

Best view:
Zoom=200%

ZAMM · Z. angew. Math. Mech. 74 (1994) 1, 69–71

Akademie Verlag

HOFFMANN, G.

Zeitsynchrone sprunghfreie Bahngeneratoren

MSC (1980): 70B15, 70-04, 65C20, 68J10, 93C15

1. Einleitung

Der Verfasser beschäftigt sich mit der Entwicklung rechnergesteuerter Marionetten, die nach Musik „tanzen“ sollen. Beispielsweise bewegt sich die gesamte Marionette mit gegebener Umlaufzeit und bekanntem Radius auf einer Kreisbahn. Beide Parameter werden aus der Rhythmusinformation der aufgenommenen Klänge ermittelt. Für die Erzeugung der Kreisbahn ist offensichtlich ein Sinus-Kosinus-Generator nötig, dessen Frequenz und Amplitude aber nicht fest sind, sondern so geändert werden können, daß der Generator die neuen Werte sprunghfrei annimmt. Außerdem soll die relative Phase der beiden Signale gleichermaßen sprunghfrei verstellbar sein.

Die Generatorsignale müssen in Echtzeit erzeugt werden. Auf einem Personalcomputer mit dem Betriebssystem MS-DOS ergibt sich das Problem, daß die eingebaute Uhr nur sehr grob abgefragt werden kann. Daher muß vorab eine Zeitvariable erzeugt werden, die feiner gestuft ist, aber im Mittel synchron mit der Uhr läuft.

2. Zeitsynchronisation

In Bild 1 sieht man den Verlauf der Rechneruhr „Clockzeit t_c “ über dem Schleifenzähler eines Programms aufgetragen, dessen Schleifenzeit dt in der Größenordnung 4 ms liegt und konstant ist. Die Clockzeit liefert im Mittel alle 55 ms eine neue Information (18,2 Hz), aber die Inkremente sind mit 40, 50 oder 60 ms nicht konstant. Dann sieht man im Bild eine Variable „Programmzeit t_p “, die mit möglichst guter Anpassung der mittleren Clockzeit folgt. Dies ist im Rechner der Ersatz für die physikalische Zeit t , die sonst nicht verfügbar ist. Die Programmzeit wird mit Hilfe eines programmierten Abtastreglers mit der Clockzeit synchronisiert. Schließlich ist im Bild als Anwendungsbeispiel eine mit t_p gebildete Sinusfunktion dargestellt.

Wären die Variablen kontinuierlich, so nähme man einen Proportional-Integral-Regler mit der folgenden Aufschaltung:

$$\dot{t}_p = 1 + c_p(t_c - t_p) + c_i \int_0^t (t_c - t_p) dt. \quad (1)$$

Einmal differenziert erhält man eine lineare Differentialgleichung 2. Ordnung:

$$\ddot{t}_p + c_p \dot{t}_p + c_i t_p = c_p \dot{t}_c + c_i t_c. \quad (2)$$

Offensichtlich ist $c_p = 2D\omega_i$ und $c_i = \omega_i^2$ mit bis jetzt beliebiger Kreisfrequenz ω_i und Dämpfung D .

Der Dämpfungsgrad darf nicht zu groß gemacht werden, weil sonst der Differentialanteil auf der rechten Seite (die differenzierten Sprünge der Clockzeit) die Regelgüte verschlechtert.

Dieses Verfahren kann ohne Änderung als programmierter Abtastregler installiert werden und lautet dann, bei Integration mit

der Eulerformel:

$$dv = t_c - t_p, \quad (3)$$

$$t_i = t_i + dt dv, \quad (4)$$

$$t_p = t_p + dt + dt(c_p dv + c_i t_i). \quad (5)$$

Hierbei ist t_i die Zustandsvariable für den Integralanteil. Um stets synchron zu starten, muß zu Beginn der Rechnung in einer schnellen Schleife die nächste Transition der Rechneruhr erfaßt werden. Die Clockzeit t_c ergibt sich dann durch Subtraktion der absoluten Uhrzeit und Verschieben um die halbe mittlere Clockperiode (27,5 ms), wie in Bild 1.

