Proračunska aerodinamika
Nedelja 2

A. Simonović & J. Svorcan

Mašinski fakultet, Katedra za vazduhoplovstvo

2020/2021.

Sadržaj

Zakoni održanja

- Zakoni održanja mogu biti formulisani za *kontrolnu masu* (CM), što se često koristi u dinamici krutog tela.
- U mehanici fluida je medjutim teško pratiti kretanje odredjene mase, već uobičajeniji pristup podrazumeva analizu dešavanja u odredjenom delu prostora – *kontrolnoj zapremini* (CV).
- Zakon održanja mase za kontrolnu masu (materija se ne može stvoriti niti uništiti):

$$\frac{\mathrm{d}m}{\mathrm{d}t} = 0.$$

- Količina kretanja sistema se menja dejstvom sila na sistem:

$$\frac{\mathrm{d}(m\vec{v})}{\mathrm{d}t} = \sum \vec{f}.$$

- Prelaz sa jednog pristupa na drugi (CM na CV) vrši se pomoću Rejnoldsove jednačine prenosa:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CM}} \rho\phi \mathrm{d}\Omega = \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CV}} \rho\phi \mathrm{d}\Omega + \int_{S_{CV}} \rho\phi(\vec{v} - \vec{v}_b)\cdot\vec{n}\mathrm{d}S.$$

Zakoni održanja u konzervativnom obliku

Zakon održanja mase

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0$$

Zakon održanja količine kretanja

$$\frac{\partial (\rho \vec{v})}{\partial t} + \nabla \cdot (\rho \vec{v} \vec{v}) = -\nabla p + \nabla \cdot \tau + \rho \vec{b}$$

Zakon održanja energije

$$\frac{\partial}{\partial t}\left(\rho\left(e + \frac{1}{2}\vec{v} \cdot \vec{v}\right)\right) + \nabla \cdot \left(\rho \vec{v}\left(e + \frac{1}{2}\vec{v} \cdot \vec{v} + \frac{p}{\rho}\right)\right) = -\nabla \cdot \vec{q} + \nabla \cdot (\tau \cdot \vec{v}) + \rho \vec{v} \cdot \vec{b}$$

Zakon održanja veličine ϕ

$$\frac{\partial (\rho \phi)}{\partial t} + \nabla \cdot (\rho \phi \vec{v}) = \nabla \cdot (\Gamma \nabla \phi) + q_\phi$$

Uprošćenja

Kako su navedene j-ne prilično složene (nelinearne, spregnute, teške za rešavanje), uglavnom se uvode odredjena uprošćenja.

- Nestišljivo strujanje: gustina fluida je konstantna (pa najčešće i temperatura i viskoznost) (primenljivo na tečnosti i gasove na malim brzinama, $M < 0.3$).
- Neviskozno strujanje: zanemaruju se viskozni efekti (i članovi u j-nama). Tangentna komponenta brzine na zidu $\neq 0$.
- Potencijalno strujanje: jedno od najjednostavnijih, neviskozno i nevrtložno.
- Stoks (strujanje vrlo viskoznog fluida malim brzinama), Businesk (kod strujanja sa prenosom toplote), granični sloj (1D, nema povratnog strujanja, blaga promena geometrije) ...

Kratko podsećanje

Postojeći niz **a**, koji je zabeležen u datoteci **niz.txt**:
10
12 -51 -63 148 56 12 32 65 78 279
transformisati u niz $a^2 - 2a - 3$ i ispisati na ekran.

Provera

U Matlab-u/QtOctave-u skicirati ovu kvadratnu fju kao liniju, a niz a i njegovu transformaciju kao pojedinačne tačke.

Uvod u FORTRAN – Funkcije i sabrutine

Ukoliko je program složen i dugačak ili postoji skup naredbi koji se ponavlja korisno je podeliti program na potprograme – funkcije (**FUNCTION**) ili sabrutine (**SUBROUTINE**).

Razlika je što funkcija na osnovu ulaznih parametara vraća samo jednu vrednost izlaznog parametra, dok su kod sabrutine parametri istovremeno i ulazni i izlazni i moguće ih je sve menjati.

