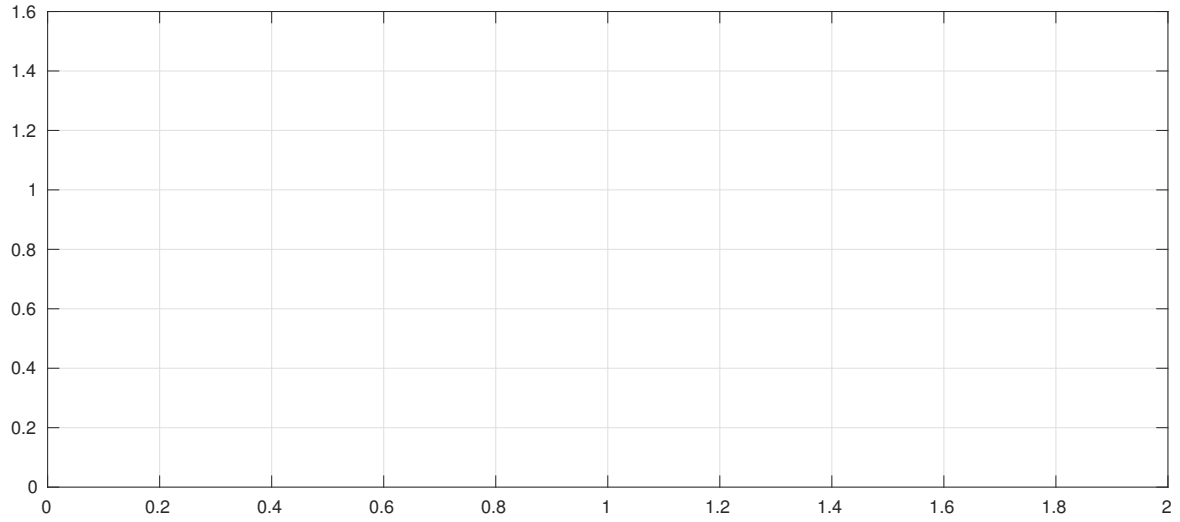
Tutorial: Mit Taylor alles auf die Reihe kriegen

Aufgabenstellung: Wir wollen eine Taylorentwicklung für die Wurzelfunktion

$$f(x) = \sqrt{x}$$

aufstellen und experimentell den Fehler untersuchen.

Schritt A: Graphische Anschauung & theoretische Überlegungen (erinnern Sie sich!)



1. Zeichnen Sie den Graphen $f(x) = \sqrt{x}$ in das obige Achsenkreuz. Werten Sie die Funktion dazu (ohne TR!) bei

   $f(0) =$ _____ $f(1) =$ _____ $f(2) =$ _____ aus.

2. Notieren Sie sich noch einmal die Tangentengerade $T_{f,x_0}(x)$ an eine beliebige Funktion f im Punkt $(x_0, f(x_0))$ (ganz allgemein; ohne nachschlagen :):

   $T_{f,x_0}(x) =$ _____

3. Berechnen Sie die Tangentengerade $T_{f,1}(x)$ am Punkt $(1, f(1))$ von $f(x) = \sqrt{x}$.

   $T_{f,1}(x) =$ _____

4. Zeichnen Sie die Tangentengerade $T_{f,1}(x)$ in das obige Achsenkreuz. Werten Sie die Funktion dazu (ohne TR!) bei

   $T_{f,1}(0) =$ _____ $T_{f,1}(1) =$ _____ $T_{f,1}(2) =$ _____ aus.

5. Notieren Sie sich noch einmal das Taylorpolynom n-ten Grades um den Entwicklungspunkt x_0 einer beliebigen Funktion f (ganz allgemein; ohne nachschlagen :):

   (mit Summenzeichen) $T_{f,n}(x, x_0) =$ _____

   (mit Pünktchen) $T_{f,n}(x, x_0) =$ _____

6. Berechnen Sie das Taylorpolynom 2-ten grades $T_{f,2}(x, 1)$ am Punkt $(1, f(1))$ von $f(x) = \sqrt{x}$.

   $T_{f,2}(x, 1) =$ _____

7. Zeichnen Sie $T_{f,2}(x, 1)$ in das obige Achsenkreuz. Werten Sie die Funktion dazu (ohne TR!) an folgenden Stellen aus:

   $T_{f,2}(0, 1) =$ _____ $T_{f,2}(1, 1) =$ _____ $T_{f,2}(2, 1) =$ _____

8. Nun kommen wir zur **Königsetappe**: Wie lautet allgemein die n-te Ableitung $f^{(n)}(x)$ der Funktion $f(x) = \sqrt{x}$? Berechnen Sie so viele Ableitungen bis Sie ein Muster erkennen. **Tipp:** Das Muster beginnt erst ab der 2-ten Ableitung.

