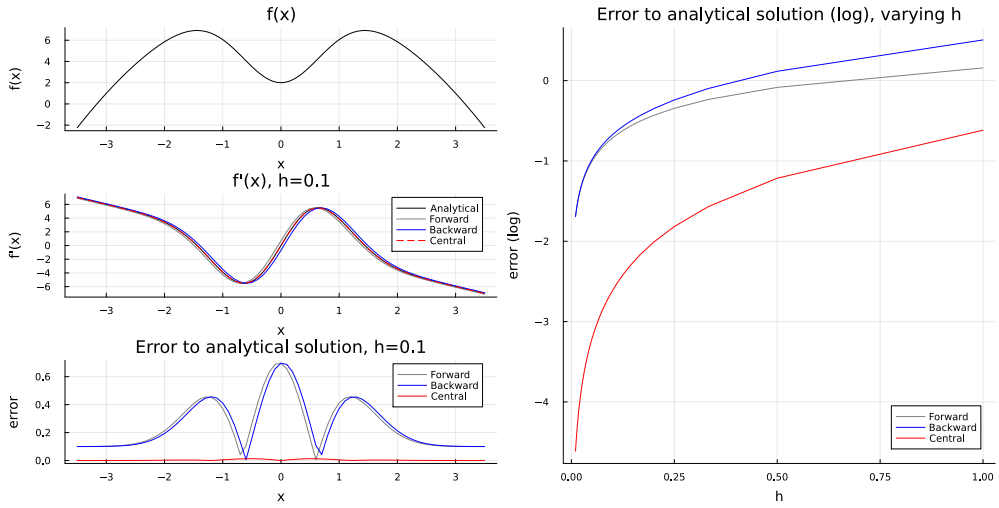
Vergleichen Sie die Ergebnisse, indem Sie den Fehler zwischen den Näherungen und der analytischen Lösung unter Verwendung der absoluten Differenz berechnen. *Z.B.* für die Vorwärts Differenz:

$$\varepsilon^+ = |D_+ [f(0.0)] - f'(0.0)|.$$

Führen Sie die Berechnungen manuell durch und geben Sie alle notwendigen Schritte in der schriftlichen Lösung an.

c) (0.25 Punkte) Implementieren Sie die analytische Lösung der Ableitung $f'(x)$ in der Methode **f1(x::Float64)::Float64**.

d) (1 Punkt) Implementieren Sie die Finite Differenzen Methoden **forward()**, **backward()** und **central()** mit der Signatur **func(x::Float64, h::Float64, f::Function)::Float64** im Modul **FiniteDiff**. Die resultierenden Plots sollten aussehen wie in der folgenden Abbildung.



e) (0.25 Punkte) Betrachten Sie das Ergebnis und erklären Sie, was Sie beobachten können. Was ist der Grund dafür, dass einige Methoden im Vergleich zu anderen genauer sind (kleinere Fehler)?

(2) (2.5 Punkte) Gegeben ist die Selbe Funktion wie in Aufgabe (1):

$$f(x) = 10 - x^2 - 8e^{-x^2}.$$

Das Integral dieser Funktion ist:

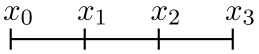
$$F(x) = \int f(x)dx = -4\sqrt{\pi} \cdot \text{erf}(x) - \frac{x^3}{3} + 10x + C,$$

mit $\text{erf}(x)$ der Gaußschen Fehlerfunktion definiert als:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

Verwenden Sie das Julia Template **2-quadrature.jl** für alle Implementierungen in dieser Aufgabe.

a) (0.5 Punkte) Berechnen Sie das bestimmte Integral $\int_{-3}^{3} f(x)dx$ indem sie die Grenzen für $F(x)$ einsetzen. Da $\text{erf}(x)$ nicht analytisch berechnet werden kann verwenden Sie die Approximation $\text{erf}(3) = 0.9999779$. In einem weiteren Schritt, approximieren Sie das bestimmte Integral $\int_{-3}^{3} f(x)dx$ numerisch. Verwenden Sie für die numerische Berechnung drei äquidistante Intervalle ($N = 2$) und geben Sie zunächst alle notwendigen Funktionswerte an.



