JMKMOD

Models and Software for the Ground Calibration of AXAF

Version 7.0

Eugene Y. Tsiang
The Smithsonian Astrophysical Observatory
of the
Harvard-Smithsonian Center for Astrophysics
60 Garden Street
Cambridge
MA 02138
email: **tsiang@cfa.harvard.edu**

Contents

Part I

Introduction

This manual describes JMKMOD, a suite of models that may be used for:

1. Characterizing the flow proportional counters (fpc's) and solid-state detectors (ssd's) used in the ground calibration of the NASA Advanced X-ray Astrophysical Facility (AXAF) at the X-ray Calibration Facility (XRCF, Marshall Space Flight Center).

2. Extracting the detector line and continuum counts for absolute (using the PTB/BESSY synchrotron white beam) and relative calibration of the detector quantum efficiencies.

3. Extracting the High Resolution Mirror Assembly (HRMA) effective area.

This manual consists of three parts:

- *JMKMOD User's Guide*
- *JMKMOD Reference Guide*
- *JMKMOD Software Guide*

0.1 JMKMOD User's Guide

The *JMKMOD User's Guide* is for the data analyst who needs the knowledge to use the JMKMOD model within XSPEC. It contains a table of the XSPEC parameters, and a table of the source/filter functions for the spectral continuum. A brief description of the parameters are given.

0.2 JMKMOD Reference Guide

The *JMKMOD Reference Guide* forms the theoretical backbone. It is for the data analyst who wants a knowledge of the physical meaning of the parameters she gets from her XSPEC fits. It gives the origins of the models and their parameters, with some previously unpublished difference-differential equation solutions given in the **Appendix**. The author feels that further model development should trace its roots to the basic physics described here. It provides an example of how to develop a semi-empirical model for any charge-multiplying detector, gaseous-electronic or otherwise, starting from basic principles.

0.3 JMKMOD Software Guide

The *JMKMOD Software Guide* is for the programmer who wants to extend JMKMOD. It gives the software architecture and a listing of the Fortran code with comments.

0.4 Acknowledgments and a Development History

The author would like to thank the HEASARC staff in general and Keith Annaud(GSFC) in particular for their roles in the development and support of XSPEC. He would also like to thank Allyn Tennant(MSFC) for answering his numerious questions about XSPEC.

Edwin Kellogg(SAO) first brought the existence of XSPEC as a best-fit modeling package to the attention of the author around May, 1995. Not recognizing the adaptability of XSPEC to handle the simultaneous fitting of source and detector model parameters, the author did not pursue the idea of using XSPEC until July, 1995, when Jack Hughes(then SAO, now Rutgers) and Richard Edgar(SAO) suggested the use of the diagonal matrix within XSPEC, as well as the Jahoda-McCammon model for FPCs.

The author worked out a closed-form proportional counter model, and tested the algorithm for implementing it on a PC *Mathematica* platform in July, 1995. Sarah Vitek(SAO) and the author took advantage of some previous IDL code left over from the AXAF VETA experiment[13] that reads in the atomic absorption coefficients from the Henke Tables[22]. They tested the algorithm in routines written in the IDL language[23] between August and September, 1995. Vitek then ported the IDL code into Fortran(the base language of XSPEC) between September and November, 1995. They successfully tested the first JM-Kramers model on real data between September and December, 1995. Vitek applied the basic JM-Kramers model to many FPC spectra in early 1996.

Even before he knew about XSPEC and the Jahoda-McCammon model, the author had worked out an algorithm of spectral pileup removal in early 1955, based on a theory due to Tenney, which showed promise when used on Fe55 data accumulated on SSDs at SAO. By happenstance, this method depended on the Fourier Transform function of the count spectrum, which fit in most harmoniously with the moment generating function approach he took with the JM model later.

Unlike the Kramers or Pella model of the electron impact point source (EIPS), the absolute synchrotron 'white-beam' spectrum can be calculated and measured to a high rate of precision, and can be used for absolute calibration. In early 1996, the author worked out the synchrotron radiation flux through a circular aperture placed on and above the cyclotron orbit plane, as part of a PTB/BESSY white beam calibration study. He also gave a formal meaning to absolute calibration in the parameter fitting context.

The author brought all these strands together in JMKMOD, a complete suite of models for the relative and absolute calibration of flow proportional counters and solid-state detectors, between July and December 1996. The present version is Version 7.0.

Part II

JMKMOD User's Guide

We present a semi-empirical model suitable for the spectroscopic analysis of proportional counters, which serves, by extension, for solid-state detectors as well. These counters have been undergoing tests at SAO and MSFC for use in the ground calibration of the NASA Advanced X-ray Astrophysical Facility (AXAF) X-ray mirrors (High Resolution Mirror Assembly, or HRMA). The motivation for developing the model is the extraction of photon counts from line emissions to characterize the AXAF mirrors. The model, based on the theories of Prescott, Alkhazov and Jahoda-McCammon, has been incorporated into XSPEC, an X-ray spectral-fitting program widely used by the astrophysical community. The model can extract discrete line strengths and the continuum from the count spectrum, as well as the parameters characterizing the spectral response function. The novel features of this model are:

1. The considerable insight provided by the traceback of the parameters of the model back to the underlying physics of ionization and recombination. Thus, the Fano factor F is related to the ratio of the rates of ionization and recombination. The Polya h factor is expressed as the ratio of the actual gas amplication factor to that predicted by Townsend's avalanche theory of electron multiplcation. They both give rise, in general, to non-Poissonian gain variances.

2. The deduction of a closed-form expression for the moment generating function (Laplace Transform) of the detector spectral response function. For calibration, it is normally only possible, if not extremely advantageous, to use closed-form expressions, as opposed to models based on extensive simulations or calculations, where each run is a brand-new mathematical problem based on some assumed parameters.

3. The inversion of the the moment generating function to give the final spectral response function by inverse Fast Fourier Transform. This is also the only way of getting a convenient distribution based on the theories of Alkhazov and Jahoda-McCammon distribution to extract the parameters of the detector.

4. The reduction of the general detector response function to the classic Prescott function in the limit Fano factor $F \rightarrow 1$, Polya h factor $\rightarrow 1$.

Chapter 1
JMKMOD

1.1 JMKMOD, an XSPEC model

JMKMOD is the software for calibrating, using semi-empirical detector/source models, the ground-based AXAF X-ray detectors at several sites: the X-ray Calibration Facility(XRCF) of the Marshall Space Flight Center (MSFC, Huntsville, AL), the Smithsonian Astrophysical Observatory(SAO, Cambridge, MA) and the PTB(BESSY, Berlin). JMKMOD includes the suite of models for analyzing the XRCF, SAO and BESSY spectral data, with each individual XSPEC model within this suite being denoted by ⟨jmkmod⟩. Each ⟨jmkmod⟩ is a model with various numbers of incoming spectral lines and source/filter components. The "off-the-shelf" ⟨jmkmod⟩ models in Version 7.0 (January 29, 1997) are given in the following table:

⟨jmkmod⟩	**lines**	**filters**	**no. pars incl. global norm**
jmkmod	2	6	59
jmkmod4	4	6	63
jmkmod8	8	6	71
jmkmod16	16	6	87

Global norm is the normalizing constant at the very end of each XSPEC model. The basic model is jmkmod for 2 spectral lines, which is also the name of the Fortran subroutine **jmkmod.f** incorporated into the XSPEC local functions subdirectory

$XANADU/local/spectral/xspec/src/functions

. Models for more lines and filters all call **jmkmod.f**.

The number of lines and filters can be changed by an authorized XSPEC 'superuser' by editing the file **lmodel.dat** in the local model subdirectory

$XANADU/local/spectral/xspec/src/model

. The user may desire to do this if the number of lines and filters are insufficient, or if she wants to make the program run more efficiently by having a custom number of lines and filters. She would change the parameters **numfil** and **numcomp** in the parameter table. For example, jmkmod8 MUST have numfilter = 6, numcomp = 8. The actual parameters for the lines and filters come after all the other parameters

of JMKMOD. It will be apparent to the user what to modify by looking at the off-the-shelf examples in **lmodel.dat**.

JMKMOD is a suite of models of the **output count spectrum** in the detectors. The output count spectrum is the detector response function (Jahoda-McCammon, or JM model, and its various extensions) that has been convolved with line and continuum source spectrum, with counting artifacts such as pileup added. It is intended for use with a **unit diagonal response matrix** within XSPEC. Its usage differs from the usual mode of XSPEC in the following way. In the usual mode, the user numerically convolves the input model spectrum and the detector response function to give an output count spectrum which is compared to the observed spectrum, and the parameters are adjusted to give a "best" fit. For detector calibration work, this detector response itself is not known, and indeed is part of the goal of the calibration itself. Therefore, we include the response matrix (JM model, gaussian, etc.) in the model of the incoming spectrum. We define the XSPEC response matrix, i.e. the one invoked by the XSPEC command

$$\text{XSPEC> response } [\langle\text{diagresponse}\rangle \ldots] \tag{1.1}$$

to be a **unit square diagonal matrix** file ⟨diagresponse⟩, which being a unit matrix, does nothing to the output count spectrum. The ⟨diagresponse⟩ used most often are **diag512.rmf** and **diag4096.rmf** for proportional counters and solid-state detectors respectively.

1.2 Using JMKMOD

To analyze a single spectrum, first convert the **.pha** files of interest into **.fits** files, using a PERL script such as **pha2fits[37]**. Invoke XSPEC with the command

$$\text{> xspec data } \langle\text{filename}\rangle \tag{1.2}$$

or read in the observed count spectrum within XSPEC with the command:

$$\text{XSPEC> data } \langle\text{filename}\rangle \tag{1.3}$$

where ⟨filename⟩ is the FITS data file. Bring in the response with

$$\text{XSPEC> response } \langle\text{diagresponse}\rangle \tag{1.4}$$

Two commonly used ⟨diagresponse⟩s are **diag512.rmf** and **diag4096.rmf** for use with data containing 512 (flow proportional counters) and 4096 channels (solid-state detectors) respectively. Make sure these are in your directory or search path. Invoke JMKMOD. For a 4-line spectrum, type

$$\text{XSPEC> model jmk mod 4} \tag{1.5}$$

As a rule, invoke JMKMOD as a standalone model only—**the additive and multiplicative aspects of XSPEC should not be used in conjunction with JMKMOD**. This is because the electronic pileup effects (if the pileup switch is turned on or set to a nonzero number) mixes the different spectral components of JMKMOD with each other, so that various components are no longer distinct. Thus with pileup included, the invocation of a 4-line model with (1.5) is correct, while

$$\text{XSPEC> model jmk mod + jmkmod} \tag{1.6}$$

would be wrong. Of course, if pileup is not included, (1.5) is permissible.

Following invocation of JMKMOD, XSPEC responds with a query for parameter values, described in the following section.

1.3 JMKMOD parameters

The number of parameters for **V.7.0** stands at 59 for 2 lines and 6 filters, including switches. Some typical values are shown in the following table. Not all the parameters are active at any one fitting. For example, if 'sestype 2' (which stands for single electron spectral type 2) is chosen (usually for ssd spectra), then "polyah" is ignored, because sestype 2 corresponds to an SES that is a delta function ($h \to \infty$). As another example, the gas temperature and pressure have no meaning for SSDs. **It is important for the user to freeze these inactive parameters.** At best, XSPEC will waste time calculating unnecessary elements in the sensitivity matrix.

A word about the counts: the final count rates are the products of the global normalization N_{global}(last parameter 59) and the individual line or continuum components. One tactic is to get a reasonably good fit with parameter 59 free and either the line or continuum component fixed at 1. The final fit is then done with N_{global} fixed at 1, and letting the line and continuum norms go free, after setting their best guesses equal to the products of the old line and continuum norms with the old N_{global}. The disadvantage of freezing N_{global} is that the renormalization command

XSPEC>ren

loses its effect.

The parameter table for jmkmod (2 lines, 6 filters) with some typical SSD values is as follows.

Table 1.1: JMKMOD parameter table

No.	parameter	value	par comp	comments
1	fano F	0.166682	+/- 0.	Fano factor (JM model)
2	polyah h	1.20000	frozen	Polya h factor (JM model)

Table 1.1: JMKMOD parameter table (*cont.*)

No.	parameter	value	par comp	comments
3	E_offset E_0	0.	frozen	Offset energy (keV, JM model)
4	i_pot w	3.000000E-03	frozen	pair creation energy (keV, JM)
5	gain μ	0.428890	+/- 0.	(channels per pair, JM)
6	ch_off q_o	73.1796	+/- 0.	zero offset (channels)
7	broad σ_{ch}	20.6944	+/- 0.	sigma (channels)
8	d_gain $\Delta\mu$	0.	frozen	gain variation (channels)
9	nchan N_{ch}	4096.00	frozen	no. actual channels (512/4096)
10	sestype	2.00000	frozen	switch: 1 (JM), 2 (ssd), 3 (broad JM)
11	contin	1.00000	frozen	switch, n.e. 0 for continuum
12	econt E_c	13.2840	+/- 0.	continuum energy (keV): upper cutoff of bremsstrahlung, or critical energy of synchrotron
13	contnorm N_{cont}/N_{global}	9.035257E-02	+/- 0.	continuum norm(dimensionless)
14	E_max E_{max}	13.2840	= par 12	convolution upper limit (keV)
15	E_min E_{min}	1.000000E-02	frozen	convolution lower limit (keV)
16	E_char E_{char}	2.00000	frozen	characteristic line energy (keV) with natural width
17	E_width E_{width}	0.100000	frozen	natural width of line (keV)
18	nInt N_{int}	512.000	frozen	no. convolution points
19	temper T	15.0000	frozen	temperature of fpc (deg. C)
20	pressure P	400.000	frozen	pressure of fpc (Torr)
21	argrat	0.900000	frozen	argon ratio for P10

Table 1.1: JMKMOD parameter table (*cont.*)

No.	parameter	value	par comp	comments
22	shelfsw	1.00000	frozen	switch, n.e. 0 for shelf
23	t1norm β_{tail}	0.	frozen	descending tail norm (dimensionless)
24	t1par x_1	0.999000	frozen	descending tail parameter
25	t2norm γ_{tail}	0.	frozen	ascending tail norm (dimensionless)
26	t2par x_2	1.00100	frozen	asending tail parameter
27	shelfnm δ_{shelf}	5.894007E-02	+/- 0.	flat shelf norm (dimensionless)
28	pulser	0.	frozen	switch, n.e. 0 for pulser
29	pulsepos q_{pulser}	100.000	frozen	pulser position(channels)
30	pulsesig σ_{pulser}	1.00000	frozen	pulser sigma (channels)
31	pulsenrm N_{pulser}/N_{global}	1.00000	frozen	pulser norm (dimensiless)
32	numcomp	2.00000	frozen	number of line components
33	numfil	5.00000	frozen	number of source/filters
34	pileup	1.00000	frozen	switch, 1 for periodic pulser, 0 no pulser, -1 random pulser
35	pilepar $\exp(-N_{true}\tau_{pu})$	1.00000	+/- 0.	pileup parameter (dimensionless)
36	bfield B_{synch}	14500.0	frozen	magnetic induction (gauss)
37	current I_{synch}	1.00000	frozen	current (mA)
38	incline α_{synch}	0.	frozen	inclination of aperture from orbital plane (radians)
39	abswitch	0.	frozen	absolute calibration switch
40	phi ϕ_{HRMA}	0.	frozen	HRMA pitch angle in degrees

Table 1.1: JMKMOD parameter table (*cont.*)

No.	parameter	value	par comp	comments
41	theta θ_{HRMA}	0.	frozen	HRMA yaw angle in minutes
42	combo	0.	frozen	HRMA mirror combination
43	E_line1 $E_{\gamma 1}$	1.49000	frozen	line 1 energy (keV)
44	E_line2 $E_{\gamma 2}$	1.49000	= par 39	line 2 energy (keV)
45	E_norm1 $N_{\gamma 1}/N_{global}$	1.00000	frozen	line 1 norm (dimensionless)
46	E_norm2 $N_{\gamma 2}/N_{global}$	0.	frozen	line2 norm (dimensionless)
47	src1	35.0000	frozen	source/filter 1
48	src2	2.00000	frozen	source/filter 2
49	src3	5.00000	frozen	source/filter 3
50	src4	2.00000	frozen	source/filter 4
51	src5	37.0000	frozen	source/filter 5
52	src6	42.0000	frozen	source/filter 6
53	srcthk1	0.	frozen	source/filter 1 parameter
54	srcthk2	1.185682E-03 +/-	0.	source/filter 2 parameter
55	srcthk3	1.200000E-04	frozen	source/filter 3 parameter
56	srcthk4	2.000000E-05	frozen	source/filter 4 parameter
57	srcthk5	0.500000	frozen	source/filter 5 parameter
58	srcthk6	0.500000	frozen	source/filter 6 parameter
59	global norm N_{global}	2413.54	+/- 0.	scaling norm (Hz)

A brief explanation of the various parameters follows. Instead of working with probability distribution functions (pdf's), we work with their moment generating functions (mgf's). This mgf-based approach, the physics of the model and detailed derivations are given in the *JMKMOD Reference Guide* and the **Appendix**. Where appropriate, the symbols in the parameter table are the same as those used in the *Guide*.

1.3.1 FPC/SSD response function parameters

The basic detector response function is given by its moment generating function (mgf) $\Phi_a(s)$, where s is the Laplace Transform parameter conjugate to the channel number:

Fano	**response function mgf $\mathbf{\Phi}_a(\mathbf{s},\mathbf{E}_\gamma)$**
$F \neq 1$	$R_a(s)(1-(1-F)(1-R_a(s)))^{\frac{\langle n_{se}\rangle}{1-F}}$
$F = 1$	$R_a(s)exp\left(-\langle n_{se}\rangle\left(1-R_a(s)\right)\right)$

in which the mgfs $R_a(s)$ for single electron specta (SES) of different types are:

type / SES	**description**	**mgf $\mathbf{R}_a(\mathbf{s})$**
1 (FPC/SSD)	'Polya' or Erlangian	$\left[1+\frac{\mu}{h}s\right]^{-h}$
2 (SSD)	Delta function	$\lim_{h\to\infty}\left[1+\frac{\mu}{h}s\right]^{-h} \to exp\left(-\mu s\right)$
3 (FPC)	Broadened gain (flat)	$\frac{\ln\left(\frac{1+(\mu+\Delta\mu)s}{1+(\mu-\Delta\mu)s}\right)}{2s\Delta\mu},\ h=1$
	$\int_{\mu-\Delta\mu}^{\mu+\Delta\mu}\left[1+\frac{\mu'}{h}s\right]^{-h}d\mu'$	$\frac{\left[1+\frac{(\mu-\Delta\mu)}{h}s\right]^{1-h}-\left[1+\frac{(\mu+\Delta\mu)}{h}s\right]^{1-h}}{2s\Delta\mu\left(1-\frac{1}{h}\right)},\ h\neq 1$

One can imagine adding more SES types to this list, such as one where the the probability distribution of the gain is a known function across an FPC wire.

As shown in the *Reference Guide*, the special case of $h = 1$ and $F = 1$ corresponds to the mgf of the Prescott pdf. The mean number of secondary pairs (electron-ion or electron-hole) created are

$$\langle n_{se}\rangle = \frac{E_\gamma - E_0}{\langle w\rangle}$$

where $\langle w\rangle$ is the average energy in keV to produce a pair ($28eV$ for electron-ions and $3eV$ for electron-holes), E_γ is the energy of the incoming X-ray in keV, E_0 is the offset energy in keV.

In addition, electronic broadening due to preamplifier noise in the detector electronics contribute a shift and a broadening in channels of q_{ch} and σ_{ch} respectively, giving for the detector mgf

$$\Phi_a(s,E_\gamma) \underset{\text{electronic broadening}}{\longrightarrow} \exp\left(-q_{ch}\,s+\frac{\sigma_{ch}^2 s^2}{2}\right)\Phi_a(s,E_\gamma)$$

The electronic broadening is the dominant source of response function broadening for SSDs, but there is a small Fano component as well.

1.3.2 Channel-energy conversion ratio

The ratio

$$\frac{\mu}{w} \tag{1.7}$$

(channels per keV) gives the conversion from energy to channels. The reciprocal gives the conversion from channels to energy:

$$\frac{w}{\mu} \tag{1.8}$$

1.3.3 Shelf-tail parameters

Fano/mgf	**tail** $\Phi_{tail}(s)$	**shelf** $\Phi_{shelf}(s)$
$F \neq 1$	$\frac{y^{\langle n_{se}\rangle} x^{n\max} - 1}{\left(y^{\langle n_{se}\rangle} - 1\right)\left[1 + \frac{\ln x}{(1-F)\ln y}\right]}$	$\frac{(1-F)(1-x^{n\max})}{-\langle n_{se}\rangle \ln x}$
$F = 1$	$\frac{-1 + y^{\langle n_{se}\rangle} \exp[\langle n_{se}\rangle (R_a(s)-1)]}{\left(y^{\langle n_{se}\rangle} - 1\right)\left[1 + \frac{R_a(s)-1}{\ln y}\right]}$	$\frac{\exp[\langle n_{se}\rangle (R_a(s)-1)] - 1}{(R_a(s)-1)\langle n_{se}\rangle}$

where

$$x \equiv F + (1 - F)\, R_a(s) \tag{1.9}$$

$$y = \text{tail parameter}$$

$$R_a(s) = SES$$

These mgfs are all unit normalization. Two mgfs for tails and a shelf mgf are added to the basic single line mgf $\Phi_a(s)$ to give the line mgf

$$\Phi_{line}(s) = \frac{\Phi_a(s) + \beta_{tail}\Phi_{tail}(s, t_{l+}) + \gamma_{tail}\Phi_{tail}(s, 0) + \delta_{shelf}\Phi_{tail}(s, t_{l-})}{1 + \beta_{tail} + \gamma_{tail} + \delta_{shelf}} \tag{1.10}$$

1.3.4 Continuum parameters

When the continuum switch is non-zero, then a continuum spectrum $N_c(E_\gamma)$as modified by the detector quantum efficiency $\eta(E_\gamma)$and any transmission filters $exp\left[-\mu_{source}(E_\gamma)\, t_{source}\right]$, is included as part of the fit. The continuum mgf is

$$\Phi_{cont}(s) = \int_{E\min}^{E_{\max}} \Phi_a(s, E_\gamma) \cdot \eta(E_\gamma) \cdot \; exp\left[-\mu_{filter}(E_\gamma)\, t_{filter}\right] \cdot N_c(E_\gamma)\, dE_\gamma$$

where $\Phi_a(s, E_\gamma)$ is the selected response function mgf above for incoming X-ray energy E_γ. This convolution integral is numerically evaluated from some minimum energy (typically $10eV$ or some lower energy cutoff where the transmission or q.e. becomes negligible) to some maximum energy $E_{\max}$. The number of **uniformly spaced** numerical integration points chosen is N_{int}, which is practically chosen to be 512, but may be set to a maximum of 1024 (for Version 7.0). More points may be added if needed by changing the program. The user may experiment with this parameter to see if there is a difference in how closely she choses his integration points. For a list of the continuum sources and filters available, see the **Source/Filter Table** below.

1.3.5 Pulser parameters

To fit the pulser set the pulser switch to a non-zero number. At the moment the pulser mgf is assumed to be gaussian:

$$\Phi_{pulser}(s) = \exp\left(-q_{pulser}\, s + \frac{\sigma_{pulser}^2\, s^2}{2}\right)$$

where q_{pulser} and σ_{pulser} are the pulser location and broadening in channels respectively. More general profiles such as the Voight profile may be added later as the actual pulser profile is often not gaussian, or the user may choose not to fit the pulser channels, since doing so is inherently inaccurate.

1.3.6 Pileup parameters

To turn pileup on set the pileup switch to a non-zero number. Let the combined mgf from the lines $E\gamma$, continuum and pulser be

$$\Phi_{total}(s) = \frac{N_{pulser}\Phi_{pulser}(s) + N_{cont}\Phi_{cont}(s) + \sum_{\gamma} N_{\gamma}\Phi_{a}(s, E_{\gamma})}{N_{total}} \tag{1.11}$$

where N_{pulser}, N_{cont} and N_{γ} are the pulser, continuum and line norms(count rates), and

$$N_{total} = N_{pulser} + N_{cont} + \sum_{\gamma} N_{\gamma} \tag{1.12}$$

is the total incoming count rate. Then the final mgf $\Phi_{pileup}(s)$ after pileup is

type of pulser	$\mathbf{\Phi}_{pileup}(\mathbf{s})$
1 or periodic	$\Phi_{total}(s)\frac{e^{-N_{total}\ \tau_{pu}}}{1-\left(1-e^{-N_{total}\ \tau_{pu}}\right)\Phi_{total}(s)}$
2 or random	$\Phi_{total}(s)\frac{e^{-N_{total}\ \tau_{pu}}}{1-\left(1-e^{-N_{total}\tau}\right)\Phi_{part}(s)}$

where

$$\Phi_{part}(s) = \frac{N_{cont}\Phi_{cont}(s) + \sum_{\gamma} N_{\gamma}\Phi_{a}(s, E_{\gamma})}{N_{cont} + \sum_{\gamma} N_{\gamma}}$$

The combination

$$\text{pilepar} = e^{-N_{total}\ \tau_{pu}} \tag{1.13}$$

is the pileup parameter in the fit, where τ_{pu} is the residual counter pileup time, which is typically about 10% of the countrate deadtime. Strictly speaking N_{total} should be N_{true}, the true count rate when corrected for deadtime. The pileup parameter is always less one.

1.3.7 Absolute value switch

Note that all the mgfs have unit norms, with one exception. When the absolute value switch, 'abswitch' is set to a non-zero number, absolute calibration is in effect for synchrotron white-beam (or any continuum) data. The mgf for the continuum $\Phi_{cont}(s)$ is then **not** normalized to unity, but to the actual count rate predicted by theory. The user should

1. Freeze the continuum normalization 'cont norm' at 1.
2. Let the global norm N_{global} go free (the default anyway).
3. Let XSPEC do the fit and compare the fitted value of N_{global} with unity. If the theory of the source (synchrotron radiation model) is correct, if all the filter thicknesses are correctly fitted or measured, and if a model of the detector

quantum efficiency (relative q.e. model) is correct, the N_{global} should be close to unity, and the error bounds on N_{global} should provide a metric of the overall absolute calibration accuracy.

