Simulation of a Hydraulic System with the Simulation Package Simnon

Braun, Konrad

1985

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
INTERNAL REPORT

Date of issue
February 1985

Document Number
CODEN: LUTFD2/(TFRT-7306)/1-36./(1985)

Supervisor

Sponsoring organisation

Author(s)
Konrad Braun

Title and subtitle
Simulations of a hydraulic system with the simulation package Simnon

Abstract

This report presents simulations of a hydraulic system with the simulation package Simnon. The four integration algorithms available in the package were evaluated for simulation of this system. The last chapter summarizes the experiences made by using this package the first time.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
36

Security classification

Recipient's notes

CODEN: LUTFD2/(TFRT-7306)/1-36/(1985)

Simulation of a hydraulic system with the simulation package Simnon

Konrad Braun

Department of Automatic Control
Lund Institute of Technology

February 1985

I CONTENTS

II SYMBOLS

A	Area	[m^2]
B	Bulk modulus of elasticity	[N/m^2]
d	Diameter	[m]
F	Force	[N]
g	Acceleration of gravity	[m/s^2]
K_v	Throttle constant	[-]
m	Mass	[kg]
p	Pressure	[N/m^2]
Δp	Pressure difference	[N/m^2]
r	Friction coefficient in the cylinder	[m]
t	Time	[m^2]
V	Volume flow	[m^3/s]
x	State vector	
y	Position	[m]
δ	Constant	
ς	Pressure loss coefficient	[-]
ν	Kinematic viscosity	[m^2/s]
ϱ	Density	[kg/m^3]
κ	Specific-heat ratio	[-]

INDICES

a	Gas volume storage
b	Oil volume storage
c	Throttle after storage
d	Pipe d
e	Damper e
f	Oil volume f in cylinder
g	Oil volume g in cylinder
h	Damper h

i	Pipe i
k	Piston
l	Pipe l
m	Throttle m
oil	Oil
s	Piston of the storage

1 INTRODUCTION

The simulation package Simnon, which was designed for simulations of nonlinear systems, was tested by simulations of this hydraulic system. The mathematical model of the system was derived in [2], the equations were taken from there.

2 DESCRIPTION OF THE SYSTEM

In high-voltage power transmission switches are needed which are able to interrupt the contact within few milliseconds. The arc is extinguished by special gas. Short switching periods and the compression of the gas need a high driving power. That is why presently most of the switches are driven by hydraulic power. Fig. 2.1 shows the schema of the hydraulic equipment for such a switch-drive.

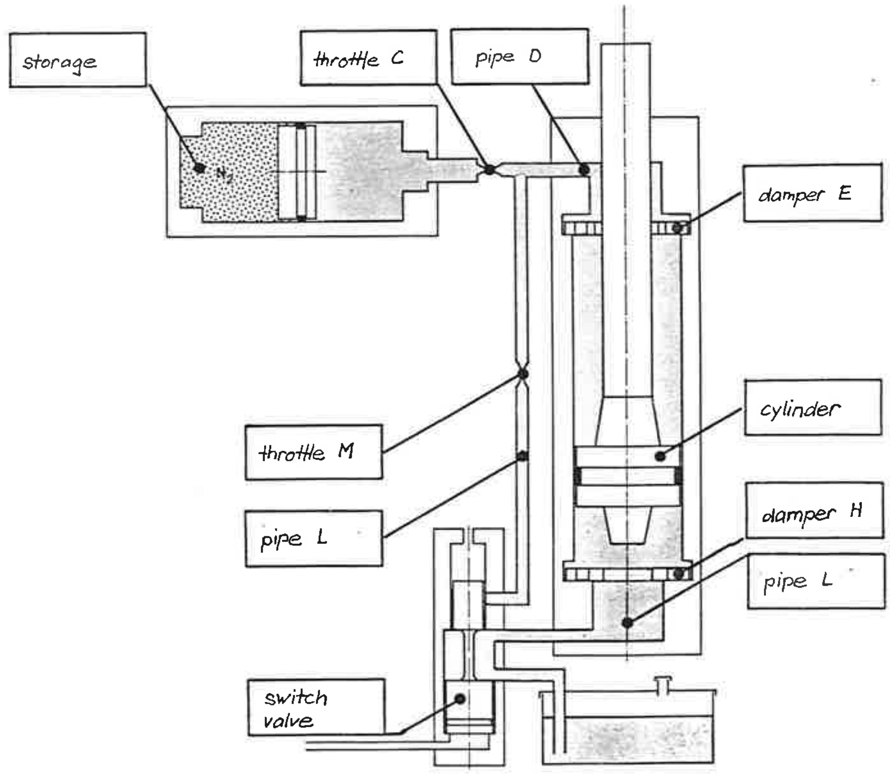


Fig. 2.1: Hydraulic switch-drive

Energy is stored in compressed CO_2 gas. The pressure is transmitted to the oil by a piston. The switching movement is performed by the differential cylinder where the position is controlled by the 2-way shut-off valve. If the pipe from the cylinder is connected with the tank, the rod moves in. This is called an opening-cycle because the electrical contact is interrupted. Conversely, when the pipe is connected with the storage the rod moves out. This is called a closeing-cycle. On both sides of the cylinder there are dampers to reduce the chock of the collision between the piston and cylinder. The damping is performed by diminishing the outlet area of the cylinder.

3 MODELL OF THE SYSTEM

The deductive modelling in [2] led to a 10th order system with the following states:

$x_1 = p_d$ pressure in pipe D
$x_2 = p_i$ pressure in pipe I
$x_3 = p_l$ pressure in pipe L
$x_4 = p_b$ pressure in pipe B
$x_5 = v_s$ velocity of the storage piston
$x_6 = y_s$ position of the storage piston
$x_7 = p_f$ pressure in volume F
$x_8 = p_g$ pressure in volume G
$x_9 = v_k$ velocity of the piston
$x_{10} = y_k$ position of the piston

The differential equations were derived with the following physical laws and heuristic relations:

- Newtons law

- law of adiabat compression of an ideal gas

- heuristic equation of the turbulent flow through a throttle

- continuity equation of flow.

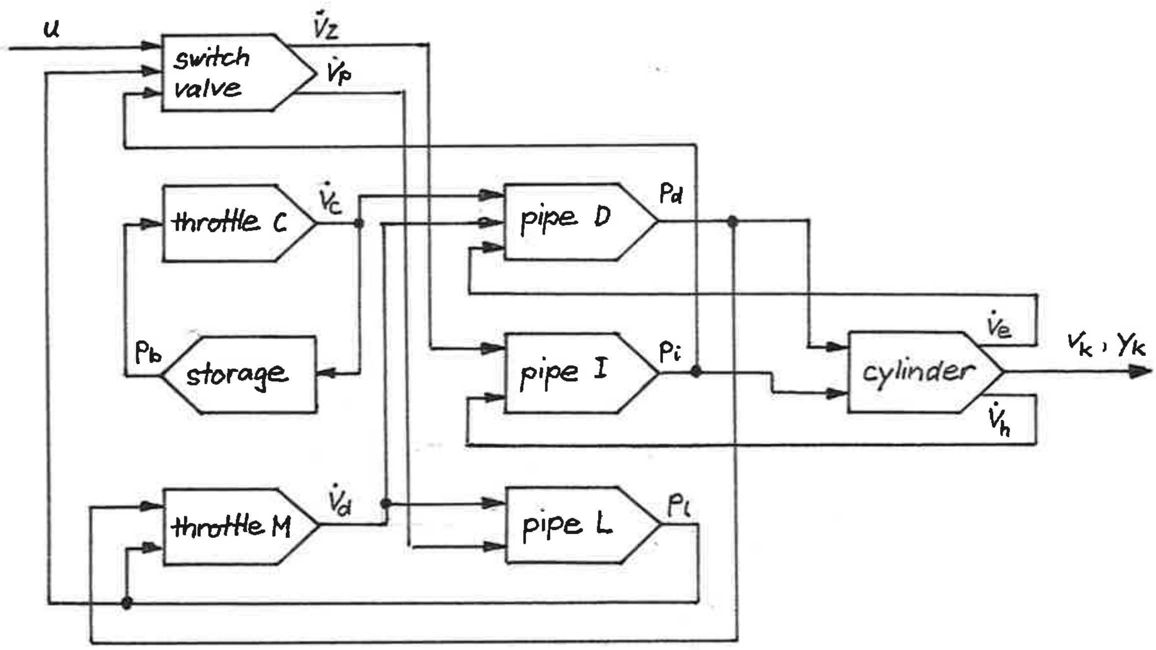


Fig. 4.1: Structure of the SIMNON-program

The dynamics of the switching valve are neglected because the time constant is very short.
The equations of the model can be found in Appendix A.

4 SIMULATIONS

4.1 Simulation Program in Simnon

The system description in SIMNON is structured as shown in fig. 4.1.

The listings of the SIMNON-program are added in Appendix B.

4.2 Integration Routines available in Simnon

Four different integration routines can be used for simulations in Simnon:

- Hamming predictor corrector (default) (HAMPC)
- Runge-Kutta variable step size (RK)
- Runge-Kutta fixed step size (RKFIX)
- Integration routine for stiff systems (DAS)

a) Hamming predictor corrector

This is a routine which yields good results for a wide range of problems. It is an implicit multistep method of order four with variable step size. The integration is started with a fourth order Runge-Kutta method.

b) Runge-Kutta variable step size

This is the well-known explicit method with variable step size. In some situations the step size can be set to very small values, and the computation requires a lot of time. A special step size strategy takes care of jump discontinuities in the right hand side of the differential equation by taking a very small step across the jump.

c) Runge-Kutta fixed step size

The step size of the explicit algorithm can be provided by the user or it will use by default one hundredth of the total integration time.
If the right step size is known and the optimal step size does not differ much over the whole integration interval this routine is very efficient. The main drawback is that there is no error control.

d) Integration routine for stiff systems

DAS is a special routine to integrate stiff systems. The fast modes should be known and indicated before the integration is started. It is also possible to solve implicit equations with this routine. A main drawback of this routine is that it can not handle discontinuities of the right hand side of the differential equation. If such a discontinuity occurs the integration is stopped and a message is given that the requested accuracy may not be reached. This algorithm is very time consuming.

