VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

# Návrh a realizace algoritmu pro systém limitního ozařování

# Design and Implementation of an Algorithm for the System Limit Radiation

2015 David Oczka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

# Zadání bakalářské práce

Student: **David Oczka**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Návrh a realizace algoritmu pro systém limitního ozařování
Design and Implementation of an Algorithm for the System Limit Radiation

Zásady pro vypracování:

1. Seznámení se s problematikou onkologických aplikací terapeutického záření.
2. Seznámení se s prostředím MATLAB.
3. Seznámení se s problematikou plánování radioterapeutických zákroků.
4. Návrh a realizace systému vizualizace onkologického plánování s omezujícími podmínkami polohy pacienta.
5. Provedení měření a testů.
6. Zhodnocení výsledků měření a práce.

Seznam doporučené odborné literatury:

[1] BRONZINO, Joseph D. et al. *The biomedical engineering handbook.* Boca Raton(USA): CRC Press, 1995. 2896 s. ISBN 0849383463/978-0849383465.
[2] SVATOŠ, Josef. *Biologické signály I.* Praha: ČVUT Praha, 1998. 202 s. ISBN 8001018229.
[3] PENHAKER, M., M. IMRAMOVSKÝ a P. TIEFENBACH. *Lékařské diagnostické přístroje: učební texty.* 1. vyd. Ostrava: VŠB-TU Ostrava, 2004. 320 s. ISBN 8024807513/978-8024807515.
[4] CARR, Joseph J. a John M. BROWN. *Introduction to biomedical equipment technology.* 4th ed. Upper Saddle River(USA): Prentice Hall, 2001. 743 s. ISBN 978-0130104922.
[5] MACKAY, Stuart R.*Bio-Medical Telemetry: Sensing and Transmitting Biological Information from Animals and Man.* 2nd Edition. Wiley-IEEE Press, 1998. ISBN: 978-0-7803-4718-2.
[6] FRANDEN, Jacob. *Handbook of Modern Sensors: Physics, Design, and Applications.* 4th edition. Springer, 2010. 663s. ISBN 978-1441964656.
[7] BRONZINO, Joseph D. et al. *The biomedical engineering handbook.* CRC Press, Boca Raton, 1995.
[8] WEBSTER,John G. *Medical instrumentation : application and design.* Hoboken (USA): Wiley, 1998. ISBN 0-471-15368-0.
[9] PEREZ,Reinaldo. *Design of medical electronic devices.* San Diego (USA): Academic press, 2002. ISBN 0125507119/9780125507110.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Marek Penhaker, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015


doc. Ing. Jiří Koziorek, Ph.D.
*vedoucí katedry*

prof. RNDr. Václav Snášel, CSc.
*děkan fakulty*

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Oczka

Datum odevzdání bakalářské práce: 4. 5. 2015

# Poděkování

Rád bych poděkoval panu Ing. Marku Penhakerovi, Ph.D. za cenné rady, konzultace a odborný dohled při vedení mé bakalářské práce. Dále také Ing. Lukáši Knybelovi za konzultace a zkušenosti z praxe týkající se plánovacího procesu systému CyberKnife na onkologické klinice FN Ostrava.

# Abstrakt

Práce popisuje realizaci aplikace pro efektivní plánování procesu radioterapie u systému CyberKnife pomocí zpracování CT snímků. Systém CyberKnife, který zaměřuje terapeutický cíl pomocí dvou rentgenek, jež snímkují pacienta pod úhlem 45 stupňů, umožňuje v oblasti plic lokalizaci ozařovaného nádorového ložiska na základě rozdílu denzit (míry absorpce záření) mezi ložiskem a okolní tkání. Některá ložiska je obtížné lokalizovat vzhledem k sumaci a tedy překryvu sledované oblasti s jinými vysoce denzitními strukturami (např. páteř, paže, žebra). Úkolem práce bylo vytvoření snímků z úhlu 45 stupňů, kde je následně v aplikaci simulována rotace pacienta na CT tak, aby byl překryv nežádoucích tkáňových struktur eliminován. Aplikace je řešena jako klient vytvořený v jazyce C#, připojený na COM server výpočetního systému MATLAB, který zajišťuje většinu výpočtů. Doplňující zpracování obrazu je zajišťováno přímo v klientu, implementováno v jazyce C#. Systém byl testován na jedno jádrovém systému čipů a rychlost načtení a zpracování se pohybuje průměrně kolem 1 s. Systém byl vyvíjen, realizován a nyní úspěšně testován na Onkologické klinice FN Ostrava.

# Klíčová slova

Radioterapie, Radiodiagnostika, CyberKnife, zpracování CT obrazu, MATLAB, C#

# Abstract

This thesis describes the implementation of application for effective planning process of CyberKnife system by processing CT images. CyberKnife system, which focuses therapeutic target using two X-ray tubes and takes images of the patient at an angle of 45 degrees, allows localization of lung irradiated tumor bearing based on the difference in density (absorption rate of radiation) between the bearing and the surrounding tissue. Some tumor bearings are difficult to localize due to summation and overlay of monitoring area with other high densities structures (e.g. the spine, arms and ribs). The task of the work was to create images from an angle of 45 degrees with simulated rotation of patient during CT until the overlay is eliminated. The application is designed as a client created in C#, connected to the COM server of computing system MATLAB, which provides most of the calculations. Additional image processing is provided directly in the client, implemented in C#. The application was tested on a single core chip system and speed of acquisition and processing is in average around 1 s. The application was developed, implemented and now successfully tested on the Oncology Clinic at FN Ostrava.

# Key words

Radiotherapy, Radiodiagnostics, CyberKnife, CT image processing, MATLAB, C#

# Seznam použitých zkratek

| | |
|---|---|
| CT | Computed Tomography / Počítačová tomografie |
| MRI | Magnetic Resonance Imaging / Magnetická rezonance |
| PET | Positron Emission Tomography / Pozitronová emisní tomografie |
| DRR | Digitally Reconstructed Radiograph / Digitálně rekonstruovaný snímek |
| DICOM | Digital Imaging and Communications in Medicine |
| PACS | Picture Archiving and Communication System |
| 3D-CRT | 3D Conformal Radiation Therapy / Trojrozměrná konformní radioterapie |
| IMRT | Intensity Modulated Radiation Therapy / Radioterapie s modulovanou intenzitou |
| SRT | Stereotactic Radiation Therapy / Stereotaktická radioterapie |
| COM | Component Object Model |
| GUI | Graphical User Interface / Grafické uživatelské rozhraní |

# Obsah

# Úvod

Systém CyberKnife je ozařovací systém, který šetrně a účinně ničí nádorová ložiska. Funguje na principu stereotaktického ozařování, což je ozařování nádorového ložiska z mnoha různých směrů paprsky s malou dávkou, která je pro okolní tkáně bezpečná, ovšem v ložisku se dávka sčítá a tím ložisko účinně ničí. Celý proces si systém monitoruje sám pomocí dvou rentgenek, které snímají cíl pod úhlem 45 stupňů po celou dobu terapie.

Než může dojít k samotné terapii, je zapotřebí vše pečlivě naplánovat. Základem plánování je vyšetření na CT. Následuje zakreslování kontur nádoru a také okolních orgánů. Dále výpočet ozařovacích směrů.

Ovšem během tohoto plánování se může objevit případ, kdy je ložisko překryto jinou tkání s vyšší mírou absorpce záření a stane se tak pro přístroj neviditelným. Řešením tohoto problému mohou být klíny různých velikostí, které vytočí pacienta o úhel, jež je potřebný k odkrytí ložiska na zaměřovacích snímcích. Bohužel, v současné době nemá systém CyberKnife k dispozici žádný software, který by umožnil zobrazit snímek rotovaného pacienta a tedy říct, jakým klínem by se měl pacient podložit. Proto se v této chvíli podobné problémy řeší odhadem.

Cílem této práce je proto navrhnout a realizovat algoritmus, který by, na základě CT snímků a kontur, vytvořil snímky pod úhlem 45 stupňů pacienta, který je rotován o určitý úhel, a řekl, jaký klín je třeba použít, a tím zefektivnit plánovací proces.

# 1 Radiodiagnostické metody

Jde o metody v radiologii, které na principu rozdílné absorpce rentgenového záření zobrazují měkké a tvrdé tkáně uvnitř lidského těla. Tyto metody jsou v medicíně velmi důležitým nástrojem k určení správné diagnózy.

## 1.1 Skiagrafie

Skiagrafie je metoda, kdy se výsledné získané zobrazení zaznamenává na nosné médium. Nejčastěji se užívá k zobrazení struktur, které není třeba zvýrazňovat kontrastními látkami, tedy kostí, zubů, případně kloubů. Je možné zobrazovat i měkké tkáně, jako jsou plíce, svaly, atd.

## 1.2 Skiaskopie

Skiaskopie se používá pro zobrazení dynamických dějů ať už za užití kontrastní látky či nikoliv. Výsledná zobrazení nejsou zaznamenávána na média, ale je možné uložení záznamu v podobě videa. Ačkoliv by principiálně mělo jít o kontinuální zobrazení, z důvodů snížení radiační zátěže pacienta se užívá pulzního snímání, kdy je zobrazení obnovováno s určitou frekvencí, a nízkých hodnot energií. [1]

## 1.3 Kontrastní látky

Kontrastní látky jsou látky, které zvyšují či snižují absorpci rentgenové záření, a tím zobrazují struktury, ve kterých se nachází. Ve většině případů jde o tekutiny, ale existují i plynné či pevné kontrastní látky.

### 1.3.1 Negativní

Negativní kontrastní látky snižují absorpci rentgenového záření. Většinou se jedná o plyny. Příkladem negativní kontrastní látky může být vzduch (v plicích). Laicky řečeno, výsledkem negativních kontrastních látek je více černé barvy ve výsledném zobrazení.

### 1.3.2 Pozitivní

Pozitivní kontrastní látky absorpci rentgenového záření zvyšují. Řadí se sem jódové (zobrazení cév) či baryové (zobrazení trávicího traktu) kontrastní látky. Opět laicky řečeno, výsledkem pozitivních kontrastních látek ve výsledném zobrazení je více bílé barvy. [1]

## 1.4 Snímače

Snímače rentgenového záření jsou důležitým prvkem při vytváření výsledného zobrazení, protože určují hodnotu absorbovaného záření.

### 1.4.1 Analogové snímače

Analogové snímače jsou stínítka obsahující rentgenový film, na který se zachycuje údaj o absorpci záření, tento film se pak vyvolá. Z tohoto plyne, že se musí film při každém použití vyměnit.

### 1.4.2 Digitální snímače

Digitálními snímači jsou speciální polovodičové detektory, jejichž výstupem je elektrický signál, který je následně převeden do digitálního snímku.

## 1.5 Přenosová média

Na přenosových médiích se uchovává záznam radiodiagnostického vyšetření. Opět máme média analogová a digitální.

### 1.5.1 Analogová média

Analogová média jsou hmotnými médii, například vyvolaný snímek. Nevýhodou těchto médií je nižší kvalita snímků, bez možnosti jakéhokoliv pozdějšího zpracování obrazu. Výroba a uskladnění těchto médií je, oproti těm digitálním, dražší.

### 1.5.2 Digitální média

Digitální médium je snímek uložený v paměti počítačového zařízení. Jeho výhodou je možnost pozdějšího zpracování a bez fyzické existence jednoduchá přenositelnost či uskladnění.

#### 1.5.2.1 Souborový formát DICOM

Jedná se o standardizovaný formát pro uchování obrazových dat z CT, MRI nebo také ultrazvuku. Mimo informací o obrazových datech a jeho parametrech, obsahuje tento formát také informace o pacientu, daném vyšetření, instituci a přístroji, kde byl snímek pořízen.

Díky tomuto standardizovanému formátu je možné sdílet obrazová data o pacientech mezi zdravotnickými zařízeními velmi snadno a rychle. Slouží k tomu systém PACS, který zajišťuje archivaci, správu a zobrazení těchto snímků napříč nemocnicemi.

## 1.6 Klasické snímkování

Touto metodou se vytváří dvourozměrné snímky trojrozměrného objektu, vyšetřované části pacienta. Výsledný snímek je zaznamenán na rentgenový film, který se následně vyvolá stejným způsobem jako obyčejná fotografie, ovšem výsledným rentgenovým snímkem je negativ. Na výsledném zobrazení jsou světlejší místa struktury s vyšší absorpcí záření, přičemž tmavší místa jsou struktury s absorpcí nižší.

Snímky pak vyhodnocuje lékař na negatoskopu, přístroji, který má rovnoměrně podsvícenou matnou plochu, na kterou se připevní snímek.

[1]

## 1.7 Hounsfieldovy (denzitní) jednotky

Jedná se o číselný údaj, určující hodnotu absorpce záření vztaženou k hodnotě absorpce záření vody v daném bodě na digitálním snímku. Každý bod na digitálním snímku se nazývá pixel, ovšem na snímcích (řezech) z CT reprezentuje trojrozměrnou jednotku objemu, tzv. voxel.

## 1.8 CT

Počítačová tomografie je radiodiagnostická metoda, kdy je výsledný obraz matematickou rekonstrukcí z nespočtu rentgenových projekcí, které jsou pořízeny z různých úhlů. Výsledkem je série snímků, řezů, které nesou informaci o absorpci záření v podobě Hounsfieldových denzitních jednotek pro jednotlivé voxely. Výstupním formátem je standardizovaný formát DICOM.

### 1.8.1 Princip

Pacient je fixován na pohyblivém lehátku, které prochází snímacím stojanem, který má nejčastěji tvar torusu. Uvnitř torusu rotuje rentgenka se štěrbinovým paprskem. Naproti rentgence jsou umístěny detektory.

Detektory jsou provedeny několika způsoby, starší typy tomografů mají detektory připevněny naproti rentgence, a tedy rotují s rentgenkou. V nové generaci jsou detektory statické a jsou připevněny po vnitřním obvodu torusu.

Záření z rentgenky projde skrze pacienta, určitá část se absorbuje a zbytek dopadá na detektory, ty zaznamenají míru absorpce, rentgenka se pootočí o určitý úhel a proces se opakuje. Takto to probíhá, dokud není zaznamenán dostatečný počet úhlů pro výpočet jednotlivých voxelů (ukázka výpočtu níže). Po dokončení řezu se lehátko s pacientem posune a celý cyklus se opakuje, dokud není zachycena celá vyšetřovaná oblast.

Hodnota voxelů je vypočítávána na základě znalosti součtu v jednotlivých směrech. Tím je získáno n rovnic o n neznámých, které je možné, díky znalosti součtu z tolika směrů, jednotlivě vypočítat.

[2]

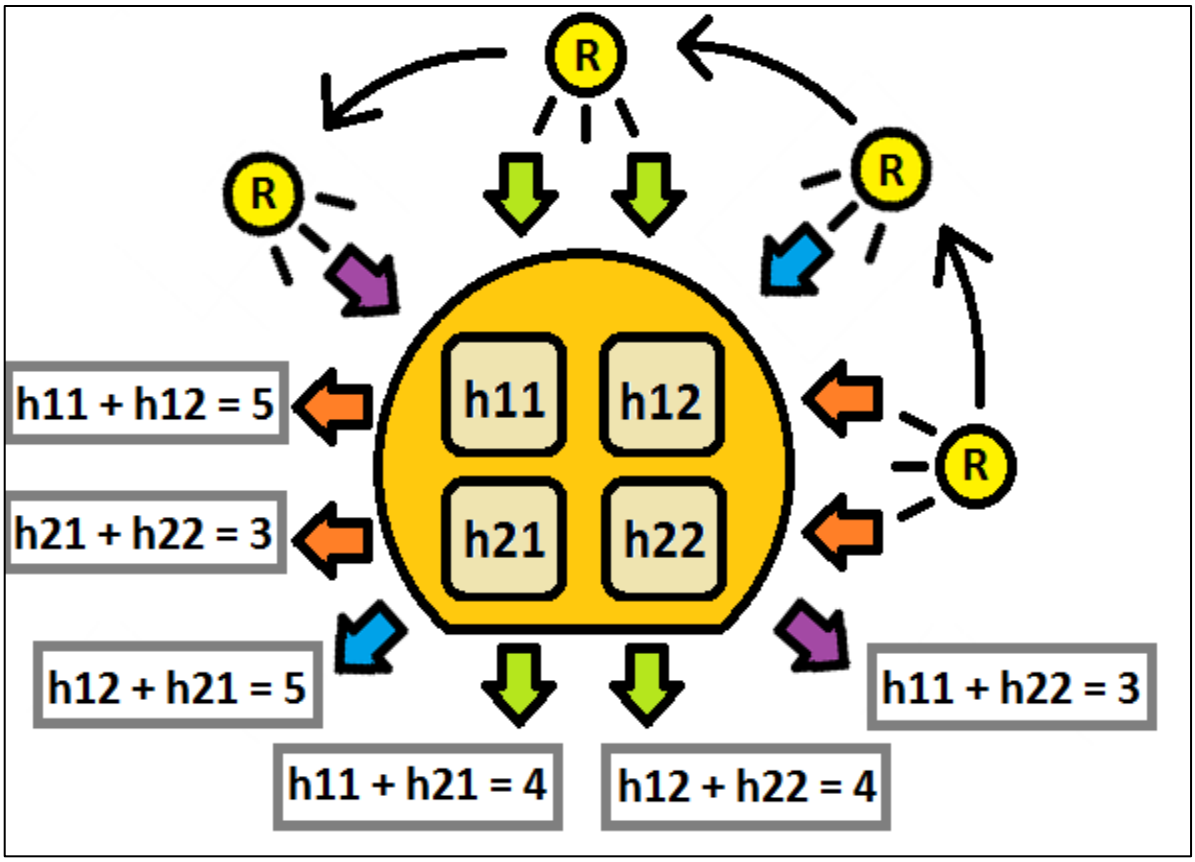
#### 1.8.1.1 Příklad výpočtu voxelů

Na obrázku níže vidíme zjednodušené vysvětlení principu výpočtu jednotlivých voxelů. Písmenem R je označena rentgenka a šipkami průchod záření pacientem. V tomto příkladu máme vyšetřovaný řez pacientem rozdělen na čtyři voxely. Ovšem, jednoduše řečeno, je možné plochu řezu rozdělit na libovolný počet voxelů, pokud máme součty z dostatečného počtu směrů.

