DIPLOMARBEIT

Berechnen von Zustandsereignissen in dynamischen Rotor/Stator Modellen

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs
unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker
Insitut für Analysis und Scientific Computing, E101
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Horst Ecker
Institut für Mechanik und Mechatronik, E325

eingereicht an der Technischen Universität Wien
Fakultät für Mathematik und Geoinformation

von

Gernot Steiner
9525326
A-2340 Mödling, In den Messerern 6

Wien, im Jänner 2013

Kurzfassung

Die vorliegende Arbeit behandelt die Simulation von drei kontaktbehafteten Rotor-Stator-Systemen, insbesondere die Simulation des Kontaktvorgangs und dient zum Vergleich der Ergebnisse mit der Diplomarbeit "Numerische Untersuchung von selbsterregten Rotor-Stator-Systemen mit Begrenzung durch ein starres oder elastisches Fanglager"von Sebastian Popprath aus dem Jahr 2003, eingereicht an der TU Wien, Fakultät Maschinenbau.

Für die Analyse solcher kontaktbehafteten Rotor-Stator-Systeme sind aufwändige Berechnungen notwendig, da diese Systeme durch den Kontaktvorgang nichtlinear werden. Es wurden drei Modelle ausgewählt, die im Rahmen von numerischen Untersuchungen im Programm Maple und im Simulationsprogramm Maplesim getestet wurden.

Das erste Modell besteht aus einem Rotor mit starrem Stator mit kreisförmigen Lagerspiel. Der Kontaktvorgang wird mit Hilfe der Newton'schen Stoßhypothese modelliert. Zur Resultatsveranschaulichung wurden Bifurkationsdiagramme erstellt.

Im den verbesserten Modellen 2 und 3 wird die Hertz'sche Theorie zur Ermittlung einer mathematischen Formulierung für den Kontaktvorgang verwendet. Vorteile gegenüber dem ersten Modell ergeben sich durch die Möglichkeit der Berechnung der Kontaktkraft.

Das dritte Modell unterscheidet sich vom Zweiten durch einen elastisch gelagerten Stator. Hier wurden Bifurkationsdiagramme mit der Rotordrehzahl und der Fanglagermasse als Bifurkationsparameter durchgeführt.

Abschließend erfolgte der Vergleich der Ergebnisse mit der oben erwähnten Diplomarbeit.

Abstract

Calculation of Events within dynamic Rotor-Stator Sytems

This thesis deals with the simulation of three rotor-stator systems with rubs, especially the simulation of the event of the rub and serves as a comparison to the results of the diploma thesis "Numerical Investigation of Self-Exited Rotor-Stator Systems with Constraint by a Rigid or Elastic Retainer Bearing"by Sebatian Popprath from the year 2003, submitted at the technical university of vienna at the department of machine engineering.

It takes complex calculations for the analysis of these rotor-stator systems with rub, because it shows that due to the contact this systems become nonliniear. Three modells have been chosen for numerical testing with the software program Maple and the simulation software Maplesim.

The first model consists of a rotating part and a rigid stator with a circular clearance. The contact is modelled by means of the Newton impact model. For visulalization of the results bifurcation diagrams were aquired.

For the improved modells two and three the theory of contact between two bodies of H. Hertz is used to derive a mathematical expression for the contact between rotor and stator. Advantage is the possibility to calculate the contact force.

The third model differs from the second model by an elastically supported stator. Bifurcation diagrams using the number of rotations and the stator mass were used as bifurcation parameters.

Finally the results were compared with the diploma thesis mentioned above.

Inhaltsverzeichnis

Kapitel 1

Einleitung: Wichtige Grundbegriffe aus der Rotordynamik

Dieser Abschnitt dient einer Kurzerläuterung der wichtigsten Begriffe dieser Thematik. Kontaktvorgänge zwischen Rotor und Stator sind von großer technischer Bedeutung, da diese Vorgänge in praktisch allen Maschinen mit rotierenden Komponenten auftreten können. Durch Kontaktvorgänge entstehen schwerwiegende Auswirkungen auf das dynamische Verhalten des gesamten Systems, beziehungsweise führen zu nichtlinearen Systemgleichungen.

Rotor

In der Realität besteht der Rotor aus vielen Komponenten, bei der numerischen Untersuchung von Kontaktvorgängen wird aber meist das Minimalmodell der Rotordynamik verwendet: Dieses Minimalmodell besteht aus einer masselosen Welle und einer Rotorscheibe, welche als konzentrische Masse in der Mitte sitzt.

Stator

Beim Stator handelt es sich üblicherweise um den feststehenden Teil des Systems, zum Beispiel einem Gehäuse, einer Dichtung, einer Stopfbüchse oder um ein Fanglager.

Orbit

Die Darstellung der Bahn eines ausgezeichnenten Rotorpunktes, meistens der Wellenmittelpunkt, bezeichnet man als Orbit.

Kapitel 2

Modell 1: Simulationsmodell mit Newton'scher Stoßhypothese

2.1 Theoretische Grundlagen und Modellbildung

Grundlage für alle Simulationsmodelle der Dynamik sind die Bewegungsgleichungen eines betrachteten Systems. Die Bewegungsgleichungen werden hier nur angeführt, aber nicht hergeleitet. Interessierte Leser können dies in der Fachliteratur (z.B. [2], [7], [15], [19], [11]) nachlesen. Dieses Kapitel ist eine Kurzzusammenfassung der Theorie aus [13].
Beim Minimalmodell der Rotordynamik wird die Welle masselos, biegeelastisch und entweder isotrop oder anisotrop modelliert. Isotropie bezeichnet die Unabhängigkeit einer Eigenschaft von der Richtung. Zusätzlich kann zur äußeren Dämpfung auch ein innerer Dämpfungsmechanismus berücksichtigt werden. Auch die Einbringung einer Vorkrümmung im unbelasteten Zustand ist möglich.
Der Rotor wird als starre Scheibe mit statischer und/oder dynamischer Unwucht modelliert und besitzt einen axialen Freiheitsgrad, einen Torsionsfreiheitsgrad und 4 Biegefreiheitsgrade. Zusätzlich wird noch die Lagerung der Welle modelliert.
Bei diesem Modell trifft man die Annahme, dass die Rotordrehzahl konstant gehalten wird. Der Rotor wird mit zwei lateralen Freiheitsgraden, den Auslenkungen in $x-$ und $y-$Richtung modelliert. Die Welle soll einen kreisrunden Querschnitt und isotropes Materialverhalten aufweisen. Die Masse der Welle wird vernachlässigt. Die Wellenachse fällt mit der $z-$Achse des gewählten Koordinatensystems zusammen. Das Lager wird als isotrop, starr und nicht mit dem Fundament wechselwirkend angenommen.
Die Gesamtsteifigkeit je nach Koordinatenrichtung $k_{W,x}$, $k_{W,y}$ setzt sich aus der Summe der Biegesteifigkeit der Welle am Ort der Scheibe k_{Welle} und den Lagersteifigkeiten k_{Lager} zusammen. Dadurch erhält man die Beziehnung:

$$k_{W,i} = \frac{1}{\frac{1}{2*k_{Lager,i}} + \frac{1}{k_{Welle}}} = \frac{\frac{48*E*I}{L^3}}{1 + \frac{24*E*I}{k_{Lager,i}*L^3}} \qquad i = x, y \tag{2.1}$$

Dabei bezeichnen E den Elastizitätsmodul des Wellenwerkstoffs, I das axiale Flächenträgheitsmoment des Wellenquerschnitts und L die Länge der Welle.

Mit all diesen Annahmen erhält man folgende Bewegungsgleichungen:

$$m_W \ddot{x}_W + c_{W,x}\dot{x}_W + k_{W,x}x_W = m_W r\omega^2 \cos(\omega t) \tag{2.2}$$

$$m_W \ddot{y}_W + c_{W,y}\dot{y}_W + k_{W,y}y_W = m_W r\omega^2 \sin(\omega t) \tag{2.3}$$

m_W bezeichnet die Rotormasse, x_W und y_W bezeichnen die Rotorauslenkungen in $x-$ bzw. $y-$Richtung, $c_{W,x}$ und $c_{W,y}$ die Dämpfung in $x-$ bzw. $y-$Richtung, ω die Winkelgeschwindigkeit des Rotors, der zugleich die Kreisfrequenz der Unwuchterregung darstellt, und t die Zeit. Die Definition eines Dämpfungsmaßes (Lehr'sche Dämpfung) D_W und der Einführung der ungedämpften Eigenkreisfrequenz Ω_0 in beide Hauptrichtungen

$$D_{W,x} = \frac{c_{W,x}}{2\sqrt{k_{W,x} * m_W}} \qquad D_{W,y} = \frac{c_{W,y}}{2\sqrt{k_{W,y} * m_W}} \tag{2.4}$$

$$\Omega_{0,W,x} = \sqrt{\frac{k_{W,x}}{m_W}} \qquad \Omega_{0,W,y} = \sqrt{\frac{k_{W,y}}{m_W}} \tag{2.5}$$

lassen eine Umformung der Bewegungsgleichungen auf folgende Gestalt zu.

$$\ddot{x}_W + 2D_{W,x}\Omega_{0,W,x}\dot{x}_W + \Omega_{0,W,x}^2 x_W = r\omega^2 \cos(\omega t) \tag{2.6}$$

$$\ddot{y}_W + 2D_{W,y}\Omega_{0,W,y}\dot{y}_W + \Omega_{0,W,y}^2 y_W = r\omega^2 \sin(\omega t) \tag{2.7}$$

Dazu definiert man noch das Frequenzverhältnis η, welches auch als bezogene Erregerfrequenz oder dimensionslose Drehzahl bezeichnet wird.

$$\eta = \frac{\omega}{\Omega_{0,W,y}} \tag{2.8}$$

Zwei zusätzliche Normierungen machen die Durchführung von Simulationsstudien praktikabel: Die erste Normierung betrifft die Zeitachse. Ziel ist es, dass eine Umdrehung des Rotors genau einer Zeiteinheit entspricht. Damit eine Umdrehung genau einer Zeiteinheit der dimensionslosen Zeit τ entspricht, muss die folgende Beziehung erfüllt sein:

$$\omega t = 2\pi\tau \tag{2.9}$$

Aus diesem Ansatz müssen nun weitere, für die einzelnen Ableitungen der Freiheitsgrade gültige Beziehungen abgeleitet werden. Diese Form der Zeitnormierung wird bei allen drei Modellen angewandt.

$$t = \frac{2\pi}{\omega}\tau \tag{2.10}$$

$$dt = \frac{2\pi}{\omega}d\tau \tag{2.11}$$

$$\dot{x}_W = \frac{dx_W}{dt} = \frac{\omega}{2\pi}\frac{dx_W}{d\tau} = \frac{\omega}{2\pi}x'_W \tag{2.12}$$

$$\dot{y}_W = \frac{dy_W}{dt} = \frac{\omega}{2\pi}\frac{dy_W}{d\tau} = \frac{\omega}{2\pi}y'_W \tag{2.13}$$

$$\ddot{x}_W = \frac{d^2x_W}{dt^2} = (\frac{\omega}{2\pi})^2\frac{d^2x_W}{d\tau^2} = (\frac{\omega}{2\pi})^2 x''_W \tag{2.14}$$

$$\ddot{y}_W = \frac{d^2y_W}{dt^2} = (\frac{\omega}{2\pi})^2\frac{d^2y_W}{d\tau^2} = (\frac{\omega}{2\pi})^2 y''_W \tag{2.15}$$

Durch Einsetzen dieser Gleichungen in die umgeformten Bewegungsgleichungen erhält man die folgenden Beziehungen.

$$x''_W + 2D_{W,x}(2\pi\frac{\Omega_{0,W,x}}{\omega})x'_W + (2\pi\frac{\Omega_{0,W,x}}{\omega})^2 x_W = (2\pi)^2 r\cos(2\pi\tau) \tag{2.16}$$

$$y''_W + 2D_{W,y}(2\pi\frac{\Omega_{0,W,y}}{\omega})y'_W + (2\pi\frac{\Omega_{0,W,y}}{\omega})^2 y_W = (2\pi)^2 r\sin(2\pi\tau) \tag{2.17}$$

Die zweite Normierung betrifft Auslenkungen, Geschwindigkeiten und Beschleunigungen. All diese Größen werden auf das Lagerspiel c bezogen und damit dimensionslos gemacht.

$$u_W = \frac{1}{c}x_W \quad u'_W = \frac{1}{c}x'_W \quad u''_W = \frac{1}{c}x''_W \quad v_W = \frac{1}{c}y_W \quad v'_W = \frac{1}{c}y'_W \quad v''_W = \frac{1}{c}y''_W \tag{2.18}$$

Für den dimensionslosen Unwuchtradius e gilt:

$$e = \frac{r}{c} \tag{2.19}$$

Somit lauten die Bewegungsgleichungen:

$$u''_W = (2\pi)^2 e\cos(2\pi\tau) - 2D_{W,x}(2\pi\frac{\Omega_{0,W,x}}{\omega})u'_W - (2\pi\frac{\Omega_{0,W,x}}{\omega})^2 u_W \tag{2.20}$$

$$v''_W = (2\pi)^2 e\sin(2\pi\tau) - 2D_{W,y}(2\pi\frac{\Omega_{0,W,y}}{\omega})v'_W - (2\pi\frac{\Omega_{0,W,y}}{\omega})^2 v_W \tag{2.21}$$

Durch Rückeinsetzen erhält man die Bewegungsgleichungen, wie sie in Maple eingegeben wurden.

$$u''_W = (2\pi)^2 e\cos(2\pi\tau) - 2\frac{c_{W,x}}{2\sqrt{k_{W,x} * m_W}}(2\pi\frac{\sqrt{\frac{k_{W,x}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})u'_W - (2\pi\frac{\sqrt{\frac{k_{W,x}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2 u_W \tag{2.22}$$

$$v''_W = (2\pi)^2 e\sin(2\pi\tau) - 2\frac{c_{W,y}}{2\sqrt{k_{W,y} * m_W}}(2\pi\frac{\sqrt{\frac{k_{W,y}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})v'_W - (2\pi\frac{\sqrt{\frac{k_{W,y}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2 v_W \tag{2.23}$$

Nun muss noch der Kontaktvorgang formelmäßig ausgearbeitet werden, um ihn in das Simulationsmodell implementieren zu können. Für die theoretischen Grundlagen verweise ich auf die Fachliteratur, z.B. [12], [18]. Zu diesem Zweck wird eine weitere Abstraktion des realen Systems vorgenommen. Die Rotorscheibe wird auf den Wellendurchtrittspunkt zusammengezogen, woduch der Statordurchmesser den doppelten Wert des Lagerspiels annimmt. Daduch fällt der Kontaktpunkt des Rotors mit dem Stator mit dem Wellendurchtrittspunkt zusammen.
Nun werden die Freiheitsgrade x und y aus dem Rotorsystem in das Fanglagersystem transformiert. Diese zwei Koordinatensysteme kommen durch die Berücksichtigung eines Versatzes zwischen dem Fanglagermittelpunkt und dem Ursprung des Rotorkoordinatensystems zusatande. Da ein Versatz in der Praxis fast immer auftritt wurden zwei Offsetparameter dem Modell hinzugefügt.

$$\lambda_W = u_W - u_{OFF} \qquad \nu_W = v_W - v_{OFF} \tag{2.24}$$

Da es sich hierbei nur um translatorische, konstante Verschiebungen handelt, bleiben abgeleitete Größen (Geschwindigkeiten und Beschleunigungen) unverändert. Damit kann die dimensionslose radiale Auslenkung des Wellenmittelpunktes σ_W bezüglich des Ursprungs des Fanglagersystems ermittelt werden.

$$\sigma_W = \sqrt{\lambda_W^2 + \nu_W^2} \tag{2.25}$$

Da alle Größen auf das Lagerspiel bezogen sind und das Fanglager kreisförmig ist, nimmt die Kontaktbedingung mit Berücksichtigung der oben vorgenommenen Abstraktion folgende Form an.

$$\sigma_W - 1 > 0 \tag{2.26}$$

Für die eigentliche Berechnung des Kontaktvorganges benötigt man noch ein lokales Koordinatensystem, welches mit dem Normal- und dem Tangentialvektor an die kreisförmige Lagerkontur im Kontaktpunkt zusammenfällt. Definiert man den Winkel ψ als den Winkel zwischen der $\lambda-$Achse und der Strecke vom Fanglagermittelpunkt zum Kontaktpunkt (= Wellendurchtrittspunkt), so kann man die Rotationstransformationsmatrizen aufstellen.

$$\tan\psi = \frac{\nu_W}{\lambda_W} \qquad \sin\psi = \frac{\nu_W}{\sigma_W} \qquad \cos\psi = \frac{\lambda_W}{\sigma_W} \tag{2.27}$$

$$T_{iU} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \tag{2.28}$$

$$T_{gU} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \tag{2.29}$$

Die Geschwindigkeiten unmittelbar vor dem Kontakt in Normalenrichtung $v_{a,N}$ und tangentialer Richtung $v_{a,T}$ lassen sich nun ermitteln.

$$\begin{bmatrix} v_{a,N} \\ v_{a,T} \end{bmatrix} = T_{iU} \begin{bmatrix} u'_W \\ v'_W \end{bmatrix} = \begin{bmatrix} u'_W \cos\psi + v'_W \sin\psi \\ -u'_W \sin\psi + v'_W \cos\psi \end{bmatrix} \tag{2.30}$$

Damit werden über die Stoßgleichungen die Geschwindigkeiten unmittelbar nach dem Stoä berechnet. In diese Beziehungen gehen zusätzlich der Restitutionskoeffizient ε und der Coulomb'sche Reibbeiwert μ ein. Für die Herleitung der Beziehungen wird auf [5] verwiesen. Die Geschwindigkeiten unmittelbar nach dem Kontakt in Normalenrichtung $v_{e,N}$ und tangentialer Richtung $v_{e,T}$ lassen sich nun ermitteln.

$$v_{e,N} = -\varepsilon * v_{a,N} \tag{2.31}$$

$$v_{e,T} = v_{a,T} - \mu(1+\varepsilon)v_{a,N} \tag{2.32}$$

Durch Rücktransformation erhält man die Geschwindigkeiten für das Rotor/Stator System.

$$\begin{bmatrix} u'_W \\ v'_W \end{bmatrix} = T_{gU} \begin{bmatrix} v_{e,N} \\ v_{e,T} \end{bmatrix} = \begin{bmatrix} v_{e,N}\cos\psi - v_{e,T}\sin\psi \\ v_{e,N}\sin\psi + v_{e,T}\cos\psi \end{bmatrix} \tag{2.33}$$

Nun sind alle für die Simulation notwendigen Gleichungen formuliert und können in Maple implementiert werden.

2.2 Modellimplementierung und Simulationsergebnisse

Der folgende Abschnitt enthält den Programmiercode mit anschließender Erklärung.

