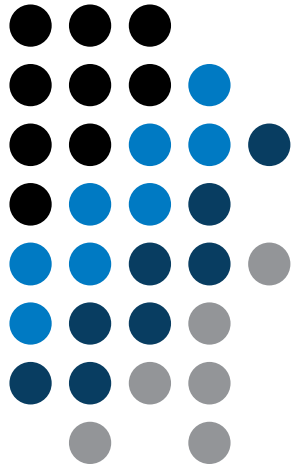
A0B17MTB – Matlab

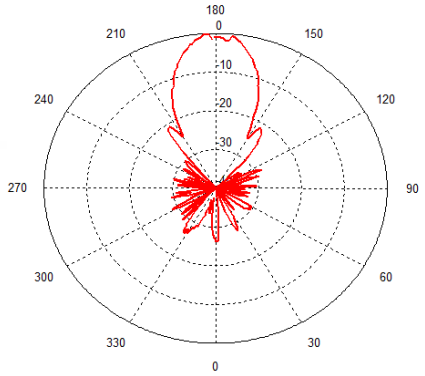
Část #13

Miloslav Čapek
miloslav.capek@fel.cvut.cz

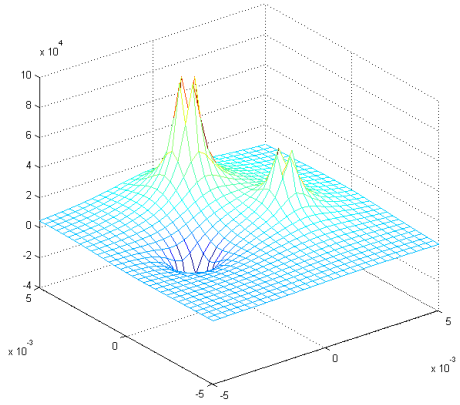
Katedra elektromagnetického pole
B2-626, Dejvice



Naučíte se …

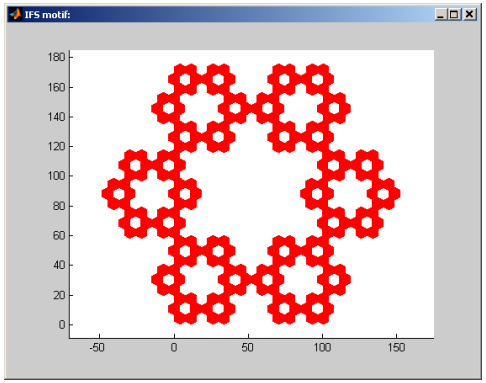


Cvičení #1 – zpracování VD antény



Cvičení #2 – zobrazení pole kolem bodových nábojů

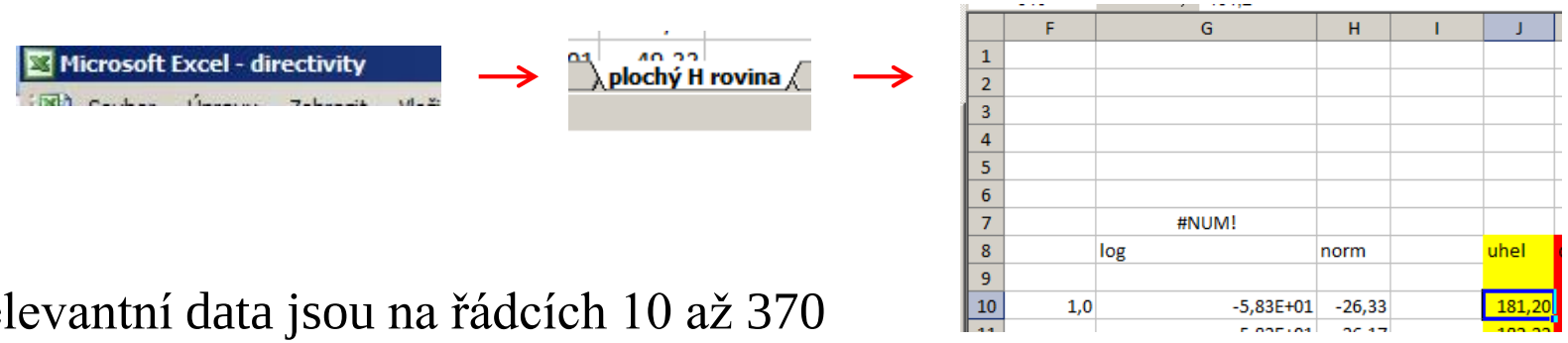
Cvičení #3 – generace IFS fraktálních struktur



Cvičení #1/01

- načteme si ze souboru `directivity.xls`, listu `plochý H rovina` sloupečky `uhel` a `dBm`

```
>> num = xlsread('directivity.xls','plochý H rovina','J10:K370');
```



- relevantní data jsou na řádcích 10 až 370
  → měli bychom získat 361 hodnot

```
>> size(num) % = [361 2]
```

- úhly si uložíme do matice U, změřené hodnoty výkonu do D

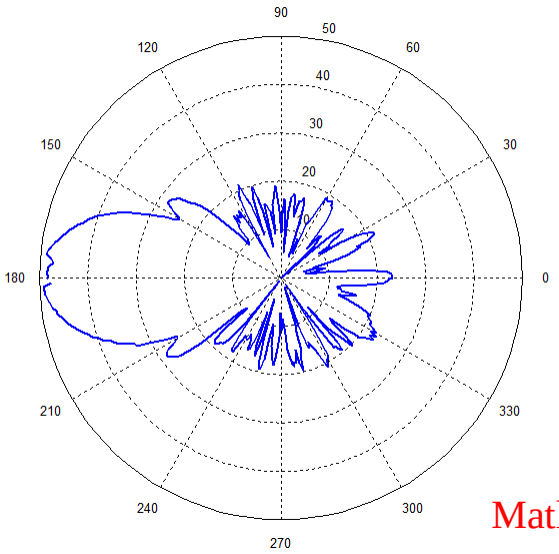
```
>> U = num(:,1);
>> D = num(:,2);
```

Cvičení #1/02

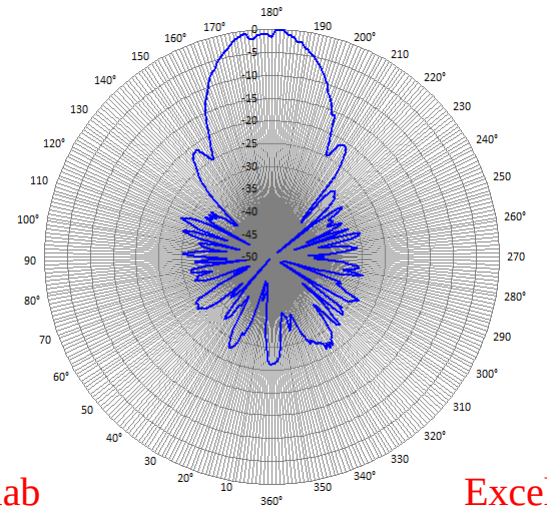
- dále můžeme s daty pracovat dle libosti
  - např. úprava jako v Excelu, se stejným výstupem:

```
>> Ur = pi*U/180;          % polar vykresluje podle úhlu (0:2*pi)
>> Dr = D + max(abs(D)); % polar vykresluje argument jako
>>                         % vzdálenost od osy (= kladné číslo!)
>> figure; polar(Ur,Dr); % vykreslení polárního grafu
```

vidíte, že v Matlabu je jiný popis osy (v kontextu anténní techniky nesprávný → musíme event. použit jiný typ grafu (A.P.P.)



Matlab



Excel

Cvičení #1/03

- graf `polar` nám stále zcela nevyhovuje
  - chceme hlavní lalok ve směru osy *y*
  - správně cejchovaná je osa v případě grafu z Excelu

- možnosti:
  - nová funkce pro polární graf (obtížné, avšak ne nemožné)
  - najdeme vyhovující alternativu:

`http://www.mathworks.com/matlabcentral/fileexchange/`

  - do vyhledávače napíšeme např. „`polar plot`“
  - a seřadíme např. podle hodnocení od nejlepšího

Cvičení #1/04

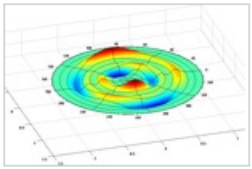
updated 4 years ago

Bullseye Polar Data Plot by Daniel Ennis

This function creates smooth patches of polar data in "bullseye" plot. (bullseye, polar, plotting)

fx [h,t]=bullseye(data,varargin);

5 Ratings
6 Comments
10 Downloads (30 Days)



updated 6 days ago

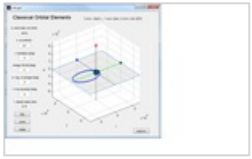
3D Polar Plot by Ken Garrard

Plots 3d polar data with polar axis and polar grid (polar, plot, specialized)

fx polarplot3d(Zp,varargin)

polarplot3d_demo.m

5 Ratings
10 Comments
141 Downloads (30 Days)



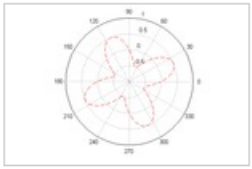
updated 1 year ago

Classical Orbital Elements GUI by Anders Edfors Vannevik

Visualize Classical Orbital Elements (aerospace, aeronautics, aerodef)

coe_gui(varargin)