Na obrázku je naznačeno, jak rentgenka rotuje a získá součet voxelů ze čtyř směrů. Tím získá šest rovnic vždy o dvou neznámých z celkového počtu čtyř neznámých. Po úpravě a vyřešení těchto rovnic získáme hodnoty pro jednotlivé voxely.

V našem případě jsou rozměry 2x2 a hodnoty pro voxely: $h11 = 2$, $h12 = 3$, $h21 = 2$ a $h22 = 1$.

V takto triviálním příkladu je výpočet snadnou záležitostí, ovšem reálné rozlišení CT snímků je většinou 512x512 bodů a počet voxelů, a tím i počet neznámých dosahuje hodnoty 262 144.




Obrázek 1.1 Zjednodušený princip CT výpočtu

## 1.9 Angiografie

Angiografie je metoda užívaná k zobrazování cév pomocí kontrastní látky. Nejčastějšími užívanými kontrastními látkami jsou látky na jódovém základu. V případě rizika vzniku alergické reakce či poruše funkce ledvin, protože jodové kontrastní látky se vylučují ledvinami, se užívají kontrastní látky gadoliniové, případně oxid uhličitý.

### 1.9.1 Digitální subtrakční angiografie

Jedná se o invazivní metodu, kdy je ve vyšetřované oblasti nejprve sejmut tzv. nativní snímek (snímek bez kontrastní látky), tento snímek je převeden v negativ, poté se vstřikuje kontrastní látka a oblast se opět snímá. Výsledný obraz je pak rozdílem nativního snímku a snímku s kontrastní látkou. Díky substrakci nejsou na výsledném snímku vidět statické struktury (např. kosti), ale pouze cévy, skrze které protéká kontrastní látka. Tímto získáme přesný obraz cév ve vyšetřované oblasti.

### 1.9.2 CT Angiografie

CT angiografie je metoda, kdy je snímání pomocí CT načasováno tak, aby zachytilo okamžik, kdy jsou cévy naplněny kontrastní látkou. Na výsledných snímcích je pak možné vidět cévy a jejich větvení v prostoru.

[3]

# 2 Účinky ionizujícího záření na živé tkáně

Ionizující záření má stejné účinky, jak na živou, tak na neživou hmotu. Rozdílem je ovšem reakce na tyto účinky. Během absorpce záření dochází k ionizaci, excitaci, a tedy i absorpci energie. Tyto procesy jsou pak v organismech spouštěcími mechanismy nespočtu biochemických změn. Výsledkem může být poškození či dokonce usmrcení organismu.

[4] [5]

## 2.1 Buněčné dělení

K vysvětlení účinků ionizujícího záření na buňku je nutné popsat nejprve způsoby jejího dělení. Existuje několik způsobů dělení buněk.

### 2.1.1 Amitóza

Jde o způsob dělení buňky zaškrcením, kdy nezáleží na přesném rozdělení počtu chromozómů uvnitř buňky. Tento způsob dělení je specifický pro nekontrolovaná bujení, jaká probíhají, například v nádorech.

### 2.1.2 Mitóza

U tohoto způsobu probíhá dělení v několika fázích a výsledkem jsou dvě buňky, mateřská a dceřiná, které mají stejný, plný, počet chromozómů. Probíhající fáze dělení jsou profáze, metafáze, anafáze, telofáze. V rámci těchto fází dochází k transkripci, přepisu (vytvoření kopie) genetické informace, která pak bude obsažena v dceřiné buňce. Fáze mezi dvěma mitózami se nazývá interfáze. Tento způsob je nejčastějším způsobem dělení, ke kterému v organismu dochází.

### 2.1.3 Meióza

Meióza, neboli redukční dělení, je způsob dělení, kdy je výsledný počet chromozómů snížen na polovinu. Jde o způsob dělení pohlavních typů buněk. Plného počtu chromozómů je docíleno až splynutím mužské a ženské pohlavní buňky.

[6]

## 2.2 Účinky záření na buňku

### 2.2.1 Usmrcení buňky

Pokud se buňka nachází v interfázi, pak v závislosti na velikosti dávky záření, může buňka zemřít okamžitě na následek denaturace buněčného obsahu, anebo může dojít k poškození buňky během některé z fází dělení, které nebude mít za následek okamžitou smrt, ale učiní buňku neschopnou se dále dělit.

Případ, kdy buňka není schopna se dělit, nazýváme mitotická smrt a je pozorován již při menších dávkách záření. Tento účinek se snadno projeví v tkáních, kde probíhá rychlé buněčné dělení.

[4] [5]

### 2.2.2 Změna genetické informace

Dalším výsledkem ozáření buňky může být změna genetické informace neboli mutace. Tyto mutace vznikají v rámci ozáření během transkripce genetické informace a mohou pak způsobovat poruchy organismu. V případě mutace pohlavních buněk je možnost přenosu poruchy do další generace. Zmutované buňky mohou mít také vztah ke vzniku rakoviny.

[5]

# 3 Radioterapie

Radioterapie je léčebná metoda, která využívá k léčbě nádorových i nenádorových tkání účinky ionizujícího záření. Toto záření při průchodu tkání předává svou energii a tím spouští sérii pochodů vedoucích k degeneraci tkáně. Ionizující záření může být vyzařováno radioaktivní látkou nebo zařízením k tomu určeném.

Cílem radioterapie je dodat co nejvíce energie ionizujícího záření do cílové oblasti nádoru, přičemž bude okolní tkáň zasažená co nejméně.

Po chirurgickém zákroku je radioterapie nejvíce efektivní metodou léčby onkologických onemocnění a užívá se dnes až v polovině případů. Tato metoda je často kombinována s chemoterapií nebo imunoterapií.

Možná je také kombinace s termoterapií, kdy je tkáň zahřívána na určitou teplotu či rozmezí teplot, tím je ovlivněno prokrvení a okysličení. Záření má pak zvýšený účinek na okysličenou nádorovou tkáň.

Radioterapii dělíme podle vzdálenosti zdroje záření na brachyterapii a teleterapii. Tyto metody se během léčby často kombinují.

## 3.1 Cíle aplikace radioterapie

### 3.1.1 Kurativní radioterapie

Cílem je zničit nádor a plně vyléčit pacienta. Indikace je buď primární, nebo jsou výsledky srovnatelné s jiným typem léčby. Uplatňuje se také v případech, kdy je třeba zachovat funkce daného orgánu.

### 3.1.2 Paliativní radioterapie

Primárním cílem paliativní radioterapie je odstranit či alespoň zmírnit příznaky nádorového onemocnění (bolest, krvácení, atd.). Až druhotným cílem je prodloužení života pacienta.

### 3.1.3 Nenádorová radioterapie

Cílem je zmírnit obtíže způsobené nenádorovými onemocněními nebo zabránit zhoršení funkce postiženého orgánu. Indikace je až při vyčerpání standartních léčebných postupů. Nejčastěji a s nejlepšími výsledky se využívá u bolestivých onemocnění pohybového aparátu. Tento typ léčby není vhodný pro mladé pacienty a ženy v plodném věku.

[7] [8]

## 3.2 Brachyterapie

Brachyterapie je léčebná metoda v radiologii, kdy je zářič v malé vzdálenosti od ložiska či přímo v orgánu nebo tkáni s nádorem.

Tato metoda je indikována ve čtyřech případech. Prvním případem je primární radikální léčba, druhým je tzv. boost neboli podpora k teleterapii, pak paliativní léčba a ozařování recidiv.

### 3.2.1 Vývoj

Brachyterapie je známá už dlouhou dobu. Ve svých začátcích byl zdroj záření zaváděn manuálně za zvýšené radiační zátěže zdravotnického personálu. V dnešní době už existují různé nosiče, které se zavedou jako první a až pak se zavádí aktivní zářič, což zvyšuje přesnost účinku zářiče i bezpečnost personálu. Zavedení zářiče probíhá buď manuálně, nebo automaticky pomocí přístroje. Tento postup, kdy je prvně zaveden nosič a až poté zářič se nazývá afterloading.

### 3.2.2 Rozdělení podle času ozařování

Podle času dělíme brachyterapii na dočasnou a permanentní.

#### 3.2.2.1 Dočasná brachyterapie

V rámci dočasné brachyterapie je postižené místo jednorázově či opakovaně ozařováno. Ozařování se provádí, například jednou denně, ale jsou možné i častější ozařování.

#### 3.2.2.2 Permanentní brachterapie

Druhou možností je permanentní zavedení zářiče do postižené tkáně. Zdroje mají většinou podobu malé tyčinky. Místa užití jsou nejčastěji mozek a prostata.

### 3.2.3 Rozdělení podle umístění zdroje záření

Umístění zdroje záření je důležité, protože určuje efektivitu terapie, protože intenzita záření klesá s kvadrátem vzdálenosti.

#### 3.2.3.1 Intersticiální

Zdroj záření je umístěn přímo do místa postižené tkáně.

#### 3.2.3.2 Intrakavitární

Zdroj záření je zaveden do tělesných dutin pacienta (jícen, rektum, pochva, děloha, průdušnice). Zářiče jsou často v podobě zrn.

#### 3.2.3.3 Povrchová/kontaktní

Jde o tzv. muláž, zdroj záření je umístěn ve speciálních aplikátorech na povrchu těla.

### 3.2.4 Výhody a nevýhody

Výhodou je dodání vysoké dávky přesně do místa nádoru, přičemž jsou zdravé okolní tkáně bez poškození nebo jsou poškozeny minimálně. Aplikace probíhá v krátkém čase a je velice účinná na malé a přesně lokalizované nádory.

Nevýhodou je fakt, že jde o metodu invazivní, která může být velice bolestivá, jsou tedy většinou potřebná anestetika, a mohou vzniknout komplikace, které upoutají pacienta na lůžko.

[8] [9]

## 3.3 Teleterapie – zevní radioterapie

Teleterapie je léčebná metoda, kdy je zdroj záření od pacienta vzdálen. Nejčastějším zdrojem záření jsou v dnešní době lineární urychlovače, protože umožňuji regulaci energie záření, což ovlivňuje průnik záření do hloubky.

### 3.3.1 Rozdělení teleterapie

Místo, resp. směr, ze kterého se ozařuje, se nazývá pole. Během terapie se využívá různý počet polí, různá velikost jednotlivých polí a intenzita záření z daného pole. V závislosti na těchto parametrech dělíme teleterapii na konvenční, konformní, radioterapii s modulovanou intenzitou a stereotaktickou radioterapii.

Metody, používající více ozařovacích polí, využívají skutečnosti, že se dávka v ohnisku, kde se paprsky polí protínají, sčítá. Takto je možné docílit vysoké dávky uvnitř ložiska, které je situováno v ohnisku, zatímco okolní zdravá tkáň je sice zasažena, ale menší dávkou.

#### 3.3.1.1 Konvenční (2D) radioterapie

Ozařování ložiska probíhá z jednoho či několika málo míst (polí). Jedná se o dvourozměrné ozařování, kdy svazek záření není přizpůsoben tvaru nádorového ložiska. Terapie se provádí bez plánovacího CT vyšetření.

#### 3.3.1.2 Trojrozměrná konformní radioterapie (3D-CRT)

Při této metodě je svazek záření (pole) přizpůsoben tvaru nádorového ložiska. Výsledkem je ozáření nádorové tkáně s menším zasažením zdravé tkáně než při konvenčním ozařování. Terapie se provádí s plánovacím CT, kdy je známa přesná poloha nádorového ložiska a kritických orgánů.

#### 3.3.1.3 Radioterapie s modulovanou intenzitou (IMRT)

Radioterapie s modulovanou intenzitou ozařovacího svazku je modernější verze konformní radioterapie, kdy je možné kromě přizpůsobení svazku záření (pole) tvaru ložiska, přizpůsobit i energii záření.

Touto metodou je také možné ozářit určité části ložiska různou intenzitou záření, a tím zvýšit efektivitu léčby. Výsledkem je možnost ozářit složitější geometrické útvary a méně zatížit zdravou tkáň.

#### 3.3.1.4 Stereotaktická radioterapie (SRT)

Rozdíl této metody od ostatních je v počtu ozařovacích polí, který je v řádech desítek až stovek. Využívají se zde složité trojrozměrné plánovací systémy.

Jednoduše řečeno, se ložisko ozařuje z mnoha různých směrů, paprsky záření nízké intenzity, které jsou pro okolní zdravou tkáň bezpečné, ale v ohnisku polí, v ložisku, se dávka sčítá a efektivně tak ničí nádorovou tkáň. Přičemž proměnlivý může být tvar i intenzita jednotlivých polí.

[7]

### 3.3.2 Výhody a nevýhody

Výhodou teleterapie je, že se jedná o neinvazivní metodu, která je schopna ozářit větší objemy nádorové tkáně i ve špatně přístupných místech.

Nevýhodou může být, že některé typy metod v teleterapii potřebují i pro malou dodanou dávku více procedur a naopak jiné výrazně zatěžují kromě nádorové tkáně také zdravou tkáň.

# 4 CyberKnife

Systém CyberKnife je robotický ozařovač, který využívá stereotaktické metody ozařování k ničení nádorových ložisek. Je schopen ozařovat nádory téměř v celém těle. Řízení systému je postaveno na technologii řízení obrazem. Obrazová data získává systém pomocí dvou rentgenek umístěných na pravé a levé straně od pacienta. Tyto rentgenky snímají pacienta pod úhlem 45 stupňů.

Díky nepřetržitému sledování cíle, je schopen systém CyberKnife ozařovat i pohyblivé cíle, například v plicích či játrech. Pokud se jedná o špatně viditelná ložiska v měkkých tkáních, je možné implantovat do cílové oblasti zlatá zrna, která budou pro systém CyberKnife vztyčné body. Přesnost ozařování je i pro pohyblivé cíle 2 mm. Celý proces ozařování je předem naplánován podle plánovacího CT nebo MRI.

[10]

## 4.1 Historie

Systém CyberKnife vynalezl v roce 1987 John R. Adler, profesor neurochirurgie a radiační onkologie, na univerzitě ve Stanfordu v přátelské spolupráci se zakladatelem radiochirurgie Švédem Larsem Leksellem. Adlerova představa byla vytvořit neinvazivní robotický radiochirurgický systém s vysokou přesností v léčbě nádorů v jakékoliv části těla. Tento revoluční nápad sahal daleko za běžnou radiochirurgickou praxi té doby, která se tehdy zaměřovala pouze na léčbu nitrolebečních nádorů.

První pacient byl léčen systémem CyberKnife v roce 1994. Až do roku 2001 byl tento systém pouhým prototypem, určeným ke klinickému výzkumu, než byl schválen FDA (U. S. Food and Drug Administration). Pak se začal tento systém průmyslově vyrábět. Vyrábí a dodává jej firma Accuray.

[11] [12]

## 4.2 Konstrukce

Konstrukce systému CyberKnife se sestává z několika funkčních bloků. Patří sem robotický manipulační systém nesoucí lehátko pacienta a ozařovač, samotný ozařovač, kolimační systém určující tvar svazku a zobrazovací systém k navigaci během terapie.

### 4.2.1 Robotický manipulační systém

#### 4.2.1.1 Léčebný manipulátor

Léčebný manipulátor je název pro robotické rameno nesoucí ozařovač. Rameno je složeno z šesti článků, což zajišťuje jeho dobrou pohyblivost a obratnost. Tento robot je totožný s roboty používanými v automatizovaných provozech, například automobilový průmysl. Využit je pouze minimální potenciál rychlosti tohoto robota, asi 5-7%, z důvodů bezpečnosti a ohledu na strach pacienta.

Toto rameno má vymezený prostor, ve kterém se smí pohybovat. Díky sledování procesu v reálném čase pomocí dvou rentgenek je rameno svými pohyby schopno kompenzovat sebemenší odchylky při pohybu pacienta. Během terapie komunikuje toto rameno s druhým ramenem nesoucím lehátko s pacientem, v případě, že rameno nesoucí ozařovač není schopné odchylku vykompenzovat, například ve chvíli, kdy by muselo opustit svůj vymezený prostor, vykompenzuje tuto odchylku rameno nesoucí lehátko s pacientem.

#### 4.2.1.2 Lehátko nesené robotickým ramenem (RoboCouch )

Lehátko s pacientem je neseno obdobným robotickým ramenem, jako je popsáno výše a nazývá se RoboCouch. Vzhledem k zobrazovacímu systému, musí být lehátko vyrobeno z materiálu vhodného k průchodu rentgenového záření. Lehátko je proto vyrobeno z uhlíkových vláken.

[13]

### 4.2.2 Bezpečnostní systém

#### 4.2.2.1 Fixní zóna

Fixní zóna je prostor, kde se nachází pacient. Tento prostor si systém sám vymezí na základě CT snímků. Robotické rameno do této zóny nesmí nikdy zasáhnout ani svou částí a je tak zabráněno kolizi ramena s pacientem.

#### 4.2.2.2 Dynamická zóna

Dynamická zóna je ve své podstatě bezpečností lem kolem fixní zóny. Na začátku zákroku je řídicímu systému zadáno, jak velký je pacient, podle toho se pak robotické rameno orientuje a v případě, že zasahuje do této dynamické zóny, je obsluha upozorněna a vyzvána ke zvýšené opatrnosti z důvodu možné kolize ramena s pacientem.

### 4.2.3 Zdroj záření

Ozařovač nesený robotickým ramenem se sestává z elektronového děla, které má na konci připevněn malý lineární urychlovač.

#### 4.2.3.1 Elektronové dělo

Elektronové dělo je zdrojem elektronů, které jsou určeny k ozařování. Elektrony jsou vytvořeny a následně nasáty do urychlovací struktury odkud jsou „vystřeleny“.

#### 4.2.3.2 Lineární urychlovač

Lineární urychlovač je zařízení, které je schopné díky elektromagnetickému poli, které vytváří, urychlovat částice po jejich přímé, lineární, dráze. Částice jsou urychleny na 99% rychlosti světla na vzdálenosti přibližně 50 cm.

[13]

### 4.2.4 Kolimační systém

Kolimátory jsou clony zajišťující výstup záření z ozařovače ve správném směru a tvaru. Existují kolimátory primární a sekundární.

Primární kolimátory jsou napevno umístěny uvnitř ozařovače a zajišťuji, aby se záření nedostalo ven jinudy, než odkud je to vyžadováno.

Sekundární kolimátory jsou clony, které jsou vyměnitelné a mění se během zákroku. Nasazují se na hlavici ozařovače a tím mění tvar svazku z něj vycházejícího.