Verwenden Sie die Rechtssummen-, die Trapez- und die Simpson-Regel für numerische Quadratur, um das Integral auszuwerten. Die drei Regeln lauten wie folgt:

$$\mathcal{I}_R = \sum_{i=1}^{N} f(x_i) \cdot h,$$

$$\mathcal{I}_T = \sum_{i=1}^{N} \frac{f(x_{i-1}) + f(x_i)}{2} \cdot h,$$

$$\mathcal{I}_S = \sum_{i=1}^{N} \frac{f(x_{i-1}) + 4f((x_i + x_{i-1})/2) + f(x_i)}{6} \cdot h.$$

Berechnen Sie die Fehler zwischen den numerischen Lösungen und der Lösung für das Einsetzen in die Stammfunktion $F(x)$ mit Hilfe von absoluten Differenzen. *Z.B.* für die Rechtssummen-Regel:

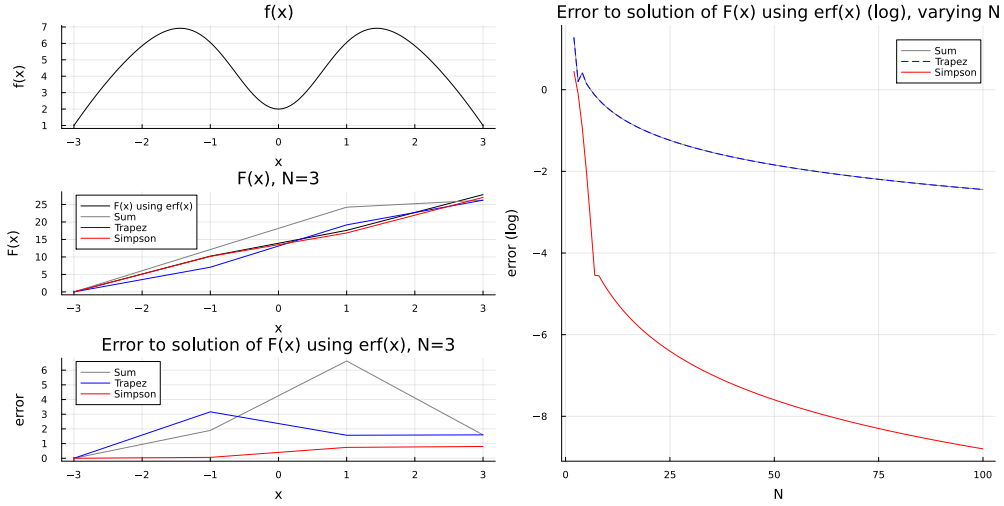
$$\varepsilon_U = \left| \mathcal{I}_R - F(x)|_{-3}^{3} \right|.$$

Führen Sie die Berechnungen manuell durch und geben Sie alle notwendigen Schritte in der schriftlichen Lösung an.

b) (0.25 Punkte) Implementieren Sie die Stammfunktion des Integrals von **f(x::Float64)** in der Funktion **F(x::Float64)**. Verwenden Sie dazu das gegebene unbestimmte Integral ohne die Konstante. Für erf(x) kann das Julia Paket **SpecialFunctions** installiert werden. Anschließend muss das Paket mit **using SpecialFunctions** importiert werden um die Funktion **erf(x)** zu verwenden.

c) (1.5 Punkte) Als nächstes implementieren Sie die numerischen Integrations-Methoden im Modul **Quadratur** in den Funktionen **right()**, **trapez()** und **simpson()**. Die Signatur dieser Funktionen ist **func(s::Float64, e::Float64, N::Int, f::Function)::Float64**, wobei **s** die Unter- und **e** die Obergrenze der Integration, **N** die Anzahl der Intervalle und **f(x::Float64)::Float64** die zu integrierende Funktion ist.

Die resultierenden Plots sollten wie in der folgenden Abbildung aussehen.



d) (0.25 Punkte) Betrachten Sie die Plots und beschreiben Sie, was Sie beobachten können. Welche Methode ist die Beste? Warum denken Sie, dass dies der Fall ist? Warum ist der Fehler der Rechtsummen- und der Trapez-Regel gleich? Denken Sie das ist immer der Fall? Oder finden Sie einen Fall bei dem das anders ist?