3 Ratings
2 Comments
28 Downloads (30 Days)



updated 2 years ago

Polar 2 by Daniel Armyr

An update to Matlabs built-in polar.m (graph types, plotting, graphics)

fx polar2(varargin)

2 Ratings
12 Comments
63 Downloads (30 Days)

updated 2 years ago

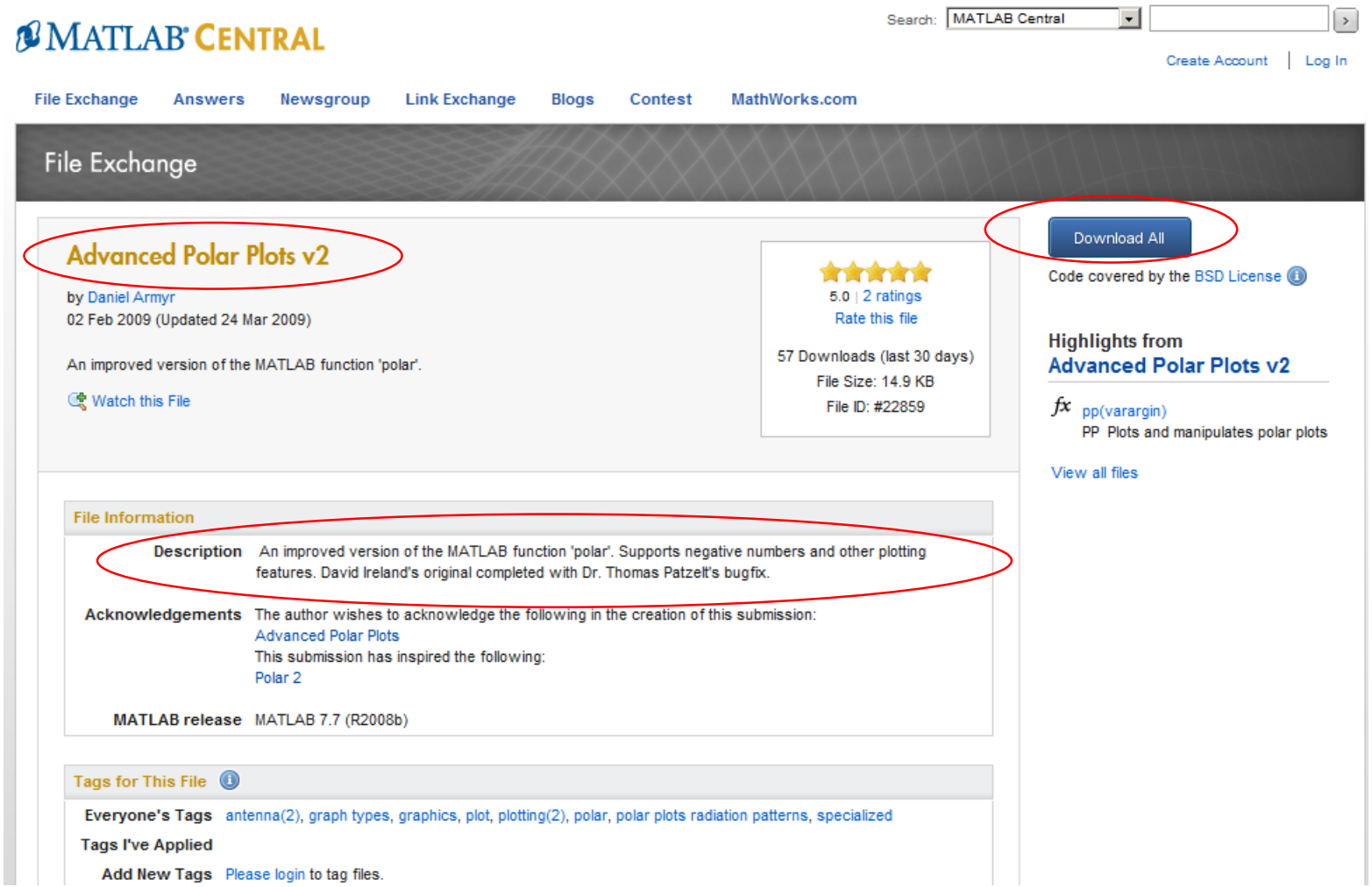
Advanced Polar Plots v2 by Daniel Armyr

An improved version of the MATLAB function 'polar'. (antenna, plotting, graphics)

fx pp(varargin)

2 Ratings
18 Comments
57 Downloads (30 Days)

Cvičení #1/05

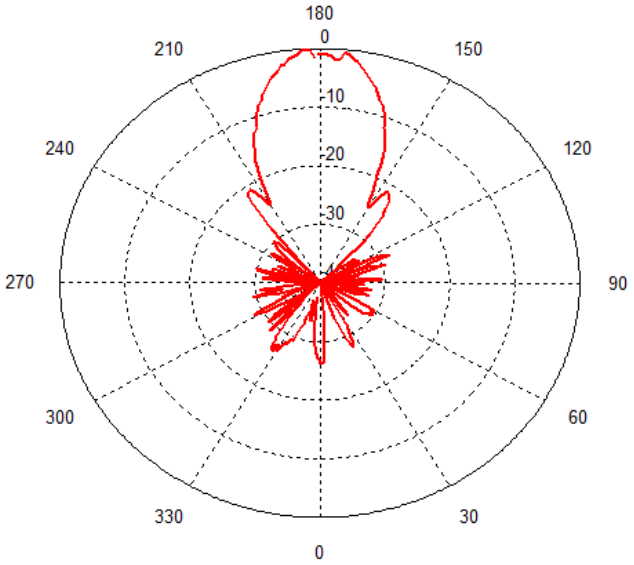

MATLAB CENTRAL
Search: MATLAB Central
Create Account | Log In
File Exchange Answers Newsgroup Link Exchange Blogs Contest MathWorks.com
File Exchange
Advanced Polar Plots v2
by Daniel Armyr
02 Feb 2009 (Updated 24 Mar 2009)
An improved version of the MATLAB function 'polar'.
Watch this File
5.0 | 2 ratings
Rate this file
57 Downloads (last 30 days)
File Size: 14.9 KB
File ID: #22859
Download All
Code covered by the BSD License
Highlights from
Advanced Polar Plots v2
pp(varargin)
PP Plots and manipulates polar plots
View all files
File Information
Description An improved version of the MATLAB function 'polar'. Supports negative numbers and other plotting features. David Ireland's original completed with Dr. Thomas Patzelt's bugfix.
Acknowledgements The author wishes to acknowledge the following in the creation of this submission:
Advanced Polar Plots
This submission has inspired the following:
Polar 2
MATLAB release MATLAB 7.7 (R2008b)
Tags for This File
Everyone's Tags antenna(2), graph types, graphics, plot, plotting(2), polar, polar plots radiation patterns, specialized
Tags I've Applied
Add New Tags Please login to tag files.

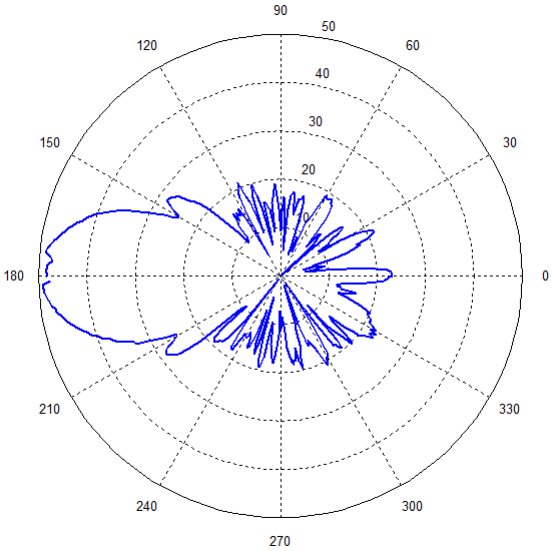
Cvičení #1/06

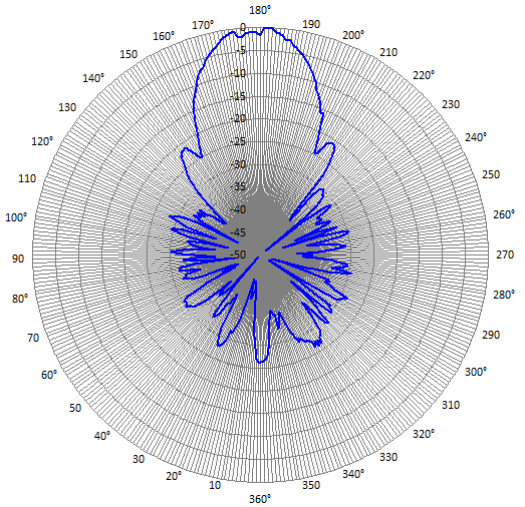
- stáhneme soubor
- podíváme se na nápovědu

```
>> help pp
```

```
>> pp(Ur,D,[-40 0],'MagMarkAngle',90,'ThetaStartAngle',270, ...
     'LineWidth',2,'LineColor','r','FigureBackgroundColor','w');
```

