In[∘]:=
```
ClearAll
```
Out[∘]=
```
ClearAll
```

biological parameters

In[∘]:=
```
c = 299 792 458 ;
hb = 1.0545715964207855 * 10^-34;
ϵ0 = 8.85 * 10^-12;
re = 2.82 * 10^-13 * 10^-2;
q = 1.602 * 10^-19;
```

In[∘]:=
```
wsig = 9 * 10^3 * q / hb;
```

In[∘]:=
```
λsig = 2 * Pi * c / wsig;
```

In[∘]:=
```
f1C = 6.01; f1N = 7.02; f1H = 1; f1O = 8.04; f1S = 16.27; f1CI = 17.31;
```

In[∘]:=
```
AC = 12.011; AN = 14.007; AH = 1.008; AO = 15.999; AS = 32.06; ACI = 35.45;
NA = 6.022 * 10^23;
cmCubeToMeterCube = 100^3;
```

In[∘]:=
```
phase[sH_, sC_, sN_, sO_, sS_, sCI_, thickness_, materialDensity_, λ_] :=
 (2 π/λ) * thickness * (re/(2 π)) * materialDensity * NA *
  (1/(sH * AH + sC * AC + sN * AN + sO * AO + sS * AS + sCI * ACI)) *
  (sH * f1H + sC * f1C + sN * f1N + sO * f1O + sS * f1S + sCI * f1CI) * λ^2 * cmCubeToMeterCube
```

In[∘]:=
```
phase[48.6, 32.9, 8.9, 8.9, 0.6, 0, 5 * 10^-9, 1.35, λsig]
```
Out[∘]=
```
0.000843499
```

cropping functions

In[∘]:=
```
Clear[lowerleft]
lowerleft =
  Function[biosample, For[i = 1, i < 0.2 * Dimensions[biosample][[1]], i++, For[j = 1,
     j < 0.2 * Dimensions[biosample][[2]], j++, If[biosample[[i, j]] == 0, Break[]]];
    If[biosample[[i, j]] == 0, Break[]]];
   {i, j}] (*{minindex1,minindex2}*);
```

In[∘]:=
```
Clear[upperright]
upperright = Function[biosample, For[i = Floor[0.8 * Dimensions[biosample][[1]]],
    i < Dimensions[biosample][[1]], i++, For[j = Floor[0.8 * Dimensions[biosample][[2]]],
     j < Dimensions[biosample][[2]], j++, If[biosample[[i, j]] == 1, Break[]]]];
   {i, j}] (*{maxindex1,maxindex2}*);
```

In[ ]:=
```
Clear[croping]
croping[biosample_] := ImageData[
  ImageTake[Image[biosample], {Dimensions[biosample][[1]] - upperight[biosample][[1]] + 1,
    Dimensions[biosample][[1]] - lowerleft[biosample][[1]] + 1},
   {lowerleft[biosample][[2]], upperight[biosample][[2]]}]]
```

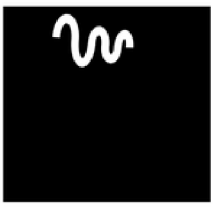
uploading the biological features and assigning them the biological data

uploadthefoldedproteinpattern

In[ ]:=
```
Clear[foldedProtein]
foldedProtein =
  Import["G:\\Haim\\SU 11 project\\papers\\Imaging\\My paper\\codes\\biological
     imaging\\biological features\\foldedProtein.JPG"];
```

In[ ]:=
```
temp = foldedProtein;
foldedProtein = ImageResize[temp, 150] (*ImageResize[temp,150]*)
```

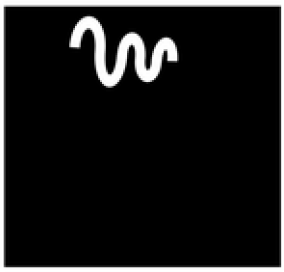
Out[ ]=



In[ ]:=
```
Clear[temp]
temp = foldedProtein;
foldedProtein = ImageData[ColorConvert[temp, "Grayscale"]];
```

In[ ]:=
```
Image[foldedProtein]
```

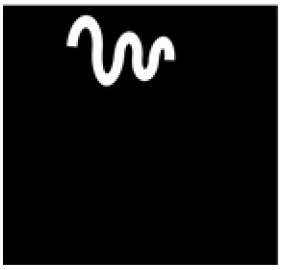
Out[ ]=



In[ ]:=
```
Clear[temp]
temp = croping[foldedProtein];
Clear[foldedProtein]
foldedProtein =
  Table[temp[[i, j]], {i, 1, Dimensions[temp][[1]]}, {j, 1, Dimensions[temp][[2]]}];
```

In[ ]:=
```
Image[foldedProtein]
```

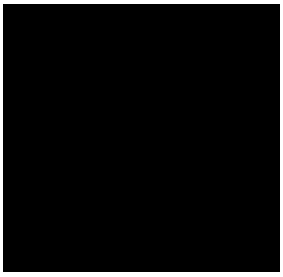
Out[ ]=



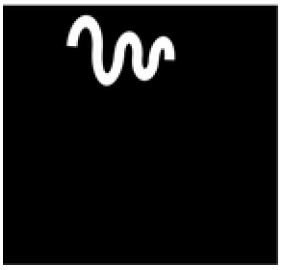
assign corresponding refractive coefficients to the folded protein pattern

In[ ]:=
```
Clear[temp]
temp = foldedProtein;
biosample01 = temp * phase[48.6, 32.9, 8.9, 8.9, 0.6, 0, (5 * 5 / 2) * 10^-9, 1.35, λsig];
Image[biosample01]
Image[biosample01 / Max[biosample01]]
```

