CODEN: LUTFD2/(TFRT-5238)/1-119/(1980)

CONDIS - EN VIDAREUTVECKLING AV SIMULERINGSPAKETET COMBINEDSIMULATION

LENNART MÅNSSON

INSTITUTIONEN FÖR REGLERTEKNIK
LUNDS TEKNISKA HÖGSKOLA
APRIL 1980

Organization
LUND INSTITUTE OF TECHNOLOGY
Department of Automatic Control
P O Box 725
S-220 07 LUND 7, Sweden

Document name
MASTER THEIS

Date of issue
APRIL 1980

CODEN: LUTFD2/(TFRT-5238)/1-119/(1980)

Author(s)
Lennart Månsson

Sponsoring organization

Title and subtitle
CONDIS - En vidareutveckling av simuleringspaketet Combinedsimulation

(CONDIS - A development of the simulation package Combinedsimulation)

A4 A5

Abstract

Extensions of the simulation package Combinedsimulation is given in the report. Combinedsimulation is suited for simulation of differential and difference equations in combination with events.

There are two main extensions done in Condis. First a fourth order predictor corrector method is used for the integration. Secondly, new ways to continue a run is done by changing the error and exit handling.

Key words A4 A5

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
Swedish

ISSN and key title

ISBN

Recipient's notes

Number of pages
119

Price

Security classification

Distribution by (name and address)

CONDIS

EN VIDAREUTVECKLING
AV SIMULERINGSPAKETET
COMBINEDSIMULATION

EXAMENSARBETE I ÄMNET
REGLERTEKNIK
UTFÖRT VID
FOA SEKT 221
AV

LENNART MÅNSSON

CONDIS, En vidareutveckling av simuleringspaketet Combinedsimulation.

Lennart Månsson

Antal sidor 119

Sammanfattning

I denna rapport beskrivs utvidgningar av simuleringspaketet COMBINEDSIMULATION (Keld Helsgaun, Roskilde Universitet, Danmark)

COMBINEDSIMULATION är en Simula-klass avsedd för simulering av dynamiska system i en miljö med diskreta ändringar av tillstånd och differentialekvationer. Klassen innehåller alltså verktyg för att lösa ordinära differentialekvationer med användning av numerisk integration. De diskreta ändringarna av tillstånd och differentialekvationer kan planeras med normala begrepp från SIMULATION, vilket innebär att komplicerade händelsestyrda simuleringar kan utföras.

Huvuddelen av arbetet har resulterat i en Simula-klass kallad CONDIS.

CONDIS är avsett att användas på DEC-10 systemet på Stockholms Datamaskincentral QZ och använder en del biblioteksrutiner därifrån.

CONDIS skiljer sig från COMBINEDSIMULATION främst i två avseenden. Dels är en ny alternativ integrationsmetod införd, nämligen en fjärde ordningens prediktor-korrektor-metod med diskontinuitetsdetektion, dels finns i CONDIS möjlighet att efter vissa fel som upptäcks av programmet få studera och ändra parametrar för att om möjligt kunna fortsätta körningen.

INNEHÅLLSFÖRTECKNING

0 LÄSANVISNING

För att kunna följa med i framställningen av kap 2 och 3, dvs de kapitel som behandlar integrationsmetoder, rekommenderas läsaren att studera dokumentationen av COMBINEDSIMULATION (ref [2]) först. Finns denna inte tillgänglig kan kap 8 ge en introduktion till grundbegreppen

En läsare som inte är intresserad av hur CONDIS fungerar utan inriktar sig på hur man använder programmet börjar med fördel med kap 8. Kap 2 och 3 och bilagorna 1 och 2 kan förmodligen överhoppas utan större saknad. Som komplement kan COMBINED-SIMULATION: INTRODUKTION och BRUGERHÅNDBOG (ref [2.1] resp [2.2]) vara nyttiga.

I en framtid planeras kap 4 tom 8 och bilagorna 3, 4 och 5 utges i en speciell rapport "CONDIS Användarhandledning".

1 INLEDNING

Bakgrunden till att detta arbete blivit utfört är önskemål på inst 220, FOA 2, om ett nytt simuleringspaket. Paketet är avsett att användas vid de blandade simuleringar som utförs vid institutionen. Med ´blandade simuleringar´ avses simulering av dynamiska system där det även förekommer diskreta ändringar av systemvariabler eller av de differentialekvationer som styr systemet.

Som utgångspunkt för mitt arbete fanns ett arbetsblad med önskemål om ett nytt paket som i huvudsak innehöll följande punkter:

1. Snabbt och effektivt.
2. Användarvänligt.
3. Möjligheter att behandla diskontinuiteter i integranderna.
4. Integrerade variabler skall nås med naturliga namn, inte konstruktioner av typ ´variabel[1]´ el dyl.
5. Möjligheter att fortsätta körningar som har avbrutits pga att integrationsnoggrannheten inte kunnat uppfyllas.
6. Möjligheter att ha olika tidssteg för olika kontinuerliga objekt.
7. En lätthanterlig och väl fungerande utmatning.
8. De kontinuerliga objekten skall deklareras som en speciell klass.
9. Möjlighet till fast integrationssteg.

Grundläggande förutsättning var att paketet skulle skrivas i Simula för att användas på QZ:s DEC-10 system.

På institutionen fanns redan ett antal simuleringspaket som CSMP, CADSIM och COMBINEDSIMULATION.

CADSIM (se [1]) var tidigare det mest använda simuleringshjälpmedlet vid institutionen.

COMBINEDSIMULATION (se [2]) hade institutionen precis erhållit från Roskilde Universitetcenter.

Som inledande uppgift, för att få någon inblick i problemställningarna och i användningen av DEC-10-systemet, skrev jag några program för simulering av projektiler avfyrade med begynnelsehastighet rakt uppåt. Jämförelser gjordes mellan program skrivna med CADSIM och med COMBINEDSIMULATION avseende strukturering av användarprogrammen och effektiviteten i beräkningarna. Resultaten av dessa tester finns i bilaga 1.

Dessutom studerades åtskilliga artiklar om olika integrationsmetoder som kunde vara intressanta i sammanhanget. Som exempel kan nämnas

"Solving nonstiff ordinary differential equations- The state of art", [3].

"Subroutine HPGC and DHPGC", [4].

Utvidgningen av dessa till:
"DHAMDI a FORTRAN subroutine to integrate a set of first order, ordinary differential equations containing discontinuities", [5].

"Numerical integration methods for the solution of ordinary differential equations", [6].

"Integration across discontinuities in ordinary differential equation using power series", [7].

"Linear methods for ordinary differential equations: Method formulations, stability and the method of Nordsieck and Gear", [8].

I samråd med min handledare forskare Göran Lyman och laborator Inga Nordström, båda inst 220, FOA 2, beslöts därefter att det fortsatta arbetet skulle utgå från Helsgauns paket COMBINEDSIMULATION.

COMBINEDSIMULATION ansågs i stort vara ändamålsenligt för institutionens behov. Flera av de på sidan 5 uppställda önskemålen är tillgodosedda och dessutom talar även följande punkter för COMBINEDSIMULATION:

Då aktuella tillämpningar ofta kräver att vissa funktioner som ingår i differentialekvationerna specifieras som uppslagningstabeller, inte som analytiska uttryck, är metoder som bygger på funktionalmatriser inte lämpliga.

Lämpligt sätt att ange differentialekvationerna är

$$\dot{y}_i(t) = f_i(t, y_1(t), \ldots, y_i(t), \ldots, y_n(t))$$

där $y_i(t)$ är variablernas värde vid tiden t.

COMBINEDSIMULATION ansågs ha en vettig struktur för att kombinera de diskreta händelserna med de kontinuerliga objekten.

Möjligheterna till klar och lättläst struktur i användarprogrammen ansågs goda i COMBINEDSIMULATION.

Jämförelser mellan COMBINEDSIMULATION och CADSIM avseende snabbhet och effektivitet visade att COMBINEDSIMULATION i varje fall inte uppvisade sämre resultat.

Nackdelarna med COMBINEDSIMULATION ansågs ligga i begränsningar i integrationsmetoden. Användningen av uppslagningstabeller medför nämligen att funktionsuttrycken som integreras innehåller mer eller mindre kraftiga diskontinuiteter. Detta medför behovet av en integrationsmetod som klarar sådana diskontinuiteter. Dessutom finns i COMBINEDSIMULATION mycket små möjligheter att fortsätta exekveringen efter felavbrott t ex orsakade av att begärd integrationsnoggrannhet inte kunde erhållas.

På grundval av detta inriktades det fortsatta arbetet på alternativa integrationsmetoder och på dialog i samband med uppträdande fel i programmet. Speciellt ansågs en integrationsmetod av typ DHAMDI med möjlighet till behandling av diskontinuiteter vara värdefull att införa.

Som första uppgift konstruerades en möjlighet att i COMBINEDSIMULATION byta mellan olika enstegs integrationsmetoder i huvudsak avsett för varianter av Runge Kutta . Detta arbete ufördes dels för att testa en metod RK4F (Fjärde ordningens Runge-Kuttametod enl Fehler), dels för att få studera mer i detalj hur integrationen i COMBINEDSIMULATION fungerar.

Huvudidén är att bryta ut de delar av integrationen som är metodspecifika i virtuella procedurer, som användaren sedan kan omdefiniera till den metod som önskas.

Arbetet är beskrivet i kap 2, den modifierade källkoden av ´themonitor´, det enda avsnittet av COMBINEDSIMULATION som behövde ändras finns i avsnitt 8.1

Nästa uppgift bestod i att med utgångspunkt från metodbeskrivningarna av HPGC och DHAMDI (se [4] resp [5]) implementera en liknande metod i COMBINEDSIMULATION. Detta arbete är beskrivet i kap 3.

Åtgärder för att kunna rätta felaktigheter, ändra integrationsparametrar och fortsätta körningen efter upptäckta fel i programmet finns beskrivet i kap 4. Där finns också beskrivningen av en viss dialog vid start av programmet.

I kap 5 finns en ny klass ´object´ som avser att underlätta strukturering för användaren.

De arbeten som är beskrivna i kap 3,4 och 5 har resulterat i ett komplett simuleringspaket som i fortsättningen benämns ´CONDIS´.

Kap 6 behandlar en separatkomplilerad klass ´filequ´, som underlättar behandlingen av filer, både för CONDIS och för användaren av CONDIS. Klassen kan också användas i andra sammanhang där systemprogrammeraren behöver tillgång till referenser på de filer som användarprogrammeraren har skapat.

CONDIS utnytttjar ett antal externkompilerade procedurer och klasser Detta innebär även att användaren kan utnyttja dessa programdelar. Under DECsystem-10 SIMULA version 4 måste de även deklareras i användarprogrammet. Vilka externa klasser och procedurer som finns och hur de skall anges under olika SIMULA system finns beskrivet i kap 7.

Kap 8 innehåller en kort användarhandledning till CONDIS.

I kap 9 slutligen finns källkoden till de olika program som diskuterats.

2 ANVÄNDNING AV ANDRA INTEGRATIONSMETODER I COMBINEDSIMULATION

För att möjliggöra användning av olika integrationsprocedurer i COMBINEDSIMULATION har koden ändrats så att den innehåller två virtuella procedurer, take_a_step och act_error. Dessa procedurer innehåller det som är specifikt för den metod som används i orginalversionen av COMBINEDSIMULATION, nämligen Runge-Kutta-England. Genom att omdefiniera dessa procedurer kan man möjligöra användning av annan procedur. Metoden är främst anpassad till användning av någon annan Runge-Kutta-variant än den enl England som finns i COMBINEDSIMULATION.

Huvudidén är att take_a_step vid anrop har tillgång till en variabel "dtnow" som anger hur långt det aktuella integrationssteget skall vara. Vid återhoppet skall attributet "ds" till varje "variable" i COMBINEDSIMULATION:s mening innehålla tillskottet vid stegets slut. Någon gång i take_a_step skall ett anrop av error_test göras för att avgöra om det aktuella steget kan godkännas ur noggrannhetssynpunkt. Error_test anropar i sin tur act_error som skall innehålla algoritmer för beräkning av dels en feluppskattning i steget, dels det maximalt tillåtna felet. Act_error kommer att anropas en gång för varje "variable" som är ´aktiva´ dvs är startade med kommandot ´start´.

Error_test sätter en global boolesk variabel error_flag till TRUE om steget inte kan godkännas utan måste tas om med kortare steglängd. Denna variabel kan utnyttjas av användaren i take_a_step.

De olika variablerna har bl a följande attribut som kan vara intressanta i sammanhanget:

state rate abserror relerror oldstate ds dsh a1...a5

Variablerna ligger i en länkad lista. Det finns en pekare till den första variblen som heter "firstvar" och varje variabel har en pekare till nästa "sucvar". Detta medför att man kan utföra en beräkning för samtliga variabler med följande konstruktion:

```
var :- firstvar ;
WHILE var =/= NONE DO INSPECT var DO
BEGIN
  (
  ( SATSER ATT UTFÖRAS PÅ VARJE VARIABEL. ATTRIBUTEN
  ( ÅTKOMLIGA UTAN PUNKTNOTATION.
  (
  var :- sucvar;
END;
```

ANVÄNDNING AV ANDRA INTEGRATIONSMETODER ...

De beteckningar som förekommer i följande avsnitt förutsätter att systemet som skall integreras är definierat enligt:

$$\dot{y}_i = f_i(t,y(t))$$

Tillståndsvariablernas värde vid stegets början ligger i attributet oldstate. State kan användas för lagring av värdet i mellanled, oldstate får ej ändras. För att räkna ut f(t,y(t)) för samtliga variabler skall först "Time" tilldelas värdet t och state skall ha värdet y(t). Skrives sedan satsen "Resume(firstcont);" så beräknas f(t,y(t)) och läggs i attributet "rate" till varje variabel.

Vid anrop av take_a_step finns följande värden tillgängliga:

$f_i(t_o,y(t_o))$	i	rate
$y(t_o)$	i	oldstate
tidp för stegets slut	i	nexttime
t_o	i	Time
aktuell steglängd	i	dtnow

Av dessa får dtnow och dtnext inte ändras. Rate får bara ändras via satsen resume(firstcont).

Attributen dsh,h,a1,...,a5 kan användas för lagring av koeffcienter och mellanresultat. Dock skall observeras att a4 ändras vid anrop av error_test.

För att interpoleringsalgoritmerna vid tillståndshändelser och "sampling" skall fungera måste följande gälla då proceduren lämnas:

dsh skall innehålla tillskott vid $\frac{dtnow}{2}$

a1 ------"-------- $\frac{dtnow}{2} * f(t_o,y(t_o))$

a5 ------"-------- $\frac{dtnow}{2} * f(t_o + \frac{dtnow}{2}, y(t_o + \frac{dtnow}{2}))$

a4 ------"-------- uppskattat fel för variabeln i steget

h ------"-------- $\frac{dtnow}{2}$

Proceduren act_error skall dels tilldela a4 en uppskattning av felet i steget, dels tilldela temp ett värde på det maximalt tillåtna steget. Till det senare används attributen relerror och abserror.

För att komma åt de olika variablerna måste satserna i båda procedurerna stå i ett inspect-block "INSPECT themonitor DO BEGIN".

För att take_a_step och act_error skall kunna omdefinieras av användaren måste de vara VIRTUAL-deklarerade och deklarationen av dem måste finnas i det yttersta blocket av COMBINED-SIMULATION. Procedurerna take_a_step och act_error som finns i denna modifierade version av COMBINEDSIMULATION avser samma metod, Runge-Kutta-England som finns i orginalversionen.

Källkoden till denna modifierade version av COMBINEDSIMULATION finns i avsnitt 8.1. De delar av den som exakt överensstämmer med orginalversionen är bara skissade med klass-, block- eller procedurhuvuden.

I take_a_step kan noteras att den tidigare nämnda variabeln error_flag är här använd för att spara in lite på räkningarna efter ett underkänt steg.

```
PROCEDURE take_a_step;
! INTEGRATIONS-PROCEDUREN FÖR RK-ENGLAND;
INSPECT themonitor DO
BEGIN
  h:= 0.5 * dtnow;

  IF NOT error_flag THEN
  BEGIN
    var :- firstvar;
    WHILE var =/= NONE DO INSPECT var DO
    BEGIN
      a1:= h* rate;
      state := oldstate + 0.5*a1;
      var :- sucvar;
    END;
  END *** IF NOT ERROR_FLAG ***
  ELSE
  BEGIN
    var:-firstvar;
    WHILE var=/=NONE DO INSPECT var DO
    BEGIN
      a1:=frac*a1;
      state:=oldstate+0.5*a1;
      rate:=0;
      var:-sucvar;
    END;
  END *** IF ERROR_FLAG ***;
```

```
dt:=0.5*h; Time:=lasttime+dt;
Resume(firstcont);

var:-firstvar;
WHILE var=/=NONE DO INSPECT var DO
BEGIN
  a2:=h*rate;
  state:=oldstate+0.25*(a1+a2);
  var:-sucvar;
END;

Resume(firstcont);

var:-firstvar;
WHILE var=/=NONE DO INSPECT var DO
BEGIN
  a3:=h*rate;
  state:=oldstate+(2*a3-a2);
  var:-sucvar;
END;

dt:=h; Time:=lasttime+dt;
Resume(firstcont);

var:-firstvar;
WHILE var=/=NONE DO INSPECT var DO
BEGIN
  a4:=h*rate;
  dsh:=((a1+a4)+4*a3)/6;
  state:=oldstate+dsh;
  var:-sucvar;
END;

Resume(firstcont);

var:-firstvar;
WHILE var=/=NONE DO INSPECT var DO
BEGIN
  a5:=h*rate;
  ds:=(-dsh+(24*a5-20*a4))+16*(a3-a2);
  a4:=(-a1+4*a3)+(17*a4-23*a5);
  state:=oldstate+(dsh+0.5*a5);
  var:-sucvar;
END;

dt:=1.5*h; Time:=lasttime+dt;
Resume(firstcont);
```

```
    var:-firstvar;
    WHILE var=/=NONE DO INSPECT var DO
    BEGIN
      a2:=h*rate;
      state:=oldstate+(dsh+0.25*(a5+a2));
      var:-sucvar;
    END;

    Resume(firstcont);

    var:-firstvar;
    WHILE var=/=NONE DO INSPECT var DO
    BEGIN
      a3:=h*rate;
      state:=oldstate+(ds+(a2-2*a3));
      var:-sucvar;
    END;

    dt:=dtnow; Time:=nexttime;
    Resume(firstcont);

    error_test;

    IF NOT error_flag THEN
    BEGIN
      Resume(firstcont);

      var:- firstvar;
      WHILE var =/= NONE DO INSPECT var DO
      BEGIN
        ds:=(a4+dsh)+((a5+h*rate)+4*a3)/6;
        var :- sucvar;
      END;


    END *** IF NOT ERROR_FLAG ***;
  END *** TAKE_A_STEP ***;


  PROCEDURE act_error(var); REF(variable) var;
  INSPECT var DO
  INSPECT themonitor DO
  BEGIN
    a4 := ((4*a3 - h*rate) + a4)/90;
    temp := Abs(abserror) + Abs(relerror*(oldstate + dsh));
  END *** ACT_ERROR ***;
```

3 ALTERNATIV INTEGRATIONSMETOD AV PREDIKTOR-KORREKTORTYP

3.1 INLEDNING

Ett av önskemålen med simuleringspaketet var möjligheten att behandla diskontinuiteter i derivatorna. För att lösa detta problem infördes en alternativ integrationsmetod, arbetande enl samma princip som "DHAMDI" av Göran Fick [5], som i sin tur bygger på en SSP-rutin "HPCG". Se [4]

3.2 GRUNDMETODEN

Grundläggande metod är Hamming´s modifierade prediktor-korrektor metod. Som alla prediktor-korrektor (PC) -metoder är den inte självstartande utan behöver känna till tillståndsvariablernas värde och derivator i fyra ekvidistanta punkter för att kunna startas. För detta ändamål används i regel en Runge-Kutta metod, i detta fall den metod enl England som är normal integrations-metod i COMBINEDSIMULATION. Eftersom ett steg med denna metod består av två delsteg finns efter ett sådant steg tre punkter tillgängliga. För att erhålla den fjärde kan man t ex ta ett ensamt delsteg. Detta medför dock problem, dels eftersom det är svårt att skapa ett numeriskt motiverat feltest i detta "halvsteg", dels skulle det komplicera interpoleringsalgoritmen vid sökandet efter tillståndshändelser och "tidsfixerade samplingar".

För att undvika dessa problem föreslås istället att start-proceduren består av två hela Runge-Kutta-England steg varvid steglängden låses efter det första. Detta medför dessutom att man efter startproceduren har tillgång till fem punkter, dvs det behövs bara ett PC-steg till innan en dubblering av steglängden kan tillåtas.

Eftersom Runge-Kutta-England metoden i COMBINEDSIMULATION inte bara ändrar steglängen med dubblering eller halvering används ett interpoleringsförfarande för att fixera rapportering till vissa tidpunkter. Därför behöver problemet med när dubbling är tillåtet med hänsyn till rapporttider, se [5], inte beaktas.

För att kunna dubblera steglängden krävs att state och rate är definierade vid sex tidpunkter bakåt. Efter uppstart med Runge-Kutta finns fem, efter ett normalt PC-steg ökas antalet med ett, efter dubblering av steglängen finns fyra och efter ett under-känt steg finns fem punkter definierade. För hålla reda på hur läget är införs en variabel ´normal_pc_steps´ som skall innehålla hur många punkter utöver fyra som är klara.

Normal_pc_steps sätts alltså till 1 efter RK-uppstart, till 0 efter dubblering av steglängd, till 1 efter halvering av steglängd resp ökas med 1 efter ett normalt steg. För att kunna dubbla steglängden måste normal_pc_steps alltså vara 2.

3.1.2 'UPPTÄCKT' AV DISKONTINUITET

För att klara av diskontinuiteter måste steglängden delas ner så många gånger att integrering kan ske över diskontinuiteten utan att steget blir underkänt pga för stort lokalt fel. När väl diskontinuiteten är passerad kan steglängen åter öka. Men eftersom detta sker med dubblering av steglängden och dubblering bara kan ske om värden på tillståndsvariablerna och derivatorer är kända sex steg tillbaka, så måste efter varje dubblering två steg utan dubblering tas innan nästa dubblering kan äga rum. Således krävs det tre gånger så många steg innan steglängden är tillbaka på ursprunglig nivå än vad det krävdes halveringar för att klara diskontinuiteten.

Metoden som Fick föreslår i [5] bygger på att det som kännetecknar en diskontinuitet är:

1. Ett stort antal halveringar av steglängden.
2. Därefter följande dubbleringar.

Om man med dessa kriterier kan "detektera" en diskontinuitet kan sedan en effektivisering ske genom att återstarta med Runge-Kutta metoden och en normal steglängd.

Detta vore i och för sig möjligt att genomföra utan att införa PC-metoden. Tester har dock visat att det totala felet vid integrering med Runge-Kutta_England i vissa fall blivit c:a 10 ggr större än det maximalt tillåtna lokala felet, medan PC-metoden sällan givit mer än två ggr så stort totalt fel. Se bilaga 2. Dessutom effektiviseras beräkningarna genom att antalet beräkningar av den funktion som skall integreras blir färre med PC än med RKE.

Eftersom möjligheterna till effektiv hantering av diskontinuiteter ligger i PC-delen är det önskvärt att undvika diskontinuiteter under RK-uppstartningen. Följande situation kan då ställa till problem. Antag att en stegfunktion H(t-4) skall integreras och att startsteglängden är 2 tidsenheter (te). Det första RK-steget till t=2 går då bra. Det andra går däremot inte pga att ändringen i H blir för stor, en minskning av steglängden blir alltså nödvändig. Eftersom en viktig princip är att en gång godkänt steg inte skall rivas upp igen blir det alltså det andra steget som får tas om. Detta innebär i sin tur att värden från det första steget inte längre kan användas som startvärden till PC-metoden eftersom de två stegen kommer att ha olika steglängd. Två nya godkända RKE-steg behövs alltså.

Halveras nu steglängden tas det första nya steget med längden 1 te från t=2 till t=3, vilket går bra. Det andra steget skall då utföras från t=3 till t=4, men stöter på diskontinuiteten och ny delning blir nödvändig! Detta upprepas nu tills antingen steglängden blir mindre än den minsta tillåtna eller att steget blir så litet att diskontinuiteten klaras av med RK-metoden. I det senare fallet blir följden, förutom dålig noggrannhet, att diskontinuiteten inte "upptäcks" eftersom de möjligheterna ligger i PC-delen.

För att undvika den här situationen, har två lösningar betraktats. Den första går ut på att om felkriteriet inte uppfylls under uppstartningen med RK, tas steget om efter fjärdedelning av steglängden. Den andra metoden halverar steglängden, men om det finns ett godkänt första RK-steg används interpolation för att beräkna startvärden till PC-metoden i ekvidistanta punkter. På detta sätt behövs inte mer än två godkända RK-steg oberoende av hur många underkända som kommer mellan dem.

De båda metoderna har jämförts, dels vad avser CPU-tid för ett testproblem, dels avseende sannolikheten för att metoderna ändå inte klarar av att starta pc-integrationen före en eventuell diskontinuitet. Jämförelsen visade att metod 2 bör vara den bättre, och den används också i CONDIS.

För att realisera ´diskontinuitetsupptäckten´ föreslås följande lösning, som i hög grad överensstämmer med den i DHAMDI. Antalet halveringar av steglängden i förhållande till dtmax noteras i attribut till themonitor kallat ´halvings´. Halvings räknas även ner när steglängen fördubblas. Efter varje godkänt steg testas om kvoten mellan det maximalt tillåtna felet och det uppskattade lokala felet för den variabel med minst sådan kvot, dvs errorratio använt på samma sätt som i orginalversionen, är tillräcklig stor för att tillåta dubblering av steglängden. Eftersom metoden är av fjärde ordningen, dvs felet är proportionellt mot (steglängden)^5, skall teoretiskt sett kvoten vara $2^5 = 32$ för att en dubbling av steglängden skall vara motiverad. Väljes detta värde är dock risken stor att nästa steg inte kan godkännas pga att svårighetsgraden på integranden kanske ökar något lite. Helsgaun föreslår i [2] att 64 är ett lämpligt värde, medan Fick i [5] använder 50. I nuvarande version av CONDIS används 50 men detta kan ev ändras efter mera ingående uttestning.

Skulle nu errorratio vara större än 50 samtidigt som antalet halveringar är stort anses en diskontinuitet vara upptäckt. Antalet halveringar som behövs har Fick satt till 10, i CONDIS föreslås användandet av en variabel ´disc_halvings´ som normalt sättes till 10, men som användaren kan ändra i sitt program eller interaktivt i samband med att ett integreringsfel uppstår. Se även kap 4 om felhantering. Svårigheten är nu att det inte går att starta om med Runge-Kutta så fort dessa villkor är uppfyllda. Betrakta nämligen följande situation:

Antag att en stegfunktion H(t-3) skall integreras, samt att dtmax är 1. Efter två RK-steg till t=2 börjar nu PC-metoden med initialsteglängd 0.5. Första steget går bra, andra till t=3 går inte om det tillåtna felet är tillräckligt litet. Ett halverat steg till 2.75 gå bra, nästa till 3 går inte osv. När nu antalet halveringar överskrider disc_halvings kan situationen vara den att diskontinuiteten fortfarande inte är passerad, ett steg fram till diskontinuiteten är precis underkänt och nästa steg går alldeles utmärkt ty diskontinuiteten nås inte. Detta steg kommer förmodligen att få ett errorratio som är tillräckligt stort för att tillåta dubbling, och antalet halveringar överskrider nu disc_halvings. Men det går inte att starta om med Runge-Kutta innan diskontinuiteten är passerad, så något måste göras för att fördröja omstarten tills så har skett.