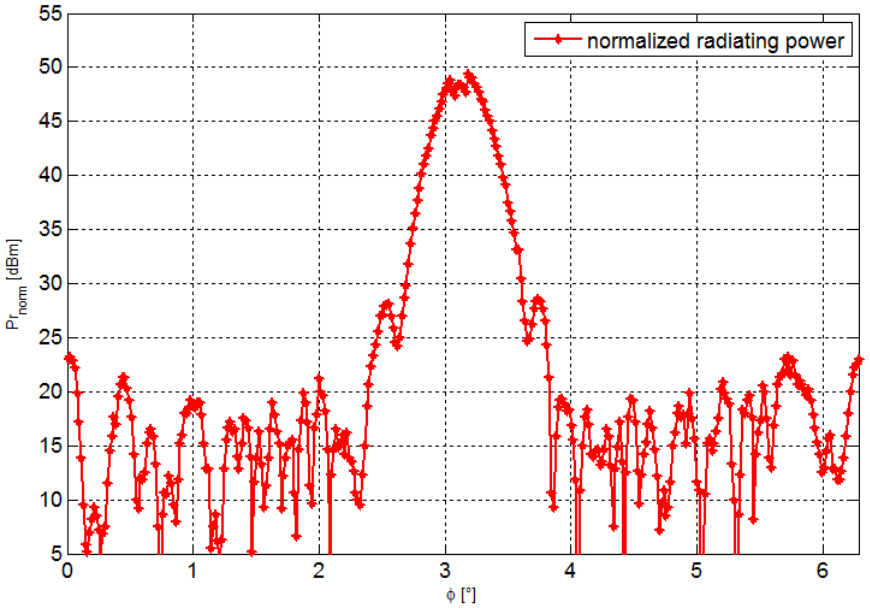






Cvičení #1/07

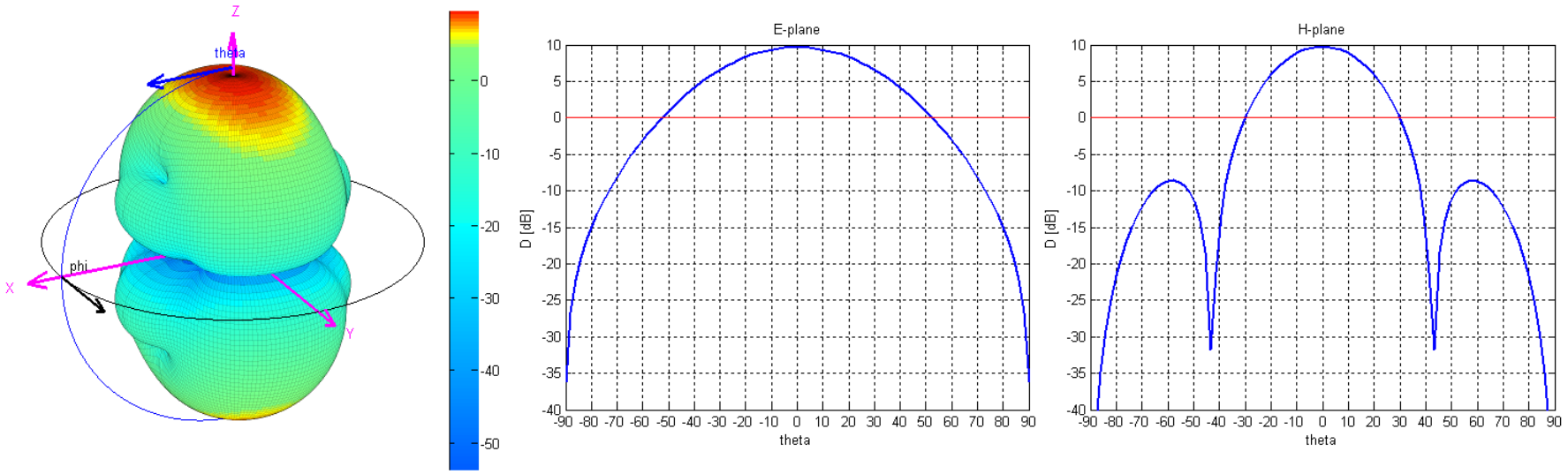
- ukažme si ještě pokročilejší úpravy
- využijeme klasický *x*-*y* graf




```
>> figure('Color','w','Position',[100 100 850 550],'MenuBar','none');
>> plot(Ur,PrdB,'*-r','LineWidth',2);
>> grid on;
>> xlim([0 2*pi]);
>> ylim([5 55]);
>> xlabel('\phi [°]');      % Matlab respektuje i vstup pomocí TeXu
>> ylabel('Pr_{norm} [dBm]');
>> legend('normalized radiating power');
>> set(gca,'FontSize',13);
```

Cvičení #1/08

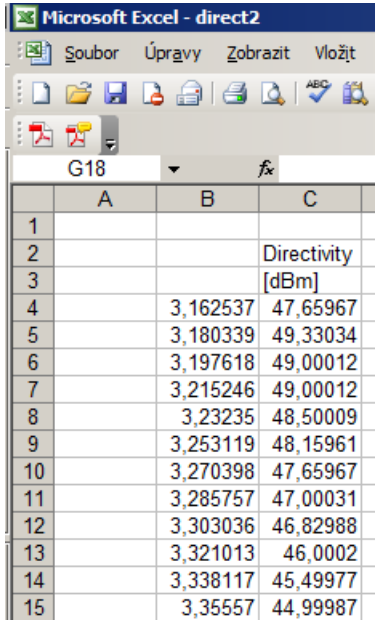
- naše vlastní řešení, vzniklé na katedře elmag.pole:
  - (Ing. Jan Eichler)



Cvičení #1/09

- opačně, můžeme rovněž zapisovat (i do nového) `xls` souboru
  - zapisujeme postupně vybraná data

```
>> d = {'Directivity';'[dBm]'};
>> xlswrite('direct2.xls',d,2,'C2');
>> xlswrite('direct2.xls',Ur,2,'B4');
>> xlswrite('direct2.xls',Dr,2,'C4');
```



	A	B	C
1			
2			Directivity
3			[dBm]
4		3,162537	47,65967
5		3,180339	49,33034
6		3,197618	49,00012
7		3,215246	49,00012
8		3,23235	48,50009
9		3,253119	48,15961
10		3,270398	47,65967
11		3,285757	47,00031
12		3,303036	46,82988
13		3,321013	46,0002
14		3,338117	45,49977
15		3,35557	44,99987

Cvičení #1/10

- k načítání využijeme funkci `importdata`
  - můžete však využít již zmíněnou funkci `uiimport`

```
>> TXT = importdata(...
          '39364,7134632176.txt');
```

jméno souboru (Matlab ho musí vidět)

```
>> TXT
```

```
>> TXT = importdata(...
          '39364,7134632176.txt');
>> TXT

TXT =

          data: [361x3 double]
      textdata: {3x3 cell}
    colheaders: {'uhel[°]'  'vykon[W]'  'faze[°]'}

>>
```

39364,7134632176 – Poznámkový blok

Soubor Úpravy Formát Zobrazení Nápověda

```
WinEman 2.4.1

uhel[°]          vykon[W]          faze[°]

0.59      1.469000000000000E-0009      0.000000000000000E+0000
1.21      1.524000000000000E-0009      0.000000000000000E+0000
2.21      1.413000000000000E-0009      0.000000000000000E+0000
3.39      1.211000000000000E-0009      0.000000000000000E+0000
4.39      7.079000000000000E-0010      0.000000000000000E+0000
5.37      3.828000000000000E-0010      0.000000000000000E+0000
6.37      1.778000000000000E-0010      0.000000000000000E+0000
7.41      6.561000000000000E-0011      0.000000000000000E+0000
8.24      2.818000000000000E-0011      0.000000000000000E+0000
9.22      2.415000000000000E-0011      0.000000000000000E+0000
10.27      3.690000000000000E-0011      0.000000000000000E+0000
11.27      4.819000000000000E-0011      0.000000000000000E+0000
12.27      6.310000000000000E-0011      0.000000000000000E+0000
13.26      5.212000000000000E-0011      0.000000000000000E+0000
14.26      3.828000000000000E-0011      0.000000000000000E+0000
15.24      2.239000000000000E-0011      0.000000000000000E+0000
16.30      3.548000000000000E-0011      0.000000000000000E+0000
17.36      4.140000000000000E-0011      0.000000000000000E+0000
18.25      1.040000000000000E-0010      0.000000000000000E+0000
19.23      2.075000000000000E-0010      0.000000000000000E+0000
20.25      2.818000000000000E-0010      0.000000000000000E+0000
21.29      4.295000000000000E-0010      0.000000000000000E+0000
22.27      3.690000000000000E-0010      0.000000000000000E+0000
23.30      6.561000000000000E-0010      0.000000000000000E+0000
24.28      8.570000000000000E-0010      0.000000000000000E+0000
25.27      1.000000000000000E-0009      0.000000000000000E+0000
26.25      1.000000000000000E-0009      0.000000000000000E+0000
27.28      7.943000000000000E-0010      0.000000000000000E+0000
28.32      6.067000000000000E-0010      0.000000000000000E+0000
29.30      4.295000000000000E-0010      0.000000000000000E+0000
30.34      1.919000000000000E-0010      0.000000000000000E+0000
31.32      7.362000000000000E-0011      0.000000000000000E+0000
32.34      6.067000000000000E-0011      0.000000000000000E+0000
33.32      1.167000000000000E-0010      0.000000000000000E+0000
```

Cvičení #1/11

- přiřazení dat, následují identické operace jako z Excelu

```
>> U    = TXT.data(:,1);
>> P    = TXT.data(:,2);
>> P(10:20)

>> Ur   = pi*U/180;
>> PdB  = 10*log10(P*1e3);
>> PrdB = PdB + max(abs(PdB));
>> figure; polar(Ur,PrdB);
```