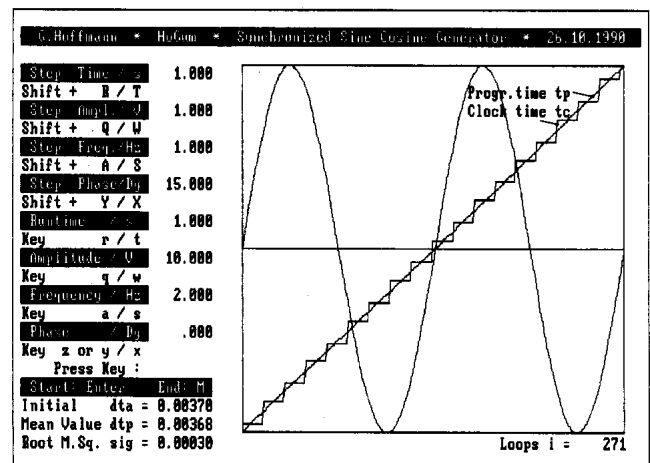


Bild 1. Clockzeit, Programmzeit und Generatorsignal über dem Schleifenindex

Das Zeitinkrement dt ist beim ersten Start des Programms ein Schätzwert. Nach diesem Lauf wird das tatsächlich benötigte dt und dessen Standardabweichung infolge der Reglerkorrektur berechnet. Im folgenden Lauf wird das so ermittelte dt eingesetzt. Dieses Verfahren konvergiert extrem schnell. Wenn das Zeitinkrement um den Faktor Zehn zu groß oder zu klein angenommen wurde, sind allenfalls drei Läufe von einer Sekunde Dauer nötig, in allen praktischen Fällen aber nur einer. Die Messung ist außerdem viel genauer als mit der Rechneruhr, weil diese die mittlere Meßunschärfe von 55 ms besitzt.

Für die Regleraufschaltfaktoren c_p und c_i wurden $\omega_i = 8$ und $D = 0,3$ experimentell optimiert. Mit zunehmendem Dämpfungsgrad wächst die Standardabweichung des über die Laufzeit gemittelten Zeitinkrements. Die genannten Werte garantieren eine Standardabweichung von 0,1 dt .

Nun kann man die Programmzeit t_p anstelle der nicht greifbaren physikalischen Zeit im Programm verwenden. Beispielsweise wird ein Signalgenerator programmiert. Den zusätzlichen Zeitaufwand pro Schleife stellt man schon nach dem ersten Lauf mit guter Genauigkeit fest.

Das Programm verschwendet keine Zeit mit Warteschleifen für Zeitgeber-Interrupts, sondern rechnet mit maximalem Durchsatz. Wenn nun während des Laufs ein Parameter geändert wird, zum Beispiel die Amplitude des Generators, so muß für die Bearbeitung der Tastaturbedienung eine kleine Zeitspanne geopfert werden. Danach wird die Programmzeit wieder mit der Clockzeit synchronisiert, wobei der Generator für kurze Zeit etwas schneller läuft. Etwas später hat das Signal nicht nur im Mittel die quartzgenaue Frequenz, sondern sogar die korrekte Phasenlage in Bezug auf den Zeitnullpunkt beim Start. Aufgrund dieser ungewöhnlichen Eigenschaft erhält die Programmzeit auch die Bezeichnung „Gummizeit“. Die rekursive Korrektur des Zeitinkrements dt für den folgenden Programmlauf, wie oben beschrieben, wird ausgesetzt, wenn während des Programmablaufs Tastatur-Bedienungen aufgetreten sind.

Zusammenfassend kann gesagt werden, daß die hier beschriebene Zeitsynchronisation auf jedem Personalcomputer PC AT mit Uhr ohne Zusatzkarten und ohne Zeitgeber-Interrupt unter ausschließlicher Verwendung von Hochsprachenbefehlen einsetzbar ist.

3. Synchronisierte Generatoren

In (1) und (2) werden Laufmustergeneratoren für die Signale zur Ansteuerung von Robotern mit Beinen beschrieben. Man erkennt die Ähnlichkeit mit dem Marionettenproblem. Als Signalquelle dient eine Differentialgleichung von Van der Pol. Die verwendete Differentialgleichung hat den Nachteil, daß die stabilisierte Schwingung nur näherungsweise harmonisch wird, wenn die Amplitudenregelung hart ist. Besser wäre eine zweite Form der Gleichung von Van der Pol, siehe (3), die in jedem Fall einen harmonischen Grenzyklus liefert. Beide Formen sind noch verbesserungsbedürftig, weil kein Gebrauch von der inneren Symmetrie des Generators gemacht wird. Zunächst soll aber gezeigt werden, wie der Signalgenerator unter Verwendung der Programmzeit (Gummizeit) direkt zu programmieren ist.

