NUMERISK LÖSNING AV OPTIMALA STYRNINGSPROBLEM MED BRYSON - KELLEY METODIK

IVAR GUSTAVSSON

Rapport RE - 7 nov 1966

NUMERISK LÖSNING AV OPTIMALA STYRNINGSPROBLEM MED BRYSON-KELLEY METODIK

av

Ivar Gustavsson

Inledning

Det problem, som skall behandlas här, ligger inom det område av regleringstekniken, som kallas optimeringsteori. Detta är ett mycket vidsträckt område med stor praktisk betydelse. Det gäller ju ofta att maximera eller minimera en viss storhet, och detta oftast då samtidigt som vi måste införa begränsningar av olika slag för andra storheter.

Optimala styrningsproblem uppkommer i samband med processer, där det gäller att bestämma en eller flera styrvariabler, så att vissa slutvillkor blir uppfyllda. Problemet är alltså att bland alla möjliga styrprogram välja ut det, som minimerar (eller maximerar) en viss storhet, medan samtidigt andra storheter antar vissa givna värden vid slutpunkten. Vi vill t. ex. minimera tiden för ett projekt, då vi har begränsad tillgång på kapital och arbetskraft.

Klassiskt användes variationskalkyl för lösning av sådana problem. Men endast relativt enkla problem kunde lösas på detta sätt. Även med snabba datamaskiner står man inför en besvärlig uppgift, då dessa problem i regel i sin klassiska formulering utgör ett randvärdesproblem för olinеära, ordinära differentialekvationer, där en del randvärden är givna som initialvärden och en del som slutvärden ("two-point boundary problem"). För numerisk lösning krävs att man gör en god gissning av de initialvärden, som saknas, integrerar differentialekvationerna numeriskt till sluttidpunkten, jämför de erhållna slutvillkoren med de givna och sedan försöker

göra en bättre gissning av de saknade initialvärdena. Detta måste upprepas tills alla slutvillkor är uppfyllda. Denna metod är besvärlig och fungerar i vissa fall inte alls. Den är t.ex. känslig även för små ändringar i begynnelsevillkoren.

Andra metoder att komma till rätta med dessa svårigheter har under senare år framkommit. Det finns i huvudsak två olika metoder, dynamisk programmering samt de metoder, som bygger på Pontryagins maximiprincip och går ut på att med successiva approximationer nå den optimala lösningen. Pontryagins maximiprincip ger dessutom möjlighet att direkt införa begränsningar på styrvariablerna, något som inte är möjligt vid klassisk lösning med variationskalkyl. Bland metoderna med successiva approximationer märks speciellt två, dels den där iteration sker på randvillkoren, och dels den där iteration sker i rummet av styrsignaler. Vi skall här begränsa oss till den sistnämnda, den s.k. Bryson-Kelley-metoden. Redan nu kan sägas att den i huvudsak bygger på att vi gissar en styrsignal och sedan successivt förbättrar denna styrsignal, så att vi hela tiden närmar oss den optimala lösningen.

Formulering av ett allmänt optimeringsproblem

Innan vi närmare undersöker Bryson-Kelley-metoden formulerar vi ett allmänt optimeringsproblem.

Bestäm en styrsignal $u(t)$ i intervallet $t_0 \leq t \leq T$, så att förlustfunktionen

$$V = \phi(x(T),T) + \int_{t_0}^{T} L(x(t), u(t), t)\, dt \qquad (1)$$

minimeras (eller maximeras) under bivillkoren

$$\frac{dx_i}{dt} = f_i(x(t),u(t),t) \qquad (2)$$

$x = (x_1, x_2, \ldots, x_n)$ (tillståndsvariablerna)

$u = (u_1, u_2, \ldots, u_m)$ (styrvariablerna)

$x_i(t_0)$ och t_0 givna (3)

$\psi_j(x(T),T) = 0 \qquad j = 1,2,\ldots, q;\ q \leq n \qquad (4)$

T bestämmes av $\Omega(x(T),T) = 0 \qquad (5)$

I variationskalkylen kallas ett sådant problem för Bolzas problem. Specialfall uppkommer för $\phi = 0$ (Lagranges problem) och för $L = 0$ (Mayers problem). Dessa tre problem är emellertid ekvivalenta och kan relativt lätt transformeras från den ena formen till den andra.

Förlustfunktionen V är en funktion ϕ av sluttillståndet $x(T)$ och sluttidpunkten T plus en integral av en funktion L av x, u och t.

ϕ kan sägas representera förluster, som uppkommer, då vi ej når målet, och L representerar då förluster längs vägen till slutpunkten.

Bivillkoren (2) består av n stycken första ordningens olinеära, ordinära differentialekvationer. Vi antar initialvärdena (3) givna. Slutvillkoren är givna enligt (4) och (5). Sluttiden, T, kan vara en del av förlustfunktionen (ϕ = T) eller given implicit av (5), dvs antingen fixt T eller givet som t.ex. $x_k(T)=0$.

Ett sätt att lösa Bolzas problem är genom införandet av Lagrange-multiplikatorer (även kallade adjungerade funktioner).

Vi definierar då en funktion H enligt följande

$$H(x(t),u(t),\lambda(t),t) = L(x(t),u(t),t) + \sum_{i}^{n} \lambda_i f_i(x(t),u(t),t) \quad (6)$$

$\lambda_i(t)$ är tills vidare n stycken obestämda funktioner.

Vi definierar även en funktion

$$\Phi(x(T),T,\ ) = \phi(x(T),T) + \sum_{i}^{q} \nu_i \psi_i(x(T),T) \quad (7)$$

ν_i (q stycken) är tills vidare obestämda.

Vi väljer sedan $\lambda_i(t)$ så att de satistierar

$$\frac{d\lambda_i}{dt} = -\frac{\delta H}{\delta x_i} \qquad i = 1, 2, \ldots, n \quad (8)$$

$$\lambda_i(T) = \frac{\delta \Phi}{\delta x_i} \quad (9)$$

I så fall kan man visa att infinitesimala ändringar av styrvariablerna, $\delta u_i(t)$, infinitesimala ändringar i initialvillkoren, $dx_i(t_0)$, i slutvillkoren, $d\psi_i$, i begynnelsetiden, dt_0, samt i sluttiden, dT, ger följande infinitesimala ändring av förlustfunktionen

$$dV = \int_{t_0}^{T} \frac{\delta H}{\delta u_i} \delta u_i(t)\, dt + \lambda_i(t_0)\, dx_i(t_0) - \nu_i\, d\psi_i - H(t_0)\, dt_0 + \left(\frac{\delta \Phi}{\delta T} + H(T) \right) dT \quad (10)$$

En optimal lösning är nu en lösning, som satisfierar (2) - (5) och som dessutom är sådan att en liten ändring i styrvariablerna, $\delta u_i(t)$, inte ger någon ändring i förlustfunktionen och så att, om T är given implicit av (4), en liten ändring av sluttiden, dT, inte ger någon ändring av förlustfunktionen, dV.
Detta betyder enligt (10) och tidigare ekvationer att vi sammanfattningsvis har följande nödvändiga villkor för den optimala lösningen:

$$\frac{dx_i}{dt} = f_i(x,u,t) \qquad i = 1,2,\ldots,n \tag{11}$$

$$\frac{d\lambda_i}{dt} = -\frac{\partial H}{\partial x_i} = -\lambda_j \frac{\partial f_j}{\partial x_i} - \frac{\partial L}{\partial x_i} \tag{12}$$

$$\frac{\partial H}{\partial u_i} = \lambda_j \frac{\partial f_j}{\partial u_i} + \frac{\partial L}{\partial u_i} = 0 \qquad i = 1,2,\ldots,m \tag{13}$$

$x_i(t_0)$ givna (14)

$$\lambda_i(T) = \frac{\partial \phi}{\partial x_i} + \nu_j \frac{\partial \psi_j}{\partial x_i} \tag{15}$$

$$H(T) = -\frac{\partial \phi}{T} - \nu_j \frac{\partial \psi_j}{\partial T} \tag{16}$$

$\psi_j(T)$ givna (17)

(11) utgöres av n stycken differentialekvationer, systemekvationerna, liksom (12), som omfattar de n stycken adjungerade ekvationerna. (13) erhålles ur (10) som nödvändigt villkor för optimal lösning, och dessa ekvationer ger $(u_1,\ldots,u_m)$. (14) omfattar de givna initialvillkoren (n stycken) för systemekvationerna medan (15) ger slutvillkoren för de adjungerade ekvationerna. Ekvation (16) bestämmer T.(17) består av q stycken slut-

villkor, som bestämmer $(\nu_1,\ldots,\nu_q)$.

Ekvationerna (12) och (13) är de från variationskalkylen kända sk. Euler-Lagrange ekvationerna. Ekvationerna (11) - (17) utgör tillsammans ett "two-point boundary problem", och ett sådant problem är, som nämndes i inledningen komplicerat att lösa, även om man har en snabb datamaskin till förfogande.

Vi skall också kort se på det fall då styrvariablerna måste väljas begränsad, dvs. $|u_i(t)| \leq U$. I stället för ekvation (10) får vi, om vi för enkelhets skull här sätter $dx_i(t_0) = 0$, $dt_0 = 0$, $d\nu_i = 0$ och $dT = 0$,

$$dV = \int_{t_0}^{T} \delta_u H(x(t),u(t), \dot{\lambda}(t),t)\, dt \qquad (18)$$

där $\delta_u H$ betyder ändringen av H, då u ändras, men då x, λ och t hålles fixa.

För att maximera V skall alltså u väljas inom sina begränsningar och så att H maximeras för varje tidpunkt t i intervallet $t_0 \leq t \leq T$.

Styrningsproblemet, då T är fixt

Teori

Vi skall se närmare på lösningen av ett något mindre generellt formulerat optimalt styrningsproblem.

Låt oss betrakta följande problem:

Bestäm $u(t)$ så att

$$V = \phi(x(T)) + \int_{t_0}^{T} L(x(t),u(t))\,dt \tag{19}$$

minimeras då

$$\dot{x} = \frac{dx}{dt} = f(x(t),u(t)) \tag{20}$$

$$x(t_0),\ t_0 \text{ och } T \text{ givna} \tag{21}$$

$$|u(t)| \leq U \tag{22}$$

(Här och i fortsättningen av detta avsnitt användes vektorbeteckningar för $x = (x_1,\ldots,x_n)$ respektive $\lambda = (\lambda_1,\ldots,\lambda_n)$. Med en punkt över en storhet avses tidsderivatan av storheten, t.ex. $\dot{x}_1 = \frac{dx_1}{dt}$. x^T betyder den transponerade vektorn. Med beteckningen L_x avses vektorn $\left(\frac{dL}{dx_1},\ldots,\frac{dL}{dx_n}\right)^T$.)

För att lösa problemet införes Lagrangemultiplikatorer $\lambda(t)$, så att hänsyn kan tas till bivillkoren (20).

Vi skall då i stället minimera

$$V(x,u,\lambda) = \phi(x(T)) + \int_{t_0}^{T} (L(x,u) + \lambda^T(f(x,u) - \dot{x}))\,dt \tag{23}$$

Euler-Lagranges ekvationer ger ett nödvändigt villkor för minimum:

$$\begin{cases} L_x(x,u) + \lambda^T f_x(x,u) + \dot{\lambda} = 0 \\ f(x,u) - \dot{x} = 0 \\ L_u(x,u) + \lambda^T f_u(x,u) = 0 \end{cases} \tag{24}$$

Som randvillkor för den första av ekvationerna fås

$$\phi_x - \lambda(T) = 0 \tag{25}$$

För den andra av ekvationerna, som ju är våra ursprungliga systemekvationer, gäller att $x(t_0)$ är given.
Ur den tredje ekvationen kan u lösas ut och sättas in i de andra två ekvationerna, och vi har fått ett "two-point boundary problem".
Vi kan också betrakta funktionen

$$H(x, u, \lambda) = L(x,u) + \lambda^T f(x,u) \tag{26}$$

och bestämma u(t) så att $H_u = 0$ (för $u = u^0(x,\lambda)$), dvs så att H stationärt med avseende på u. Detta är ju detsamma som att lösa ut u ur den tredje av ekvationerna (24). Vi minimerar med andra ord H med avseende på u, och får

$$H^0(x, \lambda) = H(x, \lambda, u^0(x, \lambda)) \tag{27}$$

Vidare erhålles

$$\begin{cases} \dot{x} = H_\lambda \\ \dot{\lambda} = -H_x \end{cases} \tag{28}$$

Villkoren (27) och (28) är enligt Pontryagins sats nödvändiga villkor för minimum. Ekvationerna (27) och (28) motsvarar helt ekvationerna i (24).

Det finns nu olika sätt att lösa det "two-point boundary problem", som har framkommit. Ett är att gissa initialvärden för de adjungerade funktionerna, $\lambda_i(t)$, och sedan iterera sig fram till den optimala lösningen.
Ett annat sätt är att gissa en styrsignal, $u(t) = u^0(t)$, integrera systemekvationerna framåt i tiden från t_0 till T och de adjungerade ekvationerna bakåt i tiden

från T till t_0, beräkna förlustfunktionen V och sedan ändra styrsignalen med $\delta u(t)$, så att förlustfunktionen avtar. Det sistnämnda steget kräver att vi beräknar förlustfunktionens gradient med avseende på u. Orsaken till att vi integrerar systemekvationerna framåt och de adjungerade ekvationerna bakåt är den form på vilken randvärdena är givna. En upprepning av hela proceduren med den nya styrsignalen, $u^{k+1}(t) = u^k(t) + \delta u(t)$, i stället sker sedan så länge förlustfunktionens gradient med avseende på u är mindre än noll. Denna metoden kallas Bryson-Kelley metoden.

Vi vill nu se hur förlustfunktionen ändras då vi gör en liten ändring $\delta u(t)$ i styrsignalen u(t).

Vi har

$$V(x^k,u^k) = \phi(x^k) + \int_{t_0}^{T} L(x^k,u^k)\,dt \tag{29}$$

$$\dot{x}^k = f(x^k,u^k) \tag{30}$$

Om vi nu ändrar styrsignalen $u^k(t)$ med $\delta u(t)$ till $u^{k+1}(t) = u^k(t) + \delta u(t)$, kommer detta att ge en ändring i tillståndsvariablerna, $x^{k+1}(t) = x^k(t) + \delta x(t)$.

Vi får

$$V(x^{k+1},u^{k+1}) = V(x^k+\delta x,\ u^k+\delta u) =$$

$$= \phi(x^k+\delta x) + \int_{t_0}^{T} L(x^k+\delta x,\ u^k+\delta u)\,dt \tag{31}$$

En Taylorserieutveckling av detta uttryck kring (x^k,u^k) ger

$$V(x^k+\delta x,\ u^k+\delta u) = \phi(x^k) + \phi_x\delta x + \frac{1}{2}\phi_{xx}(\delta x)^2 +$$

$$+ \int_{t_0}^{T}\{L(x^k,u^k) + L_x\delta x + L_u\delta u + \tfrac{1}{2}L_{xx}(\delta x)^2 + \tfrac{1}{2}L_{uu}(\delta u)^2 +$$

$$+ L_{xu}\delta x\delta u\Big\}\, dt + \text{termer av högre ordning i } \delta x \text{ och } \delta u \qquad (32)$$

I fortsättningen kommer i regel linearisering av förlustfunktionen att göras, dvs termerna med ordning två eller högre försummas. För vissa korrigeringar behövs emellertid i vissa fall termer av ordning två. Det gäller alltså att välja δu så litet att linearitetenen inte störs.