Föreslagen lösning på problemet utgår från att högst ett godkänt steg kan tas mellan varje underkänt när man närmar sig en diskontinuitet. En boolesk varabel ´disc_imp´ införs som sätts TRUE efter varje halvering av steglängden, den blir FALSE igen efter ett normalt steg. Testas nu, förutom halvings > disc_halvings och errorratio > 50, även NOT disc_imp erhålls den önskade effekten ty är diskontinuiteten inte passerad blir nästa steg antingen inte godkänt eller är disc_imp TRUE. Är däremot diskontinuiteten passerad, vilket ju måste ske med ett normalt steg där disc_imp sätts FALSE, så fungerar det hela enligt önskan.

3.3 ANPASSNING AV METOD "DHAMDI" TILL COMBINEDSIMULATION

I COMBINEDSIMULATION är det "themonitor" som hanterar själva integrationen. Orginalversionen av "themonitor" kan skissas enl följande:

```
   nexttimeevent:-controller2.Nextev;
   WHILE nexttimeevent =/= NONE OR firstwait =/= NONE DO
B1 BEGIN
     nexteventtime:=nextsampletime:=nexttimeevent.Evtime;
     resume(firstcont);
     resume("alla sample");
     WHILE Time < nexteventtime DO
B2   BEGIN
       IF firstcont =/= NONE THEN
B3     BEGIN
         integration:
         IF firstvar == NONE THEN
B4       BEGIN
           "RÄKNA FRAM TIDEN ETT STEG
           BERÄKNA DERIVATORER"
E4         END
           ELSE
B5         BEGIN
             "INTEGRERA FRAM ETT STEG MED RUNGE-KUTTA-ENGLAND
              STEGLÄNGD : MIN(Dtnext,Nexteventtime-lasttime,Dtmax)"
E5         END;
E3       END
         ELSE
B6       BEGIN
           "RÄKNA FRAM TIDEN ETT STEG"
E6       END *** IF FIRSTCONT =/= NONE ***;

         IF "TILLSTÅNDSHÄNDELSE INTRÄFFAT UNDER STEGET" THEN
B7       BEGIN
           "BERÄKNA INTERPOLATIONSPOLYNOM"
           WHILE "TIDEN FÖR TILLSTÅNDSHÄNDELSEN EJ BEST MED"
                        "NOGGR DTMIN"   DO
B8         BEGIN
             "BINÄR SÖKNING EFTER TIDEN DÅ"
             "TILLSTÅNDSHÄNDELSEN INTRÄFFAT"
             "Time := "DEN SÖKTA TIDEN"
E8         END;
           IF Time < nexteventtime THEN
B9         BEGIN
             nexteventtime:=Time
E9         END;
           nextsampletime:= Time;
E7       END *** IF "TILLSTÅNDSHÄNDELSE UNDER STEGET" ***;
         IF firstpossample =/= NONE AND nextsampletime < Time THEN
B10      BEGIN
           "UTFÖR ALLA TIDSBEST SAMPLINGAR FRAM TILL Time"
E10      END;
         Resume(firstzerosample);
E2     END *** WHILE TIME < NEXTEVENTTIME ***;
       Resume(firstnegsample);
E1   END *** WHILE NEXTTIMEEVENT=/=NONE OR FIRSTWAIT=/=NONE ***;
```

3.3.1 PRINCIPSKISS FÖR PREDIKTOR-KORREKTORMETOD.

För att införa PC-metoden föreslås följande ändringar i COMBINEDSIMULATION:

1. En global Boolesk variabel "pc" införes, analog med euler, adams och trapez. PC = true innebär alltså att PC-metoden skall användas.

2. Attributen till CLASS variable utökas med:

```
state3,state2,state1,state0,state_1,state_2,
rate3,rate2,rate1,rate0,rate_1,rate_2,
p3_c3,p4,c4
```

   State- resp rate- 3 tom -_2 defineras som variablens värde resp derivatans värde vid tider enl fig 1.

   Observera att i variabelnamnen används "_" för att beteckna "-".

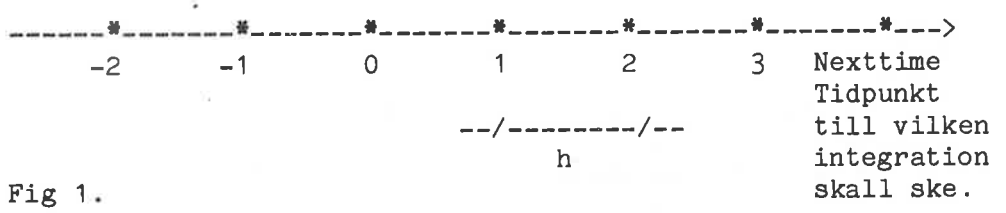

Fig 1.

   Betydelsen av p3_c3,p4 och c4 framgår av metodbeskrivningen, se [4].

3. Blocket som i skissen betecknas B3 ändras enl följande principskiss.

```
    IF firstcont =/= NONE THEN
B3  BEGIN
      "BERÄKNA VÄRDEN PÅ DTNOW OCH NEXTTIME"
      WHILE "STEGET INTE GODKÄNT" DO
      BEGIN
        IF euler OR adams OR trapez OR firstvar == NONE THEN
        BEGIN
          "TAG ETT STEG ENL RESP METOD"
        END
        ELSE
        IF "PC-STEG INTE MÖJLIGT" THEN
        BEGIN
          IF "FÖRSTA FÖRSÖKET TILL GODKÄNT STEG" THEN
          BEGIN
            "RK-BERÄKNINGAR DEL 1."
          END;
          "RK-BERÄKNINGAR DEL 2."
          IF "STEGET INTE KAN GODKÄNNAS" THEN
          BEGIN
            IF "MÖJLIGT ATT MINSKA STEGLÄNGDEN" THEN
```

```
            BEGIN
              "FÖRBERED FÖR NYTT STEG MED MINSKAD STEGLÄNGD"
              "RK-BERÄKNINGAR DEL 1."
            END
            ELSE
            BEGIN
              "FELÅTGÄRDER SE KAP 4"
            END;
          END
          ELSE
          BEGIN
            "RK-BERÄKNINGAR DEL 3."
          END;
          IF "STEGET KUNDE GODKÄNNAS" THEN
          BEGIN
            "RK-BERÄKNINGAR DEL 4."
            "BESTÄM DTNEXT"
            IF "BEGÄRD METOD ÄR PC" THEN
            "AVGÖR OM RK-DELEN ÄR KLAR";
          END;
        END
        ELSE
        BEGIN
          IF "FÖRRA STEGET VAR RK-STEG" THEN
          BEGIN
            "FÖRBÄTTRA STARTVÄRDEN FÖR PC MED ITERATION"
            "PC-BERÄKNINGAR DEL 1."
          END;
          IF "STEGET INTE KAN GODKÄNNAS" THEN

          BEGIN
            IF "MÖJLIGT ATT MINSKA STEGLÄNGDEN" THEN
            BEGIN
              "FÖRBERED NYTT STEG MED HALVERAD STEGLÄNGD"
            END
            ELSE
            BEGIN
              "FELÅTGÄRDER SE KAP 4"
            END;
          END;
          IF "STEGET KUNDE GODKÄNNAS" THEN
          BEGIN
            IF "DISKONTINUITET UPPTÄCKT" THEN
            "FÖRBERED FÖR DISKONTINUITET"
            ELSE
            IF "MÖJLIGT ATT DUBBLA STEGLÄNGDEN" THEN
            BEGIN
              "FÖRBERED STEG MED DUBBLERAD STEGLÄNGD"
            END
            ELSE "FÖRBERED STEG MED OFÖRÄNDRAD STEGLÄNGD";
          END;
        END *** PC-STEG ***;
      END *** WHILE "STEGET INTE GODKÄNT" ***;
E3  END *** IF FIRSTCONT =/= NONE ***
    ELSE  ...
```

3.2.2 KOMMENTARER TILL PRINCIPSKISS AV THEMONITOR.

"PC-STEG INTE MÖJLIGT"

För att pc-steg SKALL vara möjligt krävs för det första pc = TRUE. Dessutom måste två kompletta RK-steg ha tagits tidigare för att state- och rate- 2 tom 0 skall vara bestämda. För att hålla reda på detta införs den Booleska variablen ´pc_possible som ett attribut till themonitor. Om pc = TRUE sätts pc_possible till TRUE då de bägge inledande RK-stegen är klara.

För att veta vilket av de två RK-stegen som utförs, införs en Boolesk variabel ´start_rk´. Start_rk = TRUE anger att det är första steget, FALSE det andra. Start_rk behöver kännas av dels för att bestämma nästa steglängd (den får ju inte ändras om det är det andra steget som skall påbörjas), dels för att veta vilka state och rate variabler som skall få värde (_1 och 0 bestämms under första, 1,2 och 3 under andra RK-steget). Dessutom skall interpolering i föregående steg för att ändra state- resp rate_1 och -0 vid halvering av steglängd inte ske om start_rk är TRUE. Start_rk sätts FALSE när det första steget är klart, den blir TRUE samtidigt som pc_possible blir TRUE för att ha rätt värde nästa gång det blir aktuellt med RK-steg.

"FÖRRA STEGET VAR RK-STEG"

En Boolesk variabel ´last_pc´ införs som attribut till the-monitor. Den sättes till TRUE i slutet av varje PC-steg och till FALSE vid varje RK-steg.

Villkoret blir alltså:
```
IF NOT last_pc
```

"FÖRBÄTTRA STARTVÄRDEN FÖR PC MED ITERATION"

Det är mycket viktigt att startvärdena är så bra som möjligt. Därför används en iterativ interpolationsprocedur för att förbättra state- och rate 3 tom 1.Se [4] sid 339.

"BERÄKNA VÄRDEN PÅ DTNOW OCH NEXTTIME"

Innan exekveringen når hit är dtnow satt till nexteventtime - lasttime, dvs ett steg till nästa tidshändelse. Detta värde måste nu korrigeras, varvid fyra möjligheter finns. Två uppträder om pc_possible är FALSE dvs om nästa steg skall tas med RK-metod. Om nämligen dtnow > dtnext skall dtnow := dtnext, annars skall dtnow behållas men start_rk måste sättas till TRUE eftersom steglängden ändras. Om däremot pc_possible = TRUE

kontrolleras om dtnow är större än h (=steglängden i förra PC-steget alt halva steglängden om förra steget var RK-steg.), i så fall sätts dtnow := h. Är däremot dtnow < h måste steglängden minskas och övergång till RK-steg ske. Dtnow behöver alltså inte ändras men pc_possible := FALSE och start_rk := TRUE.

I samtliga fall sätts nexttime till (i princip) lasttime + dtnow.

"STEGET INTE GODKÄNT"
"FÖRSTA FÖRSÖKET TILL GODKÄNT STEG"

Slingan har införts för att undvika ett antal go_to-satser då noggrannheten vid integrationen inte kan uppfyllas. Styrande för testet är en boolesk variabel ´ok´ som sätts FALSE i satsen före WHILE-satsen, och därefter TRUE som första sats i slingan. Om steget nu inte kan godkännas sätts ok till FALSE i samband med att förberedelser för omtagning av steget utförs. Repetionen av slingan slutar alltså när slingan är genomlöpt en gång utan att ok har ändrats till FALSE.

När ett steg har blivit underkänt och skall tas om kan räkne-arbetet förenklas genom att utnyttja några värden från det underkända steget. Dessa beräkningar utförs i slutet på det underkända steget, varför primärräkningarna bara skall utföras i det första försöket. För att styra detta införs en boolesk variabel ´firsttime´, som är TRUE första gången slingan "WHILE NOT ok DO" genomlöps.

Strukturen av denna slinga blir alltså:

```
firsttime := TRUE
ok := FALSE;
WHILE NOT ok DO
BEGIN
  ok := TRUE;
  ...
  IF firsttime THEN
  BEGIN
    ...; firsttime := FALSE;
  END;
  ...
  IF "STEGET MÅSTE TAS OM" THEN OK := FALSE; (Förekommer
  ...                                         på flera
END *** WHILE NOT OK DO ***;                  platser i
                                              slingan)
```

"STEGET INTE KAN GODKÄNNAS"

Det uppskattade lokala felet läggs i attributet a4 till resp variabel. Det maximalt tillåtna felet räknas ut som Abs(abserror) + Abs(relerror*state) och läggs i temp.

Villkoret blir alltså " IF Abs(a4) > temp THEN ..."

"MÖJLIGT ATT MINSKA STEGLÄNGDEN"

För att bestämma minsta tillåtna steglängd används olika metoder under RK- resp PC-integrering.

Vid RK anges dtmin som minsta tillåtna steglängd. Minskning sker genom halvering om den resulterande steglängden inte blir för liten, annars minskas till dtmin.

Vid PC anges i stället max_halvings som det totala antalet halveringar som tillåts i förhållande till dtmax. Max_halvings är en global integer variabel som sätts till 30 av CONDIS. Användaren kan ändra den i sitt program eller interaktivt om ett integrationsfel uppstår. Se även kap 4 om felhantering.

"BESTÄM DTNEXT"

Under ren RK-integrering bestäms dtnext ur förhållandet mellan det lokala och det maximalt tillåtna felet i det utförda steget. En begränsning sker dock till intervallet [dtnow,min(2*dtnow,dtmax)].

Efter det första RK-steget under uppstartning av PC-metoden sätts dtnext till dtnow, dvs ingen ändring av steglängden tillåts.

Efter det andra RK-steget sätts dtnext till dtmax eftersom dtnext inte utnyttjas under PC-steg, och vid omstart med RK skall steglängden vara dtmax.

"AVGÖR OM RK-DELEN ÄR KLAR"

Se "PC-STEG INTE MÖJLIGT".

"STEGET KUNDE GODKÄNNAS"

IF ok THEN ...

"FÖRBERED NYTT STEG MED HALVERAD STEGLÄNGD"

Här utförs interpolering i det föregående steget för att få värden på state resp rate som överensstämmer med den nya steglängden. Även p3_c3 räknas om, för formlerna hänvisas till [4]. Dtnow och h halveras och nexttime räknas ut som lasttime + dtnow. Eftersom state och rate bara blir uträknade från -1 och uppåt sätts normal_pc_steps till 1. Ok sätts FALSE och disc_imp sätts TRUE.

"DISCONTINUITET UPPTÄCKT"

Enligt den inledande diskussionen blir detta testet: "IF errorratio > 50 AND halvings > disc_halvings AND NOT disc_imp"

"FÖRBERED FÖR DISKONTINUITET"

Tillräckliga åtgärder här är att sätta pc_possible till FALSE och halvings till 0.

"MÖJLIGT ATT DUBBLA STEGLÄNGDEN"

Kravet blir dels att errorratio är tillräckligt stort, dels måste tillräcligt många state resp rate vara definierade dvs normal_pc_steps >= 2. Dessutom kontrolleras att den dubblade steglängden inte blir större än dtmax. Villkoret blir alltså:

"IF erorratio > 50 AND normal_pc_steps >= 2 AND 2*h <= dtmax"

"FÖRBERED STEG MED DUBBLERAD STEGLÄNGD"

State och rate omstuvas, oldstate, lasttime, h och dtnow ändras enl fig 2.

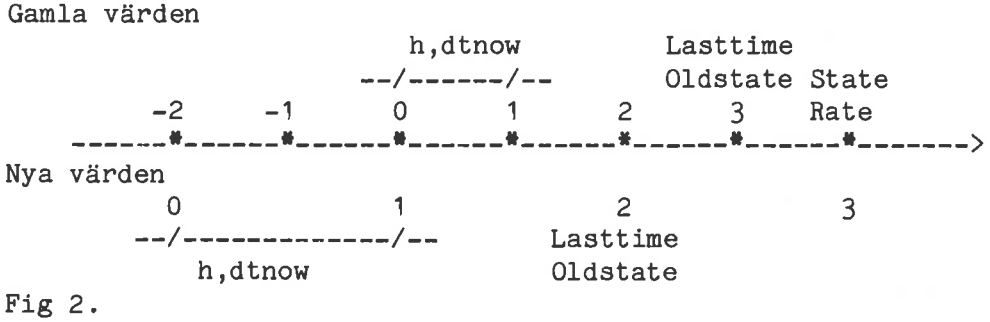

Fig 2.

Dessutom måste p3_c3 räknas om enl formel på sid 338 i [4]. Slutligen sätts normal_pc_steps till 0 och halvings minskas med 1.

"FÖRBERED STEG MED OFÖRÄNDRAD STEGLÄNGD"

State- och ratevariablerna, oldstate mm. uppdateras enl fig 3.

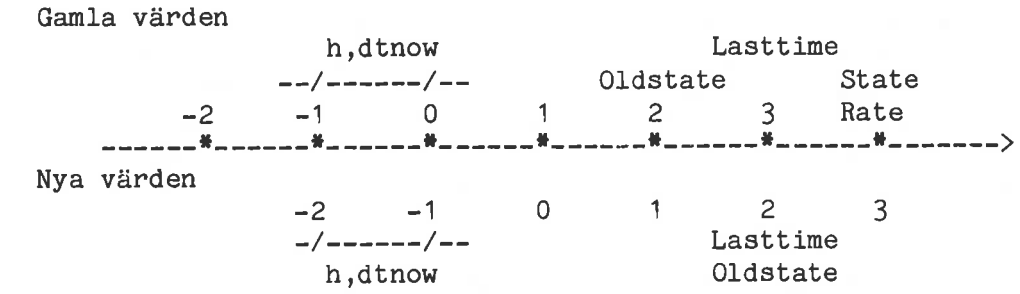

Fig 2.

Dessutom måste p3_c3 uppdateras. Eftersom p3_c3 är uppskattningen av det lokala felet i förra steget sätts den till a4 som innehåller detta värde. Vidare ökas normal_pc_steps med 1 och disc_imp sätts TRUE.

4 INTERAKTIVITET I CONDIS

I COMBINEDSIMULATION förekommer ingen dialog mellan programmet och användaren. Då COMBINEDSIMULATION upptäcker ett fel skrivs ett felmeddelande ut, och sedan avslutas exekveringen. Dock finns det förberett för att exekveringen kan fortsätta efter ett anrop av integrationerror (anropas då integrationsnoggrannheten inte kan uppfyllas). Genom att procedurerna integrationerror och simulationerror är virtuella kan användaren genom att skriva egna procedurer specifiera vad som skall hända vid ett sådant fel.

Ett starkt önskemål angående simuleringspaketet är att man skall kunna fortsätta exekveringen efter så många olika typer av fel som möjligt, där så är tilllämpligt efter att ha ändrat på integrationsparametrar. För att möjligöra detta ingår i CONDIS ett antal procedurer som anropas vid olika fel. Vissa justeringar av koden i klassen ´monitor´ har utförts i anslutning till detta.

En procedur som frågar efter värden på vissa parametrar innan integrationen startar är också inlagd.

För denna dialog används paketet SAFEIO, ett paket för inmatningssäker konversation mellan användare och SIMULA-program se [9]. Genom att CONDIS via filequ (se kap 6) är en underklass till SAFEIO kan användaren av CONDIS även använda SAFEIO:s möjligheter utan fler åtgärder än de som måste till tack vare att CONDIS använder SAFEIO. Se även kapitel 7 om externdeklarationer.

För den som måste svara på frågor levererade av SAFEIO kan följande korta beskrivning kanske vara till hjälp:

Det finns två typer av frågor. I den första ställs en enkel fråga som oftast skall besvaras med ett tal eller med ´yes´ alternativt ´no´. På detta svarar användaren med talet resp ordet följt av <RETURN>. Gäller det ett tal finns då ofta en kontroll av att talet ligger inom vissa gränser. Skulle det avgivna svaret inte uppfylla detta krav skrivs ett meddelande ut om varför talet inte blev godkänt, därefter kommer frågan igen. Kommer frågan tillbaks utan något meddelande innebär det att programmeraren inte orkat skriva någon upplysning om begränsningarna som gäller.

Den andra typen är så kallad menyfråga, vilket innebär att ett antal alternativ till svar radas upp. För att välja ett av alternativen skriver man då den delen av förslaget som inte står inom parantes, eller en entydig förkortning av det, följt av <RETURN>.

För båda frågetyperna gäller att om man svarar med ett ´?<RETURN>´ så erhålls i regel någon upplysning om vad man skall svara. Dessutom kan förstahandssvar förekomma, vilket innebär att frågan följs av någonting som står mellan snedstreck. Anser man då att uttrycket inom snedstrecken är ett bra svar på frågan så räcker det med att slå <RETURN>.

I avsnitt 4.3 finns ett exempel på dialogen vid körning av ett CONDIS-program.

4.1 INLÄSNING AV INTEGRATIONSPARAMETRAR VID START AV PROGRAMMET.

I de exekverbara satserna av CONDIS´ yttersta block finns ett anrop av proceduren prologue inlagt före INNER. Detta innebär att proceduren anropas precis innan de exekverbara satserna i användarens eget program börjar exekveras.

Prologue innehåller anrop av procedurerna method och new_dtmax_dtmin samt inläsning av värde till variablerna maxrelerror och maxabserror. Eftersom prologue är deklarerad virtuell kan användaren slippa denna dialog genom att själv deklarera prologue t ex med en tom klasskropp.

Method är en procedur som frågar efter önskad integrationsmetod. Vilken metod som önskas bestämms av de booleska variablerna pc, adams, trapez och euler. Önskas Runge-Kutta-England skall ingen av variablerna vara TRUE annars skall den som motsvarar önskad metod vara TRUE. Om önskad metod är HEUN skall både adams och trapez vara TRUE. Method börjar med att sätta alla fyra till FALSE, sedan ställs en menyfråga om vilken metod som önskas. Svaret på denna medför att rätt variabel (variabler) blir TRUE. Förstahandssvar på frågan är RK (Runge-Kutta-England).

New_dtmax_dtmin frågar efter värde på först dtmin därefter dtmax, kontroll är inlagd så att dtmin blir större än noll samt att dtmax blir större än eller lika med dtmin. Förstahandssvar på frågorna är det värde som dtmin resp dtmax har vid anrop av proceduren dvs båda är noll då proceduren anropas från prologue. I detta fall kan med andra ord förstahandssvar inte användas !

Både method och new_dtmin_dtmax används även i samband med dialog vid fel i körningen.

Vid skapandet av nya variabler med uttrycket ´NEW variable(begynnelsevärde)´ får attributen relerror resp abserror samma värde som de globala variablerna maxrelerror och maxabserror. Vill man ha olika felgränser för olika variabler kan man tilldela relerror resp abserror värde med punktnotation till en referens till variablen ifråga, detta måste dock göras i användarens program. Dessutom finns möjlighet att ändra relerror

och abserror för den variabel som orsakar ett felavbrott vid integration. Se avsnitt 4.2.1. För maxrelerror krävs det ett värde som är större än eller lika med noll, medan maxabserror måste vara strängt större än noll pga risken för att feltoleransen vid variabelvärden kring noll annars blir mycket liten.

Om användaren i sitt program själv anger värden på metod, dtmin, dtmax, maxrelerror eller maxabserror blir det dessa som gäller eftersom den dialog som är beskriven här inträffar först. Vill användaren inte ha dessa inledande frågor kan han utnyttja att prologue är deklarerad virtuell och själv deklarera en procedure prologue med tom klasskropp. Noteras bör att maxrelerror och maxabserror måste ges värden innan nya variabler skapas annars får dessa variablers relerror och abserror värdet noll.

4.2 ÅTGÄRDER VID FEL I PROGRAMMET.

I COMBINEDSIMULATION finns ett antal tester av att vissa variabler inte har värden som gör fortsatt exekvering odefinierad eller meningslös. Då en sådan situation uppkommer anropas proceduren error som skriver ut ett meddelande om felets art. För att om möjligt kunna fortsätta exekveringen, eller i varje fall kunna få mer information om felet, är i CONDIS proceduren utökad. Först lämnas samma meddelande som förut, beroende på felets art ges därefter möjligheter att stoppa körningen, ändra på vissa parametrar eller gå in i SIMDDT. SIMDDT är ett avlusningspaket för SIMULA-program som finns på DEC-systemet, för beskrivning hänvisas till manual för SIMDDT, [10]. Då så är möjligt kan exekveringen därefter startas igen. De olika felutskrifterna och handlingsalternativen behandlas i avsnitt 4.2.2.

Den typ av fel som beror på att integrationsnoggranheten inte kan uppfyllas hanteras av en annan procedur, integrationerror. I COMBINEDSIMULATION är denna procedur deklarerad VIRTUAL, detta är bortaget i CONDIS men proceduren är i stället utökad så att man kan byta integrationsmetod, ändra minsta och största steglängd, ändra noggranhetskraven mm, därefter kan exekveringen om så önskas återupptas. Mera om detta i nästa avsnitt.

4.2.1 INTEGRATIONSFEL.

När PC- eller RK-integrationsmetoderna används testas för varje integrationssteg, om det lokala integrationsfelet för någon variabel överstiger det maxmalt tillåtna, värdet specificerat via relerror och abserror. Om begärd noggrannhet inte kan uppnås anropas procerduren integrationerror. Det första som händer är då att ett meddelande om detta skrivs ut tillsammans med tid-

punkten när det hände samt aktulla värde på dtmin, dtmax och dt. Därefter kommer frågan om man vill avsluta direkt. Om så inte är fallet ges en menyfråga om vad man vill göra. Alternativen är, om integrationerror är anropad under RK-integrering:

```
(Change) method
(Change) dtmin (and dtmax)
(Change rel- and abs-) error
                (for the variable that caused the error)
(Enter) SIMDDT
Stop (execution)
```

Under PC-integrering tillkommer även :

```
(Change) disc_halvings
(Change) max_halvings
```

"(Change) method" och "(Change) dtmin (and dtmax)" innebär att procedurerna method resp new_dtmax_dtmin (se 4.1) anropas. Efter "(Enter) SIMDDT" anropas en extern procedur ´enterdebug(TRUE)´ vilket innebär att SIMDDT startas. Parametern TRUE betyder att det är tillåtet att fortsätta programmet via kommandot ´proceed´.

Före anrop av integrationerror läggs en referens till den variabel vars integrationsfel inte kunde godkännas i variabeln ´errorvariable´. Detta används då "(Change rel- and abs-) errror (for...)" ges som svar. Först ges följande utskrift:

(Siffrorna är ett exempel på vad som kan stå)

```
Actual values:
    Abserror+Abs( Relerror *   State)=Max allowed < Estimated
                                          error         error
   1.00E-05      1.00E-05      3.24E-01    1.32E-05     8.37E-04
```

´Max allowed error´ står alltså för det det fel som högst kunde tolereras, uträknat från relerror,abserror och state. ´Estimated error´ är den uppskattning av det aktuella felet som gjorts av integrationsalgoritmen. Eftersom steget inte blivit godkänt är alltså ´Estimated error´ större än ´Max allowed error´.

Med ledning av dessa värden kan man göra en uppskattning av hur rel- alternativt abserror bör ändras för att steget skall kunna godkännas. Därefter frågar programmet efter önskade nya värden på relerror och abserror, kontroller sker så att abserror är större än noll samt att relerror är större eller lika med noll. Anledningen till att abserror inte får vara noll är att om state för en variabel antar värdet noll och bara relerror är skiljt från noll, så blir även det maximalt tillåtna felet noll med ett så gott som oundvikligt integrationsfel som följd.

Förstahandssvar för frågorna om abserror och relerror är det gamla värdet på resp variabel.

Max_halvings anger det maximala antalet halveringar av steglängden som är tillåtet innan integrationerror anropas. Disc_halvings anger antalet halveringar som behövs innan en diskontinuitetsupptäckt ´tillåts´. Ingen av variablerna har någon mening vid RK-integrering. Förstahandssvar är för max_halvings 30 och för disc_halvings 10, dvs samma som sätts vid start av programmet.

