The MATLAB/C program *take* - a program for simulation of X-ray projections from 3D volume data. Demonstration of beam-hardening artefacts in subsequent CT reconstruction.

Olle Seger*, Maria Magnusson Seger**
Computer Engineering (*), Computer Vision Laboratory(**)
Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
Email: olles@isy.liu.se, maria@isy.liu.se

March 15, 2005

LiTH-ISY-R-2682
ISSN 1400-3902

Contents

1 Introduction

The MATLAB/C program *take* version 3.1 is a program for simulation of X-ray projections from 3D volume data. It is based on an older C version by Müller-Merbach [4] as well as an extended C version by Turbell [7]. The program can simulate 2D X-ray projections from 3D objects. These data can then be input to 3D reconstruction algorithms. Here however, we only demonstrate a couple of 2D reconstruction algorithms, written in MATLAB. Simple MATLAB examples show how to generate the *take* projections followed by subsequent reconstruction.

Compared to the old *take* version, the C code have been carefully revised.

A preliminary, rather untested feature of using a polychromatic X-ray source with different energy levels was already included in the old *take* version. The current polychromatic feature X-ray is however carefully tested. For example, it has been compared with the results from the program described in [3]. We also demonstrate experiments with a polychromatic X-ray source and a Plexiglass object giving the beam-hardening artefact.

Detector sensitivity for different energy levels is not included in *take*. However, in section 6.2, we describe a technique to include the detector sensitivity into the energy spectrum.

Finally, an experiment with comparison of real and simulated data were performed. The result wasn't completely successful, but we still demonstrate it.

2 Manual for *take*

2.1 General structure of *take*

The *take* environment is shown in Figure 1. All parameters to the program are given in a number of text-files. By default `take.txt` is used as the initialization file. In this file, all the other files are specified. Also, in some cases, a voxel volume might serve as in-data. The out-data is normally projection data from a mathematical phantom. However, the out-data can also be a voxel volume computed from a mathematical phantom. *Take* can be executed in two ways

- as a stand-alone C program. The out-data is then returned in a binary file.
- as a MATLAB command. The out-data is then returned in a MATLAB variable.

2.2 MATLAB interface

The general format is

```
[y,params] = take(commands);
```

where `params` and `commands` are optional. The simplest version is

```
y = take;
```

which computes a projection (or a voxelized phantom) according to the contents of the parameter files. The character strings `commands` can be used to override the parameters given in the parameter files. A call by name convention is used so no special order or number of commands is required. For instance

```
y = take('initfile=myinit.txt');
```

will use `myinit.txt` instead of `take.txt`. The table below gives a list of available commands:

Command	Comment
`'attenuation=log'`	do take the logarithm of the projection data
`'debug'`	debug printouts during execution
`'detector=file.txt'`	use the detector definition in file.txt
`'energyspectrum=file.txt'`	use the spectrum definition in file.txt
`'initfile=file.txt'`	use `file.txt` instead of `take.txt` as initfile
`'initrot=fi'`	rotates the phantom fi degrees before taking projections
`'mono=energy'`	use monoenergetic spectrum with energy given in keV
`'phantom=file.txt'`	use the phantom definition in file.txt
`'trajectory=file.txt'`	use the trajectory definition in file.txt
`'verbose'`	prints logging information to log.txt in case of projection generation
`'voxelcenter=x y z'`	center of voxel volume given by the doubles x,y,z and measured in meters
`'voxelization'`	computes a voxelized phantom instead of projection data
`'voxelnr=x y z'`	size of voxel volume given by the integers x,y,z and measured in voxels
`'voxelsize=x y z'`	size of voxel volume given by the doubles x,y,z and measured in meters

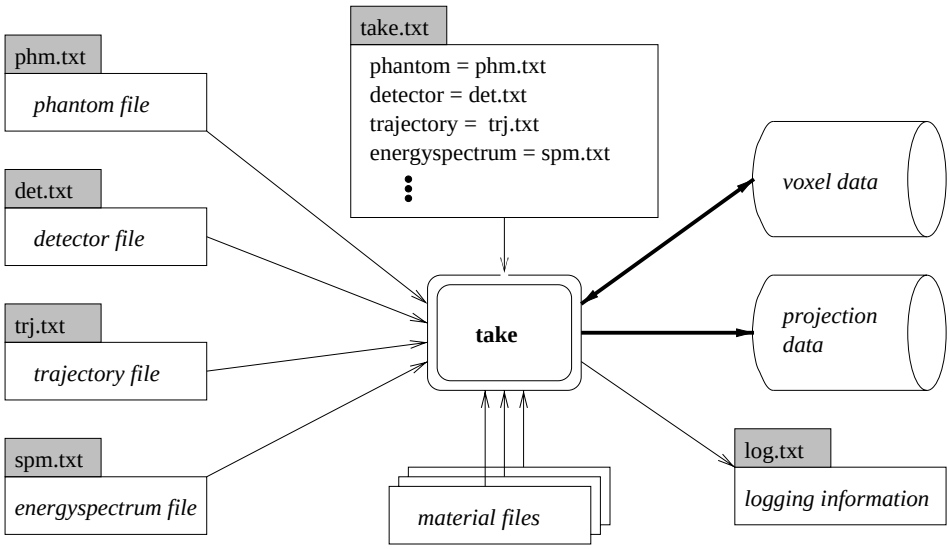

Figure 1: The in-files and the out-data for the program *take*.

Most of these commands can alternatively be given directly in the initialization file `take.txt`:

Command	Comment
`'attenuation=log'`	do take the logarithm of the projection data
`'detector=file.txt'`	use the detector definition in file.txt
`'energyspectrum=file.txt'`	use the spectrum definition in file.txt
`'mono=energy'`	use monoenergetic spectrum with energy given in keV
`'phantom=file.txt'`	use the phantom definition in file.txt
`'trajectory=file.txt'`	use the trajectory definition in file.txt
`'verbose='`	prints logging information to log.txt in case of projection generation
`'voxelcenter=x y z'`	center of voxel volume given by the doubles x,y,z and measured in meters
`'voxelization='`	computes a voxelized phantom instead of projection data
`'voxelnr=x y z'`	size of voxel volume given by the integers x,y,z and measured in voxels
`'voxelsize=x y z'`	size of voxel volume given by the doubles x,y,z and measured in meters

The command

```
[y geo] = take;
```

will return the projection data in `y`. The 1×2-array `geo` returns some geometrical parameters, where geo(1) is the distance in meters between the X-ray source and the rotation center and geo(2) is the angle in radians between two fan rays.

2.3 The stand-alone syntax

The command

```
take
```

will execute the stand-alone C program with `take.txt` as in-data and initialization file, i.e. `take.txt` is the default file. The projection data will by default be returned in the file `proj.dat` or alternatively to a file specified in the initialization file. If another initialization file than `take.txt` is desired, the command

```
take file.txt
```

will execute the stand-alone C program with `file.txt` as in-data and initialization file. Alternatively, the out-data will be voxelized data, if so specified in the initialization file. Additional possible commands to be given directly in the initialization file `take.txt` is then:

Command	Comment
`'projection=proj1.dat'`	saves the projection data in the file `proj1.dat` instead of in `proj.dat`
`'voxelization='`	computes a voxelized phantom instead of projection data and saves it in the file `vox.dat`
`'voxelization=vox1.dat'`	computes a voxelized phantom instead of projection data and saves it in the file `vox1.dat`

The stand-alone syntax cannot read .mat-files. Spectrum files and material files must be given in text-format in .txt-files, see Section 2.8 and 2.9.

The format of the projection and voxelized data is our own so called bvv-format. It is simply three integers followed by the data in floating point. Actually, the projection data is saved in the same order as the c-declaration

```
float[noProj][cSize][aSize]
```

where `noProj` is the number of projections, and `aSize` and `cSize` is the width and height of the detector, respectively. The voxelized data is saved in the same order as the c-declaration

```
float[zSize][ySize][xSize]
```

2.4 The initialization file

A typical initialization file might look like:

```
#
# init file for take
#
#-------------------------
verbose
phantom      = phm.txt
detector     = det.txt
trajectory   = trj.txt
#------------------------
#voxelization = vox.dat
voxelnr      = 128 64 32
voxelsize    = 0.6 0.6 0.6
voxelcenter  = 0.0 0.0 0.0
voxelpoints  = 1
#-------------------------
energyspectrum = spm.txt
```

Lines beginning with a # are treated as comments. If the parameters `phantom`, `detector` and `trajectory` are not given, they are set to the default values `phm.txt`, `det.txt` and `trj.txt`. These files are necessary to enable a meaningful calculation.

The program *take* will voxelize the phantom if the parameter `voxelization` is specified. (Then the # character before `voxelization` in the example file must naturally be removed.) In this case the parameters `voxelnr` and `voxelsize` must be specified. The parameter `voxelcenter` is optional with default value `0.0 0.0 0.0`. The meaning of `voxelnr`, `voxelsize` and `voxelcenter` was given in the tables in section 2.2. The measures are given in relation to the reference coordinate system, see below. If the integer parameter `voxelpoints` $= \alpha > 1$, each voxel will be split into α^3 sub-voxels. The resulting value for a voxel is then the average of the subvoxels, which will reduce the jaggedness close to boundaries.

The `energyspectrum` parameter is given as an ASCII .txt-file or alternatively as a MATLAB .mat-file, see also Section 2.8.

2.5 The phantom file

Below we show an example of a simple phantom file describing a phantom, consisting of a Plexiglass cylinder with three drilled holes. The line continuation symbol "\" used in this example is not supported by *take*.

```
# plexi phantom
#-----------------------------
# cylinder
cylind  a=0.040 b=0.040 c=0.01 x=0.0 y=0.0 z=0.0 \
        theta=0.0 phi=0.0 dens=1.19 mat=0
# hole D = 20mm
cylind  a=0.010 b=0.010 c=0.01 x=-0.02 y=0.0 z=0.0 \
        theta=0.0 phi=0.0 dens=-1.19 mat=0
# hole D = 15mm
cylind  a=0.0075 b=0.0075 c=0.01 x=0.01 y=-0.0173 z=0.0 \
        theta=0.0 phi=0.0 dens=-1.19 mat=0
# hole D = 10mm
cylind  a=0.005 b=0.005 c=0.01 x=0.01 y=0.0173 z=0.0 \
        theta=0.0 phi=0.0 dens=-1.19 mat=0
#
# plexiglass
material = 0 ../materials/plexi.txt
```

The program *take* allows three kinds of geometrical objects, namely `ellipsoids`, `cylinders` and `boxes`. Moreover, *take* allows voxel volumes, see the end of this section. To specify them only the first letter is significant, that is `e`, `c`, `b` or `v`. These objects may be thought of as unit objects at the origin of a reference coordinate system. For instance the unit ellipsoid is the unit sphere. The rest of the line in the phantom file will then scale, rotate and translate the object. To describe these operations we use 4-dimensional homogeneous transformation matrices. These matrices will all have the same fourth row, which is not stored in *take*. The scaling matrix is given by

$$S(s_a, s_b, s_c) = \begin{pmatrix} s_a & 0 & 0 & 0 \\ 0 & s_b & 0 & 0 \\ 0 & 0 & s_c & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

and the translation matrix is given by

$$T(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2}$$

An arbitrary coordinate $(x_0, y_0, z_0, 1)'$ on the unit sphere is transformed to $(s_a x_0, s_b y_0, s_c z_0, 1)'$ by the S-matrix, i.e.

$$\begin{pmatrix} s_a x_0 \\ s_b y_0 \\ s_c z_0 \\ 1 \end{pmatrix} = \begin{pmatrix} s_a & 0 & 0 & 0 \\ 0 & s_b & 0 & 0 \\ 0 & 0 & s_c & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}. \tag{3}$$

In a similar way an arbitrary point $(x_0, y_0, z_0, 1)'$ in the 3D world is translated to $(x_0 + t_x, y_0 + t_y, z_0 + t_z, 1)'$ by the T-matrix, i.e.

$$\begin{pmatrix} x_0 + t_x \\ y_0 + t_y \\ z_0 + t_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}. \tag{4}$$

See also Figure 2 showing the result of transforming the unit sphere by the S- and T-matrices.

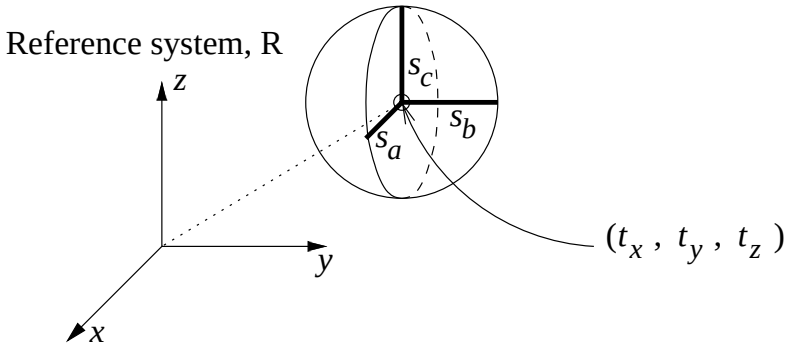

Figure 2: The unit sphere transformed by the S- and T-matrices.

A rotation in the xy-plane around the z-axis is given by

$$R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5}$$

The rotation matrices R_x and R_y are defined in a similar manner. The program *take* uses three different ways of specifying the rotation of an object, namely by specifying

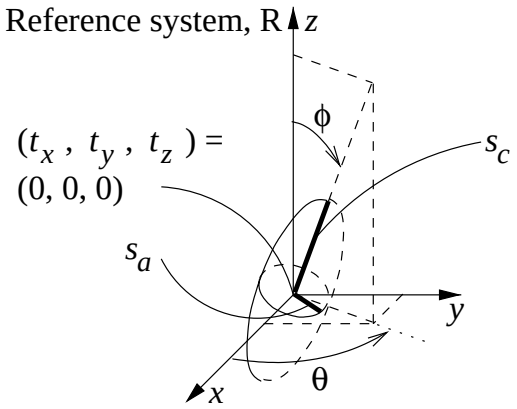

Figure 3: The unit sphere transformed by the $R_z(\theta) \cdot R_y(\phi) \cdot S(s_a, s_b, s_c)$. Here $(s_a, s_b, s_c) = (0.1, 0.1, 0.3)$, $\theta = 60^\circ$ and $\phi = 30^\circ$.

- 2 angles, θ (`theta`) and ϕ (`phi`). The two angles specifies a point on the unit sphere, θ is the longitude and ϕ is the colatitude. For instance

```
a=0.1 b=0.1 c=0.3 x=0.0 y=0.0 z=0.0 \
theta=60 phi=30
```

  corresponds to the transformation matrix

$$\begin{aligned}\Omega &= T(t_x, t_y, t_z) \cdot R_z(\theta) \cdot R_y(\phi) \cdot S(s_a, s_b, s_c) \\ &= T(0, 0, 0) \cdot R_z(60) \cdot R_y(30) \cdot S(0.1, 0.1, 0.3)\end{aligned}$$

  where the matrices are applied from right to left. See also Figure 3 showing the result of transforming the unit sphere this way.

- 3 rotations, r_x, r_y, r_z (`rotx, roty, rotz`) around the axes of the reference system. For instance

```
a=0.1 b=0.1 c=0.1 x=-0.2 y=0.0 z=0.0 \
rotx=30 roty=45 rotz=90
```

  which should be interpreted: rotate first 30 degrees around the x-axis, then 45 degrees around the y-axis and finally 90 degrees around the z-axis. The complete transformation matrix is then given by

$$\begin{aligned}\Omega &= R_z(r_z) \cdot R_y(r_y) \cdot R_x(r_x) \cdot T(t_x, t_y, t_z) \cdot S(s_a, s_b, s_c) \\ &= R_z(90) \cdot R_y(45) \cdot R_x(30) \cdot T(-0.2, 0, 0) \cdot S(0.1, 0.1, 0.1).\end{aligned}$$

  Translation is performed before rotating the object.

- 3 rotations, x_r, y_r, z_r (`xrot, yrot, zrot`) around the object axes. The following examples uses the Euler angles

```
a=0.1 b=0.1 c=0.1 x=-0.2 y=0.0 z=0.0 \
zrot=30 yrot=45 zrot=90
```

  which should be interpreted as: begin by rotating 30 degrees around the z-axis, then rotate 45 degrees around the *new* y-axis and finally 90 degrees around the *new* z-axis. The same 3D-rotation can be performed by rotating relative to the fixed xyz-coordinate system just by reversing the order of the rotations, that is

$$\begin{aligned}\Omega &= T(t_x, t_y, t_z) \cdot R_z(z_r) \cdot R_y(y_r) \cdot R_z(z_r) \cdot S(s_a, s_b, s_c). \\ &= T(-0.2, 0, 0) \cdot R_z(30) \cdot R_y(45) \cdot R_z(90) \cdot S(0.1, 0.1, 0.1).\end{aligned}$$

The density of the object is given by, for instance, `dens=1.19`. To handle polychromatic spectrums, a material number `mat=0` and a corresponding material file

```
material = 0 ../materials/plexi.txt
```

should be specified in the phantom file, see the example in the beginning of this section.

The voxel volume object can be seen as a box, the only difference being that the density variation is described by a voxel file instead of being homogeneous. The voxel data may be a reconstructed CT-volume or a voxelization of a phantom not possible to construct from the existing primitives. The declaration is on the form

```
# voxel volume object
#---------------------
voxel a=0.5 b=0.5 c=2 x=0 y=0 z=2 theta=0 phi=0 \
file=head.bvv
```

where the last argument specifies the name of the file containing the voxel density data. The file should be of bvv-type. The data is stretched to fully fill the box as shown in Figure 4. The projections are generated by interpolating along the rays using the technique of Joseph [1]. The projection generating procedure through voxel volumes is currently not implemented for different energy levels.

2.6 The detector file

A typical detector file might look like:

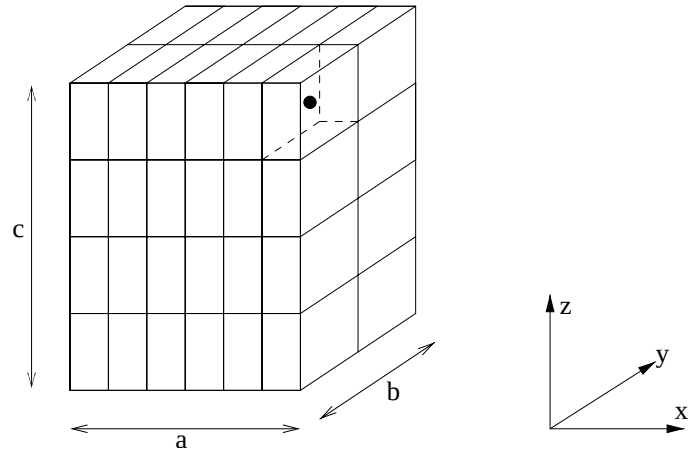

Figure 4: A small voxel volume with $6\times2\times4$ voxel values. Note that a, b, and c define the outer dimensions of the voxel volume and that the voxels are non-cubic in order to fill the box.

```
# detector file
#--------------------
xlen=0.092816
ylen=0.092816
xpix=256
ypix=256
xypoints=4
```

The size of the detector is given by `xlen` and `ylen`, see Figure 7. The parameters `xpix` and `ypix` specifies the number of detector elements in the x- and y-direction, respectively. The parameter `xypoints` gives sub-sampling of the detector elements, in this case `4x4=16` subsamples are specified.

The default detector shape is a flat rectangular detector. Cylindrical detectors of two kinds are also allowed: cylindrical around the X-ray source or cylindrical around the rotation axis. The first one is illustrated in Figure 5. The second one is similar, the only difference being that the detector is more curved since it is cylindrical around the rotation axis. In these cases a number of extra parameters are needed.

- `shape`. Specify `cylindricalAroundSource` or `cylindricalAroundRotationAxis`.
- `fanangle`. Specifies the angle, measured in radians, between the detector center and the outer border of the border detector cell.
- `height`. Denotes the height of the detector projected onto the rotation axis.

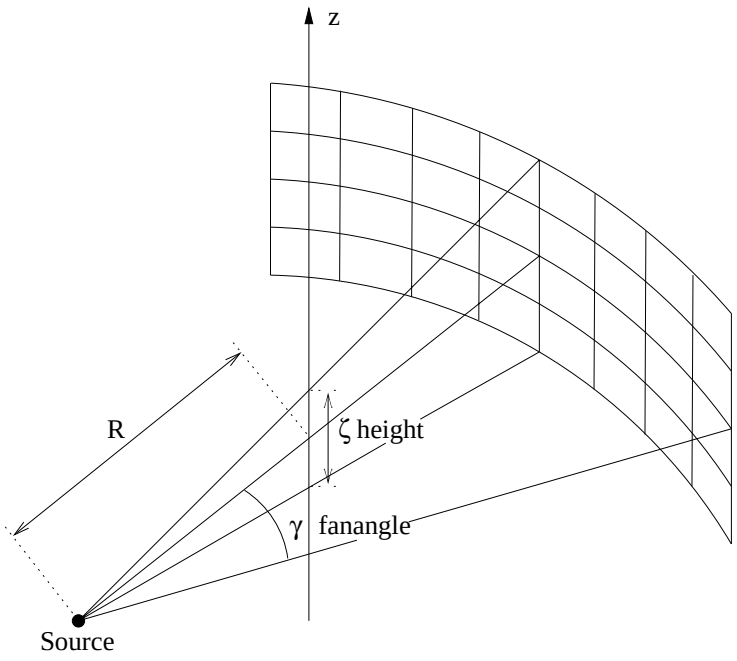

Figure 5: The parameters of a cylindrical detector centered around the source. The detector consists of 4 rows of 8 channels, each.

- `channels`. Denotes the number of rays in a fan, i.e. the number of detector elements in a row.
- `rows`. Denotes the number of fans in a cone, i.e. the number of detector rows.
- `skew`. This parameter is only valid for the `cylindricalAroundRotationAxis` case. It is used when slanted rows are desired. It denotes the ζ-displacement per fan-angle for each row, i.e. $\Delta\zeta/\Delta\gamma$. It can be used if a detector perfectly fitted to the helical scan path is desired.

All types of detectors can be translated horizontally. In the case of cylindrical detectors, this translation is a rotation.

- `channel_offset`. Translates or rotates the detector in the positive γ-direction the given number of detector channels. If this parameter is set to 0.25, quarter offset is achieved.

The default case in *take* is to use a point-shaped source. More realistically, the source can be modeled as a small rectangular plane parallel to the detector plane. This feature only makes a difference if several rays are measured per detector cell,

i.e. when `xypoints > 1`. See Figure 6. The rays then start from a grid on the source rectangle. This grid is defined in an identical way to the grid on each detector cell. It would be too expensive to cast a ray from each source grid point to each detector cell grid point, so a simple scheme is used. From each source grid point two rays are casted to each detector cell, one to the corresponding detector grid point and another to the mirrored detector grid point. For example, from the upper left corner of the source rays are cast both to the upper left corner and to the lower right corner of each detector cell. The number of grid points on the source and detector are therefore the same, namely $\texttt{xypoints}^2$. This results in that $2 \cdot \texttt{xypoints}^2$ rays are casted per detector cell. The following parameters defining the source should be placed in the detector file:

- `sourceWidth`. Specifies the width of the source rectangle.
- `sourceHeight`. Specifies the height of the source rectangle.

Note that when `xypoints` is odd, the central ray is measured twice. This is however not the case if `xypoints=1` or if
`sourceWidth = sourceHeight = 0`.

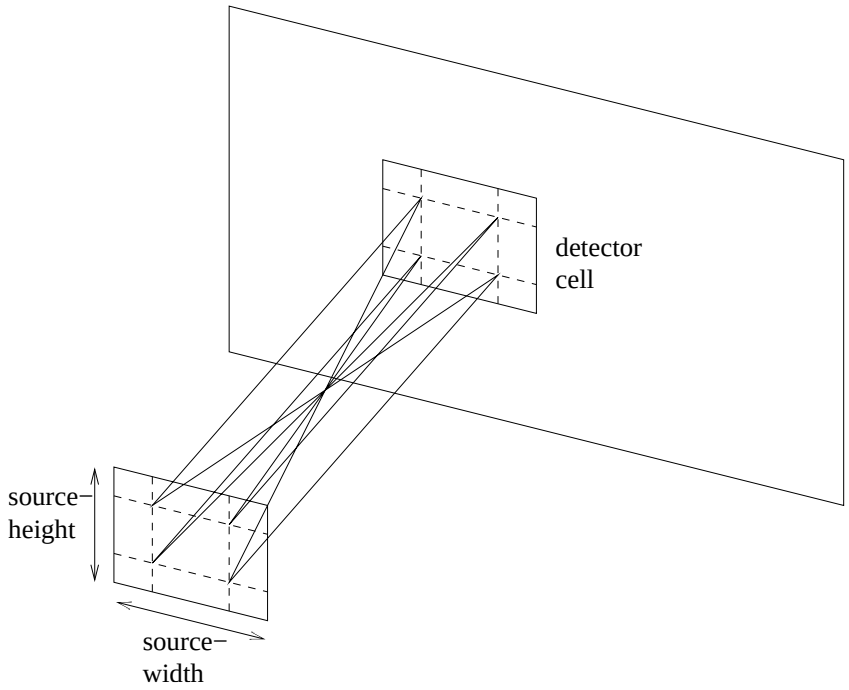

Figure 6: Here, both source and detector are sub-sampled. The parameter `xypoints = 2`.

2.7 The trajectory file

The trajectory file describes the trajectories of the X-ray source and detector. It must begin with the word `projections` and the number of projections. The trajectories can then be specified in two ways:

- by specifying in order, a vector for the first source position, a matrix for the first detector position and a matrix for the transformation from one position to the next.
- by specifying the word `explicit` immediately after the projections line. Then comes in order source position 0, detector position 0 (matrix), source position 1, ..., source position $N-1$ and detector position $N-1$.

We explain the first method with an example:

```
# trajectory file
# ----------------
projections = 254
#
# source position s0
  0.725712    0.00000000  0.00000000
#
# detector position matrix D0
  0.00000000 -1.00000000  0.00000000 -0.72571200
  1.00000000  0.00000000  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
#
# transformation matrix T (pos i => pos i+1)
  0.99991660 -0.01291508  0.00000000  0.00000000
  0.01291508  0.99991660  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
```

See Figure 7. The first source position is given by $s_0 = (0.725712, 0, 0)$, that is on the positive part of the x-axis in the reference coordinate system R. The detector coordinate system D is defined as follows. Originally D coincide with R, so that the basis vectors $\mathbf{a} = \mathbf{x}$, $\mathbf{b} = \mathbf{y}$, $\mathbf{c} = \mathbf{z}$ and the two origins coincide. In homogeneous matrix notation we have for the original detector position

$$R = I_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6}$$

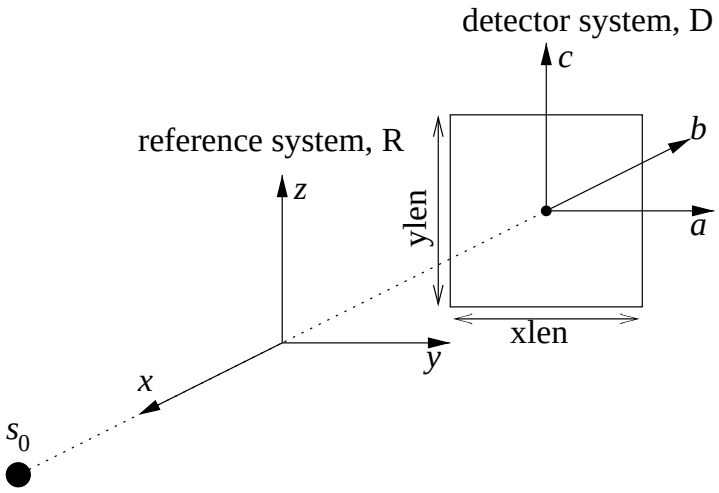

Figure 7: The source position s_0 and the detector coordinate system D are specified in relation to the reference coordinate system R.

The first detector position is then given by

$$D_0 \cdot I_4 = D_0 = \begin{pmatrix} 0 & -1 & 0 & -0.725712 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} & t \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix}. \qquad (7)$$

which is a translation by -0.725712 in the x-direction and a $90°$ rotation around the z-axis. Equivalently, the new basis vectors in the detector system are $\mathbf{a} = \mathbf{y}$, $\mathbf{b} = -\mathbf{x}$, $\mathbf{c} = \mathbf{z}$. Finally the i-th position of the source and detector is given by

$$\begin{aligned} s_i &= T^i \cdot s_0 \\ D_i &= T^i \cdot D_0, \end{aligned} \qquad (8)$$

where T is the transformation matrix

$$T = \begin{pmatrix} 0.99991660 & -0.01291508 & 0 & 0 \\ 0.01291508 & 0.99991660 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad (9)$$

i.e. this matrix rotates the source and the detector $0.74°$ around the z-axis of the reference system.

2.8 The energy-spectrum files

A spectrum file for an X-ray source, specifying the number of photons as a function of energy $N = N(E)$, is shown below. It consists of a table where the first column gives the energy E [keV] and the second column the number of photons N. The second column must begin and end with a zero.

```
# Name: 45.0 kV
# Continuous spectrum
# Dimension: 42
# E [keV]       N
5.0000000e+00   0.0000000e+00
6.0000000e+00   0.0000000e+00
7.0000000e+00   0.0000000e+00
8.0000000e+00   0.0000000e+00
9.0000000e+00   3.1155915e-09
1.0000000e+01   4.0548318e-05
1.1000000e+01   6.8623098e-04
1.2000000e+01   1.8461402e-02
...
4.6000000e+01   0.0000000e+00
```

The energy-spectrum file can also be given in .mat-format containing two columns and a number of rows. Note, however, that the .mat-format does not work for a stand-alone program. The whole energy-spectrum file is given in section 8.5.1.

2.9 The material files

A material file for Plexiglass is shown below. It consists of a table where the first column gives photon energy [MeV]. The other columns list so called cross sections, for coherent and incoherent scattering and photo-electric effect, respectively. The cross-sections combined with the density gives the X-ray attenuation coefficient. The density should be given in the phantom file.

```
# Material: Plexiglass
# Density: 1.19 g/cm^3
# Dimension: 201
# Energy      Co_CS     In_CS     Ph_CS
# [MeV]       [cm^2/g]  [cm^2/g]  [cm^2/g]
 1.00000e-03  1.15e+00  1.42e-02  2.73e+03
 1.50000e-03  1.03e+00  2.83e-02  8.91e+02
 2.00000e-03  9.13e-01  4.36e-02  3.92e+02
 3.00000e-03  6.92e-01  7.21e-02  1.19e+02
 4.00000e-03  5.27e-01  9.46e-02  5.03e+01
 5.00000e-03  4.13e-01  1.11e-01  2.55e+01
```

```
6.00000e-03  3.34e-01  1.23e-01  1.45e+01
...
2.00000e-01  1.06e-03  1.31e-01  1.61e-04
```

We make the assumption that a photon that has interacted by any of these three effects does not reach the detector. The linear attenuation coefficient μ can thus be calculated by the formula

$$\mu(E) = 100 \cdot \rho \cdot (\sigma_{Co}(E) + \sigma_{In}(E) + \sigma_{Ph}(E)) \qquad [\mathrm{m}^{-1}], \tag{10}$$

where ρ is the density of the material measured in [g/cm^3], σ_{Co} is the cross-section for coherent scattering, σ_{In} is the cross-section for incoherent scattering, and σ_{Ph} is the cross-section for the photo-electric effect. The cross-sections are all measured in [cm^2/g]. The material file can also be given in .mat-format containing four columns and a number of rows. Note, however, that the .mat-format does not work for a stand-alone program. The whole material file for Plexiglass is given in section 8.5.2.

2.10 Computation of a detector value given a polychromatic X-ray source

Detector values are estimated by computing line integrals through the phantom. The intensity that reaches the detector is given by

$$S_L = \int_0^{E_{max}} E\ N(E) \cdot \exp\left(- \int_L \mu(l, E)\ dl \right)\ dE, \tag{11}$$

where $EN(E)$ is the intensity at the energy level E, the linear attenuation coefficient μ is given by (10) and L is a straight line through the phantom. For an unattenuated ray, that does not intersect the phantom, we get

$$S_0 = \int_0^{E_{max}} E\ N(E)\ dE. \tag{12}$$

Projection data, that is used as input to a reconstruction algorithm, is given by

$$p_L = \ln\left(\frac{S_0}{S_L}\right), \tag{13}$$

where S_0 and S_L are defined by (11) and (12), respectively. For a phantom consisting of K materials (11) is modified to

$$S_L = \int_0^{E_{max}} E\ N(E) \cdot \exp\left(- \sum_{k=0}^{K-1} \mu_k(E) l_k \right)\ dE, \tag{14}$$

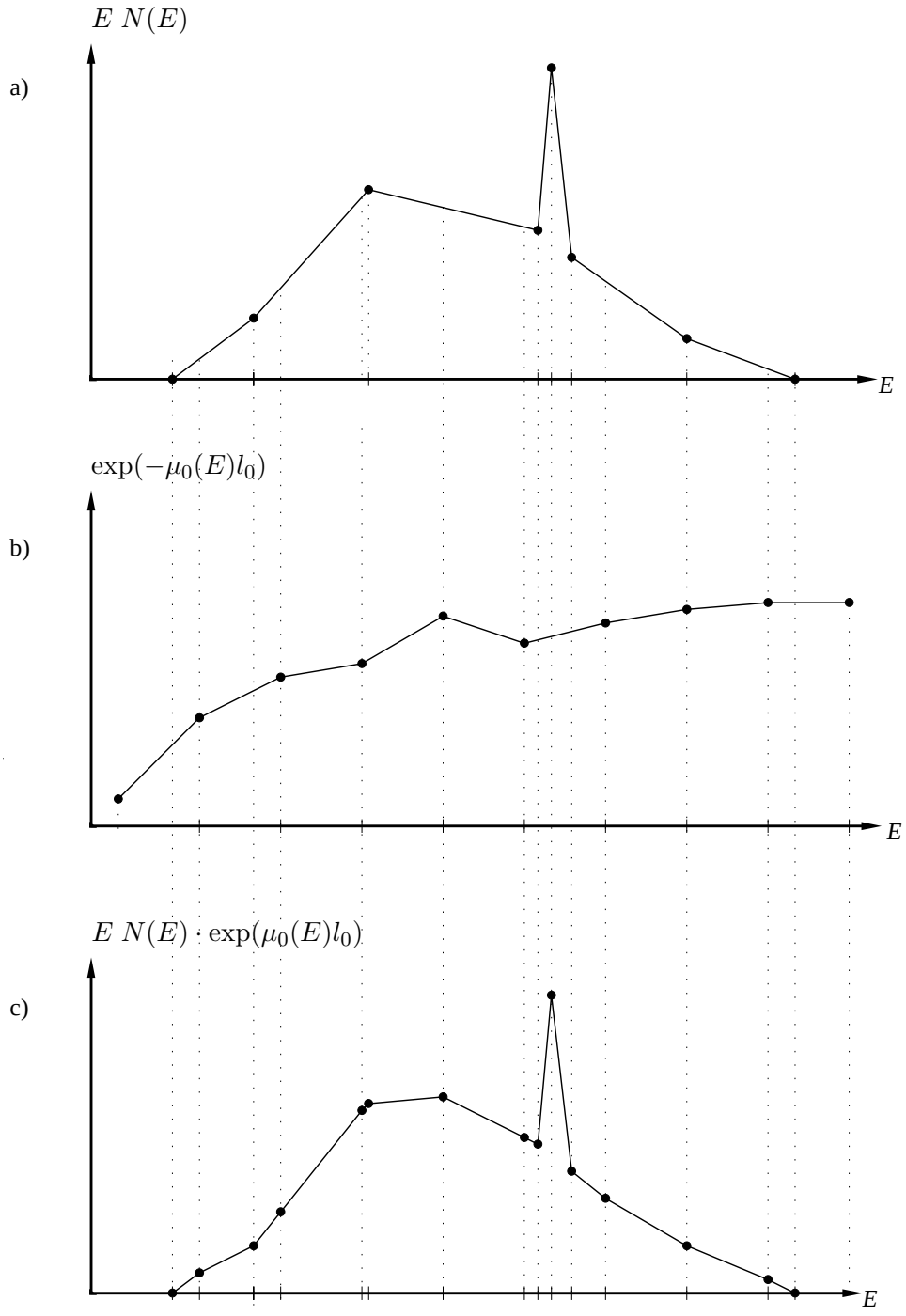

Figure 8: a) Intensity spectrum at the source, $E \cdot N(E)$. b) Attenuation as a function of energy $\exp(-\mu_0(E)l_0)$. c) Intensity spectrum at the detector, $E \cdot N(E) \cdot \exp(-\mu_0(E)l_0)$.

where l_k is the length of the intersection with material k. If the phantom consists of only one material (14) simplifies to

$$S_L = \int_0^{E_{max}} E\ N(E) \cdot e^{-\mu_0(E) l_0}\ dE. \tag{15}$$

In Figure 8 we show how the integral in (15) is computed. The intensity spectrum at the source $E\ N(E)$, Figure 8 a), and the attenuation as a function of energy $\exp(-\mu_0(E) l_0)$, Figure 8 b), are typically sampled on different grids. The intensity spectrum and attenuation function are re-sampled to a common grid and multiplied as shown in Figure 8 c). We designate these sample points $\{E_k\}_{k=0\ldots K}$. Now (15) can be computed using Simpson's formula

$$S_L \approx \frac{1}{2} \sum_{k=1}^{K-1} E_k N(E_k) \cdot e^{-\mu_0(E_k) l_0} \cdot (E_{k+1} - E_{k-1}). \tag{16}$$

Finally we can compute a monochromatic version of (11),

$$S_L = E_0\ N(E_0) \cdot \exp\left(- \int_L \mu(l, E_0)\ dl \right). \tag{17}$$

Note that the detector sensitivity is not included in this computation of the detector value. The program *take* does not support detector sensitivity either. However, the experiment in section 6.2 shows a technique to include the detector sensitivity into the energy spectrum.

3 Reconstruction with Computed Tomography (CT)

When a set of projections have been generated by a CT-scanner or computed by a simulation program such as *take*, the behind-lying object function can be reconstructed from its projections provided that

- The projections span the whole 180° angular interval around the object.
- The projections cover the whole object, i.e. the projections are not truncated.

The above mentioned is valid for the 2D case, a 2D slice through the object and a set of 1D parallel projections.

A modern CT tomograph does not create parallel rays, but a set of rays diverging from one point, the X-ray source. Such a set of rays gives a 2D function, a cone-beam projection. If the rays are collimated so that they all travel in the same 2D object slice, a 1D fan-beam projection is produced instead, see Figure 10.

The simulation program *take* is able to produce 2D cone-beam projections and of course also 1D fan-beam projections. From 2D cone-beam projections there is, under certain circumstances, possible to reconstruct a 3D object. We will, however, not treat 3D reconstruction in this report.

The most commonly used reconstruction algorithm is the filtered back-projection method. There are three variants of this method depending on the ray geometry, parallel rays, fan-beam rays with curved detector and fan-beam rays with flat detector. In the following we will describe the first two ones, which both have been implemented in the *take* environment. There also exists another possibility, namely to *rebin* the fan-beam rays into parallel rays before applying the reconstruction. Rebinning from fan-beam rays with flat detector to parallel rays have been implemented in the *take* environment and is described below. A more thorough treatise of all variants of the filtered back-projection method can be found in for example [2] or [5].

3.1 The parallel filtered back-projection method

The whole process of parallel computed tomography (CT), is illustrated in Figure 9 and contains the following steps, *parallel projection generation*, *ramp-filtering* and *back-projection*. The two last steps constitutes the parallel filtered back-projection method. Note that the object is denoted $f(x, y)$ and the reconstructed object $\hat{f}(x, y)$.

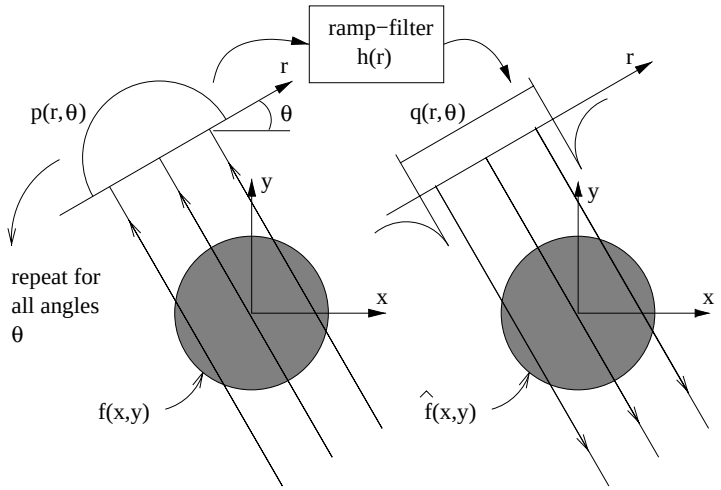

Figure 9: Parallel computed tomography illustrated.

- *Projection* generation is described by

$$p(r, \theta) = \int_{-\infty}^{\infty} f(x, y)\, ds, \tag{18}$$

where

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} r \\ s \end{pmatrix}. \tag{19}$$

- At first, the *ramp-filter* is applied according to

$$q(r,\theta) = \mathcal{F}_r^{-1}\Big[\mathcal{F}_r[p(r,\theta)] \cdot \mathcal{F}_r[h(r)]\Big], \tag{20}$$

where $\mathcal{F}_r$ denotes the Fourier transform in the r-direction, $\mathcal{F}_r^{-1}$ denotes the inverse Fourier transform, and

$$\mathcal{F}[h(r)] = H(\rho) = \begin{cases} |\rho|, & \text{if } |\rho| \leq \rho_{max}, \\ 0, & \text{elsewhere.} \end{cases} \tag{21}$$

- Then *back-projection* is applied, which means smearing of filtered projection data over the image plane according to

$$\hat{f}(x,y) = \int_0^{\pi} q(x\cos\theta + y\sin\theta, \theta)\, d\theta. \tag{22}$$

3.2 The fan-beam filtered back-projection method with curved detector

The whole process of fan-beam computed tomography with curved detector, is illustrated in Figure 10 and contains the following steps, *fan-beam projection generation, pre-weighting, filtering with a modified ramp-filter* and *weighted back-projection*. The three last steps constitutes the fan-beam filtered back-projection method. Note that the object is denoted $f(x,y)$ and the reconstructed object

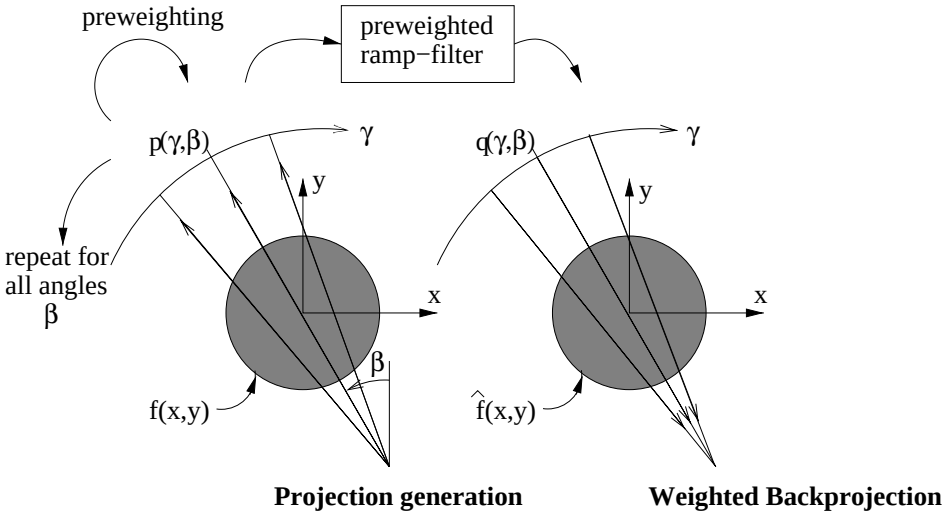

Figure 10: Fan-beam computed tomography illustrated.

$\hat{f}(x,y)$.

- The *fan-beam projection* $p_f(\gamma, \beta)$ contains line integrals through $f(x, y)$ as the geometry in Figure 10 illustrates.

- At first, the projection data is *pre-weighted* with

$$D \cdot \cos(\gamma). \tag{23}$$

- Then the modified *ramp-filter* is applied according to

$$q_f(\gamma, \beta) = \mathcal{F}_\gamma^{-1}\big[\mathcal{F}_\gamma[D \cdot \cos(\gamma) \cdot p_f(\gamma, \beta)] \cdot \mathcal{F}_\gamma[g(\gamma)]\big], \tag{24}$$

  where $\mathcal{F}_\gamma$ denotes the Fourier transform in the γ-direction, $\mathcal{F}_\gamma^{-1}$ denotes the inverse Fourier transform and the relation between the original ramp-filter $h(\ )$ in Equation (21) and the modified ramp-filter $g(\ )$ is

$$g(\gamma) = \frac{1}{2}\left(\frac{\gamma}{\sin\gamma}\right)^2 h(\gamma). \tag{25}$$

- The *weighted back-projection*

$$\hat{f}(x, y) = \int_0^{2\pi} \frac{1}{L(x, y, \beta)^2}\, q_f(\beta, \gamma(x, y, \beta)\, d\beta. \tag{26}$$

  includes a space dependent factor, $1/L^2$, where L is the distance between the actual point and the source and R is the distance between the origin and the source,

$$L(x, y, \beta)^2 = (R - x\sin\beta + y\cos\beta)^2 + (x\cos\beta + y\sin\beta)^2. \tag{27}$$

  The position γ, where the ray intersects the pixel (x, y) is given by

$$\gamma(x, y, \beta) = \arctan\left(\frac{x\cos\beta + y\sin\beta}{R - x\sin\beta + y\cos\beta}\right) \tag{28}$$

The explanation for Equation (27) and Equation (28) can be realized by this equation describing the rotation between two coordinate systems,

$$\begin{pmatrix} r \\ s \end{pmatrix} = \begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \tag{29}$$

and the geometry in Figure 11.

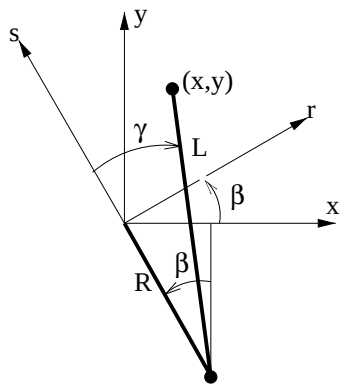

Figure 11: The geometry of a fan-beam system illustrated.

3.3 Rebinning from fan-beam rays and flat detector to parallel rays

The parallel filtered back-projection method has some advantages over the fan-beam filtered back-projection method. At first, it is simpler and faster. Secondly, the demanded projection angular interval is smaller. As can be seen in the back-projection equations, Equation (22) and (26), the projection interval for parallel data is $\theta \in [0, \pi[$, whereas it is $\beta \in [0, 2\pi[$ for fan-beam data. Nevertheless, it can be shown that the projection interval can be diminished to $\beta \in [-\gamma_{max}, \pi + \gamma_{max}[$, where γ_{max} is the maximum fan-angle. The cost for this is a more complicated reconstruction algorithm.

Another possibility is to collect fan-beam in the interval $\beta \in [-\gamma_{max}, \pi + \gamma_{max}[$ and rebin them to parallel data in the interval $\theta \in [0, \pi[$, followed by simple parallel filtered back-projection reconstruction. The name *rebinning* suggests that the fan-beam data could simply be resorted into parallel data, but the fact is that interpolation is also necessary.

Figure 12 shows the fan-beam geometry with flat detector. The S-axis is the axis along the detector and the s-axis is simply a scaled version translated to the origin. The fan-beam source angle is β and the fan-angle is γ. The maximum fan-angle is denoted γ_{max}. The coordinate axes for the parallel geometry (θ, r) is also included in the figure. From the figure, it it easy to derive the following relations between the different coordinates,

$$\theta = \beta + \gamma, \quad r = s\cos(\gamma), \tag{30}$$

or equivalently

$$\theta = \beta + \arctan\left(\frac{s}{D}\right), \quad r = \frac{sD}{\sqrt{D^2 + s^2}}, \tag{31}$$

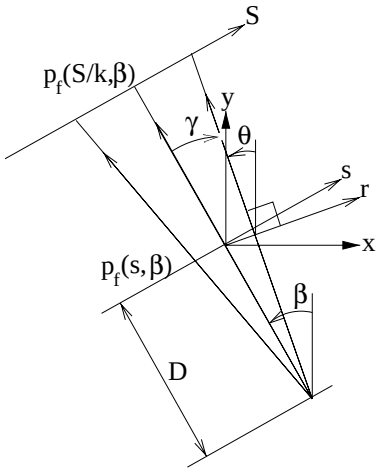

Figure 12: Fan-beam computed tomography illustrated.

or equivalently

$$\beta = \theta - \arcsin\left(\frac{r}{D}\right). \quad s = \frac{rD}{\sqrt{D^2 - r^2}}, \tag{32}$$

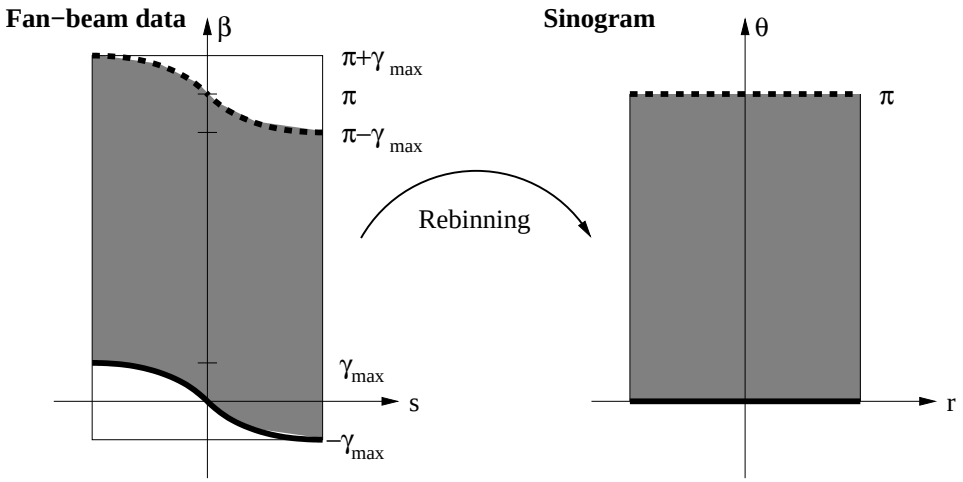

Figure 13: Rebinning.

Figure 13 now shows rebinning from the fan-beam projection coordinates (β, s) to the parallel coordinates (θ, r). The (θ, r)-scheme consequently shows 1D parallel projections piled up on each other. Such a scheme is often called a *sinogram*. As mentioned above, the indicated 2D interval with $\theta \in [0, \pi[$ and $r \in [-r_{max}, r_{max}]$ is sufficient for reconstruction. This interval can be mapped onto the (β, s)-space

by using Equation (31). The location of the interval is indicated in the figure and shows that $\beta \in [-\gamma_{max}, \pi + \gamma_{max}[$ is necessary and sufficient for reconstruction. The maximum fan-angle γ_{max} can be calculated from Equation (30) by inserting $r = r_{max}$. From figure 13, it can also be noted that some of the data is redundant, i.e. they need not to be utilized for rebinning to the (θ, r)-space.

4 The physics of X-ray projections

Ideally, an X-ray with N_0 photons transversing an object along a line L is attenuated to N photons according to

$$N = N_0 \cdot e^{-\int_L \mu(l)dl}, \tag{33}$$

where μ is the attenuation coefficient of the material in the object. By taking the natural logarithm on both sides we can express the line integral along L as

$$\int_L \mu(l)dl = \ln \frac{N_0}{N}. \tag{34}$$

This is the monochromatic case, compare these equations with the polychromatic case in Equation (11), (12) and (13). A set of line integrals forms a projection and a set of projections can be reconstructed to a CT-image, see section 3. Consequently, a CT-image $\hat{f}(x, y)$ shows the attenuation coefficients in the reconstructed slice, $\hat{f}(x, y) = \mu(x, y)$.

More precisely, the μ-value is the probability for interaction per unit length. Normally, the unit for μ is attenuation per meter or simply $[1/m]$. The μ-value is composed of three different interaction processes,

$$\mu = \sigma_{Co} + \sigma_{In} + \sigma_{Ph} \tag{35}$$

where the three interaction processes are

- Coherent scattering (Rayleigh scattering) σ_{Co}: This is the interaction of an X-ray photon and an atom. As a result of the interaction, the X-ray photon does not loose any energy, but is deflected from its original path.

- Incoherent scattering (Compton scattering) σ_{In}: This is the interaction of an X-ray photon and a free or loosely bound electron in one of the outer shells of an atom. As a result of the interaction, the X-ray photon looses some of its energy and is deflected from its original path. The energy loss is gained by the electron.

- The photo-electric absorption σ_{Ph}: An X-ray photon imparts all its energy to a tightly bound inner electron in an atom. The electron uses some of the acquired energy to overcome the binding energy within its shell. The rest of the energy is used for kinetic energy.

A more thorough treatise of these three types of interactions can be found in any basic literature on modern physics.

Coherent and incoherent scattering do not only contribute to the attenuation coefficient μ, they also produce new X-rays traveling in a new direction, so called scattering. If these rays enters the detector, they cannot be distinguished from primary rays. The scattered rays are not wanted since they cause errors in the measurement of the line integral in equation (34). It is possible to diminish scattering by using mechanical arrangements of collimators and rasters.

A good simulation of scattering is nevertheless valuable and can be realistically done with so called Monte Carlo simulation. There is one disadvantage with Monte Carlo simulation, however, it is very slow. The simulation program *take* cannot do scatter simulation. Instead, we hope to combine high resolution projection generation in *take* with low resolution scattering simulation such as the one in [3]. We think that this is possible since scattering seems to be a low frequency distortion.

5 Experiments with simulated data

5.1 Demonstration of voxel data and projection generation.

The program *take* was used operate on a Plexiglass object consisting of a cylinder with a cylindrical hole. At first, the object was voxelized. In Figure 14, three cross-sections of the voxelized object are shown. Then *take* was used to generate

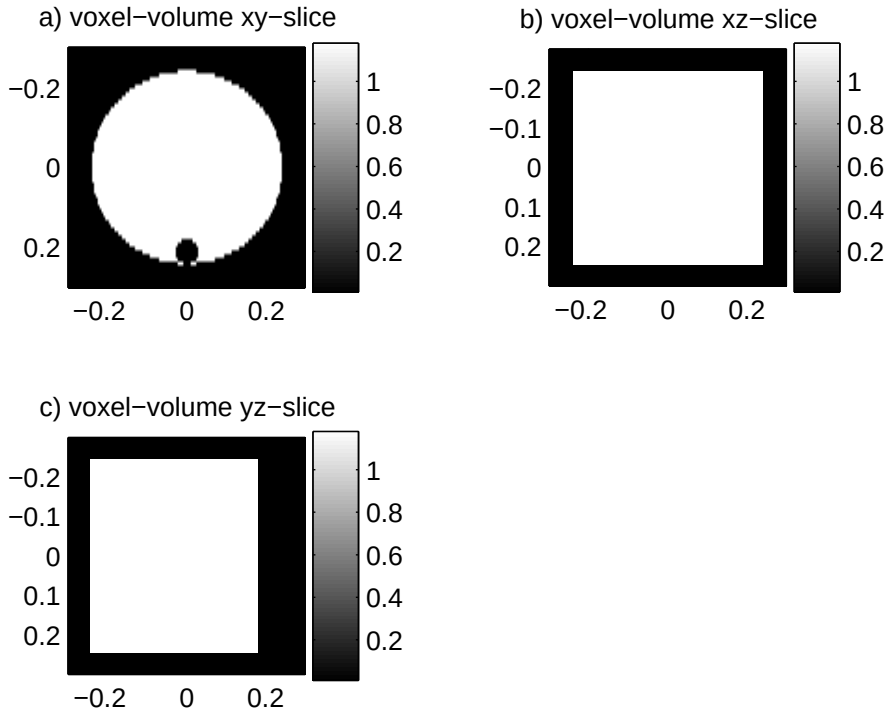

Figure 14: Three cross-sections of the voxelized 3D object. a) xy-slice b) xz-slice c) yz-slice

cone-beam projections for the Plexiglass object with a monochromatic spectrum at 30 keV. The number of projections was 180, spanning an angular interval of $0^\circ - 360^\circ$. The detector was specified to be 64 channels and 65 rows. The projection data are shown in Figure 15. Figure 15a-c show 2D projection data at projection angle 0°, 60° and 90°. Note how the little hole in the object change position in projection data. Figure 15d shows the central row of projection data at the whole angular interval, $0^\circ - 360^\circ$, i.e. this is equal to a complete set of fanbeam projection data. Note how the little hole in the object gives rise to a skewed sinusoid in the fan-beam projection data. The MATLAB code for this experiment is given in

Appendix, Section 8.1.

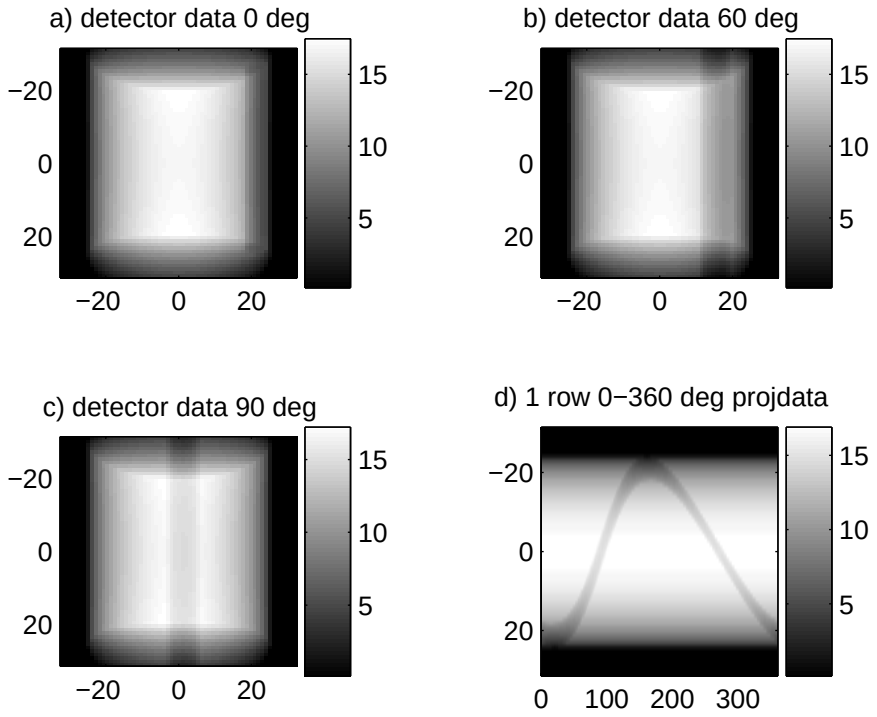

Figure 15: 2D projection data from the 3D object. a) 2D projection data at $0°$. b) 2D projection data at $60°$. c) 2D projection data at $90°$. d) One row (the central) of projection data, $0° - 360°$.

5.2 Demonstration of fan-beam CT reconstruction.

The program *take* was used to generate fan-beam projections for a Plexiglass object and monochromatic spectrum at 30 keV. The material file for Plexiglass is given in Section 8.5.2 and a material curve for Plexiglass is shown in Figure 17 b). The number of projections was 180 and the detector was specified to be 64 channels but only one row. Therefore, one projection can be regarded as 1D instead of 2D. The fan-beam projection data for all projection angles is shown as a 2D image in Figure 16a. Note how the little hole in the object gives rise to a skewed sinusoid in the fan-beam projection data. The data was reconstructed by the fan-beam filtered back-projection method described in Section 3.2. The reconstruction result is shown in Figure 16b. Cross-sections along the middle horizontal and vertical

lines are shown in Figure 16c. The MATLAB code for this experiment is given in Appendix, Section 8.2.

Remember that the current experiment uses a monochromatic spectrum at 30 keV, recall Equation (10) and see Section 8.5.2 with the material data for Plexiglass. Also, check the file `phm.txt` in Section 8.2 giving that the density is $1.19\mathrm{g}/\mathrm{cm}^3$. This altogether gives that

$$\mu(30\mathrm{keV}) = 100 \cdot 1.19 \cdot \left(3.68 \cdot 10^{-2} + 1.78 \cdot 10^{-1} + 8.35 \cdot 10^{-2}\right) = 35.5\mathrm{m}^{-1}.$$

This value is in accordance with Figure 16c, which gives that the μ-value for the reconstructed object is approximately 35.

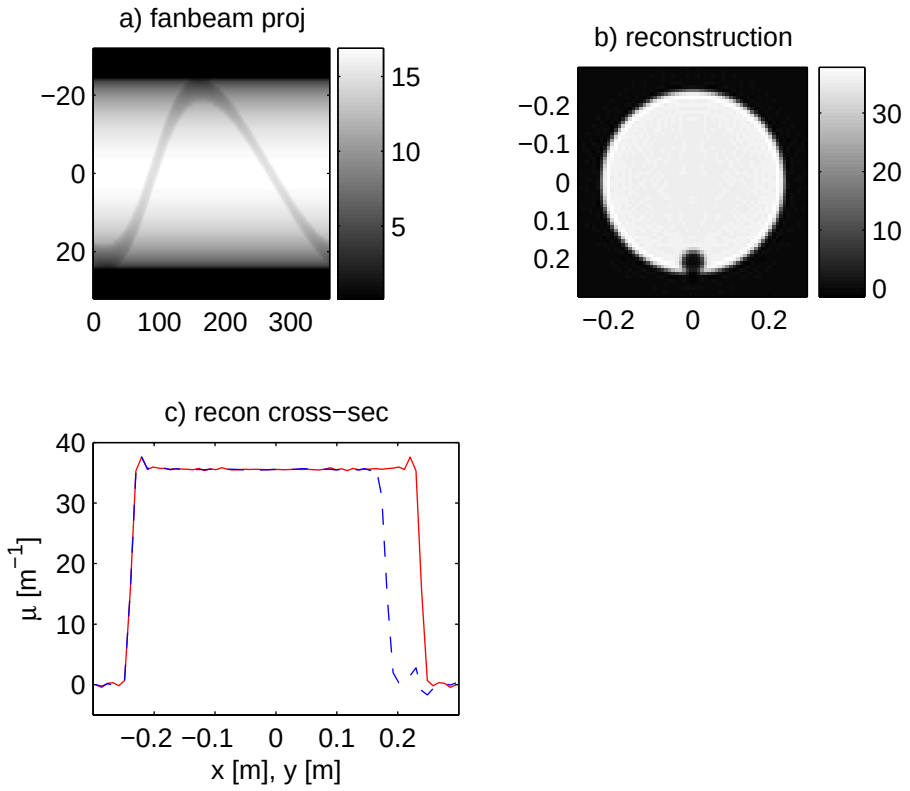

Figure 16: a) Fan-beam projection data. b) Reconstruction result. c) Cross-sections along the middle horizontal and vertical lines in the reconstruction.

5.3 Demonstration of 1D beam hardening.

In Figure 17 we show how beam hardening artefacts can be demonstrated with *take*. Figure 17 a) and b) show the X-ray source spectrum and the attenuation for Plexiglass, respectively. Compare these curves with the data in Section 8.5. In the experiment depicted in Figure 17 c), projection data is simulated for a slab of Plexiglass where the thickness is varied from 1 cm to 10 cm. For a monochromatic beam we have

$$p_{mono}(l) = \mu(E_0) \cdot l, \tag{36}$$

where $E_0 = 30$ keV. Consequently, for a monochromatic beam projection data is proportional to the thickness of the slab. The polychromatic formula is more complicated

$$p_{poly}(l) = \ln\left(\frac{\int E\ N(E)\ dE}{\int E\ N(E) \exp(-\mu(E)l)\ dE}\right), \tag{37}$$

which is a combination of Equation (11), (12) and (13). Since the attenuation coefficient $\mu(E)$ decreases with increasing photon energy E, the lower energy part of the spectrum will be more attenuated than the higher energy part. Therefore, along the path of the beam through the object, the percentage of high energy in the beam will increase. This is beam hardening. In particular if we try to estimate the thickness of an object by measuring projection data we will get too low values, as shown in Figure 17 c). The MATLAB code for this experiment is given in Appendix, Section 8.3.

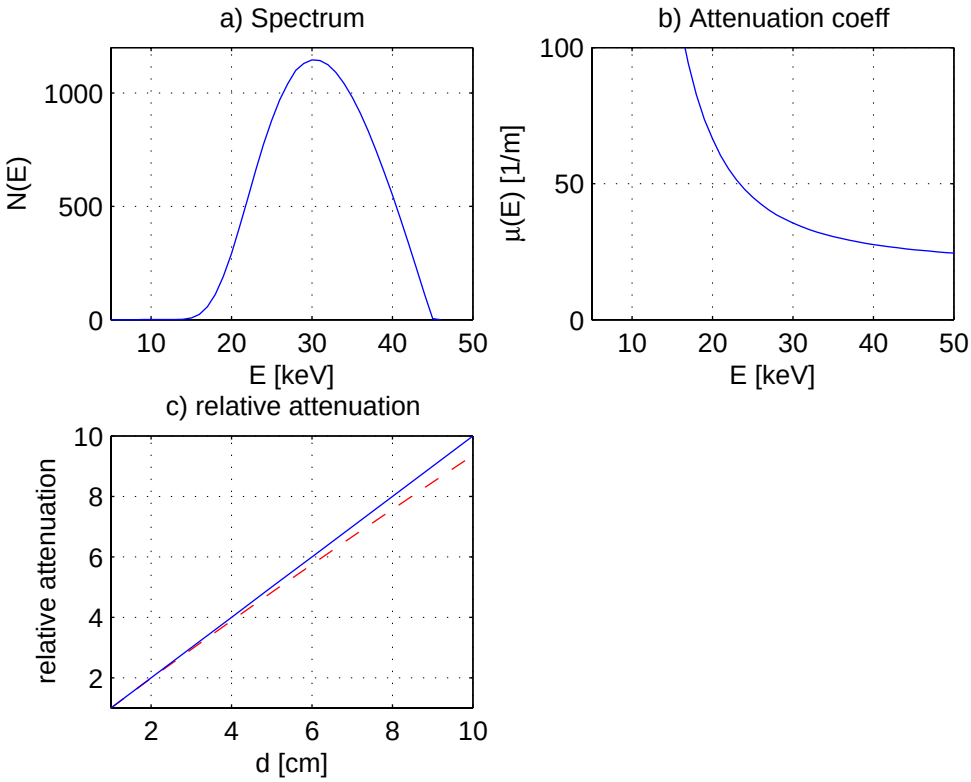

Figure 17: a) X-ray spectrum $N(E)$. b) Attenuation coefficient $\mu(E)$ for Plexiglass. c) Normalized projection data from a slab of Plexiglass as a function of thickness. Monochromatic beam (solid line) and polychromatic beam (dashed line).

5.4 Demonstration of beam hardening in CT reconstruction.

The program *take* was used to generate fan-beam projections for both monochromatic and polychromatic spectrum. An energy level of 30 keV was used for the monochromatic case. The object consisted of Plexiglass. Figure 17 a) and b) show the polychromatic X-ray source spectrum and the attenuation for Plexiglass, respectively. Compare these curves with the data in Section 8.5. The data was reconstructed by first using rebinning as described i Section 3.3 followed by the parallel filtered back-projection method as described in Section 3.2. In Figure 18a, the monochromatic fan-beam projection data is shown and in Figure 18b, the monochromatic rebinned projection data is shown. The polychromatic projection data is of course a little different, but will not be distinguishable in an image. The reconstructions from monochromatic and polychromatic data are shown in

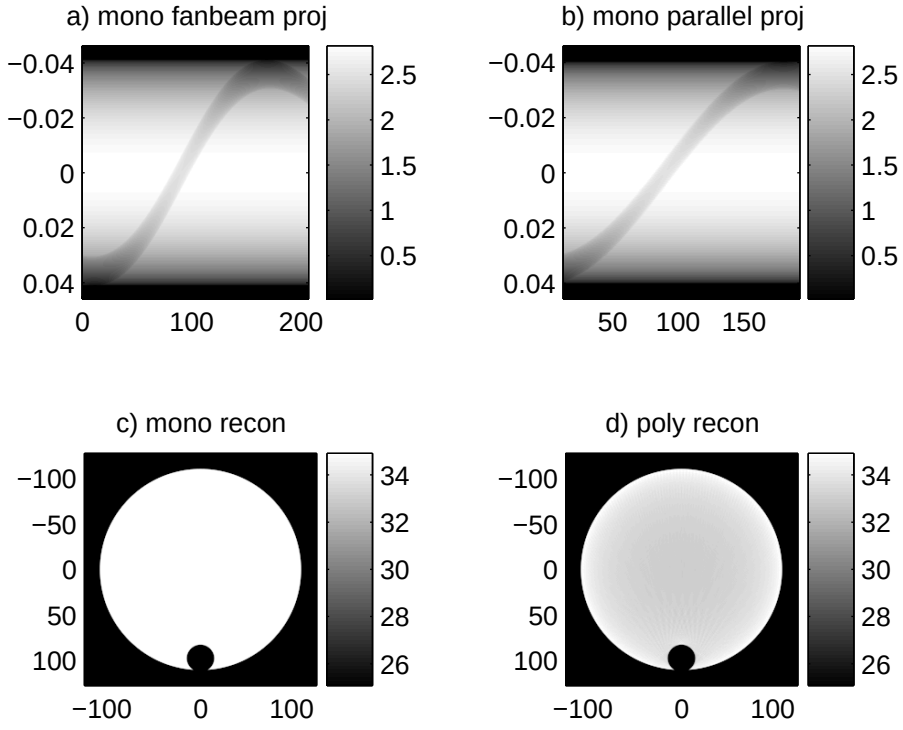

Figure 18: a) Monochromatic fan-beam projection data. b) Monochromatic rebinned projection data. c) Reconstruction from monochromatic data. d) Reconstruction from polychromatic data.

Figure 18c and d, respectively. Cross-sections along the middle vertical line are shown in Figure 19. The typical beam hardening artefact is shown to be a cup-

ping artefact. This artefact is due to that the rays passing through the center of the cylinder are under-estimated. The MATLAB code for this experiment is given in Appendix, Section 8.4.

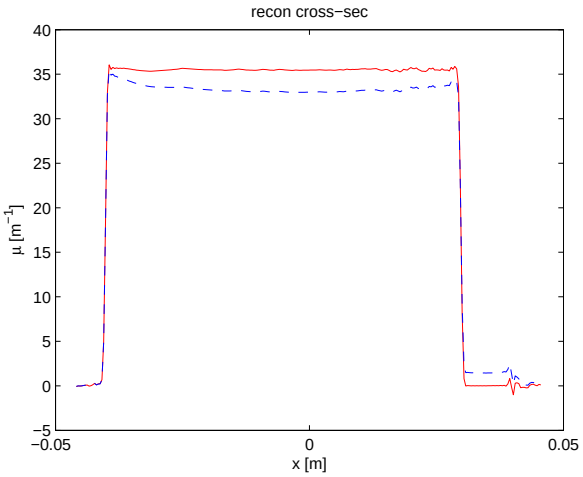

Figure 19: Reconstructed Plexiglass cylinder. Monochromatic beam (solid line) and polychromatic beam (dashed line).

6 Experiment with real data

6.1 The Tomohawk system

Tomohawk is a system for industrial X-ray radiography and computed tomography [6]. The basic components are shown in Figure 20. An X-ray source generates a cone-beam projection on the detector, which consists of an image intensifier and a CCD-camera. A turntable is used for positioning the test object. The image intensifier converts the X-rays into visible light, which generates an image in the CCD-camera. The image is then transferred to the personal computer. In the case of radiography, this image is the final result. In the case of computed tomography several images collected from different rotational positions of the turntable serves as in-data for a reconstruction algorithm giving slices or a whole 3D-volume of the object. The personal computer both processes the reconstruction algorithm and controls the turntable via the manipulator control unit.

Unfortunately, an image intensifier introduces geometric distortion. Thus, it is necessary to perform a two-dimensional geometrical correction of the image obtained from the CCD-camera.

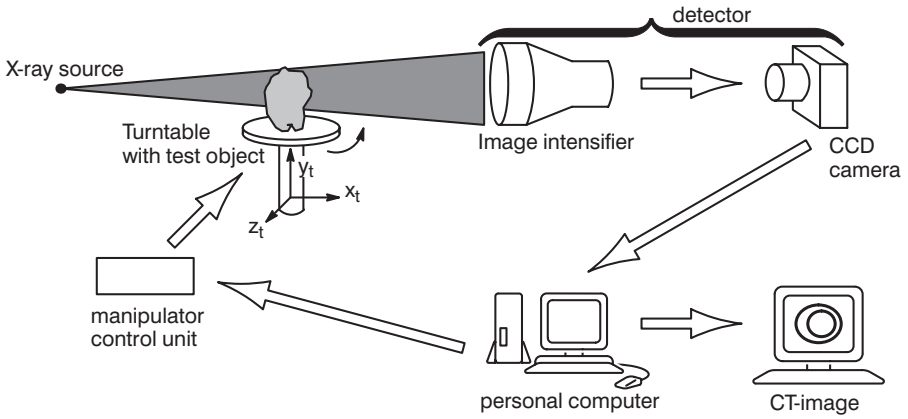

Figure 20: The Tomohawk system.

6.2 The experiment

To test our model we have carried out a series of measurements on the Tomohawk system. The phantom was a Plexiglass cylinder, diameter 80 mm, with three drilled holes of diameters 10 mm, 15 mm, 20 mm, respectively. We had access to a data base of previously measured spectrums for the Tomohawk X-ray tube for varying acceleration voltages. Before using the spectrum in *take* we compensated for the attenuation in air and a $l_{Al} = 2$ mm thick aluminum filter:

$$N_f(E) = N(E) \cdot e^{-\mu_{air} l_{air} - \mu_{Al} l_{Al}}, \tag{38}$$

see Figure 21 a),b).

We had also access to a simulated single-event distribution $\Omega(E', E)$ for the detector used in the Tomohawk system. In $\Omega(E', E)$, E is the incoming energy and E' is the outgoing energy, i.e. one energy level E will be converted to other energies in the detector. We now modify (11) to allow for detector sensitivity

$$S_L = \int E' \left(\int \Omega(E', E) A_L(E) N_f(E)\, dE \right) dE', \tag{39}$$

where $A_L(E)$ is the attenuation along the line L,

$$A_L(E) = \exp\left(- \int_L \mu(l, E)\, dl \right). \tag{40}$$

By rearranging (39) we get

$$\begin{aligned} S_L &= \int \left(\int E' \Omega(E', E)\, dE' \right) A_L(E) N_f(E)\, dE \\ &= \int \Omega_E(E)\, A_L(E)\, N_f(E)\, dE, \end{aligned} \tag{41}$$

where

$$\Omega_E(E) = \int E'\Omega(E', E)\, dE'. \tag{42}$$

This equation can be precomputed. It is illustrated for detector used in the Tomohawk system in Figure 21 c).

We finish this theoretical discussion by pointing out that by inserting (40) into our original formula for calculation of a detector value (11) we get

$$S_L = \int E\, A_L(E)\, N(E)\, dE, \tag{43}$$

which is (41) without compensation for detector sensitivity, air and aluminum filter. See Figure 8a. The energy spectrum $E \cdot N(E)$ should be replace by $\Omega_E(E) \cdot N_f(E)$ if detector sensitivity is to be regarded. The combination of $N(E)$, $N_f(E)$ and $\Omega_E(E)$ to the effective spectrum $\Omega_E(E) \cdot N_f(E)$ is shown Figure 21.

