Lernhilfe zur Vorlesung: Kap. 11, Numerische Lösung von GDGL

Anfangswertaufgabe

Wir ziehen das Beispiel aus der Vorlesung nach. Wir betrachten also das Anfangswertproblem

$$y'(t) = 1 + y(t)^2, y(0) = 0$$

auf dem Interval $[0, 1.4]$. Mit den Definitionen aus dem Satz von Picard-Lindelöf haben wir damit

$$f(t, y) = 1 + y^2, a = 0, b = 1.4, y_0 = 0.$$

Wie in der Vorlesung dargelegt, ist diese Funktion zwar nicht global, aber lokal lipschitzstetig, was für den Satz bereits ausreicht. Wir wissen also, dass es auf einem Intervall $[0, \varepsilon]$ eine Lösung gibt. Tatsächlich existiert die Lösung auf dem Intervall $[0, \pi/2)$, es gilt

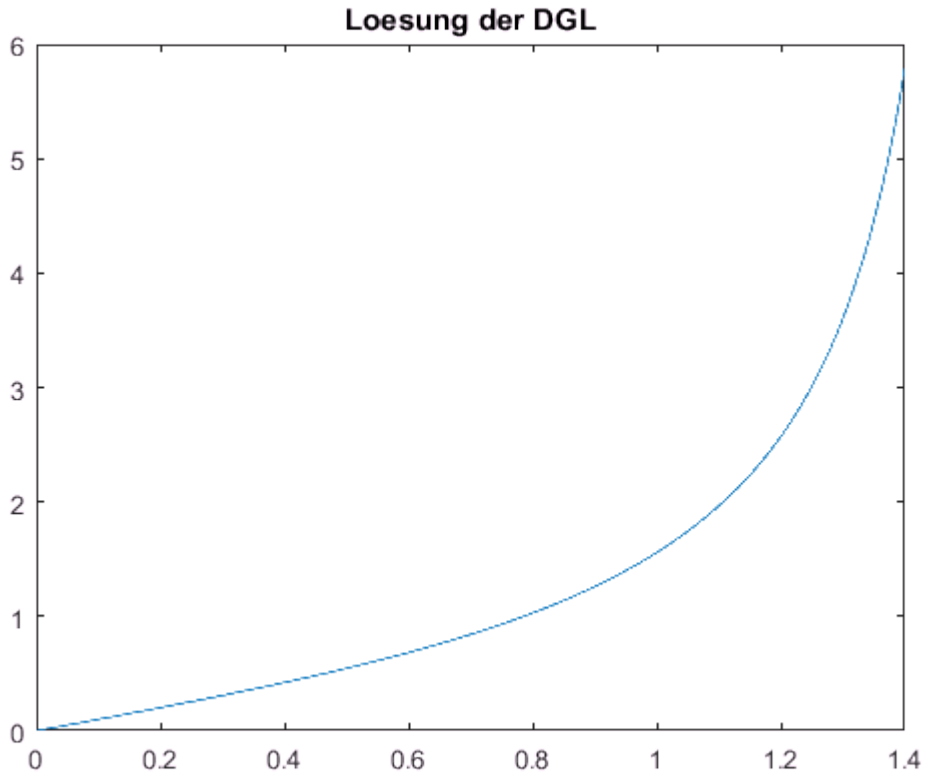
$$\widetilde{y}(x) = tan(x)$$

löst das Anfangswertproblem. Wir übernehmen diese Angaben.

```
f=@(t,y) 1+y^2;
a=0;
b=1.4;
y0=0;
yloesung=@(x) tan(x);
```

Zunächst mal plotten wir die Lösung.

```
N=10000;
h=(b-a)/N;
x=a:h:b;
plot(x,yloesung(x));
title('Loesung der DGL');
```



