Physik am Komputer 12. Juni 2010

Übung 04

Michael Kopp

Aufgabe 1: Anfangswertproblem lösen: Getriebener Oszillator Das Problem ist in `pendel01.cpp` auf vier Arten gelöst:

- Euler,
- Euler-Chromer,
- Verlet,
- Velocity Verlet.

In Abb. 2 sieht man, dass die Verfahren bei schwachem Antrieb nährungsweise die selben Lösungen bringen, doch auch hier weicht das Verlet-Verfahren schon weit ab. In 3 sieht man, wie die Verfahren Verlet und Velocity Verlet schon ab $t \approx 5\,\text{sec}$ von den beiden anderen wegdriften. Weil Euler nicht symplektisch ist und Euler-Chromer nur in erster Ordnung symplektisch, nehme ich an, dass die Verlets hier ein falsches Bild liefern, weil das System nicht hamilton'sch ist und sich dies bei den höhrern Symplektischen Verfahren stärker auswirkt. Vgl. dazu auch Abb. 1, in dem Euler-Chromer und Verlet nocheinmal gesondert für eine schwache Anregung untersucht werden – hier sieht man, dass die Abweichung der beiden Verfahren absolut kleiner wird, jedoch deutlich ist.

Die Periodizität ist nach einer gewissen Einschwingdauer – soweit erkennbar – Ω.

Aufgabe 2: Dynamik einer Federkette Die von mir verwendetet Bewegungsgleichung basiert basiert auf folgender Überlegung. Seien x_1, x_2 und x_3 die Auslenkungen der Massen aus ihrern Ruhelagen und D_a die Federhärte der ersten Feder (zwischen Wand und x_1) und entspr. weiter D_b, D_c und D_d definiert, dann erfährt die erste Masse x_1 nach Hook die Kraft

$$F_1 = \underbrace{-D_a x_1}_{\text{Von Feder } D_a} + \underbrace{D_b x_2 - D_b x_1}_{\text{Von Feder } D_b} . \tag{1}$$