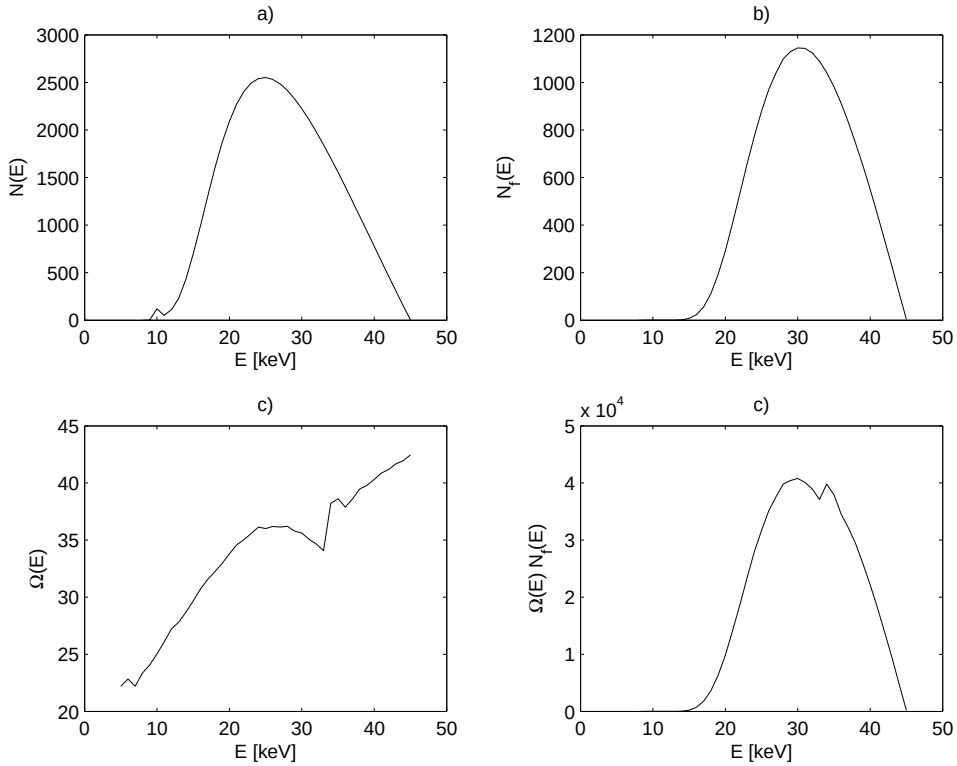

Figure 21: a) Original spectrum $N(E)$. b) Filtered spectrum $N_f(E)$. c) Detector sensitivity $\Omega_E(E)$. d) Effective spectrum $\Omega_E(E) \cdot N_f(E)$.

Projection data, see Figure 22 a), was collected from the Tomohawk and reconstructed with a fan-beam algorithm, see Figure 22 b). The experiment was also

simulated in *take*, see Figure 22 d), and reconstructed, see Figure 22 e). The simulated reconstruction result, see Figure 22 f), shows the typical beam-hardening cupping effect, which is however unfortunately not visible in the measured reconstruction result, see Figure 22 c).

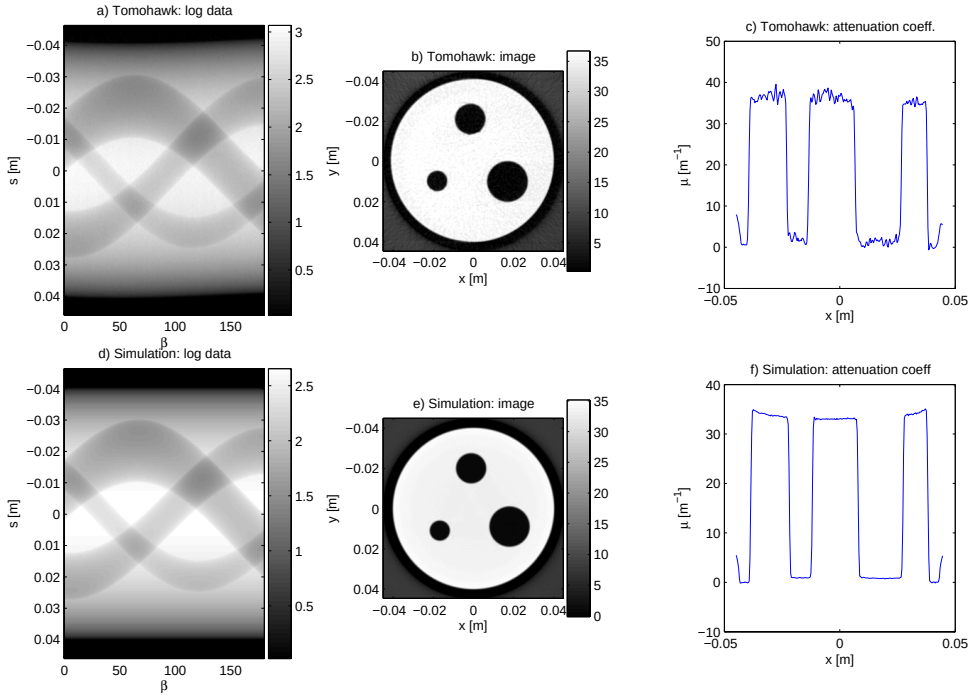

Figure 22: a) Measured projection data from a Plexiglass phantom b) Reconstruction from measured projection data c) Reconstructed attenuation coefficient (through the two lower holes) d) Simulated projection data from a Plexiglass phantom e) Reconstruction from simulated projection data f) Reconstructed attenuation coefficient (through the two lower holes).

7 Future work

As mentioned in Section 4 scattering is not implemented in *take*. Instead, we want to combine high resolution projection data generated by *take*, with low resolution scattering generated by, for example, [3]. In [3], scattering is computed by Monte Carlo simulation, a good but time consuming method. Therefore, we would like to have a simpler approximative scatter simulation included in *take*. Modeling of noise, both Poisson and thermic, are lacking in *take*. It is desired that the projection generating procedure through voxel volumes would be implemented for different energy levels.

The last experiment in this report, a comparison between real and simulated data, wasn't completely successful and should therefore be analyzed and improved.

References

[1] P. M. Joseph, An Improved Algorithm for Reprojecting Rays Through Pixel Images, *IEEE Transactions on Medical Imaging MI-1, 192-196*, 1982

[2] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*, ISBN 0-87942-198-3, IEEE PRESS, 1987

[3] A. Malušek, Maria Magnusson Seger, Michael Sandborg, Gudrun Alm Carlsson, Effect of Scatter on Reconstructed Image Quality in Cone Beam CT: Evaluation of a Scatter-reduction Optimization Function, *Accepted for publication in Radiation Protection Dosimetry*, 2005

[4] J. Müller-Merbach, Simulation of X-ray Projections for Experimental 3D Tomography, *Report LiTH-ISY-R-1866*, Dept of EE, Linköpings University, SE-581 83, Linköping, Sweden, 1996

[5] F. Natterer, *The Mathematics of Computerized Tomography*, ISBN 0-471-90959-9, Wiley, 1986

[6] Tomohawk 2, Linköping University, Industrial Computed Tomography System. *Operators manual*, 1992

[7] H. Turbell, New Functionality in *take* version 2.1, *Web material only, currently on*, Linköping, Sweden, 1999

8 Appendix

8.1 Experiment 1

8.1.1 main program

```
f = take('mono=30', 'initfile=take1.txt');

sizevol = size(f);
N1 = sizevol(1);                        % N1*N2*N3 = volume size
N2 = sizevol(2);
N3 = sizevol(3);

rec = 0.6;
vec1 = (-(N1-1)/2:(N1-1)/2)*rec/N1;     % image axis
vec2 = (-(N2-1)/2:(N2-1)/2)*rec/N2;     % image axis
vec3 = (-(N3-1)/2:(N3-1)/2)*rec/N3;     % image axis

f1 = zeros(N1,N2);
f1(:,:) = f(:,:,N3/2);
f3 = zeros(N2,N3);
f3(:,:) = f(N1/2,:,:);
f2 = zeros(N1,N3);
f2(:,:) = f(:,N2/2,:);

figure(1);
colormap(gray);
subplot(2,2,1), imagesc(vec2, vec1, f1');
colorbar;
title('a) voxel-volume xy-slice')
axis image;
subplot(2,2,2), imagesc(vec1, vec3, f2');
colorbar;
title('b) voxel-volume xz-slice')
axis image;
subplot(2,2,3), imagesc(vec2, vec3, f3');
colorbar;
title('c) voxel-volume yz-slice')
axis image;

f = take('mono=30', 'attenuation=takelog','initfile=take2.txt');

sizevol = size(f);
N1 = sizevol(1);                % N1 = nr of channels
N2 = sizevol(2);                % N2 = nr of rows
N3 = sizevol(3);                % N3 = nr of projections


vec1 = (-(N1-1)/2:(N1-1)/2);    % image axes
vec2 = (-(N2-1)/2:(N2-1)/2);
vec3 = 1:360;
f1 = zeros(N1,N2);
f1(:,:) = f(:,:,1);
f2 = zeros(N1,N2);
f2(:,:) = f(:,:,N3/6);
f3 = zeros(N1,N2);
f3(:,:) = f(:,:,N3/4);
f4 = zeros(N1,N3);
f4(:,:) = f(:,(N2+1)/2,:);

figure(2);
```

```
colormap(gray);
subplot(2,2,1), imagesc(vec2, vec1, f1');
colorbar;
title('a) detector data 0 deg')
axis image;
subplot(2,2,2), imagesc(vec2, vec1, f2');
colorbar;
title('b) detector data 60 deg')
axis image;
subplot(2,2,3), imagesc(vec2, vec1, f3');
colorbar;
title('c) detector data 90 deg')
axis image;
subplot(2,2,4), imagesc(vec3, vec1, f4);
colorbar;
title('d) 1 row 0-360 deg projdata')
```

8.1.2 take1.txt

```
# init file for take
#-------------------------
verbose
phantom      = phm.txt
detector     = det.txt
trajectory   = trj.txt
#-------------------------
voxelization = vox.dat
voxelnr = 128 64 32
voxelsize = 0.6 0.6 0.6
voxelcenter = 0.0 0.0 0.0
voxelpoints = 1
#-------------------------
energyspectrum = spm.mat
```

8.1.3 take2.txt

```
# init file for take
#-------------------------
verbose
phantom      = phm.txt
detector     = det.txt
trajectory   = trj.txt
#-------------------------
#voxelization = vox.dat
voxelnr = 128 64 32
voxelsize = 0.6 0.6 0.6
voxelcenter = 0.0 0.0 0.0
voxelpoints = 1
#-------------------------
energyspectrum = spm.mat
```

8.1.4 det.txt

```
# detector file
#------------------
shape=cylindricalAroundSource
fanangle=0.4571
height=0.6
channels=63
rows=65
xypoints=1
```

8.1.5 trj.txt

```
# trajectory file
# ----------------
projections = 180
#
# source pos. 0
  0.700       0.00000000  0.00000000
#
# detector pos. 0
  0.00000000 -1.00000000  0.00000000  0.00000000
  1.00000000  0.00000000  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
#
# transformation matrix
  0.99939083 -0.03489950  0.00000000  0.00000000
  0.03489950  0.99939083  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
```

8.1.6 phm.txt

```
# PlexiGlass phantom
#-----------------------------
cylind  a=0.240 b=0.240 c=0.25 x=0.0 y=0.0 z=0.0 theta=0.0 phi=0.0 dens=1.19 mat=0
cylind  a=0.030 b=0.030 c=0.25 x=0.0 y=0.21 z=0.0 theta=0.0 phi=0.0 dens=-1.19 mat=0
#
material = 0 ../materials/plexi.mat
```

8.2 Experiment 2

8.2.1 main program

```
g = take('mono=30');

rec = 0.6;          % reconstructed image size

[N,R,M] = size(g);  % N = nr of pixels per projection
                    % R = nr of rows
                    % M = nr of projections

x = (-(N-1)/2:(N-1)/2);          % detector axis
a = (0:M-1)*360/M;               % projection angles

xvec = (-(N-1)/2:(N-1)/2)*rec/N; % image axis

geo = [0.7 0.4571*2/64];         % fanbeam geometry
firstang = 0;

g1 = zeros(N,M);
g1(:,:) = g(:,1,:);
g1 = log(max(max(g1))) - log(g1); % take the logarithm
fhat = fanFB(g1, geo, rec, firstang);      % fanbeam reconstr

figure(1);
subplot(2,2,1), imagesc(a, x, g1);
colorbar;
title('a) fanbeam proj')

subplot(2,2,2), imagesc(xvec, xvec, fhat);
colorbar;
title('b) reconstruction')
axis image;
```

```
subplot(2,2,3), plot(xvec, fhat(32,:), '-r', ...
     xvec, fhat(:,32), '--b');
title('c) recon cross-sec')
xlabel('x [m], y [m]');
ylabel('\mu [m^{-1}]');
axis([-rec/2 rec/2 -5 40]);
```

8.2.2 take.txt

```
# init file for take
#-------------------------
verbose      = 1
phantom      = phm.txt
detector     = det.txt
trajectory   = trj.txt
#
energyspectrum = spm.mat
```

8.2.3 det.txt

```
# detector file
#-------------------
shape=cylindricalAroundSource
fanangle=0.4571
height=0.0014
channels=64
rows=1
xypoints=1
```

8.2.4 trj.txt

```
# trajectory file
# ----------------
projections = 180
#
# source pos. 0
  0.700       0.00000000  0.00000000
#
# detector pos. 0
  0.00000000 -1.00000000  0.00000000  0.00000000
  1.00000000  0.00000000  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
#
# transformation matrix
  0.99939083 -0.03489950  0.00000000  0.00000000
  0.03489950  0.99939083  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
```

8.2.5 phm.txt

```
# PlexiGlass phantom
#-----------------------------
cylind a=0.240 b=0.240 c=0.25 x=0.0 y=0.0 z=0.0 theta=0.0 phi=0.0 dens=1.19 mat=0
cylind a=0.030 b=0.030 c=0.25 x=0.0 y=0.21 z=0.0 theta=0.0 phi=0.0 dens=-1.19 mat=0
#
material = 0 ../materials/plexi.mat
```

8.2.6 fanFB.m

```
function f = fanFB(g, geo, rec, firstang)
```

```
%
% f = fanFB(g, geo, rec, firstang)
%
% geo(1):   distance source <-> origo
% geo(2):   angle between rays
% rec:      image x size
% firstang: angle for first projection [rad]

if nargin ~= 4
  fprintf('The number of arguments must be 4!\n');
  return;
end

[M,N] = size(g);   % M = nr of pixels per projection,
                   % N = nr of projections

D = geo(1);
dalfa = geo(2);

if N == 1          % special case for rot. sym. obj.
  N = 2*M;
  g = g*ones(1,N);
end

Sx = rec;
Sy = Sx;
m0 = M;
n0 = M;
startx = -Sx/2 + 0.5*Sx/M;
starty = startx;

% projection angles
dbeta = 2*pi/N;
beta = 0+firstang:dbeta:2*pi+firstang-dbeta;


% detector angles
alfa = dalfa*(-(M-1)/2:(M-1)/2)';

% preweight the projections with D * cos(alfa)
g = g.*(D*cos(alfa)*ones(1,N));

% rampfilter the projections
q = ramp(g,dalfa);

% the grid for the reconstructed image
dx = Sx/m0;
dy = Sy/n0;

[X,Y] = meshgrid(startx:dx:startx+(m0-1)*dx, starty:dy:starty+(n0-1)*dy);

f = zeros(size(X));

% weighted backprojection
for i=1:N
   x0 = D*sin(beta(i));
   y0 = D*cos(beta(i));
   L2 = (x0-X).^2 + (y0+Y).^2;
   gamma = -adjust(atan2(x0-X,y0+Y) - beta(i));
   f = f + interp2(ones(M,1)*(1:N), alfa*ones(1,N), q, i*ones(m0,m0), gamma)./L2;
end
```

```
% find NaN:s and replace them with 0
loc = find(isnan(f)==1);
f(loc) = 0;

f = f*dbeta*dalfa;

function y = adjust(x)
y=x;
y(find(x>=pi)) = y(find(x>=pi)) - 2*pi;
y(find(x<-pi)) = y(find(x<-pi)) + 2*pi;
```