#### 4.2.4.1 Pevné kolimátory

Pevné kolimátory jsou kruhové clony vyrobené z wolframu. Mají různou délku a vnitřní otvor. Během zákroku jsou buď kolimátory měněny manuálně personálem, nebo jsou vyměněny automaticky přístrojem na speciálním stolku, který obsahuje k tomuto účelu potřebnou aparaturu.

Tento speciální kolimátorový stůl se nazývá Xchange. Nicméně, výměna těchto kolimátorů ať už manuálně či automaticky je časově náročná operace.

#### 4.2.4.2 Motoricky regulované (Iris) kolimátory

Druhou možností jsou kolimátory s proměnlivým propustným otvorem, jehož velikost je automaticky počítačově ovládána. Tento typ kolimátoru uvnitř obsahuje dvě řady šestihranných wolframových clon, které společně vytváří otvor dvanáctiúhelníkového tvaru. Během zákroku se velikost otvoru mění bez nutnosti měnit celý kolimátor, čímž je zkrácena i doba zákroku.

[13]

### 4.2.5 Zobrazovací systém

Jedná se o stereotaktický rentgenový zobrazovací systém, který se sestává ze dvou vůči sobě pravoúhle umístěných rentgenek a zobrazovacích flat-panelů. Rentgenky jsou zavěšeny u stropu po bočních stranách lehátka pod úhlem 45 stupňů. Flat-panely pak mohou být na stojanech nebo zapuštěny v podlaze.

Centrální paprsek rentgenek prochází vždy bodem, který má hodnotu nula ve všech osách. Tento bod se nazývá imaging center a ozařované ložisko by mělo vždy být v tomto bodě či v jeho těsné blízkosti.

Systém vytváří snímky na začátku zákroku pro určení správné pozice pacienta a pak během zákroku, kdy je kontrolována poloha pacienta. Správná pozice je kontrolována na základě porovnání digitálně rekonstruovaných snímků (DRR) vygenerovaných z plánovacího CT a snímků z rentgenek.

### 4.2.6 Zaměřovací systémy

Pro přesné určení polohy ozařovaného ložiska a tzv. online verifikaci pozice pacienta disponuje systém CyberKnife několika zaměřovacími technologiemi. Tyto technologie jsou určeny pro různé oblasti v lidském těle.

Všechny systémy jsou založeny na porovnání digitálně rekonstruovaných snímků a reálně snímaných snímků. Systémy je možné používat jednotlivě nebo v kombinaci.

#### 4.2.6.1 6D systém sledování lebky

Tato technologie rozpoznává a sleduje kostní znaky lebky, které jsou vzaty jako orientační body. Následně je na základě těchto bodů vyhodnoceno, zda a kam se pacient pohnul.

Díky této technologii odpadá potřeba stereotaktického rámu, ale i přesto je potřeba užití termoplastické masky, která je vyrobena na míru každému pacientu a je během terapie připevněna k lehátku.

Systém nese označení 6D, protože se jedná o pohyb ve třech osách a rotaci v těchto osách.

#### 4.2.6.2 Režim sledování zaměřovacích bodů

Tato technologie si vyžaduje invazivní zákrok a tím je implantace zaměřovacích bodů do okolí ozařované oblasti. Počet implantátů bývá 3-6. Implantáty mohou být, například zlatá zrna (1x5mm) nebo ocelové šroubky (2x5mm).

Používá se v případě, kdy se ozařovaná oblast nachází v měkkých tkáních a není dobře viditelná na rentgenových snímcích. Po zavedení zlatých zrn má pak systém CyberKnife v této oblasti orientační body.

#### 4.2.6.3 Režim Xsight Spine Tracking

V tomto případě jsou sledovány kostní znaky páteře a na jejich základě je zjišťován pohyb pacienta. Pohyb je opět sledován ve třech osách a třech rotacích.

#### 4.2.6.4 Režim Xsight Lung Tracking

Tento systém se používá v kombinaci s režimem Xsight Spine Tracking a systémem Synchrony Respiratory Tracking. Používá se při ozařování plicních nádorů a zaměřuje ložisko v plicích na základě rozpoznání stupňů šedi, přičemž ložisko musí mít ve všech směrech alespoň 15mm.

#### 4.2.6.5 Systém Synchrony Respiratory Tracking

Tato technologie kontroluje pohyb ložiska v rámci dýchání pacienta pomocí speciální vesty, kterou má pacient na sobě. Ozařování je pak synchronizováno s dýchacími pohyby pacienta.

[13]

### 4.2.7 Plánovací systém terapie

Plánovací proces systému CyberKnife zajišťuje systém Multiplan, který vypočítává rozložení dávky pomocí algoritmu Monte Carlo.

#### 4.2.7.1 MultiPlan systém

Jedná se o počítačový program, ve kterém se vytváří ozařovací plán systému CyberKnife. V tomto systému se nastavují všechny nezbytné parametry ozařovacího plánu. Do tohoto systému se nahrávají obrazová data pacienta z různých zobrazovacích metod, přičemž jako základ jsou použita data z CT vyšetření. Tato data jsou případně ještě sloučena s daty z MRI nebo PET.

Do sloučených výsledných dat zakresluje lékař kontury označující polohu nádorového ložiska a kritických orgánů. Na základě obrazových dat se zakreslenými konturami jsou pak pomocí algoritmu Monte Carlo vypočteny ozařovací směry (pole) a jejich tvar a intenzita.

Výsledný plán mívá od 100 do 250 ozařovacích polí. Po vytvoření plánu se již neprovádí žádná simulace.

# 5 Výpočetní systém MATLAB

Matlab (zkratka z názvu Matrix laboratory) je interaktivní výpočetní prostředí pro vědeckotechnické výpočty, tvorbu modelů, algoritmů, simulací, analyzování a zobrazování dat. Je schopen paralelních výpočtů, měření a zpracovávání signálů, navrhování řídících a komunikačních systémů. Tento výpočetní systém je také možné propojit s nespočtem různých rozhraní a jeho možnosti jsou tedy velmi rozsáhlé. Mimo to je možno vytvářet jednoduše i uživatelská grafická rozhraní s dvourozměrnými či trojrozměrnými komponentami.

Matlab sám o sobě není jednotným celkem, ale obsahuje hlavní výpočetní jádro a pak balíčky funkcí zvané toolboxy. Touto strukturou pak nabízí uživateli možnost výběru balíčků s funkcemi v jeho oblasti zájmu. Další důležitou částí je nástavba Simulink. Jedná se o grafické prostředí, kde je možné modelovat různé děje pomocí blokových schémat, která opět nabízí jednotlivé toolboxy. Simulink je taktéž možné propojit oboustranně s algoritmy v Matlabu (Možnost spojení kódu se simulací, ale také simulace s kódem).

[14]

## 5.1 Historie

Matlab byl vytvořen na konci sedmdesátých let dvacátého století profesorem Clevem Molerem, z katedry informačních technologií na univerzitě v Novém Mexiku. Ten jej vytvořil, aby jeho studenti mohli snadněji využívat složitější matematické operace bez vyšší znalosti nějakého z programovacích jazyků, ve kterých by tyto počty byly ještě o mnoho složitější. Když se Matlab rozšířil i na další univerzity a stal se tak známým, byl Matlab přepsán do jazyka C a stal se plnohodnotným produktem, který nabídla na trh firma Mathworks, která jej dodnes vyvíjí a stále vylepšuje.

Na počátku své kariéry a to v roce 1985, kdy byla vydána první verze pro stolní počítače, měl Matlab poměrně velké problémy s operační pamětí, která byla na těchto počítačích dostupná, protože ta omezovala maximální velikost matic, které jsou základem Matlabu. S postupným vývojem silnějších počítačů se vyvíjel také Matlab a velkým pokrokem byla nová verze Matlabu 386, která umožňovala práci s virtuální pamětí a tedy převyšovala skutečnou operační paměť, díky čemuž bylo možné vyřešit i problém s velikostmi matic, ovšem toto řešení se projevilo snížením rychlosti výpočtů.

[15]

## 5.2 Programovací jazyk

Matlab má vlastní programovací jazyk postavený podobně jako kdybychom spojili základy jazyků C, Fortran, Pascal a dalších. Jde o velmi jednoduchou syntaxi, kterou je schopen ovládat i uživatel s pouhými základy programovací logiky. V tomto programovacím jazyce se nevyužívá deklarace proměnných jako objektů s určeným datovým typem, nýbrž si proměnné uživatel přímo definuje a Matlab sám, podle zadaného vstupu, určuje jejich datový typ. Uživatel si tedy vystačí se znalostí základních klíčových slov, jako jsou slova pro podmínky a cykly. Ostatní slova jsou již proměnné či funkce, které jsou definovány buď uživatelem či obsahem nainstalovaných toolboxů.

[15]

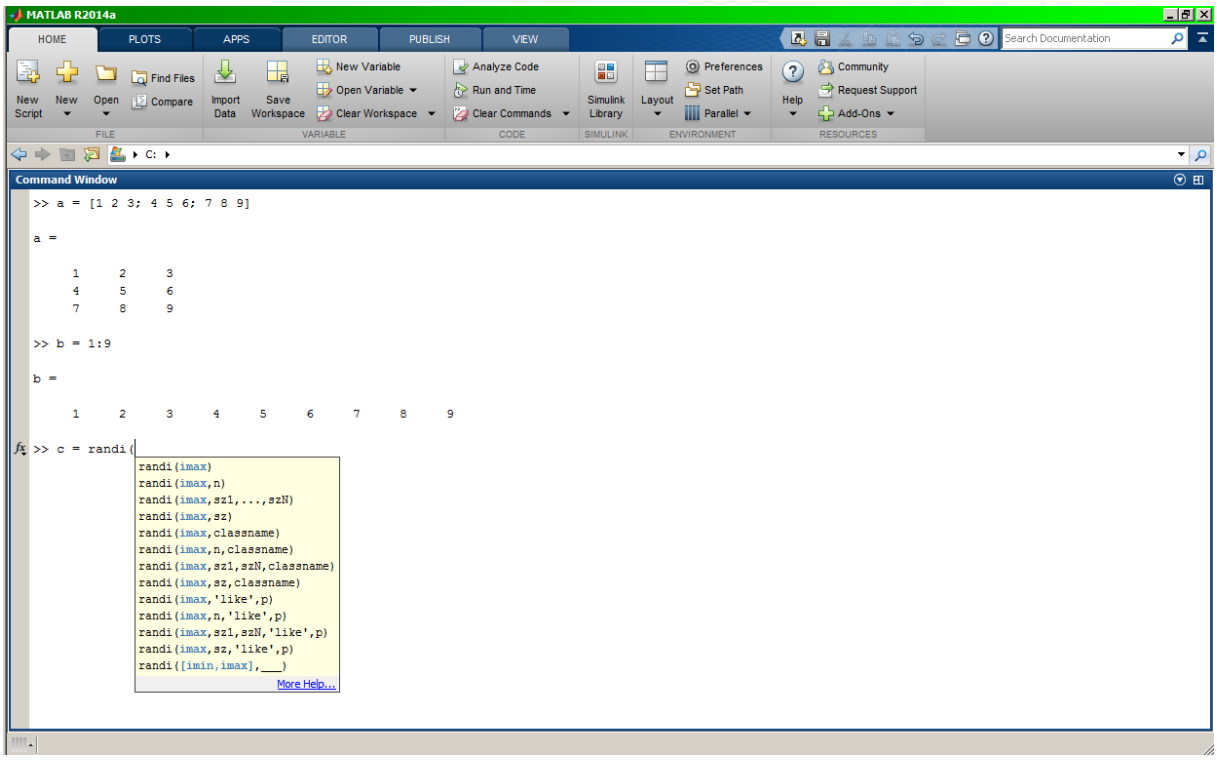
## 5.3 Práce v systému MATLAB

V Matlabu je možné pracovat dvěma způsoby. Prvním způsobem je práce s příkazovou řádkou, zvanou Command Window. Tento způsob je vhodný pro krátké rychlé výpočty, protože uživatel brzy ztrácí přehled o tom, co již zapsal, jelikož veškeré textové výstupy se vypisují do Command Window. Nicméně je možné veškerý postup v Command Window ukládat do souboru pomocí funkce *diary*.

Druhou možností je práce s textovým editorem, který je integrovanou komponentou Matlabu a v tomto editoru je možné psát libovolně dlouhé úseky kódu, tedy skripty. Ovšem pro spuštění těchto skriptů je vyžadováno uložit svůj kód do souboru. Pro tyto skripty má Matlab vlastní typ souboru, takzvaný M-file.

### 5.3.1 Command Window

Command Window je příkazová řádka neboli konzole pro veškerý textový vstup a výstup programu Matlab. Je možné zde definovat proměnné, volat funkce, měnit nastavení (pomocí příkazů). Matlab zde také vypisuje všechna upozornění a chyby, jež nastaly za běhu programu.




Obrázek 5.1 Ukázka práce s Command Window

### 5.3.2 Current directory – Aktivní složka

Current directory je složka, kterou systém MATLAB považuje za běhu programu jako používanou. Spouštěný skript se musí nacházet v této složce. Veškeré vlastní funkce či soubory k jakékoliv operaci, které nemají udanou celou cestu umístění, budou hledány právě v této složce.

### 5.3.3 Typy souborů používané systémem MATLAB

Systém MATLAB využívá několik vlastních typů souborů pro ukládání skriptů, grafických oken a ostatních dat. V následujícím seznamu si popíšeme nejdůležitější z nich.

#### 5.3.3.1 Soubory obsahující kód (přípona *.m)

Jsou to takzvané M-soubory (M-file), které obsahují skript či implementaci funkce. Chovají se stejně jako obyčejný textový soubor. Je možné je upravovat v kterémkoliv textovém editoru.

#### 5.3.3.2 Uzamčené soubory obsahující kód (přípona *.p)

Tento typ souboru je ve své podstatě taktéž M-soubor, avšak jeho obsah je uživatelům skryt a přečíst jej dokáže pouze Matlab. Jde o nevratné „zatemnění“ souboru. Do tohoto typu se převádí soubory pro autorskou ochranu kódů. Využívá se pro skrytí obsahu funkcí, které jsou v toolboxech anebo jsou komerčně (i nekomerčně) šířeny.

#### 5.3.3.3 Soubory grafických oken (přípona *.fig)

V souborech tohoto typu jsou uloženy veškeré informace týkající se grafických oken zvaných figure. Pro lepší představu, tento soubor je jako plán rozložení grafického okna, který obsahuje všechny komponenty (tlačítka, textová pole, grafy) a jejich nastavení, včetně informací o tom, jaká funkce se má spustit na jakou událost.

#### 5.3.3.4 Soubory pro uchování informací (přípona *.mat)

Matlab dokáže kteroukoliv proměnnou či skupinu proměnných uložit jako balíček dat do takzvaného MAT-souboru. Je možné zde uložit nespočet proměnných a vůbec nezáleží na tom, jakého datového typu tyto proměnné jsou.

### 5.3.4 Proměnné

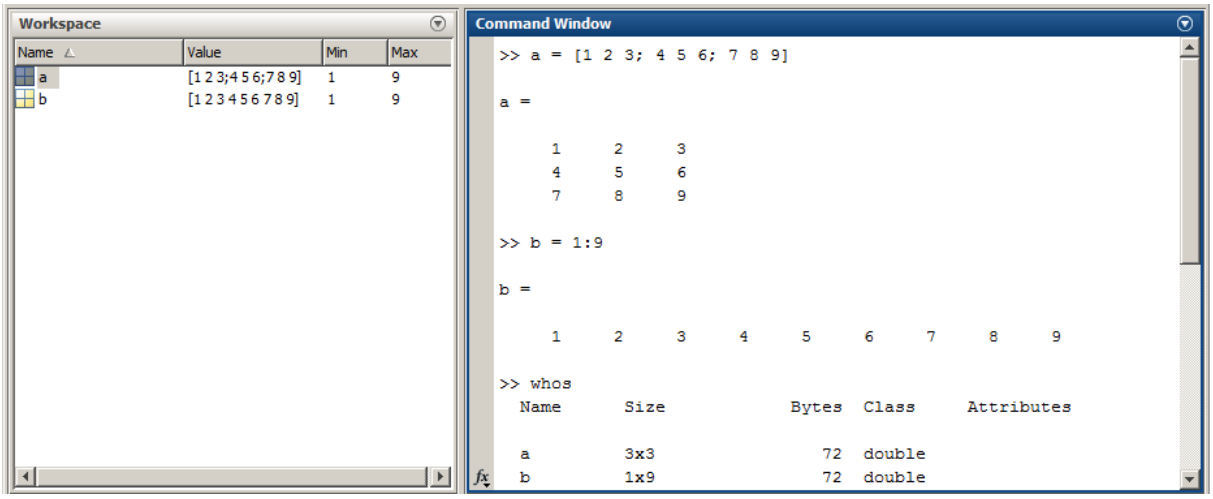
Jak již bylo řečeno výše, základem Matlabu jsou matice, čili na veškeré proměnné ať už jsou jakéhokoliv typu, Matlab pohlíží jako na matice. Jedná-li se o jednorozměrnou proměnnou, například číslo, jde o matici velikosti 1x1.

Datové typy všech proměnných určujeme při jejich definici vstupními hodnotami. Pokud bychom chtěli číslo o daném datovém typu, je zapotřebí použít k vytvoření některou z funkcí, kterou Matlab nabízí.

#### 5.3.4.1 Workspace

Jedná se o úložnou plochu pro proměnné, které jsou právě vytvořeny. Ve Workspace je možné tyto proměnné prohlížet, upravovat, mazat nebo třeba ukládat do MAT-souborů. Jde to i opačně, můžeme proměnné z MAT-souborů skrze Workspace zase načíst. Můžeme Workspace tedy nazvat takovým manažerem proměnných. Ve Workspace je zobrazena velikost a datový typ dané proměnné. Přepis do přehledné textové podoby nám zajišťuje funkce whos, která po zavolání vypíše všechny proměnné do Command Window.

Vedle hlavní Workspace, která je zobrazena v grafickém rozhraní Matlabu, existují ještě vedlejší Workspace, které jsou nám skryty. Vedlejší Workspace se vytváří jako uložiště proměnných pro běžící funkce. Pro každou funkci se vytváří vlastní Workspace, kam se za běhu funkce ukládají vnitřní proměnné. Tyto Workspace nemůžeme vidět, jedinou možností, jak přistupovat k proměnným z vedlejších Workspace, je přiřadit k nim nějakou proměnnou z hlavní Workspace pomocí funkce *assignin*, kterou musíme zavolat uvnitř funkce, jejíž proměnné chceme odchytit.




