Úvod, seznámení s Octave (spuštění, hlášení chyb, psaní komentářů, ukládání)

Octave

 - programovací jazyk podobný Matlabu

Instalace

- pro Windows - ze stránky ftp://ftp.gnu.org/gnu/octave/windows/ stáhnout instalační soubor octave-4.0.2-installer.exe popř. jakoukoli novější verzi

 - dvakrát na něj kliknout a sledovat instrukce

 - doporučuju domů

Stránky, které mohou pomoci

 - octave.cz – český průvodce programem

 - stackoverflow.com

 - google.com, co si budeme povídat

Octave v učebně

- na virtuálním disku - Virtualization - Shortcut -> VirtualBox -> UFY -> UFY.vbox (modrá krychle).

Spustíme Octave a začneme

Nový soubor

 - pomocí

Ukládání

 - pomocí nebo

Otevření souboru

 - pomocí

Spouštění

 - pomocí

Psaní komentářů pomocí # nebo % - co je na řádku za # nebo za % Octave ,,nevidí“

Další řádek

 - buď přes Enter, nebo přes ;

Help

- do příkazového řádku napsat help a a jméno příkazu, který chete použít (ten znát musíte)

- př. help plot

- stránka helpu je dlouhá - nezobrazí se celá

- stisknutím ENTERu se posunete o řádek, stisknutím MEZERNÍKU se posunete o stránku a klávesa Q tento pager ukončuje

Hlášení chyb

a) Parse error

- př. y = x\*\*\*2 – objeví se parse error

- parse error se objeví, pokud Octave nerozumí něčemu, co jste napsali

- například když se překlepnete - prohodíte slovosled

- přibližně napíše a ukáže, kde by mohla být chyba

- používá symbolu ^ pro označení místa na řádku, kde nebyl schopen pochopit vstup

b) Zprávy, které se objeví přímo během vykonávání příkazu - ve výpočetní době

 - př.

c = 10 + b

- proměnná b v souboru Pokus.m není nikde definovaná

- Octave hlásí:

 error: 'b' undefined near line 1 column 10

error: called from

 Pokus at line 1 column 3

- pokud ukončíte příkaz středníkem, říkáte Octave, aby výsledek nevypisoval

D = 10;

vs

D = 10

Octave vypíše:

D = 10

Instalace balíčků

- ze stránky <http://octave.sourceforge.net/packages.php> stáhněte požadovaný balíček do složky C:\Octave\Octave-4.0.2\lib\octave\packages (záleží, kam jste instalovali Octave) a do příkazového řádku napište pkg install ...

- místo tří teček doplňte balíček, např. pro balíček image bude vypadat instalace jako pkg install image-1.0.0.tar.gz

Nahrání balíčku

- na začátku programu napiště pkg load ...

- místo tří teček doplňte balíček, např. pro balíček optim bude vypadat načtení jako pkg load optim

Octave: datové typy, funkce, velikosti objektů & práce s maticemi, aritmetické operace

Zabudované datové typy

a) Numerické objekty

- reálné a komplexní skaláry a matice (česky – číslo, vektor nebo matice, které mohou být reálné, nebo komplexní)

- komplexní číslo – např. 1 + 2i (zápis v Octave – 1 + 2i, NE 1 + 2 i (mezi 2 a i nesmí být mezera)

 - ukládány jako čísla s dvojnásobnou přesností

 - maticové objekty mohou být libovolné velikosti a dají se dynamicky rozšiřovat a

tvarovat. Je jednoduché vyjmout konkrétní řádek, sloupec, submatici pomocí rozmanitých a účinných indexovacích vlastností.

b) Řetězcové objekty

- sekvence znaků ohraničených dvojitými nebo jednoduchými uvozovkami (”text” nebo ‘text‘)

 - veškeré indexové operace tedy fungující pro matice fungují i pro řetězce

- většinou se jedná o text (třeba pro napsání něčeho na příkazový řádek, např. ,,Zadejte číslo:“