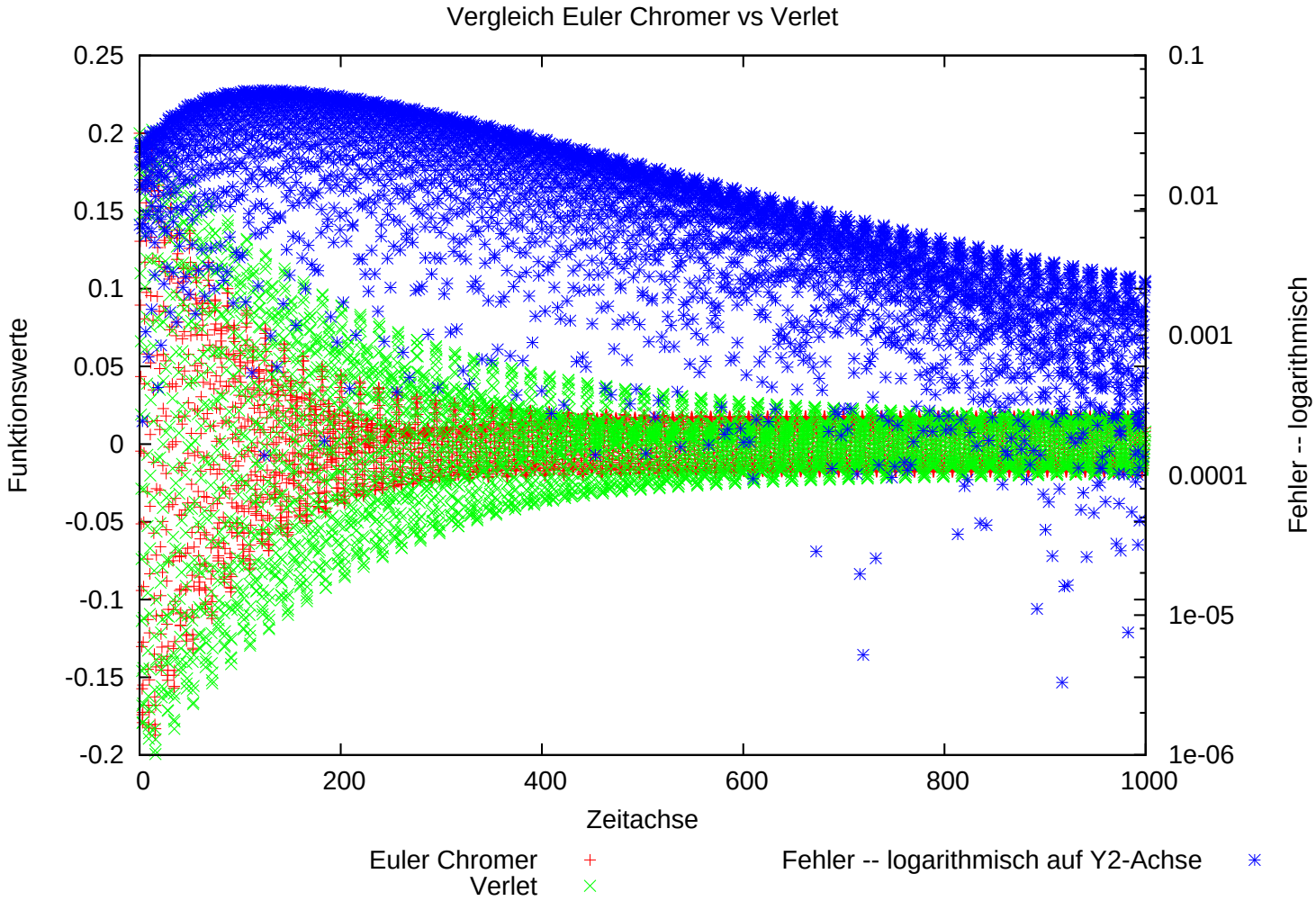


Abbildung 1: Vergleich von Euler und Verlet-Verfahren