Out[ ]=



Out[ ]=



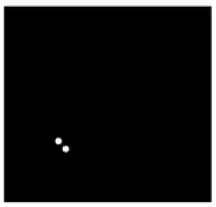
```
(*1. Import,Resize,and Convert to Grayscale*)
ClearAll[foldedProtein];
SetDirectory["G:\\Haim\\SU 11 project\\papers\\Imaging\\My
    paper\\codes\\biological imaging\\biological features"];

(*Import the image and resize it to 150 pixels width*)
foldedProtein = Import["foldedProtein.JPG"] // ImageResize[#, 150] &;

(*Convert to Grayscale and extract numerical ImageData for mathematical processing*)
foldedProteinData = ImageData[ColorConvert[foldedProtein, "Grayscale"]];

(*2. Dimension Checks (if needed for subsequent steps)*)
Print["Dimensions: ", Dimensions[foldedProteinData]];
(*Example:{143,150}*)

(*3. Cropping*)
(*Apply the pre-defined'croping' function and update the main variable*)
foldedProteinData = croping[foldedProteinData];

(*Print["Cropped Dimensions: ",Dimensions[foldedProteinData]];
Image[foldedProteinData] (*Display the cropped image*)
*)
(*4. Create the Bio Sample*)
(*Multiply the image data by the pre-defined'phase' function*)
biosample01 =
  foldedProteinData * phase[48.6, 32.9, 8.9, 8.9, 0.6, 0, (5 * 5 / 2) * 10^-9, 1.35, λsig];

(*Display the processed sample image*)
Image[biosample01]
(*Display the normalized image for better visualization*)
Image[biosample01 / Max[biosample01]]
```

uploadthe smallproteinpattern

In[ ]:=
```
Clear[twoSmallProteins]
twoSmallProteins =
  Import["G:\\Haim\\SU 11 project\\papers\\Imaging\\My paper\\codes\\biological
     imaging\\biological features\\twoSmallProteins.JPG"];
```

In[ ]:=
```
Clear[temp]
temp = twoSmallProteins;
twoSmallProteins = ImageResize[temp, 150] (*ImageResize[temp,150]*)
```

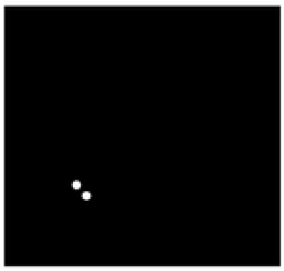
Out[ ]=



In[ ]:=
```
Clear[temp]
temp = twoSmallProteins;
twoSmallProteins = ImageData[ColorConvert[temp, "Grayscale"]];
```

In[ ]:=
```
Image[twoSmallProteins]
```

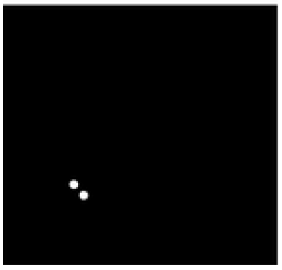
Out[ ]=



In[ ]:=
```
Clear[temp]
temp = croping[twoSmallProteins];
Clear[twoSmallProteins]
twoSmallProteins =
  Table[temp[[i, j]], {i, 1, Dimensions[temp][[1]]}, {j, 1, Dimensions[temp][[2]]}];
```

In[ ]:=
```
Image[twoSmallProteins]
```

Out[ ]=

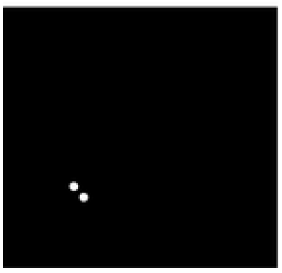


assign corresponding refractive coefficients to the small protein pattern

In[ ]:=
```
Clear[temp]
temp = twoSmallProteins;
biosample02 = temp * phase[48.6, 32.9, 8.9, 8.9, 0.6, 0, (3 * 5 / 2) * 10^-9, 1.35, λsig];
Image[biosample02]
Image[biosample02 / Max[biosample02]]
```

Out[ ]=

Out[ ]=

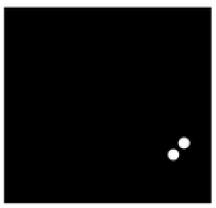


upload the large protein in pattern

In[ ]:=
```
Clear[twoLargeProteins]
twoLargeProteins =
  Import["G:\\Haim\\SU 11 project\\papers\\Imaging\\My paper\\codes\\biological
     imaging\\biological features\\twoLargeProteins.JPG"];
```

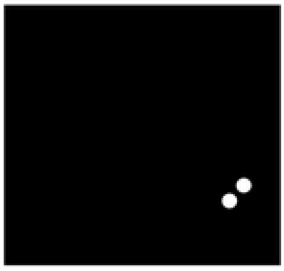
In[ ]:=
```
Clear[temp]
temp = twoLargeProteins;
twoLargeProteins = ImageResize[temp, 150] (*ImageResize[temp,150]*)
```

Out[ ]=



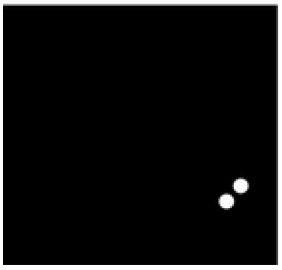
In[ ]:=
```
Clear[temp]
temp = twoLargeProteins;
twoLargeProteins = ImageData[ColorConvert[temp, "Grayscale"]];
```

In[ ]:=
```
Image[twoLargeProteins]
```
Out[ ]=



In[ ]:=
```
Clear[temp]
temp = croping[twoLargeProteins];
Clear[twoLargeProteins]
twoLargeProteins =
  Table[temp[[i, j]], {i, 1, Dimensions[temp][[1]]}, {j, 1, Dimensions[temp][[2]]}];
```