Aritmetické operace

+ - plus zadání v Octave: +

- - mínus zadání v Octave: -

\* - krát zadání v Octave: \*

/ - děleno zadání v Octave: /

2 - na druhou zadání v Octave: buď \*\* nebo ^

- je třeba si dávat pozor u matic, zda jde o mativé operace, nebo operace po složkách, viz dále

exp(x) - počítá exponenciální hodnotu x

log(x) - přirozený logaritmus prvku x

log2(x) - logaritmus základu 10 pro každý prvek x

log10(x) - logaritmus základu 2 pro každý prvek x

sqrt(x) - odmocnina z x

* spousta dalších funkcí je k nalezení na octave.cz

Matice

- obdélníkové či čtvercové schéma čísel nebo nějakých matematických objektů – prvků matice (též elementů matice)

- obsahuje obecně m řádků a n sloupců

- používá se např. při výpočtu soustav lineárních rovnic

- př. $\left(\begin{matrix}1&0\\0&1\end{matrix}\right)$ je matice o velikosti 2x2, nebo $\left(\begin{matrix}1&2&3\\4&5&6\end{matrix}\right)$ je matice o velikosti 2x3

- Octave chápe každou proměnnou jako matici určité velikosti (např. číslo 5 chápe jako matici 1x1, která pro něj vypadá jako [5])

 - matice se v Octave zapisují pomocí hranatých závorek

 - př.

 a = [ 1, 1, 2; 3, 5, 8; 12, 21, 34 ]

- příklad jednoduché matice 3x3, která vypadá takto: $\left(\begin{matrix}1&1&2\\3&5&8\\12&21&34\end{matrix}\right)$

- hodnoty na řádku se oddělují mezerou nebo čárkou

- řádky se oddělují středníkem nebo novým řádkem

 - skládání matice z menších matic

 P = [1, 2; 3, 4; 5, 6];

 Q = [1, 1; 2, 2];

 R = [0, 0];

 S = [P,[Q;R]];

 ans = 1 2 1 1

 3 4 2 2

 5 6 0 0

- b = rand (3, 2);

 - příkaz rand (m,n) vrátí matici o m řádcích a n sloupcích

- c = [1,1,1;2,2,2];

- c

 - příi použití středníku za zadáním matice se matice nevytiskne

- k tomu, abyste zobrazili hodnotu jakékoli proměnné stačí jednoduše napsat její název.

 - m = [1 2; 3] - chyba, protože v každém řádku je jiný počet sloupců

- operace s maticemi

 - sčítání a odčítání matic po prvcích:

 [1,2,3;4,5,6] + [1,1,1;2,2,2]

 - Octave vypíše výsledek

ans = 2 3 4

 6 7 8

 - umocňování matice reálným číslem

 Z = [1,2;4,5];

Z^2 - !!! Pozor, neumocní každé číslo v matici na druhou, ale maticově vynásobí matici Z s maticí Z

ans = 9 12

 24 33

Z. ^2 – umocní každý prvek matice na druhou

ans = 1 4

 16 25

Z^-1 – umocněním na -1 získáme inverzní matici k matici původní

Z\*(Z^-1)

ans = 1 0

 0 1

vynásobením původní a inverzní matice získáme matici jednotkovou (má na diagonále jedničky a jinde nuly)

 - transpozice (operace, která přemění řádky na sloupce)

 Z‘

 ans = 1 4

 2 5

- dělení – pravostranné pomocí / a levostranné pomocí \ (obě odpovídají vynásobení matice s maticí inverzní)

- násobení a dělení po prvcích

 - pokud před \* nebo / dáme . (tečku), bude operace probíhat po prvcích

 M = [1, 2, 3; 4, 5, 6]

 N = [2, 4, 6; 8, 10, 12]

 M.\*N

 ans = 2 8 18

 32 50 72

 M./N

 ans = 0.5 0.5 0.5

 0.5 0.5 0.5

- další užitečné funkce

 - inv - výpočet inverzní matice

 - det - výpočet determinantu

 - rank - výpočet hodnosti matice

Hledání prvku matice – indexy

- pokud máte matici a chcete z ní zobrazit jeden prvek, jehož znáte pozici

- např matice A = [1, 2, 3; 4, 5, 6], tedy

 A = $\left(\begin{matrix}1&2&3\\4&5&6\end{matrix}\right)$

- chceme-li prvek na pozici 1, 1 (první řádek, první sloupec) což je v našem případě 1, napíšeme

 A(1,1)

- a Octave vypíše 1

- pro prvek na pozici 2, 3 (druhý řádek, třetí sloupec) tedy šestku, napíšeme

 A(2,3)

- a Octave vypíše 6

- požadované číslo můžeme uložit do nové proměnné, např. B

 B = A(2,3)

- pokud chceme část sloupce nebo řádku, nebo část matice, použijeme dvojtečku

 A(2,2:3)

 - což znamená, že chceme druhý řádek a v něm sloupec dva až tři

 - Octave tiskne

 5 6

- pokud chceme celý řádek, nebo sloupec, opět použije dvojtečku

 A(:,1)

 - což znamená, že chceme všechny řádky a sloupec jedna

 - Octave vytiskne

 1

 4

Velikosti objektů

Funkce, které se dají použít

- columns (a)

- vrátí počet sloupců v proměnné a.

