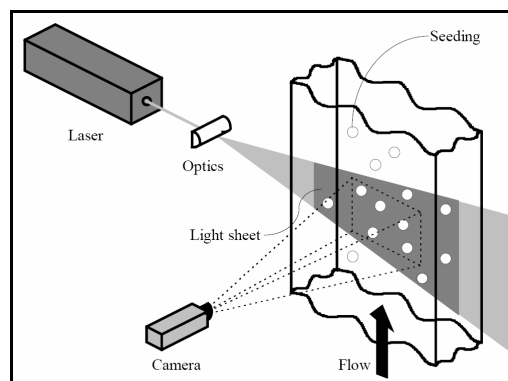


# Vyhodnocení 2D rychlostního pole metodou PIV programem Matlab

(zpracoval Jan Kolínský, dle programu ing. Jana Novotného)

## 1 Obecný popis metody

Particle Image Velocimetry, nebo-li zkráceně PIV, je měřicí metoda fungující na poměrně jednoduchém principu. Tok částic unášených tekutinou je vyfotografován digitální kamerou ve dvou okamžicích bezprostředně jdoucích za sebou. Vzhledem k tomu, že se tyto části za dobu mezi těmito snímky posunou, známe-li časový interval mezi snímky a vzdálenost posunutí, můžeme vypočítat, jak rychle se tyto částice pohybují. Výsledné rychlosti proudění získané z takovéto dvojice snímků jsou polem okamžitých rychlostí v oblasti zachycené kamerou.



Uspořádání měřicí aparatury PIV

Při použití jedné kamery provádíme měření vždy v jedné rovině a výsledkem je tedy pole složek rychlostí v této rovině. Tomuto uspořádání proto říkáme 2D PIV.

## 2 Získání vektorů rychlostí

Dva snímky zachycené po sobě v časovém intervalu  $\Delta t$  jsou označeny jako dvojsnímek či dvojobrázek a v dalším zpracování vytvářejí základní datovou sadu pro vyhodnocení rychlosti proudění.

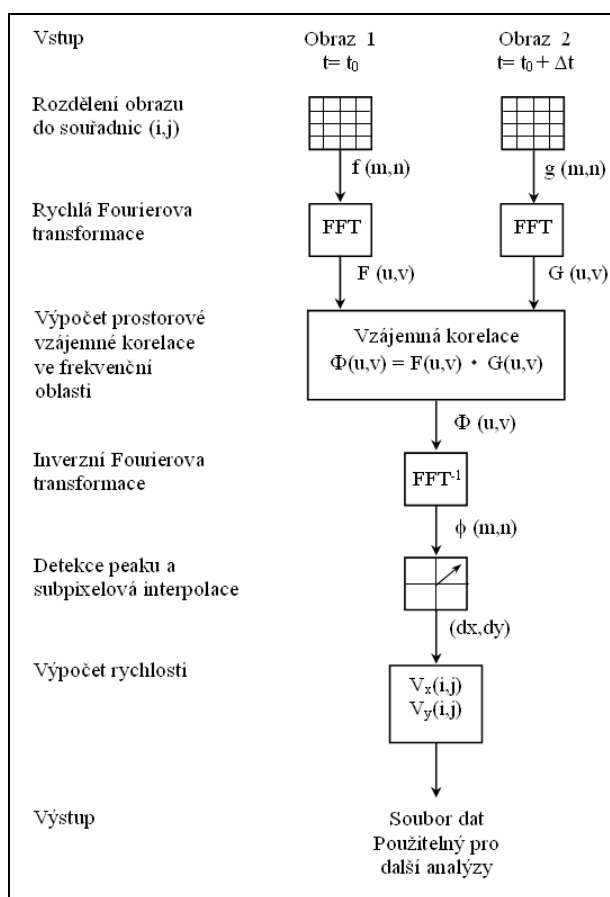
Snímky jsou rozděleny do obdélníkových oblastí, které se nazývají vyšetřované oblasti. Pro každou z těchto vyšetřovaných oblastí obrázek částic zachycený prvním snímkem a obrázek částic zachycených druhým snímkem spoluvytváří náhradní vektor. Ten je vypočítán pomocí cross-corelační analýzy. Výsledkem cross-corelace je plocha peaků v korelační rovině. Každý peak svojí výškou a plochou vůči ostatním určuje, s jakou statistickou pravděpodobností došlo v příslušné vyšetřované oblasti k posunutí částic, které je reprezentováno vektorem s počátečním bodem ve středu nulového peaku (odpovídá počátku souřadnicového systému v korelační rovině) a koncovým ve středu každého dalšího peaku. Výpočet

nejpravděpodobnějšího posunutí se tak zúží na hledání nejvyššího peaku s nejpříznivějšími statistickými parametry.