1.3.8 Number of channels parameter N_{chan}

The count rate spectrum $r\,(n)$ in channel n is obtained by numerically inverting $\Phi_{pileup}(s)$

$$\begin{aligned} r(n) &= \frac{1}{N\delta q}\sum_{k=0}^{N-1}\Phi_{pileup}\,(s)\left(j\frac{2\pi k}{N\delta q}\right)e^{j\frac{2\pi kn}{N}}, \qquad n=0,1,\ldots,N-1 \\ &= 0 \qquad\qquad\qquad \text{otherwise} \end{aligned}$$

As shown in the *Reference Guide*, we exploit the fact that $r(n)$ is real to cut down on the number of complex arithmetical operations to half the number of computational channels N. We have chosen

$$N = 2N_{chan}$$

The parameter N_{chan} itself, which is a frozen parameter, can be chosen to be a power of 2 **which is equal to or greater than the number of physical channels** N_{phys}, which is 4096 for SSD spectra and 512 for FPC spectra. Channels in excess of the number of N_{phys} are simply not plotted in XSPEC's graphical commands **plot** and **iplot**. Note that it is not necessary to set $N_{chan} = N_{phys}$, but it may be necessary to choose a large enough N_{chan} when there are continuum or pileup counts at very high channel numbers beyond what are actually counted in within N_{phys}.

1.3.9 Global norm N_{global}

When we perform an XSPEC fit we are actually fitting the inverse transform (inverse Fourier Transform for s imaginary) of:

$$\begin{aligned} &\frac{N_{pulser}\Phi_{pulser}(s)+N_{cont}\Phi_{cont}(s)+\sum_{\gamma}N_{\gamma}\Phi_{a}(s,E_{\gamma})}{N_{global}} \\ =\;& \left(\frac{N_{pulser}}{N_{global}}\right)\Phi_{pulser}(s)+\left(\frac{N_{cont}}{N_{global}}\right)\Phi_{cont}(s)+\sum_{\gamma}\left(\frac{N_{\gamma}}{N_{global}}\right)\Phi_{a}(s,E_{\gamma}) \\ =\;& (\text{pulser norm})\times\Phi_{pulser}(s)+(\text{cont norm})\times\Phi_{cont}(s)+\sum_{\gamma}(\text{line norm})_{\gamma}\times\Phi_{a}(s,E_{\gamma}) \end{aligned} \tag{1.4}$$

to the data. The last XSPEC fit parameter is the global norm parameter N_{global}. The actual number of counts for each of the components is therefore

$$N_{pulser} = N_{global}\times\left(\frac{N_{pulser}}{N_{global}}\right) = N_{global}\times\text{pulser norm}$$

$$N_{cont} = N_{global} \times \left(\frac{N_{cont}}{N_{global}} \right) = N_{global} \times \text{cont norm}$$

$$N_{\gamma} = N_{global} \times \left(\frac{N_{\gamma}}{N_{global}} \right) = N_{global} \times \text{line norm}_{\gamma} \tag{1.15}$$

Usually we freeze $(\text{line norm})_{\gamma}$ for one of the lines (a prominent line which takes center stage, say) at unity, then the global norm N_{global} is just N_{γ} for that line. Sometimes, it is more natural to freeze the continuum norm at unity, as when we are doing absolute calibration using synchrotron radiation using abswitch $\neq 0$.

1.4 Source/Filter Table

The source/filter types and associated parameters must be specified for the continuum. Sources and filters are treated the same. Imagine the source to be a white spectrum that passes through a number of filters, which are listed in the following table. The parameter associated with the filter is usually the **thickness of the material in centimeters**, except where noted. An exception is the synchrotron model, where the filter parameter is the sine of the half-angle of the aperture subtended at the tangent to the orbit. It may seem strange that a source should be treated like a filter, till one realizes that the source spectrum is the same as the spectrum produced by white light that has gone through a source 'filter'. The various parts of the detector that make up the detectors quantum efficiency are also treated as filters.

The source/filter table is as follows:

Table 1.2: JMKMOD source/filter table

filter	name	symbol	par (usually cm)
1	molybdenum	mo Mo	2.d-4
2	aluminum	al Al	1.d-4, 2.d-6 on fpc
3	zirconium	zr Zr	2.d-4
4	copper	cu Cu	0.5d-4
5	parylene	p1	
6	carbon	c C	2.5d-4
7	beryllium	be Be	2.d-4
8	boron	b B	1.d-4
9	chromium	cr Cr	1.d-4
10	iron	fe Fe	0.6d-4
11	nickel	ni Ni	0.5d-4
12	zinc	zn Zn	2.5d-4
13	magnesium	mg Mg	1.5d-3
14	titanium	ti Ti	4.d-4

Table 1.2: JMKMOD source/filter table (*cont.*)

filter	name	symbol	par (usually cm)
15	silver	ag *Ag*	1.d-4
16	indium	in *In*	5.d-4
17	vanadium	v *V*	2.d-3
18	cobalt	co *Co*	2.5d-3
19	polyimide	p2	1.065d-4
20	oxygen	o *O*	NA
21	hydrogen	h *H*	NA
22	nitrogen	n *N*	NA
23	argon	ar *Ar*	NA
24	xenon	xe *Xe*	NA
25	tin	sn *Sn*	1.d-4
26	niobium	nb *Nb*	1.d-4
27	tungsten	w *W*	1.d-4
28	mylar	pe	not implemented
29	teflon	p4	not implemented
30	ammonium dihydrogen phosphate pn	p5	not implemented
31	pvc	p6	not implemented
32	germanium	ge *Ge*	1.d-4
33	sillicon	si *Si*	1.d-4
34	fluorine	f *F*	NA
35	Kramers model	ks	NA
36	synchrotron model	sy	sin (half-aperture)
37	germanium detector q.e.	gx	1.d-1
38	fpc p10 q.e.	ax	5.36
39	fpc methane q.e.	mx	5.36
40	HRMA effective area	ea	meaninc switch
41	gaussian line	gn	NA
42	pella continuum	px	to be implemented
43	gold	au *Au*	1.d-4
44	iridium	ir *Ir*	1.d-4
45	unity(all-pass)	un	NA
46	manganese	mn *Mn*	1.d-4
47	polypropylene	p7	1.d-4
48	ice	ic	1.d-4

The absorption or transmission coefficients are calculated by looking up the Henke Tables for the components of the filters, where needed.

Filter 45, the unity or all-pass filter, is very useful as a 'filler' filter. When there are only four filters being used out of six, the remaining two may be set to 45.

The thicknesses of the filters are an important part of the fit, especially for absolute calibration.

1.4.1 Kramers Source

For the electron impact point source (EIPS), Kramers's bremstrahllung spectrum is:

$$N_c\left(E_\gamma\right) dE_\gamma = kZ\frac{\left(E - E_\gamma\right) dE_\gamma}{E_\gamma}$$

with

$$k = \frac{8\pi e^2}{3\sqrt{3}4\pi\varepsilon_0 hc^5 ml}$$

where the constants have their usual physics textbook meanins. Only the dependence $\frac{(E-E_\gamma)}{E_\gamma}$ is useful. The constant kZ multiplying into it is not used at all, and indeed is discarded when the spectrum is normalized to one (when abswitch = 0). Absorption effects of X-rays within the source, not included in Kramers' model, in included in Pella's model, but are not implemented in Version 7.0.

1.4.2 PTB/BESSY Source

When the detector lies in the orbital plane of the cyclotron orbit($\alpha_{synch} = 0$), the count rate within a dimensionless spectral range dy through an aperture whose half-angle is Δ is

$$\begin{aligned}\frac{N_{aperture}\left(y, \alpha_{synch} = 0\right)}{N_{synch}} dy \quad = \quad & \frac{3\sqrt{3}y}{20\pi^3} \cdot 4 \cdot \\ & \int_0^{\gamma \sin\Delta} \int_0^{2\pi} \left\{\left(1 + p_y^2\right)^2 K_{2/3}^2\left(\xi\right) + \left(1 + p_y^2\right) p_y^2 K_{1/3}\left(\xi\right)\right\} \cdot \\ & \arctan\left(\frac{\gamma^2 \sin^2\Delta - p^2}{\gamma\cos\Delta}\right) dp_y dy\end{aligned}$$

where

$$\xi = \frac{y}{2}\left(1 + p_y^2\right)^{3/2}$$

$$y = \frac{E_x}{E_s}$$

the ratio of the X-ray energy E_x(frequency ν_x)to the critical energy E_s(critical frequency ν_s)

$$E_s = h\nu_s$$

$$\nu_s = \frac{3}{2} \cdot \left(\frac{E_{synch}}{m_e c^2}\right)^2 \cdot \frac{eB_{synch}}{m_e c} \cdot \frac{1}{2\pi}$$

E_{synch} being the electron energy, B_{synch} being the magnetic induction, and the other symbols having their conventional meanings. The variable ψ is

$$N_{synch} = \frac{10\pi^2 I_{synch}}{\sqrt{3}e} \cdot \frac{E_{synch}}{m_e c^2} \cdot \frac{e^2}{hc}$$

for a current I_{synch}. Other symbols retain their usual physics textbook meanings.

The following is not implemented in JMKMOD Version 7.0:

1. The flux $N_{aperture}\,(y, \alpha_{synch})$when $\alpha_{synch} \neq 0$.
2. Finite source emittance.

1.4.3 Line source

This 'continuum' source is useful when the natural width of the line straddles a prominent feature in the filter, such as an absorption edge. The line pdf is a shifted gaussian:

$$\frac{1}{\sqrt{2\pi}\sigma_{line}} \exp\left(\frac{-\left(E_\gamma - E_{line}\right)}{2\sigma_{line}^2}\right)$$

Part III

JMKMOD Reference Guide

Chapter 2
PROPORTIONAL COUNTERS

2.1 Proportional Counter models

The operation of a proportional counter is well-known and described qualitatively and semi-quantitavely in many places (Blum and Rolandi[7], Fraser[16], Kleinknecht[27], Knoll[29], Fraser[16], Ku and Novick[32]). We use a model of the FPC response function which is based on the work of a long line of workers beginning with Prescott[38], Alkhazov[1], and leading up to Jahoda and McCammon[26]. In these works, the underlying probability distribution functions (Poisson, binomial, Polya pdfs, etc.) are often assumed and then used to build up a composite distribution for the pulse height spectra. Quantities such as the Fano factor and Polya h factor are introduced *ad hoc* without much physical basis, except as fitting parameters. All these pdfs come from the solutions to a model that treat the detector as a medium in which particles are created through ionization and lost through recombination. In many other disciplines, such as cosmic ray physics or queuing theory, these are known as a birth-and-death processes. In the simple birth-and-death process which we consider here, the birth(ionization) rates and death(recombination) rates are assumed to be constants or linear functions of the particles already present. In the so-called drift region far from the anode wire, we find that the solution is the binomial distribution, with the Fano factor being directly related to the ratio of the death rate to the birth rate. The bigger the Fano factor, the bigger the variance of the distribution. In the avalanche region where the electric field is high, far from the drift region where the field is low, we find that the appropriate distribution is the negative binomial or Polya distribution, with the Polya h factor being the ratio of the actual gain to the gain calculated by classical Townsend discharge theory. This physical interpretation of the Fano factor F and the Polya h factor seems to be new and gives a good deal of insight into the model. The birth-and-death model may therefore taken as fundamental which provides a framework for further modifications and improvements, to account for spatial inhomogeneities, geometrical effects, incomplete charge collection through electron loss, etc. Birth-and-death calculations gives us not the probability distribution functions(pdfs) themselves, but the moment generating functions (mgfs or Laplace Transforms) of the pdfs. These cannot be inverted analytically, but can be inverse Fourier transformed into the desired pdfs using the Fast Fourier Transform. The pdfs constitute the JMKMOD detector models.

A proportional counter is a gas-filled chamber with a high positive voltage on an anode wire running through it. An X-ray enters the counter through a window, ionizes a gas atom and produces an immediate *primary* photoelectron. This first photoelectron has an energy equal to the photon's energy minus an offset. This primary photoelectron then ionizes several other atoms, producing a cloud of secondary electrons. Some of these secondary electrons can be lost via electron diffusion. The primary photoelectron is also accompanied by a cascade of Auger electrons and/or fluorescent photons. Typically these photo- and Auger mean free paths are short (0.1 - 1 mm) in common counting gases at atmospheric pressure. Fluorescent yields for K- and L-shell escape peaks are small and may be treated as a second order effect. The average number of secondary electrons produced by the primary photoelectron is proportional to the photon's original energy. After the cloud of secondary electrons are created, they drift towards the high-field anode wire region.

When each electron gets within a critical radius of the anode wire, the kinetic energy gained from the accelerating voltage produces a second ionization process: the so-called Townsend discharge. Each secondary electron from the intial ionization produces a barrage of about 10^4 electrons (called avalanche electrons), which appears as a voltage pulse in the measuring electronics. The pulse height distribution of this pulse which is launched by a single secondary electron is called the single electron spectrum (SES)[38]. Since there are many secondary electrons, the mean magnitude of the signal, obtained by summing over all the individual pulses, is proportional to the energy of the incident photon. Because of the statistical uncertainties in the ionization and discharge process, there is a good deal of spread about the mean magnitude of the signal. The probability distribution function (pdf) of this spread is known as the detector's *pulse height response function*.

For detector modeling, we must translate the qualitative picture above into a quantitative one. This quantification has been attempted by many authors. Prescott[38] proposed a response function for a scintillator counter, then suggested that the model of a proportional counter should be similar to it. He used a simple exponential distribution for the SES from the photomultiplier surface, but he mentioned the possible use of a Polya distribution for the SES, and the possibility of non-Poissonian statistics in the initial ionization. Alkhazov[1] proposed a pdf which uses a Polya distribution weighted by a binomial distribution, but did not give an *analytical* expression for its moment generating function (mgf), which we have found to be so crucial for data fitting. Jahoda and McCammon[26] adopted Alkhazov's suggestion of using the Polya distribution for the SES, but computed a pdf which consists of Polya distributions weighted by gaussians whose variances are characterized by a Fano factor. Jahoda and McCammon also suggested another modification: replacing the weighting gaussians by modified gaussians which takes into account electron diffusion losses. Their method of calculating a detector response function from *known* parameters is computationally very intensive, and is not useful as a fitting function when the detector parameters themselves are unknown. Moreover, their accounting of secondary elec-

tron losses through electron diffusion, using Inoue's model,[25] awaits further study.

We derive a simple model for the proportional counter response function based on a Polya (also known as a negative binomial) distribution weighted by a binomial distribution. We show that both the binomial distribution for the secondary electrons and the negative binomial distribution distribution for the avalanche electrons come from a basic birth-and-death stochastic model for electron creation (birth) through ionization, and loss(death) through recombination. The FPC response function so obtained is then used in various ways to fit some measured spectra, which may include discrete lines and bremsstrahlung continuum from an electron impact point source.

2.2 Photoionization: the Binomial Distribution

In this section, we derive the distribution of the secondary electrons created by a primary photoelectron — the binomial distribution, from the ionization and recombination process described by a difference-differential equation commonly used in the theory of cosmic rays. The deduction of the Fano factor in terms of the ratio of the recombination to ionization rates (Eq. (2.11) below) as a consequence of this formulation is new.

As described above, the basic model of the proportional counter is that of a photoionization process in the body of the gas followed by an electron multiplication process in the high field region close to the anode wire. Let E_γ be the energy of the incident photon, and let $E_\mathbf{0}$ be the work function or energy lost in the process of converting the photon energy into a photoelectron. We assume here there are no losses due to diffusion, which cause incomplete charge collection effects, L-shell photon escapes, and Auger electron losses. This fundamental photoelectron is created with a probability given by a so-called **quantum efficiency** η_e. Let $\langle w \rangle$ be the average photoelectron energy required to produce an ion-electron pair along the entire track length of the ionizing particle. Values of w measured with photons and electrons are experimentally found to be the same[7]. Then the average number $\langle n \rangle$ of secondary electrons produced is given by

$$\langle n_{se} \rangle = \frac{E_\gamma - E_\mathbf{0}}{\langle w \rangle} \tag{2.1}$$

not counting the fundamental photoelectron. Values for $E_\mathbf{0}$ range from 0 to 100 eV, and have in general different values between absorption edges the gas. [6]. Values for w range experimentally from 20 to 30 eV in counter gases[6]. The following table for w is taken out of a compilation by Christophorou[14]:

Table 1. Average energy w spent for the creation of one ion-electron pair in various gases and a mixture; $\mathbf{w}_\alpha$ and $\mathbf{w}_\beta$ are from measurements using alpha-particles and electrons(or photons), respectively. The lowest(first) ionization potential I is typically 1.5 to 3 times smaller. From Christophorou[14].

Gas	$\langle w_\alpha \rangle\,(eV)$	$\langle w_\beta \rangle\,(eV)$	$I\,(eV)$
Ar	26.4	26.3	15.76
CH_4	29.1	27.1	12.99
Ar(97%)+CH_4 (3%)†	26.0	$\sim$	$\sim$
Air	35.0	33.8	12.15
H_2O	30.5	29.9	12.60

†The quoted concentration is the one giving the smallest w.

We normally use 28 eV for FPCs (and 3 eV for SSDs).

Let the *maximum* number $n_{\max}$ of secondary electrons created (the number is finite since E_γ is finite) be related to $\langle n \rangle$ by

$$n_{\max} \equiv \frac{\langle n_{se} \rangle}{(1-F)} = \frac{E_\gamma - E_0}{\langle w \rangle\,(1-F)} \tag{2.2}$$

where F will be identified with the Fano factor and has values less than 1. The combination $\langle w \rangle\,(1-F)$ may be taken to be the value of the *smallest* electron creation energy, $\langle w \rangle$ being the average. The following table is taken from Hendricks[21] showing the fluctuations of the average potential required to produce one electron, albeit in the high-field avalanche region(see discussion on gas gain below).

Table 2. Measured values of w for various gases (from Hendricks[21]), showing fluctuations in w in the high field region. The ratio of the fluctuation in w to w itself may be taken to be an effective Fano factor.

Gas	$\langle w \rangle \pm \langle w \rangle\,F\,(eV)$	F
Ar(90%) + CH_4 (10%)	23.6±5.4	0.23
Ar(95%) + CH_4 (5%)	21.8±4.4	0.20
CH_4	36.5 ± 5.0	0.14

Let x be the distance traversed by the primary photoelectron in this weak field region (sometimes known as the 'drift' region), creating secondary electrons while on its way. Let $p_n(x)$ be the probability of n secondary electrons generated by the primary photoelectron as a function of the distance x traversed by the photoelectron. We assume that

1. The probability of another secondary electron being created (birth) via ionization in a small distance Δx, when there are already n secondary electrons present, is proportional to Δx and to the number of secondary electrons $n_{\max} - n$ still capable of being created, $\frac{\lambda(n_{\max}-n)}{n_{\max}}\Delta x$, where λ, the effective ionization rate is assumed to be a constant and $n \leq n_{\max}$.

2. Each electron has the probability $\mu \Delta x$ of 'dying' via recombination during the same interval Δx, where μ, the effective recombination rate is assumed to be a constant.

Let $p_n(x)$ be the probability of there being n secondary electrons generated by the initial photoelectron. Let x be the path length traversed by this photoelectron as it goes about creating more and more secondaries, whose own path lengths are parameterized by the same variable x. The difference-differential equations describing this birth-and-death process for $p_{n_{se}}(x)$ is

$$\begin{aligned} p'_{n_{se}}(x) &= -\left(\frac{\lambda(n_{\max}-n_{se})}{n_{\max}}+n_{se}\mu\right)p_{n_{se}}(x)+(n_{se}+1)\mu p_{n_{se}+1}(x)+ \\ &\quad \frac{\lambda\left[n_{\max}-(n_{se}-1)\right]}{n_{\max}}p_{n_{se}-1}(x) \qquad (2.3) \\ \text{for } 0 &\leq n_{se}\leq n_{\max}, \\ &= 0, \text{ otherwise} \end{aligned}$$

and

$$p'_0(x) = -\lambda p_0(x)+\mu p_1(x) \qquad (2.4)$$

The initial condition is that there are no secondary electrons(the initial photoelectron doesn't count since $p_n(x)$ is the pdf for secondaries created by it),

$$p_0(0) = 1 \qquad (2.5)$$

We show in the Appendix that this set of equations has the solution

$$p_{n_{se}}(x) = \frac{n_{\max}!}{n_{se}!\,(n_{\max}-n_{se})}F(x)^{n_{\max}-n_{se}}\left[1-F(x)\right]^{n_{se}} \qquad (2.6)$$

with mean

$$\langle n_{se}(x)\rangle = \frac{n_{\max}\lambda(1-e^{-\frac{(\lambda+\mu n_{\max})}{n_{\max}}x})}{\lambda+\mu n_{\max}} = n_{\max}\left[1-F(x)\right] \qquad (2.7)$$

This has the steady state limit as $x\to\infty$ of

$$\langle n_{se}\rangle = \frac{n_{\max}\lambda}{\lambda+\mu n_{\max}} = n_{\max}\left[1-F\right] \qquad (2.8)$$

and variance

$$\left\langle \Delta n_{se}^2\right\rangle = \langle n_{se}\rangle F \qquad (2.9)$$

where $F\equiv F(\infty)$. This shows the significance of the Fano factor F — it is the fraction of the variance relative to Poisson variance. Using this relationship, Fano calculated F directly for hydrogen[15]. It is also significant that F physically relates the 'death' rate μ to the birth rate λ, which we get by solving for F from Eqs. (2.8) and (2.2):

$$\frac{\mu n_{\max}}{\lambda} = \frac{\mu \langle n_{se} \rangle}{\lambda (1-F)} = \frac{F}{1-F} \tag{2.10}$$

or

$$F = \frac{\frac{\mu n_{\max}}{\lambda}}{1 + \frac{\mu n_{\max}}{\lambda}} = \frac{\mu \langle n_{se} \rangle}{\lambda} \tag{2.11}$$

Physically, the Fano factor is therefore essentially the ratio of a mean recombination to the ionization rate. In general F is a function of energy. From Eq. (2.11) we also get a relation between the mean and the maximum number of secondaries

$$\langle n_{se} \rangle = \frac{n_{\max}}{1 + \frac{\mu n_{\max}}{\lambda}} \tag{2.12}$$

Eliminating F entirely between Eq. (2.9) and Eq. (2.11), we get the relationship

$$\frac{\langle \Delta n_{se}^2 \rangle}{\langle n_{se} \rangle^2} = \frac{\mu}{\lambda} \tag{2.13}$$

Eq. (2.6) is nothing more than the binomial distribution. When

$$F = \frac{\mu \langle n_{se} \rangle}{\lambda} \to 1 \tag{2.14}$$

$n_{\max} \to 1$, and the binomial distribution becomes the Poisson distribution, which is the distribution for secondaries used by Prescott ([38]).

2.3 Statistics of electron avalanche growth: the Negative Binomial (Polya) Distribution for Avalanche Electrons

Each secondary electron causes an avalanche of further ionizations in the high-field region close to the anode wire. In what follows we show that a simple "birth-and-death" process, which is a generalization of the Poisson and Furry process[41] in cosmic ray theory, produces a single electron spectrum solution known variously as the negative binomial pdf, or Polya pdf. This pdf is found to be a close fit to experimental single electron data from UV excitation[26].

The high-field region is entirely distinct from the drift region of the previous section[16]. Let x be the distance traversed by a secondary electron in this high-field region. We assume, following Arley[2] that

1. The probability of an electron being created (birth) in a small distance Δx is proportional to Δx, $\lambda \Delta x$, where λ , the effective ionization rate is assumed to be a constant, and this probability is independent of the number already present. These *trigger* electrons (Loeb[33], p.65), which are generated in the background (by the secondary ionization process described above) , are essential to the avalanche process

2. One electron by traveling a distance Δx can be converted into two electrons with a probability proportional to Δx, $\frac{\lambda \Delta x}{h}$, where h is a further constant. This is the so-called *Furry* or *avalanche* component.

3. Each electron has a probability $\mu \Delta x$ of 'dying' during the same distance Δx, where the effective recombination rate μ is assumed to be constant. We note, however, that Arley obtained a solution only by assuming that this effective recombination rate is zero (i.e. a pure birth process).

The birth and death rates λ and μ in the avalanche region are given the same notation as the birth and death rates in the drift region for economy of notation, although they are physically distinct. Since we don't use them beyond the sections where they are defined, there is no confusion.