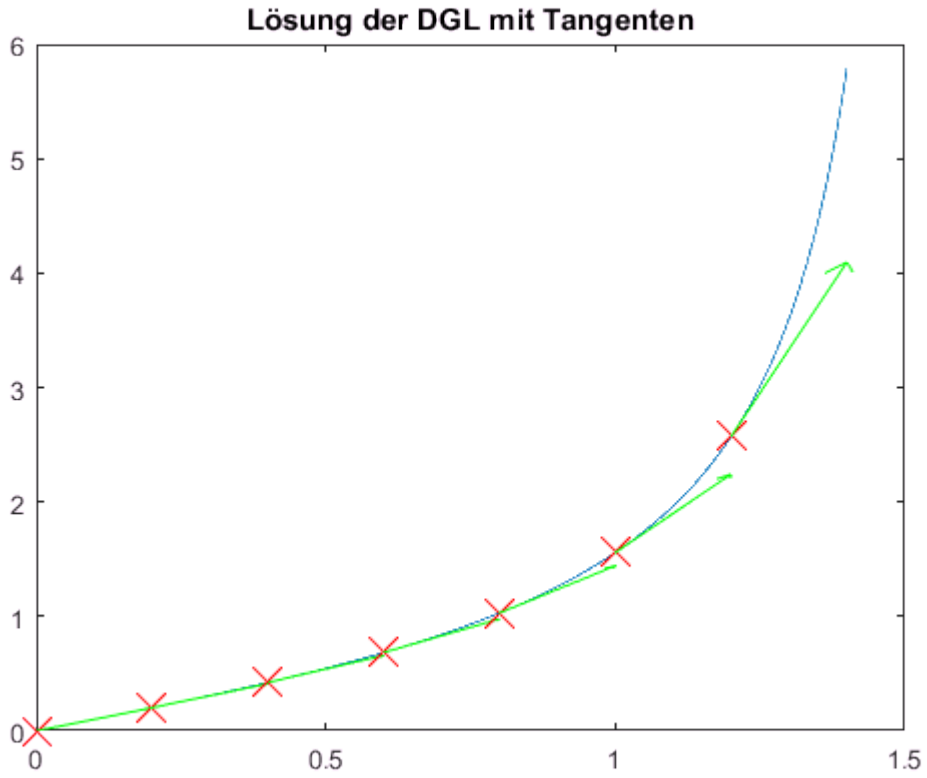
Numerische Approximation

Diese wollen wir nun numerisch approximieren. Wir fragen uns zunächst: Was bedeuten die Angaben der Anfangswertaufgabe geometrisch für die gesuchte Kurve?

Zum Anfangswert: wir wissen, die Kurve startet im Punkt (0,0), geht also durch diesen Punkt.

Zur Differentialgleichung: falls die gesuchte Kurve durch den Punkt (t,z) geht, also $y(t) = z$, so hat die Kurve dort die Tangentensteigung $y'(t) = f(t,z)$. Wir zeichnen in unseren Plot für einige Punkte die Steigungen ein und überpüfen, dass die Tangentensteigung korrekt ist.

```
plot(x,yloesung(x));
hold on;
M=7;
for i=0:M-1
    t=a+(b-a)*i/M;
    h=(b-a)/(2*M);
    plot(t,yloesung(t),'Xr','MarkerSize',15);
    abl=f(t,yloesung(t));
    %line([t t+h],[yloesung(t) yloesung(t)+h*abl],'Color','green');
    quiver(t,yloesung(t),h,h*abl,'AutoScaleFactor',2,'Color','green');
end
hold off;
title('Lösung der DGL mit Tangenten');
```



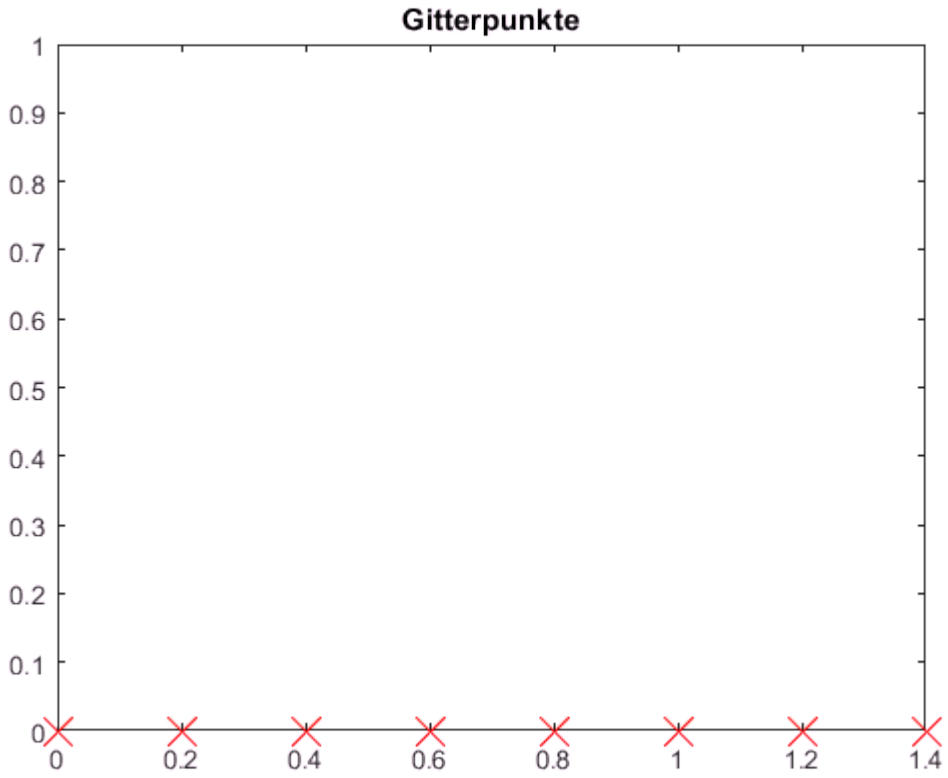
Ok, das scheint zu stimmen. Wir versuchen jetzt, diese Kurve numerisch zu approximieren. Die Idee ist: Wir wählen einige Werte $a = t_0 \dots t_M = b$ im Intervall $[a, b]$ und versuchen, an diesen Punkten Näherungen $y_0 \dots y_M$ für den Wert der Funktion y bzw. die Punkte der Kurve zu finden. Wir beginnen zunächst mal mit 5 Punkten und zeichnen deren x-Positionen.