Protože pro určení vzdáleností obrazu používáme proces rychlé Fourierovy transformace (FFT), neobdržíme kontinuální korelační funkci. Na místo toho máme konečný počet bodů reprezentujících korelační rovinu, kde prostor mezi pixely odpovídá prostoru mezi diskrétními hodnotami FFT funkce. Jelikož známe teoretický tvar křivky v korelační rovině, a proložíme-li tuto známou křivku danými diskrétními body FFT funkce, můžeme přesněji určit střed značkovací částice. Jestliže známe přesně polohu středu každé značkovací částice ve vyšetřované oblasti, můžeme určit vzdálenost mezi prvním a druhým obrázkem částice s přesností menší než jeden pixel pitch. Tento proces se nazývá subpixelová interpolace s jejíž použitím jsme schopni vyhodnotit posunutí částice o velikosti 1/64 pixel pitch.

### 3 Zpracování dat

Na obrázku vidíte schéma algoritmu, kde program nejprve načte dvojici snímků a každému pixelu dle bitové hloubky obrazu přiřadí hodnotu. Poté provede rychlou Fourierovu transformaci těchto polí a ze dvou obdržených funkcí vypočte vzájemnou korelaci a poté na této funkci provede inverzní rychlou Fourierovu transformaci. V takto získané rovině korelačních peaků program vyhledá nejvyšší peak a případně provede subpixelovou interpolaci pro nalezení přesnější polohy maxima. Tímto program vypočte vektor posunutí, který vydělením délkou časového intervalu  $\Delta t$  mezi pořízením jednotlivých snímků a vynásobením rozměrovým měřítkem pixelů vyjádří výslednou rychlost.



Při analýze dat metodou PIV tedy program aplikací korelační funkce na dva snímky vyhodnotí nejpravděpodobnější posunutí v dané oblasti. Teprve poté z posunutí a ze známého časového kroku určí rychlost.

## 4 Jednotlivé kroky programu

### 4.1 Jedna vyhodnocovaná oblast

Pro nejjednodušší případ, kdy nebudeme uvažovat překrývání vyhodnocovaných oblastí a celý čtvercový snímek považujeme za jednu vyhodnocovanou oblast, tedy k získání právě jednoho vektoru posunutí, lze použít následný program bez subpixelové interpolace, jehož kroky jsou tyto:

Pro načtení dvojice snímků do proměnných snímekA a snímekB použijeme příkaz `imread`

```
snimekA=double(imread('a10_01a.tif'));  
snimekB=double(imread('a10_01b.tif'));
```

Určíme velikost (odpovídá velikosti snímku 64 x 64 pixelů)

```
vel=64;
```

provedeme výpočet korelace pomocí funkce `Korelace`, kterou popíšeme následně

```
[c] = Korelace (OB1,OB2,vel);      %výpočet korelace
```

Program `Korelace` je uložen v samostatném matlabovském souboru jako funkce na dvou obrázcích a provádí rychlou Fourierovu transformaci obou snímků a do proměnné `c` vloží korelaci těchto proměnných.

```
function [c] = Korelace (snimekA,snimekB,vel);  
N = 2*vel;  
c = zeros(N,N);  
    ffta=fft2(OB1,N,N);  
    fftb=fft2(OB2,N,N);  
c = real(ifft2(ffta.*fftb));
```

Poté lze najít nejpravděpodobnější polohu posunutí jako maximum z korelační funkce

```
[x,y] = find(c == max(c(:)));      % nalezení celočíselné pozice maxima
```

Vzhledem k tomu, že matlab přiřazuje počátek souřadného systému do středu obrázku je třeba pro složky vektoru posunutí provést korekci

```
posunx=x-vel  
posuny=y-vel
```

Tento program nám umožnil z dvojice snímků vypočítat vektor (`posunx, posuny`) nejpravděpodobnějšího směru posunutí částic mezi těmito snímky.