Struktura potprograma veoma je slična strukturi programa.

function Ime(x1, x2)	**subroutine Ime(x1, x2)**
...	
Ime = f(**x1**,**x2**)	...
return	
end function	**end subroutine**

Funkcije u FORTRAN-u – Najveći član niza

```
program niz
  real a(100), amax
  integer i, n
  open(1,file='niz.txt',status='old')
  read(1,*) n
  read(1,*) (a(i), i = 1,n)
  close(1)
  amax = maxniza(a,n)
  open(1,file='niz2.txt',status='new')
  write(1,*) 'Najveci clan je ', amax
  close(1)
end
```

```
function maxniza(f,n)
  real f(n), x
  x = f(1)
  do i = 2,n
   if (x.lt.f(i)) then
    x = f(i)
   end if
  end do
  maxniza = x
  return
end function
```

Sabrutine u FORTRAN-u – Sortiranje niza u rastućem poretku

```
program nizsort
  real a(100), b(100)
  integer i, n
  open(1,file='niz.txt',status='old')
  read(1,*) n
  read(1,*) (a(i), i = 1,n)
  close(1)
  call sortiraj(a,b,n)
  open(1,file='niz2.txt',status='new')
  write(1,*) (b(i), i = 1,n)
  close(1)
end
subroutine zameni(a,b)
  real a, b, pom
  pom = a
  a = b
  b = pom
end subroutine
```

```
subroutine sortiraj(a,b,n)
  real a(n), b(n)
  integer i, n
  do i = 1,n
   b(i) = a(i)
  end do
  do i = 1,n-1
   do j = i+1,n
    if (b(i).gt.b(j)) then
     call zameni(b(i),b(j))
    end if
   end do
  end do
end subroutine
```

Potprogrami u MATLAB/QtOctave-u

Na vrlo sličan način moguće je definisati potprogram u MATLAB-u:

$$\text{function } [y1,\ldots,yN] = \text{myfun}(x1,...,xM)$$

Primer fje sa jednim izlazom:

```
function y = average(x)
if ~isvector(x)
    error('Input must be a vector')
end
y = sum(x)/length(x);
end
```

Ukoliko je više izlaza:

```
function [m,s] = stat(x)
...
```

Početno Python-u

Python se može nabaviti na više načina:

- Python, verzija 2 ili 3, https://www.python.org
- Numerical Python, https://numpy.org
- Matplotlib, https://matplotlib.org

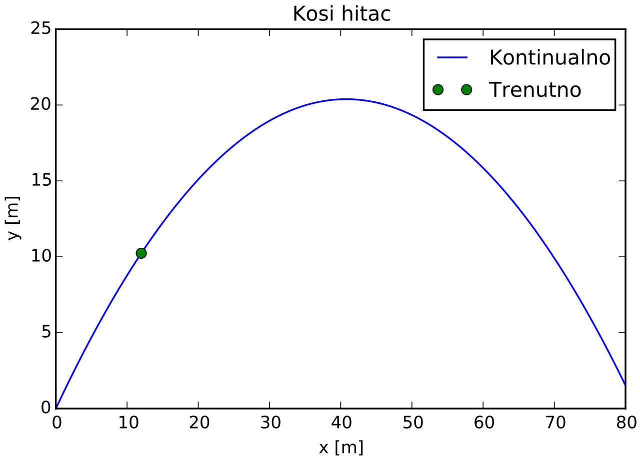
Veoma korisna je i Anaconda (Distribution), besplatna platforma koja sadrži oko 200 Python paketa, kao i sam Python https://www.anaconda.com/distribution/. Takodje sadrži i *Spyder*, integrisano razvojno okruženje, naročito korisno za pisanje programčića kakvi su nama potrebni.