Achtung: Matlab und Octave numerieren ab 1, nicht Null, deshalb ist in den Skripten der Index dort immer einen höher. Im Fließtext folgen wir den Angaben der Vorlesung.

```
M=7;
h=(b-a)/M;
t=a+(0:M)*h
```

```
t =
         0    0.2000    0.4000    0.6000    0.8000    1.0000    1.2000    1.4000
```

```
y=zeros(M+1,1);
plot(t,0,'Xr','MarkerSize',15);
ylim([0 1]);
title('Gitterpunkte');
```



Da wir wissen, dass die Kurve im Punkt (t_0, y_0) beginnt, können wir diesen Wert gleich mal eintragen.

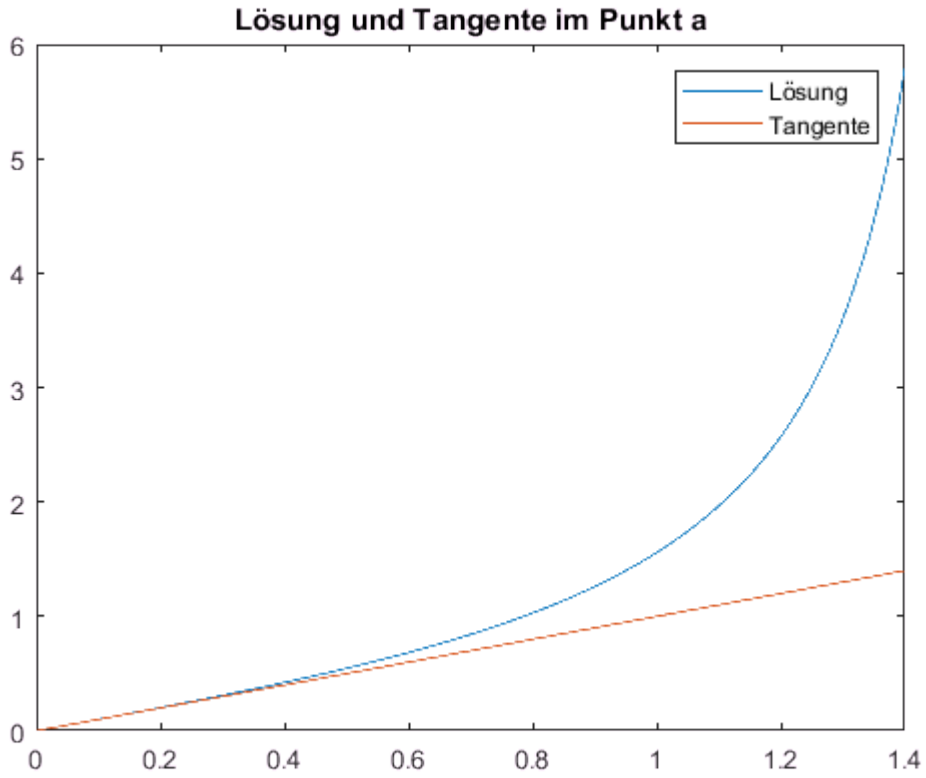
```
y(1)=y0;
```

Am Punkt t_1 kennen wir die Kurve nicht. Wir kennen aber im Punkt (t_0, y_0) die Tangente. Sie hat die Gleichung

