Pub 214

RECLBL LIBRARY USERS MANUAL

Donner Algorithms
for
Reconstruction Tomography

R. H. Huesman
G. T. Gullberg
W. L. Greenberg
T. F. Budinger

October, 1977

Lawrence Berkeley Laboratory
University of California

This work was performed under the auspices of the
U. S. Department of Energy (Contract W-7405-ENG-48) for the
National Cancer Institute (Contract Y01-CB-50304).

Contents

1 INTRODUCTION

1.1 The RECLBL Library Package

The RECLBL Library is a package of computational subroutines that apply to the reconstruction of transverse sections from projection data. The subroutines are written in the FORTRAN programming language (ANSI standard) and have been tested on CDC 6400, 6600, and 7600 computers and on a PDP 11/45 system. The package applies to three-dimensional reconstruction problems that arise in the medical and physical sciences. The package includes programs for medical applications that can be used both for the determination of tissue attenuation coefficients using x-ray transmission data and for the determination of radionuclide concentration using data from nuclear medicine detectors. This manual contains descriptive material that gives the physical and mathematical bases for the algorithms, examples of the use of the algorithms, and FORTRAN listings of the algorithms.

1.2 The Reconstruction Problem

The reconstruction problem consists of generating a two-dimensional picture from its projections. The reconstructed picture consists of a quantitative set of numbers specifying source density or attenuation coefficient on a two-dimensional grid. The picture represents a transverse section of an object such as a human head shown in Figure 1.

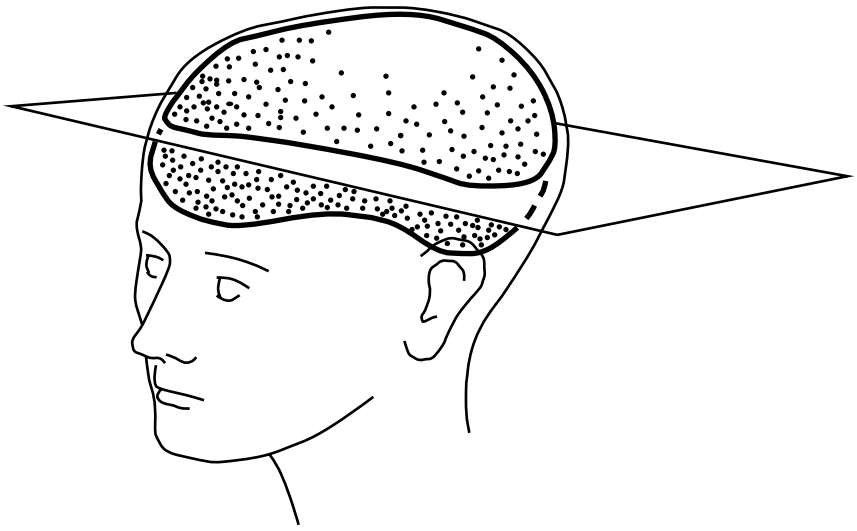

Figure 1: Concept of a transverse section.

The RECLBL Library applies to data that represent the projection of density along parallel or diverging sets of straight-line paths (rays) through an object. The algorithms transform one-dimensional projections from multiple angles around the object to a corresponding transverse section through the object. Three-dimensional information is obtained by stacking successive transverse sections.

1.3 Description of Library Contents

The reconstruction algorithms in the library are supplied as the following subroutines:

- `BJECT` — Simple back-projection.
- `BKFIL` — Back-projection of filtered projections (Fourier space).
- `CONGR` — Iterative least-squares minimization by the method of conjugate gradients.
- `CONVO` — Back-projection of convolved projections (configuration space).
- `ENTPY` — Iterative dual-space entropy maximization by the method of conjugate gradients.
- `FILBK` — Two-dimensional filtering of the simple back-projection (Fourier space).
- `GVERS` — Direct least-squares minimizations using the generalized inverse.
- `GRADY` — Iterative least-squares minimization by the method of steepest descent.
- `MARR` — Direct least-squares minimization using orthogonal polynomials on the unit circle.

These reconstruction algorithms execute with the following geometry options:

- Parallel-beam geometry with weighting by the area of the pixel intersected by the ray.
- Parallel-beam geometry assuming that all the activity is in the center of the pixel.
- Parallel-beam geometry with weighting by the length of the line that transverses the pixel.
- Fan-beam geometry with weighting by the area of the pixel intersected by the diverging ray.
- Fan-beam geometry assuming that all the activity is in the center of the pixel.

The methods of compensating for attenuation use attenuation factors calculated by the subroutines:

- `EVATN` — Incorporation of attenuation from a user provided array of attenuation coefficients.
- `EVATU` — Incorporation of constant attenuation coefficient within a convex boundary.

An overview of the library is shown in Figure 2. The figure gives the names of the essential library subroutines with which the user will need to be familiar.

Several reconstruction algorithms that this library does not contain (e.g., ART, the Algebraic Reconstruction Technique and SIRT, the Simultaneous Iterative Reconstruction Technique) may be found in G.T. Herman and S.W. Rowland, *SNARK77: A Programming System for the Reconstruction of Pictures from Projections,* State University of New York at Buffalo, Department of Computer Science, Technical Report No. 130 (1977).

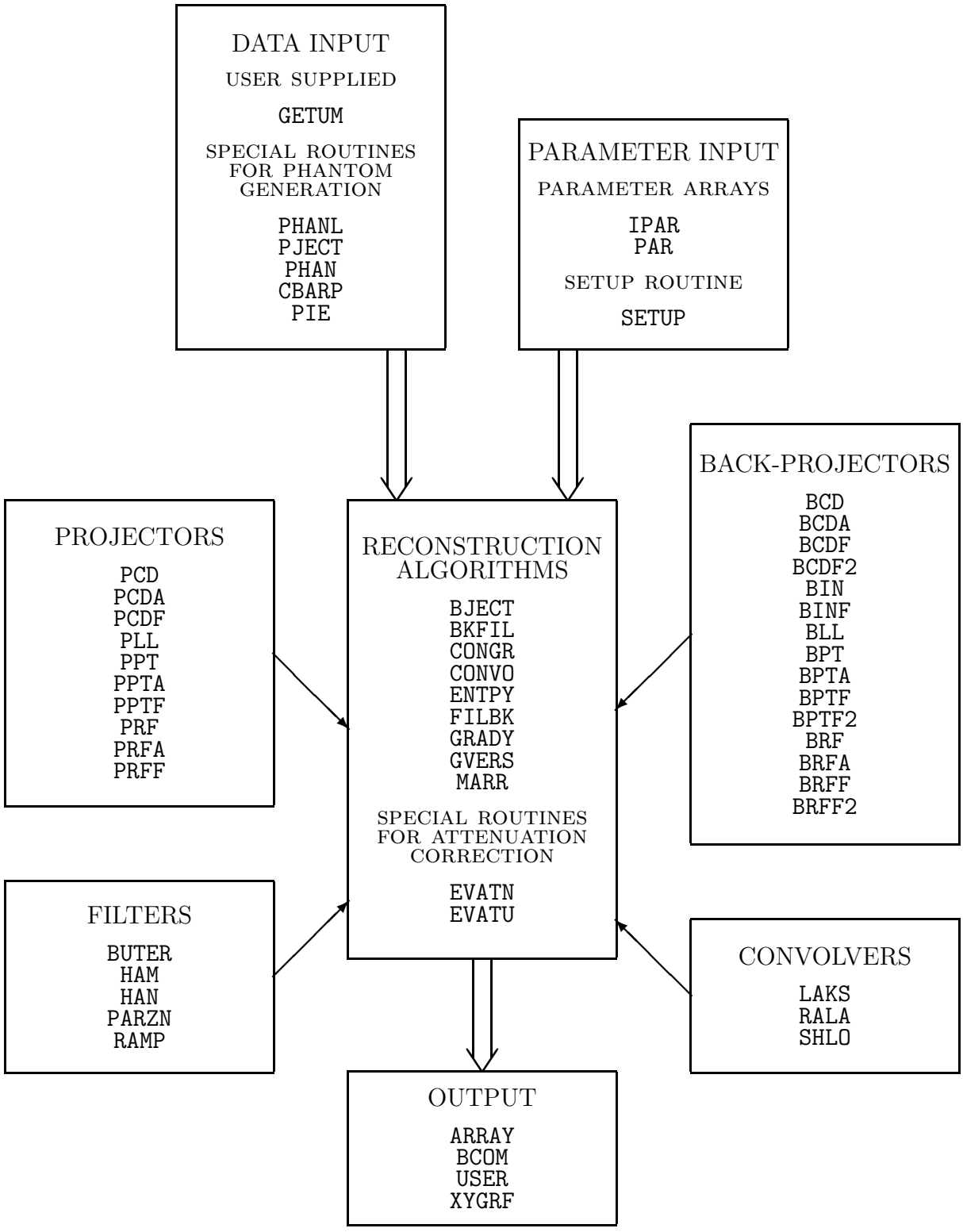

Figure 2: The RECLBL Library has 9 user called reconstruction subroutines. Projectors, back-projectors, convolvers and filters are passed to the reconstruction algorithms as external subroutines. The data are input using the subroutine `GETUM`, and the parameter arrays `IPAR` and `PAR` are input using the subroutine `SETUP`. The reconstructions may be displayed using special output subroutines.

1.4 Distribution of Documentation and Programs

Subscribers to the RECLBL Library will receive the Users Manual and the library source material, which is distributed on magnetic tape. The magnetic tape can be either 7 or 9 track, depending on the user's hardware requirements. A charge of $20.00 will be made for each magnetic tape provided to cover the cost of the tape and mailing. The user will receive library revisions and additions after they have been tested and implemented.

The last page of this manual contains an order blank for a magnetic tape containing the source material of the RCLBL Library. The contents and format of the magnetic tape are given in Section 2.5.

Corrections or comments on the RECLBL Library or this manual should be sent to:

Research Medicine Group
Donner Laboratory
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720
Attention: RECLBL Library

2 LIBRARY CHARACTERISTICS

2.1 Philosophy

The RECLBL Library is a collection of subroutines. The user is expected to have a working knowledge of the FORTRAN computer language. He must write a main program that calls various subroutines of the RECLBL Library. These include setup, data input, and display routines, as well as the major routines that execute the reconstruction algorithms.

The user must also be familiar with the names of another class of library subroutines that are used as external parameters of major reconstruction algorithms. These routines specify that type of weighting factor and the convolution or filter function to be used. All of the subroutine names that the user might need to use are shown in Figure 2.

The structure of the RECLBL Library provides the user with a great deal of flexibility while requiring a minimum knowledge of computer programming.

2.2 Operating Environment

The programs have been designed to accommodate both small and large computer implementation. The RECLBL Library has been written and tested on CDC 6400, 6600, and 7600 computers. Parts of the library have been put into operation on PDP 11/45 and HP2100 systems. Because the HP2100 does not allow labeled common, this package must be modified for full implementation on that system.

The library has been designed to be used in an operating system that has the ability to load into memory only those routines that are necessary to execute the user's code. Because of the structure of the RECLBL Library, a minimum amount of computer memory is required.

2.3 Coding Conventions

The subroutines of the RECLBL Library were coded in the FORTRAN computer language using the guidelines of:

American National Standard FORTRAN
American National Standards X3.9-1966
United States of America Standards Institute
New York, 1966

Clarification of American National Standards X3.9-1966 was prepared by a Subcommittee of the American Standards Committee X3, Computers and Information Processing, and published in the Communications of the Association for Computing Machinery:

Clarification of Fortran Standards–Initial Progress, Comm. ACM, Vol. 12, No. 5, May 1969, pp. 289-294.
Clarification of Fortran Standards–Second Report, Comm. ACM, Vol. 14, No. 10, October 1971, pp. 628-642.

Table 1: Common block and subroutine names used by the RECLBL Library.

Common Blocks			
//	/FANCOM/	/OUTCOM/	/STRCOM/
/ATNCOM/	/FILCOM/	/PHNCOM/	/TRGCOM/
/CNVCOM/	/GNVCOM/	/PRTCOM/	/WRKCOM/
/DATCOM/	/ITRCOM/	/PTRCOM/	
/ENTCOM/	/MARCOM/	/RAYCOM/	
Subroutines			
ARRAY	BUTER	GRADY	PPTF
ATENF	CBARP	GVERS	PRF
BCD	CISQ	HAM	PRFA
BCDA	CONGR	HAN	PRFF
BCDF	CONVO	IOCTL	RADAL
BCDF2	DOT	LAKS	RALA
BCOM	DULFC	LGTXT	RAMP
BIN	EMESG	MARR	RAYST
BINF	ENTPY	MEMST	RCHEK
BJECT	EVATN	PARZN	SETIT
BKFIL	EVATU	PCD	SETUP
BLL	FFTC	PCDA	SHLO
BPT	FFTR	PCDF	SQINT
BPTA	FFTR2	PHAN	SRCH
BPTF	FILBK	PHANL	STATN
BPTF2	FMCG	PIE	STPTR
BRF	FTATN	PJECT	XYGRF
BRFA	GETDE	PLL	ZERO
BRFF	GETDM	PPT	
BRFF2	GINV	PPTA	

2.4 User Coding Restrictions

Within the RECLBL Library there are various common blocks and subroutines, with which the user need not be familiar, but whose names are a possible source of conflict with user-created common blocks and subroutines. In order that the library as a whole operate correctly, the user must not use the common block and subroutine names listed in Table 1. Note that blank common (//) is one of the common blocks used by the library.

2.5 Magnetic Tape Structure

The following describes the file structure of the magnetic tapes containing the source code of the RECLBL Library routines. The first file of the tape is a label and contains information such as the version number, the date of the last revision, etc. The subsequent 80 files (2-81) contain the routines that make up the library (cf. Section 10.2). The last 18 files (82-99)

contain examples (cf. Section 9). Two file marks follow the last example. The format of the tape depends on whether the tape is 7 or 9 track. Each record of each file on the magnetic tape contains an 80-character card image. Each character is represented by either 6 or 8 bits, depending on whether the tape is 7 or 9 tracks, respectively.

The 7-track magnetic tapes are written in EXTERNAL BCD format with 80 characters per record. This is an industry standard, even parity format. The 6-bit octal EXTERNAL BCD code for the standard FORTRAN character set is shown in Table 2.

The 9-track magnetic tapes are written in either ASCII or EBCDIC format. These are both industry standard, odd parity formats. The 7-bit octal ASCII code and the 8-bit octal EBCDIC code for the standard FORTRAN character set are shown in Table 2. Because of tape writing restrictions at the Lawrence Berkeley Laboratory Computer Center, the 9-track magnetic tapes contain 90 characters per record. The first 80 characters contain the 80-character card image, and the last 10 characters contain blank fill.

Table 2: EXTERNAL BCD, ASCII and EBCDID octal codes for the standard FORTRAN character set.

Standard FORTRAN Character	6-Bit EXTERNAL BCD Octal Code	7-Bit ASCII Octal Code	8-Bit EBCDIC Octal Code
A	61	101	301
B	62	102	302
C	63	103	303
D	64	104	304
E	65	105	305
F	66	106	306
G	67	107	307
H	70	110	310
I	71	111	311
J	41	112	321
K	42	113	322
L	43	114	323
M	44	115	324
N	45	116	325
O	46	117	326
P	47	120	327
Q	50	121	330
R	51	122	331
S	22	123	342
T	23	124	343
U	24	125	344
V	25	126	345
W	26	127	346
X	27	130	347
Y	30	131	350
Z	31	132	351
0	12	060	360
1	01	061	361
2	02	062	362
3	03	063	363
4	04	064	364
5	05	065	365
6	06	066	366
7	07	067	367
8	10	070	370
9	11	071	371
+	60	053	116
-	40	055	140
	54	052	134
/	21	057	141
(	34	050	115
)	74	051	135
$	53	044	133
+	13	075	176
blank	20	040	100
,	33	054	153
.	73	056	113

3 USER PROGRAM STRUCTURE

3.1 General Description

Since the RECLBL Library is a collection of subroutines, the user must provide a program that performs such functions as: set parameters that define the geometry as well as determine control operations within the library subroutines, call reconstruction subroutines of the library, call display routines of the library, and save results if desired. In addition, the user must provide a subroutine `GETUM` for data input. A skeleton program that outlines the recommended structure of a main program and a data input routine (`GETUM`) is shown in Figure 3.

The variables `LUNOUT` and `I80132` of `COMMON/OUTCOM/` must be set by the user prior to the execution of any of the library subroutines.

`LUNOUT` is the logical unit number of the print file. The library communicates with the user via this file.

`I80132` is a flag indicating whether to print 80 or 132 characters per line on `LUNOUT`. `I80132`=0 indicates 80 characters per line, otherwise the library prints 132 characters per line.

Before any of the reconstruction algorithms are called, the user must call the subroutine `SETUP`. The arguments of `SETUP` include control options that describe the geometry as well as some computer operation parameters. Subroutine `SETUP` is called as follows:

```
CALL SETUP (IPAR,PAR,ANG)
```

where

`IPAR` is an array of integer parameters,

`PAR` is an array of floating point parameters, and

`ANG` is an array of projection angles and is needed only when `MODANG`=`IPAR(4)` is equal to zero or one.

Parameters of the `IPAR` and `PAR` arrays are described in Sections 3.2 and 3.3 below. *Throughout this manual they will be referred to by the variable names given in the* `EQUIVALENCE` *statement of Figure 3.*

A description of the input data format for the user provided subroutine `GETUM` (cf. Figure 3) is given in Section 3.4.

3.2 Geometry Parameters

Of the 15 parameters of the `IPAR` and `PAR` arrays, 10 describe aspects of the geometry to be used in the reconstruction. In conjunction with the definitions given below the reader is referred to Figures 4-7.

Program card (machine/compiler dependent)	`PROGRAM MAIN ( )`
Reconstruction array and uncertainites	`DIMENSION X("NDIMU","NDIMU"),E("NDIMU","NDIMU")`
Array of projection angles	`DIMENSION ANG("NANG")`
Working space in blank common (see Section 3.3)	`COMMON WORK(2000)`
Output file and flag for number of characters per line (see Section 3.1)	`COMMON/OUTCOM/LUNOUT,I80132`

Integer and real parameter arrays (see Sections 3.2 and 3.3)	`DIMENSION IPAR(12),PAR(3)`
	`EQUIVALENCE (NDIMU ,IPAR( 1)),(ICIR ,IPAR( 2)),(IGEOM ,IPAR( 3)),`
	`1 (NANG ,IPAR( 4)),(MODANG,IPAR( 5)),(KDIMU ,IPAR( 6)),`
	`2 (IMIT ,IPAR( 7)),(NWORK ,IPAR( 8)),(NFLOAT,IPAR( 9)),`
	`3 (ISTORE,IPAR(10)),(IPRINT,IPAR(11)),(LUNATN,IPAR(12)),`
	`4 (PWID , PAR( 1)),(AXISU , PAR( 2)),(RFAN , PAR( 3))`

Back-projection and convolution subroutines that are passed as externals (see Section 5.2)	`EXTERNAL BCK,CNV`
Output file (see Section 3.1) Output line length flag (see Section 3.1)	`LUNOUT= ....` `I80132= ....`

Input parameters (see Sections 3.2 and 3.3)	`NDIMU= ....`
	`ICIR= ....`
	`IGEOM= ....`
	`NANG= ....`
	`MODANG= ....`
	`KDIMU= ....`
	`IMIT= ....`
	`NWORK= ....`
	`NFLOAT= ....`
	`ISTORE= ....`
	`IPRINT= ....`
	`LUNATN= ....`
	`PWID= ....`
	`AXISU= ....`
	`RFAN= ....`

	`CALL SETUP(IPAR,PAR,ANG)`
Reconstruct the array `X` using the convolution algorithm (see Section 5 for a description of all the reconstruction algorithms	`CALL CONVO(X,E,CNV,BCK,1)`
Displays the reconstructed array `X` (see Section 6.3)	`CALL ARRAY(X,NDIMU)`
	`END`

Data input routine (see Section 3.4)
`M` is the angle index, `DATA` is the projection data array, and `ERR` is an array of projection errors.
	`SUBROUTINE GETUM(M,DATA,ERR)`
	(Here is where data and errors for the $\mathtt{M}^{th}$ projection are supplied by the user; see Section 3.4 and examples in Section 9.)
	`RETURN`
	`END`

Figure 3: Skeleton program to show recommended user program structure.

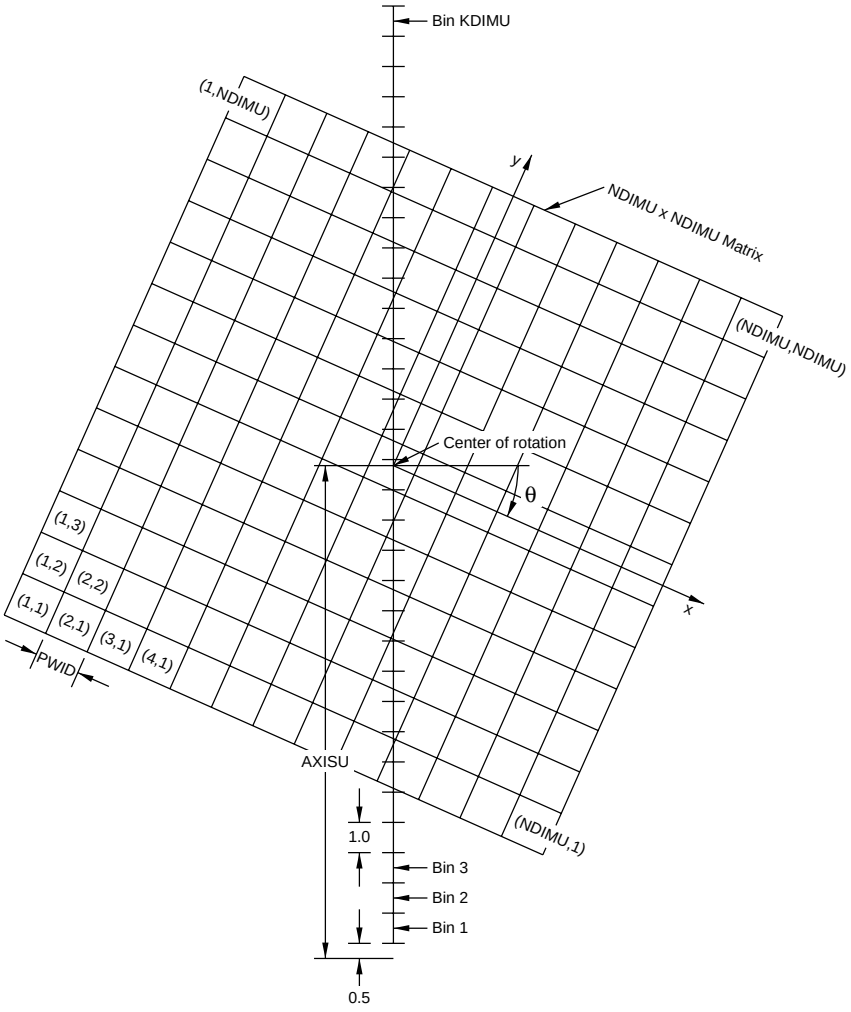

Figure 4: Parallel-beam geometry for data collected at projection angle θ. `NDIMU` may be either even or odd and the center of rotation is at the exact center of the `NDIMU` $\times$ `NDIMU` reconstruction array. The indices of the array are denoted by (`I`,`J`), each representing a pixel with linear dimension `PWID`, where projection bins are defined to have unit width. `AXISU` is 0.5 greater than the distance from the center of rotation to the lower bin edge of the first of `KDIMU` projection bins.