2.2.1 Maple Code des 1. Modells

```
(1) restart
(2) with(plots):
(3) Digits:=15;

(4) RoSt1 := dsolve({ diff(x(t), [`$`(t, 2)]) = (2*Pi)^2*uwr*cos(2*Pi*t)-
2*daempfx/(2*sqrt(kx*mass))*(2*Pi*sqrt(kx/mass)/(rdz*sqrt(ky/mass)))*
(diff(x(t), t))- (2*Pi*sqrt(kx/mass)/(rdz*sqrt(ky/mass)))^2*x(t),
x(0) = 0, D(x)(0) = 0, diff(y(t), [`$`(t,2)])= (2*Pi)^2*uwr*sin(2*Pi*t)-
2*daempfy/(2*sqrt(ky*mass))*(2*Pi*sqrt(ky/mass)/(rdz*sqrt(ky/mass)))*
(diff(y(t), t))-(2*Pi*sqrt(ky/mass)/(rdz*sqrt(ky/mass)))^2*y(t),
y(0) = 0,D(y)(0) = 0}, numeric, parameters = [uwr, rdz, daempfx, daempfy,
kx, ky, mass, xoff, yoff, rkf, rbw], events = [[[y(t), t >= 270], halt],
[sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)-1, [diff(x(t), t) = -cos(arctan(y(t)-yoff,
x(t)-xoff))*(cos(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(x(t), t))+
sin(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(y(t), t)))*rkf-
sin(arctan(y(t)-yoff, x(t)-xoff))* (-sin(arctan(y(t)-yoff, x(t)-xoff))*
pre(diff(x(t), t))+cos(arctan(y(t)-yoff, x(t)-xoff))*
pre(diff(y(t), t))-(1+rkf)*rbw*(cos(arctan(y(t)-yoff, x(t)-xoff))*
pre(diff(x(t), t))+sin(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(y(t), t)))),
diff(y(t), t) = -sin(arctan(y(t)-yoff, x(t)-xoff))*(cos(arctan(y(t)-yoff,
x(t)-xoff))*pre(diff(x(t), t))+sin(arctan(y(t)-yoff, x(t)-xoff))*
pre(diff(y(t), t)))*rkf+ cos(arctan(y(t)-yoff, x(t)-xoff))*
(-sin(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(x(t), t))+
cos(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(y(t), t))-(1+rkf)*rbw*
(cos(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(x(t), t))+ sin(arctan(y(t)-yoff,
x(t)-xoff))*pre(diff(y(t), t)))]]], event_maxiter = 100000, maxfun = 0,
abserr = 10^(-200), relerr = 10^(-8), optimize = true);

(5) RoSt1(parameters = [0.5, 0.7650, 150, 150, 150000, 150000, 8.75, 0.2,
0.2, 1, 0.3]):
(6) a := odeplot(RoSt1, [x(t), y(t)], 240 .. 270, numpoints = 3000):
(7) b := implicitplot((x-0.2)^2+(y-0.2)^2 = 1, x = -0.8 .. 1.2, y = -0.8
.. 1.2, colour = blue):
(8) display(a,b);

(9) allRoots := proc (dsn, tf)
local result, val;
_Env_dsolve_nowarnstop := true;
result := Vector(0, 'datatype = float');
val := dsn(tf);
while op([1, 2], val) <tf do
```

```
result(numelems(result)+1) := op([1, 2], val);
dsn(':-eventclear');
val := dsn(tf)
end do;
result
end proc

(10) k:=1;
(11) for j from 0.7650 by 0.0001 to 0.8150 do
RoSt1(parameters=[0.5, j,150,150,150000,150000,8.75,0.2,0.2,1,0.3]):
zpkte:=allRoots(RoSt1,300):

RoSt1listop := dsolve({diff(x(t), [`$`(t, 2)]) = (2*Pi)^2*uwr*cos(2*Pi*t)-
2*daempfx/(2*sqrt(kx*mass))*(2*Pi*sqrt(kx/mass)/(rdz*sqrt(ky/mass)))*
(diff(x(t), t))- (2*Pi*sqrt(kx/mass)/(rdz*sqrt(ky/mass)))^2*x(t),
x(0) = 0, D(x)(0) = 0, diff(y(t), [`$`(t,2)])= (2*Pi)^2*uwr*sin(2*Pi*t)-
2*daempfy/(2*sqrt(ky*mass))*(2*Pi*sqrt(ky/mass)/(rdz*sqrt(ky/mass)))*
(diff(y(t), t))-(2*Pi*sqrt(ky/mass)/(rdz*sqrt(ky/mass)))^2*y(t), y(0)
= 0, D(y)(0) = 0}, numeric, parameters = [uwr, rdz, daempfx, daempfy, kx,
ky, mass, xoff, yoff, rkf, rbw], events = [ [sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)
-1, [diff(x(t), t) = -cos(arctan(y(t)-yoff, x(t)-xoff))*(cos(arctan(y(t)-yoff,
x(t)-xoff))*pre(diff(x(t), t))+sin(arctan(y(t)-yoff, x(t)-xoff))*
pre(diff(y(t), t)))*rkf-sin(arctan(y(t)-yoff, x(t)-xoff))*
(-sin(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(x(t), t))+cos(arctan(y(t)-yoff,
x(t)-xoff))* pre(diff(y(t), t))-(1+rkf)*rbw*(cos(arctan(y(t)-yoff, x(t)-xoff))
*pre(diff(x(t), t))+sin(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(y(t), t)))),
diff(y(t), t) = -sin(arctan(y(t)-yoff, x(t)-xoff))*(cos(arctan(y(t)-yoff,
x(t)-xoff))*pre(diff(x(t), t))+sin(arctan(y(t)-yoff, x(t)-xoff))*
pre(diff(y(t), t)))*rkf+ cos(arctan(y(t)-yoff, x(t)-xoff))*
(-sin(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(x(t), t))+cos(arctan(y(t)-yoff,
x(t)-xoff))*pre(diff(y(t), t))-(1+rkf)*rbw*(cos(arctan(y(t)-yoff, x(t)-xoff))*
pre(diff(x(t), t))+ sin(arctan(y(t)-yoff, x(t)-xoff))*pre(diff(y(t), t))))]]],
event_maxiter = 100000, maxfun = 0, abserr = 10^(-200), relerr = 10^(-8),
optimize = true, output = listprocedure);

RoSt1listop(parameters=[0.5,j,150,150,150000,150000,8.75,0.2,0.2,1,0.3]):

for i from 1 to numelems(zpkte) do
nst[i]:=rhs(RoSt1listop[2](zpkte[i])):
end do;
liste[k]:=[[j,nst[n]]$(n=1..numelems(zpkte))]:
c[k]:=pointplot(liste[k]):
k:=k+1;
end do;

(12) display(seq(c[l], l = 1 .. k-1));
```

Nach der Initialisierung (1) und dem Laden des Pakets für das Erstellen von Grafiken (2) wird die Anzahl der Stellen für die Rechengenauigkeit (3) festgelegt. Als nächster Schritt (4) folgt die Eingabe der Differenzialgleichung und der Anfangswerte mittels $dsolve\{...\}$. $numeric$ legt fest, daß eine numerische Lösung gesucht werden soll, in Maple ist die Standardberechnungsmethode das Runge-Kutta Fehlberg Verfahren 5. Ordnung. $parameters$ legt die Parameter der Differenzialgleichung fest. $events[...]$ steuert das Eventhandling: Das erste Event $y(t), t >= 270$ sucht nach allen Nullstellen nach dem Zeitpunkt t = 270 und hält dann die Berechnung wegen dem Befehl *halt* an. Das zweite Event $\sqrt{(x(t) - xoff)^2 + (y(t) - yoff)^2} - 1 = 0$ beschreibt jenen Ort wo sich Rotor und Stator berühren, und was dann beim Eintreten des Events geschehen soll (Abänderung der ersten Ableitungen). Der Parameter *pre* in den Bedingungen für die Änderung der Ableitungen sorgt dafür, dass der letzte Wert vor Eintreten des Events herangezogen wird. $event\_maxiter$ legt die maximale Anzahl der Berechnungen des Events fest, $maxfun = 0$ deaktiviert das Berechnugslimit für die numerische Lösung, damit die Berechnung bei sehr vielen Berechnungsschritten nicht vorzeitig abbrechen kann. $abserr$ und $relerr$ legen die absoluten und relativen Fehlertoleranzen für einen erfolgreichen Schritt des Anfangswertproblems fest.$optimize = true$ optimiert intern das Differenzialgleichungssystem für eine eventuell schnellere Berechnung. (5) legt die Werte der Parameter fest. (6) berechnet nun die Bewegung des Stators um sie für eine grafische Ausgabe vorzubereiten (in diesem Beispiel im Zeitbereich von 240 bis 270), $numpoints$ legt die Anzahl der zu berechnenden Punkte im Intervall fest, um eine glatte Kurve zu erhalten. (7) zeichnet den starren Stator, (8) zeichnet nun die Ergebnisse aus (6) und (7) in einer Grafik. Dies dient zur Visualisierung der Orbits.
Die nächsten Schritte dienen zur Erstellung eines Bifurkationsdiagramms in Maple:
Die Prozedur (9) erstellt, ohne Zwischenausgaben wie die Warung eines stattfindenden Events, zuerst einen leeren Vektor in Gleitkommadarstellung. dsn is die Variable der zu berechnenden Funktion, in diesem Fall das Differenzialgleichungssystem, tf is die Obergrenze für das Berechnugsintervall der Nullstellen des Stators. Danach wird der Zeitpunkt einer Nullstelle berechnet, das Ergebnis im Vektor gespeichert, das Event gelöscht, damit bei den weiteren Berechnungen nicht wieder bei den bereits gespeicherten Nullstellen angehalten wird. Anders ausgedrückt, die offene Untergrenze für das Berechnungsintervall der Nullstellen wird immer auf die letzte gefundene Nullstelle gesetzt. Das Ergebnis der Prozedur (9) ist also ein Vektor mit allen Zeitpunkten von Nulldurchgängen des Stators im Intervall $[270, tf]$.
Die Schleife (11) iteriert den gewünschten Parameter, hier die Rotordrehzahl rdz im gewünschten Bereich mit gewählter Schrittweite. Die Zeitpunkte der Nullstellen werden in das Differenzialgleichungssystem eingesetzt um in der nächsten Schleife die x-Koordinaten der Nullstellen zu berechnen. Es wird nun für jeden Wert des Parametes eine Liste an Punkten erstellt, deren x-Koordinate den Wert des Parameters hat, und deren y-Koordinate den Wert der Nullstelle annimmt. Diese Punkte werden in einer Grafik gespeichert. (12) fasst all diese Grafiken zu einer Grafik zusammen, dem Bifurkationsdiagramm.

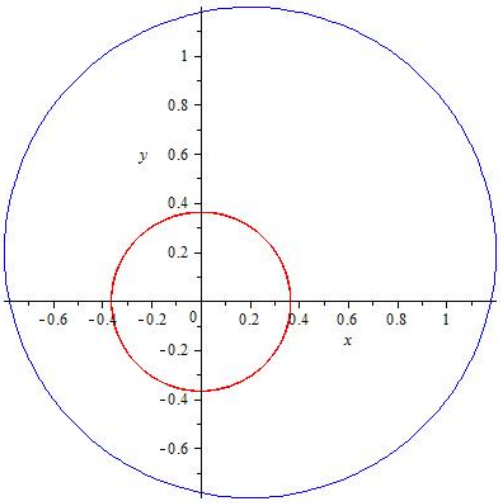
2.2.2 Simulationsergebnisse

Zunächst werden die Parameter für das Simulationsmodell festgelegt. Dabei erfolgt eine Aufteilung in dimensionsbehaftete und dimensionslose Parameter. Als Simulationsdauer wurden 300 Zeiteinheiten, das entspricht 300 Rotorumdrehungen, gewählt. In diesem Abschnitt wurden Simulationen mit verschieden großer Unwucht durchgeführt. Aufgrund der gewählten Steifigkeitsisotropie ($k_{W,x} = k_{W,y}$) stellt sich im kontaktfreien Bereich ein kreisförmiger Orbit ein. Der gewählte Bereich der Unwucht beziehungsweise die gewählten Parameter sind in folgender Tabelle aufgelistet.

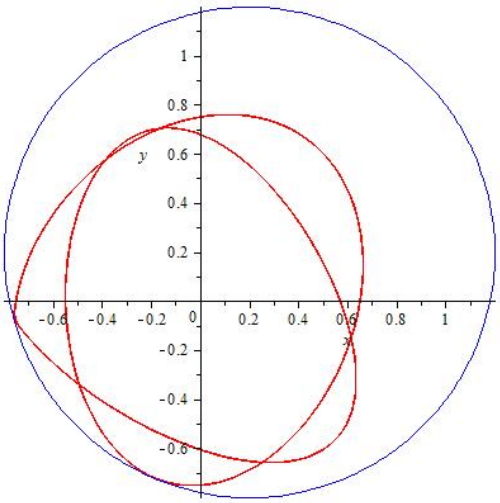
Bezeichnung	**Parameter**	**Wert**	**Einheit**
Rotormasse	m	8.75	kg
Steifigkeit in x-Richtung	$k_{W,x}$	150000	N/m
Steifigkeit in y-Richtung	$k_{W,y}$	150000	N/m
Dämpfung in x-Richtung	$c_{W,x}$	150	Ns/m
Dämpfung in y-Richtung	$c_{W,y}$	150	Ns/m
Offset in x-Richtung	u_{OFF}	0.2	
Offset in y-Richtung	v_{OFF}	0.2	
Restitutionskoeffizient	ε	1.0	
Coulomb'scher Reibbeiwert	μ	0.0	
Dimensionslose Drehzahl	η	0.775	
Bezogener Unwuchtradius	e	0.25 - 1.50	

Tabelle 2.1: Simulationsparameter für die Untersuchung des Unwuchteinflusses

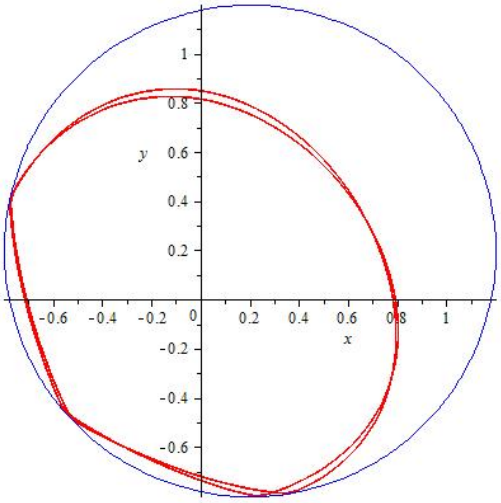
Ausgewählte Simulationsergebnisse sind in Abbildung 2.1 in Form von Orbits des Wellendurchtrittspunktes dargestellt. Aus den dargestellten Orbits erkennt man, dass mit zunehmender Unwucht immer mehr Kontakte pro Umdrehung des Rotors auftreten. Eine weitere Steigerung der Unwucht führt zum sogenannten *Full Annular Rub*: Dabei bewegt sich der Rotor entgegen seiner Drehrichtung entlang des Lagerspiels. Dieser Zustand zeigt die numerischen Grenzen des Simulationsprogramms auf. Da der zeitliche Abstand zwischen zwei aufeinanderfolgenden Kontakten immer kleiner wird, wird irgendwann die minimal mögliche Integrationsschrittweite zwischen zwei Kontakten unterschritten und, da vorher bereits das nächste *event* stattfindet welches nun nicht mehr berechnet wird, platziert das Programm den nächsten Kontaktpunkt hinter dem Stator. Ab diesem Simulationszeitpunkt sind alle weiteren Ergebnisse natürlich unbrauchbar.



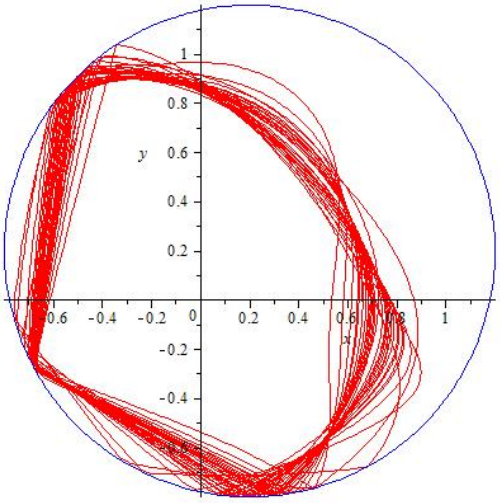
(a) Unwuchtradius 0.25



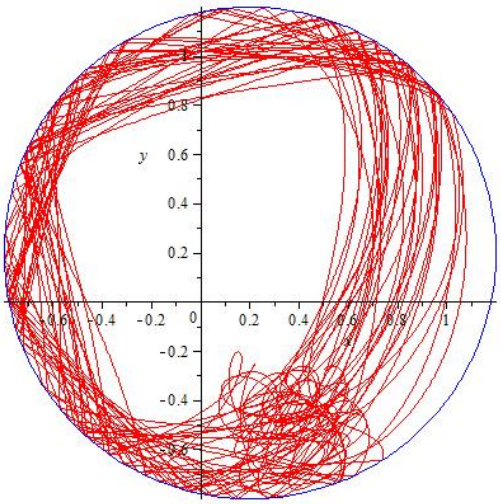
(b) Unwuchtradius 0.50



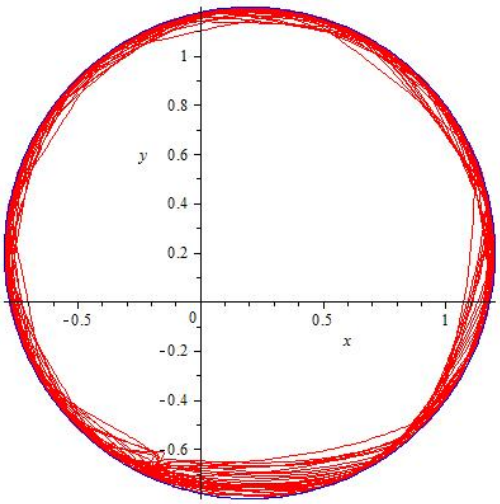
(c) Unwuchtradius 0.75



(d) Unwuchtradius 1.00



(e) Unwuchtradius 1.25



(f) Unwuchtradius 1.50

Abbildung 2.1: Rotor-Orbits für Rotordrehzahl 0.775 bei variierter Unwucht

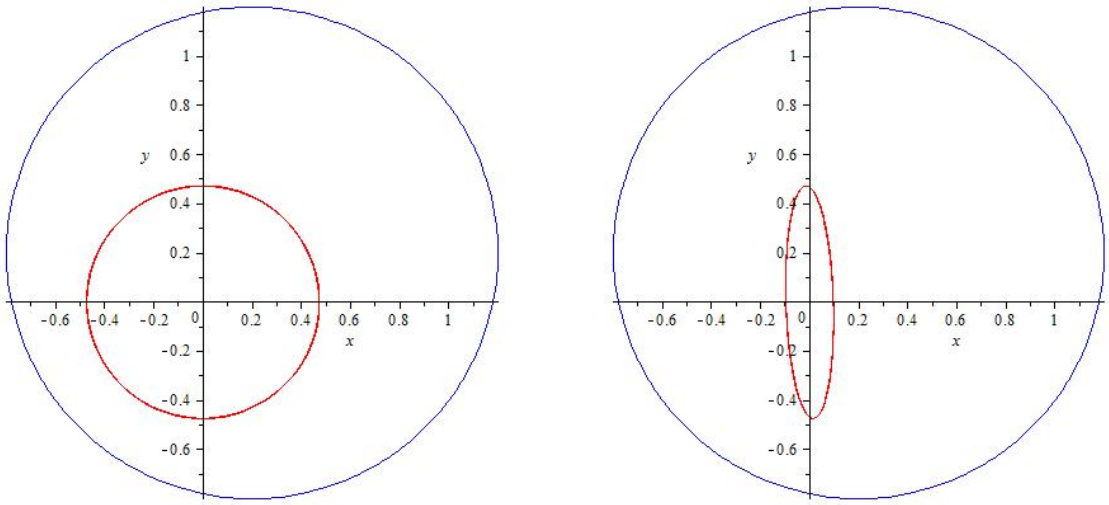
2.2.3 Anisotrophie der Lagerung

In diesem Abschnitt wurde die Steifigkeit anisotroph ($k_{W,x} \neq k_{W,y}$) gewählt. Im kontaktfreien Bereich stellt sich nun ein ellipsenförmiger Orbit ein. Die gewählten Parameter, beziehungsweise deren Variationsbereiche, sind in folgender Tabelle aufgelistet.