In[ ]:=
```
Image[twoLargeProteins]
```
Out[ ]=

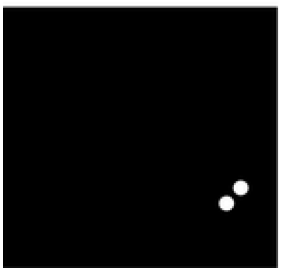


assign corresponding refractive coefficients to the large protein pattern

In[ ]:=
```
Clear[temp]
temp = twoLargeProteins;
biosample03 = temp * phase[48.6, 32.9, 8.9, 8.9, 0.6, 0, (5 * 5 / 2) * 10^-9, 1.35, λsig];
Image[biosample03]
Image[biosample03 / Max[biosample03]]
```

Out[ ]=

Out[ ]=



assign corresponding refractive coefficients to the lipid layer

In[ ]:=
```
lipid =
  Table[phase[62.5, 31.5, 0, 6.3, 0, 0, 45 * 10^-9, 1, λsig], {i, 1, 139}, {j, 1, 144}];
```

In[ ]:=
```
Image[lipid]
```

Out[ ]=

assign corresponding refractive coefficients to the lice cube

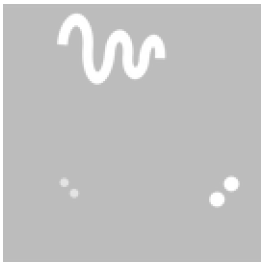
In[ ]:=
```
water = Table[phase[2, 0, 0, 1, 0, 0, 5 * 10^-6, 0.92, λsig], {i, 1, 139}, {j, 1, 144}];
```

In[ ]:=
```
Image[water]
```
Out[ ]=

creating the total biological sample

In[ ]:=
```
Clear[totalbiosample]
totalbiosample = biosample01 + biosample02 + biosample03 + lipid(*+water*);
```

In[ ]:=
```
Clear[totalbiosample]
totalbiosample = ImageData[%292];
Image[totalbiosample / Max[totalbiosample]]
```
Out[ ]=



In[ ]:=
```
Image[1 - (totalbiosample / Max[totalbiosample])]
```
Out[ ]=

In[ ]:=
```
Clear[totalimageBG]
totalimageBG = Table[phase[2, 0, 0, 1, 0, 0, 5 * 10^-6, 0.92, λsig],
   {i, 1, Dimensions[totalbiosample][[1]]}, {j, 1, Dimensions[totalbiosample][[2]]}]
 (* this is for 5 micron water *);
```

In[ ]:=
```
Clear[temp]
temp = totalbiosample;
totalbiosample = temp + totalimageBG;
```

In[ ]:=
```
Image[totalbiosample]
```
Out[ ]=

In[ ]:=
```
totalbiosample
```
Out[ ]=

```
{{0.605691, 0.605691, 0.605691, 0.605691, ... 126 ...,
  0.605691, 0.605691, 0.605691, 0.605691}, ... 132 ..., {... 1 ...}}
```

large output | show less | show more | show all | set size limit...

In[ ]:=
```
Image[totalbiosample / Max[totalbiosample]]
```
Out[ ]=

In[ ]:=
```
Image[1 - (totalbiosample / Max[totalbiosample])]
```
Out[ ]=

creating a matrix for the signal (“S”) angular deviations

the object pixel size must be smaller than the imaging resolution →

$$\frac{\text{FOV\_sig}}{\text{minimal resolution}} = \frac{10^{-6}}{10^{-9}} = 10^3 < \text{ number of object 1D pixels}$$

In[∘]:=
```
σθp0 = 4.13 * 10^-6
 (* σθp should be of the order of 0.77*objXpixel to achieve the Rayleigh limit *);
θp0 = -1.1556175944769262`;
ks = 4.559401012488846`*^10;
kp = 4.560414567875652`*^10;
ki = 2.446282748793892`*^7;
```

In[∘]:=
```
fs = 0.003;
```

In[∘]:=
```
res = fs kp σθp0 / (ks * Sec[θp0])
```

Out[∘]=

4.99866×10^{-9}

In[∘]:=
```
pixelsize = 0.5 * res (* the smallest separation in my image is 2 pixels than
   we need to multiply by 0.5 for that separation to correspond to 5 nm *)
```

Out[∘]=

2.49933×10^{-9}

In[∘]:=
```
angularPixel = 0.5 * res / fs
```

Out[∘]=

8.3311×10^{-7}

In[∘]:=
```
2 * angularPixel / σθp0
```

Out[∘]=

0.403443

In[∘]:=
```
angularPixel * (Dimensions[totalbiosample][[1]])
```

Out[∘]=

0.000111637

In[∘]:=
```
δsθxmin = -angularPixel * (Dimensions[totalbiosample][[1]]) / 2
δsθxmax = angularPixel * (Dimensions[totalbiosample][[1]]) / 2
```

Out[∘]=

-0.0000558184

Out[∘]=

0.0000558184

In[∘]:=
```
ScientificForm[δsθxmax]
```

Out[∘]//ScientificForm=

5.58184×10^{-5}

In[∘]:=
```
δsθymin = -angularPixel * (Dimensions[totalbiosample][[2]]) / 2
δsθymax = angularPixel * (Dimensions[totalbiosample][[2]]) / 2
```
Out[∘]=
```
-0.0000558184
```
Out[∘]=
```
0.0000558184
```

In[∘]:=
```
jmax = Dimensions[totalbiosample][[2]];
jmin = 1;
imax = Dimensions[totalbiosample][[2]];
imin = 1;
```

In[∘]:=
```
(δsθxmax - δsθxmin) / (imax - imin + 1)
```
Out[∘]=
8.3311×10^{-7}

In[∘]:=
```
objXpixel = (δsθxmax - δsθxmin) / (imax - imin + 1)
  (* this should TURN OUT TO BE EQUAL to angularPixel *)
```
Out[∘]=
8.3311×10^{-7}

In[∘]:=
```
N[objXpixel]
```
Out[∘]=
8.3311×10^{-7}

In[∘]:=
```
objYpixel = (δsθymax - δsθymin) / (jmax - jmin + 1)
  (* this should TURN OUT TO BE EQUAL to angularPixel *)
```
Out[∘]=
8.3311×10^{-7}