$$y(t) = y_0 + (t - t_0) f(t_0, y_0).$$

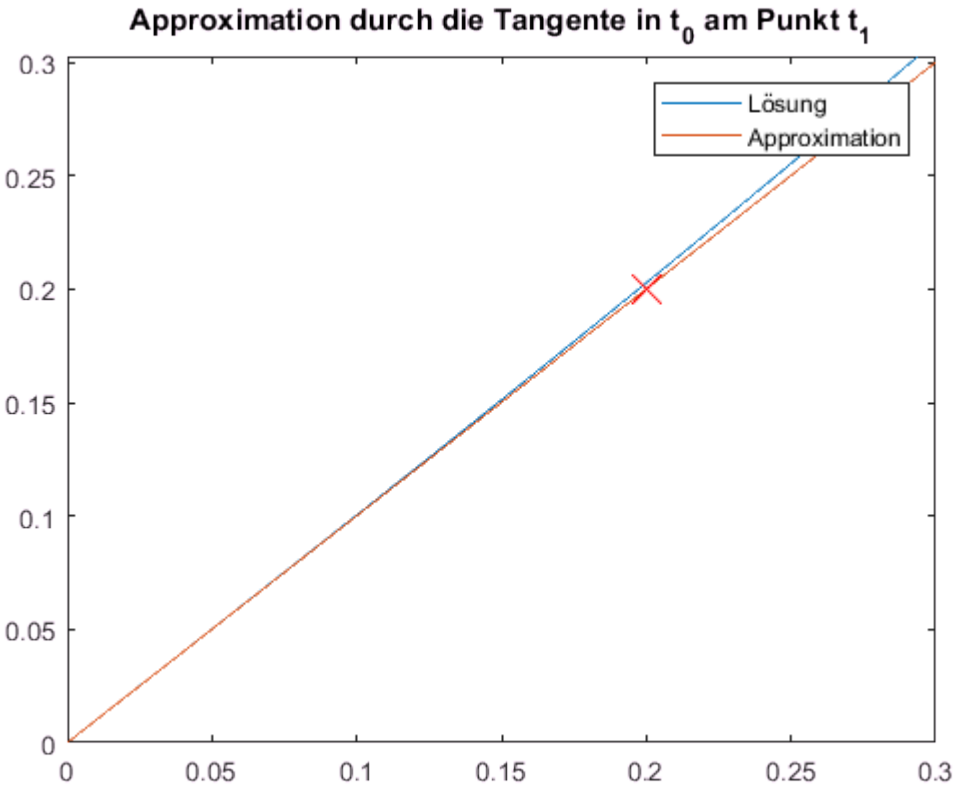
Wir plotten Kurve und Tangente.

```
plot(x,yloesung(x),x,y(1)+(x-a)*f(t(1),y(1)));
title('Lösung und Tangente im Punkt a');
legend('Lösung','Tangente');
```



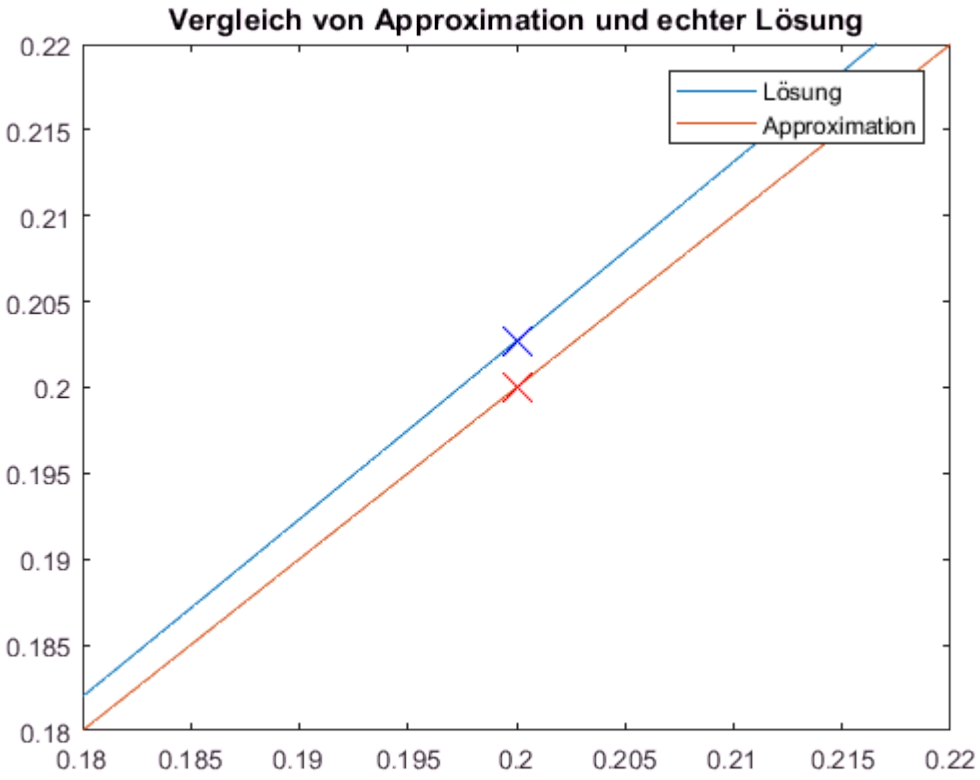
In der Nähe des Anfangswerts ist die Tangente eine gute Nächerung für die Funktion. Wir approximieren daher den Wert der Funktion im Punkt t_1 durch den Wert der Tangente im Punkt t_1, setzen also t_1 in die Tangentengleichung ein. Dieses Verfahren heißt **Eulersches Polygonzugverfahren**. Wir erhalten

```
% Tangentensteigung im Punkt (t_0,y_0)
m=f(t(1),y(1));
% t_1 eingesetzt in die Tangentengleichung
y(2)=y(1)+(t(2)-t(1))*m;
plot(x,yloesung(x),x,y(1)+(x-t(1))*f(t(1),y(1)));
hold on;
plot(t(2),y(2),'Xr','MarkerSize',15);
hold off;
title('Approximation durch die Tangente in t_0 am Punkt t_1');
xlim([t(1) t(2)+0.1]);
ylim([y(1) 0.1+max(y(2),yloesung(t(2)))]);
legend('Lösung','Approximation');
```