Bezeichnung	**Parameter**	**Wert**	**Einheit**
Rotormasse	m	8.75	kg
Steifigkeit in x-Richtung	$k_{W,x}$	150000 - 450000	N/m
Steifigkeit in y-Richtung	$k_{W,y}$	150000	N/m
Dämpfung in x-Richtung	$c_{W,x}$	150	Ns/m
Dämpfung in y-Richtung	$c_{W,y}$	150	Ns/m
Offset in x-Richtung	u_{OFF}	0.2	
Offset in y-Richtung	v_{OFF}	0.2	
Restitutionskoeffizient	ε	1.0	
Coulomb'scher Reibbeiwert	μ	0.0	
Dimensionslose Drehzahl	η	0.700 - 0.800	
Bezogener Unwuchtradius	e	0.50	

Tabelle 2.2: Simulationsparameter für die Untersuchung des Einflusses von anisotropen Lagern

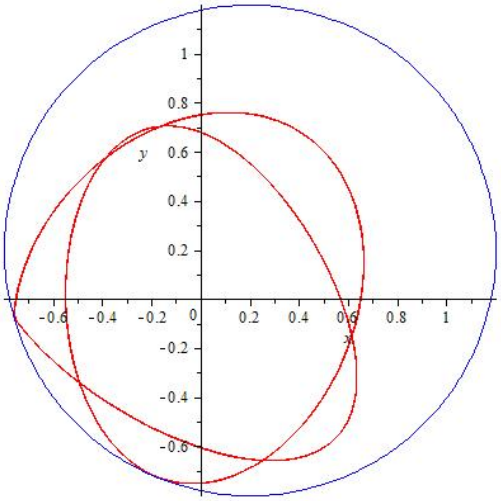
Ausgewählte Simulationsergebnisse sind in den folgenden Abbildungen 2.2 - 2.4 dargestellt.



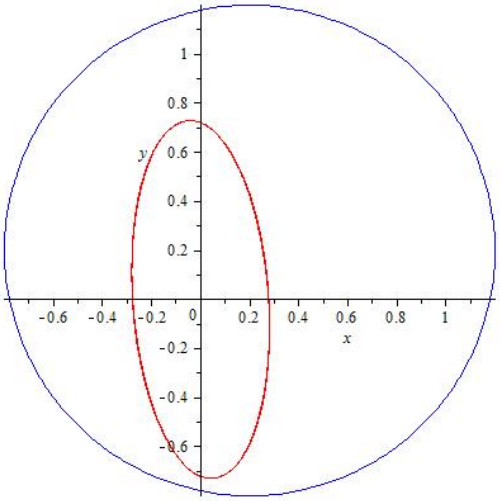
(a) Steifigkeit in x-Richtung 150000

(b) Steifigkeit in x-Richtung 450000

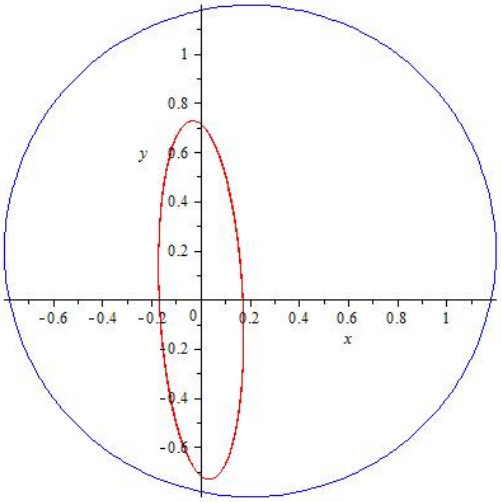
Abbildung 2.2: Rotor-Orbits für Rotordrehzahl 0.700 bei variierter Gesamtsteifigkeit in x-Richtung



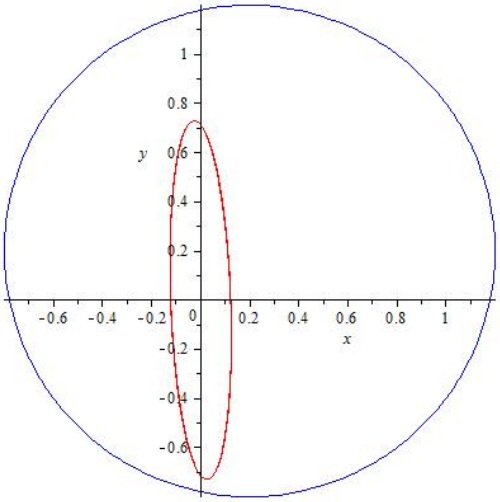
(a) Steifigkeit in x-Richtung 150000



(b) Steifigkeit in x-Richtung 250000

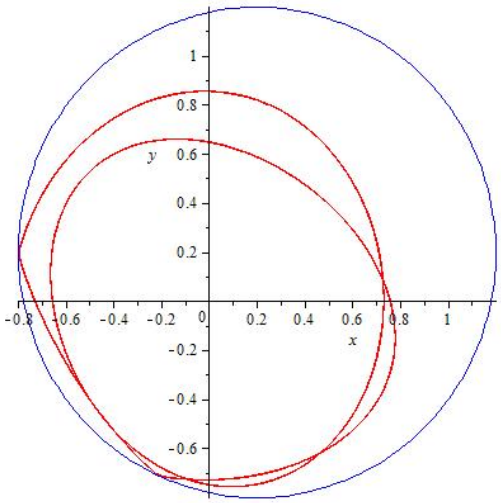


(c) Steifigkeit in x-Richtung 350000

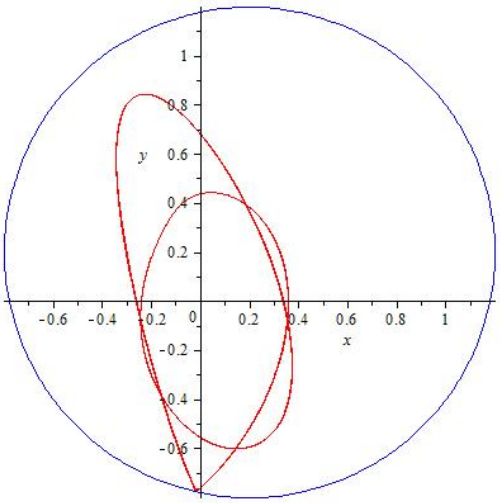


(d) Steifigkeit in x-Richtung 450000

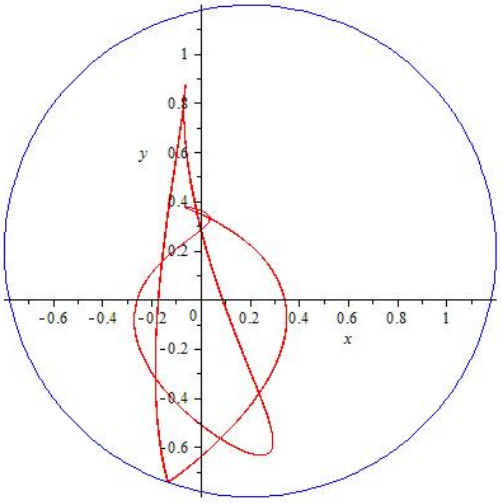
Abbildung 2.3: Rotor-Orbits für Rotordrehzahl 0.775 bei variierter Gesamtsteifigkeit in x-Richtung



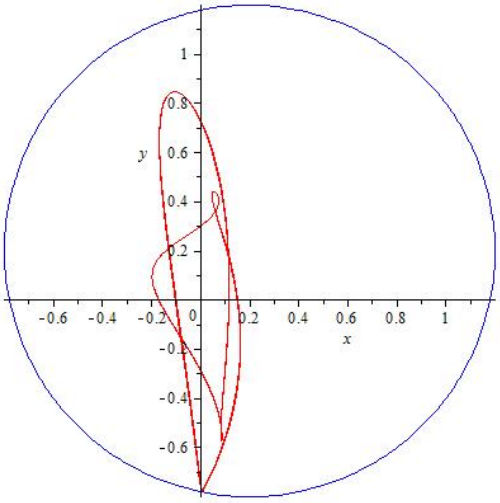
(a) Steifigkeit in x-Richtung 150000



(b) Steifigkeit in x-Richtung 250000



(c) Steifigkeit in x-Richtung 350000



(d) Steifigkeit in x-Richtung 450000

Abbildung 2.4: Rotor-Orbits für Rotordrehzahl 0.800 bei variierter Gesamtsteifigkeit in x-Richtung

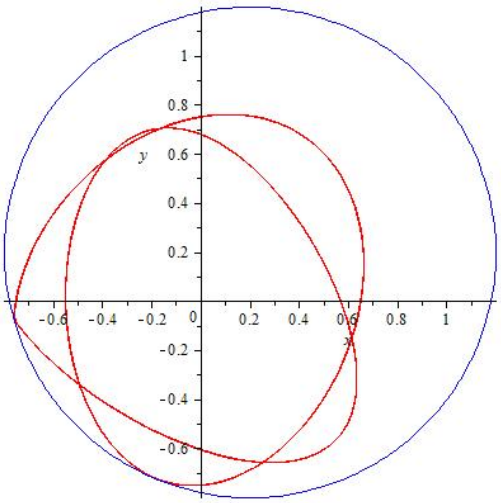
2.2.4 Verzweigungsverhalten

Die Untersuchung des Verzweigungsverhaltens stellt den Hauptpunkt der Analyse dieses Simulationsmodelss dar. Die gewählten Parameter, beziehungsweise deren Variationsbereiche, sind in folgender Tabelle aufgelistet.

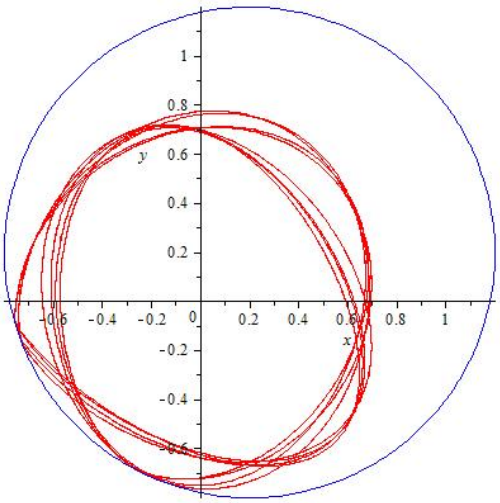
Bezeichnung	**Parameter**	**Wert**	**Einheit**
Rotormasse	m	8.75	kg
Steifigkeit in x-Richtung	$k_{W,x}$	150000	N/m
Steifigkeit in y-Richtung	$k_{W,y}$	150000	N/m
Dämpfung in x-Richtung	$c_{W,x}$	150	Ns/m
Dämpfung in y-Richtung	$c_{W,y}$	150	Ns/m
Offset in x-Richtung	u_{OFF}	0.2	
Offset in y-Richtung	v_{OFF}	0.2	
Restitutionskoeffizient	ε	1.0	
Coulomb'scher Reibbeiwert	μ	0.0 - 0.3	
Dimensionslose Drehzahl	η	0.765 - 0.815	
Bezogener Unwuchtradius	e	0.50	

Tabelle 2.3: Simulationsparameter für die Untersuchung des Verzweigungsverhaltens

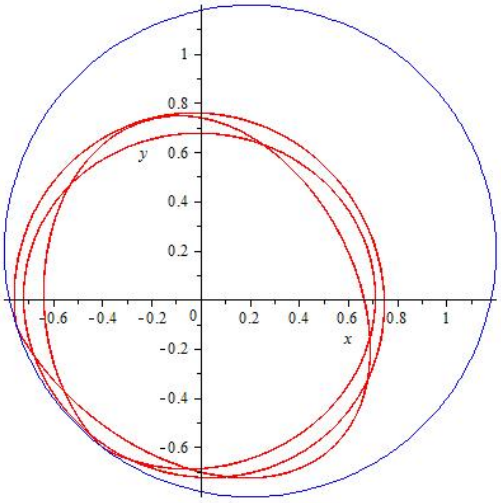
Die Abbildungen 2.5 und 2.8 zeigen wieder ausgewählte Orbitdiagramme, hier bei unterschiedlichem Reibbeiwert bei variierter Rotordrehzahl. Zum beseren Verständnis werden die in Abbildung 2.6 und 2.7 dargestellten Verzweigungsdiagramme nun kurz erklärt.
Bei der Simulation wurde der Bereich zwischen 270 und 300 Zeiteinheiten hergenommen und alle Durchgänge des Rotororbits durch die $x-$Achse gespeichert. Diese Nullstellen wurden dann als $y-$Koordinaten im Diagramm aufgetragen. Die $x-$Koordinaten der Punkte geben den variierten Parameter, in diesem Fall die Rotordrehzahl, an. Dadurch eliminiert man den Einschwingvorgang und kann auch hochperiodische Orbits erkennen.



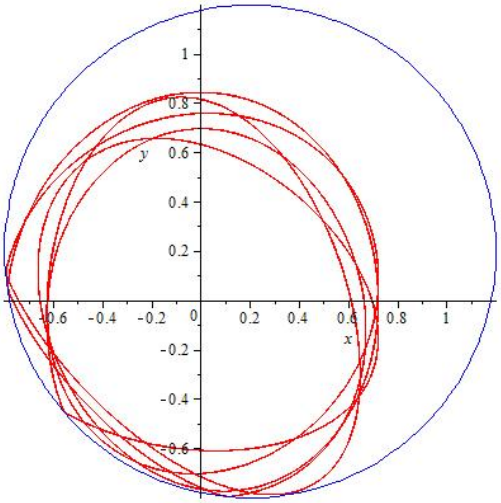
(a) Rotordrehzahl 0.7750



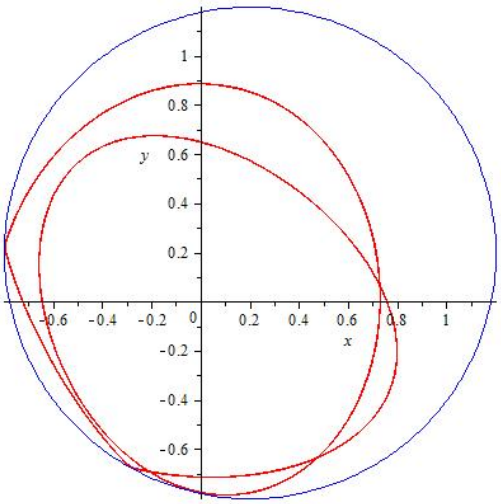
(b) Rotordrehzahl 0.7770



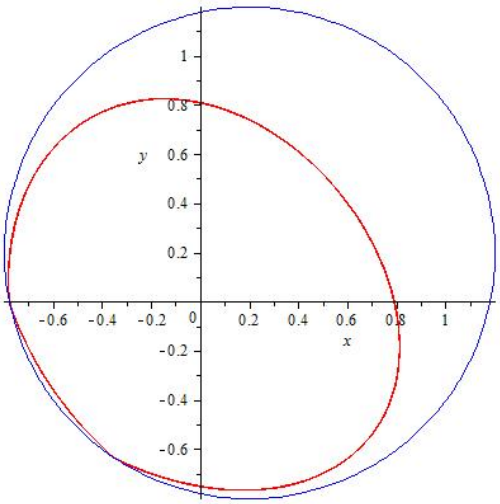
(c) Rotordrehzahl 0.7790



(d) Rotordrehzahl 0.7882



(e) Rotordrehzahl 0.8040



(f) Rotordrehzahl 0.8100

Abbildung 2.5: Orbits für ausgewählte Drehzahlen. Restitutionskoeffizient 1, Reibbeiwert 0