- rows (a)

- vrátí počet řádků v proměnné a.

- length (a)

- vrátí "délku" objektu a. Pro maticové objekty je délkou myšlen počet řádků nebo sloupců, kterýkoli je větší

- size (a)

- vrátí počet řádků a sloupců proměnné a.

- isempty (a)

- vrátí 1 pokud se jedná o prázdnou matici (buď počet řádků, počet sloupců nebo obojí je rovno nule). Jinak vrátí 0

- dále existuje: isscalar, isvector, ismatrix.

Příklad

mat = [1, 2; 3, 4; 5, 6]

sloupce = columns (mat);

radky = rows(mat);

delka = length(mat);

radkyasloupce = size(mat);

jeprazdna = isempty(mat);

jematice = ismatrix(mat)

Funkce

- účel – zjednodušení delších programů

- mohou být v externích souborech a pak je možné je volat stejně, jako zabudované funkce

- soubor s funkcí se musí jmenovat stejně jako funkce (v následujícím příkladu má funkce název funkce1)

- funkce se skládá ze tří částí:

 a) je třeba jí uvést pomocí příkazu function a za ním napsat název funkce

 function funkce1(x,y)

- v kulaté závorce za funkce1 jsou dvě proměnné, to znamená, že až ji budeme volat, zadáme jí dvě proměnné

 b) další částí je tzv. Body, neboli tělo, kde je definováno, co bude funkce dělat

 z=x^2+y

- zadání toho, co má funkce s přidělenými hodnotami dělat (jsou zde dvě proměnné, x a y)

- výsledkem je hodnota z

 c) poslední částí je ukončení funkce pomocí příkazu endfunction

- celkově tedy

 function funkce1(x,y)

z=x^2+y

endfunction

- pokud chceme funkci zavolat, vytvoříme nový soubor ve stejném adresáři, jako je soubor funkce1.m a napíšeme do něj

 funkce1(2,5)

 - funkce jedna má dvě proměnné, proto jsou v závorce dvě čísla oddělená čárkou

- Octave vypíše z = 9

- zkuste do závorky napsat jen jedno číslo

- s proměnnou z se ale nedá dál pracovat

- pokud bychom s vypočítanou proměnnou chtěli pracovat dál, musíme funkci definovat trochu jinak a to následujícím způsobem (tato funkce bude počítat průměr prvků vektoru - součet prvků vektoru dělený jeho délkou)

 function retval = funkce2 (v)

retval = sum (v) / length(v) %místo retval můžete použít jakýkoli název chcete

endfunction

- zavolání funkce

 funkce2([2,3]);

- zde se provede pouze funkce a výsledek se nikam neuloží

- v kulaté závorce je prvek, se kterým má funkce pracovat

- v hranaté závorce je ten prvek - vektor - a ten se píše do hranaté, proto jsou zde nejprve kulatá a pak hranatá závorka

- pokud zavoláme funkci jako

 retval = funkce2([2,3]);

pak se výsledek uloží do proměnné retval

- pokud chceme vytvořit funkci, která vypočítá a uloží více proměnných, vypadá funkce následovně

 function [x, y, z] = funkce3 (e,f,g)

 x = 3\*e;

 y = 3+f;

 z = 2-g;

endfunction

- a aby byly výsledky uloženy v proměnných a, b, c se volá jako

[a, b, c] = funkce3(1,1,1);

- funkce lze zadávat i pomocí @, ale to až později

Octave: příkazy (if, switch, while, do-until, for, break), proměnné (stálé, globální)

