Appendix A
Plotting Transfer Functions

Abstract This appendix gives several simple MATLAB scripts to interactively plot the transfer function for CTEM and STEM. This is a simple and relatively easy to use approach to investigate the transfer function.

The following scripts interactively calculate and plot transfer functions for the CTEM and STEM on the screen or in publication quality hardcopy using Matlab (distributed and trademarked by: The MathWorks, Inc., www.mathworks.com). Matlab is relatively easy to use and provides a graphical output on many popular computers in a nearly machine independent manner. It is a complete programming language and has a variety of sophisticated mathematical functions and procedures. Matlab's ease of use comes at a price however. It is mostly an interpreted language (newer versions have a just-in-time compiler that improves performance) with the inherent speed penalty. However Matlab's fundamental operations are on matrices and vectors. If the problem is vectorized (i.e., the operation are on a whole array or vector of numbers at one time) then the performance penalty typically associated with an interpreted language is partially overcome. Matlab is well suited for small to medium calculations (such as calculating and plotting transfer functions) but probably should not be used for large numerical simulation.

Each of the three MATLAB programs ctemtf.m, stempsf.m and stemmtf.m should be called directly from the MATLAB command line. All files should be in the default directory. The MATLAB functions do not need to be called directly but are called from the other three programs. Each program first asks for the electron optical parameters. Then it calculates and plots the appropriate function on the screen. The STEM programs may take a significantly longer time because they must calculate a Fourier-Bessel transform. Once the graph appears on the screen it can be printed from the command line using the standard Matlab print command.

E.J. Kirkland, *Advanced Computing in Electron Microscopy*,
DOI 10.1007/978-1-4419-6533-2_9,

A.1 CTEM

There is one Matlab programs and one Matlab functions (each is a separate file). A sample output of each program is shown later (Fig. A.1) along with the source listing.

- **ctemtf.m** Plot CTEM transfer function (calls ctemh.m below).
- **ctemh.m** Calculate the CTEM transfer function (3.40).

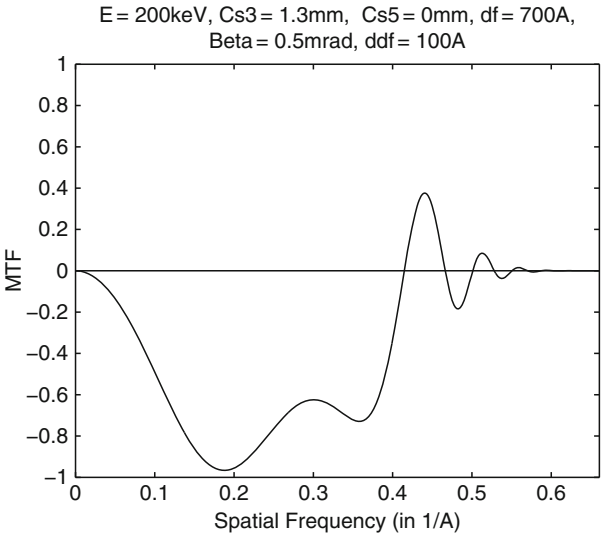

Fig. A.1 Example of CTEM transfer function from the MATLAB program **ctemtf.m**

ctemtf.m

```
%
%  MATLAB script ctemtf.m to plot CTEM transfer functions
%  this script calls ctemh.m
%
%  Cs3,5 = Spherical Aberration
%  df = defocus
%  kev = electron energy in keV
%  ddf = chromatic aberation defocus spread
%  beta =  spread in illumination angles
%
disp( 'Plot CTEM transfer function' );
p.kev  = input('Type electron energy in keV : ');
p.Cs3  = input('Type spherical aberration Cs3 in mm : ');
p.Cs5  = input('Type spherical aberration Cs5 in mm : ');
p.df   = input('Type defocus df in Angstroms : ');
p.ddf  = input('Type defocus spread ddf in Angs. : ');
```

```
p.beta = input('Type illumination semiangle in mrad : ');
%
% electron wavelength
wav = 12.3986/sqrt((2*511.0+p.kev)*p.kev);
Cs = abs(p.Cs3);
if( Cs < 0.1 )
    Cs = 0.1;
end
ds = sqrt( sqrt( Cs*1.0e7*wav*wav*wav ));
kmax = 2.5/ds;
k = 0.:(kmax/500):kmax;  % 500 points
sinw = ctemh( k, p, 0 );
plot( k, sinw  );
axis([0, kmax, -1, +1]);
xlabel( 'Spatial Frequency (in 1/A)');
ylabel( 'MTF' );
s1 = sprintf('E= %gkeV, Cs3= %gmm, ', p.kev, p.Cs3);
s2 = sprintf(' Cs5= %gmm, df= %gA,  ', p.Cs5, p.df);
s3 = sprintf('Beta= %gmrad, ddf= %gA', p.beta, p.ddf);
title([s1 s2 s3]);
hold on;         % plot line through zero
x = [0, kmax];
y = [0, 0];
plot( x, y );
hold off;
```