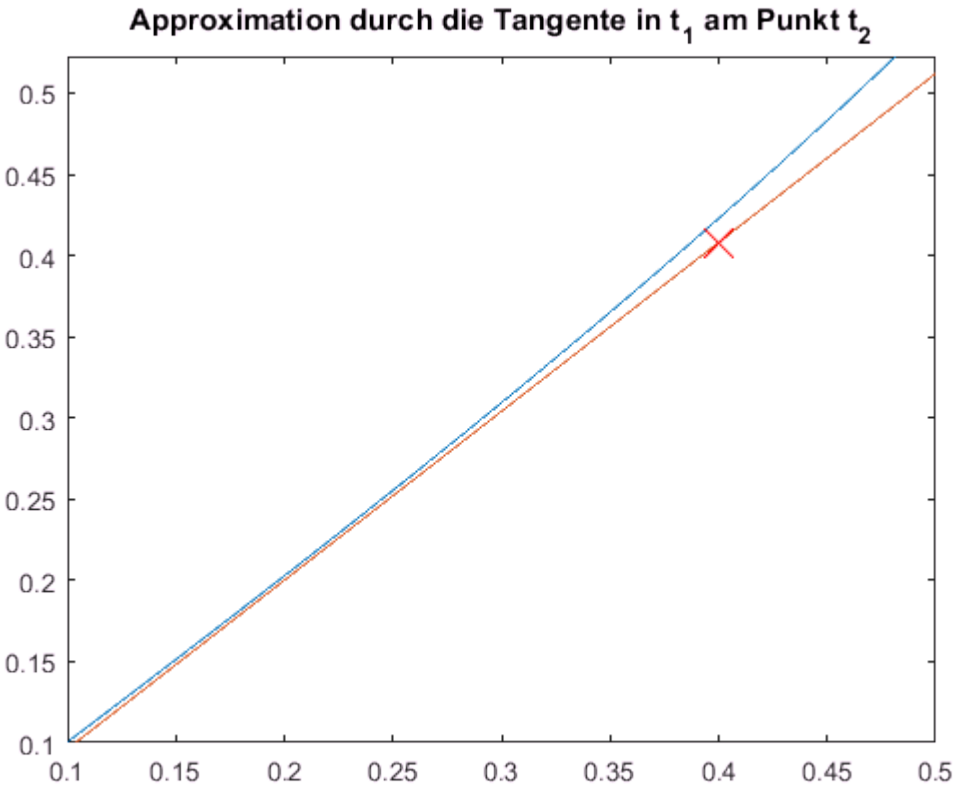
Natürlich ist diese Näherung nicht exakt, Wir vergrößeren die Zeichnuing an diesem Punkt.

```
plot(x,yloesung(x),x,y(1)+(x-a)*f(t(1),y(1)));
hold on;
plot(t(2),y(2),'Xr','MarkerSize',15);
plot(t(2),yloesung(t(2)),'Xb','MarkerSize',15);
hold off;
xlim([t(2)-0.02 t(2)+0.02]);
ylim([y(2)-0.02 y(2)+0.02]);
title('Vergleich von Approximation und echter Lösung');
legend('Lösung','Approximation');
```



Dies führen wir nun genau so fort. Wir approximieren die Kurve durch die Tangente am Punkt (t_1,y_1) und berechnen daraus die Approximation y_2 am Punkt t_2.

```
% Tangentensteigung im Punkt (t_1,y_1)
m=f(t(2),y(2));
% t_1 eingesetzt in die Tangentengleichung
y(3)=y(2)+(t(3)-t(2))*m;
plot(x,yloesung(x),x,y(2)+(x-t(2))*f(t(2),y(2)));
hold on;
plot(t(3),y(3),'Xr','MarkerSize',15);
hold off;
title('Approximation durch die Tangente in t_1 am Punkt t_2');
xlim([t(2)-0.1 t(3)+0.1]);
ylim([y(2)-0.1 0.1+max(y(3),yloesung(t(3)))]);
```



