Programm zur Modellierung des kippenden Besenstiels

Zuerst wurde das Zeitschritt-Verfahren als Programm geschrieben. Dieses Prinzip basiert darauf, dass man einen Punkt hat an dem man die Beschleunigung und damit die Geschwindigkeit berechnet. Nun schätzt man mit der Steigung bei diesem Punkt, die Steigung über ein Zeitintervall δt ab und schätzt damit den nächsten Punkt ab. Von dort aus wird der Vorgang bis zum Ende der Annäherung wiederholt. Durch dieses Verfahren ist es numerisch möglich, weitere Steigungen abzuschätzen ohne eine Funktion zu haben. Um dieses Verfahren zu übernehmen wurde zuerst ein Modul-Befehl verwendet. Dieser Befehl erlaubt es jeder Größe einen Wert zuzuordnen, ohne diesen darauf festzulegen. Also können im Programm die definierten Werte geändert werden. Dabei steht hier t für τ, z für die Zeit, p0 für den im Schritt eingesetzten Winkel, ps der Startwinkel, dp0 und dps deren erste Ableitungen, ddp die zweite Ableitung des Winkels, FktList der x/y Wert zum ausgeführten Schritt, d für den festgelegten Zeitschritt und l die Länge des Besenstiels. Die Grundlage dieses Programms liegt bei einer While-Schleife. Diese sorgt dafür, dass ein vorher definierter Vorgang so oft ausgeführt wird, bis eine Vorgabe nicht mehr erfüllt ist. In diesem Fall ist die Vorgabe, dass der Wert unter pi/2 liegt. Dies entspricht 90° und damit dem Moment, wo der Besen auf dem Boden liegt. Der Vorgang funktioniert wie folgt. Zuerst wird p0, welches mit einem Startwert im Modul definiert wird, in die Definition der zweiten Ableitung ddp eingesetzt. Damit wird dann mit der Definition der ersten Ableitung diese berechnet und in die Funktion des Winkels eingesetzt. Nachdem dieser dann auch berechnet wurde, werden die Werte der Zeit und des Winkels in die Liste der Ergebnisse eingetragen. Jetzt werden p0 und dp0 durch die berechneten p und dp ersetzt, um im nächsten Schritt im neuen Punkt zu rechnen. Darauf folgend wird überprüft, ob die Vorgabe noch erfüllt ist und der Vorgang beginnt von vorne.

```
KipMod[ps_, dps_, d_, l_] :=
 Module[{t, z = 0, p0 = ps, dp0 = dps, p  = 0 , dp = 0, ddp = 0, FktList = {{0, ps}}},
 Modul
  t = Sqrt[2 * l / (3 * 9.81)];
      Quadratwurzel
  While[p0 < π / 2, ddp = Sin[p0] / t^2;
  solange                  Sinus
    dp = dp0 + d * ddp;
    p = p0 + d * dp;
    z = z + d;
    AppendTo[FktList, {z, p}];
    hänge an bei
    p0 = p;
    dp0 = dp
  ];
  {z, FktList}]
```

Der Werte-Befehl ist dann dazu da, die Wertetabellen zu erhalten und zu exportieren, um sie später in QtiPlot einzufügen und dort die Graphen anzufertigen. Dazu muss aber erst einmal die Liste definiert werden. Diese beginnt wieder mit einem Module-Befehl um den Startwinkel p0 = 0 und die Liste zu Ausgabe der Werte als leere Liste zu definieren. Damit wird die Ausgangssitua-

tion bei mehrfacher Ausführung wiederhergestellt. Nun folgt eine For-Schleife. Diese führt ab einem Startwert eines Laufparameters einen Vorgang solange lange durch, bis die Vorgabe (2. Eintrag) nicht mehr erfüllt wird. Hier wird zuerst der Startwert von i auf s festgelegt, welches im Befehl selbst definiert werden kann. Die Vorgabe ist, dass i ≤ freq ist. Dabei beschreibt freq die Anzahl der Einträge in die Liste (ein Eintrag besteht aus Zeit und Winkel). Der Vorgang besagt, dass p0 zuerst auf den ersten Zeitwert gesetzt wird. Dieser ist über "ende*i/freq" definiert. Dabei steht "ende" für den letzten Wert der Messreihe. Also wird p0 zum i-ten Wert zwischen 0 und "ende". Danach folgt das Eintragen des Funktionswertes in die List für den Graphen. Dazu wird das Neue p0 eingesetzt und nach dem oberen Schema der Zeitpunkt berechnet, wo der Besen den Boden berührt und diese beiden Werte für p0 und z als Wertepaar eingetragen. Die For-Schleife endet dann mit dem Erhöhen des Laufparameters, um den nächsten Wert zu bestimmen. Wenn nach dem Erhöhen von i die Vorgabe nicht mehr erfüllt ist, endet das Programm. Nach der For-Schleife wird die Tabelle in Excel importiert. Dies geht mit dem Export-Befehl. ".qti" ist Mathematica nicht bekannt, weswegen man diesen Umweg gehen muss.

```
Werte[s_, freq_, ende_, d_, l_] := Module[{p0 = 0, FktList = {}},
                                    Modul
  For[i = s, i ≤ freq, p0 = ende * i / freq;
  For–Schleife
   AppendTo[FktList, {p0, KipMod[p0, 0, d, l][[1]]}];
   hänge an bei
   i++];
  Export["Fktlist1.xsl", FktList]]
  exportiere
```

Leider geht es nicht dem Namen einen Parameter zu geben. Deswegen wurde der Name separat im Programm selber geändert und ausgeführt. Darum ist hier auch nur ein Beispiel zu sehen.

```
Werte[0.05, 40, 1.5, 0.2, 1.45]
```

Out[8]=
```
{{0.001875, 2.4}, {0.039375, 1.4}, {0.076875, 1.2}, {0.114375, 1.}, {0.151875, 1.},
 {0.189375, 1.}, {0.226875, 0.8}, {0.264375, 0.8}, {0.301875, 0.8}, {0.339375, 0.8},
 {0.376875, 0.8}, {0.414375, 0.6}, {0.451875, 0.6}, {0.489375, 0.6}, {0.526875, 0.6},
 {0.564375, 0.6}, {0.601875, 0.6}, {0.639375, 0.6}, {0.676875, 0.6}, {0.714375, 0.4},
 {0.751875, 0.4}, {0.789375, 0.4}, {0.826875, 0.4}, {0.864375, 0.4}, {0.901875, 0.4},
 {0.939375, 0.4}, {0.976875, 0.4}, {1.01438, 0.4}, {1.05188, 0.4}, {1.08938, 0.4},
 {1.12688, 0.4}, {1.16438, 0.4}, {1.20188, 0.2}, {1.23938, 0.2}, {1.27688, 0.2},
 {1.31438, 0.2}, {1.35188, 0.2}, {1.38938, 0.2}, {1.42688, 0.2}, {1.46438, 0.2}}
```