   $f^{(n)}(x) =$ _____ , $n = 0$

   $f^{(n)}(x) =$ _____ , $n = 1$

   $f^{(n)}(x) =$ _____ , $n \geq 2$

9. Berechnen Sie das Taylorpolynom $T_{f,n}(x, x_0)$ der Funktion $f(x) = \sqrt{x}$ an einer beliebigen Stelle (welche sind erlaubt?) x_0 für beliebigen Grad n.

   $T_{f,n}(x, x_0) =$ _____

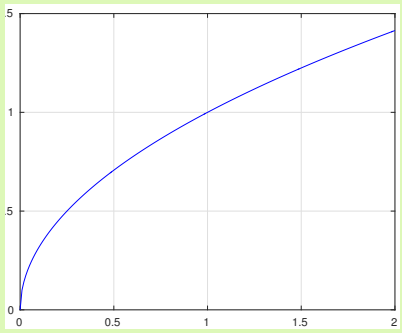
   Tipp: (a) Verwenden Sie Ihr Ergebnis aus Teil (9) und (b) Bringen Sie n und k nicht durcheinander!

Schritt B: Mit dem Algorithmus Kurven ertasten

10. Wir fangen einmal ganz langsam an: Erstellen Sie eine Datei **TutoTaylor.m** und darin die Funktion `function TutoTaylor()` mit dem Inhalt

```
function TutoTaylor()

clear all
close all

x=0.1:0.1:2; % Vorsicht: x darf
nicht Null sein. Warum nicht?
y=sqrt(x)

plot(x,y,'b-')
grid on

end
```

So sollte Ihr plot aussehen:



11. Im nächsten Schritt schreiben wir eine Unterfunktion namens **y=DSQRT(x,n)** im Anschluss an die Funktion `TutoTaylor` (nach `end` in der letzten Zeile), welche die n-te Ableitung der Funktion $f(x) = \sqrt{x}$ an der Stelle x ausgibt.

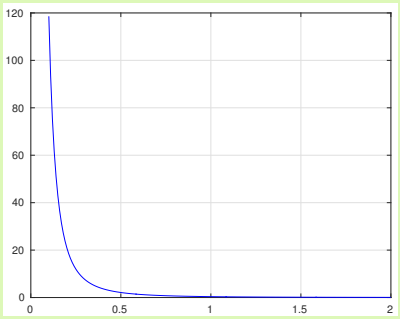
```
function y = DSQRT(x,n)

switch n
  case 0 % die Funktion f
    y= ...

  case 1 % erste Ableitung der Funktion f
    y= ...

  otherwise % n-te Ableitung der Funktion f mit n≥ 2
    prod=1;
    for j=0:n-2
      prod = prod* ...
    end
    y = prod* ...
end

end
```

Tipp: Ihr Ergebnis aus Punkt 8 kann Ihnen hier eine Hilfe sein.

12. Die Funktion `DSQRT(x,n)` brauchen wir dann für die Berechnung des Taylorpolynoms. Es liefert ja gerade $f^{(n)}(x) = \frac{d^n}{dx^n}\sqrt{x}$. Es ist sinnvoll wenn man in kleinen Schritten arbeitet und jede Funktion für sich prüft. **Testen** Sie mit Ihrem Programm `TutoTaylor()`, ob die Funktion `DSQRT(x,n)` fehlerfrei rechnet. Etwa indem Sie 0-te, 1., 2. und vielleicht auch noch 3. Ableitungen berechnen lassen. Am Beispiel dritte Ableitung sollte es bei Ihnen nun so aussehen:

```
function TutoTaylor()

clear all
close all

x=0:0.1:2;
y=DSQRT(x,3)

plot(x,y,'b-')
grid on

end
```

So sollte Ihr plot aussehen:



Nur die Graphik anschauen, gibt allenfalls einen qualitativen Eindruck, quantitativ fühlen Sie der Routine auf den Zahn, indem Sie mit (sehr wahrscheinlich) korrekten Zahlen vergleichen. Zum Beispiel, in dem Sie die Norm bilden aus der Differenz des durch `DSQRT` erzeugten Feldes `y` mit dem Feld `w`, dass Sie direkt mit der Funktion der dritten Ableitung gefüllt haben. Also so:

```
K>> w=3/8./sqrt(x).^5;
K>> norm(w-y)

ans =

     0
```

