Appendix A

Extinction and reddening

A.1 Extinction and reddening for NGC 6204 and Hogg 22

NGC 6204

$(B-V)-(B_0)-(V_0)=0.430$ (Obtained from WEBDA)
$\therefore (B-B_0)-(V-V_0)=0.430$
but $(B-B_0)=A_b$ and $(V-V_0)=A_v$ (Extinction in B and V respectively)
$\therefore A_B-A_v=0.430$ (Reddening)
but $A_b/A_v=1.324$ (From (Rieke & Lebofsky, 1985))
$\therefore 1.324A_v-A_v=0.324A_v=0.430$
$\therefore A_v=0.430/0.324$
$\therefore A_v=1.32$

Hogg 22

$(B-V)-(B_0)-(V_0)=0.647$ (Obtained from WEBDA)
$\therefore (B-B_0)-(V-V_0)=0.647$
but $(B-B_0)=A_b$ and $(V-V_0)=A_v$ (Extinction in B and V respectively)
$\therefore A_B-A_v=0.647$ (Reddening)
but $A_b/A_v=1.324$ (From (Rieke & Lebofsky, 1985))
$\therefore 1.324A_v-A_v=0.324A_v=0.647$
$\therefore A_v=0.647/0.324$
$\therefore A_v=1.99$

Appendix B

Computer code

The following part of the computer code is responsible for calculating image rotation and translation after which the coordinate transformation from image coordinates to the master list coordinates is done.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char **argv)
{
  if (argc != 2)
  {
    printf("usage: %s filename \n", argv[0]);
    }

//#########################################################################
  FILE *fp1;                                                  /*variables*/
  FILE *fp2;
  FILE *fp3;
  FILE *fp4;
  FILE *fp5;
  FILE *fp6;
  FILE *fp7;
  FILE *fp8;
  float  b1[5000], b2[5000], b3[5000], b4[5000], b5[5000], b6[5000], b7[5000];
  float  v1[5000], v2[5000], v3[5000], v4[5000], v5[5000], v6[5000], v7[5000];
  float  i1[5000], i2[5000], i3[5000], i4[5000], i5[5000], i6[5000], i7[5000];
  float ximage[20000], yimage[20000], alphaRad[5000], alphaDeg[500], pixdistA[500],
        pixdistB[500], coords[46][6], coordsA[1000][6], coordsB[1000][6];
  float a1[200], a2[200], a3[200], a4[200], a5[200], a6[200], x1[200], x2[200],
        x3[200], c1[200], c2[200], c3[200], e1[200], e2[200], e3[200];
```

```
  float d1, d2, d3, d4, d5, d6, d7, x0, y0, t[200], t1[200], t2[200], f1, f2, f3,
        f4, f5, f6;
  int i,j,k,l,y,g,q, nlines1, nlines2, nlines3;

//#########################################################################

 if((fp1=fopen("master.fit.coo.1", "r")) == NULL) {
    printf("Cannot open file.\n");
           exit(1);
      }

if((fp2=fopen(argv[1], "r")) == NULL) {
    printf("Cannot open file.\n", argv[1]);
           exit(1);
      }

  if((fp3=fopen("masterbright.coords", "w+")) == NULL) {
      printf("Cannot open file.\n");
           exit(1);
 }

 if((fp4=fopen("image.coords", "w+")) == NULL) {
      printf("Cannot open file.\n");
           exit(1);
 }
  if((fp5=fopen("out1", "w+")) == NULL) {
        printf("Cannot open file.\n");
                   exit(1);

 }

 if((fp6=fopen("out2", "w+")) == NULL) {
         printf("Cannot open file.\n");
                             exit(1);

                                }

 if((fp7=fopen("out.coords", "w+")) == NULL) {
         printf("Cannot open file.\n");
                             exit(1);

                                }

if((fp8=fopen("master.coords", "r")) == NULL) {
         printf("Cannot open file.\n");
                                      exit(1);

                                }
```

```
//#########################################################################
 char line[256], *tmpchar;                          /*read number of lines*/
 nlines1 = 0;
 while (fgets(line, sizeof(line), fp1) != NULL)
   {
    tmpchar = &line[0];
     while((!feof(fp1)) && (*tmpchar == ' ') && (*tmpchar != '\0'))
        {
         tmpchar++;
         fscanf (fp1, "%f%f%f%f%f%f%f", &d1, &d2, &d3, &d4, &d5, &d6, &d7);
          nlines1 = nlines1 + 1;
        }
    }
nlines1 = nlines1 - 1;
printf("%d \n", nlines1);

nlines2 = 0;
 while (fgets(line, sizeof(line), fp2) != NULL)
    {
     tmpchar = &line[0];
      while((!feof(fp2)) && (*tmpchar == ' ') && (*tmpchar != '\0'))
        {
         tmpchar++;
         fscanf (fp2, "%f%f%f%f%f%f%f", &d1, &d2, &d3, &d4, &d5, &d6, &d7);
          nlines2 = nlines2 + 1;
         }
     }
nlines2 = nlines2 - 1;
printf("%d \n", nlines2);

nlines3 = 0;
 while (fgets(line, sizeof(line), fp8) != NULL)
     {
          tmpchar = &line[0];
        while((!feof(fp8)))
         {
        tmpchar++;
         fscanf (fp8, "%f%f", &d1, &d2);
          nlines3 = nlines3 + 1;
         }
     }