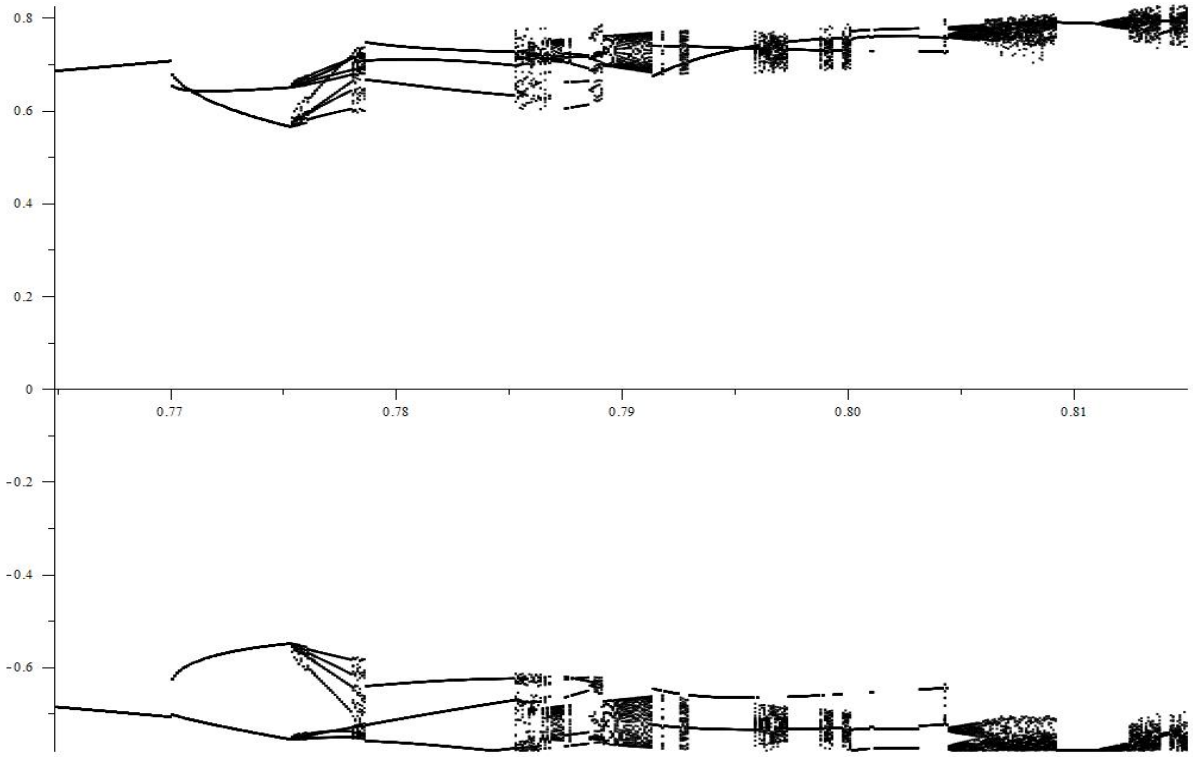


Abbildung 2.6: Restitutionskoeffizient = 1 und Reibbeiwert = 0

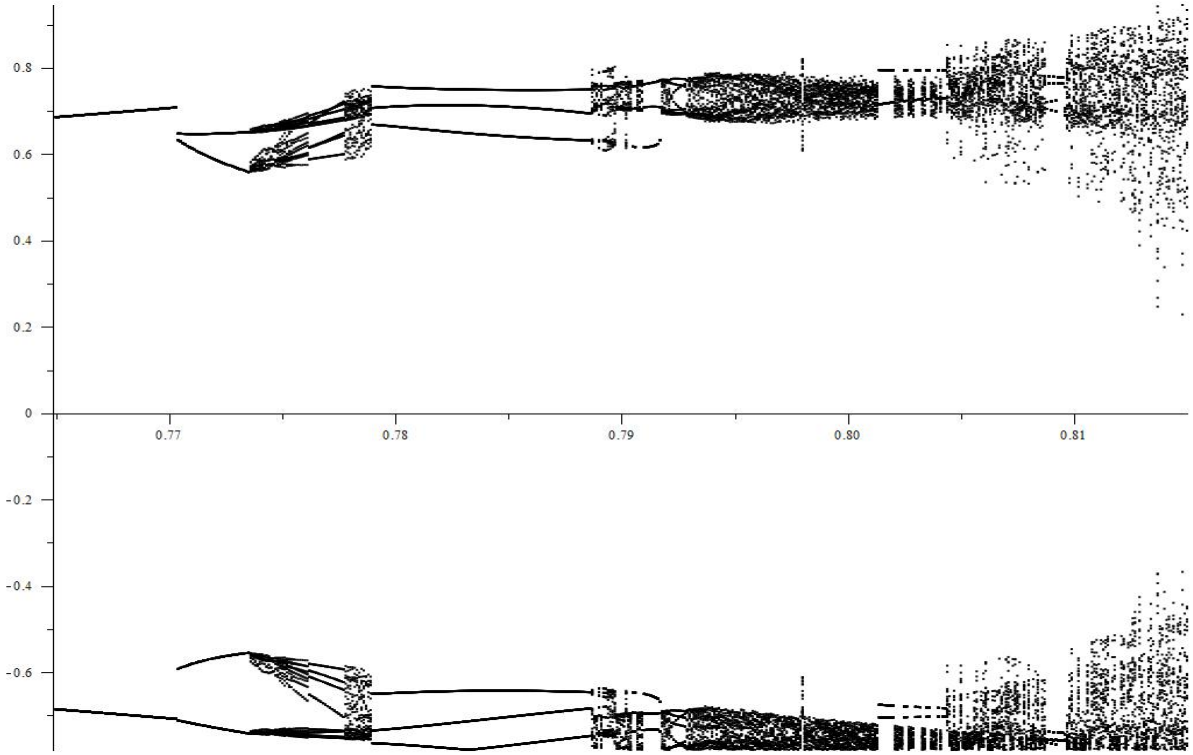
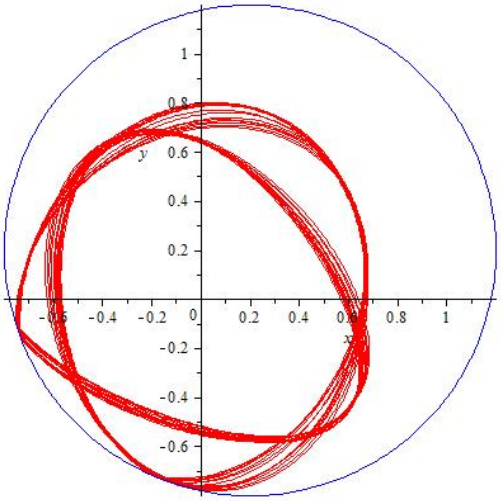
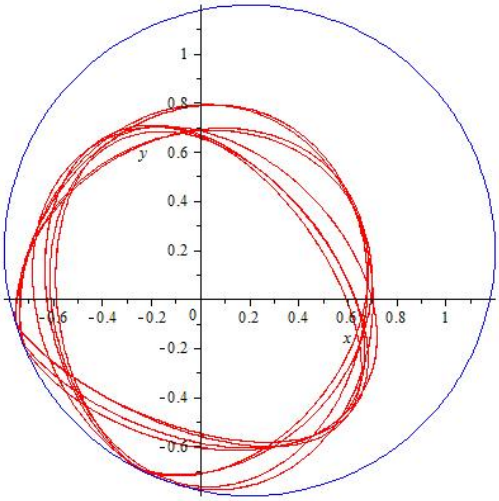


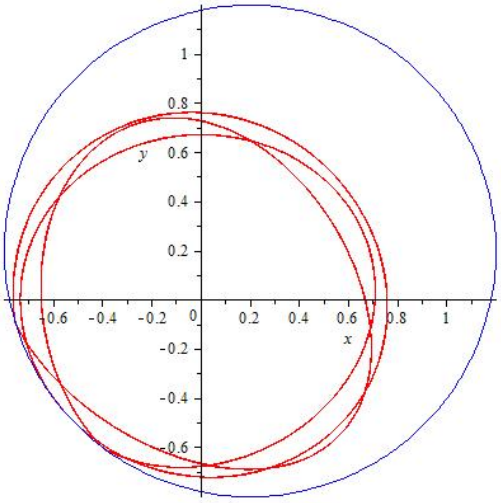
Abbildung 2.7: Restitutionskoeffizient = 1 und Reibbeiwert = 0.3



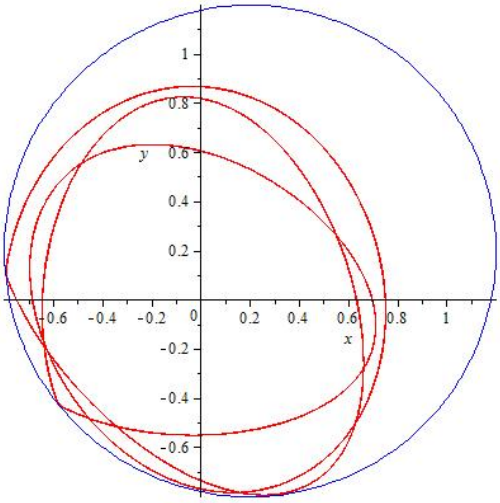
(a) Rotordrehzahl 0.7750



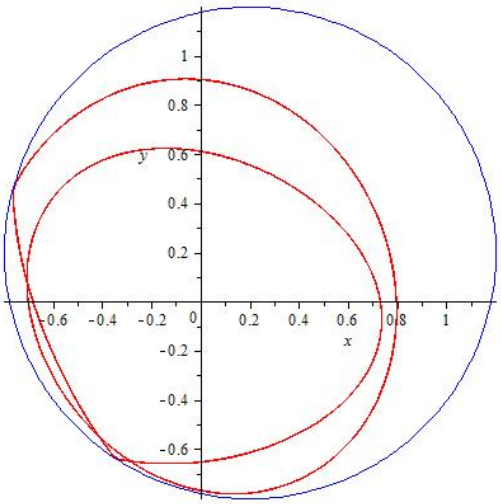
(b) Rotordrehzahl 0.7770



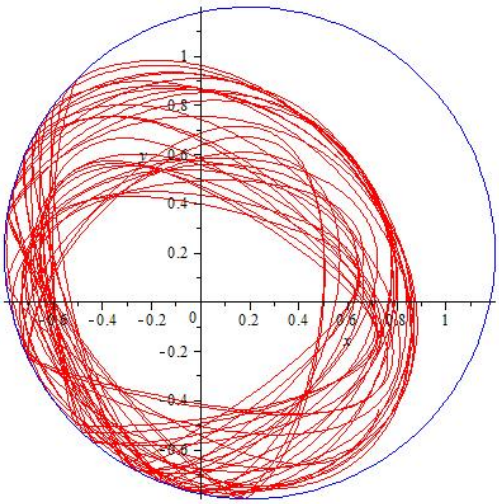
(c) Rotordrehzahl 0.7790



(d) Rotordrehzahl 0.7882



(e) Rotordrehzahl 0.8040



(f) Rotordrehzahl 0.8100

Abbildung 2.8: Orbits für ausgewählte Drehzahlen. Restitutionskoeffizient 1, Reibbeiwert 0.3

Kapitel 3

Modell 2: Simulationsmodell mit kontaktmechanischer Hypothese

3.1 Theoretische Grundlagen und Modellbildung

Dieses Kapitel ist eine Kurzzusammenfassung der Theorie aus [13]. Wie beim Modell mit Newton'scher Stoßhypothese wird auch hier wieder ein vertikaler Rotor in einem kreisrunden Fanglager bei konstanter Drehzahl betrachtet. Hier kommt jedoch ein verbesertes Kontaktmodell zum Einsatz. Bei Berührung von Rotor und Stator tritt eine Normalkraft und wegen der Relativbewegung in tangentialer Richtung auch eine Tangentialkraft auf. Die Tangentialkraft $F_{K,T}$ ist über den Coulomb'schen Gleitreibungskoeffizienten μ mit der Normalkraft verknüpft.

$$F_{K,T} = \mu(k_h d + c_h \dot{d}) \tag{3.1}$$

d bezeichnet die Eindringtiefe und $\dot{d}$ die Eindringgeschwindigkeit des Rotors in Normalenrichtung in den Stator. k_h und c_h stehen für die Kontaktsteifigkeit bzw. für die Kontaktdämpfung. Der Offset bzw. die Kontaktbedingung sind gleich wie beim ersten Modell (2.24), (2.25) und (2.26). Auch die Gleichungen für den Winkel ψ und die Transformationsmatrizen haben sich nicht geändert (2.27) - (2.29).
Daduch lassen sich die Geschwindigkeiten im lokalen Koordinatensystem berechnen.

$$\dot{d} = \dot{x}_W \cos\psi + \dot{y}_W \sin\psi \tag{3.2}$$

$$v_t = -\dot{x}_W \sin\psi + \dot{y}_W \cos\psi + \frac{D_R}{2}\omega \tag{3.3}$$

v_t steht für die Geschwindigkeit in Tangentialrichtung, der Term $\frac{D_R}{2}\omega$ repräsentiert dabei die zusätzliche Relativgeschwindgkeit zufolge der Drehung des Rotors.
Da die Richtung der Kontaktkraft in tangentialer Richtung von der Relativbewegung zwischen Rotor und Stator im Kontaktpunkt abhängt, muss eine Schaltfunktion δ_f eingeführt werden, die garantiert, dass die Kontaktkraft in tangentialer Richtung immer entgegen der Relativbewegung wirkt.

$$\delta_f = \begin{cases} 1 & \text{wenn } v_t > 0 \\ -1 & \text{wenn } v_t < 0 \\ 0 & \text{wenn kein Kontakt} \end{cases} \tag{3.4}$$

Aus der Kontaktnormal- und Kontakttangentialkraft werden nun die Anteile in beide Koordinatenrichtungen ermittelt.

$$F_{K,x} = (k_h d + c_h \dot{d})(-\cos\psi + \delta_f \mu \sin\psi) \tag{3.5}$$

$$F_{K,y} = (k_h d + c_h \dot{d})(-\sin\psi - \delta_f \mu \cos\psi) \tag{3.6}$$

Somit erhält man die analog zu (2.2) und (2.3) die Bewegungsgleichungen des betrachteten Systems in dimensionsbehafteter Form.

$$m_W \ddot{x}_W + c_{W,x}\dot{x}_W + k_{W,x}x_W = m_W r\omega^2 \cos(\omega t) + (k_h d + c_h \dot{d})(-\cos\psi + \delta_f \mu \sin\psi) \tag{3.7}$$

$$m_W \ddot{y}_W + c_{W,y}\dot{y}_W + k_{W,y}y_W = m_W r\omega^2 \sin(\omega t) + (k_h d + c_h \dot{d})(-\sin\psi - \delta_f \mu \cos\psi) \tag{3.8}$$

Nun werden analog zum ersten Modell wieder die Zeit- und Längennormierungen durchgeführt (2.9) bis (2.15) und (2.18),(2.19). Die Längennormierungen für d und $\dot{d}$ lauten

$$\delta = \frac{d}{c} \qquad \delta' = \frac{\dot{d}}{c} \tag{3.9}$$

Somit erhält man die Bewegungsgleichungen in $x-$ und $y-$Richtung für das zweite Modell, wie sie in Maple eingegeben wurden.

$$\begin{aligned} u''_W = {} & (2\pi)^2 e\cos(2\pi\tau) - 2\frac{c_{W,x}}{2\sqrt{k_{W,x} * m_W}}(2\pi\frac{\sqrt{\frac{k_{W,x}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})u'_W - (2\pi\frac{\sqrt{\frac{k_{W,x}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2 u_W + \\ & + ((2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2\delta + 2\frac{c_h}{2\sqrt{k_h m}}2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}}(u'_W\cos\psi + v'_W\sin\psi))(-\cos\psi + \delta_f\mu\sin\psi) \end{aligned} \tag{3.10}$$

$$\begin{aligned} v''_W = {} & (2\pi)^2 e\sin(2\pi\tau) - 2\frac{c_{W,y}}{2\sqrt{k_{W,y} * m_W}}(2\pi\frac{\sqrt{\frac{k_{W,y}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})v'_W - (2\pi\frac{\sqrt{\frac{k_{W,y}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2 v_W + \\ & + ((2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2\delta + 2\frac{c_h}{2\sqrt{k_h m}}2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}}(u'_W\cos\psi + v'_W\sin\psi))(-\sin\psi - \delta_f\mu\cos\psi) \end{aligned} \tag{3.11}$$

3.2 Modellimplementierung und Simulationsergebnisse

Der folgende Abschnitt enthält den Programmiercode mit anschließender Erklärung.

3.2.1 Maple Code des 2. Modells mit Gear Verfahren

```
(1) restart
(2) with(plots):
(3) Digits:=15;

(4) uwr := 0.5; rdz := 0.765; daempfx := 150; daempfy := 150; kontdaempf
:= 0; kx := 150000; ky := 15000; mass := 8.75; xoff := 0.2; yoff := 0.2;
kontsteif := 1500000000; rbw := 0.3;

(5) RoSt2 := dsolve({diff(x(t), [`$`(t, 2)]) = (2*Pi)^2*uwr*cos(2*Pi*t)-
2*daempfx/(2*sqrt(kx*mass))*(2*Pi*sqrt(kx/mass))/(rdz*sqrt(kx/mass))*
(diff(x(t),t)))- (2*Pi*sqrt(kx/mass)/(rdz*sqrt(kx/mass)))^2*x(t)+
(piecewise(sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)>=1,
(2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2*(sqrt((x(t)-xoff)^2+
(y(t)-yoff)^2)-1)+2*kontdaempf/(2*sqrt(kontsteif*mass))*
piecewise(sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t))*(x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+
(diff(y(t), t))*(y(t)-yoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)))*
(-(x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+
piecewise(-(diff(x(t), t))*(y(t)-yoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+
(diff(y(t), t))* (x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2) < 0, -1,
1)*rbw*(y(t)-yoff)/sqrt((x(t)-xoff)^2+ (y(t)-yoff)^2)), x(0) = 0,
D(x)(0) = 0, diff(y(t), [`$`(t, 2)]) = (2*Pi)^2*uwr*sin(2*Pi*t)-
2*daempfy/(2*sqrt(ky*mass))*(2*Pi*sqrt(ky/mass))/(rdz*sqrt(kx/mass))*
(diff(y(t),t))- (2*Pi*sqrt(ky/mass)/(rdz*sqrt(kx/mass)))^2*y(t)+
(piecewise(sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)>=1,
(2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2*(sqrt((x(t)-xoff)^2+
(y(t)-yoff)^2)-1)+ 2*kontdaempf/(2*sqrt(kontsteif*mass))*
piecewise(sqrt((x(t)-xoff)^2+ (y(t)-yoff)^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*((diff(x(t), t))*
(x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+(diff(y(t), t))*
(y(t)-yoff)/sqrt((x(t)-xoff)^2+ (y(t)-yoff)^2)))*
(-(y(t)-yoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)-
piecewise(-(diff(x(t), t))*(y(t)-yoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+
(diff(y(t), t))* (x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2) < 0, -1,
1)*rbw*(x(t)-xoff)/sqrt((x(t)-xoff)^2+ (y(t)-yoff)^2)), y(0) = 0 ,
D(y)(0) = 0}, numeric, method = gear, minstep = 10^(-10), maxstep = 0.1,
maxfun = 0, output = listprocedure);
```

```
(6) a := odeplot(RoSt2, [x(t), y(t)], 270 .. 300, numpoints = 3000)
(7) b := implicitplot((x-0.2)^2+(y-0.2)^2 = 1, x = -0.8 .. 1.2, y = -0.8
.. 1.2, colour = blue)
(8) display(a,b)

(9) allRoots := proc (f::procedure, rng::range, ` $`)
local result, root;
result := Vector(0, 'datatype = float');
root := RootFinding:-NextZero(f, lhs(rng), 'maxdistance' = rhs(rng)-lhs(rng));
while root ≠ FAIL and root ≤ rhs(rng) do result(numelems(result)+1) :=
root;
root := RootFinding:-NextZero(f, root, 'maxdistance' = rhs(rng)-root);
end do;
return result;
end proc;
(10) k:=1;
(11) for j from 0.765 by 0.0001 to 0.815 do
rdz:=j;
RoSt2 := dsolve({diff(x(t), [`$`(t, 2)]) = (2*Pi)^2*uwr*cos(2*Pi*t)-
2*daempfx/(2*sqrt(kx*mass))*(2*Pi*sqrt(kx/mass))/(rdz*sqrt(kx/mass))*
(diff(x(t),t)))-(2*Pi*sqrt(kx/mass)/(rdz*sqrt(kx/mass)))^2*x(t)
+(piecewise(sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)>=1,
(2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2 *(sqrt((x(t)-xoff)^2+
(y(t)-yoff)^2)-1)+2*kontdaempf/(2*sqrt(kontsteif*mass))*
piecewise(sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*((diff(x(t), t))*
(x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+(diff(y(t), t))*
(y(t)-yoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)))*
(-(x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+
piecewise(-(diff(x(t), t))*(y(t)-yoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+
(diff(y(t), t))* (x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2) < 0, -1,
1)*rbw*(y(t)-yoff)/sqrt((x(t)-xoff)^2+
(y(t)-yoff)^2)), x(0) = 0, D(x)(0) = 0, diff(y(t), [`$`(t, 2)]) = (2*Pi)^2*
uwr*sin(2*Pi*t)-2*daempfy/(2*sqrt(ky*mass))*
(2*Pi*sqrt(ky/mass))/(rdz*sqrt(kx/mass))*(diff(y(t),t))-
(2*Pi*sqrt(ky/mass)/(rdz*sqrt(kx/mass)))^2*y(t)+
(piecewise(sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)>=1,
(2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2*
(sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)-1)+2*kontdaempf/(2*sqrt(kontsteif*mass))*
piecewise(sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t))*(x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+
(diff(y(t), t))* (y(t)-yoff)/sqrt((x(t)-xoff)^2+ (y(t)-yoff)^2)))*
(-(y(t)-yoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)-
piecewise(-(diff(x(t), t))*(y(t)-yoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2)+
(diff(y(t), t))* (x(t)-xoff)/sqrt((x(t)-xoff)^2+(y(t)-yoff)^2) < 0, -1,
1)*rbw*(x(t)-xoff)/sqrt((x(t)-xoff)^2+ (y(t)-yoff)^2)), y(0) = 0 ,
D(y)(0) = 0}, numeric, method = gear, minstep = 10^(-10), maxstep = 0.1,
maxfun = 0, output = listprocedure);
```

```
zpkte:=allRoots(rhs(RoSt2_4),270..300):

for i from 1 to numelems(zpkte) do
nst[i]:=rhs(RoSt2_2(zpkte[i])):
end do;
liste[k]:=[[j,nst[n]]$(n=1..numelems(zpkte))]:
c[k]:=pointplot(liste[k]):
k:=k+1;
end do;

(12) display(seq(c[l], l = 1 .. k-1));
```

Wie bereits vom Programmcode des 1. Modells bekannt, wird nach der Initialisierung (1) und dem Laden des Pakets für das Erstellen von Grafiken (2) wird die Anzahl der Stellen für die Rechengenauigkeit (3) festgelegt. Die Parameter werden als globale Variablen gespeichert (4). Dies wurde durch einen Fehler in Maple im Zusammenhang mit dem verwendeten Verfahren notwendig. In späteren Versionen sollte dieses Modell analog wie im ersten Modell auch durch ein lokales Speichern der Parameter funktionieren. In (5) wird die Differenzialgleichung mit den Anfangswerten mittels $dsolve\{...\}$ eingegeben. $numeric, method = gear$ sucht die numerische Lösung mittels eines Ein-Schritt-Extrapolationsverfahrens. $minstep$ und $maxstep$ legen die minimale und maximale Schrittweite zur besseren Kontrolle des Verfahrens fest, und $maxfun = 0$ deaktiviert wieder das Berechnugslimit für die numerische Lösung.
Die Schritte (6), (7) und (8) dienen, analog zum ersten Modell, wieder dem Erstellen von Orbitgrafiken.
Da es nicht möglich ist $events = [...]$ im Zusammenhang mit dem Gear-Verfahren zu verwenden wurde für das zweite Modell eine alternative Methode zum Auffinden der Nullstellen gewählt: Die Prozedur (9) findet alle Nullstellen eines Intervalls mittels des $RootFinding$ Pakets und dem Befehl $NextZero$, welcher die nächste Nullstelle einer Funktion in einem gegebenen Intervall sucht. Nach der Variablendeklaration wird ein leerer Vektor in Gleitkommadarstellung erzeugt. Dieser Vektor wird nun mit den Zeitpunkten der Nullstellen befüllt. Wird eine Nullstelle im vorgegebenen Intervall gefunden, so wird sie im Vektor gespeichert, und die linke Seite des Intervalls für $NextZero$ auf den Zeitpunkt der letzten gefundenen Nullstelle gesetzt. Dies wird solange gemacht bis die rechte Seite des Intervalls erreicht wird.
Die Schleife (11) iteriert den gewünschten Parameter, hier die Rotordrehzahl rdz im gewünschten Bereich mit gewählter Schrittweite. Da der Parameter global festgelegt wurde, muss die Differenzialgleichung (5) bei jedem Schleifendurchlauf neu initialisiert werden. Im nächsten Schritt werden nun die Nullstellen der Differenzialgleichung mittels (9), wieder im Intervall [270,300] berechnet. $output = listprocedure$ lässt mittels Index den Zugriff auf die in der Differenzialgleichung verwendeten Variablen zu. Im Fall des zweiten Modells sind das $(t, x(t), \dot{x}(t), y(t), \dot{y}(t))$. Daher berechnet der Index 4 die Nullstellen von $y(t)$. Die folgende for-Schleife berechnet die zugehörigen x-Koordinaten (Index 2) an den gefundenen Zeitpunkten der Nullstellen. Wie im ersten Modell wird eine Liste von Punkten generiert, die als 1. Koordinate den variierten Parameter, und als 2. Koordinate die x-Koordinate der Nullstellen haben, welche mittels (12) zum Bifurkationsdiagramm zusammengefasst werden.