Potprogrami u Python-u (primer kosog hica)

```
# Program za proracun vertikalnog hica, potrebne biblioteke:
from math import *
from numpy import *
import matplotlib.pyplot as plt

# pomocna fja koja racuna trenutni polozaj
def xy(v0x, v0y, t):
    g = 9.81
    return v0x*t, v0y*t - (1./2)*g*t**2

# komponente pocetne brzine
v0x = 20.; v0y = 20.
# trenutno vreme
time = 0.6
# trenutni polozaj
x, y = xy(v0x, v0y, time)
print 'Polozaj posle', time, 's je', (x, y)
# ugao izbacivanja
angle = atan(v0y/v0x)*180/pi
print 'Ugao izbacivanja je', '%.1f' % angle, 'deg'

# promena po vise vremenskih trenutaka
N = 101
t = linspace(0., 4.0, N); x1 = linspace(0., 0., N); y1 = linspace(0., 0., N)
for i in range(N):
    x1[i], y1[i] = xy(v0x, v0y, t[i])

# grafik
plt.plot(x1, y1, '-', x, y, 'o')
plt.xlabel('x [m]'); plt.ylabel('y [m]')
plt.legend(['Kontinualno', 'Trenutno'])
plt.title('Kosi hitac')
plt.axis([0, 80, 0, 25])
plt.savefig('kosi_hitac.pdf')
plt.show()
```

Potprogrami u Python-u (primer kosog hica)

Rezultat prethodnog koda prikazan je na slici ...



Uvod u FORTRAN – Formatiranje teksta

Naredbom FORMAT moguće je definisati izgled ulaznih i izlaznih stringova. Ovu specifikaciju koriste naredbe READ, WRITE i PRINT. Najčešće korišćene oznake su:

I*w*.*m*	*w*-dužina (broj mesta), *m*-broj nula – celi brojevi
F*w*.*d*	*w*-dužina, *m*-broj decimalnih mesta – realni brojevi
E*w*.*d*[E*e*]	*e*-eksponent – realni, eksponencijalni zapis
L*w*	*w*-dužina – logičke promenljive
A[*w*]	*w*-dužina – string
"a", 'a'	*a*-string
*n*X	*n*-broj blanko znakova (razmaka)
*n*H	*n*-broj karaktera
$	ne prelazi u novu liniju

Uvod u FORTRAN – Formatiranje teksta, primeri

Na prvo mesto treba upisati broj **r**.

r FORMAT(f1,f2,...,fn)

PRINT **r**, e1,e2,. . . ,en (umesto *)

WRITE(*,**r**) e1,e2,...,en

READ **r**, e1,e2,...,en

READ(*,**r**) e1,e2,...,en

Ili direktno: PRINT '(f1,f2,...,fn)', e1,e2,...,en

Npr:

```
br1 = 15
br2 = 3
PRINT '(A,I3,A$)','Aha!!! Prvi broj ',br1,' je veci od 10, dok je'
PRINT '(A,I3,A)',' drugi broj ',br2,'manji.'
```

Za dalji rad na formatiranju u FORTRAN-u

Korisni linkovi:

- http://force.lepsch.com/
-

Iterativno rešavanje jednačina

Postoje direktne metode rešavanja linearnih jednačina (čak i sistema linearnih jednačina). Medjutim, **ne postoje direktne** metode rešavanja **nelinearnih** jednačina, te se takve jednačine rešavaju **aproksimativnim** metodama (iterativno). Aproksimativne metode uvek prave izvesnu grešku i ne moraju uvek dovesti do rešenja. Korisnik/inženjer treba da bude u stanju da oceni grešku i odabere pogodnu metode zadovoljavajuće efikasnosti.

Iterativno rešavanje (metod sukcesivne zamene) podrazumeva:

1. početnu pretpostavku rešenja x_o,
2. definisanje pravila proračuna rešenja u sledećoj iteraciji $x_{n+1} = g(x_n)$ (slično matematičkoj indukciji).

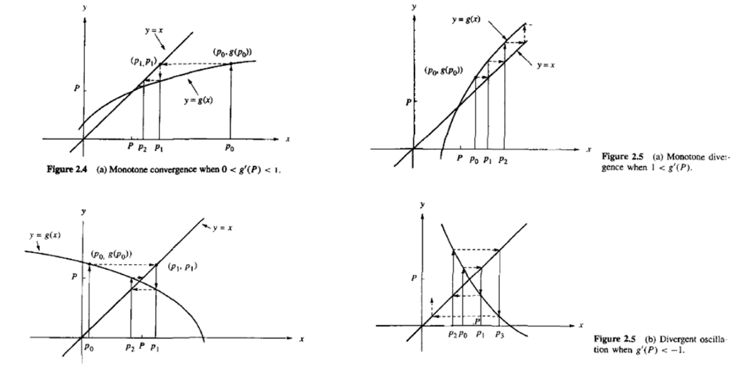
Iz jednačine koja se rešava $f(x) = 0$ treba izvesti potrebni oblik