3.1 Sprungfreier parametervariabler Generator mit Tabellen

Die Aufgabe besteht darin, einen Sinus-Kosinus-Generator mit vorgegebener Frequenz und Amplitude sowie veränderlicher relativer Phasenlage des „Kosinus“-Signals zu programmieren, wobei während des Laufs alle Parameter geändert werden können und vom Generator sprunfrei angenommen werden müssen.

Die Winkelfunktionen kann man direkt berechnen, und die stufenlosen Übergänge der Parameter werden über Tiefpässe erster Ordnung erzeugt:

$$y_s = a \sin(\omega t_p + p), \quad (6)$$

$$y_c = a \sin(\omega t_p + p + c), \quad (7)$$

$$\dot{a} = \omega_r(a_k - a), \quad (8)$$

$$\dot{c} = \omega_r(c_k - c), \quad (9)$$

$$\dot{\omega} = \omega_r(\omega_k - \omega). \quad (10)$$

Die Parameter mit dem Index k sind die kommandierte Amplitude, Phase und Frequenz, und ω_r ist die Kreisfrequenz über die Dynamik der Übergänge. Weiterhin ist p die Startphase beider Signale und c die relative Phase.

Bei der Erprobung zeigt sich, daß schon mäßige Frequenzänderungen zu erheblichen Phasensprüngen führen. Man müßte für die Frequenzübergänge einen Tiefpaß höherer Ordnung oder eine sehr große Zeitkonstante verwenden, beides keine guten Lösungen. Besser ist es, die Frequenzänderungen sprunghaft zuzulassen, die Signale aber sprunfrei ineinander zu überführen.

Weiterhin kann man Zeit sparen durch Abruf der Winkelfunktionen aus einer Sinustabelle $s(i)$ mit $i = 0$ bis $i = n$. Anstelle des Sinus darf auch die erste Periode einer ganz beliebigen Funktion gespeichert werden.

Zunächst wird der Tabellenindex über die Modulo-Funktion berechnet, das ist hier nichts anderes als die Nachkommazahl des ersten Operanden. Der Index wird automatisch gerundet. Weiterhin ist p die Startphase, ausgedrückt als Index zwischen 0 und n :

$$is = n \bmod (t_p f_k, 1) + p, \quad (11)$$

$$\left. \begin{array}{l} \text{if } is > n \text{ then } is = is - n \\ \text{if } is < 0 \text{ then } is = is + n \end{array} \right\} \quad (12)$$

Dann wird der Sinus aufgerufen

$$ys = as(is) \quad (13)$$

und der phasenvariable „Kosinus“ berechnet, z. B. mit $c = n/4$ für den echten Kosinus:

$$ic = is + c, \quad (14)$$

$$\left. \begin{array}{l} \text{if } ic > n \text{ then } ic = ic - n \\ \text{if } ic < 0 \text{ then } ic = ic + n \end{array} \right\}, \quad (15)$$

$$yc = as(ic). \quad (16)$$

Bei Frequenzänderungen $f_k = f_k + df$ über die Tasten wird ein sprunghafter Übergang in der Tabelle geschaffen:

$$\text{if key input } f_k \text{ then } p = is - n \bmod (t_p f_k, 1). \quad (17)$$

Schließlich werden die Kommandowerte für die Amplitude und die Phase wie oben gefiltert, hier mittels Euler-Integration:

$$a = a + dt \omega_r(a_k - a), \quad (18)$$

$$c = c + dt \omega_r(c_k - c). \quad (19)$$

Danach wiederholt sich die Schleife.

In Bild 2 sieht man den Sinus-Ausgang eines derartigen Generators, wobei während des Laufs die Amplitude und die Frequenz verändert wurden.