In[∘]:=
```
N[objYpixel]
```
Out[∘]=
8.3311×10^{-7}

In[∘]:=
```
δsθx = Range[δsθxmin * 0.995, δsθxmax * 0.995, objXpixel];
Dimensions[δsθx]
```
Out[∘]=
```
{134}
```

In[∘]:=
```
δsθy = Range[δsθymin * 0.995, δsθymax * 0.995, objYpixel];
Dimensions[δsθy]
```
Out[∘]=
```
{134}
```

In[∘]:=
```
N[129 * (5 * 10^-9) / 2] (*  this is the FOV of this image in meters *)
```
Out[∘]=
3.225×10^{-7}

creating a matrix for the idelr (“I”) angular

deviations

In[∘]:=
```
δiθxmin = δsθxmin * ks / ki
δiθxmax = δsθxmax * ks / ki
δiθymin = δsθymin * ks / ki
δiθymax = δsθymax * ks / ki
```

Out[∘]=
```
-0.104035
```

Out[∘]=
```
0.104035
```

Out[∘]=
```
-0.104035
```

Out[∘]=
```
0.104035
```

In[∘]:=
```
imageXpixel = (δiθxmax - δiθxmin) / (imax - imin + 1)
```

Out[∘]=
```
0.00155276
```

In[∘]:=
```
δiθx = Range[δiθxmin * 0.995, δiθxmax * 0.995, imageXpixel];
Dimensions[δiθx]
```

Out[∘]=
```
{134}
```

In[∘]:=
```
imageYpixel = (δiθymax - δiθymin) / (jmax - jmin + 1)
```

Out[∘]=
```
0.00155276
```

In[∘]:=
```
δiθy = Range[δiθymin * 0.995, δiθymax * 0.995, imageYpixel];
Dimensions[δiθy]
```

Out[∘]=
```
{134}
```

In[∘]:=
```
Dimensions[δiθy][[1]]
```

Out[∘]=
```
134
```

writing the input pump function

writing the matrix of the idler angular deviations based on the pump beam width

In[∘]:=
```
Clear[func]
```

$$\text{func}[\sigma\theta p\_, i\_, j\_, m\_, n\_] := \left(* \frac{1}{\sqrt{2*\pi}\ \sigma\theta p} **\right) e^{-\frac{\left(\left(ki\ \delta i\theta x[[m]] + ks\ \delta s\theta x[[i]]\right)^2 + \left(ki\ \delta i\theta y[[n]] + ks\ \delta s\theta y[[j]]\right)^2\right) \text{Sec}[\theta p\theta]^2}{kp^2}}{\sigma\theta p^2}}$$

writing the matrix of the idler image

part 1:

In[∘]:=
```
(*  nij = (#nPDC per unit area)*(area of the ENTIRE object)  *)
```

In[∘]:=
```
Dimensions[δiθy]
```

Out[∘]=
```
{134}
```

In[∘]:=
```
Clear[normalization]
normalization[σθp_] := ParallelTable[
   Sum[func[σθp, i, j, m, n], {i, 1, Dimensions[δsθx][[1]]}, {j, 1, Dimensions[δsθy][[1]]}],
   {m, 1, Dimensions[δiθx][[1]]}, {n, 1, Dimensions[δiθy][[1]]}];
```

In[∘]:=
```
Clear[sf]
sf[σθp_] := (1 / normalization[σθp]) *
   ParallelTable[Sum[(*2*nPDC/objarea**)Sin[totalbiosample[[i, j]] * func[σθp, i, j, m, n],
      {i, 1, Dimensions[δsθx][[1]]}, {j, 1, Dimensions[δsθy][[1]]}],
     {m, 1, Dimensions[δiθx][[1]]}, {n, 1, Dimensions[δiθy][[1]]}];
```

In[∘]:=
```
Image[Abs[1 - totalimageBG]]
Image[Abs[totalimageBG]]
Clear[sbg]
sbg[σθp_] := (1 / normalization[σθp]) *
   ParallelTable[Sum[(*2*nPDC/objarea**)Sin[totalimageBG[[i, j]] * func[σθp, i, j, m, n],
      {i, 1, Dimensions[δsθx][[1]]}, {j, 1, Dimensions[δsθy][[1]]}],
     {m, 1, Dimensions[δiθx][[1]]}, {n, 1, Dimensions[δiθy][[1]]}];
```

Out[∘]=

Out[∘]=

In[∘]:=
```
Clear[addΔtotal]
addΔtotal[q_, n1_, objarea_] :=
  ParallelTable[RandomVariate[NormalDistribution[0, Sqrt[2 * (1 + q) * n1/objarea]]],
   {m, 1, Dimensions[δiθx][[1]]}, {n, 1, Dimensions[δiθy][[1]]}] (* e^(-(x-μ)^2/(2σ^2)),
 x→noise_sf[[m,n]], μ[[m,n]]=0, σ^2=Δ2sf[[m,n]]+Δ2sbg[[m,n]]=2*nPDC/objarea +2*nPDC/objarea *);
```

part 2:

In[∘]:=
```
Clear[Sidler]
Sidler[σθp_] := sf[σθp] - sbg[σθp];
```

choosing Region Of Interest (ROI):

In[∘]:=
```
SidlerNσθp0 = Sidler[σθp0];
```

In[∘]:=
```
Image[SidlerNσθp0 / Max[Image[SidlerNσθp0]]]
```

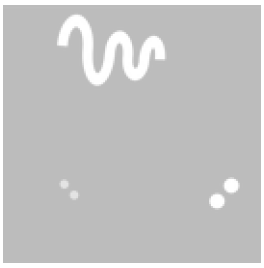
Out[∘]=

In[∘]:=
```
Clear[SidlerROIn]
SidlerROIn[i1_, i2_, j1_, j2_] :=
 ImageData[ImageTake[Image[SidlerNσθp0(*/Max[SidlerNσθp0]*)],
   {Dimensions[δiθx][[1]] - i2 + 1, Dimensions[δiθx][[1]] - i1 + 1},
   {Dimensions[δiθy][[1]] - j2 + 1, Dimensions[δiθy][[1]] - j1 + 1}]]
```