//#########################################################################
  rewind(fp1);                              /*read from iraf coords file*/
  rewind(fp2);

  i=1;
```

```
  while (fgets(line, sizeof(line), fp1) != NULL)
     {
         tmpchar = &line[0];
  while (((*tmpchar == ' ')) && (*tmpchar != '\0'))
         tmpchar++;
         i++;
         if ((*tmpchar == '#') || (*tmpchar == '\0') || (*tmpchar == '\n'))
          continue;
      sscanf(tmpchar, "%f %f", &b1[i], &b2[i]);
     // fprintf(stdout, "%f \n" ,b1[i]);
     // fprintf(stdout, "%f \t %f \n", b1[i], b2[i]);
     fprintf(fp3, "%f \t %f \n",b1[i], b2[i]);
     }

j=1;
  while (fgets(line, sizeof(line), fp2) != NULL)
     {
          tmpchar = &line[0];
   while (((*tmpchar == ' ')) && (*tmpchar != '\0'))
         tmpchar++;
         j++;
        if ((*tmpchar == '#') || (*tmpchar == '\0') || (*tmpchar == '\n'))
        continue;
    sscanf(tmpchar, "%f %f", &b3[j], &b4[j]);
   // fprintf(stdout, "%f \n" ,b1[i]);
   // fprintf(stdout, "%f \t %f \n", b1[i], b2[i]);
   fprintf(fp4, "%f \t %f \n", b3[j], b4[j]);
     }

//#########################################################################
rewind(fp3);                /*read from output coords files into v and i*/
rewind(fp4);

for(i=0;i<nlines1;i++){
     fscanf(fp3, "%f \t %f \n", &v1[i], &v2[i]);     // Scan masterbright.coords
     }

for(i=0;i<nlines2;i++){
    fscanf(fp4, "%f \t %f \n", &i1[i], &i2[i]);             // Scan image.coords
     }

//#########################################################################
   for(i=0;i<nlines1;i++){        /*cross identify stars in two images with distances*/
    for(j=i+1;j<nlines1-1;j++){


/*distances between two stars in first frame*/
     pixdistA[i]=sqrt(pow((v1[i]-v1[j]),2)+pow((v2[i]-v2[j]),2));
```

```
     fprintf(fp5, "%i %f %f %f %f \n", i, v1[i],v2[i],v1[j],v2[j], pixdistA[i]);
//   fprintf(stdout, "%i %f %f %f %f \t %f %f %f %f \n", i, v1[i],v2[i],
//        v1[j],v2[j], pixdistA[i], i1[i], i2[i], i1[j], i2[j], pixdistB[i]);


   }
   }
   printf("\n");

  for(i=0;i<nlines2;i++){
    for(j=i+1;j<nlines2-1;j++){

/*distances between two stars in second frame*/
   pixdistB[i]=sqrt(pow((i1[i]-i1[j]),2)+pow((i2[i]-i2[j]),2));
     fprintf(fp6, "%i %f %f %f %f %f \n", i, i1[i],i2[i],i1[j],i2[j], pixdistB[i]);

      }
        }
     printf("\n");

rewind(fp5);
rewind(fp6);

for (i=0;i<100;i++)
  {

fscanf(fp5, "%f %f %f %f %f %f \n", &coordsA[i][0], &coordsA[i][1], &coordsA[i][2],
                                    &coordsA[i][3], &coordsA[i][4], &coordsA[i][5]);
  }

                                                        /*Print unsorted arrays*/
/*for (k=0;k<nlines1-1;k++)
   {
   for (l=0;l<6;l++)
   {
    printf("%8f ", coordsA[k][l]);
   }
  printf("\n");
  }
  printf("\n\n");
*/

for (i=0;i<100;i++)
  {

  fscanf(fp6, "%f %f %f %f %f %f \n", &coordsB[i][0], &coordsB[i][1], &coordsB[i][2],
                                      &coordsB[i][3], &coordsB[i][4], &coordsB[i][5]);
```

```
    }
/*    for (k=0;k<nlines2-1;k++)
      {
  for (l=0;l<6;l++)
   {
    printf("%8f ", coordsB[k][l]);
   }
      printf("\n");
    }
 printf("\n\n");
*/

//################ Initialize coords arrays ################################

for(i=0;i<900;i++)                  /*Sort coordinate arrays by pixel distances*/
 {
    for(j=0; j<900-1-i; j++)
  {
     if(coordsA[j+1][5] > coordsA[j][5])
      {
        f1 = coordsA[j][5];
        coordsA[j][5] = coordsA[j+1][5];
        coordsA[j+1][5] = f1;
        f2 = coordsA[j][4];
        coordsA[j][4] = coordsA[j+1][4];
        coordsA[j+1][4] = f2;
        f3 = coordsA[j][3];
        coordsA[j][3] = coordsA[j+1][3];
        coordsA[j+1][3] = f3;
        f4 = coordsA[j][2];
        coordsA[j][2] = coordsA[j+1][2];
        coordsA[j+1][2] = f4;
        f5 = coordsA[j][1];
        coordsA[j][1] = coordsA[j+1][1];
        coordsA[j+1][1] = f5;
        f6 = coordsA[j][0];
        coordsA[j][0] = coordsA[j+1][0];
        coordsA[j+1][0] = f6;
      }
   }
}

for(i=0;i<900-1;i++)
 {
    for(j=0; j<900-1-i; j++)
  {
     if(coordsB[j+1][5] > coordsB[j][5])
      {
```

```
        f1 = coordsB[j][5];
        coordsB[j][5] = coordsB[j+1][5];
        coordsB[j+1][5] = f1;
        f2 = coordsB[j][4];
        coordsB[j][4] = coordsB[j+1][4];
        coordsB[j+1][4] = f2;
        f3 = coordsB[j][3];
        coordsB[j][3] = coordsB[j+1][3];
        coordsB[j+1][3] = f3;
        f4 = coordsB[j][2];
        coordsB[j][2] = coordsB[j+1][2];
        coordsB[j+1][2] = f4;
        f5 = coordsB[j][1];
        coordsB[j][1] = coordsB[j+1][1];
        coordsB[j+1][1] = f5;
        f6 = coordsB[j][0];
        coordsB[j][0] = coordsB[j+1][0];
        coordsB[j+1][0] = f6;
      }
   }
}