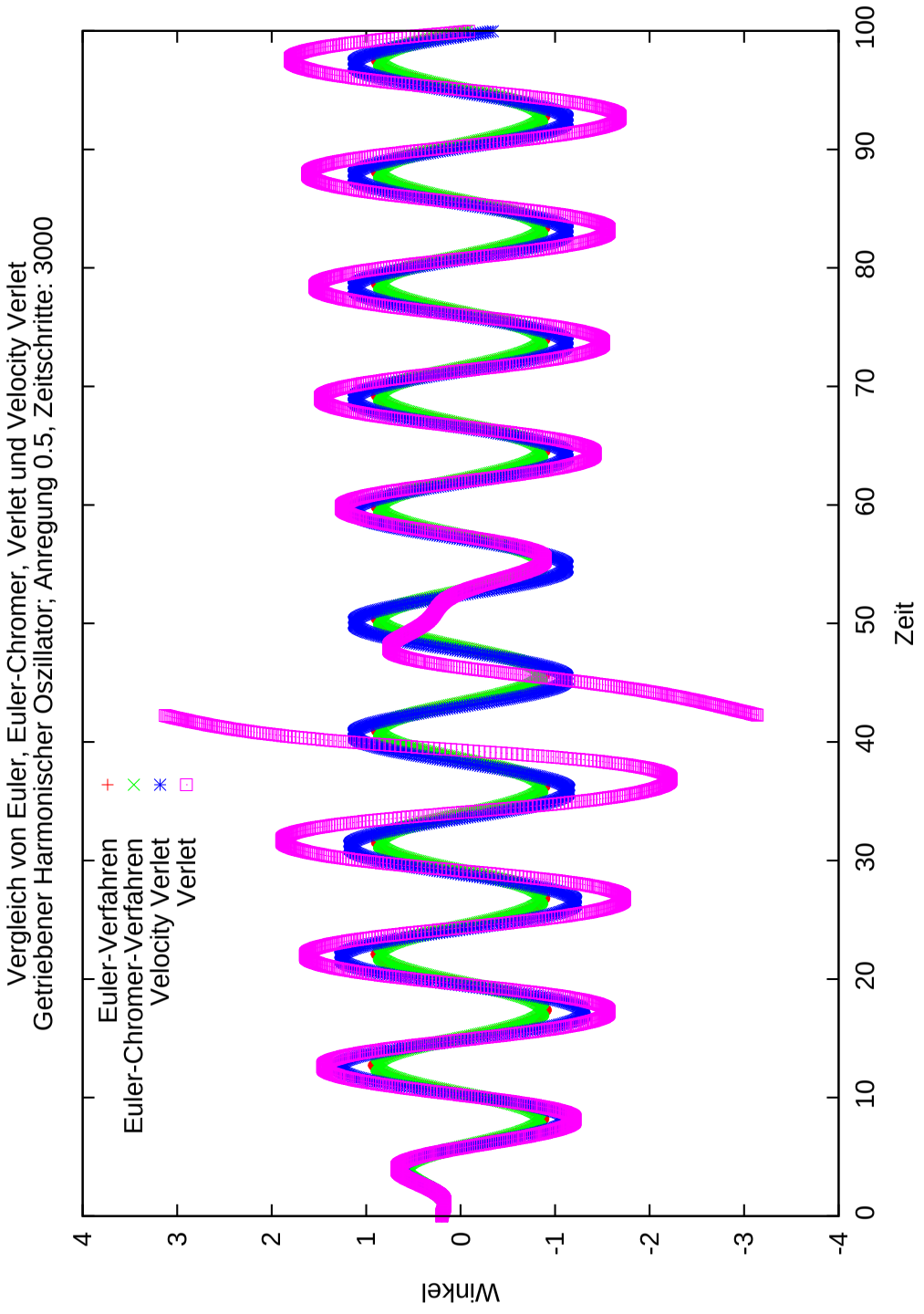


Abbildung 2: Vergleich der vier Verfahren für den getriebenen Oszillator bei schwachem Antrieb