Efter kommandot "stop (execution)" utförs först en aktivering av alla instanser av CLASS sample. Detta sker genom ett anrop av proceduren leave. Observera att eftersom det senaste steget inte behöver vara beräknat för samtliga ´variabler´ så kan vissa värden komma från det föregående steget trots att Time pekar på tiden då det underkända steget avslutas. Dessutom har ju det senaste steget blivit underkänt pga för dålig noggrannhet.

I leave stängs därefter alla filer som är inlagda i listan ´files´ (se kap 6). Exekveringen avslutas sedan med ett anrop av den externa proceduren exit.

Efter att åtgärderna vid resp kommando är utförda (vid SIMDDT när man skriver proceed) får man, utom efter kommandot ´stop´, på nytt samma menyfråga. I fortsättningen finns dock ytterligare ett kommando med nämligen :

"Continue (execution)".

Efter "Continue (execution)" återupptas integrationen på så vis att det senaste, underkända, steget tas om för samtliga variabler, varvid de nya parametervärden som har getts under dialogen används.

Exempel på dialog vid en körning som råkat ut för problem finns i bilaga 4.

4.2.2 ÖVRIGA FEL.

I COMBINEDSIMULATION förekommer 20 numrerade felutskrifter. Alla dessa finns kvar i CONDIS men åtgärderna i samband med fel är inte längre att bara stänga av exekveringen. Dessutom är ytterligare en felutskrift (nr 21) inlagd.

Följande lista omfattar samtliga felutskrifter samt de åtgärder som vidtas i samband med dem.

1:THE REQUESTED INTEGRATION ACCURACY CAN NOT BE ACHIVED : (typ)
I stället för typ står RK eller PC beroende på vilken integrationsmetod som användes då felet uppstod. Begärd integrationsnoggrannhet kan inte uppfyllas. Se föregående avsnitt.

2:THE CURRENT TIME STEP IS TOO SMALL TO ADVANCE TIME
Tidstillväxten dt är så liten att Time inte ändras vid addition av dt.

3:THERE ARE NO DISCRETE EVENTS SCHEDULED
Det finns inte fler planerade händelser, varken tids- eller tillståndshändelser.

4-6: Se nedan.

7:TIME IS AT ITS MAXIMUM VALUE AND NO EVENTS OCCUR
Time har blivit större än det största flyttal som kan representeras i datorn. Det finns fortfarande planerade tillståndshändelser men simuleringen kan inte fortsätta.

8:ILLEGAL USE OF PAUSE
9:ILLEGAL USE OF CANCELSTATEEVENT
10:ILLEGAL USE OF WAITUNTIL
11:ILLEGAL USE OF SETPRIORITY (CLASS CONTINUOUS)
12:ILLEGAL USE OF START (CLASS CONTINUOUS)
13:ILLEGAL USE OF STOP (CLASS CONTINUOUS)
14:ILLEGAL USE OF SETFREQUENCY (CLASS SAMPLE)
15:ILLEGAL USE OF START (CLASS SAMPLE)
16:ILLEGAL USE OF STOP (CLASS SAMPLE)
17:ILLEGAL USE OF (RE) ACTIVATE
18:ILLEGAL USE OF PASSIVATE (OR CANCEL(CURRENT))
19:ILLEGAL USE OF HOLD (OR REACTIVATE CURRENT)
20:ILLEGAL USE OF CANCEL
Samtliga felen 8-20 beror på ett försök att ändra planläggningen för de diskreta händelserna vid ett tillfälle då det inte är tillåtet. Sådan ändring får bara utföras av en diskret process, dvs i en class deklarerad PROCESS, eller i huvudprogrammet.

Efter var och ett av felen 2,3 och 7-20 kommer frågan om man vill gå in i SIMDDT. Under SIMDDT-sessionen kommer det inte att vara tillåtet att ge kommandot ´proceed´. Svarar man nej på frågan avslutas programmet efter att aktiverat alla instanser av CLASS sample och stängt alla filer i fillistan ´files´.

4:DTMIN < 0
5:DTMIN > DTMAX
Efter dessa bägge fel anropas proceduren new_dtmax_dtmin (se avsnitt 4.1) varefter exekveringen fortsätter direkt.

6:FREQUENCY IS TOO SMALL TO ADVANCE TIME (CLASS SAMPLE)
Det finns en instans av CLASS sample som borde ha positiv ´frekvens´ men för vilken det gäller att smptime + frq <= smptime.

Efter detta fel ges möjlighet att avbryta körningen, ändra frq för den instans av sample som gav upphov till felet eller att gå in i SIMDDT. Under SIMDDT-sessionen ligger en referens till den felaktiga sampleinstansen i ´themonitor.errorsample´. Det är tillåtet att fortsätta exekveringen med ´proceed´.

21:ELEMENT OF ILLEGAL KIND IN CLASS OBJECT
De procedurer start resp stop som finns i Head CLASS object kan pga kvalificeringsreglerna bara hantera start- och stopprocedurer i instanser av klasserna continuous, sample och variable med underklasser. Upptäcks andra element i listan skrivs ovanstående felmeddelande ut, därefter kommer frågan om man vill gå in i SIMDDT. Under SIMDDT-sessionen finns en referens till det Link-objekt som har orsakat felet i den globala variabeln errorelement. Det är tillåtet att fortsätta exekveringen med proceed. Om man väljer att inte gå in i SIMDDT fortsätter körningen direkt. Vid anrop av procedurerna START och STOP i object behandlas felaktiga element som om de inte fanns förutom att felutskriften sker och tillfälle till inspektion ges.

5 KLASSEN OBJECT

För att underlätta för användaren av CONDIS att strukturera sina program, kan olika instanser av klasserna variable, continuous resp sample som har fysisk anknytning sammanföras i en klass ´object´.

Object är en underklass till Head, vilket möjliggör att instanser av CLASS Link med underklasser kan läggas i en lista med en instans av CLASS object som huvud. Eftersom variable, continuous och sample alla är underklasser till Link, kan man alltså lägga samhörande instanser av dessa klasser i en kö under ett objekt.

En av fördelarna med detta är att man sedan, via procedurerna start och stop som finns i CLASS object, kan starta resp stoppa alla de element som hör till objektet via ett anrop, istället för ett till varje element. Vidare får man genom att ha deklarationerna av samtliga variabler, continuous och sample i CLASS object en mera logisk och lättläst uppbygnad av programmet.

I bilaga 3 finns exempel på två alternativa sätt att använda klassen. Båda behandlar samma problem, en båts dynamik. Exemplet är hämtat från Åström K J,Källström C : Identification of Ship Steering Dynamics, [11].

I det första exemplet finns deklarationerna av klasserna continuous CLASS shipdynamic och sample CLASS report utanför object CLASS ship. Deklarationerna av referensvariabler till instanser av klasserna finns däremot inuti. Detta medför att överskådligheten i CLASS ship blir bättre, men att en ref till aktuellt skepp måste överföras till shipdynamic resp report. I det andra exemplet finns hela deklarationen av shipdynamic och report inne i CLASS object, vilket medför att man slipper parametern till dessa klasser. Exemplen är bara att betrakta som möjligheter med CLASS object. En användare av CONDIS kan säkert finna andra alternativ.

Källkoden till CLASS object finns i avsnitt 9.2 sid 75.

6 KLASSEN FILEQU

En användare av CONDIS kan i regel förväntas använda en eller flera filer för att hämta data från eller lagra data på. Detta kan ställa till problem vid exekvering av ett program som inte kan avslutas på normalt sätt, t ex pga ett fel av något slag. Samtliga filer måste nämligen vara stängda innan ett Simula-program avslutas om inte ett Run-Time Error skall bli följden.

På DEC-system utgör detta visserligen inget större problem, eftersom kontrollen i så fall övergår till SIMDDT med en kommentar om att alla filer inte är stängda. Genom att använda SIMDDT-kommandot ´close´ följt av ´exit´ löses sedan problemet enkelt. Men eftersom samtliga användare av CONDIS kanske inte behärskar SIMDDT vore det ändå bra om problemet kunde undvikas.

En lösning är att använda den externkompilerade klassen ´filequ´. Denna innehåller en ´Head CLASS file_list´, skapandet av en instans av file_list med namnet ´files´ samt fyra procedurer ´opendirectfile´, ´openoutfile´, ´openinfile´ och ´openprintfile´. Genom att istället för det vanliga attributet open, använda den av de fyra procedurerna som är tillämplig, kommer filen samtidigt som den öppnas att läggas in i listan file_list. När det sedan blir aktuellt att stänga filerna används proceduren close som är ett attribut till file_list. Close letar igenom hela listan, stänger filerna och tar ur elementen ur listan.

Files.close anropas av CONDIS före varje "icke-normal" avslutning av programmet. Dessutom kan användaren själv stänga alla sina filer med ett enda kommando genom att skriva ´files.close´.

Procedurerna open . file har två parametrar. Den första som skall vara specifierad REF(...file) skall innehålla en referens till den fil som skall öppnas. Den andra är en integer som anger buffertstorlek på filen. T ex för att kunna använda en outfile ´utfil behövs följande satser av användaren:

```
REF(Outfile) utfil;
...
utfil :- NEW Outfile("filnamn.ext");
openoutfile(utfil,80);
...
```

Dessa satser medför att utfil läggs in i files samt att den öppnas med satsen "utfil.open(Blanks(80));"

Om det aktuella SIMULA-systemet inte är DECsystem SIMULA version 5 måste klassen filequ externdeklareras i användarprogrammet, se kap 7.

7. EXTERNDEKLARATIONER

Hanteringen av externdeklarationer i program som använder CONDIS är beroende av vilket SIMULA-system som används.

Under DECsystem-10 SIMULA version 5 är principen att bara de externa moduler som verkligen refereras i ett program behöver deklareras. Detta innebär att början på ett program som skall använda CONDIS kan se ut så här:

```
BEGIN
EXTERNAL CLASS condis;
condis(NOTEXT,NOTEXT)
BEGIN
  ...
```

Mer än ovanstående behövs inte för att få tillgång till procedurer eller klasser deklarerade i CONDIS, FILEQU eller SIMEIO. Skulle man däremot t ex vilja använda den externa proceduren exit måste denna denna externdeklareras trots att den används av och är externdeklarerad i CONDIS.

DECsystem-10 SIMULA version 4 kräver däremot även att alla externa moduler som används av moduler högre upp i hierarkin måste deklareras Eftersom CONDIS är en underklass till FILEQU som i sin tur är underklass till SIMEIO innebär detta att väsentlig fler deklarationer behövs. För det första måste FILEQU och SIMEIO själva deklareras, men dessutom alla externa moduler som används av FILEQU respektive SIMEIO.

Deklarationerna på nästa sida är de som behövs för att ett program som använder CONDIS skall fungera under SIMULA version 4.

Lägg märke till att de källkoder som finns i kap 9 är avsedda för version 4.

De flesta deklarationerna avser procedurer som används av SAFEIO, det paket som svarar för inmatningsäker dialog. Klassen SIMEIO är SAFEIO prefixerat med SIMULATION för att även få tillgång till simulerings- och listhanteringsmöjligheter vilket krävs av CONDIS.

Filequ är den klass som ger tillgång till kontroll av de filer som användaren har deklarerat. Se kap 6.

Procedurerna EXIT och ENTERDEBUG används av CONDIS i samband med felhantering. Se även kap 4.

EXTERNDEKLARATIONER

```
BEGIN
EXTERNAL REF(Infile) PROCEDURE findinfile;
EXTERNAL REF(Outfile) PROCEDURE findoutfile;
EXTERNAL TEXT PROCEDURE conc,upcase,frontstrip,rest,
checkextension;
EXTERNAL CHARACTER PROCEDURE fetchar,findtrigger;
EXTERNAL LONG REAL PROCEDURE scanreal;
EXTERNAL INTEGER PROCEDURE checkreal,checkint,scanint,ilog;
EXTERNAL BOOLEAN PROCEDURE menu;
EXTERNAL CLASS simeio,filequ,condis;
EXTERNAL PROCEDURE exit,enterdebug;

condis(NOTEXT,NOTEXT)
BEGIN
```

Parametrarna efter condis är parametrar till SAFEIO. Båda är deklarerade som TEXT-variabler. Den första kan innehålla namnet på en fil som skall användas som logfil, dvs en fil där alla inmatade data till programmet sparas. Om ingen sådan fil önskas eller om man avser att ge sådant filnamn under dialogen sätts aktuell parameter till NOTEXT. Den andra parametern anger på vilket språk man önskar kommentarer från SAFEIO. Möjliga värden är "SWE", "ENG" eller NOTEXT. NOTEXT innebär att "ENG" kommer att användas. Det finns även möjlighet att göra egna meddelandefiler vilket i så fall skall anges i denna parameter.

För utförlig beskrivning av SAFEIO.s möjligheter hänvisas till rapporten om SAFEIO, [9].

Övriga externa procedurer finns beskrivna i SIMULA LANGUAGE HANDBOOK Part 3, [12].

8 KORT ANVÄNDARHANDLEDNING

Detta kapitel är avsett att ge en kort beskrivning av hur CONDIS kan användas för simuleringar.

Bara de viktigaste verktygen är medtagna, för mera detaljerad information se "COMBINEDSIMULATION Brugerhåndbog" [2.2]. [2.2] beskriver COMBINEDSIMULATION version 1, i detta kapitel beskrivs även ändringarna i version 2 som är den version CONDIS bygger på. Dessa ändringar utgörs av införandet av tre nya integrationsmetoder ADAMS, TRAPEZ och HEUN utöver RUNGE-KUTTA-ENGLAND och EULER som fanns i version 1.

De ändringar av COMBINEDSIMULATION som är införda i CONDIS och som berör användaren, är att de tillgängliga integrationsmetoderna utökats med en prediktor-korrektormetod typ DHAMDI, klassen objekt samt dialogen vid felavbrott och start av programmet. Beskrivning av dessa nya möjligheter och övriga avvikelser från [2.2] tas upp i detta kapitel. För felavbrottshanteringen hänvisas till kap 4.

Prediktor-korrektormetoden av typ DHAMDI kommer i fortsättningen ofta att benämnas PC-metoden.

8.1 GRUNDBEGREPP.

Klassen CONDIS är en underklass till FILEQU, som är ett verktyg för att ge CONDIS möjlighet att stänga användardefinierade filer innan ett program avslutas pga ett fel. FILEQU är beskrivet i kap 6. FILEQU är i sin tur en underklass till SIMEIO, ett paket för säker inmatning från terminal som finns på DEC 10 systemet vid Stockholms Datamaskincentral QZ. SIMEIO är slutligen en underklass till den systemdefinierade klassen SIMULATION som används vid diskret simulering. SIMEIO är alltså paketet SAFEIO prefixerat med SIMULATION. För kort beskrivning av SAFEIO se kap 4, i övrigt hänvisas till [9]. De externa klasserna är alltså deklarerade enligt följande skiss:

```
SIMULATION CLASS simeio;...samma klasskropp som i SAFEIO...;
simeio CLASS filequ;...;
filequ CLASS condis;...;
```

CONDIS är en SIMULA klass konstruerad för att möjliggöra beskrivning och simulering av system som innehåller både diskreta och kontinuerliga element (sk blandade system).

Följande avsnitt ger en kort beskrivning av klassen. Läsaren bör vara förtrogen med SIMULA, speciellt klassen SIMULATION.

CONDIS liksom förebilden COMBINEDSIMULATION bygger på SIMULA´s processinriktade syn på simulering. Ett system uppfattas som en samling processer som genomlöper aktiva och passiva faser och vars samverkan styr systemets uppförande.

Processerna indelas i två typer kallade diskreta (CLASS Process) resp kontinuerliga (CLASS continuous). Diskreta processer har, från simulerad tid sett, ögonblickliga aktiva faser som kallas händelser (eng events). Dessa händelser orsakar diskreta ändringar i systemets tillstånd. Kontinuerliga processer däremot har aktiva faser under tidsintervall och orsakar kontinuerliga tillståndsändringar.

En process kan skapas, aktiveras, deaktiveras eller avlägsnas från systemet vid godtyckliga tidpunkter. Men CONDIS tillåter mer än existens av samtidiga processer, genom att processerna kan kommunicera med varandra och påverka varandra på ett ganska generellt sätt. En process kan refera till och ändra varje variabel i en annan process och kan påverka uppdelningen och ordningen av aktiva faser.

CONDIS möjligör simulering av system som har styckvis kontinuerliga tillståndsvariabler (CLASS variable). Diskontinuiteterna kan dels läggas in som händelser, tids-bestämda med hjälp av HOLD, ACTIVATE m fl eller tillståndsstyrda med hjälp av PROCEDURE waituntil. Tidsbestämda händelser kallas tidhändelser och tillståndsstyrda kallas tillståndshändelser.

Dessutom klarar den i CONDIS inlagda integrationsmetoden PC av diskontinuiteter som bara uppträder i funktionsuttrycken i differentialekvationerna.

Löpande insamling av tillståndsdata kan ske under simuleringens gång genom användning av CLASS sample.

Nedan visas en skiss av CONDIS innehållande de viktigaste attributen.

```
Filequ CLASS condis
BEGIN
  CLASS continuous;
  BEGIN
    PROCEDURE start; ... ;
    PROCEDURE stop; ... ;
  END;

  CLASS variable(state); REAL state;
  BEGIN
    REAL rate;
    PROCEDURE start;
    PROCEDURE stop;
  END;

  CLASS sample;
  BEGIN
    PROCEDURE setfrequency(f); REAL f; ... ;
    PROCEDURE start; ... ;
    PROCEDURE stop; ... ;
  END;
```

```
  CLASS object;
  BEGIN
    PROCEDURE start,...;
    PROCEDURE stop;...;
  END;

  PROCEDURE waituntil(b); NAME b; BOOLEAN b; ... ;

  REAL dtmin , dtmax , maxrelerror , maxabserror ;
  INTEGER max_halvings,disc_halvings;
END;
```

Att CONDIS är en underklass till SIMULATION kan användas för att beskriva diskreta tillståndsändringar. Alla SIMULATION´s möjligheter är tillgängliga för användaren. CLASS Process kan alltså användas för att beskriva diskreta processer.

CLASS continuous används alltså för att beskriva de kontinuerliga tillståndsändringarna som definieras med system av första ordningens ordinära differential- och/eller differensekvationer.. Beskrivningarna ges i underklasser till continuous som formler för att beräkna tidsderivatorna för varje tillståndsvariabel. Ett objekt från continuous blir aktiverat när dess START-procedur anropas, dvs från och med anropet utförs de användarspecifierade uppdateringarna av tillståndsvariablerna ´kontinuerligt´. Den aktiva fasen upphör då objektets procedur STOP anropas.

CLASS variable används för att representera styckvis kontinuerliga tillståndsvariabler som mellan händelser varierar enligt första ordningens differential- eller differensekvationer. Värdet av en sådan variabel betecknas med <namn>.STATE, <namn>.RATE betecknar dess derivata. Variable aktiveras med START på samma sätt som continuousobjekt. När variabeln är aktiverad blir STATE ´kontinuerligt´ uppdaterad mellan händelser. Uppdateringen sker enligt den numeriska integrationsmetod som är begärd och tillskottet i variabelns värde blir beroende på det värde RATE får av det aktiva continuousobjektet. Variabelns aktiva period upphör då STOP anropas.

CLASS sample kan användas för regelbunden registrering av tillståndsdata. Efter att ha blivit aktiverad med START blir de operationer som blivit definierade av användaren i en underklass till sample utförda med det tidsmellanrum (i simulerad tid) som bestäms av argumentet till proceduren setfrequency. Setfrequency är ett attribut till klassen sample. Som vanligt upphör den aktiva perioden om STOP anropas.

CLASS object är ett hjälpmedel för att samla de deklarationer av continuous-, variable- och sampleobjekt som tillhör ett ´fysiskt föremål´. För att göra START resp STOP på alla continuous, variable och sample som tillhör ett speciellt föremål räcker det då att göra START eller STOP på den instans av CLASS object som hör till föremålet . CLASS object är närmare beskrivet i kap 5.

Proceduren waituntil används för att planera en händelse till en tidpunkt då ett speciellt systemtillstånd är uppnått. Om ett anrop waituntil(B) sker i den aktiva process som just är 'Current' i SIMULATION's mening så blir processen passiv så länge som det booleska uttrycket B inte är sant. Waituntil fungerar alltså som ett slags Hold fast med ett booleskt villkor som skall bli sant istället för ett specifierat tidsuppehåll.

Mellan de diskreta händelserna uppdateras systemets tillstånd med variabla eller fasta tidssteg beroende på integrationsmetod. DTMAX anger alltid det största tillåtna steget. När PC metoden används anges det minsta tillåtna steget med MAX_HALVINGS, dvs det största antal halveringar av dtmax som är tillåtet. När RK-metoden används bestäms minsta tillåtna steget av DTMIN, medan vid övriga metoder används konstant steglängd DTMAX. Värdet av dtmin och max_halvings är utan betydelse när inte RK- resp PC-metoden används. Dock skall noteras att PC-metoden måste inledas med två RK-steg för att få startvärden, varför dtmin har en viss betydelse även här.

Vid PC- och RK-metoderna används MAXRELERROR och MAXABSERROR för att specifiera de maximala relativa resp absoluta felen i STATE för aktiva variabelobjekt vid uppdateringen. Det maximalt tillåtna uppskattade felet beräknas i princip som :

ABS(maxrelerror*state) + ABS(maxabserror).

Vid övriga metoder sker ingen feluppskattning så MAXRELERROR och MAXABSERROR har där ingen betydelse.

8.1.1 ETT ENKELT EXEMPEL

Följande exempel behandlar banan hos en trestegsraket som skjuts upp vertikalt från en punkt på jordytan.

Under raketens tid i banan ändras dess massa, hastighet och höjd enligt kända fysikaliska lagar som beskrivs nedan i CLASS rocketmotion.

```
1  continuous CLASS rocketmotion;
2  BEGIN
3    mass.rate := -massflow;

4    thrust   := massflow*flowvelocity;
5    drag     := c1*area*Exp(-c2*altitude.state)*velocity.state**2;
6    gravity  := mass.state / (c3+c4*altitude.state)**2;
7    velocity.rate := (thrust - drag - gravity) / mass.state;

8    altitude.rate := velocity.state;
9  END;
```

Kommentarer :

Rad

3 Ändringen i massa beror på bränsleförbrukningen och sker med konstant storlek : - MASSFLOW.

4-7 Ur Newtons andra lag bestämms raketens acceleration genom att addera de krafter som verkar på raketen och dividera med dess massa.

I detta exempel betraktar vi tre krafter: THRUST, DRAG och GRAVITY.

Thrust är dragkraften pga utströmmande förbränt bränsle. Drag är luftmotståndet som bl a beror på lufttätheten, raketens tvärsnittsarea och dess hastighet. Gravity är jordens gravitation som beror på raketens massa och höjd.

8 Raketens höjdändring per tidsenhet, altitude.rate, är lika med hastigheten (velocity).

Klassen rocketmotion används här för att definiera endast en kontinuerlig process bestående av ett system med tre första ordningens diffrentialekvationer. I detta exempel liksom i allmännhet gäller det för användaren att se till att de inblandade ekvationerna utförs i rätt ordning. Dvs variablerna som uppträder i högerledet ska ha värden som motsvarar det aktuella tillståndet i systemet. I exemplet uppfylls detta eftersom thrust, drag och velocity beräknas före velocity.rate.

Nästa steg blir att skissa hela programmet för simuleringen.

De diskreta händelserna, dvs när de olika raketstegen släpps är beskrivna i huvudprogrammet nedan. Den kontinuerliga tillståndsändringen mellan händelserna definieras av klassen rocketmotion.

```
 1  condis(NOTEXT,"eng")
 2  BEGIN
 3    continuous CLASS rocketmotion; ... ;

 4    REF(variable) mass , velocity, altitude ;
 5    REAL thrust , drag , gravity,
 6         massflow , flowvelocity , area , c1,c2,c3,c4 ;
 7    c1:= ... ; c2:= ... ; c3:= ... ; c4:= ... ;

 8    dtmin:= ... ; dtmax:= ... ;  maxrelerror:= ... ;
 9    mass     :- NEW variable(...);
10    velocity :- NEW variable( 0 );
11    altitude :- NEW variable( 0 );
```

```
12    COMMENT: *** FIRST STAGE *** ;
13    mass.start;  velocity.start;  altitude.start;
14    NEW rocketmotion.start;
15    massflow:= ... ; flowvelocity:= ... ; area:= ... ;
16    Hold(...);

17    COMMENT: *** SECOND STAGE *** ;
18    mass.state:= ... ;
19    massflow:= ... ; flowvelocity:= ... ; area:= ... ;
20    Hold(...);

21    COMMENT: *** THIRD STAGE *** ;
22    mass.state:= ... ;
23    massflow:= ... ; flowvelocity:= ... ; area:= ... ;
24    Hold(...);

25    COMMENT: *** FREE FLIGHT ***;
26    mass.state:= ...;
27    massflow := 0; flowvelocity:=0; area:=...;
28    waituntil(altitude.state < 0);
29  END;
```

Kommentarer:

Rad

8 Eftersom vi i exemplet planerar att använda RK-metoden skall dtmax, dtmin och minst en av maxrelerror resp maxabserror tilldelas värde. Dtmin och dtmax sätts till den undre resp övre tillåtna steglängden under integreringen. Maxrelerror sätts till det maximala tillåtna relativa felet i de olika tillståndsvariablerna.

Eftersom det i CONDIS finns en procedur, som anropas när programmet startar, och som frågar efter integrationsmetod, dtmax, dtmin , maxrel resp maxabserror är dessa satser inte nödvändiga. De är dock tillåtna och medför att de här angivna värdena används, inte de som ges under dialogen.

9-11 De tre variable-objekten mass,velocity och altitude skapas med begynnelsevärden på resp state-värde.

13-14 Raketen börjar ´flyga´ när dessa tre variabler och ett objekt av rocketmotion startas med attributet start.

För att slippa starta alla samhörande instanser av klasserna variable, continuous och sample var för sig skulle man kunna definiera en object CLASS rocket. Alla satserna på raderna 13-14 hade då gått att ersätta med NEW rocket.start. Klassen object finns beskriven i kap 5. I bilaga 3 finns två exempel på användningen av klassen.

15-24 Diskreta ändringar i mass, massflow och flowvelocity äger rum varje gång ett raketsteg släpps .

25-28 Efter att det sista raketsteget släppts upphör dragkraften från motorn vilket modelleras med att massflow och flowvelocity sätts till 0.

Simuleringen fortsätter sedan till raketen landar vilket uttrycks med Waituntil(altitude.state < 0)

I föregående exempel demonstrerades två typer av diskret-kontinuerlig växelverkan nämligen diskret ändring av ´kontinuerliga´ variabler, och möjligheten att låta en diskret händelse inträffa då något villkor på systemets tillstånd är uppfyllt. För detta användes proceduren waituntil.

Den sista typen är möjligheten att starta resp stoppa kontinuerliga processer från en diskret process under simuleringens gång. Denna metod gör det möjligt att byta uppsättningen av differentialekvationer genom att sedan starta ett annat objekt ur klassen continuous och låta denna styra integrationen.