                                        /*Print coordinate arrays */
for (k=0;k<100;k++)
  {
    for(l=0;l<6;l++)
    {
      printf("%8f ", coordsA[k][l]);
     }
      printf("\n");
   }

  printf("\n\n");

for (k=0;k<100;k++)
  {
    for(l=0;l<6;l++)
    {
      printf("%8f ", coordsB[k][l]);
     }
      printf("\n");
   }
  printf("\n\n");


for(i=0;i<800;i++){       /*Assign new sorted values to coordsA & coordsB arrays */
for(j=0;j<800;j++){
 v1[i+i]=coordsA[i][1];
```

```
 v2[i+i]=coordsA[i][2];
 v1[2*j+1]=coordsA[j][3];
 v2[2*j+1]=coordsA[j][4];
}
 }

for(i=0;i<800;i++){
for(j=0;j<800;j++){
 i1[i+i]=coordsB[i][1];
 i2[i+i]=coordsB[i][2];
 i1[2*j+1]=coordsB[j][3];
 i2[2*j+1]=coordsB[j][4];
    }
  }

printf("\n");
printf("\n\n");


for(g=0;g<800;g++){
for(i=0;i<900;i++){
//fprintf(stdout, "%i \n", g);
 if ((coordsB[i][5] > coordsA[g][5] - 2.0)&&(coordsB[i][5] < coordsA[g][5] + 2.0)){
  if (((coordsA[g+1][5]<coordsA[g][5]-3)&&(coordsA[g-1][5]>coordsA[g][5]+3))&&
     ((coordsB[i+1][5]<coordsB[i][5]-3)&&(coordsB[i-1][5]>coordsB[i][5]+3))){

 fprintf(stdout, "%f \n",coordsA[g][5]);
 printf("\n\n");
 fprintf(stdout, "%f \n",coordsB[i][5]);
 q=5;

v1[0]=coordsA[g][1];
v2[0]=coordsA[g][2];
v1[1]=coordsA[g][3];
v2[1]=coordsA[g][4];
i1[0]=coordsB[i][1];
i2[0]=coordsB[i][2];
i1[1]=coordsB[i][3];
i2[1]=coordsB[i][4];
break;
}
}
else {
//printf("not found \n");
continue;
}
}
if (q==5)
```

```
break;
}


printf("\n\n");
printf("\n\n");

//#########################################################################

a1[0]=v1[0]-v1[1];             /*a1-a6 : simplify x0 y0 elliminated equation*/
a2[0]=i1[0]+(-1)*i1[1];
a3[0]=(-1)*i2[0]+i2[1];
a4[0]=v2[0]-v2[1];
a5[0]=i1[0]+(-1)*i1[1];
a6[0]=i2[0]+(-1)*i2[1];

x1[0]=a1[0]+a4[0];
x2[0]=a2[0]+a6[0];
x3[0]=a5[0]+a3[0];

c1[0]=x1[0]/x1[0];
c2[0]=x2[0]/x1[0];
c3[0]=x3[0]/x1[0];

e1[0]=c1[0]+c2[0];
e2[0]=(-1)*2*c3[0];
e3[0]=c1[0]-c2[0];

t1[0]=(((-1)*e2[0])-sqrtf((e2[0]*e2[0])-(4*e1[0]*e3[0])))/(2*e1[0]);
t2[0]=(((-1)*e2[0])+sqrtf((e2[0]*e2[0])-(4*e1[0]*e3[0])))/(2*e1[0]);



/*Choose between two roots of quadratic eguation*/
if ( fabsf(0 - t1[0]) < fabsf(0 - t2[0]) )
   t[0] = t1[0];
    else
    {
        t[0] = t2[0];
            }


                alphaRad[0] = atan(t[0])*2;
                alphaDeg[0] = alphaRad[0] * (180/3.141592654);


 fprintf(stdout, "%f %f %f %f %f %f %f %f \t %f \t %f %f %f \n", v1[0], v2[0],
       v1[1], v2[1], i1[0], i2[0], i1[1], i2[1], t[0], t1[0],t2[0], alphaDeg[0]);
```

```
/*Translation in x and y direction*/
   x0=v1[1]+i2[1]*sin(alphaRad[0])-i1[1]*cos(alphaRad[0]);
   y0=v2[1]-i2[1]*cos(alphaRad[0])-i1[1]*sin(alphaRad[0]);

printf("\n\n");

      fprintf(stdout, "%f %f %f \n", x0, y0, alphaDeg[0]);



rewind(fp7);                    /*read from output coords files into b1 & b2*/
rewind(fp8);

     for(i=0;i<nlines3;i++){
          fscanf(fp8, "%f \t %f \n", &b1[i], &b2[i]);



/*Transformation from master to image coordinates*/
         ximage[i]=(b1[i]-x0)*cos(alphaRad[0])+(b2[i]-y0)*sin(alphaRad[0]);
         yimage[i]=(b2[i]-y0)*cos(alphaRad[0])-(b1[i]-x0)*sin(alphaRad[0]);

           fprintf(fp7, "%f %f \n", ximage[i], yimage[i]);



            }
         printf("%d \n", nlines3);


  fclose(fp1);
  fclose(fp2);
  fclose(fp3);
  fclose(fp4);
  fclose(fp5);
  fclose(fp6);
  fclose(fp7);
  return 0;
}
```

The following part of the computer code is the IRAF photometry script, calling the above coordinate transformation code and doing both PSF photometry and aperture photometry.

