Rekonstrukce simulovaných dat DCE-MRI

Implementace L+S modelu **skript\_simulovane**

Ondřej Mokrý, [xmokry12@vut.cz](mailto:xmokry12@vut.cz)

# Podvzorkování

## Obraz

V rámci úspory času a paměti se rekonstruuje menší obraz, než je ten simulovaný. Původní obraz je rozměru 1024 × 1024 px, rekonstruovaný obraz je v obou rozměrech zmenšený v měřítku 1 : factor.

## Souřadnice v k-prostoru

V simulátoru pracujeme se souřadnicemi v k-prostoru v rozsahu[[1]](#footnote-1)

0.5\*[-Nsamples/xsize, Nsamples/xsize],[[2]](#footnote-2)

kde Nsamples je počet vzorků na jednu radiálu, xsize je rozměr obrázku (ve stávající verzi je Nsamples = 128 a xsize = 1024).

V rekonstrukci však pracujeme se souřadnicemi, které jsou v rozsahu

0.5\*[-Nsamples/(xsize/factor), Nsamples/(xsize/factor)],

tedy vzhledem k rekonstruovanému obrázku pokrývají větší část k-prostoru. Ve skutečnosti však tyto souřadnice odpovídají stejným frekvencím jako v simulaci, protože podvzorkováním obrazu se jeho spektrum „ořízne“, nikoliv jenom zmenší (ilustruje to skript testsubsample).

# Normalizace citlivostí

K rekonstrukci se používají citlivosti ze simulátoru. A priori nejsou normalizované, tudíž se po načtení normalizují a následně se podvzorkují na velikost výstupního obrazu. Normalizaci je třeba vnímat následovně: Pro každý pixel odpovídající citlivosti tvoří vektor takové délky, kolik cívkových elementů je použito pro (simulaci) snímání dat; tyto vektory jsou normalizovány.

Takto normalizované citlivosti bychom v praxi (mimo simulaci) získali procedurou ESPIRiT, což ilustruje skript sensitivity\_testing.

# Kompenzace hustoty

Pro výpočet vah se používá funkce DoCalcDCF, která přijímá jako argument souřadnice v k-prostoru vztažené k intervalu –1/2 až 1/2. Ve funkci byla vynechána normalizace výsledných vah, tudíž výstup odpovídá plochám Voroného buněk na jednotkovém čtverci. Aby váhy odpovídaly rozměru rekonstruovaného obrázku, jsou vynásobeny faktorem X\*Y, tedy plochou rekonstruovaného obrázku. S touto kompenzací operátor zpětné NUFFT aplikovaný po NUFFT produkuje přibližně stejný obrázek (ilustruje to skript testnufft).

Druhá úprava ve funkci je řazení souřadnic v k-prostoru, které bylo překombinované a váhy se tudíž ne vždy přiřazovaly správným bodům v k-prostoru. To, že po úpravě (zjednodušení) již váhy vypadají tak, jak lze očekávat, ilustruje skript testweights.

## Případ radiálního snímání

Při radiálním snímání středu k-prostoru jsou krajní Voroného buňky neuzavřené, mají tedy nekonečný obsah. Funkce DoCalcDCF přiřazuje těmto buňkám nulovou váhu. Protože však v našem případě jsou váhy po dané radiále od počátku k okraji lineární, lze poslední hodnotu snadno extrapolovat.

# Volba parametrů algoritmu Chambolle–Pock

Pro garantovanou konvergenci algoritmu požadujeme, aby parametry splňovaly

kde je operátor NUFFT včetně kompenzace hustoty (, kde je kompenzace hustoty a je operátor samotné neuniformní Fourierovy transfornace), reprezentuje násobení obrazu všemi citlivostmi a konkatenaci výsledků, je diference ve směru času. Uvažujme, že operátory a pracují s jediným snímkem obrazové sekvence. Víme

Norma je přibližně určena maximální hodnotou . Především je však možné ji efektivně aproximovat pomocí (normalizované) mocninné metody jakožto odmocninu z největšího vlastního čísla operátoru (ilustruje to skript nufftnorm, popis metody je na konci tohoto textu). Poslední nerovnost platí proto, že (protože citlivosti jsou normované) a (dizertace Marušky, str. 59).