### 4.2 Více vyhodnocovaných oblastí

Pro případ více vyhodnocovaných oblastí snímku je třeba tento snímek rozdělit na příslušné oblasti a v každé oblasti provést výše zmíněný postup. Bežně se udává velikost vyhodnocované oblasti a procento překrytí jednotlivých oblastí. Celý skript pak má tento tvar, kde je ještě doplněna subpixelová interpolace pro přesnější nalezení vektoru posunutí.

```
snimekA=double(imread('a10_01a.tif'));  
snimekB=double(imread('a10_01b.tif'));
```

```

vel=64; % velikost integracni oblasti
prekryvani=75;
konecX=256;
konecY=256;
krok=uint16(vel-vel*prekryvani/100);
i=0;

for pX=1:krok:konecX-vel+1;
    i=i+1;
    j=0;
    for pY=1:krok:konecY-vel+1;
        j=j+1;
        OB1 = snimekA(pX:pX+vel-1,pY:pY+vel-1); % vyřiznutí podoblasti
        OB2 = snimekB(pX:pX+vel-1,pY:pY+vel-1);
        [c] = Korelace (OB1,OB2,vel);
        [x,y] = find(c == max(c(:))); % nalezení celočíselné pozice maxima
        [x2,y2]=SPinterpolace(c,x,y,vel); % poloha středu peaku vypočtená pomocí sub pixel interpolace
        posunutiX(i,j)=[x2-vel];
        posunutiY(i,j)=[y2-vel];
    end
end
end

```

Funkce SPinterpolace je funkce, která z nepochteného celočíselného maxima  $x,y$  v korelační rovině  $c$  najde přesnější polohu maxima při předpokládaném tvaru peaku. Je zapsaná jako samostatný soubor v tomto tvaru

```

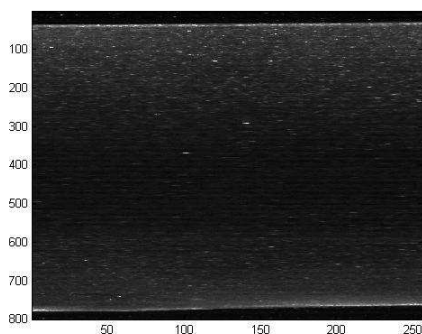
function [x2,y2]=SPinterpolace(c,x,y,vel);
f0 = log(c(x,y));
f1 = log(c(x-1,y));
f2 = log(c(x+1,y));
x2 = x + (f1-f2)/(2*f1-4*f0+2*f2);
f0 = log(c(x,y));
f1 = log(c(x,y-1));
f2 = log(c(x,y+1));
y2 = y + (f1-f2)/(2*f1-4*f0+2*f2);

```

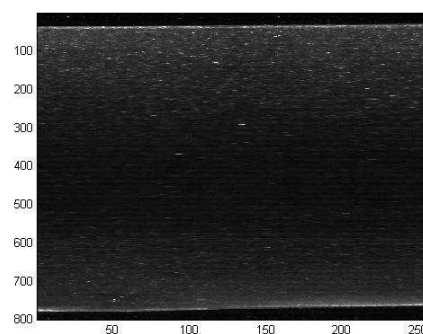
Takto zapsaný program vrací matice vektorů posunutí ve směru  $x$  (posunutiX) a  $y$  (posunutiY).

## 5 Příklad použití programu

Uvedme stručně příklad použití toho programu. Provedeme výpočet posunutí částic mezi těmito dvěma snímky.

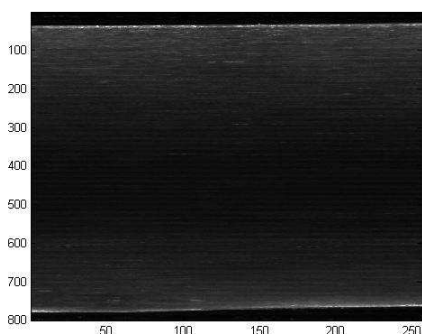


snímek A

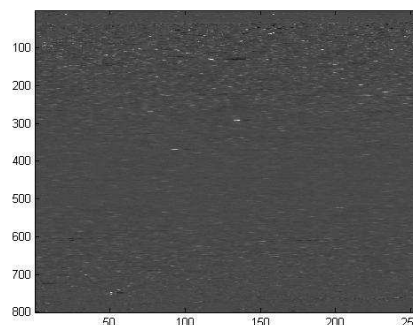


snímek B

Vzhledem k tomu, že máme k dispozici i další snímky z této série, můžeme provést filtraci obrázků pomocí odečtení pozadí. Samotné pozadí získáme jako průměrnou hodnotu ze všech snímků této série, v našem případě zprůměrujeme sedm snímků a toto pozadí odečteme od obou původně vyšetřovaných snímků.