```
 procedure autophot(iraflist)
string  iraflist  {"list",prompt="list of images to reduce"}
string  simages   {"list.allnn",prompt="list of images in which PSFs found"}
real    scale     {1.0,prompt="Image scale in units per pixel"}
real    fwhm      {3.5,min=1,max=8,prompt="full width at half maximum"}
real    threshold {4.0,min=1,prompt="threshold in sigma for feature detection"}
real    thresh2   {6.0,min=1,prompt="second iteration threshold in sigma for feature detection"}
real    nsigma    {15.,min=1,prompt="number of sigma below mean for datamin"}
real    datamax   {45000.,prompt="Maximum good data value"}
real    aperture  {4.,prompt="aperture radius in scale units"}
real    sannulus  {10.,prompt="inner radius of sky annulus in scale units"}
real    wannulus  {10.,prompt="width of sky annulus in scale units"}
string  salgo     {"mode",prompt="Sky fitting algorithm"}
real    psfrad    {14.,min=1,prompt="radius of psf in scale units"}
real    nspsfrad  {11.,min=1,prompt="radius of psf for nstar in scale units"}
real    fitrad    {4.,prompt="fit radius in scale units"}
int     psfiter   {2,min=1,max=2,prompt="number of PSF iterations (1 | 2)"}
int     allsiter  {2,min=1,max=2,prompt="number of allstar iterations (1 | 2)"}
real    gain      {0.75,prompt="gain of CCD in electrons per counts"}
real    readnoise {6.5,prompt="CCD readout noise in electrons"}
int     maxnopsf  {20,min=1,prompt="Maximum number of PSF star to select automatically"}
int     nclean    {2,prompt="Number of cleaning iterations for computing psf"}
real    mergerad  {INDEF,prompt="Critical object merging radius in scale units"}
string  errlog    {"bug_log",prompt="log of problems file"}
string  exposurek {"exptime",prompt="Exposure time header keyword"}
string  airmassk  {"airmass",prompt="Airmass header keyword"}
string  filterk   {"filters",prompt="Filter header keyword"}
string  timek     {"ut",prompt="Time of observation header keyword"}
string  datek     {"date-obs",prompt="Date of observation header keyword"}
bool    del_sub   {no,prompt="Delete *.sub.# images after reduction"}
bool    im_disp   {no,prompt="Display subtracted image after processing"}
struct *lis
struct *lis1
struct *pstlis
begin
real     sky_noise       # sky noise
real     sky             # sky level
real     datamin         # minimum good value to use
real     zmag
string   imname
real     im_sigma        # image standard deviation
string   pst_line        # line from pst file
string   coords          # name of stars plate coordinates file
string   head           # file with Gaussian parameters
string   pst_st
```

```
string   str_psfiter     # string of psf number of iteration
int      npst            # number+65 of PSF stars found by pstselect
string   last_psf        # psf image name
zmag = 25.0
delete (simages,verify=no,>>&"/dev/null")
lis=iraflist
head="tmp.header"
while (fscan (lis, imname) != EOF)
{
   coords = imname // '.coo.1'
   delete ('tmp_trsh',verify=no,>>&"/dev/null")
   # find threshold in ADU
   findthresh (images=imname, gain=gain, readnoi=readnoise, ,coaddtype="average",
               nframes=1, center="mode", verbose=no) | scan(sky_noise, im_sigma)
   sky = (((gain*sky_noise)**2) - (readnoise**2))/gain  # calculate sky value
   datamin = sky - nsigma*sky_noise


   #-----------------------------------------------------------------------
   # DAOFIND - Find stars in image
   #-----------------------------------------------------------------------
   print ('  Find stars using Daofind')
   delete (imname//'.coo.*',verify=no,>>&"/dev/null")
   daofind(image=imname, output="default", verify=no, verbose=yes, thresho=threshold,
           scale=scale, fwhmpsf=fwhm, sigma=sky_noise, datamin=datamin, datamax=datamax,
           noise="poisson", readnoi=readnoise, epadu=gain, exposur=exposurek, airmass=airmassk,
           filter=filterk, obstime=timek)
   psort (imname // ".coo.1","MAG",ascend=yes)
   print ("!coordtransform " // imname // ".coo.1") | cl()
   mv ("out.coords",   imname // ".coo.1")

   #-----------------------------------------------------------------------
   # PHOT - aperture photometry for stars in the image
   #        used as initial guess for DAOPHOT
   #-----------------------------------------------------------------------
   phot: #    doing the aperture photometry
   print ('  Doing the aperture photometry for image : ',imname)
   delete (imname//'.mag.*',verify=no,>>&"/dev/null")
   phot (image=imname, coords="default", output=imname//'.mag.0', interac=no, verify=no,
         verbose=yes, scale=scale, fwhmpsf=fwhm, sigma=sky_noise, datamin=datamin,
         datamax=datamax, noise="poisson", readnoi=readnoise, epadu=gain, exposur=exposurek,
         airmass=airmassk, filter=filterk, obstime=timek, weighti="constant", apertur=aperture,
         zmag=zmag, salgori=salgo, annulus=sannulus, dannulu=wannulus, calgori="none")

    # Delete pst files
    delete (imname//'.pst.*',verify=no,>>&"/dev/null")

   cp ("pstlist",   imname // ".pst.0")
```

```
   imdelete (imname//'.psf.*',verify=no,>>&"/dev/null")
   delete (imname//'.psg.*',verify=no,>>&"/dev/null")
   delete ('plots',verify=no,>>&"/dev/null")
   print('  Calculating the PSF for image : ',imname)
   # check if psf list is empty
   pst_st = imname // ".pst.0"
   count (pst_st) | fields("","1") | scan(npst)
   imdel (imname//'.sub.*',verify=no)
   delete (imname//'.nst.*',verify=no,>>&"/dev/null")
   delete (imname//'.nrj.*',verify=no,>>&"/dev/null")

if (npst > 66)
   {
      psf(image=imname, photfile=imname//'.mag.0', pstfile=imname //'.pst.0',
          psfimage=imname//'.psf.0', opstfile=imname //'.pst.1', groupfil=imname//'.grp.0',
          interac=no, verify=no, scale=scale, fwhmpsf=fwhm, sigma=sky_noise, datamin=datamin,
          datamax=datamax, noise="poisson", readnoi=readnoise, epadu=gain, exposur=exposurek,
          airmass=airmassk, filter=filterk, obstime=timek, functio="gauss",
          varorde=2, saturat=no, psfrad=psfrad, fitrad=fitrad, recente=yes, fitsky=yes,
          sannulu=sannulus, wsannul=wannulus, nclean=nclean)