Från (30) fås

$$\dot{x}^k + \delta\dot{x} = f(x^k+\delta x,\ u^k+\delta u) \qquad (33)$$

Utveckling i Taylorserie kring (x^k,u^k) ger

$$\dot{x}^k + \delta\dot{x} = f(x^k,u^k) + f_x\delta x + f_u\delta u + \tfrac{1}{2} f_{xx}(\delta x)^2 +$$

$$+ \tfrac{1}{2} f_{uu}(\delta u)^2 + f_{xu}\delta x\delta u + \text{termer av högre ordning i } \delta x \text{ och } \delta u \qquad (34)$$

Om vi försummar termer med ordningen två eller högre, har vi fått

$$V(x^k+\delta x,u^k+\delta u) = V(x^k,u^k) + \phi_x\delta x + \int_{t_o}^{T}(L_x\delta x + L_u\delta u)\, dt \qquad (35)$$

$$\delta\dot{x} = f_x\delta x + f_u\delta u \qquad (36)$$

Det gäller nu att bestämma $\delta u(t)$ så att förlustfunktuionen (35) får minimum, då hänsyn tas till bivillkoret (36).

För att finna detta minimum inför de tidigare nämnda Lagrangemultiplikatorerna, $\lambda(t)$. Vi får

$$V(x^k+\delta x,u^k+\delta u) = V(x^k,u^k) + \phi_x\delta x + \int_{t_o}^{T}(L_x\delta x + L_u\delta u)\, dt +$$

$$+ \int_{t_o}^{T}\lambda^T(f_x\delta x + f_u\delta u - \delta\dot{x})\, dt =$$

$$= V(x^k,u^k) + \phi_x\delta x + \int_{t_o}^{T}(L_x + \lambda^T f_x)\delta x\, dt + \int_{t_o}^{T}(L_u + \lambda^T f_u)\delta u\, dt -$$

$$- \int_{t_o}^{T}\lambda^T\delta\dot{x}\, dt =$$

$$= V(x^k,u^k) + (\phi_x(x(T)) - \lambda(T))\,\delta x(T) +$$

$$+ \int_{t_0}^{T} (L_x + \lambda^T f_x + \dot{\lambda}^T)\delta x\, dt + \int_{t_0}^{T} (L_u + \lambda^T f_u)\delta u\, dt \qquad (37)$$

Om nu $\lambda(t)$ väljes så att

$$\begin{cases} \dot{\lambda} = -f_x^T \lambda - L_x \\ \lambda(T) = \phi_x(x(T)) \end{cases} \qquad (38)$$

fås

$$dV = V(x^k+\delta x, u^k+\delta u) - V(x^k,u^k) = \int_{t_0}^{T} (L_u + \lambda^T f_u)\delta u\, dt \qquad (39)$$

Om vi infört

$$H(x, \lambda, u) = L(x,u) + \lambda^T f(x,u) \qquad (40)$$

hade vi alltså fått

$$\dot{\lambda} = - H_x \qquad (41)$$

och

$$dV = \int_{t_0}^{T} H_u \delta u\, dt \qquad \text{(jfr (8))} \qquad (42)$$

Första ekvationen i (38) är identisk med (41) och (39) motsvarar (41). Denna sista härledningen av störtermen $\delta u(t)$:s inverkan på förlustfunktionen leder alltså fram till systemet av ekvationer i (24) och randvillkoren enligt (25), vilket är helt naturligt. Tack vare (32) och (34) kan vi emellertid, om så behövs, även ta hänsyn till termer av ordning två. Vi måste nämligen vid valet av $\delta u(t)$, välja ändringen så liten att linearitetenen inte störs, dvs termerna av ordning två och högre måste vara små relativt termerna av första ordningen i δu och δx.

Sammanfattningsvis kan vi i några punkter ställa upp Bryson-Kelley metoden för lösning av det aktuella styrproblemet:

(1) Gissa $u^0(t)$

(2) Integrera systemekvationerna framåt i tiden och spara $x^k(t)$

(3) Integrera de adjungerade ekvationerna bakåt i tiden ($p^k(t)$ kan ju fås då vi vet $x^k(t)$ och $u^k(t)$)

(4) Ändra styrvariablen u^k till $u^{k+1}(t) = u^k(t) + \delta u(t)$, där $\delta u(t)$ t.ex. lika med $\varepsilon \cdot H_u(x^k, {}^k, u^k)$. $\delta u(t)$ måste väljas så liten att linearitetenen inte störs och så att $|u^{k+1}(t)| < U$.

(5) Upprepa proceduren tills H_u inte längre är mindre än noll.

Program

Låt oss nu övergå till att se hur det program, som skall lösa det i föregående avsnitt givna optimala styrningsproblemet, är konstruerat. Programmet är skrivet i ALGOL och avsett att köras på SMIL (Siffermaskinen i Lund).

Programmet inleds med fyra procedurer, varav den första av dessa innehåller de aktuella systemekvationerna (FKT1), den andra de aktuella adjungerade ekvationerna (FKT2). Dessa procedurer,liksom den tredje (FKT3), som innehåller störningsekvationer (jfr nedan), måste utbytas, då vi löser andra problem, eftersom de just innehåller de för problemet aktuella systemekvationerna, adjungerade ekvationerna och störningsekvationerna. Den sista av de fyra procedurerna (RK1ST) användes för lösning av system av differentialekvationer av första ordningen. Den löser sådana system med Runge-Kutta metod. Steglängden h vid integrationen inläses i huvudprogrammet.

Efter procedurerna kommer själva programmet för styrproblemet. Först integreras systemekvationerna framåt i tiden från t_0 till T med hjälp av Runge-Kutta proceduren. I datamaskinens minne sparas $x(t_0+k\cdot h)$. t_0, T samt de gissade värdena på $u(t_0+ k\cdot h)$ har lästs in från en särskild dataremsa, liksom startvärdena för tillståndsvariablerna. Programmet är i sin numarande form konstruerat enbart för system av högst andra ordningen, bl.a. för att spara minnesutrymme.Det kan emellertid lätt utvidgas att lösa problem av högre ordning.

Efter integrationen av systemekvationerna sker en beräkning av förlustfunktionen V. Om denna är mindre än ett på förhand givet värde (konstanten k2) stannar programmet, i annat fall fortsätter det med att integrera de adjungerade ekvationerna bakåt från T till t_0. Härvid väljes $\lambda(T) = \phi_x(x(T))$ (se (38). Värdena $\lambda(T - k \cdot h)$ lagras också. Därefter väljes $\delta u(t_0+k \cdot h) = - \operatorname{sgn}(H_u)$ men samtidigt så att $|u(t_0+k \cdot h) + \delta u(t_0+k \cdot h)| \leq U$, begränsningen på styrsignalen.

Så sker en kontroll av att linearitetenen, som var ett krav för lösningen (se (32) och fortsättningen), ej störs av detta val av δu. Detta sker genom integration av störningsekvationen (36). Vidare sker en beräkning av andra ordningens termer vid Taylorutvecklingen (se (32)) och en kontroll av att valet av δu, och därmed av δx, ej gör dessa termer stora i förhållande till första ordningens termer. Då dessa termers utseende beror av förlustfuntionen, är denna del av programmet inte generellt utan konstruerad för att passa de två testexempel, som körts.

För dessa två testexempel(se nedan) gäller

$$V = x_1^2(T) + x_2^2(T) \tag{43}$$

och vi får

$$dV = (2x_1(T) - \lambda_1(T))\,\delta x_1(T) + (2x_2(T) - \lambda_2(T))\,\delta x_2(T) + \\ + (\delta x_1(T))^2 + (\delta x_2(T))^2 + \int_{t_0}^{T} \lambda_2 \cdot \delta u \, dt \tag{44}$$

Men vi har valt

$\lambda_1(T) = 2\,x_1(T)$ och $\lambda_2(T) = 2\,x_2(T)$ (se ovan)

och linearitetenen störs alltså om $\delta x_1(T)$ resp. $\delta x_2(T)$

är stora i förhållande till $2x_1(T)$ resp. $2x_2(T)$.

För att försäkra sig om god linearitet kan vi välja

$$c \cdot |\delta x_1(T)| \leq |x_1(T)| \quad \text{och} \quad c \cdot |\delta x_2(T)| \leq |x_2(T)|,$$

med någon lämplig konstant c.

Vi får

$$c^2 \, (\delta x_1^{\,2}(T) + \delta x_2^{\,2}(T)) \leq x_1^{\,2}(T) + x_2^{\,2}(T) = V \qquad (45)$$

Vi sätter $c^2 = kl$. Det har vid testkörningarna visat sig lämpligt att välja kl mellan 2 och 50 för erhållande av hygglig konvergens.

Om δu (δx) valts för stora sker en nedskalning enligt formeln

$$\delta u_{nytt} = \delta u_{gammalt} \sqrt{\frac{V}{kl(\delta x_1^{\,2}(T) + \delta x_2^{\,2}(T))}} \qquad (46)$$

Formeln har visat sig lämplig beroende på att δx nästan lineärt i δu.

Slutligen sättes styrsignalen för nästa iteration lika med summan av den föregående styrsignalen och den nu bestämda ändringen, δu. Därefter sker hopp till programmets början, och programmet genomlöpes tills slutvillkoren är tillfredsställande uppfyllda.

```
begin integer m, n, i, j, k, l;
      real t, T, h, te, tf, k1, k2, V;
      array x1, x2, u, du, p1, p2 [1:101],
            y, ye, z, dy,dye, dz, p, pe, q [1:2];


procedure FKT1 (x,y,z); real x; array y,z;
begin z[1]:=y[2]; z[2]:=u[j-1]
end FKT1;


procedure FKT2 (x,p,q); real x; array p, q;
begin q[1]:=0; q[2]:=-p[1]
end FKT2;

procedure FKT3 (x,dy,dz); real x; array dy,dz;
begin dz[1]:=dy[2]; dz[2]:=du[j-1]
end FKT3;


procedure RK1ST (t,y,h,te,ye,f,p); real t,te,h;integer p; array y,ye;
          procedure f;
begin integer j,k; array w[1:10], a[1:5];
a[1]:=a[2]:=a[5]:=h/2; a[3]:=a[4]:=h;
te:=t;
for k:=1 step 1 until n do ye[k]:= w[k]:=y[k];
for j:=1 step 1 until 4 do
begin f(te,w,z); te:=t + a[j];
for k:=1 step 1 until n do
begin w[k]:=y[k] + a[j] × z[k];
      ye[k]:=ye[k] + a[j+1] × z[k]/3;
end k
end j;
if p=1 then begin print (3,6,te); punch (8);
for j:=1 step 1 until n do
begin print (3,6,ye[j]); punch (0);
end j;
punch (1);
end p
end RK1ST;


n:=read; m:=read; h:=read;
for i:=1 step 1 until m do u[i]:= read;
y[1]:=read; y[2]:=read;
t:=read; T:=read; k1:=read; k2:=read;
```

```
x1[1]:=y[1]; x2[1]:=y[2];
P:j:=2; tf:=t;
L1:RK1ST(tf,y,h,te,ye,FKT1,2);
tf:=te;x1[j]:=ye[1]; x2[j]:=ye[2];
y[1]:=ye[1]; y[2]:=ye[2];
j:=j+1;
if abs(T-te)>10-2 then go to L1;

print (3,8,x1[m]); print (3,8,x2[m]); punch (1); punch(1);

V:=x1[m]↑2 + x2[m]↑2;

if V<k2 then go to ut;

p1[m]:=p[1]:=2 × x1[m]; p2[m]:=p[2]:= 2 × x2[m];
Q:k:=m-1; tf:=T;
L2:RK1ST(tf,p,-h,te,pe,FKT2,2);
tf:=te; p1[k]:=pe[1]; p2[k]:=pe[2];
p[1]:=pe[1]; p[2]:=pe[2];
k:=k-1;
if abs(t-te)>10-2 then go to L2;


for i:=1 step 1 until m do
    begin du[i]:=-sign(p2[i]);  if abs(u[i]+du[i])> 1 then
          du[i]:=sign(du[i]) × (1 - abs(u[i])) end;

dy[1]:=dy[2]:=0;
S:j:=2; tf:=t;
L3:RK1ST(tf,dy,h,te,dye,FKT3,2);
tf:=te; dy[1]:=dye[1]; dy[2]:=dye[2];
j:=j+1;
if abs(T - te)>10-2 then go to L3;

if k1 × (dy[1]↑2 + dy[2]↑2) > V
   then for i:=1 step 1 until m do
        du[i]:=du[i] × sqrt(V/k1/(dy[1]↑2 + dy[2]↑2));

l:=1;
R: if du[l]=0 then begin l:=l+1; if l<m then go to R else go to ut end;

for i:=1 step 1 until m do
         begin u[i]:=u[i]+ du[i]; print (3,8,u[i]); punch(1)
         end;
         punch(1);
   y[1]:=x1[1]; y[2]:=x2[1];
   go to P;

ut:punch(8)

end
```

Testexempel 1 (dubbelintegratorn)

Betrakta systemet

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

Finn en styrsignal sådan att

$V = x_1^2(T) + x_2^2(T)$ minimeras då

$x_1(0) = 1$

$x_2(0) = 0$

$T = 2 \quad (T = 1{,}9)$

$|u| \leq 1$

Exakt lösning

Enligt (26) erhålles

$H = \lambda_1 x_2 + \lambda_2 u$

u skall väljas så att H minimeras med avseende på u, dvs

$u = - \operatorname{sgn} \lambda_2$

Vi har alltså följande system

$$\begin{cases} \dfrac{dx_1}{dt} = x_2 \\ \dfrac{dx_2}{dt} = - \operatorname{sgn} \lambda_2 \end{cases} \qquad \begin{cases} x_1(0) = 1 \\ x_2(0) = 0 \end{cases}$$

och vidare fås enligt (28)

$$\begin{cases} \dfrac{d\lambda_1}{dt} = 0 \\ \dfrac{d\lambda_2}{dt} = - \lambda_1 \end{cases}$$

Härur fås

$\lambda_1(t) = c_1$

$\lambda_2(t) = - c_1 t + c_2$

dvs $\lambda_2(t)$ har endast en teckenväxling.

Antag att $\lambda_2(t)$ ändrar tecken för $t=t_1$.

Då erhålles för $o \leq t \leq t_1$

$$\frac{dx_2}{dt} = -1$$

dvs.

$$x_2(t) = -t \qquad (x_2(0)= 0)$$

samt vidare

$$x_1(t) = 1 - \frac{t^2}{2} \qquad (x_1(0)=1)$$

Vi får härur

$$x_1(t_1) = 1 - \frac{t_1^2}{2} \quad \text{och} \quad x_2(t_1)= - t_1$$

Vidare erhålles för $t_1 \leq t \leq T$

$$\frac{dx_2}{dt} = 1$$

dvs

$$x_2(t) = t - 2t_1 \qquad (\text{för att uppfylla } x_2(t_1)= - t_1)$$

samt vidare

$$x_1(t) = \frac{t^2}{2} - 2t_1 t + 1 + t_1^2 \qquad (\text{för att uppfylla } x_1(t_1)=1-\frac{t_1^2}{2})$$

Vi får sedan

$$\begin{cases} x_1(T) = \frac{T^2}{2} - 2t_1 T + 1 + t_1^2 \\ x_2(T) = T - 2t_1 \end{cases}$$

Om vi bildar

$$V = x_1^2(T) + x_2^2(T)$$

och deriverar detta uttryck med avseende på t_1, finner man att dess derivata har endast ett nollställe, och att detta värde på t_1 ger minimum.