4.3 Usage of the different Integration Routines

4.3.1 Hamming Predictor Corrector

It was impossible to use this routine for simulations of this system, because the algorithm could nor be started with the fourth order Runge-Kutta. Even if the maximal increment chosen was very small, the integration was stopped and the message given that the initial step size has been divided in ten bisections by the starting Runge-Kutta steps.

4.3.2 Runge-Kutta variable Step Size

This routine was the easiest one to use. It yielded good results with the default values. The plots of an opening and a closeing-cycle are added in Appendix C. The global course looks quite good. In the two pressures pd and pl there seem to be some numerical instabilities, but they do not influence the global course.

4.3.3 Runge-Kutta fixed Step Size

The problem in integrating a system with this algorithm is to find an appropriate step size. This was done executing simulations with different step sizes. Starting with the default value the step size was continuously diminished. With the increment of 0.00001 sec the result looked quite good. To verify the result the

step size was divided by two and the courses compared. As the results did not deviate much, it could be concluded that this step size was appropriate.
The results of an opening and a closeing-cycle with this integration algorithm are shown in Appendix D.

4.3.4 Integration Routine for stiff Systems

This routine should be best suited for this problem, because the differential equations are stiff. But when integrating the system with this routine a problem occurs. When the piston runs into the end of the cylinder its velocity is set to zero. This is a discontinuity which causes a termination of the integration because the requested accuracy can not be reached. The result up to this event diverges only slightly from the results yielded by the other algorithms.
The plots of the simulations performed with this routine are added in Appendix E.

4.3.5 Comparison of the Algorithms

The algorithms are compared in respect to the needed CPU-time and the ease of usage.
The needed CPU-time with a VAX 11-780 is listed in Tab. 4.1

	OPEN	CLOSE
HAMPC	-	-
RK	280.52	463.18
RKFIX		
increm.: 0.00001	122.65	125.84
increm.: 0.000005	243.67	247.55
DAS	801.42	-

Tab. 4.1: CPU-time in seconds used to simulate the system

The Runge-Kutta algorithm with variable step size was easiest to use. It could be applied without making any special arrangements.
With the appropriate step size the Runge-Kutta with fixed step size needs about the same CPU-time as the one with variable step size. Because there is no error control this routine should not be used.
To integrate the system with the DAS-routine it should be known, which modes are fast. Normally this is fulfilled. Integration with this routine consumes a lot of CPU-time and in this case the result is not better then the one yielded by the RK-routine.

5 CONCLUSIONS

This chapter summerizes the experiences made with the Simnon package.
It must be mentioned, that I was a novice using this program. I have had some experiences with other simulation packages, but these were not interactive ones.
I was impressed by how easily this program can be utilized. All the commands are very logical and therefor easy to remember.

A system consisting of some subsystems can be programmed in a natural way with the definition of a "connecting system". That makes is handy to translate a block diagram into a Simnon program.

In the formulation of a dynamic system it is advantageous to be able to program no-sort-blocks; that is blocks which are treated as a unity by the sorting procedure. In physical systems one often has to distinguish between different cases. Thus it would be good to have the feature of flow control as in higher level languages. Table 5.1 shows how a part of the Simnon program could be written in a more readable way.

```lmin=(not yk>0.0) and (fres<0.0 or vk<0.0)```
```gmax=(not yk<ykm) and (fres>0.0 or vk>0.0)```
```dvkn=fres/(mk+mn)```
```dvk=if lmin>0.0 then 0.0 else if gmax>0.0 then dvkt else dvkn```
```dyk=if lmin>0.0 then 0.0 else if gmax>0.0 then 0.0 else vk```
	```>>no-sort```
	```if (yk<=0 and (fres<0 or vk<0)) then```
	``` dvk=0.0```
	``` dyk=0.0```
	```else if (yk>=0 and (fres>0 of vk>0) then```
	``` dvk=0.0```
	``` dyk=0.0```
	```else```
	``` dvk=fres/(mk+mn)```
	``` dyk=vk```
	```endif```
	```>>sort```

Table 5.1: The advantage of no-sort-blocks

The macro facility saves a lot of time. For sequences of commands, which are often used, a macro can be defined. The whole sequence of commands can then be executed by typing the name of the macro. It is also possible to execute macros with arguments of different values. Two examples of such macros are given in Appendix C.

Making simulations with a interactive program is very efficient. In particular, debugging is very fast because the course of the simulation can be watched and the integration stopped when something unusual happens. Then it is possible to examine all the variables and parameters. With this information it is relatively easy to find the bugs.
When the program was running smoothly and simulations were done, the interactive way was no longer as advantageous, that is, some simulations took a very long time. Thus sometimes it would have been better to run a batch job.

Compared with programming simulations in FORTRAN, the usage of this package brings big advantages. About 30 hours were needed to transform the problem from the given differential equations to the Simnon program and to debug this program. A programmer with more experience could have done this job even quicker. To program these simulations with a higher level language like Pascal or FORTRAN would certainly take two or three times as much time.

In short, this package is a powerful tool in the hand of an engineer. It allows one to use the computer for simulations in a very efficient way.

6 REFERENCES

[1] Åström, K.J.
A SIMNON Tutorial
Department of Automatic Control
Lund 1984

[2] Kaufmann, A.
Digitale Simulation eines Leistungsantriebes
Diplomarbeit an der Abteilung für Maschineningenieurwesen
der ETH Zürich
Zürich 1984 (not published)

APPENDIX A: EQUATIONS OF THE MODEL

Switch Valve:

Open-cycle:

$$\dot{V}_z = \dot{V}_p = \sigma_{in} \cdot \mathrm{sign}(p_i - p_l) \cdot \sqrt{|p_i - p_l|}$$

$$\dot{V}_t = 0$$

Close-cycle:

$$\dot{V}_z = \dot{V}_t = \sigma_{out} \cdot \mathrm{sign}(p_i - p_u) \cdot \sqrt{|p_i - p_u|}$$

$$\dot{V}_p = 0$$

Pipe:

$$\frac{dp}{dt} = (\dot{V}_{in} - \dot{V}_{out}) \cdot \frac{B}{V_p}$$

Throttle:

$$\Delta p = p_{before} - p_{after}$$

$$\dot{V} = K_v \cdot A \cdot \sqrt{2/\varrho} \cdot \mathrm{sign}(\Delta p) \cdot \sqrt{|\Delta p|}$$

Storage:

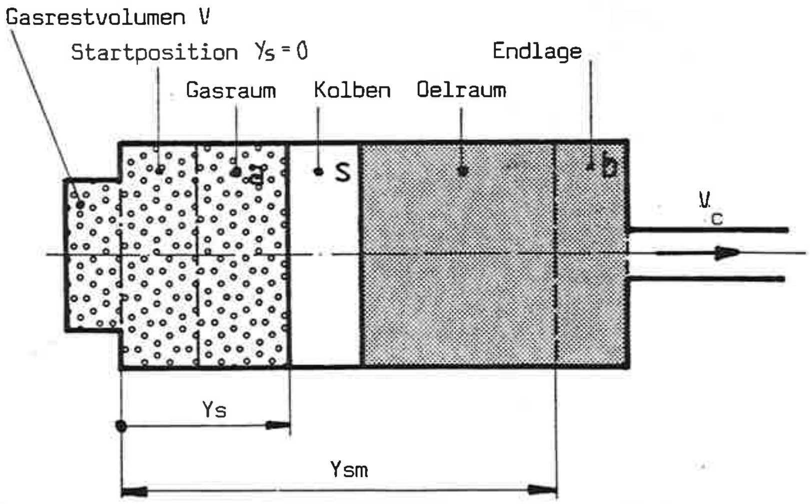


Fig. A.1: Storage

$$V_G = A_s \cdot y_s + V_r$$

$$p_g = p_{g0} \cdot (V_{g0} / V_g)^{\kappa}$$

$$V_{oil} = A_s \cdot (y_{sm} - y_s) + V_r$$

$$m = m_s + V_{oil} \cdot \varrho$$

$$\dot{p}_b = \frac{B}{V_{oil}} \cdot (A_s \cdot \dot{y}_s - \dot{V}_c)$$

$$\frac{d^2y}{dt^2} = \frac{1}{m} \cdot A_s \cdot (p_g - p_b)$$

Cylinder:

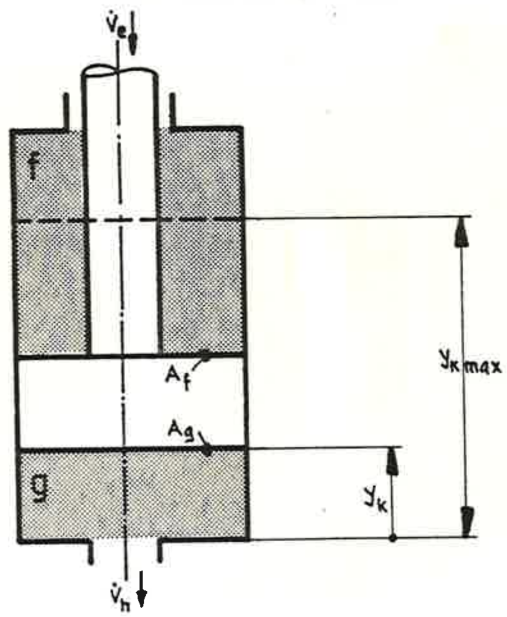


Fig. A.2: Cylinder

$$\Delta p_e = p_d - p_f$$

$$\dot{V}_{e1} = K_v \cdot A_e\ (y_k) \cdot \sqrt{2/\varrho} \cdot \mathrm{sign}(\Delta p_e) \cdot \sqrt{|\Delta p_e|}$$

$$\dot{V}_{e2} = \zeta_e \cdot \mathrm{sign}(\Delta p_e) \cdot \sqrt{|\Delta p_e|}$$

$$\dot{V}_e = \dot{V}_{e1} \qquad \text{if } \dot{V}_{e2} < 0$$

$$\dot{V}_e = \dot{V}_{e1} + \dot{V}_{e2} \qquad \text{if } \dot{V}_{e2} \geq 0$$

$$\Delta p_h = p_g - p_i$$

$$\dot{V}_{h1} = K_v \cdot A_h\ (y_k) \cdot \sqrt{2/\varrho} \cdot \mathrm{sign}(\Delta p_h) \cdot \sqrt{|\Delta p_h|}$$

$$\dot{V}_{h2} = \zeta_h \cdot \mathrm{sign}(\Delta p_h) \cdot \sqrt{|\Delta p_h|}$$

$$\dot{V}_h = \dot{V}_{h1} \qquad \text{if } \dot{V}_{h2} > 0$$

$$\dot{V}_h = \dot{V}_{h1} + \dot{V}_{h2} \qquad \text{if } \dot{V}_{h2} \leq 0$$

$$V_f = A_f \cdot (y_{km} - y_k) + V_r$$

$$\dot{p}_f = \frac{B}{V_f} \cdot (\dot{V}_e + A_f \cdot \dot{y}_k)$$

$$V_g = A_g \cdot y_k + V_r$$

$$\dot{p}_g = \frac{B}{V_g} \cdot (-\dot{V}_h - A_g \cdot \dot{y}_k)$$

$$F_v = r \cdot \dot{y}_k \cdot \varrho \cdot \nu$$

$$F_{res} = A_g \cdot p_g - A_f \cdot p_f - F_v - m_n \cdot g$$

Logic to describe the physical behaviour at the ends of the cylinder:

if $(y_k <= 0$ and $(F_{res} < 0$ or $v_k < 0))$ then

$\quad \dot{v}_k = 0.0$

$\quad \dot{y}_k = 0.0$

else if $(y_k >= 0$ and $(f_{res} > 0$ of $v_k > 0)$ then

$\quad \dot{v}_k = 0.0$

$\quad \dot{y}_k = 0.0$

else

$$\dot{v}_k = \frac{1}{m_k + m_n} \cdot F_{res}$$

$\quad \dot{y}_k = v_k$

endif

APPENDIX B: SIMNON PROGRAM