    # number of PSF iterations
    if (psfiter==1)
      {
         # one PSF iteration, allready done
      }
      else
      {

# doing the second PSF iteration
 nstar(image=imname, groupfil=imname//'.grp.0',psfimage=imname//'.psf.0',
       nstarfil=imname//'.nst.0', rejfile="default", verify=no, scale=scale, fwhmpsf=fwhm,
         sigma=sky_noise, datamin=datamin, datamax=datamax, noise="poisson",
         readnoi=readnoise, epadu=gain, exposur=exposurek, airmass=airmassk,
         filter=filterk, obstime=timek, functio="gauss", varorde=2, psfrad=nspsfrad,
         fitrad=fitrad, recente=yes, fitsky=yes, sannulu=sannulus, wsannul=wannulus,
         mergerad=mergerad)
 #imdel (imname//'.sub.*',verify=no)
 substar(image=imname, photfile=imname//'.nst.0', exfile="pstlist", psfimage=imname//'.psf.0',
         subimag=imname//'.sub.0', verify=no, scale=scale, fwhmpsf=fwhm, sigma=sky_noise,
         datamin=datamin, datamax=datamax, noise="poisson", readnoi=readnoise, epadu=gain,
         exposur=exposurek, airmass=airmassk, filter=filterk, obstime=timek, functio="gauss",
         varorde=2, psfrad=nspsfrad, fitrad=fitrad, recente=no, fitsky=yes, sannulu=sannulus,
         wsannul=wannulus)
 delete (imname//'sub.1.pst.*',verify=no,>>&"/dev/null")
 psf(image=imname//'.sub.0', photfile=imname//'.nst.0', pstfile=imname //'.pst.1',
     psfimage=imname//'.psf.1', opstfile=imname //'.pst.2', groupfil=imname//'.grp.1',
     interac=no, verify=no, scale=scale, fwhmpsf=fwhm, sigma=sky_noise, datamin=datamin,
```

```
     datamax=datamax, noise="poisson", readnoi=readnoise, epadu=gain, exposur=exposurek,
     airmass=airmassk, filter=filterk, obstime=timek, functio="gauss", varorde=2, saturat=no,
     psfrad=psfrad, fitrad=fitrad, recente=yes, fitsky=yes, sannulu=sannulus, wsannul=wannulus,
     nclean=nclean)

# doing the third PSF iteration
 nstar(image=imname, groupfil=imname//'.grp.1', psfimage=imname//'.psf.1',
       nstarfil=imname//'.nst.1', rejfile="default", verify=no, scale=scale,
       fwhmpsf=fwhm, sigma=sky_noise, datamin=datamin, datamax=datamax,
       noise="poisson", readnoi=readnoise, epadu=gain, exposur=exposurek,
       airmass=airmassk, filter=filterk, obstime=timek, functio="gauss",
       varorde=2, psfrad=nspsfrad, fitrad=fitrad, recente=yes, fitsky=yes,
       sannulu=sannulus, wsannul=wannulus, mergerad=mergerad)
 #imdel (imname//'.sub.*',verify=no)
 substar(image=imname, photfile=imname//'.nst.1', exfile="pstlist",
         psfimage=imname//'.psf.1', subimag=imname//'.sub.1', verify=no,
         scale=scale, fwhmpsf=fwhm, sigma=sky_noise, datamin=datamin,
         datamax=datamax, noise="poisson", readnoi=readnoise, epadu=gain,
         exposur=exposurek, airmass=airmassk, filter=filterk, obstime=timek,
         functio="gauss", varorde=2, psfrad=nspsfrad, fitrad=fitrad,
         recente=no, fitsky=yes, sannulu=sannulus, wsannul=wannulus)
 delete (imname//'sub.2.pst.*',verify=no,>>&"/dev/null")
 psf(image=imname//'.sub.1', photfile=imname//'.nst.1', pstfile=imname //'.pst.2',
     psfimage=imname//'.psf.2', opstfile=imname //'.pst.3', groupfil=imname//'.grp.2',
     interac=no, verify=no, scale=scale, fwhmpsf=fwhm, sigma=sky_noise, datamin=datamin,
     datamax=datamax, noise="poisson", readnoi=readnoise, epadu=gain, exposur=exposurek,
     airmass=airmassk, filter=filterk, obstime=timek, functio="gauss", varorde=2, saturat=no,
     psfrad=psfrad, fitrad=fitrad, recente=yes, fitsky=yes, sannulu=sannulus, wsannul=wannulus,
     nclean=nclean)

#-------------------------------------------------------------------------
# starting allstar
#-------------------------------------------------------------------------
allstar1: #   fitting the PSF and measuring all the stars
print (' measuring allstars in ',imname)
#imdelete (imname//'.sub.*',verify=no,>>&"/dev/null")
delete (imname//'.als.*',verify=no,>>&"/dev/null")
delete (imname//'.arj.*',verify=no,>>&"/dev/null")
# ----------- First Sweep of ALLSTAR -----------------------------------
   last_psf = imname//'.psf.2'
allstar(image=imname, photfile=imname//'.mag.0', psfimage=last_psf,
        allstarf=imname//".als.1", rejfile="default", subimage=imname//".sub.2",
        verbose="yes", verify=no, scale=scale, fwhmpsf=fwhm, sigma=sky_noise,
        datamin=datamin, datamax=datamax, noise="poisson", readnoi=readnoise, epadu=gain,
        exposur=exposurek, airmass=airmassk, filter=filterk, obstime=timek, functio="gauss",
        varorde=2, psfrad=psfrad, fitrad=fitrad, recente=yes, fitsky=yes,
        sannulu=sannulus, wsannul=wannulus, mergerad=mergerad)
```