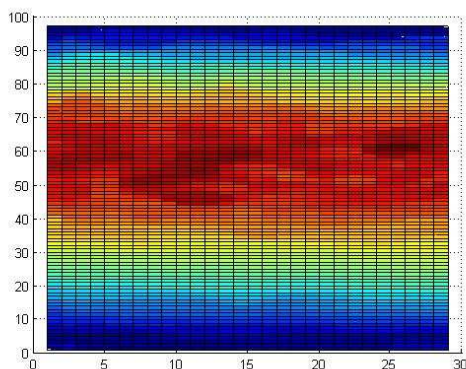


pozadí

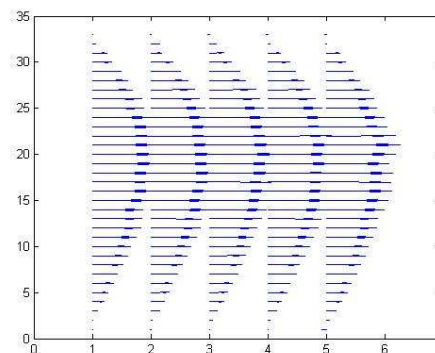


snímek A po odečtení pozadí

Tím získáme snímky, které podrobíme výše zmíněné analýze. Výsledná pole vektorů posunutí můžeme dále filtrovat a různě zobrazovat. Pro ukázkou je zde provedeno provedeno filtrování mediánem (funkce medfilt2) a poté vykreslení vybraných vektorů posunutí šipkami a barevná mapa absolutních velikostí posunutí.



absolutní velikost posunutí



vybrané vektory posunutí

Pro úplnost uvedme ještě celý program včetně filtrování pozadí a filtrování vektorů posunutí až po vykreslení zobrazených grafů.

```
clear all
close all
pozadi=double(imread('aa3_004.bmp'));
pozadi=pozadi+double(imread('aa3_005.bmp'));
pozadi=pozadi+double(imread('aa3_008.bmp'));
pozadi=pozadi+double(imread('aa3_009.bmp'));
pozadi=pozadi+double(imread('aa3_010.bmp'));
pozadi=pozadi+double(imread('aa3_011.bmp'));
pozadi=pozadi+double(imread('aa3_012.bmp'));
pozadi=pozadi./7;
```

```

snimekA=double(imread('aa3_004.bmp'))-pozadi;
snimekB=double(imread('aa3_005.bmp'))-pozadi;

vel=32; % velikost integracni oblasti
prekryvani=75;
konecX=800;
konecY=256;
krok=uint16(vel-vel*prekryvani/100);
i=0;
for pX=1:krok:konecX-vel+1; % zacatek posouvani oblasti IO
    i=i+1;
    j=0;
    for pY=1:krok:konecY-vel+1;
        j=j+1;
        OB1 = snimekA(pX:pX+vel-1,pY:pY+vel-1); % vyřiznutí podoblasti
        OB2 = snimekB(pX:pX+vel-1,pY:pY+vel-1);
        [c] = Korelace (OB1,OB2,vel); % výpočet korelace
        minC=min(min(c));
        c=c+abs(minC)+0.01; % zajištění aby korelační rovina nebyla záporná
        [x,y] = find(c == max(c(:))); % nalezení celočíselné pozice maxima
        [x2,y2]=SPinterpolace(c,x,y,vel);
        posunutiX(i,j)=(x2-vel);
        posunutiY(i,j)=-(y2-vel);
        vektor(i,j)=sqrt(posunutiX(i,j)^2+posunutiY(i,j)^2);
    end
end

posunutiY = medfilt2(posunutiY,[3 3])
posunutiX = medfilt2(posunutiX,[3 3])
vektor = medfilt2(vektor,[3 3])

figure(1)
quiver(posunutiY(1:3:end,1:6:end),posunutiX(1:3:end,1:6:end),1)
figure(2)
surf (vektor); figure(gcf)

% Konec programu.

```