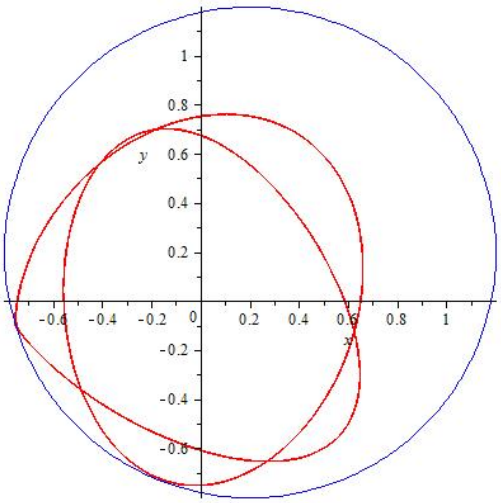
3.2.2 Simulationsergebnisse

Alle Werte der Modellparameter aus dem ersten Modell, die auch im zweiten Modell vorkommen, wurden übernommen. Die Wahl der Kontaktsteifigkeit k_h wurde in Anlehnung an die Wertewahl in [10] gewählt.
Die gewählten Parameter, beziehungsweise deren Variationsbereiche, sind in folgender Tabelle aufgelistet. Dabei ist zu beachten, dass die Kontaktsteifigkeit k_h nicht variiert wurde, was den Vergleich zwischen Modell 1 und Modell 2 bei sonst gleichen Parametern etwas verzerrt. Da a priori nicht klar ist ob es bei Variation von k_h zu einer Übereinstimmung beider Modelle kommen kann und ob der zu findende Zusammenhang linear ist wurde hier von der zusätzlichen Variation von k_h abgesehen.

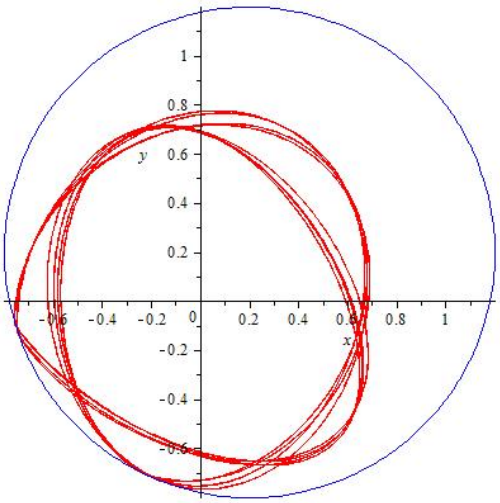
Bezeichnung	**Parameter**	**Wert**	**Einheit**
Rotormasse	m	8.75	kg
Steifigkeit in x-Richtung	$k_{W,x}$	150000	N/m
Steifigkeit in y-Richtung	$k_{W,y}$	150000	N/m
Dämpfung in x-Richtung	$c_{W,x}$	150	Ns/m
Dämpfung in y-Richtung	$c_{W,y}$	150	Ns/m
Kontaktsteifigkeit	k_h	1500000000	N/m
Kontaktdämpfung	c_h	0	Ns/m
Offset in x-Richtung	u_{OFF}	0.2	
Offset in y-Richtung	v_{OFF}	0.2	
Coulomb'scher Reibbeiwert	μ	0.0 - 0.3	
Dimensionslose Drehzahl	η	0.765 - 0.815	
Bezogener Unwuchtradius	e	0.50	

Tabelle 3.1: Simulationsparameter für die Untersuchung des Verzweigungsverhaltens

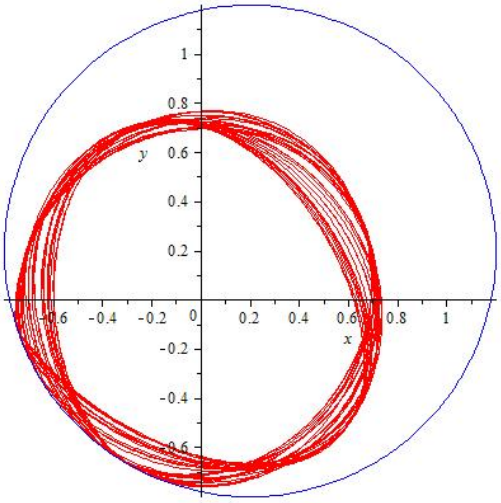
Wie die folgenden Abbildungen 3.1 - 3.4 zeigen, stimmen die beiden Modelle qualitativ trotzdem praktisch überein. Nur in einem kleinen Bereich knapp rechts von der Mitte im Verzweigungsdiagramm gibt es kleinere Unterschiede. Als Vergleich dienen die Abbildungen 2.5 - 2.8. Daraus lässt sich schlussfolgern, dass beide Simulationsmodelle bei richtiger Parameterwahl und angepassen Rechengenauigkeiten vergleichbare Ergebnisse liefern.



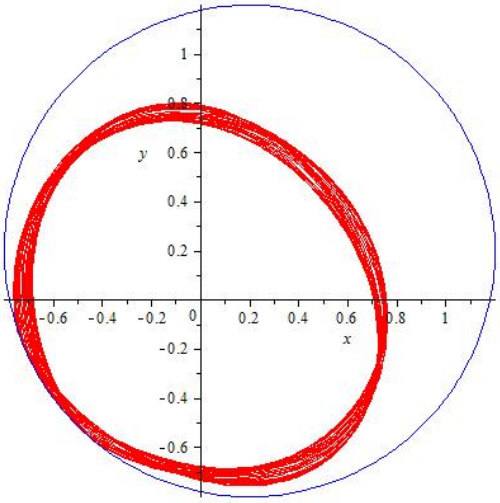
(a) Rotordrehzahl 0.7750



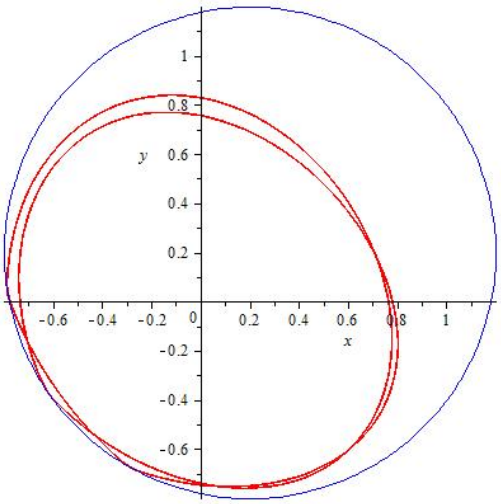
(b) Rotordrehzahl 0.7770



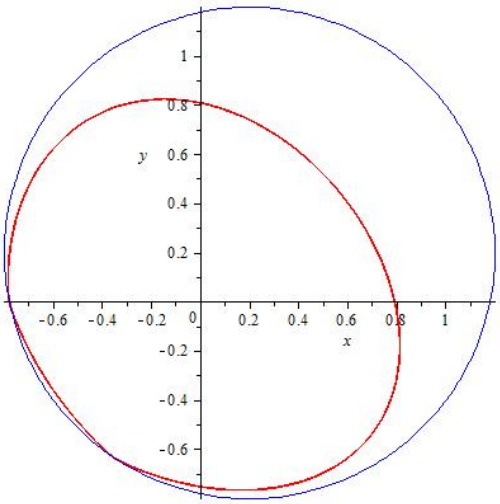
(c) Rotordrehzahl 0.7790



(d) Rotordrehzahl 0.7882



(e) Rotordrehzahl 0.8040



(f) Rotordrehzahl 0.8100

Abbildung 3.1: Orbits für ausgewählte Drehzahlen. Kontaktdämpfung 0, Reibbeiwert 0

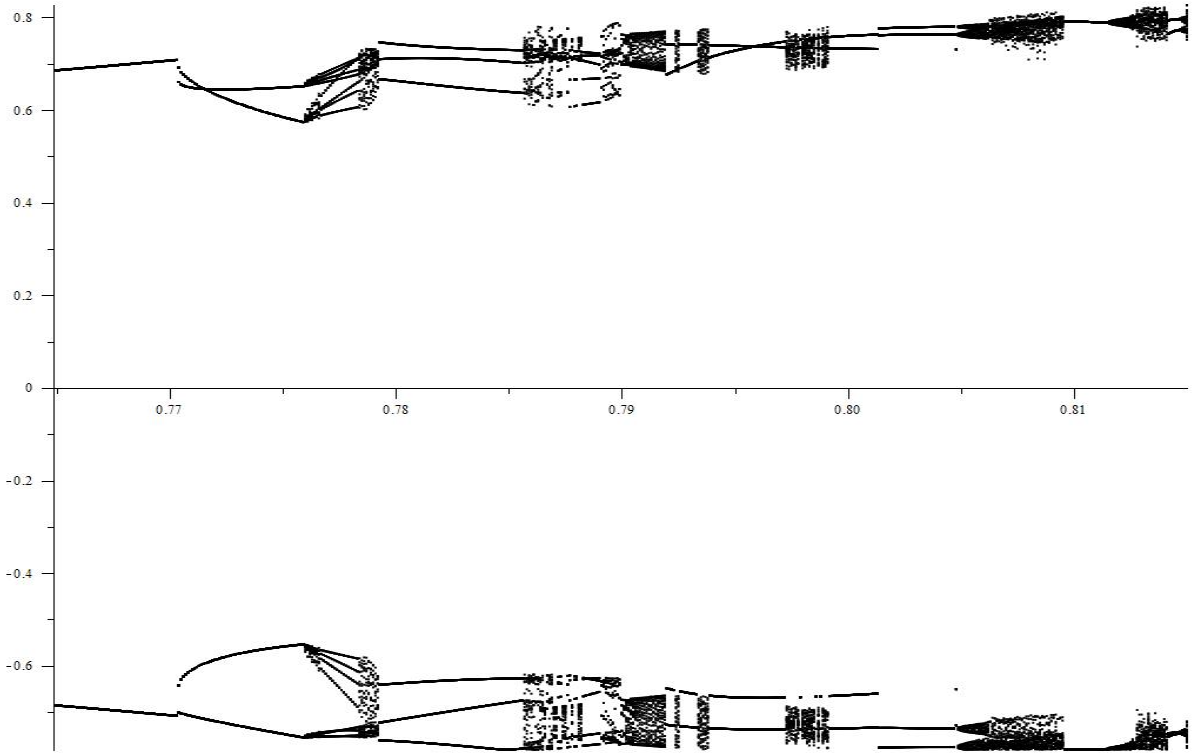


Abbildung 3.2: Kontaktdämpfung = 0 und Reibbeiwert = 0

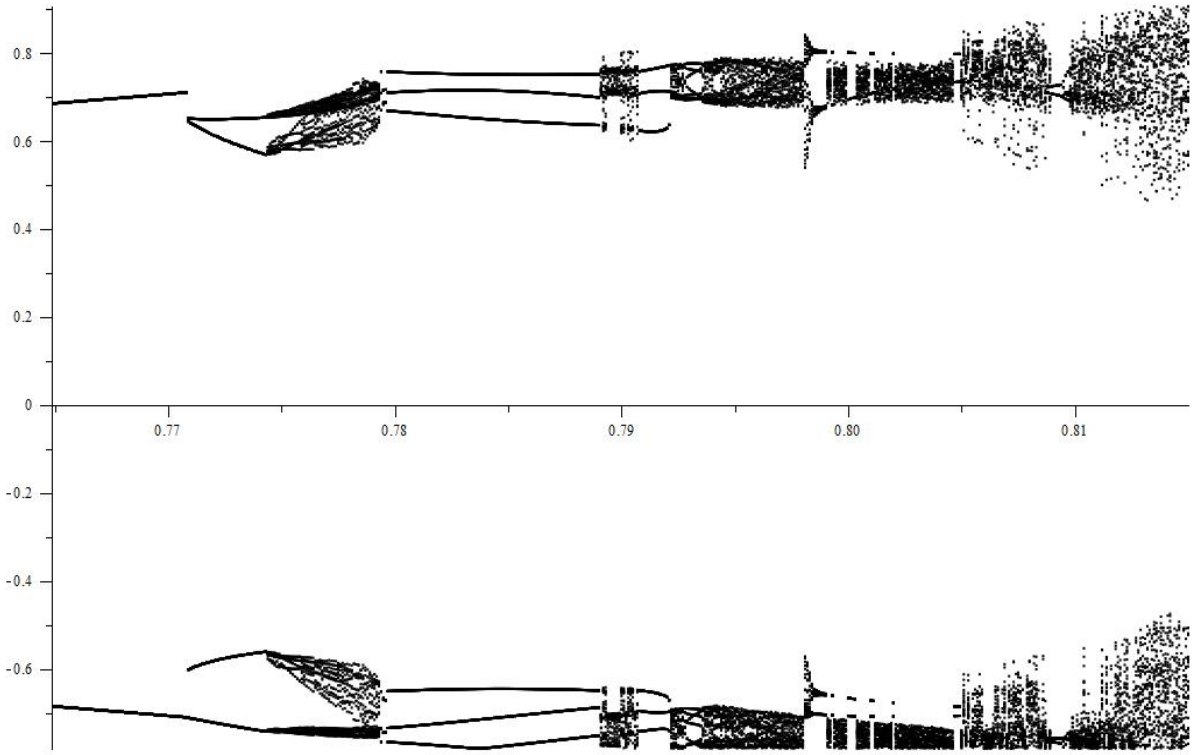
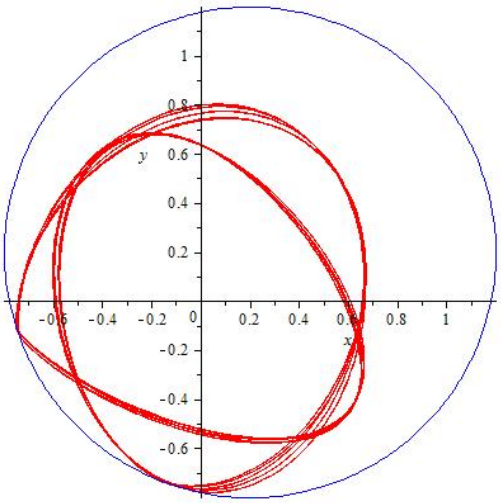
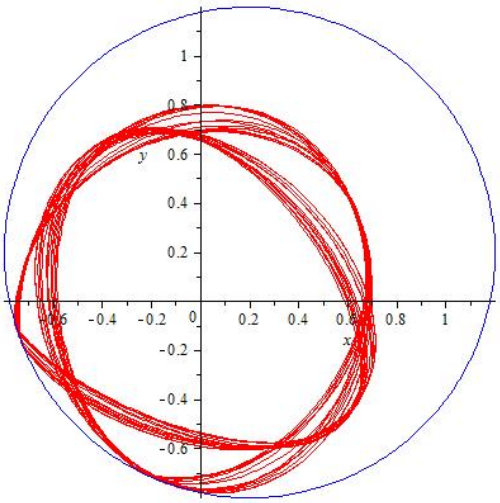


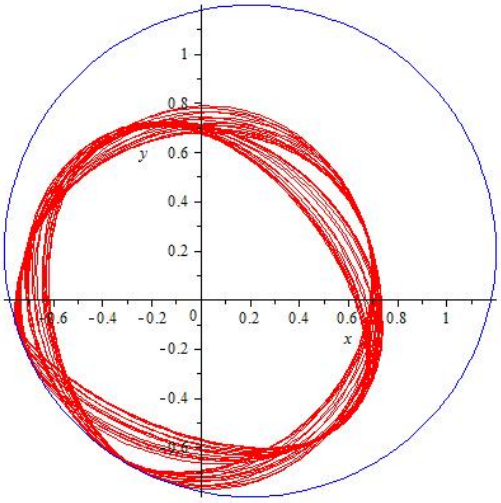
Abbildung 3.3: Kontaktdämpfung = 0 und Reibbeiwert = 0.3