Obrázek 5.2 Ukázka Workspace a přepisu do Command Window pomocí funkce whos

**Poznámka**

Ještě než začneme s popisem možností definování proměnných, je nutné uvést fakt, že proměnné, které jsou již ve Workspace můžeme používat v rámci Command Window a jakéhokoliv skriptu (ovšem nikoliv funkce) do doby než budou smazány. Pro názornost si představme, že v Command Window nadefinujeme proměnnou *A* nebo ji nahrajeme skrze Workspace ze souboru a poté spustíme skript, který pracuje s proměnnou, která se také jmenuje *A*. Matlab nebude mít žádný problém s tím, že proměnná *A* není nadefinována v rámci spouštěného skriptu. Je tedy důležité pochopit, že se Matlab dívá na kód a proměnné svým způsobem odděleně a také to, že kód se spouští postupně od shora dolů a nekontroluje jej jako celek.

#### 5.3.4.2 Definice číselné proměnné

Proměnnou můžeme nadefinovat několika způsoby. Bude uveden vždy typ proměnné a pak příklad v kódu s následujícím slovním komentářem.

**Jednorozměrné proměnné**

```
a = 10
b = .5;
```

Úsek kódu 5.1

Tento úsek kódu nám definuje dvě proměnné. Proměnnou *a*, která bude mít hodnotu 10. Proměnnou *b*, která má hodnotu 0.5, kde si povšimněme, že je možné nulu vynechat. Jak můžete dále vidět, za proměnnou *a* není napsán středník, kdyžto za proměnnou *b* ano. Středník má v kódu jeden poměrně zásadní účel a tím je výpis výsledku do Command Window. Proměnná *a* se nám tedy uloží do Workspace a vypíše do Command Window, přičemž proměnná *b* se nám pouze uloží.

**Vektory**

```
a = [1, 2, 3];
b = [4; 5; 6];
```

Úsek kódu 5.2

Základem definování vícerozměrné proměnné jsou hranaté závorky. Zde definujeme dva vektory, které se na první pohled můžou zdát stejné, ale pro objasnění si hned vysvětlíme rozdíly. Proměnná *a* je sloupcový vektor, tedy matice o velikost 1x3 a proměnná *b* je vektor řádkový, tedy matice 3x1. Z toho vyplývá, že čárka odděluje sloupce a středník řádky.

**Matice**

```
A = [1, 2, 3; 4, 5, 6; 7, 8, 9];
B = [1 2; 3 4];

C = [a; a];
```

Úsek kódu 5.3

Definice matice je úplně stejná jako definice vektoru, akorát přidáme další dimenze. Matice *A* má tedy rozměry 3x3 a matice *B* 2x2. Dále si můžete povšimnout, že jsou při definování matice *B* vynechány čárky a prvky jsou odděleny pouze mezerami. Matlab nabízí také tuto možnost zápisu pro oddělování sloupců matice. A pro představu, jak snadno se dá v Matlabu pracovat s maticemi, je zde uvedena matice *C* o rozměrech 2x3, která vzniká spojením dvou sloupcových vektorů *a* z předchozí ukázky.

#### 5.3.4.3 Buňky

Buňka je speciální typ matice, která dokáže pojmout za každý prvek jiný datový typ, tedy každý prvek této matice může mít jinou velikost. Můžeme proto v buňce kombinovat všechny datové typy a také rozměry. Například můžeme do jednoho prvku vložit dvourozměrnou matici, do dalšího jen číslo, do dalšího text a tak dál, více v úseku kódu níže.

```
C = {[1 2; 3 4], 'Tohle je text'; 1000, @ukazatel_funkce};
```

Úsek kódu 5.4

Buňka se podobně jako běžná matice definuje závorkami, ovšem závorky jsou složené, pak už platí stejná pravidla jako u matice. Čárky oddělují sloupce a středníky řádky. V tomto úseku kódu definujeme buňku *C* o rozměrech 2x2, kde na první pozici je dvourozměrná matice, na další textový řetězec, pak na dalším řádku jednorozměrné číslo a nakonec ukazatel na funkci. Je tedy vidět, že opravdu můžeme skombinovat různé typy proměnných.

#### 5.3.4.4 Definice textové proměnné

U textových proměnných bohužel neplatí úplně stejná pravidla jako u číselných a to z toho důvodu, že textový řetězec je již sám o sobě vektor znaků, proto s ním nemůžeme pracovat stejně jako s čísly.

**Textový řetězec**

```
a = 'Tohle je textový řetězec';
```

Úsek kódu 5.5

V ukázce definujeme proměnnou *a*, která je vektorem znaků, tedy textovým řetězcem. Hlavním rozdílem mezi ostatními programovacími jazyky je to, že se zde textové řetězce zapisují do apostrofů na rozdíl od uvozovek.

**Vektor textových řetězců**

```
M = ['Řádek1'; 'Řádek2'; 'Řádek3'; 'Řádek4'];

T = {'Pondělí'; 'Úterý'; 'Středa'; 'Čtvrtek'; 'Pátek'; 'Sobota'; 'Neděle'};
```

Úsek kódu 5.6

Pokud chceme vytvořit vektor řetězců, jak již bylo řečeno výše, nemůžeme použít stejný postup jako s číselnými hodnotami. V ukázce výše vidíme, že se pravděpodobně jedná o dva vektory textu. Ovšem mezi proměnnými *M* a *T* jsou značné rozdíly. V případě proměnné *M* totiž nevytváříme vektor textových řetězců, nýbrž matici znaků o velikosti 4x6 a tento zápis je možný pouze pokud jsou všechny řetězce stejně dlouhé. Kdyby nebyly, Matlab zareaguje chybovou hláškou o nerovnoměrných velikostech jednotlivých řádků matice. Proto se vektor textových řetězců zapisuje jako buňka, která je popsána výše. Proměnná *T* je tedy buňka o rozměrech 7x1.

#### 5.3.4.5 Struktura

Struktura je dalším speciálním typem proměnné. Jedná se o takový balíček, do kterého můžeme vkládat další proměnné různých typů. Můžeme mít tedy pod jedním jménem skupinu proměnných týkající se stejného problému.

```
Struktura.cislo = 1;
Struktura.text = 'Text';
Struktura.matice = [1, 2, 3; 4, 5, 6; 7, 8, 9];

Struktura2.struktura = Struktura;
```

Úsek kódu 5.7

Při definování struktury se používá zápis, kdy první je název struktury, pak následuje tečka, název vnitřní proměnné a nakonec rovná se s hodnotou proměnné. Do struktury *Struktura* postupně ukládáme číslo, text i celou matici a jak můžete vidět na struktuře *Struktura2* můžeme do vnitřních proměnných ukládat i celou strukturu a vniká nám tedy struktura ve struktuře.

### 5.3.5 Základní operace

#### 5.3.5.1 Indexování

Indexování je jednou z nejdůležitějších věcí při práci s maticemi v Matlabu. Ovšem oproti klasickému programování, kde všude je první pozice označená jako nultá, v Matlabu je prvním indexem číslo jedna. Například, budeme-li mít matici o rozměrech 2x2, bude první prvek označen indexy jako pozice 1,1. Pro získání některého prvku z matice, napíšeme za názvem proměnné kulaté závorky a uvnitř indexy oddělené čárkou. Pokud se jedná o buňku, zápis je obdobný, s tím rozdíle, že závorky budou složené. Ukázka v úseku kódu níže.

```
A = [1, 2, 3; 4, 5, 6; 7, 8, 9];
A(2,1)

C = {[1 2; 3 4], 'Tohle je text'; 1000, @ukazatel_funkce1};
C{1,2}
```

Úsek kódu 5.8

```
ans =
            4

ans =
            Tohle je text
>>
```

Výstup do Command Window z úseku kódu 5.8

**Interval indexů (Dynamické indexování)**

Dalším důležitou znalostí je výběr rozsahu indexů, například potřebujeme vybrat celý sloupec či řádek matice nebo jen jeho část. Matlab má pro tyto případy speciální znak a tím je dvojtečka. Chceme-li vybrat celý sloupec či řádek, zadáme na místo čísla indexu dvojtečku a tím říkáme, že chceme všechno. Chceme-li vybrat určitý rozsah, tak na místo čísla indexu zadáme čísla od – do oddělené dvojtečkou. Více v následujícím úseku kódu.

```
A = [1, 2, 3; 4, 5, 6; 7, 8, 9]

b = A(:,1)

c = A(2,:)

d = A(1:2, 2:3)
```

Úsek kódu 5.9

```
A =
      1     2     3
      4     5     6
      7     8     9

b =
      1
      4
      7

c =
      4     5     6

d =
      2     3
      5     6
```

Výstup do Command Window z úseku kódu 5.9

Opět definujeme matici *A*, která má rozměry 3x3. V prvním případě (proměnná *b*) chceme vypsat všechny řádky prvního sloupce. V druhém případě (proměnná *c*) všechny sloupce druhého řádku a ve třetím případě (proměnná *d*) chceme vypsat řádky jedna až dvě ze sloupce dvě až tři.

**Poslední index**

Poslední index je užitečnou vlastností, pokud potřebujeme vypsat poslední prvek například z vektoru o neznámém počtu prvků nebo pokud vektor rozšiřujeme a chceme přidat nový prvek na konec. Tento poslední index se značí slovíčkem *end*.

```
v = [1, 2, 3, 4, 5];

v(end)

v(end+1) = 6

v(3:end)
```

Úsek kódu 5.10

```
ans =
      5

v =
      1     2     3     4     5     6

ans =
      3     4     5     6
```

Výstup do Command Window z úseku kódu 5.10

Jak je vidět, definujeme řádkový vektor *v*, který má pět prvků. Nejprve vypíšeme jeho poslední prvek. V dalším případě přidáváme na konec další prvek a tím je číslo šest. Nakonec si necháváme vypsat všechny prvky od třetího prvku až na konec.

#### 5.3.5.2 Generování posloupností

Potřebujeme-li rychle a snadno vytvořit posloupnost čísel, tak speciální znak dvojtečka, zmíněný výše v kapitole Indexování, je možné použít také zde. Tento znak se v Matlabu využívá na více místech. Dalším využitím bude například cyklus *for*. Chceme-li generovat posloupnost s určitým krokem, uvedeme počáteční hodnotu, oddělíme dvojtečkou, uvedeme krok, opět oddělíme dvojtečkou a uvedeme koncovou hodnotu posloupnosti. Více o práci s generováním posloupností v úseku kódu níže.

```
v = 0: .25 :1

A = [1:3; 4:6; 7:9]
```

Úsek kódu 5.11

```
v =
      0    0.2500    0.5000    0.7500    1.0000

A =
      1     2     3
      4     5     6
      7     8     9
```

Výstup do Command Window z úseku kódu 5.11

Pro vektor *v* definujeme posloupnost od 0 do 1 s krokem 0.25 (Pro lepší přehlednost v kódu uvedeno s mezerami). Dále definujeme matici *A*, která vzniká spojením tří posloupností, tedy tří řádkových vektorů, které jsou, pokud není krok uveden, s krokem 1.

#### 5.3.5.3 Práce s maticemi a vektory

**Transponování**

Chceme-li transponovat matici či vektor, je to velmi snadná operace, kterou zajišťuje speciální znak a tím je apostrof. Pro transponování vektoru či matice jednoduše napíšeme její název a následně apostrof. Ukázka v následujícím úseku kódu.

```
A = [1, 2; 3, 4]

A'
```

Úsek kódu 5.12

```
A =
      1      2
      3      4

ans =
      1      3
      2      4
```

Výstup do Command Window z úseku kódu 5.12

V tomto úseku kódu vidíme, jak jednoduše můžeme transponovat matici *A*.

**Sčítání a odčítání matic**

Sčítat a odečítat matic se provádíme úplně stejně jako sčítání a odečítání jednorozměrných proměnných. Tuto operace je možné použít pouze v případě, kdy jsou rozměry sčítaných či odčítaných matic totožné. Výsledkem operace je matice stejných rozměrů, která je naplněna hodnotami součtu či rozdílu prvků na dané indexové pozici.

Velmi jednoduše můžeme také přičíst či odečíst od všech prvků v matici konstantu.

```
A = [1,2;3,4];

B = [5,6;7,8];

A + B

A - 5
```

Úsek kódu 5.13

```
ans =
      6     8
      10    12

ans =
      -4    -3
      -2    -1
```

Výstup do Command Window z úseku kódu 5.13

**Násobení a dělení konstantou**

Násobení se provádí pomocí znaku * a dělení pomocí znaku /. Při použití této operace jednoduše vynásobíme či podělíme všechny prvky matice naší zadanou konstantou.

```
A = [1,2;3,4];

A * 5

A / 2
```

Úsek kódu 5.14

```
ans =
       5    10
      15    20


ans =
      0.5000    1.0000
      1.5000    2.0000
```

Výstup do Command Window z úseku kódu 5.14

**Násobení a dělení matic**

Pro násobení a dělení matic opět použijeme znaky * a / a tato operace může proběhnout dvěma způsoby.

Prvním způsobem je běžné maticové násobení a dělení tak, jak je známe z matematiky. U násobení tedy řádek krát sloupec. A u dělení jsou možnosti dvě. Pravostranné a levostranné dělení. Můžeme totiž použít / a \, a tím určit, která strana bude kterou dělená. V případě $A \,/\, B$ jde o dělení zprava ($A \cdot B^{-1}$) a v případě $A \setminus B$ jde o dělení zleva ($A^{-1} \cdot B$).

Druhým způsobem je násobení a dělení po prvcích. V tomto případě musí mít matice totožné rozměry. Před znaky pro násobení a dělení se přidá tečka, a tím říkáme, že chceme násobit či dělit po prvcích, tedy prvky na stejných indexových pozicích se mezi sebou násobí či dělí.

```
A = [1,2;3,4];
B = [5,6;7,8];

A * B

A / B

A .* B

A ./ B
```

Úsek kódu 5.15

```
ans =
      19    22
      43    50

ans =
      3.0000   -2.0000
      2.0000   -1.0000

ans =
       5    12
      21    32

ans =
      0.2000    0.3333
      0.4286    0.5000
```

Výstup do Command Window z úseku kódu 5.15

## 5.3.6 Podmínky

Podmínky jsou funkční příkazy, které vytváří bloky kódu, které se spouští v závislosti na platnosti dané podmínky. Každý blok podmínky začíná funkčním slovem a ukončuje se slovem *end*.

### 5.3.6.1 Podmínka if

Základní typ podmínky, blok začíná slovem *if* a může být rozšířen slovem *else*, neboli jinak. Tento blok se používá, pokud chceme zajistit blok kódu pro platnost i neplatnost podmínky. Chceme-li porovnávat více podmínek, je zde ještě možnost použít slovo *elseif*, kdy podmínka začíná blokem *if*, následují bloky *elseif* a případně *else*, když žádná podmínka nevyhovuje.

Pokud máme bloků a tedy i podmínek k testování moc, je výhodnější použít přepínač, *switch*, více níže.

```
a = 1;

if a == 1
    a = 2;
elseif a == 2
    a = 3;
else
    a = 1;
end
```

Úsek kódu 5.16

#### 5.3.6.2 Switch

Tento typ podmínky se nazývá přepínač a už podle názvu je jeho funkce jasná. Switch přechází na úseky kódu v těle svého bloku podle hodnoty zadané vstupní proměnné.

Využívá se v případech, kdy máme proměnnou a velký počet úkonů při jejích různých hodnotách. Blok začíná slovem *switch*, vnitřní úseky kódů se označují slovem *case* následovaným hodnotou proměnné. Nakonec každého úseku se dává slovo *break*, které způsobí vystoupení programu z bloku přepínače.

Pro případ, kdy žádná podmínka, tedy žádný úsek kódu, nevyhovuje hodnotě proměnné, používá se úsek kódu, který začíná slovem *otherwise*.

```
switch a
    case 1
        % Implementace pro případ, kdy se a rovná 1
        break;

    case 2
        % Implementace pro případ, kdy se a rovná 2
        break;

    otherwise
        % Implementace pro případ, kdy se a nerovná ani 1, ani 2
end
```

Úsek kódu 5.17

### 5.3.7 Cykly

Cykly jsou bloky kódu, které se opakují, dokud je splňována určitá podmínka.

#### 5.3.7.1 Cyklus for

Tento typ cyklu má konečný počet opakování a je dán vektorem definovaným za funkčním slovem *for*. Vektor musí být řádkový, ale nemusí jít o po sobě jdoucí celá čísla. Proměnná, které je vektor přiřazen, je pomocná proměnná, která se využívá uvnitř cyklu. Více v názorné ukázce v úseku kódu níže.

```
for a = [.5, 8, 100]
    disp(['Proměnná a = ', num2str(a)]);
end
```

Úsek kódu 5.18

```
Proměnná a = 0.5
Proměnná a = 8
Proměnná a = 100
```

Výstup do Command Window z úseku kódu 5.18

#### 5.3.7.2 Cyklus while

Tento cyklus není konečný a je závislý na vstupní podmínce, která se vyhodnocuje vždy na začátku bloku, tedy před vstupem a každým opakováním. Tento typ cyklu je jednoduché zacyklit, protože hodnoty pro podmínku cyklu jsou upravovány uvnitř cyklu a stačí malá chyba, aby vznikla nekonečná smyčka.

Blok cyklu začíná slovem *while*, následuje podmínka a končí slovem *end*.

```
a = 1;

while a < 5
   a  = a + 1;
end
```

Úsek kódu 5.19

#### 5.3.7.3 Cyklus parfor

Tento typ cyklu je takřka stejný jako cyklus *for*, ovšem rozdílem je v tom, že tento cyklus je využíván při paralelním výpočtu. Z tohoto užití plyne, že proměnné uvnitř bloku nesmí být závislé na výsledcích z předešlých kroků.

### 5.3.8 Funkce

Funkce jsou příkazy, které v sobě skrývají blok kódu, jenž se vykonává při zavolání funkce. Tento blok kódu může mít nějaké vstupy a nějaké výstupy, ovšem nemusí mít ani jedno a výstupem funkce bude samotné provedení bloku kódu uvnitř funkce.

Je zde nutno zmínit, že každá funkce si vytváří svou vlastní oddělenou Workspace, která není nijak propojena s hlavní Workspace. Dále platí, že funkce vnořená může přistupovat do Workspace funkce nad ní, ale nikoliv opačně.

Systém MATLAB nabízí nespočet funkcí, které jej činí velmi výkonným v mnoha oblastech. Dá se říct, že takřka jakákoliv oblast má své zastoupení mezi funkcemi ať už v základní verzi nebo v přidružených toolboxech.

Máme několik typů funkcí. Níže popíšu jednotlivé typy a způsob jejich definice.

#### 5.3.8.1 Obyčejná funkce

Funkce je definována podle vzoru níže a musí být uložena v souboru typu M se stejným názvem, jako je název funkce. Pokud chceme tuto funkci zavolat ve skriptu, musí být soubor funkce ve stejné složce jako skript.

```
function [vystup1, vystup2] = NazevFunkce(vstup1, vstup2)
% Implementace funkce
end
```

Úsek kódu 5.20

#### 5.3.8.2 Vedlejší funkce

Vedlejší funkce je v podstatě stejná jako funkce uvedená výše, rozdílem je, že se nachází pod funkcí hlavní, obyčejnou funkcí, podle které je pojmenovaný soubor, a tuto vedlejší funkci je pak možné použít pouze v rámci souboru hlavní funkce. Tedy v hlavní funkci a ostatních vedlejších či vnořených funkcích.

#### 5.3.8.3 Vnořená funkce

Vnořená funkce je definovaná v rámci jiné funkce. Vnořenou funkci můžeme použít pouze v rámci mateřské funkce.

```
function [] = HlavniFunkce()

VnorenaFunkce();

    function VnorenaFunkce()
        VedlejsiFunkce();
    end
end

function [] = VedlejsiFunkce()
% Implementace
end
```

Úsek kódu 5.21 Ukázka definice vedlejší a vnořené funkce

#### 5.3.8.4 Anonymní funkce

Tento typ funkce je definován podobně jako proměnná. Tuto funkci můžeme definovat v jakékoliv části kódu a není nutné pro ni vytvářet vlastní soubor. Používá se jako tzv. makro.

```
AnonymniFunkce = @(x) x^2 + 3*x + 2;

A = AnonymniFunkce(2);
```

Úsek kódu 5.22

### 5.3.9 Grafické výstupy

V této kapitole budou popsány některé funkce, které zajišťují vykreslení dat do grafického okna. Základem práce s grafickými výstupy je funkce *figure*, která vytvoří nové grafické okno, do kterého je pak možné přidávat grafy, případně jiné grafické komponenty (tlačítka, textová pole, atd.).

#### 5.3.9.1 Funkce subplot

Tato funkce zajišťuje členění grafického okna do mřížky. Takto můžeme jednoduše v jediném grafickém okně vykreslit například čtyři grafy či jiné grafické výstupy.

#### 5.3.9.2 Funkce plot

Tato funkce zajišťuje vykreslení lineárních grafů. Po zavolání buď vykreslí data do grafu v aktivním grafickém okně (figure) nebo vytvoří nové a vykreslí je tam.

#### 5.3.9.3 Funkce imshow

Tato funkce zajišťuje vykreslení obrazových matic do grafického okna. Funkce je schopna vykreslit takřka jakýkoliv typ obrazových dat.

#### 5.3.9.4 Doplňující funkce grafů – nadpis, popisky os

Každý graf by měl být opatřen nadpisem a popiskami os. V systému MATLAB pro tento účel slouží k nastavení nadpisu funkce *title* a k popiskům os funkce *xlabel* a *ylabel*.

### 5.3.10 Souborové vstupy a výstupy

Systém MATLAB obsahuje nespočet funkcí pro načítání a ukládání souborů různých typů. Dokáže pracovat jak na binární, tak na textové úrovni. Nejdůležitější funkce užitá v rámci této práce k načítání souborů je funkce *dicomread*.

#### 5.3.10.1 Funkce fopen

Funkce k otevření souboru pro různé typy operací (čtení, zápis) a vytvoření ukazatele na daný soubor. Po dokončení práce se souborem je nutné soubor uzavřít pomocí funkce *fclose*.

#### 5.3.10.2 Funkce dicomread

Tato funkce je schopna načíst soubory typu DICOM. Vstupem je cesta k souboru. Výstupem je matice obsahující hodnoty absorpce záření.

#### 5.3.10.3 Funkce fread

Funkce čte soubory na binární úrovni. Pro použití funkce je nejprve nutné vytvořit ukazatel a otevřít soubor pomocí funkce *fopen*.

#### 5.3.10.4 Funkce fscanf

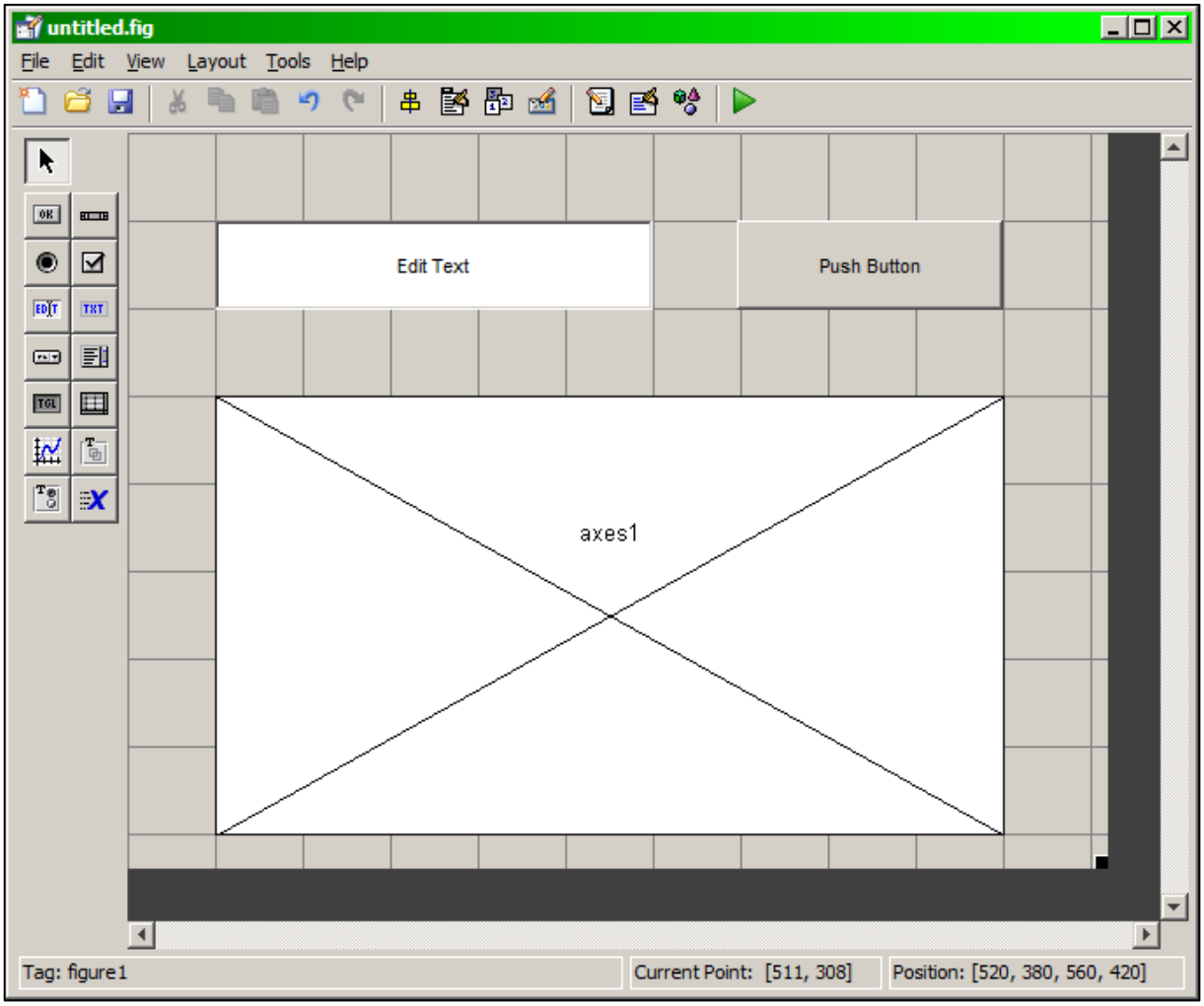
Tato funkce čte soubor na textové úrovni. Vstupem je ukazatel navrácený funkcí *fopen*.

### 5.3.11 Tvorba grafických prostředí (GUI)

Ačkoliv k tomu není primárně určen, je možné v systému MATLAB vytvářet grafická rozhraní, ovšem jejich dokonalost tomu také odpovídá. Vytvořit grafické rozhraní je možné dvěma způsoby.

#### 5.3.11.1 Tvorba pomocí GUIDE

První možností je grafický průvodce tvorbou grafického prostředí zvaný GUIDE. Tohoto průvodce vyvoláme zavoláním stejnojmenné funkce *guide*. V tomto rozhraní je možné jednoduše vkládat komponenty, jako jsou tlačítka, textová pole, grafy a tak dále. Vývojář má okamžitý přehled o grafickém vzezření aplikace. Výstupem je soubor typu FIG, který nese rozložení grafických komponent, a soubor typu M, který nese veškerý kód vztahující se k danému grafickému rozhraní.




Obrázek 5.3 Ukázka grafického průvodce tvorbou grafického prostředí GUIDE

#### 5.3.11.2 Tvorba skrze kód

Druhou možností je tvorba grafického rozhraní skrze kód. Výhodou tohoto postupu je větší nastavitelnost výsledného rozhraní, přehlednost komponent a jejich implementace, a také snížení počtu souborů na pouze jeden soubor typu M.

Základem tvorby grafického rozhraní je vytvoření grafického okna pomocí funkce *figure*. Další komponenty je možné umístit do tohoto okna pomocí funkce *uicontrol* a nastavení jejich vlastností pomocí funkce *set*.

Nejlepším postupem při volbě této možnosti vývoje, je vytvoření hlavní funkce, ve které bude veškerá implementace rozložení grafického okna. Odezvy jednotlivých komponent a ostatní funkce je pak možné implementovat jako funkce vnořené či vedlejší.

#### 5.3.11.3 Funkce figure

Tato funkce zajišťuje vytvoření nového grafického okna. Jejím výstupem je ukazatel na sebe sama a vstupy mohou být různá nastavení vlastností okna, jako jsou například nadpis, číslo okna, velikost, měnitelnost velikosti, atd.

#### 5.3.11.4 Funkce uicontrol

Výstupem funkce je vytvoření grafické komponenty a navrácení ukazatele na tuto komponentu. Ve vstupních parametrech je pak možné nastavit typ této komponenty, například tlačítko.

#### 5.3.11.5 Funkce get

Funkce *get* navrací hodnotu či hodnoty vlastností dané grafické komponenty, jejíž ukazatel byl uveden jako vstup.

#### 5.3.11.6 Funkce set

Funkce *set* nastavuje hodnotu či hodnoty vlastností dané grafické komponenty, jejíž ukazatel byl uveden jako vstup.

# 6 Propojení systému MATLAB s jazykem C#

Propojení systému MATLAB s jazykem C# je možné několika způsoby. Jedním způsobem je kompilace kódu do podoby knihovny, na kterou je pak možné se v kódu odkázat a z ní zavolat danou funkci. Tento způsob je poměrně pomalým (z pohledu implementace) a velice neohebným řešením, protože při jakékoliv změně ve funkci, je nutné vytvořit novou knihovnu. V tomto případě ovšem není nutné mít na cílovém počítači nainstalovaný systém MATLAB.

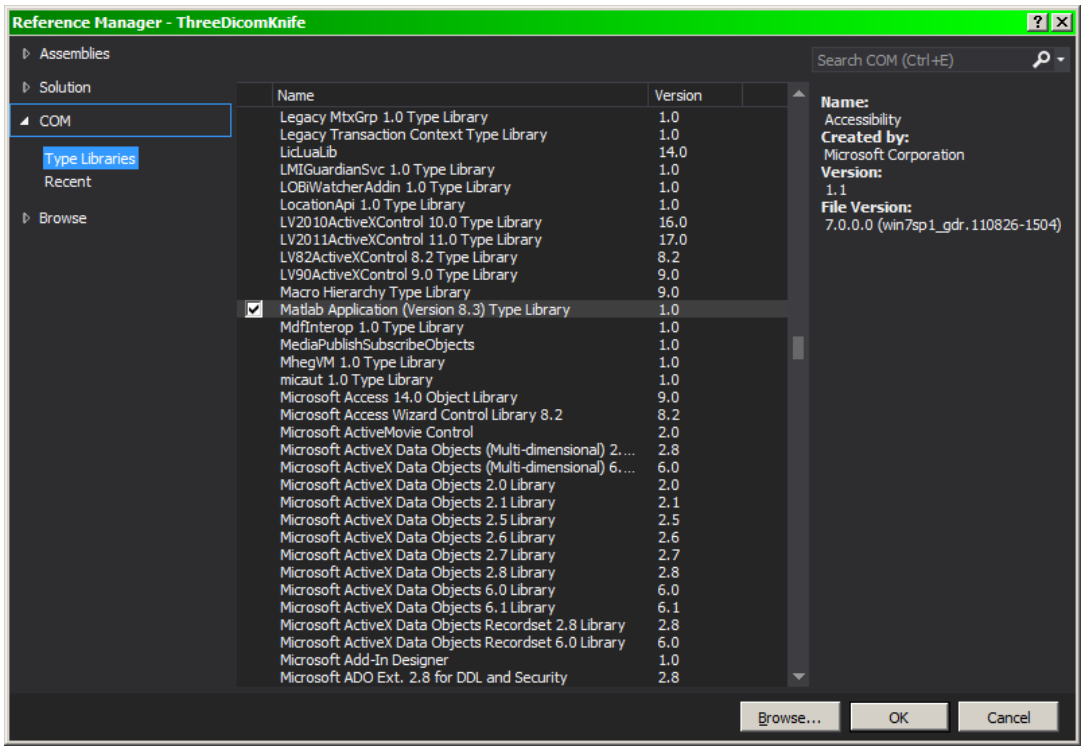
Jednodušším, rychlejším (z pohledu implementace) a ohebnějším řešením je přidání reference na systém MATLAB v projektu vytvořeném v jazyce C#, a tím vytvoření klienta, který bude připojen na COM server systému MATLAB. Jak již plyne z kontextu, v tomto případě je nutné mít na cílovém počítači nainstalovaný systém MATLAB.

## 6.1 Vytvoření spojení systému MATLAB s jazykem C# v podobě COM serveru

### 6.1.1 COM object

Jedná se o způsob komunikace mezi dvěma softwarovými komponentami. Přenos je zprostředkováván na binární úrovni, a proto nezáleží, na jaké platformě se dané komponenty nachází.

### 6.1.2 Přidání reference v projektu jazyka C#




Obrázek 6.1 Přidání COM objektu do referencí projektu jazyka C#

### 6.1.3 Vytvoření COM objektu v jazyce C#

Po přidání reference v projektu, můžeme vytvořit instanci třídy *MLApp.MLApp*, která je jednoduše ukazatelem na COM objekt systému MATLAB. Skrze tuto instanci pak řešíme veškerou komunikaci.

#### 6.1.3.1 Metoda Execute

Metodu voláme z instance třídy *MLApp.MLApp*. Do této metody vstupuje textový řetězec obsahující kód v jazyce systému MATLAB, který, jednoduše řečeno, posíláme do Command Window. Návratovou hodnotou je, taktéž, textový řetězec obsahující výstup z Command Window.

```
MOut = Matlab.Execute(@"A = bitshift(uint16(gather(A)), -4);");
```

Úsek kódu 6.1 Ukázka použití metody Execute

#### 6.1.3.2 Metoda GetVariable

Metodu voláme z instance třídy *MLApp.MLApp*. Vstupem je název proměnné, která by se měla nacházet ve Workspace serveru systému MATLAB a typ Workspace. Návratem je obsah proměnné ve stejném datovém typu.

```
UInt16[,] image = Matlab.GetVariable("A", "base");
```

Úsek kódu 6.2 Ukázka použití metody GetVariable

# 7 Definice problému

## 7.1 Překryv cíle jinou tkání

Jak již bylo zmíněno na úplném začátku v úvodu, hlavním problémem, který tato práce řeší, je možnost sumace a tím i překryvu ozařovaného cíle tkání s vyšší denzitou. V takovém případě je nádorové ložisko pro systém CyberKnife viditelné pouze na jedné rentgence či úplně neviditelné. Tento problém je možné řešit klíny, kterými se pacient na plánovacím CT podloží, a tím bude rotován o úhel potřebný k odkrytí cíle na zaměřovacích snímcích.

Klín, který bude použit, se momentálně vybírá odhadem a je tedy možné, že cíl nebude viditelný ani po rotaci, což vede k dalšímu opakování CT vyšetření, které je náročné nejen na čas, ale také zvyšuje radiační zátěž pacienta.

Je proto potřeba zajistit správný výběr úhlu, o který bude pacient rotován, aby byla zajištěna co nejlepší viditelnost cíle na obou rentgenkách.

## 7.2 Současné možné řešení

V současné době pro systém CyberKnife existuje softwarové rozšíření, které se nazývá OneView. Při použití tohoto rozšíření je možné se řídit informacemi získanými pouze z jedné rentgenky, tedy obejít neviditelnost cíle na jedné z rentgenek, ovšem za cenu snížení efektivity v podobě zvětšení ochranného lemu, a tím ozáření většího objemu zdravé tkáně. Navíc je toto rozšíření nákladnou záležitostí, což přináší onkologické klinice jen další nevýhody a žádná plus.

# 8 Návrh a postupný vývoj řešení

## 8.1 Představa řešení

Ideou řešení bylo vytvořit aplikaci ve výpočetním systému MATLAB, která načte, zpracuje CT data a na jejich základě vytvoří snímky pod úhlem 45 stupňů, které budou dále ještě rotovány o úhel klínu. Tento úhel by pak mohl technik měnit a vidět okamžitě změnu zaměřovacích snímků.

Součástí aplikace mělo být také základní zpracování a úprava obrazu jako jsou změny kontrastu, jasu, inverze obrazu a také možnosti lupy, tedy zvětšení vybrané oblasti snímku. Důležitou součástí měla být také možnost zobrazení kontury cíle pro snazší orientaci technika na výsledných snímcích. Celá aplikace pak měla mít grafické rozhraní vytvořené v sytému MATLAB.