If

- příkaz, který dělá rozhodnutí

- jsou tři základní možnosti, jak může vypadat

- vždy musí začínat příkazem “if“ a končit příkazem “endif“

a) if (podmínka)

 udělej toto, pokud je podmínka pravdivá

 endif

b) if (podmínka)

 udělej toto, pokud je podmínka pravdivá

 else

 udělej toto, pokud není podmínka pravdivá

 endif

c) if (podmínka1)

 udělej toto, pokud je podmínka1 pravdivá

 elseif (podmínka2)

 udělej toto, pokud je podmínka2 pravdivá

 else

 udělej toto, pokud není ani jedna z podmínek pravdivá

 endif

- u této varianty je možné udělat libovolné množství podmínek

- příklady k jednotlivým možnostem

a)

 a = 10;

if a == 10

 disp("a je rovno deseti")

endif

 - !!! u prvního a = 10 je pouze jedno “=“, proměnné a přiřazujeme číslo deset

- !!! u příkazu if jsou mezi “a“ a číslem použité dvě rovná se “==“. To proto, že se ptáme, jestli je a identické s desítkou

 - pokud je tedy a rovno deseti, vypíše Octave “a je rovno deseti“

 - pokud a není rovno deseti, Octave nevypíše nic

b)

 a = 11;

if a == 10

 disp("a je rovno deseti")

else

 disp("a neni rovno deseti")

endif

- v tomto příkladu Octave vypíše “a je rovno deseti“ pokud je a rovno deseti, jinak vypíše “a není rovno deseti“

c)

 a = 11;

if a == 10

 disp("a je rovno deseti")

elseif a == 11

 disp("a je rovno jedenácti")

else

 disp("a neni rovno deseti ani jedenácti")

endif

- v tomto příkladu Octave vypíše “a je rovno deseti“ pokud je a rovno deseti nebo “a je rovno jedenácti“ pokud je a rovno jedenácti nebo “a není rovno deseti ani jedenácti“, pokud a není rovno deseti ani jedenácti

While

- nejjednodušší smyčkový příkaz v Octave

- smyčka se opakuje, dokud není porušena podmínka

 while (podmínka)

 udělej toto, pokud je podmínka splněna a pokračuj, dokud nebude porušena

 endwhile

- příklad

 a = 10;

while a <= 15

 disp(a)

 a = a + 1;

endwhile

- přičítej k “a“ jedničku, dokud nebude porušena podmínka, že “a“ je menší nebo rovno deseti – Octave vytiskne 10, 11, 12, 13, 14, 15

- !!! pozor na nekonečnou smyčku

- příklad nekonečné smyčky

 a = 10

while a = 10

 disp(a)

endwhile

- Octave bude donekonečna tisknout 10 a Vám nezbyde, než ho vypnout natvrdo přes Správce úloh

Do-Until

- to stejné jako while, pouze podmínka je uvedena na konci

 do

 udělej toto, pokud je podmínka splněna a pokračuj, dokud nebude porušena

 until (podmínka)

- příklad

 a = 10;

do

 disp(a)

 a = a + 1;

until(a == 16)

- přičítej k “a“ jedničku, dokud se “a“ nebude rovnat 16 – Octave vytiskne 10, 11, 12, 13, 14, 15

- zde nemůžeme do podmínky dát je menší nebo rovno, protože deset je vždy menší než šestnáct a Octave by tedy nic neudělal (příkaz by znamenal: dělej, dokud nebude 10 < 16)

For

- počítá iterace pro zadaný interval

- všechny smyčky while lze přepsat na smyčky for

- pokud znáte dopředu počet iterací (opakování) je výhodnější for

 for i = x0:krok:xn

 udělej toto

 endfor

 - kde x0 znamená, krok znamená a xn znamená, např. 0:1:5 znamená 0, 1, 2, 3, 4, 5

 - může se jít i sestupně, např. 10:-1:0 znamená 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

- příklad

 for i = 0:1:5

 disp(i)

endfor

- pro i od nuly do pěti, s krokem jedna, zobrazí i

- Octave tedy tiskne 0, 1, 2, 3, 4, 5

Break

- možnost, jak se dostat ze smyček while a for

- tento příkaz lze použít pouze v těle smyčky

- tam kde chcete smyčku ukončit, napíšete break

Octave: tvorba grafů a histogramů, fitování

- grafy se tvoří pomocí příkazu plot(x, y, ...), kde x jsou x-ové souřadnice, y jsou y-ové souřadnice a ... jsou další volitelné vlastnosti grafu, viz níže a internet