Při počítání normy , kde a operují s celou sekvencí, je operátor blokově diagonální, tudíž jeho norma je maximem z norem pro jednotlivé bloky, tedy jednotlivé snímky. Poznamenejme, že pro jednotlivé snímky se liší operátor , citlivosti jsou v čase konstantní.

# Normalizace dat

Cílem je, aby data odpovídala tomu, jako by byla vygenerována operátorem NUFFT (tím stejným, který je používán v rekonstrukci).

K dispozici máme dva vztahy: Již v simulačním skriptu (simulate\_acq\_12) je signál násobený tak, aby řádově odpovídal lineárně interpolované 2D Fourierově transformaci, tedy výstup NUFFT je násoben konstantou sqrt(xsize\*ysize), kde xsize a ysize jsou rozměry simulovaného obrazu. Tento vztah popisuje obecně souvislost 2D FFT v Matlabu a NUFFT.

Druhým vztahem je, že data získaná pomocí FFT ze simulovaného obrazu jsou oproti datům, která by byla získána pomocí FFT z podvzorkovaného obrazu, zvětšená v poměru factor^2.

Ze schématu Obrázek 1je tudíž zřejmé, že správný faktor násobení vstupních dat je

kde označuje factor, a jsou rozměry původního simulovaného obrázku (tedy xsize a ysize). Správnost tohoto násobení dat ilustruje skript testnufft\_scaled.

# Výpočet lineárního operátoru

FFT podvzorkovaného obrázku

FFT celého obrázku

NUFFT celého obrázku

NUFFT podvzorkovaného obrázku

Obrázek : Schéma škálovacích koeficientů

V rámci algoritmu Chambolle–Pock aplikujeme lineární operátor  popsaný výše a jeho adjungovaný operátor .

## Citlivosti

Definujme operátory , které reprezentují násobení obrázku po složkách příslušnou citlivostní mapou. Ve vektorizovaném případě se tedy jedná o diagonální operátory řádu X\*Y, kde X\*Y je délka vektorizovaného obrázku původních rozměrů X krát Y. Potom lze psát

tedy adjungovaný operátor reprezentuje násobení konkatenace obrázků po složkách komplexně sdruženými citlivostmi a následný součet.

## NUFFT

Operátory i implementuje balíček NUFFT.

## Časová diference

Diferenci aplikujeme ve směru času pro každou prostorovou souřadnici zvlášť, tudíž ji aplikujeme na vektor. Pro Matlabovou funkci diff platí:

If X is a vector of length m, then Y = diff(X) returns a vector of length m-1. The elements of Y are the differences between adjacent elements of X.

Y = [X(2)-X(1) X(3)-X(2) ... X(m)-X(m-1)]

Tedy tento operátor lze reprezentovat maticí rozměru m-1 krát m:

Aby operátor diference neredukoval dimenzi, doplňujeme matici o m-tý řádek samých nul. Dostáváme tedy následující podobu:

V implementaci to znamená následující:[[3]](#footnote-3) Při aplikaci využijeme funkci diff(X) (dostáváme vektor délky m-1) a poslední, m-tou hodnotu Y(m) doplníme jako nulovou. Při aplikaci nastavíme první hodnotu jako opačnou hodnotu prvního prvku vstupního vektoru, tedy X(1) = -Y(1), hodnoty X(2:m-1) získáme jako opačnou hodnotu funkce –diff(Y(1:m-1)) a poslední hodnota X(m) = Y(m-1).

# Určení ground truth

## Výpočet obrazové sekvence

V rámci skriptu getImages se využívá část kódu ze simulátoru, která generuje obrazová data pro dané echo. Vstupem funkce jsou hodnoty času (v sekundách) a faktor zmenšení výstupních dat (potřebný proto, že chceme generovat ground truth pro podvzorkovanou rekonstrukci). Zadané hodnoty času jsou poděleny periodou snímání radiál, čímž získáme – vzhledem k simulaci celého signálu – indexy pro jednotlivá echa, která chceme simulovat. Simulovaný obrázek vytváříme v původním rozměru 1024 × 1024 px a následně jej pro každé echo zvlášť podvzorkujeme.