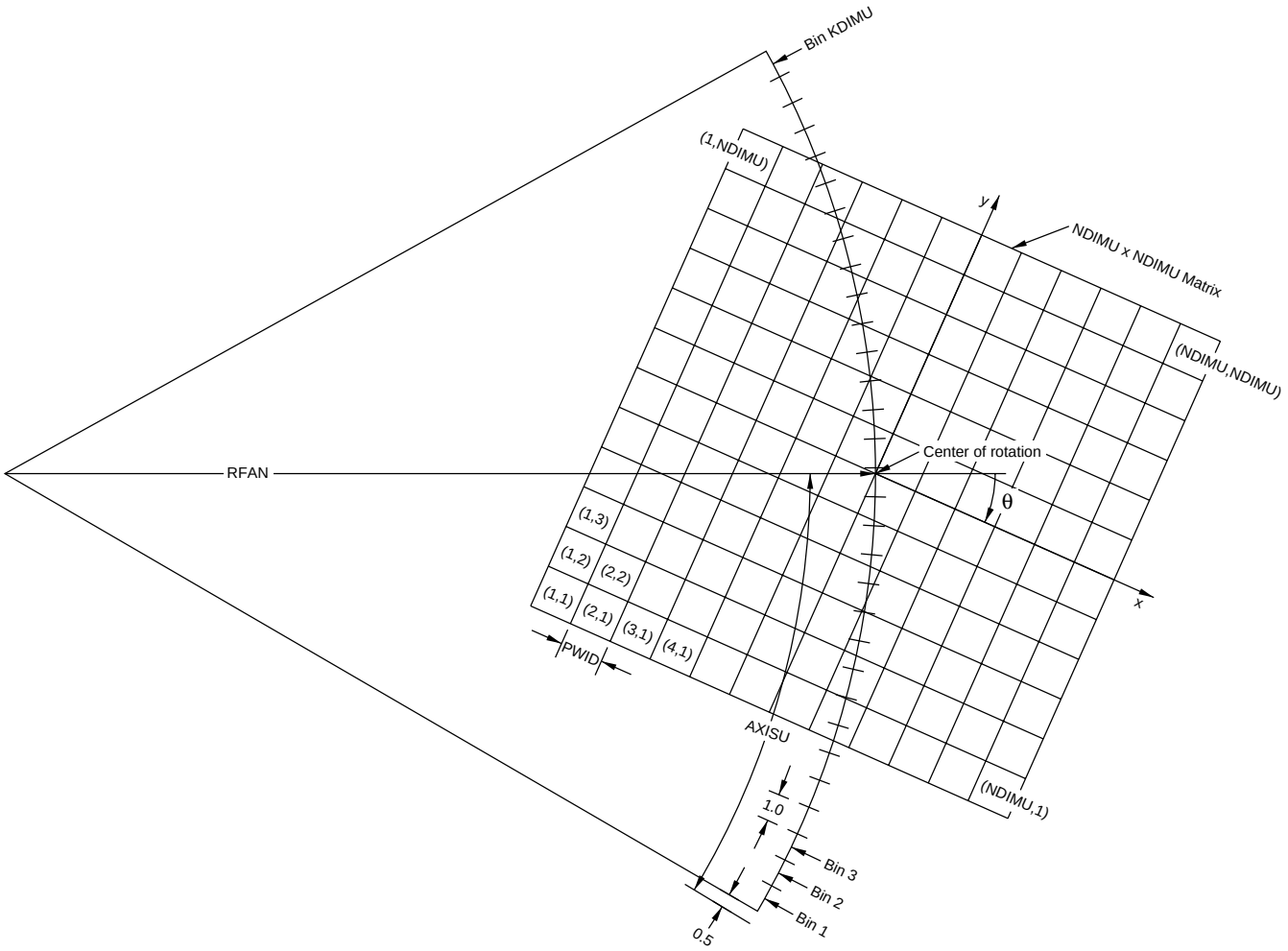

Figure 5: Fan-beam geometry for data collected at projection angle θ using a curved detector. `NDIMU` may be either even or odd and the center of rotation is at the exact center of the `NDIMU` $\times$ `NDIMU` reconstruction array. The indices of the array are denoted by (`I`,`J`), each representing a pixel with linear dimension `PWID`. The diverging projection bins are defined to have unit width measured at the center of rotation, a distance `RFAN` from the vertex of the fan. `AXISU` is 0.5 greater than the distance from the center of rotation to the lower bin edge of the first of `KDIMU` projection bins.

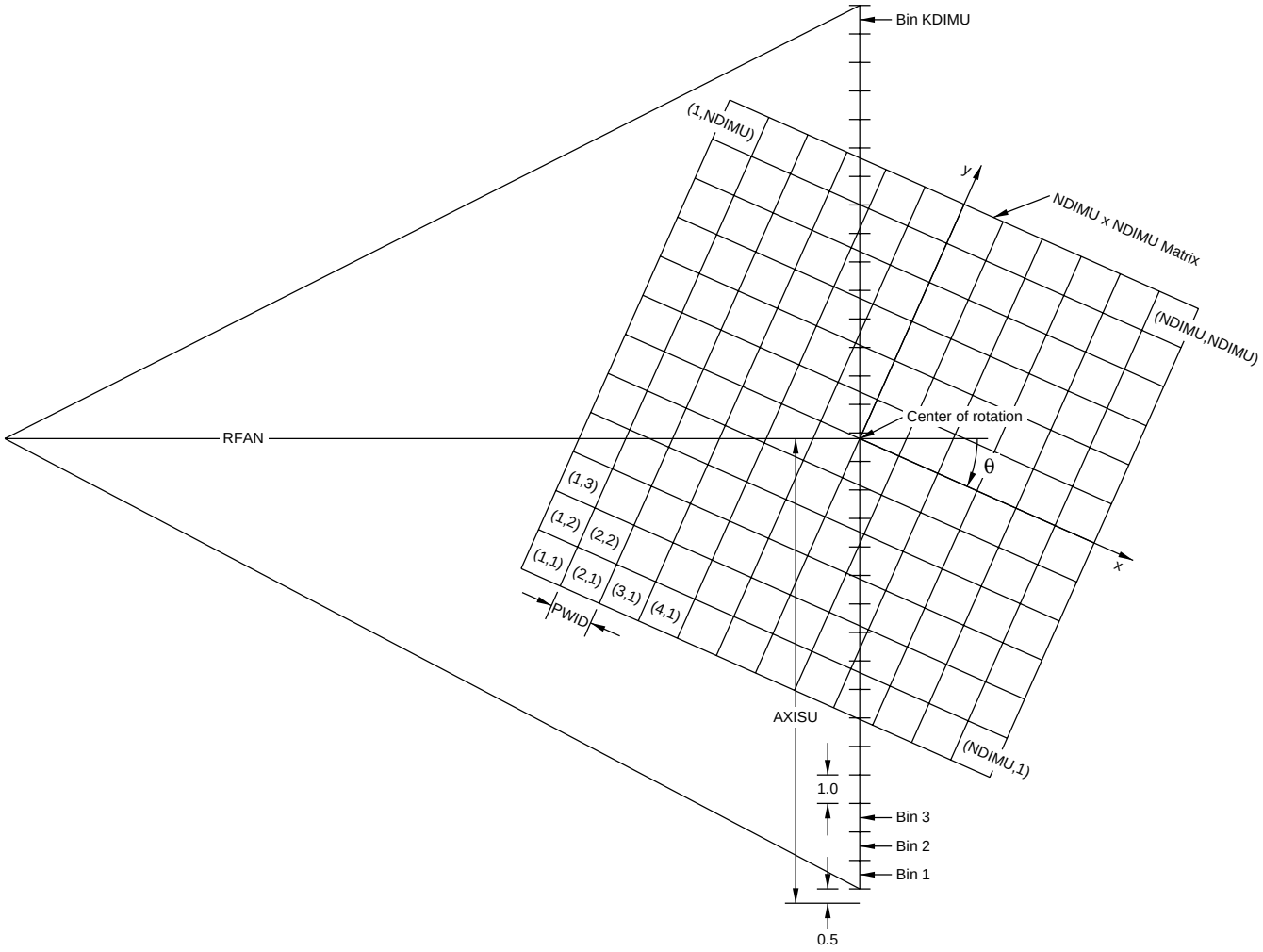

Figure 6: Fan-beam geometry for data collected at projection angle θ using a flat detector. `NDIMU` may be either even or odd and the center of rotation is at the exact center of the `NDIMU` $\times$ `NDIMU` reconstruction array. The indices of the array are denoted by (`I`,`J`), each representing a pixel with linear dimension `PWID`. The diverging projection bins are defined to have unit width measured at the center of rotation, a distance `RFAN` from the vertex of the fan. `AXISU` is 0.5 greater than the distance from the center of rotation to the lower bin edge of the first of `KDIMU` projection bins.

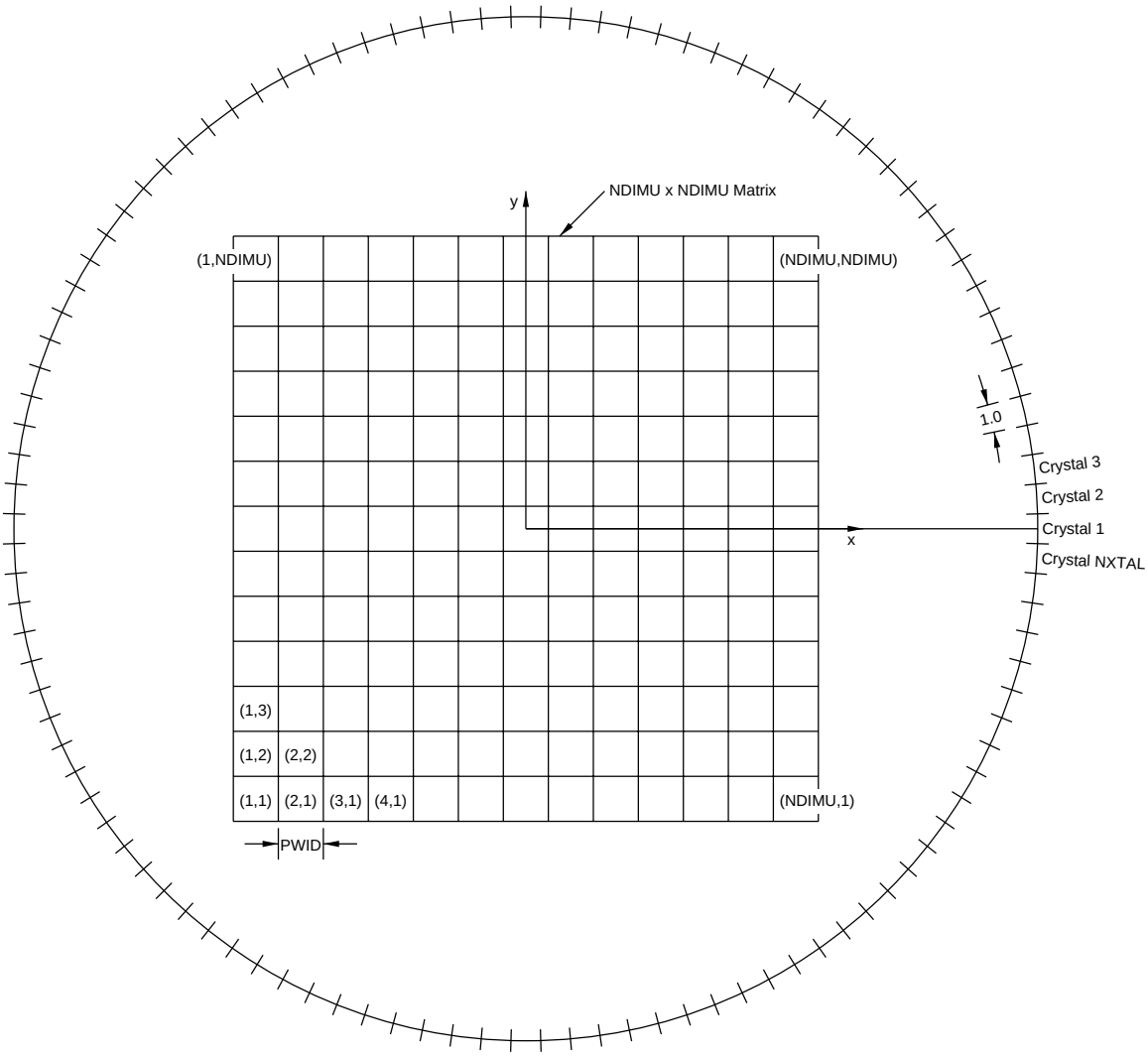

Figure 7: Geometry for data collected using a ring of `NXTAL` = `NANG` detectors. `NDIMU` may be either even or odd and the center of the ring is at the exact center of the `NDIMU` × `NDIMU` reconstruction array. The indices of the array are denoted by (`I`,`J`), each representing a pixel with linear dimension `PWID`, where the center-to-center distance between adjacent detectors is defined to be unity.

`NDIMU` is the linear dimension of the reconstruction array, i.e., a reconstruction algorithm will return an array of `NDIMU`×`NDIMU` values that represent reconstructed intensities on an `NDIMU`×`NDIMU` grid.

`ICIR` is a flag indicating whether the reconstructed intensities are to be calculated for the entire `NDIMU`×`NDIMU` square grid or only for points lying within a circle inscribed in the square. A 25% reduction in computer time can be expected for certain algorithms if only the inscribed circle is used. To reconstruct on a circle, set `ICIR`=0; otherwise the entire square will be reconstructed.

`IGEOM` is a flat indicating the type of geometry to be used in the reconstruction. `IGEOM`=0, 1, 2, 3 indicates parallel-beam, fan-beam (curved detector), fan-beam (flat detector), and ring geometry, respectively. These types of geometry are shown in Figures 4-7.

`NANG` is the number of projection angles to be used for the reconstruction in parallel-beam or fan-beam geometries (`IGEOM`=0, 1, 2). For the ring geometry (`IGEOM`=3), `NANG` is the number of detectors around the circle (an even number). Therefore, the exact meaning of `NANG` depends on `IGEOM`.

`MODANG` is a mode flag for the input of projection angle values. For `MODANG`=0 or `MODANG`=1, the user supplies projection angles in the array `ANG` in degrees or radians, respectively. For `MODANG`=2 or `MODANG`=4, `SETUP` generates `NANG` projection angles equally spaced between 0 and π starting with 0.5π/`NANG` or 0, respectively. For `MODANG`=3 or `MODANG`=5, `SETUP` generates `NANG` projection angles equally spaced between 0 and 2π starting with π/`NANG` or 0, respectively. For `MODANG` between -2 and -5, `SETUP` generates the same angles as for `MODANG` between 2 and 5, respectively, but in reverse order.

`KDIMU` is the dimension of the user's projection array for all geometries except ring geometry (`IGEOM`=3). The user is expected to input a projection data array of length `KDIMU` for each projection angle using his own subroutine `GETUM`. Subroutine `GETUM` is described in Section 3.4 below.

`IMIT` is a flag indicating whether the reconstruction is of emission or transmission data. For emission data the reconstructed intensities will be in terms of events per pixel, i.e., for unattenuated data the sum of all reconstructed intensities should equal the sum of the projected data (for all angles). For transmission data the reconstructed intensities will be attenuation coefficients in units of inverse pixel width. To reconstruct emission data, set `IMIT`=0; if `IMIT`$\neq$0 the library assumes transmission data.

`PWID` is the distance between neighboring reconstruction grid points relative to the projection bin width. Projection bin width for fan-beam geometries is described in the definition `RFAN` below. Projection bin width for the ring geometry (`IGEOM`=3) is defined as the distance between the `NANG` equally spaced points on the circle.

`AXISU` is the location within the projection array (of length `KDIMU`) where the rotation axis is projected. The rotation axis is defined to be in the exact center of the `NDIMU`×`NDIMU` reconstruction grid. `AXISU` is assumed to be the same for every projection angle but

need not be integer valued. AXISU will be an integer equal to the number of the projection bin into which the rotation axis projects if it projects into the exact center of a bin.

RFAN is the distance between the rotation axis and the origin of the fan for fan-beam geometries (IGEOM=1,2). RFAN is measured in terms of projection bin width, which is the distance between neighboring projection bins as they cross the rotation axis.

3.3 Computer Operation Parameters

The remaining five parameters of the IPAR array relate to the internal operations of the RECLBL subroutine package.

NWORK is the number of floating point words that have been set aside by the user in blank common (//). It must be set by the user to the dimension of WORK, the array in blank common. This space will be used as a working area by the library, and is not available to the user. See also the description of ISTORE below.

NFLOAT is the number of computer words required for the storage of a single floating point variable. (It is assumed that integer variables require one memory location.) NFLOAT is needed for the management of the working area in blank common.

ISTORE is a flat indicating whether to actually execute library code or to only estimate the size of blank common needed in order to accomplish the reconstruction. The amount of blank common needed is printed on the logical unit given by LUNOUT. To perform a reconstruction set ISTORE=0, otherwise only a storage size test is performed. In case the user has set NWORK too small, reconstruction will halt and from that point on only a storage size test will continue.

IPRINT is a flag that indicates the various print options for output onto the logical unit given by LUNOUT. The six low-order bits of IPRINT determine the following options:

- bit 0 Print the number of floating point variables in blank common whenever changed.
- bit 1 Print the projection data and uncertainties.
- bit 2 Print the IPAR and PAR arrays when SETUP is called.
- bit 3 Print the filter function for the convolution and filter routines.
- bit 4 Print the values of the Lagrange multipliers and gradient of the function that is optimized in the maximum entropy reconstruction.
- bit 5 Print pointers in blank common whenever changed (debug).

LUNATN is the logical unit number of a scratch file that is required when compensating for attenuation.

3.4 Data Input

Projection data (and possibly their uncertainties) must be supplied to the RECLBL Library subroutines by the user-coded subroutine `GETUM`. The arguments to this subroutine are:

```
SUBROUTINE GETUM (M,DATA,ERR)
```

where

`DATA` is an array of projection data to be returned by the user, and

`ERR` is an array of uncertainties of the respective values returned in `DATA`. The uncertainties, `ERR`, need only be supplied if the user desires to take account of uncertainties when using the algorithms `CONGR`, `GRADY`, or `GVERS`; or if the user desires that the resulting uncertainties of the reconstruction be calculated when using the algorithms `CONVO` or `GVERS`.

For parallel- or fan-beam geometries (`IGEOM`=0,1,2) `M` is the angle index number for which `GETUM` is to return projection data. `KDIMU` values are to be returned in the `DATA` array corresponding to the `KDIMU` projection bins for the $\texttt{M}^{th}$ angle as shown in Figures 4-6.

For ring geometry (`IGEOM`=3) there are `NANG`(`NANG`-1)/2 possible pairs of detectors and hence `NANG`(`NANG`-1)/2 different projection data values. `M` is an index that indicates the detector separation for the set of data values `GETUM` must supply. `M` will vary from 1 to `NANG`/2, and `GETUM` must supply `NANG` values in the `DATA` array for `M` between 1 and `NANG`/2-1. For `M`=`NANG`/2, `GETUM` need only supply `NANG`/2 values since for this case, the detectors are diametrically opposed. If `K` is an index that indicates the order in which the `NANG` (or `NANG`/2) values are to be returned in the `DATA` array, the projection data are from between the $\texttt{K}^{th}$ and $(\texttt{K+M})^{th}$ detectors.

4 PROJECTION AND BACK-PROJECTION ROUTINES

4.1 Models of Intensity Distribution

In order to perform the reconstruction of a transverse section from its projections on a digital computer, it is necessary to characterize the two-dimensional intensity distribution by a finite number of parameters. In the RECLBL Library the transverse section is divided into NDIMU^2 small square areas (pixels), and the reconstruction results in an array of intensity values (one for each pixel). These values may represent either the intensity at the center of the pixel or the total (or average) intensity within the pixel.

The reconstruction algorithms of the RECLBL Library may be divided into two categories: those for which there is an implied distribution of intensity within each pixel, and those which are analytic by nature and require no such assumption. In the first category are the iterative methods (CONGR, GRADY), the maximum entropy method (ENTPY), and the generalized inverse method (GVERS). In all of these the reconstructed intensities are chosen such that when projected they are, in some sense, close to the user-supplied projection data. In order to perform this projection, the model of intensity distribution (distribution within each pixel) is required.

A natural choice for the model is that intensity within each pixel is uniformly distributed. This is the most realistic model in the library. The projection of one such pixel has a trapezoidal shape for all angles except multiples of $\pi/4$. Degenerate cases exist at multiples of $\pi/2$ (square shape) and odd multiples of $\pi/4$ (triangular shape).

A good approximation to the uniform square model is what has been termed the concave disk model. For this model the projection of a single pixel has a square shape independent of angle. This is particularly good approximation when the pixel size is the same as the projection bin size (PWID=1). The third model of intensity distribution assumes that all intensity within a pixel is concentrated at its center, and is called the delta function model.

4.2 Relationship of Models and Geometry

Within the RECLBL Library the projection of a two-dimensional intensity distribution may be over infinitesimally narrow paths (line integrals) or over finite width paths (ray sums) where a single projection bin extends in width to both of its neighbors (without overlap). This section describes the relationship between the models of Section 4.1 and the various geometry options of Section 3.2. Note that the projection operation is required only for the model-dependent algorithms: CONGR, GRADY, ENTPY, and GVERS.

Projection routines are intended to mimic the data-taking process under the assumptions of the model of intensity distribution within each pixel (excluding statistical fluctuations). In the RECLBL Library, projections may be performed using any of the four geometry options, which are described in Section 3.2 and illustrated in Figures 4-7.

For the model of uniform intensity within each pixel, the projections may be performed either as line integrals or as ray sums. For parallel-beam geometry (IGEOM=0) the corresponding routines are PLL (line length) and PRF (ray factors), respectively. For fan-beam

geometries (IGEOM=1,2) only ray sum projection exists at this time, and the routine is PRFF (ray factors, fan).

For the concave disk model and the delta function model only ray sum projection routines are necessary. For parallel-beam geometry the corresponding routines are PCD (concave disk) and PPT (point), respectively. For fan-beam geometries they are PCDF (concave disk, fan) and PPTF (point, fan).

The model-dependent algorithms must also perform a back-projection, that is, the adjoint or transpose of the projection operation. Thus the library contains the seven back-projection routines corresponding to the projection routines described above. In the names of these routines the first letter (P) has been replaced with the letter B: BLL, BRF, BRFF, BCD, BPT, BCDF and BPTF.

4.3 Incorporation of Attenuation

For x-ray imaging the transmitted beam intensity $I(\xi,\theta)$ is equal to

$$I(\xi,\theta) = I_o(\xi,\theta)\ e^{-\int\int \mu(x,y)K(\xi,\theta,x,y)dxdy}\ , \tag{4.3.1}$$

where $\mu(x,y)$ is the distribution of attenuation coefficients, I_o is the incident-beam intensity, and $K(\xi,\theta,x,y)$ is a function whose distribution corresponds to either parallel-beam, fan-beam, or ring geometry. The projection $p(\xi,\theta)$ is thus equal to

$$p(\xi,\theta) = -\log\left[\frac{I(\xi,\theta)}{I_o(\xi,\theta)}\right] = \int\int \mu(x,y)K(\xi,\theta,x,y)dxdy\ . \tag{4.3.2}$$

This equation does not represent the line integrals measured in emission tomography.

The projection $p_{\gamma\gamma}(\xi,\theta)$ for positron annihilation coincidence imaging is defined by the integral equation

$$p_{\gamma\gamma}(\xi,\theta) = e^{-\int\int \mu(x,y)K(\xi,\theta,x,y)dxdy}\ \int\int \rho(x,y)K(\xi,\theta,x,y)dxdy\ , \tag{4.3.3}$$

where $\rho(x,y)$ is the concentration of positron emitter. Therefore, the projection is the line integral of the positron concentration distribution multiplied by an exponential attenuation factor determined from the line integral of attenuation coefficients over the total ray path. The projection data that should be supplied to a RECLBL reconstruction algorithm are given by

$$p(\xi,\theta) = e^{\int\int \mu(x,y)K(\xi,\theta,x,y)dxdy}\ p_{\gamma\gamma}(\xi,\theta) = \int\int \rho(x,y)K(\xi,\theta,x,y)dxdy\ . \tag{4.3.4}$$

Thus, the user must modify the observed data $p_{\gamma\gamma}(\xi,\theta)$ by the appropriate attenuation factor.

The projection $p_\gamma(\xi,\theta)$ for single photon imaging in emission tomography is defined by the integral equation

$$p_\gamma(\xi,\theta) = \int\int \rho(x,y)e^{-\int_x^{\text{detector}}\int_y \mu(x',y')K(\xi,\theta,x',y')dx'dy'}\ K(\xi,\theta,x,y)dxdy\ . \tag{4.3.5}$$

Note the difference between equations (4.3.4) and (4.3.5). A single photon projection is the summation of isotope concentration at points (x,y) modified by an exponential e^{-Z} where

Z is the line integral of attenuation coefficients from the point (x, y) to the detector. Thus, the attenuation compensation needed for single photon emission computed tomography is not a simple multiplicative correction of the observed projection data as in the case of positron emission tomography.

The attenuation problem for single photon imaging is handled in a straight-forward manner in the model dependent algorithms (`CONGR`, `GRADY`, and `GVERS`). Using prior information of the attenuation coefficient distribution, equation (4.3.5) is implemented for the various models by the projection routines `PRFA` (ray factors, attenuated), `PCDA` (concave disk, attenuated) and `PPTA` (point, attenuated). Note that only parallel-beam geometry with ray sum routines have been implemented. The corresponding back-projection routines are `BRFA`, `BCDA`, and `BPTA`, respectively. For these reconstruction algorithms, the uncorrected projections of equation (4.3.5) should be supplied. When compensating for attenuation, one of the subroutines `EVATN` or `EVATU` must be called before these algorithms are executed (cf. Section 5.7).

For the model-independent algorithms (`CONVO`, `BKFIL`, `FILBK`, and `MARR`) the data must be preprocessed to take account of the attenuation problem. This problem is discussed in: T.F. Budinger and G.T. Gullberg in *Reconstruction Tomography in Diagnostic Radiology and Nuclear Medicine,* M.M. Ter-Pogossian, et al., eds., University Park Press, Baltimore, 1977, pp. 315-342.