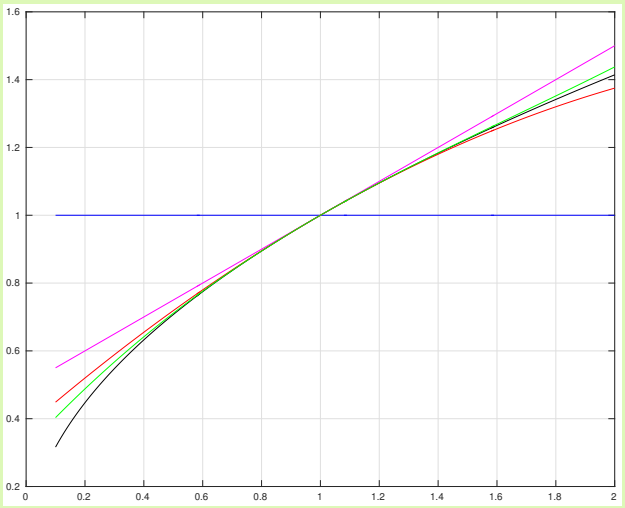
13. Jetzt der **Endspurt**: Eine Funktion namens `TaPoSQRT(x,x0,n)`, welche das Taylorpolynom vom Grad n, entwickelt um x_0, von der Funktion $f(x) = \sqrt{x}$ kann jetzt ganz einfach (doch wirklich) aufgestellt werden.

```
function y=TaPoSQRT(x,x0,n)

y=0;
for k=0:n
  y = y + DSQRT( ...
end

end
```

Tipp: Den Inhalt des Platzhalters können Sie direkt aus Punkt 5 (der Teil mit Summenzeichen) entnehmen.

14. Rufen Sie nun `TaPoSQRT(x,x0,N);` für verschiedene Werte für x_0 und N an. Diskutieren Sie Ihr Ergenis. Verstehen Sie was da passiert?

So sollte Ihr plot aussehen:



```
function TutoTaylor()

clear all
close all

x=0.1:0.1:2;
x0 = 1;

ye = DSQRT(x,0);
y0 = TaPoSQRT(x,x0,0);
y1 = TaPoSQRT(x,x0,1);
y2 = TaPoSQRT(x,x0,2);
y3 = TaPoSQRT(x,x0,3);


plot(x,ye,'k-',x,y0,'b-',x,y1,'m-',x,y2,'r-',x,y3,'g-');
grid on

end
```

Sie können auch einmal den Bereich von x variieren. **Tipp:** Wenn die graphische Ausgabe wenig aussagekräftig ist, weil die Näherung im Bereich betragsmäßig große Werte annimmt, dann beschränken Sie den Bildbereich einfach mit dem Befehl

```
ylim([0 sqrt(max(x))]);
```

Nach dem Plot-Befehl. Es `ylim([a b])` begrenzt den Bildbereich (y-Achse) auf das Intervall $[a\,,\,b]$.

15. Beim Ergebnis aus 14: Würden Sie aus rein qualitativen Gesichtspunkten sagen: Mit Erhöhung des Polynomgrades wird die Näherung

    ☐ besser ☐ schlechter .

16. Füren Sie mal bitte die Rechnung mit mit $x_0 = 1$ auf dem Intervall $[0.01, 3]$ durch und wählen Sie für N Werte aus der Menge $\{0, 1, 2, 10, 20, 30\}$ durch. **Tipp:** Mit

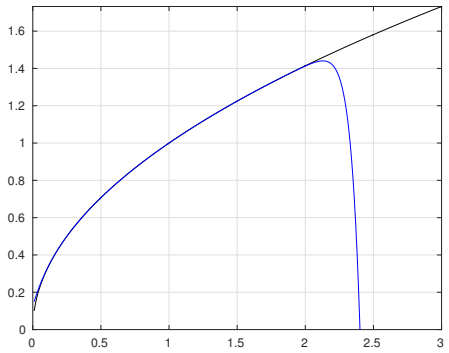
```
plot(x,ye,'k-',x,y,'b-');
ylim([0 sqrt(max(x))]);
grid on
```

    "sehen" Sie besser. Was beobachten Sie? Erkennen Sie einen Konvergenzbereich? Welchen?