Jednoduché příklady grafů

x=[0 5 1 3];

y=[4 -2 -3 8];

plot(x,y)

 - první bod [0,4], druhý [5,-2], třetí [1,-3] atd., spojené čárou

x10=0:pi/5:2\*pi;

 - vytvoření proměnné x10, která je od 0 do 2π s krokem po π/5

plot(x10,sin(x10))

 - graf sinu(x10)

x200=0:pi/100:2\*pi;

plot(x200,sin(x200))

 - to samé jeko vpředchozím případě, ale s více body

Popis os, titulek grafu

- pomocí příkazů title(), xlabel() a ylabel() pod příkazem plot()

- př.

 plot(x, y)

title('Zavislost vysky na case')

xlabel('Casove kroky')

ylabel('Vyska [m]')

- text pro titulek i popis os musí být v uvozovkách “ “ nebo ‘ ‘, české znaky nepoužívejte

Změna barvy, stylu, popisky

plot(0:5,5:-1:0,'^',0:6,6:-1:0,'-.',0:8,8:-1:0,'g\*;popisek;' , 0:.1:9,9:-.1:0,'.r',1:10,9:-1:0,'--g\*',2 :10,9:-1:1,'c;usecka;')

- 0:5 - x-ová souřadnice

- 5:-1:0 - y-ová souřadnice (začíná v 5 a s krokem -1 jde do 0)

- barva: r, g, b, m, c nebo w (červená, zelená, modrá, purpurová, azurová a bílá)

- styl: čárový (výchozí) -, čárkovaný --, tečkovaný ., čárkatečka -., hvězdičky \*, trojúhelníčky ^

- bodový graf: uvedením symbolu, který se má v daném bodě vykreslit \*, +, o, x (pokud se před znak umístí --, budou body spojeny čárkovaně pokud se před znak umístí -, budou body spojeny čarou

- mezi ; a ; lze umístit popisek

Více grafů do jednoho grafu

a) první možnost - zadání všech grafů do jediného příkazu plot, který může mít variabilní počet parametrů:

xVice=0:pi/100:2\*pi;

plot(xVice,sin(xVice),xVice,cos(xVice))

b) druhá možnost - pomocí funkce hold, která podrží aktuální obsah výstupního zařízení

plot(xVice,sin(xVice)) #udělá graf

ishold #jaká je hodnota funkce hold? Pokud je zaplá, vytiskne se ans = 0

hold on #zapnutí funkce hold (podržet aktuální obsah výstupního zařízení)

plot(xVice,cos(xVice))

hold off #vypnutí funkce hold

Více grafů vedle sebe

- pomocí příkazo subplot() před příkazem plot()

- subplot vypadá následovně: subplot(počet řádků, počet sloupců,index pozice grafu)

- př.

 subplot(2,1,1)

plot(x1, y1)

title('Prvni graf')

subplot(2,1,2)

plot(x2, y2)

title('Druhy graf')

- celkový obraz má tedy dva řádky a jeden sloupce

Histogram

 data = load(‘rychlosti.txt‘);

 načtení dat

 x = data(:,1);

z dat vyber jen první sloupec (dvojtečka znamená vyber vše, tzn. od začátku do konce (např. 1:10 by bylo od prvního do desátého místa))

 hist(x,15)

 vytvoř histogram z načteného x, histogram bude mít 15 sloupců

 prumer = mean(x)

 výpočet průměru

m = median(x)

 výpočet mediánu

Fitování

- pomocí metody nejmenších čtverců



- výsledné řešení má minimalizovat součet čtverců odchylek vůči každé rovnici (na obrázku se zeleně vyznačené odchylky umocní na druhou a sečtou a tento součet se sanží metoda minimalizovat)

- nutné načíst balíček optim

 pkg load optim

- dále definovat x-ové a y-ové souřadnice dat, která mají být fitována

 x = 0:0.005:5\*T0;

y = (n0.\*( exp((-lam).\*x))).+ 10000000.\*randn (size (x));

- pak nadefinovat funkci, která bude data fitovat s proměnnými, které budou optimalizovány metodou (v našem případě proměnná p(1))

 function y = ffun(x, p)

y = (p(1).\*exp(-p(2)\*x));

endfunction

- dále zvolit výchozí bod pro proměnnou p(1)

 p = [3\*(10^8) (log(2))/T0];

- a zbývá použít funkci leasqr, která počítá několik věcí (yfit, pfit, cvg, iter, viz. nápověda, co znamená co)

 [yfit pfit cvg iter] = leasqr(x, y, p, "ffun");

Příklady

1. Míč je vypuštěn z výšky 1 m. Po každém nárazu do podložky míč vyskočí do poloviny výšky, ze které padal. Jakou celkovou dráhu urazí, než se zastaví? Vytvořte dva grafy: na prvním zobrazte zavislost výšky na čase a na druhém závislost celkové dráhy na čase.