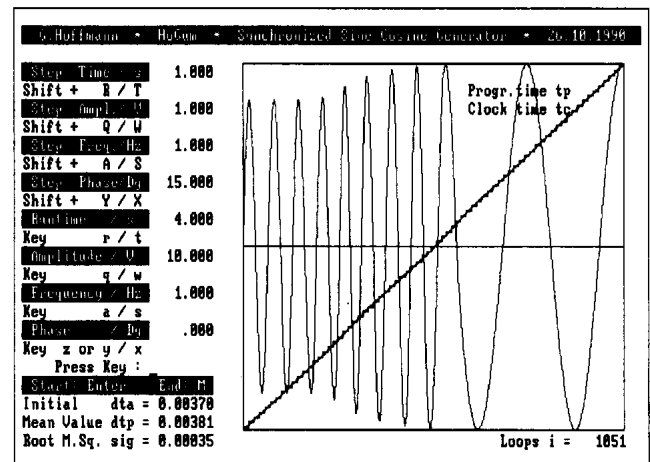


Bild 2. Ändern der Amplitude und Frequenz beim Generator mit Tabellen

3.2 Sprungfreier parametervariabler Generator mit Baumgarte-Stabilisierung

Der Generator wird durch die numerische Integration einer Schwingungsdifferentialgleichung gebildet. Aufgrund der äußerst groben Euler-Integration entstehen große Schrittfelder, auch stimmt die Frequenz nicht, obwohl die quarsynchrone Programmzeit verfügbar ist. Die Differentialgleichung muß stabilisiert werden. Dies ist generell auch beim Einsatz zeitaufwendiger Integrationsalgorithmen höherer Ordnung (Heun, Runge-Kutta) nötig.

Der Grundgedanke des folgenden Verfahrens stammt von BAUMGARTE [4]. Weiterentwicklungen findet man bei OSTERMEYER [5], [6].

Der Gleichungssatz lautet:

$$\omega = 2\pi f_k, \quad (20)$$

$$z_1 = (1/a_k) \sqrt{y_1^2 + y_2^2} - 1, \quad (21)$$

$$\dot{y}_1 = \omega y_2 - (k_1 z_1 + k_2 y_3) y_1 - k_3 (y_1 - a_k \sin(\omega t_p)), \quad (22)$$

$$\dot{y}_2 = -\omega y_1 - (k_1 z_1 + k_2 y_3) y_2 - k_3 (y_2 - a_k \cos(\omega t_p)), \quad (23)$$

$$\dot{y}_3 = z_1. \quad (24)$$

Man erkennt die Oszillorgleichung mit der Kreisfrequenz ω . Mit den Anfangswerten $y_1(0) = 0$ und $y_2(0) = a_k$ hätte man eine Sinusschwingung für y_1 und eine Kosinusschwingung für y_2 . Sodann wird die Amplitude stabilisiert, und zwar, anders als in den Differentialgleichungen von Van der Pol, völlig symmetrisch im Zustandsraum. z_1 ist der normierte Amplitudenfehler. Mit y_3 wird zusätzlich dessen Integral gebildet. Der Amplitudenregler enthält die Proportionalaufschaltung k_1 und die Integralaufschaltung k_2 , wobei aber der Regelungsterm durch Multiplikation mit y_1 bzw. y_2 auf die entsprechenden Achsen projiziert wird. Dieser Oszillator schwingt dann hochgenau in der Amplitude, während die Phase und die Frequenz aufgrund der numerischen Integration nicht stabil sind. Es wird nun angenommen, daß zwei harmonische Referenzsignale zur Verfügung stehen, die entweder wie in den Gleichungen mittels der Programmzeit erzeugt werden oder von einem anderen Generator stammen. Dabei muß aber der Zahlenwert für ω explizit bekannt sein. Mit diesen substantiell gleichwertigen Phasenreferenzen wird über k_3 die Phase des Oszillators stabilisiert und synchronisiert.

Bei Anwesenheit kleiner Fehler δa in der vektoriellen Amplitude und $\delta\varphi$ im Phasenwinkel hat man für y_1 diese Lösung:

$$y_1 = (a_k + \delta a) \sin(\omega t_p + \delta\varphi). \quad (25)$$

Dann lauten die linearen Variationsgleichungen für die Amplitude und den Phasenwinkel:

$$\delta\ddot{a} + k_1 \delta\dot{a} + k_2 \delta a = 0, \quad (26)$$

$$\delta\ddot{\varphi} + k_3 \delta\dot{\varphi} = 0. \quad (27)$$

- 6 OSTERMEYER, G.-P.: Baumgarte stabilization for differential algebraic equations. In DEYO, R.; HAUG, E. (eds.): NATO Advanced Workshop on real time integration methods for mechanical system simulation. Springer, Heidelberg, to appear 1990.
7 True Basic Inc., 12 Commerce Avenue, West Lebanon, N.H. 03784, U.S.A.

Eingegangen am 14. 11. 1990

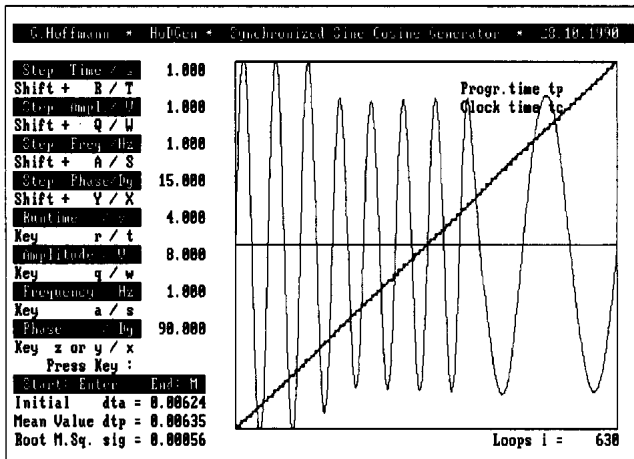


Bild 3. Ändern der Amplitude und Frequenz beim Generator mit Baumgarte-Stabilisierung

Hierin kommt die Oszillatorfrequenz nicht mehr vor, die Dynamik der Regelung ist völlig frei wählbar. Dies und die Darstellung im Zustandsraum ist der wesentliche Kern der Methode nach BAUMGARTE. BAUMGARTE ging aus von den Lagrange-Gleichungen erster Art, und er betrachtete die Regelungsterme als Zwangskräfte zur Rückführung auf eine Referenztrajektorie.

Natürlich setzt man im Amplitudenregler $k_1 = 2D\omega_r$ und $k_2 = \omega_r^2$. Der Phasenregler kann mit $k_3 = 1/\tau$ auf eine geeignete Zeitkonstante eingestellt werden.

Die Regelung ist auch im Großen stabil. In Bild 3 wird die Amplitude und die Frequenz beim laufenden Generator geändert. Man beachte, daß am Ende der Laufzeit die Phase sehr genau stimmt. Die Fehler der numerischen Integration sind kompensiert, und die Programmzeit sorgt für den synchronisierten Zeitbezug. Es soll noch erwähnt werden, daß der Phasenregler hier keinen Integralanteil hat, bei Bedarf wird er sich leicht ergänzen lassen.

4. Programme

Die Rechenprogramme HoGum und HoDGen können im Quellcode vom Verfasser bezogen werden. Darin ist auch ein Assemblerprogramm für die Bedienung der Digital-Analog-Umsetzkarte ME 66 (Meilhaus) für PC AT enthalten.

Die Programme sind in **True Basic** geschrieben, einer strukturierten Sprache, die generell kompiliert arbeitet. Die Programme lassen sich „leicht“ in jede andere Hochsprache umschreiben, sofern die Uhr abgefragt werden kann. Gegenwärtig kann man **True Basic** nur beim Hersteller [7] beziehen.

Literatur

- 1 FRIK, M.; AHANIKAMANGAR, M.: Entwurf eines nichtlinearen Laufmustergenerators. ZAMM 70 (1990) 7, 279–284.
- 2 AHANIKAMANGAR, M.; FRIK, M.: Realisierung periodischer Bewegungen von Gehmaschinen mit Hilfe gekoppelter Van-der-Pol-Gleichungen. ZAMM 70 (1990) 4, T 146–T 148.
- 3 GILOI, W.; LAUBER, R.: Analogrechnen. Springer-Verlag, Berlin–Göttingen–Heidelberg 1963.
- 4 BAUMGARTE, G.: Stabilization of constraints and integrals of motion of dynamical systems. Computat. Math. Appl. Mech. Eng. 1 (1972), 1–16.
- 5 OSTERMEYER, G.-P.: Die Stabilisierung von Bindungen und ersten Integralen als Regelungsproblem und ihre Konsequenzen. ZAMM 65 (1985), 185–189.