Let $\mathrm{r}_m(x)$ be probability of m avalanche electrons being generated. The difference-differential equations for the probabilities $r_m(x)$ are

$$r'_m(x) = -\left[\lambda(1+\frac{m}{h}) + m\mu\right] r_m(x) + (m+1)\mu r_{m+1}(x) + \lambda\left(1+\frac{m-1}{h}\right) r_{m-1}(x) \tag{2.15}$$

for all m except

$$r'_0(x) = -\lambda r_0(x) + \mu r_1(x) \tag{2.16}$$

The solution to these equations is the negative binomial or Polya distribution given in the Appendix:

$$\begin{aligned} r_m(x) &= \frac{\Gamma(m+h)}{\Gamma(h)\Gamma(m+1)} \frac{\langle m(x)\rangle^m h^h}{(h+\langle m(x)\rangle)^{m+h}} \\ &= \frac{\Gamma(m+h)}{\Gamma(h)\Gamma(m+1)} \left[1+\frac{h}{\langle m(x)\rangle}\right]^{-m} \left[1+\frac{\langle m(x)\rangle}{h}\right]^{-h} \end{aligned} \tag{2.17}$$

with mean

$$\langle m(x)\rangle = \frac{h\lambda(e^{(\frac{\lambda}{h}-\mu)x}-1)}{\lambda-\mu h} \tag{2.18}$$

and variance

$$\begin{aligned} \left\langle \Delta m^2(x)\right\rangle &= \left\langle m^2(x)\right\rangle - \langle m(x)\rangle^2 = \left.\frac{\partial^2 \ln P(z,t)}{\partial \ln z^2}\right|_{z=1} \\ &= \langle m(x)\rangle\,(1+\frac{\langle m(x)\rangle}{h}) \end{aligned} \tag{2.19}$$

Since $\langle m\,(x_d)\rangle$ is the mean number of avalanche electrons created by a single secondary electron in a high-field region of thickness x_d, it is called the *gas amplification factor* A_g[27], or the *gas gain*:

$$A_g \equiv \langle m\,(x_d)\rangle = \frac{h\lambda(e^{(\frac{\lambda}{h}-\mu)x_d}-1)}{\lambda-\mu h} \tag{2.20}$$

Notice that the gain shows that there is a threshold for the ionization rate

$$\lambda = \mu h \tag{2.21}$$

beyond which gas amplification sets in. This threshold behavior, which is predicted by our our simple birth-and-death model, is exactly the behavior described in Kleinknecht[27]. Eq.(2.20)can be written as an integral:

$$A_g = \frac{h\lambda(e^{(\frac{\lambda}{h}-\mu)x_d}-1)}{\lambda-\mu h} = \lambda\int_0^{x_d} e^{(\frac{\lambda}{h}-\mu)x}dx \tag{2.22}$$

This shows that $e^{(\frac{\lambda}{h}-\mu)x}$ is the amplification for a single electron, which is integrated over the background trigger electron level (assumed to be constant) to give A_g[70]. Comparing our single electron amplification factor $e^{(\frac{\lambda}{h}-\mu)x}$ with the expression in Kleinknecht[27], which we will call here the Townsend gas gain $A_{Townsend}$, viz.

$$A_{Townsend} \equiv \exp\left[k\sqrt{U_0}\left(\sqrt{\frac{U_0}{U_c}}-1\right)\right] \tag{2.23}$$

we may make the following identifications:

$$\mu x = k\sqrt{U_0} \tag{2.24}$$

and

$$\frac{\lambda}{h}x = \frac{kU_0}{\sqrt{U_c}} \tag{2.25}$$

Here U_0 is the high voltage applied to the counter, U_c is the ionization threshold voltage, and k a constant. There are many formulations for k itself, some more empirical than others. For instance, Rose and Korff[42] have derived a formula for k which depends on the wire radius, capacitance of the wire per unit length and the density of the gas. Others have worked out their own semi-empirical models of the gas gain[10]. After making these identifications (2.24) and (2.25), we substitute the expressions back into the equation for $\langle m\,(x_d)\rangle$ to get our expression for gas gain:

$$A_g \equiv \langle m\,(x_d)\rangle = h\frac{\exp\left[k\sqrt{U_0}\left(\sqrt{\frac{U_0}{U_c}}-1\right)\right]-1}{1-\sqrt{\frac{U_c}{U_0}}} \tag{2.26}$$

This is typically a large number. In our calibration work on the proportional counters, the gas pressure and U_0 are adjusted so that A_g ranges from $20,000$ for 100

eV X-rays to 2,000 for 10 keV X-rays, using methane gas (400 T) and P10 (150 T) respectively[67]. This adjustability is needed to bring the X-ray region of interest within range of the available number of channels. The formula for gas gain (2.26) and the identities (2.24) and (2.25) also give h as the ratio:

$$\frac{1}{h} = \frac{\mu}{\lambda}\sqrt{\frac{U_0}{U_c}} = \frac{A_{Townsend} - 1}{A_g\left(1 - \sqrt{\frac{U_c}{U_0}}\right)} \approx \frac{A_{Townsend}}{A_g} \tag{2.27}$$

for $U_0 \gg U_c$. The Polya h factor, which is the relative gas gain variance, is thus traced to *the ratio of the actual gain to that predicted by the Townsend theory* or the average *number of particles*, originating from a secondary electron, *that started the Townsend avalanche* (which is, by definition, started by a single electron). It is also the ratio of the ionization rate to the recombination rate, multiplied by the ratio of the speed of the electron at threshold to that at collection. Comparing Eq. (2.27) with Eq. (2.11) we see that $\langle n_{se}\rangle$ and $\sqrt{\frac{U_0}{U_c}}$ appear in similar roles as amplification factors. In analogy to Eq. (2.13) we also have the relation

$$\frac{\langle \Delta m^2\,(x_d)\rangle}{A_g^2} = \frac{1}{A_g} + \frac{1}{h} \tag{2.28}$$

obtained by eliminating h between Eqs.(2.19) and (2.27).

The parameter

$$\chi_{Raether} \equiv \frac{h}{\lambda}\frac{U_0}{U_c x_d} = \frac{1}{\mu x_d} \tag{2.29}$$

corresponds to Raether's parameter. Raether[39] showed that $h \to 1$ (simple exponential SES) as $\chi_{Raether} \gg 1$ (pure Furry process, see Appendix).

2.4 Combining Secondary Electron Distribution with Avalanche Distribution to give Distribution of Electrons Hitting Wire

In this section we combine the distributions for the secondary and avalanche electrons and deduce the moment generating function (mgf) of the detector response function in closed form. This approach was used by Prescott. He inverted his mgf in closed form to obtain the Prescott function. In general, however, the mgf for the detector response function must be inverted numerically, an approach we take below.

Summarizing our results in the previous sections, we have shown that the pdf of the number n of secondary ion-electron pairs created is the binomial distribution given above

$$p_{n_{se}} = \frac{n_{\max}!}{(n_{\max} - n_{se})!n_{se}!}F^{n_{\max}-n_{se}}(1-F)^{n_{se}} \tag{2.30}$$

Each one of these ion-electron pairs then produce avalanche electrons whose numbers m are governed by the negative binomial single electron spectrum (SES)

$$r_m = \frac{\Gamma(m+h)}{\Gamma(h)\Gamma(m+1)} \frac{A^m h^h}{(h+A)^{m+h}} \tag{2.31}$$

Let the total number of electrons hitting the wire be l. Then the probability ϕ_l of l electrons hitting the wire is

$$\phi_l = \sum_{n_{se}=0}^{\infty} p_{n_{se}} \sum_{l=m_0+m_1+m_2+\ldots m_{n_{se}}} \underbrace{r_{m_0} r_{m_1} r_{m_2} \ldots r_{m_{n_{se}}}}_{\text{product of } n_{se}+1 \text{ SES}} \tag{2.32}$$

In the second summation, the sum is over all combinations of $m_0, m_1, m_2, \ldots m_{n_{se}}$ subject to the constraint $l = m_0 + m_1 + m_2 + \ldots m_{n_{se}}$. This is just the *discrete convolution* of the r_ms. Note that the convolution is performed over $n_{se}+1$ SES, because of the avalanche bunch created by the initial photoelectron *in addition* to those created by the n secondary electrons. If we take the Z Transform of ϕ_l, we get

$$\begin{aligned} \Phi(z) &\equiv \sum_{n_{se}=0}^{\infty} p_{n_{se}} R(z)^{n_{se}+1} \\ &= R(z) P[R(z)] \end{aligned} \tag{2.33}$$

where the Z-Transform (defined in the Appendix) of the binomial distribution is

$$P(z) = [1 - (1-F)(1-z)]^{n_{\max}} \tag{2.34}$$

and the Z-Transform of the negative binomial distribution (also defined in the Appendix) is

$$R(z) = \left[1 + \frac{A_g}{h}(1-z)\right]^{-h} \tag{2.35}$$

In the last step we have invoked the theorem[35] that the Z Transform of a convolution is the product of Z Transforms of the individual sequences.

2.5 Charge-to-Voltage Conversion in the MCA

The basic detector electronics consist of standardized nuclear instrumentation modules (NIM), the key components of which are the preamplifier and the amplifier[12]. Leaving aside its important pulse-shaping function, the preamplifier simply provides an output pulse proportional to the incoming electron count from the detector. The amplifier then further amplifies this input signal. Let the effective capacitance of the detector-preamplifier combination be $C_{\rm det}$, and let the amplifier output be $V_{amplifier}$. The multichannel analyzer (MCA) collects voltage pulses that fall into N_{MCA} bins. Then the 'channel number' q into which a pulse falls due to an incoming electron count of n is

$$q = \frac{eN_{MCA}}{C_{\rm det}V_{amplifier}}n \tag{2.36}$$

where e is the electron charge 1.6×10^{-19} coulombs giving a scale change or 'electronic gain' μ_e of

$$\begin{aligned}\mu_e &\equiv \frac{q}{n} = \frac{eN_{MCA}}{C_{\rm det}V_{amplifier}} \\ &= 55\times10^{-6}\left(\frac{N_{MCA}}{512}\right)\left(\frac{C_{\rm det}}{0.15\ pF}\right)^{-1}\left(\frac{V_{amplifier}}{10\ V}\right)^{-1}\end{aligned} \tag{2.37}$$

channels per incoming electron

This is a small number, but by adjusting the gas gain so that the number of incoming electrons is of the order 10^5-10^6 between 100 eV and 10 keV, we can position the peaks of the spectra (or other region of interest) comfortably within the range of available channels. The other significance of μ_e is that it is the 'granularity' along the MCA 'axis'. Its smallness suggests that we replace the pdf for discrete avalanche electrons (the negative binomial distribution above) by a pdf for channels, but treating the channel numbers as continuous. This resulting pdf for continuous channels is called the single electron spectrum (SES)[38]. The discrete-to-continuous conversion of the pdfs is a problem of digital-to-analog conversion(DAC) in digital signal processing(DSP). In DSP[35] it is covenient to work directly with the Z Transforms of the distributions, rather than the distributions themselves. One transformation which maps the Z-plane of discrete sequences into the s-plane of continuous functions is the transformation that replaces derivatives by differences[35]:

$$z = 1 - \mu_e s \tag{2.38}$$

to the Z-Transform of the SES

$$R(z) = \left[1 + \frac{A_g}{h}(1-z)\right]^{-h} \tag{2.39}$$

we get the Laplace Transform $P(s)$, which we will call the mgf of the SES:

$$R_a(s) = \left[\frac{1}{1+\frac{\mu s}{h}}\right]^h \tag{2.40}$$

Let the channel number variable q be conjugate to the Laplace parameter s. The mean and variance of the pdf corresponding to this mgf are given by

$$\mu \equiv \langle q\rangle = A_g\mu_e \tag{2.41}$$

and

$$\left\langle \Delta q^2 \right\rangle = \frac{\mu^2}{h} \tag{2.42}$$

The inverse Laplace Transform of Eq. (2.40) is the SES

$$r(q)\,dq = \frac{1}{\Gamma(h)} \left(\frac{\mu}{h}\right)^{-h} q^{h-1} e^{-\frac{hq}{\mu}}\,dq \tag{2.43}$$

for the continuous channel variable q. This pdf is what Jahoda and McCammon calls a "Polya Distribution"[26], but we identify it more properly as a Gamma Distribution or an Erlangian Distribution[28]. The 'true' Polya Distribution is, strictly speaking, the negative binomial distribution Eq. (2.31) above for discrete variables.

The detector response function or the probability distribution of electrons hitting the anode wire is then a weighted sum of SESs over a binomial distribution. We take the Z-transform of the response function and map the Z-plane into the s-plane. We get for the pulse-height mgf

$$\begin{aligned} \Phi_a(s) &= R_a(s) Q(R_a(s)) \\ &= R_a(s)(1-(1-F)(1-R_a(s)))^{\frac{\langle n_{se}\rangle}{1-F}} \\ &= \left[\frac{1}{1+\frac{\mu}{h}s}\right]^h \left[1-(1-F)\left(1-\left[\frac{1}{1+\frac{\mu}{h}s}\right]^h\right)\right]^{\frac{\langle n_{se}\rangle}{1-F}} \end{aligned} \tag{2.44}$$

The mean and variance for this mgf are the first two cumulants in the series expansion of this mgf:

$$\langle q \rangle = \mu(\langle n_{se}\rangle + 1) \tag{2.45}$$

and

$$\left\langle \Delta q^2 \right\rangle = \langle q \rangle \left[\mu \frac{\langle n_{se}\rangle}{(\langle n_{se}\rangle + 1)} \left(F + \frac{1}{h}\right)\right] \tag{2.46}$$

respectively. The resolution

$$\begin{aligned} \frac{FWHM}{\langle q \rangle} &\equiv \frac{2\sqrt{2\ln 2}\sqrt{\langle \Delta q^2 \rangle}}{\langle q \rangle} \\ &= \frac{2.36\sqrt{\mu \frac{\langle n_{se}\rangle}{(\langle n_{se}\rangle+1)}\left(F+\frac{1}{h}\right)}}{\sqrt{\mu(\langle n_{se}\rangle+1)}} \\ &\approx \frac{2.36\sqrt{\left(F+\frac{1}{h}\right)}}{\sqrt{\langle n_{se}\rangle}} \qquad for \quad \langle n_{se}\rangle \gg 1 \end{aligned} \tag{2.47}$$

is of the same form as the resolution of a proportional counter described in Fraser[16]. The fraction $f = \frac{1}{h}$ is referred to by Fraser as the *relative variance of the gas gain*.

2.6 Reduction of the General Response Function to the Prescott functions

In this section we deduce the now-classic Prescott functions by specializing our detector response function. In general, the response function mgfs with arbitrary F, h and μ do not have analytical pdfs, but must be inverted numerically, with *one* exception. In the limit $F \to 1$, $h \to 1$, the mgf becomes

$$\Phi_a(s) = \frac{1}{1+\mu s} \cdot e^{-\langle\ n_{se}\rangle} e^{\frac{\langle n_{se}\rangle}{1+\mu s}} \tag{2.48}$$

which is the Laplace Transform of the pdf

$$\phi\left(q, \langle n_{se}\rangle\right) = \frac{e^{-\langle n_{se}\rangle - \frac{q}{\mu}} I_0(2\sqrt{\frac{\langle n_{se}\rangle q}{\mu}}}{\mu} \tag{2.49}$$

with mean

$$\langle q\rangle = \mu(\langle n_{se}\rangle + 1) \tag{2.50}$$

and variance

$$\left\langle \Delta q^2 \right\rangle = 2\mu \left\langle q\right\rangle \frac{\langle n_{se}\rangle}{(\langle n_{se}\rangle + 1)} \tag{2.51}$$

I_0 is the zeroth order Bessel function. By taking the Inverse Laplace Transform of the mgf *with the primary electron omitted*, viz.

$$\Phi_a(s) = e^{-\langle n_{se}\rangle} e^{\frac{\langle n_{se}\rangle}{1+\mu s}} \tag{2.52}$$

we get [38]:

$$\varphi_{\mathrm{Prescott}}\left(q, \langle n_{se}\rangle\right) = \frac{e^{-\langle n_{se}\rangle - \frac{q}{\mu}} \left[\sqrt{\frac{\mu\langle n_{se}\rangle}{q}} I_1\left(2\sqrt{\frac{\langle n_{se}\rangle q}{\mu}}\right) + \delta\left(\frac{q}{\mu}\right)\right]}{\mu} \tag{2.53}$$

with mean

$$\langle q\rangle = \mu \left\langle n_{se}\right\rangle \tag{2.54}$$

and variance

$$\left\langle \Delta q^2 \right\rangle = 2\mu \left\langle n_{se}\right\rangle \tag{2.55}$$

I_1 is the first order Bessel function. Note the delta function $\delta\left(\frac{q}{\mu}\right)$ within the Prescott function at the origin, as first noted by Prescott[38]. If the number of secondary electrons created is large, the absence and presence of the primary electron is practically insignificant. However, for low energy incoming X-rays, the presence of the primary electron may not be negligible. Prescott considered the statistics of electrons in a photomultiplier tube due to incoming Poisson light, with and without spontaneous emisson of an electron from the photocathode, and Eqs.(2.49)and (2.53)are functions named after him for these two cases.

2.7 FPC Operational Settings during Experiments

During actual calibration experiments the FPC is operated so that the FWHM is held nearly constant[68], as part of a procedure to 'optimize' the operational settings (anode wire voltage, amplifier gain, lower level discriminator, etc.). The anode wire voltage, the gas pressure and the electronic gain combine to produce the total gain. By adjusting the gas pressure, we can adjust the total gain μ so that, from Eq. 2.47 above,

$$FWHM = 2.36\sqrt{\left(F+\frac{1}{h}\right)}\mu\sqrt{\frac{E_\gamma}{w}} \text{ is a constant} \tag{2.56}$$

This may be done to make quick-look analyses of main lines easier. The mean channel corresponding to a main peak at E_γ is given by Eq. (2.45) above (for negligible E_0)

$$\langle q\rangle = \mu\frac{E_\gamma}{w} \tag{2.57}$$

Eliminating the gain μ (which is adjusted to vary as $E_\gamma^{-1/2}$) between these two equations, we get a constant channel width of the main peak:

$$\begin{aligned} FWHM &= 2.36\sqrt{\left(F+\frac{1}{h}\right)}\frac{\langle q\rangle}{\sqrt{\frac{E_\gamma}{w}}} \\ &= 2.36\times 250\times\sqrt{\frac{28}{10,000}}\sqrt{\left(F+\frac{1}{h}\right)}\frac{\frac{\langle q\rangle}{250}\sqrt{\frac{w}{28eV}}}{\sqrt{\frac{E_\gamma}{10\,keV}}} \\ &= 31.2\sqrt{\left(F+\frac{1}{h}\right)}\frac{\frac{\langle q\rangle}{250}\sqrt{\frac{w}{28eV}}}{\sqrt{\frac{E_\gamma}{10\,keV}}}\ channels \end{aligned} \tag{2.58}$$

If the main peak at 10 keV is chosen to be at channel 250, the $FWHM$ is about 40 channels.

2.8 Asymptotic formulae

The author has found the approximate expression for the JM model by inverting its Fourier transform in terms of elementary functions (powers and exponentials). This is an approximate expression for the JM model in the same sense that

$$f(q) = (4\pi)^{-\frac{1}{2}} N^{\frac{1}{4}} \left(\frac{q}{\mu}\right)^{-\frac{1}{4}} \exp\left\{2\sqrt{Nq/\mu} - q/\mu - N\right\} \tag{2.59}$$

is an approximation to the Prescott model pdf for large q:

$$f(q) = N^{\frac{1}{2}} \mu^{-\frac{1}{2}} e^{-N} q^{-\frac{1}{2}} \exp\left(-\frac{q}{\mu}\right) I_1\left\{2\sqrt{Nq/\mu}\right\} \tag{2.60}$$

where q is the channel number, μ is the gain, N is the average number of ion-electron pairs. Since the Bessel function is expensive to compute, all codes (Chartas, current MSFC XSPEC model, etc.) uses Eq. (2.59). However, it suffers from the defect that it does not have unit normalization that a true pdf does, and therefore the line count obtained from XSPEC using Eq. (2.59) will be an approximate line count.

I will quote my result

$$\begin{aligned} f(q) \;=\; & \frac{1}{\sqrt{2\pi}} \exp(qs_o) \frac{1}{\left(1+\frac{\mu s_o}{h}\right)^h} \left\{1 + (1-F)\left[-1 + \frac{1}{\left(1+\frac{\mu s_o}{h}\right)^h}\right]\right\}^{\frac{N}{1-F}} \times \\ & \left[\frac{h\mu^2}{2(h+\mu s_o)^2} \left\{1 + \frac{N\left[1-F+F(1+h)\left(1+\frac{\mu s_o}{h}\right)^h\right]}{\left[1-F+F\left(1+\frac{\mu s_o}{h}\right)^h\right]^2}\right\}\right]^{-\frac{1}{2}} \end{aligned} \tag{2.61}$$

where s_o a root of

$$q - \frac{h\mu}{h+\mu s_o} + \frac{h\mu N}{(h+\mu s_o)\left[-1+F-F\left(1+\frac{\mu s_o}{h}\right)^h\right]} = 0 \tag{2.62}$$

which must be solved for numerically for general h, but is a quadratic for the special case of $h = 1$. Like the approximate Eq.(2.59), it will not have unit normalization. Eq.(2.61) is the first term of an asymptotic expansion, and is therefore good for large q. I have not investigated this question, but I am pretty sure that it valid under the same conditions that (2.59) is valid. Preliminary calculations indicate that (2.61) reduces to (2.59)when $h \to 1$ and $F \to 1$.

A remark about this asymptotic development. The asymptotic expression for the JM model is much more complicated than that for the Prescott model. Like the approximate Prescott model, it also suffers from the fact that **it does not have exact unit normalization**, and this will affect the all-important line count we are trying to fit.

Chapter 3
SOLID-STATE DETECTORS

3.1 SSD mgf

A rough first model for the SSD response function is a gaussian with a constant preamp noise variance. However, we know that there is a fundamental Fano limit for these detectors[16], and the fwhm of George Chartas' old data shows a non-negligible Fano component[9]. The average number $\langle n \rangle$ of secondary electrons produced by a photon of energy E in the depletion layer is given by

$$< n >= \frac{E - E_0}{w} = (1 - F) n_{\max} \tag{3.1}$$

not counting the fundamental photoelectron. Here f is the Fano factor, and n_{max} is the largest possible number of secondary electrons produced. We assume that the discrete pdf of the secondary electrons in the SSD is, like the FPC, given by the binomial distribution

$$p_n = \binom{n_{\max}}{n} F^{n_{\max} - n} (1 - F)^n \tag{3.2}$$

with its Z-Transform

$$Q(z) = (1 - (1 - F)(1 - z))^{\frac{\langle n \rangle}{1 - F}} \tag{3.3}$$

Typically, w for SSDs is 10 times smaller than w for FPCs.

For SSDs, we assume there is no further charge multiplication within the detector itself, so that the charge collected is turned into a voltage, thence into channels. This is handled by the substitution

$$R_a(s) = z = \exp(-\mu s) \tag{3.4}$$

in the Z Transform $Q(z)$ of the binomial distribution to get

$$\begin{aligned} \Phi_a(s, E_\gamma) &= e^{\frac{s^2 \sigma_{chan}^2}{2}} R_a(s) Q(R_a(s)) \\ &= e^{\frac{s^2 \sigma_{chan}^2}{2}} R_a(s) (1 - (1 - F)(1 - R_a(s)))^{\frac{\langle n \rangle}{1 - F}} \end{aligned} \tag{3.5}$$

3.2 HyperJM: modified Hypermet tails and shelves

Hypermet is a set of 5 functions used by the high energy physics community for empirically fitting their detector response functions. PTB has changed one of the Hypermet functions. These are all based on Gaussians. Here we would like to base our set of "Hypermet-like" functions on the extende JM model above. We will tentatively call this set HyperJM.

Since FPC spectra also contain tails and shelves, and since in the last section we saw that SSD models are just a special case of FPC models, we will continue to use the extended JM model as a basis for the following discussion. Let us suppose that electrons are lost via diffusion, and obey some kind of distribution $P_e(E_e)$ over the energy range $E_e$from 0 to the energy of the incoming monoenergetic line E_γ. Then the mgf of the tail/shelf is

$$\Phi(s) = e^{\frac{s^2\sigma_A^2}{2}} R_a(s) \int_{E_0}^{E_\gamma} (1-(1-F)(1-R_a(s)))^{\frac{E_\gamma - E_e}{w(1-F)}} P_e\left(E_e\right) dE_e \tag{3.6}$$

where $\mathrm{R}_a\left(s\right)$ is the SES.

In particular, let us suppose that this distribution is a "tail", viz.

$$P_e\left(E_e\right) = \frac{e^{-t_l \frac{E_e}{w}} t_l}{w} \tag{3.7}$$

with the tail parameter t_l. Then the result of the integration is

$$\begin{aligned} \Phi_{tail}\left(s, t_l\right) &= e^{\frac{s\sigma_A^2}{2}} R_a(s) \int_{E_0}^{E_\gamma} (1-(1-F)(1-R_a(s)))^{\frac{E_\gamma - E_e}{w(1-f)}} P_e\left(E_e\right) dE_e \\ &= \frac{e^{\frac{s\sigma_A^2}{2}} R_a(s) \left(a^{\frac{E_0}{pw}} e^{\frac{E_\gamma t_l}{w}} - e^{\frac{E_0 t_l}{w}} a^{\frac{E_\gamma}{pw}}\right) p t_l}{e^{\frac{E_0}{pw}} \left(e^{\frac{E_\gamma t_l}{w}} - e^{\frac{E_0 t_l}{w}}\right) \left(-\ln(a) + p t_l\right)} \end{aligned} \tag{3.8}$$

where

$$\begin{aligned} a &\equiv 1-(1-F)(1-R_a(s)) \\ &= R_a(s)\left(1-F\right) + F \end{aligned} \tag{3.9}$$

and

$$p = 1 - F \tag{3.10}$$

This holds for both descending tails ($t_l \succ 0$)and for ascending tails ($t_l \prec 0$). The limit $t_l = 0$ corresponds to a flat shelf:

$$\Phi_{tail}\left(s,0\right)=\frac{e^{\frac{s\sigma_A^2}{2}}R_a(s)\left(a^{\frac{E_0}{pw}}-a^{\frac{E_\gamma}{pw}}\right)pw}{e^{\frac{E_0}{pw}}\left(E_\gamma-E_0\right)\left(-\ln\left(a\right)\right)} \tag{3.11}$$

These all have the limit

$$\Phi_{tail}(0,t_l)=1 \tag{3.12}$$

For a single incoming line, in general we need response functions for a line, a flat shelf, an ascending and descending tail, as for instance in the following combination:

$$\Phi_{line}(s)=\frac{\Phi_a(s)+\beta_{tail}\Phi_{tail}\left(s,t_{l+}\right)+\gamma_{tail}\Phi_{tail}\left(s,0\right)+\delta_{shelf}\Phi_{tail}\left(s,t_{l-}\right)}{1+\beta_{tail}+\gamma_{tail}+\delta_{shelf}} \tag{3.13}$$

The fifth Hypermet-like function is that for an escape peak, which is essentially the same combination as above, except with a shifted peak.

The coefficients $\beta_{tail},\gamma_{tail}$ and δ_{shelf} are fractions of the main peak and are independent. They are not prescribed functions of energy in Version 7.0. The PTB team seems to have a theory for these coefficients which fits a wide range of energies.

Chapter 4
CONTINUUM MODELS

4.1 Continuum X-ray emission - Kramer model

A number of different X-ray sources are being used for HXDS calibration: synchrotron radiation from BESSY, radioactive sources (X-kits), Penning plasma source, rotating anode sources and electron impact (Manson anode) sources. In the last two sources mentioned, X-rays are generated by deceleration of charged particles, producing continuum radiation, or by electron-transition from a filled electron state into a vacancy in a lower atomic shell, producing characteristic X-ray lines. The vacancy is caused by charged-particle ionization or photo-ionization(fluorescence). Both line and continuum radiation are present in the measured spectra, and it becomes important to use a semi-empirical model for the pure and detected continuum as well to extract the line count and continuum contribution.

Kramers[31], using a semi-classical treatment, derived the following bremstrahllung spectrum for incoming electron energy E_e :

$$N_c\left(E_\gamma\right) dE_\gamma = kZ\frac{\left(E_e - E_\gamma\right) dE_\gamma}{E_\gamma} \tag{4.1}$$

with

$$k = \frac{8\pi e^2}{3\sqrt{3}4\pi\varepsilon_0 hc^5 ml} \tag{4.2}$$

Absorption effects of X-rays within the source are not included in Kramers' model.

The "quantum efficiency" $\eta\left(E_\gamma\right)$ of recording a photon of energy E_γ depends on the transmission through a detector window of thickness t_{window} and the probability of absorption inside the proportional counter itself, of thickness t_{fpc}:

$$\eta\left(E_\gamma\right) = \zeta\left(E_\gamma\right)\exp\left[-\mu_{window}\left(E_\gamma\right)t_{window}\right]\left\{1-\exp\left[-\mu_{\det ector}\left(E_\gamma\right)t_{\det ector}\right]\right. \tag{4.3}$$

assuming normal incidence of the X-rays, plane-parallel surfaces for the detector, window and filter, and negligible edge effects[40]. The 1.5" by 5" proportional counter detector window is made of $1\,\mu m$ polyimide with a 200 $\mathring{A}$ aluminum coating, reinforced

by a periodic two-dimensional grid of 2-mm pitch, 100-μm diameter Au-coated W wires. The values of the absorption coefficents are tabulated in the Henke Tables[22], which are available on-line.

We weight the mgf of the single line spectral response over

$$exp\left[-\mu_{source}\left(E_\gamma\right)t_{source}\right]\eta\left(E_\gamma\right)N_c\left(E_\gamma\right)$$

to get the mgf of the continuum radiation. For the JMKramer models we have to evaluate numerically the complex mgf:

$$\begin{aligned}\Phi_{Kramer}(s) &= \left[\frac{1}{1+\frac{\mu}{h}s}\right]^h \int_{E\,\min}^{E_{\max}} \left[1-(1-F)\left(1-\left[\frac{1}{1+\frac{\mu}{h}s}\right]^h\right)\right]^{\frac{E_\gamma-E_0}{w(1-f)}} \cdot \\ &\qquad \eta\left(E_\gamma\right)\ exp\left[-\mu_{source}\left(E_\gamma\right)t_{source}\right]N_c\left(E_\gamma\right)dE_\gamma \end{aligned} \tag{4.4}$$

In JMKMOD this integral is done without recalculating

$$F(E_\gamma,s)\equiv\left[1-(1-F)\left(1-\left[\frac{1}{1+\frac{\mu}{h}s}\right]^h\right)\right]^{\frac{E_\gamma-E_0}{w(1-f)}} \tag{4.5}$$

using complex arithmetic during the numerical integration, an operation which is very inefficient.

Let us assume that the integral ($E_{\min}=E_0, E_{\max}=E_e$) is divided into N_E equal segments and we apply Simpson's Rule to get

$$\begin{aligned}\Phi_{Kramer}(s) &= \left[\frac{1}{1+\frac{\mu}{h}s}\right]^h \left[1-(1-F)\left(1-\left[\frac{1}{1+\frac{\mu}{h}s}\right]^h\right)\right]^{\frac{E_{\min}-E_0}{w(1-f)}} \Delta E_\gamma \times \\ &\left\{\begin{array}{c} \frac{1}{2}\left(p_0(s)+jq_0(s)\right)\eta\left(E_0\right)\ exp\left[-\mu_{source}\left(E_0\right)t_{source}\right]N_c\left(E_0\right)+ \\ \sum_{k=1}^{N_E-1}\left(p_k(s)+jq_k(s)\right)\eta\left(E_0+k\Delta E_\gamma\right)\times \\ exp\left[-\mu_{source}\left(E_0+k\Delta E_\gamma\right)t_{source}\right]N_c\left(E_0+k\Delta E_\gamma\right)+ \\ \frac{1}{2}\left(p_{N_E}(s)+jq_{N_E}(s)\right)\eta\left(E_{\max}\right)exp\left[-\mu_{source}\left(E_{\max}\right)t_{source}\right]N_c\left(E_{\max}\right) \end{array}\right\} \end{aligned} \tag{4.6}$$

where

$$\Delta E_\gamma=\frac{E_{\max}-E_{\min}}{N_E} \tag{4.7}$$

is the integration increment, and

$$\begin{aligned} p_k(s) &= Re\left[\left[1-(1-F)\left(1-\left[\frac{1}{1+\frac{\mu}{h}s}\right]^h\right)\right]^{k\Delta\varepsilon}\right] \\ q_k(s) &= Im\left[\left[1-(1-F)\left(1-\left[\frac{1}{1+\frac{\mu}{h}s}\right]^h\right)\right]^{k\Delta\varepsilon}\right] \end{aligned} \tag{4.8}$$

$$\Delta\varepsilon \equiv \frac{\Delta E_\gamma}{w\,(1-F)} \tag{4.9}$$

This integral must be evaluated for each value of $s = j\frac{2\pi k}{N\delta q}$ that enters into the final IFFT, and we have found that this is computationally very intensive. However, there is an efficient recursive scheme of evaluating the following expression in the integrand

$$F(E_0 + k\Delta E_\gamma, s) = \left[1-(1-F)\left(1-\left[\frac{1}{1+\frac{\mu}{h}s}\right]^h\right)\right]^{k\Delta\varepsilon} = p_k\,(s) + jq_k\,(s) \tag{4.10}$$

which avoids slow complex arithmetic for the most part. We need the following values to start:

$$\begin{aligned} p_0 &= 1 \\ q_0 &= 0 \end{aligned} \tag{4.11}$$

and

$$\begin{aligned} p_1\,(s) &= Re\left[F\,(E_0+\Delta E_\gamma, s)\right] = Re\left\{\left[1-(1-F)\left(1-\left[\frac{1}{1+\frac{\mu}{h}s}\right]^h\right)\right]^{\Delta\varepsilon}\right\} \\ q_1\,(s) &= Im\left[F\,(E_0+\Delta E_\gamma, s)\right] = Im\left\{\left[1-(1-F)\left(1-\left[\frac{1}{1+\frac{\mu}{h}s}\right]^h\right)\right]^{\Delta\varepsilon}\right\} \end{aligned} \tag{4.12}$$

They are the only expressions that have to be evaluated using complex arithmetic. Since by definition

$$p_k\,(s) + jq_k\,(s) = (p_1\,(s) + jq_1\,(s))\,(p_{k-1}\,(s) + jq_{k-1}\,(s)) \tag{4.13}$$

we can write the following recursion relations:

$$\begin{aligned} p_k\,(s) &= p_1\,(s)\,p_{k-1}\,(s) - q_1\,(s)\,q_{k-1}\,(s) \\ q_k\,(s) &= q_1\,(s)\,p_{k-1}\,(s) + p_1\,(s)\,q_{k-1}\,(s) \end{aligned} \tag{4.14}$$

for $k = 1, \ldots, N_E$. Applying the recursion relations (4.14) twice, we get

$$\begin{aligned} p_k\,(s) &= \left(p_1^2\,(s) - q_1^2\,(s)\right)p_{k-2}\,(s) - 2p_1\,(s)\,q_1\,(s)\,q_{k-2}\,(s) \\ q_k\,(s) &= 2p_1\,(s)\,q_1\,(s)\,p_{k-2}\,(s) + \left(p_1^2\,(s) - q_1^2\,(s)\right)q_{k-2}\,(s) \end{aligned} \tag{4.15}$$

From Eqs.(4.14) and (4.15) we get the alternative recursive relations which apply equally to $p_k(s)$ and $q_k(s)$:

$$\begin{pmatrix} p_k(s) \\ q_k(s) \end{pmatrix} = 2p_1(s) \begin{pmatrix} p_{k-1}(s) \\ q_{k-1}(s) \end{pmatrix} - \left(p_1^2(s) + q_1^2(s)\right) \begin{pmatrix} p_{k-2}(s) \\ q_{k-2}(s) \end{pmatrix} \tag{4.16}$$

for $k \geq 2$. This happens to be a relationship used in analog circuit design[30].

Alternatively, the number of points chosen N_E can be reduced, and should be left as a user-chosable parameter, since it is possible that choosing more points may bring diminishing returns in continuum modeling, given the basic assumptions in the model itself.

The number of integration points chosen N_E is a user-chosable parameter, typically chosen to be 512.

4.2 Radiation through a circular aperture at BESSY

The photon flux $N_{aperture}(y, \alpha)$ passing through an aperture which subtends an angle 2Δ at the instantaneous position of the electron and whose surface normal is inclined at angle α with respect to the orbital plane of the electron is given by[**?**]

$$\begin{aligned} \frac{N_{aperture}(y,\alpha)}{N_{sync}} &= \frac{3\sqrt{3}y}{20\pi^3} \int_0^{\gamma \sin\Delta} \int_0^{2\pi} \left\{ \left(1+\psi^2\right)^2 K_{2/3}^2(\xi) + \left(1+\psi^2\right)\psi^2 K_{1/3}(\xi) \right\} \cdot \\ &\quad \frac{p}{\sqrt{\gamma^2 - p^2}} dp d\varphi' dy \end{aligned} \tag{4.17}$$

with

$$\xi = \frac{y}{2}\left(1+\psi^2\right)^{3/2} \tag{4.18}$$

$$y = \frac{E_x}{E_s} \tag{4.19}$$

the ratio of the X-ray energy E_x(frequency ν_x)to the critical energy E_s(critical frequency ν_s)

$$\nu_s = \frac{3}{2} \cdot \left(\frac{E}{m_e c^2}\right)^2 \cdot \frac{eB}{m_e c} \cdot \frac{1}{2\pi} \tag{4.20}$$

$$E_s = h\nu_s$$

E being the electron energy, B being the magnetic induction, and the other symbols having their conventional meanings. The variable ψ is

$$\psi = \sin\alpha\sqrt{\gamma^2 - p^2} - p\cos\alpha\sin\varphi' \tag{4.21}$$

$$N_{sync} = \frac{10\pi^2 I}{\sqrt{3}e} \cdot \frac{E}{m_e c^2} \cdot \frac{e^2}{hc} \tag{4.22}$$

for a current I. When $\alpha = 0$, the integral (4.17)reduces to a single integral which is much easier to evaluate:

$$\begin{aligned} \frac{N_{aperture}(y,\alpha)}{N_{sync}} &= \frac{3\sqrt{3}y}{20\pi^3} \cdot 4 \cdot \int_0^{\gamma \sin\Delta} \int_0^{2\pi} \left\{ \left(1+p_y^2\right)^2 K_{2/3}^2(\xi) + \left(1+p_y^2\right) p_y^2 K_{1/3}(\xi) \right\} \cdot \\ &\quad \arctan\left(\frac{\gamma^2 \sin^2\Delta - p^2}{\gamma \cos\Delta} \right) dp_y dy \end{aligned} \tag{4.23}$$

where

$$\xi = \frac{y}{2}\left(1+p_y^2\right)^{3/2} \tag{4.24}$$

Using an aperture of 2.5mm placed at a distance 16139mm from the tangent point on the cyclotron circle, we get a half-angle of $\Delta = 0.000155$ and this is the associated parameter of the "synchrotron filter" in the source/filter table.

Chapter 5
PILEUP AND DEADTIME

5.1 Pulser peak

We will approximate the pulser peak by a Gaussian located at channel q_{pulser}, whose mgf is given by

$$R_{pulser}(s) = e^{-sq_{pulser}} e^{\frac{s^2\sigma^2_{pulser}}{2}} \tag{5.1}$$

5.2 Deadtime

The pulser allows the deadtime of the counter to be measured directly. Let the true total count rate be N_{true}, , the measured count rate be N_{total}, the measured pulser count rate be $N_{pulser\ fit}$, the true pulser count rate be N_{pulser} (which is known exactly) and the counter deadtime be τ_d. Then the following relations hold for paralyzable and non-paralyzable counters

rate / counter	**paralyzable**	**non-paralyzable**
$\langle N_{pulser\ fit} \rangle$	$N_{pulser} \exp\left(-N_{true}\tau_d\right)$	$N_{pulser}\left(1+N_{true}\tau_d\right)^{-1}$
$\left\langle \Delta N^2_{pulser\ fit} \right\rangle$	$\frac{N_{pulser}}{t}\left(1+N_{true}\tau_d\right)^{-3}$	$\frac{N_{pulser\ fit}}{t}$
N_{total}	$N_{true}\exp\left(-N_{true}\tau_d\right)$	$N_{true}\left(1+N_{true}\tau_d\right)^{-1}$
deadtime τ_d	$\frac{1}{N_{true}}\ln\left(\frac{N_{pulser}}{N_{pulser\ fit}}\right) = \frac{N_{pulser\ fit}}{N_{total}N_{pulser}}\ln\left(\frac{N_{pulser}}{N_{pulser\ fit}}\right)$	$\frac{1}{N_{true}}\left(\frac{N_{pulser}}{N_{pulser\ fit}}-1\right) = \frac{N_{puls}}{N_{total}}$

$$N_{total} = N_{pulser} + N_{continuum} + \sum_{\gamma} N\gamma \tag{5.2}$$

$$N_{true} = N_{total}\frac{N_{pulser}}{N_{pulser\ fit}} \tag{5.3}$$

$$t = true\ counting\ time \tag{5.4}$$

$$\tau_d = dead\ time \tag{5.5}$$

5.3 Pileup and Inverse FFT

Let

$$\Phi_{pileup}(s) = mgf\ of\ measured\ count\ pectrum$$

$$\Phi_a(s) = mgf\ of\ true\ count\ spectrum$$

$$s = \ Laplace\ transform\ parameter\ conjugate\ to\ channel\ number$$

$$\rho = N_{true}$$

$$\tau_{pu} = residual\ characteristic\ pileup\ time$$

then the author has shown elsewhere that

$$\Phi_a\left(s\right) = \frac{\Phi_{pileup}\left(s\right) e^{\rho\tau_{pu}}}{1 + \Phi_{pileup}\left(s\right)\left(e^{\rho\tau_{pu}} - 1\right)} \tag{5.6}$$

which may be used for pileup removal, or

$$\Phi_{pileup}(s) = \Phi_a(s) \frac{e^{-\rho\tau_{pu}}}{1 - \left(1 - e^{-\rho\tau_{pu}}\right)\Phi_a(s)} \tag{5.7}$$

which is implemented in JMKMOD. This mgf $\Phi_{pileup}(s)$ is then inverted in the usual way:

$$\begin{aligned} r(n) &= \frac{1}{N\delta q} \sum_{k=0}^{N-1} \Phi_{pileup}\left(j\frac{2\pi k}{N\delta q}\right) e^{j\frac{2\pi kn}{N}}, \qquad n = 0, 1, \ldots, N-1 \\ &= 0 \qquad\qquad\qquad \text{otherwise} \end{aligned} \tag{5.8}$$

Chapter 6
INVERSION OF MOMENT GENERATING FUNCTIONS

6.1 Inversion of the Moment Generating Function for the Count spectrum

We carry out a program to invert the mgf, which is the Laplace Transform of the pdf. The mgf goes to 0 as $s \to \infty$ in all parts of the complex plane s, and in particular along the imaginary axis. Along the imaginary s axis, the mgf is just the Fourier Transform of the response function pdf. Therefore we can invert the mgf along the imaginary axis[4], by taking the inverse-transform of the mgf. Let the count in the nth channel be $\phi(n)$, and let this be the sampled impulse response of the true analog response $\phi_a(q)$. That is,

$$\phi(n) = \phi_a(n\delta q) \tag{6.1}$$

where $\delta q = 1$ is the channel 'sampling period'(in channel units, of course). We keep δq in the formulae explicitly for clarity, then set it to unity in the end.

The digital Fourier transform Φ is related to the analog Fourier transform Φ_a by[35]

$$\Phi(e^{j\theta}) = \frac{1}{\delta q} \sum_{k=-\infty}^{\infty} \Phi_a \left(j\frac{\theta}{\delta q} + j\frac{2\pi}{\delta q}\, k \right) \tag{6.2}$$

where θ ranges from 0 to 2π. By enforcing

$$\Phi_a\left(j\Omega\right) = 0, \qquad |\Omega| \geq \frac{\pi}{\delta q} \tag{6.3}$$

all the higher frequencies (which are usually very small anyway) are eliminated, and there would be no aliasing, so that

$$\Phi(e^{j\theta}) = \frac{1}{\delta q}\Phi_a\left(j\frac{\theta}{\delta q}\right), \qquad |\theta| \leq \pi \tag{6.4}$$

Let this Fourier response be sampled at N equal intervals to give a *finite* Fourier Transform

$$\tilde{\Phi}\left(k\right) = \Phi(z)|_{z=e^{j\frac{2\pi k}{N}}} = \frac{1}{\delta q}\Phi_a\left(j\frac{2\pi k}{N\delta q}\right) = \sum_{n=0}^{N-1} \phi\left(n\right) e^{-j\frac{2\pi kn}{N}} \tag{6.5}$$

This can be inverted (via inverse FFT) to give the response function pdf in counts per channel:

$$\begin{aligned}\phi(n) &= \frac{1}{N}\sum_{k=0}^{N-1}\widetilde{\Phi}(k)\,e^{j\frac{2\pi kn}{N}}, \qquad n=0,1,\ldots,N-1 \\ &= \frac{1}{N\delta q}\sum_{k=0}^{N-1}\Phi_a\left(j\frac{2\pi k}{N\delta q}\right)e^{j\frac{2\pi kn}{N}}, \qquad n=0,1,\ldots,N-1 \\ &= 0 \qquad\qquad \text{otherwise} \end{aligned} \tag{6.6}$$

For FPCS, we replace δq with

$$\delta q = 1 \tag{6.7}$$

and $< n_{se} >$ with Eq. (2.1) above

$$< n_{se} >= \frac{E_\gamma - E_0}{\langle w\rangle}$$

noting that $\langle w\rangle$ is the *mean ionization energy* per ion-electron pair created by the incoming X-ray with energy E_γ [1], and E_0 is the energy offset[26], to get

$$\phi(n) = \frac{1}{N}\sum_{k=0}^{N-1} R_a\left(j\frac{2\pi k}{N\delta q}\right)\left[1-(1-F)\left(1-R_a\left(j\frac{2\pi k}{N\delta q}\right)\right)\right]^{\frac{E_\gamma - E_0}{\langle w\rangle(1-F)}} e^{j\frac{2\pi kn}{N}} \tag{6.8}$$

for $n = 0, 1, \ldots, N-1$,and 0 otherwise. Here R_a is the SES.

This is the final form of our pulse height response function for a sharp incoming line of energy $E\gamma$, which we shall call the *Extended JM (EJM) pdf.* It is characterized by 6 parameters: E_γ, $\mathrm{E_0}$, w, F, μ and h. That we are able to characterize it so succinctly is due to the availability of a closed form for the analog mgf Φ_a.

The number of computational channels N does not have to be and indeed must not be the same as the number of physical channels N_c. We often choose

$$N = 2N_{chan} \tag{6.9}$$

so that there is enough of a 'buffer' between the last computational channel and the first, bearing in mind that the Discrete Fourier Series wraps around after a 'period' of N. If a continuum background is present up to an energy $E_{\max}$, the number of channels should be chosen so that $N \succ \frac{\mu E_{\max}}{w}$.

Chapter 7
RELATIVE AND ABSOLUTE CALIBRATION GOALS

7.1 AXAF calibration goals

The Advanced X-ray Astrophysical Facility (AXAF) is NASA's third Great Space Observatory, and is scheduled for a shuttle launch in the late 1990's[69]. Of central interest to the calibration of the four Wolter Type-I mirror pairs, known as the High Resolution Mirror Assembly (HRMA), is the prelaunch measurement of the intensity distribution of the X-ray beam spot or point response function of the mirror in the focal plane.

The measurements, conducted in 1996 and 1997 at the X-ray Calibration Facility (XRCF) of the NASA Marshall Space Flight Center (MSFC), consist of sampling the intensity distribution of the beam in the vicinity of the beam center and at or near the focal plane. The photon counts are recorded by X-ray detectors: 7 flow(gas) proportional counters (FPCs) and 2 Ge solid state detectors (SSDs) located at different stations along the facility optical axis. Some of the criteria to be met for detection calibration are:

1. Detector livetimes to within 0.25%.

2. Detector absolute QEs to 1%.

3. Detector relative QEs to less than 0.17%.

4. X-ray line counts to 0.2% by fitting the detected count spectrum to models of the FPCs and SSDs.

In these criteria, the *absolute quantum efficiency* (q.e.) η ($0 \leq \eta \leq 1$) of the detector is defined as the probability that a single photon incident on the detector generates an ion-electron pair that contributes to the detector count[44]. Not all ion-electron pairs produced are collected because of electron diffusion losses or recombination. The absolute q.e. of recording a photon of energy E_γ depends on the transmission through a detector window of thickness x_{window} and the probability of absorption inside the proportional counter itself, of thickness x_{fpc}, and may be written as

$$\eta\left(E_\gamma\right) = \zeta\left(E_\gamma\right)\exp\left[-\left(\frac{\mu_{window}\left(E_\gamma\right)}{\rho_{window}}\right)\rho_{window}x_{window}\right] \left\{1-\exp\left[-\left(\frac{\mu_{fpc}\left(E_\gamma\right)}{\rho_{fpc}}\right)\rho_{fpc}x_{fpc}\right]\right\} \quad (7.1)$$

assuming normal incidence of the X-rays, plane-parallel surfaces for the detector and window. Here $\zeta\left(E_\gamma\right)$ is a catch-all term for the fraction of electron-ion pairs that do not contribute to useful photocurrent, and also for any edge effects not accounted for by the simple geometry assumed in the transmission and absorption factors[40].. Absolute q.e. calibration reduces to the precise determination of this correction term $\zeta\left(E_\gamma\right)$. This quantity, $\zeta\left(E_\gamma\right)$, must be close to unity across a broad range of E_γ, for if it were not, the continuum would have been poorly fit.

The absolute q.e. is only part of the detector pulse height response function, which is defined as the product

$$\varphi\left(q,E_\gamma\right) = \eta\left(E_\gamma\right)\phi\left(q,E_\gamma\right) \quad (7.2)$$

Here $\phi\left(q,E_\gamma\right)$,the probability distribution function (pdf) for the fluctuations in pulse height channel number q as a result of an incoming photon of energy E_γ. Sometimes this pdf itself is called the detector response function.

The predicted pulse height $C_p\left(q\right)$ at channel q is the convolution of the detector response function $\varphi\left(q,E_\gamma\right)$with a source spectrum $S(E_\gamma)$ and any intervening filters $f(E_\gamma)$ between the source and detector

$$C_p(q) = \int_0^\infty \varphi\left(q,E_\gamma\right) f(E_\gamma)S(E_\gamma)dE_\gamma \quad (7.3)$$

From the point of view of the calibration of the X-ray mirror optics, the HRMA effective area (defined as reflection coefficient $\times$ mirror area) may be considered just a factor in the $f(E_\gamma)$. The $S(E_\gamma)$ consists principally of characteristic K and L emission lines from the anode material superimposed on a bremsstrahlung continuum of an electron impact X-ray source[72]. Properly selected filters, which form part of the $f(E_\gamma)$, suppress the bremsstrahlung continuum and low energy photons. As a rule which is not strictly enforced, filters are made of the same material as the target source since the characteristic emission energy of an element is slightly below the absorption edge energy. A pure element is "transparent" to its own emission lines. Filter thicknesses are chosen to attenuate the X-ray beam by a factor of 2 - 5 at the energy of interest.

For spectral calibration, the principal task is to determine the *strengths* of the emission lines from the source over and above the continuum and their overlap with other lines in the pulse height count spectrum. It cannot be more fully emphasized, therefore, that all the other parameters, such as detector characteristics, source characteristics, etc., while important in their own supporting roles and which form the

backbone of the detector modeling described in this paper, are in the end *subservient* to the the determination of the strengths of the emission lines.

The *relative q.e.* is the ratio of the q.e.s of two detectors at some energy, obtained by simultaneously measuring the flux from the same source. For this measurement, the source may be time varying and its absolute flux needs not be known to great precision. This imprecision cannot be tolerated for absolute q.e. measurements, which demand a flux from a source known to great precision, such as that from the BESSY synchrotron.

Preliminary calibration of the HXDS detectors and software testing have been conducted at the Smithsonian Astrophysical Observatory (SAO) X-ray Pipe facility, at the XRCF at MSFC before December 1996, and by SAO at the PTB (BESSY synchrotron in Berlin). These are all dedicated to the calibration of the detectors themselves, without the presence of the HRMA. The HRMA was introduced after December 1996. JMKMOD can be used for detector calibration, HRMA effective area determination and the absolute calibration of the

Chapter 8
CONCLUSIONS

We have shown that the distributions, taken as assumptions by various authors for the secondary and the avalanche electrons in a proportional counter, viz. the binomial and negative binomial distributions, respectively, can be deduced from birth-and-death processes commonly used to study cosmic rays and other stochastic processes. One immediate insight is to arrive at the actual physical signficance of the Fano factor f and the Polya h factor. The former is related to the ratio of the recombination to the ionization rates. The latter is the number of ratio of the actual gas gain to the gain calculated from Townsend's theory of discharges.

We conclude that the present fpc model as implemented by XSPEC is an excellent tool for fitting fpc data. Specifically, what we have done is to have:

1. Summed the Laplace Transform of the Jahoda-McCammon-Alkhazov (JM) distribution (or moment generating function, mgf) in closed form. This Laplace Transform cannot be inverted analytically to give the JM distribution in channel space.

2. Checked that a special case of JM inverted analytically exactly to the well-known Prescott function by specialization of two parameters, Eq.

3. Numerically inverted the mgf. Straightforward inversion of the Laplace Transform being unstable, we exploited the fact that that we are dealing with a probability distribution function that vanishes at ∞ to invert the mgf along the imaginary axis, which is just a Fourier Transform. We successfully ran a series of tests to check that the numerically inversion of a special case of the JM distribution did indeed to the Prescott function to within the desired computational precision.

4. Applied the algorithm to fit line(s) and continuum (Kramers model smeared out by the detector response function) to determine a single set of parameters of the detector response function which best fitted the data for line and continuum simultaneously.

The inversion of the moment generating function by the fast Fourier Transform method is the only convenient way of getting a distribution based on the theories of

Alkhazov and Jahoda-McCammon to fit the data. Jahoda and McCammon in their paper[26] used an infinite series for their distribution, each term of which contains a Gaussian multiplying into an Erlangian (approximate Polya) distribution. They had to truncate their series at some point, and that point is actually one of the parameters we are trying to fit: the maximum number of secondary electrons created, Eq. The computer program they have for carrying out their calculations is directed towards generating the detector response function when the parameters are already known. The model we have developed in this paper, which is in a sense an extension to their model and Alkhazov's, reduces their entire computer program into an analytical expression suitable for calibration.

Further developments in spectral fitting will be pursued in the following areas:

1. Enhancing the models. Some areas of concern, like shelves and tails, need a comprehensive theory.

2. Improving the continuum bremsstrahlung model by including effects of X-ray absorption in the target, such as the Pella model.

3. Improving the pileup model to account for 'non-ideal' effects.

4. Incorporating a HRMA model which can be fitted.

Part IV

JMKMOD Software Guide

Chapter 9
ARCHITECTURE

9.1 Software architecture

The overall architecture of JMKMOD is given in Figure 1. A more detailed modular layout is shown in Figure 2. The data flow is shown in Figure 3.

Note that we are addressing the first-level processing, when all the parameters are determined as a function of energy as a result of the measured response to discrete lines. In second-level processing, it may be envisaged that the energy dependence of these parameters may be used to improve the continuum fitting and the line fitting as well.

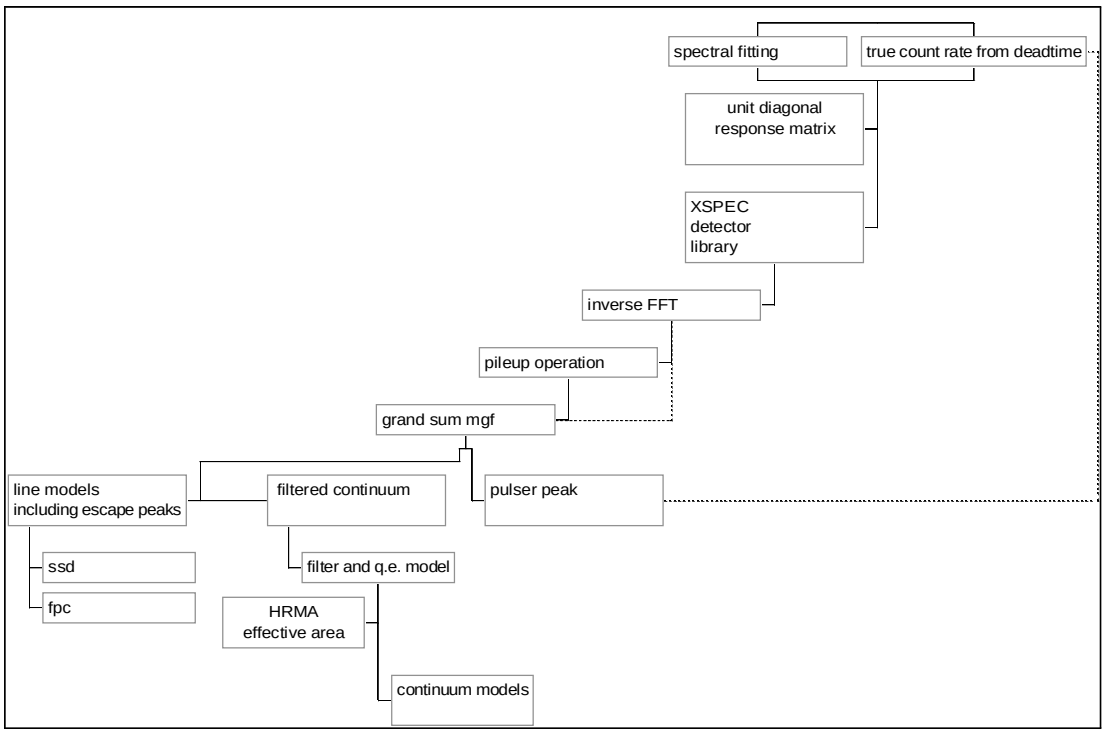

Figure 1 Overall architecture of JMKMOD.

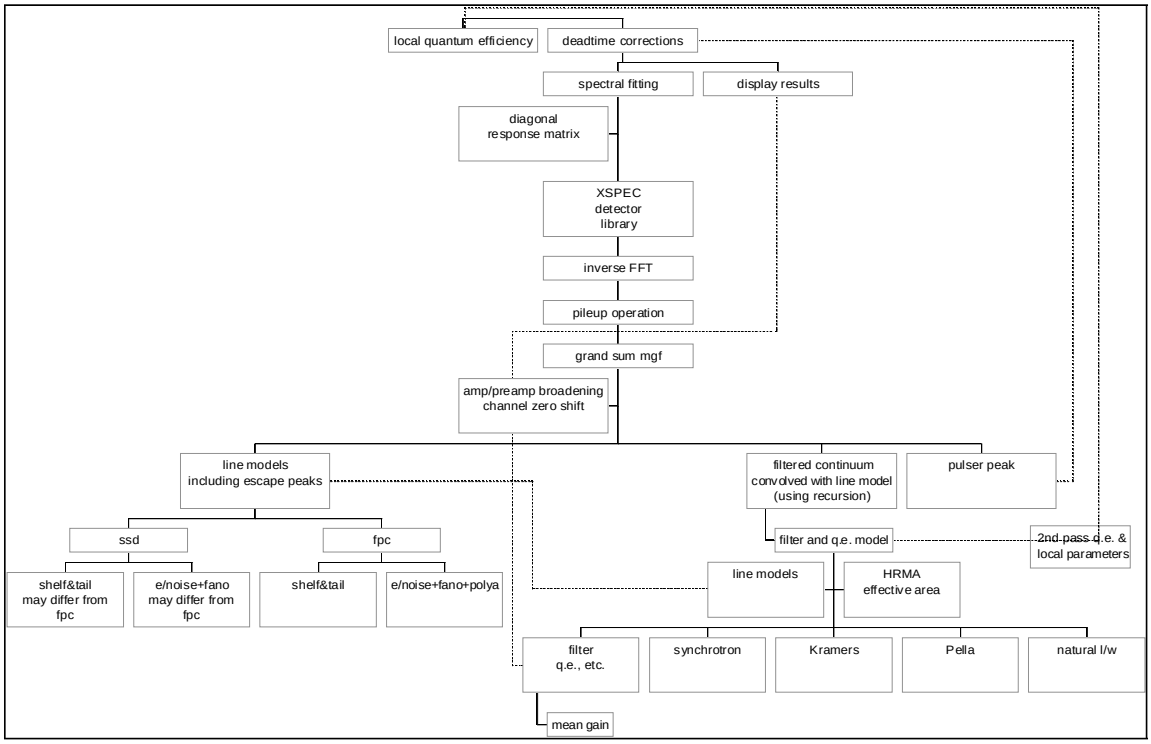

Figure 2 Detailed layout of software code showing major modules.

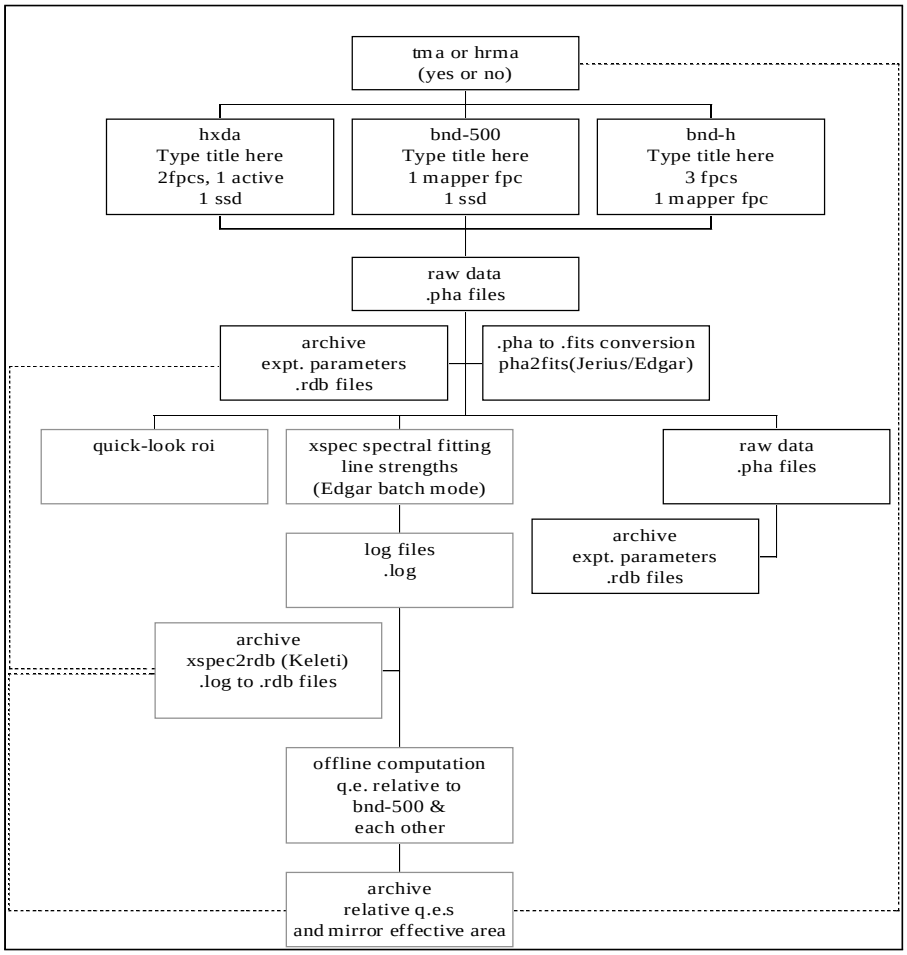

Figure 3 Data flow

Chapter 10
CODE

10.0.1 params.for

Remark 1 *An include file for Fortran commons*

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Programmer: Eugene Tsiang c
c AXAF Mission Support Team c
c Date: 8/26/96 c
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
c Module name: params.for, an include file
c include file "params.for" defines all the common blocks
c10/9/96 added synchrotron block
c 12/30/96 added ice
c 12/31/96 added absolute switch and effective area parameters
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c added 10/9/96 for synchrotron radiation
common/bessyfac/yenergy, bessyEcrit, bessyfluxnorm,
1 bessyEnorm, bessyBnorm, bessyInorm, bessyEnergy, bessyBfield,
2 bessyI, boost, sind, boostsind, boostsind2,
3 boostcosd, tand, enercrit, fluxconst, boostnorm,
4 bfield, current, incline
double precision bessyEcrit, bessyfluxnorm,
1 bessyEnorm, bessyBnorm, bessyInorm, bessyEnergy, bessyBfield,
2 bessyI, boost, sind, boostsind, boostsind2,
3 boostcosd, tand, enercrit, fluxconst, boostnorm,
4 bfield, current, incline
double precision yenergy(512)
c above added 10/9/96 for synchrotron radiation, abswitch added 12/31/96
c added 12/31/96 effective area option
common/hrma/phi, theta, shell
double precision phi, theta
integer shell, singlelimit
```

```
c above added 12/31/96 for hrma effective area
integer integral, linelist, elemlist, srclist, maxhenke
c the array sizes for integration and number of lines,
c number of filter elements, number of source filters, initialized by
c blockdata statement
parameter (integral = 1024, elemlist = 50, singlelimit = 42,
1 srclist = 16, linelist = 32, maxhenke = 1024)
c singlelimit was 42, 12/31/96
double precision fano, polyaH, eOff,
1 ionEn, gain, dgain, shift, sigma
common/modvar/ fano, polyaH, eOff,
1 ionEn, gain, dgain, shift, sigma
double precision eChar, eWidth, eCont, eMax, eMin, contnorm,
1 pulsepos, pulsesigma, pulsenorm,
2 linenergy(linelist), linenorm(linelist)
common/spread/eChar, eWidth, eCont, eMax, eMin, contnorm,
1 pulsepos, pulsesigma, pulsenorm, linenergy, linenorm
c WARNING! Make sure linenorm has enough room for all line AND
c continuum components!!!! 1/2/97
double precision temper, pressure, argrat
common/detfpc/temper, pressure, argrat
integer shelfswitch, pulser, continuum, sestype, abswitch
double precision t1norm, t2norm, shelfnorm, t1par, t2par
common/shelf/t1norm, t2norm, shelfnorm, t1par, t2par,
1shelfswitch, abswitch, sestype, pulser, continuum
integer pileup
double precision pilepar
common/pur/ pilepar, pileup
character*2 elemName(elemlist)
double precision elemWate(elemlist), elemDens(elemlist),
1 elemThick(elemlist)
integer parylene,polyimide,polypro,kramers,synchro,effarea,
1 ssdgerm,fpcp10,fpcmeth,gnplace,unplace,geplace,
2 oplace,hplace,nplace,arplace,xeplace,cplace,alplace,
3 pella, ice
common/element/
1 elemWate, elemDens, elemThick, elemName,
2 parylene,polyimide,polypro,kramers,synchro,
3 effarea,ssdgerm,fpcp10,fpcmeth,gnplace,unplace,geplace,
4 oplace, hplace, nplace, arplace, xeplace, cplace, alplace,
5 pella, ice
c initialized by blockdata statement
```

```
integer srcstack(srclist)
double precision srcthick(srclist), srcdamp(integral,srclist)
common/filstack/srcstack, srcthick, srcdamp
double precision mypi, pebble, half, one, two, torr, stddens,
1 abzero
common/diamond/mypi, pebble, half, one, two, torr, stddens,
1 abzero
c initialized by blockdata statement
double precision dEmax, eArray(integral), evArr(integral),
1 alldamp(integral)
common/enerMax/dEmax, eArray, evArr, alldamp
integer nInt, nInt1, nChan, tChan, tChan1, numfilter, numcomp
double precision deltaE
double complex piDelt
common/enint/piDelt, deltaE,
1nInt, nInt1, nChan, tChan, tChan1, numfilter, numcomp
common/flagship/eMaxOld, eMinOld, dEmaxOld, enerflag
double precision eMaxOld, eMinOld, dEmaxOld
integer enerflag
```

10.0.2 SUBROUTINE beschb(x,gam1,gam2,gampl,gammi)

Remark 2 *A Bessel function evaluation routine, from Numerical Recipes.*

```
INTEGER NUSE1,NUSE2
DOUBLE PRECISION gam1,gam2,gammi,gampl,x
PARAMETER (NUSE1=5,NUSE2=5)
CU USES chebev
REAL xx,c1(7),c2(8),chebev
SAVE c1,c2
DATA c1/-1.142022680371172d0,6.516511267076d-3,3.08709017308d-4,
*-3.470626964d-6,6.943764d-9,3.6780d-11,-1.36d-13/
DATA c2/1.843740587300906d0,-.076852840844786d0,1.271927136655d-3,
*-4.971736704d-6,-3.3126120d-8,2.42310d-10,-1.70d-13,-1.d-15/
xx=8.d0*x*x-1.d0
gam1=chebev(-1.,1.,c1,NUSE1,xx)
gam2=chebev(-1.,1.,c2,NUSE2,xx)
gampl=gam2-x*gam1
gammi=gam2+x*gam1
return
END
C (C) Copr. 1986-92 Numerical Recipes Software %5'KNp='.
```

10.0.3 subroutine bessyflux(damping)

Remark 3 *BESSY number flux calculation*

```
include 'params.for'
double precision damping(nInt)
real*4 yvalue
common/besspec/yvalue
real*4 bessyhole
external bessyhole
real*4 pihalf, evalue, syncons, result
integer i
pihalf = 1.570796327
syncons = 0.03351677775621519
* print *, 'In bessflux, i, yvalue, evalue, normresult, result = '
do 500 i = 1, nInt
evalue = earray(i)
yvalue = evalue/bessyEcrit
yenergy(i) = yvalue
* print *, i, yvalue, evalue
call qsimp(bessyhole,0.,pihalf,result)
damping(i) = bessyfluxnorm*evalue*result
* print *, i, yvalue, evalue, syncons*yvalue*result, damping(i)
500 continue
return
end
```

*10.0.4 real*4 function bessyhole(alpha)*

Remark 4 *BESSY synchrotron radiation integrand*

```
include 'params.for'
real*4 alpha
real*4 yvalue
common/besspec/yvalue
real*4 shalf, third, sone, onehalf, pi
parameter (shalf = 0.5, third = 1./3., sone = 1.)
parameter (onehalf = 1.5, pi = 3.141592654)
real*4 gamcosangle, angpart, dummy1, xsi, ri, rip, py, py2
real*4 koneth, konethp, ktwoth
py = boostsind*sin(alpha)
py2 = py*py
gamcosangle = sqrt(abs(boostsind2 - py2))
angpart = atan(gamcosangle/boostcosd)
* print *, 'In bessyhole, gamsind, gamcosd, angpart = '
```

```
* print *, gamsind, gamcosd, angpart
dummy1 = sone + py2
xsi = yvalue*shalf*dummy1**onehalf
call bessik(xsi, third, ri, koneth, rip, konethp)
* print *, 'xsi, ri, rip, koneth, konethp = '
* print *, xsi, ri, rip, koneth, konethp
ktwoth = -konethp - third*koneth/xsi
bessyhole = gamcosangle*angpart*dummy1*(dummy1*ktwoth*ktwoth +
1 py2*koneth)
* print *, 'yenergy, py2, dummy1, ktwoth, integrand = '
* print *, yenergy, py2, dummy1, ktwoth, integrand
return
end
```

10.0.5 subroutine bessik(x,xnu,ri,rk,rip,rkp)

Remark 5 *Bessel function calculation, from Numerical Recipes*

```
INTEGER MAXIT
REAL ri,rip,rk,rkp,x,xnu,XMIN
DOUBLE PRECISION EPS,FPMIN,PI
PARAMETER (EPS=1.e-10,FPMIN=1.e-30,MAXIT=10000,XMIN=2.,
*PI=3.141592653589793d0)
CU USES beschb
INTEGER i,l,nl
DOUBLE PRECISION a,a1,b,c,d,del,del1,delh,dels,e,f,fact,fact2,ff,
*gam1,gam2,gammi,gampl,h,p,pimu,q,q1,q2,qnew,ril,ril1,rimu,rip1,
*ripl,ritemp,rk1,rkmu,rkmup,rktemp,s,sum,sum1,x2,xi,xi2,xmu,xmu2
if(x.le.0..or.xnu.lt.0.) pause 'bad arguments in bessik'
nl=int(xnu+.5d0)
xmu=xnu-nl
xmu2=xmu*xmu
xi=1.d0/x
xi2=2.d0*xi
h=xnu*xi
if(h.lt.FPMIN)h=FPMIN
b=xi2*xnu
d=0.d0
c=h
do 11 i=1,MAXIT
b=b+xi2
d=1.d0/(b+d)
c=b+1.d0/c
del=c*d
```

```
h=del*h
if(abs(del-1.d0).lt.EPS)goto 1
11 continue
pause 'x too large in bessik; try asymptotic expansion'
1 continue
ril=FPMIN
ripl=h*ril
ril1=ril
rip1=ripl
fact=xnu*xi
do 12 l=nl,1,-1
ritemp=fact*ril+ripl
fact=fact-xi
ripl=fact*ritemp+ril
ril=ritemp
12 continue
f=ripl/ril
if(x.lt.XMIN) then
x2=.5d0*x
pimu=PI*xmu
if(abs(pimu).lt.EPS)then
fact=1.d0
else
fact=pimu/sin(pimu)
endif
d=-log(x2)
e=xmu*d
if(abs(e).lt.EPS)then
fact2=1.d0
else
fact2=sinh(e)/e
endif
call beschb(xmu,gam1,gam2,gampl,gammi)
ff=fact*(gam1*cosh(e)+gam2*fact2*d)
sum=ff
e=exp(e)
p=0.5d0*e/gampl
q=0.5d0/(e*gammi)
c=1.d0
d=x2*x2
sum1=p
do 13 i=1,MAXIT
```

```
ff=(i*ff+p+q)/(i*i-xmu2)
c=c*d/i
p=p/(i-xmu)
q=q/(i+xmu)
del=c*ff
sum=sum+del
del1=c*(p-i*ff)
sum1=sum1+del1
if(abs(del).lt.abs(sum)*EPS)goto 2
13 continue
pause 'bessk series failed to converge'
2 continue
rkmu=sum
rk1=sum1*xi2
else
b=2.d0*(1.d0+x)
d=1.d0/b
delh=d
h=delh
q1=0.d0
q2=1.d0
a1=.25d0-xmu2
c=a1
q=c
a=-a1
s=1.d0+q*delh
do 14 i=2,MAXIT
a=a-2*(i-1)
c=-a*c/i
qnew=(q1-b*q2)/a
q1=q2
q2=qnew
q=q+c*qnew
b=b+2.d0
d=1.d0/(b+a*d)
delh=(b*d-1.d0)*delh
h=h+delh
dels=q*delh
s=s+dels
if(abs(dels/s).lt.EPS)goto 3
14 continue
pause 'bessik: failure to converge in cf2'
```

```
3 continue
h=a1*h
rkmu=sqrt(PI/(2.d0*x))*exp(-x)/s
rk1=rkmu*(xmu+x+.5d0-h)*xi
endif
rkmup=xmu*xi*rkmu-rk1
rimu=xi/(f*rkmu-rkmup)
ri=(rimu*ril1)/ril
rip=(rimu*rip1)/ril
do 15 i=1,nl
rktemp=(xmu+i)*xi2*rk1+rkmu
rkmu=rk1
rk1=rktemp
15 continue
rk=rkmu
rkp=xnu*xi*rkmu-rk1
return
END
C (C) Copr. 1986-92 Numerical Recipes Software %5'KNp='.
```

10.0.6 FUNCTION chebev(a,b,c,m,x)

Remark 6 *Chebechev routine in Bessel Function computation, from Numerical Recipes.*

```
INTEGER m
REAL chebev,a,b,x,c(m)
INTEGER j
REAL d,dd,sv,y,y2
if ((x-a)*(x-b).gt.0.) pause 'x not in range in chebev'
d=0.
dd=0.
y=(2.*x-a-b)/(b-a)
y2=2.*y
do 11 j=m,2,-1
sv=d
d=y2*d-dd+c(j)
dd=sv
11 continue
chebev=y*d-dd+0.5*c(1)
return
END
C (C) Copr. 1986-92 Numerical Recipes Software %5'KNp='.
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Programmer: Eugene Y. Tsiang c
```

```
c AXAF Mission Support Team c
c Date: 8/26/96 c
c Change dates: 8/27/96
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
c Module name: ctmmgf.f c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

10.0.7 double complex function ctmmgf(slap,mgf)

Remark 7 *Continuum moment generating function calculation, accelerated method*

```
c moment generating function for continuum smoothed by mgf, at slap
include 'params.for'
* double complex slap, sglmgf, sglshelf, mgf, dMgf
double complex slap, sglmgf, mgf, dMgf
external mgf
double precision reMgf(integral), imMgf(integral), tmpor1, tmpor2
double precision oldE, twoMgf, sqrMgf
integer k
c the array 'alldamp(nInt)' is already in place due to damper
c called in jmkmod
c start the recursion, do for incremental dEmax
oldE = eChar
eChar = dEmax
* if (shelfswitch.eq.0) then
* dMgf = sglmgf(slap, mgf)
* else
* dMgf = sglshelf(slap,mgf)
c wrong!
c shelves and tails demand their own accelerators, 8/27/96, EYT
* endif
dMgf = sglmgf(slap, mgf)
c convolve only bare mgfs with continuum
c this is the germ mgf
eChar = oldE
c reset eMax to old eMax
** print *, 'dMgf in ctmmgf =', dMgf
reMgf(1) = one
imMgf(1) = 0.
reMgf(2) = dreal(dMgf)
imMgf(2) = dimag(dMgf)
twoMgf = two*reMgf(2)
sqrMgf = reMgf(2)*reMgf(2) + imMgf(2)*imMgf(2)
```

```
Do 300 k = 3, nInt
reMgf(k) = twoMgf*reMgf(k-1) - sqrMgf*reMgf(k-2)
imMgf(k) = twoMgf*imMgf(k-1) - sqrMgf*imMgf(k-2)
300 continue
c this generates the real and imaginary parts of mgf at different
c energies, for a fixed Laplace parameter slap
** print *, 'slap = ', slap, ', Mgf in ctmmgf = '
** do 150 k = 1, nInt
** print *, reMgf(k), ',', imMgf(k)
** 150 continue
tmpor1 = (reMgf(1)*alldamp(1) + reMgf(nInt)*alldamp(nInt))*half
tmpor2 = (imMgf(1)*alldamp(1) + imMgf(nInt)*alldamp(nInt))*half
c Simpson's rule of integration
do 400 k = 2, nInt1
tmpor1 = tmpor1 + reMgf(k)*alldamp(k)
tmpor2 = tmpor2 + imMgf(k)*alldamp(k)
400 continue
tmpor1 = tmpor1*dEmax
tmpor2 = tmpor2*dEmax
c this normalization wrt dEmax may not be needed!
** print *, 'tmpor1 in ctmmgf = ', tmpor1
** print *, 'tmpor2 in ctmmgf = ', tmpor2
c this is the convolution
ctmmgf = dcmplx(tmpor1,tmpor2)
** print *, 'last stand: ctmmgf in ctmmgf = ', ctmmgf
c this is the continuum mgf for a fixed Laplace slap
return
end
```

10.0.8 Block Data initje

Remark 8 *Initialization of source/filter tables, definition of constants*

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Programmer: Eugene Tsiang c
c AXAF Mission Support Team c
c Date: 8/26/96 c
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
c Module name: initje.f c
c Modifications:
c Added polypro, Mn 9/11/96
c 9/16/96 Changed stddens to 4.4614113d-5
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
block data initJe
include 'params.for'
c special nonelements:
c 'p1' is parylene
c 'p2' is polyimide
c 'p3' is mylar (density wrong)
c 'p4' is teflon (polytetrafluoroethylene, CF2)
c 'p5' is pn (ammonium dihydrogen phosphate, density wrong)
c 'p6' is pvc (polyvinyl chloride, density wrong)
c 'p7' is polypropylene (CH2), density 0.9
c 'ks' is the kramers continuum
c 'sy' is the synchrotron radiation spectrum
c 'gx' is the germanium quantum efficiency
c 'ax' is the fpc p10 quantum efficiency
c 'mx' is the fpc methane quantum efficiency
c 'ea' is the effective area of the mirror
c 'gn' is the gaussian
c 'px' is the pella continuum
c 'un' is the all-pass (unity) filter
data parylene/5/
data polyimide/19/
data polypro/47/
data kramers/35/
data synchro/36/
data effarea/40/
data ssdgerm/37/
data fpcp10/38/
data fpcmeth/39/
data alplace/2/
data cplace/6/
data oplace/20/
data hplace/21/
data nplace/22/
data arplace/23/
data xeplace/24/
data gnplace/41/
data unplace/45/
data geplace/32/
data pella/42/
data ice/48/
data elemname/'mo','al','zr','cu','p1',
1 'c' ,'be','b' ,'cr','fe',
```

```
2 'ni','zn','mg','ti','ag',
3 'in','v' ,'co','p2','o' ,
4 'h' ,'n' ,'ar','xe','sn',
5 'nb','w' ,'p3','p4','p5',
6 'p6','ge','si','f' ,'ks',
7 'sy','gx','ax','mx','ea',
8 'gn','px','au','ir','un',
9 'mn','p7','ic',' ',' '/
c Nominal aluminum thickness of fpc window is 2.d-6 cm or 200A
c Polyimide window is 1.065 microns thick
data elemThick/ 2.0d-4, 1.0d-4, 2.0d-4, 0.5d-4, 0.0d0 ,
1 2.5d-4, 2.0d-4, 1.0d-4, 1.0d-4, 0.6d-4,
2 0.5d-4, 2.5d-4, 1.5d-3, 4.0d-4, 1.0d-4,
3 5.0d-4, 2.0d-3, 2.5d-3, 1.065d-4, 0.0d0 ,
4 0.0d0 , 0.0d0 , 0.0d0 , 0.0d0 , 1.0d-4,
5 1.0d-4, 1.0d-4, 0.0d0 , 0.0d0 , 0.0d0 ,
6 0.0d-4, 1.0d-4, 1.0d-4, 0.0d0 , 0.0d0 ,
7 1.55d-4,1.0d-1, 5.36 , 5.36 , 0.0d0 ,
8 0.0d0 , 0.0d0 , 1.d-4 , 1.d-4 , 0.0d0 ,
9 1.0d-4, 1.d-4 , 1.d-4 , 0.0d0 , 0.0d0 /
data elemWate/ 95.94000, 26.98154, 91.22000, 63.54600, 0.00000,
1 12.01115, 9.01218, 10.81400, 51.99600, 55.84700,
2 58.69000, 65.38700, 24.31000, 47.87900,107.86800,
3 114.81800, 50.94150, 58.93300, 0.00000, 25.99940,
4 1.00797, 14.00672, 39.94800,131.29000,118.69000,
5 92.90640,183.85000, 0.00000, 0.00000, 0.00000,
6 0.00000, 72.59000, 28.08550, 18.99840, 0.00000,
7 0.00000, 0.00000, 0.00000, 0.00000, 0.00000,
8 0.00000, 0.00000,196.96650,192.22000, 0.00000,
9 54.93805, 0.00000, 0.00000, 0.00000, 0.00000/
data elemDens/ 10.20, 2.70, 6.49, 8.96, 1.10,
1 2.66, 1.82, 2.47, 7.19, 7.87,
2 8.91, 7.13, 1.74, 4.51, 10.50,
3 7.29, 6.09, 8.90, 1.44, 0.00,
4 0.00, 0.00, 0.00, 0.00, 7.30,
5 8.55, 19.30, 1.00, 1.00, 1.00,
6 1.00, 5.32, 2.33, 0.00, 0.00,
7 0.00, 0.00, 0.00, 0.00, 0.00,
8 0.00, 0.00, 19.30, 22.50, 0.00,
9 7.47, 0.95, 1.00, 0.00, 0.00/
data mypi, pebble, half, one, two, torr, stddens, abzero/
1 3.141592653589793,1.d-8,5.d-1, 1.d0, 2.d0, 760.0,
```

```
2 4.4614113d-5, 273.15/
data bessyEnorm, bessyBnorm, bessyInorm, fluxconst, boostnorm,
1enercrit/
27.8e5, 1.45e4, 1., 6.139996366423063e6, 1526.406838302636,
3 0.5866605793269655/
end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Programmer: Eugene Tsiang c
c AXAF Mission Support Team c
c Date: 8/26/96 c
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
c Module name: invert.f
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

10.0.9 subroutine invert(mgfarr, pdfarr)

Remark 9 *inversion of moment generating function given an array of mgfs*

```
c inverts a general mgf array mgfarr(tChan)
c to give a real pdf pdfarr(tChan)
c max channel 4096*4 = 16384
include 'params.for'
double precision dummy1, dummy2
integer i
double precision dumarr(16384), pdfarr(tChan)
double complex mgfarr(tChan)
* print *, 'In invert, mgfarr = ', mgfarr
* print *, 'In invert,', 'nchan = ', nchan
c convert double complex array mgfarr to single real array dumarr
do 10 i = 1, tChan
dumarr(2*i -1) = dble(mgfarr(i))
dumarr(2*i) = dimag(mgfarr(i))
10 continue
* print *, 'Before four1, dumarr = '
* do 850 i = 1, tChan
* print *, dumarr(2*i-1), dumarr(2*i)
* 850 continue
c Inverse Fourier transform
call four1(dumarr, tChan, -1)
* print *, 'After four1, dumarr = '
* do 860 i = 1, tChan
* print *, dumarr(2*i-1), dumarr(2*i)
* 860 continue
```

```
c take the absolute value, divide by tChan for correct normalization
do 40 i = 1, tChan
dummy1 = dumarr(2*i - 1)
dummy2 = dumarr(2*i)
pdfarr(i) = (dsqrt(dummy1*dummy1 + dummy2*dummy2))/tChan
40 continue
return
end
```

10.0.10 SUBROUTINE FOUR1(DATA,NN,ISIGN)

Remark 10 *Inverse Fast Fourier Transform, from Numerical Recipes*

```
DOUBLE PRECISION WR,WI,WPR,WPI,WTEMP,THETA, TEMPR, TEMPI
DOUBLE PRECISION DATA(*)
INTEGER NN,ISIGN,N,J,I,M,MMAX,ISTEP
N=2*NN
J=1
DO 11 I=1,N,2
IF(J.GT.I)THEN
TEMPR=DATA(J)
TEMPI=DATA(J+1)
DATA(J)=DATA(I)
DATA(J+1)=DATA(I+1)
DATA(I)=TEMPR
DATA(I+1)=TEMPI
ENDIF
M=N/2
1 IF ((M.GE.2).AND.(J.GT.M)) THEN
J=J-M
M=M/2
GO TO 1
ENDIF
J=J+M
11 CONTINUE
MMAX=2
2 IF (N.GT.MMAX) THEN
ISTEP=2*MMAX
THETA=6.28318530717959D0/(ISIGN*MMAX)
WPR=-2.D0*DSIN(0.5D0*THETA)**2
WPI=DSIN(THETA)
WR=1.D0
WI=0.D0
DO 13 M=1,MMAX,2
```

```
      DO 12 I=M,N,ISTEP
      J=I+MMAX
      TEMPR=WR*DATA(J)-WI*DATA(J+1)
      TEMPI=WR*DATA(J+1)+WI*DATA(J)
      DATA(J)=DATA(I)-TEMPR
      DATA(J+1)=DATA(I+1)-TEMPI
      DATA(I)=DATA(I)+TEMPR
      DATA(I+1)=DATA(I+1)+TEMPI
   12 CONTINUE
      WTEMP=WR
      WR=WR*WPR-WI*WPI+WR
      WI=WI*WPR+WTEMP*WPI+WI
   13 CONTINUE
      MMAX=ISTEP
      GO TO 2
      ENDIF
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Programmer: Eugene Tsiang c
c AXAF Mission Support Team c
c Date: 8/26/96 c
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
c Module name: jmkmod.f
c Modifications
c 8/30/96 distinction between random and periodic pulsers
c routines ranpile and perpile replace pilesim
c 9/4/96 correction to periodic pulser
c modified 10/9/96, added synchrotron
c modified 10/25/96, corrected wrong normalization in periodic
c pulser
c eliminated dparam, filnum 12/23/96, multiline capability
c added version control 12/23/96,
c corrected mistake in last virtual spectral bin (wrong index)
c version 6.4
c version 6.5 added ice filter 12/30/96
c added HRMA parameters and abswitch 12/31/96
c increased number of filters to 8 12/31/96, or any number
c implemented non-normalization of continuum for abswitch 1/2/97
c implemented HRMA effective area 1/6/97
c Version 7.0 released 1/6/97
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

10.0.11 subroutine jmkmod(ear, ne, param, worksp, photar)

Remark 11 *Main JMKMOD subroutine called by XSPEC, passes parameters to jmkply*

```
real*4 param(*), ear(0:ne)
real*4 worksp(*), photar(ne), grit
integer*4 ne, i, intdummy
double precision contpdf(8192)
double complex mgfarr(8192)
integer numcomp, numfilter, tally
integer maxcomp, maxfilter, intdum
parameter (maxcomp = 16, maxfilter = 16)
c increased number of components and filters 12/23/96
double precision
1 jfano, jpolyah, jeoff, jionen, jgain, jshift, jsigma, jdgain,
2 jnchan, jsestype,
3 jcontinuum, jecont, jcontnorm, jemax, jemin, jechar, jewidth,
4 jnInt, jtemper, jpressure, jargrat,
5 jshelfswitch, jt1norm, jt1par, jt2norm, jt2par, jshelfnorm,
6 jpulser, jpulsepos, jpulsesigma, jpulsenorm,
7 jnumcomp, jnumfilter, jpileup, jpilepar,
8 jbfield, jcurrent, jincline,
9 jabswitch, jphi, jtheta, jshell,
1 jlinenergy(maxcomp), jlinenorm(maxcomp),
2 jsrcstack(maxfilter), jsrcthick(maxfilter)
c singlelimit + 2*maxcomp + 2*maxfilter parameters 12/23/96 eyt
c passed to jmkply
* character*79 dummy
* write(dummy,800)ear(0),ear(ne),(param(i),i=1,5)
*800 format('JMmodel: ',1p,7g10.2)
*call xwrite(dummy,25)
c version control
character*96 dummy
real*4 version
version = 7.0
write(dummy, 800) version
800 format(' 1/6/96, JMKMOD version: ', f10.2)
c print *, dummy
call xwrite(dummy,11)
c uncomment this last statement within xspec
c now give them their assigned meanings and convert to dbl precision
```

jfano = param(1)
jpolyah = param(2)
jeoff = param(3)
jionen = param(4)
jgain = param(5)
jshift = param(6)
jsigma = param(7)
jdgain = param(8)
jnchan = param(9)
jsestype = param(10)
jcontinuum = param(11)
jecont = param(12)
jcontnorm = param(13)
jemax = param(14)
jemin = param(15)
jechar = param(16)
jewidth = param(17)
jnInt = param(18)
jtemper = param(19)
jpressure = param(20)
jargrat = param(21)
jshelfswitch = param(22)
jt1norm = param(23)
jt1par = param(24)
jt2norm = param(25)
jt2par = param(26)
jshelfnorm = param(27)
jpulser = param(28)
jpulsepos = param(29)
jpulsesigma = param(30)
jpulseNorm = param(31)
jnumcomp = param(32)
jnumfilter = param(33)
jpileup = param(34)
jpilepar = param(35)
jbfield = param(36)
jcurrent = param(37)
jincline = param(38)
jabswitch = param(39)
jphi = param(40)
jtheta = param(41)
jshell = param(42)

```
numcomp = jnumcomp
tally = 42
c singlelimit must be changed in params.for AND here, 1/2/97
intdum = tally + numcomp
do 2000 i = 1, numcomp
jlinenergy(i) = param(tally +i)
jlinenorm(i) = param(intdum + i)
2000 continue
numfilter = jnumfilter
tally = tally + 2*numcomp
intdum = tally + numfilter
do 3000 i = 1,numfilter
jsrcstack(i) = param(tally + i)
jsrcthick(i) = param(intdum + i)
3000 continue
* print *, 'In jmkmod, jsrcstack = ', jsrcstack
* print *, 'In jmkmod, jsrcthick = ', jsrcthick
*
tally = tally + 2*numfilter
* print *, 'In jmkmod, the total number of parameters is ', tally
call jmkply(contpdf, mgfarr,
1 jfano, jpolyah, jeoff, jionen, jgain, jshift, jsigma, jdgain,
2 jnchan, jsestype,
3 jcontinuum, jecont, jcontnorm, jemax, jemin, jechar, jewidth,
4 jnInt, jtemper, jpressure, jargrat,
5 jshelfswitch, jt1norm, jt1par, jt2norm, jt2par, jshelfnorm,
6 jpulser, jpulsepos, jpulsesigma, jpulsenorm,
7 jnumcomp, jnumfilter, jpileup, jpilepar,
8 jbfield, jcurrent, jincline,
9 jabswitch, jphi, jtheta, jshell,
1 jlinenergy, jlinenorm, jsrcstack, jsrcthick)
GRIT = 1.d-8
** print *, 'In jmkmod, '
DO 10, I = 1, NE
INTDUMMY = 0.5*((EAR(I-1)+EAR(I))) + GRIT
PHOTAR(I) = contpdf(INTDUMMY)
** print *, intdummy, photar(i)
10 CONTINUE
c photar is the returned array, ear, ne, param are input
return
END
```

10.0.12 subroutine jmkply(contpdf, mgfarr,

1 jfano, jpolyah, jeoff, jionen, jgain, jshift, jsigma, jdgain,
2 jnchan, jsestype,
3 jcontinuum, jecont, jcontnorm, jemax, jemin, jechar, jewidth,
4 jnInt, jtemper, jpressure, jargrat,
5 jshelfswitch, jt1norm, jt1par, jt2norm, jt2par, jshelfnorm,
6 jpulser, jpulsepos, jpulsesigma, jpulsenorm,
7 jnumcomp, jnumfilter, jpileup, jpilepar,
8 jbfield, jcurrent, jincline,
9 jabswitch, jphi, jtheta, jshell,
1 jlinenergy, jlinenorm, jsrcstack, jsrcthick)

Remark 12 *Takes parameters from jmkmod and generates continuum, calculates mgfs, pileup and IFT*

```
include 'params.for'
double precision contpdf(tChan)
double complex mgfarr(tChan)
double precision
1 jfano, jpolyah, jeoff, jionen, jgain, jshift, jsigma, jdgain,
2 jnchan, jsestype,
3 jcontinuum, jecont, jcontnorm, jemax, jemin, jechar, jewidth,
4 jnInt, jtemper, jpressure, jargrat,
5 jshelfswitch, jt1norm, jt1par, jt2norm, jt2par, jshelfnorm,
6 jpulser, jpulsepos, jpulsesigma, jpulsenorm,
7 jnumcomp, jnumfilter, jpileup, jpilepar,
8 jbfield, jcurrent, jincline,
9 jabswitch, jphi, jtheta, jshell,
1 jlinenergy(*), jlinenorm(*), jsrcstack(*), jsrcthick(*)
c total of singlelimit + 2*maxcomp + 2*maxfilter parameters 10/17/96 eyt
c singlelimit = 42 12/31/96
double complex jmmgf, contply, dgjmgf, contdg,
1 binmgf, contexp, brshft
external jmmgf, contply, dgjmgf, contdg, binmgf, contexp, brshft
integer k, tally
fano = jfano
polyaH = jpolyah
eOff = jeOff
ionen = jionen
gain = jgain
shift = jshift
sigma = jsigma
```

```
dgain = jdgain
nchan = jnchan
sestype = jsestype
continuum = jcontinuum
econt = jecont
contnorm = jcontnorm
emax = jemax
emin = jemin
echar = jechar
ewidth = jewidth
nInt = jnInt
temper = jtemper
pressure = jpressure
argrat = jargrat
shelfswitch = jshelfswitch
t1norm = jt1norm
t1par = jt1par
t2norm = jt2norm
t2par = jt2par
shelfnorm = jshelfnorm
pulser = jpulser
pulsepos = jpulsepos
pulsesigma = jpulsesigma
pulsenorm = jpulsenorm
numcomp = jnumcomp
numfilter = jnumfilter
pileup = jpileup
pilepar = jpilepar
bfield = jbfield
current = jcurrent
incline = jincline
abswitch = jabswitch
phi = jphi
theta = jtheta
shell = jshell
c strip parameters from call and put in common blocks
do 10 k = 1, numcomp
linenergy(k) = jlinenergy(k)
linenorm(k) = jlinenorm(k)
10 continue
do 20 k = 1, numfilter
srcstack(k) = jsrcstack(k)
```

```
srcthick(k) = jsrcthick(k)
20 continue
c end source stack
tChan = nChan*2
tChan1 = tChan + two
nInt1 = nInt - 1
deltaE = 1.d0
piDelt = dcmplx(0.d0, - mypi/(dble(nChan)*deltaE))
dEmax = (eMax - eMin)/ nInt1
if (eMax.ne.eMaxOld.or.
1eMin.ne.eMin.or.
2 dEmax.ne.dEmaxOld) then
enerflag = 1
c this means that some energy parameters have changed
eMaxOld = eMax
eMinOld = eMin
dEmaxOld = dEmax
do 200 k = 1, nInt
eArray(k) = eMin + dEmax*(k - 1)
evArr(k) = 1000.*eArray(k)
200 continue
else
enerflag = 0
c this means that no energy parameters have changed
endif
tally = singlelimit + 2*numcomp + 2*numfilter
* print *, 'In jmkply, the total number of parameters is ', tally
*
* print *, 'In jmkply',
* 1' fano = ', fano, ' polyaH = ', polyah, ' eoff = ', eoff,
* 2' ionEn = ', ionEn, ' gain = ', gain, ' shift = ', shift,
* 3' sigma = ', sigma, ' dgain = ', dgain, ' nchan = ', nchan,
* 4' sestype = ', sestype, ' continuum = ', continuum,
* 5' cont = ', econt, ' emax = ', emax, ' emin = ', emin,
* 6' echar = ', echar, ' ewidth = ', ewidth, ' nInt = ', nInt,
* 7' temperature = ', temper, 'pressure = ', pressure,
* 8' argon ratio = ', argrat, ' shelfswitch = ', shelfswitch,
* 9' t1norm = ', t1norm, ' t1par = ', t1par, ' t2norm = ', t2norm,
* 1' t2par = ', t2par, ' shelfnorm = ', shelfnorm,
* 2' pulser = ',pulser, ' pulsepos = ', pulsepos,
* 3' pulsesigma = ',pulsesigma,
* 4' number of components = ',numcomp,
```

```
* 5' numfilter = ', numfilter, ' pileup = ', pileup,
* 6' pilepar = ', pilepar, ' line energies = ', linenergy,
* 7' linenorm = ', linenorm, ' filter stack = ', srcstack,
* 8' filter thicknesses = ', srcthick,
* 9' bfield = ', bfield, 'current = ', current,
* 1' incline = ', incline, ' abswitch = ', abswitch,
* 2' phi = ', phi, ' theta = ', theta, ' shell = ', shell
* print *, 'In jmkply, auxiliaries: ', ' tChan = ', tChan,
* 1 ' tChan1 = ', tChan1, ' nInt1 = ', nInt1,' deltaE = ',deltaE,
* 2 ' piDelt = ', piDelt, ' dEmax = ', dEmax
*
** print *, 'eArray in jmkply '
** do 260 k = 1, nInt
** print *, eArray(k), ' ', evArr(k)
** 260 continue
if (continuum.ne.0) call damper
c damper returns an array 'alldamp(nInt)' containing the
c action of all the sources(filters)
c single electron spectral type sestype selected:
if (sestype.eq.2) then
call mgfmake(binmgf, contexp, mgfarr)
else if (sestype.eq.3) then
call mgfmake(dgjmgf, contdg, mgfarr)
else
call mgfmake(jmmgf, contply, mgfarr)
c this is the default (sestype.eq.1)
endif
c pileup if any is performed within mgfmake 8/30/96
** print *, 'pileup parameter in jmkply ', pileup, pilepar
call invert(mgfarr, contpdf)
c jmpdf(tChan) returned by Fourier inverting mgfarr(tChan)
** print *, 'In jmkply '
** do 2000 k = 1,tChan
** print *, contpdf(k), mgfarr(k)
** 2000 continue
return
end
```

10.0.13 subroutine ranpile(mgfarr)

Remark 13 *Random pulser pileup routine*

```
c for random pulsers
include 'params.for'
```

```
double complex mgfarr(tChan)
integer i
double precision someval, cpilepar, normsum
double complex mgfdum
if (pilepar.eq.one) return
if (pilepar.eq.0.0) then
c almost never happens, because this implies unlimited pileup
do 800 i = 1,tChan
mgfarr(i) = (0.0,0.0)
800 continue
return
endif
normsum = mgfarr(1)
cpilepar = one - pilepar
someval = cpilepar/normsum
do 400 i = 1, tChan
mgfdum = mgfarr(i)
mgfarr(i) = pilepar*mgfdum/(one - someval*mgfdum)
400 continue
return
end
```

10.0.14 subroutine perpile(mgfarr, pulsarr)

Remark 14 *Periodic pulser pileup routine*

```
c for periodic pulsers, mgfarr not normalized
include 'params.for'
double complex mgfarr(tChan), pulsarr(tChan)
integer i
double precision someval, cpilepar, normsum
double complex mgfdum
if (pilepar.eq.one) return
if (pilepar.eq.0.0) then
c almost never happens, because this implies unlimited pileup
do 800 i = 1, tChan
mgfarr(i) = (0.0, 0.0)
800 continue
return
endif
normsum = mgfarr(1)
cpilepar = one - pilepar
someval = cpilepar/normsum
do 400 i = 1, tChan
```

```
mgfdum = mgfarr(i)
mgfarr(i) = pilepar*(pulsarr(i) + mgfdum)/
1 (one - someval*mgfdum)
400 continue
return
end
```

10.0.15 subroutine mgfmake(linemgf, contmgf, mgfarr)

Remark 15 *Generates total mgf from line, pulser and continuum mgfs, adjusts for pileup.*

```
double complex linemgf, contmgf, brshft
external linemgf, contmgf, brshft
include 'params.for'
double precision oldenergy, oldshift, oldsigma
integer j, k, dumbcomp
double complex dumbarr(8192), lcparr(8192,32)
c make sure these arrays are large enough to hold components
double complex mgfarr(tChan)
double complex dumbtemp, sum
** double precision dumbpdf(8192)
c numcomp is the number of line components
oldenergy = eChar
c oldenergy saves the current of eChar, while eChar
c is being "borrowed"
do 650 k = 1, numcomp
eChar = linenergy(k)
call mgftbl(linemgf,dumbarr)
** call invert(dumbarr, dumbpdf)
do 660 j = 1, tChan
lcparr(j,k) = dumbarr(j)
** print *, dumbpdf(j), dumbarr(j)
660 continue
650 continue
dumbcomp = numcomp
c this equates dumbcomp to the number of line components
eChar = oldenergy
c this restores eChar back to its original value
if (continuum.ne.0) then
call mgftbl(contmgf, dumbarr)
dumbcomp = dumbcomp+1
dumbtemp = dumbarr(1)
c normalization constant for continuum
```

```
if (abswitch.eq.0) then
c abswitch.eq.0 (relative calibration)
do 670 j = 1, tChan
dumbarr(j) = dumbarr(j)/dumbtemp
lcparr(j,dumbcomp) = dumbarr(j)
670 continue
c case of relative calibration where just the counts matter
else
c abswitch.ne.0 (absolute calibration)
do 780 j = 1, tChan
lcparr(j,dumbcomp) = dumbarr(j)
780 continue
c no normalization for absolute calibration 1/2/97
endif
** call invert(dumbarr, dumbpdf)
** print *,'In mgfmake, the cont pdf/mgf is '
** do 770 j = 1, tChan
** print *, dumbpdf(j), dumbarr(j)
** 770 continue
linenorm(dumbcomp) = contnorm
c "borrowing" linenorm for the continuum
endif
c at this point, dumbcomp is the number of lines and continuum,
c if any
c by convention
c pulser>0, periodic pulser
c pulser = 0, no pulser
c pulser<0, random pulser
if (pulser.ne.0) then
c there is a pulser
oldshift = shift
oldsigma = sigma
shift = pulsepos
sigma = pulsesigma
call mgftbl(brshft, dumbarr)
c the pulser is a shifted gaussian
** call invert(dumbarr, dumbpdf)
** print *, 'In mgfmake, the pulser/pdf/mgf is '
dumbcomp = dumbcomp+1
do 680 j = 1, tChan
lcparr(j,dumbcomp) = dumbarr(j)
** print *, dumbpdf(j), dumbarr(j)
```

```
680 continue
shift = oldshift
sigma = oldsigma
linenorm(dumbcomp) = pulsenorm
endif
c at this point, dumbcomp is the number of lines and continuum
c and pulser, if any
c correction added 10/25/96 due to wrong normalization for
c periodic pulser
if (pileup.ne.0.and.pulser.gt.0) then
do 840 j = 1, tChan
dumbarr(j) = linenorm(dumbcomp)*lcparr(j,dumbcomp)
840 continue
dumbcomp = dumbcomp - 1
endif
c this is the case of pileup with a periodic pulser
c dumbcomp is lowered by 1 to exclude the pulser, which is last
do 700 j = 1, tChan
sum = 0.0
do 750 k = 1, dumbcomp
sum = sum + linenorm(k)*lcparr(j,k)
750 continue
mgfarr(j) = sum
700 continue
if (pileup.eq.0) then
return
c the case of NO pileup, NO further processing of mgfarr
c the remaining cases have pileup
else if (pulser.gt.0) then
c the case of a periodic pulser with pileup
call perpile(mgfarr, dumbarr)
c dumbarr is just lcparr(j,dumbcomp+1)*linenorm(dumbcomp+1),
c the mgf of the pulser. Remember that dumbcomp has been set
c to dumbcomp-1, one less because of pulser
else
c this includes the general cases of random pulser and no pulser
c pulser.le.0
call ranpile(mgfarr)
endif
return
end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
c Programmer: Eugene Tsiang c
c AXAF Mission Support Team c
c Date: 8/26/96 c
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
c Module name: jmmgf1.f
c Modifications:
c 9/4/96 tail parameters changed using logs to prevent instability
c10/31/96 previous tail parameters implementation corrected
c 11/4/96 NaN in xspec caused by tail parameters overflowing
c cured by taking inverse and causing underflowing
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

10.0.16 double complex function jmmgf(slap)

Remark 16 *Mgf with SES type 1 mgf, with Laplace parameter 'slap'*

```
c moment generating function for a line
include 'params.for'
double complex slap, sglmgf, sglshelf, plymgf, brshft
external plymgf
if (shelfswitch.eq.0) then
jmmgf = sglmgf(slap,plymgf)*brshft(slap)*plymgf(slap)
else
jmmgf = sglshelf(slap,plymgf)*brshft(slap)*plymgf(slap)
* jmmgf = sglmgf(slap,plymgf)
endif
return
end
```

10.0.17 double complex function dgjmgf(slap)

Remark 17 *Mgf with SES type 3 mgf.*

```
c moment generating function for a line
include 'params.for'
c make sure this is changed to 'params.f' in xspec
* include 'params.f'
double complex slap, sglmgf, sglshelf, dgmgf, brshft
external dgmgf
if (shelfswitch.eq.0) then
dgjmgf = sglmgf(slap, dgmgf)*brshft(slap)*dgmgf(slap)
else
dgjmgf = sglshelf(slap, dgmgf)*brshft(slap)*dgmgf(slap)
* dgjmgf = sglmgf(slap, dgmgf)
endif
```

```
      return
      end
```

10.0.18 double complex function binmgf(slap)

Remark 18 *Mgf with SES type 2 Mgf*

```
      c moment generating function for a line
      include 'params.for'
      double complex slap, sglmgf, sglshelf, expmgf, brshft
      external expmgf
      if (shelfswitch.eq.0) then
      binmgf = sglmgf(slap, expmgf)*brshft(slap)*expmgf(slap)
      * binmgf = sglmgf(slap, expmgf)
      else
      binmgf = sglshelf(slap, expmgf)*brshft(slap)*expmgf(slap)
      endif
      return
      end
```