```
#   clean the als file from INDEF stars
print(' cleaning '//imname//'.als.1')
delete (imname//'.sls.*',verify=no,>>&"/dev/null")
txdump(textfile=imname//'.als.1', fields="ID,XCENTER,YCENTER,MAG,MERR,MSKY,
       NITER,SHARPNESS, CHI", expr="MAG[1] !=INDEF", headers=yes, >imname//'.sls.1')
#-------------------------------------------------------------------------

hselect (imname//'.psf.2.fits',"PAR1", "PAR1>0", missing="INDEF") | scan(x)
hselect (imname//'.psf.2.fits',"PAR2", "PAR2>0", missing="INDEF") | scan(y)
z =(x + y)

txdump(textfile=imname//'.als.1',fields="XCENTER,YCENTER,MAG,MERR,SHARPNESS,ID",
       expr="ID>0", headers=yes,>imname//'.als.2')

         #---------------------------------------------------------------
         # PHOT - aperture photometry for stars in the image
         #        used as initial guess for allstar 2nd iteration
         #---------------------------------------------------------------
         print ('  Doing the aperture photometry for image : ',imname)
         delete (imname//'.sub.*.mag.*',verify=no,>>&"/dev/null")

         phot (image=imname, coords=imname //'.als.2', output=imname//'.mag.1',
               interac=no, verify=no, verbose=yes, scale=scale, fwhmpsf=fwhm,
               sigma=sky_noise, datamin=datamin, datamax=datamax,
               noise="poisson", readnoi=readnoise, epadu=gain, exposur=exposurek,
               airmass=airmassk, filter=filterk, obstime=timek, weighti="constant",
               apertur=z, zmag=zmag, salgori=salgo, annulus=sannulus,
               dannulu=wannulus, calgori="none")
        txdump(textfile=imname//'.mag.1',fields="ID,XCENTER,YCENTER,MAG,MERR,SHARPNESS",
               expr="ID>0", headers=yes,>imname//'.ape.1')

x=1.5*z
         phot (image=imname, coords=imname //'.als.2', output=imname//'.mag.2', interac=no,
               verify=no, verbose=yes, scale=scale, fwhmpsf=fwhm, sigma=sky_noise,
               datamin=datamin, datamax=datamax, noise="poisson", readnoi=readnoise,
               epadu=gain, exposur=exposurek, airmass=airmassk, filter=filterk,
               obstime=timek, weighti="constant", apertur=x, zmag=zmag, salgori=salgo,
               annulus=sannulus, dannulu=wannulus, calgori="none")
         txdump(textfile=imname//'.mag.2',fields="ID,XCENTER,YCENTER,MAG,MERR,SHARPNESS",
                expr="ID>0", headers=yes,>imname//'.ape.2')

x=2.0*z
         phot (image=imname, coords=imname //'.als.2', output=imname//'.mag.3',
               interac=no, verify=no, verbose=yes, scale=scale, fwhmpsf=fwhm,
               sigma=sky_noise, datamin=datamin, datamax=datamax, noise="poisson",
               readnoi=readnoise, epadu=gain, exposur=exposurek, airmass=airmassk,
               filter=filterk, obstime=timek, weighti="constant", apertur=x, zmag=zmag,
               salgori=salgo, annulus=sannulus, dannulu=wannulus, calgori="none")
```

```
        txdump(textfile=imname//'.mag.3',fields="ID,XCENTER,YCENTER,MAG,MERR,
               SHARPNESS", expr="ID>0", headers=yes,>imname//'.ape.3')

x=2.5*z
         phot (image=imname, coords=imname //'.als.2', output=imname//'.mag.4',
               interac=no, verify=no, verbose=yes, scale=scale, fwhmpsf=fwhm,
               sigma=sky_noise, datamin=datamin, datamax=datamax, noise="poisson",
               readnoi=readnoise, epadu=gain, exposur=exposurek, airmass=airmassk,
               filter=filterk, obstime=timek, weighti="constant", apertur=x,
               zmag=zmag, salgori=salgo, annulus=sannulus, dannulu=wannulus,
               calgori="none")
         txdump(textfile=imname//'.mag.4',fields="ID,XCENTER,YCENTER,MAG,MERR,
                SHARPNESS", expr="ID>0", headers=yes,>imname//'.ape.4')

#-------------------------------------------------------------------------
del header
del tmp.jd
imhead (imname, imlist=imname, longheader=yes, userfields=yes, > "header")
!awk '{if ($1=="JD") printf("%s \n",$3)}'  header >> tmp.jd
#-------------------------------------------------------------------------

print ("!./pasteall "//imname) | cl()
print ("!./diffphot "//imname//'.all') | cl()
mv ("diffout.dat",   imname // ".diff")

      print(imname,>>simages)