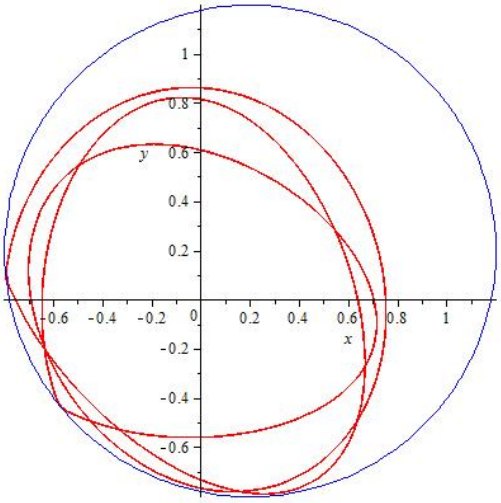
(a) Rotordrehzahl 0.7750



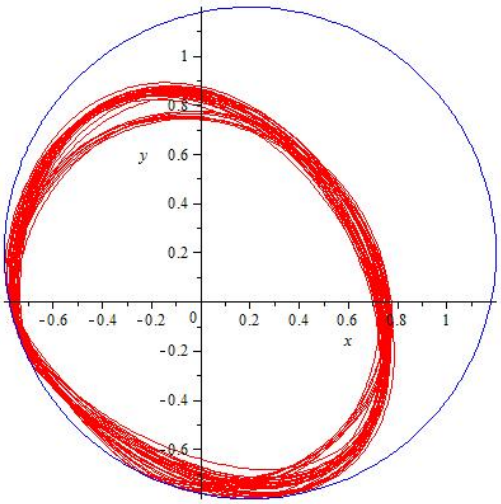
(b) Rotordrehzahl 0.7770



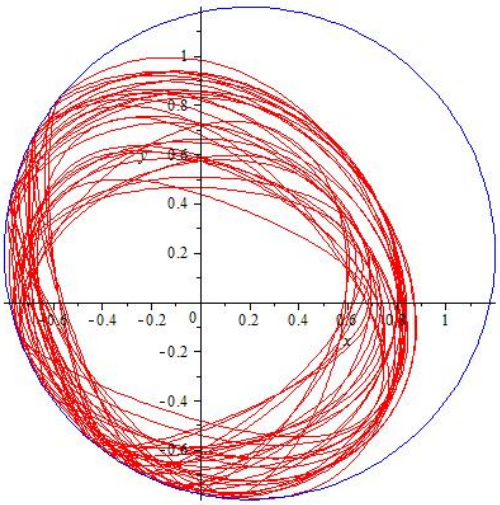
(c) Rotordrehzahl 0.7790



(d) Rotordrehzahl 0.7882



(e) Rotordrehzahl 0.8040



(f) Rotordrehzahl 0.8100

Abbildung 3.4: Orbits für ausgewählte Drehzahlen. Kontaktdämpfung 0, Reibbeiwert 0.3

3.2.3 Maple Code des 2. Modells mit Rosenbrock Verfahren und Simulationsergebnisse mit Verzweigungsverhalten

Maple steht nebem den *Gear* Verfahren noch ein ein weiteres Verfahren für die numerische Lösung steifer Probleme zur Verfügung: Das *Rosenbrock* Verfahren.
Dies ist ein implizietes Runge-Kutta Verfahren dritter bis vierter Ordnung mit Interpolanten vom Grad Drei, und ist das numerische Standardlösungsverfahren für steife Probleme in Maple. Der einzige Unterschied im Code im Vergleich zum Kapitel 3.2.1 ist, dass der Befehl $method = gear$ durch $method = rosenbrock$ ersetzt wird und es keine Schrittweitenvorgaben ($minstep$, $maxstep$) gibt. Die Simulationsdauer ist mit dem Gear Verfahren vergleichbar. Ein möglicher Vorteil ist, dass man *events* im Rosenbrock Verfahren verwenden kann, im Gear Verfahren allerdings nicht. Daher kann man für das Auffinden der Nullstellen wieder die Schleife $allRoots$ des ersten Modells heranziehen und muss nicht mit der $RootFinding$ Funktion arbeiten. Das spart insgesamt etwas Zeit gegenüber dem Gear Verfahren ein.
Die qualitative Lösung stimmt auch mit der Lösung des Gear Verfahrens großteils überein, nur bei manchen Orbitübergängen von stabil nach instabil liefert das Gear Verfahren bessere Lösungen, die auch näher am ersten Modell liegen als das Rosenbrock Verfahren. Da alle Parameter zum vorigen Kapitel ident geblieben sind, wurde auf eine eigene Parametertabelle verzichtet. Da auch die Orbits praktisch ident zum Gear Verfahren sind wurde auch auf ausgewählte Orbitdiagramme in diesem Kapitel verzichtet, nur die Verzweigungsdiagramme, siehe Abbildungen 3.5 und 3.6, wurden erstellt (für den Vergleich zu Abbildung 3.2 und 3.3 des Gear Verfahrens).

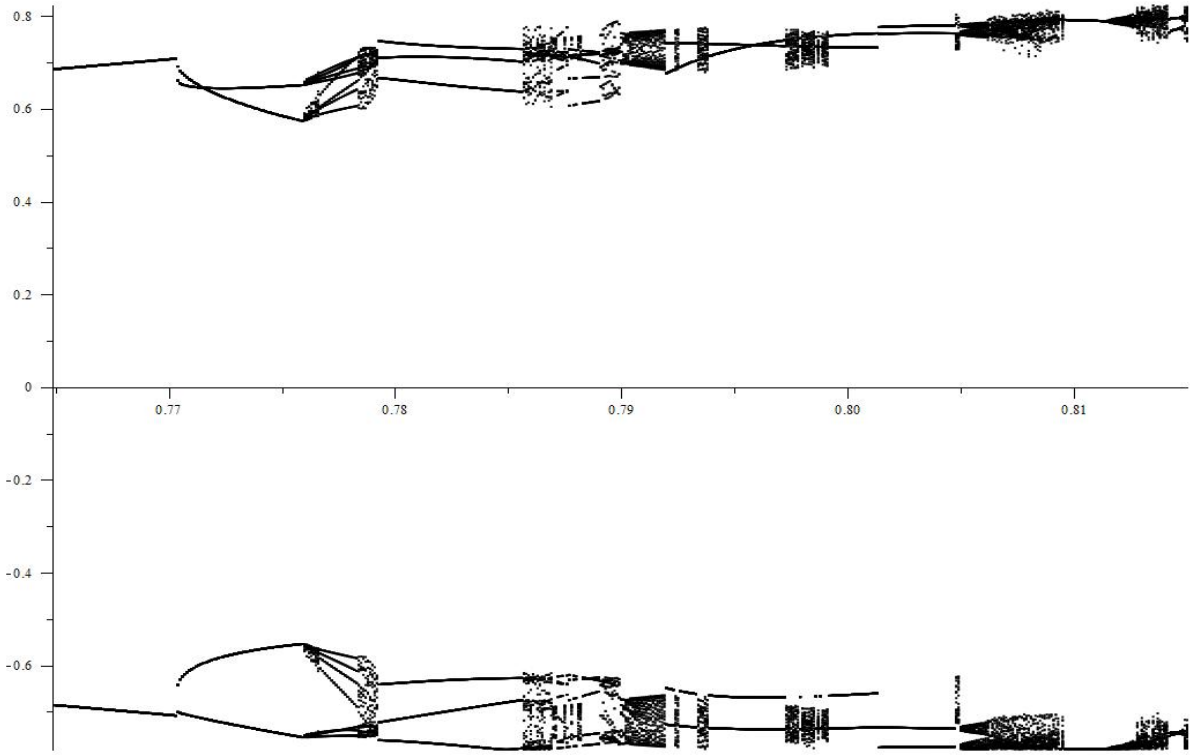


Abbildung 3.5: Kontaktdämpfung = 0 und Reibbeiwert = 0

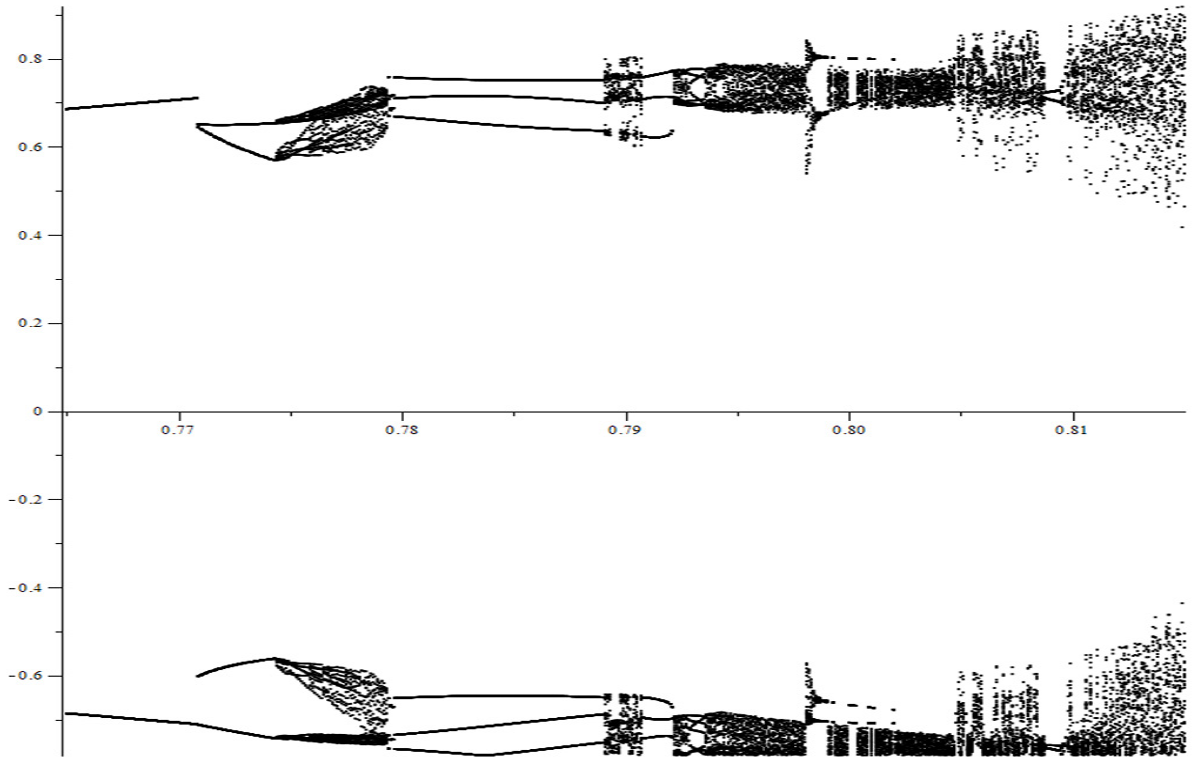


Abbildung 3.6: Kontaktdämpfung = 0 und Reibbeiwert = 0.3

3.2.4 Umsetzung des zweiten Modells in MapleSim

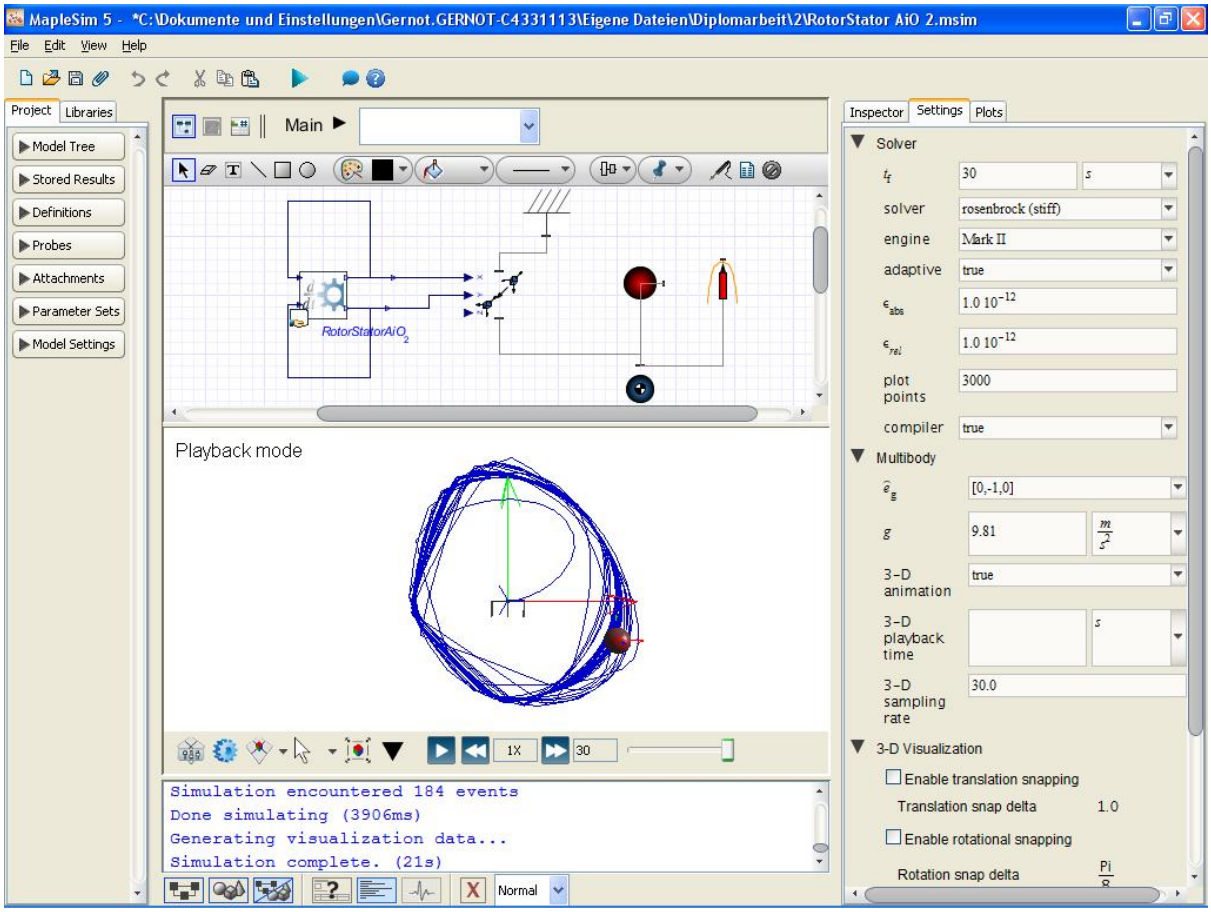


Abbildung 3.7: Modell 2 in MapleSim 5

Wie in Abbildung 3.7 zu sehen, ist die Modellierung des zweiten Modells auch problemlos in MapleSim möglich. Da die Differenzialgleichungen die das Modell beschreiben bekannt sind, ist es nicht notwendig das Modell duch "vorgefertigte"Blöcke zu modellieren. Es genügt eine sogenannte *Custom Modeling Component*, in diesem Modell $RotorStatorAiO_2$ genannt, in der man die Differenzialgleichungen (3.10) und (3.11) genau wie im Maple Code unter (5) beschrieben, nur ohne Angabe der Anfangswerte und der weiteren Parameter, angibt. Die Angabe aller Parameter und der Ein- bzw. Ausgänge der Komponente vervollständigen diese. Die Komponenten rechts davon (*Fixed Frame*, *Prescribed Translation*, *Spherical Geometry*, *Rigid Body* und *Path Trace*) dienen zur Visualisierung der Daten mittels einer sich bewegenden roten Kugel inklusive blauer Bewegungsspur, die den Wellendurchtrittspunkt repräsentiert, zu sehen im *Playback mode* Fenster. Für die optimale Darstellung werden die Daten, die ja in der $xy-$Ebene liegen, in die $xz-$Ebene platziert. Für die Berechnung wurde der einzig in MapeSim implementierte Stiff-Solver (Rosenbrock) verwendet. Die maximal möglichen absoluten- und relativen Fehlertoleranzen betragen hierbei $5 * 10^{-13}$. Die Anfangswerte und die Werte aller Parameter lassen sich, wie auch die Zeitdauer der Simulation und ob eine adaptive Schrittweitensteuerung erwünscht ist oder nicht, in MapleSim direkt eingeben.

Kapitel 4

Modell 3: Simulationsmodell mit kontaktmechanischer Hypothese und elastisch gelagertem Stator

4.1 Theoretische Grundlagen und Modellbildung

Dieses Kapitel ist eine Kurzzusammenfassung der Theorie aus [13]. Die Bewegungsgleichungen für den Rotor können direkt aus dem zweiten Modell übernommen werden. Für den Stator sind die Bewegungsgleicgungen analog aufgebaut, der Unwuchtterm tritt allerdings nicht auf und die Richtungen der Kontaktkraftkomponenten müssen umgedreht werden. Die verwendeten Steifigkeiten $k_{L,x}$ und $k_{L,y}$ und Dämpfungen $c_{L,x}$ und $c_{L,y}$ sind direkte Statorparameter und setzen sich nicht aus mehreren Anteilen zusammen. Der Index "L"kennzeichnet hier, dass es sich um einen Statorparameter handelt.
Da in diesem Modell wieder ein Offset zwischen dem Ursprung des Statorruhesystems und dem Rotorsystem berücksichtigt wird und sich hier der Ursprung des Fanglagersystems zufolge des Kontakts zwischen Rotor und Stator verschieben kann, muss dies auch bei der ermittling des Winkels ψ und bei der Formulierung der Kontaktbedingung beachtet werden.

$$\lambda_{WL} = u_W - (u_L + u_{off}) \tag{4.1}$$

$$\nu_{WL} = v_W - (v_L + v_{off}) \tag{4.2}$$

Dabei geben die Größen λ_{WL} und ν_{WL} die dimensionslosen Auslenkungen des Rotormittelpunktes bezüglich des Fanglagermittelpunktes in den beiden Koordinatenrichtungen an.Mit deren Hilfe kann die dimensionslose radiale Auslenkung σ_{WL}, und somit auch die Eindringtiefe δ ermittelt werden.

$$\sigma_{WL} = \sqrt{\lambda_{WL}^2 + \nu_{WL}^2} \tag{4.3}$$

$$\delta = \sigma_{WL} - 1 > 0 \tag{4.4}$$

Der Winkel ψ wird analog zu den anderen beiden Modellen ermittelt.

$$\tan\psi = \frac{\nu_{WL}}{\lambda_{WL}} \qquad \cos\psi = \frac{\lambda_{WL}}{\sigma_{WL}} \qquad \sin\psi = \frac{\nu_{WL}}{\sigma_{WL}} \tag{4.5}$$

Zusätzlich muss noch die notwendige Beziehung für die dimensionslose Geschwindigkeit in Normalenrichtung δ' aufgestellt werden. Da sich auch der Stator bewegt, muss mit den Relativgeschwindigkeiten gerechnet werden.

$$\lambda'_{WL} = u'_W - u'_L \tag{4.6}$$

$$\nu'_{WL} = v'_W - v'_L \tag{4.7}$$

$$\delta' = \lambda'_{WL}\cos\psi + \nu'_{WL}\sin\psi \tag{4.8}$$

Die Bewegungsgleichungen für den Stator und den Rotor werden in gleicher Weise zeit- und längennormiert, wie bei den anderen beiden Modellen. Somit erhält man die Bewegungsgleichungen in $x-$ und $y-$Richtung für das dritte Modell, wie sie in Maple eingegeben wurden.