10.0.19 double complex function plymgf(slap)

Remark 19 *SES type 1 mgf: Polya distribution*

```
      c single electron spectral types
      include 'params.for'
      double complex slap
      plymgf = (one/(one + gain*slap/polyaH))**polyaH
      return
      end
      double complex function expmgf(slap)
      c case when polyaH goes to infinity – mgf for delta function ses
      include 'params.for'
      double complex slap
      expmgf = exp(-gain*slap)
      return
      end
```

10.0.20 double complex function dgmgf(slap)

Remark 20 *SES type 3 mgf: delta function*

```
      c case when gain varies between gain - dgain and gain + dgain
      include 'params.for'
      double complex slap
      double complex temp, temp1, temp2
      if (slap.eq.(0.0,0.0)) then
```

```
dgmgf = (1.0,0.0)
return
endif
if (dgain.gt.pebble) then
if (abs(polyaH - one).gt.pebble) then
temp1 = polyaH + (gain - dgain)*slap
temp2 = polyaH + (gain + dgain)*slap
temp = temp1/(temp1/polyaH)**polyaH -
1 temp2/(temp2/polyaH)**polyaH
dgmgf = temp/(2.*dgain*slap*(polyaH-one))
return
else
temp1 = one + (gain - dgain)*slap
temp2 = one + (gain + dgain)*slap
dgmgf = log(temp2/temp1)/(2.*dgain*slap)
return
endif
else
dgmgf = (one/(one + gain*slap/polyaH))**polyaH
return
endif
end
```

10.0.21 double complex function sglmgf(slap, sesmgf)

Remark 21 *MGF with SES type 1 mgf without broadening and extra electron*

```
c mgf for a line without broadening and extra electron
include 'params.for'
double complex slap, sesmgf
double precision pFano, nPhot
double complex temp
c local variables
** print *, 'slap in sglmgf = ', slap
pFano = one - fano
nPhot = (eChar - eOff)/ionEn
temp = sesmgf(slap)
c channel shift and broadening
if (abs(pFano) .gt. pebble) then
sglmgf = (one + pFano*(temp - one))**(nPhot/pFano)
c general jm case
else
sglmgf = exp((temp - one)*nPhot)
c jm-Prescott intermediate case (Fano factor = 1)
```

```
endif
return
end
```

10.0.22 double complex function sglshelf(slap, sesmgf)

Remark 22 *Shelf mgf*

```
c mgf for a line without broadening and extra electron
include 'params.for'
double complex slap, sesmgf
double precision pFano, nPhot
double complex temp, temp1, temp6, temp7, temp0
double precision nupper, temp2, temp3, temp4, temp5
double complex shelfpart, tpart1, tpart2
c local variables
** print *, 'slap in sglshelf = ', slap
pFano = one - fano
nPhot = (eChar - eOff)/ionEn
temp = sesmgf(slap)
if (temp.eq.(1.0,0.0)) then
sglshelf = (1.0,0.0)
return
endif
c this point evaluated separately because of mild apparent
c singularities
if (abs(pFano) .gt. pebble) then
temp1 = fano + pFano*temp
temp7 = - log(temp1)
nupper = nPhot/pFano
temp6 = temp1**nupper
temp0 = (one + pFano*(temp - one))**nupper
c the JM case for a line is temp0
if (shelfnorm.ne.0.0)
1 shelfpart = (pFano*(one - temp6))/(nPhot*temp7)
if (t1norm.ne.0.0) then
temp2 = t1par**nPhot
temp3 = pFano*log(t1par)
if (temp2.gt.one) then
temp2 = one/temp2
c taking reciprocal averts overflow problem 11/4/96
tpart1 = (temp6 - temp2)/
1 ((one - temp2)*(one - temp7/temp3))
else
```

```
tpart1 = (temp2*temp6 - one)/
1 ((temp2 - one)*(one - temp7/temp3))
endif
endif
if (t2norm.ne.0.0) then
temp4 = t2par**nPhot
temp5 = pFano*log(t2par)
if (temp4.gt.one) then
temp4 = one/temp4
c taking reciprocal averts overflow problem 11/4/96
tpart2 = (temp6 - temp4)/
1 ((one - temp4)*(one - temp7/temp5))
else
tpart2 = (temp4*temp6 - one)/
1 ((temp4 - one)*(one - temp7/temp5))
endif
endif
c general jm case
else
temp0 = exp(nPhot*(temp - one))
c the Prescott case for the line is temp0
if (shelfnorm.ne.0.0)
1 shelfpart = (temp0 - one)/((temp - one)*nPhot)
if (t1norm.ne.0.) then
temp2 = t1par**nPhot
tpart1 = (- one + temp2*temp0)/
1 ((temp2 - one)*(one + (temp - one)/log(t1par)))
endif
if (t2norm.ne.0.) then
temp4 = t2par**nPhot
tpart2 = (- one + temp4*temp0)/
1 ((temp4 - one)*(one + (temp - one)/log(t2par)))
endif
c jm-Prescott intermediate case (Fano factor = 1)
endif
* print *, "Within jmmgf1, fano, pFano,
* 1 slap, temp, temp0, temp1, temp2, temp3, temp4, temp5,
* 2 temp6, nPhot, tpart1, tpart2, shelfpart, t1norm, t2norm,
* 3 shelfnorm = ", fano, pFano,
* 4 slap, temp, temp0, temp1, temp2, temp3, temp4, temp5,
* 5 temp6, nPhot, tpart1, tpart2, shelfpart, t1norm, t2norm,
* 6 shelfnorm
```

```
sglshelf = (temp0 +
1 t1norm*tpart1 + t2norm*tpart2 + shelfnorm*shelfpart)/
2 (one + t1norm + t2norm + shelfnorm)
c sglshelf includes the line, the shelf, and two tails
return
end
```

10.0.23 double complex function brshft(slap)

Remark 23 *Broadening and shift mgfs*

```
c shifting and broadening mgf
include 'params.for'
double complex slap
if (slap.eq.(0.0,0.0)) then
brshft = (1.0,0.0)
return
endif
if (shift.eq.0.0.and.sigma.eq.0.0) then
brshft = one
return
elseif (shift.eq.0.0) then
brshft = exp(half*(slap*sigma)**2)
return
elseif (sigma.eq.0.0) then
brshft = exp(-slap*shift)
return
else
brshft = exp(-slap*shift + half*(slap*sigma)**2)
return
endif
end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Programmer: Eugene Tsiang c
c AXAF Mission Support Team c
c Date: 8/26/96 c
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
c Module name: kramer.f
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

10.0.24 double complex function contply(slap)

Remark 24 *Broadening and shift and extra electron applied to continuum mgf: SES type 1*

```
      include 'params.for'
      c make sure this is changed to 'params.f' in xspec
      * include 'params.f'
      double complex slap, plymgf, ctmmgf, brshft
      external plymgf
      ** print *, 'slap in contply = ',slap
      ** contply = ctmmgf(slap, plymgf)
      contply = ctmmgf(slap, plymgf)*brshft(slap)*plymgf(slap)
      ** print *, 'slap back in contply = ', slap
      return
      end
```

10.0.25 double complex function contexp(slap)

Remark 25 *Broadening and shift and extra electron applied to continuum mgf: SES type 2*

```
      include 'params.for'
      c make sure this is changed to 'params.f' in xspec
      * include 'params.f'
      double complex slap, expmgf, ctmmgf, brshft
      external expmgf
      ** print *, 'slap in contexp = ', slap
      ** contexp = ctmmgf(slap, expmgf)
      contexp = ctmmgf(slap, expmgf)*brshft(slap)*expmgf(slap)
      ** print *, 'slap back in contexp = ', slap
      return
      end
```

10.0.26 double complex function contdg(slap)

Remark 26 *Broadening and shift and extra electron applied to continuum mgf: SES type 3*

```
      include 'params.for'
      c make sure this is changed to 'params.f' in xspec
      * include 'params.f'
      double complex slap, dgmgf, ctmmgf, brshft
      external dgmgf
      ** print *, 'slap in contdg = ', slap
      ** contdg = ctmmgf(slap, dgmgf)
      contdg = ctmmgf(slap, dgmgf)*brshft(slap)*dgmgf(slap)
      ** print *, 'slap back in contdg = ', slap
      return
      end
      ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
c Programmer: Eugene Tsiang c
c AXAF Mission Support Team c
c Date: 8/26/96 c
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
c Module name: matnu.f
c added ice filter, lengthened data, mu and arr arrays 12/30/96
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

10.0.27 subroutine newmac(kElem, extinct)

Remark 27 *Henke Tables reader*

```
include 'params.for'
integer kElem
* double precision eoff, emax, demax
*double precision keV
integer best
* integer nInt
double precision mubest, arrbest
* integer best1
double precision data(maxhenke,3),mu(maxhenke),arr(maxhenke)
c maxhenke = 1024, should be enough 12/30/96, old array too small
c these are arrays associated with the Henke Tables
c arr is the energy array
c mu is the extinction array calculated from Henke
double precision extinct(nInt), aw
c extinct is the interpolated extinction array
character*2 el
character*1 el1
character*50 henke, header
character*44 temp
integer i, n, nhenke
temp = '/home/saolib/optical_constants/henke_ext_95/'
aw = elemWate(kElem)
el = elemName(kElem)
if (el(2:2).eq.' ') then
el1 = el(1:1)
henke = temp//el1//'.nff'
else
henke = temp//el//'.nff'
endif
* print *, 'In newmac, kElem, aw, el, el1 = ',
* 1 kElem, aw, el, el1
```

```
* print *, 'In newmac, reading ', henke
* call xwrite('jmkram: reading file '//henke,25)
open(unit =1,file = henke)
read(1,*) header
i=0
20 continue
i=i+1
read (1,*, end = 30) data(i,1), data(i,2), data(i,3)
goto 20
30 continue
n=i-1
close(1)
*print *, 'data read in'
nhenke = n
do 40 i=1,n
mu(i) = 4.208d7*data(i,3)/data(i,1)/aw
40 continue
*print *, 'mu = '
*print *, mu
*print *, 'n = ', n
*print *, 'In matnew.f, arr = '
do 50 i=1,n
arr(i)=data(i,1)
* print *, i, data(i,1), data(i,2), data(i,3),mu(i)
50 continue
* eoff = 0.00
* emax = 10.
* nInt = 512
* demax = (emax - eoff)/(nInt - 1)
* keV = 1000.
* print *, ' energy array = '
* do 650 i = 1, nInt
*evArr(i) = keV*(eoff + (i - 1)*demax)
* print *, i, evArr(i)
* 650 continue
best = 1
* best1 = 2
* print *, "In newmac, i, evArr, arrbest, best, mubest, extinct"
do 660 i = 1, nInt
* call locate(arr, n, evArr(i), best1)
call hunt(arr, n, evArr(i), best)
if (nhenke.gt.best.and.best.gt.0) then
```

```
mubest = mu(best)
arrbest = arr(best)
extinct(i) = mubest + (mu(best+1)-mubest)*
1 (evArr(i) - arrbest)/(arr(best+1)- arrbest)
else if (best.le.0) then
extinct(i) = 999999999.
arrbest = evArr(i)
mubest = extinct(i)
else
mubest = mu(best)
arrbest = arr(best)
extinct(i) = mubest
endif
* print *, i, evArr(i), arrbest, best, mubest ,extinct(i)
660 continue
return
end
```

10.0.28 SUBROUTINE hunt(xx,n,x,jlo)

Remark 28 *Interpolation routine: hunting for value, from Numerical Recipes*

```
INTEGER jlo,n
double precision x,xx(n)
INTEGER inc,jhi,jm
LOGICAL ascnd
ascnd=xx(n).gt.xx(1)
if(jlo.le.0.or.jlo.gt.n)then
jlo=0
jhi=n+1
goto 3
endif
inc=1
if(x.ge.xx(jlo).eqv.ascnd)then
1 jhi=jlo+inc
if(jhi.gt.n)then
jhi=n+1
else if(x.ge.xx(jhi).eqv.ascnd)then
jlo=jhi
inc=inc+inc
goto 1
endif
else
jhi=jlo
```

```
2 jlo=jhi-inc
if(jlo.lt.1)then
jlo=0
else if(x.lt.xx(jlo).eqv.ascnd)then
jhi=jlo
inc=inc+inc
goto 2
endif
endif
3 if(jhi-jlo.eq.1)return
jm=(jhi+jlo)/2
if(x.gt.xx(jm).eqv.ascnd)then
jlo=jm
else
jhi=jm
endif
goto 3
END
C (C) Copr. 1986-92 Numerical Recipes Software 4p='.
```

SUBROUTINE locate(xx,n,x,j)

Remark 29 *Locating a value during interpolation, due to Numerical Recipes*

```
INTEGER j,n
double precision x,xx(n)
INTEGER jl,jm,ju
* integer i
* print *, 'In locate , n = ', n
* print *, 'In locate , j = ', j
* print *, 'In locate, x = ', x
* print *, 'In locate , xx = '
* do 60 i = 1,n
* print *, i, xx(i)
* 60 continue
jl=0
ju=n+1
10 if(ju-jl.gt.1)then
jm=(ju+jl)/2
if((xx(n).gt.xx(1)).eqv.(x.gt.xx(jm)))then
jl=jm
else
ju=jm
```

```
endif
goto 10
endif
j=jl
return
END
C (C) Copr. 1986-92 Numerical Recipes Software 4p='.
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Programmer: Eugene Tsiang c
c AXAF Mission Support Team c
c Date: 8/26/96 c
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
c Module name: mgftbl.f
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

10.0.29 subroutine mgftbl(stunt,mgfarr)

Remark 30 *Generates a table of moment generating functions at a particular s.*

```
c mgf table generator prior to inverse FFT
include 'params.for'
c make sure this is changed to 'params.f' in xspec
* include 'params.f'
integer k
double complex stunt, mgfarr(tChan), piK
external stunt
** print *, 'deltaE in mgftbl = ', deltaE
** print *, 'nChan in mgftbl =', nChan
** print *, 'tChan in mgftbl =', tChan
** print *, 'tChan1 in mgftbl = ', tChan1
** print *, 'piDelt in mgftbl = ', piDelt
** print *, 'emax in mgftbl = ', emax
mgfarr(1) = stunt(dcmplx(0,0))
do 100 k = 2, nChan
piK = dble(k-1)*piDelt
mgfarr(k) = stunt(piK)
* print *, 'back in mgftbl , piK = ', piK
mgfarr(tChan1 - k) = dconjg(mgfarr(k))
100 continue
mgfarr(nChan + 1) = dreal(stunt(dble(nChan)*piDelt))
** print *, 'mgfarr in mgftbl = ', mgfarr
return
end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Programmer: Eugene Tsiang c
c AXAF Mission Support Team c
c Date: 1/6/97 c
c Place: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 7.00 c
c Module name: mirror.f
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

10.0.30 subroutine mirror(damping)

Remark 31 *Goes to a HRMA effective area file and reads in the values, a la Henke Table.*

```
include 'params.for'
integer iphi, itheta
double precision resphi, restheta
character*6 phiasc, thetasc
character*4 shellasc
character*48 temp
character*70 filename
double precision data(maxhenke,2), mu(maxhenke), arr(maxhenke)
integer i, nrecords, best
c maxhenke = 1024
c arr is the energy array
c mu is the mirror reflectivity array
double precision damping(nInt), mubest, arrbest
c damping is the interpolated dampingion array
iphi = phi
itheta = theta
resphi = phi - iphi
restheta = theta - itheta
temp = '/data/axafj/dj/axaf/cal-hdbk/v4/saohdos3/rdb/eaf'
open(unit = 10, file = 'board.dat')
write (10,14) iphi,resphi,itheta, restheta, shell
14 format(i3.3,f3.2,i3.3,f3.2,i4.4)
rewind (10)
read(10,15) phiasc, thetasc, shellasc
15 format(2a6,a4)
* print*, 'phiasc = ', phiasc, ', thetasc = ', thetasc
filename = temp//phiasc//'t'//thetasc//'s'//shellasc//'.rdb'
* print *, 'filename ' , filename
close(10)
c business end
```

```
open (unit = 11, file = filename)
i = 0
20 continue
i = i + 1
read(11, *, end = 30) data(i,1), data(i,2)
goto 20
30 continue
nrecords = i - 1
* print *, ' number of records read is ', nrecords
* print *, 'data '
* do 40 i = 1,nrecords
* print *, i, data(i,1), data(i,2)
* 40 continue
* print *, 'arr and mu = '
do 500 i = 1, nrecords
arr(i) = data(i,1)
mu(i) = data(i,2)
* print *, arr(i), mu(i)
500 continue
best = 1
* print *, 'Within mirror'
do 800 i = 1, nInt
call hunt(arr, nrecords, evArr(i), best)
if (nrecords.gt.best.and.best.gt.0) then
mubest = mu(best)
arrbest = arr(best)
damping(i) = mubest + (mu(best+1)-mubest)*
1 (evArr(i) - arrbest)/(arr(best+1)- arrbest)
else if (best.le.0) then
damping(i) = 0.
* arrbest = evArr(i)
* mubest = damping(i)
else
* mubest = mu(best)
* arrbest = arr(best)
* damping(i) = mubest
damping(i) = mu(best)
endif
* print *, i, evArr(i), damping(i)
800 continue
close(11)
return
```

```
end
```

10.0.31 SUBROUTINE qsimp(func,a,b,s)

Remark 32 *Simpson's Rule of integration, from Numerical Recipes*

```
INTEGER JMAX
REAL a,b,func,s,EPS
EXTERNAL func
PARAMETER (EPS=1.e-6, JMAX=20)
CU USES trapzd
INTEGER j
REAL os,ost,st
ost=-1.e30
os= -1.e30
do 11 j=1,JMAX
call trapzd(func,a,b,st,j)
s=(4.*st-ost)/3.
if (abs(s-os).lt.EPS*abs(os)) return
os=s
ost=st
11 continue
pause 'too many steps in qsimp'
END
C (C) Copr. 1986-92 Numerical Recipes Software %5'KNp='.
```

10.0.32 SUBROUTINE qtrap(func,a,b,s)

Remark 33 *Trapezoidal integration, from Numerical Recipes*

```
INTEGER JMAX
REAL a,b,func,s,EPS
EXTERNAL func
PARAMETER (EPS=1.e-6, JMAX=20)
CU USES trapzd
INTEGER j
REAL olds
olds=-1.e30
do 11 j=1,JMAX
call trapzd(func,a,b,s,j)
if (abs(s-olds).lt.EPS*abs(olds)) return
olds=s
11 continue
pause 'too many steps in qtrap'
END
C (C) Copr. 1986-92 Numerical Recipes Software %5'KNp='.
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cProgrammer: Eugene Tsiang c
cAXAF Mission Support Team c
cDate: 8/26/96 c
cPlace: Smithsonian Astrophysical Observatory, Cambridge, MA c
c Spectral Analysis Program for XSPEC Revision 6.00 c
cModule name: source.f c
cModifications:
c 9/11/96 Added Mn and polypro
c 9/16/96 Checked shelves, found error in Ge dens in ssdgerm
c 9/17/96 Changed P10 and CH4 using right formulae
c10/9/96 emax to eCont
c 12/30/96 added ice filter
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

10.0.33 subroutine damper

Remark 34 *Calculates the total effect of a stack of source/filters*

```
c this subroutine returns values in the array alldamp(integral) in
c the common array "filstack"
include 'params.for'
double precision dummy, dumdamp(integral)
integer j, k
integer kElem
double precision hold
* print *, "In damper, number filter, kElem, thick "
do 800 k = 1, numfilter
kElem = srcstack(k)
hold = srcthick(k)
* print *, k, kElem, hold
call source(kElem, hold, dumdamp)
c call source(scrstack(k),scrthick(k), dumdamp)
do 900 j = 1, nInt
srcdamp(j,k) = dumdamp(j)
c print *, k, j, srcdamp(j,k), dumdamp(j)
900 continue
800 continue
do 1000 j = 1, nInt
dummy = one
do 1100 k = 1, numfilter
dummy = dummy*srcdamp(j,k)
1100 continue
alldamp(j) = dummy
```

```
1000 continue
* 31 format(i8, ' ', g16.8e4,' ',g16.8e4,' ',g16.8e4,' ',g16.8e4,' ',
* 1 g16.8e4,' ', g16.8e4,' ', g16.8e4,' ', g16.8e4,' ',g16.8e4,
* 2 ' ',g16.8e4)
* open(30,file = 'damping.dat', form = 'formatted')
*
* do 2000 j = 1, nInt
* write(30,31) j, eArray(j),alldamp(j),srcdamp(j,1),
* 1 srcdamp(j,2),
* 2 srcdamp(j,3),srcdamp(j,4),srcdamp(j,5), srcdamp(j,6),
* 3 srcdamp(j,7), srcdamp(j,8)
* 2000 continue
*
* close(30)
return
end
```

10.0.34 subroutine krasub(damping)

Remark 35 *Kramers source filter*

```
include 'params.for'
c Make sure this is 'params.f' in xspec!
* include 'params.f'
double precision damping(nInt)
integer i
do 10 i=1,nInt
if (eArray(i) .eq. 0) then
damping(i)=0.0
else
damping(i) = (eCont/eArray(I) -one)
c eMax changed to Econt 10/9/96
endif
10 continue
** print *, 'In krasub, eMax is ', eMax
** print *, 'In krasub, damping is ', damping
return
end
```

10.0.35 subroutine linegauss(damping)

Remark 36 *Gaussian source/filter*

```
include 'params.for'
double precision damping(nInt), eWid2, rootpi
integer i
```

```
eWid2 = eWidth*eWidth
rootpi = one/(eWidth*sqrt(two*mypi))
do 450 i = 1, nInt
damping(i) = rootpi*
1 exp(-(eArray(i) - eChar)**2/(two*eWidth*eWidth))
450 continue
** print *, 'In linegauss, damping is ', damping
return
end
```

10.0.36 subroutine source(kElem, thick, damping)

Remark 37 *Calculates the transmission/absorption values for a filter corresponding to 'kElem'*

```
include 'params.for'
integer kElem
double precision thick, damping(nInt)
double precision muabs1(integral), muabs2(integral),
1 muabs3(integral), muabs4(integral)
double precision muabs(integral)
double precision stoich(4)
double precision elempith, cargrat, normal
integer i
c special nonelements:
c 'p1' is parylene
c 'p2' is polyimide
c 'p3' is mylar (density wrong)
c 'p4' is teflon (polytetrafluoroethylene, density wrong)
c 'p5' is pn (ammonium dihydrogen phosphate, density wrong)
c 'p6' is pvc (polyvinyl chloride, density wrong)
c 'p7' is polypropylene
c 'ks' is the kramers continuum
c 'sy' is the synchrotron radiation spectrum
c 'gx' is the germanium quantum efficiency
c 'ax' is the fpc p10 quantum efficiency
c 'mx' is the fpc methane quantum efficiency
c 'ea' is the effective area of the mirror
c 'gn' is the gaussian
c 'px' is the pella continuum
c 'un' is the all-pass (unity) filter
if (kElem.eq.parylene) then
elempith = elemDens(kElem)*thick
stoich(1) = 5.98306d-1
```

```
stoich(2) = 4.01694d-1
call newmac(cplace, muabs1)
call newmac(hplace, muabs2)
** print *, 'In parylene in source, thick, elempith =',
** 1thick, elempith
do 100 i = 1,nInt
muabs(i) = muabs1(i)*stoich(1) + muabs2(i)*stoich(2)
damping(i) = exp(-muabs(i)*elempith)
** print *, i, damping(i),muabs(i), muabs1(i), muabs2(i)
100 continue
** print *, 'In parylene in source, damping = '
** do 120 i = 1, nInt
** print *, damping(i)
** 120 continue
c stoichiometry of parylene
elseif (kElem.eq.polyimide) then
elempith = elemDens(kElem)*thick
call newmac(cplace, muabs1)
call newmac(hplace, muabs2)
call newmac(oplace, muabs3)
call newmac(nplace, muabs4)
stoich(1) = 22.0*elemWate(cplace)
stoich(2) = 10.0*elemWate(hplace)
stoich(3) = 4.0*elemWate(oplace)
stoich(4) = 2.0*elemWate(nplace)
normal = stoich(1) + stoich(2) + stoich(3) + stoich(4)
do 200 i = 1, nInt
muabs(i) = (stoich(1)*muabs1(i) + stoich(2)*muabs2(i) +
1 stoich(3)*muabs3(i) + stoich(4)*muabs4(i))/
2 normal
damping(i) = exp(-muabs(i)*elempith)
200 continue
c stoichiometry of polyimide: C22 H10 O4 N2
elseif (kElem.eq.polypro) then
elempith = elemDens(kElem)*thick
call newmac(cplace, muabs1)
call newmac(hplace, muabs2)
stoich(1) = elemWate(cplace)
stoich(2) = 2.0*elemWate(hplace)
normal = stoich(1) + stoich(2)
do 250 i = 1, nInt
muabs(i) = (stoich(1)*muabs1(i) + stoich(2)*muabs2(i))/
```

```
1 normal
damping(i) = exp(-muabs(i)*elempith)
250 continue
elseif (kElem.eq.ice) then
elempith = elemDens(kElem)*thick
call newmac(hplace, muabs1)
call newmac(oplace, muabs2)
stoich(1) = 2.0*elemWate(hplace)
stoich(2) = elemwate(cplace)
normal = stoich(1) + stoich(2)
do 270 i = 1, nInt
muabs(i) = (stoich(1)*muabs1(i) + stoich(2)*muabs2(i))/
1 normal
damping(i) = exp(-muabs(i)*elempith)
270 continue
elseif (kElem.eq.kramers) then
call krasub(damping)
c Kramers continuum
elseif (kElem.eq.synchro) then
c 10/9/96 addition
bessyEnergy = eCont
* eCont already in param list
boost = boostnorm*(bessyEnergy/bessyEnorm)
sind = thick
c thick takes on the value of detector subtended angle
boostsind = boost*sind
* nominally 0.236
* subtended angle added to param list as "thick" 10/9/96
sind = boostsind/boost
boostsind2 = boostsind*boostsind
boostcosd = sqrt(boost*boost - boostsind*boostsind)
tand = boostsind/boostcosd
bessyBfield = bfield
bessyI = current
* bessyncline = 0.
* inclination of ecliptic should be added to param list 10/9/96
* special integration for nonzero bessyncline not implemented 10/9/96
* These values are input parameters from xspec
bessyfluxnorm = fluxconst*(bessyI/bessyInorm)/
1 (((bessyEnergy/bessyEnorm)**3)*(bessyBfield/bessyBnorm))
c flux norm in photons per keV^2 per second
c It is the product of 4*3^(1.5)/(20*Pi^3) and the flux per kev^2 per sec
```