ctemh.m

```
function y = ctemh(k,params,type)
%
%  MATLAB function ctemh.m to calculate CTEM bright
%  field phase contrast transfer function with partial
%  coherence for weak phase objects
%    input array k has the spatial freq. values (in 1/A)
%    input array params has the optical parameters
%          params = [Cs, df, kev, ddf, beta]
%    input type = 0 for phase contrast
%           and 1 for amplitude contrast
%    output array contains the transfer function vs k
%
%  params.Cs3,5  = spherical aberration (in mm)
%  params.df   = defocus (in Angstroms)
%  params.kev  = electron energy (in keV)
%  params.ddf  = chrom. aberr. def. spread (in Angst.)
%  params.beta = spread in illum. angles (in mrad)
%
%  reference
%  R. H. Wade and J. Frank, Optik 49 (1977) p.81
%
Cs3 = params.Cs3*1.0e7;
Cs5 = params.Cs5*1.0e7;
df = params.df;
kev = params.kev;
```

```
ddf = params.ddf;
beta = params.beta*0.001;
mo = 511.0;       % electron rest mass in keV
hc = 12.3986;     % in keV-Angstroms
wav = (2*mo)+kev;
wav = hc/sqrt(wav*kev);
wavsq = wav*wav;
w1 = pi*Cs3*wavsq*wav;
w2 = pi*wav*df;
w3 = pi*Cs5*wavsq*wavsq*wav;
e0 = (pi*beta*ddf)^2;
k2 = k .* k;
wr = ((w3.*k2+w1).*k2-w2).*k*beta/wav;
wi = pi*wav*ddf.*k2;
wi = wr.*wr + 0.25.*wi.*wi;
wi = exp(-wi./(1+e0.*k2));
wr = w3*(1-2.0*e0.*k2)/3.0;
wr = wr.*k2 + 0.5*w1.*(1-e0.*k2);
wr = (wr.*k2 - w2).*k2./(1+e0.*k2);
if type == 0
    y = sin(wr).* wi;
else
    y = cos(wr).* wi;
end;
```

A.2 STEM

There are two Matlab programs and three Matlab functions (each is a separate file). The first integral over the lens aberration function is not well behaved so a Matlab adaptive quarature routine is used to gain efficiency and accuracy (in stmhr.m). A sample output of each program is shown later (Figs. A.2, A.3) along with the source listing.

- **stempsf.m** Plot the STEM probe profile (calls stemhr.m below).
- **stemtf.m** Plot the STEM transfer function (calls stemhr.m and stemhk.m below).
- **stemhr.m** Calculate the STEM probe profile (3.69, calls lens.m).
- **stemhk.m** Calculate the STEM transfer function(3.70, calls stemhr.m).
- **lens.m** Calculate the lens aberration function to use for the adaptive quadrature function.

stempsf.m

```
%  stempsf.m
%  Matlab file to plot the STEM probe profile
%  this script calls stemhr.m
%
clear;
clf;
disp( 'Plot STEM probe intensity' );
```