```
CONNECTING SYSTEM HYDRO
"Connecting system for simulation of a hydraulic drive.
"In this system the dynamics of the storage are included.
Time t
pl[SWIVAL]=ypl[PIPEL]
pi[SWIVAL]=ypi[PIPEI]
pd[THROTM]=ypd[PIPED]
pl[THROTM]=ypl[PIPEL]
pd[THROTC]=ypd[PIPED]
pb[THROTC]=ypb[STORE]
dvc[STORE]=dvc[THROTC]
dvc[PIPED]=dvc[THROTC]
dvd[PIPED]=dvd[THROTM]
dve[PIPED]=dve[CYLIND]
dvh[PIPEI]=dvh[CYLIND]
dvz[PIPEI]=dvz[SWIVAL]
dvd[PIPEL]=dvd[THROTM]
dvp[PIPEL]=dvp[SWIVAL]
pd[CYLIND]=ypd[PIPED]
pi[CYLIND]=ypi[PIPEI]
END
```

```
CONTINUOUS SYSTEM SWIVAL
"Algebraic equations of the switch valve
Input pl pi
Output dvz dvp
Time t
"
delp1=pi-pu
dv1=sigin*sign(delp1)*sqrt(abs(delp1))
delp2=pi-pl
dv2=sigout*sign(delp2)*sqrt(abs(delp2))
dvz=if u>0 then dv1 else dv2
dvp=if u>0 then 0.0 else dv2
"
pu: 1.0e5
sigin: 0.2e-5
sigout: 0.25e-5
tdel: 0.003
u: 1.0
END
```

```
CONTINUOUS SYSTEM STORE
"Derivatives of the storage
Input dvc
Output ypb
State pb vs ys
Der dpb dvs dys
"
pgas=pstat*((pvg0/(pstat*(as*ys+vgasr)))↑kappa)
voil=as*(ysm-ys)+voilr
dpb=b/voil*(as*vs-dvc)
mt=ms+voil*rho
dvs=(pgas-pb)*as/mt
dys=vs
ypb=pb
"
pgas0: 230.0e5
vgas0: 0.031
pvg0: 7.13e5
vgasr: 0.001
kappa: 1.84
as: 25.45e-3
ysm: 1.18
voilr: 5.0e-6
ms: 7.0
b: 1.49e9
rho: 0.85e3
pstat: 306.0e5
vs: 0.0
END


CONTINUOUS SYSTEM THROTC
"Algebraic equation of trottle c
Input pd pb
Output dvc
"
cc1=kv*a*sqrt(2.0/rho)
delp=pb-pd
dvc=cc1*sign(delp)*sqrt(abs(delp))
"
kv: 0.7
a: 7.85e-5
rho: 0.85e3
END


CONTINUOUS SYSTEM THROTM
"Algebraic equation of trottle m
Input pd pl
Output dvd
"
cc1=kv*a*sqrt(2.0/rho)
delp=pl-pd
dvd=cc1*sign(delp)*sqrt(abs(delp))
"
kv: 0.7
a: 0.127e-3
rho: 0.85e3
END
```

```
CONTINUOUS SYSTEM PIPED
"Derivative of pipe d
Input dvc dvd dve
Output ypd
State pd
Der dpd

ypd=pd
dpd=(dvc+dvd-dve)*b/vd
"
b: 1.49e9
vd: 80.9e-6
pstat: 306.0e5
END
```

```
CONTINUOUS SYSTEM PIPEI
"Derivative of pipe i
Input dvh dvz
Output ypi
State pi
Der dpi
"
dpi=(dvh-dvz)*b/vi
ypi=pi
"
b: 1.49e9
vi: 38.5e-6
pstat: 306.0e5
END
```

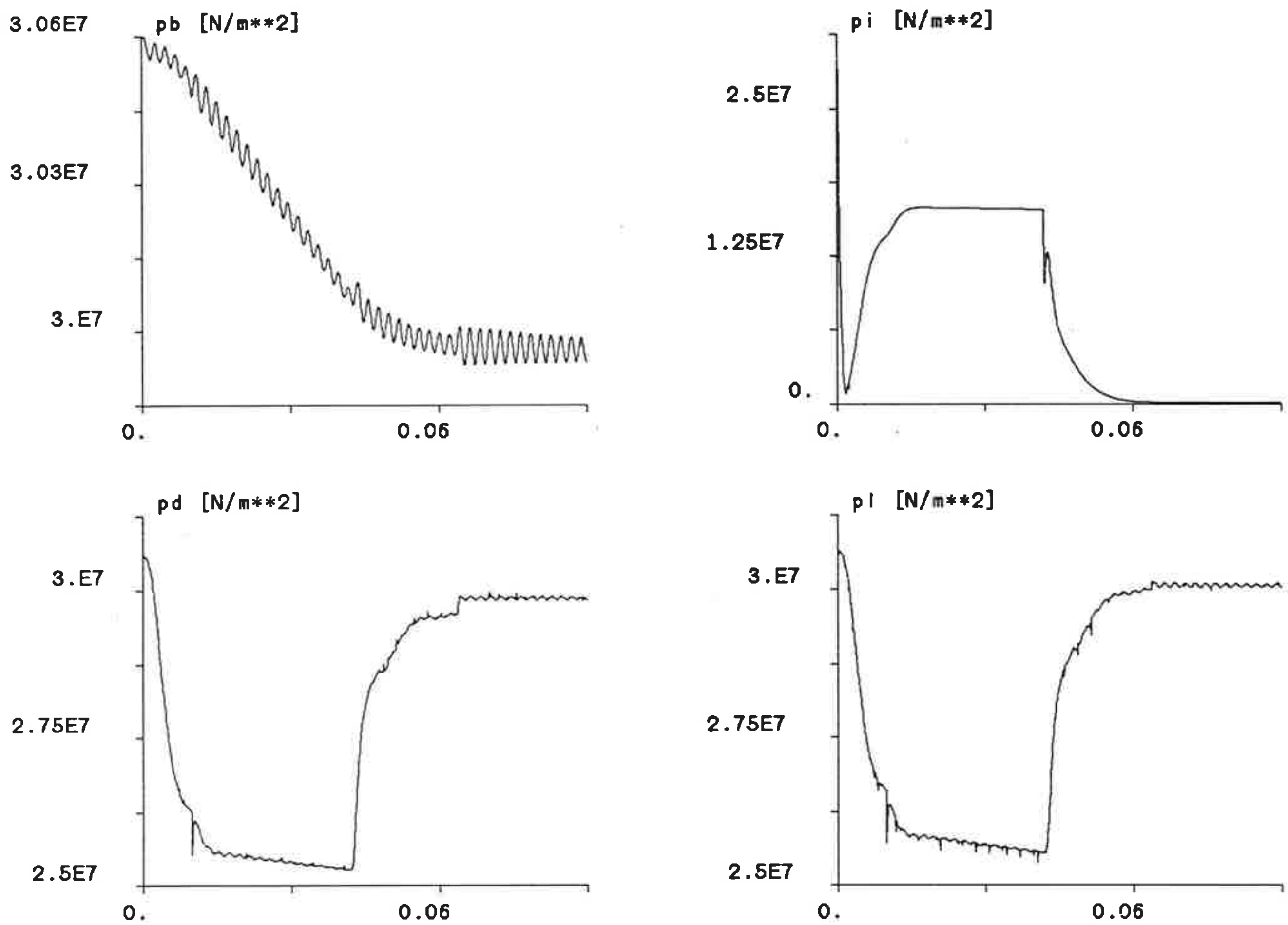
```
CONTINUOUS SYSTEM PIPEL
"Derivative of pipe l
Input dvd dvp
Output ypl
State pl
Der dpl
"
dpl=(dvp-dvd)*b/vl
ypl=pl
"
b: 1.49e9
vl: 29.0e-6
pstat: 306.0e5
END
```