In[∘]:=
```
Clear[addΔtotalROIn]
addΔtotalROIn[q_, n1_, i1_, i2_, j1_, j2_] :=
 ImageData[ImageTake[Image[addΔtotal[q, n1, 1]],
   {Dimensions[δiθx][[1]] - i2 + 1, Dimensions[δiθx][[1]] - i1 + 1},
   {Dimensions[δiθy][[1]] - j2 + 1, Dimensions[δiθy][[1]] - j1 + 1}]]
```

In[∘]:=
```
Image[(totalbiosample - totalimageBG) / Max[totalbiosample - totalimageBG]]
```

Out[∘]=



In[∘]:=
```
Clear[i1Folded, i2Folded, j1Folded, j2Folded, rowMinFolded, rowMaxFolded]
```

In[∘]:=
```
rowMinFolded = 69;
rowMaxFolded = 134;

i1Folded = Dimensions[δiθx][[1]] - rowMaxFolded + 1
i2Folded = Dimensions[δiθx][[1]] - rowMinFolded + 1
j1Folded = 24; (* the most left column *)
```

In[∘]:=
```
Clear[foldedProteinData]
foldedProteinData = ImageData[ImageTake[Image[(totalbiosample - totalimageBG)],
    {i1Folded, i2Folded}, {j1Folded, j2Folded}]];

Image[foldedProteinData / Max[foldedProteinData]]
```

Out[∘]=

Fourier transforming the folded protein

In[∘]:=
```
Clear[SidlerROInFolded]
SidlerROInFolded = SidlerROIn[i1Folded, i2Folded, j1Folded, j2Folded];

Clear[qj]
qj = Table[q, {q, {1, 5, 25}}];

Clear[n1]
n1 = Table[NPDC, {NPDC, 10^5, 1 * 10^8, 1.999 * 10^6}];
```

In[∘]:=
```
Clear[a1folded]
a1folded = ParallelTable[Fourier[2 * n1[[j]] * Sqrt[qj[[i]]] * SidlerROInFolded +
    addΔtotalROIn[qj[[i]], n1[[j]], i1Folded, i2Folded, j1Folded, j2Folded],
   FourierParameters → {1, 1}], {i, 1, Dimensions[qj][[1]]}, {j, 1, Dimensions[n1][[1]]}]
```

Out[∘]=

{ … 1 … }

large output | show less | show more | show all | set size limit...

In[∘]:=
```
a1folded = ParallelTable[RotateRight[a1folded⟦i⟧⟦j⟧, Floor[
    ({Dimensions[foldedProteinData]⟦2⟧, Dimensions[foldedProteinData]⟦1⟧} - 1) / 2]],
  {i, 1, Dimensions[qj]⟦1⟧}, {j, 1, Dimensions[n1]⟦1⟧}]
```

Out[∘]=

{ … 1 … }

large output | show less | show more | show all | set size limit...

In[∘]:=
```
Clear[a2folded]
a2folded = ParallelTable[Fourier[2 * n1⟦j⟧ * Sqrt[qj⟦i⟧] * SidlerROInFolded +
    add△totalROIn[qj⟦i⟧, n1⟦j⟧, i1Folded, i2Folded, j1Folded, j2Folded],
   FourierParameters → {1, 1}], {i, 1, Dimensions[qj]⟦1⟧}, {j, 1, Dimensions[n1]⟦1⟧}]
```

Out[∘]=

{ … 1 … }

large output | show less | show more | show all | set size limit...

In[∘]:=
```
a2folded = ParallelTable[RotateRight[a2folded⟦i⟧⟦j⟧, Floor[
    ({Dimensions[foldedProteinData]⟦2⟧, Dimensions[foldedProteinData]⟦1⟧} - 1) / 2]],
  {i, 1, Dimensions[qj]⟦1⟧}, {j, 1, Dimensions[n1]⟦1⟧}]
```

Out[∘]=

… 1 …

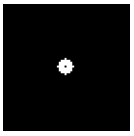
large output | show less | show more | show all | set size limit...

creating a search ring filter in the Fourier space

In[∘]:=
```
Clear[sign]
sign[i_, j_, min_, max_] := If[(i - (Floor[Dimensions[foldedProteinData]⟦1⟧ / 2]))^2 +
      (j - (Floor[Dimensions[foldedProteinData]⟦2⟧ / 2]))^2 ≥ min^2 &&
    (i - (Floor[Dimensions[foldedProteinData]⟦2⟧ / 2]))^2 +
      (j - (Floor[Dimensions[foldedProteinData]⟦2⟧ / 2]))^2 ≤ max^2, 1, 0]
```

In[∘]:=
```
Clear[pp]
pp = Table[0, {i, 1, Dimensions[foldedProteinData]⟦1⟧},
    {j, 1, Dimensions[foldedProteinData]⟦2⟧}];

For[i = 1, i ≤ Dimensions[foldedProteinData]⟦1⟧,
 i++, For[j = 1, j ≤ Dimensions[foldedProteinData]⟦2⟧,
  j++, pp⟦i, j⟧ = (*ppp⟦i,j⟧**)sign[i, j, 1, 4]]]
```

In[ ]:=
```
Image[pp]
```
Out[ ]=



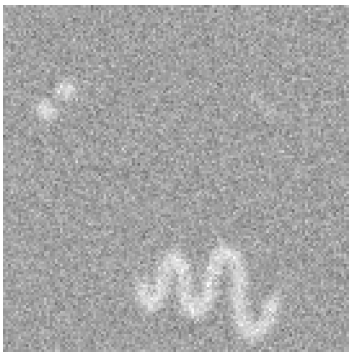
In[ ]:=
```
Clear[numpix, halfBit]
```

In[ ]:=
```
numpix[min_, max_] := Floor[Pi * (max^2 - min^2)];
halfBit[min_, max_] :=
  (0.2071 + 1.9102 / (√numpix[min, max])) / (1.2071 + 0.97102 / (√numpix[min, max]));

halfBitTable = Table[halfBit[min, min + Δr],
   {min, 1, Floor[Dimensions[foldedProteinData][[2]] / 2]}, {Δr, 1, 4}];
```