## 8.2 Výsledné řešení

Výsledná aplikace je vytvořena jako grafický klient, implementován v jazyce C#, připojený na COM server systému MATLAB, který prostřednictvím spouštění funkcí implementovaných v jazyce systému MATLAB zajišťuje načtení a zpracování CT dat, ale také výpočet snímků pod úhlem 45 stupňů. Mimo to, zajišťuje systém MATLAB také výpočet rotované kontury cíle, kterou klient následně umožňuje vykreslit do vytvořených snímků.

Doplňující funkce zpracování obrazu, kterými jsou úprava kontrastu a jasu, inverze obrazu, možnosti lupy a také úprava gamy, zajišťuje přímo klient, jsou tedy implementovány v jazyce C#.

## 8.3 Postup řešení

V této kapitole bude popsán veškerý postup řešení této práce v chronologickém pořadí tak, jak k němu docházelo. Veškeré časové údaje, není-li uvedeno jinak, jsou při výpočtu na jedno jádrovém systému čipů, za nejpříznivějších podmínek a jsou vždy průměrem několika měření z 389 snímků.

### 8.3.1 Základy načtení a práce s CT daty

Základem celého řešení bylo zajistit správné načtení CT souboru, které jsou ve formátu DICOM. Díky balíčku funkcí pro zpracování obrazu (Image Processing Toolbox), podporuje systém MATLAB načítání souborů ve formátu DICOM pomocí funkce *dicomread*. Po zavolání této funkce jsou data z CT snímku načtena jako obyčejná matice a je pak tedy možné s nimi tak i zacházet.

Další důležitou funkcí, z výše zmíněného balíčku, je funkce *dicominfo*, která umožňuje načíst veškeré informace skryté ve struktuře souboru formátu DICOM. Mezi těmito informacemi je možné nalézt například osobní údaje pacienta, údaje o nemocnici a přístroji, který snímky vytvořil, technické informace o snímku, jako například rozměry, bitová hloubka, odchylka či násobek dat.

### 8.3.2 První verze algoritmu pro snímky z úhlu 45 stupňů (3d_diag)

První pokusy o vytvoření algoritmu zahrnovaly pouze vytvoření snímků z úhlu 45 stupňů. Byl řešen triviálním přístupem a bez jakéhokoliv ohledu na výslednou rychlost.

**Princip**

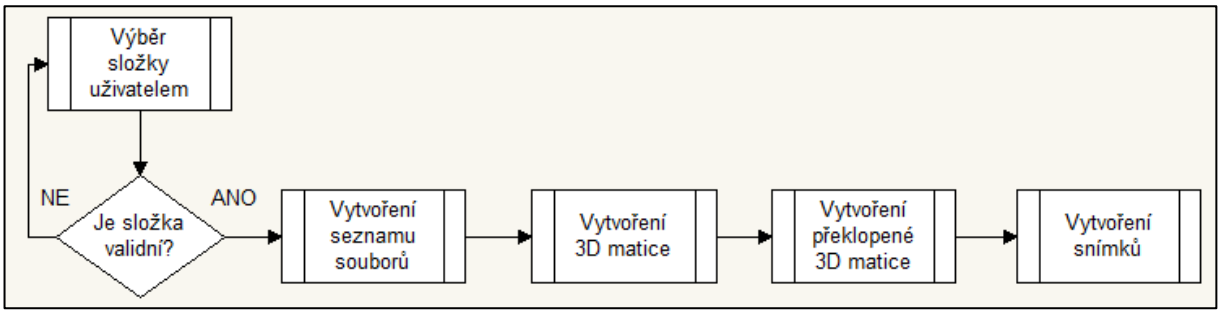
Uživatel zadá pomocí grafického průvodce cestu ke složce obsahující DICOM soubory. Ze složky jsou vybrány soubory se specifickým názvem (koncovka DCM) a jsou spočítány. Pokud je počet souborů větší než 100, je složka vyhodnocena jako validní. V opačném případě je uživatel odkázán znovu na výběr složky.

Pokud je složka vyhodnocena jako validní, je následně vytvořen seznam celkových cest k jednotlivým souborům. Z tohoto seznamu pak je po jednotlivých snímcích poskládána 3D matice s CT záznamem pacienta a hned poté také druhá 3D matice, kde jsou hodnoty v řádcích v opačném pořadí. První matice zastupuje pacienta pro rentgenku A a druhá matice pro rentgenku B.

Nakonec, je pomocí funkce *max* nalezeno maximum v každé diagonále, které prochází funkce *diag* v každém řezu pacienta. Výsledný snímek má tedy rozměry (512+512-1) x [počet snímků] a každý pixel (voxel) je maximem v dané diagonále.

**Rychlost výpočtu**

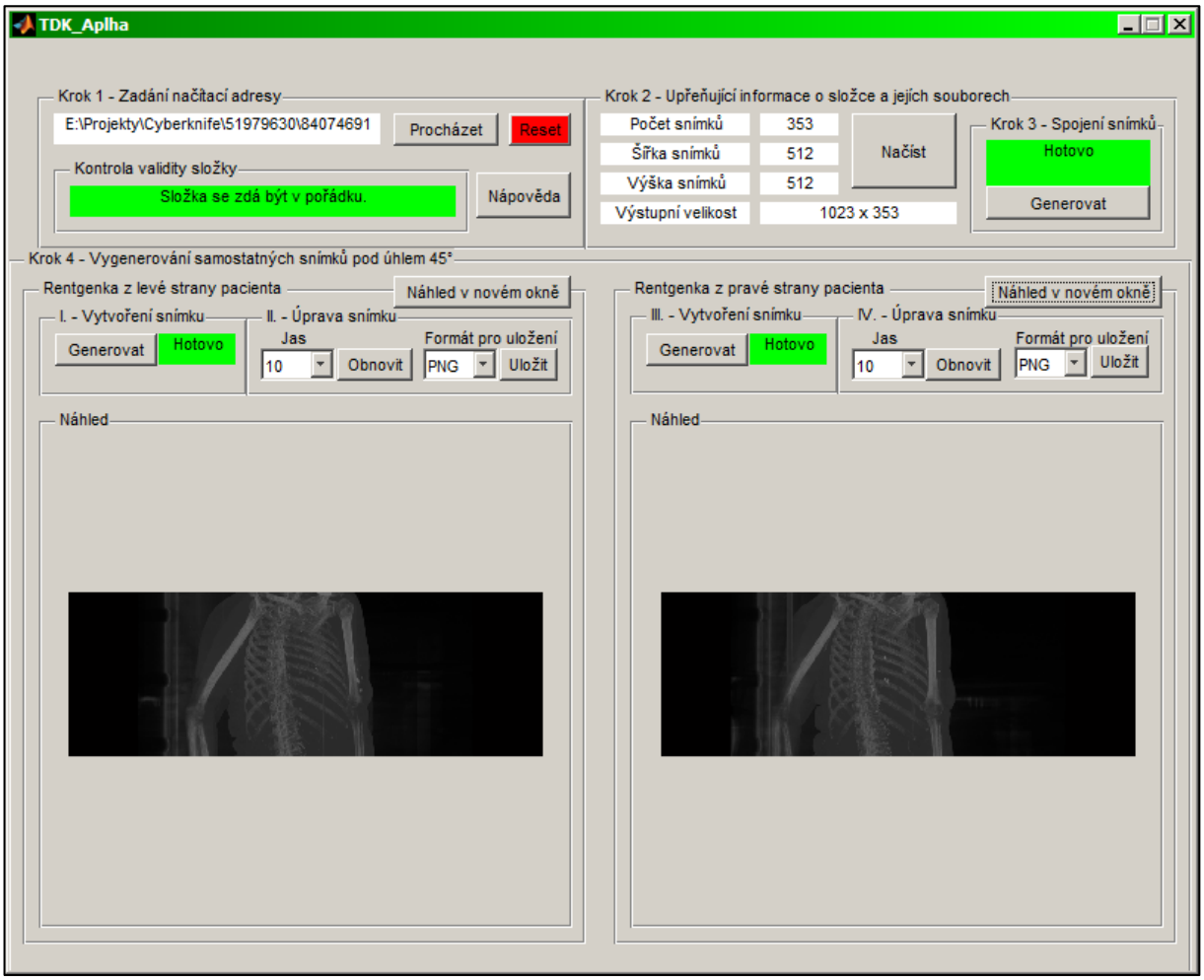
Doba naplnění matic je cca 48 sekund a doba pro vytvoření dvou snímků je cca 157 sekund, celkem **205 sekund**, což jsou přibližně **3.5 minuty**.




Obrázek 8.1 Zjednodušený vývojový diagram algoritmu 3d_diag

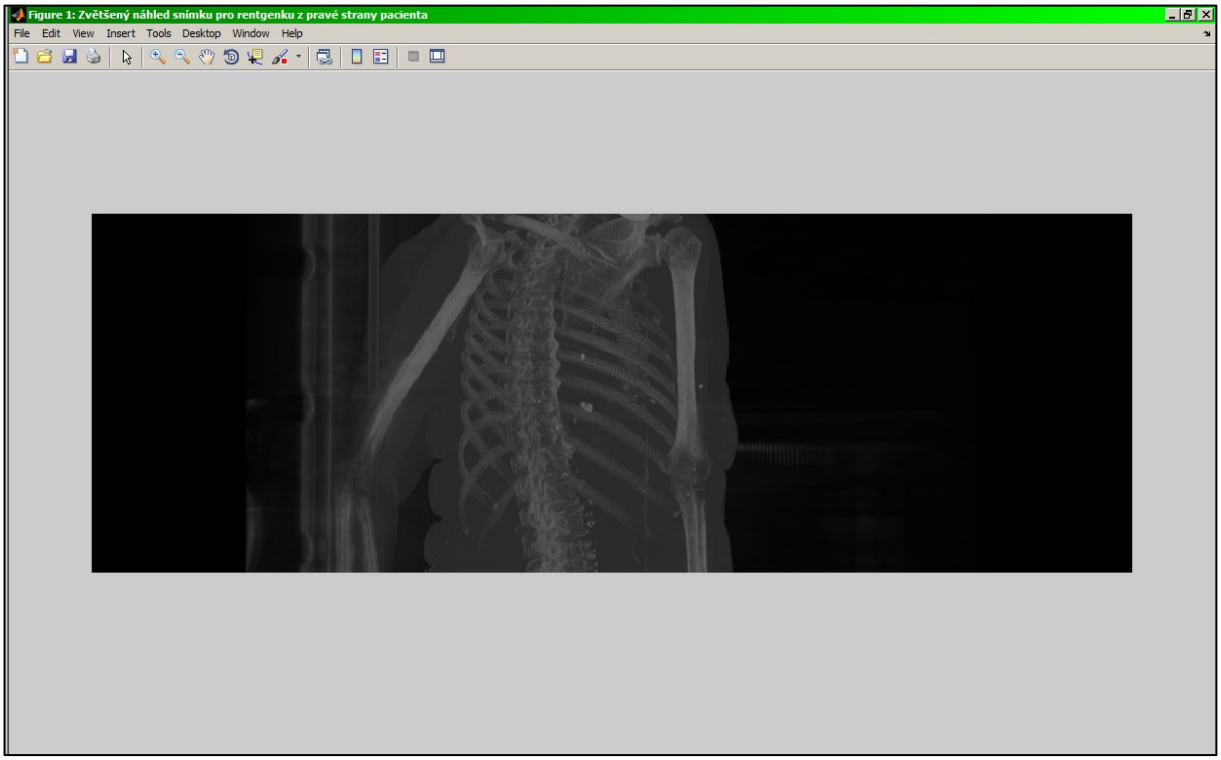
**Grafické rozhraní (MGUI_v1)**

Pro tento algoritmus bylo skrze grafického průvodce (GUIDE) vytvořeno v systému MATLAB grafické rozhraní, které poskytovalo uživateli mnohem příjemnější prostředí a ovládání.




Obrázek 8.2 Grafické prostředí MGUI_v1

Toto grafické rozhraní nabízelo uživateli informace o načtených souborech, jejich validitě a také o výstupních snímcích, které bylo možné zesvětlit, otevřít v novém okně a také uložit v několika formátech.




Obrázek 8.3 Výsledný zesvětlený snímek z grafického rozhraní MGUI_v1otevřený v novém okně

### 8.3.3 Algoritmus pro načítání a kontrolu CT snímků (ct_load)

Během vývoje jsem obdržel nové testovací snímky, které byly již ve formátu, který je momentálně na onkologické klinice běžně využíván. Jedná se o soubory ve formátu DICOM, které ovšem nemají žádnou koncovku. Proto bylo třeba vyvinout nový princip načítání a kontroly snímků.

**Princip**

Vstupním parametrem je cesta ke složce, která by měla obsahovat DICOM soubory a podle ní se vytvoří seznam souborů uvnitř této složky. Tento algoritmus pak využívá funkce *dicomread*, která je uzavřena v příkazu *try-catch*, čili pokud soubor není typu DICOM, funkce *dicomread* vyvolá u tohoto souboru výjimku, která je příkazem *try-catch* odchycena a soubor je ze seznamu vyřazen. V případě, že je soubor úspěšně načten, následuje kontrola, zdali načtená proměnná je prázdná (hodnota je rovna []), protože v tomto případě se jedná o soubor obsahující data kontury. Názvy cest validních snímků a kontur jsou následně uloženy v seznamech, které pokračují k dalšímu zpracování. Z principu algoritmu vyplývá, že je schopen načíst soubory typu DICOM ať už jsou pojmenovány jakkoli a mají jakoukoliv koncovku.

**Rychlost algoritmu**

Doba načtení snímků je cca **5 sekund**.

### 8.3.4 Druhá verze algoritmu pro snímky z úhlu 45 stupňů (file_diag)

Tato verze algoritmu vychází z první verze a cílem bylo zkrátit stávající dlouhou dobu výpočtu. Jelikož první verze načtením a držením v paměti dvou obrovských trojrozměrných matic zabírala spoustu fyzické paměti, bylo zapotřebí nalézt způsob, jak nepoužívat během výpočtu trojrozměrnou matici, tím snížit zátěž fyzické paměti počítače a tím urychlit výpočet.

Během experimentů bylo zjištěno, že data jsou po načtení do systému MATLAB v datovém typu *uint16* a rychlost načtení se zvýší, pokud je předem vytvořena proměnná, tedy matice, do které budou následně načtená data zapsána. Tato matice je vytvářena skrze funkci *zeros*, kde je možné jako vstupní parametr uvést požadovaný datový typ výstupní proměnné. Pro tento postup je ovšem nutné znát předem rozměry snímku, které jsou však v našem případě známy a jsou vždy 512x512 bodů.

**Princip**

Uživatel zadá pomocí grafického průvodce cestu ke složce obsahující DICOM soubory. Výběr složky je zprostředkován skrze funkci *uigetdir*. Algoritmus pro načítání a kontrolu CT snímků (ct_load) pak na základě zadané cesty zkontroluje soubory uvnitř složky a vytvoří seznamy cest k souborům snímků a kontur, které označil jako validní.

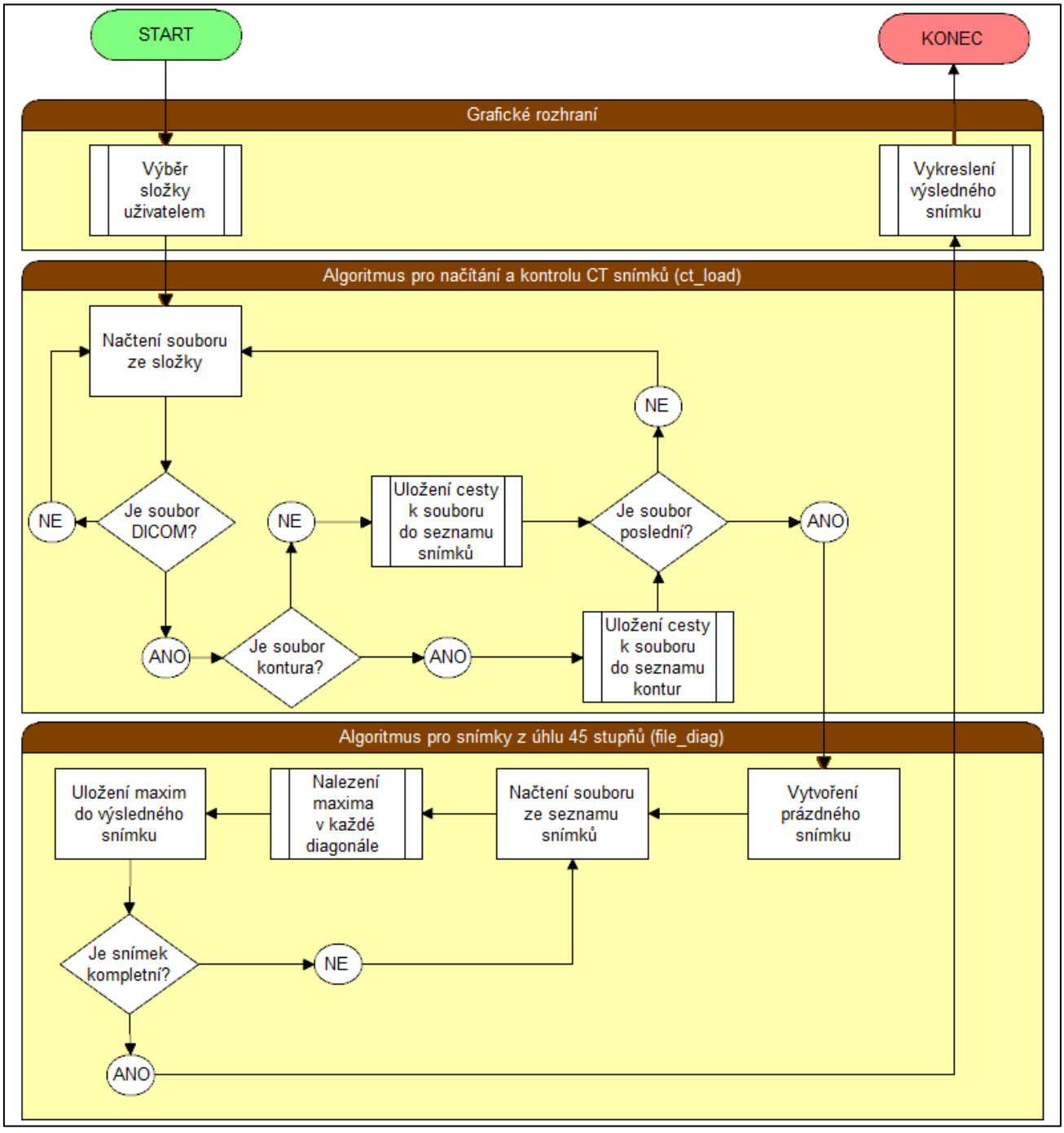
Seznam s cestami (resp. cestou, většinou je připojen pouze jeden soubor s konturami) ke konturám je v tuto chvíli nevyužíván a je proto zahozen.