8.2.7 ramp.m

```
function q = ramp(g, dalfa)
%
% q = ramp(g, dalfa)
% rampfilter the columns of g
%

[M,N] = size(g);

M2 = 2^ceil(log2(M));
r = zeros(2*M2,1);
if nargin==1
  r(M2+1) = 1/4;
  r(2:2:end) = -1 ./(pi^2*(-M2+1:2:M2-1).^2);
end
if nargin==2
  r(M2+1) = 1/(8*dalfa^2);
  r(2:2:end) = -1 ./(2*pi^2*sin((-M2+1:2:M2-1)*dalfa).^2);
end
w = cos(pi*(-M2:M2)'/(2*M2)); % low pass weighting
R = (w(1:end-1).*fftshift(fft(ifftshift(r))))*ones(1,N);

q = zpadcol(g,2*M2);
q = real(fftshift(ifft(ifftshift (R.*fftshift(fft(ifftshift(q)))))));
q = zpadcol(q,M);

function y = zpadcol(x,m2)
%
% y = zpadcol(x,m2)
% pad or unpad
%

[m,n] = size(x);

if mod(m-m2,2) == 0
  offset = abs((m-m2)/2);
else
  offset = (abs(m-m2)+1)/2;
end

if m2>m
  y = [zeros(offset,n); x; zeros(m2-m-offset,n)];
else
  y = x(offset+1:offset+m2,:);
end
```

8.3 Experiment 3

8.3.1 main program

```
% Spectrum data
```

```
load spm.mat
subplot(2,2,1), plot(f(:,1), f(:,2), '-b');
title('a) Spectrum');
xlabel('E [keV]'); ylabel('N(E)');
axis([5 50 0 1200]); grid;

% Material data
load ../materials/plexi.mat
dens = 1.19;                % Plexiglass density
pmat = 100 * dens * (f(:,2) + f(:,3) + f(:,4));
subplot(2,2,2), plot(1000*f(:,1), pmat, '-b');
title('b) Attenuation coeff');
xlabel('E [keV]'); ylabel('\mu(E) [1/m]');
axis([5 50 0 100]); grid;

% Loop over phantoms with different thickness and
% compute the relative attenuation
g1 = zeros(1,10);
for i=1:10
  p = ['phantom = phm' num2str(i) '.txt'];
  y = take(p);
  g1(i) = log(y(1)/y(33));
end
g0 = zeros(1,10);
for i=1:10
  p = ['phantom = phm' num2str(i) '.txt'];
  y = take(p,'mono=30');
  g0(i) = log(y(1)/y(33));
end
subplot(2,2,3), plot(1:10, g1/g1(1), '--r', 1:10, g0/g0(1), '-b')
title('c) relative attenuation');
xlabel('d [cm]'); ylabel('relative attenuation');
axis([1 10 1 10]); grid;
```

8.3.2 take.txt

```
#-------------------------
phantom      = phm.txt
detector     = det.txt
trajectory   = trj.txt
#
energyspectrum = spm.mat
```

8.3.3 det.txt

```
# detector file
#-------------------
xlen=0.2
ylen=0.001
xpix=65
ypix=1
xypoints=1
```

8.3.4 trj.txt

```
# trajectory file
# ----------------
projections = 1
# source pos. 0
  0.7         0.00000000  0.00000000
#
# detector pos. 0
```

```
  0.00000000 -1.00000000  0.00000000  0.00000000
  1.00000000  0.00000000  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
#
# transform
  0.99991660 -0.01291508  0.00000000  0.00000000
  0.01291508  0.99991660  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
```

8.3.5 phm1.txt

```
# plexi phantom
#------------------------------
# cylinder
cylind  a=0.005 b=0.040 c=0.01 x=0.0 y=0.0 z=0.0 theta=0.0 phi=0.0 dens=1.19 mat=0
#
material = 0 ../materials/plexi.mat
```

8.3.6 phm2.txt, phm3.txt, ..., phm10.txt

The files `phm2.txt, phm3.txt, ..., phm10.txt` look similar as `phm1.txt` above. The only exception is that `a` increaces according to:
`a=0.005`, `a=0.010`, `a=0.015`, ..., `a=0.050`.

8.4 Experiment 4

8.4.1 main program

```
L = 0.2;                % distance source - rot. center
dx0 = 0.00036;          % detector element size
dfi0 = 0.74;            % angle increment in degrees
M0 = 256;               % nr of pixels per projection
N0 = 280;               % nr of projections

dx1 = 0.00036;          % new detector element size
dfi1 = 1;               % new angle increment in degrees
M1 = 255;               % new nr of pixels per projection
N1 = 180;               % new nr of projections

gamma = 180*atan(M0/2*dx0/L)/pi;% fanbeam angle
firstang = pi/2 + pi*gamma/180; % first angle after rebinning

x0 = (-M0/2:M0/2-1)*dx0;        % detector axis
a0 = (0:N0-1)*dfi0;             % detector axis

x1 = (-M1/2:M1/2-1)*dx1;        % new detector axis
a1 = (0:N1-1)*dfi1+gamma;       % new detector axis

xvec = (-(M1-1)/2:(M1-1)/2);    % image axis

% ************ mono ***************************
g = take('mono=30');
[N,R,M] = size(g);  % N = nr of pixels per projection
                    % R = nr of rows
                    % M = nr of projections
g = log(max(max(g))) - log(g);
g1 = zeros(N,M);
g1(:,:) = g(:,1,:);

reb = rebinning(g1, L, dx0, dfi0, 1, 0, dx1, dfi1, M1, N1);
fhat0 = parFB(reb, dx1, firstang);
```

```
figure(1);
subplot(2,2,1), imagesc(a0,x0,g1);
colorbar;
title('a) mono fanbeam proj')
subplot(2,2,2), imagesc(a1,x1,reb);
colorbar;
title('b) mono parallel proj')
subplot(2,2,3), imagesc(xvec, xvec, fhat0', [25,35]);
colorbar;
title('c) mono recon')
axis image;

% ************ poly ***************************
g = take;
g = log(max(max(g))) - log(g);
g1 = zeros(N,M);
g1(:,:) = g(:,1,:);

reb = rebinning(g1, L, dx0, dfi0, 1, 0, dx1, dfi1, M1, N1);
fhat = parFB(reb, dx1, firstang);

subplot(2,2,4), imagesc(xvec, xvec, fhat', [25,35]);
colorbar;
title('d) poly recon')
axis image;

figure(2);
subplot(2,2,1), plot(x1, fhat0(128,:), '-r', ...
                     x1, fhat(128,:), '--b');
title('recon cross-sec')
xlabel('x [m]');
ylabel('\mu [m^{-1}]');
axis([-0.05 0.05 -5 40]);
```

8.4.2 take.txt

```
# init file for take
#-------------------------
verbose      = 1
phantom      = phm.txt
detector     = det.txt
trajectory   = trj.txt
#
energyspectrum = spm.mat
```

8.4.3 det.txt

```
# detector file
#-------------------
xlen=0.09216
ylen=0.00012
xpix=256
ypix=1
xypoints=4
```

8.4.4 phm.txt

```
# plexi phantom
#-----------------------------
# cylinder
cylind  a=0.040 b=0.040 c=0.01 x=0.0 y=0.0   z=0.0 theta=0.0 phi=0.0 dens=1.19 mat=0
cylind  a=0.005 b=0.005 c=0.01 x=0.0 y=0.035 z=0.0 theta=0.0 phi=0.0 dens=-1.19 mat=0
```

```
#
material = 0 ../materials/plexi.mat
```

8.4.5 trj.txt

```
# trajectory file
# ----------------
projections = 280
#
# source pos. 0
  0.20000000  0.00000000  0.00000000
#
# detector pos. 0
  0.00000000 -1.00000000  0.00000000  0.00000000
  1.00000000  0.00000000  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
#
# transform
  0.99991660 -0.01291508  0.00000000  0.00000000
  0.01291508  0.99991660  0.00000000  0.00000000
  0.00000000  0.00000000  1.00000000  0.00000000
```

8.4.6 rebinning.m

```
function y = rebinning(x, L, dt, dfi, fb, rot, dtn, dfin, Mn, Nn)
% function y = rebin(x, L, dt, dfi, rot, dtn, dfin, Mn, Nn)
% L: distance to source
% dt: detector element size
% dfi: angle increment degrees
% fb: clockwise or counterclockwise rotation?
%     try either fb = 0 and fb = 1
% rot: rotation center pixel nr
% dtn: new detector element size
% dfin: new angle increment
% Mn: new nr of pixels per proj
% Nn: new nr of projs

[M,N] = size(x);    % M = nr of pixels per projection
                    % N = nr of projections

if nargin < 10
    Nn = N;
end
if nargin < 9
    Mn = M;
end
if nargin < 8
    dtn = dt;
end
if nargin < 7
    dfin = dfi;
end
if nargin < 6
    rot = 0;
end

gamma = 180*atan(M/2*dt/L)/pi;          % fanbeam angle

% in- and out-grid
%-----------------
if fb == 0
  [F,t]   = meshgrid((0:N-1)*dfi, (-(M-1)/2:(M-1)/2)*dt);
```

```
  [Fn,tn] = meshgrid((0:Nn-1)*dfin+gamma, (-(Mn-1)/2:(Mn-1)/2)*dtn);
else
  [F,t]   = meshgrid((0:N-1)*dfi, ((M-1)/2:-1:-(M-1)/2)*dt);
  [Fn,tn] = meshgrid((0:Nn-1)*dfin+gamma, ((Mn-1)/2:-1:-(Mn-1)/2)*dtn);
end

% rebin
%------
y=interp2(F, t+rot, x, Fn-180*asin(tn/L)/pi, tn./sqrt(1-tn.^2/L^2));

% find NaN:s and replace them with 0
%-----------------------------------
loc = find(isnan(y)==1);
y(loc) = 0;
```

8.4.7 parFB.m

```
function f = parFB(g, dtau, firstang)
%
% f = parFB(g, tau)
% parallel filtered backprojection method
%
% g:        projection indata
% dtau:     detector element size
% firstang: angle for first projection [rad]
%

if (nargin == 0)|(nargin>3)
  fprintf('1, 2 or 3 arguments is desired!\n');
  return;
end
if nargin == 1
  dtau = 1;
  firstang = 0;
end
if nargin == 2
  firstang = 0;
end

[M,N] = size(g);   % M = nr of pixels per projection,
                   % N = nr of projections

Sx = M;            % image x size
Sy = M;            % image y size
m0 = M;            % image x size in pixels
n0 = M;            % image y size in pixels
if mod(M,2)==0
  startx=-M/2;
  starty=-M/2;
else
  startx=-(M-1)/2;
  starty=-(M-1)/2;
end

% projection angles
dtheta = pi/N;
%theta = -pi/2-firstang:-dtheta:-3*pi/2+dtheta-firstang;
theta = 0+firstang:dtheta:pi-dtheta+firstang;

% detector array
if mod(M,2)==0
  det = (-M/2:M/2-1)';
```

```
else
  det = (-(M-1)/2:(M-1)/2)';
end

% rampfilter the projections
q = ramp(g);

% the grid for the reconstructed image
dx = Sx/m0;
dy = Sy/n0;
[X,Y] = meshgrid(startx:dx:startx+(m0-1)*dx, starty:dy:starty+(n0-1)*dy);
f = zeros(size(X));

% backprojection
for i=1:N
  c = cos(theta(i));
  s = sin(theta(i));
  f = f + interp2(ones(M,1)*(1:N), det*ones(1,N), q, i*ones(m0,m0), X*c + Y*s);
end

% find NaN:s and replace them with 0
loc = find(isnan(f)==1);
f(loc) = 0;

f = f*dtheta/dtau;
```

8.4.8 ramp.m

```
function q = ramp(g, dalfa)
%
% q = ramp(g, dalfa)
% rampfilter the columns of g
%

[M,N] = size(g);

M2 = 2^ceil(log2(M));
r = zeros(2*M2,1);
if nargin==1
  r(M2+1) = 1/4;
  r(2:2:end) = -1 ./(pi^2*(-M2+1:2:M2-1).^2);
end
if nargin==2
  r(M2+1) = 1/(8*dalfa^2);
  r(2:2:end) = -1 ./(2*pi^2*sin((-M2+1:2:M2-1)*dalfa).^2);
end
w = cos(pi*(-M2:M2)'/(2*M2)); % low pass weighting
R = (w(1:end-1).*fftshift(fft(ifftshift(r))))*ones(1,N);

q = zpadcol(g,2*M2);
q = real(fftshift(ifft(ifftshift (R.*fftshift(fft(ifftshift(q)))))));
q = zpadcol(q,M);

function y = zpadcol(x,m2)
%
% y = zpadcol(x,m2)
% pad or unpad
%

[m,n] = size(x);

if mod(m-m2,2) == 0
```

```
  offset = abs((m-m2)/2);
else
  offset = (abs(m-m2)+1)/2;
end

if m2>m
  y = [zeros(offset,n); x; zeros(m2-m-offset,n)];
else
  y = x(offset+1:offset+m2,:);
end
```

8.5 Spectrum and Material files

8.5.1 spm.txt

```
# Name: 45.0 kV
# Continuous spectrum
# Dimension: 42
# E [keV]       N
5.0000000e+00   0.0000000e+00
6.0000000e+00   0.0000000e+00
7.0000000e+00   0.0000000e+00
8.0000000e+00   0.0000000e+00
9.0000000e+00   3.1155915e-09
1.0000000e+01   4.0548318e-05
1.1000000e+01   6.8623098e-04
1.2000000e+01   1.8461402e-02
1.3000000e+01   2.3796571e-01
1.4000000e+01   1.7065633e+00
1.5000000e+01   7.5493386e+00
1.6000000e+01   2.3382856e+01
1.7000000e+01   5.6251281e+01
1.8000000e+01   1.1165767e+02
1.9000000e+01   1.9116994e+02
2.0000000e+01   2.9152420e+02
2.1000000e+01   4.0898978e+02
2.2000000e+01   5.3109138e+02
2.3000000e+01   6.5615568e+02
2.4000000e+01   7.7426157e+02
2.5000000e+01   8.7962451e+02
2.6000000e+01   9.7048676e+02
2.7000000e+01   1.0409779e+03
2.8000000e+01   1.0995464e+03
2.9000000e+01   1.1299357e+03
3.0000000e+01   1.1454061e+03
3.1000000e+01   1.1430603e+03
3.2000000e+01   1.1240242e+03
3.3000000e+01   1.0893693e+03
3.4000000e+01   1.0417206e+03
3.5000000e+01   9.8133317e+02
3.6000000e+01   9.1089186e+02
3.7000000e+01   8.3169753e+02
3.8000000e+01   7.4364895e+02
3.9000000e+01   6.4928316e+02
4.0000000e+01   5.4973676e+02
4.1000000e+01   4.4499243e+02
4.2000000e+01   3.3716408e+02
4.3000000e+01   2.2643508e+02
4.4000000e+01   1.1309668e+02
4.5000000e+01   4.5457152e+00
4.6000000e+01   0.0000000e+00
```