```
CONTINUOUS SYSTEM CYLIND
"Derivatives of the cylinder with dampers
Input pd pi
Output dve dvh yvk yyk
State pf pg vk yk
Der dpf dpg dvk dyk
"
ae1=(dae*dae-(de1-c1*(ykm-yk))*(de1-c1*(ykm-yk)))*copi/4.0
ae2=c2*sqrt((yke-yk)*(yke-yk)+c3)
ae=if yk>yke then ae1 else if ae2<ae3 then ae2 else ae3
delpe=pd-pf
cc1=sign(delpe)*sqrt(abs(delpe))
dve1=c4*cc1*ae
dve2=zete*cc1
dvet=dve1+dve2
dve=if dve2<0.0 then dve1 else dvet
vf=af*(ykm-yk)+vfr
dpf=b/vf*(dve+af*vkh)
"
ah1=(dah*dah-(dh-c5*yk)*(dh-c5*yk))*copi/4.0
ah2=c6*sqrt(yk*yk+c7)
ah=if yk<yh then ah1 else if ah2<ah3 then ah2 else ah3
delph=pg-pi
cc2=sign(delph)*sqrt(abs(delph))
dvh1=c4*cc2*ah
dvh2=zeth*cc2
dvht=dvh1+dvh2
dvh=if dvh2>0.0 then dvh1 else dvht
vg=ag*yk+vgr
dpg=b/vg*(-dvh-ag*vkh)
"
fv=c8*vk
fres=ag*pg-af*pf-fv-9.81*(mk+mn)
lmin=(not yk>0.0) and (fres<0.0 or vk<0.0)
gmax=(not yk<ykm) and (fres>0.0 or vk>0.0)
dvkn=fres/(mk+mn)
dvk=if lmin>0.0 then 0.0 else if gmax>0.0 then dvkt else dvkn
vkh=if lmin>0.0 then 0.0 else if gmax>0.0 then 0.0 else vk
dyk=vkh
yvk=vkh
yyk=yk
"
copi: 3.1415926
dae: 30e-3
de1: 29.8e-3
c1: 1.05e-2
c2: 9.345e-2
c3: 7.57e-6
ae3: 2.16e-4
c4: 3.395e-2
zete: 1.746e-6
yke: 182.5e-3
ykm: 212.0e-3
vfr: 5.0e-6
af: 1.244e-3
dah: 19.0e-3
dh: 18.9e-3
c5: 4.888e-2
c6: 5.704e-2
c7: 1.534e-5
ah3: 2.835e-4
```

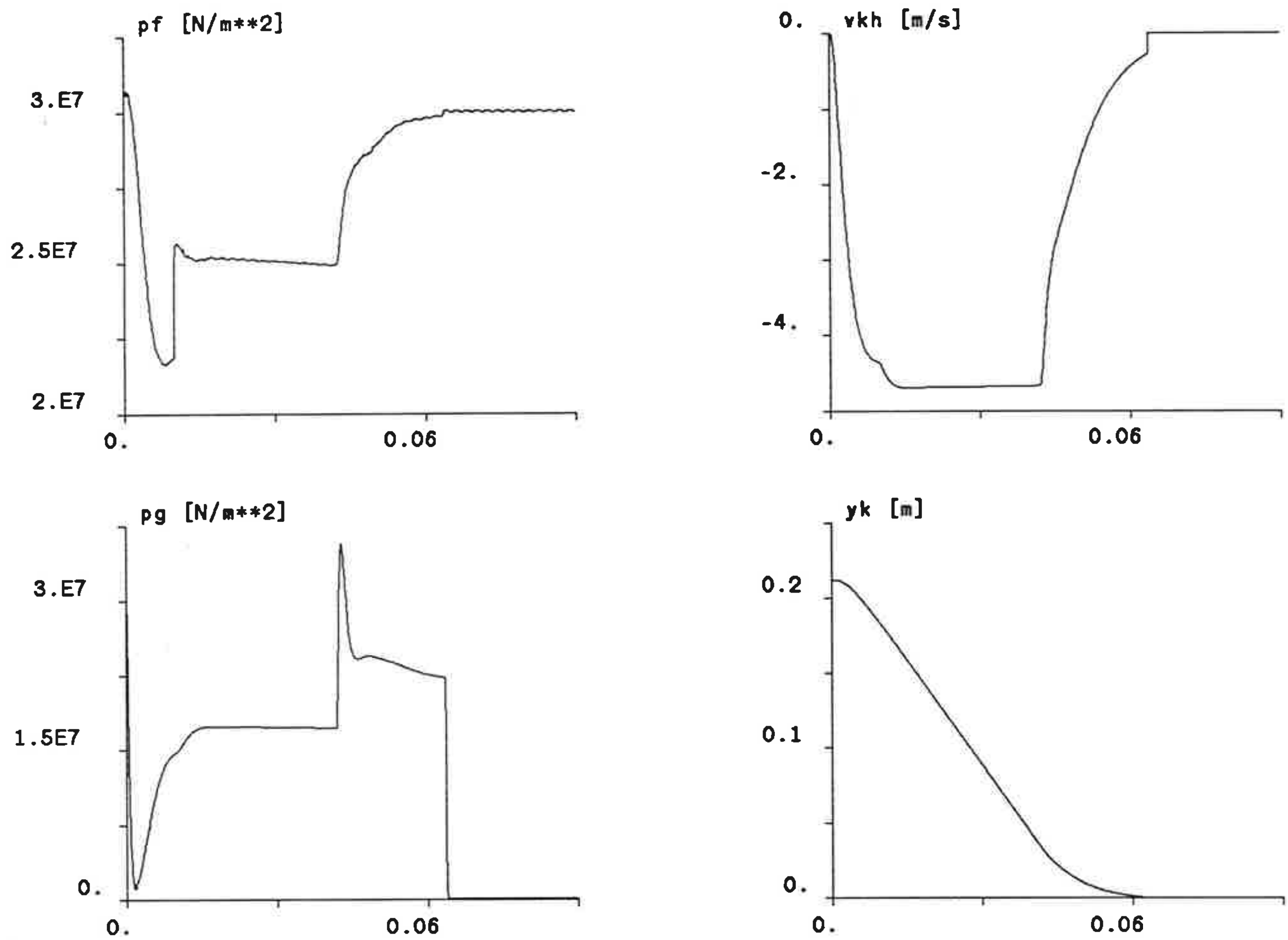
```
yh: 32.5e-3
ag: 1.735e-3
vgr: 5.0e-6
zeth: 1.819e-6
c8: 3.305e2
mk: 6.05
mn: 26.0
b: 1.49e9
vk: 0.0
yk: 212.0e-3
END


MACRO OPEN tmax ; psta
default psta=306E5
par u[swival] : 1.0
init pd[piped] : psta
init pi[pipei] : psta
init pl[pipel] : psta
init pf[cylind] : psta
init pg[cylind] : psta
init pb[store] : psta
let ys1=0.031*230.0e5
let ys2=ys1/psta
let ys3=ys2-0.001
let ys0=ys3/25.45e-3
init ys[store] : ys0
par pstat[store] : psta
switch graph on
split 1 1
axes h 0 tmax v 0 1000.0
plot mscpu
simu 0.0 tmax
END