Lokaler und globaler Diskretisierungsfehler

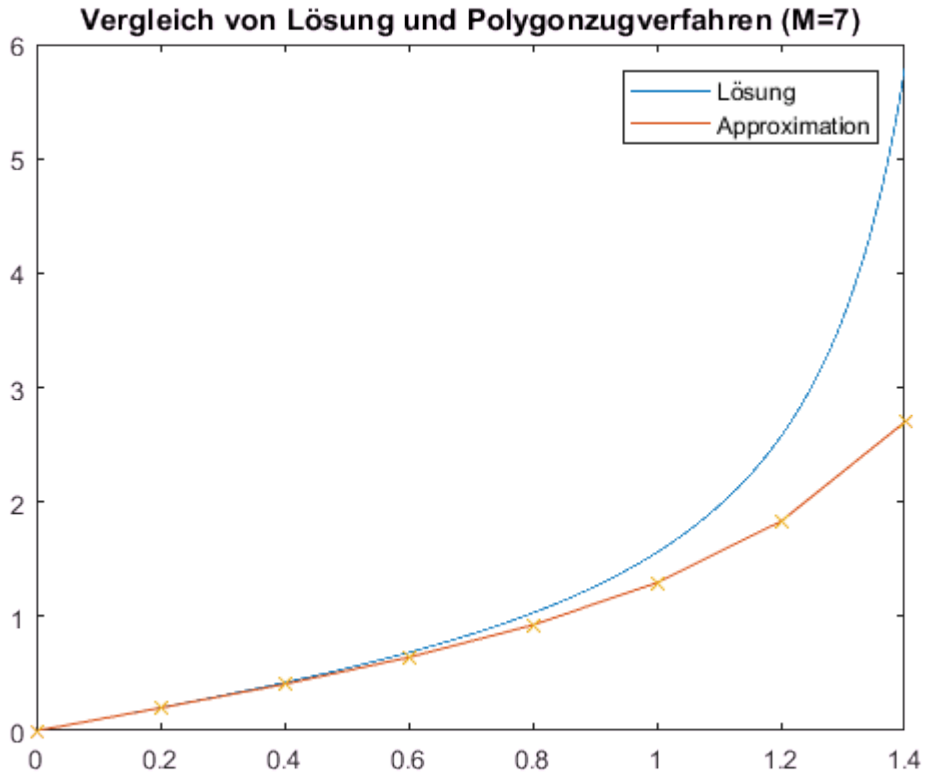
Dabei machen wir jetzt zwei Fehler: Einmal folgen wir, wie im ersten Schritt, nicht exakt der Kurve, sondern nur wieder der Tangente. Dadurch entsteht ein Fehler, diesen nennen wir den lokalen Diskretisierungsfehler - den Fehler, der in diesem Schritt entsteht. Diesen Fehler kann man leicht ausrechnen mit Taylorentwicklung. Wir skalieren ihn üblicherweise nochmal mit $1/h$ (siehe Definition in der Vorlesung).

Zusätzlich aber: Der Punkt (t_1,y_1), von dem wir ausgehen, liegt zusätzlich nicht mal auf der gesuchten Kurve, denn der war ja selbst nur eine Approximation. Den Gesamtfehler, also $y(t_2) - y_2$, nennen wir globalen Diskretisierungsfehler. Dieser Fehler ist schwer zu berechnen.