(Att vi ovan förutsatt att $\frac{dx_2(t)}{dt} = -1$ i intervallet $0 \leq t \leq t_1$ och $+1$ i det andra intervallet, beror på att förlustfunktionen självklart blir större om vi gör tvärtom, så länge $x_1(0) > 0$)

T=2 ger minimum för V för $t_1 = 1$, medan T = 1,9 ger minimum för $t_1 \approx 0{,}98$.

$\lambda_1(t)$, $\lambda_2(t)$, $u(t)$, $x_1(t)$ och $x_2(t)$ har ritats i diagram 1 för fallet T = 2. Liknande kurvor erhålles för fallet T = 1,9.

Numerisk lösning med programmets hjälp

Programmet har körts för olika värden på konstanten k1 och i ett fall dessutom för mer än en gissad styrsignal. Låt oss se på de körda fallen med T=2:

k1=10, $u^0(t) = -1$: Oscillationerna mellan värden, som ändrade sig relativt långsamt, uppträdde på ett tidigt stadium, varför körningen avbröts. Om konvergens trots detta föreligger, ger detta värde på konstanten en mycket långsam konvergens.

k1=15, $u^0(t) = -1$: Resultatet av körningen framgår av diagram 2.

k1=20, $u^0(t) = -1$: Se diagram 3

k1=20, $u^0(t) = +1$: Se diagram 4

k1=25, $u^0(t) = -1$: Se diagram 5

Av diagrammen framgår direkt att för dubbelintegratorn ger k1 =20 snabbast konvergens.

Programmet har även körts för T = 1,9, k1 = 20 och $u^0(t) = -1$. Resultatet enligt diagram 6.

Tiden för en iteration beror nästan uteslutande på valet av steglängden vid användningen av Runge-Kutta proceduren. Tiden för en iteration är ca en minut vid körning på SMIL, om steglängden h är sådan att $(T - t_0)/h = 25$.

Testexempel 2 (oscillatorn)

Betrakta systemet

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

Finn en styrsignal sådan att kriteriet

$V = x_1^2(T) + x_2^2(T)$ minimeras då

$$\begin{cases} x_1(0) = 10 \\ x_2(0) = 0 \end{cases}$$

$T = 5\pi$ $(T=15)$

$|u| \leq 1$

Exakt lösning

För detta exempel erhålles

$H = \lambda_1 x_2 - \lambda_2 x_1 + \lambda_2 u$

dvs u skall väljas enligt

$u = -\operatorname{sgn} \lambda_2$

Vi har alltså följande system

$$\begin{cases} \frac{dx_1}{dt} = x_2 \\ \frac{dx_2}{dt} = -x_1 - \operatorname{sgn} \lambda_2 \end{cases} \qquad \begin{cases} x_1(0) = 10 \\ x_2(0) = 0 \end{cases}$$

samt

$$\begin{cases} \frac{d\lambda_1}{dt} = \lambda_2 \\ \frac{d\lambda_2}{dt} = -\lambda_1 \end{cases}$$

Lösningen till systemet med de adjungerade funktionerna blir trigonometriska funktioner, och vi inser, att vi får fyra teckenväxlingar för $\lambda_2(t)$ i intervallet $0 \leq t \leq 5\pi$.

Detta problem är besvärligare att lösa än det förra, men lösningen för $T = 5\pi$, som gör $V = 0$, finns på diagram 7.

Numerisk lösning med programmets hjälp

Också för detta exempel har programmet körts för olika värden på konstanten kl, men i samtliga fall har den primärt gissade styrsignalen varit $u(t) = -1$. Programmet har körts för med i vissa fall olika värden på steglängden h. Detta har gjort att sluttiden T har varit antingen 15,6 (för h=0,4) eller 15,75 (för h=0,35), vilka värden alltså använts som approximation till 5π. I stället för T=15 har använts T=14,8 resp. T=15,05. Detta val inverkar emellertid obetydligt på resultatet.

kl = 2: T = 15,6 Se diagram 8

T = 14,8 Se diagram 9

kl = 5: T = 15,75 (och 15,6) Se diagram 10

T = 15,05 (och 14,8) Se diagram 11

kl = 10: T = 15,6 Se diagram 12

kl = 25: T = 15,6 Se diagram 13

Av dessa diagram framgår att konvergensen är snabbast för kl = 5 eller 10. Oscillatorn har emellertid visat sig betydligt okänsligare, vad beträffar konvergensen, för valet av kl än dubbelintegratorn.

Vidare framgår det att valet av steglängden h kan vara betydelsefull. Ju kortare h, desto bättre erhålles den tidpunkt, för vilken u(t) gör språng. Men samtidigt ökar räknetiden avsevärt, så det hela blir en avvägningsfråga mellan disponibel räknetid och noggrannhet.

Kommentar

Programmet har alltså visat sig ge en ganska god approximation av den optimala styrsignalen för de två testexempel, som undersökts. Om en hygglig gissning av styrsignalen u(t) göres från början och om konstanten kl väljes lämpligt, kan vi dessutom få en ganska snabb konvergens. Vad beträffar kl finns den möjligheten att vid lösningen av ett visst problem testa konvergensen genom att låta datamaskinen utför ett mindre antal iterationer för olika värden på kl. En annan och bättre möjlighet är att köra programmet och efter varje iteration läsa in ett nytt värde på kl, som då kan ändras successivt. Denna metod rekommenderas, då konvergensen oftast varierar under körningen av ett problem. Störningar i linearitetен blir större, då vi närmar oss målet, och det är då lämpligt att välja ett större värde på kl. Programmet har körts med denna modifikation, som göres mycket enkelt.

Styrningsproblemet med T variabel

I föregående avsnitt har vi löst optimala styrproblem med sluttiden T fix. Trots detta kan metoden användas att lösa det sk. minimaltidsproblemet, dvs det styrproblem, där det gäller att finna den styrsignal, som på minsta tid överför systemet från en viss tillståndskonfiguration till en annan. Vi kan nämligen lösa problemet för flera olika rimliga sluttider T och sedan bestämma för vilket minsta T, som de givna slutvillkoren är uppfyllda. Vi har alltså en möjlighet att approximativt nå lösningen, dock i regel efter en hel del arbete. Därför vore det lämpligt att finna en metod, som direkt löser minimaltidsproblemet.

En möjlig väg skulle kunna vara följande :

$$V(x^k+\delta x, u^k+\delta u) = V(x^k,u^k) + (\phi_x - \lambda(T))\,\delta x(T) + \int_{t_0}^{T} (L_x + \lambda^T f_x + \dot{\lambda}^T)\,\delta x\,dt + \int_{t_0}^{T} (L_u + \lambda^T f_u)\,\delta u\,dt \qquad (47)$$

enligt tidigare.

Om T inte längre fix tillkommer en term i δT.

Om vi ser på fallet med dubbelintegratorn erhålles

$$dV = (2x_1(T) - \lambda_1(T))\,\delta x_1(T) + (2x_2(T) - \lambda_2(T))\,\delta x_2(T) + \delta T + \int_{t_0}^{T} \lambda_2 \delta u\,dt \qquad (48)$$

Om $x_1(T)$ och $x_2(T)$ skall vara noll, kan vi integrera systemekvationerna framåt tills $x_1(t) < \varepsilon$, med $\varepsilon > 0$ och ε litet, och sätta T lika med den tidpunkten vi då nått, dvs $x_1(T) \approx 0$. Vid nästa iteration gör vi likadant men måste då integrera till $T + \delta T$ innan $x_1 < \varepsilon$, dvs $x_1(T + \delta T) \approx 0$.

Detta ger

$x_1(T + \delta T) = 0$

eller differentierat

$x_1(T + \delta T) = x_1(T) + \delta x_1(T) + \dot{x}_1(T)\,\delta T = 0$

vilket ger

$$\delta T = - \delta x_1(T)/\dot{x}_1(T) \qquad (49)$$

eftersom

$x_1(T) = 0.$

(48) och (49) ger då tillsammans att

$\lambda_1(T)$ skall väljas lika med $2x_1(T) - (\dot{x}_1(T))^{-1}$ (jfr (44) och följande). $\lambda_2(T)$ skall liksom tidigare väljas lika med $2x_2(T)$.

Men $\dot{x}_1(T) = x_2(T)$ och detta skall gå mot noll, då vi närmar oss den optimala lösningen.

Med en liten modifiering av det program, som användes för fix tid, skulle alltså minimaltidsproblemet kunna lösas, om vi kan gardera oss för de svårigheter, som uppstår, då $x_2(T)$ blir liten. Något sådant program har emellertid inte konstruerats, då svårigheterna att komma till rätta med problemet med $x_2(T)$ nära noll väntas bli stora.

I stället har direkt tillämpats den metodik, som Bryson och Denham redogjort för i sin uppsats "A steepest ascent method for solving optimum programming problems', för att konstruera ett program, som löser minimaltidsproblemet för dubbelintegratorn.

Först ges det allmänna problemets förutsättningar och lösningen till detta, och därefter undersöker vi närmare minimaltidsproblemet för dubbelintegratorn.

Formulering av det allmänna problemet

Bestäm $u(t)$ i intervallet $t_0 \leq t \leq T$ så att

$$\phi = \phi(x(T),T) \tag{50}$$

maximeras under bivillkoren

$$\psi = \psi(x(T),T) = 0 \tag{51}$$

$$\frac{dx}{dt} = f(x(t),u(t),t) \tag{52}$$

t_o och $x(t_0)$ givna (53)

T bestämmes av $\Omega = \Omega(x(T),T) = 0$ (54)

(jfr (1)-(5))

Här gäller

$$u(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}$$

(en m-dimensionell vektor av styrvariabler, som ev. kan vara begränsade, men för övrigt får väljas fritt)

$$x(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

(en n-dimensionell vektor av tillståndsvariabler)

$$\psi = \begin{bmatrix} \psi_1 \\ \vdots \\ \psi_q \end{bmatrix}$$

(en q-dimensionell vektor av slutvillkor; kända funktioner av $x(T)$ och T)

ϕ är förlustfunktionen

f är en n-dimensionell vektor av kända funktioner

Ω är det stoppvillkor, som bestämmer sluttiden T, och är en känd funktion.

Bryson och Denham menar nu, att detta optimala styrproblem kan lösas systematiskt och snabbt på en modern datamaskin genom användning av s.k. "steepest-ascent"-

metodik. Denna innebär,kort uttryckt, att man vid iterationen mot den optimala lösningen hela tiden försöker röra sig i gradientens riktning. Den metod, som ges i ovan nämnda uppsats, startar med ett gissat värde på styrsignalen, $u^0(t)$, och förbättrar sedan denna gissning steg för steg genom undersökning av föregående styrsignals inverkan. Metoden bygger på en lokal linearisering kring den väg,som följdes i föregående steg. Metoden är alltså en Bryson-Kelley metodik, men eftersom det för programkonstruktionen behövs ett explicit uttryck på $\delta u(t)$, lämnas här en sammanfattning av teorin.

Först gissas ett rimligt värde på styrsignalen, $u^0(t)$, och detta värde jämte initialvärdena (53) och differentialekvationssystemet (52) användes för att beräkna $x^0(t)$ tills $\Omega = 0$. I allmänhet satisfierar ej denna väg slutvillkoren $\psi = 0$ och ej heller nås maximum av ϕ.

Sedan betraktas en störning, $\delta u(t)$, i styrvariablen. Denna störning ger en ändring, $\delta x(t)$, i tillståndsvariablen. Insättning i (52) ger som tidigare

$$\delta \dot{x} = F(t) \cdot \delta x + G(t) \cdot \delta u \tag{55}$$

om hänsyn tas endast till termer av första ordningen.

$$F(t) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{56}$$

$$G(t) = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial u_m} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix} \tag{57}$$

Samtliga derivator beräknas längs vägen $u^0(t)$ och $x^0(t)$.

Vi kan vidare skriva

$$d\phi = \int_{t_o}^{T} \lambda^T(t)\, G(t)\, \delta u(t)\, dt + \lambda_\phi^T(t_0)\, \delta x(t_0) + \dot{\phi}\, dT \qquad (58)$$

$$d\psi = \int_{t_o}^{T} \lambda^T(t)\, G(t)\, \delta u(t)\, dt + \lambda_\psi^T(t_0)\, \delta x(t_0) + \dot{\psi}\, dT \qquad (59)$$

$$d\Omega = \int_{t_o}^{T} \lambda^T(t)\, G(t)\, \delta u(t)\, dt + \lambda_\Omega^T(t_0)\, \delta x(t_0) + \dot{\Omega}\, dT \qquad (60)$$

där elementen i λ-matriserna bestämmes genom numerisk integration av de adjungerade ekvationerna

$$\frac{d\lambda}{dt} = -F^T(t)\, \lambda \qquad (61)$$

med randvillkoren

$$\lambda_\phi^T(T) = \left(\frac{\partial \phi}{\partial x}\right) \qquad (62)$$

$$\lambda_\psi^T(T) = \left(\frac{\partial \psi}{\partial x}\right) \qquad (63)$$

$$\lambda_\Omega^T(T) = \left(\frac{\partial \Omega}{\partial x}\right) \qquad (64)$$

där derivatorna är utvärderade i punkten t=T längs vägen $u^0(t)$ och $x^0(t)$.

Vidare definieras

$$\frac{\partial \phi}{\partial x} = \left(\frac{\partial \phi}{\partial x_1}, \ldots, \frac{\partial \phi}{\partial x_n}\right) \qquad (65)$$

$$\frac{\partial \psi}{\partial x} = \begin{bmatrix} \frac{\partial \psi_1}{\partial x_1} & \cdots & \frac{\partial \psi_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial \psi_q}{\partial x_1} & \cdots & \frac{\partial \psi_q}{\partial x_n} \end{bmatrix} \qquad (66)$$

$$\frac{\partial \Omega}{\partial x} = \left(\frac{\partial \Omega}{\partial x_1}, \ldots, \frac{\partial \Omega}{\partial x_n}\right) \qquad (67)$$

och vidare

$$\dot{\phi} = \left(\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x} f\right) \qquad (68)$$

$$\dot{\psi} = \left(\frac{\partial \psi}{\partial t} + \frac{\partial \psi}{\partial x} f\right) \qquad (69)$$

$$\dot{\Omega} = \left(\frac{\partial \Omega}{\partial t} + \frac{\partial \Omega}{\partial x} f\right) \tag{70}$$

De tre sista uttrycken ((68)-(70)) är utvärderade längs vägen $u^0(t)$ och $x^0(t)$ och för $t=T$.

De adjungerade ekvationerna integraras bakåt i tiden, eftersom randvärden är givna för $t=T$.

För sk. "steepest ascent" sökes nu ett $\delta u(t)$, sådant att $d\phi$ maximeras för ett givet värde på integralen

$$(dP)^2 = \int_{t_0}^{T} \delta u^T(t)\, W(t)\, \delta u(t)\, dt \tag{71}$$

samt givna värden på $d\psi$ och $d\Omega = 0$.

