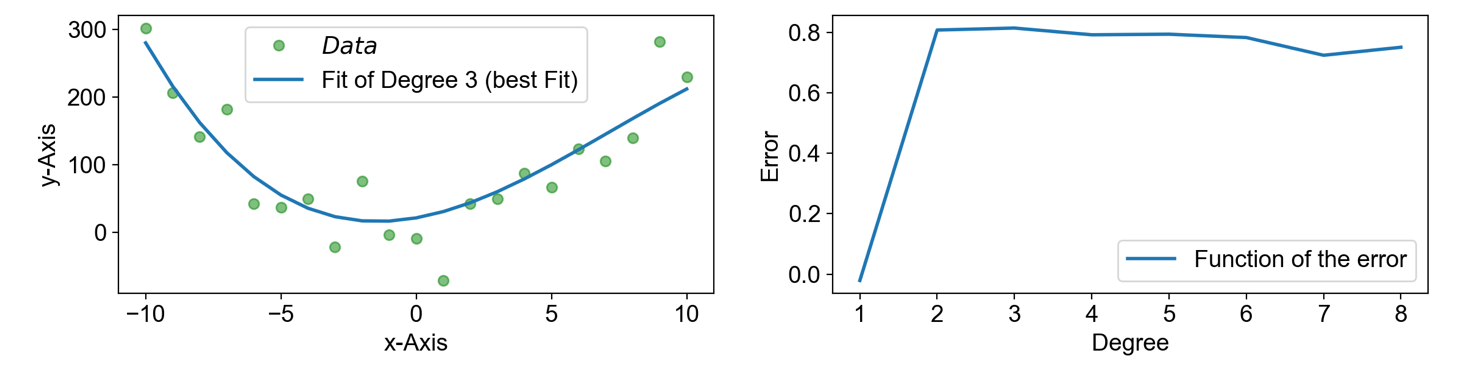
# Angewandte Mathematik: Blatt 1

## Aufgabe 1e)

Bei der linearen Regression geht es darum, die optimale Funktion durch eine gegebene Anzahl an Datenpunkten zu finden. Die Funktion sollte einen möglichst kleinen Fehler haben, aber auch nicht genau durch alle Punkte laufen (Interpolation), da sonst Prognosen deutlich ungenauer werden. Um die optimale Funktion zu finden benutzen wir das Verfahren der Kreuzvalidierung.

Wir haben zuerst unsere Funktion auf Basis jedes zweiten Datenpunktes, mithilfe der pylab.polyfit Funktion bestimmt. Nun wurde der Fehler, bezüglich aller Datenpunkte, mithilfe der folgenden Formel berechnet (R-squared Methode):

Dabei wird der Fehler zwischen Daten und Vorhersage berechnet und quadriert. Der Fehler bewegt sich damit zwischen einem Wert von null und eins, wobei eins als optimal und null als schlechtester Fit zu bewerten ist. Da wir unseren Fit nur auf Basis jedes zweiten Datenpunktes erstellt haben, können wir nun eine Bewertung hinsichtlich aller Datenpunkte treffen (Kreuzvalidierung) und eine Aussage über die Prognose der Funktion machen. Die Ergebnis haben wir hier aufbereitet:



Wir kamen zu folgendem Ergebnis: Die Funktion dritten Grades weißt den niedrigsten Fehler auf. Betrachtet man die Funktion des Fehlers, so fällt auf, dass R2 für eine Funktion ersten Grades (lineare Funktion) null ist und einen sehr schlechten Fit darstellt. Der beste Fit wird bei einer Funktion dritten Grades erreicht.

## Aufgabe 2c)

Um die beiden Gleichungen zu lösen wird erst die eine Gleichung umgestellt und dann in die andere eingesetzt. Mithilfe der SymPy Bibliothek ist dies in einem Schritt mit der solve() Methode zu erreichen:

x, y, a, b = symbols('x y a b')

solu = solve((b - a\*y - x\*\*2 \* y, -x + a \* y + x\*\*2 \* y ), (y, x))

Nun kommen wir auf die beiden Lösungen:

ii) Die beiden Gleichungen werden umgestellt: Die zweite Gleichung lösen wir wieder mit SymPy nach auf:

x,y,b,a = symbols("x y b a")

soly = solve(-x + a \* y + x\*\*2 \* y,y)

Dies liefert folgendes Ergebnis:

Da wir wissen das ist kann wie folgt nach x aufgelöst werden:

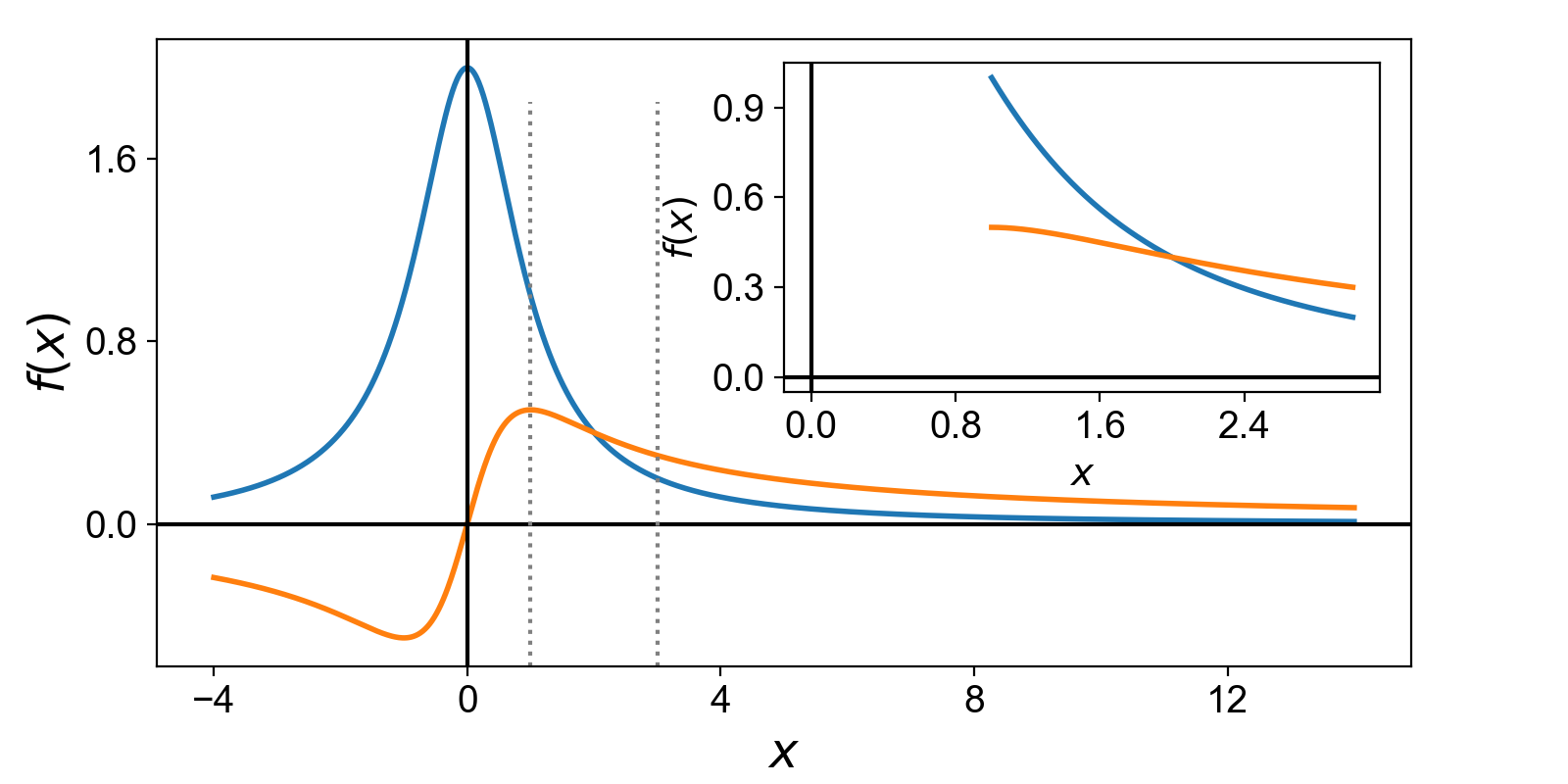
Bei einsetzen von y kürzt sich und es bleibt .

Die Fixpunktiteration muss nun neben einem x-Wert in jeder Iteration auch den y-Wert berechnen. Wenn wir die obigen Gleichungen in die Fixpunktiteration einsetzen, sehen wir dass das Ergebnis oszilliert, also immer zwischen null und neununddreißig hin- und herspringt. Die Gleichungen müssen also in eine andere Form gebracht werden:

Die erste Gleichung wird nach x aufgelöst, die zweite nach y. So ergeben sich folgende Ergebnisse:

Eingesetzt in die Fixpunktiteration ergeben sich folgende Lösungen:

Betrachten wir nun die Lösung grafisch, indem wir beide Gleichungen nach umstellen:



Die Lösung (2/0.4) ist plausibel.

## Aufgabe 2d)

Um den Lagrange Punkt L1 zu finden wird die Gleichung mithilfe von SymPy nach aufgelöst:

G,M,R,r,m,w = symbols('G,M,R,r,m,w')

sol = solve((G\*M)/(r\*\*2)-(G\*m)/((R-r)\*\*2),(w\*\*2 \* r), r)

Dies liefert das Ergebnis:

Nun werden die gegebenen Variablen eingesetzt. Durch die Wurzel kommt es zu zwei verschiedenen Lösungen:

Da der Abstand zwischen Erde und Mond aber nur 384.400 ist und der Lagrange Punkt L1 zwischen Erde und Mond liegt kommt nur die Lösung in Betracht, da der andere Punkt sich hinter dem Mond befindet.