Durchführung für die komplette Kurve

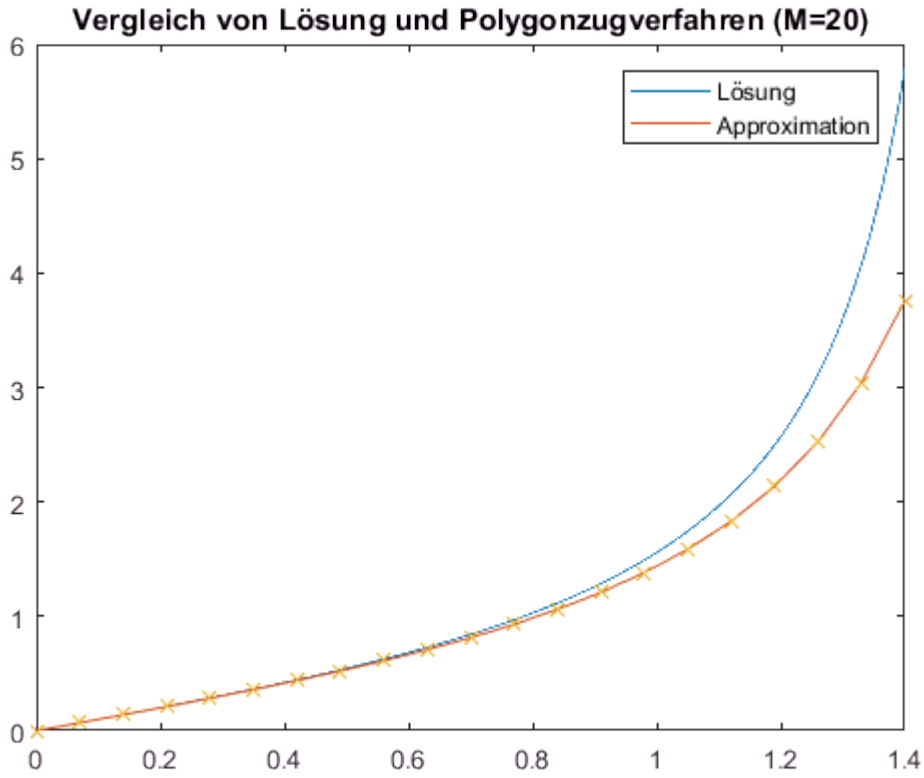
Wir machen das nun für alle Diskretisierungspunkte.

```
h=(b-a)/M;
t=a+h*(0:M);
y(1)=y0;
for i=1:M
    m=f(t(i),y(i));
    y(i+1)=y(i)+m*h;
end
plot(x,yloesung(x),t,y,t,y,'X');
title(['Vergleich von Lösung und Polygonzugverfahren (M=' num2str(M) ')']);
legend('Lösung','Approximation');
```



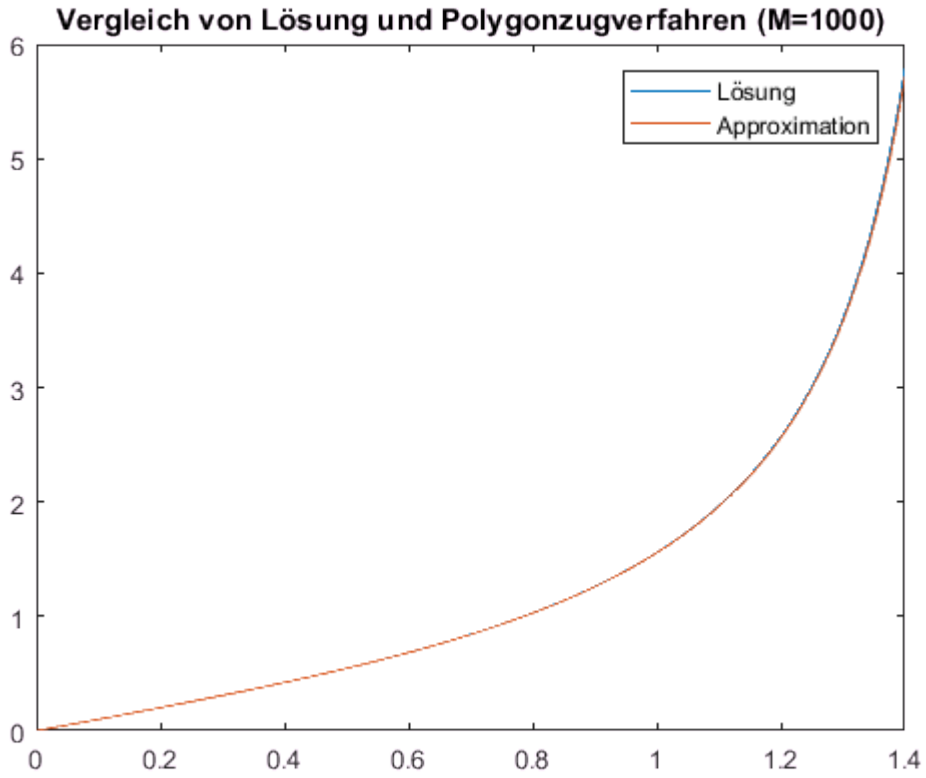
Die Approximation ist nicht sehr gut. Aber wir hatten das Gitter ja auch sehr grob gewählt. Also nun für ein größeres M:

```
M=20;
h=(b-a)/M;
t=a+h*(0:M);
y=zeros(M,1);
y(1)=y0;
for i=1:M
    m=f(t(i),y(i));
    y(i+1)=y(i)+m*h;
end
plot(x,yloesung(x),t,y,t,y,'X');
title(['Vergleich von Lösung und Polygonzugverfahren (M=' num2str(M) ')']);
legend('Lösung','Approximation');
```



Schon besser. Wir wählen nun einen sehr großen Wert für M.

```
M=1000;
h=(b-a)/M;
t=a+h*(0:M);
y=zeros(M,1);
y(1)=y0;
for i=1:M
    m=f(t(i),y(i));
    y(i+1)=y(i)+m*h;
end
plot(x,yloesung(x),t,y);
title(['Vergleich von Lösung und Polygonzugverfahren (M=' num2str(M) ')']);
legend('Lösung','Approximation');
```



Konvergenz

Hier ist die Approximation tatsächlich schon sehr gut. Es scheint also zu gelten: Je mehr Werte man für die Auflösung wählt, je kleiner der Unterschied h zwischen zwei Punkten wird, unso kleiner wird der Unterschied zwischen den Kurven, also der globale Diskretisierungsfehler.