(3) (2 Punkte) Diffusionsprozesse wie die Ausbreitung von Wärme über die Zeit in einer bestimmten Region können mit der Diffusionsgleichung modelliert werden. Sie ist eine parabolische partielle Differentialgleichung und ihre normierte Form in einer Dimension ist gegeben durch:

$$\frac{\partial}{\partial t} u(x,t) = \frac{\partial^2}{\partial x^2} u(x,t).$$

Außerdem müssen zur Lösung eines solchen Problems der Zustand $u(0,t)$ an den Grenzen (Randbedingung) und der Anfangszustand $u(x,0)$ (Anfangsbedingung) bekannt sein. Im Folgenden sind eine homogene *Dirichlet*-Randbedingung und eine anfängliche Wärmeverteilung

für $-1 < x < 1$ und $t > 0$ gegeben:

$$u(-1,t) = u(1,t) = 1.0,$$
$$u(x,0) = x^2 - |x|.$$

Ein solches Problem kann numerisch durch Finite Differenzen approximiert werden. Dazu wird der Raum in $x_0, \ldots, x_I$ mit einer Schrittweite h und der Zeitbereich in $t_0, \ldots, t_N$ mit einer Schrittweite τ diskretisiert. Auf einem solchen Gitter kann $u(x,t)$ an jedem Punkt $u(x_i, t_n)$ approximiert werden.

Dazu kann eine explizite Methode hergeleitet werden, indem die Zeitableitung $\frac{\partial}{\partial t}u(x_i,t_n)$ an der Position x_i über eine Vorwärts Differenz (siehe $D_+[f(x)]$ in (2)) berechnet wird:

$$\frac{\partial}{\partial t}u(x_i,t_n) = \frac{u(x_i,t_{n+1}) - u(x_i,t_n)}{\tau}.$$

Die Raumableitung $\frac{\partial^2}{\partial x^2}u(x_i,t_n)$ kann dann über eine Zentrale Differenz zweiter Ordnung approximiert werden. Die Formel für die Zentrale Differenz zweiter Ordnung erhält man, indem man die Zentrale Differenz erster Ordnung (siehe $D_c[f(x)]$ in (2)) zweimal anwendet:

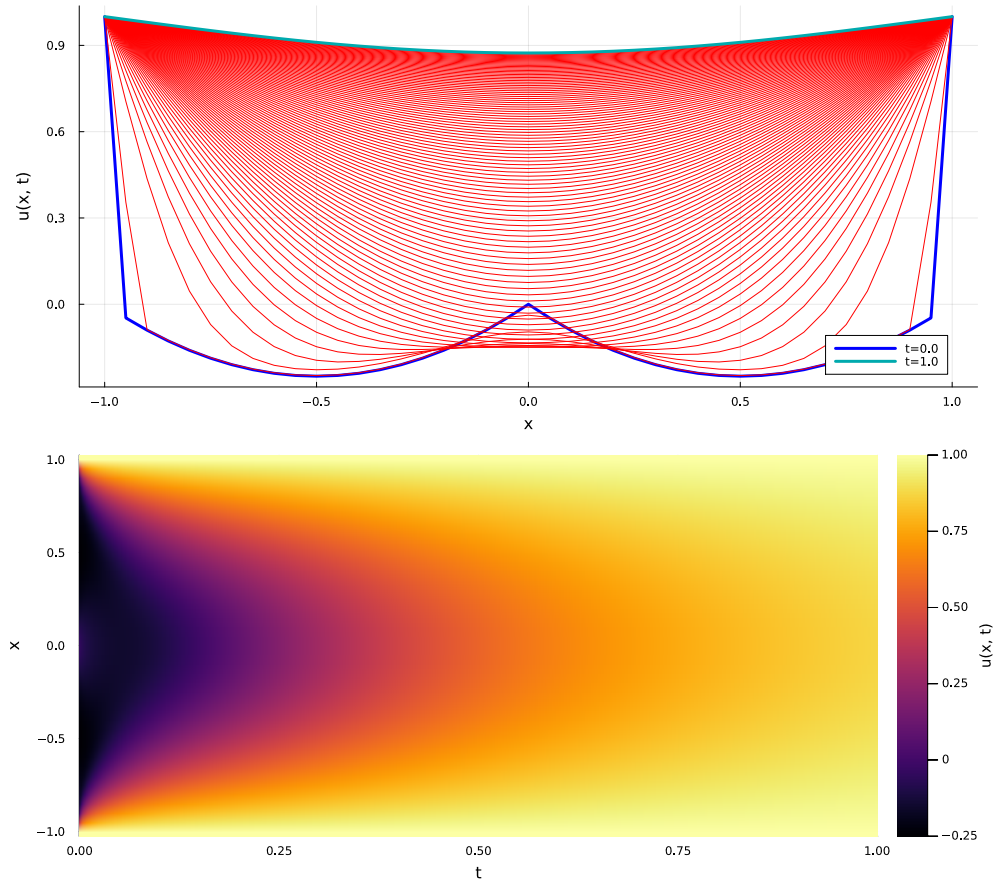
$$D_c^{(2)}[f(x)] = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