Rotor:

$$u''_W = (2\pi)^2 e\cos(2\pi\tau) - 2\frac{c_{W,x}}{2\sqrt{k_{W,x}*m_W}}(2\pi\frac{\sqrt{\frac{k_{W,x}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})u'_W - (2\pi\frac{\sqrt{\frac{k_{W,x}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2 u_W +$$
$$+((2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2\delta + 2\frac{c_h}{2\sqrt{k_h m}}2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}}((u'_W - u'_L)\cos\psi + (v'_W - v'_L)\sin\psi))(-\cos\psi + \delta_f\mu\sin\psi) \tag{4.9}$$

$$v''_W = (2\pi)^2 e\sin(2\pi\tau) - 2\frac{c_{W,y}}{2\sqrt{k_{W,y}*m_W}}(2\pi\frac{\sqrt{\frac{k_{W,y}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})v'_W - (2\pi\frac{\sqrt{\frac{k_{W,y}}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2 v_W +$$
$$+((2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2\delta + 2\frac{c_h}{2\sqrt{k_h m}}2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}}((u'_W - u'_L)\cos\psi + (v'_W - v'_L)\sin\psi))(-\sin\psi - \delta_f\mu\cos\psi) \tag{4.10}$$

Stator:

$$u''_L = -2\frac{c_{L,x}}{2\sqrt{k_{L,x}*m_L}}(2\pi\frac{\sqrt{\frac{k_{L,x}}{m_L}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})u'_L - (2\pi\frac{\sqrt{\frac{k_{L,x}}{m_L}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2 u_L -$$
$$-((2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2\delta + 2\frac{c_h}{2\sqrt{k_h m}}2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}}((u'_W - u'_L)\cos\psi + (v'_W - v'_L)\sin\psi))(-\cos\psi + \delta_f\mu\sin\psi) \tag{4.11}$$

$$v''_L = -2\frac{c_{L,y}}{2\sqrt{k_{L,y}*m_L}}(2\pi\frac{\sqrt{\frac{k_{L,y}}{m_L}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})v'_L - (2\pi\frac{\sqrt{\frac{k_{L,y}}{m_L}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2 v_L -$$
$$-((2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}})^2\delta + 2\frac{c_h}{2\sqrt{k_h m}}2\pi\frac{\sqrt{\frac{k_h}{m_W}}}{\eta\sqrt{\frac{k_{W,y}}{m_W}}}((u'_W - u'_L)\cos\psi + (v'_W - v'_L)\sin\psi))(-\sin\psi - \delta_f\mu\cos\psi) \tag{4.12}$$

4.2 Modellimplementierung und Simulationsergebnisse

Der folgende Abschnitt enthält den Programmiercode mit anschließender Erklärung.

4.2.1 Maple Code des 3. Modells mit Gear Verfahren

```
(1) restart
(2) with(plots):
(3) Digits:=15;

(4) uwr := 0.5; rdz := 0.765; daempfx := 150; daempfy := 150; kontdaempf
:= 0; kx := 150000; ky := 15000; mass := 8.75; xoff := 0.2; yoff := 0.2;
kontsteif := 1500000000; rbw := 0.3;
sdaempfx := 10000; sdaempfy:=10000; kontsteifx := 450000; kontsteify :=
450000; flmass:=300;

(5) RoSt3 := dsolve({
diff(x(t), [`$`(t, 2)]) = (2*Pi)^2*uwr*cos(2*Pi*t)-
2*daempfx/(2*sqrt(kx*mass))*2*Pi*sqrt(kx/mass)/(rdz*sqrt(kx/mass))*
(diff(x(t),t))-(2*Pi*sqrt(kx/mass)/(rdz*sqrt(kx/mass)))^2*x(t)+
(piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2) >= 1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2*
(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-1)+
2*kontdaempf/(2*sqrt(kontsteif*mass))*
piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t)-diff(u(t),t))* (x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2
+(y(t)-(v(t)+yoff))^2)+(diff(y(t), t)-diff(v(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+ (y(t)-(v(t)+yoff))^2)))*
(-(x(t)-(u(t)+xoff)/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
piecewise(-(diff(x(t), t)-diff(u(t),t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
(diff(y(t), t)-diff(v(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)<0, -1,1)*rbw*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)), x(0)=0,
D(x)(0)=0,
diff(y(t), [`$`(t, 2)]) = (2*Pi)^2*uwr*sin(2*Pi*t)-
2*daempfy/(2*sqrt(ky*mass))*
2*Pi*sqrt(ky/mass)/(rdz*sqrt(kx/mass))*(diff(y(t), t))-
(2*Pi*sqrt(ky/mass)/(rdz*sqrt(kx/mass)))^2*y(t)+
(piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2) >= 1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2
*(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-1)+
2*kontdaempf/(2*sqrt(kontsteif*mass))*
piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t)-diff(u(t), t))* (x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
```

```
(y(t)-(v(t)+yoff))^2)+(diff(y(t), t)-diff(v(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)))*
(-(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-
piecewise(-(diff(x(t), t)-diff(u(t),t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
(diff(y(t), t)-diff(v(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)<0,-1, 1)*rbw*
(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)), y(0)=0,
D(y)(0)=0,
diff(u(t), [`$`(t, 2)]) = -2*sdaempfx/(2*sqrt(skontsteifx*flmass))*
2*Pi*sqrt(skontsteifx/flmass)/(rdz*sqrt(kx/mass))*(diff(u(t), t))-
(2*Pi*sqrt(skontsteifx/flmass)/(rdz*sqrt(kx/mass)))^2*u(t)-
(piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2) >= 1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)),0)^2*
(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-1)+
2*kontdaempf/(2*sqrt(kontsteif*mass))*piecewise(sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t)-diff(u(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)+(diff(y(t), t)-diff(v(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)))*
(-(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
piecewise(-(diff(x(t), t)-diff(u(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
(diff(y(t), t)-diff(v(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)<0, -1,1)*rbw
*(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)),
u(0)=0, D(u)(0)=0,

diff(v(t), [`$`(t, 2)]) = -2*sdaempfy/(2*sqrt(skontsteify*flmass))*
2*Pi*sqrt(skontsteify/flmass)/(rdz*sqrt(kx/mass))*(diff(v(t), t))-
(2*Pi*sqrt(skontsteify/flmass)/(rdz*sqrt(kx/mass)))^2*v(t)-
(piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2) >= 1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2*
(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-1)+
2*kontdaempf/(2*sqrt(kontsteif*mass))*piecewise(sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t)-diff(u(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)+(diff(y(t), t)-diff(v(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)))*
(-(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-
piecewise(-(diff(x(t), t)-diff(u(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
(diff(y(t), t)-diff(v(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)<0, -1,1)*rbw*
```

```
(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)), v(0)=0,
D(v)(0)=0},
numeric, method = gear, minstep = 10^(-10), maxstep = 0.1, maxfun = 0);

(6) a := odeplot(RoSt3, [x(t), y(t)], 170 .. 200, numpoints = 3000, colour
= blue)
(7) b := odeplot(RoSt3, [u(t), v(t)], 170 .. 200, numpoints = 3000, colour
= red)
(8) display(a, b)

(9) allRoots := proc (f::procedure, rng::range, ` $`)
local result, root;
result := Vector(0, 'datatype = float');
root := RootFinding:-NextZero(f, lhs(rng), 'maxdistance' = rhs(rng)-lhs(rng));
while root ≠ FAIL and root ≤ rhs(rng) do result(numelems(result)+1) :=
root;
root := RootFinding:-NextZero(f, root, 'maxdistance' = rhs(rng)-root);
end do;
return result;
end proc;

(10) k:=1;

(11) for j from 0.765 by 0.0001 to 0.815 do
rdz:=j;
RoSt3 := dsolve({
diff(x(t), [`$`(t, 2)]) = (2*Pi)^2*uwr*cos(2*Pi*t)-
2*daempfx/(2*sqrt(kx*mass))*2*Pi*sqrt(kx/mass)/(rdz*sqrt(kx/mass))*
(diff(x(t),t))-(2*Pi*sqrt(kx/mass)/(rdz*sqrt(kx/mass)))^2*x(t)+
(piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2) >= 1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2*
(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-1)+
2*kontdaempf/(2*sqrt(kontsteif*mass))*
piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t)-diff(u(t),t))* (x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2
+(y(t)-(v(t)+yoff))^2)+(diff(y(t), t)-diff(v(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+ (y(t)-(v(t)+yoff))^2)))*
(-(x(t)-(u(t)+xoff)/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
piecewise(-(diff(x(t), t)-diff(u(t),t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
(diff(y(t), t)-diff(v(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)<0, -1,1)*rbw*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)), x(0)=0,
D(x)(0)=0,
diff(y(t), [`$`(t, 2)]) = (2*Pi)^2*uwr*sin(2*Pi*t)-
2*daempfy/(2*sqrt(ky*mass))*
2*Pi*sqrt(ky/mass)/(rdz*sqrt(kx/mass))*(diff(y(t), t))-
(2*Pi*sqrt(ky/mass)/(rdz*sqrt(kx/mass)))^2*y(t)+
```

```
(piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2) >= 1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2
*(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-1)+
2*kontdaempf/(2*sqrt(kontsteif*mass))*
piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t)-diff(u(t), t))* (x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)+(diff(y(t), t)-diff(v(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)))*
(-(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-
piecewise(-(diff(x(t), t)-diff(u(t),t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
(diff(y(t), t)-diff(v(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)<0,-1, 1)*rbw*
(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)), y(0)=0,
D(y)(0)=0,
diff(u(t), [`$`(t, 2)]) = -2*sdaempfx/(2*sqrt(skontsteifx*flmass))*
2*Pi*sqrt(skontsteifx/flmass)/(rdz*sqrt(kx/mass))*(diff(u(t), t))-
(2*Pi*sqrt(skontsteifx/flmass)/(rdz*sqrt(kx/mass)))^2*u(t)-
(piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2) >= 1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)),0)^2*
(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-1)+
2*kontdaempf/(2*sqrt(kontsteif*mass))*piecewise(sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t)-diff(u(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)+(diff(y(t), t)-diff(v(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)))*
(-(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
piecewise(-(diff(x(t), t)-diff(u(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
(diff(y(t), t)-diff(v(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)<0, -1,1)*rbw
*(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)),
u(0)=0, D(u)(0)=0,

diff(v(t), [`$`(t, 2)]) = -2*sdaempfy/(2*sqrt(skontsteify*flmass))*
2*Pi*sqrt(skontsteify/flmass)/(rdz*sqrt(kx/mass))*(diff(v(t), t))-
(2*Pi*sqrt(skontsteify/flmass)/(rdz*sqrt(kx/mass)))^2*v(t)-
(piecewise(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2) >= 1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)^2*
(sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-1)+
2*kontdaempf/(2*sqrt(kontsteif*mass))*piecewise(sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)>=1,
2*Pi*sqrt(kontsteif/mass)/(rdz*sqrt(kx/mass)), 0)*
((diff(x(t), t)-diff(u(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)+(diff(y(t), t)-diff(v(t), t))*
```

```
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)))*
(-(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)-
piecewise(-(diff(x(t), t)-diff(u(t), t))*
(y(t)-(v(t)+yoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)+
(diff(y(t), t)-diff(v(t), t))*(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+
(y(t)-(v(t)+yoff))^2)<0, -1,1)*rbw*
(x(t)-(u(t)+xoff))/sqrt((x(t)-(u(t)+xoff))^2+(y(t)-(v(t)+yoff))^2)), v(0)=0,
D(v)(0)=0},
numeric, method = gear, minstep = 10^(-10), maxstep = 0.1, maxfun = 0,
output = listprocedure);

zpkte:=allRoots(rhs(RoSt3_8),170..200):

for i from 1 to numelems(zpkte) do
nst[i]:=rhs(RoSt3_6(zpkte[i])):
end do;
liste[k]:=[[j,nst[n]]$(n=1..numelems(zpkte))]:
c[k]:=pointplot(liste[k]):
k:=k+1;
end do;

(12) display(seq(c[l], l = 1 .. k-1));
```

Wie bereits vom Programmcode der ersten beiden Modelle bekannt, wird nach der Initialisierung (1) und dem Laden des Pakets für das Erstellen von Grafiken (2) wird die Anzahl der Stellen für die Rechengenauigkeit (3) festgelegt. Die Parameter werden wieder als globale Variablen gespeichert (4). In (5) wird die Differenzialgleichung mit den Anfangswerten mittels $dsolve\{...\}$ eingegeben. $\ddot{x}(t) = ...$ und $\ddot{y}(t) = ...$ beschreiben die Gleichungen für den Rotor, $\ddot{u}(t) = ...$ und $\ddot{v}(t) = ...$ beschreiben die Gleichungen für den Stator. $numeric, method = gear$ sucht wieder die numerische Lösung mittels des Ein-Schritt-Extrapolationsverfahrens. $minstep$ und $maxstep$ legen die minimale und maximale Schrittweite zur besseren Kontrolle des Verfahrens fest, und $maxfun = 0$ deaktiviert wieder das Berechnugslimit für die numerische Lösung.
Die Schritte (6), (7) und (8) dienen, analog zu den anderen beiden Modellen, wieder dem Erstellen von Orbitgrafiken.
Da es nicht möglich ist $events = [...]$ im Zusammenhang mit dem Gear-Verfahren zu verwenden wurde für das zweite Modell eine alternative Methode zum Auffinden der Nullstellen gewählt: Die Prozedur (9) findet alle Nullstellen eines Intervalls mittels des $RootFinding$ Pakets und dem Befehl $NextZero$, welcher die nächste Nullstelle einer Funktion in einem gegebenen Intervall sucht. Nach der Variablendeklaration wird ein leerer Vektor in Gleitkommadarstellung erzeugt. Dieser Vektor wird nun mit den Zeitpunkten der Nullstellen befüllt. Wird eine Nullstelle im vorgegebenen Intervall gefunden, so wird sie im Vektor gespeichert, und die linke Seite des Intervalls für $NextZero$ auf den Zeitpunkt der letzten gefundenen Nullstelle gesetzt. Dies wird solange gemacht bis die rechte Seite des Intervalls erreicht wird.
Die Schleife (11) iteriert den gewünschten Parameter, hier die Rotordrehzahl rdz im gewünschten Bereich mit gewählter Schrittweite. Da der Parameter global festgelegt wurde, muss die Differenzialgleichung (5) bei jedem Schleifendurchlauf neu initialisiert werden. Im nächsten Schritt werden nun die Nullstellen der Differenzialgleichung mittels (9), diesmal im Intervall [170,200] berechnet, da sonst der Rechenaufwand zu hoch wäre. $output = listprocedure$ lässt mittels Index den Zugriff auf die in der Differenzialgleichung verwendeten Variablen zu. Im Fall des dritten Modells sind das $(t, u(t), \dot{u}(t), v(t), \dot{v}(t), x(t), \dot{x}(t), y(t), \dot{y}(t))$. Daher berechnet der Index 8 die Nullstellen von $y(t)$. Die folgende for-Schleife berechnet die zugehörigen x-Koordinaten (Index 6) an den gefundenen Zeitpunkten der Nullstellen. Wie in den anderen Modellen wird eine Liste von Punkten generiert, die als 1. Koordinate den variierten Parameter, und als 2. Koordinate die x-Koordinate der Nullstellen haben, welche mittels (12) zum Bifurkationsdiagramm zusammengefasst werden.

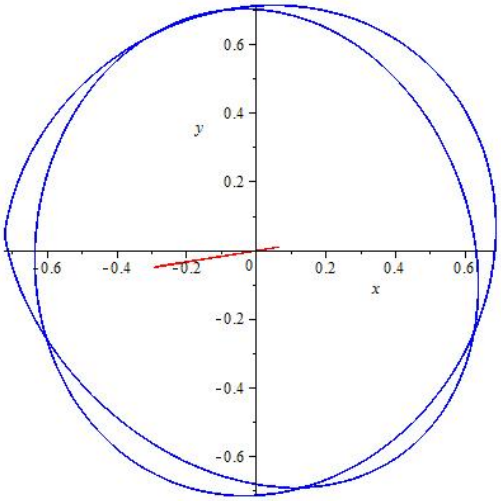
4.2.2 Simulationsergebnisse

Alle Modellparameter für den Rotor bleiben gegenüber dem zweiten Modell unverändert. Zusätzlich müssen noch die Parameter für das Fanglager festgelegt werden. Die Auswahl erfolge so, dass im betrachteten Drehzahlbereich mit den bisher erziehlten Resultaten vergleichbare Lösungen erziehlt werden.
Die gewählten Parameter, beziehungsweise deren Variationsbereich, sind in folgender Tabelle aufgelistet.

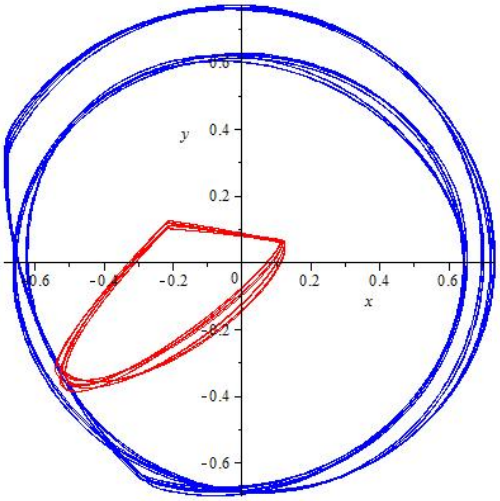
Bezeichnung	**Parameter**	**Wert**	**Einheit**
Rotormasse	m	8.75	kg
Fanglagermasse	m_L	100 - 900	kg
Rotorsteifigkeit in x-Richtung	$k_{W,x}$	150000	N/m
Rotorsteifigkeit in y-Richtung	$k_{W,y}$	150000	N/m
Rotordämpfung in x-Richtung	$c_{W,x}$	150	Ns/m
Rotordämpfung in y-Richtung	$c_{W,y}$	150	Ns/m
Fanglagersteifigkeit in x-Richtung	$k_{L,x}$	450000	N/m
Fanglagersteifigkeit in y-Richtung	$k_{L,y}$	450000	N/m
Fanglagerdämpfung in x-Richtung	$c_{L,x}$	10000	Ns/m
Fanglagerdämpfung in y-Richtung	$c_{L,y}$	10000	Ns/m
Kontaktsteifigkeit	k_h	1500000000	N/m
Kontaktdämpfung	c_h	0	Ns/m
Offset in x-Richtung	u_{OFF}	0.2	
Offset in y-Richtung	v_{OFF}	0.2	
Coulomb'scher Reibbeiwert	μ	0.0	
Dimensionslose Drehzahl	η	0.765 - 0.815	
Bezogener Unwuchtradius	e	0.50	

Tabelle 4.1: Simulationsparameter für die Untersuchung des Verzweigungsverhaltens

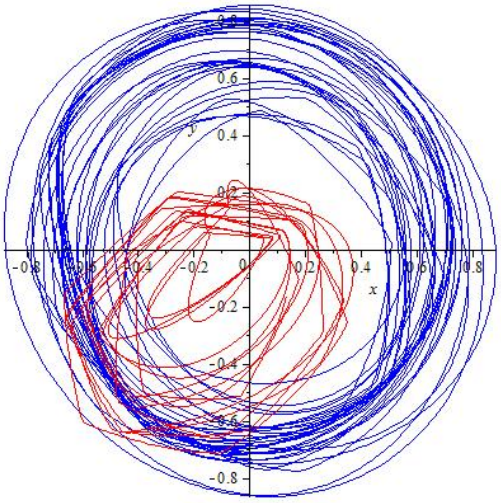
Für dieses Modell wurde das Verzweigungsverhalten mit variierter Rotordrehzahl bei konstanter Rotormasse (Abbildung 4.2) untersucht und dazugehörige Orbitdiagramme (Abbildung 4.1) erstellt. Ausserdem wurde noch das Verzweigungsverhalten bei konstanter Drehzahl mit variierter Fanglagermasse (Abbildung 4.3) erstellt.