    }
   }

   else
   {
      print ('image ',imname,' ,  psf list empty',>> errlog)
      print ('image ',imname,' PSF list is empty')
      # goto next image
   }
   # display subtracted image
   if (im_disp)
   {
      display(imname//'.sub.3',1)
   }
   # delete *.sub.# images
   if (del_sub)
   {
      imdelete(imname//'.sub.*',verify=no)
   }
}
end
```

The following code is responsible for doing differential photometry on the output from the PSF and aperture photometry.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <complex.h>



int main(int argc, char **argv)
{
  if (argc != 2)
  {
    printf("usage: %s filename \n", argv[0]);
    }


//#############################################################################
  FILE *fp1;                                                    /*variables*/
  FILE *fp2;
  FILE *fp3;
  float b[4000][14];
  float mag_const, flux_const, flux[4000], delta_mag[4000], flux_err[4000],
        flux_const_err, delta_mag_err[4000];
  int a1;
  double jd;
  char s1[4000];
  int i,j,k,l,y,g,q, nlines;
  int id1=4, id2=5, id3=6, id4=8, id5=19, num[4000] ;
//#############################################################################



if((fp1=fopen(argv[1], "r")) == NULL) {
    printf("Cannot open file.\n", argv[1]);
           exit(1);
      }


 if((fp2=fopen("diffout.dat", "w+")) == NULL) {
      printf("Cannot open file.\n");
           exit(1);
 }

 if((fp3=fopen("tmp.jd", "r")) == NULL) {
      printf("Cannot open file.\n");
```

```
          exit(1);
 }
//#############################################################################
                                          /*read number of lines in .all file*/
char line[256], *tmpchar;
 nlines = 0;
 while (fgets(line, sizeof(line), fp1) != NULL)
   {
    tmpchar = &line[0];

         tmpchar++;
         fscanf (fp1, "%f\n", &a1);
          nlines = nlines + 1;

    }
//printf("%d \n", nlines);


//#############################################################################
  rewind(fp1);                                          /*read from .all file*/
  rewind(fp2);

 mag_const=0.0;
 flux_const=0.0;
 flux[1]=0.0;
 delta_mag[1]=0.0;
flux_const_err = 0.0;
 for(i=1;i<=nlines;i++)
{

  fscanf(fp1, "%d%f%f%f%f%f%f%f%f%f%f%f%f\n", &num[i], &b[i][2], &b[i][3],
  &b[i][4], &b[i][5], &b[i][6], &b[i][7], &b[i][8], &b[i][9], &b[i][10],
  &b[i][11], &b[i][12], &b[i][13], &b[i][14]);

  flux[i]=pow(10.0, -0.4*b[i][4]);
  flux_err[i]=0.4*flux[i]*logf(10)*b[i][5];


 if ((num[i]==id1)||(num[i]==id2)||(num[i]==id3)||(num[i]==id4)||(num[i]==id5))
 {

    flux_const += flux[i];
    flux_const_err += powf(flux_err[i],2.0);


 }
}
```

```
//b1 ID
//b2, b3 Coordinates
//b4, b5 Psf MAG + MERR
//b6  Sharpness
//b7, b8 Aperture1 MAG + MERR
//b9, b10 Aperture2 MAG +MERR
//b11, b12 Aperture3 MAG +MERR
//b13, b14 Aperture4 MAG +MERR

//printf("%5.5e %5.5e  \n",flux_const, flux_const_err);

fscanf(fp3,"%lf \n", &jd);
//printf("%16.8lf  \n",jd);

 for(i=1;i<=nlines;i++)
{
  delta_mag[i]=(-2.5*log10f(flux[i]/flux_const));

  //fprintf(fp2, "%5.5e %5.5e \n",flux_err[i], flux_const_err);


  delta_mag_err[i] = sqrtf(powf(-2.5/log(10),2.0)*(powf(flux_err[i]/flux[i],2.0)
                     +flux_const_err/powf(flux_const,2.0)));


  //$1=ID, $2=xcenter, $3=ycenter, $4=mag, $5=merr, $6=diff mag, $7=diff mag err

  fprintf(fp2, "%d %f %f %f %f %f %f %16.8lf \n", num[i], b[i][2], b[i][3],
               b[i][4], b[i][5], delta_mag[i], delta_mag_err[i], jd);


}



//#############################################################################


  fclose(fp1);
  fclose(fp2);
  fclose(fp3);
  return 0;
}
```

Bibliography

American Association of Variable Star Observers. 2012, Types of Variables, `http://www.aavso.org/types-variables`, [Online; accessed 29 August 2012]

Bailey, J. D., Grunhut, J., Shultz, M., et al. 2012, Astrophysics, 17

Baker, N. & Kippenhahn, R. 1962, Z. Astrophys., 54, 114

Balona, L. A., Dziembowski, W. A., & Pamyatnykh, A. 1997, MNRAS, 289, 25

Battinelli, P., Brandimarti, A., & Capuzzo-Dolcetta, R. 1994, A&AS, 104, 379

Battinelli, P. & Capuzzo-Dolcetta, R. 1991, MNRAS, 249, 76

Bergin, E. A., Goldsmith, P. F., Snell, R. L., & Langer, W. D. 1997, ApJ, 482, 285

Bergin, E. A. & Tafalla, M. 2007, ARA&A, 45, 339

Birney, D., Gonzalez, G., & Oesper, D. 2006, Observational Astronomy (Cambridge, UK: Cambridge University Press)

Böhm-Vitense, E. 1992, Introduction to stellar astrophysics, Volume 3 (University of Washington: Cambridge University Press)

Böhm-Vitense, E. 1993, Introduction to stellar astrophysics, Volume 2 (University of Washington: Cambridge University Press)

Brown, A. G. A. 2001, in Revista Mexicana de Astronomia y Astrofisica Conference Series, Vol. 11, Revista Mexicana de Astronomia y Astrofisica Conference Series, 89

Carroll, B. & Ostlie, D. 1996, An Introduction to Modern Astrophysics (Weber State University: Addison-Wesley publishing company)

