Memorandum

To:	L3 Distribution
From:	F. Primini
Subject:	Revised Specifications for Computing Aperture Photometry Quantities
Date:	June 6, 2007

Introduction

This is a revised specification for computing the aperture photometry quantities , including errors, described in section 1.2.2.6.3 of the Chandra Source Catalog Requirements, v. 0.5. The basic paradigm remains the same as that described in the original aperture flux specification (https://icxc.harvard.edu/soft/schedule/L3/Sci_Docs/significance.ps), but I have generalized the formalism to deal explicitly with rates, photon fluxes, and energy fluxes, and I have recast the mathematics slightly to conform with the approach used by Vinay Kashyap for computing the Bayesian Posterior Probability Distribution for the source quantities. A memo describing the mathematical details of Vinay's approach accompanies this memo, and sample code for calculating the probability distribution is included as an appendix here. This probability distribution is used in the error calculations.

Deleted Quantities

In the context of the current algorithm for computing fluxes, it does not make sense to compute net source counts, rate, or flux, corrected for psf fraction, separately in source and background apertures. Such quantities are identified in section 1.2.2.6.3 by extension **aperbkg** (and **aperbkg_err** for errors). All such quantities should be deleted from the requirements.

Definitions

n	Counts in source aperture
m	Counts in background aperture
A_s	Number of pixels in source aperture
A_B	Number of pixels in background aperture
α	Fraction of PSF in source aperture
β	Fraction of PSF in background aperture
T_s	Exposure time in source aperture (sec)
T_B	Exposure time in background aperture (sec). For generality, this is defined independently from T_s , although it usually has the same value.
E_s	Average exposure map value in source aperture (cm^2-sec)
E_B	Average exposure map value in background aperture (cm^2-sec)
$\bar{\epsilon}_s$	Average photon energy in source aperture ($\bar{\epsilon}_s=\sum_{i=1}^{n}\epsilon_i/n$)
$\bar{\epsilon}_B$	Average photon energy in background aperture ($\bar{\epsilon}_B=\sum_{i=1}^{m}\epsilon_i/m$)

Basic Formalism

It is assumed that the source and background apertures are distinct (although they are not necessarily simple shapes) so that n and m are statistically independent. The general approach is to express n and m as sums of contributions from source and background quantities, yielding two simultaneous linear equations which can be solved for the source quantity. For reasons which I hope will become clear shortly, I chose to write these as follows:

$$\begin{aligned} n &= fs + cb = fs + b' \\ m &= gs + rcb = gs + rb' \end{aligned}$$

Here, the intermediate quantities c and b are introduced for consistency in describing source and background units, but do not enter into the final calculations.

Again, the equations for n and m are cast in this form so that we can easily take advantage of Vinay's formalism for computing probability distributions. The actual meanings of the parameters $f, g, c,$ and r will depend on the particular source quantity we're interested in, e.g., net counts, rate, etc.

The quantity of interest is s , and its Gaussian error, σ_s are given, in general, by

$$s = \frac{rn - m}{rf - g}, \qquad \sigma_s^2 = \frac{r^2 n + m}{(rf - g)^2} .$$

The determination of s and its error for the different source quantities net counts, net rate, net photon flux, and net energy flux, then reduces to the proper determination of the parameters f, g and r . These are described below.

Net Counts (Requirements Section 1.2.2.6.3.1.2)

Here, the units of s and b are simply $counts$ and $counts - pix^{-2}$, and $f, g, c,$ and r can be written in terms of our earlier definitions as

$$\begin{aligned} f &= \alpha \\ g &= \beta \\ c &= A_s \\ r &= \frac{A_B}{A_s} \end{aligned} .$$

Net Rate (Requirements Section 1.2.2.6.3.3)

The units of s and b are now $counts - sec^{-1}$ and $counts - sec^{-1} - pix^{-2}$, and $f, g, c,$ and r can be written in terms of our earlier definitions as

$$\begin{aligned} f &= \alpha T_s \\ g &= \beta T_B \\ c &= A_s T_s \\ r &= \frac{A_B T_B}{A_s T_s} \end{aligned} .$$

Net Photon Flux (Requirements Section 1.2.2.6.3.4)

The units of s and b are now $photons - cm^{-2} - sec^{-1}$ and $photons - cm^{-2} - sec^{-1} - pix^{-2}$, and

f , g , c , and r can be written in terms of our earlier definitions as

$$\begin{aligned} f &= \alpha E_s \\ g &= \beta E_B \\ c &= A_s E_s \quad . \\ r &= \frac{A_B E_B}{A_s E_s} \end{aligned}$$

Net Energy Flux (Requirements Section 1.2.2.6.3.4)

The units of s and b are now $ergs-cm^{-2}-sec^{-1}$ and $ergs-cm^{-2}-sec^{-1}-pix^{-2}$, and f , g , c , and r can be written in terms of our earlier definitions as

$$\begin{aligned} f &= \alpha E_s / \bar{\epsilon}_s \\ g &= \beta E_B / \bar{\epsilon}_B \\ c &= A_s E_s / \bar{\epsilon}_s \quad . \\ r &= \frac{A_B E_B \bar{\epsilon}_s}{A_s E_s \bar{\epsilon}_B} \end{aligned}$$

In this case, since $\bar{\epsilon}_s$ and $\bar{\epsilon}_B$ appear in the denominators of some quantities, we need to be able to handle the exception when they are 0. In those cases, we should set them equal to the mid-point energy of the energy band in use.

Computation of Errors

In all cases, once f , g , and r are known, the maximum likelihood estimator for s can be computed from

$$s_{MLE} = \frac{rn-m}{rf-g} \quad .$$

The confidence region on s is computed from the posterior probability distribution for s . The computation of this distribution is described in the accompanying memo, “Background Marginalized X-Ray Source Intensity”, by V. Kashyap. The distribution we're interested in is described in eq. 26. Sample s-lang code for computing this distribution given n , m , f , g , and r is provided in the appendix. The basic routine is s_pdf, whose arguments are n,m,r,f, and g, (note n=C and m=B in the documentation). The distribution is provided as an array with a binsize of 0.1 in s in the code.

To determine the energy bounds, we should find the mode of the distribution (which should be close to s_{MLE}) and sum probabilities in alternating bins on either side of the mode until we achive the desired confidence level. The values of s in the final bins determine the error bounds. If $s=0$ is reached before the confidence level is achived, the integration should continue only on the positive side of the mode, and the flux should be flagged as an upper limit.

Appendix – S-Lang Code for Computing Posterior Probability Distribution

```
#! /usr/bin/env slsh
_auto_declare =1;
_traceback=1;

()=evalfile("/Users/fap/bin/slang.sl");
require("gsl");

% For now, get variables from command line;

C = integer(__argv[1]);
B = integer(__argv[2]);
r = atof(__argv[3]);
f = atof(__argv[4]);
g = atof(__argv[5]);

define s_pdf(C,B,r,f,g){

  % Compute the posterior probability distribution for source counts or intensity     ;
  % marginalized over background, assuming non-informative gamma priors (alpha=1, beta-0) ;

  variable i,j,k;
  variable C1 = C+1;
  variable B1 = B+1;
  variable CB1= C+B+1;

  variable rf_minus_g = r*f-g;
  variable one_plus_r = 1.0+r;

  variable K = Double_Type[C1,B1];
  variable S = Double_Type[C1,B1];

  % Compute MLE value for s, to estimate reasonable range for pdf ;

  variable S_mle = (r*C-B)/rf_minus_g;
  if(S_mle<=0.0) S_mle=30;

  variable s = [0.0:3*S_mle:0.1];
  variable ps= Double_Type[length(s)];

  for(k=0;k<=C;,k++){
    for(j=0;j<=B;j++){

      K[k,j]=rf_minus_g*f^k*g^j*r^(B-j)*gamma(CB1-k-j);
      K[k,j]/=(gamma(k+1)*gamma(C1-k)*gamma(j+1)*gamma(B1-j)*one_plus_r^(CB1-k-j));

    }
  }

  for(i=0;i<length(s);i++){

    for(k=0;k<=C;,k++){
      for(j=0;j<=B;j++){

          S[k,j]=s[i]^(k+j)*exp(-1.0*s[i]*(f+g));

      }
    }

    ps[i] = sum(_reshape(S*K,[(C+1)*(B+1)]));

  }

  % Renormalize psf just in case;

  return s,ps/(0.1*sum(ps));
% return s,ps;

} % end of s_pdf;

define s_gaussian(C,B,r,f,g){
  % Compute the classical Gaussian probability distribution for source counts or intensity     ;

  variable i,j,k;
  variable C1 = C+1;
  variable B1 = B+1;
  variable CB1= C+B+1;

  variable rf_minus_g = r*f-g;
  variable one_plus_r = 1.0+r;

  variable K = Double_Type[C1,B1];
  variable S = Double_Type[C1,B1];

  % Compute MLE value for s, to estimate reasonable range for pdf ;

  variable S_mle = (r*C-B)/rf_minus_g;
  variable sigma_s = sqrt((r*r*C+B)/(rf_minus_g*rf_minus_g));

  variable S_tmp = S_mle;
  if(S_tmp<=0.0) S_tmp=30;

  variable s = [0.0:3*S_tmp:0.1];
  variable ps= Double_Type[length(s)];

  ps = (1.0/(sqrt(2.0*PI)*sigma_s))*exp(-1.0*(s-S_mle)*(s-S_mle)/(2.0*sigma_s*sigma_s));

  return s,ps;

} % end of s_gaussian;

(s,ps) = s_pdf(C,B,r,f,g);

mode_s_pdf = s[where(ps==max(ps))];
()=printf("Posterior Probability Distribution Mode:\t%f\n",mode_s_pdf[0]);
norm_s_pdf = sum(ps)*0.1;
()=printf("Posterior Probability Distribution Normalization:\t%f\n",norm_s_pdf);
```

```
fp=fopen("S_Posterior_PDF","w+");

for(i=0;i<length(s);i++) ()=fprintf(fp,"%f\t%f\n",s[i],ps[i]);

()=fclose(fp);

(s,ps) = s_gaussian(C,B,r,f,g);

mode_s_gaussian = s[where(ps==max(ps))];
()=printf("Gaussian Probability Distribution Mode:\t%f\n",mode_s_gaussian[0]);
norm_s_gaussian = sum(ps)*0.1;
()=printf("Gaussian Probability Distribution Normalization:\t%f\n",norm_s_gaussian);
S_mle = (r*C-B)/(r*f-g);
()=printf("Max. Likelihood Estimate:\t%f\n",S_mle);

fp=fopen("S_Gaussian_PDF","w+");

for(i=0;i<length(s);i++) ()=fprintf(fp,"%f\t%f\n",s[i],ps[i]);

()=fclose(fp);
```