8.5.2 plex.txt

```
# Material: Plexiglas
# Density: 1.19 g/cm^3
# Dimension: 201
# Energy      Co_CS     In_CS     Ph_CS
# (MeV)       (cm^2/g)  (cm^2/g)  (cm^2/g)
 1.00000e-03  1.15e+00  1.42e-02  2.73e+03
 1.50000e-03  1.03e+00  2.83e-02  8.91e+02
 2.00000e-03  9.13e-01  4.36e-02  3.92e+02
 3.00000e-03  6.92e-01  7.21e-02  1.19e+02
 4.00000e-03  5.27e-01  9.46e-02  5.03e+01
 5.00000e-03  4.13e-01  1.11e-01  2.55e+01
 6.00000e-03  3.34e-01  1.23e-01  1.45e+01
 7.00000e-03  2.77e-01  1.32e-01  8.98e+00
 8.00000e-03  2.36e-01  1.39e-01  5.91e+00
 9.00000e-03  2.04e-01  1.45e-01  4.08e+00
 1.00000e-02  1.79e-01  1.50e-01  2.92e+00
 1.10000e-02  1.59e-01  1.54e-01  2.16e+00
 1.20000e-02  1.42e-01  1.57e-01  1.64e+00
 1.30000e-02  1.28e-01  1.60e-01  1.27e+00
 1.40000e-02  1.16e-01  1.62e-01  9.98e-01
 1.50000e-02  1.06e-01  1.65e-01  7.99e-01
 1.60000e-02  9.71e-02  1.67e-01  6.49e-01
 1.70000e-02  8.92e-02  1.69e-01  5.33e-01
 1.80000e-02  8.22e-02  1.70e-01  4.43e-01
 1.90000e-02  7.59e-02  1.71e-01  3.72e-01
 2.00000e-02  7.03e-02  1.73e-01  3.15e-01
 2.10000e-02  6.53e-02  1.74e-01  2.68e-01
 2.20000e-02  6.08e-02  1.74e-01  2.31e-01
 2.30000e-02  5.67e-02  1.75e-01  2.00e-01
 2.40000e-02  5.30e-02  1.76e-01  1.74e-01
 2.50000e-02  4.97e-02  1.76e-01  1.52e-01
 2.60000e-02  4.66e-02  1.77e-01  1.34e-01
 2.70000e-02  4.38e-02  1.77e-01  1.18e-01
 2.80000e-02  4.13e-02  1.77e-01  1.05e-01
 2.90000e-02  3.90e-02  1.78e-01  9.34e-02
 3.00000e-02  3.68e-02  1.78e-01  8.35e-02
 3.10000e-02  3.48e-02  1.78e-01  7.50e-02
 3.20000e-02  3.30e-02  1.78e-01  6.75e-02
 3.30000e-02  3.13e-02  1.78e-01  6.10e-02
 3.40000e-02  2.98e-02  1.78e-01  5.53e-02
 3.50000e-02  2.83e-02  1.78e-01  5.03e-02
 3.60000e-02  2.70e-02  1.78e-01  4.58e-02
 3.70000e-02  2.57e-02  1.78e-01  4.18e-02
 3.80000e-02  2.46e-02  1.78e-01  3.83e-02
 3.90000e-02  2.35e-02  1.77e-01  3.52e-02
 4.00000e-02  2.24e-02  1.77e-01  3.23e-02
 4.10000e-02  2.15e-02  1.77e-01  2.98e-02
 4.20000e-02  2.06e-02  1.77e-01  2.75e-02
 4.30000e-02  1.97e-02  1.77e-01  2.55e-02
 4.40000e-02  1.90e-02  1.76e-01  2.36e-02
 4.50000e-02  1.82e-02  1.76e-01  2.19e-02
 4.60000e-02  1.75e-02  1.76e-01  2.04e-02
 4.70000e-02  1.68e-02  1.76e-01  1.90e-02
 4.80000e-02  1.62e-02  1.75e-01  1.77e-02
 4.90000e-02  1.56e-02  1.75e-01  1.65e-02
 5.00000e-02  1.51e-02  1.75e-01  1.55e-02
 5.10000e-02  1.45e-02  1.74e-01  1.45e-02
 5.20000e-02  1.40e-02  1.74e-01  1.36e-02
 5.30000e-02  1.36e-02  1.74e-01  1.27e-02
 5.40000e-02  1.31e-02  1.73e-01  1.20e-02
```

```
5.50000e-02  1.27e-02  1.73e-01  1.13e-02
5.60000e-02  1.23e-02  1.73e-01  1.06e-02
5.70000e-02  1.19e-02  1.72e-01  1.00e-02
5.80000e-02  1.15e-02  1.72e-01  9.45e-03
5.90000e-02  1.11e-02  1.72e-01  8.93e-03
6.00000e-02  1.08e-02  1.71e-01  8.44e-03
6.10000e-02  1.05e-02  1.71e-01  7.99e-03
6.20000e-02  1.02e-02  1.71e-01  7.57e-03
6.30000e-02  9.87e-03  1.70e-01  7.18e-03
6.40000e-02  9.59e-03  1.70e-01  6.81e-03
6.50000e-02  9.32e-03  1.70e-01  6.47e-03
6.60000e-02  9.05e-03  1.69e-01  6.15e-03
6.70000e-02  8.80e-03  1.69e-01  5.85e-03
6.80000e-02  8.56e-03  1.68e-01  5.57e-03
6.90000e-02  8.33e-03  1.68e-01  5.31e-03
7.00000e-02  8.11e-03  1.68e-01  5.06e-03
7.10000e-02  7.90e-03  1.67e-01  4.83e-03
7.20000e-02  7.70e-03  1.67e-01  4.61e-03
7.30000e-02  7.50e-03  1.67e-01  4.40e-03
7.40000e-02  7.31e-03  1.66e-01  4.21e-03
7.50000e-02  7.13e-03  1.66e-01  4.02e-03
7.60000e-02  6.95e-03  1.65e-01  3.85e-03
7.70000e-02  6.78e-03  1.65e-01  3.69e-03
7.80000e-02  6.62e-03  1.65e-01  3.53e-03
7.90000e-02  6.46e-03  1.64e-01  3.39e-03
8.00000e-02  6.31e-03  1.64e-01  3.25e-03
8.10000e-02  6.16e-03  1.64e-01  3.12e-03
8.20000e-02  6.02e-03  1.63e-01  2.99e-03
8.30000e-02  5.88e-03  1.63e-01  2.87e-03
8.40000e-02  5.75e-03  1.63e-01  2.76e-03
8.50000e-02  5.62e-03  1.62e-01  2.66e-03
8.60000e-02  5.50e-03  1.62e-01  2.56e-03
8.70000e-02  5.38e-03  1.62e-01  2.46e-03
8.80000e-02  5.26e-03  1.61e-01  2.37e-03
8.90000e-02  5.15e-03  1.61e-01  2.28e-03
9.00000e-02  5.04e-03  1.60e-01  2.20e-03
9.10000e-02  4.94e-03  1.60e-01  2.12e-03
9.20000e-02  4.83e-03  1.60e-01  2.04e-03
9.30000e-02  4.74e-03  1.59e-01  1.97e-03
9.40000e-02  4.64e-03  1.59e-01  1.90e-03
9.50000e-02  4.55e-03  1.59e-01  1.84e-03
9.60000e-02  4.46e-03  1.58e-01  1.77e-03
9.70000e-02  4.37e-03  1.58e-01  1.72e-03
9.80000e-02  4.28e-03  1.58e-01  1.66e-03
9.90000e-02  4.20e-03  1.57e-01  1.60e-03
1.00000e-01  4.12e-03  1.57e-01  1.55e-03
1.01000e-01  4.04e-03  1.57e-01  1.50e-03
1.02000e-01  3.97e-03  1.56e-01  1.45e-03
1.03000e-01  3.89e-03  1.56e-01  1.41e-03
1.04000e-01  3.82e-03  1.56e-01  1.36e-03
1.05000e-01  3.75e-03  1.55e-01  1.32e-03
1.06000e-01  3.68e-03  1.55e-01  1.28e-03
1.07000e-01  3.62e-03  1.55e-01  1.24e-03
1.08000e-01  3.55e-03  1.54e-01  1.20e-03
1.09000e-01  3.49e-03  1.54e-01  1.17e-03
1.10000e-01  3.43e-03  1.54e-01  1.13e-03
1.11000e-01  3.37e-03  1.53e-01  1.10e-03
1.12000e-01  3.31e-03  1.53e-01  1.07e-03
1.13000e-01  3.25e-03  1.53e-01  1.04e-03
1.14000e-01  3.20e-03  1.53e-01  1.01e-03
1.15000e-01  3.14e-03  1.52e-01  9.77e-04
1.16000e-01  3.09e-03  1.52e-01  9.50e-04
```

```
1.17000e-01  3.04e-03  1.52e-01  9.23e-04
1.18000e-01  2.99e-03  1.51e-01  8.98e-04
1.19000e-01  2.94e-03  1.51e-01  8.73e-04
1.20000e-01  2.89e-03  1.51e-01  8.50e-04
1.21000e-01  2.85e-03  1.50e-01  8.27e-04
1.22000e-01  2.80e-03  1.50e-01  8.05e-04
1.23000e-01  2.76e-03  1.50e-01  7.83e-04
1.24000e-01  2.72e-03  1.49e-01  7.63e-04
1.25000e-01  2.67e-03  1.49e-01  7.43e-04
1.26000e-01  2.63e-03  1.49e-01  7.24e-04
1.27000e-01  2.59e-03  1.49e-01  7.05e-04
1.28000e-01  2.55e-03  1.48e-01  6.87e-04
1.29000e-01  2.51e-03  1.48e-01  6.70e-04
1.30000e-01  2.48e-03  1.48e-01  6.53e-04
1.31000e-01  2.44e-03  1.47e-01  6.37e-04
1.32000e-01  2.40e-03  1.47e-01  6.21e-04
1.33000e-01  2.37e-03  1.47e-01  6.06e-04
1.34000e-01  2.33e-03  1.47e-01  5.91e-04
1.35000e-01  2.30e-03  1.46e-01  5.77e-04
1.36000e-01  2.27e-03  1.46e-01  5.63e-04
1.37000e-01  2.24e-03  1.46e-01  5.50e-04
1.38000e-01  2.20e-03  1.45e-01  5.37e-04
1.39000e-01  2.17e-03  1.45e-01  5.24e-04
1.40000e-01  2.14e-03  1.45e-01  5.12e-04
1.41000e-01  2.11e-03  1.45e-01  5.00e-04
1.42000e-01  2.08e-03  1.44e-01  4.89e-04
1.43000e-01  2.06e-03  1.44e-01  4.78e-04
1.44000e-01  2.03e-03  1.44e-01  4.67e-04
1.45000e-01  2.00e-03  1.43e-01  4.57e-04
1.46000e-01  1.97e-03  1.43e-01  4.46e-04
1.47000e-01  1.95e-03  1.43e-01  4.37e-04
1.48000e-01  1.92e-03  1.43e-01  4.27e-04
1.49000e-01  1.90e-03  1.42e-01  4.18e-04
1.50000e-01  1.87e-03  1.42e-01  4.09e-04
1.51000e-01  1.85e-03  1.42e-01  4.00e-04
1.52000e-01  1.82e-03  1.42e-01  3.91e-04
1.53000e-01  1.80e-03  1.41e-01  3.83e-04
1.54000e-01  1.78e-03  1.41e-01  3.75e-04
1.55000e-01  1.76e-03  1.41e-01  3.67e-04
1.56000e-01  1.73e-03  1.41e-01  3.60e-04
1.57000e-01  1.71e-03  1.40e-01  3.52e-04
1.58000e-01  1.69e-03  1.40e-01  3.45e-04
1.59000e-01  1.67e-03  1.40e-01  3.38e-04
1.60000e-01  1.65e-03  1.40e-01  3.31e-04
1.61000e-01  1.63e-03  1.39e-01  3.25e-04
1.62000e-01  1.61e-03  1.39e-01  3.18e-04
1.63000e-01  1.59e-03  1.39e-01  3.12e-04
1.64000e-01  1.57e-03  1.39e-01  3.06e-04
1.65000e-01  1.55e-03  1.38e-01  3.00e-04
1.66000e-01  1.53e-03  1.38e-01  2.94e-04
1.67000e-01  1.52e-03  1.38e-01  2.88e-04
1.68000e-01  1.50e-03  1.38e-01  2.83e-04
1.69000e-01  1.48e-03  1.37e-01  2.77e-04
1.70000e-01  1.46e-03  1.37e-01  2.72e-04
1.71000e-01  1.45e-03  1.37e-01  2.67e-04
1.72000e-01  1.43e-03  1.37e-01  2.62e-04
1.73000e-01  1.41e-03  1.36e-01  2.57e-04
1.74000e-01  1.40e-03  1.36e-01  2.52e-04
1.75000e-01  1.38e-03  1.36e-01  2.48e-04
1.76000e-01  1.37e-03  1.36e-01  2.43e-04
1.77000e-01  1.35e-03  1.36e-01  2.39e-04
1.78000e-01  1.34e-03  1.35e-01  2.34e-04
```

```
1.79000e-01  1.32e-03  1.35e-01  2.30e-04
1.80000e-01  1.31e-03  1.35e-01  2.26e-04
1.81000e-01  1.29e-03  1.35e-01  2.22e-04
1.82000e-01  1.28e-03  1.34e-01  2.18e-04
1.83000e-01  1.27e-03  1.34e-01  2.14e-04
1.84000e-01  1.25e-03  1.34e-01  2.11e-04
1.85000e-01  1.24e-03  1.34e-01  2.07e-04
1.86000e-01  1.23e-03  1.34e-01  2.03e-04
1.87000e-01  1.21e-03  1.33e-01  2.00e-04
1.88000e-01  1.20e-03  1.33e-01  1.97e-04
1.89000e-01  1.19e-03  1.33e-01  1.93e-04
1.90000e-01  1.17e-03  1.33e-01  1.90e-04
1.91000e-01  1.16e-03  1.32e-01  1.87e-04
1.92000e-01  1.15e-03  1.32e-01  1.84e-04
1.93000e-01  1.14e-03  1.32e-01  1.81e-04
1.94000e-01  1.13e-03  1.32e-01  1.78e-04
1.95000e-01  1.12e-03  1.32e-01  1.75e-04
1.96000e-01  1.10e-03  1.31e-01  1.72e-04
1.97000e-01  1.09e-03  1.31e-01  1.69e-04
1.98000e-01  1.08e-03  1.31e-01  1.66e-04
1.99000e-01  1.07e-03  1.31e-01  1.64e-04
2.00000e-01  1.06e-03  1.31e-01  1.61e-04
```