Chabrier, G. 2003, PASP, 115, 763

Davidson, K. & Humphreys, R. M. 1997, ARA&A, 35, 1

de La Fuente Marcos, R. 1998, PASP, 110, 1117

Elmegreen, B. G. & Efremov, Y. N. 1997, ApJ, 480, 235

Forbes, D. 1986, PASP, 98, 218

Forbes, D. & Short, S. 1996, AJ, 111, 1609

Frescura, F. A. M., Engelbrecht, C. A., & Frank, B. S. 2007, ArXiv e-prints

Frescura, F. A. M., Engelbrecht, C. A., & Frank, B. S. 2008, MNRAS, 388, 1693

Friel, E. D. 1995, ARA&A, 33, 381

Frost, E. B. 1906, ApJ, 24, 259

Horne, J. H. & Baliunas, S. L. 1986, ApJ, 302, 757

Howell, S. 2006, Handbook of CCD astronomy (Cambridge, UK: Cambridge University Press)

Humphreys, R. M. 1978, ApJS, 38, 309

Iglesias, C. A. & Rogers, F. J. 1991, ApJ, 371, L73

Kennicutt, R. C. & Evans, N. J. 2012, ARA&A, 50, 531

King, D. S. & Cox, J. P. 1968, PASP, 80, 365

King, I. 1971, PASP, 83, 3

Kippenhahn, R. & Weigert, A. 1994, Stellar structure and evolution (Göttingen, Germany: Springer-Verlag)

Klessen, R. S., Burkert, A., & Bate, M. R. 1998, ApJ, 501, L205

Kołaczkowski, Z., Pigulski, A., Soszyński, I., et al. 2004, in Astronomical Society of the Pacific Conference Series, Vol. 310, IAU Colloq. 193: Variable Stars in the Local Group, ed. D. W. Kurtz & K. R. Pollard, 225

Kroupa, P. 2002, Sci, 295, 82

Lata, S., Pandey, A. K., Maheswar, G., Mondal, S., & Kumar, B. 2011, MNRAS, 418, 1346

Lesh, J. R. & Aizenman, M. L. 1973, A&AS, 22, 229

Lesh, J. R. & Aizenman, M. L. 1978, ARA&A, 16, 215

Maciel, W. J. & Costa, R. D. D. 2010, in IAU Symposium, Vol. 265, IAU Symposium, ed. K. Cunha, M. Spite, & B. Barbuy, 317–324

Marigo, P., Girardi, L., Bressan, A., et al. 2008, A&AS, 482, 883

McKee, C. F. & Ostriker, E. C. 2007, ARA&A, 45, 565

McLean, I. 1997, Electronic imaging in astronomy Detectors and instrumentation (Chichester, West Sussex, England: Wiley)

Mermilliod, J. C. 1981, A&AS, 44, 467

Moskalik, P. 1995, in Astronomical Society of the Pacific Conference Series, Vol. 83, IAU Colloq. 155: Astrophysical Applications of Stellar Pulsation, ed. R. S. Stobie & P. A. Whitelock, 44

Moskalik, P. & Dziembowski, W. A. 1992, ApJ, 256, L5

Nasa Ames Research Center. 2011, Kepler, `http://kepler.nasa.gov/science/about/targetFieldOfView/stellarVariability/lightcurves/`, [Online; accessed 29 August 2012]

Nichols, J. S., Henden, A. A., Huenemoerder, D. P., et al. 2010, ApJS, 188, 473

Nilakshi, Sagar, R., Pandey, A. K., & Mohan, V. 2002, A&AS, 383, 153

Ofek, E. 1997, IRAF Programs, `http://wise-obs.tau.ac.il/~eran/iraf/index.html`, [Online; accessed 04-April-2012]

Paczynski, B. & Pojmanski, G. 2000, in Bulletin of the American Astronomical Society, Vol. 32, American Astronomical Society Meeting Abstracts #196, 687

Padmanabhan, T. 2001, Theoretical Astrophysics, Volume II: Stars and Stellar Systems (Inter-University Centre for Astronomy and Astrophysics, Pune, India: Cambridge University Press)

Pamyatnykh, A. A. 1999, Acta Astron., 49, 119

Pigulski, A. & Kołaczkowski, Z. 2002, A&AS, 388, 88

Pigulski, A., Kopacki, G., Kołaczkowski, Z., & Jerzykiewicz, M. 2002, in Astronomical Society of the Pacific Conference Series, Vol. 259, IAU Colloq. 185: Radial and Nonradial Pulsationsn as Probes of Stellar Physics, ed. C. Aerts, T. R. Bedding, & J. Christensen-Dalsgaard, 146

Pigulski, A. & Pojmański, G. 2008, A&AS, 477, 907

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992, Numerical Recipes in C: The Art of Scientific Computing. Second Edition

Ridpath, I. 2012, Oxford dictionary of astronomy (New York, USA: Oxford University Press)

Rieke, G. H. & Lebofsky, M. J. 1985, ApJ, 288, 618

Rybicki, G. & Lightman, A. 1979, Radiative processes in astrophysics (Harvard-Smithsonian Center for Astrophysics: John Wiley and Sons)

Salaris, M. & Cassisi, S. 2005, Evolution of stars and stellar populations (Chichester, West Sussex, England: Wiley)

Salpeter, E. E. 1955, ApJ, 121, 161

Simon, N. R. 1982, ApJ, 260, L87

Stankov, A. & Handler, G. 2005, ApJS, 158, 193

Stellingwerf, R. F. 1978, AJ, 83, 1184

Stellingwerf, R. F. 1979, ApJ, 227, 935

Sterken, C. & Jerzykiewicz, M. 1992, Space Sci. Rev., 62, 95

The International Variable Star Index. 2012, VARIABLE STAR TYPE DESIGNATIONS IN VSX, `http://www.aavso.org/vsx/help/VariableStarTypeDesignationsInVSX.pdf`, [Online; accessed 29 August 2012]

Williams, J. P., Blitz, L., & McKee, C. F. 2000, Protostars and Planets IV, 97