MACRO CLOSE tmax ; psta
"This macro is to simulate a close-cycle
default psta=306E5
let pu=1.0e5
par u[swival] : 0.0
init pd[piped] : psta
init pi[pipei] : pu
init pl[pipel] : psta
init pf[cylind] : psta
init pg[cylind] : pu
init pb[store] : psta
let ys1=0.031*230.0e5
let ys2=ys1/psta
let ys3=ys2-0.001
let ys0=ys3/25.45e-3
init ys[store] : ys0
init yk[cylind] : 0.0
par pstat[store] : psta
switch graph on
split 1 1
axes h 0 tmax v 0 1000.0
plot mscpu
simu 0.0 tmax
END
```

APPENDIX C: RESULTS WITH ALGORITHM RUNGE-KUTTA VARIABLE STEP SIZE

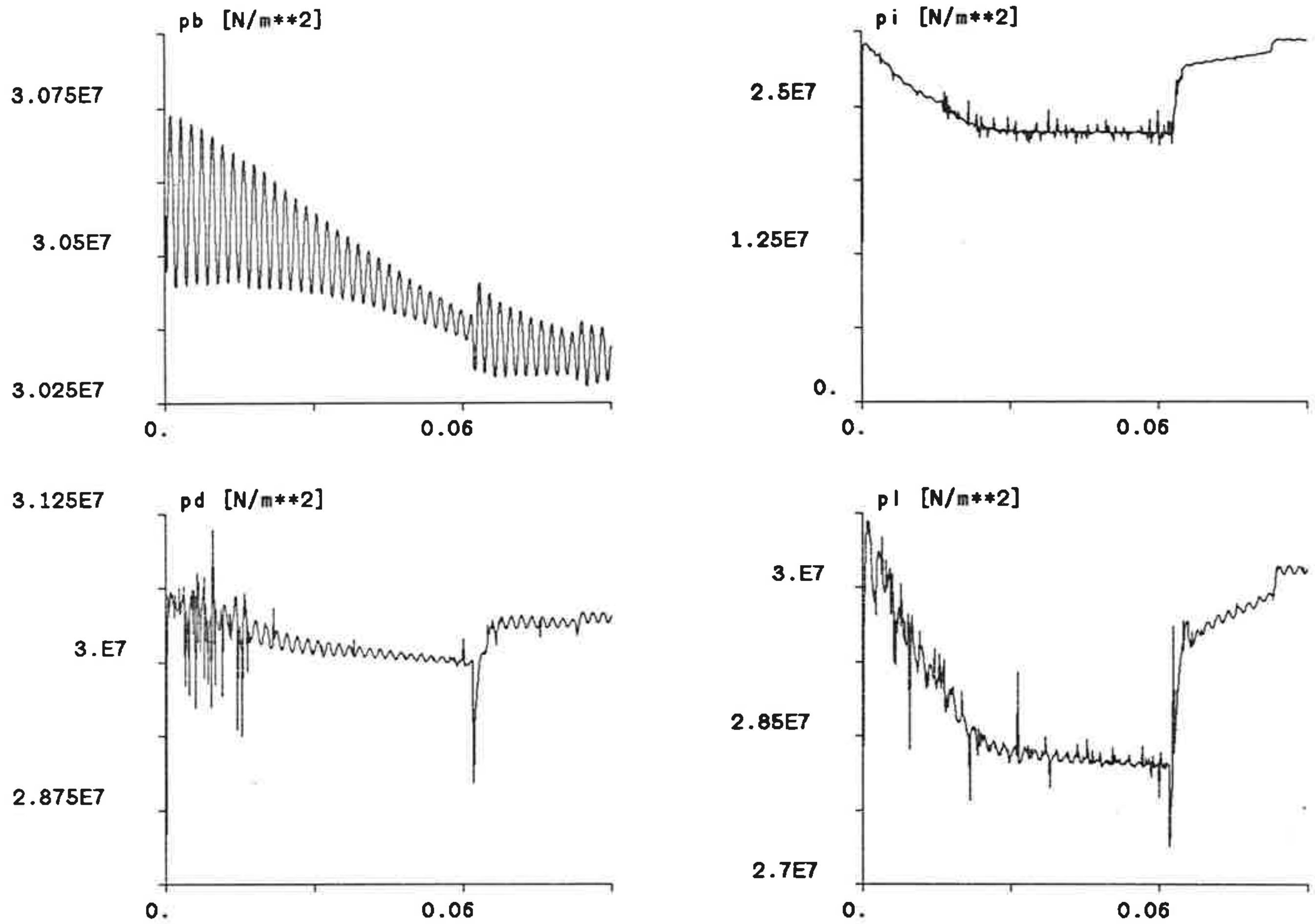