V rámci rekonstrukčního skriptu je obrazová sekvence určována vždy pro časový okamžik odpovídající středu časového úseku, po který je snímán daný obrázek.

## Kompenzace citlivostí

Předpokládejme, že platí přibližná rovnost

kde je snímek simulované sekvence a je operátor NUFFT. Takovouto přibližnou rovnost ilustruje např. skript testnufft nebo testnufft\_scaled. Jestliže doplníme operátor reprezentující rozklad snímku a násobení *normalizovanými* citlivostmi, platí

Ve zjednodušeném případě reprezentuje snímaná data a operátor reprezentuje rekonstrukci. V případě, kdy simulace probíhá s *nenormalizovanými* citlivostmi , ale rekonstrukce probíhá s normalizovanými citlivostmi (po pixelech), dostáváme relaci

Tedy rekonstruovaný snímek přibližně odpovídá vstupnímu snímku, který je po pixelech násobený normou citlivostí (kde norma se počítá pro každý pixel jako norma vektoru citlivostí pro tento pixel).

# Výsledky rekonstrukce

Rekonstruována byla data získaná ze simulátoru pomocí NUFFT. Experiment proběhl s následujícím nastavením:

% radials per frame

rpf = 66;

% number of frames

Nf = 454;

Nf = min(Nf, floor(N/rpf));

% regularizers (parameters of the model)

lambdaS = 0.125;

lambdaL = 0.5;

% parameters of the Chambolle-Pock algorithm, part one

theta = 1;

iterations = 100;

% subsampling factor (for speed-up)

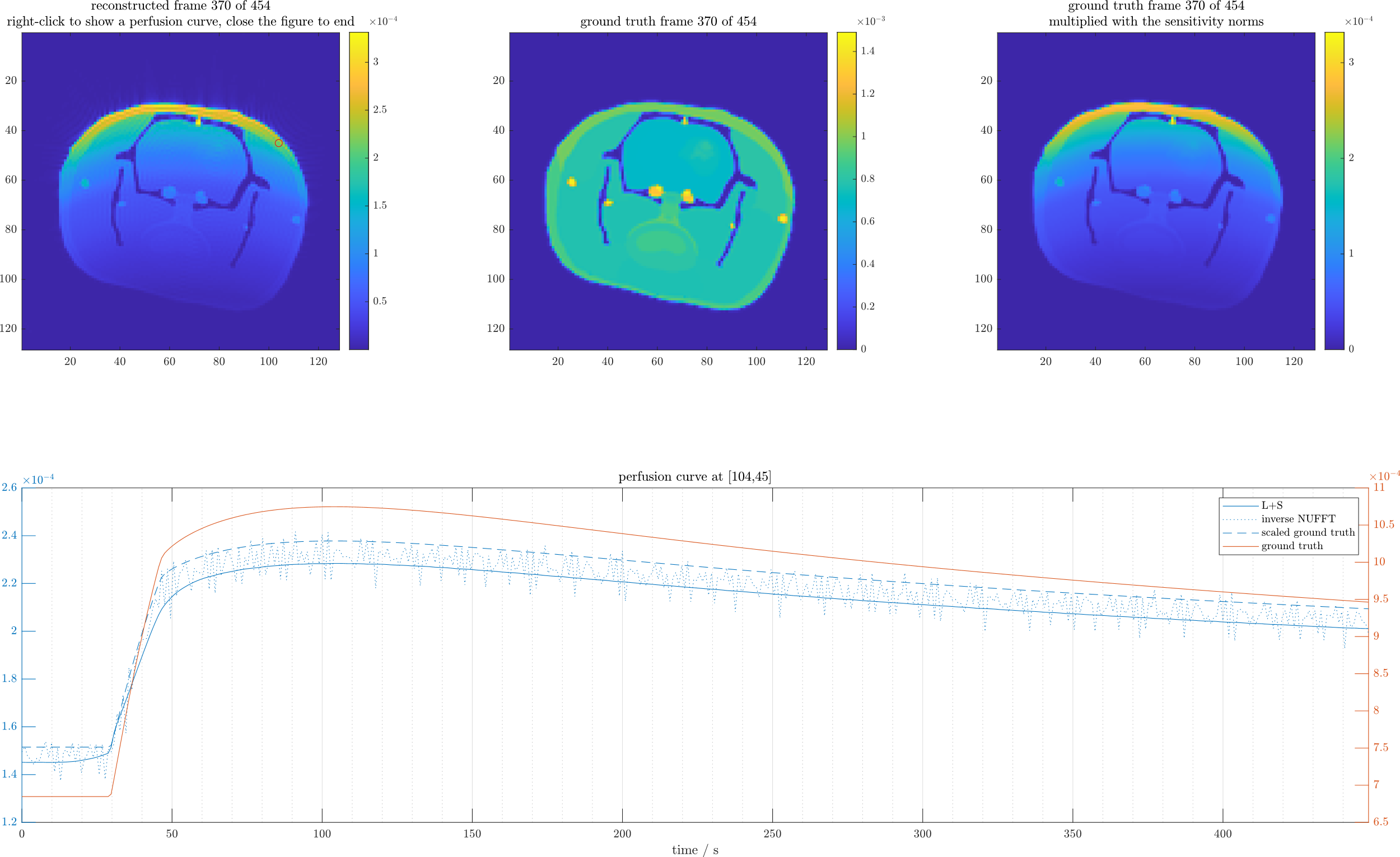
factor = 8;