Cvičení #2/01

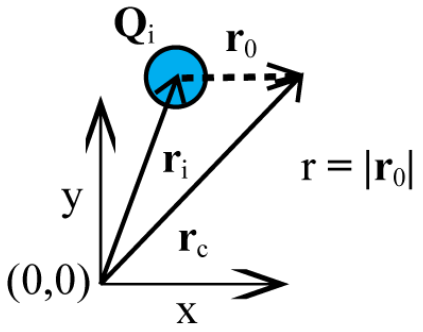
- vytvořte skript (funkci), který zobrazí elektrické pole od elektrických nábojů
  - je třeba nalézt vhodné rovnice pro výpočet grafu funkce (elmag. pole)
  - musíme uvážit fyzikální pozadí celé úlohy

- řešení
- lze využít Colombova zákona pro bodový náboj:

$$\mathbf{E}(r) = \frac{Q}{4\pi\varepsilon} \cdot \frac{1}{r^2} \cdot \mathbf{r_0} \qquad \text{kde vzdálenost} \qquad r = |\mathbf{r_c} - \mathbf{r}_i|$$

- snazší je potom výpočet pomocí potenciálu, neboť

$$\mathbf{E} = -\nabla\varphi \qquad \varphi(r) = \frac{Q}{4\pi\varepsilon} \cdot \frac{1}{r} + K$$



Cvičení #2/02

- máme-li více (n) nábojů, využijeme principu superpozice:

$$\mathbf{E}(\mathbf{r}) = \sum_{i=1}^{n} \mathbf{E}_i(\mathbf{r}) \qquad \text{a tedy} \qquad \varphi(r) = \sum_{i=1}^{n} \frac{Q_i}{4\pi\varepsilon} \cdot \frac{1}{|\mathbf{r_c} - \mathbf{r}_i|}$$

- pro dva náboje (Q_1, Q_2):

$$\varphi(r) = \frac{Q_1}{4\pi\varepsilon} \cdot \frac{1}{|\mathbf{r_c} - \mathbf{r}_1|} + \frac{Q_2}{4\pi\varepsilon} \cdot \frac{1}{|\mathbf{r_c} - \mathbf{r}_2|}$$

- což lze upravit pro známé (=zadané) polohy nábojů $[x_1,y_1]$ a $[x_2,y_2]$, umístěné ve vakuu:

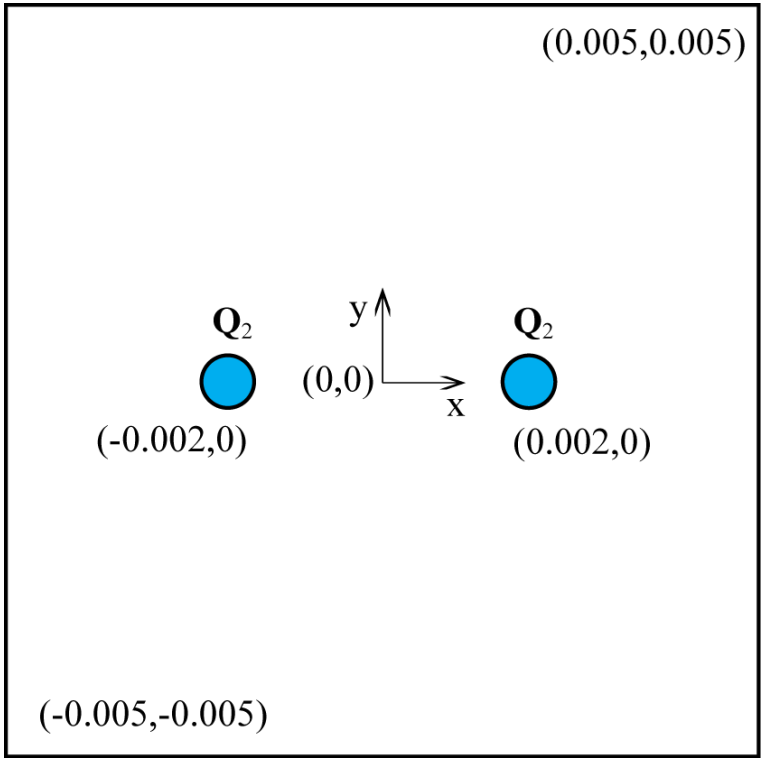
$$\boxed{\varphi(r) = \frac{1}{4\pi\varepsilon_0} \cdot \left(\frac{Q_1}{\sqrt{(x_c - x_1)^2 + (y_c - y_1)^2}} + \frac{Q_2}{\sqrt{(x_c - x_2)^2 + (y_c - y_2)^2}} \right)}$$

Cvičení #2/03

- co dále neznáme (a potřebujeme znát)?
  - permitivitu vakua: $\varepsilon_0 = 8.854 \cdot 10^{-12}\,\mathrm{Fm}^{-1}$
  - velikost elementárního náboje: $e = 1.6022 \cdot 10^{-19}\,\mathrm{C}$
  -
  - (naše náboje ponecháme o uvedené velikosti, tj. $Q_1 = +e$; $Q_2 = \pm e$)
  - pozice 1. náboje (např.): $x_1 = 2\,\mathrm{mm}$; $y_1 = 0\,\mathrm{mm}$;
  - pozice 2. náboje (např.): $x_2 = -2\,\mathrm{mm}$; $y_2 = 0\,\mathrm{mm}$;
  - velikost gridu: $\langle -5, 5\rangle \times \langle -5, 5\rangle$ mm s krokem 1/3 mm

Cvičení #2/04

- takto máme definovanou úlohu:



pozn.: Protože dělíme čísla, která jsou velikostí velmi rozdílná, je potřeba dát si pozor na využitelnou velikost mantisy (opominutí tohoto faktu může vést k nekontrolovatelnému šíření numerické chyby ve výpočtu).

$$\mathbf{E}(x_c, y_c) = -\frac{1}{4\pi\varepsilon_0} \cdot \nabla \left(\frac{Q_1}{\sqrt{(x_c - x_1)^2 + (y_c - y_1)^2}} + \frac{Q_2}{\sqrt{(x_c - x_2)^2 + (y_c - y_2)^2}} \right)$$

Cvičení #2/05

- Matlab kód (srovnejte postup s předchozí úlohou)
  - otevřete si nový skript
  - vymazání obrazovky, definice konstant

```
clear, clc, close all;
x1 = 2e-3;  y1 = 0;
x2 = -2e-3; y2 = 0;

e0   = 1.6022e-19;
q1   = +e0;
q2   = -e0;
eps0 = 8.854e-12;
```

  - tvorba gridu pro výpočet grafu funkce

```
t = (-5:1/3:5).*1e-3;
[x,y] = meshgrid(t);
```

Cvičení #2/06

- výpočet super pozice:

```
z = q1./(eps0*4*pi*sqrt((x-x1).^2+(y-y1).^2)) + ... % pole 1. náboje
    q2./(eps0*4*pi*sqrt((x-x2).^2+(y-y2).^2));      % pole 2. náboje
```

- výpočet elektrické intenzity **E**

```
[gx gy] = gradient(z);
```

Cvičení #2/07

- ošetření nefyzikálního průběhu v okolí bodového náboje: všimněte si, že gradient zde vychází nekonečný, což je způsobeno následující singularitou (v reálu problém nenastává)

```
gx(gx == +Inf) = NaN;
gy(gy == +Inf) = NaN;
gx(gx == -Inf) = NaN;
gy(gy == -Inf) = NaN;
```

$$\lim_{\substack{x_1 \to x_c \\ y_1 \to y_c}} \frac{1}{\sqrt{(x_c - x_1)^2 + (y_c - y_1)^2}} \approx \frac{1}{0} \to \infty$$

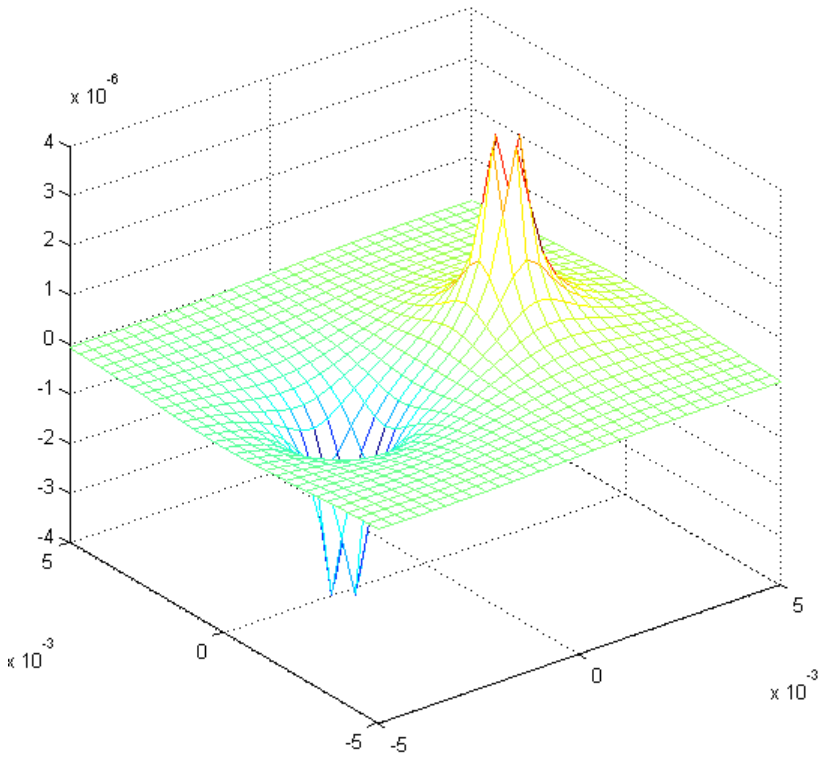
- vykreslení potenciálu a intenzity elektrického pole:

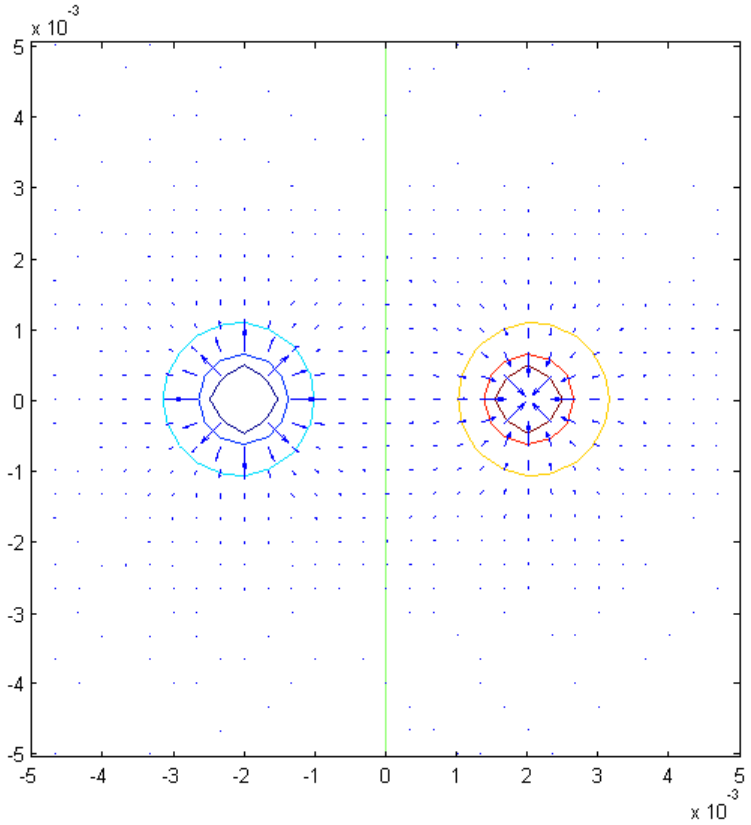
```
figure('pos',[50 50 1300 700]);
subplot(1,2,1);
mesh(x,y,z);
```

```
subplot(1,2,2);
contour(x,y,z);
hold on;
quiver(t,t,gx,gy);
```

Cvičení #2/08

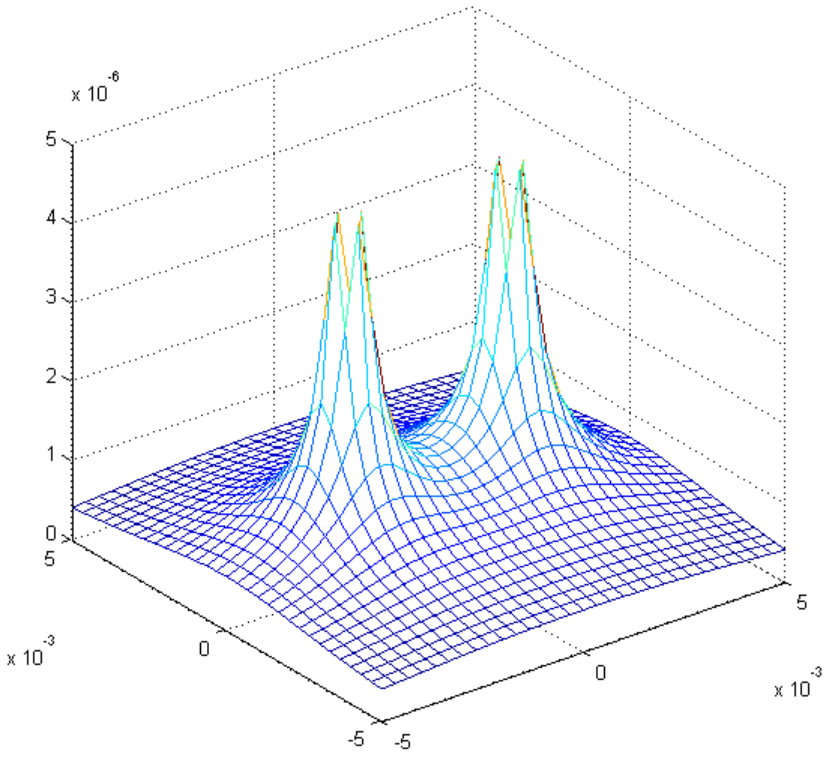
- opačná polarita nábojů:

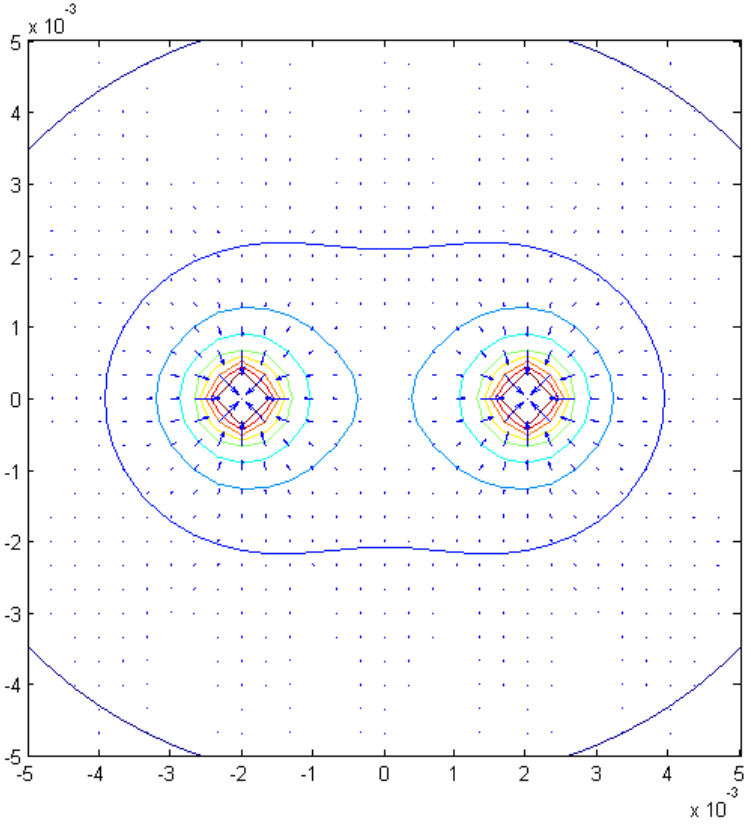




Cvičení #2/09

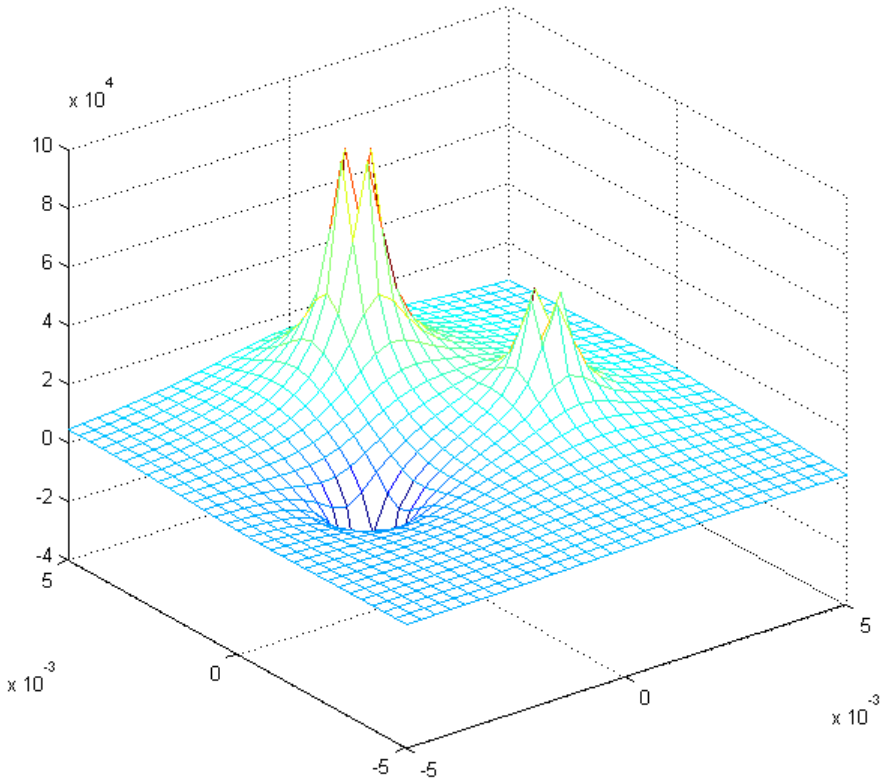
- shodná polarita nábojů:

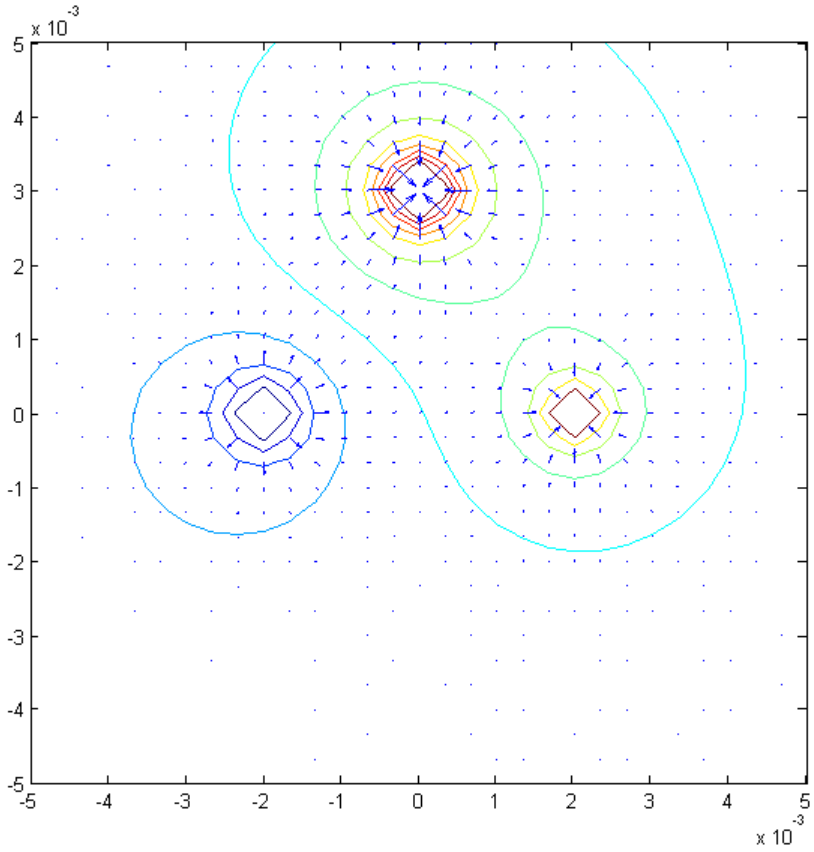




Cvičení #2/10

- a konečně pro případ tří (různě velikých) nábojů:





Cvičení #3/01

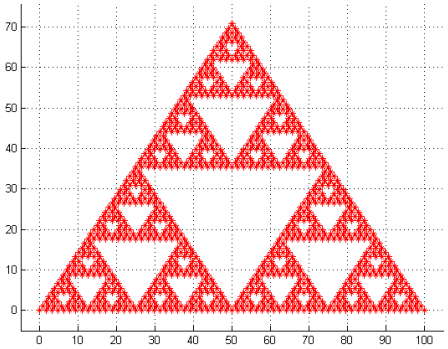
- vytvořte generátor IFS fraktálů
  - co to je IFS fraktál a jak se generuje najdete na internetu
  - pro efektivní generaci využijte jako vstup množiny bodů a transformací

  - celý problém je složitý a na první pohled nerealizovatelný; zkusme si ho tedy rozdělit na jednotlivé (= jednoduše zvládnutelné) etapy!
    - které navrhujete??

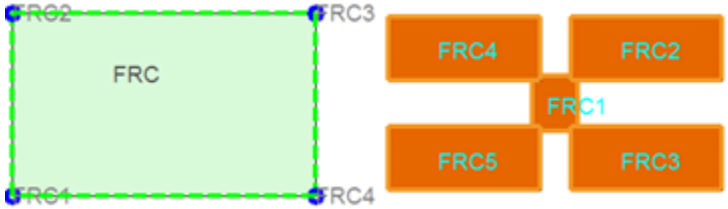
Cvičení #3/02

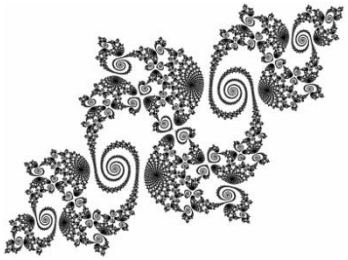
- 1) jaký je cíl naší práce – teoretický rozbor, co chceme vyřešit
  - zjistíme, co jsou pojmy: fraktál, polygon, iterace
  - co je to IFS (*Iterated Function System*)
  - jakým způsobem převedeme formální zápis generace IFS do Matlabu
- 2) jak bude vypadat plánovaný kód
  - více separátních částí?
  - jaký bude vstupní a výstupní formát dat?
  - jak budou vypadat hlavičky funkcí, budeme ošetřovat vstup (a na co)?
- 3) návrh a implementace programu
  - od jednoduchého po složitější
  - ošetření vstupů
  - test funkcí (funkčnost, rychlost, nevhodné vstupy)
- 4) další vylepšení základní verze

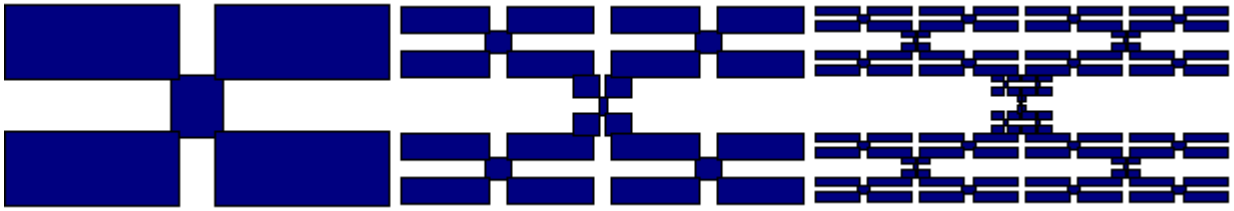
Cvičení #3/03



- Ad 1) jaký je cíl naší práce?
  - zjistíme, co je to fraktál (wiki, internet)
  - co je to IFS?







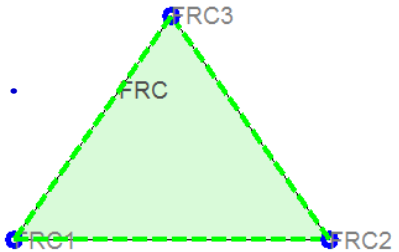
  - jakou můžeme zvolit notaci, odpovědi např. zde:
    `http://www.elmag.org/doku.php/wiki:user:capek:ifsmaker`

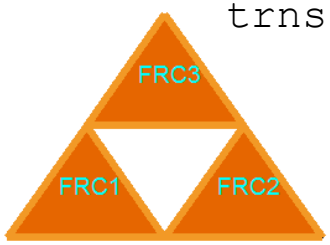
Cvičení #3/04

- formát vstupních proměnných (polygon, transformace):

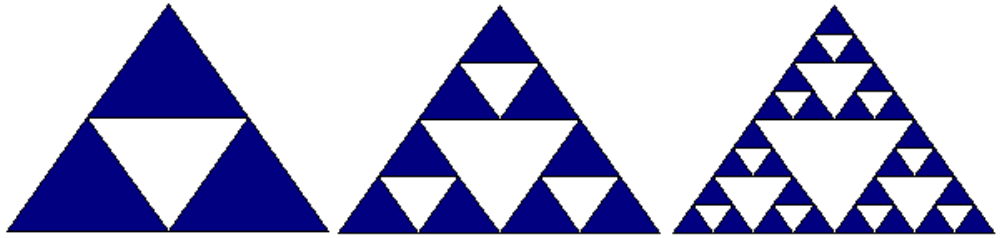
Proměnná	Velikost	Popis
`pts`	(*n*,2)	pole bodů základního útvaru (*n* je počet bodů)
`trns`	(*m*,6)	pole afinních transformací (*m* je počet transformací)
`iter`	(1,1)	počet iterací (celočíselný kladný skalár)

```
pts = [0 0; ...
       100 0; ...
       50 √2*50]
```






```
trns = [0.5 0 0 0.5 0 0; ...
        0.5 0 0 0.5 50 0; ...
        0.5 0 0 0.5 25 √2*25]
```



`iter = 1;` `iter = 2;` `iter = 3;`

Cvičení #3/05

- Ad 2) jak bude vypadat plánovaný kód?
  - více separátních částí
    - část (A) bude generovat fraktál, výsledkem bude množina polygonů
    - část (B) bude fraktál vykreslovat
  - jaký bude vstupní a výstupní formát dat?
    - část (A) vyžaduje množinu vstupních bodů (počáteční polygon), množinu afinních transformací a počet iterace (proměnné `pts`, `trns`, `iter`)
    - výstupem (A) bude 3D pole polygonů, které reprezentují výslednou fraktální koláž, označme tuto proměnnou např. `IFSfractal`
    - vstupem (B) bude výstup (A), tj. `IFSfractal`
    - výstupem (B) bude handle na figure, ve kterém bude fraktál vykreslen

Cvičení #3/06

- Ad 2) jak bude vypadat plánovaný kód?
  - jak budou vypadat hlavičky funkcí, budeme ošetřovat vstup (a na co)?

```
function IFSfractal = gen_ifs_fractal(pts,trns,iter)
%%GEN_IFS_FRACTAL: Generates iterated-function-system (IFS) motif
%
% program na generaci IFS koláže
```

```
function hndl = draw_fractal(IFSfractal)
%%DRAW_FRACTAL: Plots IFS collage (3D polygon array)
%
% program vykresluje vygenerovanou IFS koláž
```

Cvičení #3/07

FUNKCE (A)