$d\psi$ väljes så att vi närmar oss de önskade slutvillkoren $\psi = 0$. dP väljes så att linearitetenen inte störs av vårt val av $\delta u(t)$. W(t) är en godtycklig m x m-matris av viktfunktioner vald så att konvergensen förbättras. Man kan under dessa förutsättningar härleda hur $\delta u(t)$ skall väljas och följande uttryck erhålles

$$\delta u(t) = W^{-1} G^T \left(\lambda_{\phi\Omega} - \lambda_{\psi\Omega} I_{\psi\psi}^{-1} I_{\psi\phi} \right) \left(\frac{(dP)^2 - d\beta^T I_{\psi\psi}^{-1} d\beta}{I_{\phi\phi} - I_{\psi\phi}^T I_{\psi\psi}^{-1} I_{\psi\phi}} \right)^{1/2} + \\ + W^{-1} G^T \lambda_{\psi\Omega} I_{\psi\psi}^{-1} d\beta \tag{72}$$

där

$$\begin{cases} d\beta = d\psi - \lambda_{\psi\Omega}^T(t_0)\, x(t_0) \\ \lambda_{\phi\Omega} = \lambda_\phi - \dfrac{\dot{\phi}}{\dot{\Omega}} \lambda_\Omega \\ \lambda_{\psi\Omega} = \lambda_\psi - \lambda_\Omega \dfrac{\dot{\psi}^T}{\dot{\Omega}} \\ I_{\psi\psi} = \displaystyle\int_{t_0}^{T} \lambda_{\psi\Omega}^T G W^{-1} G^T \lambda_{\psi\Omega}\, dt \\ I_{\psi\phi} = \displaystyle\int_{t_0}^{T} \lambda_{\psi\Omega}^T G W^{-1} G^T \lambda_{\phi\Omega}\, dt \\ I_{\phi\phi} = \displaystyle\int_{t_0}^{T} \lambda_{\phi\Omega}^T G W^{-1} G^T \lambda_{\phi\Omega}\, dt \end{cases} \tag{73}$$

Om $d\beta$ väljes för stor kan täljaren under rottecknet bli negativ, vilket alltså betyder, att det finns en begränsning på $d\beta$,då dP är givet. Eftersom dP väljs för att säkerställa linearitetet, måste också $d\beta$ begränsas. Den beräknade ändringen i ϕ då u(t) ändras med $\delta u(t)$ blir

$$d\phi = (((dP)^2 - d\beta^T I_{\psi\psi}^{-1} d\beta)(I_{\phi\phi} - I_{\psi\phi}^T I_{\psi\psi}^{-1} I_{\psi\phi}))^{1/2} + \\ + I_{\psi\phi}^T I_{\psi\psi}^{-1} d\beta + \lambda_{\phi a}^T(t_0)\, \delta x(t_0) \qquad (74)$$

Om $d\psi = 0$, $\delta x(t_0) = 0$ erhålles

$$\frac{d\phi}{dP} = (I_{\phi\phi} - I_{\psi\phi}^T I_{\psi\psi}^{-1} I_{\psi\phi})^{1/2} \qquad (75)$$

Detta uttryck kan sägas vara gradienten i funktionsrummet, då dP är "steglängden" för styrvariablen. Då vi närmar oss den optimala lösningen och slutvillkoren är uppfyllda, går denna gradient mot noll, och detta ger ett uttryck på $d\phi$, som visar hur förlustfunktionen ändras vid små störningar i slutvillkor och initialvärden.

$$d\phi = I_{\psi\phi}^T I_{\psi\psi}^{-1} d\psi + (\lambda_{\phi a}^T(t_0) - I_{\psi\phi}^T I_{\psi\psi}^{-1} \lambda_{\psi a}^T(t_0))\, \delta x(t_0) \qquad (76)$$

Så väljes en ny styrsignal som summan av den föregående styrsignalen och $\delta u(t)$ enligt (72). Hela proceduren upprepas sedan tills slutvillkoren uppfyllda och tills gradienten är nära noll. Då så är fallet, har den optimala styrsignalen erhållits.

Denna metod kan användas för minimaltidsproblemet. För detta fås

$\phi = - t$

vilket ger

$\dot{\phi} = -1$

$\lambda_\phi = 0$ och alltså

$$\lambda_{\phi\Omega} = \frac{1}{\dot{\Omega}}$$

$$d\phi = -dT = \int_{t_0}^{T} \frac{1}{\dot{\Omega}} \lambda_\Omega{}^T G\, \delta u(t)\, dt + \frac{1}{\dot{\Omega}} \lambda_\Omega{}^T(t_0) \cdot \delta x(t_0) \qquad (77)$$

För lösning av detta problem med hjälp av ett datamaskinprogram kan följande huvudpunkter uppställas:

(1) Gissa $u^0(t)$ och integrera systemekvationerna från t_0 tills slutvillkoret $\Omega = 0$ uppfyllt. Spara $x^0(t)$ i minnet.

(2) Beräkna samtliga adjungerade ekvationerna samtidigt genom att integrera (61) bakåt, och använd härvid de under (1) beräknade partiella derivatorna. Spara $\lambda^0(t)$.

(3) Beräkna samtidigt dessutom $\lambda_{\phi\Omega}, \lambda_{\psi_1\Omega}, \ldots, \lambda_{\psi_q\Omega}$ och spara $\lambda_{\phi\Omega}^T G$ och $\lambda_{\psi\Omega}^T G$. Beräkna vidare samtidigt talen $I_{\phi\phi}$, $I_{\psi\phi}$ och $I_{\psi\psi}$ genom att integralerna utvärderas (integration bakåt i t)

(4) Välj ändringarna i slutvillkoren, $d\psi_1, \ldots, d\psi_q$, så att nästa lösning närmar sig $\psi = 0$.

(5) Välj ett värde på $(dP)^2/(T - t_0)$

(6) Använd värdena på d och dP för att beräkna $V = (dP)^2 - d\psi^T I_{\psi\psi}^{-1} d\psi$. Om V negativt skalas $d\psi$ ned så att V = 0. Om positivt görs ingen ändring.

(7) Beräkna sedan $\delta u(t)$ enligt (72). ($\delta x(t_0) = 0$).

(8) Om sluttiden T ej given eller skall minimeras, beräknas dessutom ändringen dT i nästa steg:

$$dT = -\frac{1}{\dot{\Omega}} \int_{t_0}^{T} \lambda_\Omega^T\, G\, \delta u(t)\, dt \qquad (78)$$

Om $|dT|$ större än ett på förhand valt värde, skalas $\delta u(t)$ ner, så att detta värde ej överskrides.

(9) Ett nytt värde på u(t) erhålles ur

$$u(t) = u^0(t) + \delta u(t)$$

Kontroll av att u(t) ej överskrider eventuella begränsningar sker.

Punkterna (1)-(9) upprepas sedan tills $\psi = 0$ och tills kvadraten på gradienten

$$I_{\phi\phi} - I_{\psi\phi}^{T} I_{\psi\psi}^{-1} I_{\psi\phi}$$

är nära noll.

Program för specialfallet dubbelintegratorn

Ett program, skrivet i ALGOL, har konstruerats för ett specialfall, nämligen minimaltidsproblemet för dubbelintegratorn (jfr sid. 18).

Problemet är alltså följande;

Minimera

$$\phi = -t$$

då

$$\begin{cases} \dfrac{dx_1}{dt} = x_2 \\ \dfrac{dx_2}{dt} = u \end{cases}$$

$$\psi = x_2(T) = 0$$

$$t_0 = 0,\ x_1(0) = 1,\ x_2(0) = 0$$

$$\Omega = x_1(T) = 0$$

Vi får härur med beteckningar enligt tidigare

$$F(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$G(t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

samt

$$\dot{\lambda} = -F^{T}(t)\lambda = -\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\lambda$$

Vidare erhålles

$$\begin{cases} \lambda_{\phi_1} = 0 & \lambda_{\phi_1}(T) = 0 \\ \lambda_{\phi_2} = -\lambda_{\phi_1} & \lambda_{\phi_2}(T) = 0 \end{cases}$$

dvs. $\lambda_{\phi_1}(t) = \lambda_{\phi_2}(t) = 0$

$$\begin{cases} \lambda_{\psi_1} = 0 & \lambda_{\psi_1}(T) = 0 \\ \lambda_{\psi_2} = -\lambda_{\psi_1} & \lambda_{\psi_2}(T) = 1 \end{cases}$$

dvs $\lambda_{\psi_1}(t) = \lambda_{\psi_2}(t) = 0$

$$\begin{cases} \lambda_{\Omega_1} = 0 & \lambda_{\Omega_1}(T) = 1 \\ \lambda_{\Omega_2} = -\lambda_{\Omega_1} & \lambda_{\Omega_2}(T) = 0 \end{cases}$$

dvs $\lambda_{\Omega_1}(t) = 1$ och $\lambda_{\Omega_2}(t) = -t + T$

I fortsättningen betecknas λ_{Ω_2} med p_2.

Vi får dessutom

$\dot{\phi} = -1$

$\dot{\psi} = u^0(T)$

$\dot{\Omega} = x_2{}^0(T)$

Vidare definieras

$$(dP)^2 = \int_0^T k(\delta u(t))^2 dt$$

där k väljes så att snabb konvergens erhålles.

Insättning av detta i tidigare givna formler ger

$$\lambda_{\phi\Omega}{}^T = \frac{1}{x_2{}^0(T)} (1, p_2)$$

$$\lambda_{\psi\Omega}{}^T = \left(\frac{u^0(T)}{x_2{}^0(T)}, 1 - p_2 \frac{u^0(T)}{x_2{}^0(T)}\right)$$

$d\beta = d\psi$

$$I_{\psi\psi} = \int_0^T k^{-1} \lambda_{\psi\Omega}{}^T GG^T \lambda_{\psi\Omega} dt$$

$$I_{\psi\phi} = \int_0^T k^{-1} \lambda_{\psi\Omega}{}^T GG^T \lambda_{\phi\Omega} dt$$

$$I_{\phi\phi} = \int_0^T k^{-1} \lambda_{\phi\Omega}{}^T GG^T \lambda_{\phi\Omega} dt$$

Dessa värden användes sedan då $\delta u(t)$ beräknas.

Programmet är skrivet enbart för dubbelintegratorn, vilket alltså innebär, att de speciella värden på olika storheter, som framkommit på de två senaste sidorna, använts. Härigenom har den mängd av matrismultplikationer, som förekommer i ett generellt program, undvikits.

Programmet inleds med tre procedurer, FKT1, FKT2 och RK1ST. Den första innehåller systemekvationerna, den andra de adjungerade ekvationerna (som för detta enkla fall endast blir $\dot{\lambda}_n = -F(t)\lambda_n$; de övriga adjungerade funktionerna blir identiskt noll) och den tredje proceduren är den i föregående avsnitt nämnda Runge-Kutta proceduren.

Därefter följer själva beräkningsdelen av programmet. Denna del är konstruerad i enlighet med punkterna (1) - (9) sid. 31.

Körning på SMIL

I programmet användes tre konstanter, k1, k2 och k3. Med hjälp av k1 väljes $d\varphi$ och med hjälp av k2 väljs $(dP)^2$. k3 ombesörjer nedskalningen av δu, om dT blir för stor. Valet av dessa konstanter är mycket betydelsefullt för att konvergens skall erhållas. Efter en hel del försök nåddes ett någorlunda hyggligt resultat. Se diagram 14. Som framgår av detta är emellertid tyvärr konvergensen mot den exakta styrsignalen i området kring övergången från -1 till +1 dålig. Däremot ger metoden en god approximation av den minimala tiden, varefter vi enligt sid. 24 kan använda metoden, som användes vid fixt T, för att nå den optimala styrsignalen.

```
begin integer n, i, j, j1, k, l;
      real t, T, h, te, tf, dp, k1, k2, k3, dP, V, dT;
      array x1, x2, u, du, p1, p2, Ipp, Ipf, Iff, lfo,lpo[1:110],
            y, ye, z, p, pe, q [1:2];


procedure FKT1 (x,y,z); real x; array y,z;
begin z[1]:=y[2]; z[2]:=u[j-1]
end FKT1;


procedure FKT2 (x,p,q); real x; array p, q;
begin q[1]:=0; q[2]:=-p[1]
end FKT2;


procedure RK1ST (t,y,h,te,ye,f,p); real t,te,h;integer p; array y,ye;
          procedure f;
begin integer j,k; array w[1:10], a[1:5];
a[1]:=a[2]:=a[5]:=h/2; a[3]:=a[4]:=h;
te:=t;
for k:=1 step 1 until n do ye[k]:= w[k]:=y[k];
for j:=1 step 1 until 4 do
begin f(te,w,z); te:=t + a[j];
for k:=1 step 1 until n do
begin w[k]:=y[k] + a[j] × z[k];
      ye[k]:=ye[k] + a[j+1] × z[k]/3;
end k
end j;
if p=1 then begin print (3,6,te); punch (8);
for j:=1 step 1 until n do
begin print (3,6,ye[j]); punch (0);
end j;
punch (1);
end p
end RK1ST;


n:=read; h:=read;
for i:=1 step 1 until 51 do u[i]:= read;
y[1]:=read; y[2]:=read;
p[1]:=read; p[2]:=read;
t:=read; k1:=read; k2:=read; k3:=read;


x1[1]:=y[1]; x2[1]:=y[2];
P:j:=2; tf:=t;
L1:RK1ST(tf,y,h,te,ye,FKT1,2);
tf:=te;x1[j]:=ye[1]; x2[j]:=ye[2];
y[1]:=ye[1]; y[2]:=ye[2];
if x1[j]>0 then begin j:=j+1;
                if j>50 ^dT>0 then begin j:=j1+1;T:=T+h; go to A end;
                if j>50 ^dT<0 then begin j:=j1-1;T:=T-h; go to A end;
                go to L1
                end;
T:=te;
```

```
A:p1[j]:=p[1]; p2[j]:=p[2];
Ipp[j]:=Ipf[j]:=Iff[j]:=0;
lfo[j]:=p2[j]/x2[j];
lpo[j]:=1-u[j] × p2[j]/x2[j];
k:=j;
Q: k:=k-1; tf:=T;
I2:RK1ST(tf,p,-h,te,pe,FKT2,2);
tf:=te; p1[k]:=pe[1]; p2[k]:=pe[2];
lfo[k]:=p2[k]/x2[j];
lpo[k]:=1-u[j] × p2[k]/x2[j];
Ipp[k]:=Ipp[k+1]+(lpo[k]↑2+lpo[k+1]↑2)/2 × h;
Ipf[k]:=Ipf[k+1]+(lpo[k] × lfo[k] + lpo[k+1] × lfo[k+1])/2 × h;
Iff[k]:=Iff[k+1]+(lfo[k]↑2 + lfo[k+1]↑2)/2 × h;
p[1]:=pe[1]; p[2]:=pe[2];
print(3,6,lfo[k]);print(3,6,lpo[k]);print(3,6,Ipp[k]);print(3,6,Ipf[k]);
print(3,6,Iff[k]);punch(1);
k:=k-1;
if abs (t-te)>10-2 then go to I2;

if abs(x2[j])<0.02 ^ abs(Iff[1] - Ipf[1]↑2/Ipp[1])<0.02 then go to ut;

print(2,8,T); print(3,8,x1[j]); print(3,8,x2[j]); punch(1);

dp:=-x2[j]/k1;print(3,4,dp);
dP:=k2 × T;print(3,4,dP);
S:V:=dP - dp↑2/Ipp[1];print(3,4,V);punch(1);

if V<0 then begin dp:=sqrt(dP × Ipp[1]); V:=0 end;print(3,4,dp);punch(1);

for i:=1 step 1 until j do
begin du[i]:= (lfo[i]-lpo[i]/Ipp[1]×Ipf[1]) × sqrt(V/(Iff[1]-Ipf[1]↑2/Ipp[1])) +
                lpo[i] × dp/Ipp[1];
if abs(du[i])>1 then du[i]:=sign(du[i]);
if abs(u[i]+du[i])>1 then du[i]:=sign(du[i]) × (1 - abs(u[i])) end;
l:=1;
R:if du[l]=0 then begin l:=l+1; if l<j then go to R else go to ut end;

dT:=0;
for i:=1 step 1 until j-1 do
dT:= dT - p2[i] × du[i] × h/x2[j];print(3,4,dT);punch(1);
if abs(dT)>k3 then
begin for i:=1 step 1 until j do
du[i]:=du[i] × k3/abs(dT) end;


for i:=1 step 1 until j do
          begin u[i]:=u[i]+ du[i]; print (3,8,u[i]); punch(1)
          end;
          punch(1);
   y[1]:=x1[1]; y[2]:=x2[1];p[1]:=p1[j];p[2]:=p2[j];
   u[j+1]:=u[j+2]:=u[j]; j1:=j;
   go to P;

ut:punch(8)

end
```

Litteraturförteckning

Bryson, A.E. and Denham, W.F.: A steepest ascent method for solving optimum programming problems (Trans. ASME ser. E 29:247-57)

Bryson, A.E.: Optimal programming and control (ur Proceedings IBM scientific computing symposium control theory and applications, IBM corporation 1966)

Noton, A.R.M.: Introduction to variational methods in control engineering.(Pergamon Press, London, 1965).