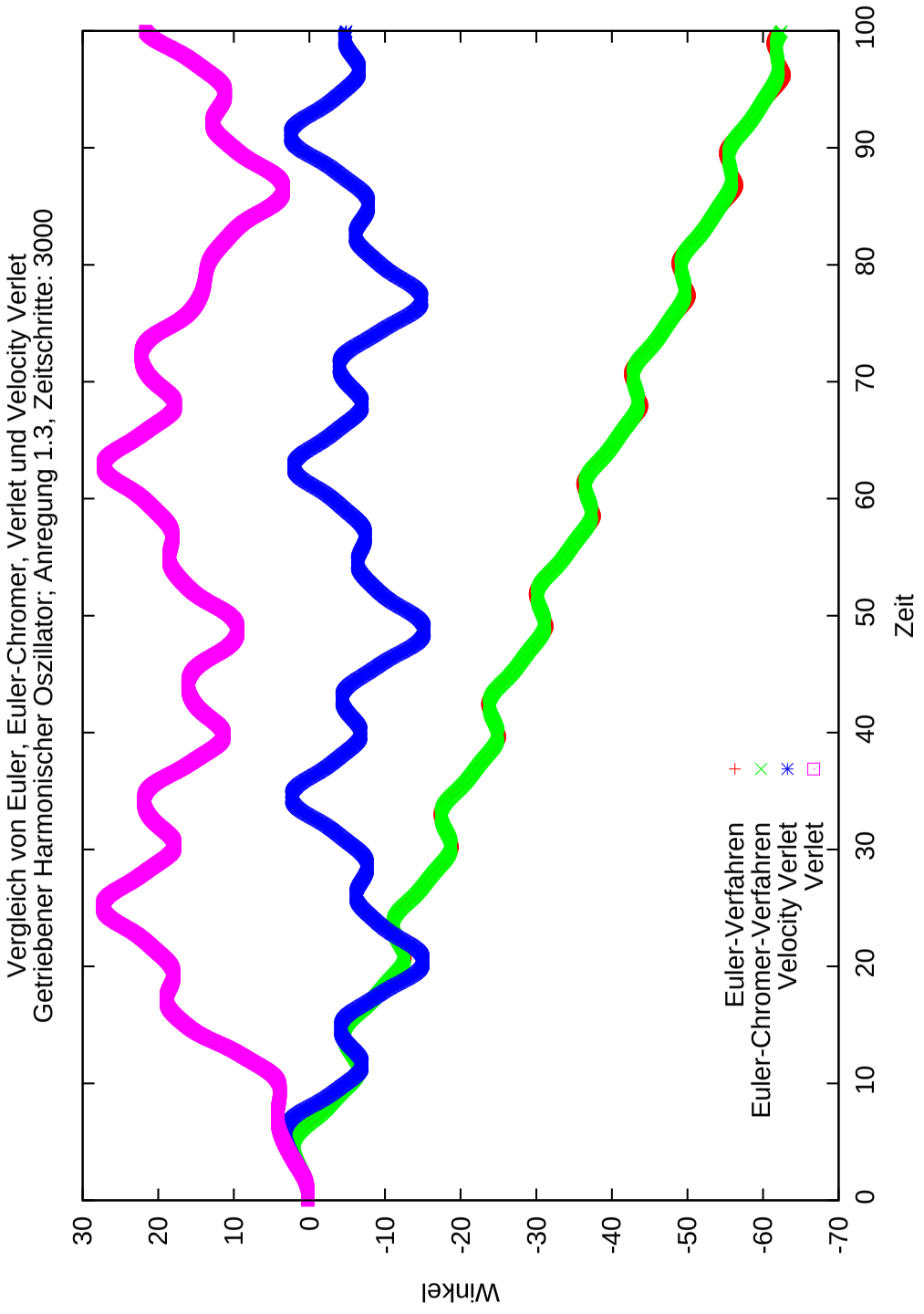


Abbildung 3: Vergleich der vier Verfahren für den getriebenen Oszillator bei starkem Antrieb

Berechnet man analog die Kräfte auf die zweite und dritte Masse so kann man dies zusammenfassen:

$$F_1 = m_1\ddot{x}_1 = -(D_b + D_a)x_1 + D_b x_2 \ , \tag{2}$$
$$F_2 = m_2\ddot{x}_2 = D_b x_1 - (D_b + D_c)x_2 + D_d x_3 \ , \tag{3}$$
$$F_3 = m_3\ddot{x}_3 = -(D_c + D_d)x_3 + D_c x_2 \ . \tag{4}$$

Um die Bewegung elegant zu notieren, verwende ich den Vektor

$$\boldsymbol{\xi} = (x_1, \dot{x}_1, x_2, \dot{x}_2, x_3, \dot{x}_3)^T \ , \tag{5}$$

dessen Ableitung

$$\dot{\boldsymbol{\xi}} = (\dot{x}_1, \frac{F_1}{m_1}, \dot{x}_2, \frac{F_2}{m_2}, \dot{x}_3, \frac{F_3}{m_3})^T \tag{6}$$

Eine Funktion von $\dot{x}_i$ und F_i ist, wopbei man die F_i durch die x_i ausdrücken kann. Es ist also

$$\dot{\boldsymbol{\xi}} = f(\boldsymbol{\xi}) \ , \tag{7}$$

und diese DGL kann man mit Runge-Kutta integrieren.

Um den Vektor $\boldsymbol{\xi}$ auch im Programm zu verwenden, habe ich die Vektorklasse `Vector6D.h` angepasst. Die Implementierung ist in `federn01.cpp` und die Ausgabe mit den Anfangsbedingungen

$$\boldsymbol{\xi}_0 = (0., 0., 0., 0., 1.1/12., 0.) \tag{8}$$

ist für $t \in [0, 10]$ mit 500 Zeitschritten in Abb. 4 dargestellt.