Damit kann $\frac{\partial^2}{\partial x^2}u(x_i,t_n)$ folgendermaßen approximiert werden:

$$\frac{\partial^2}{\partial x^2}u(x_i,t_n) = \frac{u(x_{i+1},t_n) - 2u(x_i,t_n) + u(x_{i-1},t_n)}{h^2}.$$

a) (0.5 Punkte) Leiten Sie die Formel für $u(x_i,t_{n+1})$ her. Setzen Sie $\frac{\partial}{\partial t}u(x,t)$ und $\frac{\partial^2}{\partial x^2}u(x,t)$ in die Diffusionsgleichung mit der Vorwärts Differenz über die Zeit und der Zentralen Differenz zweiter Ordnung über den Raum ein und lösen Sie nach $u(x_i,t_{n+1})$ auf.

b) (1.5 Punkte) Implementieren Sie den Aktualisierungsschritt von $u(x,t)$ für einen Zeitschritt in dem Template **3-heat.jl**. Insbesondere muss die Funktion **explicitStep(us::Vector, h::Float64, tau::Float64)::Vector** implementiert werden. Der resultierende Plot sollte gleich wie in der folgenden Abbildung die Veränderung der 2D Wärmeverteilung über die Zeit auf zwei verschiedene Arten visualisieren.


u(x, t)
x
t=0.0
t=1.0
t
u(x, t)

Bonus Aufgabe (1 Punkt) Zerlegen Sie folgende Polynome mit Hilfe des Horner-Schemas. Ermitteln Sie die Anzahl der Rechenoperationen vorher und nachher. Die Rechnungen zu (a) und (b) sind von Hand zu führen.

(a) (0.25 Punkte) Polynom in einer Variable:

$$6x^8 - 2x^6 + x^5 + 3x^2 - 7x + 2$$

(b) (0.5 Punkte) Polynom in zwei Variablen. Sortieren Sie die Terme zuerst in x-, y- und xy-abhängige Terme und zerlegen Sie die drei Gruppen getrennt.

$$3x^9 + 4x^7y^7 + 2y^7 - 4x^6y^3 + 7x^3y^3 - xy^2 + 11y - 21$$

(c) (0.25 Punkte) Implementieren Sie die zerlegte Funktion aus (b) in der vorbereiteten Julia Funktion **h(x::Float64, y::Float64)::Float64** in **bonus-horner.jl**. Das Skript misst die Laufzeit um das Polynom für jede Zelle auf einem Grid der Göße $N \times N, N = 1024$ auszuwerten und berichtet dann die durchschnittliche Laufzeit für beide Varianten über drei Ausführungen. Zusätzlich werden die Ergebnisse und die Unterschiede zwischen diesen als Heatmaps geplottet (siehe Abbildung unten). Was ergibt sich für ein Speed-Up? Verwenden Sie dafür den Mittelwert der gemessenen Laufzeiten.