$$x - g(x) = 0$$

Iterativno rešavanje jednačina

Da bi red dobijen po pravilu izračunavanja $x = g(x)$ bio konvergentan (doveo do rešenja) mora biti zadovoljen uslov:

$$|g'(x)| < 1$$



Slika: Konvergencija (divergencija) rešenja

Iterativno rešavanje jednačina

- Potrebni oblik $x - g(x) = 0$ moguće je izvesti iz polazne jne $f(x) = 0$ na neograničen broj načina.
- Iako je izbor proizvoljan, jedna (uobičajena) mogućnost je:

$$x = (1-k)x + kg(x).$$

- Proračunski zapis:

$$x_{n+1} = (1-k)x_n + kg(x_n).$$

Primer 1

Rešiti jednačinu $x^2 + 2x - 3 = 0$.

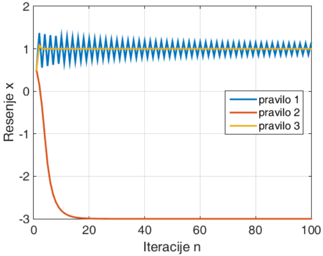
Moguća pravila:

1. $x_{n+1} = (3 - x_n^2)/2$,
2. $x_{n+1} = x_n + (x_n^2 + 2x_n - 3)/(x_n^2 + 5)$,
3. $x_{n+1} = x_n - (x_n^2 + 2x_n - 3)/(2x_n + 2) \ldots$

Šta se dogadja za različita pravila i različite početne vrednosti? Proveriti vrednost izvoda fje $g(x)$.

(Skript iter.m . . .)

Primer 1



Slika: Konvergencija rešenja pri različitim pravilima, $x_o = 0.5$

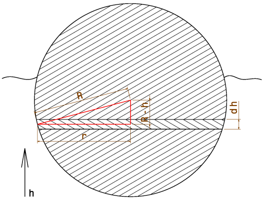
Matlab:

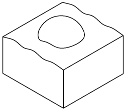
```
f = @(x) x.^2+2*x-3;
x1 = fzero(f,2)
x2 = fzero(f,-5)
% probati solve (syms), fsolve, ...
```

Primer 2

Odrediti do koje visine će biti potopljena u vodu drvena lopta poluprečnika $r = 10$ cm i gustine $\rho = 638$ kg/m^3.

Po Arhimedovom zakonu, sila potiska biće jednaka težini lopte. Zapremina potopljenog dela nalazi se integraljenjem ...





$$V_{pot} = \int_0^H r(h)^2 \pi dh$$
$$V_{pot} = \int_0^H (2Rh - h^2)\pi dh$$
$$V_{pot} = 2R\pi \frac{H^2}{2} - \pi \frac{H^3}{3}$$
...
$$\rho_{vod} V_{pot} = \rho_{drv} \frac{4}{3} R^3 \pi$$
$$f(H) = H^3 - 3RH^2 + 4R^3 \frac{\rho_{drv}}{\rho_{vod}}$$

Treba rešiti $f(H) = 0$.

Primer 3

Rešiti jednačinu
$x = \arctan(x) - a$.

Testirati šta se dogadja za različite početne uslove i željenu tačnost

...

(Rešenje: iter.f)

```
program iter
 real a, xp, xn, eps
 integer n
 write(*,*) 'Unesite broj a i zeljenu tacnost eps '
 read(*,*) a, eps
 write(*,*) 'Unesite pocetno resenje '
 read(*,*) xp
 open(1,file='rez.txt',status='unknown')
 xn = xp + 10*eps
 write(1,*) xn
 n = 0
 do while (abs(xn-xp).gt.eps)
  n = n+1
  xp = xn
  xn = atan(xp)-a
  write(1,*) xn
 end do
 write(1,*) 'Broj potrebnih iteracija je ', n
 close(unit=1)
end
```

Njutnov (Njutn-Rafson) metod nalaženja nula f-ja

Pravilo je definisano kao:

$$x_{n+1} = g(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}$$

Primer: Naći $\sqrt{n}$.

Izraz se može zapisati kao $x = \sqrt{n}$, odnosno $x^2 = n$.