1. Nahrajte data weather.ascii a vyberte z nich druhý sloupec (nadmořská výška) a třetí sloupec (teplota). Udělejte bodový graf závislosti teploty na nadmořské výšce. Vypočtěte pomocí funkce polyfit() koeficienty lineární regres a do stejného grafu toto regresi vykreslete. Udělejte popisky obou grafů, popiště osy a vypočítejte korelaci a kovarianci.



1. ,,Nasimulujte“ reálná data pro radioaktivní rozpad. Radioaktivní rozpad má vzorec

 $N= N\_{0}e^{-λt}$, kde *N* je počet částic v daném časovém okamžiku, *N*0je počet částic na začátku, *λ* je přeměnová konstanta a *t* je čas. Data vytvořte pro čas od 0 do dvou s krokem 0.005 tedy t = 0:0.005:2 a jejich rozptyl tím, že ke každému vypočítanému bodu přičtete 10000000.\*randn(size (t)) (pokud budete data vytvářet přes for smyčku, bude rozptylování vypadat trochu jinak). Tyto data poté pomocí metody nejmenších čtverců nafitujte vzorcem pro radioaktivní rozpad (budete potřebovat balíček optim).



1. Stáhněte si obrázek galaxy.jpg a zobrazte ho. Na tento obrázek dále zobrazte graf s Fibonaccoho spirálou, která je dána vztahy:

x1(end + 1) = 0.9\*cos(deg2rad(m))\*exp(0.0053468\*m)+498;

y1(end + 1) = 0.75\*sin(deg2rad(m))\*exp(0.0053468\*m)+410;

 Budete potřebovat nahrát balíčky nurbs a image.

1. Stáhněte soubor rychlosti.txt a vytvořte z nich histogram o pětadvaceti sloupcích.



1. Pro výpočet pohybu kyvadla je možné použít nástroje pro výpočet obyčejných diferenciálních rovnic zabudovaných v Octave. Nahrajte balíček odepkg, zadejte počáteční podmínky *m* = 1, *k* = 1 a *c* = 0.3 a pod ně napište

dy\_dt = @(t,y) [y(2);-(c/m) \* y(2) - (k/m) \* y(1) ];

 - zadání funkce

odeopt = odeset ('RelTol', 0.00001, 'AbsTol', 0.00001,'InitialStep',0.5,'MaxStep',0.5);

 - zadání parametrů

[t,y] = ode45(dy\_dt,[0 25], [0.0 1.0],odeopt);

 - použití příkazu ode45 (výpočet obyčejných diferenciálních rovnic)

- vytvořte grafy závislosti rychlosti a polohy na čase v jednom grafu a dráhy na rychlosti v druhém grafu a umístěte je vedle sebe



1. Strojvedoucí rychlíku, který se pohyboval rychlostí *v*1 = 108 km·h-1 spatřil ve vzdálenosti 180 m před sebou nákladní vlak pohybující se stejným směrem rychlostí *v*2 = 32,4 km·h-1. Strojvedoucí začal brzdit a vlak zpomalil se zpomalením a = 1,2 m·s-2. Zjistěte, zda se vlaky srazí, popřípadě v jakém čase a v jaké vzdálenosti se srazí. (použijte příkaz fsolve())



1. Vytvořte horní a dolní integrální součet pro funkci y = -2+x/2+3\*sin(x), tak, aby bylo možné měnit počet intervalů. Integrál zobrazte. Zobrazte a vypočítejte též lichoběžníkovou metodou (příkaz trapz())



1. Jakou dráhu urazí těleso, které se pohybuje rychlostí velikosti *v* = 3*t*2 + *t* -1 od 2 do 10 sekund od začátku pohybu. Dráha je integrálem rychlosti, a proto využijte numerický výpočet integrálu pomocí příkazu trapz().