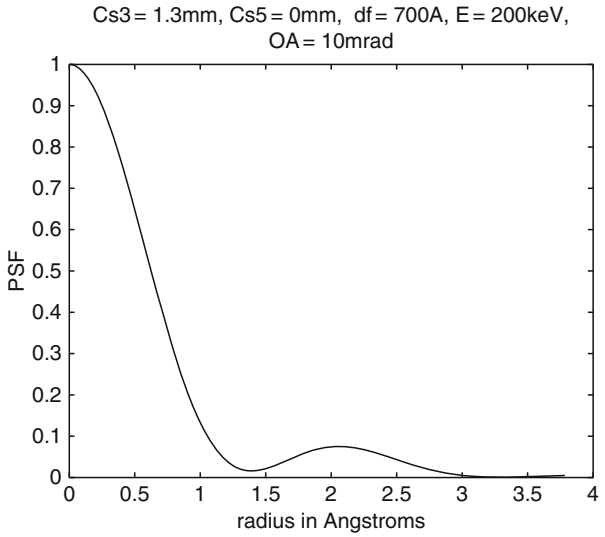

Fig. A.2 Example of STEM probe intensity profile the output from the MATLAB program **stempsf.m**

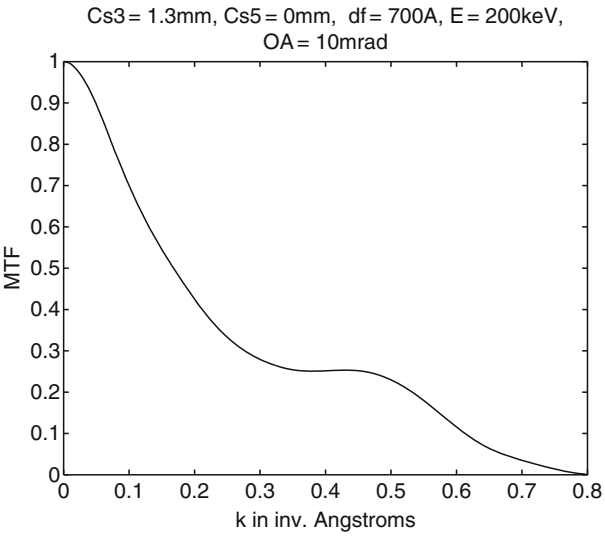

Fig. A.3 Example of the STEM transfer function output from the MATLAB program **stemtf.m**

```
p.kev  = input( 'Type electron energy in keV : ');
p.Cs3  = input( 'Type spherical aberration Cs3 in mm : ');
p.Cs5  = input( 'Type spherical aberration Cs5 in mm : ');
p.df   = input( 'Type defocus df in Angstroms : ');
p.amax = input( 'Type obj. apert. semiangle in mrad : ');
%
% electron wavelength
```

```
wav = 12.3986/sqrt((2*511.0+p.kev)*p.kev);
Cs = abs(p.Cs3);
if( Cs < 0.1 )
    Cs = 0.1;
end
rmax = sqrt( sqrt( Cs*1.0e7*wav*wav*wav ));
npts = 300;   %  number of points in curve
r = 0:(rmax/npts):rmax;
psf = stemhr( r, p );
plot( r, psf );
xlabel( 'radius in Angstroms');
ylabel( 'PSF' );
s1 = sprintf('Cs3= %gmm, Cs5= %gmm, ', p.Cs3, p.Cs5 );
s2 = sprintf(' df= %gA, ', p.df);
s3 = sprintf('E= %gkeV, OA= %gmrad', p.kev, p.amax);
title([s1 s2 s3]);
```

stemtf.m

```
%  stemtf.m
%  Matlab file to plot the STEM probe mtf
%  this script calls stemhk.m
%
clear;
clf;
disp( 'Plot STEM transfer function' );
p.kev  = input( 'Type electron energy in keV : ');
p.Cs3  = input( 'Type spherical aberration Cs3 in mm : ');
p.Cs5  = input( 'Type spherical aberration Cs5 in mm : ');
p.df   = input( 'Type defocus df in Angstroms : ');
p.amax = input( 'Type obj. apert. semiangle in mrad : ');
%
% electron wavelength
wav = 12.3986/sqrt((2*511.0+p.kev)*p.kev);
kmax = 2*0.001*p.amax/wav;
npts = 500;  % number of points
k = 0:(kmax/npts):kmax;
mtf = stemhk( k, p );
plot( k, mtf );
xlabel( 'k in inv. Angstroms');
ylabel( 'MTF' );
s1 = sprintf('Cs3= %gmm, Cs5= %gmm, ', p.Cs3, p.Cs5 );
s2 = sprintf(' df= %gA, ', p.df);
s3 = sprintf('E= %gkeV, OA= %gmrad', p.kev, p.amax);
title([s1 s2 s3]);
```