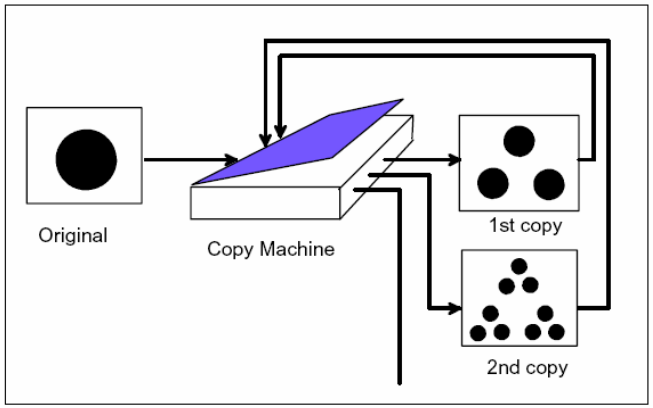
```
IFSfractal = gen_ifs_fractal(pts,trns,iter)
```

```
function IFSfractal = gen_ifs_fractal(pts,trns,iter)
%%GEN_IFS_FRACTAL: Generates iterated-function-system (IFS) motif
%
% program na generaci IFS koláže
```

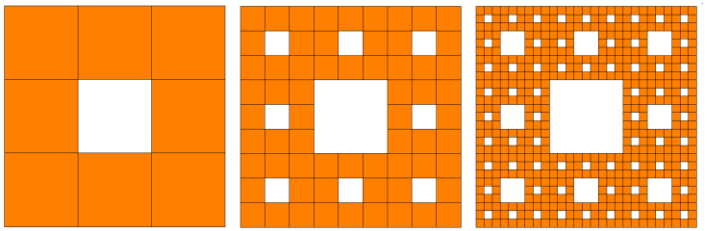
Cvičení #3/08

- Ad 3) návrh a implementace programu: **generace IFS**
  - od jednoduchého po složitější:
    PSEUDOKÓD:

```
(01) pro_vsechny_iterace
(02)     pro_kazdy_polygon
(03)         pro_kazdy_bod
(04)             generuj_novy_bod
(05)         end
(06)     end
(07) end
```



$$w : \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$



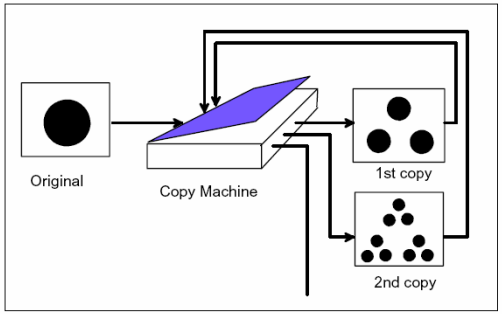
Cvičení #3/09

- Ad 3) návrh a implementace programu: **generace IFS**

  VYLEPŠENÝ PSEUDOKÓD (rozbor):

```
(01) inicializace IFSfractal
(02) pro_vsechny_iterace
(03)     vyber_polygony_posledni_iterace
(04)     pro_kazdy_polygon_posledni_iter
(05)         pro_kazdou_transformaci
(06)             pro_kazdy_bod_daneho_polygonu
(07)                 generuj_novy_bod
(08)                 /existuje_novy_bod/
(09)             end
(10)         end
(11)         /existuje_novy_polygon/
(12)     end
(13)     /existuje_nova_iterace/
(14)     uloz_ji_nakonec
(15) end
```

$$w : \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$



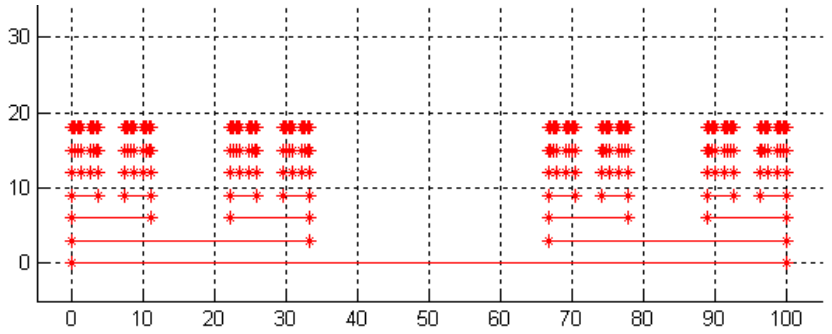
Cvičení #3/10

- Ad 3) návrh a implementace programu: **generace IFS**
- závěry:
  - potřebujeme kromě `IFSfractal` ještě jednu dočasnou proměnnou, kde se budou postupně ukládat (a znovu načítat) polygony ze související iterace
  - velikost této proměnné musí být inicializována před začátkem generace a budou v ní po iteracích uloženy všechny polygony
  - toto lze jednoduše realizovat pomocí datového typu cell, označme tedy naší proměnnou názvem `Cell`
  - poslední iteraci na závěr uložíme do `IFSfractal` a tím získáme naší koláž

- otázka je: jak vyzrát na `cell`?

Cvičení #3/11

- Ad 3) návrh a implementace programu: **generace IFS**
- nyní je vhodné pseudokód projít a pro jednoduchý případ otestovat, zda bude koncept fungovat – nechceme přeci naprogramovat hloupost
  - vybereme si např. Cantorovo mračno (objevil ho B. Mandelbrot a popisuje mj. rozložení rušení na telefonní lince)
  - základní objekt: 2 body (1 úsečka), tedy:
    ```
    pts = [0 0; 100 0]
    ```
  - transformace: zmenšení na 1/3 + posun, 2×6 koeficientů, tedy:
    ```
    trns = [1/3 0 0 0 0 3; 1/3 0 0 0 200/3 3]
    ```
  - iterace:
    - `iter = 5`



Cvičení #3/12

```
Cell =

    [2x2   double]
    [2x2x2 double]
    []
    []
    []
```

```
Cell =

    [2x2   double]
    [2x2x2 double]
    [2x2x4 double]
    []
    []
```

```
Cell =

    [2x2   double]
    [2x2x2 double]
    [2x2x4 double]
    [2x2x8 double]
    []
```

```
Cell =

    [2x2    double]
    [2x2x2  double]
    [2x2x4  double]
    [2x2x8  double]
    [2x2x16 double]
```

- v dokončeném programu by se nám tedy měl takto (nalevo) plnit `Cell` po iteracích (zde 5 iterací, `iter = 5`)
- `IFSfractal` potom obsahuje všechny polygony poslední iterace, tj.

```
IFSfractal = Cell{end};
```

```
>> size(IFSfractal)

ans =

     2     2    16
```

- `Cell` má na prvním místě počáteční polygon (0. iteraci), slouží pro generaci 1. iterace
  - zároveň je velký `iter+1` (všechny iterace + základní objekt)

```
Cell      = cell(iter+1,1);
Cell{1}   = pts;
```

Cvičení #3/13

- Ad 3) návrh a implementace programu: **generace IFS**

  SEGMENTY V MATLABU:

```
for n = 1:iter                                  % po iteracích
    thisIter  = Cell{n};                        % vezmu polygony z minulé iterace
    newPolygs = zeros(size(thisIter,1),2,size(thisIter,3)*trnsSize); % inicializace
    for m = 1:size(thisIter,3)                  % musíme aplikovat na všechny polygony
        thisPolyg(:,:) = thisIter(:,:,m);       % po jednom si je pujčím
        for cur_tr = 1:trnsSize                 % a jedu opět pro všechny transformace
            for cur_pt = 1:size(thisPolyg,1)    % všechny body polygonu
                newPolygs(cur_pt,1,(m-1)*trnsSize+cur_tr) = ...
                    thisPolyg(cur_pt,1,:)*trns(cur_tr,1) + ...
                    thisPolyg(cur_pt,2,:)*trns(cur_tr,2) + trns(cur_tr,5);
                newPolygs(cur_pt,2,(m-1)*trnsSize+cur_tr) = ...
                    thisPolyg(cur_pt,1,:)*trns(cur_tr,3) + ...
                    thisPolyg(cur_pt,2,:)*trns(cur_tr,4) + trns(cur_tr,6);
            end                                 % lze řešit i maticově!
        end
    end
    Cell{1+n} = newPolygs(:,:,:);               % celou vypočtenou iteraci uložíme
end
IFSfractal = Cell{end};                         % zajímá nás (ZDE!) jen poslední iterace
```

Cvičení #3/14

- Ad 3) návrh a implementace programu: **generace IFS**
  - přidáme alokaci `Cell` struktury a `trnsSize`
  - připíšeme hlavičku
  - napíšeme nápovědu funkce
  - funkce: `gen_ifs_fractal.m`
  - ošetření vstupů
    - ošetřujeme podle definice IFS, kterou jsme si zadali, promyslete si
  - test funkce
    - zkuste funkčnost na Cantorovo mračnu, 1. iterace

Cvičení #3/15

FUNKCE (B)

```
hndl = draw_fractal(IFSfractal)
```

```
function draw_fractal(IFSfractal)
%%DRAW_FRACTAL: Plots IFS colage (3D polygon array)
%
% program vykresluje vygenerovanou IFS koláž
```

Cvičení #3/16

- Ad 3) návrh a implementace programu: **vykreslení IFS**
  PSEUDOKÓD:

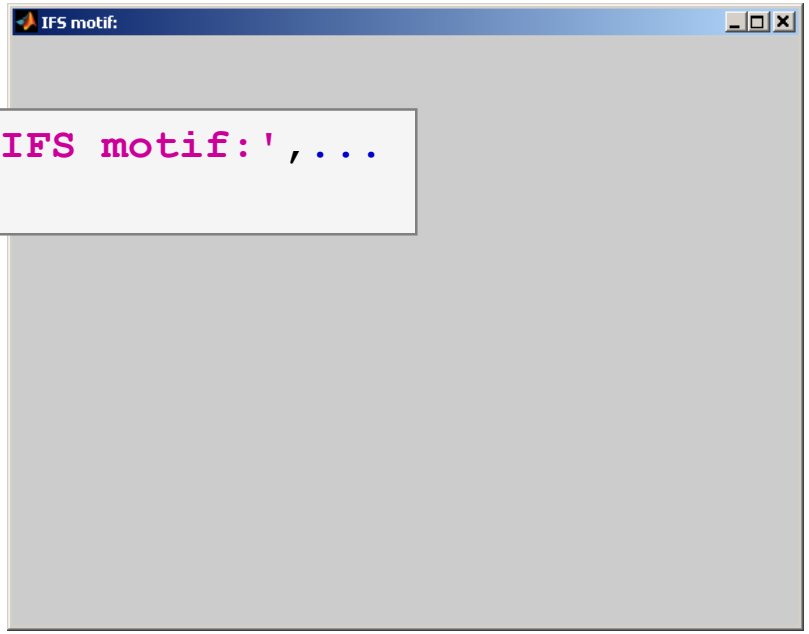
```
function draw_fractal(IFSfractal)
%%DRAW_FRACTAL: Plots IFS colage (3D polygon array)
%
% program vykresluje vygenerovanou IFS kolaz
```

```
(01) vytvoreni okna figure
(02) osa (axis)
(03) hold on
(04) pro_vsechny_polygony
(05)     pro_vsechny_body
(06)         vykresli_usecku
(07)     end
(08) end
```

Cvičení #3/17

- Ad 3) návrh a implementace programu: **vykreslení IFS**
  - zajistěte, ať se nezobrazuje menu
  - zajistěte, až se nezobrazuje číslo okna v záhlaví
  - název okna je „`IFS motif:`“

```
hndl = figure('Units','Pixels','Name','IFS motif:',...
    'NumberTitle','off','Menu','none');
```



Cvičení #3/18

- Ad 3) návrh a implementace programu: **vykreslení IFS**
  - nastavíme rozsah jednotlivých os – tento krok lze vynechat, ale osy jsou při vykreslování postupně zvětšovány!
    - nastavíme proto velikost os fixní hned na začátku
  - vyjdeme ze známých maximálních rozměrů polygonů

```
A = min(min(IFSfractal(:,1,:)));
B = max(max(IFSfractal(:,1,:)));
C = min(min(IFSfractal(:,2,:)));
D = max(max(IFSfractal(:,2,:)));
```

  - a podle těchto rozměrů nastavíme velikost osy

```
axis([A(1)-0.05*B(1) B(1)+0.05*B(1) C(1)-0.05*D(1) D(1)+0.05*D(1)]);
```

Cvičení #3/19

- Ad 3) návrh a implementace programu: **vykreslení IFS**
  - navíc nastavíme barvu pozadí os a umístíme osy do boxu
  - také nastavíme osy na hold on (zafixujeme kreslící plátno)

```
hold on
box on
```

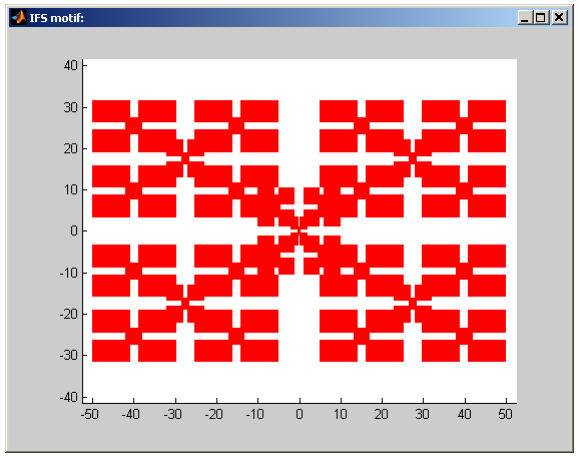
  - vykreslovat budeme po iteracích vždy všechny polygony pomocí funkce `patch`

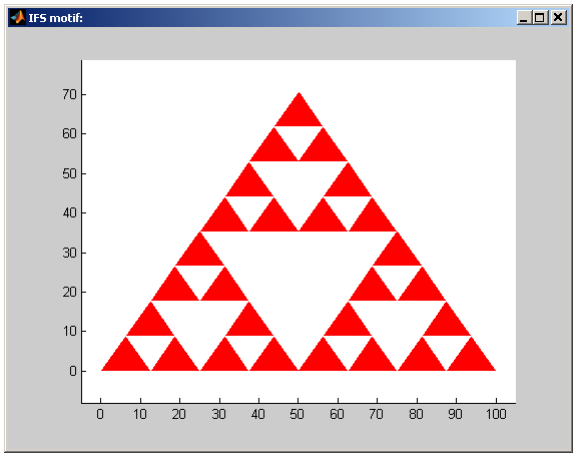
```
for o = 1:size(IFSfractal,3)
    X = IFSfractal(:, 1, o);
    Y = IFSfractal(:, 2, o);
    patch(X, Y, 'r', 'EdgeColor', 'none');
end
```

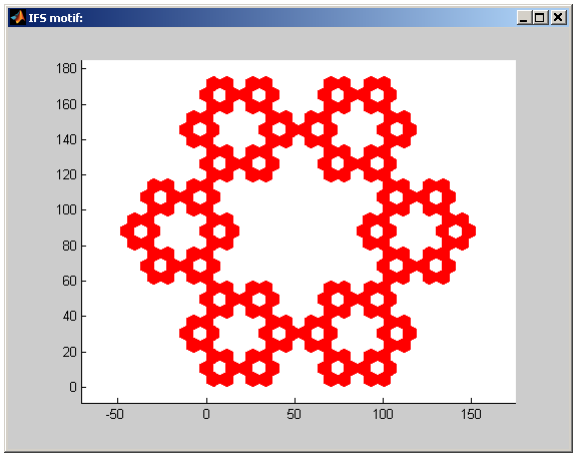
Cvičení #3/20

- nyní máme k dispozici obě funkce
  - (A): `IFSfractal = gen_ifs_fractal(pts,trns,iter)`
  - (B): `draw_fractal(IFSfractal)`

- otestujte na vlastních fraktálech
  - připravená data tří koláží jsou: `IFS1.m`, `IFS2.m` a `IFS3.m`







Cvičení #3/21

- vykreslení okna figure lze dále upravit
  - víte, jak lze získat rozlišení monitoru?
  - nastavte velikost okna figure na 0.3×0.4 násobku rozlišení monitoru a umístěte ho 0.05 násobku rozlišení od levého dolního kraje
  - zvolte si vlastní barvu okna

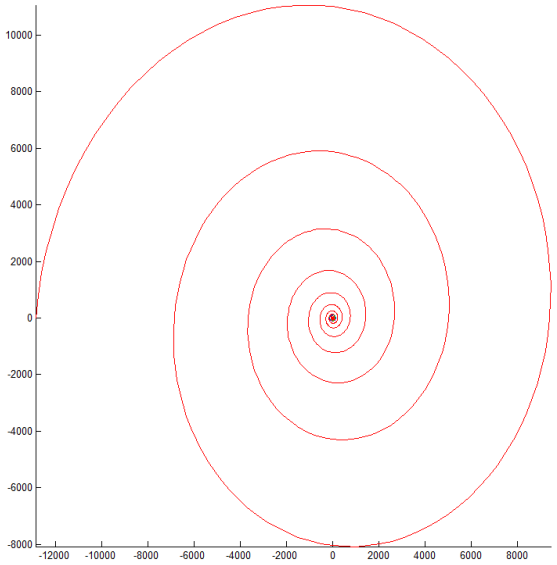
```
monitor = get(0,'Screensize');
hndl = figure('Units','Pixels','Name','IFS motif:',...
    'Position',[0.025*monitor(3) 0.05*monitor(4) 0.3*monitor(3)...
    0.4*monitor(4)],'Color',[0.9 0.9 0.9],...
    'NumberTitle','off','Menu','none');
```

Cvičení #3/22

- další úkoly:
  - zkuste ošetřit vstup proti zadání větších / menších matic, textů atp.
  - zkuste zvýšit rychlost algoritmu (zejm. vykreslovací části)
    - návod: pomocí nástroje `profile` zjistěte, co zabírá většinu času, potřebujete dané části?
  - promyslete, zda není uživatelsky výhodnější použít jednu strukturu pro úplný popis IFS koláže?
    - tip: koukněte se na datový typ `struct` (`>> doc struct`), co Vám nabízí?

Cvičení #4/01

- pomocí příkazu `comet` zobrazte zatáčející se logaritmickou spirálu




```
alpha   = 25*pi:-pi/50:0; % osa (úhly)
a       = 5; % koeficienty
b       = 0.1;
r       = a*exp(b*alpha);
[x,y,z] = sph2cart(alpha,0,r);

figure('Pos',[25 25 850 850],'Color','w');
comet3(x,y,z);
```

Děkuji!

ver. 3.1 (18/05/2015)
Miloslav Čapek
miloslav.capek@fel.cvut.cz