Tada je $f(x) = x^2 - n$ i $f'(x) = 2x$, odakle sledi $x_{n+1} = x_n - \frac{x^2-n}{2x}$.

```
program koren
 real xo, y, n
 integer i
 write(*,*) 'Unesite broj n '
 read(*,*) n
 write(*,*) 'Unesite pocetno resenje '
 read(*,*) xo
 open(1,file='rez.txt',status='unknown')
 do i = 1,10
  write(1,*) xo
  y = f(xo,n)
  xo = y
 end do
 close(1)
end

function f(x,n)
 real x, n
 f = x - (x**2-n)/(2*x)
 return
end function
```

Obične diferencijalne jednačine

ODE su jednačine u kojima figurišu funkcija jedne nezavisne promenljive i njeni izvodi. Često ih srećemo u prirodnim i inženjerskim modelima/primerima.

Npr. slobodan pad objekta sfernog oblika opisujemo sledećim izrazom:

$$m\dot{u} = mg - D(u),$$

gde je m masa objekta, u brzina i D sila otpora vazduha.

Takodje, kretanje platforme promenljive mase može se opisati izrazom:

$$\frac{d}{dt}(mu) = F \Rightarrow \dot{m}u + m\dot{u} = F,$$

gde je m masa platforme, u brzina i F rezultujuća sila koja deluje na platformu.

Obične diferencijalne jednačine

- Ponašanje fizičkih procesa (naročito nestacionarnih) možemo opisati običnim diferencijalnim j-nama.
- Iako se mnoge tipske j-ne mogu rešiti analitički, mnogo veći deo realnih problema nema opšte rešenje i potrebno je pribeći numeričkim metodama.
- Opšti zapis obične diferencijalne j-ne n-tog reda:

$$F\left(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \frac{d^3y}{dx^3}, \dots, \frac{d^ny}{dx^n}\right) = 0.$$

- Partikularno rešenje iz skupa opštih rešenja odredjujemo zadavanjem dodatnih uslova.

Obične diferencijalne jednačine

Razlikujemo dva tipa dodatnih uslova:

- početne – kojima je definisana vrednost nepoznate funkcije $u(x,t)$ na početku (vremena $u_o(x,0)$ ili proračuna), i
- granične – kojima je definisana vrednost nepoznate funkcije $u(x,t)$ po granici domena $u(x_g,t)$ u svakom vremenskom trenutku.

Treba primetiti da su početni uslovi bili potrebni i pri iterativnom rešavanju j-na (znači, uvek u numeričkom proračunu iako ne moraju uvek imati fizičkog smisla).

ODE – Zadatak 1

Rešiti slobodan pad kuglice leda koja pada sa početne visine $H_o = 3000$ m. Poluprečnik kuglice je $a = 1$ cm, gustina leda je $\rho_{led} = 917$ kg/m^3, a dinamička viskoznost vazduha je konstantna i iznosi $\mu = 1.798 \cdot 10^{-5}$ kg/m/s.

Moguće je koristiti (približne) izraze:

$$\rho_{vaz} = \rho_o \frac{20000\ [\mathrm{m}] - h}{20000\ [\mathrm{m}] + h},$$

$$\mathrm{Re} = \frac{\rho_{vaz} u (2a)}{\mu_{vaz}},$$

$$C_D = \frac{24}{\mathrm{Re}} + \frac{6}{1 + \sqrt{\mathrm{Re}}} + 0.4, \mathrm{Re} > 0.$$

Zadatak 1 – Rešenje

Potrebno je još definisati početni uslov. Na početku kretanja, brzina čestice jednaka je 0, $u(0) = u_o = 0$.

Konačno, matematički model je:

$$m\dot{u} = mg - D(u) \Rightarrow \dot{u} = g - \frac{\rho_{vaz} u^2}{2m}(a^2\pi)C_D(\mathrm{Re}).$$

Kada izrazimo masu čestice kao $m = \rho_{led} V = \rho_{led} \frac{4}{3} a^3 \pi$, dobija se:

$$\dot{u} = g - \frac{\rho_{vaz} u^2}{2(\rho_{led} \frac{4}{3} a^3 \pi)}(a^2\pi)C_D(\mathrm{Re}),$$

odnosno

$$\dot{u} = g - \frac{3}{8} \frac{\rho_{vaz} u^2}{\rho_{led} a} C_D(\mathrm{Re}) = f(u, t).$$