Výsledky ukazují obrázky Obrázek 2 a Obrázek 3. Obrázek Obrázek 2 byl získán tak, že od rekonstruované obrazové sekvence (v prvním případě pomocí L+S modelu, v druhém případě pomocí prosté inverzní NUFFT) je odečtena ground truth (škálovaná dle popisu v sekci Kompenzace citlivostí) a tento rozdíl, který je stále ještě obrazovou sekvencí, je zprůměrován vzhledem k času. Tedy pro každý pixel vidíme průměrný rozdíl rekonstruované a správné perfúzní křivky.

Z obrázku Obrázek 2 vidíme, že chyba rekonstrukce je největší při hranách a směrem od hran chyba drobně kmitá. Hypotéza je, že simulovaná k-prostorová data data vůbec neobsahují vyšší prostorové frekvence, tudíž hrany není z čeho vyvodit.



Obrázek : Rozdíly rekonstrukce a ground truth



Obrázek : Ukázka rekonstrukce

# Normalizovaná mocninná metoda

Mocninná metoda je způsob numerického výpočtu největšího vlastního čísla dané diagonalizovatelné čtvercové matice . Normalizovaná varianta této metody obsahuje jistou pojistku proti přetečení či podtečení z důvodu geometrického růstu proměnných. Algoritmus je následující:

je libovolný nenulový vektor

, kde

Potom , kde je vlastní číslo matice o největší absolutní hodnotě. [Libor Čermák, *Vybrané statě z numerických metod*, Brno, 2015]

1. Ve vertikálním směru je rozsah počítaný pomocí ysize, v našem případě je ale obraz čtvercový, tudíž v obou dimenzích je rozsah stejný. [↑](#footnote-ref-1)
2. Toto je zjednodušený zápis, ve skutečnosti – při uvažování sudého počtu vzorků Nsamples – jsou vzorky od (-Nsamples/2)\*xsize do (Nsamples/2-1)\*xsize, tedy nejsou rozložené symetricky podle středu. [↑](#footnote-ref-2)
3. Předpokládáme, že X a Y jsou vektory délky m. Pro názornost dále značíme, že operátor má jako vstup X a jako výstup Y, operátor má naopak jako vstup Y a jako výstup X. [↑](#footnote-ref-3)