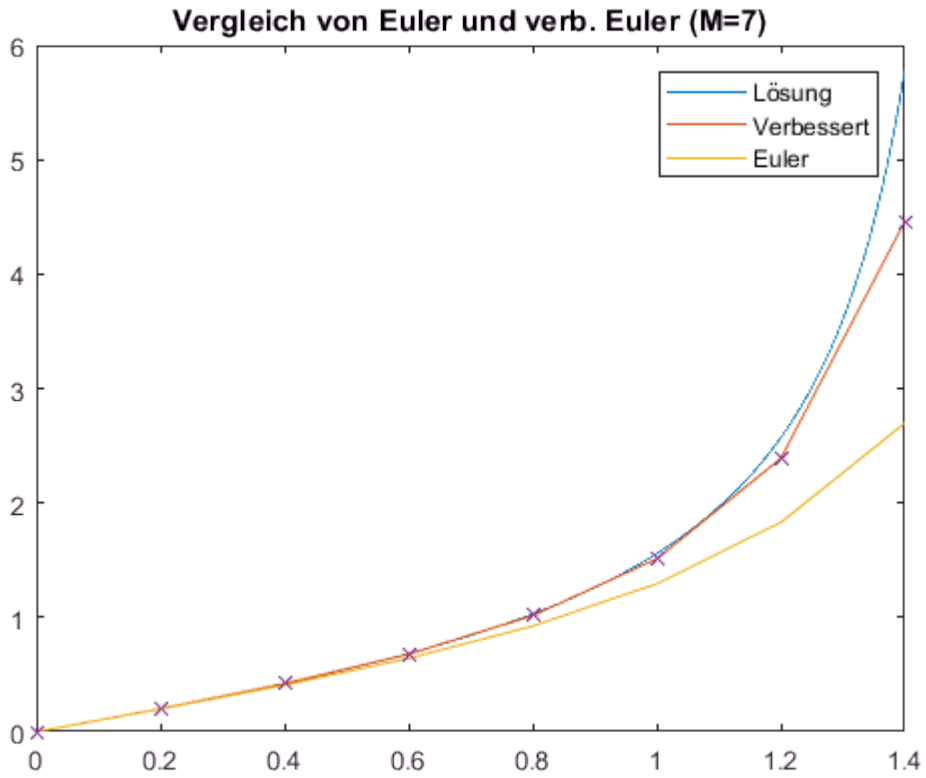
Dies bezeichnen wir als Konvergenz eines Verfahrens: Geht die Gitterweite h gegen 0, so sollte auch der maximale globale Diskretisierungsfehler gegen Null gehen.

Alternative Verfahren

Oben sind wir einfach der Tangente gefolgt, um die nächste Approximation zu berechnen. Dies führte zum Eulerschen Polygonzugverfahren. Man kann sich fragen: Gibt es bessere Techniken? In der Vorlesung haben wir mit dem verbesserten Eulerverfahren etwas anders diskretisiert. Mit der dort definierten Funktion φ gilt

```
M=7;
h=(b-a)/M;
phi=@(t,y,h) f(t+h/2,y+h/2*f(t,y));
t=a+h*(0:M);
y=zeros(M,1);
yeuler=zeros(M,1);
y(1)=y0;
yeuler(1)=y0;
for i=1:M
    mverb=phi(t(i),y(i),h);
    y(i+1)=y(i)+mverb*h;
    yeuler(i+1)=yeuler(i)+f(t,yeuler(i))*h;
end
plot(x,yloesung(x),t,y,t,yeuler,t,y,'X');
```

```
title(['Vergleich von Euler und verb. Euler (M=' num2str(M) ')']);
legend('Lösung','Verbessert','Euler');
```



Ich denke, der Unterschied ist klar ersichtlich. Der globale Diskretisierungsfehler für das verbesserte Verfahren ist deutlich kleiner. Tatsächlich wurde in der Vorlesung gezeigt: Das Verfahren ist von der Ordnung 2, geht also gegen 0 wie h^2. Allerdings muss man berücksichtigen, dass hier die Funktion in jedem Schritt zweimal ausgewertet wird - ein Schritt dauert also etwas länger.