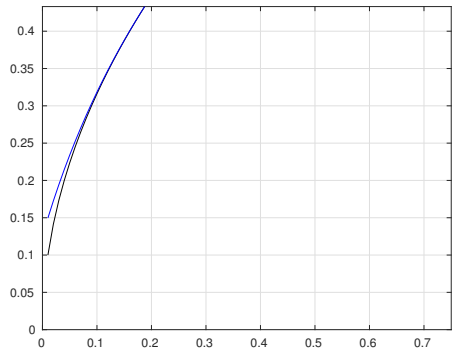
    $K = (\_\_\_\_\_, \_\_\_\_\_)$

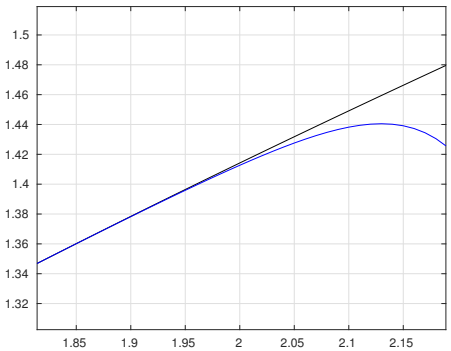
    Tipp: Zoomen Sie mit der Maus in der Graphik in die kritischen Stellen hinein.

    Sehen Sie sowas?



Zoom bei $x = 0$ und $x = 2$:





17. Wiederholen Sie 16 mit $x_0 = 2$ auf dem Bereich $[0.01, 5]$. Was beobachten Sie nun?

    $K = (\_\_\_\_\_, \_\_\_\_\_)$

18. Können Sie aus den Ergebnissen von 16 und 17 eine allgemeine Vermutung aufstellen? Wie könnte der Konvergenzbereich für die Reihe bei einem beliebigen Entwicklungspunkt x_0 aussehen?

    $K = (\_\_\_\_\_, \_\_\_\_\_)$

    Überlegen Sie sich noch eine Stichprobe.

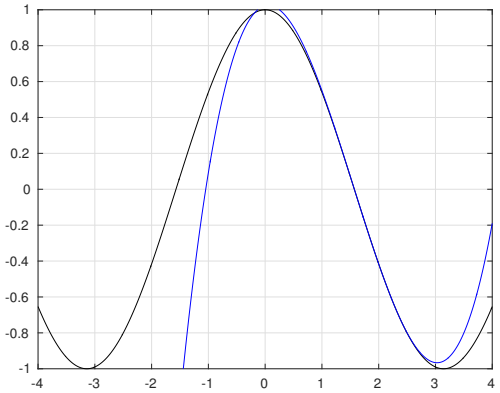
Schritt C: Nachhaltigkeit auch beim Programmieren

Es wird ein wenig langweilig, wenn man immer nur mit einer Funktion spielt, aber unsere Funktion `TaPoSQRT` ist leider auf $f(x) = \sqrt{x}$ fixiert, weil sie die Funktion `DSQRT` aufruft, welche die Ausdrücke $f^{(n)}(x)$ liefert. Etwas modularer funktioniert das Ganze, wenn statt dessen eine beliebige Funktion aufgerufen werden kann. Das erreichen wir, indem wir der Routine, die das Taylorpolynom aufstellt einen Zeiger auf eine Funktion übergeben. Worauf dieser Funktionszeiger dann zeigt, kann variieren. Wir behalten die Datei `TutoTaylor.m` und erzeugen eine neue:

19. Erzeugen Sie eine Datei `MyTaylor.m`, die eine entsprechende Funktion namens `function MyTaylor()` ohne Übergabewerte enthält und kopieren Sie zunächst den Inhalt aus `TutoTaylor.m` komplett dort hinein. Jetzt gibt es nur wenige Änderungen:

    - Die Routine `TaPoSQRT` erhält bei Definition und Aufruf den Namen `MyTaylorpolynom`. Das ist nicht wichtig aber es ist gut verschiedenen Kindern auch verschiedene Namen zu geben.
    - In der Definition von `MyTaylorpolynom` wird die Funktionsvariable `df` noch in die Übergabeliste aufgenommen - so:
      ```
      function y=MyTaylorpolynom(x,df,x0,n)
      ```
      - und das `DSQRT` beim Aufruf durch `df` ersetzt.
    - Beim Aufruf von `MyTaylorpolynom` übergeben wir nun den Zeiger auf `DSQRT`. So:
      ```
      y = MyTaylorpolynom(x,@DSQRT,x0,N);
      ```
      Das war's schon. Das neue Programm sollte nun funktionieren wie das alte.

20. Schreiben Sie nun eine neue Funktion, die die n-te Ableitung einer Funktion liefert und zwar für
    $$f(x) = \cos x$$
    namens `DCOS(x,n)`. Übergeben Sie dann einen Zeiger auf `DCOS` statt auf `DSQRT`, also so
    ```
    y = MyTaylorpolynom(x,@DCOS,x0,N);
    ```
    Spielen Sie ein wenig herum. Auf dem Interall $[-4, 4]$ mit dem Entwicklungspunkt $x_0 = 2$ erhalten Sie für $N = 3$ folgende Ausgabe:


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-4
-3
-2
-1
0
1
2
3
4