Kelley, H.J.: Method of gradients, chap. 6 Optimization techniques (ed. Leitmann) (Academic Press, New York, 1962)

Almstedt, T.A. and Gibson, D.B.: Solution of a problem of optimal control by the method of steepest descent (IBM reports 1962)

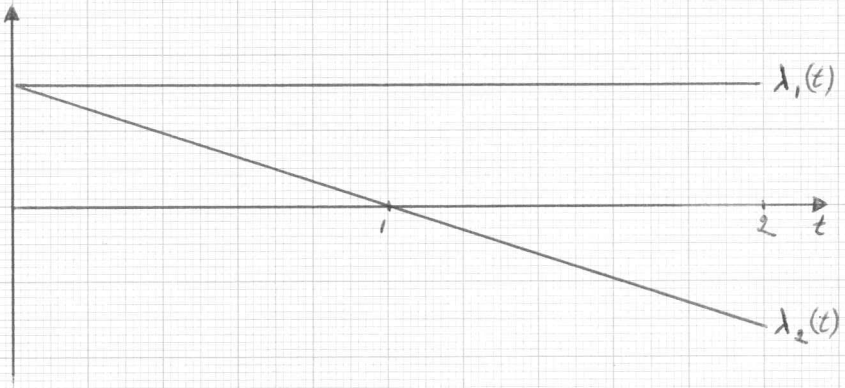
Bryson, A.E., Denham,W.F. and Dreyfus, S.E.: Optimal programming problems with inequality constraints (AIAA Journal vol. 1 no. 11 November 1963 and vol.2 no. 1 January 1964)

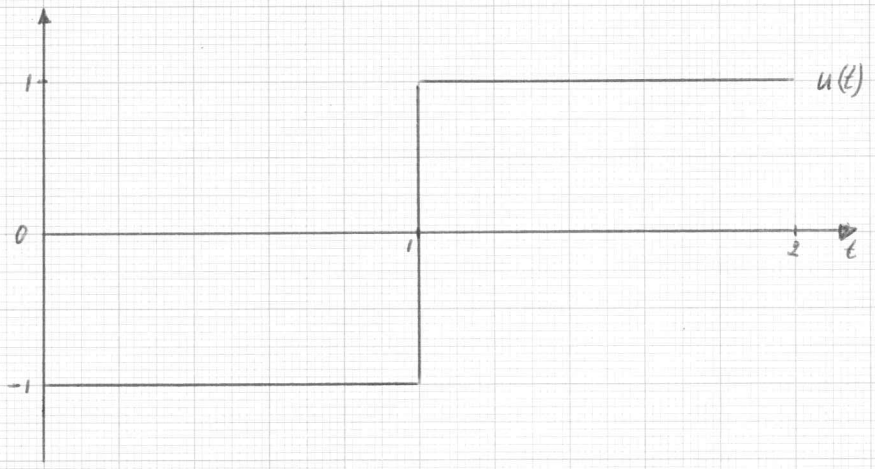
Balakrishnan, A.V. and Neustadt, L.W.: Computing Methods in optimization problems

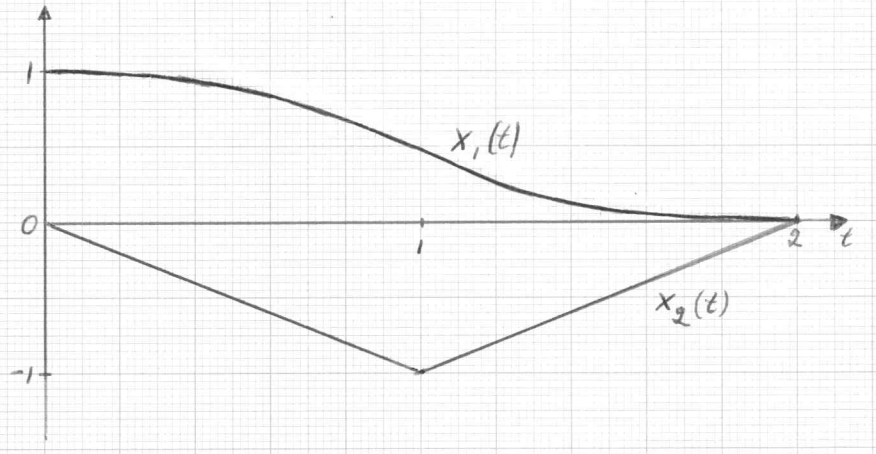
Exakt lösning för dubbelintegratorn.

T = 2.

Diagram 1



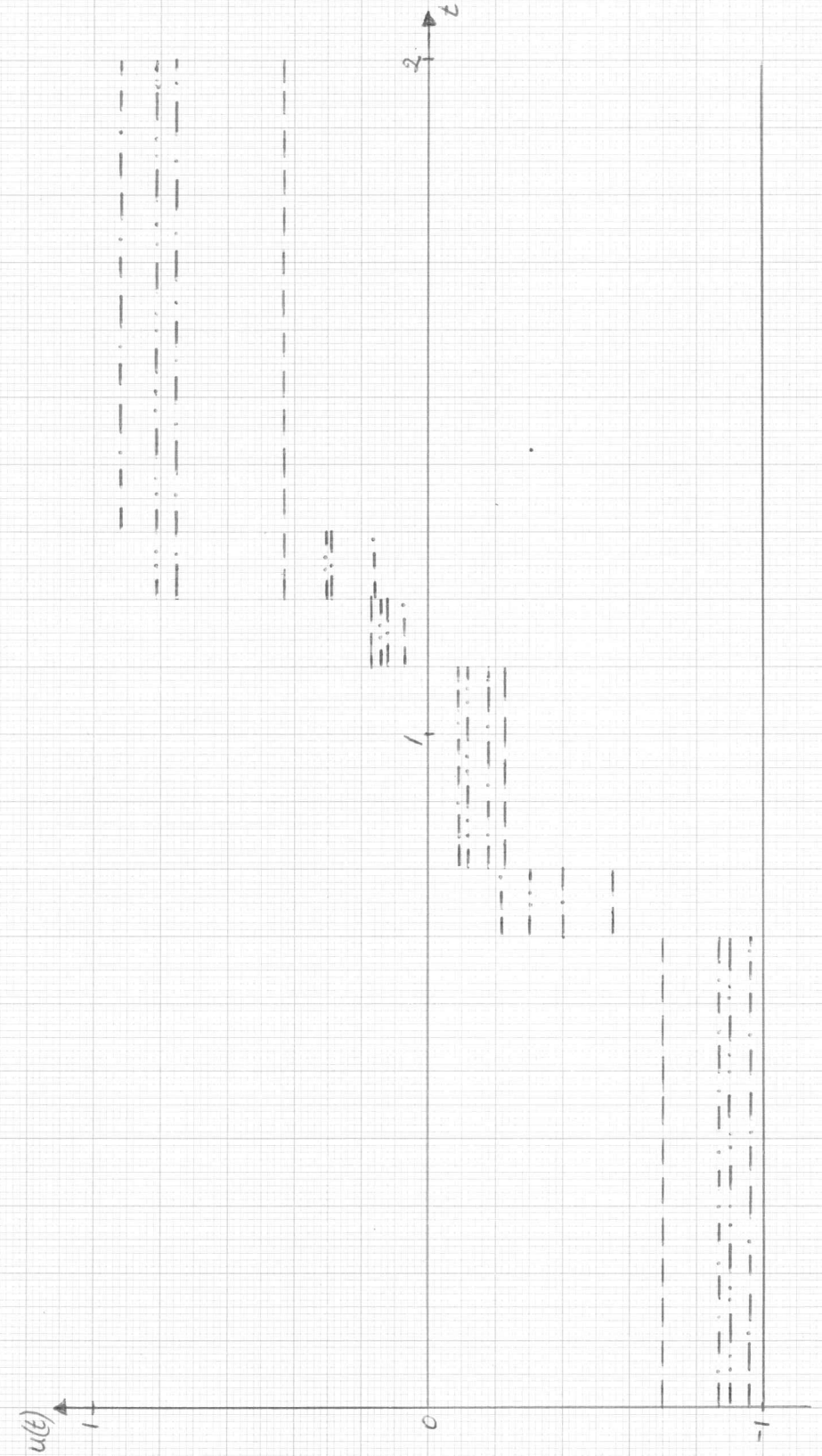




SIS 732501 - 514 A4 - 1×1 mm
ESSELTE
4441

Dubbelintegratorn. kl = 15. $u^0(t) = -1$

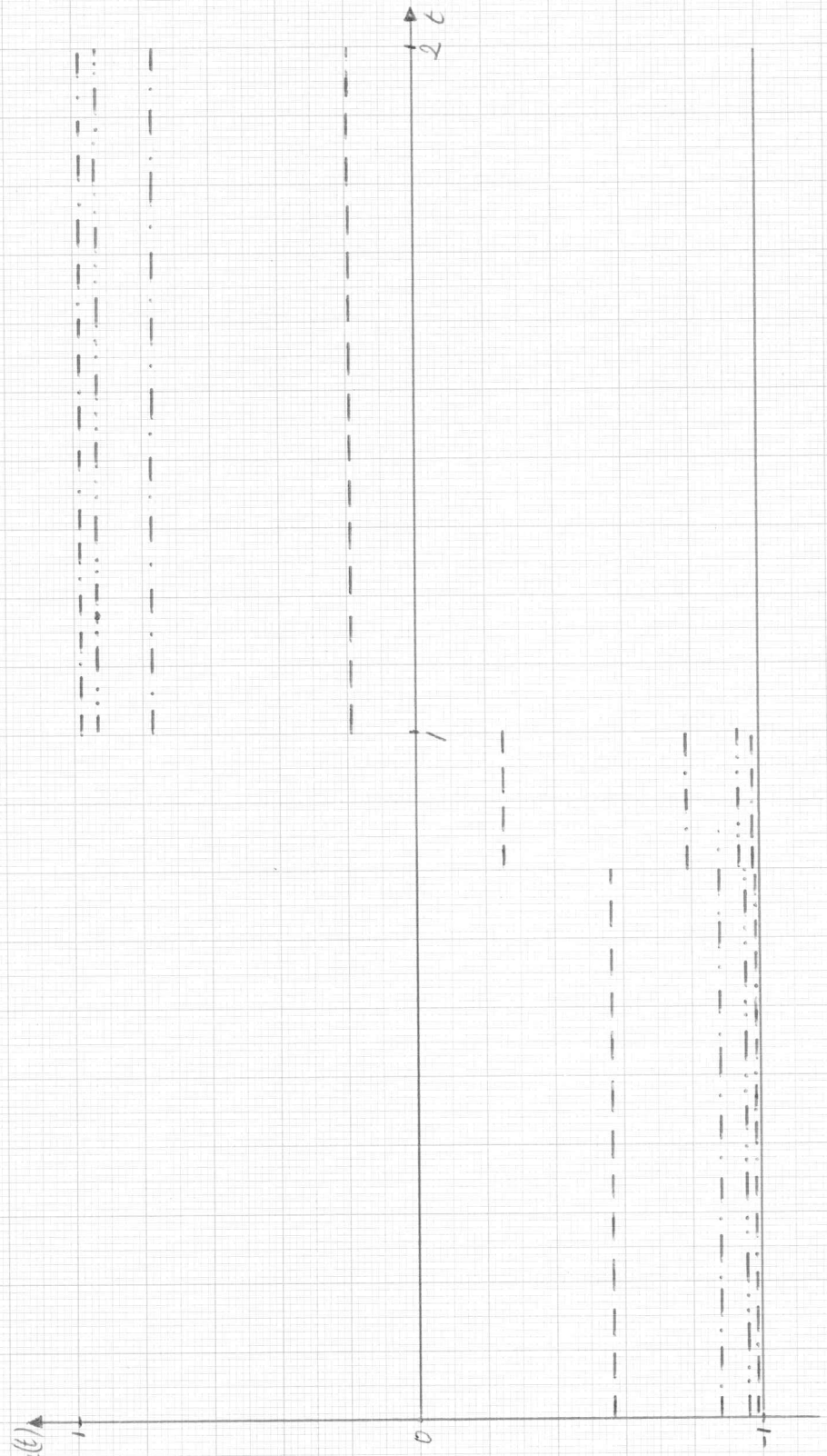
Diagram 2



732501 - 514 A4 - 1×1 mm
ESSELTE
4441

Dubbelintegratorn. kl =20. $u^0(t) = -1$

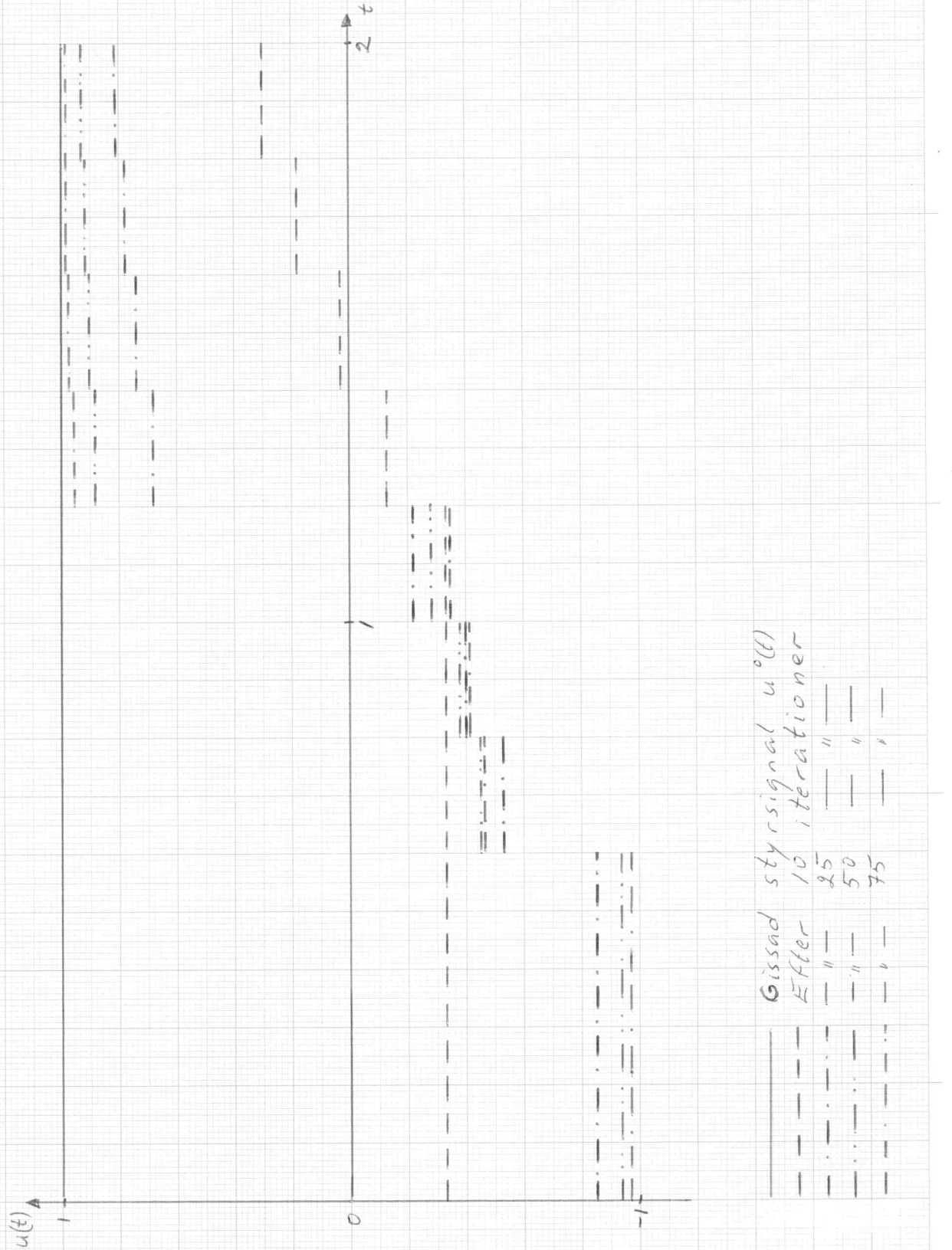
Diagram 3



73 25 01 - 514 A4 - 1 x 1 mm
ESSELTE
4441

Dubbelintegratorn. kl = 20. $u^0(t) = 1$

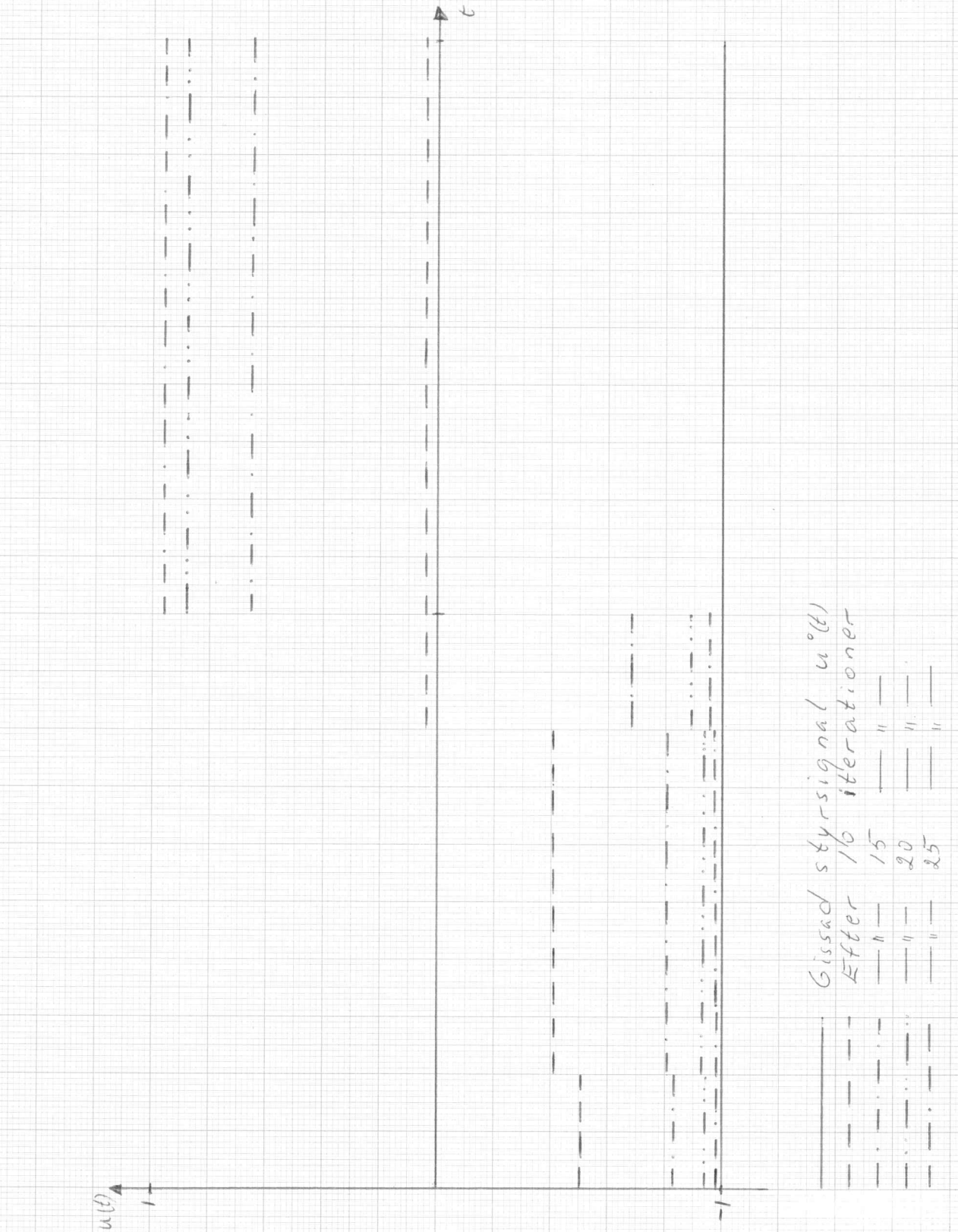
Diagram 4



73 2501 - 514 A4 - 1 x 1 mm
ESSELTE
4441

Dubbelintegratorn. kl=25. $u^0(t) = -1$

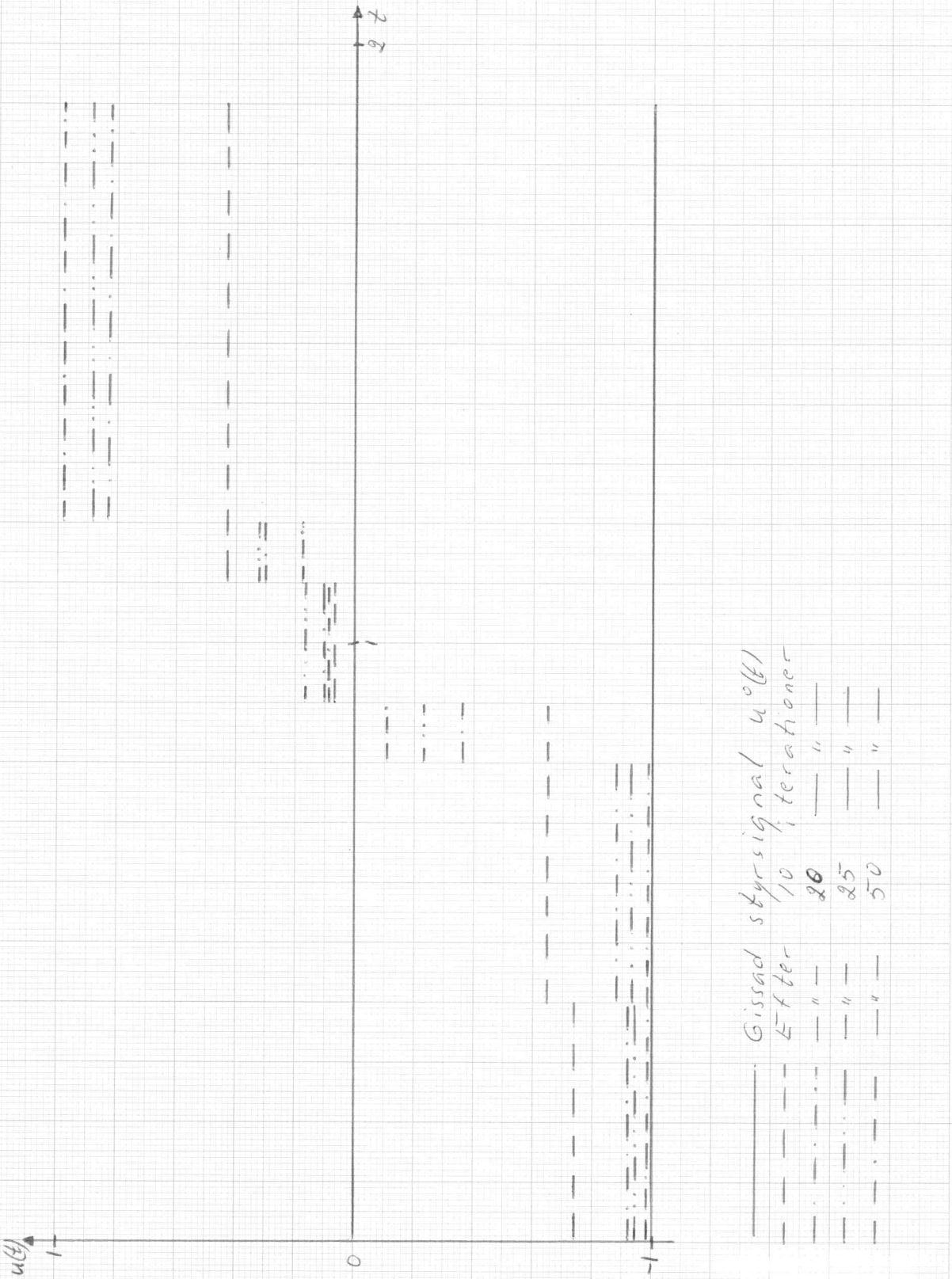
Diagram 5



SIS 732501 - 514 A4 - 1×1 mm
ESSELTE 4441

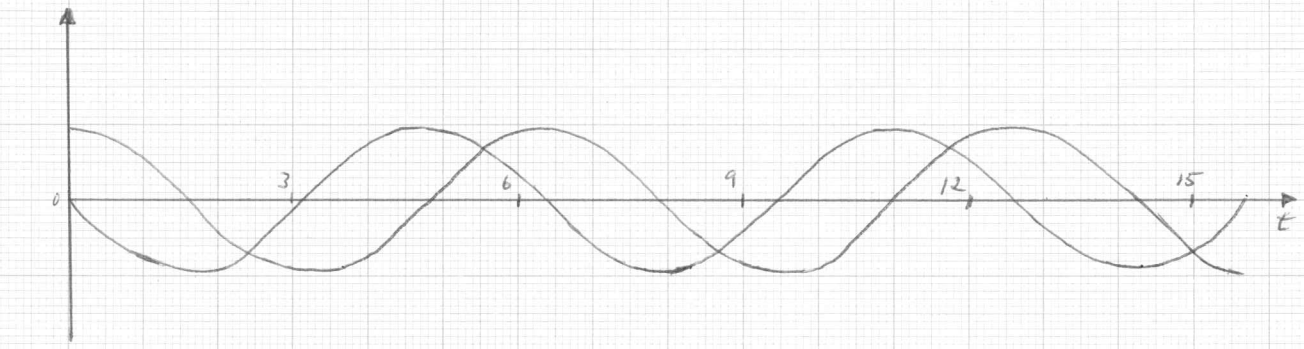
Dubbelintegratorn. kl = 20. $u^0(t)=-1$
T = 1,9.

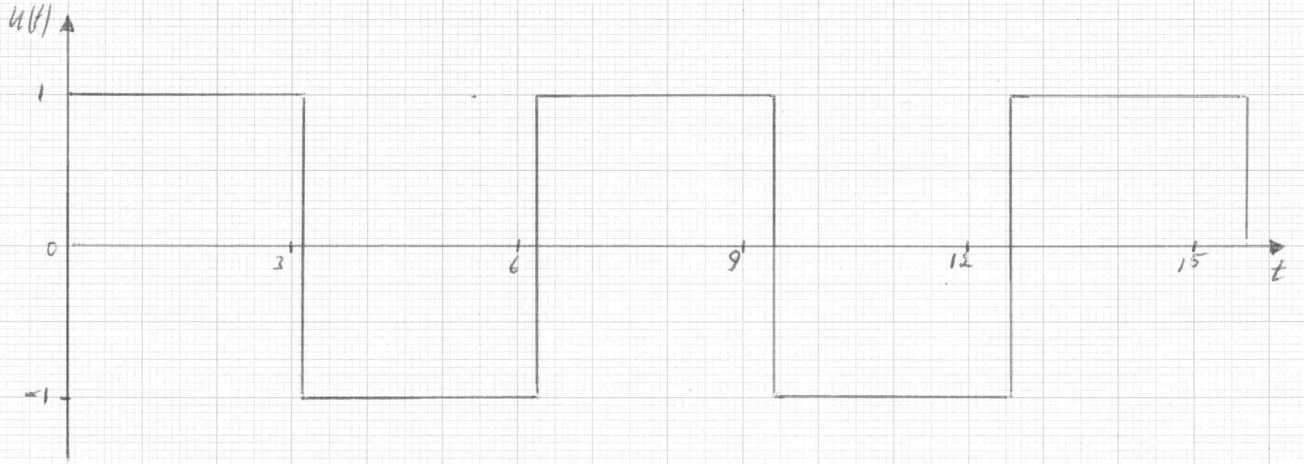
Diagram 6

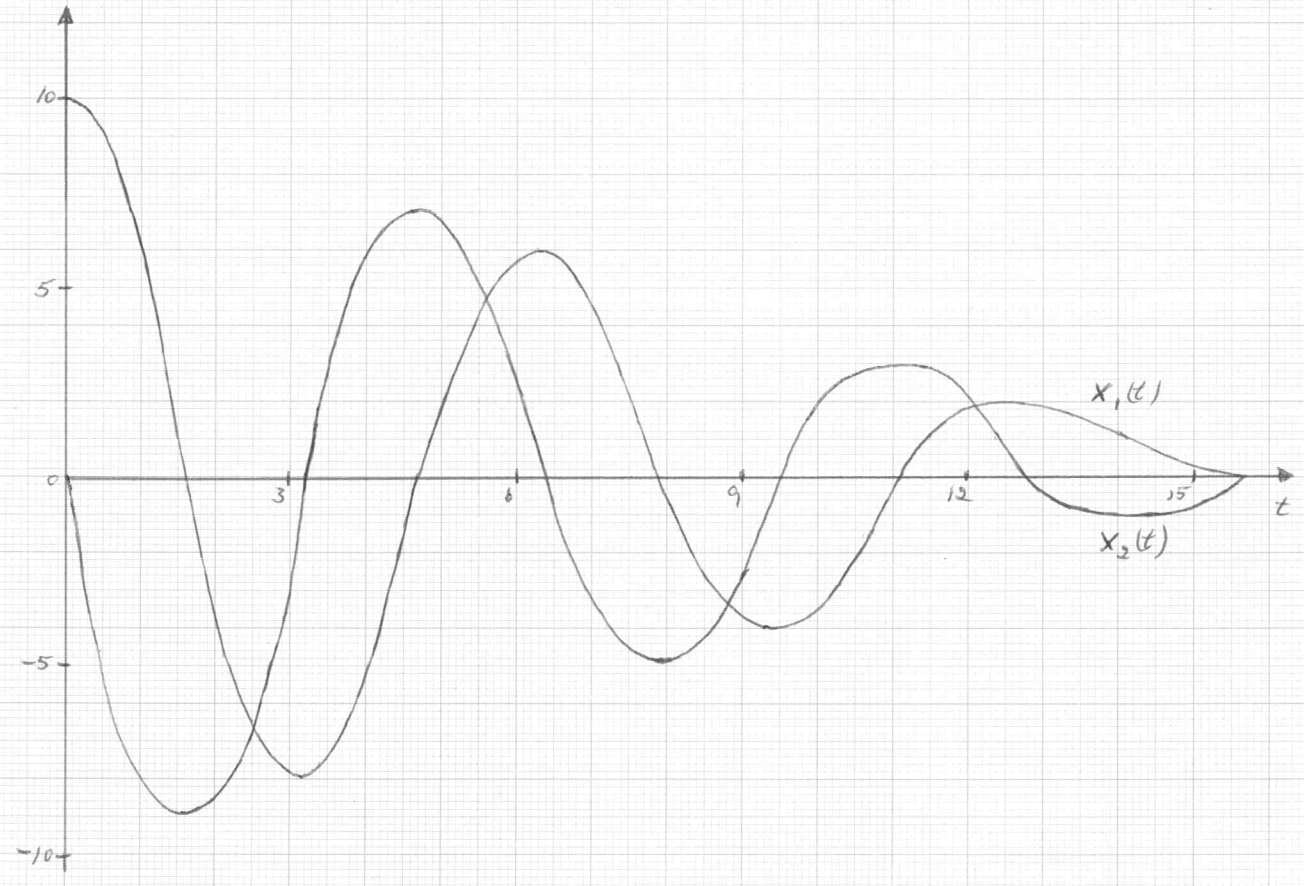


Oscillatorn. Exakt lösning.

Diagram 7



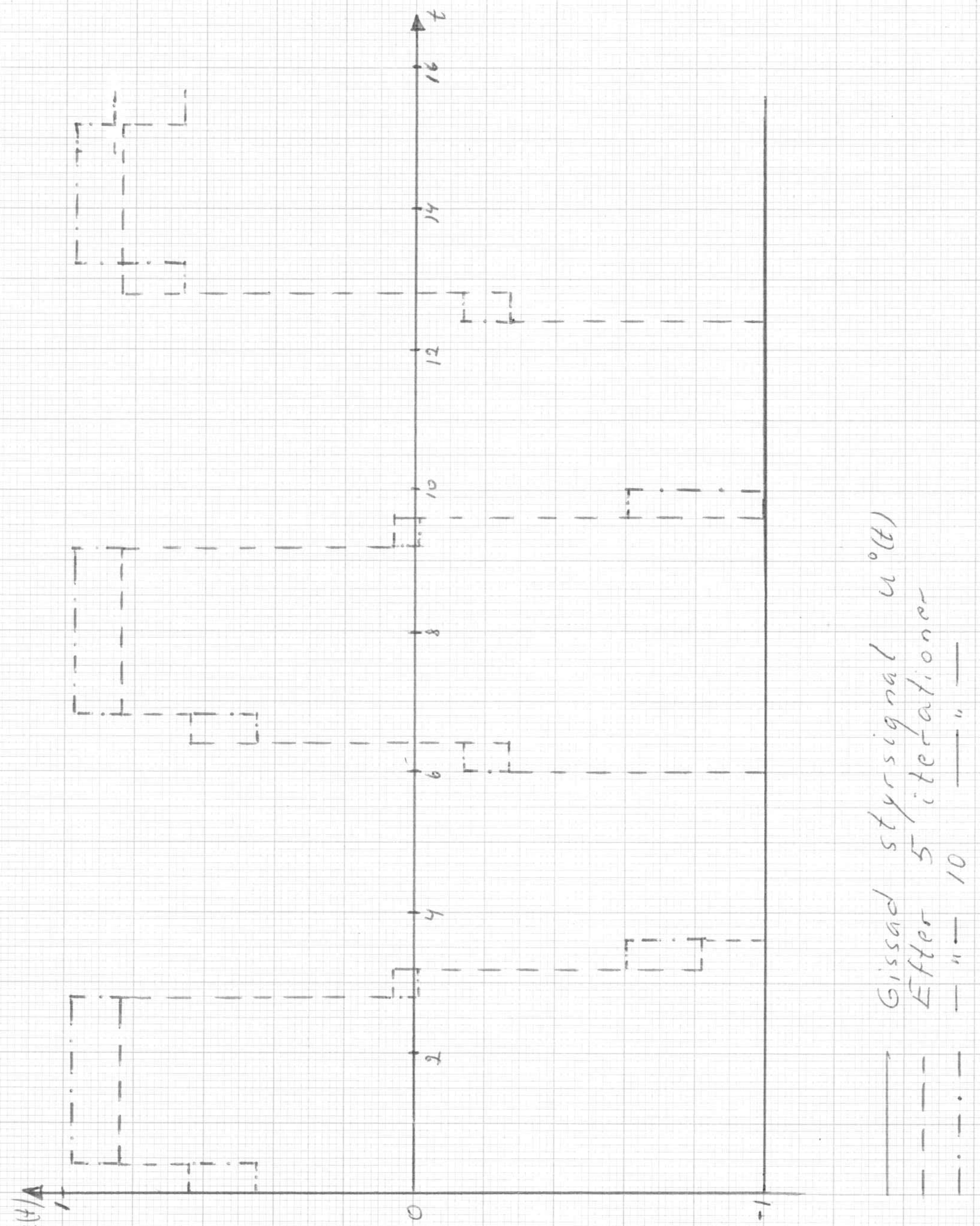




73 25 01 - 514 A4 - 1×1 mm
ESSELTE
4441

Oscillatorn. kl =2. T = 15,6.

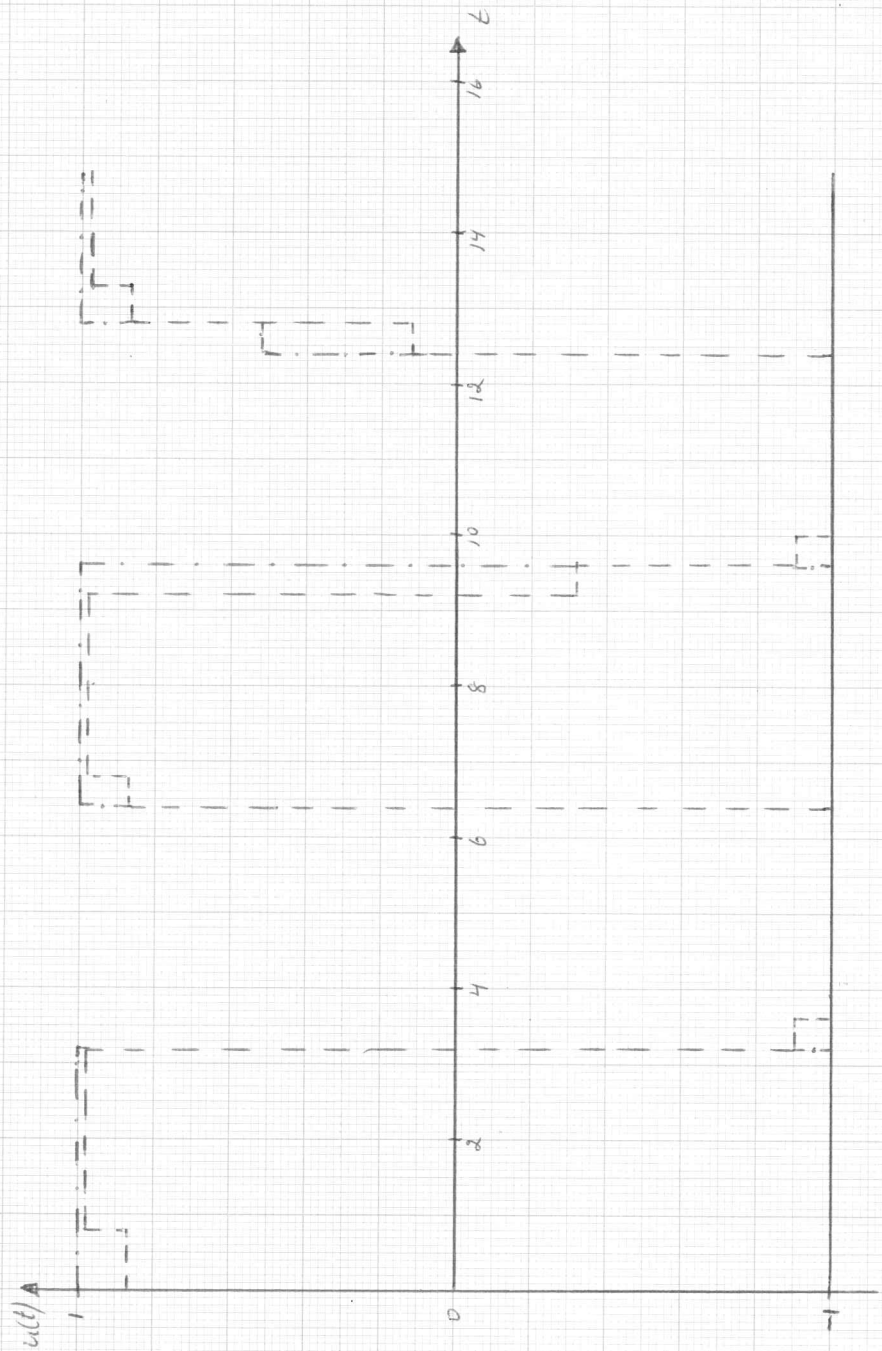
Diagram 8



SIS 732501 - 514 A4 - 1×1 mm
ESSELTE 4441

Oscillatorn. kl = 2. T = 14,8.

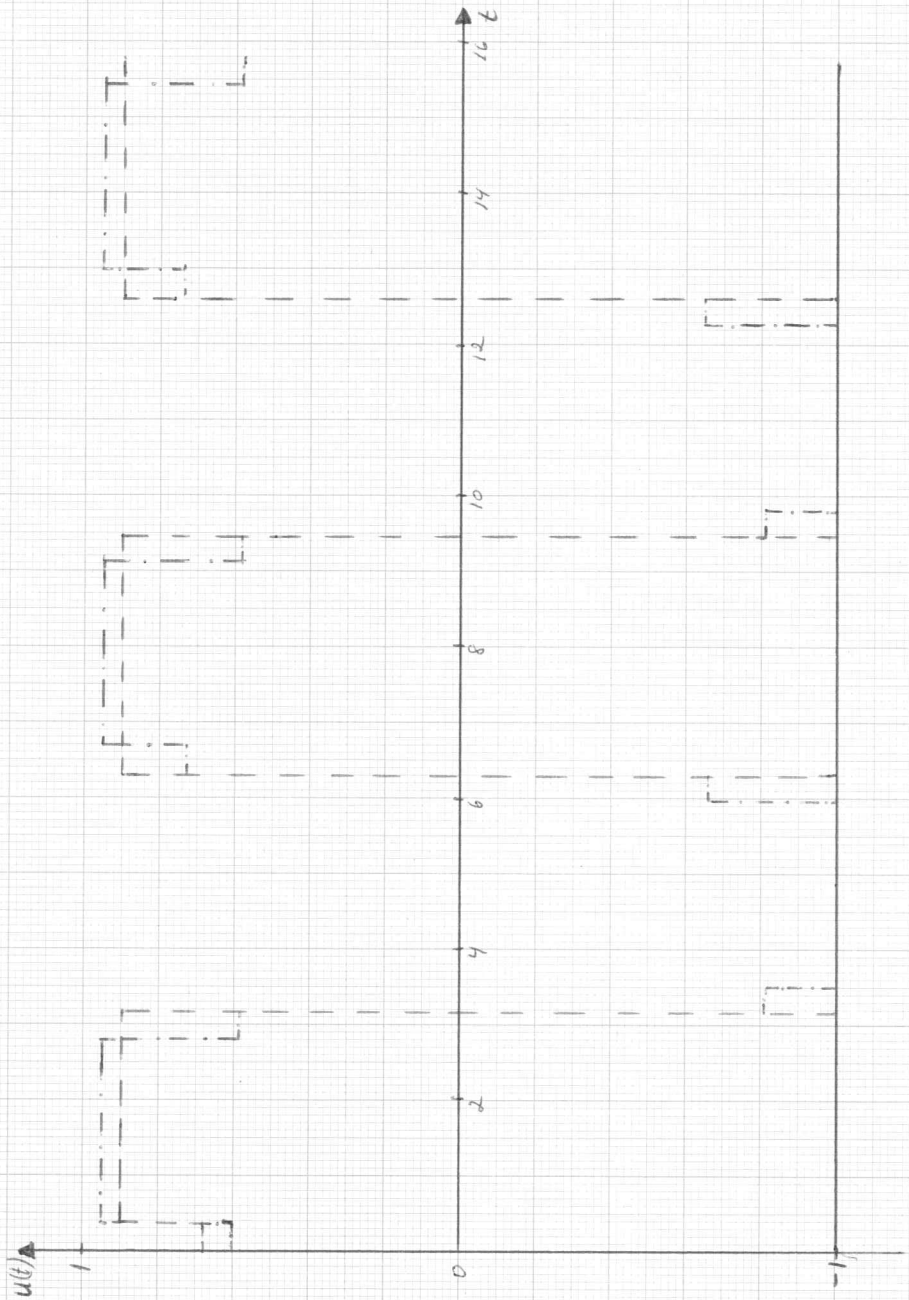
Diagram 9



73 25 01 - 514 A4 - 1×1 mm
ESSELTE
4441

Oscillatorn. kl = 5. T = 15,75.

Diagram 10

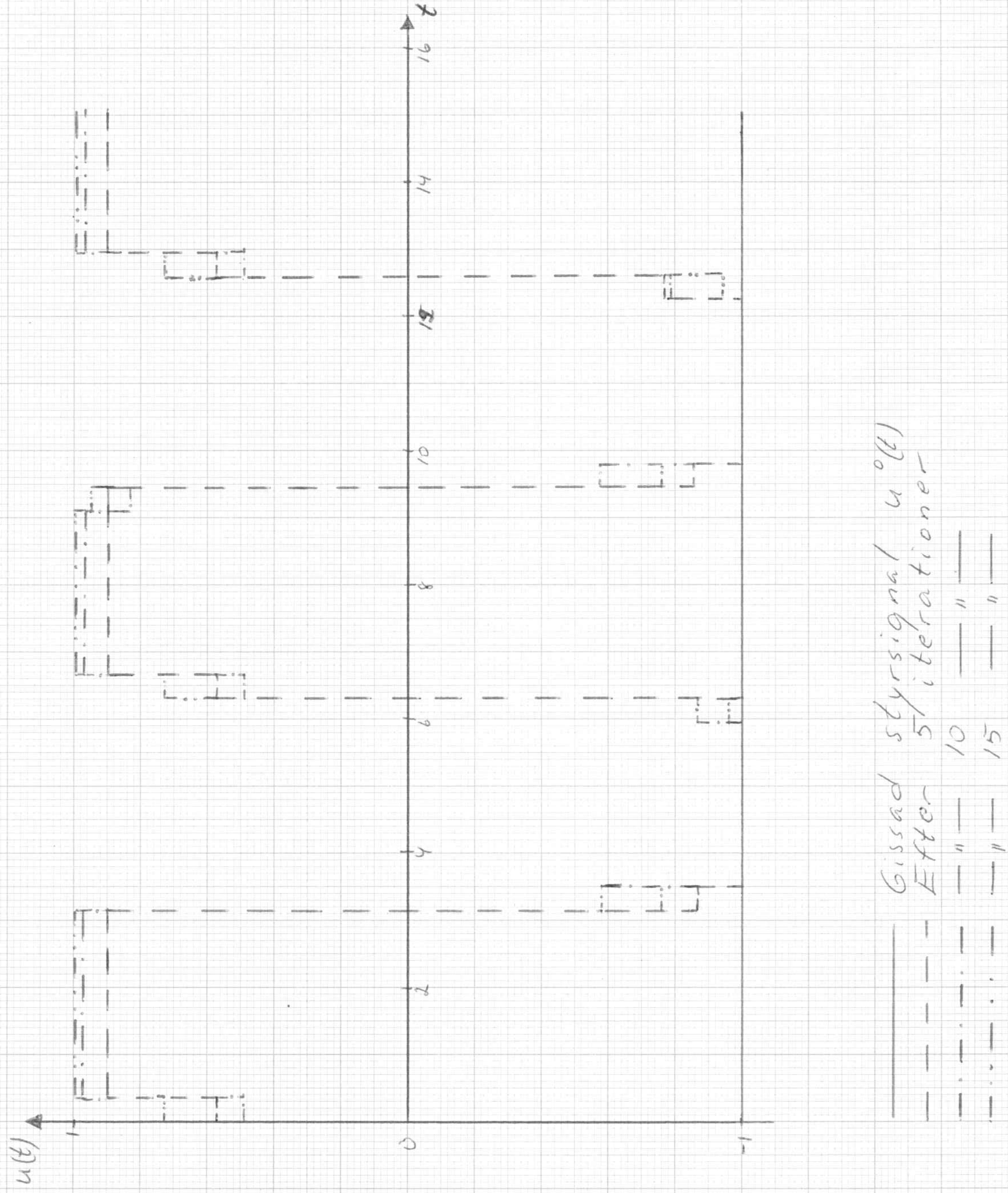


Gissad styrsignal u°(t)
Efter 5 iterationer
— " — 10 — " —

732501 - 514 A4 - 1×1 mm
ESSELTE
4441

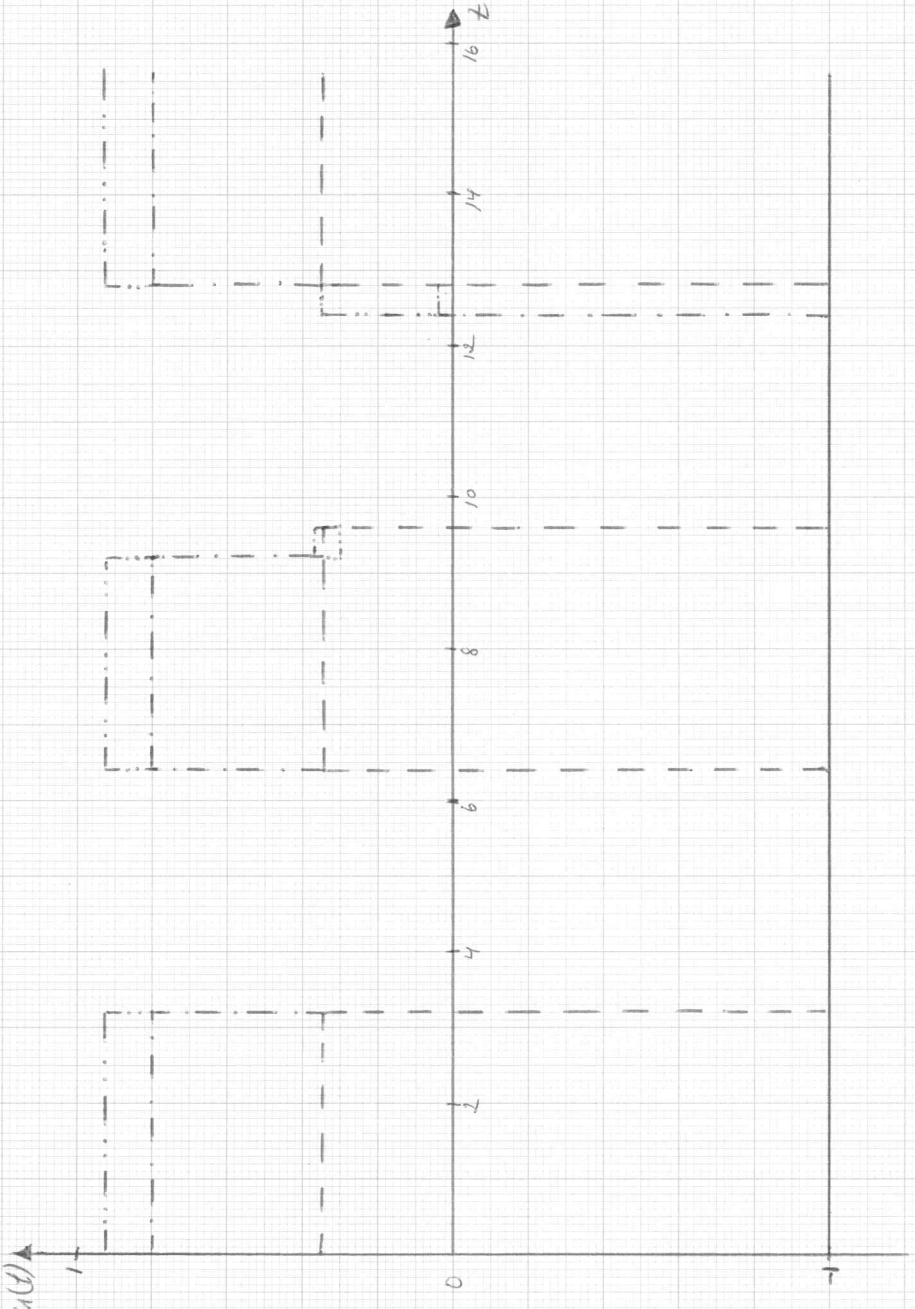
Oscillatorn. kl = 5. T = 15,05

Diagram 11



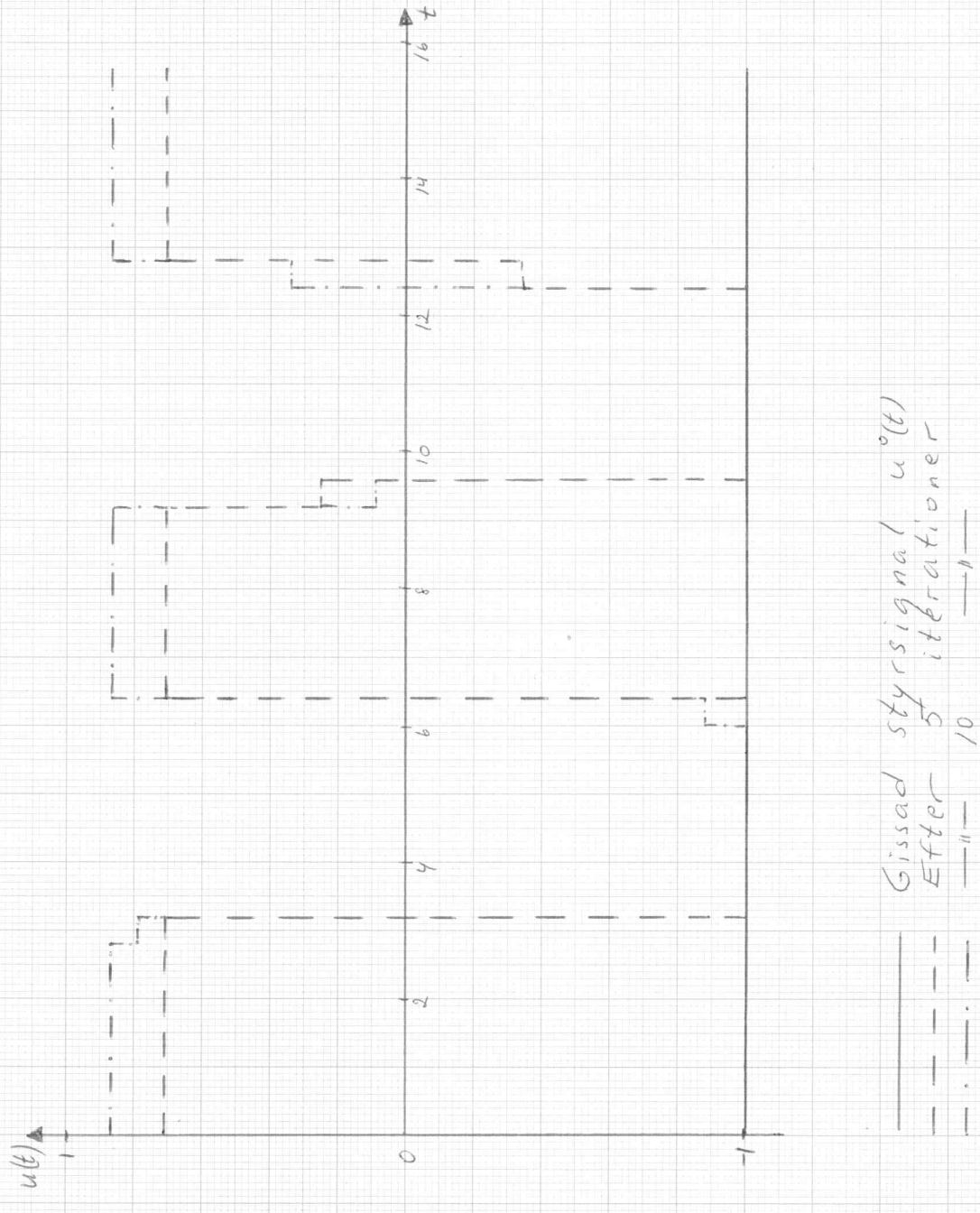
Oscillatorn. k1 = 10. T = 15,6

Diagram 12



Oscillatorn. k1 = 25. T = 15,6

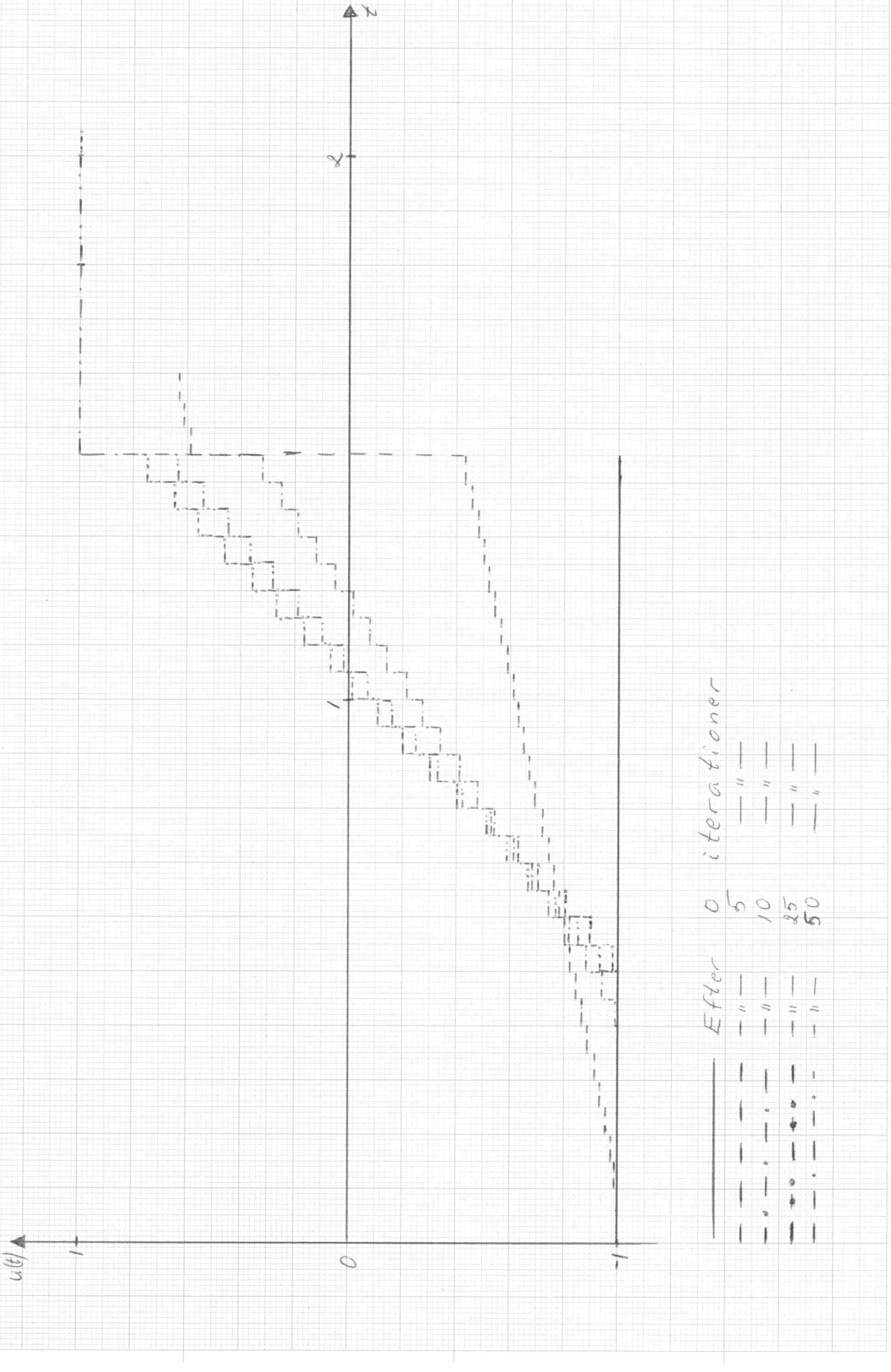
Diagram 13



732501 - 514 A4 - 1×1 mm
ESSELTE
4441

Minimaltidsproblemet för dubbelintegratorn.

Diagram 14



73 25 01 - 514 A4 - 1 x 1 mm
ESSELTE
4441