4.4 Special Back-Projection Routines

Special back-projection routines can be used with the model-independent algorithms `CONVO`, `BKFIL`, and `FILBK`. The back-projection need not be the adjoint or transpose of a projection operation, but must be the digital approximation of an angular integral that is needed by these algorithms.

For parallel-beam geometry, the routine `BIN` (interpolation) is used to reconstruct the values of the intensity distribution at the centers of the pixels. Contributions to the back-projection image are calculated by linear interpolation between the appropriate projection bins. `BIN` can be used with each of the algorithms `CONVO`, `BKFIL`, and `FILBK`. The routines `BRF`, `BCD`, and `BPT` described above can also be used with these algorithms but give the average value of the intensity distribution within each pixel. `BIN` allows the calculation of one standard deviation statistical errors of the reconstructed intensity values when used with `CONVO`.

For fan-beam geometry, the routine `BINF` (interpolation, fan) is used with `CONVO` to reconstruct intensity values at the centers of the pixels. Like `BIN` it allows calculation of errors of the reconstructed values. In addition to the type of interpolation performed by `BIN`, the routine `BINF` applies weighting according to the relative positions of the image point (pixel center) and the origin of the fan. For a curved detector (cf. Figure 5) the weighting factor is given by

$$\frac{(\texttt{RFAN})^2}{[\texttt{RFAN} + r\cos(\phi - \theta)]^2 + [r\sin(\phi - \theta)]^2} \tag{4.4.1}$$

and for a flat detector (cf. Figure 6) the weighting factor is given by

$$\frac{(\texttt{RFAN})^2}{[\texttt{RFAN} + r\cos(\phi - \theta)]^2} \tag{4.4.2}$$

RFAN and θ are defined in Figures 5 and 6 for equations (4.4.1) and (4.4.2), respectively, and (r, ϕ) are the polar coordinates of the image point. The denominators of equations (4.4.1) and (4.4.2) are the squares of the distance and the projected distance from the image point to the origin of the fan, respectively.

For FILBK, one of the back-projection routines BRFF2 (ray factors, fan), BCDF2 (concave disk, fan) or BPTF2 (point, fan) must be used for fan-beam geometry. Back-projection using these routines results in a convolution of $1/r$ with the source. (Deconvolution follows the back-projection.) A discussion of the function of BRFF2, BCDF2 and BPTF2 can be found in Section 5.3.

5 LIBRARY RECONSTRUCTION ALGORITHMS

5.1 Iterative Algorithms

5.1.1 The Function to be Minimized

Iterative methods within the RECLBL Library minimize the function

$$\chi^2(X) = \sum_{km} \frac{\left(\sum_{ij} F_{ij}^{km} X_{ij} - p_{km}\right)^2}{\sigma_{km}^2} \ , \tag{5.1.1}$$

where p_{km} is the measured projection at the m^{th} angle and bin k; σ_{km} is the uncertainty with which p_{km} was measured; X_{ij} is the intensity in pixel (i, j) to be reconstructed; and F_{ij}^{km} is the fraction of X_{ij} that projects into p_{km}. F_{ij}^{km} depends on the model of intensity distribution within each pixel and whether attenuation compensation is involved (cf. Section 4).

In order to simplify the notation in this section, equation (5.1.1) can be rewritten in matrix form by contraction of the double indices (i, j) and (k, m) to the single indices i and k, respectively,

$$\chi^2(X) = \sum_{k} \frac{\left(\sum_i F_{ik} X_i - p_k\right)^2}{\sigma_k^2} = X \cdot MX - 2v \cdot X + c \ , \tag{5.1.2}$$

where X is the vector of intensities to reconstruct, and

$$M_{ij} = \sum_k \frac{F_{ik} F_{jk}}{\sigma_k^2} \ , \tag{5.1.3}$$

$$v_i = \sum_k \frac{F_{ik} p_k}{\sigma_k^2} \ , \tag{5.1.4}$$

$$c = \sum_k \frac{p_k^2}{\sigma_k^2} \ . \tag{5.1.5}$$

Methods that minimize $\chi^2(X)$ are called weighted least-squares methods. The weighting factors are $1/\sigma_k^2$ in equation (5.1.2). These weighting factors may also be set to unity, and some savings in memory requirements will be realized at the expense of the accuracy in the estimate of X. The two iterative least-squares algorithms of this library are `GRADY` (gradient or steepest descent minimization) and `CONGR` (conjugate gradient minimization). Other notable algorithms of this class are ART and SIRT (cf. R. Gordon, R. Bender, and G.T. Herman, J. Theoret. Biol. **29**, 1970, pp. 471–481; P.F.C. Gilbert, J. Theoret. Biol. **36**, 1972, pp. 105–117).

5.1.2 Step Length Calculation

The difference between the iterative algorithms of the RECLBL Library is the manner in which they choose the direction of the next step in the iterative process. Common to these algorithms is a step length calculation after the step direction has been chosen.

The direction of the n^{th} step is denoted by Δ^n, and the step length calculation consists of finding the factor a_n such that

$$X^{n+1} = X^n + a_n \Delta^n \tag{5.1.6}$$

minimizes $\chi^2(X^{n+1})$. To accomplish this, set the derivative of $\chi^2(X^{n+1})$ (with respect to a_n) equal to zero and solve for a_n. The solution is

$$a_n = \frac{\Delta^n \cdot \alpha^n}{\Delta^n \cdot M\Delta^n} \ , \tag{5.1.7}$$

where the vector α^n is proportional to the gradient of $\chi^2(X)$ at the n^{th} step,

$$\alpha^n = -\frac{1}{2}\nabla\chi^2(X^n) = v - MX^n \ . \tag{5.1.8}$$

5.1.3 Parameter Scaling

In most cases, convergence of the iterative process may be accelerated by performing a scale change on the parameters. This is *not* true when the diagonal elements of the matrix M are nearly equal (i.e., for the case of parallel-beam geometry without attenuation and not using errors in the reconstruction). The scale change of variables performed on the pixel values is

$$Y = DX \ , \tag{5.1.9}$$

where D is a diagonal matrix with diagonal elements equal to

$$D_{ii} = \sqrt{M_{ii}} \ . \tag{5.1.10}$$

After substituting equation (5.1.10) into equation (5.1.2), the function to be minimized has the form

$$\chi^2(Y) = Y \cdot (D^{-1}MD^{-1})Y - 2(D^{-1}v) \cdot Y + c \ . \tag{5.1.11}$$

Iterative stepping (using `GRADY` or `CONGR`) is performed on the transformed variables, Y, with M replaced by $D^{-1}MD^{-1}$ and v replaced by $D^{-1}v$. The final reconstructed values are obtained by the operation

$$X = D^{-1}Y \ . \tag{5.1.12}$$

In this manual the parameter scaling described above is called "relaxation." When this scaling is performed in the gradient method (below) it becomes the iterative relaxation method (cf. M. Goitein, Nucl. Inst. Meth. **101**, 1972, pp. 509–518).

5.1.4 Gradient Method or Method of Steepest Descent (GRADY)

The gradient method of reconstruction is implemented as follows:

```
CALL GRADY (X,PRJ,BCK,ISTP,IERR,IZER)
```

where

`X` is the reconstructed transverse section;

`PRJ` is the projection subroutine;

`BCK` is the back-projection subroutine;

`ISTP` is the number of iteration steps to take;

`IRLX` is nonzero for iterative relaxation;

`IERR` is nonzero for weighted least squares (otherwise $\sigma = 1$ is assumed);

`IZER` is zero to zero the initial solution;

(cf. Examples 6, 8, 9, 10, 11, 12 of Section 9).

The parameter `PRJ` can be one of the projection subroutines: `PCD`, `PCDA`, `PCDF`, `PLL`, `PPT`, `PPTA`, `PPTF`, `PRF`, `PRFA`, or `PRFF`; and the parameter `BCK` can be one of the back-projection subroutines: `BCD`, `BCDA`, `BCDF`, `BLL`, `BPT`, `BPTA`, `BPTF`, `BRF`, `BRFA`, or `BRFF`. These parameters are externals and should be declared in an `EXTERNAL` statement.

The gradient method takes as its step direction that direction in which $\chi^2(\mathtt{X})$ locally decreases most rapidly. This direction is opposite to the gradient so that

$$\Delta^n = \alpha^n \tag{5.1.13}$$

is chosen. The step length calculation is performed yielding a_n (equation (5.1.7)) and the step is calculated by

$$\mathtt{X}^{n+1} = \mathtt{X}^n + a_n \Delta^n \ . \tag{5.1.14}$$

5.1.5 Conjugate Gradient Method (CONGR)

The conjugate gradient method of reconstruction is implemented as follows:

```
CALL CONGR (X,PRJ,BCK,ISTP,IRLX,IERR,IZER)
```

where

`X` is the reconstructed transverse section;

`PRJ` is the projection subroutine;

`BCK` is the back-projection subroutine;

`ISTP` is the number of iteration steps to take;

`IRLX` is nonzero for iterative relaxation;

`IERR` is nonzero for weighted least squares (otherwise $\sigma = 1$ is assumed);

`IZER` is zero to zero the initial solution;

(cf. Examples 5, 7 of Section 9).

The parameter `PRJ` can be one of the projection subroutines: `PCD`, `PCDA`, `PCDF`, `PLL`, `PPT`, `PPTA`, `PPTF`, `PRF`, `PRFA`, or `PRFF`; and the parameter `BCK` can be one of the back-projection subroutines: `BCD`, `BCDA`, `BCDF`, `BLLL`, `BPT`, `BPTA`, `BPTF`, `BRF`, `BRFA`, or `BRFF`. These parameters are externals and should be declared in an `EXTERNAL` statement.

The conjugate gradient method improves convergence of the iterative process by making the step direction orthogonal to all previous steps (cf. J.M. Ortega and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970). The direction of the first step is taken the same as the gradient method,

$$\Delta^o = \alpha^o \ , \tag{5.1.15}$$

$$\mathtt{X}^1 = \mathtt{X}^o + a_o \Delta^o \ . \tag{5.1.16}$$

The succeeding step directions are given by

$$\Delta^n = \alpha^n - b_n \Delta^{n-1} \ . \tag{5.1.17}$$

where

$$b_n = \frac{\alpha^n \cdot M\Delta^{n-1}}{\Delta^{n-1} \cdot M\Delta^{n-1}} \ . \tag{5.1.18}$$

This makes all steps orthogonal in the sense

$$\Delta^n \cdot M\Delta^m = 0 \ , \ \text{for } m \neq n \tag{5.1.19}$$

The step length calculation is performed yielding a_n (equation (5.1.7)), and the step is calculated by

$$\mathtt{X}^{n+1} = \mathtt{X}^n + a_n \Delta^n \ . \tag{5.1.20}$$

5.1.6 Subroutine `USER`

All of the iterative reconstruction subroutines in the RECLBL Library (`CONGR`, `ENTPY`, `GRADY`) call a subroutine named `USER` after each iteration. The library contains a default subroutine by that name, which prints out the iteration number and the value of the function being minimized. However, it has been anticipated that the user may be interested in more than this information. Thus, the user may supply a subroutine `USER` (along with the main program and subroutine `GETUM`) to satisfy his requirements. The arguments of the subroutine are

```
SUBROUTINE USER (ITER,X,FCN)
```

where

`ITER` is the iteration number;

`X` is the array of fitted parameters,
for `CONGR` and `GRADY` — reconstructed array,
for `ENTPY` — Lagrange multipliers;

`FCN` is the value of the function being optimized,
for `CONGR` and `GRADY` — chi-square,
for `ENTPY` — objective function of the dual program.

5.2 Configuration Space Convolution Algorithm

5.2.1 One-Dimensional Convolution (CONVO)

Reconstruction by the convolution method is accomplished using the statement

```
CALL CONVO (X,XE,CNV,BCK,IERR)
```

where

`X` is the reconstructed transverse section;

`XE` is an array of uncertainties for `X`;

`CNV` is the convolution subroutine;

`BCK` is the back-projection subroutine;

`IERR` is the error flag;

(cf. Example 2 of Section 9).

The parameter `CNV` can be one of the three convolution functions: `SHLO`, `RALA`, or `LAKS`; and the parameter `BCK` can be one of the back-projection subroutines: `BCD`, `BIN`, `BINF`, `BLL`, `BPT`, or `BRF`. These parameters are externals and should be declared in an `EXTERNAL` statement. The routines `LAKS` and `BINF` *are required* for reconstructing fan-beam data. If `XE` (errors of the reconstruction `X`) are desired, then *only* the back-projection routines `BIN` or `BINF` can be used and `IERR` must be set nonzero.

The algorithm `CONVO` requires the projection angles to be equally spaced over at least π radians for parallel-beam geometry. To ensure this `MODANG` *must not be* 0 or 1 in the call to `SETUP`. When reconstructing fan-beam data, the projection angles *must be* equally spaced over 2π radians. Therefore `MODANG` *must be* 3, -3, 5, or -5 in the call to `SETUP` (cf. Section 3.2).

The algorithm performs the following operations: multiply the projection data by a weight function; convolve the projection data with a convolver; and back-project the modified projection data (cf. G.N. Ramachandran and A.V. Lakshminarayanan, Proc. Natl. Acad. Sci. U.S. **68**, 1971, pp. 2236–2240). These algorithm operations are symbolized by the equation

$$\texttt{X} = \text{back-project}[(pd) * c] \ , \tag{5.2.1}$$

where `X` is the transverse section, p are the projection data, and c is the convolution function. The weight function d is unity for parallel-beam geometry. For fan-beam geometry there are two weight functions; one is used with a curved detector and the other is used with a flat detector. These functions are defined in Section 5.2.2.

The digital implementation of this algorithm by the RECLBL Library first multiplies the projection data by a weight function

$$p'_{km} = p_{km} d(k) \ , \tag{5.2.2}$$

where k is the lateral index and m is the angular index. Then modified projection q_{km} are formed using the convolution equation

$$q_{km} = \sum_{k'} c(k - k') p'_{k'm} \ , \tag{5.2.3}$$

where c is the symmetric convolution function. The convolved projections are then back-projected giving the reconstruction

$$\mathtt{X}_{ij} = \frac{1}{\mathtt{NANG}} \sum_{km} F_{ij}^{km} q_{km} \ , \tag{5.2.4}$$

where F_{ij}^{km} are the weighting factors in the back-projection routines. A factor of $\pi/\mathtt{NANG}$ is required for the numerical calculation of the back-projection integral. However, a factor of $1/\mathtt{NANG}$ is shown in the above equation and the other factor of π is incorporated in the convolution function.

The errors `XE` in the reconstructed image are returned if the error flag `IERR` is set nonzero. If errors are desired, then one of the back-projection routines `BIN` or `BINF` *must be used*, depending whether the user is reconstructing parallel- or fan-beam geometry, respectively. The `BIN` back-projection operator is represented by the equation

$$\mathtt{X}_{ij} = \frac{1}{\mathtt{NANG}} \sum_{m} \left[f_k q_{km} + (1 - f_k) q_{k+1,m} \right] \ , \tag{5.2.5}$$

where the factors f_k are determined by linearly interpolating between adjacent bins and q_{km} are the convolved projections. Thus, the error matrix `XE` has elements given by the equation

$$\mathtt{XE}_{ij} = \frac{1}{\mathtt{NANG}} \sqrt{\sum_{m} \left[f_k^2 \mathrm{var}(q_{km}) + (1 - f_k)^2 \mathrm{var}(q_{k+1,m}) + 2 f_k (1 - f_k) \mathrm{cov}(q_{km}, q_{k+1,m}) \right]} \ . \tag{5.2.6}$$

The variance of q_{km} is given by equation

$$\mathrm{var}(q_{km}) = \sum_{k'} \left[c(k - k') d(k') \right]^2 \mathrm{var}(p_{km}) \ , \tag{5.2.7}$$

and the covariance of q_{km} and $q_{k+1,m}$ is given by the equation

$$\mathrm{cov}(q_{km}, q_{k+1,m}) = \sum_{k'} c(k - k') c(k + 1 - k') d(k')^2 \mathrm{var}(p_{k'm}) \ . \tag{5.2.8}$$

The errors of the projection data, which equal the square roots of the variances $\left(\sqrt{\mathrm{var}(p_{km})}\right)$, are input to the program using the subroutine `GETUM` (Section 3.4).

5.2.2 Convolvers and Weight Functions

The analytic expressions for the convolvers are shown below. Section 9, example 2 is an example program utilizing these convolvers with the convolution algorithm.

`RALA` Convolver

The `RALA` convolver (cf. G.N. Ramachandran, and A.V. Lakshminarayanan, Proc. Natl. Acad. Sci. U.S. **68**, 1971, pp. 2236–2240) is defined by the equation

$$c(k) = \begin{cases} \frac{\pi}{4} & \text{if } k = 0 \ , \\ \frac{-1}{\pi k^2} & \text{if } k \text{ odd} \ , \\ 0 & \text{if } k \text{ even} \ . \end{cases} \tag{5.2.9}$$

This convolver must be used only for parallel-beam geometry, for which the weight function d in equation (5.2.2) is equal to 1 for all k. The `RALA` convolver is the digital representation of the `RAMP` convolution function given in Section 5.3.3.

`SHLO` Convolver

The `SHLO` convolver (cf. L.A. Shepp and B.F. Logan, IEEE Trans. Nucl. Sci. **NS-21**, 1974, pp. 21–43) is defined by the equation

$$c(k) = \begin{cases} \frac{2}{\pi} & \text{if } k = 0 \ , \\ \frac{-2}{\pi(4k^2-1)} & \text{if } k \neq 0 \ . \end{cases} \tag{5.2.10}$$

This convolver must be used only for parallel-beam geometry, for which the weight function d in equation (5.2.2) is equal to 1 for all k.

Figure 8 compares the graphs of the `RALA` and `SHLO` convolvers. The `SHLO` convolver is designed such that the convolution function $c(x)$, which is equal to $c(k)$ at $x = k$ and linear in the intervening intervals, has a filter function that is the Fourier transform of $c(x)$ equal to

$$\tilde{c}(f) = 2|\sin \pi f| \left(\frac{\sin \pi f}{\pi f} \right)^2 \ . \tag{5.2.11}$$

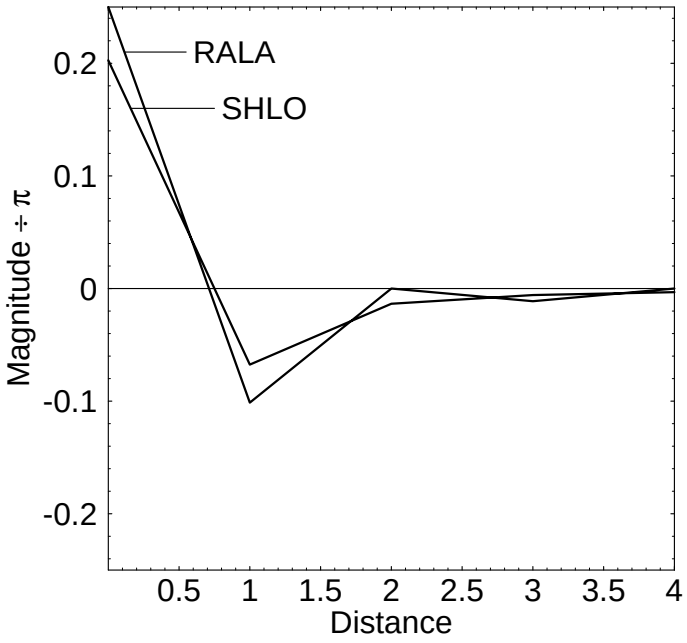

Figure 8: Two convolvers in the RECLBL Library used with parallel-beam geometry.

The `SHLO` and `RALA` convolution functions have widths for the central lobe that are nearly the same. Therefore, the resolutions in the reconstructed images are similar for perfect data. However, the side lobes for the `SHLO` convolver are damped, reducing noise amplification for data with statistical fluctuations.

LAKS Convolver

The LAKS convolver (cf. G.T. Herman, A.V. Lakshminarayanan, and A. Naparstek, Comput. Biol. Med. **6**, 1976, pp. 259–271) is used only for fan-beam geometry. For a curved detector the convolution function is defined by the equation

$$c(k) = \begin{cases} \frac{\pi}{4} & \text{if } k = 0 \ , \\ \frac{-(1/\mathtt{RFAN})^2}{\pi \sin^2(k/\mathtt{RFAN})} & \text{if } k \text{ odd} \ , \\ 0 & \text{if } k \text{ even} \ , \end{cases} \tag{5.2.12}$$

with weights d(k) defined by the equation

$$d(k) = \cos(k/\mathtt{RFAN}) \ . \tag{5.2.13}$$

For a flat detector the convolution function is defined by the equation

$$c(k) = \begin{cases} \frac{\pi}{4} & \text{if } k = 0 \ , \\ \frac{-1}{\pi k^2} & \text{if } k \text{ odd} \ , \\ 0 & \text{if } k \text{ even} \ , \end{cases} \tag{5.2.14}$$

with weights d(k) defined by the equation

$$d(k) = \frac{1}{\sqrt{1 + (k/\mathtt{RFAN})^2}} \ . \tag{5.2.15}$$

5.3 Fourier Space Convolution Algorithms

5.3.1 Back-Projection of Filtered Projections Algorithm (BKFIL)

The back-projection of filtered projections algorithm is implemented as follows:

```
CALL BKFIL (X,FIL,BCK,ORDERX,FREQX)
```

where

`X` is the reconstructed transverse section;

`FIL` is the filter subroutine;

`BCK` is the back-projection subroutine;

`ORDERX` is a filter parameter used only by the filter `BUTER`;

`FREQX` is a filter parameter;

(cf. Example 3 of Section 9).

The parameter `FIL` can be one of the five filters: `BUTER`, `HAN`, `HAM`, `PARZN`, or `RAMP`. The parameter `BCK` can be one of the back-projection subroutines: `BCD`, `BIN`, `BLL`, `BPT` or `BRF`. These parameters are externals and should be declared in an `EXTERNAL` statement.

A description of the filter options and the appropriate values for `ORDERX` and `FREQX` parameters is found in Section 5.3.3. The cutoff frequency `FREQX` for the filters has units of cycles per projection bin. Thus, for a Nyquist frequency equal to 1 cycle per projection bin,

one can choose FREQX=0.5 for most applications. Other appropriate values for FREQX are described in Section 5.3.3.

The algorithm BKFIL requires the projection angles to be equally spaced over at least π radians. To ensure this MODANG *must not be* 0 or 1 in the call to SETUP (cf. Section 3.2).

The algorithm performs the following sequence of operations: Fourier transform the projection data vector; multiply the complex values by one of the five optional filters; inverse Fourier transform these modified frequencies; and back-project the modified projection data (cf. T.F. Budinger and G.T. Gullberg, IEEE Trans. Nucl. Sci. **NS-21**, 1974, pp. 2–20). These algorithm operations are symbolized as:

$$\mathtt{X} = \text{back-project}\left\{\mathcal{F}_1^{-1}\left[\tilde{c}\mathcal{F}_1(p)\right]\right\} \ , \tag{5.3.1}$$

where X is the transverse section, p are the projection data, $\tilde{c}$ is the filter function, and $\mathcal{F}_1$ denotes one-dimensional Fourier transformation. The filter function $\tilde{c}$ is equal to the product of a window function $w(R)$ and the absolute value of the frequency:

$$\tilde{c} = |R|w(R) \ . \tag{5.3.2}$$

Due to the Fourier convolution theorem, this method of reconstruction is equivalent to the convolution method except that the convolution of the projection data is carried out in frequency space. The filter function $\tilde{c}$ is the Fourier transform of the convolution function c. The rationale for performing the filter operation in Fourier space is given in Section 5.3.3.

The digital implementation of this algorithm by the RECLBL Library performs the discrete Fourier transform of the projection data given by the equation

$$\tilde{p}_{km} = \frac{1}{\mathtt{KDIMT}} \sum_{l=0}^{\mathtt{KDIMT}-1} p_{lm} e^{-i2\pi kl/\mathtt{KDIMT}} \ , \tag{5.3.3}$$

where k is the projection bin index and m is the angle index. KDIMT is equal to $2^{\mathtt{IPOW2}}$ where IPOW2 = 2×(the smallest power of two that is greater than or equal to KDIMU). The factor of 2 is required so that the convolution result of one period does not overlap the convolution result of the succeeding period when using the discrete Fourier transform. After discrete Fourier transforming the projection data, Fourier transformed values $\tilde{p}_{km}$ are multiplied by a filter function giving

$$\tilde{q}_{km} = \tilde{c}\left(\frac{k}{\mathtt{KDIMT}}\right)\tilde{p}_{km} \ . \tag{5.3.4}$$

Then the values $\tilde{q}_{km}$ are discrete inverse Fourier transformed giving the convolved projection

$$q_{km} = \sum_{l=0}^{\mathtt{KDIMT}-1} \tilde{q}_{lm} e^{\ i2\pi kl/\mathtt{KDIMT}} \ . \tag{5.3.5}$$

The convolved projection data are then back-projected as in the convolution method to give the reconstruction

$$\mathtt{X}_{ij} = \frac{\pi}{\mathtt{NANG}} \sum_{km} F_{ij}^{km} q_{km} \ , \tag{5.3.6}$$

where F_{ij}^{km} are the weighting factors in the projection and back-projection routines. The factor $\pi/\mathtt{NANG}$ is the step size in the numerical calculation of the back-projection integral.