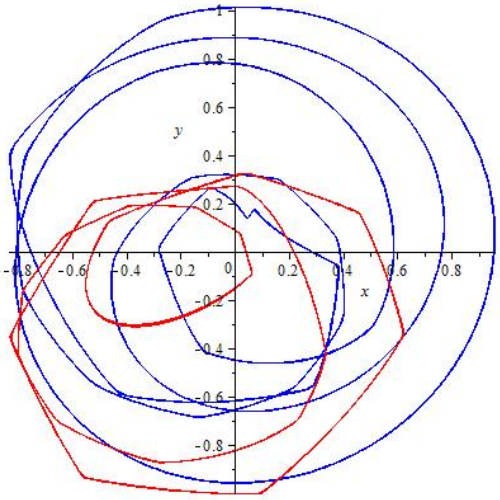
(a) Rotordrehzahl 0.7700



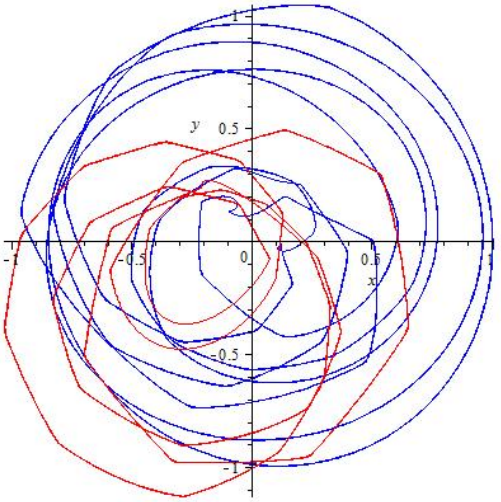
(b) Rotordrehzahl 0.7800



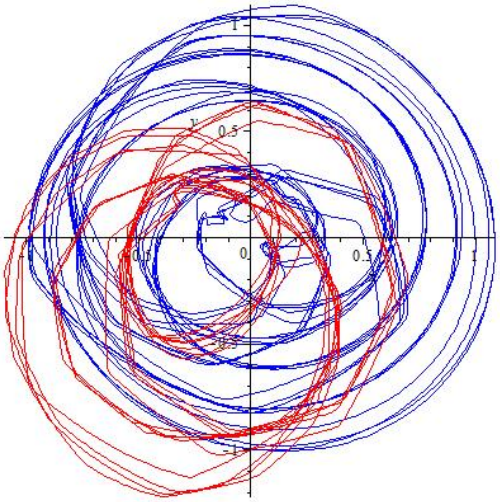
(c) Rotordrehzahl 0.7882



(d) Rotordrehzahl 0.8022



(e) Rotordrehzahl 0.8130



(f) Rotordrehzahl 0.8150

Abbildung 4.1: Orbits bei Fanglagermasse 300, Kontaktdämpfung 0, Reibbeiwert 0

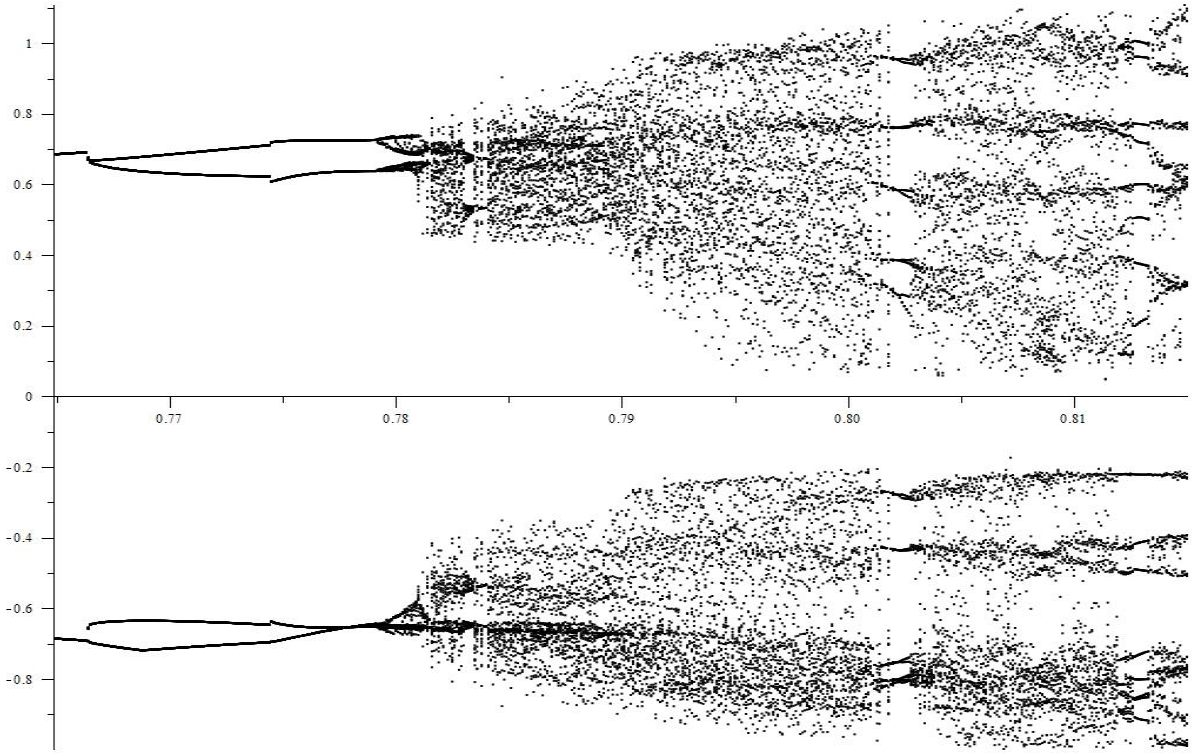


Abbildung 4.2: Fanglagermasse = 300, Kontaktdämpfung = 0 und Reibbeiwert = 0

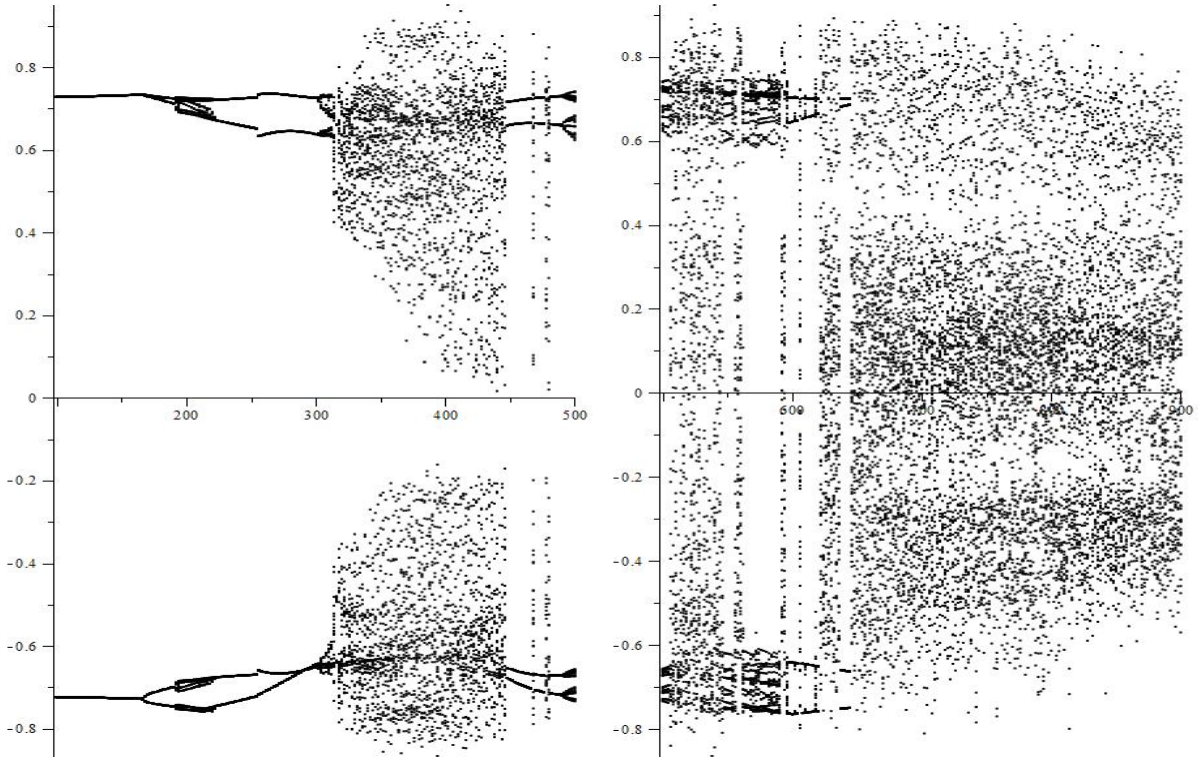


Abbildung 4.3: Modell 3 mit variierter Masse von 100 kg bis 900 kg

4.2.3 Umsetzung des dritten Modells in MapleSim

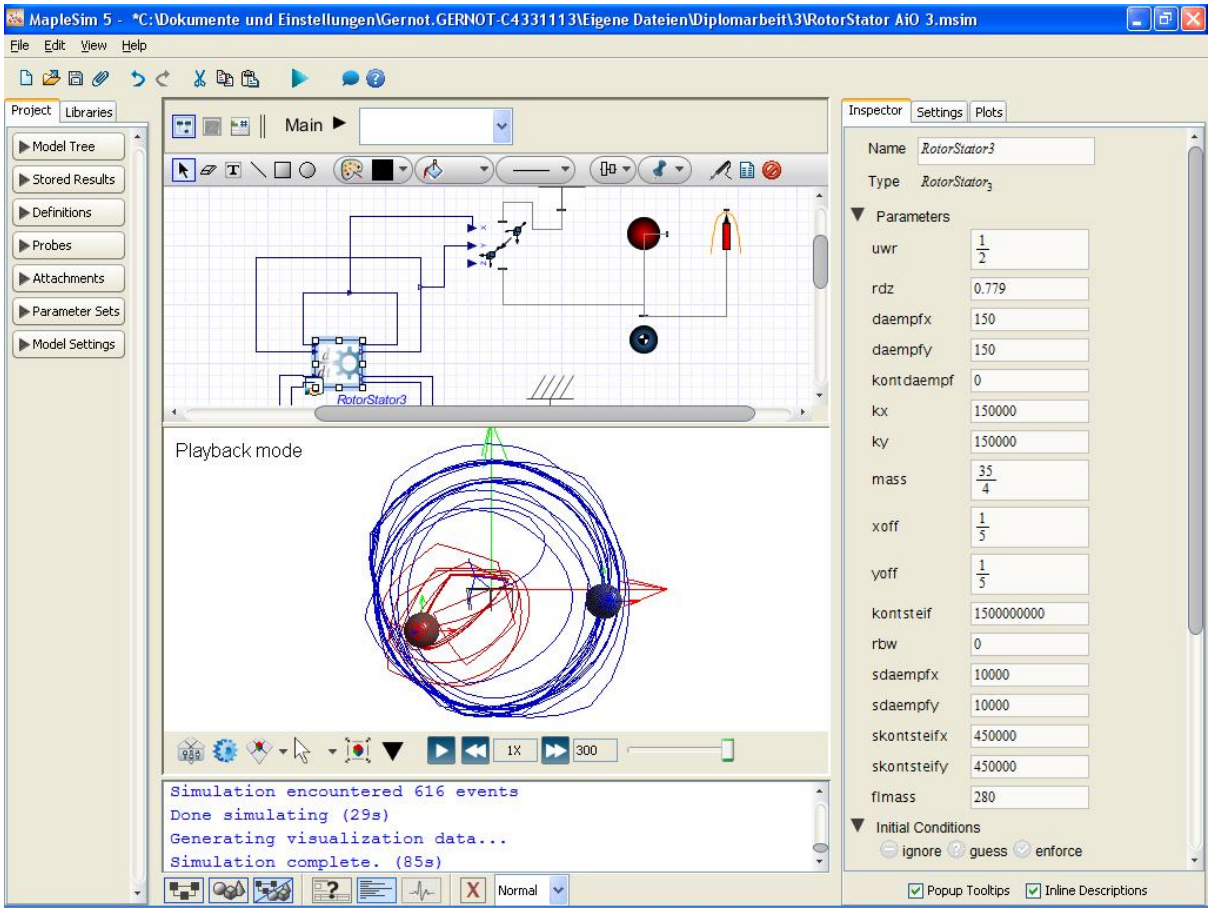


Abbildung 4.4: Modell 3 in MapleSim 5

Wie schon für das zweite Modell beschrieben, ist eine analoge Umsetzung einer Simulation des dritten Modells in MapleSim möglich. Wieder werden die Differenzialgleichungen des Modells (4.9) - (4.12) mittels Maplesyntax, wie im Maple Code unter (5), wieder ohne Angabe der Anfangswerte und mit eigener Befehlszeile zur Bezeichnung der Parameter, beschrieben, in eine *Custom Modeling Component* eingegeben, hier $RotorStatorAiO_3$ genannt (siehe Abbildung 4.4). Es wird ein zusätzlicher Ein- bzw. Ausgang für den sich bewegenden Stator gebraucht. Die Visualisierung erfolgt, inklusive der Verschiebung der Daten in die $xz-$Ebene, analog zum zweiten Modell: Die Bewegung der blauen Kugel entspricht der Bewegung des Rotors, und die Bewegung der roten Kugel entspricht der Bewegung des Stators (siehe *Playback mode* Fenster).

Kapitel 5

Schlussfolgerungen

Vergleicht man die Ergebnisse mit [13], so kann man bei den ersten beiden Modellen eine große Übereinstimmung finden. Es konnten öfters stabile Orbits gefunden werden, beziehungsweise nähern sich die beiden Modelle in der gewählten Simulationsumgebung noch besser aneinander an. Auch die Simulation des anisotrophen Verhaltens im ersten Modell scheint nun besser die Realität wiederzugeben. Beim dritten Modell konnte die Simulation der Fanglagermasse im hohen Bereich die Ergebnisse aus [13] nicht bestätigen.
Die Simulationen wurden auf einem handelsüblichen PC mit Intel© Core^{TM} i7 Vierkernprozessor und 12 GB Arbeitsspeicher durchgeführt. Leider konnte Maple nur einen der vier Kerne ansprechen, daher war die Rechenzeit höher als gedacht. Für die Berechnung der Erstellung der Bifurkationsdiagramme für das erste Modell, bei Durchlauf von 501 Parameterwerten, wurde ca. 2 Stunden gebraucht. Im zweiten Modell dauerte dies bereits ca. 50 Stunden, und im dritten Modell gar 265 Stunden. Allerdings wären hierbei auch Einsparungen möglich: Da auf die Rechengenauigkeit Wert gelegt wurde, startete der Integrationsvorgang nach dem Auffinden einer Nullstelle von vorn. Man könnte auch von der gefundenen Nullstelle weiterrechnen lassen, nur entstünde ein Rundungsfehler nach der 15. Stelle bei der Zeit, den $x-$ und $y-$Koordinaten und deren ersten Ableitungen. Da im gewählten Intervall ca. 60 Nullstellen auftreten kann dies zur Beeinflussung des Ergebnisses führen, vor allem bei instabilen Orbits. Bei sehr stabilen Orbits tritt keine Veränderung ein. Wenn auf diese Genauigkeit allerdings kein Wert gelegt wird, so lässt sich die Rechenzeit um den Faktor 60 verkürzen. Eine andere Möglichkeit wäre das Mitloggen der Daten der Orbits und das Auslesen der Nullstellen aus diesen Daten. Dadurch würde man Rechenzeit für die extra Berechnung der Nullstellen einsparen, auf Kosten der Genauigkeit der Nullstellen, und man hat das Risiko nicht alle Nullstellen im geählten Intervall zu finden, abhängig von der Zeitschrittweite der gespeicherten Punkte. Außerdem fallen bei dieser Methode große Datenmengen an, beziehungsweise braucht man ein weiteres Programm zum Auslesen der Daten. Möchte man die Genauigkeit weiter erhöhen, so müsste man auf mehr als 15 Stellen Genauigkeit rechnen lassen. Maple würde dann allerdings im *software mode* rechnen, ohne den vorgefertigten Routinen, und der Rechenaufwand würde schnell um den Faktor 100 steigen, was derzeit nur eine Aufgabe für Großrechenanlagen wäre. Auch bei der Wahl der Fehlertoleranzen kann man noch Rechenzeit einsparen. Aus den bereits erwähnten Gründen wurden sie allerdings so gewählt, dass das Programm gerade noch nicht wegen numerischen Singularitäten abbricht. Für die Simulation einzelner Orbits eignet sich auch MapleSim gut, die Berechnung ist auch etwas schneller als in Maple.

Literaturverzeichnis

[1] Brogliato B.: *Nonsmooth Mechanics.* Springer Verlag London, 2nd Edition, 1999.

[2] Childs D.: *Turbomachinery Rotordynamics.* John Wiley & Sons, 1993.

[3] Den X., Liebich R., Gasch R.: *Gekoppelte Biege und Torsionsschwingungen infolge von Anstreifvorgängen.* Schwingungen in rotierenden Maschinen V, S. 75-85, Referate der Tagung in Wien, 26.-28. Februar 2001, Vieweg-Verlag, 2001.

[4] Ecker H., Tauchner M.: *Experimentelle Untersuchung von Anstreifvorgängen mit einem Rotorprüfstand.* Schwingungen in rotierenden Maschinen VI, S.73-80, Referate der Tagung in Darmstadt, 26.-28. Februar 2003, Vieweg-Verlag, 2003.

[5] Ecker H.: *Nonlinear Stability Analysis of a Single Mass Rotor Contacting a Rigid Backup Bearing.* Proceedings of the Eurometh Colloquium, pp. 79-88, Proceedings of the Euromech Colloquium, 15.-18. September 1998, Springer Verlag, 1998.

[6] Edwards S., Lees A. W., Friswell M.I.: *The Influence of Torsion on Rotor/Stator Contact in Rotating Machinery.* Journal of Sound and Vibration, 225(4), pp. 767-778, 1999.

[7] Gacsh R., Nordmann R., Pfützner H.: *Rotordynamik.* Springer Verlag Berlin Heidelberg, 2. Auflage, 2002.

[8] Goldman P., Muszynska A.: *Chaotic Behavior of Rotor/Stator Systems with Rubs.* Journal of Engineering for Gas Turbines and Power, 116, pp. 692-699, 1994.

[9] Liebich R.: *Nichtlineare Schwingungen aus Rotor-Stator-Kontakt unter Berücksichtigung von thermischen Effekten.* Schwingungen in rotierenden Maschinen IV, S. 172-180, Referate der Tagung in Kassel, 4.-6. März 1997, Vieweg-Verlag, 1997.

[10] Lin F., Schoen M. P., Korde U. A.: *Numerical Investigation with Rub-related Vibration in Rotating Machinery.* Journal of Vibration and Control, 7, 833-848, 2001.

[11] Moser F.: *Stabilität und Verzweigungsverhalten eines nichtlinearen Rotor-Lager-Systems.* Dissertation TU Wien, Oktober 1993.

[12] Parkus H.: *Mechanik der festen Körper.* Springer Verlag, 2. Auflage, 1966, Nachdruck 1995.

[13] Popprath S.: *Numerische Untersuchung von selbsterregten Rotor-Stator-Systemen mit Bregrenzung durch ein starres oder elastisches Fanglager*, Diplomarbeit TU Wien, April 2003.

[14] Tauchner M.: *Auslegung und Bau eines Rotorprüfstandes zur Untersuchung von Anstreifvorgängen.* Diplomarbeit TU Wien, 2001.

[15] Tiwari M., Gupta K., Prakash O., *Dynamic Response of an Unbalanced Rotor Supported on Ball Bearings.* Journal of Sound and Vibration, 238(5), pp. 757-779, 2000.

[16] Tondl A.: *Some Problems of Rotor Dynamics.* Chapman & Hall, London 1965.

[17] Troger H., Steindl A.: *Nonlinear Stability and Bifurcation Theory.* Springer Verlag, 2001.

[18] Wegener G., Markert R.: *Influence of Contact and Impacts on the Dynamics of an Elastic Rotor with an Elastic Retainer Bearing.* Dynamics of Vibro-Impact Systems, pp. 89-98, Proceedings of the Euromech Colloquium, 15.-18. September 1998, Springer Verlag, 1998.

[19] Yamamoto T., Ishida Y.: *Linear and Nonlinear Rotordynamics.* John Wiley & Sons, 2001.