Ett exempel på hur detta kan se ut i fallet med raketen skissas nedan:

```
condis(notext,notext)
BEGIN
  REF(variable)  mass,velocity,altitude;
  REAL thrust,drag,gravity,massflow,flowvelocity,area,
       c1,c2,c3,c4;

  continuous CLASS rocketmotion; ... enl förra exemplet ...;

  continuous CLASS rocketflight;
  BEGIN
    drag := c1*area*EXP(-c2*altitude.state)*velocity.state**2;
    gravity := mass.state / (c3 + c4*altitude.state)**2;
    velocity.rate :=-(drag + gravity) / mass.state;
    altitude.rate := velocity.state;
  END;

  REF(rocketmotion) rockmot;
  ...
  Initiering enl förra exemplet rad 7 -11.
  ...
  COMMENT: *** FIRST STAGE ***;
  mass.start;velocity.start;altitude.start;
  rockmot :- NEW rocketmotion;
  rockmot.start;
  massflow:=...;flowvelocity:=...;area:=...;
  Hold(...);
  ...
  Ändra och tredje steget enl förra exemplet rad 17-24.
  ...
```

```
  COMMENT: *** FREE FLIGHT ***;
  rockmot.stop;
  mass.stop;
  NEW rocketflight.start;
  mass.state := ...; area := ...;
  waituntil(altitude.state < 0);
END *** PROGRAM ***;
```

Lägg märke till att variabeln mass finns kvar under "FREE FLIGHT" trots att den är stoppad. Värdet på dess state kan alltså avläsas och ändras men den ändras inte kontinuerligt som under de tidigare stegen.

8.2 NÅGOT OM IMPLEMENTATIONEN

Simuleringen kontrolleras av ett objekt ´themonitor´ ur klassen monitor.

Themonitor har som uppgift att se till att:

(1) diskreta händelser äger rum vid rätt tidpunkt,
(2) systemtillstånden varierar kontinuerligt mellan händelserna och
(3) information om systemet registreras (CLASS sample).

Themonitor ser också till att alla kontinuerliga delarna av systemet opererar parallellt och är synkroniserade med de kvasiparallella diskreta processerna. Om användaren i sitt program försöker ändra på ordningen på ett otillåtet sätt upptäcks det av themonitor och felhanteringsrutinen anropas.(Se kap 4)

Nedan följer en skiss över themonitor.

```
WHILE "fler planerade händelser" DO
BEGIN
  dt:=0;
  "exekvera alla AKTIVA continuous-objekt";
  "exekvera alla AKTIVA sample-objekt";

  WHILE "ingen händelse skall utföras" DO
  BEGIN
    "tag ett integrationssteg som uppfyller noggrannhetskrav";

    IF "tillståndshändelse inträffade under steget" THEN
    "bestäm tidpunkten för händelsen och reducera steget till den
     tidpunkten";

    "utför begärda registreringar av data (CLASS sample)"
  END;

  "låt händelsen äga rum nu";
END;
```

8.3 ANVÄNDARATTRIBUT

Skissen av CONDIS nedan innehåller de attribut som användaren bör känna till för att utföra en simulering med CONDIS.

Innebörden är i regel självförklarande, men en kort beskrivning till var och ett av attributen är tillfogat i slutet av avsnittet.

```
 1  filequ CLASS condis; VIRTUAL: PROCEDURE prologue;
 2  BEGIN
 3    PROCEDURE error; ... ;
 4    PROCEDURE integrationerror; ... ;
 5    PROCEDURE prologue;...;

 6    Link CLASS continuous; VIRTUAL: PROCEDURE prelude;
 7    BEGIN
 8      PROCEDURE prelude;;
 9      PROCEDURE start; ... ;
10      PROCEDURE stop; ... ;
11      BOOLEAN PROCEDURE active; ... ;
12      PROCEDURE setpriority(r); REAL r; ... ;
13      REAL PROCEDURE priority; ... ;
14      prelude; Detach; execute:; INNER; Resume(...); GOTO execute;
15    END;

16    Link CLASS variable(state); REAL state;
17    BEGIN
18      REAL rate;
19      PROCEDURE start; ... ;
20      PROCEDURE stop; ... ;
21      BOOLEAN PROCEDURE active; ... ;
22      REAL PROCEDURE laststate; ... ;
23      REAL relerror,abserror;
24      relerror:=maxrelerror; abserror:=maxabserror;
25    END;

26    Link CLASS sample; VIRTUAL: PROCEDURE prelude;
27    BEGIN
28      PROCEDURE prelude;;
29      PROCEDURE start; ... ;
30      PROCEDURE stop; ... ;
31      BOOLEAN PROCEDURE active; ... ;
32      PROCEDURE setfrequency(f); REAL f; ... ;
33      REAL PROCEDURE frequency; ... ;
34      REAL PROCEDURE sampletime; ... ;
35      prelude; Detach; execute:; INNER; Resume(...); GOTO execute;
36    END;
```

```
37    Head CLASS object;
38    BEGIN
39      PROCEDURE start;...;
40      PROCEDURE stop;...;
41    END;

42    PROCEDURE waituntil(b); NAME b; BOOLEAN b; ... ;
43    REAL waitpriority; BOOLEAN waitprior;
44    PROCEDURE cancelstateevent(p); REF(Process) p; ... ;
45    REAL dtmin,dtmax,maxrelerror,maxabserror,max_halvings;
46    REAL PROCEDURE Time; ... ;
47    REAL PROCEDURE lasttime; ... ;
48    REAL PROCEDURE dt; ... ;
49    BOOLEAN pc,euler,adams,trapez;
50    PROCEDURE pause; ... ;
51    REF(Process) PROCEDURE nexttimeevent(p); REF(Process) p; ... ;
52    REF(variable) PROCEDURE errorvariable; ... ;
53    REF(continuous) PROCEDURE errorcontinuous; ... ;
54    REF(sample) PROCEDURE errorsample; ... ;
55    REF(Link) PROCEDURE errorobject;...;
56    Link CLASS monitor;...;
57    REF(monitor) themonitor;

58    themonitor :- NEW monitor;
59    max_halvings := 30;
60    prologue;
61    INNER;
62    files.close;
63  END;
```

Kommentarer:

Rad

1 CONDIS är via filequ och simeio en underklass till Simulation. Diskreta händelser i det simulerade systemet beskrivs på vanligt sätt med CLASS Process.

3 ERROR är en procedur som anropas om något fel uppstår som inte berör noggranheten vid integrering. De åtgärder som utförs är beskrivna i kap 4.

4 INTEGRATIONERROR anropas då begärd integrationsnoggranhet inte kan uppfyllas. Se även kap 4.

5 Det första som sker i ett program som använder CONDIS är ett anrop av PROLOGUE. Proceduren innehåller frågor till användaren om önskad integrationsmetod, största resp minsta tillåtna steglängd samt värden på maxabserror och maxrelerror. Proceduren är virtuell och kan alltså omdefinieras av användaren, t ex kan den helt uteslutas genom att definiera en ny upplaga av PROLOGUE med tom procedurkropp.

6-15 CLASS continuous används för att beskriva de kontinuerliga tillståndsändringarna i systemet. Differential- och/eller differensekvationerna som beskriver systemet anges anges i en eller flera underklasser till continuous.

Exempel:

Differentialekvationen :

```
dy/dt = y*t
```

representeras som

```
y.rate:=y.state *time
```

där y är det aktuella variable-objektet.

Differensekvationen:

```
y(i) =3*y(i-1) + 2
```

uttryckS som

```
y.state := 3 * y.laststate + 2.
```

CLASS continuous är en underklass till Link vilket används av CLASS object. Om användaren inte utnyttjar CLASS object kan Link-egenskapen användas fritt.

8 PRELUDE är en virtuell procedur som anropas när ett objekt ur continuous skapas. Proceduren är predefinierad med en tom procedurkropp.

9-10 START och STOP aktiverar resp passiverar aktuellt continuous-objekt. Att objektet är aktivt innebär att dess användardefinierade ekvationer uppdateras ´kontinuerligt´.

11 ACTIVE ger värdet TRUE om continuous-objektet är aktivt, annars ger den värdet FALSE.

12 SETPRIORITY(R) tilldelar continuous-objeketet prioriteten R. De aktiva continuous-objekten exekveras i prioritetsordning (högst först). Proceduren kan alltså användas för att ´sortera ekvationer´.

13 PRIORITY ger aktuellt prioritetsvärde för continuous-objektet.

16-25 CLASS variable används för att representera styckvis kontinuerliga tillståndsvariabler. STATE betecknar det aktuella värdet av en sådan variabel. Om inte CLASS object används kan Link-egenskapen användas fritt.

18 RATE betecknar aktuellt värde av tillståndsvariabelns derivata med avseende på tiden.

19-20 START och STOP aktiverar resp passiverar variabeln. Att variabeln är aktiv innebär att värdet på STATE ändras kontinuerligt beroende på värdet av RATE som beräknas av något aktivt continuous-objekt.

21 ACTIVE ger värdet TRUE om variabeln är aktiv, annars ger den värdet FALSE.

22 LASTSTATE ger värdet på STATE som det var vid aktuellt stegs början.
Denna procedur kan användas för att uttrycka differens-ekvationer.

23-24 RELERROR resp ABSERROR representerar det maximalt tillåtna felet i tillståndsvariabelns tillskott i varje tidssteg när Runge-Kutta- eller Prediktor-Korrektor-integrering används.
Längden av tidsstegen anpassas automatiskt så att det uppskattade integrationsfelet i varje steg blir mindre än:

```
ABS(abserror) + ABS(relerror*state)
```

Skulle inte detta gå att uppfylla med givna krav på minsta tillåtna steglängd resp antal tillåtna halveringar av steglängden anropas felhanteringsproceduren integration-error. Denna är beskriven i kap 4.

Vid de övriga integrationsmetoderna sker ingen felupp-skattning och integrationen sker med konstant steglängd.

När ett object ur CLASS variable skapas sätts värdena på relerror och abserror till samma värde som de globala variablerna maxrelerror resp maxabserror har, men användaren kan sedan tilldela de enskilda variablerna olika värden på relerror och abserror med vanlig tilldelningssats.

26-36 CLASS sample används för att registrera information om systemets uppförande.
Varje sample-objekt kan utföra sina användardefinierade operationer vid specifierade tidpunkter.
Om CLASS object inte används är Link-egenskapen fritt användbar.

28 PRELUDE är en virtuell procedur som anropas när ett sample-objekt skapas. Proceduren är fördefinierad med en tom klasskropp, och kan t ex användas för att skapa resultatfil(er), skriva överskrifter mm.

29-30 START och STOP aktiverar resp passiverar sample-objektet.

31 ACTIVE ger värdet TRUE när sample-objektet är aktivt annars ger den värdet FALSE.

32 SETFREQUENCY(F) tilldelar sample-objektet en ´frekvens´, F.
F anges i simulerad tid och har följande innebörd :

F > 0 : Operationerna äger rum med jämna tidsmellanrum med längden F, samt vid händelsetidpunkter.

F = 0 : Operationerna äger rum i slutat av varje tidssteg, vilket inkluderar händelsetidpunkterna.

F < 0 : Operationerna äger endast rum vid händelsetidpunkterna.

Vid händelsetidpunkterna uförs de användardefinierade operationerna i samtliga aktiva sample-objekt, oberoende av F, både före och efter händelsen.

33 FREQUENCY ger det aktuella värdet på sample-objektets ´frekvens´,F se ovan.

34 SAMPLETIME ger tidpunkten för nästa tidsbestämda exekvering av operationerna om sample-objektet har en positiv frekvens. (Analogt med Process-attributet EVTIME).

37-41 CLASS object kan användas för att strukturera användar-programmet genom att lägga in sammanhörande objekt av klasserna continuous, variable och sample i en länkad lista med ett object av klass object som huvud.

CLASS object är beskrivet i detalj i kap 5, exempel på användningen finns i bilaga 3.

39-40 START och STOP aktiverar resp passiverar alla de objekt av klasserna continuous, variable och sample som finns i listan där objectet är huvud.

42 WAITUNTIL(B) passiverar det aktiva Process-objektet (Current) under en tidsperiod som är planerad att sträcka sig tills det booleska villkoret blir uppfyllt.

Det passiva tillståndet kan emellertid avslutas tidigare om det väntande Process-objektet aktiveras genom Hold, Activate eller annat SIMULATION-kommando.

Om en tillståndshändelse skulle inträffa ´samtidigt´ som en tidshändelse utförs tidshändelsen först.

43 WAITPRIORITY och WAITPRIOR används för att ordna ´samtidiga´ tillståndshändelser i prioritetsordning (Högst-först).

När WAITUNTIL anropas tilldelas den planerade tillståndshändelsen prioriteten WAITPRIORITY.

Värdet på WAITPRIOR vid anropet av WAITUNTIL avgör om en tillståndshändelse skall utföras före (TRUE) eller efter (FALSE) andra tillståndshändelser med samma prioritet.

44 CANCELSTATEEVENT(P) annulerar en tillståndshändelse som är planerad av processen P.

45 DTMIN resp MAX_HALVINGS och DTMAX används för att specifiera minsta resp största tillåtna steglängden vid integration med någon av metoderna Runge-Kutta eller Prediktor-Korrektor.

Vid Runge-Kutta anger DTMIN den minsta tillåtna steglängden. Vid Prediktor-Korrektor anger MAX_HALVINGS högsta antalet halveringar av DTMAX. Kan begärd integrationsnoggrannhet inte uppfyllas med givna krav på minsta steglängd anropas proceduren INTEGRATIONERROR. Denna procedur ger möjlighet till att inspektera variabler av intresse, ändra vissa parametrar, byta integrationsmetod mm. Se även avsnitt 4.2.

Vid övriga integrationsmetoder används fast steglängd, DTMAX.

Tidpunkten för en tillståndshändelse bestämms med ett fel högst lika med DTMIN.

MAXABSERROR och MAXRELERROR används som värde för relerror och abserror för objekt ur CLASS variable som skapas.

DTMAX, DTMIN, MAXRELERROR, MAXABSERROR initieras till värdet 0, men om inte den virtuella proceduren PROLOGUE omdefinieras får användaren vid start av programmet frågor om önskade värden på dessa variabler.

MAX_HALVINGS initieras till 30.

46-48 TIME, LASTTIME och DT ger den aktuella systemtiden, starttiden för det aktuella tidssteget resp den aktuella steglängden.

Det vill säga TIME = LASTTIME + DT.

49 De Booleska variablerna PC, EULER, ADAMS och TRAPEZ används för att välja önskad integrationsmetod enligt följande schema. Observera att Runge-Kutta-England används om PC, EULER, ADAMS and TRAPEZ alla är falska.

PC	EULER	ADAMS	TRAPEZ	Method	Order	Step-size
FALSE	FALSE	FALSE	FALSE	RKE	4	Variable
TRUE	FALSE	FALSE	FALSE	PC	4	Variable
FALSE	TRUE	FALSE	FALSE	EULER	1	Fixed
FALSE	-	TRUE	FALSE	ADAMS	2	Fixed
FALSE	-	FALSE	TRUE	TRAPEZ	2	Fixed
FALSE	-	TRUE	TRUE	HEUN	2	Fixed

RKE : Runge-Kutta-England [13]
PC : Hammings modifierade Prediktor-Korrektor metod med ´diskontinuitetsupptäckt´ enl DHAMDI. [4] och [5]
EULER : First-order Euler [14]
ADAMS : Second order Adams [14]
TRAPEZ : The trapez method, improved Euler [14]
Heun : The improved Heun method [14]

50 PAUSE kan anropas av ett Process-objekt och medför att användardefinierade operatiner hos alla continuous- och sampleobjekt utförs omedelbart med DT = 0. Efteråt återtar den anropande processen kontrollen.

51 NEXTTIMEEVENT skall användas istället för Process-attributet Nextev av säkerhetsskäl.

52-55 ERRORVARIABLE, ERRORCONTINUOUS, ERRORSAMPLE och ERROR-ELEMENT ger vid olika fel användaren en referens till det objekt som har orsakat felet. Se avsnitt 4.2.

56-58 Här deklareras och skapas den MONITOR som skissades i avsnitt 8.2.

59 Max_halvings får värdet 30 om inte användaren tilldelar den något värde.

60 Här anropas proceduren PROLOGUE.

61 Användarens exekverbara satser kommer in här.

62 Innan programmet avslutas stängs alla filer som ligger i listan files. Se kap 6.

9 KÄLLKODER

9.1 CLASS MONITOR ENL KAP 2

```
OPTIONS(/E);
Simulation CLASS combinedsimulation;
VIRTUAL: PROCEDURE integrationerror,simulationerror,
take_a_step,act_error;
BEGIN
REAL PROCEDURE maxreal; maxreal:=1.7014118&38;

REAL dtmin,dtmax,maxabserror,maxrelerror,waitpriority;

REAL PROCEDURE Time; Time:=themonitor.Time;

REAL PROCEDURE dt; dt:=themonitor.dt;

REAL PROCEDURE lasttime; lasttime:=themonitor.lasttime;

BOOLEAN waitprior,euler,adams,trapez,repeatstep;

REF(variable) PROCEDURE errorvariable; ...

REF(continuous) PROCEDURE errorcontinuous; ...

REF(sample) PROCEDURE errorsample; ...

PROCEDURE integrationerror; ...

PROCEDURE simulationerror; ...

PROCEDURE abort; GOTO stopsimulation;

PROCEDURE error(message); VALUE message; TEXT message; ...

CLASS waitnotice(proc,priority); REF(Process) proc;
REAL priority; ...

REF(Process) PROCEDURE nexttimeevent(p); REF(Process) p; ...

PROCEDURE pause; ...

PROCEDURE cancelstateevent(p); REF(Process) p; ...

PROCEDURE waituntil(b); NAME b; BOOLEAN b; ...

Link CLASS variable(state); REAL state; ...

Link CLASS continuous; VIRTUAL: PROCEDURE prelude; ...

Link CLASS sample; VIRTUAL: PROCEDURE prelude; ...
```

```
Process CLASS control1; ...

Process CLASS control2; ...

PROCEDURE take_a_step;
! INTEGRATIONS-PROCEDUREN FÖR RK-ENGLAND;
INSPECT themonitor DO
BEGIN
  h:= 0.5 * dtnow;

  IF NOT error_flag THEN
  BEGIN
    var :- firstvar;
    WHILE var =/= NONE DO INSPECT var DO
    BEGIN
      a1:= h* rate;
      state := oldstate + 0.5*a1;
      var :- sucvar;
    END;
  END *** IF NOT ERROR_FLAG ***
  ELSE
  BEGIN
    var:-firstvar;
    WHILE var=/=NONE DO INSPECT var DO
    BEGIN
      a1:=frac*a1;
      state:=oldstate+0.5*a1;
      rate:=0;
      var:-sucvar;
    END;
  END *** IF ERROR_FLAG ***;

  dt:=0.5*h; Time:=lasttime+dt;
  Resume(firstcont);

  var:-firstvar;
  WHILE var=/=NONE DO INSPECT var DO
  BEGIN
    a2:=h*rate;
    state:=oldstate+0.25*(a1+a2);
    var:-sucvar;
  END;

  Resume(firstcont);

  var:-firstvar;
  WHILE var=/=NONE DO INSPECT var DO
  BEGIN
    a3:=h*rate;
    state:=oldstate+(2*a3-a2);
    var:-sucvar;
  END;
```

```
  dt:=h; Time:=lasttime+dt;
  Resume(firstcont);

  var:-firstvar;
  WHILE var=/=NONE DO INSPECT var DO
  BEGIN
    a4:=h*rate;
    dsh:=((a1+a4)+4*a3)/6;
    state:=oldstate+dsh;
    var:-sucvar;
  END;

  Resume(firstcont);

  var:-firstvar;
  WHILE var=/=NONE DO INSPECT var DO
  BEGIN
    a5:=h*rate;
    ds:=(-dsh+(24*a5-20*a4))+16*(a3-a2);
    a4:=(-a1+4*a3)+(17*a4-23*a5);
    state:=oldstate+(dsh+0.5*a5);
    var:-sucvar;
  END;

  dt:=1.5*h; Time:=lasttime+dt;
  Resume(firstcont);

  var:-firstvar;
  WHILE var=/=NONE DO INSPECT var DO
  BEGIN
    a2:=h*rate;
    state:=oldstate+(dsh+0.25*(a5+a2));
    var:-sucvar;
  END;

  Resume(firstcont);

  var:-firstvar;
  WHILE var=/=NONE DO INSPECT var DO
  BEGIN
    a3:=h*rate;
    state:=oldstate+(ds+(a2-2*a3));
    var:-sucvar;
  END;

  dt:=dtnow; Time:=nexttime;
  Resume(firstcont);

  error_test;
```

```
    IF NOT error_flag THEN
    BEGIN
      Resume(firstcont);

      var:- firstvar;
      WHILE var =/= NONE DO INSPECT var DO
      BEGIN
        ds:=(a4+dsh)+((a5+h*rate)+4*a3)/6;
        var :- sucvar;
      END;


    END *** IF NOT ERROR_FLAG ***;
  END *** TAKE_A_STEP ***;

  PROCEDURE act_error(var); REF(variable) var;
  INSPECT var DO
  INSPECT themonitor DO
  BEGIN
    a4 := ((4*a3 - h*rate) + a4)/90;
    temp := Abs(abserror) + Abs(relerror*(oldstate + dsh));
  END ACT_ERROR;


  Link CLASS monitor;
  BEGIN
    REAL dt,dtnow,dtnext,dtfull,dtlower,
    Time,lasttime,nexttime,nexteventtime,nextsampletime,
    epstime,h,frac,errorratio,temp;

    BOOLEAN active,error_flag,ok;

    REF(Process) stateevent,nextstateevent,nexttimeevent,
    controller1,controller2;

    REF(continuous) firstcont,lastcont,errorcontinuous;

    REF(sample) firstpossample,firstnegsample,firstzerosample,
    errorsample;

    REF(waitnotice) firstwait,lastwait;

    REF(variable) firstvar,errorvariable,var;
```

```
  PROCEDURE error_test;
  !DEL AV INTEGRATIONS-PROCEDUREN FÖR RK-ENGLAND ;
  BEGIN
    error_flag:= ok := FALSE;
    errorratio:=2**5/0.5;

    var :- firstvar;
    WHILE var =/= NONE DO INSPECT var DO
    BEGIN
      act_error(var);

      IF Abs(a4)>temp THEN
      BEGIN
        error_flag := TRUE;

        IF dtnow>dtmin THEN
        BEGIN

          IF h<dtmin THEN
          BEGIN
            frac := dtmin / dtnow;
            dtnow:=dtnext:=dtmin;
          END
          ELSE
          BEGIN
            frac := 0.5;
            dtnow := dtnext := h;
          END;
          nexttime:=lasttime+(epstime+dtnow);
          IF nexttime>nexteventtime
          THEN nexttime:=nexteventtime;

          IF firstvar=/=THIS variable THEN
          BEGIN
            predvar.sucvar:-sucvar;
            IF sucvar=/=NONE THEN sucvar.predvar:-predvar;
            sucvar:-firstvar;
            firstvar:-firstvar.predvar:-
            predvar:-THIS variable;
          END;

          ok :=TRUE;
        END *** IF DTNOW > DTMIN ***
        ELSE
        errorvariable :- THIS variable;
        var :- NONE;
      END *** IF ABS(A4) > TEMP ***
      ELSE
```

```
      BEGIN
        IF temp<errorratio*Abs(a4)
        THEN errorratio:=temp/Abs(a4);

        state:=oldstate+(dsh+(2*a3-a2));
        var:-sucvar;
      END;
    END *** WHILE VAR =/= NONE DO INSPECT VAR DO ***;
    ut:
  END *** ERROR_TEST ***;

  controller1:-NEW control1; controller2:-NEW control2;
  REACTIVATE controller2 AFTER Main;
  REACTIVATE controller1 BEFORE controller2;

  Detach;

  nexttimeevent:-controller2.Nextev;

  WHILE nexttimeevent=/=NONE OR firstwait=/=NONE DO
  BEGIN
    active:=TRUE;

    IF dtmin<0 THEN error("4: DTMIN<0");
    IF dtmin>dtmax THEN error("5: DTMIN>DTMAX");

    dt:=0; lasttime:=Time;

    nexteventtime:=nextsampletime:=
    IF nexttimeevent=/=NONE THEN nexttimeevent.Evtime ELSE maxreal;

    var:-firstvar;
    WHILE var=/=NONE DO INSPECT var DO
    BEGIN
      IF NOT ds=0 THEN
      epsstate:=IF state=oldstate+(epsstate+ds)
      THEN (epsstate+ds)-(state-oldstate) ELSE 0;

      oldstate:=state;
      rate:=ds:=0;
      var:-sucvar;
    END;

    IF firstcont=/=NONE THEN
    BEGIN
      Resume(firstcont);
      IF euler OR adams OR trapez OR dtnext=0 OR dtnext>dtmax
      THEN dtnext:=dtmax
      ELSE IF dtnext<dtmin THEN dtnext:=dtmin;
    END;
```

```
    IF firstpossample=/=NONE THEN Resume(firstpossample);
    IF firstnegsample=/=NONE THEN Resume(firstnegsample);
    IF firstzerosample=/=NONE THEN Resume(firstzerosample);

    IF firstwait=/=NONE AND Time<nexteventtime THEN
    BEGIN
      Resume(firstwait.proc);
      IF stateevent=/=NONE THEN nexteventtime:=Time;
    END;

    WHILE Time<nexteventtime DO
    BEGIN
      lasttime:=Time;
      dtnow:=(nexteventtime-lasttime)-epstime;

      IF firstcont=/=NONE THEN
      BEGIN
        IF dtnow>dtnext THEN
        BEGIN
          dtnow:=dtnext;
          nexttime:=lasttime+(epstime+dtnow);
          IF nexttime>nexteventtime THEN nexttime:=nexteventtime;
        END
        ELSE nexttime:=nexteventtime;
        integration:
        IF euler OR adams OR trapez OR firstvar==NONE THEN
        BEGIN
          IF adams THEN temp:=dtnow-dt;
          var:-firstvar;
          WHILE var=/=NONE DO INSPECT var DO
          BEGIN
            IF NOT ds=0
            THEN epsstate:=(epsstate+ds)-(state-oldstate);

            oldstate:=state;
            ds:=a1:=dtnow*rate;
            IF adams THEN
            BEGIN IF temp=0 THEN ds:=1.5*a1-dsh; dsh:=0.5*a1; END;
            state:=oldstate+(epsstate+ds);
            rate:=0;
            var:-sucvar;
          END;
          dt:=dtnow; Time:=nexttime;
          Resume(firstcont);
          IF trapez OR (adams AND NOT temp=0) THEN
          BEGIN
```

```
        var:-firstvar;
        WHILE var=/=NONE DO INSPECT var DO
        BEGIN
          ds:=0.5*(a1+dtnow*rate);
          state:=oldstate+(epsstate+ds);
          rate:=0;
          var:-sucvar;
        END;
        Resume(firstcont);
      END;
    END
    ELSE
    BEGIN
      var:-firstvar;
      WHILE var=/=NONE DO INSPECT var DO
      BEGIN
        IF NOT  ds=0 THEN
        epsstate:=(epsstate+ds)-(state-oldstate);
        oldstate:=state;
        var:-sucvar;
      END;

      steg:
      take_a_step;

      IF error_flag THEN
      BEGIN
        IF ok THEN GO TO steg;

        integrationerror;

        IF repeatstep OR errorvariable.predvar==NONE THEN
        BEGIN
          var:-firstvar;
          WHILE var=/=NONE DO INSPECT var DO
          BEGIN
            state:=oldstate;
            rate:=2*(a1/dtnow);
            var:-sucvar;
          END;
          error_flag := FALSE;
          GOTO integration;
        END;
        errorratio:=0;
      END *** IF ERROR_FLAG ***
      ELSE
      BEGIN
        IF dtnow=dtnext AND errorratio>1/0.5 THEN
        BEGIN
          dtnext:=(0.5*errorratio)**(1/5)*dtnow;
          IF dtnext>dtmax THEN dtnext:=dtmax;
        END;
```

```
            var :- firstvar;
            WHILE var =/= NONE DO INSPECT var DO
            BEGIN
              state := oldstate +(epsstate + ds);
              var :- sucvar;
            END;

            Resume(firstcont);

          END *** IF NOT ERROR_FLAG ***;
        END *** RKE ***;
      END *** FIRSTCONT=/=NONE ***
      ELSE
      BEGIN
        ...
      END;

      ... RESTEN AV THEMONITOR ÄR HELT ENL ORGINALVERSIONEN

    END *** WHILE TIME<NEXTEVENTTIME ***;

    ...

  END *** WHILE NEXTTIMEEVENT=/=NONE OR FIRSTWAIT=/=NONE ***;

  error("3: THERE ARE NO DISCRETE EVENTS SCHEDULED");
END *** CLASS MONITOR ***;

REF(monitor) themonitor; themonitor:-NEW monitor;
INNER;
stopsimulation:;
END *** CLASS COMBINEDSIMULATION ***;
```

9.2 CONDIS

```
00010   OPTIONS(/E/-W/C);
00020   EXTERNAL REF(Infile) PROCEDURE findinfile;
00030   EXTERNAL REF(Outfile) PROCEDURE findoutfile;
00040   EXTERNAL TEXT PROCEDURE conc,upcase,frontstrip,rest,
00050   checkextension;
00060   EXTERNAL CHARACTER PROCEDURE fetchar,findtrigger;
00070   EXTERNAL LONG REAL PROCEDURE scanreal;
00080   EXTERNAL INTEGER PROCEDURE checkreal,checkint,scanint,ilog;
00090   EXTERNAL BOOLEAN PROCEDURE menu;
00100   EXTERNAL PROCEDURE exit,enterdebug;
00110   EXTERNAL CLASS simeio,filequ;
00120   filequ CLASS condis; VIRTUAL: PROCEDURE prologue;
00130   BEGIN
00140   REAL PROCEDURE maxreal; maxreal:=1.7014118&38;
00150   REAL dtmin,dtmax,maxabserror,maxrelerror,waitpriority;
00160
00170   REAL PROCEDURE Time; Time:=themonitor.Time;
00180
00190   REAL PROCEDURE dt; dt:=themonitor.dt;
00200
00210   REAL PROCEDURE lasttime; lasttime:=themonitor.lasttime;
00220
00230   BOOLEAN waitprior,euler,adams,trapez,repeatstep,pc;
00240
00250   INTEGER max_halvings,disc_halvings;
00260
00270   REF(variable) PROCEDURE errorvariable;
00280   errorvariable:-themonitor.errorvariable;
00290
00300   REF(continuous) PROCEDURE errorcontinuous;
00310   errorcontinuous:-themonitor.errorcontinuous;
00320
00330   REF(sample) PROCEDURE errorsample;
00340   errorsample:-themonitor.errorsample;
00350
00360   REF(Link) errorelement;
00370
00380   PROCEDURE leave;
00390   BEGIN
00400     themonitor.resumesample;
00410     files.Close;
00420     exit(1);
00430   END *** LEAVE ***;
00440
00450   PROCEDURE simddt;
00460   BEGIN
00470     themonitor.resumesample;
00480     files.Close;
00490     enterdebug(FALSE);
00500   END *** SIMDDT ***;
00510
```

```
00520   PROCEDURE errormessage(message);VALUE message ;TEXT message;
00530   BEGIN
00540     Outimage;Outtext("***COMBINEDSIMULATION");Outimage;
00550     Outtext("***ERROR ");Outtext(message);Outimage;
00560     Outtext("***ENCOUNTERED AT TIME ");Outreal(Time,5,11);
00570     Outimage;Outimage;
00580     Outtext("            DT           DTMIN          DTMAX");
00590     Outimage;Outimage;
00600     Outreal(dt,5,15);Outreal(dtmin,5,15);Outreal(dtmax,5,15);
00610     Outimage;Outimage;
00620   END *** ERRORMESSAGE ***;
00630
00640   PROCEDURE new_dtmax_dtmin;
00650   BEGIN  TEXT t; t:- Blanks(10);
00660     Outtext("Actual dtmin :");Outreal(dtmin,4,10);Outimage;
00670     t.Putreal(dtmin,5);
00680     request("New dtmin :",t,realinput(dtmin,dtmin>0),
00690     NOTEXT,help("Dtmin indicates the minimum steplength during"
00700     "RK-integration. It has no meaning during other methods."));
00710     Outtext("Actual dtmax :");Outreal(dtmax,4,10);Outimage;
00720     t.Putreal(dtmax,5);
00730     request("New dtmax :",t,realinput(dtmax,dtmax>=dtmin),
00740     "Dtmax must be >= dtmin.",help("During RK- and PC-intgration "
00750     "dtmax is the maximum steplength. During Euler,Adams or Trapez "
00760     "it is the constant steplength"));
00770   END *** NEW DTMAX DTMIN ***;
00780
00790
00800   PROCEDURE integrationerror(type); VALUE type; TEXT type;
00810   BEGIN
00820     TEXT t;
00830     BOOLEAN flag,cont;
00840     INTEGER index,i;
00850     TEXT ARRAY table[1:8],choice[0:8];
00860
00870     SWITCH action :=
00880     lmethod,ldtmin,lerror,lsimddt,lstop,lmax_halvings,
00890     ldisc_halvings,lcont;
00900
00910     choice[0] :- Copy("Do you want to :");
00920     table[1]:-Copy("METHOD"); choice[1]:-Copy("(Change) method");
00930     table[2]:-Copy("DTMIN"); choice[2]:-Copy("(Change) dtmin (and "
00940     "dtmax)");
00950     table[3]:-Copy("ERROR");
00960     choice[3]:-Copy("(Change rel and abs) error"
00970     " (for the variable that caused the error)");
00980     table[4]:-Copy("SIMDDT"); choice[4]:-Copy("(Enter) SIMDDT");
00990     table[5]:-Copy("STOP"); choice[5]:-Copy("Stop (execution)");
01000     table[6]:-Copy("MAX_HALVINGS");
01010     choice[6]:-Copy("(Change) max_halvings");
01020     table[7]:-Copy("DISC_HALVINGS");
01030     choice[7]:-Copy("(Change) disc_halvings");
01040     table[8]:-Copy("CONTINUE");
01050     choice[8]:-Copy("Continue (execution)");
01060
```

```
01070     t:-
01080     conc("1: THE REQUESTED INTEGRATION ACCURACY CAN NOT BE ACHIVED "
01090     ," :",type);
01100     errormessage(t);
01110
01120     request("Do you want to stop execution ?","yes",boolinput(flag),
01130     NOTEXT,help("If you choose to continue you´ll be able to change"
01140     " integration parameters, change method or enter SIMDDT"));
01150
01160     IF flag THEN leave;
01170
01180     FOR i := 0,1,2,4,5,3 DO Outtext(conc(choice[i]," , "));
01190     IF type = "PC" THEN FOR i:= 6,7 DO
01200     Outtext(conc(choice[i]," , "));
01210     Outimage;
01220
01230     WHILE NOT cont DO
01240     BEGIN
01250       request("*",nodefault,textinput(t,menu(t,index,table,8)),
01260       commandmessage(index),commandhelp(table,8));
01270
01280       GO TO action(index);
01290
01300       lmethod:
01310       flag := FALSE;
01320       method;
01330       IF euler OR adams OR trapez THEN
01340       BEGIN
01350         Outtext("Do you want to change dtmax "); Outimage;
01360         request("?","yes",boolinput(flag),NOTEXT,
01370         help("When you are using EULER, ADAMS or TRAPEZ the "
01380         " integration proceeds with a konstant step of DTMAX !"));
01390       END;
01400       IF NOT flag THEN GO TO Out;
01410
01420       ldtmin:
01430       new_dtmax_dtmin;
01440       GO TO Out;
01450
01460       lerror:
01470       INSPECT themonitor.errorvariable DO
01480       BEGIN TEXT t; t:- Blanks(10);
01490         Outtext("Actual values:"); Outimage;
01500         Outtext("    Abserror+Abs( Relerror *   State)=Max allowed "
01510         "< Estimated");Outimage;
01520         Setpos(41);Outtext("error       error");Outimage;
01530         Outreal(abserror,3,11);
01540         Outreal(relerror,3,13);
01550         Outreal(state,3,12);
01560         Outreal(themonitor.temp,3,12);
01570         Outreal(a4,3,12);Outimage;
01580         t.Putreal(relerror,4);
```

```
01590         request("New relerror :",t,
01600         realinput(relerror,relerror >=0),NOTEXT,nohelp);
01610         t.Putreal(abserror,4);
01620         request("New abserror :",t,realinput(abserror,abserror > 0),
01630         "Abserror must not be 0, to avoid problems when state is 0."
01640         ,nohelp);
01650       END;
01660       GO TO Out;
01670
01680       lsimddt:
01690       Outtext("If you give SIMDDT the comand ´proceed´ "
01700       "you´ll get the last menu again.");Outimage;
01710       enterdebug(TRUE);
01720       GO TO Out;
01730
01740       lstop: leave;
01750
01760       lmax_halvings:
01770       Outtext("Actual value of max_halvings :");
01780       Outint(max_halvings,5);Outimage;Outimage;
01790       request("New max_halvings :","30",
01800       intinput(max_halvings,max_halvings>0),NOTEXT,
01810       help("Max_halvings indicates the number of bisections"
01820       " of the steplength during PC-integration, that is allowed"
01830       " before the execution is interrupted with an error."));
01840       GO TO Out;
01850
01860       ldisc_halvings:
01870       Outtext("Actual value of disc_halvings :");
01880       Outint(disc_halvings,5);Outimage;Outimage;
01890       request("New disc_halvings :","10",
01900       intinput(disc_halvings,disc_halvings>0),NOTEXT,
01910       help("Disc_halving indicates the number of bisections"
01920       " of the steplength, during PC-integration, before a "
01930       "discontinuity is said to be found."));
01940       GO TO Out;
01950
01960       lcont:
01970       cont := TRUE;
01980
01990       Out:
02000       IF NOT cont THEN
02010       BEGIN
02020         FOR i := 0,8,1,2,4,5,3 DO Outtext(conc(choice[i]," , "));
02030         IF type = "PC" THEN
02040         FOR i := 6,7 DO Outtext(conc(choice[i]," , "));Outimage;
02050       END *** IF NOT CONT ***;
02060     END *** WHILE NOT CONT DO ***;
02070   END *** INTEGRATIONERROR ***;
02080
```

```
02090   PROCEDURE error(nr,message);VALUE message;TEXT message;
02100   INTEGER nr;
02110   BEGIN
02120     BOOLEAN flag;
02130
02140     errormessage(message);
02150
02160     IF nr = 2 OR nr = 3 OR nr >= 7 AND NOT nr = 21 THEN
02170     BEGIN
02180       request("Do you want to enter SIMDDT ?","no",
02190       boolinput(flag),NOTEXT,
02200       help("You are probably not able to correct"
02210       " this error during this execution. However you may"
02220       " enter SIMDDT which allows you to study properties of"
02230       " the program. You will have all the possibilities included"
02240       " in SIMDDT, except that you will not be allowed to"
02250       " restart the execution by ´proceed´."));
02260
02270       IF flag THEN
02280       BEGIN
02290         IF nr >= 11 AND nr <= 13 THEN
02300         BEGIN
02310           Outtext("The variable ´themonitor.errorcontinuous´"
02320           " will contain a  reference to the instance of"
02330           " class continuous that caused the error"); Outimage;
02340         END;
02350
02360         IF nr >= 14 AND nr <= 16 THEN
02370         BEGIN
02380           Outtext("The variable ´themonitor.errorsample´"
02390           " will contain a  reference to the instance of"
02400           " class sample that caused the error"); Outimage;
02410         END;
02420         simddt;
02430       END
02440       ELSE leave;
02450     END *** IF NR = 2,3 OR >= 7 ***;
02460
02470     IF nr = 6 THEN
02480     BEGIN
02490       request("Do you want to stop execution ?","yes",
02500       boolinput(flag),NOTEXT,
02510       help("One instance of class sample has a positive"
02520       " ´frequency´ with to small magnitude. If you choose to "
02530       "continue you will be able to change the frequency"
02540       " or invoke SIMDDT"));
02550
02560       IF flag THEN
02570       BEGIN
02580         files.Close;
02590         exit(1);
02600       END;
```

```
02610       request("Do you want to change the frequency ?","yes",
02620       boolinput(flag),NOTEXT,
02630       help("If you answer YES you´ll be asked for new"
02640       " frequency then execution continues. Otherwise"
02650       " SIMDDT will be called, and ´themonitor.erorrsample´ will"
02660       " then contain a reference to the sample that"
02670       " caused the error"));
02680
02690       IF flag THEN
02700       BEGIN
02710         Outtext("Actual value of frequency :");
02720         Outreal(themonitor.errorsample.frq,5,10);
02730         Outimage;
02740         request("New value :",nodefault,
02750         realinput(themonitor.errorsample.frq,
02760         themonitor.errorsample.frq > 0),
02770         "Frequency must be > 0 . The type of sample can"
02780         " not be changed in this situation.",
02790         help("Give a positive real, greater than the actual value."
02800         " If you make too small, you will get the same errormessage"
02810         " again."));
02820       END
02830       ELSE enterdebug(TRUE);
02840     END *** IF NR = 6 ***;
02850
02860     IF nr = 4 OR nr = 5 THEN new_dtmax_dtmin;
02870
02880     IF nr = 21 THEN
02890     BEGIN
02900       request("Do you want to enter SIMDDT ?","no",boolinput(flag),
02910       NOTEXT,help("If you choose to enter SIMDDT you may"
02920       " inspect the erroneous element through the reference"
02930       " ´errorelement´. Otherwise execution will continue."));
02940
02950       IF flag THEN enterdebug(TRUE);
02960     END *** IF NR = 21 ***;
02970
02980
02990   END *** PROCEDURE ERROR ***;
03000
```

```
03010   PROCEDURE method;
03020   BEGIN
03030     TEXT t;
03040     TEXT ARRAY table[1:6];
03050     INTEGER index;
03060     SWITCH action := lrk,lpc,leuler,ladams,ltrapez,lheun;
03070     table[1] :- Copy("RK");
03080     table[2] :- Copy("PC");
03090     table[3] :- Copy("EULER");
03100     table[4] :- Copy("ADAMS");
03110     table[5] :- Copy("TRAPEZ");
03120     table[6] :- Copy("HEUN");
03130     pc:=euler:=adams:=trapez:= FALSE;
03140
03150     Outtext("Which integrationmethod do you want ?");Outimage;
03160     Outtext("        RK (Runge-Kutta-England)");Outimage;
03170     Outtext("        PC (Predictor-correktormethod type DHAMDI)");
03180     Outimage;
03190     Outtext("        EULER                    ");Outimage;
03200     Outtext("        ADAMS                    ");Outimage;
03210     Outtext("        TRAPEZ                   ");Outimage;
03220     Outtext("        HEUN                     ");Outimage;
03230     request("*","RK",textinput(t,menu(t,index,table,6)),
03240     commandmessage(index),commandhelp(table,6));
03250
03260     GO TO action[index];
03270
03280     lpc:     pc:= TRUE; GO TO lrk;
03290     leuler:  euler:= TRUE; GO TO lrk;
03300     ladams:  adams := TRUE; GO TO lrk;
03310     ltrapez: trapez := TRUE; GO TO lrk;
03320     lheun:   trapez := adams := TRUE;
03330     lrk:
03340   END *** PROCEDURE METHOD ***;
03350
03360   PROCEDURE prologue;
03370   BEGIN
03380     method;
03390     new_dtmax_dtmin;
03400     request("Max relerror ?",nodefault,realinput(maxrelerror,
03410     maxrelerror>=0),"Max relerror >= 0 !",nohelp);
03420     request("Max abserror ?",nodefault,realinput(maxabserror,
03430     maxabserror > 0),"Max abserror > 0 !",
03440     help("Max Abserror must be > 0 to avoid problems when state"
03450     " is 0."));
03460
03470   END *** PROCEDURE PROLOG ***;
03480
03490
03500   CLASS waitnotice(proc,priority); REF(Process) proc; REAL priority;
03510   BEGIN REF(waitnotice) predwait,sucwait; END;
03520
03530
```

```
03540   REF(Process) PROCEDURE nexttimeevent(p); REF(Process) p;
03550   IF p=/=NONE THEN
03560   nexttimeevent:-
03570   IF themonitor.active OR p.Nextev==themonitor.controller1
03580   THEN themonitor.controller2.Nextev
03590   ELSE p.Nextev;
03600
03610   PROCEDURE pause;
03620   BEGIN
03630     IF themonitor.active THEN error(8,"8: ILLEGAL USE OF PAUSE");
03640     IF Current.Nextev=/=themonitor.controller1 THEN
03650     BEGIN
03660       REACTIVATE themonitor.controller1 AFTER Current;
03670       REACTIVATE themonitor.controller2 AFTER
03680       themonitor.controller1;
03690     END;
03700     REACTIVATE Current AFTER themonitor.controller2;
03710   END *** PROCEDURE PAUSE ***;
03720
03730   PROCEDURE cancelstateevent(p); REF(Process) p;
03740   IF themonitor.active THEN error(9,"9: ILLEGAL USE OF"
03750   " CANCELSTATEEVENT")
03760   ELSE
03770   INSPECT p DO
03780   BEGIN
03790     REF(waitnotice) w;
03800     w:-themonitor.lastwait;
03810     WHILE (IF w=/=NONE THEN w.proc=/=THIS Process ELSE FALSE) DO
03820     w:-w.predwait;
03830
03840     INSPECT w DO
03850     BEGIN
03860       IF predwait==NONE THEN themonitor.firstwait:-sucwait
03870       ELSE predwait.sucwait:-sucwait;
03880
03890       IF sucwait==NONE THEN themonitor.lastwait:-predwait
03900       ELSE sucwait.predwait:-predwait;
03910
03920       predwait:-sucwait:-THIS waitnotice;
03930     END;
03940   END *** PROCEDURE CANCELSTATEEVENT ***;
03950
```

```
03960   PROCEDURE waituntil(b); NAME b; BOOLEAN b;
03970   INSPECT NEW waitnotice(Current,waitpriority) DO
03980   BEGIN
03990     INSPECT themonitor DO
04000     IF active THEN error(10,"10: ILLEGAL USE OF WAITUNTIL") ELSE
04010     IF firstwait==NONE THEN firstwait:-lastwait:-THIS waitnotice
04020     ELSE
04030     IF priority>firstwait.priority
04040     OR (waitprior AND  priority=firstwait.priority) THEN
04050     BEGIN
04060       sucwait:-firstwait;
04070       firstwait:-sucwait.predwait:-THIS waitnotice;
04080     END
04090     ELSE
04100     BEGIN
04110       predwait:-lastwait;
04120       WHILE priority>predwait.priority DO
04130       predwait:-predwait.predwait;
04140
04150       IF waitprior THEN
04160       WHILE priority=predwait.priority DO
04170       predwait:-predwait.predwait;
04180
04190       sucwait:-predwait.sucwait; predwait.sucwait:-THIS waitnotice;
04200
04210       IF sucwait=/=NONE THEN sucwait.predwait:-THIS waitnotice
04220       ELSE lastwait:-THIS waitnotice;
04230     END;
04240
04250     Passivate;
04260
04270     WHILE proc=/=Current DO
04280     IF b THEN BEGIN themonitor.stateevent:-proc; Resume(themonitor);
04290     END
04300     ELSE Resume(IF sucwait=/=NONE THEN sucwait.proc ELSE
04310     themonitor);
04320
04330     IF predwait==NONE THEN themonitor.firstwait:-sucwait
04340     ELSE predwait.sucwait:-sucwait;
04350
04360     IF sucwait==NONE THEN themonitor.lastwait:-predwait
04370     ELSE sucwait.predwait:-predwait;
04380   END *** PROCEDURE WAITUNTIL ***;
04390
```

```
04400   Link CLASS variable(state); REAL state;
04410   BEGIN
04420     REAL rate,
04430     abserror,relerror,
04440     oldstate,epsstate,ds,dsh,a1,a2,a3,a4,a5,p3_c3,p4,c4,
04450     state_2,state_1,state0,state1,state2,state3,
04460     rate_2,rate_1,rate0,rate1,rate2,rate3;
04470
04480     REF(variable) predvar,sucvar;
04490
04500     REAL PROCEDURE laststate; laststate:=oldstate;
04510
04520     BOOLEAN PROCEDURE active; active:=predvar=/=NONE;
04530
04540     PROCEDURE start;
04550     INSPECT themonitor DO
04560     IF predvar==NONE THEN
04570     BEGIN
04580       IF firstvar==NONE THEN firstvar:-predvar:-THIS variable
04590       ELSE
04600       BEGIN
04610         sucvar:-firstvar;
04620         firstvar:-firstvar.predvar:-predvar:-THIS variable;
04630       END;
04640     END *** PROCEDURE START ***;
04650
04660     PROCEDURE stop;
04670     IF predvar=/=NONE THEN
04680     BEGIN
04690       IF predvar=/=THIS variable THEN predvar.sucvar:-sucvar
04700       ELSE themonitor.firstvar:-predvar:-sucvar;
04710
04720       IF sucvar=/=NONE
04730       THEN BEGIN sucvar.predvar:-predvar; sucvar:-NONE; END;
04740
04750       predvar:-NONE;
04760       rate:=0;
04770     END *** PROCEDURE STOP ***;
04780
04790     abserror:=maxabserror; relerror:=maxrelerror;
04800   END *** CLASS VARIABLE ***;
04810
```

```
04820   Link CLASS continuous; VIRTUAL: PROCEDURE prelude;
04830   BEGIN
04840     REAL pri;
04850
04860     REF(continuous) predcont; REF(Link) succont;
04870
04880     PROCEDURE prelude; ;
04890
04900     REAL PROCEDURE priority; priority:=pri;
04910
04920     PROCEDURE setpriority(r); REAL r;
04930     IF themonitor.active THEN
04940     BEGIN
04950       themonitor.errorcontinuous:-THIS continuous;
04960       error(11,"11: ILLEGAL USE OF SETPRIORITY (CLASS CONTINUOUS)");
04970     END
04980     ELSE
04990     BEGIN
05000       pri:=r;
05010       IF succont=/=NONE THEN BEGIN stop; start; END;
05020     END *** PROCEDURE SETPRIORITY ***;
05030
05040     BOOLEAN PROCEDURE active; active:=succont=/=NONE;
05050
05060     PROCEDURE start;
05070     INSPECT themonitor DO
05080     IF active THEN
05090     BEGIN
05100       errorcontinuous:-THIS continuous;
05110       error(12,"12: ILLEGAL USE OF START (CLASS CONTINUOUS)");
05120     END
05130     ELSE
05140     IF succont==NONE THEN
05150     BEGIN
05160       IF firstcont==NONE THEN
05170       BEGIN
05180         firstcont:-lastcont:-THIS continuous; succont:-THIS monitor;
05190       END
05200       ELSE
05210       IF pri>firstcont.pri THEN
05220       BEGIN
05230         succont:-firstcont;
05240         firstcont:-firstcont.predcont:-THIS continuous;
05250       END
05260       ELSE
```

```
05270       BEGIN
05280         predcont:-lastcont;
05290         WHILE pri>predcont.pri DO predcont:-predcont.predcont;
05300
05310         succont:-predcont.succont;
05320         predcont.succont:-THIS continuous;
05330
05340         IF succont==THIS monitor THEN lastcont:-THIS continuous
05350         ELSE succont QUA continuous.predcont:-THIS continuous;
05360       END;
05370     END *** PROCEDURE START ***;
05380
05390     PROCEDURE stop;
05400     INSPECT themonitor DO
05410     IF active THEN
05420     BEGIN
05430       errorcontinuous:-THIS continuous;
05440       error(13,"13: ILLEGAL USE OF STOP (CLASS CONTINUOUS)");
05450     END
05460     ELSE
05470     IF succont=/=NONE THEN
05480     BEGIN
05490       IF predcont=/=NONE THEN predcont.succont:-succont
05500       ELSE firstcont:-IF succont==THIS monitor THEN NONE ELSE
05510       succont;
05520
05530       IF succont==THIS monitor THEN lastcont:-predcont
05540       ELSE succont QUA continuous.predcont:-predcont;
05550
05560       succont:-predcont:-NONE;
05570     END *** PROCEDURE STOP ***;
05580
05590     prelude;
05600     Detach;
05610
05620     execute:;
05630     INNER;
05640     Resume(succont);
05650     GOTO execute;
05660   END *** CLASS CONTINUOUS ***;
05670
```

```
05680   Link CLASS sample; VIRTUAL: PROCEDURE prelude;
05690   BEGIN
05700     REAL frq,smptime;
05710
05720     REF(sample) predsmp; REF(Link) sucsmp;
05730
05740     PROCEDURE prelude; ;
05750
05760     REAL PROCEDURE sampletime; sampletime:=smptime;
05770
05780     REAL PROCEDURE frequency; frequency:=frq;
05790
05800     PROCEDURE setfrequency(f); REAL f;
05810     INSPECT themonitor DO
05820     IF active THEN
05830     BEGIN
05840       errorsample:-THIS sample;
05850       error(14,"14: ILLEGAL USE OF SETFREQUENCY (CLASS SAMPLE)");
05860     END
05870     ELSE
05880     IF sucsmp==NONE OR Sign(frq)=Sign(f) THEN
05890     BEGIN smptime:=Time; frq:=f; END
05900     ELSE BEGIN stop; frq:=f; start; END;
05910
05920     BOOLEAN PROCEDURE active; active:=sucsmp=/=NONE;
05930
05940     PROCEDURE start;
05950     INSPECT themonitor DO
05960     IF active THEN
05970     BEGIN
05980       errorsample:-THIS sample;
05990       error(15,"15: ILLEGAL USE OF START (CLASS SAMPLE)");
06000     END
06010     ELSE
06020     IF sucsmp==NONE THEN
06030     BEGIN
06040       REF(sample) First;
06050       smptime:=Time;
06060       First:-IF frq>0 THEN firstpossample ELSE
06070       IF frq<0 THEN firstnegsample
06080       ELSE firstzerosample;
06090
06100       IF First==NONE THEN
06110       BEGIN First:-THIS sample; sucsmp:-THIS monitor; END
06120       ELSE BEGIN sucsmp:-First; First:-First.predsmp:-THIS sample;
06130       END;
06140
06150       IF frq>0 THEN firstpossample:-First ELSE
06160       IF frq<0 THEN firstnegsample:-First
06170       ELSE firstzerosample:-First;
06180     END *** PROCEDURE START ***;
06190
```

```
06200     PROCEDURE stop;
06210     INSPECT themonitor DO
06220     IF active THEN
06230     BEGIN
06240       errorsample -THIS sample;
06250       error(16,"16: ILLEGAL USE OF STOP (CLASS SAMPLE)");
06260     END
06270     ELSE
06280     IF sucsmp=/=NONE THEN
06290     BEGIN
06300       REF(sample) First;
06310       IF predsmp=/=NONE THEN predsmp sucsmp -sucsmp
06320       ELSE
06330       BEGIN
06340         First:-IF sucsmp==THIS monitor THEN NONE ELSE sucsmp;
06350         IF frq>0 THEN firstpossample -First ELSE
06360         IF frq<0 THEN firstnegsample  First
06370         ELSE firstzerosample. First;
06380       END;
06390
06400       IF sucsmp=/=THIS monitor
06410       THEN sucsmp QUA sample predsmp -predsmp;
06420
06430       sucsmp -predsmp -NONE;
06440     END *** PROCEDURE STOP ***;
06450
06460     prelude;
06470     Detach;
06480
06490     execute:;
06500     INNER;
06510     IF frq<=0 THEN Resume(sucsmp)
06520     ELSE INSPECT themonitor DO
06530     BEGIN
06540       IF smptime<=Time THEN
06550       BEGIN  REAL oldsmptime;
06560         oldsmptime:=smptime;
06570         smptime =oldsmptime+frq;
06580         errorsample  THIS sample;
06590         WHILE smptime <= Time DO
06600         BEGIN
06610           error(6 "6  FREQUENCY IS TOO SMALL TO ADVANCE TIME "
06620           "(CLASS SAMPLE)");
06630           smptime  = oldsmptime + frq;
06640         END;
06650         errorsample    NONE;
06660       END;
06670       repeat:
06680       IF smptime<nextsampletime THEN nextsampletime =smptime;
06690       Resume(sucsmp);
06700       IF smptime>Time AND Time<nexteventtime AND dt>0 THEN GOTO
06710       repeat;
06720     END;
06730     GOTO execute;
06740   END *** CLASS SAMPLE ***;
```

```
06750   Head CLASS object;
06760   BEGIN
06770     PROCEDURE start;
06780     BEGIN
06790       REF(Link) part;
06800       part :- First;
06810       WHILE part =/= NONE DO
06820       BEGIN
06830         INSPECT part
06840         WHEN variable DO start
06850         WHEN continuous DO start
06860         WHEN sample DO start
06870         OTHERWISE
06875         BEGIN
06878           errorelement :- part;
06880           error(21,"21:ELEMENT OF ILLEGAL KIND IN CLASS OBJEKT.");
06885         END;
06890         part :- part.Suc;
06900       END *** WHILE PART =/= NONE DO ***;
06910     END *** START ***;
06920
06930     PROCEDURE stop;
06940     BEGIN
06950       REF(Link) part;
06960       part :- First;
06970       WHILE part =/= NONE DO
06980       BEGIN
06990         INSPECT part
07000         WHEN variable DO stop
07010         WHEN continuous DO stop
07020         WHEN sample DO stop
07030         OTHERWISE
07034         BEGIN
07037           errorelement :- part;
07040           error(21,"21:ELEMENT OF ILLEGAL KIND IN CLASS OBJEKT.");
07045         END;
07050         part :- part.Suc;
07060       END *** WHILE PART =/= NONE DO ***;
07070     END *** STOP ***;
07080   END *** CLASS OBJECT ***;
07090
07100   Process CLASS control1;
07110   WHILE NOT themonitor.active DO Resume(themonitor);
07120
07130   Process CLASS control2;
07140   IF themonitor.controller1.Idle THEN
07150   BEGIN
07160     IF themonitor.controller1.Terminated
07170     THEN error(17,"17: ILLEGAL USE OF (RE)ACTIVATE");
07180     error(18,"18: ILLEGAL USE OF PASSIVATE (OR CANCEL(CURRENT))");
07190   END
07200   ELSE error(19,"19: ILLEGAL USE OF HOLD (OR REACTIVATE CURRENT)");
07210
```

```
07220   Link CLASS monitor;
07230   BEGIN
07240     REAL dt,dtnow,dtnext,dtfull,dtlower,
07250     Time,lasttime,nexttime,nexteventtime,nextsampletime,
07260     epstime,h,frac,errorratio,temp;
07270
07280     BOOLEAN
07290     active,pc_possible,last_pc,firsttime,start_rk,ok,disc_imp;
07300
07310     INTEGER normal_pc_steps,halvings;
07320
07330     REF(Process) stateevent,nextstateevent,nexttimeevent,
07340     controller1,controller2;
07350
07360     REF(continuous) firstcont,lastcont,errorcontinuous;
07370
07380     REF(sample) firstpossample,firstnegsample,firstzerosample,
07390     errorsample;
07400
07410     REF(waitnotice) firstwait,lastwait;
07420
07430     REF(variable) firstvar,errorvariable,var;
07440
07450     PROCEDURE resumesample;
07460     BEGIN
07470       IF firstpossample =/= NONE THEN Resume(firstpossample);
07480       IF firstnegsample =/= NONE THEN Resume(firstnegsample);
07490       IF firstzerosample =/= NONE THEN Resume(firstzerosample);
07500     END *** RESUMESAMPLE ***;
07510
07520     controller1:-NEW control1; controller2:-NEW control2;
07530     REACTIVATE controller2 AFTER Main;
07540     REACTIVATE controller1 BEFORE controller2;
07550
07560     Detach;
07570
07580     nexttimeevent:-controller2.Nextev;
07590
07600     WHILE nexttimeevent =/= NONE OR firstwait=/= NONE DO
07610     BEGIN
07620       start_rk := TRUE;
07630       active:=TRUE;
07640
07650       IF dtmin<0 THEN error(4,"4: DTMIN<0");
07660       IF dtmin>dtmax THEN error(5,"5: DTMIN>DTMAX");
07670
07680       dt:=0; lasttime:=Time;
07690
07700       nexteventtime:=nextsampletime:=
07710       IF nexttimeevent=/=NONE THEN nexttimeevent.Evtime ELSE
07720       maxreal;
07730
```

```
07740       var:-firstvar;
07750       WHILE var=/=NONE DO INSPECT var DO
07760       BEGIN
07770         IF NOT ds=0 THEN
07780         epsstate:=IF state=oldstate+(epsstate+ds)
07790         THEN (epsstate+ds)-(state-oldstate) ELSE 0;
07800
07810         oldstate:=state;
07820         rate:=ds:=0;
07830         var:-sucvar;
07840       END;
07850
07860       IF firstcont=/=NONE THEN
07870       BEGIN
07880         Resume(firstcont);
07890         IF euler OR adams OR trapez OR dtnext=0 OR dtnext>dtmax
07900         THEN dtnext:=dtmax
07910         ELSE IF dtnext<dtmin THEN dtnext:=dtmin;
07920       END;
07930
07940       IF firstpossample=/=NONE THEN Resume(firstpossample);
07950       IF firstnegsample=/=NONE THEN Resume(firstnegsample);
07960       IF firstzerosample=/=NONE THEN Resume(firstzerosample);
07970
07980       IF firstwait=/=NONE AND Time<nexteventtime THEN
07990       BEGIN
08000         Resume(firstwait.proc);
08010         IF stateevent=/=NONE THEN nexteventtime:=Time;
08020       END;
08030
08040       WHILE Time < nexteventtime DO
08050       BEGIN
08060         lasttime := Time;
08070         dtnow := (nexteventtime-lasttime)-epstime;
08080         IF firstcont =/= NONE THEN
08090         BEGIN
08100
08110           !*** ASSIGN VALUES TO DTNOW AND NEXTTIME ACCORDING METHOD***;
08120           IF NOT pc_possible THEN
08130           BEGIN
08140             IF dtnow > dtnext THEN
08150             BEGIN
08160               dtnow := dtnext;
08170               nexttime := lasttime + (epstime + dtnow);
08180               IF nexttime > nexteventtime THEN nexttime := nexteventtime;
08190             END
08200             ELSE
08210             BEGIN
08220               start_rk := TRUE;
08230               nexttime := nexteventtime;
08240             END;
08250           END *** NOT PC_POSSIBLE ***
08260           ELSE
```

```
08270           BEGIN !*** PC_POSSIBLE *** ;
08280             IF dtnow < h THEN
08290             BEGIN
08300               pc_possible := FALSE;
08310               start_rk := TRUE;
08320               nexttime := nexteventtime;
08330             END
08340             ELSE
08350             BEGIN
08360               dtnow := h;
08370               nexttime := lasttime + (h + epstime);
08380             END;
08390           END *** ASSIGN VALUES TO DTNOW AND NEXTTIME ***;
08400
08410           firsttime := TRUE;
08420           ok := FALSE;
08430           WHILE NOT ok DO
08440           BEGIN
08450             ok := TRUE;
08460
08470             IF euler OR adams OR trapez OR firstvar == NONE THEN
08480             BEGIN !*** TAKE A STEP ACCORDING TO METHOD ***;
08490               IF adams THEN temp:=dtnow-dt;
08500               var:-firstvar;
08510               WHILE var=/=NONE DO INSPECT var DO
08520               BEGIN
08530                 IF NOT ds=0
08540                 THEN epsstate:=(epsstate+ds)-(state-oldstate);
08550
08560                 oldstate:=state;
08570                 ds:=a1:=dtnow*rate;
08580
08590                 IF adams THEN
08600                 BEGIN
08610                   IF temp=0 THEN ds:=1.5*a1-dsh;
08620                   dsh:=0.5*a1;
08630                 END;
08640                 state:=oldstate+(epsstate+ds);
08650                 rate:=0;
08660                 var:-sucvar;
08670               END;
08680
08690               dt:=dtnow; Time:=nexttime;
08700               Resume(firstcont);
08710
```

```
08720              IF trapez OR (adams AND NOT temp=0) THEN
08730              BEGIN
08740                var:-firstvar;
08750                WHILE var=/=NONE DO INSPECT var DO
08760                BEGIN
08770                  ds:=0.5*(a1+dtnow*rate);
08780                  state:=oldstate+(epsstate+ds);
08790                  rate:=0;
08800                  var:-sucvar;
08810                END;
08820                Resume(firstcont);
08830              END;
08840              last_pc := FALSE;
08850            END
08860            ELSE
08870            IF NOT pc_possible THEN
08880            BEGIN !*** TAKE A RK-STEP.START_RK = FALSE INDICATES ***;
08890              !    *** 2:ND RK-STEP PREPARING FOR PC.           ***;
08900
08910              IF start_rk  AND firsttime THEN
08920              BEGIN
08930                h := 0.5*dtnow;
08940                last_pc := FALSE;
08950              END;
08960
08970              IF firsttime THEN
08980              BEGIN
08990                var :- firstvar;
09000                WHILE var =/= NONE DO INSPECT var DO
09010                BEGIN
09020                  IF start_rk THEN
09030                  BEGIN
09040                    state_1:= state;rate_1:=rate;
09050                  END
09060                  ELSE
09070                  BEGIN
09080                    state1:=state;rate1:=rate;
09090                  END;
09100
09110                  IF NOT ds=0
09120                  THEN epsstate:=(epsstate+ds)-(state-oldstate);
09130
09140                  oldstate:=state;
09150                  a1:=h*rate;
09160                  state:=oldstate+0.5*a1;
09170                  rate:=0;
09180                  var:-sucvar;
09190                END ** WHILE  VAR =/= NONE INSPECT  VAR DO ***;
09200                firsttime := FALSE;
09210              END *** IF FIRSTTIME ***;
09220
09230              dt:=0.5*h; Time:=lasttime+dt;
```

```
09240               Resume(firstcont);
09250
09260               var:-firstvar;
09270               WHILE var=/=NONE DO INSPECT var DO
09280               BEGIN
09290                 a2:=h*rate;
09300                 state:=oldstate+0.25*(a1+a2);
09310                 var:-sucvar;
09320               END;
09330
09340               Resume(firstcont);
09350
09360               var:-firstvar;
09370               WHILE var=/=NONE DO INSPECT var DO
09380               BEGIN
09390                 a3:=h*rate;
09400                 state:=oldstate+(2*a3-a2);
09410                 var:-sucvar;
09420               END;
09430
09440               dt:=h; Time:=lasttime+dt;
09450               Resume(firstcont);
09460
09470               var:-firstvar;
09480               WHILE var=/=NONE DO INSPECT var DO
09490               BEGIN
09500                 a4:=h*rate;
09510                 dsh:=((a1+a4)+4*a3)/6;
09520                 state:=oldstate+dsh;
09530                 var:-sucvar;
09540               END;
09550
09560               Resume(firstcont);
09570
09580               var:-firstvar;
09590               WHILE var=/=NONE DO INSPECT var DO
09600               BEGIN
09610                 IF start_rk THEN
09620                 BEGIN
09630                   state0 := state; rate0 := rate;
09640                 END
09650                 ELSE
09660                 BEGIN
09670                   state2 := state; rate2 := rate;
09680                 END;
09690
09700                 a5:=h*rate;
09710                 ds:=(-dsh+(24*a5-20*a4))+16*(a3-a2);
09720                 a4:=(-a1+4*a3)+(17*a4-23*a5);
09730                 state:=oldstate+(dsh+0.5*a5);
09740                 var:-sucvar;
09750               END;
09760
```

```
09770               dt:=1.5*h; Time:=lasttime+dt;
09780               Resume(firstcont);
09790
09800               var:-firstvar;
09810               WHILE var=/=NONE DO INSPECT var DO
09820               BEGIN
09830                 a2:=h*rate;
09840                 state:=oldstate+(dsh+0.25*(a5+a2));
09850                 var:-sucvar;
09860               END;
09870
09880               Resume(firstcont);
09890
09900               var:-firstvar;
09910               WHILE var=/=NONE DO INSPECT var DO
09920               BEGIN
09930                 a3:=h*rate;
09940                 state:=oldstate+(ds+(a2-2*a3));
09950                 var:-sucvar;
09960               END;
09970
09980               dt:=dtnow; Time:=nexttime;
09990               Resume(firstcont);
10000               errorratio:=2**5/0.5;
10010
10020               var:-firstvar;
10030               WHILE var =/= NONE AND ok DO INSPECT var DO
10040               BEGIN
10050                 a4:=((4*a3-h*rate)+a4)/90;
10060                 temp:=Abs(abserror)+Abs(relerror*(oldstate+dsh));
10070
10080                 IF Abs(a4) > temp THEN
10090                 BEGIN
10100                   IF NOT pc AND dtnow > dtmin OR
10110                   pc AND halvings < max_halvings THEN
10120                   BEGIN ! *** HALVING POSSIBLE ***;
10130
10140                     IF NOT pc AND h < dtmin THEN
10150                     BEGIN ! *** REDUCING STEP TO DTMIN ***;
10160                       frac := dtmin/dtnow; dtnow:=dtnext:=dtmin;
10170                     END
10180                     ELSE
10190                     BEGIN ! *** HALVING OF STEP ***;
10200                       frac := 0.5 ;dtnow:=dtnext:= h;
10210                       halvings := halvings + 1;
10220
```

```
10230                     IF NOT start_rk THEN
10240                     BEGIN
10250                       var :- firstvar;
10260                       WHILE var =/= NONE DO INSPECT var DO
10270                       BEGIN
10280                         state :=((45*state1 + 72*state0 + 11*state_1)
10290                         + h*(-9*rate1 + 36*rate0 + 3*rate_1))/128;
10300                         state_1 := state0;
10310                         rate_1  := rate0;
10320                         var :- sucvar;
10330                       END *** WHILE VAR =/= NONE DO INSPECT VAR DO ***;
10340                       Time := lasttime - 0.5*h;
10350                       Resume(firstcont);
10360
10370                       var :- firstvar;
10380                       WHILE var =/= NONE DO INSPECT var DO
10390                       BEGIN
10400                         state0:= state;
10410                         rate0 := rate;
10420                         var :- sucvar;
10430                       END *** WHILE VAR =/= NONE DO INSPECT VAR DO ***;
10440                     END *** IF NOT START_RK ***;
10450                   END *** HALVING OF STEP ***;
10460
10470                   nexttime := lasttime+(epstime+dtnow);
10480                   IF nexttime > nexteventtime
10490                   THEN nexttime := nexteventtime;
10500
10510                   h:= 0.5*dtnow;
10520
10530                   var :- firstvar;
10540                   WHILE var =/= NONE DO INSPECT var DO
10550                   BEGIN
10560                     a1:=frac*a1;
10570                     state := oldstate + 0.5*a1;
10580                     rate := 0;
10590                     var :- sucvar;
10600                   END *** WHILE VAR =/= NONE DO INSPECT VAR DO ***;
10610
10620                   IF firstvar=/=THIS variable THEN
10630                   BEGIN
10640                     predvar.sucvar:-sucvar;
10650                     IF sucvar=/=NONE THEN sucvar.predvar:-predvar;
10660                     sucvar:-firstvar;
10670                     firstvar:-firstvar.predvar:-
10680                     predvar:-THIS variable;
10690                   END;
10700                 END *** HALVING POSSIBLE ***
10710                 ELSE
```

```
10720                  BEGIN !*** HALVING NOT POSSIBLE ***;
10730                    errorvariable :- THIS variable ;
10740                    integrationerror("RK");
10750                    errorvariable :- NONE;
10760
10770                    dtnext := dtnext*(0.5/a4*
10780                    (Abs(abserror)+Abs(relerror*(oldstate+dsh))))**(1/5);
10790
10800                    IF dtnext > dtmax THEN dtnext := dtmax
10810                    ELSE IF dtnext < dtmin THEN dtnext := dtmin;
10820
10830                    var:-firstvar;
10840                    WHILE var=/=NONE DO INSPECT var DO
10850                    BEGIN
10860                      state:=oldstate;
10870                      rate:=2*(a1/dtnow);
10880                      ds := 0;
10890                      var:-sucvar;
10900                    END;
10910
10920                    dtnow:=(nexteventtime-lasttime)-epstime;
10930
10940                    IF dtnow > dtnext THEN
10950                    BEGIN
10960                      dtnow:=dtnext;
10970                      nexttime := lasttime + (dtnow+epstime);
10980
10990                      IF nexttime > nexteventtime THEN
11000                      nexttime := nexteventtime;
11010                    END
11020                    ELSE nexttime := nexteventtime;
11030
11040                    start_rk := TRUE; firsttime := TRUE;
11050
11060                    IF euler OR adams OR trapez THEN
11070                    BEGIN dt:=0; dtnext:=dtmax; END;
11080
11090                  END *** HALVING NOT POSIBLE ***;
11100                  ok := FALSE;
11110                END *** IF ABS(A4) > TEMP ***
11120                ELSE
11130                BEGIN
11140                  IF temp < errorratio*Abs(a4)
11150                  THEN errorratio := temp/Abs(a4);
11160
11170                  state := oldstate+(dsh+(2*a3-a2));
11180                  var :- sucvar;
11190                END;
11200              END *** WHILE VAR =/= NONE AND OK DO ***;
11210
```

```
11220               IF ok THEN
11230               BEGIN !*** FINISH RK-STEP ***;
11240                 Resume(firstcont);
11250                 var:-firstvar;
11260                 WHILE var=/=NONE DO INSPECT var DO
11270                 BEGIN
11280                   ds:=(a4+dsh)+((a5+h*rate)+4*a3)/6;
11290                   state:=oldstate+(epsstate+ds);
11300                   var:-sucvar;
11310                 END;
11320
11330                 Resume(firstcont);
11340
11350                 IF NOT start_rk THEN
11360                 BEGIN
11370                   var :- firstvar;
11380                   WHILE var =/= NONE DO INSPECT var DO
11390                   BEGIN
11400                     state3:=state; rate3:=rate;
11410                     var :- sucvar;
11420                   END;
11430                 END;
11440
11450                 IF NOT pc THEN
11460                 BEGIN
11470                   IF dtnow=dtnext AND errorratio> 2 THEN
11480                   BEGIN
11490                     dtnext:=(0.5*errorratio)**(1/5)*dtnow;
11500                     IF dtnext>dtmax THEN dtnext:=dtmax;
11510                   END;
11520                 END
11530                 ELSE IF start_rk THEN
11540                 BEGIN
11550                   start_rk := FALSE;
11560                   dtnext := dtnow
11570                 END
11580                 ELSE
11590                 BEGIN
11600                   pc_possible := TRUE;
11610                   normal_pc_steps := 1;
11620                   dtnext := dtmax;
11630                   start_rk := TRUE;
11640                 END;
11650               END *** FINISH RK-STEP ***;
11660               last_pc := FALSE;
11670             END *** TAKE A RK-STEP ***
11680             ELSE
```

```
11690             BEGIN !*** TAKE A PC-STEP;
11700               IF NOT last_pc THEN
11710               BEGIN !*** REFINING OF STARTING VALUES ***;
11720                 var :- firstvar;
11730                 WHILE var =/= NONE DO INSPECT var DO
11740                 BEGIN
11750                   ds := 0;
11760                   epsstate := 0;
11770                   p3_c3:=0;
11780                   state:=state1:=state0+
11790                   h/24*(9*rate0+19*rate1-5*rate2+rate3);
11800                   var:-sucvar;
11810                 END *** WHILE VAR =/= NONE  DO ***;
11820
11830                 Time := lasttime - 2*h;
11840                 Resume(firstcont);
11850
11860                 var:-firstvar;
11870                 WHILE var=/=NONE DO INSPECT var DO
11880                 BEGIN
11890                   rate1:= rate;
11900                   state:=state2:=state0+
11910                   h/3*(rate0+4*rate1+rate2);
11920                   var:-sucvar;
11930                 END *** WHILE VAR =/= NONE DO ***;
11940
11950                 Time := lasttime - h;
11960                 Resume(firstcont);
11970
11980                 var:-firstvar;
11990                 WHILE var=/=NONE DO INSPECT var DO
12000                 BEGIN
12010                   rate2:= rate;
12020                   state:=state3:=state0+
12030                   h*3/8*(rate0+3*rate1+3*rate2+rate3);
12040                   var:-sucvar;
12050                 END *** WHILE VAR =/= NONE DO ***;
12060
12070                 Time := lasttime;
12080                 Resume(firstcont);
12090
12100                 var:-firstvar;
12110                 WHILE var=/=NONE DO INSPECT var DO
12120                 BEGIN
12130                   rate3:=rate;
12140                   var:-sucvar;
12150                 END *** WHILE VAR =/= NONE DO ***;
12160               END *** REFINING OF STARTING VALUES ***;
12170
```

```
12180               var :- firstvar ;
12190               WHILE var =/= NONE DO INSPECT var DO
12200               BEGIN
12210                 p4 := state0 + (2*rate3 - rate2 + 2*rate1)*4*h/3 ;
12220                 state := p4 -(p3_c3)*112/121;
12230                 var :- sucvar;
12240               END *** WHILE VAR =/= NONE INSPECT VAR ***;
12250
12260               Time:=nexttime; dt:=dtnow;
12270
12280               Resume(firstcont);
12290               errorratio := 64;
12300               var :- firstvar;
12310               WHILE var =/= NONE AND ok DO INSPECT var DO
12320               BEGIN
12330                 c4 := (9*state3-state1+3*h*(rate+2*rate3-rate2))/8;
12340                 a4 := p4 - c4;
12350                 state := c4 + 9*a4/121;
12360                 temp:= Abs(abserror)+ Abs(relerror*state);
12370                 IF Abs(a4) > temp THEN
12380                 BEGIN
12390                   IF firstvar =/= THIS variable THEN
12400                   BEGIN
12410                     predvar.sucvar :- sucvar;
12420                     IF sucvar =/= NONE THEN sucvar.predvar :- predvar;
12430                     sucvar :- firstvar;
12440                     firstvar:-firstvar.predvar:-predvar:-THIS variable;
12450                   END *** IF FIRSTVAR =/= THIS VARIABLE ***;
12460
12470                   halvings := halvings + 1;
12480
12490                   IF halvings <= max_halvings THEN
12500                   BEGIN
12510                     dtnow := 0.5*dtnow;
12520
12530                     var :- firstvar ;
12540                     WHILE var =/= NONE DO INSPECT var DO
12550                     BEGIN
12560                       state_1 := state1;
12570                       rate_1 := rate1;
12580                       state1 := state2;
12590                       rate1 := rate2;
12600                       state := ((12*state3+135*state1+108*state_1+state0)
12610                       +h*(-3*rate3-54*rate1+27*rate_1))/256;
12620                       var :- sucvar;
12630                     END *** WHILE VAR =/= NONE INSPECT VAR ***;
12640
12650                     Time := lasttime -3*dtnow;
12660                     Resume(firstcont);
12670
```

```
12680                   var :- firstvar ;
12690                   WHILE var =/= NONE DO INSPECT var DO
12700                   BEGIN  REAL tempstate;
12710                     tempstate := state0;
12720                     state0 := state;
12730                     rate0 := rate;
12740                     state :=
12750                     ((80*state3+135*state1+40*state_1+tempstate)
12760                     +h*(-15*rate3+90*rate1+15*rate_1))/256;
12770                     var :- sucvar;
12780                   END *** WHILE VAR =/= NONE INSPECT VAR ***;
12790
12800                   Time := lasttime - dtnow;
12810                   Resume(firstcont);
12820                   h:=dtnow;
12830
12840                   var :- firstvar ;
12850                   WHILE var =/= NONE DO INSPECT var DO
12860                   BEGIN
12870                     state2 := state;
12880                     rate2 := rate;
12890                     p3_c3:=242/27*(state3-state0) -
12900                     121/36*h*(rate3+3*rate2+3*rate1+rate0);
12910                     var :- sucvar;
12920                   END *** WHILE VAR =/= NONE INSPECT VAR ***;
12930
12940                   nexttime:=lasttime+(epstime+dtnow);
12950                   IF nexttime>nexteventtime THEN
12960                   nexttime:=nexteventtime;
12970
12980                   normal_pc_steps := 1;
12990                   ok := FALSE;
13000                   disc_imp := TRUE;
13010                 END *** HALVINGS <= MAXHALVINGS ***
13020                 ELSE
13030                 BEGIN
13040                   errorvariable :- THIS variable;
13050                   integrationerror("PC");
13060                   errorvariable :- NONE;
13070                   ok := FALSE;
13080                   IF NOT pc THEN
13090                   BEGIN
13100                     firsttime := TRUE;
13110                     dt := 0;
13120                     pc_possible := FALSE;
13130                     var :- firstvar;
13140                     WHILE var =/= NONE DO INSPECT var DO
13150                     BEGIN
13160                       state:=state3;rate:=rate3;
13170                       ds := 0;
13180                       var:-sucvar;
13190                     END;
13200                   END *** NOT PC ***;
13210                 END *** HALVINGS > MAXHALVINGS ***;
```

```
13220                 END *** IF ABS(A4) > TEMP
13230                 ELSE IF temp < Abs(a4)*errorratio
13240                 THEN errorratio := temp / Abs(a4);
13250
13260                 var :- sucvar;
13270               END *** WHILE VAR =/= NONE AND OK DO INSPECT VAR DO ***;
13280
13290               IF ok THEN
13300               BEGIN
13310                 Resume(firstcont);
13320
13330                 IF errorratio > 50 AND halvings >= disc_halvings
13340                 AND NOT disc_imp THEN
13350                 BEGIN ! *** PREPARE FOR DISCONTINUITY ***;
13360                   pc_possible := FALSE;
13370                   start_rk := TRUE;
13380                   halvings := 0;
13390                 END *** PREPARE FOR DISCONTINUITY ***
13400
13410                 ELSE IF errorratio > 50 AND normal_pc_steps >= 2
13420                 AND 2*h <= dtmax THEN
13430                 BEGIN ! *** UPDATE FOR DOUBLING ***;
13440                   var :- firstvar;
13450                   WHILE var =/= NONE DO INSPECT var DO
13460                   BEGIN
13470                     state1 := state0;
13480                     rate1  := rate0;
13490                     state0 := state_2;
13500                     rate0  := rate_2;
13510                     state3 := state;
13520                     rate3 := rate;
13530                     oldstate:= state2;
13540                     p3_c3:=242/27*(state3-state0)-
13550                     121/18*h*(rate3+3*rate2+3*rate1+rate0);
13560                     var :- sucvar;
13570                   END *** WHILE VAR =/= NONE INSPECT VAR DO ***;
13580
13590                   normal_pc_steps := 0;
13600                   lasttime := lasttime - h;
13610                   dtnow := h := 2*h;
13620                   halvings := halvings -1;
13630
13640                 END *** UPDATE FOR DOUBLING
13650                 ELSE
```

```
13660                BEGIN ! *** UPDATE NORMAL ***;
13670                  var :- firstvar;
13680                  WHILE var =/= NONE DO INSPECT var DO
13690                  BEGIN
13700                    state_2 := state_1;
13710                    rate_2  := rate_1;
13720                    state_1 := state0;
13730                    rate_1  := rate0;
13740                    state0  := state1;
13750                    rate0   := rate1;
13760                    state1  := state2;
13770                    rate1   := rate2;
13780                    oldstate:= state2  := state3;
13790                    rate2   := rate3;
13800                    state3  := state;
13810                    rate3   := rate;
13820                    p3_c3 := a4;
13830                    var :- sucvar;
13840                  END *** WHILE VAR =/= NONE INSPECT VAR DO ***;
13850                  normal_pc_steps:=normal_pc_steps+1;
13860                  disc_imp := FALSE;
13870                END *** UPDATE NORMAL ***;
13880
13890                last_pc := TRUE;
13900
13910              END *** IF OK ***;
13920            END *** PC-STEP ***;
13930          END *** WHILE NOT OK DO ***;
13940        END *** IF FIRSTCONT =/= NONE ***
13950        ELSE
13960        BEGIN !*** IF FIRSTCONT == NONE ***;
13970          IF dtnow>dtmax AND firstwait=/=NONE THEN
13980          BEGIN
13990            dtnow:=dtmax;
14000            nexttime:=lasttime+(epstime+dtnow);
14010            IF nexttime>nexteventtime THEN nexttime:=nexteventtime;
14020          END
14030          ELSE nexttime:=nexteventtime;
14040
14050          dt:=dtnow; Time:=nexttime;
14060        END *** IF FIRSTCONT == NONE ***;
14070
```

```
14080         !STATEEVENT ;
14090         nextstateevent:-NONE;
14100
14110         IF firstwait=/=NONE AND (dt>dtmin OR Time<nexteventtime)
14120         THEN Resume(firstwait.proc);
14130
14140         IF stateevent=/=NONE THEN
14150         BEGIN
14160           pc_possible:=FALSE;
14170
14180           IF dtnow<=dtmin
14190           THEN nexteventtime:=Time
14200           ELSE
14210           BEGIN
14220             nextstateevent:-stateevent; stateevent:-NONE;
14230             dt:=0.5*dtnow; dtlower:=0;
14240             Time:=lasttime+(epstime+dt);
14250
14260             IF firstcont=/=NONE THEN
14270             BEGIN
14280               dtfull:=dtnow;
14290               frac:=dt/dtfull;
14300
14310               var:-firstvar;
14320               IF euler THEN
14330               BEGIN
14340                 WHILE var=/=NONE DO INSPECT var DO
14350                 BEGIN
14360                   a5:=a4:=a3:=a2:=0; a1:=ds;
14370                   ds:=a1*frac;
14380                   state:=oldstate+(epsstate+ds);
14390                   var:-sucvar;
14400                 END;
14410               END
14420               ELSE
14430               WHILE var=/=NONE DO INSPECT var DO
14440               BEGIN
14450                 IF last_pc THEN
14460                 BEGIN
14470                   a5:=(-3*state3+3*state1+h*(rate3+4*rate2+rate1))/4;
14480                   a4:=(-2*state3+4*state2-2*state1+h*(rate3-rate1))/4;
14490                   a3:=(+5*state3-5*state1+h*(-rate3-8*rate2-rate1))/4;
14500                   a2:=(+4*state3-8*state2+4*state1+h*(-rate3+rate1))/4;
14510                   a1:=h*rate2;
14520                 END
14530                 ELSE
14540                 BEGIN
14550                   temp:=h*rate; dsh:=dsh+0.5*a4;
14560                   a4:=4*((4*dsh+13*ds)-(4*temp+6*a1+20*a5));
14570                   a3:=2*((5*temp+13*a1+32*a5)-(16*dsh+17*ds));
14580                   a2:=(7*ds+16*dsh)-(2*temp+12*a1+16*a5);
14590                   a5:=8*((a1+temp+4*a5)-3*ds);
14600                   a1:=2*a1;
14610                 END;
```

```
14620                 ds:=((((a5*frac+a4)*frac+a3)*frac+a2)*frac+a1)*frac;
14630                 state:=oldstate+(epsstate+ds);
14640                 var:-sucvar;
14650               END *** WHILE VAR =/= NONE DO ***;
14660
14670               Resume(firstcont);
14680
14690             END *** IF FIRSTCONT =/= NONE ***;
14700
14710             WHILE stateevent==NONE DO
14720             BEGIN
14730               Resume(firstwait.proc);
14740
14750               IF stateevent=/=NONE THEN
14760               BEGIN
14770                 nextstateevent:-stateevent; stateevent:-NONE;
14780                 nexttime:=Time;
14790                 dtnow:=temp:=dt;
14800               END
14810               ELSE dtlower:=temp:=dt;
14820
14830               dt:=0.5*dtlower+0.5*dtnow;
14840
14850               IF dtnow-dtlower<=dtmin OR dt<=dtlower OR dt>=dtnow THEN
14860               BEGIN
14870                 stateevent:-nextstateevent;
14880                 Time:=nexttime;
14890                 dt:=dtnow;
14900               END
14910               ELSE Time:=lasttime+(epstime+dt);
14920
14930               IF firstcont=/=NONE AND NOT dt=temp THEN
14940               BEGIN
14950                 frac:=dt/dtfull;
14960
14970                 var:-firstvar;
14980                 WHILE var=/=NONE DO INSPECT var DO
14990                 BEGIN
15000                   ds:=((((a5*frac+a4)*frac+a3)*frac+a2)*frac+a1)*frac;
15010                   state:=oldstate+(epsstate+ds);
15020                   var:-sucvar;
15030                 END;
15040
15050                 Resume(firstcont);
15060               END;
15070             END *** WHILE STATEEVENT==NONE ***;
15080
```

```
15090             IF Time<nexteventtime THEN
15100             BEGIN
15110               IF dtnow=dtfull AND firstcont=/=NONE THEN
15120               BEGIN
15130                 stateevent:-NONE;
15140                 Resume(firstwait.proc);
15150                 IF stateevent=/=NONE THEN nexteventtime:=Time;
15160               END
15170               ELSE nexteventtime:=Time;
15180             END
15190             ELSE
15200             BEGIN
15210               stateevent:-NONE; Time:=nexttime:=nexteventtime;
15220             END;
15230
15240           END *** DTNOW>DTMIN ***;
15250
15260           IF nextsampletime>nexteventtime
15270           THEN nextsampletime:=nexteventtime;
15280
15290         END *** STATEEVENT=/=NONE ***;
15300
```

```
15310         ! *** SAMPLE ***;
15320         IF firstpossample=/=NONE AND nextsampletime<=Time THEN
15330         BEGIN
15340           IF nextsampletime=Time THEN
15350           BEGIN
15360             nextsampletime:=nexteventtime;
15370             Resume(firstpossample);
15380           END
15390           ELSE
15400           BEGIN
15410             IF firstcont=/=NONE AND nextstateevent==NONE THEN
15420             BEGIN
15430               Time:=nextsampletime;
15440               dtfull:=dtnow;
15450               dt:=(Time-lasttime)-epstime;
15460               frac:=dt/dtfull;
15470
15480               var:-firstvar;
15490               IF euler OR adams OR trapez THEN
15500               BEGIN
15510                 WHILE var=/=NONE DO INSPECT var DO
15520                 BEGIN
15530                   a5:=a4:=0; a3:=(a1+dtnow*rate)-2*ds; a2:=ds-(a1+a3);
15540                   state:=oldstate+(epsstate+
15550                   ((a3*frac+a2)*frac+a1)*frac);
15560                   var:-sucvar;
15570                 END;
15580               END
15590               ELSE
15600               WHILE var=/=NONE DO INSPECT var DO
15610               BEGIN
15620                 IF last_pc THEN
15630                 BEGIN
15640                   a5:=(-3*state3+3*state1+h*(rate3+4*rate2+rate1))/4;
15650                   a4:=(-2*state3+4*state2-2*state1+h*(rate3-rate1))/4;
15660                   a3:=(+5*state3-5*state1+h*(-rate3-8*rate2-rate1))/4;
15670                   a2:=(+4*state3-8*state2+4*state1+h*(-rate3+rate1))/4;
15680                   a1:=h*rate2;
15690                 END
15700                 ELSE
15710                 BEGIN
15720                   temp:=h*rate; dsh:=dsh+0.5*a4;
15730                   a4:=4*((4*dsh+13*ds)-(4*temp+6*a1+20*a5));
15740                   a3:=2*((5*temp+13*a1+32*a5)-(16*dsh+17*ds));
15750                   a2:=(7*ds+16*dsh)-(2*temp+12*a1+16*a5);
15760                   a5:=8*((a1+temp+4*a5)-3*ds);
15770                   a1:=2*a1;
15780                 END;
15790                 state:=oldstate+(epsstate+
15800                 ((((a5*frac+a4)*frac+a3)*frac+a2)*frac+a1)*frac);
15810                 var:-sucvar;
15820               END;
15830
```

```
15840               Resume(firstcont);
15850               nextsampletime:=nexteventtime;
15860               Resume(firstpossample);
15870             END;
15880
15890             WHILE nextsampletime<nexttime DO
15900             BEGIN
15910               Time:=nextsampletime;
15920               dt:=(Time-lasttime)-epstime;
15930
15940               IF firstcont=/=NONE THEN
15950               BEGIN
15960                 frac:=dt/dtfull;
15970                 var:-firstvar;
15980                 WHILE var=/=NONE DO INSPECT var DO
15990                 BEGIN
16000                   state:=oldstate+(epsstate+
16010                   ((((a5*frac+a4)*frac+a3)*frac+a2)*frac+a1)*frac);
16020                   var:-sucvar;
16030                 END;
16040                 Resume(firstcont);
16050               END;
16060
16070               nextsampletime:=nexteventtime;
16080               Resume(firstpossample);
16090             END;
16100
16110             dt:=dtnow; Time:=nexttime;
16120             IF firstcont=/=NONE THEN
16130             BEGIN
16140               var:-firstvar;
16150               WHILE var=/=NONE DO INSPECT var DO
16160               BEGIN
16170                 state:=oldstate+(epsstate+ds);
16180                 var:-sucvar;
16190               END;
16200               Resume(firstcont);
16210             END;
16220
16230             IF Time=nextsampletime THEN
16240             BEGIN
16250               nextsampletime:=nexteventtime;
16260               Resume(firstpossample);
16270             END;
16280           END;
16290         END *** FIRSTPOSSAMPLE=/=NONE ***;
16300
16310         IF firstzerosample=/=NONE THEN Resume(firstzerosample);
16320
16330         temp:=epstime+dtnow;
16340         IF temp<=epstime AND Time<nexteventtime THEN
16350         error
16360         (2,"2: THE CURRENT TIME STEP IS TOO SMALL TO ADVANCE TIME");
16370
16380         epstime:=temp-(Time-lasttime);
16390       END *** WHILE TIME < NEXTEVENTTIME ***;
16400
```

```
16410       pc_possible := FALSE;
16420
16430       IF firstnegsample=/=NONE AND dt>0 THEN Resume(firstnegsample);
16440
16450       IF nexttimeevent==NONE AND stateevent==NONE THEN
16460       BEGIN
16470         IF firstwait=/=NONE THEN Resume(firstwait.proc);
16480         IF stateevent==NONE THEN
16490         error
16500         (7,"7: TIME IS AT ITS MAXIMUM VALUE AND NO EVENTS OCCUR");
16510       END;
16520
16530       IF stateevent=/=NONE THEN
16540       BEGIN
16550         REACTIVATE stateevent AT Time;
16560         nexttimeevent:-stateevent; stateevent:-NONE;
16570       END;
16580
16590       REACTIVATE controller2 AFTER nexttimeevent;
16600
16610       IF controller1.Nextev=/=nexttimeevent THEN
16620       BEGIN
16630         IF nexttimeevent.Idle THEN error(20,"20: ILLEGAL USE OF"
16640         " CANCEL");
16650         error(17,"17: ILLEGAL USE OF (RE)ACTIVATE");
16660       END
16670       ELSE
16680       IF NOT Time=nexttimeevent.Evtime
16690       THEN error(17,"17: ILLEGAL USE OF (RE)ACTIVATE");
16700
16710       active:=FALSE;
16720
16730       REACTIVATE controller1 BEFORE controller2;
16740
16750       nexttimeevent:-controller2.Nextev;
16760     END *** WHILE NEXTTIMEEVENT=/=NONE OR FIRSTWAIT=/=NONE ***;
16770
16780     error(3,"3: THERE ARE NO DISCRETE EVENTS SCHEDULED");
16790   END *** CLASS MONITOR ***;
16800
16810
16820   REF(monitor) themonitor; themonitor:-NEW monitor;
16830   disc_halvings := 10;
16840   max_halvings := 30;
16850   prologue;
16860   INNER;
16870   files.Close;
16880   END *** CLASS CONDIS ***;
```

9.3 FILEQU

```
00100   OPTIONS(/E);
00200   EXTERNAL REF(Infile) PROCEDURE findinfile;
00300   EXTERNAL REF(Outfile) PROCEDURE findoutfile;
00400   EXTERNAL TEXT PROCEDURE conc,upcase,frontstrip,
00500   rest,checkextension;
00600   EXTERNAL CHARACTER PROCEDURE fetchar,findtrigger;
00700   EXTERNAL LONG REAL PROCEDURE scanreal;
00800   EXTERNAL INTEGER PROCEDURE checkreal,checkint,scanint,ilog;
00900   EXTERNAL BOOLEAN PROCEDURE menu,isopen;
01000   EXTERNAL CLASS simeio;
01100
01200   simeio  CLASS filequ;
01300   BEGIN
01400   Head CLASS file_list;
01500   BEGIN
01600     PROCEDURE Close;
01700     BEGIN
01800       WHILE NOT Empty DO
01900       BEGIN
02100         INSPECT First
02200         WHEN inf DO IF isopen(the_file) THEN the_file.Close
02301         WHEN outf DO IF isopen(the_file) THEN the_file.Close
02400         WHEN directf DO IF isopen(the_file) THEN the_file.Close
02500         WHEN printf DO IF isopen(the_file) THEN the_file.Close
02600         OTHERWISE Outtext("FILEQU: ELEMENT OF"
02700         " ILLEGAL QULIFICATION IN FILELIST.");
02800         First.Out;
02900       END *** WHILE FILE =/= NONE DO ***;
03000     END *** PROCEDURE CLOSE ***;
03100   END *** CLASS FILE_LIST ***;
03200
03300   Link CLASS inf(the_file); REF(Infile) the_file;;
03400   Link CLASS outf(the_file); REF(Outfile) the_file;;
03500   Link CLASS printf(the_file); REF(Printfile) the_file;;
03600   Link CLASS directf(the_file); REF(Directfile) the_file;;
03700
03800   PROCEDURE opendirectfile(filename,postsize);
03900   REF(Directfile) filename;INTEGER postsize;
04000   BEGIN
04100     NEW directf(filename).Into(files);
04150     filename.Open(Blanks(postsize));
04200   END *** OPENDIRECTFILE ***;
04300
04400   PROCEDURE openinfile(filename,postsize);
04500   REF(Infile) filename;INTEGER postsize;
04600   BEGIN
04700     NEW inf(filename).Into(files);
04750     filename.Open(Blanks(postsize));
04800   END *** OPENINFILE ***;
04900
```

```
05000   PROCEDURE openoutfile(filename,postsize);
05100   REF(Outfile) filename;INTEGER postsize;
05200   BEGIN
05300     NEW outf(filename).Into(files);
05350     filename.Open(Blanks(postsize));
05400   END *** OPENOUTFILE ***;
05500
05600   PROCEDURE openprintfile(filename,postsize);
05700   REF(Printfile) filename;INTEGER postsize;
05800   BEGIN
05900     NEW printf(filename).Into(files);
05950     filename.Open(Blanks(postsize));
06000   END *** OPENPRINTFILE ***;
06100
06200   REF(file_list) files;
06300
06400   files :- NEW file_list;
06500
06600   END *** CLASS FILEQU ***;
```

JÄMFÖRELSE AV CPU-TID MELLAN COMBINEDSIMULATION OCH CADSIM

För att testa de båda simuleringspaketen COMBINEDSIMULATION och CADSIM simulerades en situation där 10 projektiler sköts rakt upp i luften med utgångshastigheten 1000 m/s. Tiden mellan projektilerna samt deras vikt slumpades. Luftmotståndet beräknades via tabeller över densitet som funktion av höjden, machtal som funktion av höjden samt luftmotståndskoefficient som funktion av machtal.

Simuleringen av resp projektil avbröts då villkoret v < 0 uppfylldes, dvs då projektilen vände. För varje projektil registrerades tiden för avfyring, massa, tiden då projektilen vände, maximala höjden, antal beräkningar av funktionsuttrycken i diffrentialekvationerna samt antalet utförda tidsteg under den den tid projektilen simulerats.

Två program skrevs, ett med vardera simuleringspaketet. Upplägningen av programmen var lika i båda fallen. Båda fanns som laddningsmoduler då simuleringen startade för att inte få med kompilering och länkning i åtgången tid.

Båda programmen testades med olika värden på maximalt tillåten integrationssteglängd.

Resultaten tyder på att integrationsnoggranheten är ganska lika, att COMBINEDSIMULATION kräver fler beräkningar av funktionsuttrycken, men att åtgången av CPU-tid är likvärdig under förutsättning att maximala integrationssteglängden inte valts för liten. I det sista fallet är CADSIM bättre.

Noteras bör också hur kraftigt exekveringstiden beror på valet av dtmax.

****

CADSIM Maxsteg 0.2

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.297	9.9230E+03	1066	200
8.731	1.193	48.109	1.0373E+04	1086	204
9.676	0.897	44.297	8.3168E+03	956	180
10.667	1.095	48.522	9.6965E+03	1041	198
10.809	0.985	46.897	8.9328E+03	986	187
15.438	1.096	53.322	9.7094E+03	1036	196
16.044	0.930	51.234	8.5535E+03	956	179
18.140	1.108	56.203	9.7883E+03	1031	195
18.648	0.852	52.494	7.9981E+03	911	172
23.731	0.939	59.069	8.6155E+03	951	179

68 garb.coll in 4220ms. CPU: 22.04. Elapsed 1:18.30.

****

BILAGA 1 JÄMFÖRELSE AV CPU-TID ...

****

COMSIM Maxsteg 0.2

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.299	9.9230E+03	1791	197
8.731	1.193	48.113	1.0373E+04	1873	203
9.676	0.897	44.299	8.3168E+03	1627	177
10.667	1.095	48.524	9.6965E+03	1817	196
10.809	0.985	46.900	8.9328E+03	1698	185
15.438	1.096	53.324	9.7094E+03	1852	197
16.044	0.930	51.237	8.5534E+03	1689	181
18.140	1.108	56.206	9.7884E+03	1859	197
18.648	0.852	52.497	7.9982E+03	1633	174
23.731	0.939	59.072	8.6155E+03	1741	183

95 garb.coll in 6329 ms.CPU :29.81. Elapsed 2:46.32.

****

CADSIM Maxsteg 0.4

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.297	9.9230E+03	596	105
8.731	1.193	48.109	1.0373E+04	601	108
9.676	0.897	44.297	8.3169E+03	531	94
10.667	1.095	48.522	9.6964E+03	571	104
10.809	0.985	46.897	8.9325E+03	536	97
15.438	1.096	53.322	9.7094E+03	566	103
16.044	0.930	51.234	8.5533E+03	521	93
18.140	1.108	56.203	9.7885E+03	561	100
18.648	0.852	52.494	7.9983E+03	491	88
23.731	0.939	59.069	8.6155E+03	516	91

41 garb.coll in 2249 ms.CPU :12.30. Elapsed 43.06.

****

COMSIM Maxsteg 0.4

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.299	9.9230E+03	937	102
8.731	1.193	48.113	1.0373E+04	1004	106
9.676	0.897	44.299	8.3168E+03	855	91
10.667	1.095	48.524	9.6965E+03	976	102
10.809	0.985	46.900	8.9328E+03	891	95
15.438	1.096	53.324	9.7094E+03	1016	103
16.044	0.930	51.237	8.5535E+03	912	94
18.140	1.108	56.206	9.7883E+03	1024	103
18.648	0.852	52.496	7.9981E+03	885	90
23.731	0.939	59.072	8.6155E+03	970	96

54 garb.coll in 3409 ms.CPU :16.22. Elapsed 2:04.42.

****

BILAGA 1 JÄMFÖRELSE AV CPU-TID ...

****

CADSIM Maxsteglängd 0.8

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.297	9.9235E+03	356	57
8.731	1.193	48.113	1.0374E+04	366	60
9.676	0.897	44.300	8.3173E+03	316	51
10.667	1.095	48.522	9.6959E+03	331	57
10.809	0.985	46.897	8.9327E+03	316	53
15.438	1.096	53.325	9.7101E+03	326	56
16.044	0.930	51.231	8.5530E+03	301	48
18.140	1.108	56.203	9.7882E+03	321	53
18.648	0.852	52.494	7.9980E+03	286	48
23.731	0.939	59.069	8.6154E+03	301	48

27 garb.coll in 1381 ms. CPU : 7.39. Elapsed 32.58.

****

COMSIM Maxsteglängd 0.8

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.299	9.9229E+03	498	53
8.731	1.193	48.113	1.0373E+04	558	56
9.676	0.897	44.299	8.3168E+03	463	47
10.667	1.095	48.524	9.6964E+03	540	53
10.809	0.985	46.899	8.9325E+03	481	49
15.438	1.096	53.324	9.7093E+03	582	54
16.044	0.930	51.237	8.5534E+03	513	49
18.140	1.108	56.206	9.7885E+03	592	54
18.648	0.852	52.497	7.9984E+03	504	47
23.731	0.939	59.072	8.6156E+03	566	50

33 garb.coll in 1207 ms. CPU : 9.24. Elapsed 31.16.

****

CADSIM Maxsteglängd 1.6

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.297	9.9235E+03	251	36
8.731	1.193	48.113	1.0375E+04	256	36
9.676	0.897	44.297	8.3171E+03	276	38
10.667	1.095	48.525	9.6979E+03	276	40
10.809	0.985	46.894	8.9320E+03	241	33
15.438	1.096	53.322	9.7089E+03	246	35
16.044	0.930	51.231	8.5527E+03	231	30
18.140	1.108	56.203	9.7875E+03	241	34
18.648	0.852	52.494	7.9979E+03	231	35
23.731	0.939	59.072	8.6156E+03	241	35

22 garb.coll in 1074 ms. CPU : 5.87. Elapsed 2:08.00.

****

BILAGA 1 JÄMFÖRELSE AV CPU-TID ...

****

COMSIM Maxsteglängd 1.6

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.299	9.9232E+03	282	29
8.731	1.193	48.114	1.0374E+04	337	31
9.676	0.897	44.299	8.3174E+03	275	26
10.667	1.095	48.524	9.6959E+03	339	30
10.809	0.985	46.901	8.9332E+03	286	27
15.438	1.096	53.326	9.7101E+03	375	30
16.044	0.930	51.235	8.5530E+03	321	27
18.140	1.108	56.206	9.7883E+03	377	29
18.648	0.852	52.496	7.9980E+03	314	25
23.731	0.939	59.070	8.6147E+03	361	26

CPU : 5.88. Elapsed 1:12.74.

****

CADSIM Maxsteglängd 3.2

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.297	9.9230E+03	226	29
8.731	1.193	48.113	1.0374E+04	216	26
9.676	0.897	44.297	8.3171E+03	261	33
10.667	1.095	48.538	9.7045E+03	241	28
10.809	0.985	46.894	8.9320E+03	231	28
15.438	1.096	53.325	9.7105E+03	226	29
16.044	0.930	51.241	8.5558E+03	196	24
18.140	1.108	56.203	9.7875E+03	231	29
18.648	0.852	52.494	7.9979E+03	226	31
23.731	0.939	59.072	8.6156E+03	226	30

21 garb.coll in 1051 ms. CPU : 5.49. Elapsed 55.42.

****

COMSIM Maxsteglängd 3.2

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.299	9.9228E+03	277	26
8.731	1.193	48.111	1.0373E+04	318	27
9.676	0.897	44.299	8.3179E+03	263	23
10.667	1.095	48.524	9.6964E+03	320	26
10.809	0.985	46.900	8.9329E+03	265	23
15.438	1.096	53.324	9.7093E+03	350	25
16.044	0.930	51.235	8.5530E+03	295	22
18.140	1.108	56.203	9.7874E+03	337	23
18.648	0.852	52.496	7.9978E+03	281	20
23.731	0.939	59.071	8.6150E+03	313	19

22 garb.coll in 1176 ms. CPU : 5.68. Elapsed 1:24.32.

****

BILAGA 1 JÄMFÖRELSE AV CPU-TID ...

****

CADSIM Maxsteglängd 6.4

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.297	9.9228E+03	266	27
8.731	1.193	48.106	1.0373E+04	266	25
9.676	0.897	44.297	8.3168E+03	276	29
10.667	1.095	48.525	9.6966E+03	296	31
10.809	0.985	46.903	8.9345E+03	251	26
15.438	1.096	53.325	9.7109E+03	241	27
16.044	0.930	51.238	8.5547E+03	251	25
18.140	1.108	56.206	9.7897E+03	256	25
18.648	0.852	52.494	7.9982E+03	286	31
23.731	0.939	59.072	8.6156E+03	246	31

24 garb.coll in 1192 ms. CPU :6.26. Elapsed 1:06.06

****

COMSIM Maxsteg 6.4

Skott_tid	massa	maxhöjdstid	maxhöjd	derivationer	tidssteg
5.928	1.127	44.299	9.9228E+03	268	25
8.731	1.193	48.111	1.0373E+04	339	27
9.676	0.897	44.299	8.3179E+03	262	22
10.667	1.095	48.525	9.6964E+03	342	26
10.809	0.985	46.901	8.9329E+03	287	23
15.438	1.096	53.323	9.7093E+03	395	26
16.044	0.930	51.236	8.5530E+03	340	23
18.140	1.108	56.203	9.7874E+03	382	24
18.648	0.852	52.495	7.9978E+03	327	21
23.731	0.939	59.070	8.6150E+03	364	20

23 garb.coll in 1227 ms. CPU : 6.01. Elapsed 1:03.42.

****

JÄMFÖRELSE AV NOGGRANNHET MELLAN RUNGE-KUTTA OCH PREDIKTOR-KORREKTORMETOD VID DISKONTINUITET

För att jämföra hur Runge-Kutta-England resp prediktor-korrektor enl DHAMDI uppför sig då den integrerade funktionen innehåller diskontinuiteter utfördes tester på följande problem:

$$\text{Beräkna } y(t) = \int_0^t f(s)\,ds \;,\; y(0) = 0 \text{ för } 0 <= t <= 4$$

$$\text{då } f(t) = \begin{cases} 0 & ; \; t < 1 \;,\; t >= 3 \\ 1 & ; \; 1 <= t < 2 \\ -1 & ; \; 2 <= t < 3 \end{cases}$$

Alla lösningarna är utförda med CONDIS. Vissa integrations-parametrar varierades enl nedanstående tabell. Maxrelerror, maxabserror och dtmax hade samma värde i alla fyra körningarna, 0, 1E-5 resp 0.2.

Lösn	PC	Dtmin	Spion.frq
1	FALSE	1E-9	0
2	FALSE	1E-9	0.2
3	TRUE	0.01	0
4	TRUE	0.01	0.2

Antalet derivationer kan jämföras med 347 som anges i [5] där samma problem lösts med integrationsmetoden DHAMDI.

Noteras bör också att Runge-Kutta-England klarar av diskontinuiteterna, bara dtmin är tillräckligt litet, men att onoggranheten i beräkningarna då blir upp till en storleksording högre än det begärda lokala felet. Med prediktor-korrektormetoden enligt DHAMDI blir däremot även det totala felet ungefär lika med det lokala. Detta tyder på att det inte är lämpligt att använda RKE-metoden och bara förse den med "diskontinuitetsdetektering" av samma typ som i DHAMDI, vilket ju annars vore en mycket lättare ändring av COMBINEDSIMULATION än införandet av PC-metoden.

BILAGA 2 JÄMFÖRELSE AV NOGGRANNHET ...

Lösning 1.

t	y_{ber}	y_{korr}	$(y_{ber} - y_{korr})*10^5$
0.00000000	0.00000000	0.00000000	0.000
0.20000000	0.00000000	0.00000000	0.000
0.40000000	0.00000000	0.00000000	0.000
0.60000000	0.00000000	0.00000000	0.000
0.80000000	0.00000000	0.00000000	0.000
0.90000000	0.00000000	0.00000000	0.000
0.95000000	0.00000000	0.00000000	0.000
0.97499999	0.00000000	0.00000000	0.000
0.98750000	0.00000000	0.00000000	0.000
0.99375000	0.00000000	0.00000000	0.000
0.99687500	0.00000000	0.00000000	0.000
0.99843749	0.00000000	0.00000000	0.000
1.00000000	0.00012153	0.00000000	12.153
1.00156249	0.00168403	0.00156249	12.154
1.00468750	0.00480903	0.00468750	12.152
1.01093750	0.01105903	0.01093750	12.153
1.02343750	0.02355903	0.02343750	12.153
1.04843749	0.04855903	0.04843749	12.154
1.09843750	0.09855903	0.09843750	12.153
1.19843750	0.19855903	0.19843750	12.153
1.39843750	0.39855903	0.39843750	12.153
1.59843750	0.59855903	0.59843750	12.153
1.79843749	0.79855903	0.79843749	12.154
1.99843749	0.99855904	0.99843749	12.154
1.99921875	0.99934028	0.99921875	12.153
2.00000000	1.00000000	1.00000000	0.000
2.00078124	0.99921875	0.99921876	-0.001
2.00234374	0.99765626	0.99765626	0.000
2.00546876	0.99453126	0.99453124	0.001
2.01171875	0.98828126	0.98828125	0.001
2.02421874	0.97578125	0.97578126	-0.001
2.04921874	0.95078126	0.95078126	0.000
2.09921876	0.90078125	0.90078124	0.001
2.19921875	0.80078126	0.80078125	0.001
2.39921874	0.60078125	0.60078126	-0.001
2.59921876	0.40078125	0.40078124	0.001
2.79921874	0.20078125	0.20078126	-0.001
2.99921873	0.00078125	0.00078127	-0.002
3.00000000	0.00006076	0.00000000	6.076
3.00080368	0.00006076	0.00000000	6.076
3.00241101	0.00006076	0.00000000	6.076
3.00562572	0.00006076	0.00000000	6.076
3.01205513	0.00006076	0.00000000	6.076
3.02491391	0.00006076	0.00000000	6.076
3.05063152	0.00006076	0.00000000	6.076
3.10206673	0.00006076	0.00000000	6.076
3.20493713	0.00006076	0.00000000	6.076
3.40493715	0.00006076	0.00000000	6.076
3.60493714	0.00006076	0.00000000	6.076
3.80493715	0.00006076	0.00000000	6.076
4.00000000	0.00006076	0.00000000	6.076

****

THE INTEGRATION HAS BEEN PERFORMED WITH 668 EVALUATIONS OF THE DERIVATIVES

****

BILAGA 2 JÄMFÖRELSE AV NOGGRANNHET ...

Lösning 2.

t	y_{ber}	y_{korr}	$(y_{ber} - y_{korr})*10^5$
0.00000000	0.00000000	0.00000000	0.000
0.20000000	0.00000000	0.00000000	0.000
0.40000000	0.00000000	0.00000000	0.000
0.60000000	0.00000000	0.00000000	0.000
0.80000000	0.00000000	0.00000000	0.000
1.00000000	0.00012153	0.00000000	12.153
1.20000000	0.20012154	0.20000000	12.153
1.40000001	0.40012154	0.40000001	12.153
1.60000001	0.60012154	0.60000001	12.153
1.80000001	0.80012155	0.80000001	12.154
2.00000000	1.00000000	1.00000000	0.000
2.19999999	0.80000001	0.80000001	0.000
2.39999998	0.60000002	0.60000002	0.000
2.59999996	0.40000003	0.40000004	-0.000
2.79999995	0.20000004	0.20000005	-0.001
2.99999994	0.00006076	0.00000006	6.070
3.19999993	0.00006076	0.00000000	6.076
3.39999992	0.00006076	0.00000000	6.076
3.59999990	0.00006076	0.00000000	6.076
3.79999989	0.00006076	0.00000000	6.076
3.99999988	0.00006076	0.00000000	6.076
4.00000000	0.00006076	0.00000000	6.076

****

THE INTEGRATION HAS BEEN PERFORMED WITH 698 EVALUATIONS OF THE DERIVATIVES

****

Lösning 3.

t	y_{ber}	y_{korr}	$(y_{ber} - y_{korr})*10^5$
0.00000000	0.00000000	0.00000000	0.000
0.20000000	0.00000000	0.00000000	0.000
0.40000000	0.00000000	0.00000000	0.000
0.50000000	0.00000000	0.00000000	0.000
0.60000000	0.00000000	0.00000000	0.000
0.80000000	0.00000000	0.00000000	0.000
0.90000000	0.00000000	0.00000000	0.000
0.95000000	0.00000000	0.00000000	0.000
0.97500000	0.00000000	0.00000000	0.000
0.98750000	0.00000000	0.00000000	0.000
0.99375000	0.00000000	0.00000000	0.000
0.99687500	0.00000000	0.00000000	0.000
0.99843750	0.00000000	0.00000000	0.000
0.99921875	0.00000000	0.00000000	0.000
0.99960937	0.00000000	0.00000000	0.000
0.99980468	0.00000000	0.00000000	0.000
0.99990235	0.00000000	0.00000000	0.000
0.99995117	0.00000000	0.00000000	0.000
0.99997558	0.00000000	0.00000000	0.000
1.00000000	0.00000847	0.00000000	0.847
1.00001220	0.00002397	0.00001220	1.176
1.00001831	0.00003028	0.00001831	1.197
1.00002441	0.00003640	0.00002441	1.199
1.00003052	0.00004235	0.00003052	1.183
1.00003661	0.00004843	0.00003661	1.182
1.20003662	0.20004843	0.20003662	1.182
1.40003662	0.40004843	0.40003662	1.182
1.50003661	0.50004844	0.50003661	1.182
1.60003662	0.60004845	0.60003662	1.182
1.80003662	0.80004844	0.80003662	1.182
1.90003662	0.90004845	0.90003662	1.183
1.95003662	0.95004846	0.95003662	1.185
1.97503662	0.97504846	0.97503662	1.184
1.98753662	0.98754846	0.98753662	1.184
1.99378662	0.99379846	0.99378662	1.184
1.99691162	0.99692345	0.99691162	1.184
1.99847412	0.99848596	0.99847412	1.184
1.99925537	0.99926721	0.99925537	1.184
1.99964599	0.99965783	0.99964599	1.184
1.99984130	0.99985314	0.99984130	1.184
1.99993896	0.99995080	0.99993896	1.184
1.99998780	0.99999963	0.99998780	1.184
2.00000000	1.00000337	1.00000000	0.337
2.00000611	0.99999397	0.99999389	0.008
2.00000915	0.99999072	0.99999085	-0.013
2.00001222	0.99998765	0.99998778	-0.013
2.00001526	0.99998476	0.99998474	0.001

2.00001830	0.99998172	0.99998170	0.002
2.20001832	0.79998172	0.79998168	0.004
2.40001830	0.59998173	0.59998170	0.003
2.50001830	0.49998173	0.49998170	0.003
2.60001832	0.39998173	0.39998168	0.005
2.80001831	0.19998173	0.19998169	0.004
2.90001830	0.09998173	0.09998170	0.004
2.95001832	0.04998173	0.04998168	0.005
2.97501832	0.02498173	0.02498168	0.005
2.98751831	0.01248173	0.01248169	0.004
2.99376830	0.00623173	0.00623170	0.004
2.99689332	0.00310673	0.00310668	0.005
2.99845582	0.00154423	0.00154418	0.005
2.99923706	0.00076298	0.00076294	0.004
2.99962768	0.00037236	0.00037232	0.004
2.99982300	0.00017704	0.00017700	0.005
2.99992067	0.00007939	0.00007933	0.005
2.99996948	0.00003056	0.00003052	0.004
2.99999389	0.00000615	0.00000611	0.004
3.00001833	-0.00000979	0.00000000	-0.979
3.00003052	-0.00001196	0.00000000	-1.196
3.00003663	-0.00001189	0.00000000	-1.189
3.00004274	-0.00001200	0.00000000	-1.200
3.00004882	-0.00001183	0.00000000	-1.183
3.00005493	-0.00001183	0.00000000	-1.183
3.20005494	-0.00001183	0.00000000	-1.183
3.40005493	-0.00001183	0.00000000	-1.183
3.50005493	-0.00001183	0.00000000	-1.183
3.60005492	-0.00001183	0.00000000	-1.183
3.80005494	-0.00001183	0.00000000	-1.183
4.00000000	-0.00001183	0.00000000	-1.183

****

THE INTEGRATION HAS BEEN PERFORMED WITH 368 EVALUATIONS OF THE DERIVATIVES

****

Lösning 4.

t	y_{ber}	y_{korr}	$(y_{ber} - y_{korr})*10^5$
0.00000000	0.00000000	0.00000000	0.000
0.20000000	0.00000000	0.00000000	0.000
0.40000000	0.00000000	0.00000000	0.000
0.60000000	0.00000000	0.00000000	0.000
0.80000000	0.00000000	0.00000000	0.000
1.00000000	0.00000847	0.00000000	0.847
1.20000000	0.20001196	0.20000000	1.196
1.40000001	0.40001197	0.40000001	1.196
1.60000001	0.60001183	0.60000001	1.182
1.80000001	0.80001184	0.80000001	1.182
2.00000000	1.00000337	1.00000000	0.337
2.19999999	0.79999990	0.80000001	-0.011
2.39999998	0.59999991	0.60000002	-0.011
2.59999996	0.40000008	0.40000004	0.005
2.79999995	0.20000009	0.20000005	0.004
2.99999994	0.00000032	0.00000006	0.026
3.19999993	-0.00001183	0.00000000	-1.183
3.39999992	-0.00001183	0.00000000	-1.183
3.59999990	-0.00001183	0.00000000	-1.183
3.79999989	-0.00001183	0.00000000	-1.183
3.99999988	-0.00001183	0.00000000	-1.183
4.00000000	-0.00001183	0.00000000	-1.183

****

THE INTEGRATION HAS BEEN PERFORMED WITH 398 EVALUATIONS OF THE DERIVATIVES

****

EXEMPEL PÅ STRUKTURERING AV ANVÄNDARPROGRAM VID ANVÄNDNING AV KLASSEN OBJEKT

Ex 1.

```
condis(NOTEXT,NOTEXT)
BEGIN
continuous CLASS shipdynamic(the_ship); REF(ship) the_ship;
INSPECT the_ship DO
BEGIN
  x.rate := u0*Cos(psi.state) + v.state*Sin(psi.state);
  y.rate := u0*Sin(psi.state) + v.state*Cos(psi.state);
  v.rate := a1*v.state + a11*v.state*Abs(v.state) + a2*r.state
  + b1*delta.state;
  r.rate := a3*v.state + a4*r.state + a22*r.state*Abs(v.state)
  + b2*delta.state;
  psi.rate := r.state;
END *** CLASS SHIPDYNAMIC ***;

sample CLASS rapport(the_ship); REF(ship) the_ship;
INSPECT the_ship DO INSPECT utfil DO
BEGIN
  Outfix(Time,3,11);
  Outreal(x.state,4,11);
  Outreal(y.state,4,11);
  Outreal(v.state,4,11);
  Outreal(r.state,4,11);
  Outfix(psi.state,4,11);
  Outreal(delta.state,4,11);
  Outimage;
END *** CLASS RAPPORT ***;


object CLASS ship(shipname);VALUE shipname;TEXT shipname;
BEGIN
  REF(variable) x,y,psi,v,r,delta;
  REF(shipdynamic) dyn;
  REF(sample) spy;
  REAL u0,a1,a2,a3,a4,b1,b2,a11,a22;

  x :- NEW variable(0); x.Into(THIS ship);
  y :- NEW variable(0); y.Into(THIS ship);
  psi :- NEW variable(0); psi.Into(THIS ship);
  v :- NEW variable(0); v.Into(THIS ship) ;
  r :- NEW variable(0); r.Into(THIS ship) ;
  delta :- styrautomat.delta;
  dyn :- NEW shipdynamic(THIS ship); dyn.Into(THIS ship);
  spy :- NEW rapport(THIS ship); spy.Into(THIS ship);
  BEGIN REAL frq;
    request("Resultat 'frekvens' ?",
    "0",realinput(frq,TRUE),NOTEXT,nohelp);
    spy.setfrequency(frq);
  END;
```

Ex 1 forts.

```
  u0:=1;a1:= -0.581;a2:= -0.335;a3:= -4.945;a4:= -1.822;
  b1:= 0.106;b2:= -0.790;a11:= -2.017;a22:= -12.88;

END *** CLASS SHIP ***;

Process CLASS  regulator(the_ship);REF(ship) the_ship;
BEGIN
  REF(variable) delta;
  delta :- NEW variable(0);
  Hold(1);delta.state:=0.1;
  Hold(1);delta.state:=0;
END *** CLASS REGULATOR ***;

REF(Outfile) utfil;
REF(regulator) styrautomat;
REF(ship) sea_splendid;
utfil:- NEW Outfile("ship.res");
utfil.Open(Blanks(80));
styrautomat :- NEW regulator(sea_splendid);
ACTIVATE styrautomat;
sea_splendid :- NEW ship("SEA SPLENDID");

sea_splendid.start;
Hold(10);
sea_splendid.stop;
utfil.Close;
END *** CONDIS ***;
```

Ex 2.

```
condis(NOTEXT,NOTEXT)
BEGIN
object CLASS ship(namn);VALUE namn;TEXT namn;
BEGIN
  continuous CLASS shipdynamic;
  BEGIN
    x.rate := u0*Cos(psi.state) + v.state*Sin(psi.state);
    y.rate := u0*Sin(psi.state) + v.state*Cos(psi.state);
    v.rate := a1*v.state + a11*v.state*Abs(v.state) + a2*r.state
    + b1*delta.state;
    r.rate := a3*v.state + a4*r.state + a22*r.state*Abs(v.state)
    + b2*delta.state;
    psi.rate := r.state;
  END *** CLASS SHIPDYNAMIC ***;

  sample CLASS rapport;
  INSPECT utfil DO
  BEGIN
    Outfix(Time,3,11);
    Outreal(x.state,4,11);
    Outreal(y.state,4,11);
    Outreal(v.state,4,11);
    Outreal(r.state,4,11);
    Outfix(psi.state,4,11);
    Outreal(delta.state,4,11);
    Outimage;
  END *** CLASS RAPPORT ***;

  REF(variable) x,y,psi,v,r,delta;
  REF(shipdynamic) dyn;
  REF(sample) spy;
  REAL u0,a1,a2,a3,a4,b1,b2,a11,a22;

  x :- NEW variable(0); x.Into(THIS ship);
  y :- NEW variable(0); y.Into(THIS ship);
  psi :- NEW variable(0); psi.Into(THIS ship);
  v :- NEW variable(0); v.Into(THIS ship) ;
  r :- NEW variable(0); r.Into(THIS ship) ;
  delta :- styrautomat.delta;
  dyn :- NEW shipdynamic; dyn.Into(THIS ship);
  spy :- NEW rapport; spy.Into(THIS ship);
  BEGIN REAL frq;
    request("Resultat 'frekvens' ?",
    "0",realinput(frq,TRUE),NOTEXT,nohelp);
    spy.setfrequency(frq);
  END;

  u0:=1;a1:= -0.581;a2:= -0.335;a3:= -4.945;a4:= -1.822;
  b1:= 0.106;b2:= -0.790;a11:= -2.017;a22:= -12.88;

END *** CLASS SHIP ***;
```

Ex 2 forts.

```
Process CLASS regulator(the_ship); REF(ship) the_ship;
BEGIN
  REF(variable) delta;
  delta :- NEW variable(0);
  Hold(1);delta.state:=0.1;
  Hold(1);delta.state:=0;
END *** CLASS REGULATOR ***;

REF(Outfile) utfil;
REF(regulator) styrautomat;
REF(ship) sea_splendid;
utfil:- NEW Outfile("ship.res");
utfil.Open(Blanks(80));
styrautomat :- NEW regulator(sea_splendid);
ACTIVATE styrautomat;
sea_splendid :- NEW ship("SEA SPLENDID");

sea_splendid.start;
Hold(10);
sea_splendid.stop;
utfil.Close;
END **** CONDIS ***;
```

DIALOGEXEMPEL

Nedanstående dialog är från ett program som skall beräkna följande problem:

$$\text{Lös } y(t) = \int_0^t f(s)\,ds \;;\; y(0) = 0 \text{ för } 0 <= t <= 4 \quad \text{om}$$

$$f(t) = \begin{cases} 0 & ; \; t < 1 \;,\; t >= 3 \\ 1 & ; \; 1 <= t < 2 \\ -1 & ; \; 2 <= t < 3 \end{cases}$$

Genom att sätta vissa integrationsparametrar till dåliga värden framkallas en del fel. Den del av texten som är understruken anger det som användaren har skrivit på sin terminal. Text efter utropstecken i slutet av rader är kommentarer inlagda i efterhand.

Dialogen är upplagd på fil med ett program SIMSES som registerar all dialog mellan program och användare.

```
.ex nysteg

SIMULA:  NYSTEG
LINK:  Loading
[LNKXCT        NYSTEG execution]
Which integrationmethod do you want ?
        RK (Runge-Kutta-England)
        PC (Predictor-correctormethod type DHAMDI)
        EULER
        ADAMS
        TRAPEZ
        HEUN
*/RK/:                              <RETURN> !Förstahandssvar.
Actual dtmin : 0.000E+00
New dtmin :/0.0000E+00/:            0.2      !Detta min-steg är
                                              för stort.
Actual dtmax : 0.000E+00
New dtmax :/0.0000E+00/:            1
Max relerror ?                      E-5
Max abserror ?                      0        !Ej tillåtet värde.
Max abserror > 0 !
Max abserror ?                      ?        !För att få ev ytter-
                                             !liggare information.
Max Abserror must be > 0 to avoid problems when state is 0.
Max abserror ?                      1E-5
```

```
 Result on file ?/yes/:              <RETURN> !Denna fråga kommer
                                              !inte från CONDIS.
 'Frequency  of result ?             0        ! -- - ."- --- -

      tid         y        yprick
   0.00000    0.00000    0.00000       Last rk !Resultatutskrift
   0 50000    0 00000    0.00000       Last rk !från programmet
   0 75000    0 00000    0.00000       Last rk
   0 95000    0.00000    0 00000       Last rk

***COMBINEDSIMULATION
***ERROR 1  THE REQUESTED INTEGRATION ACCURACY CAN NOT BE ACHIVED  .RK
***ENCOUNTERED AT TIME  1.1500E+00

            DT            DTMIN           DTMAX

     2.0000E 01      2 0000E-01      1.0000E+00

Do you want to stop execution ?/yes/ no

Do you want to      (Change) method    (Change) dtmin (and dtmax) ,
(Enter) SIMDDT , Stop (execution) ,
(Change rel and abs) error (for the variable that caused the error) ,
*                                   dtmin
Actual dtmin   2.000E-01
New dtmin  /2.0000E 01/:            ?
Dtmin indicates the minimum steplength during RK integration
It has no meaning during other methods
New dtmin  /2.0000E-01/             0  !Ej tillåtet värde

New dtmin :/2.0000E-01/:            .1
Actual dtmax   1.000E+00
New dtmax :/1.0000E+00/:            ?
During RK- and PC intgration dtmax is the maximum steplength
During Euler Adams or Trapez it is the constant steplength
New dtmax  /1.0000E+00/:            05
Dtmax must be >= dtmin
New dtmax :/1 0000E+00/             .1
Do you want to      Continue (execution)   (Change) method ,
(Change) dtmin (and dtmax)    (Enter) SIMDDT    Stop (execution) ,
(Change rel and abs) error (for the variable that caused the error)
*                                   cont

***COMBINEDSIMULATION
***ERROR 1: THE REQUESTED INTEGRATION ACCURACY CAN NOT BE ACHIVED  :RK
***ENCOUNTERED AT TIME  1 0500E+00

            DT            DTMIN           DTMAX

     1.0000E-01      1 0000E-01      1.0000E 01
```

```
Do you want to stop execution ?/yes/:no
Do you want to : , (Change) method , (Change) dtmin (and dtmax) ,
(Enter) SIMDDT , Stop (execution) ,
(Change rel and abs) error (for the variable that caused the error) ,
*                                   method

Which integrationmethod do you want ?
        RK (Runge-Kutta-England)
        PC (Predictor-correktormethod type DHAMDI)
        EULER
        ADAMS
        TRAPEZ
        HEUN
*/RK/                               pc
Do you want to : , Continue (execution) , (Change) method ,
(Change) dtmin (and dtmax) , (Enter) SIMDDT , Stop (execution) ,
(Change rel and abs) error (for the variable that caused the error) ,
*                                   dtmin
Actual dtmin : 1.000E-01
New dtmin :/1.0000E-01/:            1E-5 !Eftersom två RK-steg måste

                                         !vara godkända innan PC kan
                                         !starta, måste dtmin ändras.
Actual dtmax : 1.000E-01
New dtmax :/1.0000E-01/: <RETURN>
Do you want to : , Continue (execution) , (Change) method ,
(Change) dtmin (and dtmax) , (Enter) SIMDDT , Stop (execution) ,
(Change rel and abs) error (for the variable that caused the error) ,
*                                   error
Actual values:
    Abserror+Abs( Relerror *   State)=Max allowed < Estimated
                                        error       error
   1.00E-05     1.00E-05    1.42E-01    1.01E-05    1.67E-03
New relerror :/ 1.000E-05/:         0
New abserror :/ 1.000E-05/:         <RETURN>

Do you want to : , Continue (execution) , (Change) method ,
(Change) dtmin (and dtmax) , (Enter) SIMDDT , Stop (execution) ,
(Change rel and abs) error (for the variable that caused the error) ,
*                                   simddt
If you give SIMDDT the comand ´proceed´ you´ll get the last menu
again.

ZYQ212 DEBUG MODE ENTERED FROM LINE  CONDIS:1680
SIMDDT>out themonitor.halvings      !Ex på vad man kan göra i SIMDDT.
6
SIMDDT>inp max_halvings := 10       !Detta kommer att medföra ett nytt
                                    !integrationsfel.
SIMDDT>proceed
```

```
Do you want to : , Continue (execution) , (Change) method ,
(Change) dtmin (and dtmax) , (Enter) SIMDDT , Stop (execution) ,
(Change rel and abs) error (for the variable that caused the error)
*                                   cont
   0.97500    0.00000    0.00000      Last rk
   0.98750    0.00000    0.00000      Last rk
   0.99375    0.00000    0.00000      Last pc
   0.99687    0.00000    0.00000      Last pc

***COMBINEDSIMULATION
***ERROR 1: THE REQUESTED INTEGRATION ACCURACY CAN NOT BE ACHIVED  :PC
***ENCOUNTERED AT TIME  1.0000E+00

            DT            DTMIN           DTMAX

     3.1250E-03      1.0000E-05      1.0000E-01

Do you want to stop execution ?/yes/:no
Do you want to : , (Change) method , (Change) dtmin (and dtmax) ,
(Enter) SIMDDT , Stop (execution) ,
(Change rel and abs) error (for the variable that caused the error) ,
(Change) max_halvings , (Change) disc_halvings
*                                   max_halvings
Actual value of max_halvings :    10

New max_halvings :/30/:           ?
Max_halvings indicates the number of bisections of the steplength
during PC-integration, that is allowed before the execution is
interrupted with an error.

New max_halvings :/30/:           <RETURN>

Do you want to : , Continue (execution) , (Change) method ,
(Change) dtmin (and dtmax) , (Enter) SIMDDT , Stop (execution) ,
(Change rel and abs) error (for the variable that caused the error) ,
(Change) max_halvings , (Change) disc_halvings ,
*                                   cont
```

```
      0.99844   0.00000   0.00000      Last pc
      0.99922   0.00000   0.00000      Last pc
      .......
      ......
      2.99689   0.00311  -1.00000      Last pc
      2.99846   0.00154  -1.00000      Last pc
      2.99924   0.00076  -1.00000      Last pc
      2.99963   0.00037  -1.00000      Last pc
      2.99982   0.00018  -1.00000      Last pc
      2.99992   0.00008  -1.00000      Last pc
      2.99997   0.00003  -1.00000      Last pc
      2.99999   0.00001  -1.00000      Last pc
      3.00002  -0.00001   0.00000      Last pc
      3.00003  -0.00001   0.00000      Last pc
      3.00004  -0.00001   0.00000      Last pc
      3.00004  -0.00001   0.00000      Last pc
      3.00005  -0.00001   0.00000      Last pc
      3.00005  -0.00001   0.00000      Last pc
      3.10005  -0.00001   0.00000      Last rk
      .....
      .....
      3.90005  -0.00001   0.00000      Last pc
      4.00000  -0.00001   0.00000      Last rk
*******************************************************************************
THE INTEGRATION HAS BEEN PERFORMED WITH   472 EVALUATIONS OF THE DERIVATIVES
*******************************************************************************
                                    !Utskriften ovan kommer inte från CONDIS,
5 garbage collection(s) in 74 ms    !men den går lätt att få genom att
                                    !lägga in en uppräkning av en variabel
End of SIMULA program execution.    !i varje continuous CLASS.
CPU time: 2.12 Elapsed time: 8:35.38
```

REFERENSER

1. SIM, R:
CADSIM. Users Guide and Refernce Manual.
Imperial College, Publ no. 75/23, London 1975.

2. HELSGAUN,Keld : COMBINEDSIMULATION
2.1 RAPPORT NR 5: INTRODUKTION Kompendium december 1978.
2.2 RAPPORT NR 4: BRUGERHÅNDBOG Kompendium november 1978.
2.3 RAPPORT NR 6: DOKUMENTATION Kompendium januari 1979.
Samtliga:
Datologi,Roskilde Universitetscenter,4000 Roskilde

3. SHAMPINE, L F; WATTS, H A; DAVENPORT, S M (1975) :
Solving nonstiff ordinary differential equations - the state of art.
SIAM REVIEW Vol 18, No 3, July 1976.

4. Subroutines HPCG and DHPCG
Scientific Subroutine Package, Version III, IBM Manual H20-0205, pp 337-339.

5. FICK, Göran (1971) :
DHAMDI a FORTRAN subroutine to integrate a set of first order, ordinary differential equations containing discontinuities.
FOA 2 rapport : C2504 - E5, november 1971.

6. RALSTON, Anthony
Numerical integration methods for the solution of ordinary differential equations.
In "Mathematical Methods for Digital Computers" edited by Ralston, Anthony and Wilf, Herbert S, John Wiley and Sons, Inc, New York 1960, pp 95-109.

7. HALIN, H J:
Integration across discontinuities in ordinary differential equations using power series.
SIMULATION, Vol 32, No 2, January 1979.

8. HINDMARSCH, A C (1972) :
Linear multistep methods for ordinary differential equations: Method formulations, stability, and the methods of Nordsieck and Gear.
UCRL-51186 Rev 1, Lawrence Livermore Laboratory, University of California, Livermore, California, March 20 1972.

9. OHLIN, Mats
Safe Conversional SIMULA Programs Using SAFEIO.
FOA Report C10044-M3(E5), Swedish National Defense Research Institute, April 1979.

10. Debugging SIMULA programs with SIMDDT.
DECsystem 10/20 SIMULA Language Handbook Part 2,
FOA 1 Report C8399-M3(E5), Swedish National Defense Research Institute, Revised edition December 1977, pp 57-71.

11. ÅSTRÖM, K J; KÄLLSTRÖM C (1976):
Identification of Ship Steering Dynamics.
Automatica, Vol 12, pp 9-22.

12. DECsystem 10/20 SIMULA Language Handbook Part 3,
FOA Report C10045-M3(E5), Swedish National Defense Research Institute, August 1979.

13. ENGLAND, R :
Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations.
Computer Journal, Vol. 12, 1969, pp 116-170.

14. YOUNG, D M; GREGORY, R T :
A survey of numerical mathematics.