Seznam souborů snímků vstupuje již do samotného algoritmu pro vytváření snímků z úhlu 45°. Na začátku je vytvořen prázdný snímek, který bude následně naplněn výslednými hodnotami. Skrze cyklus *for* je načten do paměti vždy pouze jeden soubor a z něj je vzato maximum z každé diagonály. Tento postup je proveden totožným způsobem jako v předešlé verzi algoritmu, a to pomocí funkcí *diag* a *max*. Z každého souboru vzniká vektor hodnot, který je uložen do předem připraveného snímku, který byl vytvořen na začátku, tak jak jdou soubory postupně za sebou. Po vykonání tohoto postupu u všech souborů je výsledný snímek naplněn a může být vykreslen.

Pro snímek z druhé rentgenky je proces takřka totožný. Jediným rozdílem je horizontální překlopení každého načteného snímku, které zajišťuje funkce *flip*.

**Rychlost algoritmu**

Doba načtení snímků je cca **5 sekund** (rychlost algoritmu ct_load) a vytvoření dvou snímků trvá cca **8 sekund**.




Obrázek 8.4 Vývojový diagram aplikace pro vytvoření snímků z úhlu 45° složené z algoritmů ct_load, file_diag a grafického prostředí

### 8.3.5 Návrh podpory výpočtu na grafické kartě

Z důvodů poměrně dlouhé doby zpracování byla uvážena možnost výpočtu snímků na grafické kartě. Tato možnost je v systému MATLAB prováděna pomocí funkce *gpuArray*, která převádí proměnné do datového typu stejného názvu (*gpuArray*) a s těmito proměnnými je pak automaticky za běhu programu pracováno na grafické kartě. Převod do původního formátu zajišťuje funkce *gather*.

Bohužel, tento postup stávající algoritmus ještě více zpomalil a proto bylo od podpory výpočtu na grafické kartě upuštěno.

### 8.3.6 Návrh podpory paralelního výpočtu

Jelikož výpočet na grafické kartě algoritmus neurychlil, dalším pokusem, jak algoritmus zrychlit, byla podpora paralelního výpočtu. Paralelní výpočet funguje tak, že rozdělí úkol mezí více procesorů a tím urychluje výpočet. V systému MATLAB je spuštění paralelní podpory zprostředkováváno skrze funkci *parpool* a následný blok kódu k rozdělení a výpočtu se určí pomocí cyklu *parfor*.

**Paralelní podpora pro algoritmus file_diag**

Pro urychlení algoritmu byla přidána podpora paralelního výpočtu, která výpočet urychlila téměř dvojnásobně s každým přidaným jádrem. Výsledné rychlosti pro různé počty jader jsou uvedeny níže.

| Počet jader | Časy pro jednotlivé snímky | |
|---|---|---|
| | Snímek A (bez překlopení) | Snímek B (s překlopením) |
| 1 | 3,770568 s | 3,84427 s |
| 2 | 2,29597 s | 2,04572 s |
| 4 | 1,262843 s | 1,120922 s |

Tabulka 8.1 Výpočetní časy pro čtyřjádrový čip Intel i7-2600

### 8.3.7 Modulární grafické rozhraní v systému MATLAB (MGUI_v2)

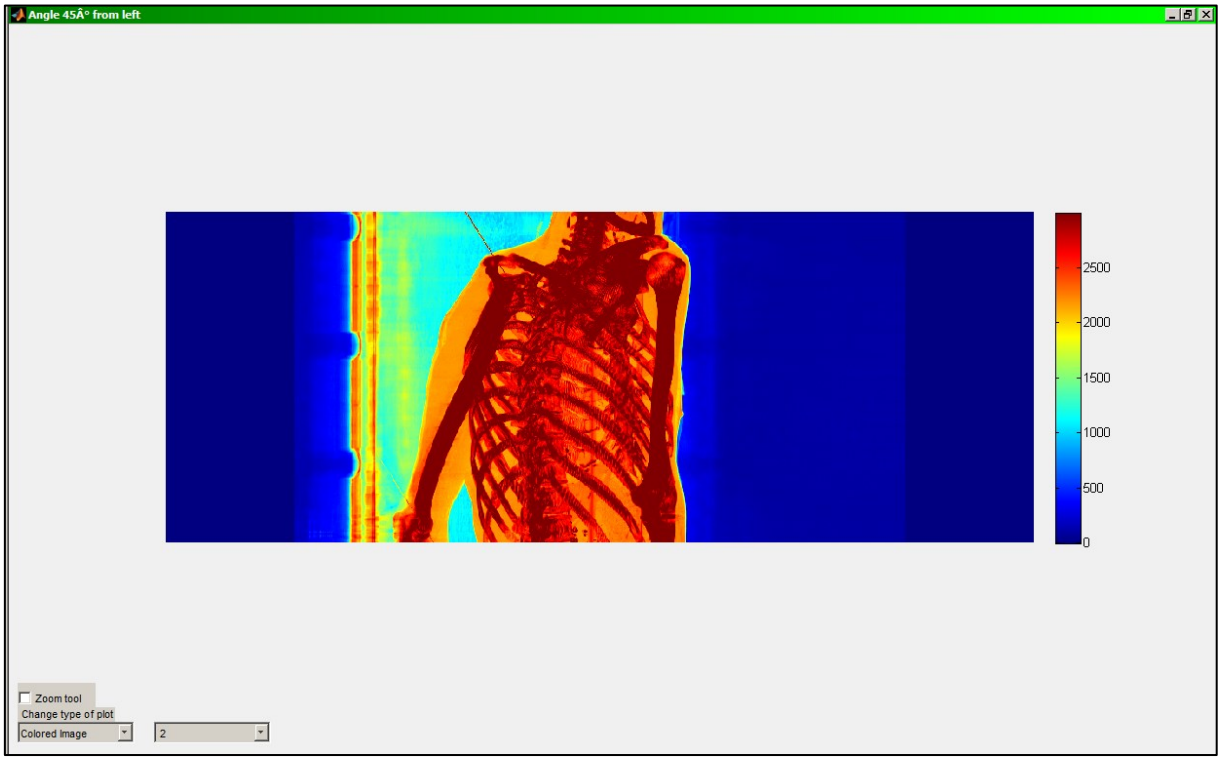
Pro jednoduchost prvního grafického rozhraní MGUI_v1, bylo vytvořeno nové grafické rozhraní, které je více propracované a uživatelsky přívětivé. Toto rozhraní bylo modulárně zpracované, tedy je složeno z jednotlivých modulů. Mimo jiné, má toto rozhraní podporu českého a anglického jazyka.

Moduly jsou napsány v jazyku systému MATLAB a prakticky se jedná o obyčejné funkce, které na základě vstupních dat, vytvoří nové okno a v něm zobrazí výsledek dané operace (např. Prohlížeč snímků, 3D prohlížeč) nebo navrací výsledek z operace, z nového okna, zpět do hlavního okna grafického rozhraní (např. Grafický průvodce výběru složky).

**Vlastní moduly**

- Grafický průvodce výběrem složky
- Prohlížeč snímků s jednoduchým post zpracováním
- Prohlížeč trojrozměrného modelu pacienta
- Logovací panel s podporou exportu do souboru


Obrázek 8.5 Hlavní okno grafického rozhraní MGUI_v2




Obrázek 8.6 Prohlížeč snímků s jednoduchým post zpracováním grafického rozhraní MGUI_v2

### 8.3.8 Algoritmus pro načítání a zpracování kontur

Systém MATLAB v současné době nemá implementovanou žádnou funkci, která by umožňovala načítání a zpracování souborů kontur (RT souborů, formát DICOM). Z tohoto důvodu byla pro načtení a zpracování kontur do systému MATLAB použita funkce, která je volně dostupná na internetu.

**Funkce dicomrt2matlab**

Autory této funkce jsou Jens T. Olesen a Ulrik Landberg Stephansen z Aalborg University, Dánsko. Funkce je dostupná na webových stránkách: https://github.com/ulrikls/dicomrt2matlab.

Vstupem funkce je cesta k souboru kontur a výstupem je struktura obsahující nespočet údajů, kde je pro nás nejdůležitější 3D matice logických hodnot obsahující umístění zakreslených kontur.

**Rychlost algoritmu**

Algoritmu trvá vytvoření struktury přibližně **1 minutu**.

### 8.3.9 Algoritmus pro nalezení přibližného směru pro ozařování (angle_search)

Jako doplňující funkce aplikace byl algoritmus pro nalezení přibližného směru pro ozařování, tím urychlit výpočet algoritmu, který je nyní v systému CyberKnife implementován, a tím zefektivnit terapii.

**Princip**

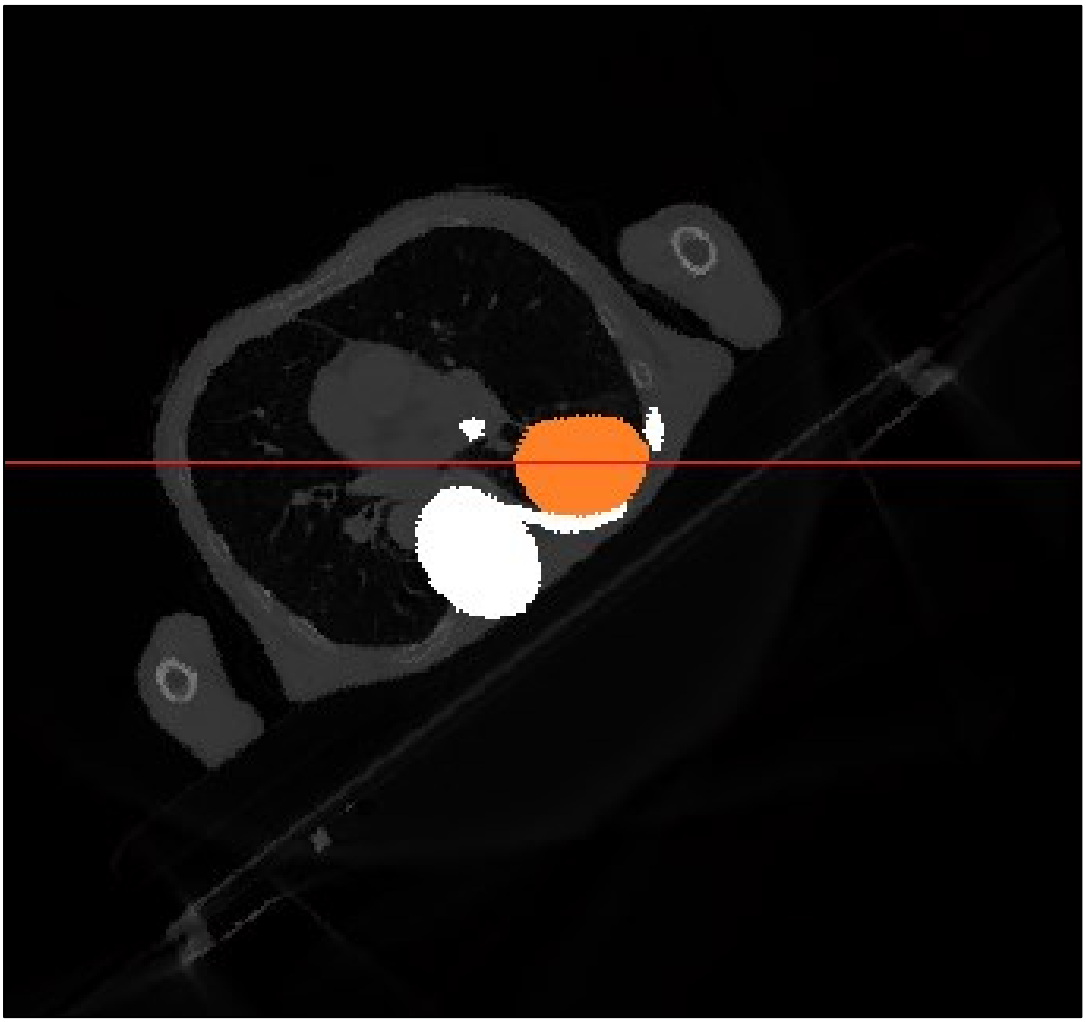
Uživatel zadá pomocí grafického průvodce cestu ke složce obsahující DICOM soubory. Výběr složky je zprostředkován skrze modul *Grafický průvodce výběrem složky* grafického rozhraní MGUI_v2, který je napsán v systému MATLAB pomocí jazyka Java. Rozdílem oproti funkci *uigetdir* je zobrazení souborů uvnitř složky a možnost výběru složky pomocí buď kliknutí přímo na složku, anebo na soubor uvnitř dané složky. Algoritmus pro načítání a kontrolu CT snímků (ct_load) pak na základě zadané cesty zkontroluje soubory uvnitř složky a vytvoří seznamy cest k souborům snímků a kontur, které označil jako validní.

Seznam s cestami (resp. cestou, většinou je připojen pouze jeden soubor s konturami) ke konturám je zadán jako vstup do funkce *dicomrt2matlab* a algoritmus čeká na vytvoření souboru s výsledky, který bude umístěn v aktivní složce algoritmu.

Po vytvoření souboru jsou kontury načteny a roztříděny na konturu ložiska a kontury orgánů. Kontury orgánu jsou spojeny v jednu konturu, aby s nimi bylo možné jednodušeji pracovat. Na základě kontur ložiska je nalezen přibližný střed ložiska ve všech osách a je vypočtena také kompenzace ve všech osách tak, aby ložisko bylo ve všech osách ve středu 3D matice.

Seznam souborů snímků vstupuje již do samotného algoritmu pro nalezení přibližného směru pro ozařování. Je vytvořena prázdná trojrozměrná matice s datovým typem *uint16*. Data jednotlivých snímků jsou načtena, kompenzována tak, aby ložisko bylo ve středu snímku, jsou přidány umělé denzity na základě kontur, které symbolizují kritické orgány, a snímek je přidán do matice. Když je matice naplněna, je ještě kompenzována v počtu snímků, tedy tak, aby střed ložiska byl v ose X (osa páteře pacienta) uprostřed.

Následně vstupuje matice do cyklu, kdy je rotována, pomocí funkce *imrotate*, v daných rozpětích úhlů v osách X a Z. Tato rotace simuluje pohyb ozařovací hlavice kolem pacienta, ovšem v našem případě rotuje pacient a hlavice je statická. V každém kroku cyklu algoritmus prochází řádek, jehož hodnota indexu je zároveň i středem v ose Z. Tímto je simulován průchod paprsku skrze střed ložiska. Ještě než začne algoritmus procházet daný řádek, je kontrolně sečten pomocí funkce *sum*, čímž okamžitě zjistíme, pokud je hodnota součtu vysoká, že v cestě paprsku se nachází nějaká vysokodenzitní struktura (kost) nebo orgán, jehož denzita byla na základě kontury změněna na umělou a extrémně vysokou hodnotu. Je-li hodnota pod určenou prahovou hodnotou součtu, algoritmus prochází jednotlivé hodnoty a vyhodnocuje jejich vhodnost průchodu na základě tabulkových hodnot denzit jednotlivých orgánů, případně kontur. Výsledky vyhodnocení jednotlivých rotací, pak jsou uloženy do struktury obsahující informace o rotaci v ose X, Z a vhodnosti k ozáření určené logickými hodnotami 0 a 1. Pro představu, je zobrazen průchod paprsku na obrázku níže.



Obrázek 8.7 Průchod paprsku (červená) skrze ložisko (oranžová), včetně zobrazení kontury orgánů (bílá)

**Rychlost algoritmu**

Doba výpočtu průchodu paprsku pro jednu danou rotaci je cca **0.8 sekund**.

**Výsledek**

Z důvodu časové tísně, byl vývoj tohoto algoritmu ukončen, a proto nebyl přidán jako doplňující součást aplikace.

### 8.3.10 Třetí verze algoritmu pro snímky z úhlu 45 stupňů (3D_index)

Zkušenosti, získané při vývoji algoritmu angle_search, vedly k novému algoritmu pro rotované snímky z úhlu 45 stupňů. Jehož doba výpočtu v první verzi byla cca **2 sekundy** na snímek, proto byla uvážena podpora paralelního výpočtu.

Během implementace paralelní podpory pomocí cyklu *parfor*, kde je nutné mít kód uvnitř bloku nezávislý na předešlých krocích cyklu, jsem byl nucen použít dynamické (vektorové) indexování, abych tohoto docílil.

Nicméně, paralelní výpočet tento algoritmus jen zpomalil a výpočet jednoho snímku trval cca **10 sekund**, ale když byla paralelní podpora odstraněna a bylo použito pouze dynamické indexování, bylo docíleno rychlosti, kdy doba výpočtu dosáhla cca **0.5 sekundy** na snímek.

**Princip**

Uživatel zadá pomocí grafického průvodce cestu ke složce obsahující DICOM soubory. Výběr složky je zprostředkován skrze modul *Grafický průvodce výběrem složky* grafického rozhraní MGUI_v2, který je napsán v systému MATLAB pomocí jazyka Java. Rozdílem oproti funkci *uigetdir* je zobrazení souborů uvnitř složky a možnost výběru složky pomocí buď kliknutí přímo na složku, anebo na soubor uvnitř dané složky. Algoritmus pro načítání a kontrolu CT snímků (ct_load) pak na základě zadané cesty zkontroluje soubory uvnitř složky a vytvoří seznamy cest k souborům snímků a kontur, které označil jako validní.

Seznam s cestami (resp. cestou, většinou je připojen pouze jeden soubor s konturami) ke konturám je v tuto chvíli nevyužíván a je proto zahozen.

Seznam s cestami k souborům vstupuje do cyklu, kde se postupně načtou všechny soubory do trojrozměrné matice. Tato matice je rotována o požadovaný úhel v ose X pomocí funkce *imrotate*. Pro rentgenku A je matice rotována o úhel 45 stupňů. Pro rentgenku B je matice rotována o úhel -45 stupňů.

