# Laboratorijske vaje pri predmetu Računalniško podprto konstruiranje

Leon Kos, Laboratorij za CAD - LECAD

## Naloga

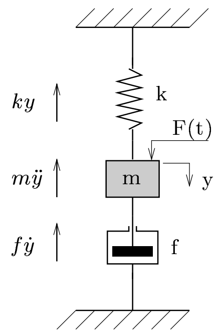
Izdelajte računalniški program za prostorsko simulacijo mehanskega sistema sestavljenega iz mas, dušilk in vzmeti ter program za prikaz odziva z uporabo grafičnega standarda GKS (level 1a). Za simulator uporabite teorijo prostora stanj v pomičnih lokalnih koordinatnih sistemih masih točk z virtualnimi pomiki.

Programa morata biti ločeni zaključeni celoti (simulator in animator), ki si posredujeta potrebne podatke po datoteki podatkov, ki jo morate izvesti z neformatirano (binarno) direktno datoteko. Za vsak program izdelajte poročilo z opisom delovanja programa, omejitvami, teoretičnimi osnovami in rezultati izračuna.

## 1 Prostor stanj

Osnovna ideja zapisa v prostoru stanj je, da zapišemo dinamično obnašanje sistema z nizom diferencialnih enačb prvega reda, katere lahko enostavno integriramo ter tako dobimo odziv sistema. S to metodo je možno analizirati linearne ali nelinearne multivariabilne sisteme s časovno spremenljivim vzbujanjem. Metoda je splošna in v uporabi na različnih področjih kot je elektrotehnika, ekonomija, biologija...

Oglejmo si enodimenzionalni linearni mehanski sistem mase, dušilke in vzmeti na sliki 1.




Slika 1: Mehanski sistem

Za maso lahko napišemo ravnotežno enačbo

$$m\ddot{y} = -ky - f\dot{y} + F(t)$$

ali

$$m\ddot{y} + ky + f\dot{y} = F(t) \tag{1}$$

Iz enačbe (1) moramo napisati sistem diferencialnih enačb prvega reda, zato uvedemo substitucijo. Za spremenljivki stanj izberemo

$$\begin{aligned} x_1(t) &= y(t) \\ x_2(t) &= \dot{y}(t) \end{aligned} \tag{2}$$

Enačbe v prostoru stanj je potrebno napisati tako, da so na levi strani enačbe prvi odvodi spremenljivk stanj, na desni pa ostali členi izraženi s spremenljivkami stanj. Enačba (1) je tako

$$\begin{aligned} \dot{x}_1 &= \dot{y} = x_2 \\ \dot{x}_2 &= \ddot{y} = \frac{1}{m}(-ky - f\dot{y} + F(t)) \end{aligned} \tag{3}$$

Enačbo (3) je potrebno v celoti zapisati s spremenljivkami stanj in vhodnimi signali:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \ddot{y} = -\frac{k}{m}x_1 - \frac{f}{m}x_2 + \frac{1}{m}F(t) \end{aligned} \tag{4}$$

Gornji enačbi v matrični obliki sta

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{f}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t) \tag{5}$$

### 1.1 Diskretizacija časa

Dobljeni sistem dveh linearnih diferencialnih enačb prvega reda (5) enostavno rešimo z diskretizacijo časovne osi. Odvod po definiciji zapišemo kot

$$\dot{x} = \frac{dx}{dt} = \lim_{h\to 0} \frac{x(t+h) - x(t)}{h} \tag{6}$$

Numerično reševanje odvoda zahteva končno majhno časovno razliko

$$h = \Delta t = t_n - t \tag{7}$$

kar nam da v praktičnem reševanju enačbe (6) dovolj točno aproksimacijo odvoda

$$\dot{x} = \frac{dx}{dt} \doteq \frac{x_n - x}{h} \tag{8}$$

V sistemu enačb (5) nadomestimo odvode z aproksimativno enačbo

$$\begin{bmatrix} \frac{x_{1n} - x_1}{h} \\ \frac{x_{2n} - x_2}{h} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{f}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t) \tag{9}$$

Enačbo (9) preuredimo tako, da je na levi strani enačaja le vrednost spremenljivk stanja v naslednjem trenutku $x_n = x(t+h)$

$$\begin{bmatrix} x_{1n} \\ x_{2n} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + h \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{f}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + h \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t) \tag{10}$$

Z izborom utrezno majhnega časovnega koraka lahko gornjo enačbo večkrat uporabimo in tako dobimo odziv mehanskega sistema pri poljubnem vzbujanju.

Spremenljivki stanja (2) predstavljata pomik in hitrost mase. Preden začnemo s simulacijo odziva je potrebno nastaviti začetne velikosti pomika in hitrosti v času

$$x(t = 0) = x(t_0)$$

### 1.2 Splošne enačbe v prostoru stanj

Obravnava sistemov v prostoru stanj obravnava vhodne spremenjivke stanj, izhodne spremenljivke stanj in spremenlivke stanj. Izhodne spremenljivke stanj so največkrat enake (notranjim) spremenljivkam stanj. Standardni zapis v prostoru stanj ima obliko

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad (11)$$

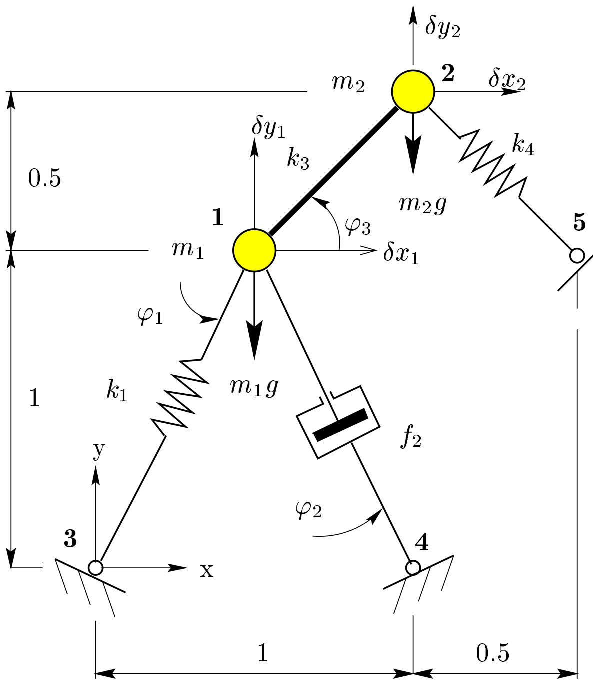
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \quad (12)$$

kjer je $\mathbf{u}$ vzbujanje sistema (vhodne s.s.), $\mathbf{y}$ pa odziv sistema (izhodne s.s.). Mastno pisane spremenljivke pomeni matriko ali vektor.

Naše obravnavanje prostora stanj je tako, da ni potrebe po enačbi (12), saj nam vektor $x$ direktno predstavlja pomike mas v lokalnem koordinatnem sistemu.

## 2 Virtualni pomiki

Običajno vsaki masi za katero napišemo ravnotežne enačbe določimo lokalni koordinatni sistem z začetni legi. Oglejmo si primer dvodimenzionalnega translatornega sistema na sliki 2.




Slika 2: Dvodimenzionalni mehanski sistem

Med masama je mišljena toga povezava, ki pa jo lahko nadomestimo z dovolj veliko togostjo vzmeti $k_3$ in si s tem prihranimo dodatne analogije v modeliranju sistema.

Tu za razliko od prejšnjega primera upoštevamo lastno težo masnih točk, ki se v prostoru stanj obravnava kot zunanje vzbujanje.

Prvi ravnotežni enačbi v lokalnem kartezijevem koordinatnem sistemu prve mase sta

$$m_1\ddot{u}_1 = k_1\Delta l_1 \cos\varphi_1 + f_2\Delta\dot{l}_2 \cos\varphi_2 + +k_3\Delta l_3 \cos\varphi_3 + m_1 g \cos\varphi_4 \quad (13)$$

$$m_1\ddot{v}_1 = k_1\Delta l_1 \sin\varphi_1 + f_2\Delta\dot{l}_2 \sin\varphi_2 + +k_3\Delta l_3 \sin\varphi_3 + m_1 g \sin\varphi_4 \quad (14)$$

S črko $\Delta l$ smo označili posamezne raztege vzmeti ali dušilke. Za prvo vzmet je ta razteg enak

$$\Delta l_1 = \sqrt{(x_1+\Delta x_1)^2 + (y_1+\Delta y_1)^2} - \sqrt{x_1^2+y_1^2} \quad (15)$$

Vidimo, da razteg posamezne vzmeti ali dušilke s takim izborom koordinatnega sistema ni več enostavna linearna funkcija, ki jo potrebujemo za zapis v prostoru stanj. Tudi drugačni koordinatni sistemi ne odpravijo te težave, zato je potrebno uvesti dodatne poenostavitve.

Če so pomiki dovolj majhni lahko pomik aproksimiramo s totalnim diferencialom. Kaj pomeni to: ,,dovolj majhni pomiki"? Diferencialno majhni pomiki so v primerjavi z dolžimi vzmeti ali dušilk zanemarljivi. V mehaniki pravimo takim pomikom *virtualni pomiki.* Po definiciji morajo biti taki pomiki majhni in seveda možni. Za globalni kartezijev koordinatni sistem v treh dimenzijah je virtualni pomik

$$\delta l = \frac{\partial l(x,y,z)}{\partial x}\,\delta x + \frac{\partial l(x,y,z)}{\partial y}\,\delta y + \frac{\partial l(x,y,z)}{\partial z}\,\delta z \quad (16)$$

ali v splošnih koordinatah

$$\delta l = \sum_i \frac{\partial l(q_1,q_2,...,q_n)}{\partial q_i}\delta q_i \quad (17)$$

Realne pomike torej nadomestimo z virtualnimi. Da ločimo med pomiki jih dodatno označimo s črko $\delta$. Enačbo (15) tako napišemo kot

$$\delta l_1 = \frac{x_1}{\sqrt{x_1^2+y_1^2}}\,\delta x_1 + \frac{y_1}{\sqrt{x_1^2+y_1^2}}\,\delta y_1 \quad (18)$$

Dolžina tretje vzmeti je enaka

$$l_3 = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2} \quad (19)$$

Virtualni pomik te vzmeti je torej odvisen od štirih virtuanih pomikov lokalnih koordinatnih sistemov masne točke ena in dve

$$\delta l_3 = \frac{x_2-x_1}{l_3}\delta x_2 + \frac{y_2-y_1}{l_3}\delta y_2 - \frac{x_2-x_1}{l_3}\delta x_1 - \frac{y_2-y_1}{l_3}\delta y_1 \quad (20)$$

### 2.1 Smerni kosinusi

Poglejmo podrobneje enačbo (20). Pred vsakim virtualnim pomikom so si členi zelo podobni. Iz trigonometrije vemo, da so to smerni kosinusi

$$cos\varphi_3 = \frac{x_2-x_1}{l_3} \qquad cos\varphi_3' = \frac{y_2-y_1}{l_3} \quad (21)$$

kar je razmerje dolžine v smeri željene koordinatne smeri in celotne dolžine daljice. V treh dimenzijah so smerni kosinusi podobni le da se dolžina povezave izračuna še z upoštevanjem koordinate $z$.

$$l_{21} = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2 + (z_2-z_1)^2} \quad (22)$$

Zaradi krajšega pisanja označimo smerne kosinuse kot

$$x_{21} = cos\varphi_3 = \frac{x_2-x_1}{l_{21}} = -x_{12} \quad (23)$$

$$y_{21} = cos\varphi_3' = \frac{y_2-y_1}{l_{21}} = -y_{12} \quad (24)$$

Ravnotežni enačbi prve mase (14) in (15) lahko zapišemo linearizirano

$$
\begin{aligned}
m_1\delta\ddot{x}_1 &= k_1(-x_{31}\delta x_1 - y_{31}\delta y_1)x_{31} + \\
&+ f_2(-x_{41}\delta\dot{x}_1 - y_{41}\delta\dot{y}_1)x_{41} + \\
&+ k_3(x_{21}\delta x_2 + y_{21}\delta y_2 - x_{21}\delta x_1 - y_{21}\delta y_1)x_{21} \\
&+ F_{1x} \qquad (25) \\
m_1\delta\ddot{y}_1 &= k_1(-x_{31}\delta x_1 - y_{31}\delta y_1)y_{31} + \\
&+ f_2(-x_{41}\delta\dot{x}_1 - y_{41}\delta\dot{y}_1)y_{41} + \\
&+ k_3(x_{21}\delta x_2 + y_{21}\delta y_2 - x_{21}\delta x_1 - y_{21}\delta y_1)y_{21} \\
&+ F_{1y} \qquad (26)
\end{aligned}
$$

S primerno organizacijo lahko sistematično napišemo enačbe stanja. Za primer na sliki 2 je potrebno napisati osem enačb stanja. To je pomik in hitrost v obeh dimenzijah (x,y) za obe masni točki. Substitucije spremenljivk so podobne kot v enodimenzionalnem translatornem problemu. Izberimo take substitucije, da bodo v prvih štirih enačbah v desni diagonali same enice. Tako bodo pri sukcesivnem reševanju diferencialnih enačb prve štiri enačbe predstavljale pomike mas, naslednje štiri pa hitrosti.

Matrika $\mathbf{A}\delta\mathbf{q}$ prostora stanj brez upoštevanje mas (11) izgleda v takole

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
s_{11} & s_{12} & s_{13} & s_{14} & b_{11} & b_{12} & b_{13} & b_{14} \\
s_{21} & s_{22} & s_{23} & s_{24} & b_{21} & b_{22} & b_{23} & b_{24} \\
s_{31} & s_{32} & s_{33} & s_{34} & b_{31} & b_{32} & b_{33} & b_{34} \\
s_{41} & s_{42} & s_{43} & s_{44} & b_{41} & b_{42} & b_{43} & b_{44}
\end{bmatrix}
\begin{bmatrix}
\delta x_1 \\ \delta y_1 \\ \delta x_2 \\ \delta y_2 \\ \delta\dot{x}_1 \\ \delta\dot{y}_1 \\ \delta\dot{x}_2 \\ \delta\dot{y}_2
\end{bmatrix}
$$

Napišimo člene podmatrik vzmeti in dušilk in uvedimo dodatne formalizme, tako da označimo konstante vzmeti in dušilk z indeksi vozlišč med katerimi se povezava nahaja. Primer: $f_2 = b_{41} = b_{14}$

$$
\begin{aligned}
s_{11} &= - k_{31}x_{31}x_{31} - k_{21}x_{21}x_{21} \\
s_{12} &= - k_{31}x_{31}y_{31} - k_{21}x_{21}y_{21} \\
s_{13} &= k_{21}x_{21}x_{21} \\
s_{14} &= k_{21}x_{21}y_{21} \\
s_{21} &= - k_{31}x_{31}y_{31} - k_{21}x_{21}y_{21} \\
s_{22} &= - k_{31}y_{31}y_{31} - k_{21}y_{21}y_{21} \\
s_{23} &= k_{21}y_{21}x_{21} \\
s_{24} &= k_{21}y_{21}y_{21} \\
b_{11} &= - f_{41}x_{41}x_{41} \\
b_{12} &= - f_{41}x_{41}y_{41} \\
b_{13} &= 0 \\
b_{14} &= 0 \\
b_{21} &= - f_{41}y_{41}x_{41} \\
b_{22} &= - f_{41}y_{41}y_{41} \\
b_{23} &= 0 \\
b_{24} &= 0
\end{aligned}
$$

Če podmatrike zapišemo tako , da združimo vse koordinate posamezne mase vidimo, da so diagonalni elementi negativna vsota vsek povezav z obravnavano masno točko, izvendiagonalni elementi pa predstavljajo povezave med posameznimi masami. Podmatrike so zaradi enakosti medsebojnih vplivov simetrične.

Algoritem gradnje matrike je lahko tak, da za vsako povezavo med dvema vozloma naredimo:

- Preverimo, če je tip povezave dušilka, potem je gradnja podmatrike enaka vzmetem, le da se členi pišejo v desno stran matrike **A**
- Prvi člen povezave pove, za ketero maso je potrebno dopisati diagonalne člene podmatrike
- Člene je potrebno napisati za vse koordinate. Členi podvrstice se množijo s smernim kosimusom povezave v smeri vrstice
- Za vse podkolone je potrebno napisati člen, ki se množi s smernim kosinusom povezave v smeri kolone.
- Izvendiagonalni členi so tisti, za katere je drugi člen povezave manjši ali enak številu mas. Ta člen tudi pove, za katere maso je potrebno napisati povezavo. Tu smo predpostavili, da bomo vozlišča številčili tako, da bomo najprej označili mase, šele nato statična vozlišča!
- Če je drugi člen v povezavi tudi vozel, ki je gibljiv (masa) je potrebno dopisati diagonalne člene še za to maso.

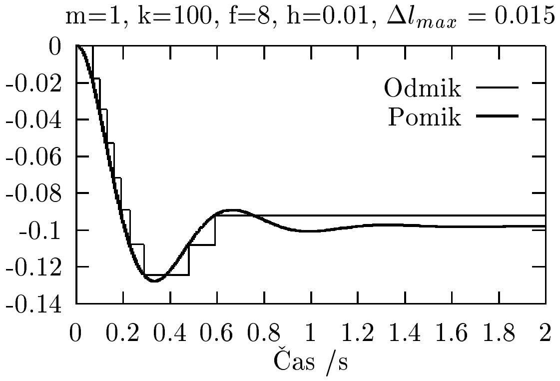
## 2.2 Pomični lokalni koordinatni sistemi

Z uvedbo virtualnih pomikov smo predpostavili, da bodo pomiki oz začetne lege majhni in s tem je napaka tudi zanemarljivo majhna. Pogled na sliko 2 nam da vedeti, da pomiki pri simulaciji nikakor ne bodo majhni, saj sistem izgleda kot elastični mehanizem. Linearizacijsko napako je možno odpraviti tako, da se lokalni koordinatni sistem pomika skupaj z maso. Pri vsakem premiku v nov lokalnei koordinatni sistem masne točke je potrebno popraviti začetne pogoje vzbujanja. Tako uvedemo v v enačbe prostora stanj začetna vzbujanja, ki so posledica prednapetih vzmeti.

S tem, ko uvedemo korekcijo linearizacijske napake, pa ne smemo zanemariti numeričnih napak, ki nastopajo pri seštevanju velikih števil z malimi. Če to ne upoštevamo, se lako pojavi tipična zaokrožitvena napaka, ki v skrajnem primeru vodi tudi k nestabilnim izračunom. Zato bomo korigirali pozicijo masne točke le takrat, ko bo podmik katerekoli masne točke večji kot neka predpisana vrednost $\Delta l_{max}$

Na sliki 3 vidimo, kako za primer na sliki 1 premikamo koordinatni sistem skupaj z maso. Ker je sistem enodimenzionalen ni napake zaradi velikega odmika od začetne lege. Tako mora biti odziv s pomikanjem enak odzivu brez pomikanja lokalnega koordinatnega sistema.

Za vsako masno točko moramo kontrolirati odmike od lokalnega koordinatnega sistema. Vsakič ko kašksen pomik preseže dovoljeno vrednost ponovno izračunati matriko stanj z novimi smernimi kosinusi povezav med vozlišči. Potrebno je tudi pomniti začetno pozicijo pomičnih vozlišč, ker moramo izračunati še prednapetje vzmeti, ki ji vstavimo ob prestavitivi lokalnega koordinatnega sistema v novo lego, kot vzbujanje s silo prednapetja v vektor $u$.




Slika 3: Pomični lokalni koordinatni sistem

Nove vektorje vzbujanja $u$ ob korekciji izračunamo kot

$$u_i = u_{0i} + \sum \frac{k_i}{m_i}(l_i - l_{0i}) sc_i \tag{27}$$

kjer je $sc_i$ smerni kosinus za ustrezno smer, $l_i - i_{0i}$ sprememba dolžine vzmeti. Indeks 0 pomeni začetno stanje sistema. Pozameznim vbujanjem je potrebno torej dodati silo prednapetja vsake vzmeti, ki vpliva na maso v ustrezni smeri.

Omenjen pristop nam tako omogoča analizo mehanizmov in večino translatornih sistemov.

# 3 Predprocesor

Predprocesor je del simulatorja, ki bere vhodno datoteko z opisom sistema in modelira problem v obliki primerni za numerično reševanje. Ob branju vhodnih podatkov tako gradi tudi potrebne podatke in postavlja na ustrezno mesto, katere kasneje simulator uporablja pri simulaciji. Vhodna datoteka za primer na sliki 2 naj ima sledeč format:

```
simulate Primer 2
plane     ; ravninsiki ali 3D (space) problem
constants ; h, t1, t2, deltal, gx, gy, gz
0.0005 0.0 15.0 0.005 0.0 -9.81 0.0

nodes 5     ; stevilo vozlisc
1  0.5  1.0 ; najprej masne tocke
2  1.0  1.5
3  0.0  0.0
4  1.0  0.0
5  1.5  1.0

masses 2   ; mase vozlisc
1 1.0
2 0.8

connections 4 ; st. povezav
k 1           ; tip povezave in st.
1 3 500.0     ; vozla in lastnost
f 2
1 4 100.0
k 3
1 2 5000.0
k 4
2 5 500.0

solve         ; konec obdelave
```

Togo povezavo med masama smo modelirali z togo vzmetjo. V praksi to največkrat ni dovolj, saj je vsaka vzmet v realnem tudi dušilka. Tako se tudi za simulacijo priporoča vgradnja dušilke vzporedno z vzmetjo, saj to prispeva k realnosti s tem tudi povečuje stabilnost simulacije. Splošno pravilo je, da mora biti $h$ vsaj deset krat manjši od nihajne dobe najbolj toge vzmeti, če nimamo paralelno vezano dušilko. V nasprotnem primeru se lako zgodi, da je simulacija nestabilna (beri kaotična).

Matrika stanj $\mathbf{A}$ že deljena z masami, za podano vhodno datoteko ima v začetnem stanju sledeče vrednosti

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -2600 & -2700 & 2500 & 2500 & -20 & 40 & 0 & 0 \\ -2700 & -2900 & 2500 & 2500 & 40 & -80 & 0 & 0 \\ 3125 & 3125 & -3438 & -2813 & 0 & 0 & 0 & 0 \\ 3125 & 3125 & -2813 & -3438 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dodatno vzbujanje v sistem bomo uvedli s harmoničnimi silami, ki delujejo na masne točke. Samo vzbujanje se razlikuje od gravitacjiske sile le v tem, da je to časovno odvisna funkcija

$$F_i(t) = A_i \sin(\omega t + \theta_i) \quad , \qquad i = \{x, y, z\} \tag{28}$$

V predprocesorju bomo vzbujalne sile modelirali z

```
perturbation 1       ; stevilo sil vzbujanja
1 12.5  25.0 0.0  25.0 1.57  0.0 0.0
```

kjer števila v drugi vrstici pomenijo indeks mase, $\omega$, $A_x$, $\theta_x$, $A_y$, $\theta_y$, $A_z$, $\theta_z$. Uporabljeni način vzbujanja v praksi ni najbolj realen, saj bi bilo bolje, če bi za vzbujanje uvedli pomično vozlišče. Zaradi enostavnosti izdelave programa bomo uporabili le harmonično silo s katero pa je ravno tako možno ugotoviti frekvenčno odvisnost sistema in s tem tudi kritične frekvence.

# 4 Prikaz rezultatov

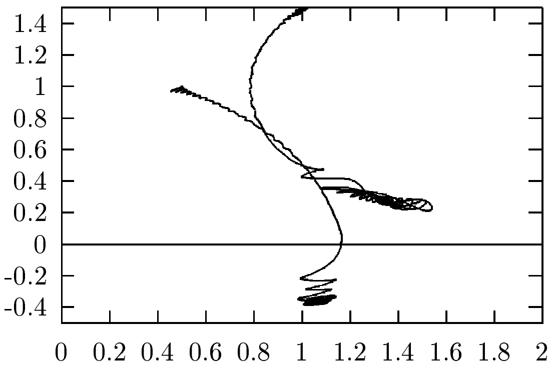
Za prikaz in kontrolo rezultatov uporabite program za risanje grafov *gnuplot*. Program ima mnoge možnosti izrisa podatkov na zaslon kot tudi na mnoge druge izhodne naprave. Uporaba je dokaj enostavna saj program sam poišče obseg podatkov in jih tudi izriše. Datoteka podatkov je lahko sestavljena iz več kolon, vendar je potrebno pri ukazu za izris podati tudi kateri dve koloni se uporabljati za izris grafa.

Primer uporabe za izris gibanja ene mase na sliki 4 s podatki v drugi in tretji koloni je sledeč:

```
gnuplot> plot "fig4.dat" using 2:3  with lines
gnuplot> help plot
gnuplot> plot [0:2] [-0.5:1.5] "fig4.dat" u 2:3
```

Program za simulacijo mora torej izpisati vse pomike za vse masne točke. S programom za prikaz pa lahko pravilnost delovanja prekontroliramo. Najenostavneje je, če nam program izpisuje rezultate kar na zaslon z ukazom `print *, ...`. Izpis na zaslon pa lahko preusmerimo v datoteko z ukazom >, ki naredi redirekcijo standardnega izhoda v datoteko. Primer:

```
f:\home\lecad01\mechsim > fig4.dat
```




Slika 4: Gibanje masnih točk v dveh dimenzijah

### 4.1 Animacija

Za grafičen prikaz rezultatov v programu uporabite GKS [2] grafični jezik. V ta namen uporabite homogene matrike in homogene transformacije. Ker bi risanje vzmeti in dušilk v treh dimenzijah vzelo preveč časa (našega in računalnikovega), bomo vzmeti risali z rdeču barvo, dušilke pa z zeleno.

Program za animacijo mora brati direktno datoteko podatkov in sproti prikazovati gibanje sistema. S tem vnesemo dodatne zakasnitve v program. Animator mora izpisovati trenutni čas, da lahko ocenimo lastna nihanja sistema.

Na zahtevo mora program izpisati tudi vozlišča in povezave (npr. rumeno). Omogočati mora tudi ogled sistema s poljubnega zornega kota (rotacija, translacija in skaliranje v homogenih koordinatah).

Osnovni program, ki demonstrira uporabo jezika GKS je:

```
      program GKS skeleton
      implicit none
      real x(3)  /-200.0, 350.0, 250.0/
      real y(3)  /-200.0, 300.0,-250.0/
      real kx(2) /0.0, 100.0/
      real ky(2) /50.0, 120.0/
      character*80 datrec(1)

c     Odpre GKS (Gks OPen gKS)
      call gopks()

c     Nastavitev zaslonskih koordinat
c     (Gks Set WorKstation ViewPort)
      call gswkvp(1, 20.0, 50.0, 600.0, 450.0)

c     Nastavitev uporabniskega koord. sistema
c     (Gks Set WiNdow)
      call gswn(1, -300.0, -300.0, 400.0, 400.0)

c     Izris lika z dvema crtama
c     (GksSetPolyLineColorIndex, GksPolyLine)
c     Indeksi:1 modra, 2 zelena, 3 cyan, 4 rdeca,
c             5 violicna, 6 rjava, 7 bela, 8 siva,
c             9 svetlomodra, ..., 14 rumena, 15
      call gsplci(3)
      call gpl(3, x, y)

c     Izris teksta (GksSetCharacterHeight,
c     ...TextColorIndex, GksTeXt)
      call gstxci(14)
      call gschh(20)
      call gtx(-250.0, 300.0, 'GKS skeleton')

c     Izris kroga
c     (Generalized Drawing Primitive)
      call gdp(2, kx, ky, -1, 1, datrec)

c     Zapre GKS (Gks CLose gKS)
      call gclks()
      end
```

Animacijo sistema izvedemo tako, da po prebranih podatkih za določen čas izrišemo na zaslon sistem, preberemo nove podatke, zbrišemo staro stanje tako, da rišemo s črnim peresom, ter narišemo novo stanje. Postopek seveda ponavljamo dokler ga lahko.

## 5 Poročilo

Poročilo se oddaja v dveh delih, enega za prvi program, drugega na koncu. V poročilu opišite problem, ki ga rešujete (zahteve), podajte teoretične osnove in bistvene dele rešitve, ki ste jih uporabili v programu.

Izpisa programa ni potrebno prilagati poročilu, potrebno pa je predstaviti pomembne dele programa na način, ki čimbolj jasno predstavlja rešitev. To je lahko z besednim opisom, diagramom poteka ali z delom kode v fortranu. Priložite tudi rezultate izračuna sistemov, ki ste jih izračunali.

Celotno poročilo naj ima klasično obliko (uvod, zahteve, teoretične osnove, problemi, rezultati, možne izboljšave, zaključek, literatura, priloge). Izberite si primer mehanskega in na njem pokažite reševanje problema. Diskutirajte izboljšave programa (*Runge-Kutta*), kot tudi omejitve, ki so posledica vgrajenih algoritmov (stabilnost, ...). Torej: poročilo mora biti napisano tako, da je možno iz njega rekonstrurirati program.

Pišite čitljivo; priporočen je izpis s tiskalnikom. Ovojnico za poročilo dobite pri vratarju (Računalniško podprto konstruiranje).

## 6 Delo na računalniku

Na računalniku PC je instalirano programsko okolje s fortranskim prevajalnikom po standardu F77. Okolje je integrirano in se imenuje *Programmers WorkBench*. Integrirano okolje z požene z ukazom pwb ime.for v katerem lahko program popravljamo, prevajamo, zaganjamo in iščemo napake. Osnovni ukazi so dosegljiv s pritiskom na tipko skupaj s tipko **ALT**. Tako je izhod iz **PWB ALT+F,X**, Prevajanje in zaganjanje programa z ukazom **Alt+R,E,B,R**. Za pomoč lahko uporabite tipko **F1**, ki je kontekstno občutljiva.

## 7 Fortran

Tu bom prikazal standardne posebnosti v Fortranu, ki jih boste s pridom uporabili pri programiranju. Priporoča se tipkanje programa z malimi črkami, saj je tako kodiran program bolj čitljiv.

Če želimo interpretirati (parsing) vhodne podatke, ki imajo (na)ključne besede skupaj s številčnim parametrom je potrebno najprej prebrati celotno vrstico kot niz znakov in ga nato interpretirati glede na vrsto ključne besede. Naslednji primer programa prebere vhodno datoteko in izpiše ustrezne podatke, ki jih je interpretiral.

```
      program Prep Demo
      character*80 line
      real node(100,3)
      integer nodes, i, j, k
      open(1, 'primer2.sim', status='old')
10    read(1,'(A)', err=200) line
      if(line(1:5) .eq. 'nodes') then
         read(line(6:80), '(I5)') nodes
         print *, 'Nodes = ', nodes
         do 20, i=1, nodes
           read(1,*) j, (node(j,k), k=1,3)
20         print *, i, j, (node(j,k), k=1,3)
      endif
      if(line(1:5) .eq. 'solve') stop 'OK'
      goto 10
200   stop 'Pricakoval ukaz solve'
      end
```

Navedeni primer vsebuje formatne stavke vključene kar v ukaz read, kar je ugodno, če imamo male formatne stavke. Podobno bi lahko napisali formatni stavek za ukaz read/write:

```
character ime*65
write(*, '('' Ime vhodne datoteke: '',$)')
read(*,'(A)') ime
```

Datoteke delimo na tekstne in binarne datokeke. Direktna datoteka je poseben primer binarne datoteke, ki ima fiksno dolžino zapisa (record). Fortran omogoča branje/pisanje v prostem formatu (*), uporabniškem formatu (format()) in brez formata (samo v binarne datoteke). Neformatirano direktno datoteko z dolžino enega zapisa 58 bytov se odpre z ukazom

```
  open(1, 'results.bin', status='NEW',
* access='DIRECT', form='UNFORMATTED',
* recl=58)
```

Dolžina enega zapisa RECL se določa z maksimalno velikostjo zapisa v datoteki. Zapisi imajo lahko različne vsebine podatkov, dolžina zapisa pa je fiksna zaradi hitrejšega določanja pozicije datotečnega kazalca. Določi se jo s seštevkom dolžine vseh spremenljivk in upoštevanjem, da zavzame realna spremenljivka 4 byte, celoštevilčna 2, znakovna pa toliko, kolikor znakov ima rezerviranih.

Za branje in pisanje enote se uporablja standardni read/write stavek brez podatka o formatu.

```
write(1, rec=2) (x(i), i = 1, n/2)
read(1, rec=2)  (x(i), i = 1, n/2)
```

Pri vseh branjih in pisanjih se lahko testira napaka ob konverziji ali napaka na vhodno/izhodno enoti s parametrom err. Primer:

```
    ...
    read(line(6:80), '(I6)', err=102) nodes
    ...
    return
102 stop 'Napaka pri branju parametra nodes!'
    end
```

Fortran omogoča pisanje in branje v svoje interne enote v obliki `character` polja. To nam omogoča pretvarjanje numeričnih spremenljivk v znakovne. Največkrat se to uporablja za podprograme, ki zahtevajo kot parameter znakovno spremenljivko, na voljo pa imamo le numerično spremenljivko. Tipičen primer uporabe je izpis teksta v jeziku GKS. Primer:

```
character cnode*3
integer node
...
write(cnode, '(I3)') node
call gtx(x, y, cnode)
...
end
```

## 7.1 Spolšna priporočila za delo

Noben program ne deluje na mah. Ker je problem kompleksen se ga je potrebno lotiti sistematično. Posamezne dele program moramo sprotno kontrolirati. Začnemo z majhnim (enodimenzionalnim) sistemom, preverimo delovanje in šele nato dodamo več dimenzij. Pomembno je tudi dokazovanje pravilnosti delovanja, kjer si logično razlagamo obnašanje programa za različne vhodne parametre. Švejkova metoda pri problemih kot je naš nikoli ni bila uspešna.

Fortran ima to (ne)ugodno lastnost, da so vse spremenljivke ob začetku postavljene na nič. Ker je naš program do neke mere iterativen, ne pozabite bisati spremenljivk!

Po končanem delu brišite `del` vse nepotrebne datoteke, ki jih imate na svojem področju. Dateteke simulacij znajo biti velike preko 1 MB, kar pri velikem številu uporabnikov in majhnem disku prav rado zapolni disk mrežnega serverja.

# Literatura


[1] Divjak J. Dovedan Z. *Fortran 77 s tehnikama programiranja.* Zveza organizacij za tehnično kulturo Slovenije, Ljubljana, 1988.

[2] Kansy K. Plaff G. Enderle G. *Computer Graphics Programming.* Springer-Verlag, New York, 1984.

[3] William H. et.al. *Numerical Recipes in Fortran; The Art of Scientific programing.* Cambridge university press, Cambridge, 1989.

[4] Pernuš Franjo. *Zbirka rešenih nalog iz teorije vezij in sistemov.* Akademska založba, Ljubljana, 1992.

[5] Gyergyék Ludvik. *Teorija vezij in sistemov.* Fakulteta za elektrotehniko, Ljubljana, 1975.


Laboratorij za CAD, Soba N-17. Govorilne ure za študente so od ponedeljka do četrtka od $13^{00}$ do $14^{00}$. Prosimo, da se držite razporeda urnika. Nadomeščanje zamujenih laboratorijskih vaj načeloma ni možno zaradi zasedenosti računalniške učilnice. Pred vajami na računalniku si pripravite gradivo za delo, da ne boste gledali v zrak (k sosedu).

Ljubljana 9. oktober 1995

Typeset in LaTeX