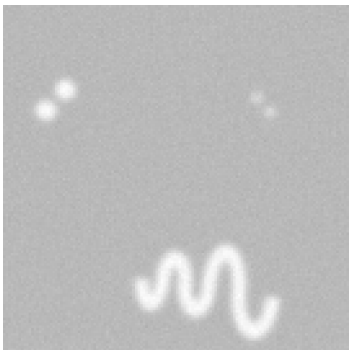
plotting at different exposures

In[ ]:=
```
NPDCs = Table[NPDC, {NPDC, 5 * 10^6, 5 * 10^8, 7.07143 * 10^7}];
```

In[ ]:=
```
image1 = NPDCs[[1]] * SidlerNσθp0 + addΔtotalROIn[NPDCs[[1]],
    1, Dimensions[SidlerNσθp0][[1]], 1, Dimensions[SidlerNσθp0][[2]]];
Image[image1 / Max[image1]]
```
Out[ ]=

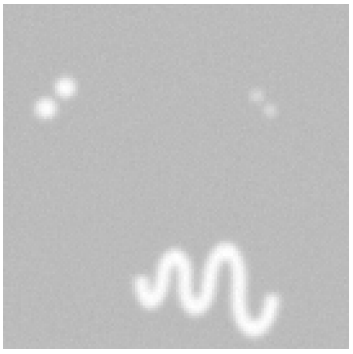


In[ ]:=
```
image4 = NPDCs[[4]] * SidlerNσθp0 + addΔtotalROIn[NPDCs[[4]],
    1, Dimensions[SidlerNσθp0][[1]], 1, Dimensions[SidlerNσθp0][[2]]];
Image[image4 / Max[image4]]
```
Out[ ]=



In[∘]:=
```
image7 = NPDCs[[7]] * SidlerNσθp0 + addΔtotalROIn[NPDCs[[7]],
    1, Dimensions[SidlerNσθp0][[1]], 1, Dimensions[SidlerNσθp0][[2]]];
Image[image7 / Max[image7]]
```

Out[∘]=



showing the ROI image of the the proteins:

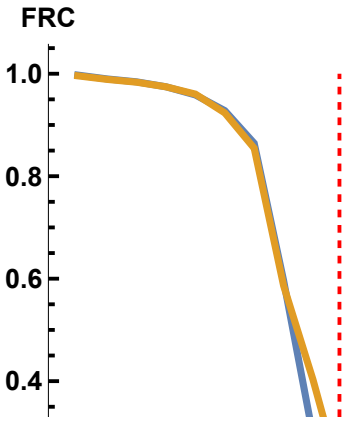
In[∘]:=
```
Image[SidlerROIn[i1Folded, i2Folded, j1Folded, j2Folded] /
  Max[SidlerROIn[i1Folded, i2Folded, j1Folded, j2Folded]]]
```

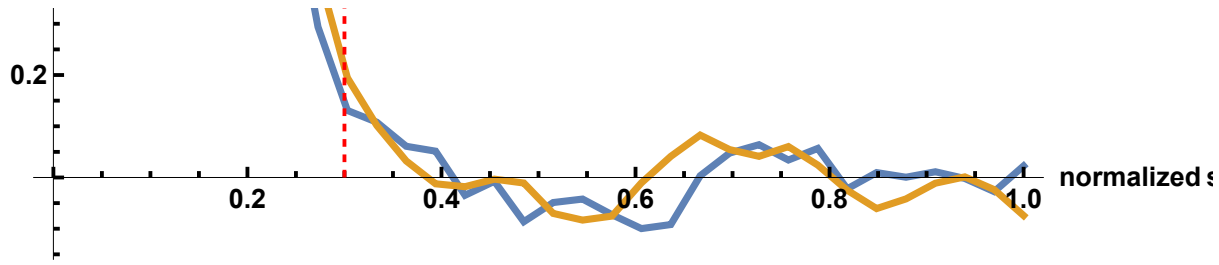
Out[∘]=

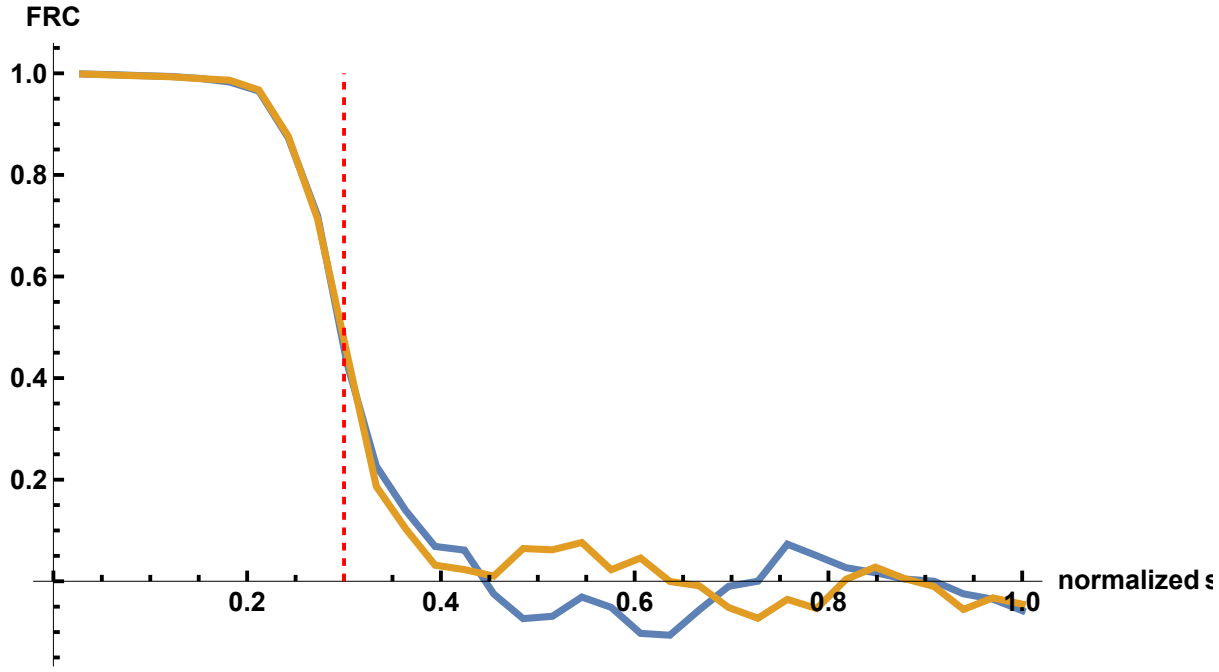
FRC of the folded protein:

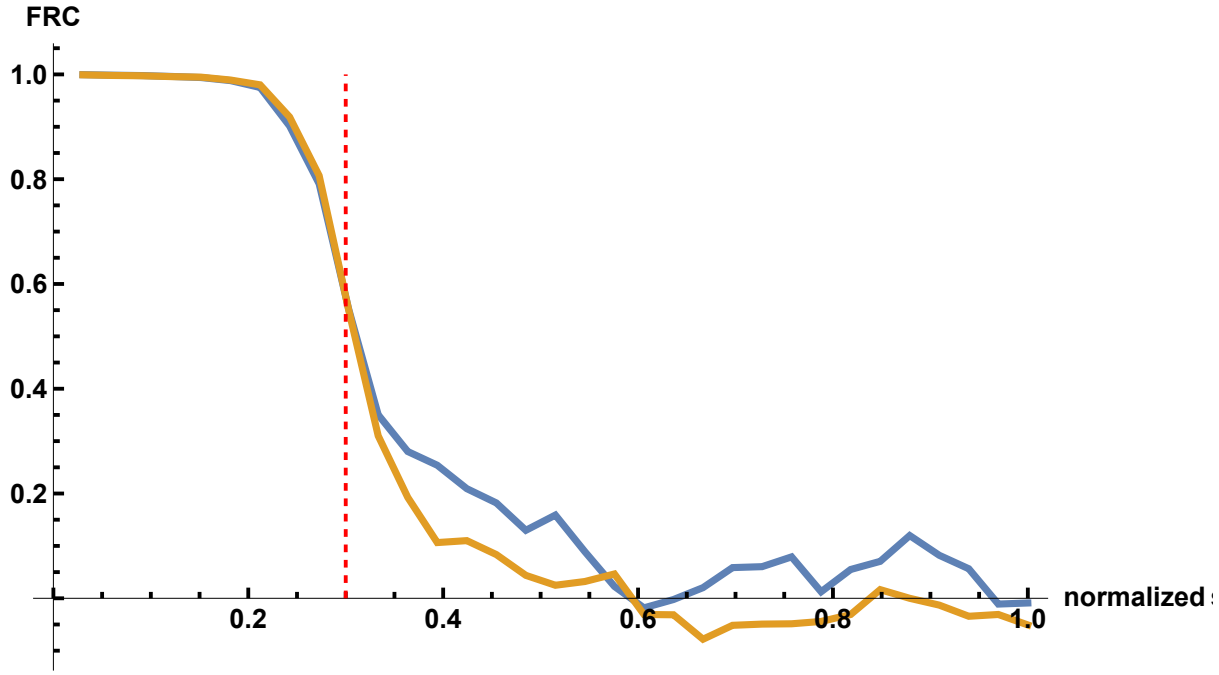
```
Clear[p1, p2]
p1 = {50, 29, 8};
p2 = {30, 18, 5};

Clear[Δr]
Δr = 3;
```

```
Table[Clear[frcDatan1, frcDatanq];
 frcDatan1 = ParallelTable[
   (Sum[Re[a1folded[[1]][[p1[[k1]]]][[i, j]] * Conjugate[a2folded[[1]][[p1[[k1]]]][[i, j]]] *
        sign[i, j, min, min + Δr]], {i, 1, Dimensions[foldedProteinData][[1]]},
      {j, 1, Dimensions[foldedProteinData][[2]]}]) /
    (Sqrt[(Sum[sign[i, j, min, min + Δr] * (Abs[a1folded[[1]][[p1[[k1]]]][[i, j]]])^2,
          {i, 1, Dimensions[foldedProteinData][[1]]},
          {j, 1, Dimensions[foldedProteinData][[2]]}]) *
        (Sum[sign[i, j, min, min + Δr] * (Abs[a2folded[[1]][[p1[[k1]]]][[i, j]]])^2,
          {i, 1, Dimensions[foldedProteinData][[1]]},
          {j, 1, Dimensions[foldedProteinData][[2]]}])]),
   {min, 1, Floor[Dimensions[foldedProteinData][[2]] / 2]}];

 frcDatanq = ParallelTable[
   (Sum[Re[a1folded[[2]][[p2[[k2]]]][[i, j]] * Conjugate[a2folded[[2]][[p2[[k2]]]][[i, j]]] *
        sign[i, j, min, min + Δr]], {i, 1, Dimensions[foldedProteinData][[1]]},
      {j, 1, Dimensions[foldedProteinData][[2]]}]) /
    (Sqrt[(Sum[sign[i, j, min, min + Δr] * (Abs[a1folded[[2]][[p2[[k2]]]][[i, j]]])^2,
          {i, 1, Dimensions[foldedProteinData][[1]]},
          {j, 1, Dimensions[foldedProteinData][[2]]}]) *
        (Sum[sign[i, j, min, min + Δr] * (Abs[a2folded[[2]][[p2[[k2]]]][[i, j]]])^2,
          {i, 1, Dimensions[foldedProteinData][[1]]},
          {j, 1, Dimensions[foldedProteinData][[2]]}])]),
   {min, 1, Floor[Dimensions[foldedProteinData][[2]] / 2]}];

 Show[
  ListPlot[{Table[{i / (Floor[Dimensions[foldedProteinData][[2]] / 2]), frcDatan1[[i]]},
      {i, 1, Floor[Dimensions[foldedProteinData][[2]] / 2]}],
     Table[{i / (Floor[Dimensions[foldedProteinData][[2]] / 2]), frcDatanq[[i]]},
      {i, 1, Floor[Dimensions[foldedProteinData][[2]] / 2]}]}, Joined → True,
   PlotStyle → Thickness[0.007], LabelStyle → {13, GrayLevel[0], Bold},
   TicksStyle → Thick],
  AxesLabel → {HoldForm[normalized spatial frequency[1/(2.5 nm)]], HoldForm[FRC]},
  PlotLabel → None, LabelStyle → {14, GrayLevel[0], Bold},
  Epilog → {Dashed, Red, AbsoluteThickness[2], Line[{{0.3, 0}, {0.3, 1}}]}]
 , {k1, 1, Length[p1]}, {k2, 1, Length[p2]}]
```