Následně matice vstupuje do cyklu, kdy je procházen každý řez a díky dynamickému indexování, za pomoci funkce *max*, je vzato maximum v každém řádku jednotlivého řezu najednou. Z každého řezu vzniká vektor hodnot, který je uložen do předem připraveného snímku, vytvořeného na začátku, tak jak jdou řezy postupně za sebou.

Výsledné snímky jsou nakonec vykresleny v oknech grafického prostředí MGUI_v2.

### 8.3.11 Algoritmus pro zpracování kontur

Jedním z posledních kroků, bylo třeba zobrazit ve vygenerovaných rotovaných snímcích také rotovanou konturu. Proto jsem napsal algoritmus, který ve spojení s funkcí dicomrt2matlab a algoritmem 3D_index načte, zpracuje a vykreslí obrysy rotovaných kontur.

**Princip**

Kontury jsou načteny a zpracovány pomocí funkce dicomrt2matlab a výstupní data z funkce jsou uloženy na pevný disk. Soubor s daty je načten jako trojrozměrná matice stejných rozměrů, jako je trojrozměrná matice CT dat, obsahující logickou jedničku v každém prvku, kde byla zakreslena kontura.

Matice následně vstupuje do algoritmu 3D_index, kde je rotována o potřebný úhel a naprosto stejným způsobem jsou vzata maxima (tedy logické jedničky) v každém řádku. Vznikne nám pohled na rotovanou konturu.

Ve výsledném snímku z algoritmu 3D_index jsou pomocí funkce *edge* nalezeny okraje zakreslené kontury a takto získaná data jsou pak zpracována pomocí funkce *find*, která hledá v matici, podle zadané podmínky, indexy vyhovující zadaným podmínkám.

Výsledkem algoritmu je vektor bodů, které určují indexovanou pozici okraje kontury ve výsledném snímku.

**Rychlost algoritmu**

Doba pro výpočet dvou rotovaných kontur je stejná jako rychlost algoritmu 3D_index, tedy cca **0.5 sekundy**.

### 8.3.12 Grafické rozhraní v jazyce C#

Po dokončení výpočetního algoritmu bylo zapotřebí vykreslovat výsledné snímky v dostatečné velikosti a rozlišení. Pro tento účel bylo potřeba vytvořit více uživatelsky přívětivé grafické rozhraní, čehož v systému MATLAB nebylo možné dosáhnout, jelikož jde o výpočetní systém a v něm vytvořená grafická prostředí se hodí spíše na práci s grafy než zpracování obrazu.

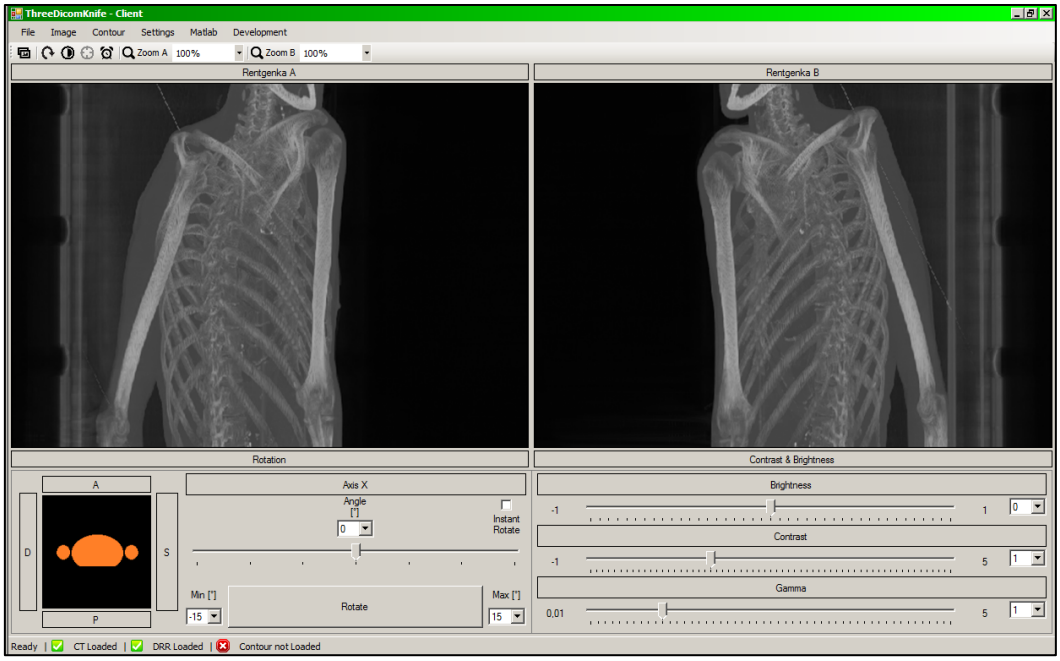
Proto bylo vytvořeno nové grafické rozhraní, implementováno v jazyce C#, které funguje jako klient připojený na COM server systému MATLAB.

**Princip**

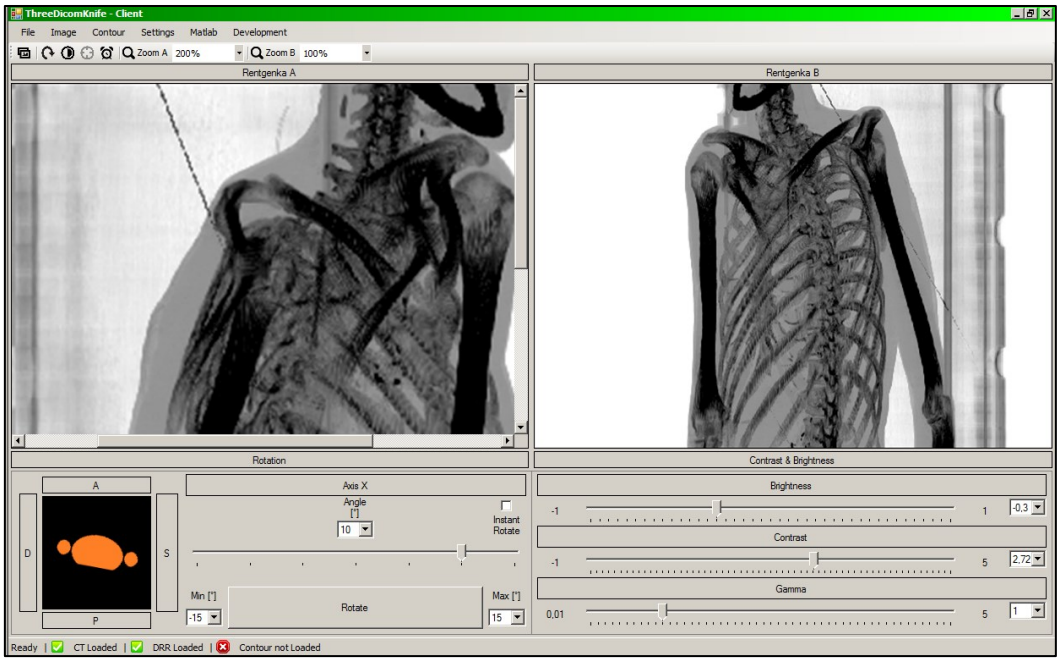
Vytvoření rotovaných snímků a kontur zajišťuje systém MATLAB skrze funkce volané v rámci implementovaného kódu klienta. Tyto snímky jsou načítány do klienta dvěma způsoby.

Prvním způsobem je jednoduché uložení na disk ze systému MATLAB a načtení klientem. Druhý způsob je přímé předání dat ze serveru na klienta, kdy jsou data navrácena jako dvojrozměrná matice s hodnotami pro jednotlivé pixely (voxely). Takto získaná data je nutné převést na proud pixelů a následně poskládat ve snímek. Ve výsledku jsou oba způsoby téměř stejně rychlé a v tuto chvíli je použit způsob načítání skrze přímé předání.

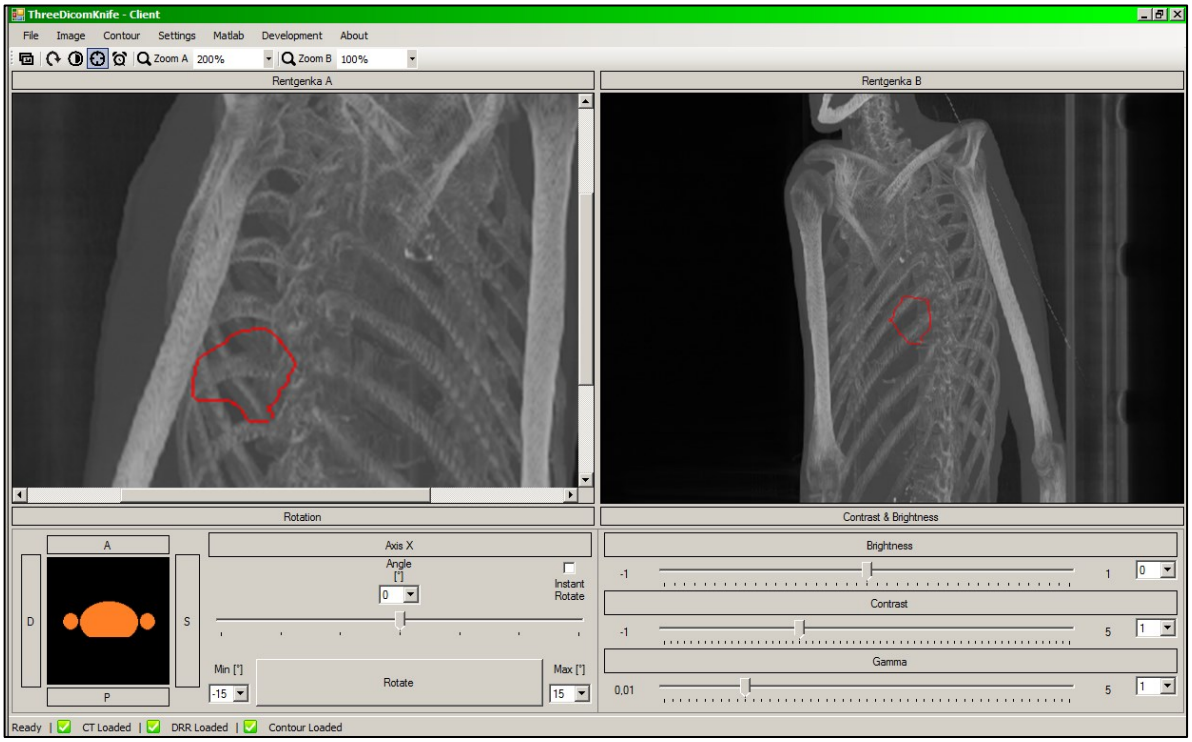
Doplňující zpracování obrazu, jako je změna jasu, kontrastu, gamy či inverze obrazu, je zajištěno na straně klienta.




Obrázek 8.8 Ukázka grafického rozhraní implementovaném v jazyce C#




Obrázek 8.9 Ukázka zpracování obrazu v grafického rozhraní implementovaném v jazyce C#




Obrázek 8.10 Zobrazení kontury v grafického rozhraní implementovaném v jazyce C#

### 8.3.13 Výsledná verze algoritmu pro snímky z úhlu 45 stupňů (3D_index_gpu)

Po dokončení grafického klienta byla doba výpočtu snímků rotovaného pacienta a výpočtu rotované kontury cca **2 sekundy** (1 sekunda na dva snímky a 1 sekunda na kontury). Tato rychlost byla přijatelná, ale stále to bylo z mého pohledu málo.

Při dalších pokusech o zrychlení byla snaha znovu se pokusit o implementaci podpory výpočtu na grafické kartě, ale tentokrát se na grafické kartě vypočítávala pouze určitá část kódu. Tento postup zrychlil celý algoritmus dvojnásobně a výsledná doba výpočtu je v tuto chvíli **0.25 sekundy** na snímek a taktéž pro výpočet jedné rotované kontury.

Výsledná rychlost algoritmu, při výpočtu dvou snímků a dvou kontur, je okolo **1 sekundy**.

# 9 Závěr

Cílem této bakalářské práce bylo navrhnout a zrealizovat algoritmus, který by dokázal, na základě obrazových dat z CT vyšetření, vytvořit snímky pod úhlem 45 stupňů pacienta, který je rotován o určitý úhel. Tento algoritmus by pak v praxi usnadnil řešení problematiky překryvu ozařovaného cíle v plicích tkání s vyšší denzitou a tím zvýšil efektivitu plánovacího procesu.

V rámci teoretické části jsou popsány základy z radiodiagnostiky, radioterapie, dále je uvedeno stručné seznámení se systémem CyberKnife, jeho konstrukcí a technologiemi využívanými při ozařování. Následuje kapitola o systému MATLAB a jeho možnostech, které jsou zde poměrně podrobně popsány. Poslední kapitolou teoretické části je seznámení s problematikou spojení systému MATLAB s projektem v jazyce C#.

Výše zmíněný algoritmus byl navržen a vyvíjen ve výpočetním systému MATLAB. Během vývoje algoritmus prošel několika verzemi, kdy se postupně snižovala doba výpočtu výsledných snímků. Doba výpočtu dvou snímků byla v první verzi algoritmu až 3.5 minuty a ve výsledné verzi pouhé 0.5 sekundy. Tato konečná rychlost pro dva snímky je výsledkem dlouhé cesty vývoje, kdy byl testován nespočet různých přístupů k řešení.

Dalšími důležitými doprovodnými algoritmy je algoritmus pro načítání grafických dat z CT ve formátu DICOM a také algoritmus pro zpracování kontur. Oba tyto algoritmy byly implementovány v jazyce systému MATLAB.

Pro algoritmus bylo také souběžně vyvíjeno grafické uživatelské rozhraní, které prošlo od své první verze také spoustou změn. První verze grafického rozhraní byla vytvořena v systému MATLAB a byla úzce spjata s první verzí algoritmu. Jak algoritmus, tak grafické rozhraní bylo později nahrazeno. V případě grafického rozhraní, byla náhrada v podobě druhé verze, která byla vytvořena modulárně, připravena na to, že se algoritmus, který byl skrze grafické rozhraní prezentován uživateli, může ještě několikrát změnit.

Nakonec se ukázalo grafické rozhraní vytvořené v systému MATLAB jako nevhodné a bylo nahrazeno grafickým rozhraním v jazyce C#. S tímto řešením se však objevil problém s komunikací a předáním dat mezi systémem MATLAB a jazykem C#. Tyto problémy však byly brzy vyřešeny a výsledné grafické rozhraní pracuje jako klient implementovaný v jazyce C# připojen na COM server systému MATLAB. Klient odesílá na server požadavky na výpočet a jsou mu navráceny výsledky. V tomto případě jsou ze serveru získávány snímky pod úhlem 45 stupňů. Doplňující zpracování obrazu je pak zajišťováno na straně klienta.

Výsledkem práce je aplikace vytvořená v jazyce C#, která splňuje všechny zadané cíle. Aplikace vyžaduje ke své funkci na cílovém počítači operační systém Windows a nainstalovaný systém MATLAB.

I přestože, jsou všechny cíle práce splněny, vždy existuje prostor pro možné vylepšení. V tomto případě by se v budoucnu mohlo zvážit další zjednodušení kódu, které by algoritmus pravděpodobně dokázalo ještě urychlit. Dále by pravděpodobně bylo výhodné uvážit použití jiných přístupů k řešení problému nebo vytvoření dalších doplňujících funkcí.

# Literatura


[1] Skiagrafie. WikiSkripta. [Online] [Citace: 1. 5 2015.] http://www.wikiskripta.eu/index.php/Skiagrafie.

[2] Počítačová tomografie a Hounsfieldovy jednotky. WikiSkripta. [Online] [Citace: 30. 4 2015.] http://www.wikiskripta.eu/index.php/Počítačová_tomografie_a_Hounsfieldovy_jednotky.

[3] Angiografie. WikiSkripta. [Online] [Citace: 30. 4 2015.] http://www.wikiskripta.eu/index.php/Angiografie.

[4] Státní úřad pro jadernou bezpečnost. [Online] [Citace: 1. 2 2015.] http://www.sujb.cz/.

[5] ULLMANN, Vojtěch. Astro Nukl Fyzika. [Online] [Citace: 1. 2 2015.] http://astronuklfyzika.cz/.

[6] Dělení buňky a buněčný cyklus. BIOMACH, výpisky z biologie. [Online] [Citace: 30. 4 2015.] http://www.biomach.cz/biologie-bunky/deleni-bunky-a-bunecny-cyklus.

[7] Radioterapie. Onkokurz. [Online] [Citace: 30. 4 2015.] http://www.onkokurz.cz/lekce/28/.

[8] SPURNÝ, Vladimír a Pavel ŠLAMPA. Moderní radioterapeutické metody VI. Díl - Základy radioterapie. Brno : IDVPZ, 1999.

[9] Brachyterapie. WikiSkripta. [Online] [Citace: 30. 4 2015.] http://www.wikiskripta.eu/index.php/Brachyterapie.

[10] CyberKnife - Kybernůž. Fakultní nemocnice Ostrava. [Online] [Citace: 30. 4 2015.] http://cyberknife.fno.cz/cs/clanky/cyberknife-kybernuz.

[11] History of CyberKnife. CyberKnife Centers of the Tampa Bay. [Online] [Citace: 30. 4 2015.] http://www.cyberknifetampabay.org/index.php/about-cyberknife-home/history-of-cyberknife.

[12] Historie. Fakultní nemocnice Ostrava. [Online] [Citace: 30. 4 2015.] http://cyberknife.fno.cz/cs/clanky/historie.

[13] Cyberknife. Wikipedia: the free encyclopedia. [Online] 2001-2015. [Citace: 30. 4 2015.] http://en.wikipedia.org/wiki/Cyberknife.

[14] Matlab - Jazyk pro technické výpočty. Humusoft. [Online] 1991 - 2015. [Citace: 17. 1 2015.] http://www.humusoft.cz/produkty/matlab/matlab/.

[15] MATLAB. Wikipedia: the free encyclopedia. [Online] San Francisco (CA): Wikimedia Foundation, 19. 9 2014. [Citace: 17. 1 2015.] http://cs.wikipedia.org/wiki/MATLAB#Historie.

[16] PENHAKER, Marek a Jan KUBÍČEK. MATLAB a simulace pro biomedicínské obory. Ostrava : VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA OSTRAVA, 2013.