stemhr.m

```
function psf = stemhr(r,params)
%
%  MATLAB function stemhf.m to calculate
```

```
%            STEM probe profile vs. r
%    input array r has the radial positions (in Angs.)
%    input variable params has the optical parameters
%          <Cs, df, kev, amax> as elements
%    output array contains the transfer function
%
%  param.Cs3  = third order spherical aberration (in mm)
%  param.Cs5  = fifth order spherical aberration (in mm)
%  param.df   = defocus (in Angstroms)
%  param.kev  = electron energy (in keV)
%  param.amax = objective aperture  (in mrad)
%
global w2 w4 w6 intr;  % constants for lens.m
df = params.df;
kev = params.kev;
amax = params.amax*0.001;
% electron wavelength
wav = 12.3986/sqrt((2*511.0+kev)*kev);
kmax = amax/wav;
w2 = wav*pi*df;
w4 = 0.5*pi*params.Cs3*1.0e7*wav*wav*wav;
w6 = pi*params.Cs5*1.0e7*wav*wav*wav*wav*wav /3.0;
nr = length( r );
for ir=1:nr,
    intr = 2*pi*r(ir);
    tol = 1.0e-7/(0.01*ir); % sliding accuracy to speed up
    % use adaptive quadrature because integrand
    %     not well behaved
    hr(ir) = quad( @lens, 0, kmax, tol );
end;
% a little faster than abs()
psf = real(hr).^2 + imag(hr).^2;
a = max(psf);
psf = psf/a;   % norm. probe intensity to a max. of 1
```

stemhk.m

```
function mtf = stemhk( k, params )
%
%  MATLAB function stemhk.m to calculate STEM mtf vs. k
%    input array k has the spatial freq. (in inv. Angs.)
%    input variable params has the optical parameters
%          [Cs, df, kev, amax] as elements
%    output array contains the transfer function
%
%  param.Cs3  = third order spherical aberration (in mm)
%  param.Cs5  = fifth order spherical aberration (in mm)
%  param.df   = defocus (in Angstroms)
%  param.kev  = electron energy (in keV)
%  param.amax = objective aperture  (in mrad)
%
Cs3 = params.Cs3;
```

```
df = params.df;
kev = params.kev;
amax = params.amax*0.001;
%  first calculate the psf using stemhr()
nr = 500;  % number of points in integral over r
wav = 12.3986/sqrt((2*511.0+kev)*kev); % elect. wavelength
Cs = abs(Cs3);
if( Cs < 0.1 )  % guess if Cs3=0
    Cs = 0.1;
end
rmax = 2.0*sqrt( sqrt( Cs*1.0e7*wav*wav*wav) );
r = 0:(rmax/nr):rmax;
psf = stemhr( r, params );
% next inverse psf to get mtf
nk = length( k );
for ik=1:nk,
h = psf .* besselj( 0, 2*pi*r*k(ik) ) .*r;
mtf(ik) = sum(h);
end;
a = mtf(1);
mtf = mtf/a;  % normalize mtf(0)=1
```

lens.m

```
function expchi = lens(k)
%
%  dummy function to integrate (used by stempsf.m)
%  MATLAB function lens.m to calculate complex
%                                  aberr. function
%    input k (in 1/Angs.), wav = electron wavelength
%
%  chi = pi*wav*k^2*[ 0.5*Cs3*wav^2*k^2
%                  + (1/3)*Cs5*wav^4*k^4 - df ]
%  return exp( -i*chi )
%
% globals:
%   w2 = pi*defocus*wav
%   w4 = 0.5*pi*Cs3*wav^3
%   w6 = (1/3)*pi*Cs5*wav^5
%   intr = 2*pi*r
%
global w2 w4 w6 intr;  % constants from lenhr.m
k2 = k.*k;
w = ( (w6.*k2 + w4) .*k2 - w2 ).*k2;
expw = exp( -i*w );
expchi = expw .* besselj( 0, intr.*k ).*k;
```