In[ ]:= `minF`

Out[ ]=

```
{13, 14, 15, 16, 18}
```

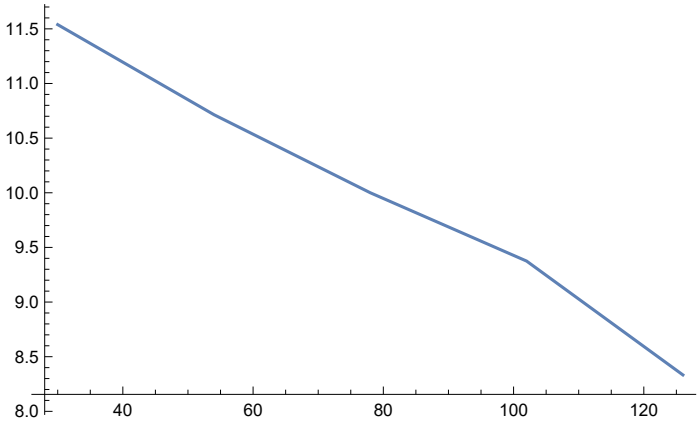
In[∘]:=
```
bla = {1 / 13, 1 / 14, 1 / 15, 1 / 16, 1 / 18}
```
Out[∘]=

$$\left\{\frac{1}{13}, \frac{1}{14}, \frac{1}{15}, \frac{1}{16}, \frac{1}{18}\right\}$$

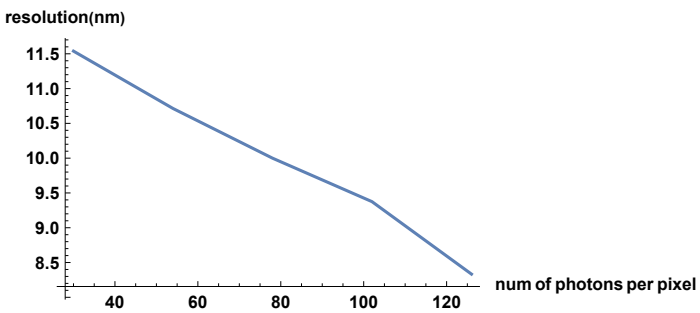
In[∘]:=
```
bla2 = Table[NPDC, {NPDC, 30, 140, 24}]
```
Out[∘]=

{30, 54, 78, 102, 126}

In[∘]:=
```
ListPlot[
 ParallelTable[{bla2[[i]], (5(**10^-9*)) / (1 / 30) * bla[[i]]}, {i, 1, 5}], Joined → True]
```



In[∘]:=
```
Show[%111, AxesLabel → {HoldForm[num of photons per pixel], HoldForm[resolution[nm]]},
 PlotLabel → None, LabelStyle → {GrayLevel[0], Bold}]
```
Out[∘]=



In[∘]:=
```
1 / ((1 / 18) * (5) / (1 / 30))
```
Out[∘]=

$$\frac{3}{25}$$

In[∘]:=
```
(1 / 18) * (5) / (1 / 30)
```
Out[∘]=

$$\frac{25}{3}$$

In[∘]:=
```
N[1 / ((1 / 18) * (5) / (1 / 30))]
```
Out[∘]=

0.12

In[∘]:=
```
0.12 * 10^9
```
Out[∘]=

1.2×10^{8}

In[∘]:=
```
N[(1 / 18) * (5) / (1 / 30)]
```
Out[∘]=

8.33333

In[∘]:=
```
N[(1 / 18) * (5 * 10^-9) / (1 / 30)]
```
Out[∘]=

8.33333×10^{-9}

In[∘]:=
```
frcMinusBit =
 Table[(Re[(Sum[a1⟦5⟧⟦i, j⟧ * Conjugate[a2⟦5⟧⟦i, j⟧] * sign[i, j, min, min + Δr],
        {i, 1, 61}, {j, 1, 61}])] / (Sqrt[
       (Sum[sign[i, j, min, min + Δr] * (Abs[a1⟦5⟧⟦i, j⟧])^2, {i, 1, 61}, {j, 1, 61}]) *
        (Sum[sign[i, j, min, min + Δr] * (Abs[a2⟦5⟧⟦i, j⟧])^2, {i, 1, 61}, {j, 1, 61}])]))
   (*-0.5*halfBit[min,min+Δr]*), {min, 1, 30}];
ListPlot[Table[{i, frcMinusBit⟦i⟧}, {i, 1, 30}], Joined → True]
```
Out[∘]=