85.02.18 - 11:44:08 nr: 3
hcopy "Open-cycle; algor: RK; error: 0.001



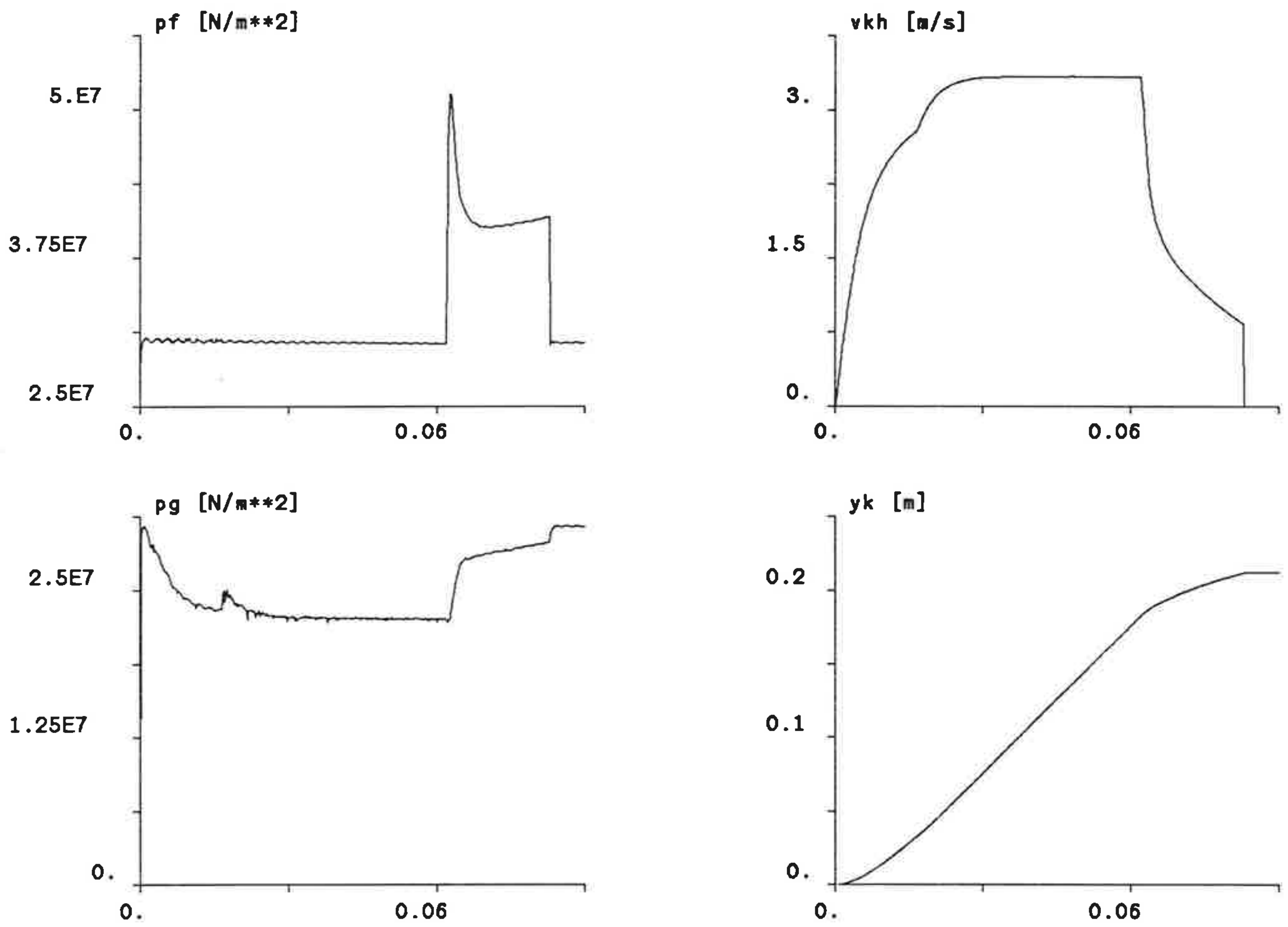
85.02.18 - 11:42:24 nr: 2
hcopy "Open-cycle; algor.: RK; error: 0.001



85.02.21 - 18:39:48 nr: 1
hcopy "Close-cycle; algor.: RK; error: 0.001

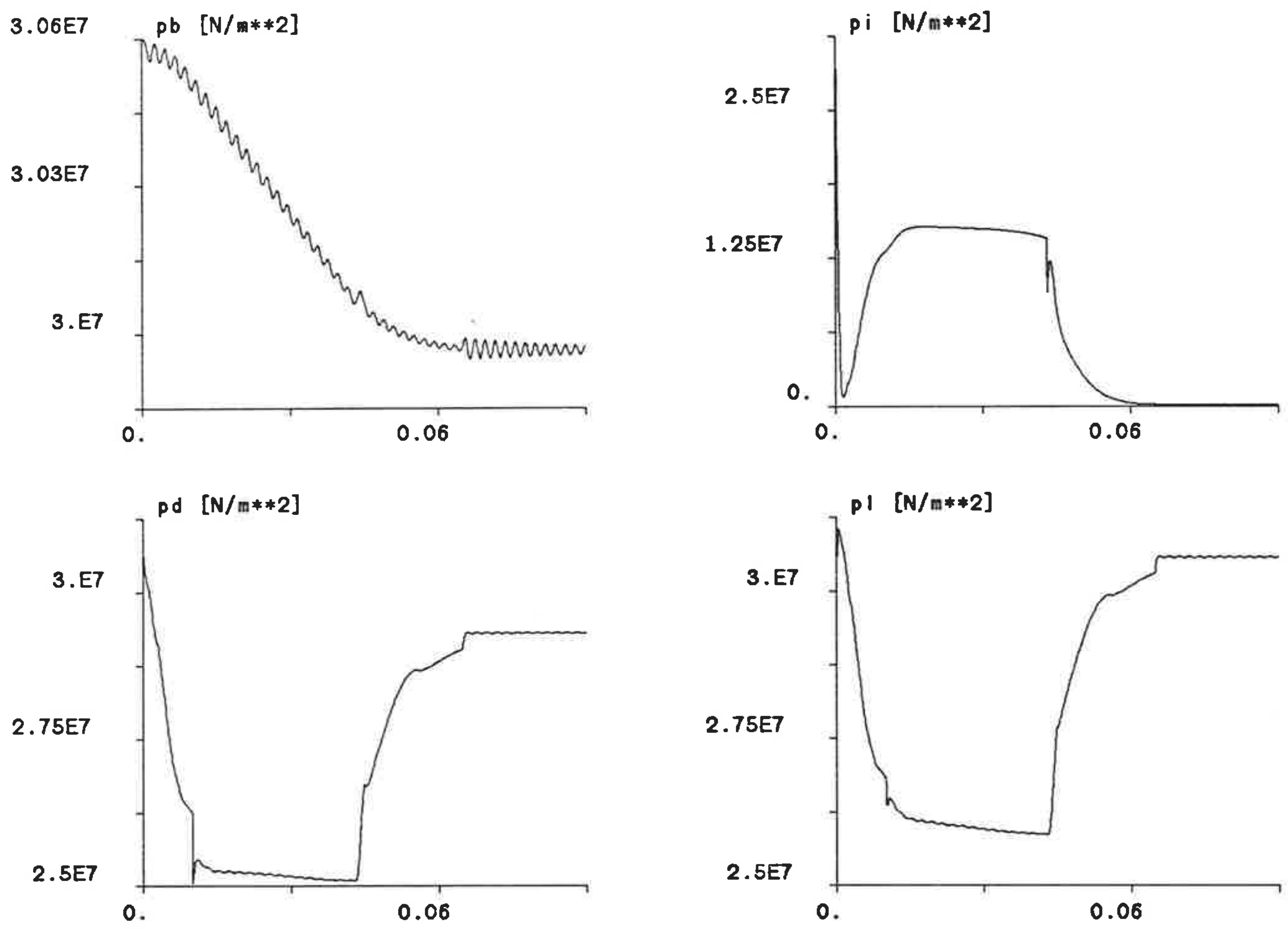


85.02.21 - 18:41:22 nr: 2
hcopy "Close-cycle; algor.: RK; error: 0.001



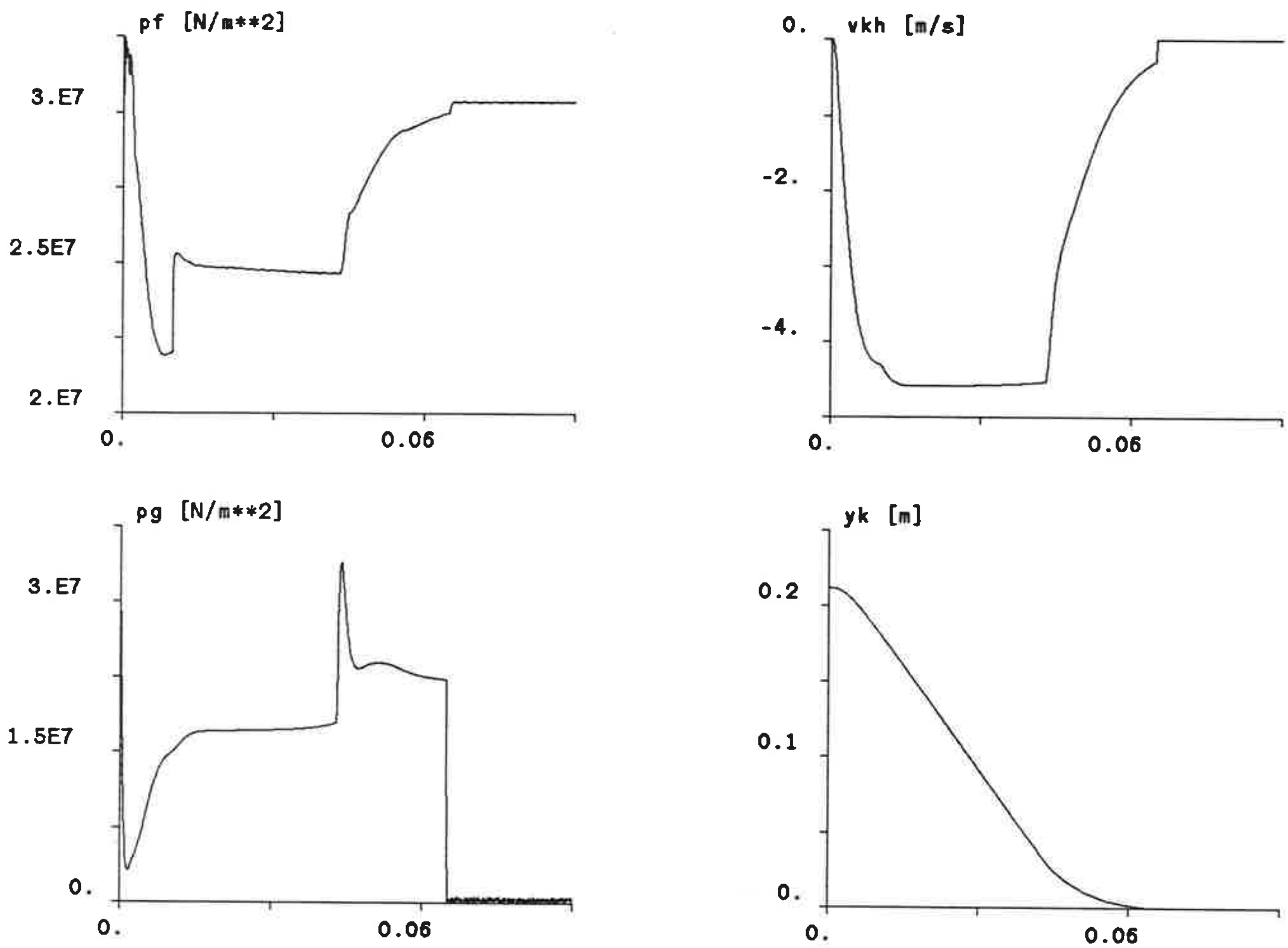
APPENDIX D: RESULTS WITH ALGORITHM RUNGE-KUTTA FIXED STEP SIZE

85.02.19 - 14:13:04 nr: 1
hcopy "Open-cycle; algor.: RKFIX; increm.: 0.00001

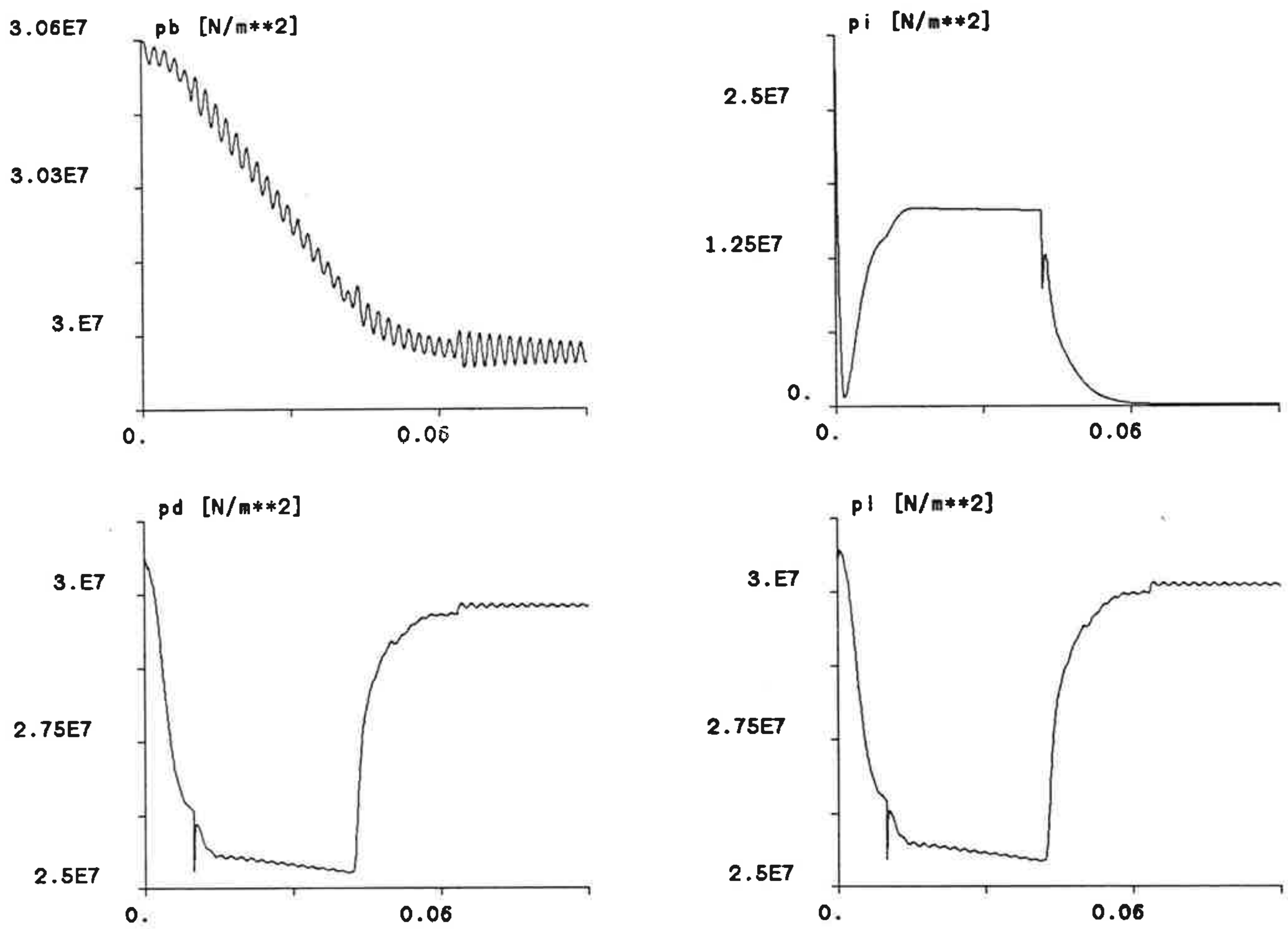


D.2

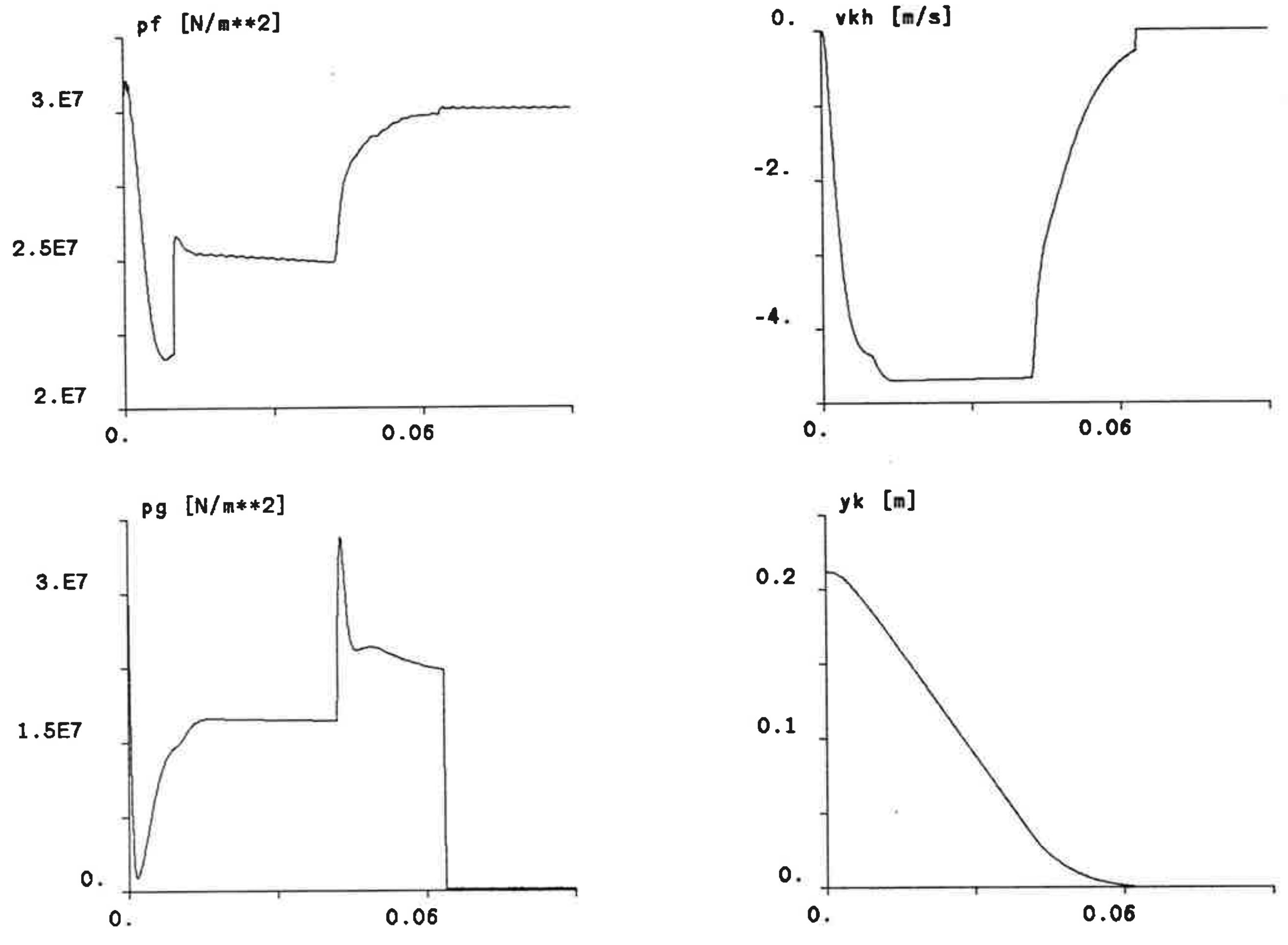
85.02.19 - 14:15:41 nr: 2
hcopy "Open-cycle; algor.: RKFIX; increm.: 0.00001



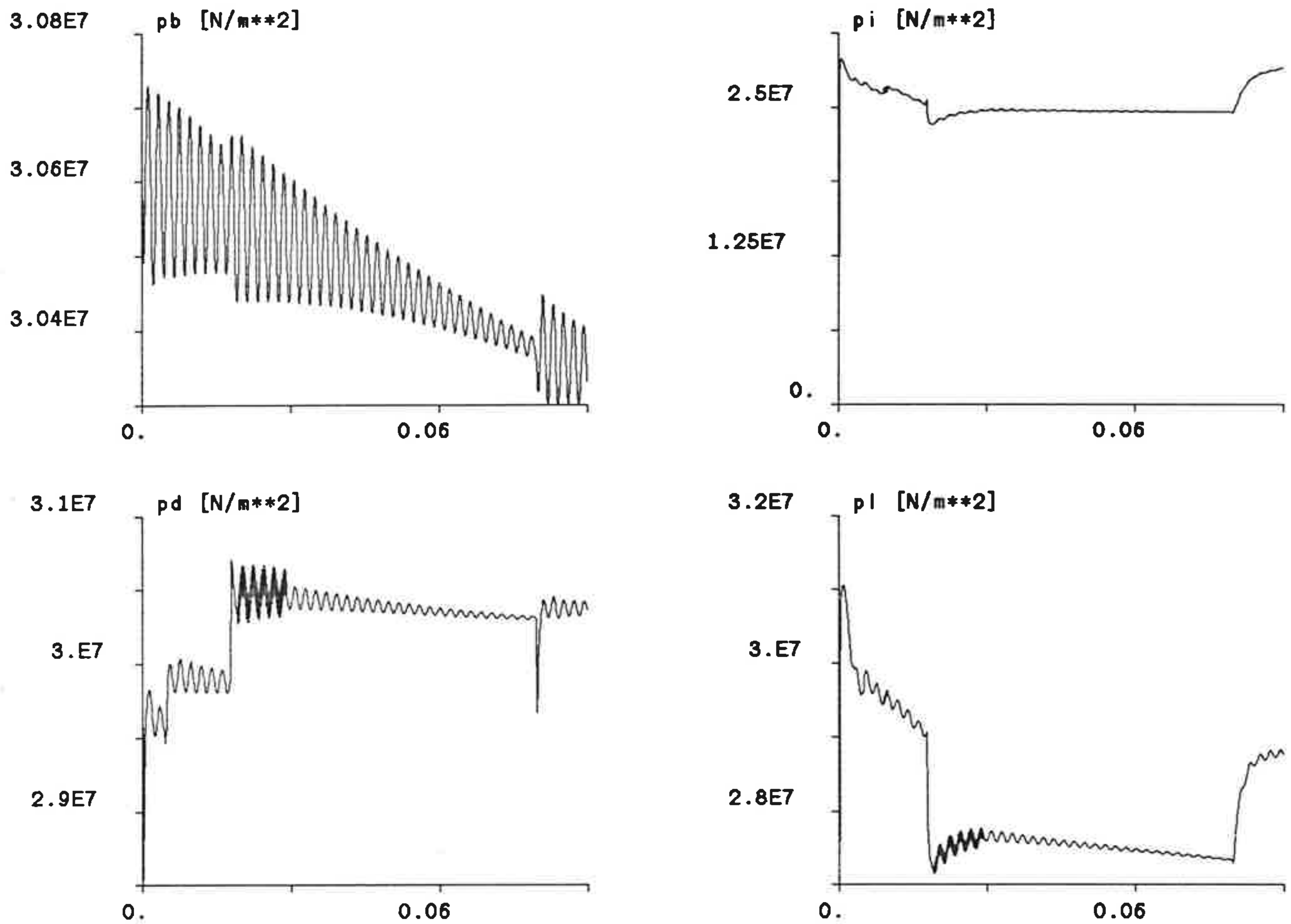
85.02.19 - 14:29:38 nr: 4
hcopy "Open-cycle; algor.: RKFIX; increm.: 0.000005



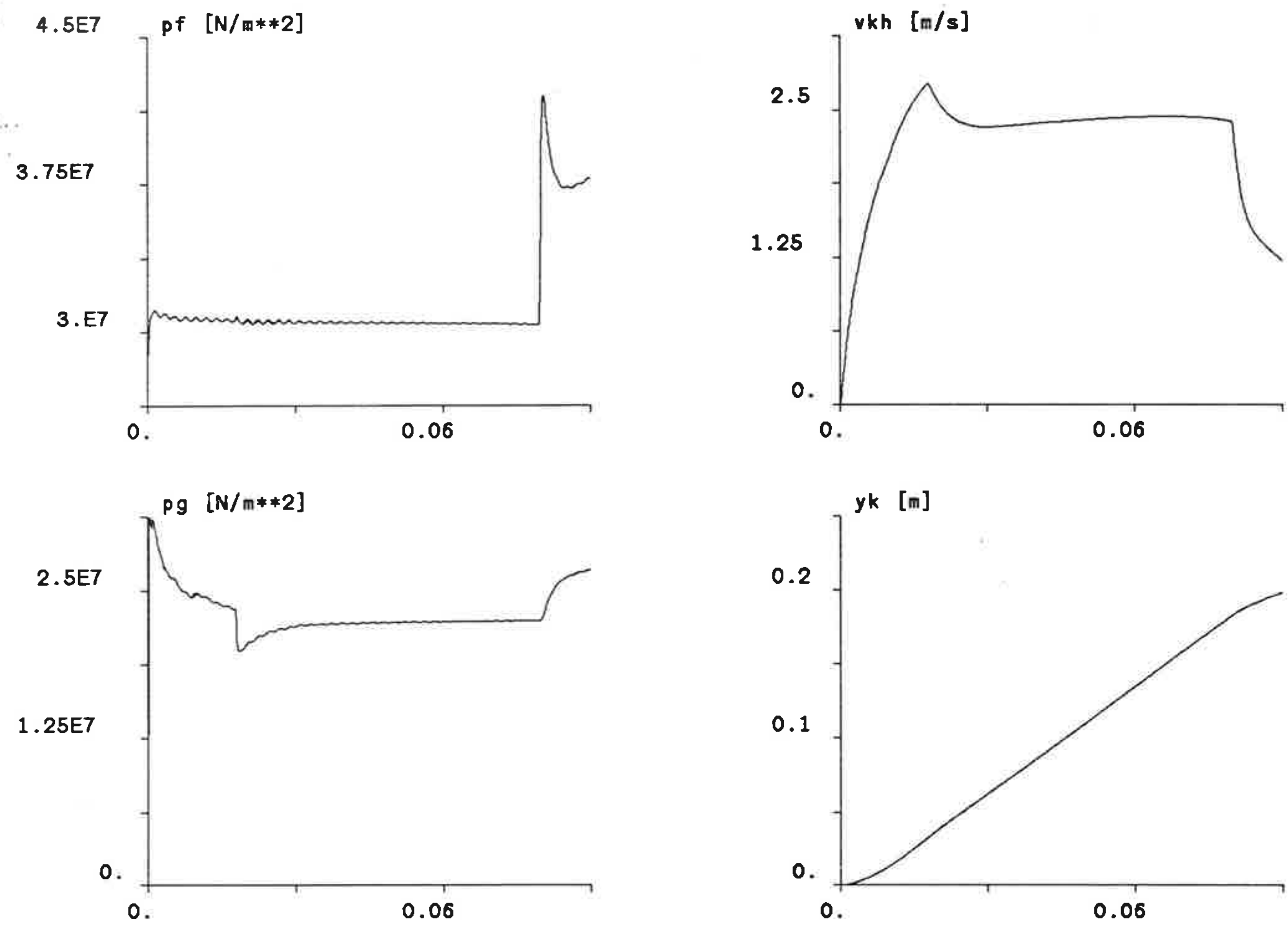
85.02.19 - 14:31:44 nr: 5
hcopy "Open-cycle; algor.: RKFIX; increm.: 0.000005



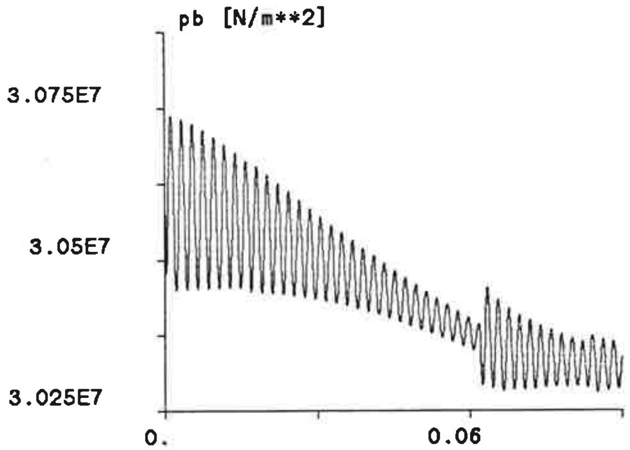
85.02.21 - 18:51:40 nr: 6
hcopy "Close-cycle; algor.: RKFIX; increm.: 0.00001

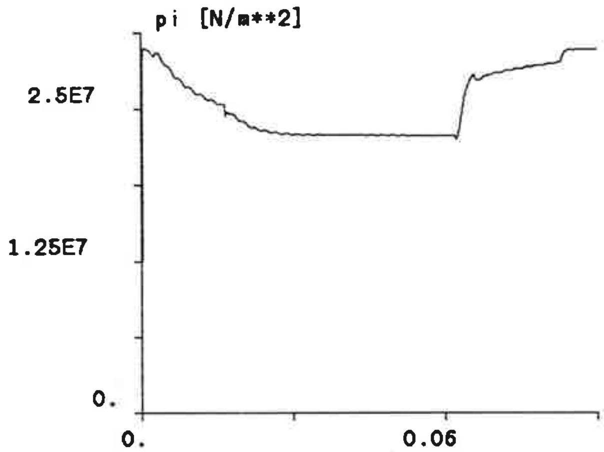


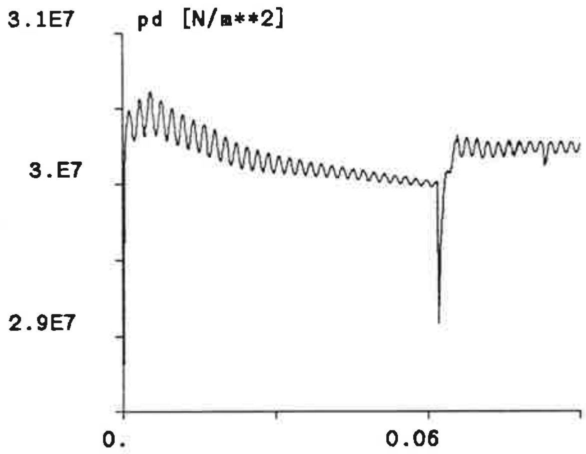
85.02.21 - 18:53:30 nr: 7
hcopy "Close-cycle; algor.: RKFIX; increm.: 0.00001

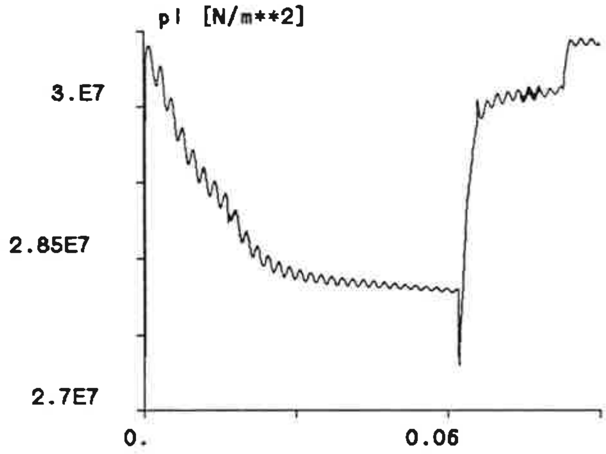


85.02.21 - 19:02:09 nr: 9
hcopy "Close-cycle; algor.: RKFIX; increm.: 0.000005

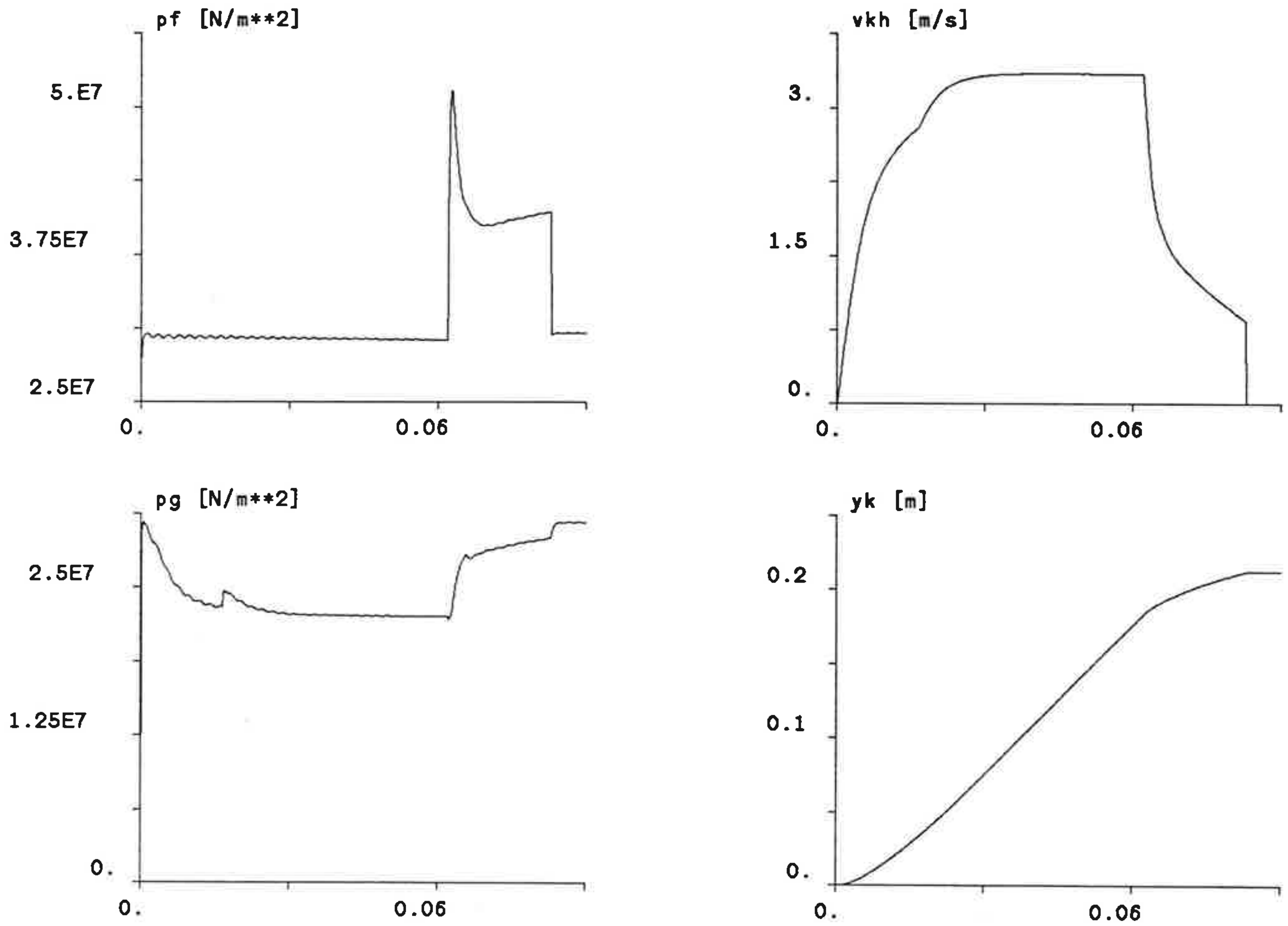






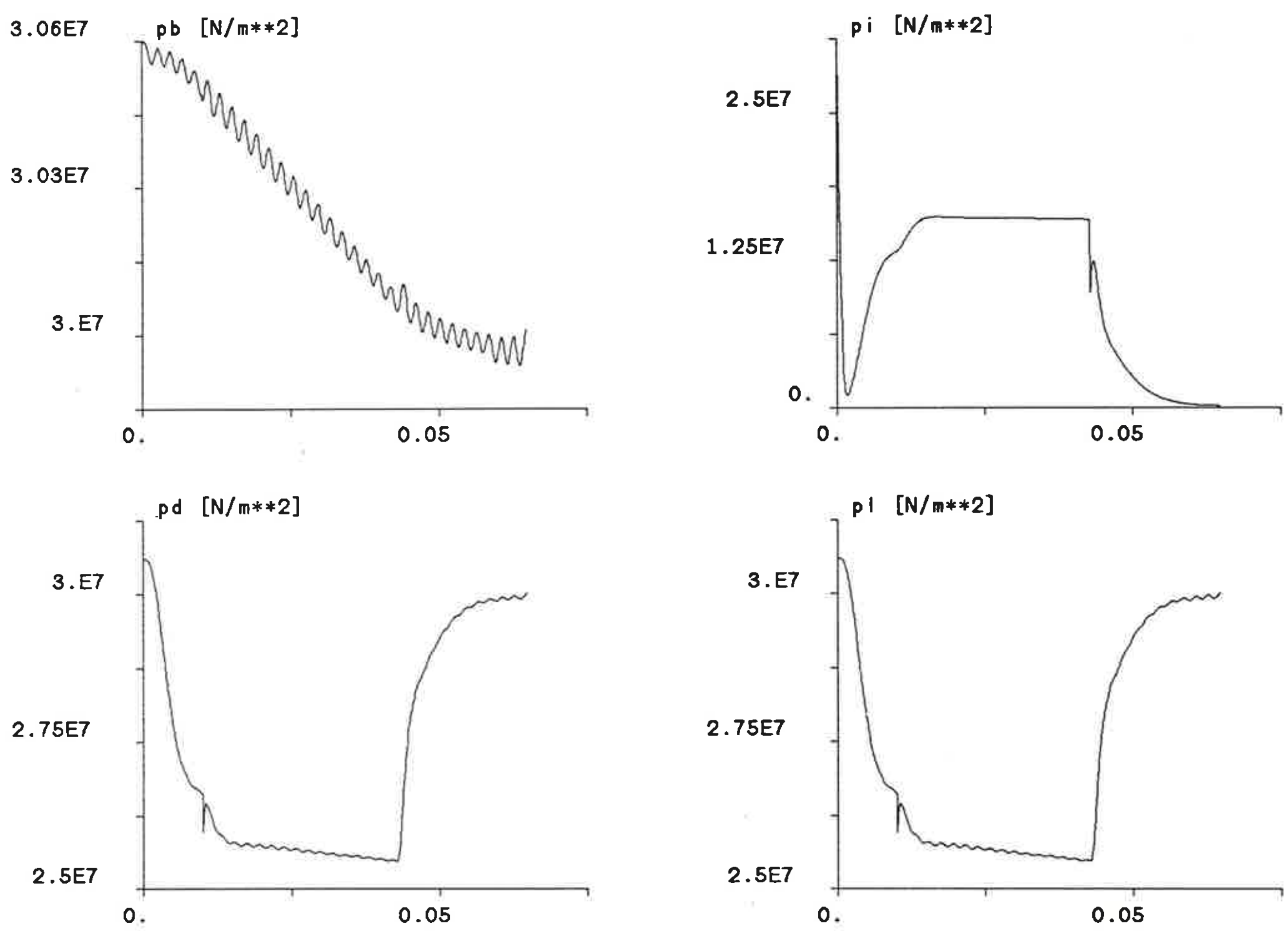


85.02.21 - 19:03:50 nr: 10
hcopy "Close-cycle; algor.: RKFIX; increm.: 0.000005



APPENDIX E: RESULTS WITH ALGORITHM FOR STIFF SYSTEMS

85.02.20 - 09:27:43 nr: 4
hcopy "Open-cycle; algor.: DAS; error: 0.001



85.02.20 - 09:17:47 nr: 2
hcopy "Open-cycle; algor.: DAS; error: 0.001