Was auffällt ist, dass sich die Trajektorien überschneiden – das ist natürlich nicht sinnvoll, weil sich in Realität die Massen nicht durchdringen können, außerdem gilt das HOOK'sche Gesetz nicht unbedingt – es ist für realie Feder nur eine Näherung.

In Abb. 5 sind die selben Anfangsbedingungen für verschiedenen Anzahlen von Zeitschritten (N) aufgetragen. Hier sieht man, dass auch für große Zeitschritte ($10/N$; also kleine N) das Verfahren noch sehr genau ist.

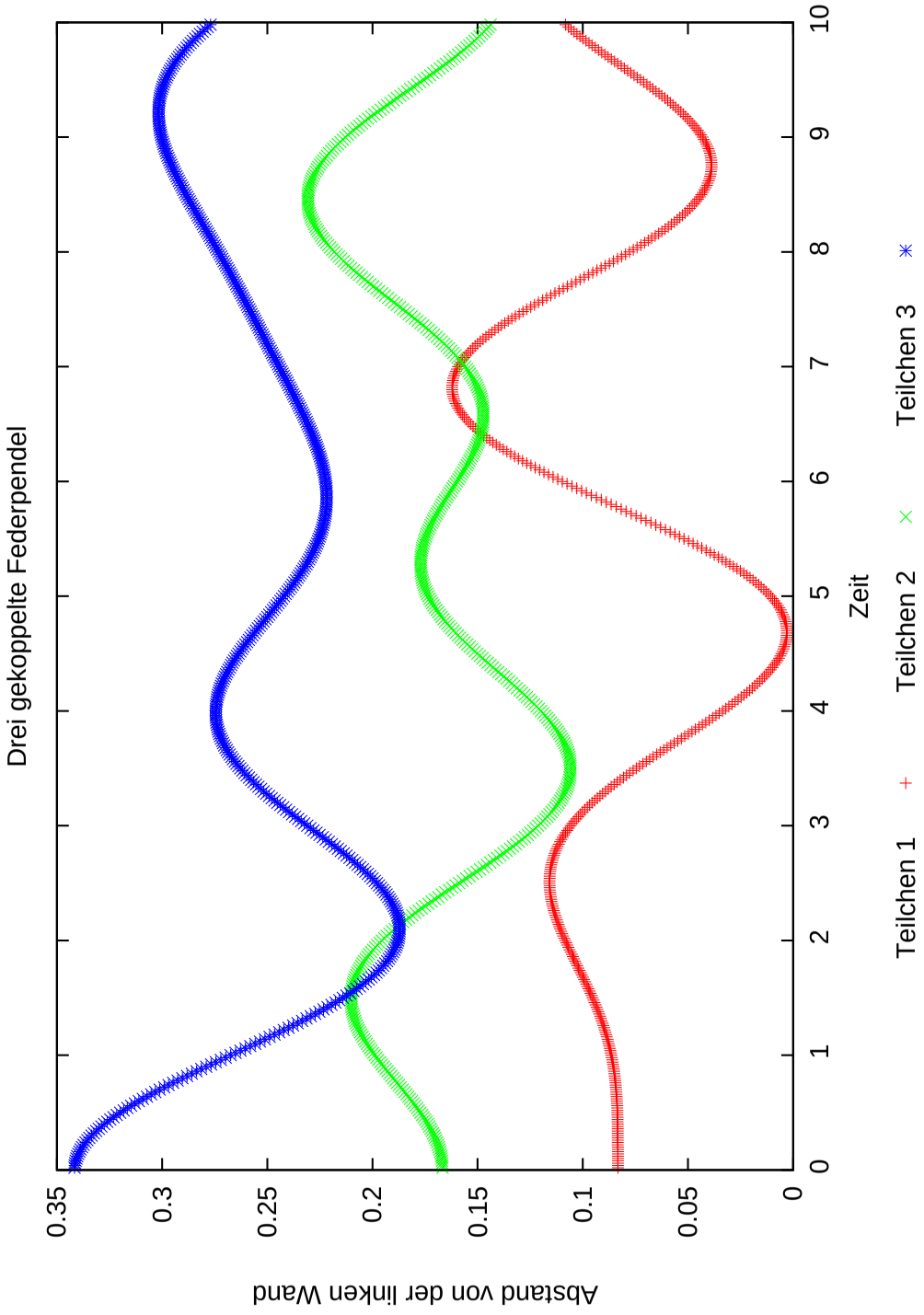
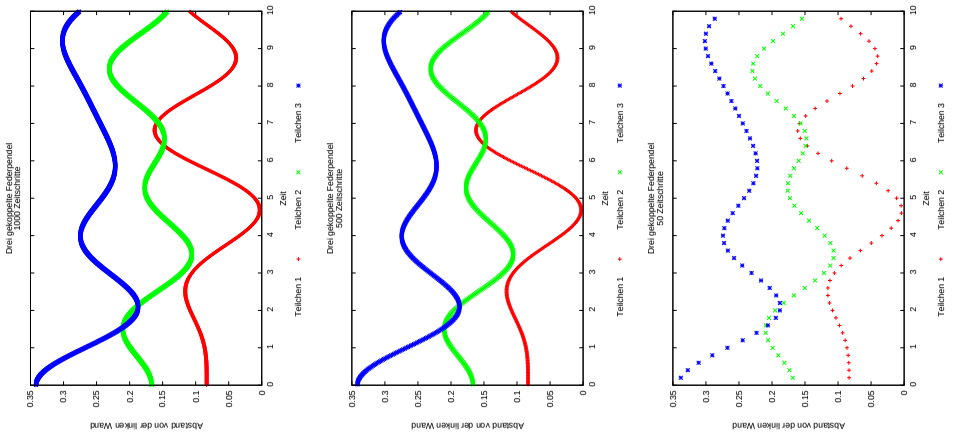