5.3.2 Filter of the Back-Projection Algorithm (FILBK)

Reconstruction by the filter of the back-projection method is accomplished using the statement

```
CALL FILBK (X,FIL,BCK,ORDERX,FREQX)
```

where

`X` is the reconstructed transverse section;

`FIL` is the filter subroutine;

`BCK` is the back-projection subroutine;

`ORDERX` is a filter parameter used only by the filter `BUTER`;

`FREQX` is a filter parameter;

(cf. Example 4 of Section 9).

The parameter `FIL` can be one of the five filters: `BUTER`, `HAN`, `HAM`, `PARZN`, or `RAMP`. The parameter `BCK` can be one of the back-projection subroutines: `BCD`, `BCDF2`, `BIN`, `BLL`, `BPT`, `BPTF2`, `BRF`, or `BRFF2`. These parameters are externals and should be declared in an `EXTERNAL` statement. The back-projection subroutines `BCDF2`, `BPTF2`, and `BRFF2` are required for reconstructing fan-beam projection data since the filter of the back-projection algorithm requires special weighting for fan-beam geometry. When reconstructing fan-beam projection data with this algorithm, the user *should not* use `BCDF`, `BPTF`, or `BRFF`.

A description of the filter options and the appropriate values for the `ORDERX` and `FREQX` parameters if found in Section 5.3.3. The cutoff frequency `FREQX` for the filters has units of cycles per pixel. (In the algorithm `BKFIL`, `FREQX` has units of cycles per projection bin.) For most applications `FREQX`=0.5 gives good results.

The algorithm `FILBK` requires that the projection angles be equally spaced over at least π radians for parallel-beam geometry. To ensure this, `MODANG` *must not be* 0 or 1 in the call to `SETUP`. When reconstructing fan-beam data, the projection angles *must be* equally spaced over 2π radians. Therefore `MODANG` *must be* 3, -3, 5, or -5 in the call to `SETUP` (cf. Section 3.2).

This algorithm performs the following sequence of operations: back-project the projection data; Fourier transform the two-dimensional back-projection image; multiply the two-dimensionally distributed Fourier coefficients by one of the optional filter functions; and perform the two-dimensional inverse Fourier transform (cf. R.H.T. Bates and T.M. Peters, New Zealand J. Sci. **14**, 1971, pp. 883–896). These algorithm operations are symbolized as:

$$\mathtt{X} = \mathcal{F}_2^{-1}\left\{\tilde{c}\mathcal{F}_2\left[\text{back-project}(p)\right]\right\} \ , \tag{5.3.7}$$

where `X` is the transverse section, p are the projection data, $\tilde{c}$ is the filter function, and $\mathcal{F}_2$ denotes the two-dimensional Fourier transform. The filter function $\tilde{c}$ is equal to the product of a window function $w(R)$ and of the absolute value of the frequency:

$$\tilde{c}(R) = |R|w(R) \ . \tag{5.3.8}$$

This method of reconstruction is equivalent to performing a two-dimensional convolution of a sharpening kernel with the back-projection data. The purpose of this method is to effect a deconvolution of the true image X from the back-projected image b given by the equation

$$b(x,y) = \int\int \frac{\mathtt{X}(x',y')}{\sqrt{(x-x')^2+(y-y')^2}} dx'dy' = \mathtt{X} * r^{-1} \ . \tag{5.3.9}$$

The derivation of the algorithm is based on the convolution theorem and the fact that $r^{-1} = \mathcal{F}_2^{-1}(R^{-1})$ where R is the frequency.

Three general geometries for back-projection are available: parallel-beam, fan-beam with curved detector and fan-beam with flat detector. The back-projection operation for parallel-beam geometry requires the summation of line integrals over the range of 180 degrees and is given by the equation

$$b_{=}(r,\phi) = \int_0^{\pi} p\left[r\sin(\phi-\theta),\theta\right] d\theta \ . \tag{5.3.10}$$

The five choices of back-projection subroutines for parallel-beam are BCD, BIN, BLL, BPT, or BRF. Fan-beam geometries require samples around the full 360 degrees for use of this algorithm (cf. G.T. Gullberg, Lawrence Berkeley Laboratory Report LBL 5604, 1977). The back-projection operations for the fan-beam geometry are given for a curved detector by the equation

$$b_c(r,\phi) = \frac{1}{2}\int_0^{2\pi} p_c(\xi^*,\theta) d\theta \ , \tag{5.3.11}$$

where

$$\xi^* = \mathtt{RFAN}\tan^{-1}\left[\frac{r\sin(\phi-\theta)}{\mathtt{RFAN}+r\cos(\phi-\theta)}\right] \ , \tag{5.3.12}$$

and for a flat detector by the equation

$$b_f(r,\phi) = \frac{1}{2}\int_0^{2\pi} \frac{p_f(\xi^*,\theta)\sqrt{r^2+\mathtt{RFAN}^2+2\ \mathtt{RFAN}\ r\cos(\phi-\theta)}}{\mathtt{RFAN}+r\cos(\phi-\theta)} d\theta \ , \tag{5.3.13}$$

where

$$\xi^* = \frac{\mathtt{RFAN}r\sin(\phi-\theta)}{\mathtt{RFAN}+r\cos(\phi-\theta)} \ . \tag{5.3.14}$$

The variable RFAN is the distance of the source in transmission tomography or of the pinhole in emission tomography to the center of rotation. Notice in equation (5.3.13) that when using a flat detector a special weighting is required for the back-projection operation. The three choices of back-projection subroutines for fan-beam geometry are BCDF2, BPTF2, and BRFF2.

The digital implementation of this algorithm by the RECLBL Library first back-projects the projection data p_{km} using the equation

$$b_{ij} = \sum_{km} F_{ij}^{km} p_{km} \ , \tag{5.3.15}$$

where k is the projection bin index, m is the angle index, and F_{ij}^{km} are the weighting factors in the projection and back-projection routines. The back-projection image is then discrete Fourier transformed using the equation

$$\tilde{b}_{kl} = \frac{1}{\texttt{NDIM}^2} \sum_{n=0}^{\texttt{NDIM}-1} \sum_{m=0}^{\texttt{NDIM}-1} b_{nm} e^{-2\pi i(kn+lm)/\texttt{NDIM}} \ . \tag{5.3.16}$$

NDIM is equal to $2^{\texttt{IPOW2}}$ where IPOW2 = 2×(the smallest power of two that is greater than or equal to NDIMU). Next the discrete Fourier transformed values $\tilde{b}_{kl}$ are multiplied by a filter function $\tilde{c}$ giving

$$\tilde{X}_{kl} = \tilde{c}\left(\frac{\sqrt{k^2+l^2}}{\texttt{NDIM}}\right)\tilde{b}_{kl} \ . \tag{5.3.17}$$

Then the values $\tilde{X}_{kl}$ are inverse Fourier transformed and multiplied by normalization factor to give the reconstruction

$$\texttt{X}_{nm} = \frac{\pi}{\texttt{NANG}\cdot\texttt{PWID}} \sum_{k=0}^{\texttt{NDIM}-1} \sum_{l=0}^{\texttt{NDIM}-1} \tilde{X}_{kl} e^{\ 2\pi i(nk+ml)/\texttt{NDIM}} \ . \tag{5.3.18}$$

The factor $\pi/$NANG is the step size in the numerical calculation of the back-projection integral. The factor $1/$PWID is the result of scaling the reconstruction space.

5.3.3 Filter Functions

The algorithms BKFIL and FILBK require a filter to be designated. These algorithms have been developed with various operations for frequency space filters because frequency space manipulation lends itself to easily changing the noise propagation vs. resolution properties of the convolution kernel. The user can improve resolution by changing the filter shape, but the noise amplification will increase. Alternatively, the user can suppress noise; however, this noise suppression will come at the cost of resolution. A second reason for incorporation of various filters with the Fourier space algorithms is that the computational method for reconstruction is more efficient using the Fast Fourier Transform than convolution in real space.

The particular filter desired by the user is evaluated by one of the five optional external subroutines: BUTER, HAM, HAN, PARZN, or RAMP. The external subroutine chosen must be designated in the main program (cf. example 3 of Section 9). These five filters correspond to multiplying the ramp function in frequency space by one of the following windows: Butterworth, Hann, Hamming, Parzen, or rectangular. (A thorough discussion of these windows and their application is found in: R.K. Otnes and L. Enochson, *Digital Time Series Analysis,* John Wiley and Sons, 1972; R.W. Hamming, *Digital Filters,* Prentice Hall, 1977.)

Texts usually define a digital filter as the real space convolution equation:

$$q_n = \sum_k c_{n-k} p_k \ . \tag{5.3.19}$$

In this manual a convolver means the convolving sequence $\{c_k\}$ and a filter is the Fourier transform of a continuous convolution function $c(x)$ such that $c_k = c(x = k)$.

Real space convolution and frequency filtering are equivalent operations. As is shown in examples 2 and 3 of Section 9, the `RAMP` filter used with the algorithm `BKFIL` achieves the same result as the `RALA` convolution function used with `CONVO`. In `BKFIL` the operation of filtering is done by multiplying the filter values by the Fourier transform of the projection data, then inverse Fourier transforming the result. Projection data modified in the same fashion is obtained by convolving the projection data with the real space equivalent of the `RAMP` function. Symbolically we have

$$\text{Modified projection} = \mathcal{F}_1^{-1}\left[|R|\ w(R)\mathcal{F}_1(\text{projection data})\right]\ , \tag{5.3.20}$$

where $\mathcal{F}_1$ denotes the one-dimensional Fourier transform and $w(R)$ is one of the window functions defining the filter $|R|\ w(R)$. From the convolution theorem, the equivalent result can be obtained as

$$\text{Modified projection} = \left\{\mathcal{F}_1^{-1}\left[|R|\ w(R)\right]\right\} * \text{projection data}\ , \tag{5.3.21}$$

where the convolver $\mathcal{F}^{-1}\left[|R|\ w(R)\right]$ is determined by the window function w and the symbol $*$ denotes convolution.

The shapes of the window functions are shown in Figure 9. The width of the window is measured as the distance between the closest zeroes on each side of the center lobe of the inverse Fourier transform of the window function. Ideally, for good resolution, the window function should have a central lobe that is tall and narrow. The size lobes for the inverse Fourier transform of these window functions give rise to the Gibbs phenomenon, which is observed as artifacts that are contamination from adjacent parts of the reconstruction.

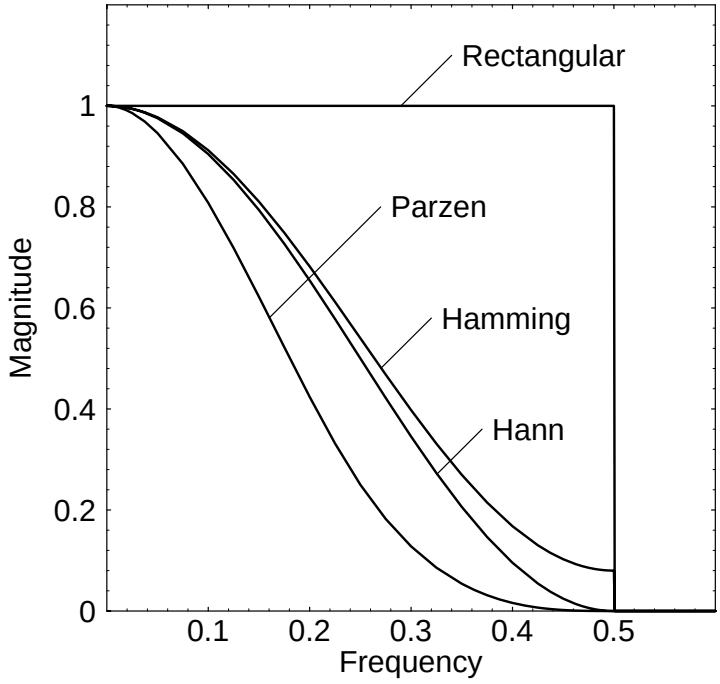

Figure 9: These window functions are multiplied by a ramp function giving the filters shown in the next figure.

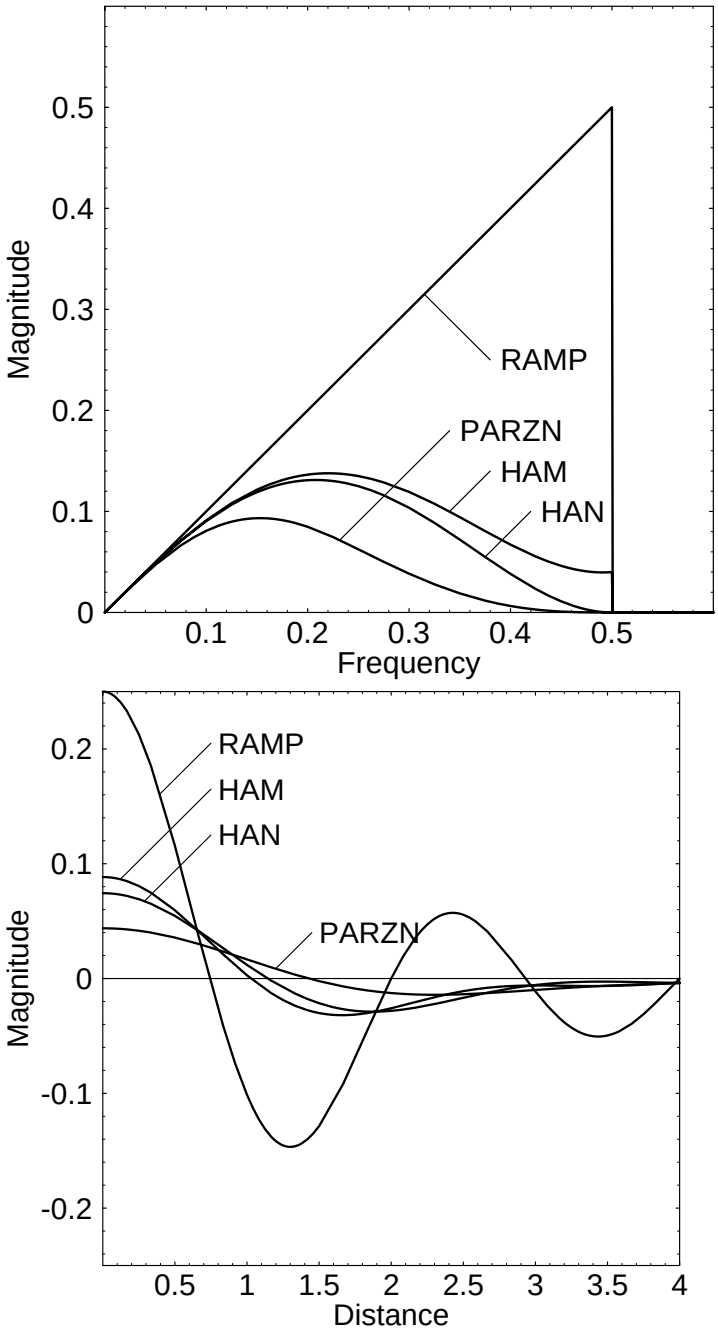

Figure 10: Window functions of the previous figure multiplied by a ramp with a cutoff frequency `FREQX` = 0.5 (upper). The inverse Fourier transform of the filters in the upper figure give the real space convolution functions (lower).

The RECLBL filters: `HAN`, `HAM`, `PARZN`, `RAMP` (Figure 10 upper) are obtained by multiplying the ramp function by the window functions in Figure 9: Hann, Hamming, Parzen, Rectangular, respectively. Figure 10 lower gives the graphs of the convolution functions that are the inverse Fourier transform of the filter functions given in Figure 10 upper. The analytic expressions for the frequency filters and the corresponding real space convolution functions are shown below. The frequency parameter f_m is the frequency parameter `FREQX`, which is input to the subroutines `BKFIL` and `FILBK`.

Rectangular Window and `RAMP` Filter

The rectangular window is defined by the equation

$$w(f) = \begin{cases} 1 & \text{if } |f| \le f_m \ , \\ 0 & \text{if } |f| > f_m \ . \end{cases} \tag{5.3.22}$$

Multiplying the rectangular window by the ramp function in frequency space gives the `RAMP` filter

$$\tilde{c}(f) = \begin{cases} |f| & \text{if } |f| \le f_m \ , \\ 0 & \text{if } |f| > f_m \ . \end{cases} \tag{5.3.23}$$

The inverse Fourier transform of the RAMP filter gives the convolution function

$$c(x) = 2f_m^2 \left(\frac{\sin 2\pi f_m x}{2\pi f_m x} \right) - f_m^2 \left(\frac{\sin \pi f_m x}{\pi f_m x} \right)^2 \ . \tag{5.3.24}$$

The `RAMP` filter gives the best resolution in the reconstructed image for perfect data but amplifies noise for data with statistical fluctuations. The sharp cutoff gives rise to intensity oscillation in regions of sharp contrast and thus generates artifacts in the reconstructed image.

Hann Window and `HAN` Filter

The Hann window is defined by the equation

$$w(f) = \begin{cases} 0.5 + 0.5 \cos \pi f / f_m & \text{if } |f| \le f_m \ , \\ 0 & \text{if } |f| > f_m \ . \end{cases} \tag{5.3.25}$$

Multiplying the Hann window by the `RAMP` function gives the `HAN` filter

$$\tilde{c}(f) = \begin{cases} 0.5|f| + 0.5|f| \cos \pi f / f_m & \text{if } |f| \le f_m \ , \\ 0 & \text{if } |f| > f_m \ . \end{cases} \tag{5.3.26}$$

The inverse Fourier transform of the filter function gives the convolution function

$$\begin{aligned} c(x) \ = \ & \frac{f_m^2}{2} \frac{\sin\left[f_m(2\pi x + \pi/f_m)\right]}{f_m(2\pi x + \pi/f_m)} - \frac{f_m^2}{4} \left(\frac{\sin\left[f_m(2\pi x + \pi/f_m)/2\right]}{f_m(2\pi x + \pi/f_m)/2} \right)^2 \\ & + f_m^2 \frac{\sin 2\pi f_m x}{2\pi f_m x} - \frac{f_m^2}{2} \left(\frac{\sin \pi f_m x}{\pi f_m x} \right)^2 \\ & + \frac{f_m^2}{2} \frac{\sin\left[f_m(2\pi x - \pi/f_m)\right]}{f_m(2\pi x - \pi/f_m)} - \frac{f_m^2}{4} \left(\frac{\sin\left[f_m(2\pi x - \pi/f_m)/2\right]}{f_m(2\pi x - \pi/f_m)/2} \right)^2 \end{aligned} \tag{5.3.27}$$

For the Hann window the central lobe of the convolution function $c(x)$ is wider than the central lobe of the corresponding convolution function for the rectangular window, but its side lobes are greatly reduced. Therefore, the reconstructed image has a smoother texture with a loss in resolution.

A frequency parameter f_m for the `HAN` filter, which is two times the value of the cutoff frequency for the `RAMP` filter, gives an approximation to the ramp function with a small rolloff near $f = f_m/2$. The `RAMP` and `HAN` filters give the same resolution in the reconstructed image when the `RAMP` has a cutoff frequency equal to one half that of the `HAN` (cf. D.A. Chesler and S.J. Riederer, Phys. Med. Biol. **20**, 1975, pp. 632–636).

Hamming Window and `HAM` Filter

The Hamming window is defined by the equation

$$w(f) = \begin{cases} 0.54 + 0.46 \cos \pi f / f_m & \text{if } |f| \le f_m \ , \\ 0 & \text{if } |f| > f_m \ . \end{cases} \tag{5.3.28}$$

Multiplying the Hamming window by the ramp function in frequency space gives the `HAM` filter

$$\tilde{c}(f) = \begin{cases} 0.54|f| + 0.46|f| \cos \pi f / f_m & \text{if } |f| \le f_m \ , \\ 0 & \text{if } |f| > f_m \ . \end{cases} \tag{5.3.29}$$

The inverse Fourier transform gives the convolution function, which is merely equation (5.3.27) with terms 1, 2, 5 and 6 reduced by 1.08 and terms 3 and 4 increased by 1.08:

$$\begin{aligned} c(x) \ = \ & 0.46 f_m^2 \frac{\sin\left[f_m(2\pi x + \pi/f_m)\right]}{f_m(2\pi x + \pi/f_m)} - 0.23 f_m^2 \left(\frac{\sin\left[f_m(2\pi x + \pi/f_m)/2\right]}{f_m(2\pi x + \pi/f_m)/2} \right)^2 \\ & + 1.08 f_m^2 \frac{\sin 2\pi f_m x}{2\pi f_m x} - 0.54 f_m^2 \left(\frac{\sin \pi f_m x}{\pi f_m x} \right)^2 \\ & + 0.46 f_m^2 \frac{\sin\left[f_m(2\pi x - \pi/f_m)\right]}{f_m(2\pi x - \pi/f_m)} - 0.23 f_m^2 \left(\frac{\sin\left[f_m(2\pi x - \pi/f_m)/2\right]}{f_m(2\pi x - \pi/f_m)/2} \right)^2 \end{aligned} \tag{5.3.30}$$

The Hamming window has smaller extreme values in the side lobes than does the Hann window. The maximum side lobe for the Hamming window is approximately one-fifth that of the Hann window.

Parzen Window and `PARZN` Filter

The Parzen window is defined by the equation

$$w(f) = \begin{cases} 1 - 6\left(\frac{|f|}{f_m}\right)^2 \left(1 - \frac{|f|}{f_m}\right) & \text{if } |f| \le f_m/2 \ , \\ 2\left(1 - \frac{|f|}{f_m}\right)^3 & \text{if } f_m/2 < |f| \le f_m \ , \\ 0 & \text{if } |f| > f_m \ . \end{cases} \tag{5.3.31}$$

where f_m is the cutoff frequency. Multiplying the Parzen window by the ramp function gives the PARZN filter

$$\tilde{c}(f) = \begin{cases} |f| - 6|f| \left(\frac{|f|}{f_m}\right)^2 \left(1 - \frac{|f|}{f_m}\right) & \text{if } |f| \le f_m/2 \ , \\ 2|f| \left(1 - \frac{|f|}{f_m}\right)^3 & \text{if } f_m/2 < |f| \le f_m \ , \\ 0 & \text{if } |f| > f_m \ . \end{cases} \tag{5.3.32}$$

The inverse Fourier transform gives the convolution function

$$\begin{aligned} c(x) \ = \ & [48\pi f_m x \cos 2\pi f_m x - 96 \sin 2\pi f_m x - 96\pi f_m x \cos \pi f_m x \\ & + 384 \sin \pi f_m x - 16\pi^3 f_m^3 x^3 - 144\pi f_m x] \, / \left(32\pi^5 f_m^3 x^5\right) \end{aligned} \tag{5.3.33}$$

and

$$c(0) = 0.175 f_m^2 \ . \tag{5.3.34}$$

The central lobe of the Parzen window is about 30than either the Hann or the Hamming window. Thus the reconstructed image resolution will be less than can be achieved with either the HAN or HAM filter. On the other hand the PARZN filter suppresses noise. The side lobes for the Hann and Hamming window oscillate between positive and negative values, whereas with the Parzen window the side lobes always remain positive.

Butterworth Filter and BUTER Filter

The major advantage of the filter BUTER is that it can be modified to suit the user. The filter is derived by using the magnitude of the Butterworth filter as a window function:

$$R(f) = \frac{1}{\sqrt{1 + (f/f_m)^{2n}}} \ , \tag{5.3.35}$$

where f_m is a frequency parameter and n is the order of the filter. This is multiplied by the ramp function giving the filter BUTER:

$$\frac{|f|}{\sqrt{1 + (|f|/f_m)^{2n}}} \tag{5.3.36}$$

The shape of the filter is designed by declaring values of FREQX $= f_m$ and ORDERX $= 2n$. Note that the order of a Butterworth filter (equation (5.3.35)) is given as $n =$ ORDERX$/2$ if ORDERX is an even integer. However, the filter BUTER in the RECLBL Library allows ORDERX to be any real value.

Usually the Butterworth filter is used in connection with recursive filtering; however, in our application the amplitude of the Butterworth filter is multiplied with a ramp function to obtain a rolloff. A filter can be selected that gives the desired result in the reconstruction by using a value for ORDERX that may be in the range of 5 to 350 and a value for FREQX between 0.25 and 1.

A filter is designed by calculating the appropriate window widths between 0 and f_p and the corresponding transition bands between the pass-band frequency f_p and the stop-band frequency f_s as illustrated in Figure 11. If the values of ϵ, A, f_p, and f_s are known for

a particular window, then the parameters ORDERX and FREQX of the Butterworth filter are determined using the equations:

$$\texttt{ORDERX} = \frac{2\log\left(\epsilon/\sqrt{A^2-1}\right)}{\log\left(f_p/f_s\right)} \tag{5.3.37}$$

$$\texttt{FREQX} = \frac{f_p}{(\epsilon)^{2/\texttt{ORDERX}}} \tag{5.3.38}$$

(cf. R.W. Hamming, *Digital Filters,* Prentice Hall, 1977, p. 189).

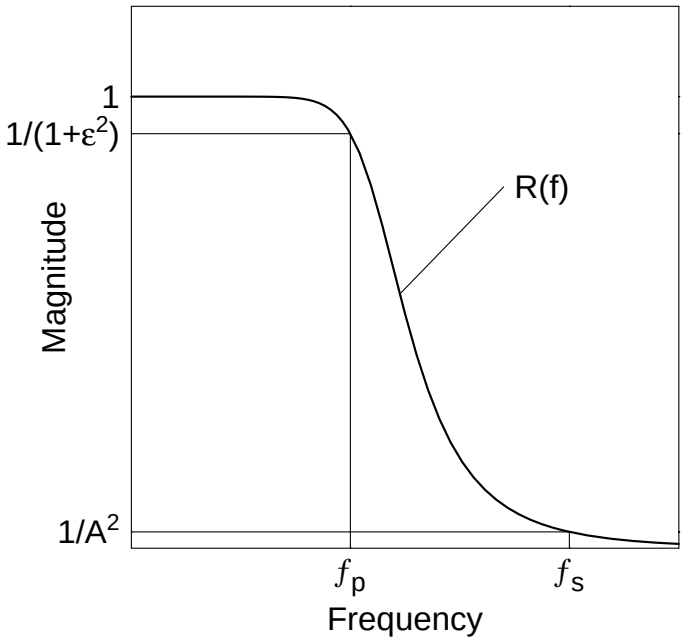

Figure 11: Method of designing a Butterworth filter. Parameters ϵ and A are calculated from ordinates at the user selected pass frequency f_p and stop frequency f_s.

The window defined by the Butterworth filter can either be designed so that it has a narrow transition band between f_p and f_s, and thus approaches a rectangular window, or can be designed so that it has a wide transition band such as the Hann or Hamming window. For example, suppose a Hann window had FREQX $= 0.5$ and we select $f_p = 0.23$ and $f_s = 0.47$, then equation (5.3.25) gives $w(0.23) = 0.563$ and $w(0.47) = 0.009$. A Butterworth filter that matches this Hann window at f_p and f_s is designed by solving equations (5.3.39) and (5.3.40) for A and ϵ:

$$R(0.23) = \frac{1}{1+\epsilon^2} = 0.563\ , \tag{5.3.39}$$

$$R(0.47) = \frac{1}{A^2} = 0.009\ . \tag{5.3.40}$$

This is illustrated in Figure 11 and the results are $A = 10.624$ and $\epsilon = 0.882$. From equations (5.3.37) and (5.3.38) we calculate the parameters of the Butterworth filter ORDERX $=$

6.95 and FREQX = 0.238. Figure 12 upper compares the window defined by the Butterworth filter for ORDERX = 6, FREQX = 0.23 with the Hann window for FREQX = 0.5. Figure 12 lower compares the corresponding filters.

Designing a window function with a narrow transition band in frequency space is equivalent to having a narrow central lobe that will give good resolution in the reconstructed image, but concurrently the side lobes for such a window function are larger, thus amplifying noise. On the other hand, a wider transition band gives poorer resolution with reduced noise amplification.

5.4 Maximum Entropy Algorithm (ENTPY)

The maximum entropy method of reconstruction is designed for projection data samples that give a system of linear equations that are underestimated. This method of reconstruction is accomplished using the statement

```
CALL ENTPY (X,PRJ,BCK,LIMITX,ERENTX)
```

where

X is the reconstructed transverse section;

PRJ is the projection subroutine;

BCK is the maximum number of iterations allowed to minimize the objective function for the dual program;

ERENTX is the test value representing the expected absolute error between the true solution and the iterative solution;

(cf. Example 14 of Section 9).

The parameter PRJ can be one of the projection subroutines: PCD, PCDF, PLL, PPT, PPTF, PRF, or PRFF; and the parameter BCK can be one of the back-projection subroutines: BCD, BCDF, BLL, BPT, BPTF, BRF, or BRFF. These parameters are externals and should be declared in an EXTERNAL statement.

The maximum entropy algorithm requires as many as 121 (LIMITX = 121) iterations for a 21×21 reconstruction array if ERENTX = 10^{-6}. Therefore, due to the computer time requirements the user might want to use this method for small array sizes and sample sizes. For larger arrays, computer tests have shown that if LIMITX = 15, ENTPY gives a good reconstruction even when the iterative procedure has not yet converged to the maximum entropy solution. ERENTX should not be any smaller than 10^{-D}, where D is the number of significant digits in floating point representation.

When reconstructing fan-beam data, the projection angles *must be* equally spaced over 2π radians. Therefore MODANG *must be* 3, -3, 5, or -5 in the call to SETUP (cf. Section 3.2).

The maximum entropy method determines a solution for the reconstructed pixel values that maximizes an entropy function subject to a consistent system of projection constraints. The problem is stated formally as follows:

Find the maximum of

$$S(\mathtt{X}) = -\sum_{ij} \frac{\mathtt{X}_{ij}}{T} \ln\left(\frac{\mathtt{X}_{ij}}{T}\right) \tag{5.4.1}$$

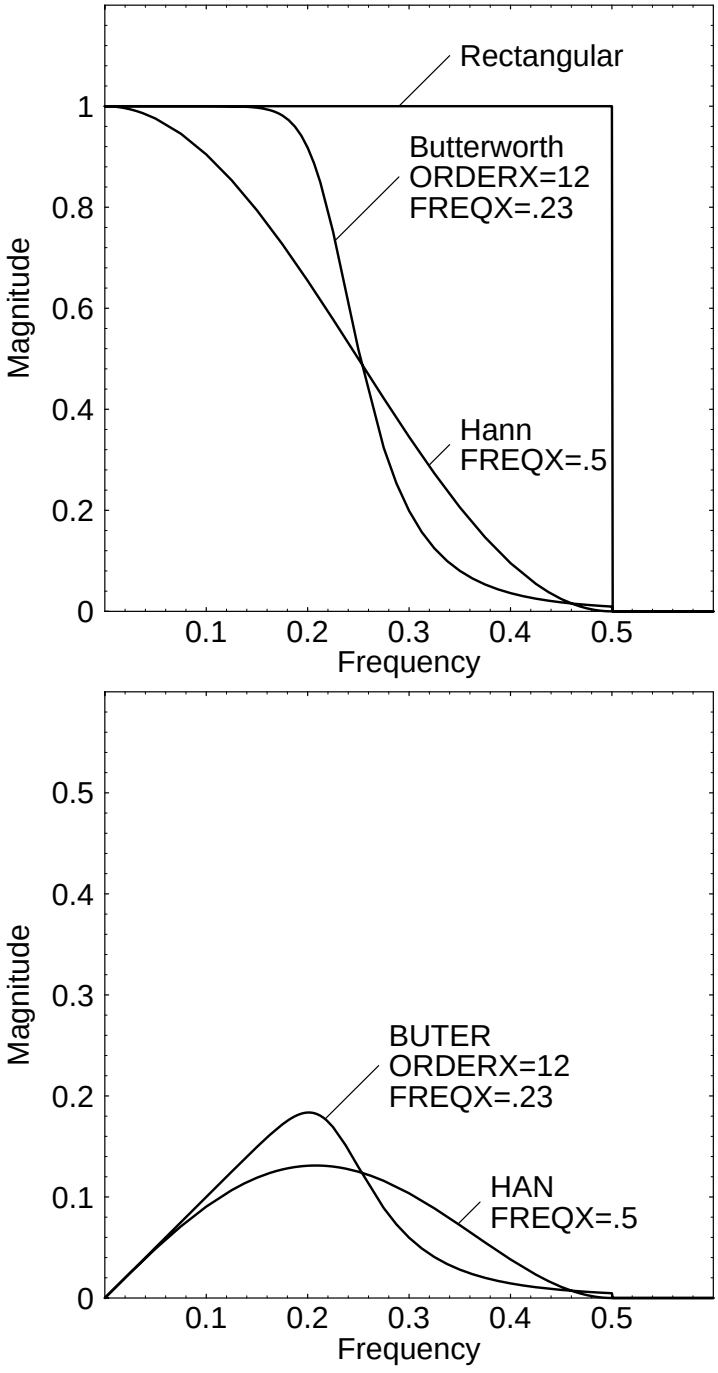

Figure 12: The upper figure shows the Hann window with a cutoff frequency FREQX = 0.5 and the Butterworth filter with ORDERX = 6 and FREQX = 0.23. The lower figure is a plot of the filters BUTER and HAN obtained by multiplying these window functions by a simple ramp.

subject to the constraints

$$\sum_{ij} F_{ij}^{km} \, \mathtt{X}_{ij} = p_{km} \ , \ \text{for all } k, m \ , \tag{5.4.2}$$

$$\sum_{ij} \mathtt{X}_{ij} = T \ , \tag{5.4.3}$$

$$\mathtt{X}_{ij} \geq 0 \ . \tag{5.4.4}$$

The intensities $\mathtt{X}_{ij}$ are elements of the array `X` representing the reconstructed transverse section, which has a total intensity T. These intensities are related to the projection values p_{km} by the weighting factors F_{ij}^{km}, which are determined by the particular choice of the projection and back-projection subroutines.

A solution for the reconstructed transverse section is solved utilizing Lagrange multipliers and duality theory (cf. R.T. Rockafellar, *Convex Analysis,* Princeton University Press, 1970). Using conjugate gradient methods, a solution to the dual program:

Find the minimum of

$$g(\lambda) = \ln \left(\sum_{ij} e^{\ z_{ij}} \right) - \sum_{km} \frac{\lambda_{km} p_{km}}{T} \ , \tag{5.4.5}$$

where $z_{ij} = \sum_{km} F_{ij}^{km} \lambda_{km}$, gives the optimum solution for the Lagrange multipliers. This solution immediately gives an optimal solution for the reconstructed image from the equation

$$\mathtt{X}_{ij} = \frac{T e^{\ z_{ij}}}{\sum_{i'j'} e^{\ z_{i'j'}}} \ , \tag{5.4.6}$$

where $z_{ij} = \sum_{km} F_{ij}^{km} \lambda_{km}$, and the λ_{km} are the optimum Lagrange multipliers (cf. G.T. Gullberg, in *Information Processing in Scintigraphy,* Proceedings of the IVth International Conference, Orsay, France, July 15–16, 1975, eds. C. Raynaud and A. Todd-Pokropek, pp. 325–332).

The maximum entropy reconstruction method will give an estimate for the reconstructed image which has less structure than any other possible solution, and thus avoids any bias while agreeing with the projection data.

5.5 Generalized Inverse Algorithm (`GVERS`)

The generalized inverse method of reconstruction is obtained by using the Fortran statement

```
CALL GVERS (X,XE,PRJ,BCK,CHISQ,IERR)
```

where

`X` is the reconstructed transverse section;

`XE` are the errors in the reconstructed image;

`PRJ` is the projection subroutine;

BCK is the back-projection subroutine;

CHISQ is the resulting chi-square;

IERR is the error indicator;

(cf. Example 15 of Section 9).

The parameter PRJ can be one of the projection subroutines: PCD, PCDA, PCDF, PLL, PPT, PPTA, PPTF, PRF, PRFA, or PRFF; and the parameter BCK can be one of the back-projection subroutines: BCD, BCDA, BCDF, BLL, BPT, BPTA, BPTF, BRF, BRFA, or BRFF. These parameters are externals and should be declared in an EXTERNAL statement. The chi-square, CHISQ, which is returned is defined by the equation

$$\texttt{CHISQ} = \chi^2(\texttt{X}) = \sum_{km} \frac{1}{\sigma_{km}^2} \left(\sum_{ij} F_{ij}^{km} \, \texttt{X}_{\,ij} - p_{km} \right)^2 \ , \tag{5.5.1}$$

where F_{ij}^{km} are the weighting factors in the projection operation, p_{km} are the projection data, and σ_{km} are the errors in the projection data. If IEFF = 1, the input projection data uncertainties are used, but no errors are calculated for the reconstructed values. If IERR = 2, the input uncertainties are used, and the errors are calculated for the reconstructed values. If IERR has any other value, then input data errors are not used, i.e., $\sigma_{km} = 1$ is assumed, and errors for the reconstructed values are not calculated.

The generalized inverse method is a direct method, as opposed to the iterative methods, for minimizing equation (5.5.1). If H is defined as

$$H_{ij}^{km} = \frac{F_{ij}^{km}}{\sigma_{km}} \ , \tag{5.5.2}$$

equation (5.5.1) can be rewritten:

$$\chi^2(\texttt{X}) = \sum_{km} \left(\sum_{ij} H_{ij}^{km} \, \texttt{X}_{\,ij} - \frac{p_{km}}{\sigma_{km}} \right)^2 \ . \tag{5.5.3}$$

If H is considered a matrix with i and j contracted to the column index and k and m contracted to the row index, then H, the Penrose generalized inverse of H, provides the reconstructed solution

$$\texttt{X}_{ij} = \sum_{km} \hat{H}_{km}^{ij} \frac{p_{km}}{\sigma_{km}} \ , \tag{5.5.4}$$

which will minimize χ^2 function. The error array XE is evaluated using

$$\texttt{XE}_{ij} = \sqrt{\sum_{km} \left(\hat{H}_{km}^{ij} \right)^2} \ . \tag{5.5.5}$$

The magnitude of the ij and km indices of the matrix H in many applications are so large that the memory requirements for the generalized inverse method are the limiting factors of its usefulness.

5.6 Orthogonal Polynomial Expansion (MARR)

The method of orthogonal polynomial expansion parameterizes the distribution to be reconstructed by a set of coefficients of polynomials orthogonal on the unit circle. The transverse section is reconstructed using the statement

```
CALL MARR (X,NDEG)
```

where

`X` is the reconstructed transverse section;

`NDEG` is the degree of the polynomial expansion;

(cf. Example 13 of Section 9).

The maximum degree `NDEG` of the polynomial expansion is two less than the number of projection angles, which is the number of detectors. The subroutine `MARR` is a modification of the program ZHEAD (Version 2.0–12/10/71) supplied to us by R. Marr. (The algorithm is described in: R.B. Marr, J. Math. Anal. and Appl. **45**, 1974, pp. 357–374.)

The data are assumed to be $N(N-1)/2$ line integrals of the source distribution for the transverse section between N equally spaced points on the periphery of a circle. In this geometry we can still represent the projection data p_{km} with k as a projection bin index and m as an angle index. With each projection measurement p_{km} there are associated two quantities z_k and θ_m, where z_k is the perpendicular distance of the center of the unit circle to the projected ray and θ_m is the angle of the projection. Polynomial coefficients ($\beta_{nn'}$ and $\gamma_{nn'}$) are calculated using the equations

$$\beta_{on'} = 0 \ , \tag{5.6.1}$$

$$\gamma_{on'} = \frac{(2n'+1)}{N^2} \sum_{km} \sin\left[(2n'+1)\cos^{-1}(z_k)\right] p_{km} \tag{5.6.2}$$

$$\left\{ \begin{array}{c} \beta_{nn'} \\ \gamma_{nn'} \end{array} \right\} = \frac{2(n+2n'+1)}{N^2} \sum_{kn} \sin\left[(n+2n'+1)\cos^{-1}(z_k)\right] \left\{ \begin{array}{c} \sin \\ \cos \end{array} \right\} (n\theta_m) p_{km} \ . \tag{5.6.3}$$

The reconstruction can then be calculated at the center of each pixel using

$$\mathtt{X}_{ij} = \sum_{n=0}^{M} \sum_{n'=0}^{[(M-n)/2]} \left(\beta_{nn'} \sin n\phi_{ij} + \gamma_{nn'} \cos n\phi_{ij}\right) r_{ij}^n Q_{nn'}\left(r_{ij}^2\right) \ , \tag{5.6.4}$$

where $M =$ `NDEG` is the degree of the polynomial expansion, (r_{ij}, ϕ_{ij}) are the polar coordinates of the pixel (i,j) with the center of the unit circle at the origin, and $Q_{nn'}(t)$ is a polynomial in t of degree n' with the explicit representation

$$Q_{nn'}(t) = \sum_{j=0}^{n'} (-1)^{n'-j} \left(\begin{array}{c} n' \\ j \end{array} \right) \left(\begin{array}{c} n+n'+j \\ n' \end{array} \right) t^j \tag{5.6.5}$$

The subroutine `MARR` is the only algorithm that reconstructs data explicitly for a ring geometry. However, by reorganization of the chords into parallel- or fan-beam projections,

the same data can be used with other algorithms. Due to the ring geometry only the SETUP parameters: NDIMU, IGEOM, NANG, IMIT, NWORK, NFLOAT, ISTORE, IPRINT, and PWID have meaning (cf. Section 3). The others need not be assigned values.

Furthermore, while the definition of PWID is not different, the presence of detectors in an entire circle lends a different implication to the values it takes on. Remembering that NANG, the number of angles at which projections are collected (in the conventional sense) also represents the number of detectors comprising the ring, we can consider the case when the ring is inscribed in the NDIMU $\times$ NDIMU reconstruction region. Then the circumference of the circle in "projection bin widths" is just NANG so that

$$\texttt{NANG} = \pi \cdot \texttt{NDIMU} \cdot \texttt{PWID} \tag{5.6.6}$$

or

$$\texttt{PWID} = \frac{\texttt{NANG}}{\pi \cdot \texttt{NDIMU}} \ . \tag{5.6.7}$$

To use PWID larger than this simply introduces zeroes in the reconstruction array outside the circle of detectors. The user can "zoom" in on the center of the reconstruction region by choosing PWID smaller than this value.

Before using the subroutine MARR the user *should study* Section 3.4, which describes the subroutine GETUM, since the data input is in a different format than for the other reconstruction algorithms.

5.7 Attenuation Correction

Transverse section emission imaging with single photon or positron annihilation photons requires compensating for attenuation effects. For positron imaging, the user must first correct the measured projection data by multiplying the sampled projection data by $\exp[\int \mu(x)dx]$, where $\exp[\int \mu(x)dx]$ is the corresponding line integral of attenuation coefficients. The transverse section can then be reconstructed using one of the algorithms available in the RECLBL Library. For single photon imaging, the user can compensate for attenuation by using attenuation coefficients, which can be determined from a transmission experiment, or may be assumed constant over a convex region. Other methods of attenuation correction (cf. T.F. Budinger and G.T. Gullberg, in *Reconstruction Tomography in Diagnostic Radiology and Nuclear Medicine,* eds: M.M. Ter-Pogossian, et al., University Park Press, Baltimore, 1977, pp. 315–342) require either preprocessing the projection data or correcting the reconstructed by uncorrected transverse section using a correction matrix based on phantom studies.

The attenuation correction schemes used in the RECLBL Library assume that the projection data for the transverse section are the summation of pixel concentrations attenuated by a factor that is a function of the attenuation between the pixel and the edge of the object. The projections p_{km} are represented by

$$p_{km} = \sum_{ij} F_{ij}^{km} A_{ij}^{km} X_{ij} \ , \tag{5.7.1}$$

where F_{ij}^{km} are weighting factors and A_{ij}^{km} are the attenuation factors, which are calculated by the subroutines EVATN or EVATU. The reconstructed pixel values X_{ij} can be determined

by one of the iterative routines CONGR or GRADY using the appropriate projection and back-projection subroutines PPTA and BPTA, or PCDA and BCDA, or PRFA and BRFA.

The attenuation factors A_{ij}^{km} are evaluated from an array of attenuation coefficients by using the Fortran statement

```
CALL EVATN (B)
```

where

B is the array of attenuation coefficients.

The attenuation factors A_{ij}^{km} are evaluated using the equation

$$A_{ij}^{km} = e^{\sum_{i'j'} L_{i'j'}^{km} \mathtt{B}(i'j')}, \tag{5.7.2}$$

where the summation is taken over the pixels (i', j') in the projection ray (k, m) from the pixel (i, j) in the direction of the measured projection. $L_{i'j'}^{km}$ is the length of that portion of a line centered in the projection ray (k, m) within the pixel (i', j'). For an NDIMU $\times$ NDIMU array and NANG projection angles, it is necessary to evaluate $(\mathtt{NDIMU})^2 \cdot \mathtt{NANG}$ attenuation factors. Due to the large number, the attenuation factors A_{ij}^{km} are stored on the file LUNATN, which is determined by the user as one of the SETUP input parameters.

A method of reconstructing single photon data and correcting for attenuation using attenuation coefficients evaluated from a transmission experiment is outlined as follows:

1. Set parameter arrays IPAR and PAR and call SETUP (cf. Section 3).
2. Input projection data from a transmission experiment using subroutine GETUM.
3. Reconstruct the array B of attenuation by resetting appropriate parameters in the arrays IPAR and PAR and call SETUP again.
4. Evaluate the attenuation factors using the statement: CALL EVATN (B).
5. Reconstruct the array of isotope concentrations using one of the iterative routines CONGR or GRADY with one of the back-projection subroutines BPTA, BCDA, BRFA, and correspondingly one of the projection subroutines PPTA, PCDA, PRFA.

Examples 8, 9, and 10 in Section 9 illustrate this method of attenuation correction, which requires two reconstructions: one for the transmission data to obtain the correction factors and one for the emission data to get the final reconstruction.

For the case of an object that has a constant attenuation coefficient, the attenuation factors A_{ij}^{km} can be determined using the statement

```
CALL EVATU (B,XLEV,ATENL)
```

B is the transverse section that has not been corrected for attenuation;

XLEV is the approximate ratio of the concentration in the object to the background for use by the boundary search routine;

ATENL is the constant attenuation coefficient per pixel width.

The attenuation factors A_{ij}^{km} are evaluated using equation (5.7.2), where `B` (i', j') = `ATENL` for all pixels (i', j') within the boundary of the object. The boundary is determined using a search routine and the corresponding distribution of attenuation coefficients is displayed so that the user can change the parameter `XLEV` is necessary to obtain the true object shape. The user may desire to interact by varying `XLEV` until the desired object shape is obtained. The attenuation factors are stored on the file `LUNATN` and are read into memory in arrays of `NDIMU`^{2} words when needed.

A method of reconstructing single photon data and correcting for attenuation assuming a constant attenuation coefficient is outlined as follows:

1. Set parameter arrays `IPAR` and `PAR` and call `SETUP`.
2. Input single photon projection data from an emission study using subroutine `GETUM`.
3. Reconstruct the uncorrected transverse section `B` using one of the reconstruction algorithms.
4. Evaluate the attenuation factors using the statement `CALL EVATU (B,XLEV,ATENL)`.
5. Reconstruct the array of isotope concentrations using one of the iterative routines `CONGR` or `GRADY` with one of the back-projection subroutines `BPTA`, `BCDA`, `BRFA`, and correspondingly one of the projection subroutines `PPTA`, `PCDA`, `PRFA`.

Examples 11 and 12 in Section 9 give example programs utilizing this method of attenuation correction.

6 HOW TO USE THE LIBRARY

6.1 Summary of Reconstruction Procedures

To execute a reconstruction using the RECLBL Library, the user must prepare a main program with a declarative as well as an executable section. The declarative statements appear first in the sequence of program lines, but some are dependent on values or operations performed in the executable section. The set of declarative statements is

```
      DIMENSION B("NDIMU", "NDIMU"), ANG("NANG")
      COMMON//WORK("NWORK")
      COMMON/OUTCOM/LUNOUT,I80132
      DIMENSION IPAR(12),PAR(3)
      EQUIVALENCE (NDIMU ,IPAR( 1)),(ICIR  ,IPAR( 2)),(IGEOM ,IPAR( 3)),
     1            (NANG  ,IPAR( 4)),(MODANG,IPAR( 5)),(KDIMU ,IPAR( 6)),
     2            (IMIT  ,IPAR( 7)),(NWORK ,IPAR( 8)),(NFLOAT,IPAR( 9)),
     3            (ISTORE,IPAR(10)),(IPRINT,IPAR(11)),(LUNATN,IPAR(12)),
     4            (PWID  , PAR( 1)),(AXISU , PAR( 2)),(RFAN  , PAR( 3))
      EXTERNAL "BCK","PRJ","CNV","FIL"
```

The meaning of each statement will become clear from the description of the executable statements below. Variable names enclosed in quotation marks represent a numeric constant whose value is dependent on the value given to that variable. For example, if `NDIMU` = 64, then the statement represented by `DIMENSION B("NDIMU","NDIMU")` is `DIMENSION B(64,64)`. Of course, the variable names used are up to the user. The `EQUIVALENCE` statement has been included above for convenience, but is not a necessary declarative statement. In what follows, the elements of the `IPAR` and `PAR` arrays are referred to by the names given them in the `EQUIVALENCE` statement. Any additional declarative statements included by the user should carefully observe the reserved common block names given in Table 1.

There are three basic steps to executing a reconstruction with the RECLBL Library. The first step is to establish the values of the input parameters using the `SETUP` subroutine. The second step is to actually execute the reconstruction. The third step is to display and/or save the resulting image.

There are 17 parametric values to be decided upon before `SETUP` can be called. These can be broken down into four categories: those that describe the projection data that is to be input (7 parameters), those that describe the computational environment in which the reconstruction will be carried out (4 parameters), and those that control the output received from the library (3 parameters).

The first three parameters are `NDIMU`, `ICIR` and `PWID`. The value for `NDIMU` is usually related to the expected resolution in the image. If a pixel is chosen much smaller than the obtainable resolution, the library may calculate values that do not have physical significance and will cost the user memory space and computation time. If the pixel size is chosen much larger than the resolution, then resolution is lost. The parameter `ICIR` can save the user

memory space and execution time. If the object to be reconstructed fits entirely within a circular domain of diameter $\leq$ NDIMU, then ICIR should be set to 0. It is important to note that the entire object to be imaged *must* lie within the NDIMU $\times$ NDIMU array. The parameter PWID gives the width of a pixel in projection bin units. For the most efficient use of memory PWID $\approx$ KDIMU/NDIMU. In most cases KDIMU (cf. Section 3.2 and below) is fixed by the data collection geometry, and thus PWID and NDIMU show an inverse relationship. If one of these is known (or its desired value is known) the value of the other is also determined. For example, it has been shown (cf. R.H. Huesman, Phys. Med. Biol. **22**, 1977, pp. 511-521) that PWID should be at least 1.5 when using iterative reconstruction techniques. Thus, if KDIMU was 100 during data collection, NDIMU should be ≤ 66. Conversely, if the reconstruction phase is considered before data collection, then knowledge of PWID and NDIMU will dictate the required spatial sampling.

The second set of SETUP parameters describes how the projection data were collected for the number of angles, NANG. MODANG is a coded parameter that defines the initial angle, angular increment, and angular range of the NANG angles. Again, consideration of how the data will be reconstructed before their actual collection, may make the reconstruction phase easier. For MODANG $\neq 0$ or 1, the library generates angles equally spaced over π or 2π radians. If these angles have been used in data collection, the user may choose one of these values for MODANG and avoid having to supply the angles to SETUP (cf. Section 3.2 for MODANG options). Three reconstruction algorithms (i.e., CONVO, FILBK, BKFIL) rely on data having been collected at equally spaced angles in order to arrive at a correct solution, and in some cases the library will not execute the reconstruction unless this is the case. Thus, the user is advised to let the library generate the data collection angles whenever possible. IGEOM describes the geometry (i.e., fan, parallel, ring) in which the data were collected. KDIMU is the parameter describing spatial sampling and represents the number of projection bins in each projection. IMIT indicates whether to reconstruct emission or transmission data.

AXISU describes where in the projection array the axis of rotation is projected. In general, if the axis is to fall in the exact center of the projection array, AXISU = (KDIMU + 1)/2. It is very important to locate the AXIS of rotation as accurately as possible (to precision of less than 1 bin if possible), since even small errors may cause artifacts in the reconstruction. RFAN is the distance from the fan beam source to the center of rotation and is only meaningful under fan-beam geometry conditions.

The third group of SETUP parameters describes the computational environment in which the reconstruction is to be carried out. NWORK is the number of floating point variables that are provided as working space for the library. There must be a COMMON//WORK("NWORK") statement reflecting the value given NWORK in the declarative portion of the main program. (For example, if NWORK=100, then there must be a COMMON//WORK(100) statement somewhere in the main program.) The minimum value of this parameter may be determined by running a "storage size test" (described below). NFLOAT gives the number of computer words that actually make up one floating point variable in the array WORK. (For example, if the computer has 16-bit words and floating point variables are represented in 32 bits, then NFLOAT=2.) LUNOUT (in common block /OUTCOM/) and LUNATN are logical unit numbers for printed output and attenuation factor scratch storage, respectively. If LUNOUT is not given a value the library cannot communicate with the user.

The fourth group of SETUP parameters governs the execution of the reconstruction and the format of the output. ISTORE may be used to implement a "storage size test" to determine the amount of blank common that is needed for the reconstruction. IPRINT contains six 1-bit print flags that govern what output is received during the reconstruction. I80132 (word 2 of common block /OUTCOM/) is a flag that determines the width of the output (either 80 or 132 characters/line).

After *all* of the above parameters are given values, SETUP is called using the statement:
CALL SETUP (IPAR,PAR,ANG) .

The second phase of execution involves the actual reconstruction. Several considerations must be made at this juncture. First, if attenuation compensation is to be done during the reconstruction, attenuation factors must be calculated first using the routines EVATN or EVATU (cf. Section 5.7). Calculation of these factors usually involves a reconstruction as well, as in the case of using a transmission scan to correct for the attenuation in an emission scan. The next step is to determine what type of weighting model is to be used in the projection/back-projection operations and/or what type of filter or convolution function is to be used. The names of the routines (such as BCD, PCD, SHLO, BUTER, etc.) decided upon should be entered in the EXTERNAL statement in the declarative portion of the program (see above). In addition, they are included in the calling sequence of the reconstruction routine.

Another integral part in the actual execution of the reconstruction is input of the projection data via the user-supplied subroutine GETUM. The user must supply a routine that returns the projection data one angle at a time with each projection being KDIMU in length. If appropriate, the measurement uncertainty for each projection value must also be returned.

If an iterative reconstruction method is being employed (GRADY, ENTPY or CONGR), subroutine USER will be called between each iteration. USER is a subroutine that allows the user access to the reconstruction array between steps in the iterative process. While it is not necessary that a subroutine of that name be supplied (since a default routine is supplied with the package), the user may desire more information than that supplied by the default routine and may supply a replacement.

When the reconstruction subroutine returns control to the main program, the reconstructed image is stored in the NDIMU $\times$ NDIMU array stipulated in the calling sequence. It is now up to the user to display it or save it as desired. Two display routines are provided in the RECLBL Library. ARRAY is a subroutine that displays a 2-dimensional, gray-scale image using an overprinting scheme. The second display routine, XYGRF, makes 1-dimensional slices through the 2-dimensional array and displays them graphically on the output device. All other graphical display or output of the image onto peripheral storage devices is left to the user.

A summary of the steps for the preparation of a FORTRAN program to obtain a reconstruction from the RECLBL Library follows:

1. Set up declarative statements that are consistent with values given to SETUP parameters in the executable section including the common blocks /OUTCOM/ and blank common //, arrays IPAR and PAR, an array for the reconstruction and one for the projection angles (if MODANG=0 or 1).

2. Choose the back-projection or projection weighting model or filter or convolution function to be used in the reconstruction as well as other options provided by the recon-

struction routine. Be sure to declare these subroutines in an EXTERNAL statement.

3. Assign values to the IPAR and PAR variables and the two variables in /OUTCOM/.
4. Call SETUP.
5. Call the reconstruction subroutine.
6. Display and/or store the reconstructed image using ARRAY, XYGRF, or a user-supplied subroutine.

In addition the user must provide a subroutine GETUM for projection data input and, if appropriate, a subroutine USER.

6.2 Library Output

As the various subroutines in the library are executing, they communicate with the user via output on the file LUNOUT. While the output is intended to be self-explanatory, this section will attempt to clarify any ambiguities that remain and point out details that may be especially useful to the user.

The output obtained on any one run is, of course, a function of what subroutines are employed during the reconstruction and what print options are chosen by the user (cf. Section 3.3). However, many portions will be common to most runs. These are the ones that will be described here.

In most cases the first output received will be from the subroutine SETUP. Figure 13 shows sample output that was obtained during a test run. As with all user-called routines, SETUP alerts the user of its initiation by printing its name in large block letters. The first part of the listing contains the values input in the integer parameter array IPAR (cf. Section 3 for details of their meanings) and the second part contains the values for the floating point array PAR. The first column gives the index of the parameter on that line. The second column gives the actual value input and the third column gives a description of what it means. Between the second and third columns flags will occasionally appear. They will be the letters NA or ER. NA means the value for this parameter is not applicable to the situation as defined by parameters above it. For example, in Figure 13 the third line under the PAR heading the reader will see NA between columns 2 and 3 (labeled A). This indicates that the distance of the fan source from the center of rotation is not necessary since IPAR(3), the geometry flag, indicates the reconstruction will use parallel-beam geometry. If the ER flag is seen, it indicates that a fatal error has been detected in the input to SETUP. At the end of the SETUP output an error message will be printed telling how many errors were detected. The lines containing the ER flag indicate the input values that were in error.

The next portion of the SETUP output will be a few lines similar to those labeled B in the figure (assuming the appropriate print option has been indicated). These are lines output by the memory management routine that tell the user how much blank common is in use at the time the line is printed. The multiple lines indicate that several requests for memory allocation have been generated by the library. The first number in the line is the decimal number of floating point variables in use and the number enclosed in parentheses is its octal equivalent.

Figure 13: Example of library output.

The next line of output, labeled C, indicates to the user which portion of the supplied projection array will be used during the reconstruction. In this case the user has supplied 100 bins when only 52 are necessary. Since the axis of rotation has been placed 50.5 projection bins from the beginning of the array, the library uses 26 on either side of it or bins 25 through 76.

Line D indicates how many bins will actually be used as the projection array during the reconstruction and how many, if any, are assumed to be zero in value. In this case a sufficient number of bins has been supplied. However, if the user's projection array is found not to be long enough to cover the entire reconstruction region, the library will add bins on either side until the projection array is long enough. These bins are assumed to have the value zero for all projection angles.

Line E is the closing line of the `SETUP` output (and several other routines). It informs the user of the largest amount of blank common that has been in use up to that point in the program. If one wishes to know the maximum amount that is ever in use, simply find the last time this message is printed during the job. `SETUP` indicates it has completed execution by printing `END SETUP` in large block letters.

In the job shown in Figure 13 a phantom was generated using the subroutine `PHAN` (cf. Section 7). The output from it is labeled F. All of the input values are printed. In this particular case the phantom was generated in a 64×64 pixel array. When a pixel was found to fall only partially inside one of the ellipses comprising the phantom, it was divided into 10×10 (or 100 total) "pixelettes" in order to determine what percentage of the full value should be assigned to that pixel. There are 1.33 pixels/projection bin. The entire phantom is composed of 4 ellipses. If the pixelized array is considered to have (x,y) coordinates with the origin in the exact center of the array, then the centers of the four ellipses are at the points (0,0), (0,-13.33), (13.33,0) and (-13.33,0). These values were arrived at by applying the conversion factor of 1.33 to the actual input values, which are in terms of projection bin width. The center points in units of projection bins are listed above the corresponding converted values that are in parentheses.

The major and minor axes of the ellipses are listed in a similar fashion as the center points under the columns headed A and B, respectively. The angle that the major axis makes with the x-axis is listed in the `PHI` column and the density assigned to each pixel in an ellipse is listed under `DENS`.

In summary, the first ellipse is centered at the origin (0,0); it has major and minor axes of 53.33 pixels (i.e., it is a circle); its major axis is colinear with the x-axis; and each pixel completely inside it has a value of 2.81. The fourth ellipse, however, is centered at (-13.33,0) in the pixel array; has major and minor axes of 18.67 and 13.33 pixels, respectively; its major axis makes an angle of 1.57 ($\pi/2$) radians with the x-axis and each pixel inside it has a value of -2.25.

The beginning of the next section of output is delimited by the large letters `CONVO`. This indicates that the user has called subroutine `CONVO` and is about to perform a convolution reconstruction. The first few lines of output from any of the major reconstruction routines (`BKFIL`, `CONGR`, `CONVO`, `ENTPY`, `FILBK`, `GRADY`, `GVERS`, or `MARR`) are a list of what the input values were and what the subroutine has interpreted them to mean. In this case the only input parameter to be interpreted is `IERR`. Its value was passed as 0 and `CONVO` interprets that to mean that the user does not want errors calculated for the reconstructed values. Next

the algorithm must determine if the back-projection and projection (or filter or convolution) routines chosen by the user are compatible. There is a single routine in the package that is used to do this and typical output from it is labeled G.

This particular output indicates that the arguments specified in the call to `CONVO` were a back-projection routine and a convolution routine, which is correct for `CONVO`. Any other combination would be an error and an error message would be printed. For the other reconstruction methods, the correct combinations are:

1. `BKFIL` and `FILBK` must have a back-projection and a filter routine,

2. `CONGR`, `ENTPY`, `GRADY`, and `GVERS` must have a back-projection and a projection routine.

3. The `MARR` reconstruction routine does not use back-projection, projection convolution or filter routines.

The column labeled "ray weighting" explains what model is employed in the projection and back-projection operations. In this case the back-projection value assigned to a pixel is proportional to the length of the line emanating from the center of a projection bin that intersects that pixel. Other entries that may appear in this column are `DELTA FUNCTION`, `CONCAVE DISK`, `UNIFORM SQUARE` or `INTERPOLATION` (cf. Section 4 for full description of what these models imply). Since the convolution routine does not perform either a back-projection or projection operation the "ray weighting" column contains an `N/A` (not applicable) entry.

The next column indicates whether the routine in question uses attenuation factors in calculating projection/back-projection values. In this case neither does. If attenuation factors are to be used in the projection/back-projection operations, it is important that the user employ either of the subroutines `EVATN` or `EVATU` to store the coefficients on the file `LUNATN` before calling the main reconstruction routine. If this has not been done an error message will be generated and execution terminated.

The last column indicates whether the routine assumes fan-beam or parallel-beam geometry. In this case, both assume parallel-beam. It is important that both the back-projection and projection (or convolution or filter) routines make similar assumptions and use the same model in order that the reconstruction be correct. If this is not the case, an error message is generated. The output just described may be consulted to locate the exact cause of the error if an error message is printed.

In addition, certain reconstruction methods or options require that special purpose routines be used with them. If the user attempts to use a routine that cannot fulfill the task requested, a self-explanatory error message will be printed after the just described output. All of the limitations that must be observed with each reconstruction method are given in the appropriate parts of Section 5.

The remaining output from `CONVO` consists of lines from the memory management routine already described. It ends with the line informing the user of the largest amount of blank common storage used to that point in the program (also described above). Termination of the subroutine `CONVO` is indicated by the large block letter message `END CONVO`.

Output that was not encountered in this run, but may be seen in others (by selecting more print options and/or using other reconstruction routines) include a printout of the projection data and their uncertainties as they are input from the user-supplied subroutine

GETUM, a printout of the values of the convolution function (for CONVO), the filter function in Fourier space (for FILBK or BKFIL) or the Lagrange multipliers and their gradient function (for ENTPY). When using the iterative reconstruction methods, the subroutine USER is called between successive iterations and output may be generated from that routine. (The default subroutine USER supplied with the package prints the value of the function being optimized and the iteration number.) All of these are accompanied by self-explanatory headings and messages.

6.3 Library Display Routines

The RECLBL Library provides the two display routines ARRAY and XYGRF. The subroutine ARRAY produces a two-dimensional gray-scale display using overprinting on the output device. The subroutine XYGRF produces a plot of intensities for one-dimensional slices through a two-dimensional array.

Any two-dimensional array may be displayed on an output device that has an overprinting capability using the statement

```
CALL ARRAY(B,NXN) ,
```

where

B is the array to be imaged;

NXN is the dimension of the array.

The subroutine is coded assuming the printing device has ten characters per inch and six lines per inch. The subroutine interpolates vertically between pixels in order that the array appear square. The format statements use carriage control statements, which are used by the line printer at Lawrence Berkeley Laboratory for implementing the overprinting capability. These statements assume that the carriage control is affected after the line has been printed. The subroutine is coded so that this gives a gray-scale image with 21 levels of gray (cf. I.D.G. McLeod, IEEE Trans. Computers **C-19**, 1970, pp. 160-162).

Cross-sectional plots of a two-dimensional array are graphically displayed using the statement

```
CALL XYGRF (B,N,NP,BMAX,BMIN,IXY,ICOR,IL,IU)
```

where

B is a square array from which plots are generated;

N is the dimension of B (i.e., B(N,N));

NP is the number of cross-sectional plots;

BMAX is the maximum value for the plot; if BMAX = 999999., the maximum will be determined from the data;

BMIN is the minimum value for the plot; if BMIN = 999999., the minimum will be determined from the data;

`IXY` determines the direction of the cross section;
if `IXY` = 0, the cross section is parallel to the x-axis;
if `IXY` = 1, the cross section is parallel to the y-axis;

`ICOR` is an array of x- or y-intercepts that determines the location of the cross section;

`IL` is the lower coordinate for the plot;

`IU` is the upper coordinate for the plot.

The maximum number of plots that can be displayed on the same graph is 10. If `NP`=1, then the cross-sectional plot will appear as a histogram where the spaces between the axis and the functional value are filled in with the symbol `X`. For `NP`>1, the functional value for each cross section will be given a different symbol and only one symbol will be plotted for each function at each coordinate value. It is assured that the array is stored such that the array value `B(I,J)` corresponds to the value at the point (I,J) relative to a standard (x,y) coordinate system with (1,1) in the bottom left corner of the first quandrant of the xy-plane. Example 7 uses the subroutine `XYGRF` in the subroutine `USER` to graph one cross-sectional plot parallel to the x-axis for each iteration.

6.4 Error Handling

During execution of a reconstruction subroutine, if errors that result from inconsistent user requests, omission of input parameters, or inappropriate data input are detected, a self-explanatory error message is printed on the file `LUNOUT`. In addition, an error number is printed in large block letters. If the detecting routine is user-called, its name is also printed in large block letters. If the error is nonfatal, the program will continue; but if it is fatal then `STOP` is printed in large block letters on `LUNOUT` and program execution is terminated.

There are a small number of errors that under normal circumstances should never occur. However, user coding errors or yet-undetected library errors may result in destruction of portions of the executable code or internal variables causing a program stop with a message to the user of a `SYSTEM ERROR` rather than just a plain `ERROR`. If this should occur, the user should carefully check the main program and any user-supplied subroutines to be sure that no addressing errors (such as incorrect array subscripts) are the cause. Ample documentation of unresolved problems should be sent to the Donner Laboratory Research Medicine Group (cf. Section 1.4). Error messages, their identification numbers, the routine that detects them and whether they are fatal or not appear in Table 3.

Table 3: RECLBL Library Error Messages.

Error No.	Routine	Fatal	Error Message
1		Y	`SETUP` must be called before ————————.
2	`SETUP`	Y	Errors in `IPAR` or `PAR` arrays.
3	`MEMST`	N	The amount of space in common block `WORK` is larger than the amount allocated by the user. This run is now a storage size test, no reconstruction will be executed.
4	`STPTR`	Y	The rotation axis (`AXISU` = XXXXX.XX) does not project into the reconstruction array. The projection is XXXXX bins long.
5	`STPTR`	Y	The fan source at a distance of XX.XXXEXXX is inside the reconstruction array. The distance from the fan source to the center of rotation must be at least XX.XXXEXXX.
6	`STPTR`	Y	The reconstruction array does not project into any user projection bins.
7	`RCHEK`	Y	These are inconsistent. Explanation: The combination of back-projector and projector/filter/convolver chosen by the user is inconsistent. Consult listing just above error message for description of the choices that were specified.
8	`RCHEK`	Y	Due to lack of appropriate filters, `BKFIL` cannot execute fan-beam reconstructions at the present time.
9	`RCHEK`	Y	When using fan-beam geometry and the subroutine `FILBK`, one of the back-projection subroutines `CDF2`, `BPTF2`, `BRFF2`, should be used.
10	`RCHEK`	Y	Should use `BCDF2`, `BPTF2`, `BRFF2` only with the subroutine `FILBK`.
11	`RCHEK`	Y	Cannot use attenuation projection and back-projection subroutines with the subroutine `ENTPY`.
12	`RCHEK`	Y	The requested back-projection subroutine will not calculate errors for convolution reconstructions.
13	`RCHEK`	Y	For this weighting model pixels and projection bins must be the same size (`PWID` = `PAR(3)` = 1.0).
14	`RCHEK`	Y	These subroutines are inconsistent with the fan-beam parameters seen by `SETUP`.
15	`RCHEK`	Y	Attempted call of a projection or back-projection subroutine requiring attenuation factors before the factors were evaluated.

Table 3. Continued.

Error No.	Routine	Fatal	Error Message
16	RCHEK	Y	For convolution and Fourier reconstruction methods, projection angles must be equally spaced over at least π radians for parallel-beam geometry. To ensure this MODANG = IPAR(4) must not be 0 or 1 in the call to SETUP.
17	RCHEK	Y	For convolution, Fourier, and entropy reconstruction methods, projection angles must be equally spaced over 2π radians for fan-beam geometry. To ensure this MODANG = IPAR(4) must be 3, -3, 5 or -5 in the call to SETUP.
18	RCHEK	Y	Must use BINF when performing convolution on fan-beam data.
19	RCHEK, BJECT, PJECT	Y	Must use the MARR reconstruction algorithm on ring geometry data.
20	CONGR, GRADY	Y	The number of steps NSTP = XXX is less than 0.
21	FMCG	Y	There is an error in the gradient calculated by the subroutine DULFC.
22	FMCG	N	Convergence was not obtained in the limit number of iterations.
23	FMCG	Y	The linear search technique indicates it is likely that there exists no minimum.
24	FILBK	Y	The dimension of the reconstruction array, NDIMU = IPAR(1), must be even for subroutine FILBK.
25	MARR	Y	The MARR reconstruction method can only be used for ring geometry (SETUP input parameter IGEOM = IPAR(3) = 3).
26	MARR	Y	The number of crystals NXTAL = XXX is not even.
27	MARR	N	A ring of XXX detectors and pixels that are XX.XXX the size of one detector implies that the entire ring will be inscribed in a square XXX pixels on a side. Using an array of XXX pixels on a side will only result in zeros outside a radius of XX.XXX.
28	MARR	N	The maximum degree of the polynomials for a ring of XXX detectors is XXX. The reconstructed values will be computed to this degree.
29	BJECT	Y	The back-projection subroutine is inconsistent with the fan-beam parameters seen by SETUP.

Table 3. Continued.

Error No.	Routine	Fatal	Error Message
30	BJECT	Y	Attempted call of a back-projection subroutine that uses attenuation factors before the factors were evaluated.
31	CBARP	Y	NREPS*NBARS = XXX is greater than 100.
32	EVATU	Y	Zero range in reconstructed array. No attenuation factors calculated.
33	EVATU	Y	Target to nontarget must be greater than 1. The value was .XXXEXXX.
34	GETUM	Y	A data input subroutine named GETUM must be supplied by the user.
35	PHANL	Y	There is a parameter error in the call to subroutine PHANL. (This is followed by a self-explanatory description of the rules for parameter values and a list of the values input.)
36	PHANL	N	Warning . . . negative source (or attenuation) detected during generation of PHANL data.
37	PJECT	Y	The subroutine PJECT cannot be called during the execution of FILBK.
38	PJECT	Y	The projection subroutine is inconsistent with the fan-beam parameters seen by SETUP.
39	PJECT	Y	Attempted call of a projection subroutine that uses attenuation factors before the factors were evaluated.
40	XYGRF	Y	Input parameter IXY = XXX is not set properly.
System Error No.			
1	FTATN	Y	No message–just SYSTEM ERROR 1 in large letters.
2	MEMST	Y	XXXXXX is not a valid pointer.
3	MEMST	Y	LPTR is negative, but not -ISET (-XXXX).
4	PHAN	Y	No message–just SYSTEM ERROR 4 in large letters.
5	STATN	Y	No message–just SYSTEM ERROR 5 in large letters.

7 GENERATION OF PHANTOMS AND PROJECTION DATA

The RECLBL Library has the ability to generate images of phantom objects and projection data that could theoretically be collected from them. If the user wishes to experiment with different reconstruction methods, he may employ the following subroutines to generate a variety of phantoms and corresponding projection data.

The general phantom generating subroutines of the RECLBL Library are `PHAN` and `PHANL`. These two subroutines have much in common and will therefore be described together. `PHAN` will generate a pixelized image of any phantom composed of ellipses and rectangles, and `PHANL` will generate analytic projections of any phantom that `PHAN` can generate. For purposes of simulating emission data with `PHANL`, each ellipse or rectangle can be specified as a source or attenuator. The value of each bin in the projection will be a line integral of source activity with attenuation included if appropriate. The amount of attenuation is a function of the density (attenuation coefficient) of the attenuator and the length of attenuating material that the source radiation must traverse on its path to the detector.

The phantom/phantom data generated by the routines `PHAN`/`PHANL` are from a superposition of rectangles and ellipses whose size and orientation are defined by the user in arrays that are arguments of these subroutines. The calling sequence for `PHAN` is

```
CALL PHAN (N,INTG,ITYPE,DENS,X,Y,A,B,PHI,BB,NBB,PIXW)
```

where

`N` is the total number of ellipses and rectangles that make up the phantom.