```
bessyEcrit = enercrit*
1 ((bessyEnergy/bessyEnorm)**2)*(bessyBfield/bessyBnorm)
c synchrotron critical energy in keV
* print *, 'In synchro '
* print *, 'boost, sind, boostsind, boostsind2, boostcosd, tand =',
* 1 boost, sind, boostsind, boostsind2, boostcosd, tand
*
* print *, 'bessyEnorm, bessyBnorm, bessyInorm = ',
* 1 bessyEnorm, bessyBnorm, bessyInorm
*
* print *, 'bessyEnergy, bessyBfield, bessyI, bessyfluxnorm = ',
* 1 bessyEnergy, bessyBfield, bessyI, bessyfluxnorm
*
* print *, 'bessyEcrit = ', bessyEcrit
call bessyflux(damping)
* print *, 'i, energy, y, damping = '
* do 401 i = 1,nInt
* print *, i, earray(i), yenergy(i), damping(i)
* 401 continue
c synchrotron radiation
elseif (kElem.eq.effarea) then
call mirror(damping)
c effective area of mirror
elseif (kElem.eq.ssdgerm) then
elempith = elemDens(geplace)*thick
call newmac(geplace, muabs)
** print *, 'In ssdgerm,elempith,thick=',elempith,thick
do 300 i = 1, nInt
damping(i) = one - exp(-muabs(i)*elempith)
** print *, i, muabs(i), damping(i)
300 continue
elseif (kElem.eq.fpcp10) then
c P10: Ar 90%, CH4 10%
elemDens(kElem) = (pressure/torr)*stddens*
1 (abzero/(abzero + temper))
elempith = elemDens(kElem)*thick
call newmac(cplace, muabs1)
call newmac(hplace, muabs2)
call newmac(arplace, muabs3)
cargrat = one - argrat
** stoich(1) = 0.2*cargrat*elemWate(cplace)
stoich(1) = cargrat*elemWate(cplace)
```

```
** stoich(2) = 0.8*cargrat*elemWate(hplace)
stoich(2) = 4.*cargrat*elemWate(hplace)
stoich(3) = argrat*elemWate(arplace)
c argrat is the proportional of Ar in P10
** normal = stoich(1) + stoich(2) + stoich(3)
do 400 i = 1, nInt
** muabs(i) = (stoich(1)*muabs1(i) + stoich(2)*muabs2(i)+
** 1 stoich(3)*muabs3(i))/normal
muabs(i) = (stoich(1)*muabs1(i) + stoich(2)*muabs2(i)+
1 stoich(3)*muabs3(i))
damping(i) = one - exp(-muabs(i)*elempith)
400 continue
elseif (kElem.eq.fpcmeth) then
c methane: CH4
elemDens(kElem) = (pressure/torr)*stddens*
1 (abzero/(abzero + temper))
elempith = elemDens(kElem)*thick
call newmac(cplace, muabs1)
call newmac(hplace, muabs2)
stoich(1) = elemWate(cplace)
stoich(2) = 4.*elemWate(hplace)
** normal = stoich(1) + stoich(2)
do 500 i = 1, nInt
** muabs(i) = (stoich(1)*muabs1(i) + stoich(2)*muabs2(i))/
** 1 normal
muabs(i) = stoich(1)*muabs1(i) + stoich(2)*muabs2(i)
damping(i) = one - exp(-muabs(i)*elempith)
500 continue
elseif (kElem.eq.gnplace) then
call linegauss(damping)
elseif (kElem.eq.unplace) then
c this is an "all-pass" filter
do 600 i = 1, nInt
damping(i) = one
600 continue
else
elempith = elemDens(kElem)*thick
* print *, "In source, elemDens ", elemDens
* print *, "In source, elemWate ", elemWate
* print *, "In souce, elemThick ", elemThick
* print *, "In source ",kElem,elemDens(kElem),elempith,thick
* print *, "damping in source = "
```

```
call newmac(kElem, muabs)
do 15 i = 1, nInt
damping(i) = exp(-muabs(i)*elempith)
* print *, i, evArr(i), eArray(i), damping(i), muabs(i)
15 continue
c the default case
endif
return
end
```

10.0.37 SUBROUTINE trapzd(func,a,b,s,n)

Remark 38 *Trapezoidal integration, from Numerical Recipes.*

```
INTEGER n
REAL a,b,s,func
EXTERNAL func
INTEGER it,j
REAL del,sum,tnm,x
if (n.eq.1) then
s=0.5*(b-a)*(func(a)+func(b))
else
it=2**(n-2)
tnm=it
del=(b-a)/tnm
x=a+0.5*del
sum=0.
do 11 j=1,it
sum=sum+func(x)
x=x+del
11 continue
s=0.5*(s+(b-a)*sum/tnm)
endif
return
END
C (C) Copr. 1986-92 Numerical Recipes Software %5'KNp='.
```

Chapter 11
XSPEC

XSPEC is an interactive X-ray spectral-fitting program which has been widely used for X-ray spectroscopic analysis in X-ray astronomy. Its considerable evolution history through a series of X-ray scientific missions is described in a NASA GSFC manual[71]. Its command structure is being augmented and updated at regular intervals. At the core of XSPEC are the following features:

1. A *curve-fitting engine*, which the user may select from the standard XSPEC Levenberg-Marquardt type routines[5] and the most recently released CERN fitting engine. Optimal fitting is determined by minimizing χ^2 or by maximizing the likelihood function, the so-called C statistic, producing best-fit parameters with goodness-of-fit confidence intervals.

2. A built-in *library* of generic and astrophysical spectral models each with their own set of fitting parameters. A single model used in the fit may consist of several of these built-in models, in which case the models are known as *additive* models. Filters and other absorption layers in the optical train may be included as *multiplicative* models, since they multiply into the additive models.

3. A user-selectable *instrumental response matrix* which describes the spectral response function of the detector in the experiment. The rows and columns of this matrix are labeled by X-ray energy values and detector channel numbers respectively.

4. Simultaneous fitting of several data sets with a model whoe parameters can be different for each data set.

5. Support for *FITS* detector data and response matrix format, which may be manipulated with the FTOOLS package, available by anonymous *ftp* from **legacy.gsfc.nasa.gov**.

6. A plotting package (PGPLOT) with device driver support.

7. Freezing (keeping value fixed) and thawing (letting value go free) of fitted parameters within prescribed hard and soft limits.

8. Optional background subtraction, ignoring of faulty channels or spurious data.

Part V

Appendix

11.1 Appendix: Solution to a Difference-differential Equation

We have used as the backbone of our basic fpc model the solution of a system of difference-differential equations describing a birth-and-death process

$$\begin{aligned} p_n'(x) &= -\left[\lambda(1+bn)+n\mu\right]p_n(x)+(n+1)\mu p_{n+1}(x)+ \\ &\quad \lambda[1+b(n-1)]p_{n-1}(x) \end{aligned} \tag{11.1}$$

$$p_0'(x) = -\lambda p_0(x)+\mu p_1(x) \tag{11.2}$$

where λ, $b\lambda$ and μ are all constants. Arley[2](see also Byrne[8]) verified that the negative binomial distribution is a solution by direct substitution, and thought that these equations for general μ, 'cannot be solved elementarily'. On the contrary, we find that the solution to these equations gives us the binomial distribution, the Poisson distribution and the negative binomial distribution (synonymous with the Polya distribution), all by simply letting the quantity b take on negative, zero and positive values respectively. The case for $b=0$ (Poisson case) is solved in Gross and Harris[19]. Here we deduce the general solution using the method of characteristics described in Ince[24]. The standard method of solving these equations is to first get rid of the difference component by taking its Z-transform[28]. Let the Z-Transform of the probability distribution $p_n(t)$ be defined as

$$P(z,x) \equiv \sum_{0}^{\infty} p_n(x)z^n \tag{11.3}$$

Multiply the difference-differential equations by the appropriate powers of z and summing, we get

$$\begin{aligned} &\sum_{n=0}^{\infty} p_n'(x)\,z^n \\ = \;& -\sum_{n=0}^{\infty}\left[\lambda(1+bn)+n\mu\right]p_n(x)z^n + \\ & \mu\sum_{n=0}^{\infty}(n+1)p_{n+1}(x)z^n+\lambda\sum_{n=1}^{\infty}[1+b(n-1)]p_{n-1}(x)z^n \end{aligned} \tag{11.4}$$

This equation may then be rewritten as

$$\begin{aligned} &\sum_{n=0}^{\infty} p_n'(x)\,z^n \\ = \;& -\lambda\sum_{n=0}^{\infty}p_n(x)z^n-(b\lambda+\mu)z\sum_{n=1}^{\infty}np_n(x)z^{n-1}+ \end{aligned}$$

$$\mu \sum_{n=0}^{\infty}(n+1)p_{n+1}(x)z^n + \lambda z \sum_{n=1}^{\infty} p_{n-1}(x)z^{n-1} \tag{11.5}$$
$$+b\lambda z^2 \sum_{n=1}^{\infty} np_n(x)z^{n-1}$$

Since

$$\sum_{n=0}^{\infty} p_n(x)\, z^n = \sum_{n=1}^{\infty} p_{n-1}(x)\, z^{n-1} = P(z,x) \tag{11.6}$$

$$\sum_{n=1}^{\infty} np_n(x)\, z^{n-1} = \sum_{n=0}^{\infty}(n+1)p_{n+1}(x)\, z^n = \frac{\partial P(z,x)}{\partial z} \tag{11.7}$$

and

$$\sum_{n=0}^{\infty} \dot{p}_n(x)\, z^n = \frac{\partial P(z,x)}{\partial x} \tag{11.8}$$

we have

$$\frac{\partial P(z,x)}{\partial x} = (\mu - \lambda bz)(1-z)\frac{\partial P(z,x)}{\partial z} - \lambda(1-z)P(z,x) \tag{11.9}$$

These so-called *Lagrange linear* (or *planar) partial differential equations*[24] may be solved by the method of *characteristics*. The characteristics are:

$$\frac{dx}{1} = \frac{dz}{-(1-z)(\mu - \lambda bz)} = \frac{dP(z,x)}{-\lambda(1-z)P(z,x)} \tag{11.10}$$

This yields by direct integration the two solutions

$$\left(\frac{1-z}{\mu - \lambda bz}\right) e^{-(\mu - \lambda b)x} = C_1 \tag{11.11}$$

and

$$P(z,x)(\mu - \lambda bz)^{\frac{1}{b}} = C_2 \tag{11.12}$$

where C_1 and C_2 are constants. Since this relationship must be true for *all* z and x, there is a functional dependence between the two forms on the left sides of Eqs. (11.11) and (11.12):

$$P(z,t)(\mu - \lambda bz)^{\frac{1}{b}} = g\left[\left(\frac{1-z}{\mu - \lambda bz}\right) e^{-(\mu - \lambda b)x}\right] \tag{11.13}$$

Now when $x = 0$, the system is empty (no avalanche electrons), that is, $p_0(0) = 1$ and $p_n(0) = 0$ for $n > 0$. Thus we have

$$P(z,0) = \sum_{n=0}^{\infty} p_n(0) z^n = 1 \tag{11.14}$$

and, from Eq.(11.13) , that

$$g\left[\left(\frac{1-z}{\mu-\lambda b z}\right)\right] = (\mu - \lambda b z)^{\frac{1}{b}} \tag{11.15}$$

Letting $y = \left(\frac{1-z}{\mu-\lambda b z}\right)$ yields

$$g(y) = \left(\frac{\mu - \lambda b}{1 - \lambda b y}\right)^{\frac{1}{b}} \tag{11.16}$$

Hence

$$\begin{aligned} & g\left[\left(\frac{1-z}{\mu-\lambda b z}\right) e^{-(\mu-\lambda b)x}\right] \\ = \; & (\mu - \lambda b z)^{\frac{1}{b}} \left[\frac{e^{-(\lambda b-\mu)x}(1-\frac{\mu}{\lambda b})}{1 - z + (z - \frac{\mu}{\lambda b}) e^{-(\lambda b-\mu)x}}\right]^{\frac{1}{b}} \end{aligned} \tag{11.17}$$

and

$$P(z,t) = \left[\frac{e^{-(\lambda b-\mu)x}(1-\frac{\mu}{\lambda b})}{1 - \frac{\mu}{\lambda b} e^{-(\lambda b-\mu)x} - z(1 - e^{-(\lambda b-\mu)x})}\right]^{\frac{1}{b}} \tag{11.18}$$

From this generating function, we get the mean

$$\langle n \rangle = - \left.\frac{\partial \ln P(z,x)}{\partial \ln z}\right|_{z=1} = \frac{\lambda(e^{(\lambda b-\mu)x} - 1)}{\lambda b - \mu} \tag{11.19}$$

Substituting this back into Eq. (11.18), we get

$$P(z) = \left[\frac{1}{1 + b \langle n \rangle (1-z)}\right]^{\frac{1}{b}} \tag{11.20}$$

all the x-dependence being contained in the mean $\langle n \rangle$. This is the generating function for three well-known distributions: the binomial distribution ($b < 0$), the Poisson distribution ($b = 0$) and the negative binomial distribution ($b > 0$), which we exhibit below. In all three cases, the variance is

$$\left\langle \Delta n^2 \right\rangle \equiv \left\langle n^2 \right\rangle - \langle n \rangle^2 = \left.\frac{\partial^2 \ln P(z,x)}{\partial \ln z^2}\right|_{z=1} \tag{11.21}$$

$$= \langle n \rangle (1 + b \langle n \rangle) \tag{11.22}$$

11.1.1 Binomial distribution ($b < 0$)

Defining

$$b \equiv -\frac{(1-F)}{\langle n \rangle} = -\frac{1}{n_{\max}} \tag{11.23}$$

where F is defined to be less than unity to keep b negative, we get the Z-Transform

$$P(z) = \left[1 - (1-F)(1-z)\right]^{\frac{\langle n \rangle}{1-F}} \tag{11.24}$$

The variance of this is

$$\left\langle \Delta n^2 \right\rangle = F \langle n \rangle \tag{11.25}$$

from which we identify F to be the Fano factor[16]. If $n_{\max} \equiv \frac{\langle n \rangle}{(1-F)}$ is an integer, we get the classic binomial distribution for n:

$$p_n = \frac{n_{\max}!}{n!\,(n_{\max}-n)!} F^{n_{\max}-n}(1-F)^n \tag{11.26}$$

11.1.2 Case 2. Poisson distribution ($b = 0$)

In the limit $b \to 0$, we get the Z-transform

$$P(z) = \exp\left[\langle n \rangle (z-1)\right] \tag{11.27}$$

with variance

$$\left\langle \Delta n^2 \right\rangle = \langle n \rangle \tag{11.28}$$

The inverse of this Z-transform is the Poisson distribution

$$p_n = \frac{\langle n \rangle^n}{n!} e^{-\langle n \rangle} \tag{11.29}$$

11.1.3 Case 3. Negative binomial distribution ($b > 0$)

Defining

$$b \equiv \frac{1}{h} \tag{11.30}$$

we get the Z-transform

$$P(z) = \left[\frac{1}{1 + \frac{\langle n \rangle}{h}(1-z)}\right]^h \tag{11.31}$$

Expanding $P(z,t)$ in a series of z, we get, by reading off the coefficients, the *negative binomial distribution* often encountered in statistical optics[18]:

$$\begin{aligned} p_n &= \frac{\Gamma(n+h)}{\Gamma(h)\Gamma(n+1)} \frac{\langle n\rangle^n h^h}{(h+\langle n\rangle)^{n+h}} \\ &= \frac{\Gamma(n+h)}{\Gamma(h)\Gamma(n+1)} \left[1+\frac{h}{\langle n\rangle}\right]^{-n} \left[1+\frac{\langle n\rangle}{h}\right]^{-h} \end{aligned} \tag{11.32}$$

where $h \equiv \frac{1}{b}$ is seen to be the parameter representing the *number of degrees* of freedom of the negative binomial distribution. The distribution may also be rewritten in the alternative forms:

$$\begin{aligned} p_n &= \frac{\langle n\rangle^n}{(1+b\,\langle n\rangle)^{n+h}\, h^n} \frac{h(h+1)\cdots(n+h-1)}{n!} \\ &= \frac{\langle n\rangle^n}{(1+b\,\langle n\rangle)^{n+h}} \frac{1(1+b)\cdots(1+(n-1)b)}{n!} \end{aligned} \tag{11.33}$$

and this is the form found in Arley[2]. He had solved the special case of the birth-and-death equations with death rate $\mu = 0$ (pure birth process) by direct substitution of the negative binomial distribution in the difference-differential equations, with mean

$$\langle n\rangle = -\left.\frac{\partial \ln P(z,x)}{\partial \ln z}\right|_{z=1} = \frac{(e^{\lambda b x}-1)}{b} \tag{11.34}$$

The case $b = h = 1$, $\mu = 0$ is a special pure-birth case still of Arley's solution, obtained earlier by Furry:

$$p_n = \frac{\langle n\rangle^n}{(1+\langle n\rangle)^{n+1}} = e^{-\lambda x}\left(1-e^{-\lambda x}\right)^n \tag{11.35}$$

Part VI

References

1. G. D. Alkhazov, Statistics of electron avalanches and ultimate resolution of proportional counters, Nucl. Instr. and Meth. 89 (1970), p. 155.
2. N. Arley, On the Theory of Stochastic Processes and Their Application to the Theory of Cosmic Radiation (Wiley, New York 1948) p. 98.
3. R E. Bellman, R. E. Kalaba and J. A. Lockett, Numerical Inversion of the Laplace Transform: Applications to Biology, Economics, Engineering and Physics, 1966 American Elsevier, New York.
4. E. Oran Brigham, The Fast Fourier Transform and its Applications, 1988 Prentice-Hall, Englewood Cliffs.
5. P. R. Bevington, Data Reduction and Error Analyses for the Physical Sciences, McGraw-Hill.
6. J. J. Bloch, Observations of the soft X-ray diffuse background from 0.07 to 1.0 keV, 1988, Thesis, University of Wisconsin-Madison.
7. W. Blum and L. Rolandi, Particle Detection with Drift Chambers, 1994 Springer-Verlag, Berlin.
8. J. Byrne, Statistics of the electron multiplication process in proportional counters, Proc. R. Soc. Edinburgh, XVI A 33(1962).
9. SAO AXAF Mission Support, Critical Design Review of the HRMA X-ray Detection System(HXDS), SAO-AXAF-DR-94-102, October 1994.
10. W. Diethorn, A methane proportional counter system for natural radiocarbon measurements, USAEC Report NY06628 (1956); also doctoral dissertation, Carnegie Inst. of Technology, 1956.
11. Calibration Group, AXAF Mission Support Team, Smithsonian Astrophysical Observatory, HXDS Error Budget and Performance Prediction, March 5, 1996.
12. Canberra Nuclear Products Group Catalog, Edition 8, Canberra Nuclear Products Group, Meriden, CT.
13. G. Chartas, K. Flanagan, J. P. Hughes, E. M. Kellogg, D. Nguyen, M. Zombeck, M. Joy and J. Kolodziejezak, Correcting X-ray spectra obtained from the AXAF VETA-I mirror calibration for pileup, continuum background and deadtime. 1992 SPIE Proceedings, Vol. 1742, p. 65.
14. L. G. Christophorou, Atomic and Molecular Radiation Physics, 1971 Wiley, London.
15. U. Fano, 1946 Phys. Rev. 70, 44; Ionization yield of radiations. II. The fluctuations of the number of ions, 1942 Phys. Rev. 72, 26-29.
16. G. W. Fraser, X-Ray Detectors in Astronomy, Cambridge University Press, Cambridge, 1989, p.48.
17. R. Jenkins, R. W. Gould and D. Gedcke, Quantitative X-ray Spectrometry, 1981 Marcel Dekker, New York.
18. J. Goodman, Statistical Optics, Wiley, New York.
19. D. Gross, C. M. Harris, Fundamentals of Queueing Theory (Wiley, New York 1985).
20. T. E. Harris, The Theory of Branching Processes (Springer-Verlag, Berlin 1963).

21. R. W. Hendricks, The gas amplification factor in xenon-filled proportional counters, Nucl. Instr. Meth. 102, 309(1972).
22. B. L. Henke, E. M. Gullikson and J. C. Davis, X-Ray Ineractions: Photoabsorption, SCattering, Transmission and Reflection $E = 50 - 30,000\, eV$, $Z = 1 - 92$, Lawrence Berkeley Laboratory Materials Sciences Division preprint, LBL-33908, March 1993.
23. IDL, an interactive data language from RSI, 777 29th Street, Suite 302, Boulder, CO 80303.
24. E. L. Ince, Ordinary Differential Equations, 1956 Dover Reprint, New York.
25. H. Inoue, K. Koyama, M. Matsuoka, T. Ohashi, Y. Tanaka, Properties of gas scintillation proportional counters for soft X-rays, Nucl. Instr. and Meth. 157 (1978), 295.
26. K. Jahoda and D. McCammon, Proportional counters as low energy photon detectors, Nucl. Instr.Meth., 1988, A272, 800.
27. K. Kleinknecht, Detectors for Particle Radiation, 1986 Cambridge University Press, Cambridge.
28. L. Kleinrock, Queueing Systems, Vol. I, 1975 Wiley, New York.
29. G. F. Knoll, Radiation Detection and Measurement, 1989 Wiley, New York.
30. P. Kokotovic and D. D. Siljak, Automatic analog solution of algebraic equations and plotting of root loci by generalized Mitrovic method, 1964 IEEE Trans. Appl. Ind., 83, 324-328.
31. H. H. Kramers, On the theory of X-ray absorption and of the continuous X-ray spectrum, Phil. Mag. 46, 836 (1923).
32. W. H.-M. Ku and Robert Novick, Gas scintillation proportional counters and other low-energy X-ray spectrophotometers, in Low Energy X-ray Diagnostics — 1981, 1981 American Institute of Physics, New York, 78 - 84.
33. Loeb, Gaseous electronics, Dover
34. G. Nicolis and I. Progogine, Self-organization in nonequilibrium systems (Wiley, NY, 1977).
35. A. V. Oppenheim and R. W. Schafer, Digital Signal Processing (Prentice-Hall, Englewood Cliffs 1975), p. 206.
36. S. D. Personick, Statistics of a general class of avalanche detectors with applications to optical communication, 1971 BSTJ, 50, 10, 3075-3095.
37. Courtesy Diab Jerius and Richard Edgar, SAO.
38. J. R. Prescott, Photomultiplier single-electron statistics and the shape of the ideal scintillation line, Nucl. Instr. and Meth., 122, 1963, p. 256.
39. H. Raether, Electron avalanches and breakdown in gases, 1964 Butterworth, London.
40. L. Reimer, Scanning Electron Microscopy (Springer-Verlag, Berlin, 1985, p. 204..
41. P. Rice-Evans, Spark, Streamer, Proportional and Drift Chambers, 1974, Richelieu Press, London., p. 77 and p. 256.
42. M. E. Rose and S. A. Korff, Phys. Rev. 59, 850(1941).

43. B. B. Rossi and H. H. Staub, Ionization Chambers and Counters: Experimental Techniques, 1949 McGraw-Hill, New York.

44. B. A. Saleh and M. C. Teich, Fundamentals of Photonics, 1991 Wiley, New York, p.649.

45. E. Tsiang, Modeling the AXAF proportional counters for ground calibration, June 12 1996.

46. E. Tsiang, Modeling the AXAF Proportional Counters for Ground Calibration of the X-Ray Detector System II: escape peaks, incomplete charge collection and pileup effects.

47. E. Tsiang, Synchrotron radiation flux through a circular aperture and the determination of the absolute quantum efficiency, SAO Memo February 2, 1996.

48. E. Tsiang, Dead time and pileup, SAO Memo January 4, 1996.

49. E. Tsiang, Special adaptation of the JM model to analyze HIREFS data, SAO Memo January 5, 1996.

50. E. Tsiang, A Proposed New Spectral Fitting Program for HXDS, SAO Memo May 25, 1995.

51. E. Tsiang, A new closed-form expression for the Jahoda-McCammon distribution without diffusion losses, SAO Memo July 20, 1995.

52. E. Tsiang, Numerical inversion of the Jahoda-McCammon moment generating function without diffusion losses, SAO Memo July 27, 1995.

53. E. Tsiang, Numerical inversion of the Prescott moment generating function and comparison with Prescott function, Fortran Inversion Routine of JM mgf for XSPEC, SAO Memo August 3, 1995.

54. E. Tsiang, XSPEC, SAO Memo August 10, 1995.

55. E. Tsiang, Detector response determination using XSPEC, August 17, 1995.

56. E. Tsiang, Line fitting using the JM model in XSPEC, the JM-Kramer continuum model, SAO Memo August 24, 1995.

57. E. Tsiang, .Escape corrections, line and continuum, SAO Memo September 5, 1995.

58. E. Tsiang, SSD pdfs — Hypermet functions and alternatives, SAO Memo October 5, 1995.

59. E. Tsiang, FPC data reduction — first results using XSPEC, SAO Memo October 19, 1995.

60. E. Tsiang, Preliminary analysis of FPC and SSD data using XSPEC, SAO Memo November 2, 1995.

61. E. Tsiang, More preliminary analysis of FPC data using XSPEC, SAO Memo Novembr 15, 1995.

62. E. Tsiang, A modification to the JM and Prescott probability density functions, SAO December 21, 1995.

63. E. Tsiang, The physics behind the Polya Distribution in the JM model, SAO Memo December 28, 1995.

64. E. Tsiang, The physics behind the Polya Distribution in the JM model, SAO Memo January 29, 1996.
65. E. Tsiang, Spectral Fitting Program for FPC–Modification of existing program, Rev. 0.3, SAO Memo June 12, 1996.
66. B. Wargelin, Primary HRMA Calibration Objectives at the X-ray Calibration Facility (XRCF), SPIE poster session, 1996.
67. B. Wargelin, private communication.
68. B. Wargelin, SAO Internal Memo, "So what's new with the FPC", March 1, 1996.
69. M. C. Weisskopf, Advanced Astrophysics Facility AXAF: an overview (Proc. SPIE 2515, 1995), p. 312.
70. E. M. Williams, The Physics and Technology of Xerographic Processes, 1984 Wiley, New York.
71. NASA Goddard Space Flight Center, XSPEC, an X-ray spectral fitting package, user's guide.
72. P. Zhao, E. M. Kellogg, D. A. Schwartz and Y. Shao, Intensity distribution of the X-ray source for the AXAF VETA-I mirror test, 1992 SPIE Vol. 1742, 26 - 39.