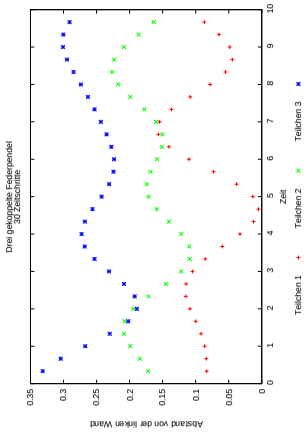
Abbildung 4: Drei durch Federn gekoppelte Massen, 500 Zeitschritte



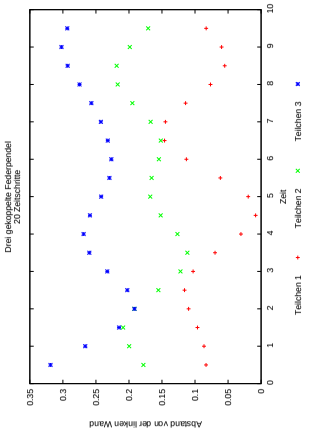
(a) $N = 1000$

(b) $N = 500$

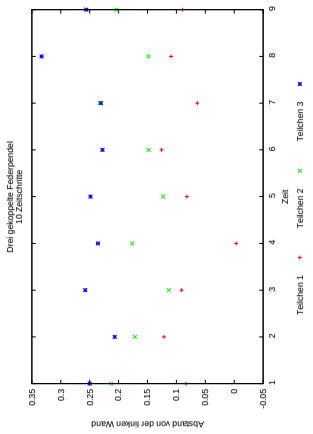
(c) $N = 50$



(d) $N = 30$



(e) $N = 20$



(f) $N = 10$

Abbildung 5: Gekoppelte Pendel mit verschieden großen Zeitschritten