`INTG` is an integration factor. `PHAN` generates the phantom in a discrete pixelized space. When the edges of an ellipse/rectangle do not coincide exactly with the boundary of a pixel, `PHAN` gives that pixel a fractional part of the full value according to what portion of the pixel lies inside the phantom. To do this, `PHAN` divides each border pixel into `INTG` × `INTG` "pixelettes," each of which is tested for "insideness" and the final value given to the pixel is the full value times the fraction of pixelettes found inside the phantom. This border "integration" gives the image a smoother appearance. We have found a reasonable value for this is 10.

`ITYPE` is an array that describes the ellipses/rectangles. For the $\texttt{I}^{th}$ ellipse/rectangle, `ITYPE(I)` can take on the following values:

1 for a source ellipse

2 for a source rectangle

`DENS` is an array that describes the density (or attenuation coefficient) of each shape. For transmission this is in units of inverse projection bin width, and for emission in units of inverse bin width squared.

`X,Y` are two arrays that describe the coordinates of the center of each ellipse/rectangle relative to the center of the image array. These are in units of projection bin width.

`A,B` are two arrays that describe the major and minor axes, respectively, of the ellipses or the lengths of the sides of the rectangles.

`PHI` is an array giving the angle in radians that the major axis (`A` above) makes with the x-axis.

`BB` is the array in which the image is generated.

`NBB` is the dimension of `BB`, which is assumed square.

`PIXW` is the pixel width in units of bin width. Since `X`, `Y`, `A` and `B` are all given in units of bin width, this is the conversion factor from bins to pixels. The sign of `PIXW` is used to normalize the total number of "counts" in the image so that it can be directly compared to a reconstructed image. It should be positive (+) for transmission and negative (–) for emission.

The calling sequence for `PHANL` is

```
CALL PHANL (N,ITYPE,DENS,X,Y,A,B,PHI,P,M)
```

where

`N` is the total number of ellipses and rectangles that make up the phantom.

`ITYPE` is an array that describes the ellipses/rectangles. For the I^{th} ellipse/rectangle `ITYPE(I)` can take on the following values:

- 1 for a source ellipse
- 2 for a source rectangle
- -1 for an attenuating ellipse
- -2 for an attenuating rectangle.

`DENS` is an array that describes the density (or attenuation coefficient) of each shape. For transmission data this is in units of inverse projection bin width, and for emission in units of inverse bin width squared.

`X,Y` are two arrays similar to `ITYPE` that describe the coordinates of the center of each ellipse/rectangle. These are in units of projection bin width.

`A,B` are two arrays that describe the major and minor axes, respectively, of the ellipses or the lengths of the sides of the rectangle.

`PHI` is an array giving the angle in radians that the major axis (`A` above) makes with the x-axis.

`P` is an array at least `KDIMU` (`SETUP` input-parameter `IPAR(6)`) long in which `PHANL` generates the projection.

`M` is the angle index at which the projection is calculated (the angles are either supplied to or generated by `SETUP`).

Arrays `X`, `Y`, `A`, `B`, `PHI`, `DENS`, and `ITYPE` in both routines must be dimensioned at least as large as the number of ellipses/rectangles in the phantom. The I^{th} entry in each of these arrays is the appropriate parameter for the I^{th} shape. Only `PHANL` uses parameters from

SETUP so phantom development can be done using PHAN without calling SETUP. PHANL and PHAN are conveniently used together since reconstruction of data generated by PHANL may be compared to the corresponding phantom generated by PHAN. However, it is *not* necessary to create a pixelized phantom image using PHAN in order that PHANL be able to generate projection data. Demonstrations of the use of these two routines are given in Section 9.

In addition to the general phantom generators, two special phantom generators in the RECLBL Library are PIE and CBARP. PIE generates an image of a circle containing equal-sized, alternating black and white sectors. CBARP generates a circular bar phantom. This is a phantom consisting of alternating black and white bars superimposed on a circular domain. The bar pattern is generated such that the last bar in each repetition of the pattern is 1/2 as wide as the previous bar, 1/3 as wide as the third last, etc. (cf. Example 16, Section 9). The bar pattern may be repeated as many times as desired across the circle with the limitation that the number of bars in the pattern times the number of repetitions of the pattern is less than 100. Projection data are obtained from these phantom generators using subroutine PJECT. PJECT generates projection data for any phantom that can be represented in a pixelized array. It gives the added freedom of allowing the user to choose which model of activity distribution he wishes to employ during the projection operation (see Section 4 for the choices available). However, it should be noted that PHANL comes the closest to mimicking the data that would be collected in a physical situation. (This can be closely approximated by use of the projector PLL in conjunction with PJECT.)

The calling sequence for PIE is

```
CALL PIE (B1,N,R,X1,Y1,Z,INTFAC,NSLIPI,ISTART)
```

where

B1 is the array where the phantom is generated.

N is the dimension of the square array B1.

R is the radius of the circle (in pixel units).

X1,Y1 are the coordinates of the center of the circle with respect to the center of the array (in pixel units).

Z is the amplitude value to be assigned to pixels that are completely in a black section.

INTFAC is an integration factor (see description of INTG under PHAN above).

NSLIPI is the total number of slices in half of the pie (or the number of black slices in the whole pie).

ISTART is an indicator of the color of the first slice (counterclockwise from straight up). If it is zero, the first section is white; otherwise it is black.

The calling sequence for CBARP is

```
CALL CBARP (B1,N,R,X1,Y1,Z,INTFAC,NBAR,NREPS,IDIREC)
```

The meaning of the parameters is the same as for PIE with the following exceptions:

NBAR is the total number of bars in one repetition of the bar pattern.

NREPS is the number of repetitions of the pattern across the entire circle.

IDIREC is the direction the bars should be in:

0 for horizontal bars,

1 for vertical bars,

2 for annular rings.

The calling sequence for PJECT is

```
CALL PJECT (B,P,M,PRJ)
```

where

B is the array containing the image from which the projection is to be calculated.

P is the array at least KDIMU (SETUP input parameter IPAR(6)) long in which the projection is generated.

M the index of the angle at which the projection is to be taken (the angles are either supplied to or generated by SETUP).

PRJ the name of the projection subroutine to be used in the projection operation. It must be declared in an EXTERNAL statement in the calling routine.

PJECT depends on values supplied to SETUP in its computations, and thus, SETUP must be called before PJECT can be used.

In Section 9, Examples 5, 6 and 17 use PIE to generate a phantom and Examples 5 and 6 use PJECT to get projection data from it. Example 16 shows the use of CBARP.

8 STORAGE REQUIREMENTS AND TIMING

8.1 Storage Requirements

The amount of storage that must be allocated in blank common is a function of the SETUP parameters NDIMU, NANG and KDIMU. The terms that make up the function vary depending on the reconstruction algorithm, the back-projection/projection weighting model, and options selected for the reconstruction via arguments in the calling statement or other SETUP parameters. It is difficult to ascertain the exact amount of storage necessary for any given program without running a storage size test (cf. Sections 3.3 and 6.1).

The expressions below are given as functions of NDIMU, NANG, KDIMU, and KDIM. KDIM may be determined from NDIMU by the relations

$$\begin{aligned} \texttt{KDIM} &= \texttt{NDIMU} + 4 \quad \text{if} \quad \texttt{ICIR} = 0, \\ \texttt{KDIM} &= \texttt{INT}(\sqrt{2}\ \texttt{NDIMU}) + 4 \quad \text{if} \quad \texttt{ICIR} \neq 0. \end{aligned}$$

When the storage requirements are dependent on values input to SETUP or the reconstruction routine, the appropriate portion of the equation is in the form of a logical statement. If the statement is true for the case being considered, then the preceding factor is included in the calculation; otherwise the value following the word "else" is used. For example, the expression

$$M = [\texttt{NDIMU}^2 \times (\pi/4 \text{ if } \texttt{ICIR} = 0, \text{ else } 1)] + [\texttt{KDIM} \cdot \texttt{NANG} \times (1 \text{ if } \texttt{IERR} = 1, \text{ else } 0)]$$

would be interpreted to mean M is equal to $\texttt{NDIM}^2$ times $\pi/4$ if ICIR=0 plus KDIM · NANG if IERR=1. If IERR≠1, the last term is zero and if ICIR≠0 the $\pi/4$ term becomes 1 in the calculation.

All reconstruction routines require that SETUP be called to initialize the values of all input parameters and to set up storage that is used by all algorithms. This amount is

$$\begin{aligned} S = {} & 3\,\texttt{NANG} + 2\,\texttt{KDIMU} + 2\,\texttt{NDIMU} + [\texttt{NDIMU}^2 \times (\pi/4 \text{ if } \texttt{ICIR} = 0, \text{ else } 1) \\ & \times (1 \text{ if using attenuation correction, else } 0)] + [62\,\texttt{NANG} \times \\ & (1 \text{ if using ray factors, else0})] \end{aligned}$$

and will be referred to below.

The storage requirements M for the nine reconstruction algorithms are as follows:

BJECT

$$M = \texttt{KDIM} + S$$

BKFIL

$$M = 3 \times 2^{(P-1)} + S$$

where

$$P = \text{smallest integer } \geq \log_2(2\ \texttt{KDIM})$$

CONGR

$$\begin{aligned} M &= 2\,\texttt{KDIM} + [3\,\texttt{NDIMU}^{\,2} \times (\pi/4 \text{ if } \texttt{ICIR} = 0,\ \text{else } 1)] \\ &\quad + [\texttt{KDIM} \cdot \texttt{NANG} \times (1 \text{ if } \texttt{IERR} = 1,\ \text{else } 0)] \\ &\quad + [\texttt{NDIMU}^2 \times (\pi/4 \text{ if } \texttt{ICIR} = 0,\ \text{else } 1) \times (1 \text{ if } \texttt{IRLX} = 1,\ \text{else } 0)] + S \end{aligned}$$

CONVO

$$\begin{aligned} M &= 4\,\texttt{KDIM} + [\texttt{KDIM} \times (1 \text{ if } \texttt{IGEOM} = 1 \text{ or } 2,\ \text{else } 0)] \\ &\quad + [6\,\texttt{KDIM} \times (1 \text{ if } \texttt{IERR} = 1,\ \text{else } 0)] + S \end{aligned}$$

ENTPY

$$M = 6\,\texttt{NANG} \cdot \texttt{KDIM} + \texttt{KDIM} + [\texttt{NDIMU}^2 \times (\pi/4 \text{ if } \texttt{ICIR} = 0,\ \text{else } 1)] + S$$

FILBK

$$M = 2^P \times (2^P + 2 + \sqrt{2}) + 2\,\texttt{KDIMU} + (3\texttt{NANG}) + 4$$

where

$$P = \text{smallest integer } \geq \log_2(2 \times \texttt{NDIMU})$$

GRADY

$$\begin{aligned} M &= 2\,\texttt{KDIM} + [2\,\texttt{NDIMU}^{\,2} \times (\pi/4 \text{ if } \texttt{ICIR} = 0,\ \text{else } 1)] \\ &\quad + [\texttt{KDIM} \cdot \texttt{NANG} \times (1 \text{ if } \texttt{IERR} = 1,\ \text{else } 0)] \\ &\quad + [\texttt{NDIMU}^2 \times (\pi/4 \text{ if } \texttt{ICIR} = 0,\ \text{else } 1) \times (1 \text{ if } \texttt{IRLX} = 1,\ \text{else } 0)] + S \end{aligned}$$

GVERS

$$\begin{aligned} M &= \{\texttt{NDIMU}^2[\texttt{NDIMU}^2 \times (\pi/4 \text{ if } \texttt{ICIR} = 0,\ \text{else } 1) \\ &\quad + (\texttt{KDIM} \cdot \texttt{NANG}) + 2] \times (\pi/4 \text{ if } \texttt{ICIR} = 0,\ \text{else } 1)\} + [\texttt{KDIM}(\texttt{NANG} + 1)] \\ &\quad + [\texttt{KDIM}(\texttt{NANG} + 1) \times (1 \text{ if } \texttt{IERR} = 1,\ \text{else } 0)] + S \end{aligned}$$

MARR

$$M = \texttt{NANG}/2 \times (25 + \texttt{NANG}) - 4$$

Table 4 shows a comparison of the estimates computed from the above expressions with the actual work space necessary for that reconstruction. The pertinent parameter values are as follows: NDIMU=32, ICIR=0, NANG=36, KDIMU=32, IGEOM=0 (except for MARR where IGEOM=3), IERR=0, IRLX=1 and using the projector/back-projector pair PCD, BCD.

Table 4: Comparison of estimates computed with actual work space necessary for reconstruction.

Reconstruction Subroutine	Actual	Estimate
BJECT	272	272
BKFIL	428	428
CONGR	3,556	3,525
CONVO	380	380
ENTPY	8,860	8,852
FILBK	4,488	4,496
GRADY	2,744	2,720
GVERS	1,714,888	1,692,296
MARR	1,094	1,094

8.2 Algorithm Timing

Table 5 gives the central processor times for reconstructing a circular array from 36 projection angles (NDIMU=32, NANG=36, ICIR=0) for various combinations of reconstruction algorithm and back-projection subroutine. These simulations were compiled using the MNF compiler and run on the CDC 7600 computer at the Lawrence Berkeley Laboratory. The times should be used only as a relative measure and not as an absolute measure of algorithm speed because speed is a function of the compiler and the particular computer used. We have found that during peak usage the computation time increases due to increased central processor overhead; thus the times listed in table 5 are only approximate values.

The speed of the algorithms BJECT, CONGR, ENTPY, and GRADY is determined entirely by the speed of the projection and back-projection subroutines. A major part of the effort of developing the RECLBL Library has been spent in optimizing the code for these subroutines. The time to project and back-project is a function of the size of the reconstruction array, the number of projections, the weighting scheme, and the type of geometry. This time is linear in both the number of elements to reconstruct and the number of projection angles.

The fan-beam projection and back-projection subroutines require the longest computation times. For example, in the CONGR and GRADY algorithms the time for fan-beam geometry with flat detector is increased more than a factor of ten over parallel-beam geometry: time for BCD is 3 sec as compared with the time for BCDF (flat detector) of 30 sec; time for BRF is 4 sec as compared with the time for BRFF (flat detector) of 69 sec. The increase in computation time of fan-beam over parallel-beam routines is less when using fan-beam geometry with a curved detector.

The data in table 5 for the algorithms CONGR and GRADY were obtained using the parameters IRLX=1, IERR=0, and NSTEP=10. Other tests were done for various combinations for IRLX and IERR but the computation times were the same. This is what one would expect when algorithm speed is primarily determined by the speed of the projection and back-projection subroutines. For these algorithms one projection and one back-projection must

Table 5: Central Processor times in seconds for constructing a 32 × 32 circular array from 36 projection angles for various combinations of reconstruction algorithms and back-projection subroutines.

	BJECT	BKFIL	CONGR for 10 Iterations	CONVO	ENTPY for 10 Iterations	FILBK	GRADY for 10 Iterations
Parallel Beam							
BPT	0.19	0.45	1.43	0.44	7.76	0.79	1.42
BCD	0.25	0.53	2.65	0.53	16.24	1.06	2.73
BIN	0.26	0.54		0.52		1.14	
BLL	0.36	0.64	5.41	0.62	31.70	1.52	5.53
BRF	0.32	0.61	4.28	0.63	23.98	1.38	4.32
Fan Beam Curved Detector							
BPTF	0.55		9.51		53.45		9.88
BPTF2	0.55					2.68	
BCDF	0.94		17.75		102.60		17.96
BCDF2	1.00					4.80	
BINF	0.54			0.91			
BRFF	2.57		52.87		310.20		54.01
BRFF2	2.58					13.28	
Fan Beam Flat Detector							
BPTF	0.27		3.29		17.98		3.25
BPTF2	0.47					2.03	
BCDF	1.47		30.16		169.80		30.19
BCDF2	1.54					8.36	
BINF	0.36			0.76			
BRFF	3.26		68.83		388.10		68.48
BRFF2	3.35					18.31	
Attenuation Correction							
BPTA	0.23		2.21				2.25
BCDA	0.29		3.80				3.72
BRFA	0.37		5.37				5.35
Other Reconstruction Subroutines							
GVERS	*						
MARR	2.23						
Special Routines for Attenuation Correction							
EVATN	2.94						
EVATU	3.02						

* Timing could not be measured since the memory required to reconstruct a 32 × 32 array is larger than 170 K, which is the memory size of the CDC 7600 computer at the Lawrence Berkeley Laboratory.

be done for each iteration. The reconstruction subroutine `ENTPY` required longer central processor time than any of the other algorithms. The reason for this is that the function optimized (equation (5.4.5) of Section 5) is not quadratic; thus more than two back-projection and projection operations are required for each iteration.

Attenuation correction increases computation time in `CONGR` and `GRADY` (`BRF` — 4.3 sec as compared with `BRFA` — 5.4 sec). These timings were measured after the attenuation factors were calculated. Notice that the time for calculation of the attenuation factors (`EVATN` — 2.9 sec and `EVATU` — 3.0 sec) is significant.

The speed of the algorithms `BKFIL`, `CONVO`, and `FILBK` is determined to a large extent by the speed of the back-projection subroutines. For these algorithms the filter or convolution operation also takes considerable time. `BKFIL`, `CONVO`, and `FILBK` require a single back-projection. However, `FILBK` requires longer computation time than `BKFIL` and `CONVO` since the data is back-projected into an array that is four times the size of the user's array.

9 EXAMPLES OF LIBRARY USE

The sample programs illustrated in this section were run on a Sun SPARCstation computer at the Lawrence Berkeley Laboratory. These programs show how the user can set up his main program in order to use the RECLBL Library. Each example program is written in standard Fortran 77 and has as its first line a program statement of the form:

```
PROGRAM XXXXX
```

This statement is optional and the user may replace it with the appropriate program statement applicable for his computer or compiler.

All programs require the declaration statements:

```
DIMENSION B( ),AG( )
COMMON WORK( )
COMMON/PARM/IPAR(12),PAR(3)
COMMON/OUTCOM/LUNOUT,I80132
EXTERNAL BCK,PRJ
```

where `B` is the reconstruction array, `AG` is the array of angles, `WORK` is an array of blank common used for working space, `IPAR` is an integer array of input parameters, and `PAR` is a real array of input parameters. Besides the 15 input parameters in the `IPAR` and `PAR` arrays (cf. Sections 3.2 and 3.3), the user must specify the logical unit number `LUNOUT` for the output file and specify whether the output line will be 80 or 132 characters long by setting the parameter `I80132` zero or nonzero, respectively (cf. Section 3.1). The `EXTERNAL` statement must specify each back-projection, projection, convolution or filter subroutine that is passed as a parameter to one of the reconstruction subroutines.

The user must supply his own subroutine `GETUM`, which is used for data input as explained in Section 3.4. The subroutine `SETUP` must be called before calling any of the reconstruction subroutines as discussed in Section 3.1.

9.1 Example 1 – Projection and Back-Projection of Parallel- and Fan-Beam Data

The program `XBJECT` gives an example of how the subroutines `PJECT` and `BJECT` are used to project and back-project data. The projection data are obtained from the array `BX`, representing a point source at position (48,48). Using parallel-beam geometry, `PJECT` is called (statement E01.072) to project data into the user's projection array `P` with `KDIMU` = 100 and `AXISU` = 50.0. In statement E01.073 the data are back-projected into the array `B` and the resulting image is displayed by the subroutine `ARRAY`. This simple back-projection image is blurred by approximately $1/r$, where r is the distance from the point source to other elements in the image.

Next we perform a test to see how close the blurring function is approximated by $1/r$. In statement E01.078 through E01.081, the back-projection result is multiplied by the distance each point (I,J) is from the point (48,48), i.e., $\sqrt{(\mathtt{I}-48)^2+(\mathtt{J}-48)^2}$. Since the back-projection result gives a $1/r$ response for a point source, the multiplied image represents a nearly uniform distribution of intensities as can be seen from the displayed image. This same procedure is applied for fan-beam data projected for both curved and flat detectors. Notice that for the fan-beam geometry the back-projection subroutine BRFF2 (statements E01.095 and E01.117) is used instead of BRFF. The subroutine BRFF2 properly weights the projection data such that the result gives a $1/r$ response for a point source.

9.2 Example 2 – Convolution

The program XCONVO uses the convolution algorithm to reconstruct emission and transmission projection data for parallel-beam, fan-beam geometry with curved detector, and fan-beam geometry with flat detector. For the parallel-beam geometry the reconstruction is performed using the convolvers SHLO and RALA in statements E02.073 and E02.086, respectively. For the fan-beam geometries the convolver LAKS is used to reconstruct simulated projection data in statement E02.106 for the curved detector and in statement E02.124 for the flat detector. The convolvers SHLO and RALA *cannot* be used for fan-beam geometry *nor can* the convolver LAKS be used for parallel-beam geometry.

The errors XE in the reconstructed image are displayed in statements E02.082, E02.095, E02.115, and E02.133. The projection errors are input by the subroutine GETUM and it is assumed that these errors are equal to the square root of the projections (statement E02.215) for emission data and are all equal to 1 (statement E02.218) for transmission data.

The largest number of floating point words required in blank common is evaluated in statement E02.137. The output indicates that this example requires a maximum of 1605 words for blank common.

The subroutine GETUM generates simulated projection data for a heart phantom. For the parallel-beam geometry the pixel width PWID is equal to 1 and there is no scaling of the image; however, for the fan-beam geometries the images are reduced in size by a scaling factor equal to $1/\,\mathtt{PWID} = 0.752$. This reduction in size is due to the fact that the parameters to the phantom generators are always in units of projection bin widths ($= 1/\mathtt{PWID}$) and the images are displayed in units of pixel widths. Likewise for transmission data the intensity in each pixel, which represents the linear attenuation coefficient in units of inverse pixel width, is increased by a scaling factor PWID = 1.33. However, for emission data the intensity is in units of concentration per pixel and is therefore increased by a scaling factor $\mathtt{PWID}^2 = 1.77$.

9.3 Example 3 – Back-Projection of Filtered Projections

The program XBKFIL uses the back-projection of filtered projections algorithm to reconstruct parallel-beam projection data utilizing the filters HAM, HAN, PARZN, and RAMP with a cutoff frequency FREQX set to 0.5 and the filter BUTER with parameters FREQX = 0.52 and ORDERX = 388. The parameter ORDERX is used only for the filter BUTER and is therefore set to zero for the other examples. The RAMP filter and BUTER filter for this example have narrow real-space convolution windows and thus the reconstructed images have sharp contrast but increased

background artifact as compared to the other filters that have wider windows and decreased amplitude in the side lobes for their respective convolution functions. These latter filters give less background artifact but also poorer resolution in the reconstructed image than available for the RAMP and BUTER filters.

The subroutine GETUM gives simulated projection data for a heart phantom. A rectangular object in the upper right is added in order to compare the sharpness of the reconstructed image for the different filters.

9.4 Example 4 – Filter of the Back-Projection

The program XFILBK uses the filter of the back-projection algorithm to reconstruct simulated projection data for parallel-beam geometry, fan-beam geometry with flat detector. These three geometries are reconstructed using the filter subroutine HAN which is declared as an external in statement E04.037. This method of reconstruction requires a larger allocation for the blank common array WORK than is required for either the convolution algorithm (Example 2) or the back-projection of filtered projections algorithm (Example 3).

The output results show good agreement between XMAX of the reconstructed images and the original phantom. However, the sum of the total intensities XSUM of the original phantom does not compare well with XSUM of the reconstructed images. The user should keep in mind that the algorithm reconstructs an array that is four times as large as the image array that is returned to the user by the subroutine FILBK. This is necessary in order to minimize the error due to the convolution result of one period overlapping the convolution result of the succeeding period when implementing the discrete Fourier transform. The XSUM for this larger reconstructed array is zero since the filter zeros the dc component of the Fourier transform of the back-projection. However, the XSUM for the reconstructed array returned to the user does not equal zero since it represents only a fourth of the larger array; but even so it only approximates XSUM of the original phantom.

9.5 Example 5 – Iterative Conjugate Gradient

The program XCONGR uses the iterative conjugate gradient algorithm to reconstruct parallel-beam projection data for a pie phantom. The parameters for the subroutine CONGR are set in statements E05.065 through E05.068. Where IRLX = 1 indicates that the iterative relaxation method is used and ISTP = 15 indicates the iterative procedure will stop after 15 steps. The other parameters IERR and IZER are set to zero, indicating that the iterative reconstruction procedure does not use errors for weighting and that the initial solution is equal to zero.

The subroutine GETUM generates a pie phantom in the array B (statement E05.133) before the first angle (M = 1) and for each angle the array B is projected using the subroutine PJECT. The values of the projection array are simulated line integrals obtained using the projection subroutine PLL. For these data the conjugate gradient algorithm gives a reconstruction with discernible background artifact but good resolution. A comparison of this algorithm (CONGR) to the iterative gradient algorithm GRADY (Example 6) reveals that the latter gives less apparent background artifact but less resolution for 15 iterations. After 15 iterations the

conjugate gradient method has chi-square equal to 443, whereas the gradient method has a chi-square six times greater.

9.6 Example 6 – Iterative Gradient

The program `XGRADY` uses the iterative gradient method with relaxation to reconstruct parallel-beam projection data. The simulated data are the same as in Example 5 where the conjugate gradient method of reconstruction was used. After 15 iterations the gradient method has a chi-square equal to 2680, whereas the conjugate gradient method has chi-square six times smaller.

9.7 Example 7 – Iterative Program for Fan-Beam Data

The program `ITRFAN` uses the conjugate gradient algorithm to reconstruct simulated fan-beam projection data from a transmission source using both a curved detector and a flat detector. The projection data for the curved detector are reconstructed by the algorithm `CONGR` at statement E07.074 and the projection data for the flat detector are reconstructed in statement E07.105. The geometry parameter `IGEOM` is set to 1 in statement E07.045 for the reconstruction of the curved detector data and set to 2 in statement E07.093 for the reconstruction of the flat detector data. Note that a change in the parameters for `IPAR` or `PAR` requires another call to `SETUP`. For this particular example the fan-beam source is at a distance `RFAN` equal to 65 bin-width units from the center of rotation and the pixel width `PWID` is equal to l.33 bin-width units.

The subroutine `GETUM` in Example 7 gives simulated projection data for a heart phantom. Example 7 also demonstrates the use of the subroutine `USER`, which gives the user the option to retrieve certain results after each iteration as desired. In this example an image is displayed after each iteration (statement E07.248) along with a graph of the cross section through the image `X(I,J)` at the `J` coordinate = 15 (statement E07.259). If the user does not code a subroutine `USER`, the default subroutine `USER` in the RECLBL Library will print out only the iteration number and the corresponding chi-square for each iteration.

9.8 Examples 8, 9, 10 – Variable Attenuation Correction

The next three examples show how to use the RECLBL Library routines to reconstruct the true distribution of isotope concentration in a transverse section by compensating for attenuation in the projection data. In the program `ATENX` below, a transverse section of attenuation coefficients is first reconstructed from simulated projection data obtained from a transmission scan by using the reconstruction subroutine `GRADY` in statement E08.077. Using the reconstructed distribution of attenuation coefficients, the attenuation factors are evaluated with the subroutine `EVATN` in statement E08.103 and then the simulated emission data are reconstructed in statement E08.112 using the subroutine `GRADY`.

The subroutines `BRF`, `PRF`, `BRFA`, and `PRFA` (projection and back-projection routines) are declared externals in statement E08.038 and are passed as externals to the subroutine `GRADY` in statements E08.077 and E08.112. The back-projection and projection subroutines `BRFA`

and PRFA are used for attenuation correction and should only be used after EVATN (or EVATU) has been called.

For the transmission study with MODANG equal to 4 (statement E08.049), the subroutine SETUP generates equal projection angles between 0 and π, while for the emission study MODANG equals 5 (statement E08.096) and SETUP generates equal projection angles between 0 and 2π. The parameter IMIT is set to 1 in statement E08.066 for reconstructing transmission data and set to 0 in statement E08.097 for reconstructing emission data in order to allow for the correct normalization of the reconstructed values.

In Example 8, the subroutine GETUM uses the subroutine PHANL to generate projection data for either a transmission study or an emission study. The variable LTYPE in the labeled common TYPE, determines whether transmission data are returned (LTYPE = 1) or whether emission data are returned (LTYPE = 2). The simulated projection data are for an elliptical source phantom with a concentration of 30 within an elliptical attenuator of the same size, which has an attenuation coefficient of 0.075 (in units of inverse pixel width). If the pixel width is 0.5 cm, then the attenuation coefficient is equal to 0.075/0.5 cm = 0.15 cm^{-1}, which is approximately the attenuation coefficient for 140 keV photons in tissue.

Example 9 uses the following subroutine GETUM and the same program ATENX to reconstruct simulated projection data for a heart phantom, which is attenuated by an attenuator consisting of chest tissue and lungs.

Example 10 uses the following subroutine GETUM and the same program ATENX to reconstruct simulated projection data for a phantom with a circular annulus and a central circular source, which is attenuated by a circular attenuator.

9.9 Examples 11, 12 – Constant Attenuation Correction

Examples 11 and 12 show how to code a program that reconstructs emission projection data with attenuation compensation implemented by assuming a constant attenuation coefficient. The simulated emission data are first reconstructed giving an approximate reconstruction using the subroutine GRADY in statement E11.076. The projection and back-projection subroutines PRF and BRF are used in this example. The attenuation factors are then evaluated by EVATU in statement E11.101 with the constant attenuation coefficient ATENL equal to 0.075 (in units of inverse pixel width). The object-to-background ratio XLEV is used for the automatic border-searching routine and is set to 3.5 here. The subroutine EVATU first does a boundary search on the approximated reconstructed image B and then displays the object with an array plot showing the distribution of the constant attenuation coefficient ATENL. The user can vary XLEV until the desired object shape is obtained. The corrected transverse section is then reconstructed in statement E11.111. The projection and back-projection subroutines PRFA and BRFA should *only* be used when correcting for attenuation with one of the iterative routines (GRADY or CONGR) and *only* after the subroutine EVATU has been implemented.

Example 11 uses the subroutine GETUM to input simulated projection data for an elliptical source phantom with a concentration of 30 and an elliptical attenuator of the same size, which has an attenuation coefficient of 0.075. This is the same phantom reconstructed in Example 8 where a transmission study was first reconstructed to determine the distribution of attenuation coefficients. If the attenuation coefficient is constant and if the source has

the same distribution domain as the attenuator, then the following program will give good results without a separate transmission study.

Example 12 uses the same program ATENUX as Example 11 to reconstruct simulated projection data for a phantom with a circular annulus and a central circular source, which is attenuated by a circular attenuator. This example was also reconstructed in Example 10, where the distribution of attenuation coefficients was determined by a transmission study.

9.10 Example 13 – Orthogonal Polynomial Expansion

The program XMARR reconstructs projection data for a ring detector using the algorithm developed by R. Marr for representing the reconstructed image as an expansion of orthogonal polynomials. The simulated data for Example 13 are for a ring detector of 64 crystals, which is equivalent to 64 projection angles. The reconstructed image has a polynomial expansion with maximum degree equal to 62 (statement E13.067).

The user should study the description of GETUM in Section 3.4 before using the MARR reconstruction algorithm. The MARR algorithm requires that the data are input first for adjacent detectors, then for detectors spaced 2 apart, and so forth. This data format is illustrated in the printout given in this example.

9.11 Example 14 – Maximum Entropy

The program XENTPY reconstructs parallel-beam projection data by maximizing the entropy of the reconstructed image while satisfying projection constraints. This method of reconstruction should be used for projection data samples that give a system of linear equations that are under determined. Also, due to the computer time requirements, this method of reconstruction is best for small array sizes and sample sizes (cf. section VIII). For the example given below, the array size NDIMU is equal to 21 and the number of angles NANG is equal to 4. The maximum number of iterations LIMITX allowed for the determination of the optimum solution is set to 1000 in statement E14.065 and the convergence criterion ERENTX is set to 10^{-6} in statement E14.066. The convergence criterion is satisfied in 167 iterations; therefore the user may want to vary LIMITX and ERENTX to obtain the best results within certain computer time requirements. The value for the entropy for the reconstruction, subject to the constraints, is 5.79.

The subroutine GETUM gives simulated projection data for two rectangular phantoms within a circular disk of a higher intensity. The data are projected using the subroutine PJECT to assure that the data give a system of linear equations that are consistent.

9.12 Example 15 – Generalized Inverse

The sample program XGVERS uses the generalized inverse to reconstruct parallel-beam projection data and compares the result with the conjugate gradient method of reconstruction. Due to memory requirements, the generalized inverse method can reconstruct only small matrices. For example, a small matrix 12×12 (NDIMU $= 12$) requires that the array WORK in blank common have dimension equal to 46000. The reconstructed array B is returned in statement E15.069 along with the error matrix BE, which gives the statistical errors in the

reconstruction; i.e., $\texttt{BE}(I,J) = \sqrt{\text{var}[\texttt{B}(I,J)]}$. The printout indicates that the largest errors are for the reconstructed values in the center four pixels. The chi-square for the conjugate gradient solution after 15 iterations is equal to 5.36×10^{-4} as compared to a chi-square equal to 1.01×10^{-12} for the generalized inverse solution. (The chi-square for the generalized inverse is nonzero due to computer rounding.)

9.13 Examples 16, 17, 18 – Phantoms

The programs `CBARPX`, `PIEX`, and `RECT` give examples of the phantoms that can be generated using the RECLBL Library. The program `CBARPX` generates a circular bar phantom using the subroutine `CBARP`. The program `PIEX` generates a pie phantom with intensity that alternates between 0 and 1 for 20 slices of a circular disk using the subroutine `PIE`. The parameter `NSLIPI` = 10 used by the subroutine `PIE` is the number of equal slices of the disk in π radians. The program `RECT` generates two rectangular and two elliptical phantoms using the subroutine `PHAN`.

In order to obtain projection data for the circular bar phantom and the pie phantom, the subroutine `PJECT` is used as illustrated in the subroutine `GETUM` of Example 5. However, the projection data for the rectangular and elliptical phantoms can be generated using the subroutine `PHANL`. The calling sequences for both `PHANL` and `PHAN` are similar in that the parameters that describe the phantoms for both subroutines are identical.

10 LIBRARY LISTING

10.1 Quick Reference

10.1.1 Parameter Input

The subroutine `SETUP` initializes certain RECLBL common blocks and must be called before any of the reconstruction subroutines.

```
SUBROUTINE SETUP (IPAR,PAR,ANGL)
```

where

`IPAR` - Integer parameter array.

`PAR` - Floating point parameter array.

`ANGL` - Array of projection angles.

The elements of the `IPAR` and `PAR` arrays are defined as follows:

`IPAR(1)` = Linear dimension of the reconstruction array.

`IPAR(2)` = 0 to reconstruct a circular array,
otherwise reconstruct a square array.

`IPAR(3)` = 0 parallel-beam geometry.
1 fan-beam geometry (curved detector).
2 fan-beam geometry (flat detector).
3 ring-detector geometry.

`IPAR(4)` = Number of projection angles.

`IPAR(5)` = 0 user supplies projection angles in degrees.
1 user supplies projection angles in radians.
2 projection angles generated between zero and π starting at the half angle.
3 projection angles generated between zero and 2π starting at the half angle.
4 projection angles generated between zero and π starting at zero.
5 projection angles generated between zero and 2π starting at zero.
–I where I is between 2 and 5 does the same as above with the order of angles reversed.

`IPAR(6)` = Number of bins for each projection angle.

`IPAR(7)` = 0 to reconstruct emission data,
otherwise reconstruct transmission data.

`IPAR(8)` = Dimension of blank common set by the user.

`IPAR(9)` = Number of words for a floating point variable.

`IPAR(10)` = 0 to perform a reconstruction,
otherwise only do a storage size test.

`IPAR(11)` = Print flags (Bit 0 = least significant bit)

- Bit 0 Print required floating point blank common whenever changed.
- Bit 1 Print projection data and uncertainties.
- Bit 2 Print setup values from `IPAR` and `PAR` arrays.
- Bit 3 Print filter function for convolution and filter routines.
- Bit 4 Print values for the Lagrange multipliers for the entropy reconstruction.
- Bit 5 Print pointers in blank common whenever changed (debug).

`IPAR(12)` = Logical unit number for attenuation factor storage.

`PAR(1)` = Pixel width in units of projection bin width.

`PAR(2)` = Location of the rotation axis in the projection array.

`PAR(3)` = The distance from the source to the center of rotation for fan-beam geometry (measured in units of projection bin widths at the center of rotation).

10.1.2 Data Input

The subroutine `GETUM` is a subroutine *supplied by the user* that returns projection data and uncertainties for each angle.

```
SUBROUTINE GETUM (M,DATA,ERR)
```

where

`M` - Angle index number.

`DATA` - Projection data array for angle `M`.

`ERR` - Array of uncertainties of `DATA`.

10.1.3 Reconstructors

The subroutine `BJECT` back-projects a single projection array `P` of length `KDIMU` with rotation axis equal to `AXISU` into the array `B`. This allows the user to use the system back-projection subroutines and back-project user data into the user's own array.

```
SUBROUTINE BJECT (B,P,M,BCK)
```

where

`B` - The back-projection array.

`P` - The projection array.

M - The angle index.

BCK - The back-projection subroutine.

The subroutine BKFIL reconstructs the array X using the back-projection of filtered projections algorithm.

```
SUBROUTINE BKFIL (X,FIL,BCK,ORDERX,FREQX)
```

where

X - The reconstruction array.

FIL - The filter subroutine.

BCK - The back-projection subroutine.

ORDERX - Filter parameter used only by the filter BUTER.

FREQX - Filter parameter.

The subroutine CONGR reconstructs the array X by minimizing the chi-square using the method of conjugate gradients.

```
SUBROUTINE CONGR (X,PRJ,BCK,ISTP,IRLX,IERR,IZER)
```

where

X - The reconstruction array.

PRJ - The projection subroutine.subroutine.

BCK - The back-projection subroutine.

ISTP - Number of iteration steps.

IRLX - IRLX is not equal to 0 for iterative relaxation.

IERR - IERR is not equal to 0 for weighted least-squares.

IZER - IZER is equal to 0 if initial solution equals 0.

The subroutine CONVO reconstructs the array X using the back-projection of the convolved projections.

```
SUBROUTINE CONVO (X,XE,CNV,BCK,IERR)
```

where

X - The reconstruction array.

XE - The errors in the reconstructed array.

CNV - The convolution subroutine.

BCK - The back-projection subroutine.

IERR - The error flag (set nonzero to return XE).

The subroutine `ENTPY` reconstructs the array `X` using a maximum entropy criterion for the reconstructed image.

```
SUBROUTINE ENTPY (X,PRJ,BCK,LIMITX,ERENTX)
```

where

`X` - The reconstruction array.

`PRJ` - The projection subroutine.

`BCK` - The back-projection subroutine.

`LIMITX` - Maximum number of iterations allowed to minimize the objective function for the dual program.

`ERENTX` - Test value representing the expected absolute error between successive iterations.

The subroutine `FILBK` reconstructs the array `X` using the filter of the back-projection algorithm.

```
SUBROUTINE FILBK (X,FIL,BCK,ORDERX,FREQX)
```

where

`X` - The reconstruction array.

`FIL` - The filter subroutine.

`BCK` - The back-projection subroutine.

`ORDERX` - Filter parameter used only by the filter BUTER.

`FREQX` - Filter parameter.

The subroutine `GRADY` reconstructs the array `X` by minimizing the chi-square using the method of steepest descent.

```
SUBROUTINE GRADY (X,PRJ,BCK,ISTP,IRLX,IERR,IZER)
```

where

`X` - The reconstruction array.

`PRJ` - The projection subroutine.

`BCK` - The back-projection subroutine.

`ISTP` - Number of iteration steps.

`IRLX` - `IRLX` is not equal to 0 for iterative relaxation.

`IERR` - `IERR` is not equal to 0 for weighted least-squares.

`IZER` - `IZER` is equal to 0 if initial solution equals 0.

The subroutine `GVERS` reconstructs the array `X` using generalized matrix inversion.

```
SUBROUTINE GVERS (X,XE,PRJ,BCK,CHISQ,IERR)
```

where

`X` - The reconstruction array.

`XE` - Array in which errors in the reconstructed values are returned if `IERR` is set to 2. Should be the same dimension as `X`.

`PRJ` - The projection subroutine.

`BCK` - The back-projection subroutine.

`CHISQ` - The resulting chi-square.

`IERR` - Error indicator, set as follows:

1 - Input data uncertainties used, but no errors calculated for reconstructed values.

2 - Input data uncertainties used and errors are calculated for the reconstructed values.

Otherwise - Input data uncertainties not used and errors not calculated.

The subroutine `MARR` reconstructs the array `X` for a given set of chords from positron annihilation events detected with a ring of crystals using orthogonal polynomial expansion.

```
SUBROUTINE MARR (X,NDEG)
```

where

`X` - The reconstruction array.

`NDEG` - Degree of the polynomial expansion.

10.1.4 Back-Projectors and Projectors

`BCD/PCD` - Back-projector/projector using the concave disk model (i.e., any pixel projects as a square wave at any angle using parallel-beam geometry.)

`BCDA/PCDA` - Same as `BCD/PCD`, but compensating for attenuation effects using factors from the file `LUNATN`.

`BCDF/PCDF` - Same as `BCD/PCD`, but using fan-beam geometry.

`BCDF2` - Back-projector using the concave disk model in fan-beam geometry that uses special weighting to achieve a true back-projection image. Should only be used with FILBK.

`BIN` - Back-projector that uses a linear interpolation model similar to concave disk for parallel-beam geometry when projection bins and pixels are of unequal size (`PWID` $\neq 1.0$).

`BLL/PLL` - Back-projector/projector using the uniform distribution model with weighting according to line length for parallel-beam geometry.

`BPT/PPT` - Back-projector/projector using the delta function model for parallel-beam geometry.

`BPTA/PPTA` - Same as `BPT/PPT`, but compensating for attenuation using factors from file `LUNATN`.

`BPTF/PPTF` - Same as `BPT/PPT`, but using fan-beam geometry.

`BPTF2` - Same as `BPTF`, but using special weighting to obtain a true back-projection image. Should only be used with `FILBK`.

`BRF/PRF` - Back-projector/projector using the uniform distribution model with weighting by ray sums using parallel-beam geometry.

`BRFA/PRFA` - Same as `BRF/PRF`, but compensating for attenuation using factors from the file `LUNATN`.

`BRFF/PRFF` - Same as `BRF/PRF`, but using fan-beam geometry.

`BRFF2` - Same as `BRFF`, but using special weighting to achieve a true back-projection image. Should only be used with `FILBK`.

10.1.5 Convolvers (used only with `CONVO`)

`LAKS` - Convolver used for fan-beam reconstruction (after Herman, Lakshminarayanan and Naparstek).

`RALA` - Convolver used for parallel-beam reconstruction (after Ramachandran and Lakshminarayanan).

`SHLO` - Convolver used for parallel-beam reconstruction (after Shepp and Logan).

10.1.6 Filters (used only with `BKFIL` and `FILBK`)

`BUTER` - Uses the Butterworth filter as a window on the ramp filter.

`HAM` - Uses the Hamming window on the ramp filter.

`HAN` - Uses the Hann window on the ramp filter.

`PARZN` - Uses the Parzen window on the ramp filter.

`RAMP` - Generates the values in frequency space for a ramp filter.

10.1.7 Phantom and Projection Generators

Subroutine PHAN generates a phantom consisting of ellipses and rectangles in the square array BB which has dimension (N,N).

```
SUBROUTINE PHAN (NPHAN,INTG,ITYPE,DENS,X,Y,A,B,PHI,BB,N,PIXW)
```

where

NPHAN - The total number of ellipses and rectangles.

INTG - An integration factor. When a pixel lies partly inside and partly outside a boundary, it is divided into INTG × INTG pixelettes, which are each checked for insideness. The final value assigned to the large pixel is the value of DENS multiplied by the fraction of pixelettes that were found to lie inside the boundary (a good value is 10).

ITYPE - An array of descriptors for the ellipses/rectangles.

1 for an ellipse.

2 for a rectangle.

DENS - An array of densities of the ellipses/rectangles. For transmission the units are inverse projection bin width. For emission the units are inverse (projection bin width)^2.

X,Y - Arrays giving the coordinates of the centers of the ellipses/rectangles with respect to the center of rotation (in units of projection bin width).

A,B - Arrays giving the major and minor axes of ellipses or the lengths of the sides of rectangles (in units of projection bin width).

PHI - An array of angles (in radians) that the major axes of the ellipses or the A sides of the rectangles make with the x-axis.

BB - Array where phantom is generated.

N - The dimension of BB.

PIXW - Pixel width that is utilized by this routine in order that the values for BB be as reconstructed (+ for transmission, – for emission).

Subroutine PHANL generates the line integral projections of a set of source ellipses and rectangles attenuated by another set of attenuating ellipses and rectangles.

```
SUBROUTINE PHANL (N,ITYPE,DENS,X,Y,A,B,PHI,P,M)
```

where

N - The total number of ellipses and rectangles.

ITYPE - An array of descriptors for the ellipses/rectangles.

1 for a source ellipse.

2 for a source rectangle.

-1 for an attenuating ellipse.

-2 for an attenuating rectangle.

`DENS` - An array of source densities or attenuation coefficients of the ellipses/rectangles. For transmission the units are inverse projection bin width. For emission the units are inverse (projection bin width)^2.

`X,Y` - Arrays giving the coordinates of the centers of the ellipses/rectangles with respect to the center of rotation (in units of projection bin width).

`A,B` - Arrays giving the major and minor axes of ellipses or the lengths of the sides of rectangles (in units of projection bin width).

`PHI` - An array of angles (in radians) that the major axes of the ellipses or the `A` sides of the rectangles make with the x-axis.

`P` - The array into which the projection is generated.

`M` - The projection angle index as defined in `SETUP`.

The subroutine `CBARP` gives a circular bar phantom.

```
SUBROUTINE CBARP (B1,N,R,X1,Y1,Z,INTFAC,NBAR,NREPS,IDIREC)
```

where

`B1` - Array where phantom is generated.

`N` - Dimension of the square array `B1`.

`R` - Radius of circle phantom.

`X1,Y1` - Center of circle relative to the center of array.

`Z` - Full value of function.

`INTFAC` - Integration factor. Each border pixel is divided into `INTFAC`^{2} pixelettes for integration.

`NBAR` - Number of bars per pattern repetition in phantom.

`NREPS` - Number of repetitions of the bar pattern. `NBAR*NREPS` must be less than or equal to 100.

`IDIREC` - Parameter telling direction of bars.

0 Horizontal.

1 Vertical.

2 Circular (concentric).

The subroutine `PIE` gives a pie phantom.

```
SUBROUTINE PIE (B1,N,R,X1,Y1,Z,INTFAC,NSLIPI,ISTART)
```

where

`B1` - Array where phantom is generated.

N - Dimension of the square array B1.

R - Radius of circle phantom.

X1,Y1 - Center of circle relative to the center of array.

Z - Full value of function.

INTFAC - Integration factor. Each border pixel is divided into INTFAC^2 pixelettes for integration.

NSLIPI - Number of slices in half the pie (in π radians).

ISTART - Indicator of the color of the first (counterclockwise) slice. 0 = white, else it is black.

The subroutine PJECT projects from the array B into a single projection array P of length KDIMU with rotation axis equal to AXISU. This allows the user to use the system projection subroutines and project data into the user's own projection array.

```
SUBROUTINE PJECT (B,P,M,PRJ)
```

where

B - The array of data for the transverse section.

P - The projection array.

M - The angle index.

PRJ - The system projection subroutine.

10.1.8 Attenuation Correction

The subroutine EVATN evaluates the attenuation factors required to correct for attenuation in an emission scan.

```
SUBROUTINE EVATN (B)
```

where

B - Array of attenuation coefficients.

The subroutine EVATU evaluates the attenuation factors required to correct for attenuation in an emission scan, assuming a constant attenuation coefficient.

```
SUBROUTINE EVATU (B,XLEV,ATENL)
```

where

B - Transverse section that has not been corrected for attenuation.

XLEV - Approximate ratio of the concentration in the object to the background.

ATENL - Constant attenuation coefficient.

The subroutine ARRAY gives an image of the array B on computer output paper where the distinct gray levels are accomplished by overprinting.

```
SUBROUTINE ARRAY (B,NXN)
```

where

B - The array to be imaged.

NXN - The dimension of the array.

Subroutine BCOM prints out and returns the largest number of floating point words required in blank common (MAXFW).

```
SUBROUTINE BCOM (MAXFW)
```

where

MAXFW - Maximum number of floating point words needed in blank common so far.

The subroutine USER gives the user the opportunity to investigate the partial reconstruction between iterations.

```
SUBROUTINE USER (ITER,X,FCN)
```

where

ITER - Iteration number.

X - Array of fitted parameters:
for CONGR and GRADY, reconstruction array;
for ENTPY, Lagrange multipliers.

FCN - Value of function being optimized:
for CONGR and GRADY, chi-square;
for ENTPY, objective function for the dual program.

The subroutine XYGRF displays NP plots of the cross-section intensities for the N × N array B.

```
SUBROUTINE XYGRF (B,N,NP,BMAX,BMIN,IXY,ICOR,IL,IU)
```

where

B - Square array from which plots are generated.

N - Dimension of B is (N,N).

NP - Number of cross-sectional plots.

BMAX - Maximum value for the plot. If BMAX = 999999., the maximum will be determined from the data.

BMIN - Minimum value for the plot. If BMIN = 999999., the minimum will be determined from the data.

`IXY` - Equals 0 if the cross section is parallel to the x-axis. Equals 1 if the cross section is parallel to the y-axis.

`ICOR` - Array of x- or y-intercepts that determines the location of the cross section.

`IL` - Lower coordinate for the plot.

`IU` - Upper coordinate for the plot.

10.2 Listing

A complete listing of the RECLBL Library source material follows.