Ojlerov metod za rešavanje

Najprostiji metod za rešavanje. Domen nezavisne promenljive delimo na sitne korake (ovde Δt) i računamo šta se dogadja sa zavisnom promenljivom u svakom koraku. Mana je akumulacija greške tokom proračuna.
Razvojem fje u Tejlorov polinom:

$$u(t^{n+1}) = u(t^n) + \Delta t \dot{u}(t^n) + \frac{1}{2}(\Delta t)^2 \ddot{u}(t^n) + o((\Delta t)^3).$$

Ukoliko koristimo samo prve članove $u(t^{n+1}) \approx u(t^n) + \Delta t \dot{u}(t^n)$.
Ako zamenimo izraz za prvi izvod iz prethodnih jednačina:

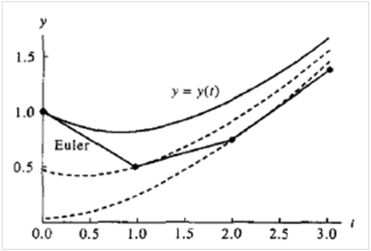
$$u(t^{n+1}) \approx u(t^n) + \Delta t \cdot f(u(t^n), t^n).$$

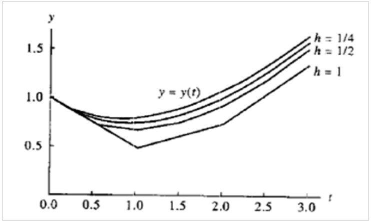
Jednostavniji zapis (šema unapred):

$$u^{n+1} \approx u^n + \Delta t \cdot f(u^n, t^n).$$

Ojlerov metod za rešavanje

U svakom koraku se sledeće rešenje "malo odmakne" od trenutnog rešenja (leva slika).
Uticaj koraka nezavisne promenljive $h = \Delta t$ je značajan (desna slika).





Zadatak 1 – Rešenje

Ojlerov metod primenjen na problem slobodnog pada kuglice leda:

$$u^{n+1} \approx u^n + \Delta t \Big[g - \frac{3}{8} \frac{\rho_{vaz} (u^n)^2}{\rho_{led} a} C_D(\mathrm{Re}) \Big].$$

Šta se dogadja ako primenimo centralnu šemu?

$$u^{n+1} \approx u^{n-1} + 2\Delta t \cdot f(u^n, t^n).$$

Tada:

$$u^{n+1} \approx u^{n-1} + 2\Delta t \Big[g - \frac{3}{8} \frac{\rho_{vaz} (u^n)^2}{\rho_{led} a} C_D(\mathrm{Re}) \Big].$$

Ojlerov metod – Kodovi

FORTRAN:

```
subroutine ojler(x,y,f,h)
  real x, y, h, f1
  f1 = f(x,y)
  y = y + h*f1
  x = x + h
end subroutine
```

MATLAB/QtOctave:

```
function [x1,y1] = ojler(x,y,fun,h)
  y1 = y + h*fun(x,y);
  x1 = x + h;
end
```

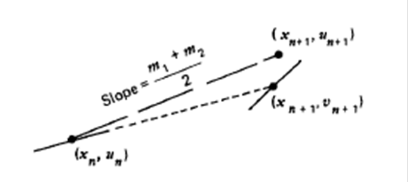
Poboljšan Ojlerov (Hjunov) metod za rešavanje

Prethodno je korišćena aproksimacija $u^{n+1} \approx u^n + \Delta t \cdot f(u^n, t^n)$. Tačnost se može povećati ako se razmatra šta se dogadja i na početku i na kraju intervala i uzme se aritmetička sredina:

$$k_1 = \Delta t \cdot f(u^n, t^n),$$

$$k_2 = \Delta t \cdot f(u^n + k_1, t^{n+1}),$$

$$u^{n+1} \approx u^n + \frac{1}{2}(k_1 + k_2) = u^n + \frac{\Delta t}{2}\left[f(u^n, t^n) + f(u^n + k_1, t^{n+1})\right].$$



Hjunov metod – Kodovi

FORTRAN:

```
subroutine ojlerp(x,y,f,h)
  real x, y, h, k1, k2
  k1 = f(x,y)*h
  k2 = f(x+h,y+k1)*h
  y = y + (k1+k2)/2
  x = x + h
end subroutine
```

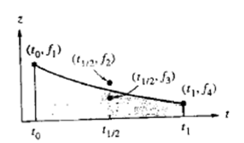
MATLAB/QtOctave:

```
function [x1,y1] = ojlerp(x,y,fun,h)
  k1 = fun(x,y)*h;
  k2 = fun(x+h,y+k1)*h;
  y1 = y + (k1+k2)/2;
  x1 = x + h;
end
```

Metod Runge-Kuta

Veoma korišćen metod 4. reda (Ojlerov je 1, a Hjunov 2. reda).
Tačnost je dodatno povećana razmatranjem 4 različite tačke:

$$
\begin{aligned}
k_1 &= \Delta t \cdot f(u^n, t^n),\\
k_2 &= \Delta t \cdot f(u^n + \tfrac{k_1}{2}, t^{n+\frac{1}{2}}),\\
k_3 &= \Delta t \cdot f(u^n + \tfrac{k_2}{2}, t^{n+\frac{1}{2}}),\\
k_4 &= \Delta t \cdot f(u^n + k_3, t^{n+1}).
\end{aligned}
$$



Tada:

$$u^{n+1} \approx u^n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

U Matlabu, postojeća funkcija **ode45**. (primer difjed.f)

Metod Runge-Kuta – Kodovi

FORTRAN:

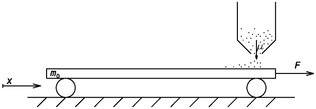
```
subroutine rkut(x,y,f,h)
  real x, y, h, k1, k2, k3, k4
  k1 = f(x,y)*h
  k2 = f(x+h/2,y+k1/2)*h
  k3 = f(x+h/2,y+k2/2)*h
  k4 = f(x+h,y+k3)*h
  y = y + (k1+2*k2+2*k3+k4)/6
  x = x + h
end subroutine
```

MATLAB/QtOctave:

```
function [x1,y1] = rkut(x,y,fun,h)
  k1 = fun(x,y)*h;
  k2 = fun(x+h/2,y+k1/2)*h;
  k3 = fun(x+h/2,y+k2/2)*h;
  k4 = fun(x+h,y+k3)*h;
  y1 = y+(k1+2*k2+2*k3+k4)/6;
  x1 = x + h;
end
```

ODE – Zadatak 2

Na pokretnu platformu mase m_o [kg] deluje konstantna sila F [N] i pesak se nanosi konstantnim protokom μ [kg/s].
Izraziti nestacionarne veličine brzinu $u(t)$ i ubrzanje $a(t)$.
Zanemariti trenje.



Problem se može opisati izrazima:

$$m(t) = m_o + \mu t, \dot{m} = \mu,$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(mu) = F \Rightarrow \mu u + (m_o + \mu t)\dot{u} = F.$$

Uporediti analitičko i numeričko rešenje. (Rešenje u pretpostaviti kao proizvod dve fje $u = cd$.)

Zadatak 2 – Rešenje

Početni uslov: platforma kreće iz mirovanja $u(0) = u_o = 0$.
Numeričko rečenje – Ojlerov metod:

$$\mu u + (m_o + \mu t)\dot{u} = F \Rightarrow \dot{u} = \frac{F - \mu u}{m_o + \mu t},$$

$$u^{n+1} \approx u^n + \Delta t \dot{u}^n = u^n + \Delta t \frac{F - \mu u}{m_o + \mu t}.$$

Analitičko rešenje – smena $u = cd \Rightarrow \dot{u} = \dot{c}d + c\dot{d}$:

$$\mu cd + (m_o + \mu t)(\dot{c}d + c\dot{d}) = F,$$

$$d[\dot{c}(m_o + \mu t) + \mu c] + (m_o + \mu t)c\dot{d} = F.$$

Fju c odabrati tako da $\dot{c}(m_o + \mu t) + \mu c = 0$ odakle se i nalazi c,
pa preostaje da se reši $(m_o + \mu t)c\dot{d} = F$ odakle se dobija d.

Zadatak 2 – Rešenje

Za $m_o = 25$ kg, $F = 1$ N i $\mu = 0.1$ kg/s moguće je formirati grafike: