T1000179 - v20

```
MaxDefocusAtITMAll = 2.1 * 10^-6;
```

H1-ITMX Vertex Hartmann Sensor

Optical Layout: [AirLens = 1.6994m, VacLens = -1.6994m]@ 820nm

Requirements

1. Image limiting aperture
2. Magnification = 1/20 x
3. Reference beam with own imaging device
4. Determine temperature sensitivity
5. Tolerancing on imaging optics

Get the Zemax Data - Import Units = mm, Convert to m

Optical Layout - X Arm

Requirements

1. HWS = Conjugate Plane of BS
2. Beam Size at ITM = Sqrt[2]*ModeSize
3. Magnification = 17.5x
4. Put the telescope outside the vacuum - Can't do it all
5. Beam is approximately collimated at the HWS

Set probe beam size at ITM = 17.5*Diameter_of_CCD/SQRT[2]

```
BeamSizeAtITM = 127 / 1000
```

$$\frac{127}{1000}$$

```
nFusedSilica840 = 1.445;
CPthick = 130 * ConvertToM;
n1 = 1.0;
TotalDeMagnification = -17.5;
```

Create Function to Determine Output Radius and Beam Size

```
OutRW[wIN_, RIN_, λIN_, ABCD_] := Module[{},
   Q1Inv = 1/RIN - ⅈ λIN/(Pi wIN^2);
   Q1a = 1 / Q1Inv;
   Q2temp = (ABCD[[1]][[1]] Q1a + ABCD[[1]][[2]])/(ABCD[[2]][[1]] Q1a + ABCD[[2]][[2]]);
   Q2Inv = 1 / Q2temp;
   ROut = Re[Q2Inv]^-1;
   wOut = Sqrt[-λIN/(Pi Im[Q2Inv])];
   Return[{ROut, wOut}];
  ];
FindNearestWaist[wIN_, RIN_, λIN_] := Module[{},
   Q1Inv = 1/RIN - ⅈ λIN/(Pi wIN^2);
   Q1a = 1 / Q1Inv;


   Q2temp = Q1a;
   Q2tempR = Re[Q2temp] + x0;
   Q2tempI = Im[Q2temp];
   Q2InvR = Q2tempR/(Q2tempR^2 + Q2tempI^2);


   ROut = Q2InvR^(-1);

   solnIN = Solve[1 / ROut == 0, x0];


   Return[x0 /. solnIN[[1]]];
  ];
```

Optical Path - determine solution: AirLens = 1.6994m, VacLens = -1.6994m

Given Physical Parameters

Distances

```
ITMXPOtoSR3 = Norm[SR3XYZ - ITMXPOXYZ]
SR3toITMX = Norm[SR3XYZ - BSXYZ] + Norm[ITMXXYZ - BSXYZ] -
  2 * BSRRT[[2]] + (nFusedSilica840 - 1) * (ITMYRRT[[2]] + CPthick)
SR3toBS = Norm[SR3XYZ - BSXYZ]
```

```
15.1729
```

```
24.663
```

```
19.4313
```

Curvatures - E0900095, T080268, E080518

```
ITMRoC = 1934 / nFusedSilica840;
SR3RoC = 36;
```

ABCD Matrices - not relying on solution

```
Requirement 2: The mode from the ITM, including the size of Sqrt[2]*Cavity-
ModeSize is defined below
```

Test Gaussian beam in and out

```
MFromITM = {{1, ITMXPOtoSR3}, {0, 1}} . {{1, 0}, {-2/SR3RoC, 1}} . {{1, SR3toITMX}, {0, 1}};
MToITM = {{1, SR3toITMX}, {0, 1}} . {{1, 0}, {-2/SR3RoC, 1}} . {{1, ITMXPOtoSR3}, {0, 1}};

wIN = BeamSizeAtITM;
RIN = ITMRoC;
pIn = {RIN, wIN}
λIN = 840 * 10^-9;

pOut1 = OutRW[wIN, RIN, λIN, MFromITM];
pOut2 = OutRW[pOut1[[2]], -pOut1[[1]], λIN, MToITM];
Print["Beam size at ITM = ", wIN * 1000, " mm"];
Print["Beam size at ITMXPO = ", pOut1[[2]] * 1000, " mm"];
Print["Beam size of retro-reflected ITMXPO beam at ITM = ",
  pOut2[[2]] * 1000, " mm"];
```

$$\left\{1338.41,\ \frac{127}{1000}\right\}$$

```
Beam size at ITM = 127 mm

Beam size at ITMXPO = 21.7538 mm

Beam size of retro-reflected ITMXPO beam at ITM = 127. mm
```

ITM to ITMXPO: From HR surface outside ITM to AR surface outside ITMXPO [in m]

```
ITMToITMXPO = {{1, ITMXPOtoSR3}, {0, 1}} . {{1, 0}, {-2/SR3RoC, 1}} . {{1, SR3toITMX}, {0, 1}};
MatrixForm[ITMToITMXPO]
```

$$\begin{pmatrix} 0.157058 & 19.0465 \\ -0.0555556 & -0.370166 \end{pmatrix}$$

BS to ITMXPO: ABCD Matrix - From BS to ITMXPO

```
BSToITMXPO = {{1, ITMXPOtoSR3}, {0, 1}} . {{1, 0}, {-2/SR3RoC, 1}} . {{1, SR3toBS}, {0, 1}};
MatrixForm[ITMToITMXPO]
```

$$\begin{pmatrix} 0.157058 & 19.0465 \\ -0.0555556 & -0.370166 \end{pmatrix}$$

Solve for the best solution

Extract Optical Information - Conjugate plane, beam size at ITMXPO

Get Output Mode from ITMXPO

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840 * 10^-9;
OutRW[wIN, RIN, λIN, {{1, 100}, {0, 1}}];
pOut = OutRW[wIN, RIN, λIN, ITMToITMXPO];
OutR = pOut[[1]];
OutW = pOut[[2]];
OutModeITMXPO = pOut;
FindNearestWaist[pOut[[2]], pOut[[1]], λIN];
```

Solve::ratnz: Solve was unable to solve the system with inexact coefficients
The answer was obtained by solving a corresponding exact system and numericizing the result >>

```
Must create mode - matching optics to produce this mode
Must create imaging optics to image the limiting aperture - the beam split-
ter - correctly
```

```
Print["Probe beam size at ITMXPO = ", OutW * 1000, " mm."]
Print["Curvature at ITMXPO = ", OutR * 1000, " mm."]
```

```
Probe beam size at ITMXPO = 21.7538 mm.

Curvature at ITMXPO = -3067.94 mm.
```

Demonstrate curvature sign by propagating a short distance further

```
pOutdz = OutRW[wIN, RIN, λIN, {{1, 0.1}, {0, 1}}.ITMToITMXPO];
OutRdz = pOutdz[[1]];
OutWdz = pOutdz[[2]];
Print["Probe beam size at ITMXPO = ", OutW * 1000, " mm."]
Print["Probe beam size 100mm out of IFO from ITMXPO = ", OutWdz * 1000, " mm."]
```

```
Probe beam size at ITMXPO = 21.7538 mm.
```

```
Probe beam size 100mm out of IFO from ITMXPO = 21.0447 mm.
```

Analytic Solution for imaging a conjugate plane with an arbitrary telescope

Solve for the solution - ITMX: Magnification = +ve!!

ABCD Matrices - dependent on solution

Do a Gaussian beam propagation

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOut = OutRW[wIN, RIN, λIN, ITMToHWS];
OutR = pOut[[1]];
OutW = pOut[[2]];
GaussianMag = wIN / OutW;
Print["Gaussian beam size DeMagnification = ", GaussianMag, "x"]
Print["Beam RoC at HWS = ", OutR, "m"]
```

```
Gaussian beam size DeMagnification = 17.5x
```

```
Beam RoC at HWS = 4.37031m
```

```
Requirement 3: The beam size at the HWS is 17.43x smaller than at the beam
splitter
Requirement 5: The beam is approximately flat at the HWS
```

Define the Input Mode

```
InputMode = {OutW, -OutR}
InputModeFromHWS = InputMode;
```

```
{0.00725714, -4.37031}
```

Propagate modes - from ITM to outside and then back again

```
Propagate the ITM mode from the ITM to the HWS
```

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
Print["Probe beam size at ITM = ", wIN * 1000, " mm."]
Print["Curvature at ITM = ", RIN, " m."]
pOut = OutRW[wIN, RIN, λIN, ITMToHWS];
Print["Probe beam size at HWS = ", pOut[[2]] * 1000, " mm."]
Print["Curvature at HWS = ", pOut[[1]] * 1000, " mm."]
Print["ITM_Beam_Size/HWS_Beam_Size = ", wIN / pOut[[2]], " x."]
```

```
Probe beam size at ITM = 127 mm.

Curvature at ITM = 1338.41 m.

Probe beam size at HWS = 7.25714 mm.

Curvature at HWS = 4370.31 mm.

ITM_Beam_Size/HWS_Beam_Size = 17.5 x.
```

Propagate the HWS mode to the ITM

```
pOut = OutRW[pOut[[2]], -pOut[[1]], λIN, HWSToITM];
Print["Probe beam size at ITM = ", pOut[[2]] * 1000, " mm."]
Print["Curvature at ITM = ", pOut[[1]], " m."]
```

```
Probe beam size at ITM = 127. mm.

Curvature at ITM = -1338.41 m.
```

Propagate from HWS mode to HWS

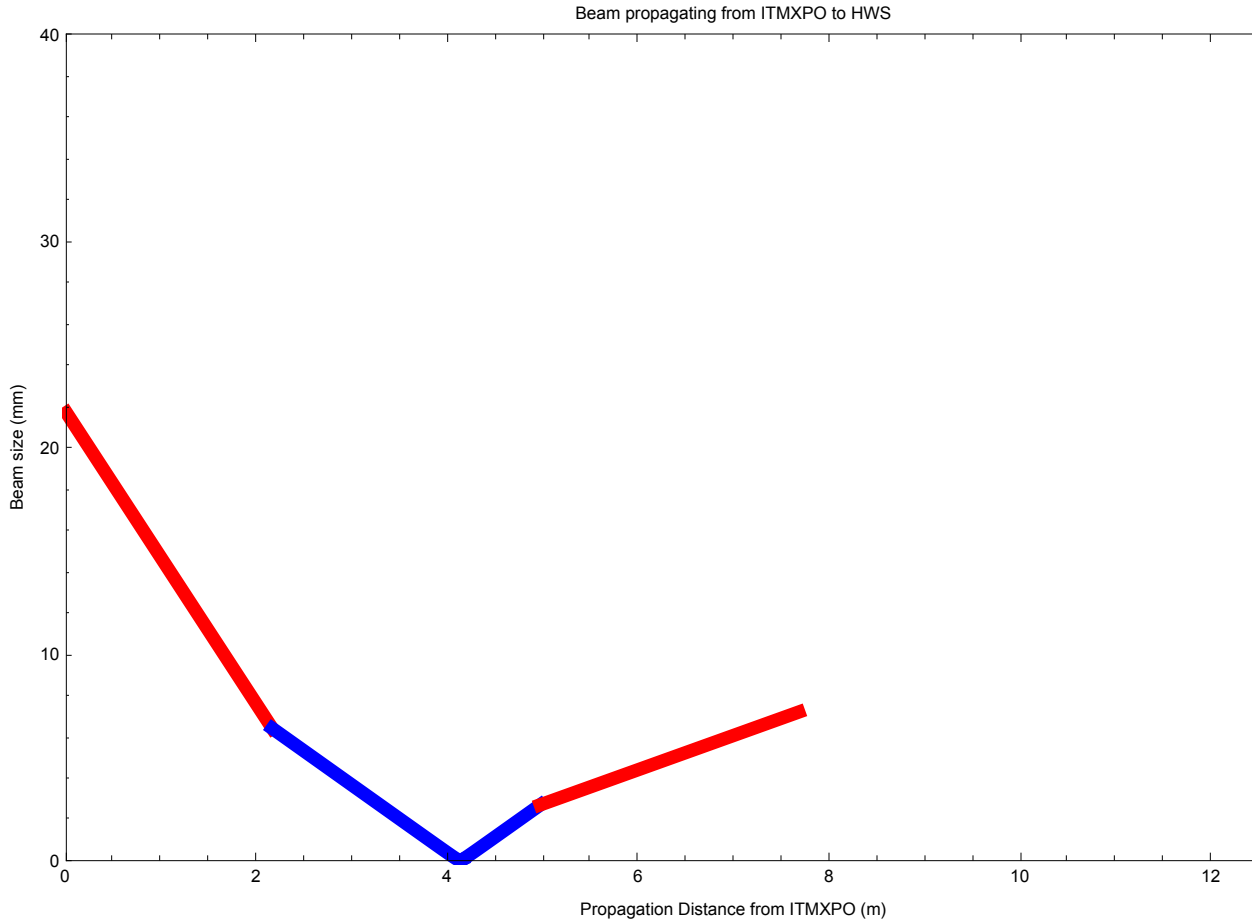
```
Print["Probe beam size at ITM = ", InputMode[[1]] * 1000, " mm."]
Print["Curvature at ITM = ", InputMode[[2]], " m."]
pOut = OutRW[InputMode[[1]], InputMode[[2]], λIN, MHWSv];
Print["Probe beam size at HWS = ", pOut[[2]] * 1000, " mm."]
Print["Curvature at HWS = ", pOut[[1]], " m."]
```

```
Probe beam size at ITM = 7.25714 mm.

Curvature at ITM = -4.37031 m.

Probe beam size at HWS = 7.25714 mm.

Curvature at HWS = 4.37031 m.
```

Plot the Solution

Plot solution from ITMXPO to HWS

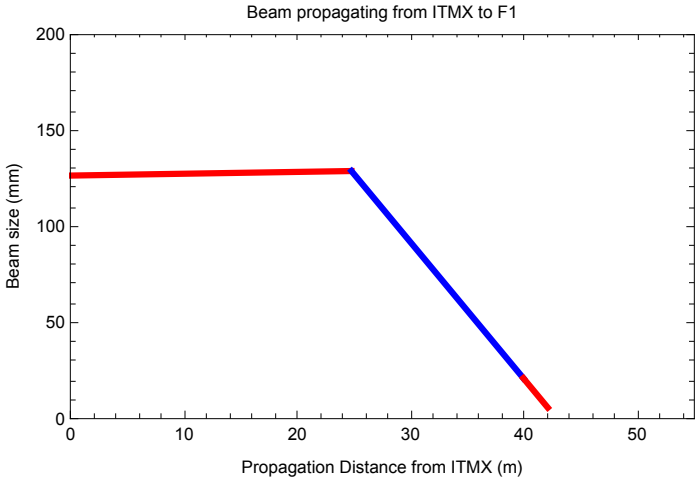
```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;

p1 = Plot[(OutRW[wIN, RIN, λIN, {{1, x}, {0, 1}}.ITMToITMXPO])[[2]] * 1000, {x, 0, Z1A},
   PlotStyle → {Red, Thickness[0.01]}, PlotRange → {{0, 14}, {0, 40}}];
BS1 = Table[{x, (OutRW[wIN, RIN, λIN, {{1, x}, {0, 1}}.ITMToITMXPO])[[2]] * 1000},
   {x, 0, Z1A, Z1A / 200}];

p2 =
  Plot[(OutRW[wIN, RIN, λIN, {{1, x - Z1A}, {0, 1}}.{{1, 0}, {-1 / F1A, 1}}.{{1, Z1A}, {0, 1}}.ITMToITMXPO])[[2]] *
    1000, {x, Z1A, Z1A + Z2A}, PlotStyle → {Blue, Thickness[0.01]}];
BS2 = Table[{x, (OutRW[wIN, RIN, λIN, {{1, x - Z1A}, {0, 1}}.{{1, 0}, {-1 / F1A, 1}}.{{1, Z1A}, {0, 1}}.
         ITMToITMXPO])[[2]] * 1000}, {x, Z1A, Z1A + Z2A, (Z2A / 200)}];
p3 = Plot[(OutRW[wIN, RIN, λIN, {{1, x - Z1A - Z2A}, {0, 1}}.{{1, 0}, {-1 / F2A, 1}}.
        {{1, Z2A}, {0, 1}}.{{1, 0}, {-1 / F1A, 1}}.{{1, Z1A}, {0, 1}}.ITMToITMXPO])[[2]] * 1000,
   {x, Z1A + Z2A, Z1A + Z2A + Z3A}, PlotStyle → {Red, Thickness[0.01]}];

BS3 = Table[
   {x, (OutRW[wIN, RIN, λIN, {{1, x - Z1A - Z2A}, {0, 1}}.{{1, 0}, {-1 / F2A, 1}}.{{1, Z2A}, {0, 1}}.{{1, 0}, {-1 / F1A, 1}}.
         {{1, Z1A}, {0, 1}}.ITMToITMXPO])[[2]] * 1000},
   {x, Z1A + Z2A, Z1A + Z2A + Z3A, Z3A / 200}];
ITMXPlottedSolutionFromITMXPO = Show[p1, p2, p3, Frame → True,
  FrameLabel → {"Propagation Distance from ITMXPO (m)",
    "Beam size (mm)", "Beam propagating from ITMXPO to HWS"}]
```




```
BS = BS1;
For[ii = 1, ii ≤ Length[BS2], ii++, {
    BS = Append[BS, BS2[[ii]]];
  }];
For[ii = 1, ii ≤ Length[BS3], ii++, {
    BS = Append[BS, BS3[[ii]]];
  }];
Export["/Advanced_LIGO/TCS/Hartmann sensor/H1_HWSX_beam_size.txt", BS, "Table"];
```

Plot from ITMX to ITMXPO

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
plotR = {{0, 55}, {0, 200}};
p1 = Plot[{OutRW[wIN, RIN, λIN, {{1, x}, {0, 1}}]}[[2]] * 1000,
    {x, 0, SR3toITMX}, PlotStyle → {Red, Thickness[0.01]}, PlotRange → plotR];
p2 = Plot[{OutRW[wIN, RIN, λIN, {{1, x - SR3toITMX}, {0, 1}}] .
          {{1, 0}, {-2 / SR3RoC, 1}} . {{1, SR3toITMX}, {0, 1}}}][[2]] * 1000,
    {x, SR3toITMX, SR3toITMX + ITMXPOtoSR3}, PlotStyle → {Blue, Thickness[0.01]}];
p3 = Plot[{OutRW[wIN, RIN, λIN, {{1, x - SR3toITMX - ITMXPOtoSR3}, {0, 1}}] .ITMToITMXPO}][[2]] *
     1000, {x, SR3toITMX + ITMXPOtoSR3, SR3toITMX + ITMXPOtoSR3 + Z1A},
    PlotStyle → {Red, Thickness[0.01]}, PlotRange → plotR ];
ITMXPlottedSolutionFromITMX = Show[p1, p2, p3, Frame → True,
  FrameLabel → {"Propagation Distance from ITMX (m)",
    "Beam size (mm)", "Beam propagating from ITMX to F1"}]
```



Tolerancing on X Arm solution

Get the Variable matrix

$$\text{SR3toHWSVar} = \begin{pmatrix} 1 & Z3A + dz3 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1 / F2A - (dSf2) & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & Z2A + dz2 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1 / F1A - (dSf1) & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & Z1A + dz1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & ITMXPOtoSR3 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ \frac{-2}{SR3RoC} & 1 \end{pmatrix};$$

$$\text{BStoHWSVar} = \text{SR3toHWSVar} . \begin{pmatrix} 1 & SR3toBS \\ 0 & 1 \end{pmatrix};$$

$$\text{ITMXtoHWSVar} = \text{SR3toHWSVar} . \begin{pmatrix} 1 & SR3toITMX \\ 0 & 1 \end{pmatrix};$$

Variation in imaging (B = 0) with length changes: (assume that FL variations can

be compensated for) - NO ISSUES - Easy to work with!

```
M1 = BStoHWSVar.{{1, x}, {0, 1}};
B1 = M1[[1]][[2]];
```

Test variation in position of F1 - solve for the position of the conjugate plane - INSENSITIVE

Set all delta values = 0 except for dz2 and dz1. Set these last two to the same value with opposite sign to model a lens out of position

```
B1A = Simplify[B1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → X1, dz1 → -X1}];
Solve[B1A == 0, x];
ConjPlaneDisplacement = x /. %[[1]];
NominalConjugatePlane = ConjPlaneDisplacement /. X1 → 0;
Print["Conjugate Plane is nominally out of position by ",
  NominalConjugatePlane, "m"];
RateCPD = (D[ConjPlaneDisplacement, X1]) /. X1 → 0;
Print["Conjugate Plane displacement for +/- 5 mm displacement of F1 = ",
  RateCPD * 1000 * 0.005, "mm"];
BTestF1OutOfPosition = {RateCPD * 1000 * 0.005, "mm", 5, "mm"};
```

```
Conjugate Plane is nominally out of position by 5.23168m
```

```
Conjugate Plane displacement for +/- 5 mm displacement of F1 = -536.785mm
```

Hence the conjugate plane is mostly invariant to the position of F1

Test variation in position of F2 - solve for the position of the conjugate plane - INSENSITIVE to distance between F1 and F2

Set all delta values = 0 except for dz2 and dz3. Set these last two to the same value with opposite sign to model a lens out of position

```
B1A = Simplify[B1 /. {dz1 → 0, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz3 → X1}];
Solve[B1A == 0, x];
ConjPlaneDisplacement = x /. %[[1]];
NominalConjugatePlane = ConjPlaneDisplacement /. X1 → 0;
Print["Conjugate Plane is nominally out of position by ",
  NominalConjugatePlane, "m"];
RateCPD = (D[ConjPlaneDisplacement, X1]) /. X1 → 0;
Print["Conjugate Plane displacement for +/- 5 mm displacement of F1 = ",
  RateCPD * 0.005, "m"];
BTestF2OutOfPosition = {RateCPD * 1000 * 0.005, "mm", 5, "mm"};
```

```
Conjugate Plane is nominally out of position by 5.23168m
```

```
Conjugate Plane displacement for +/- 5 mm displacement of F1 = -0.99378m
```

Test variation in F1-F2 distance - INSENSITIVE

```
B1A = Simplify[B1 /. {dz1 → 0, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz3 → 0}];
Solve[B1A == 0, x];
ConjPlaneDisplacement = x /. %[[1]];
NominalConjugatePlane = ConjPlaneDisplacement /. X1 → 0;
Print["Conjugate Plane is nominally out of position by ",
  NominalConjugatePlane, "m"];
RateCPD = (D[ConjPlaneDisplacement, X1]) /. X1 → 0;
Print["Conjugate Plane displacement for +/- 5 mm variation
    in distance between F1 and F2 = ", RateCPD * 0.005, "m"];
BTestF1F2Distance = {RateCPD * 1000 * 0.005, "mm", 5, "mm"};
```

```
Conjugate Plane is nominally out of position by 5.23168m
```

```
Conjugate Plane displacement for +/- 5 mm variation in distance between F1 and F2 =
 0.53747m
```

Hence the conjugate plane is totally insensitive to the distance between F2 and F1

Test variation in position of HWS - SENSITIVE = GOOD for tuning. Possibly bad for fringes

Set all delta values = 0 except for dz3.

```
B1A = Simplify[B1 /. {dz1 → 0, dSf2 → 0, dSf1 → 0, dz2 → 0, dz3 → X1}];
Solve[B1A == 0, x];
ConjPlaneDisplacement = x /. %[[1]];
NominalConjugatePlane = ConjPlaneDisplacement /. X1 → 0;
Print["Conjugate Plane is nominally out of position by ",
  NominalConjugatePlane, "m"];
RateCPD = (D[ConjPlaneDisplacement, X1]) /. X1 → 0;
Print["Conjugate Plane displacement for +/- 5 mm displacement of F1 = ",
  RateCPD * 0.005, "m"];
BTestF2HWSDistance = {RateCPD * 1000 * 0.005, "mm", 5, "mm"};
```

```
Conjugate Plane is nominally out of position by 5.23168m
```

```
Conjugate Plane displacement for +/- 5 mm displacement of F1 = -1.53125m
```

Hence the conjugate plane is totally sensitive to the position of HWS - THIS IS A GOOD THING - means it is easy to tune.

Variation in magnification (A = 1/17.5) with length changes: (assume that FL variations can be compensated for) - NO ISSUES - Easy to work with!

```
M1 = ITMXtoHWSVar;
A1 = M1[[1]][[1]];
```

Test variation in position of F1 - determine the magnification - INSENSITIVE

Set all delta values = 0 except for dz2 and dz1. Set these last two to the same value with opposite sign to model a lens out of position

```
A1A = Simplify[A1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → X1, dz1 → -X1}];
NominalMagnification = 1 / A1A /. X1 → 0;
Print["Nominal paraxial magnification is ", NominalMagnification, "m"];
RateVariationInMag = (D[A1A, X1]) /. X1 → 0;
Print["Variation in magnification for +/- 5 mm displacement of F1 = ",
  RateVariationInMag * 0.005, "x"];
```

```
Nominal paraxial magnification is -17.5m

Variation in magnification for +/- 5 mm displacement of F1 = 0.000105483x
```

Hence the paraxial magnification is VERY invariant to the position of F1

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0})];
BeamWAtHWSNom = pOutNom[[2]];
MOut = (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → X1, dz1 → -X1});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamWAtHWSX1 = pOutX1[[2]];
MagNom = BeamSizeAtITM / BeamWAtHWSNom;
MagX1 = BeamSizeAtITM / BeamWAtHWSX1;
Print["Absolute Variation in magnification for +5mm displacement = ",
  MagX1 - MagNom, "x"];
Print["Relative Variation in magnification for +5mm displacement = ",
  (MagX1 - MagNom) / MagNom * 100, "%"];
ATestF1OutOfPosition = {Round[(MagX1 - MagNom) / MagNom * 100 * 10] / 10.0, "%",  5, "mm"};
```

```
Absolute Variation in magnification for +5mm displacement = 0.0251726x

Relative Variation in magnification for +5mm displacement = 0.143844%
```

Hence the Gaussian magnification is VERY invariant to the position of F1

Test variation in position of F2 - determine the magnification - INSENSITIVE to distance between F1 and F2

Set all delta values = 0 except for dL1 and dz3. Set these last two to the same value with opposite sign to model a lens out of position

```
A1A = Simplify[A1 /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz1 → 0}];
NominalMagnification = 1 / A1A /. X1 → 0;
Print["Nominal paraxial magnification is ", NominalMagnification, "m"];
RateVariationInMag = (D[A1A, X1]) /. X1 → 0;
Print["Variation in magnification for +/- 5 mm displacement of F2 = ",
  RateVariationInMag * 0.005, "x"];
```

```
Nominal paraxial magnification is -17.5m
```

```
Variation in magnification for +/- 5 mm displacement of F2 = -0.0000996071x
```

Hence the paraxial magnification is VERY invariant to the position of F2

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0})];
BeamWAtHWSNom = pOutNom[[2]];
MOut = (ITMXtoHWSVar /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz1 → 0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamWAtHWSX1 = pOutX1[[2]];
MagNom = BeamSizeAtITM / BeamWAtHWSNom;
MagX1 = BeamSizeAtITM / BeamWAtHWSX1;
Print["Absolute Variation in magnification for +5mm displacement = ",
  MagX1 - MagNom, "x"];
Print["Relative Variation in magnification for +5mm displacement = ",
  (MagX1 - MagNom) / MagNom * 100, "%"];
ATestF2OutOfPosition = {Round[(MagX1 - MagNom) / MagNom * 100 * 10] / 10.0, "%",  5, "mm"};
```

```
Absolute Variation in magnification for +5mm displacement = -0.0435067x
```

```
Relative Variation in magnification for +5mm displacement = -0.24861%
```

Hence the Gaussian magnification is marginally sensitive to the position of F2

Test variation in F1-F2 distance: determine the magnification - INSENSITIVE

```
A1A = Simplify[A1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz1 → 0}];
NominalMagnification = 1 / A1A /. X1 → 0;
Print["Nominal paraxial magnification is ", NominalMagnification, "m"];
RateVariationInMag = (D[A1A, X1]) /. X1 → 0;
Print["Variation in magnification for +/- 5 mm displacement of F2 = ",
  RateVariationInMag * 0.005, "x"];
```

```
Nominal paraxial magnification is -17.5m
```

```
Variation in magnification for +/- 5 mm displacement of F2 = -0.0000996071x
```

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0})];
BeamWAtHWSNom = pOutNom[[2]];
MOut = (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz1 → 0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamWAtHWSX1 = pOutX1[[2]];
MagNom = BeamSizeAtITM / BeamWAtHWSNom;
MagX1 = BeamSizeAtITM / BeamWAtHWSX1;
Print[
  "Absolute Variation in mag. for +5mm distance change between F1 and F2 = ",
  MagX1 - MagNom, "x"];
Print["Relative Variation in mag. for +5mm distance change between F1 and F2 = ",
  (MagX1 - MagNom) / MagNom * 100, "%"];
ATestF1F2Distance = {Round[(MagX1 - MagNom) / MagNom * 100 * 10] / 10.0, "%",  5,  "mm"};
```

```
Absolute Variation in mag. for +5mm distance change between F1 and F2 = -0.0234457x
```

```
Relative Variation in mag. for +5mm distance change between F1 and F2 = -0.133975%
```

Hence the Gaussian magnification is very insensitive to the distance between F1 and F2

Test variation in position of HWS - determine the magnification - INSENSITIVE to position of HWS: 1% variation per 50mm

Set all delta values = 0 except for dz2 and dz1. Set these last two to the same value with opposite sign to model a lens out of position

```
A1A = Simplify[A1 /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0}];
NominalMagnification = 1 / A1A /. X1 → 0;
Print["Nominal paraxial magnification is ", NominalMagnification, "m"];
RateVariationInMag = (D[A1A, X1]) /. X1 → 0;
Print["Variation in magnification for +/- 5 mm displacement of HWS = ",
  RateVariationInMag * 0.005, "x"];
```

```
Nominal paraxial magnification is -17.5m
```

```
Variation in magnification for +/- 5 mm displacement of HWS = -1.11022 × 10^-18x
```

Hence the paraxial magnification is SUPER invariant to the position of F1

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0})];
BeamWAtHWSNom = pOutNom[[2]];
MOut = (ITMXtoHWSVar /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamWAtHWSX1 = pOutX1[[2]];
MagNom = BeamSizeAtITM / BeamWAtHWSNom;
MagX1 = BeamSizeAtITM / BeamWAtHWSX1;
Print["Absolute Variation in magnification for +5mm displacement = ",
  MagX1 - MagNom, "x"];
Print["Relative Variation in magnification for +5mm displacement = ",
  (MagX1 - MagNom) / MagNom * 100, "%"];
ATestF2HWSDistance = {Round[(MagX1 - MagNom) / MagNom * 100 * 10] / 10.0, "%",  5, "mm"};
```

```
Absolute Variation in magnification for +5mm displacement = -0.0199986x

Relative Variation in magnification for +5mm displacement = -0.114278%
```

Hence the Gaussian magnification is MOSTLY invariant to the position of HWS

Variation in collimation (C = 0) with length changes: (assume that FL variations can be compensated for) - NO ISSUES - Easy to work with!

```
M1 = ITMXtoHWSVar;
C1 = M1[[2]][[1]];
```

Test variation in position of F1 - determine the defocus - INSENSITIVE

Set all delta values = 0 except for dz2 and dz1. Set these last two to the same value with opposite sign to model a lens out of position

```
C1A = Simplify[C1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → X1, dz1 → -X1}];
NominalDefocus = C1A /. X1 → 0;
Print["Nominal paraxial defocus is ", NominalDefocus, " m^-1"];
RateVariationInDefocus = (D[C1A, X1]) /. X1 → 0;
Print["Variation in defocus for + 5 mm displacement of F1 = ",
  RateVariationInDefocus * 0.005, " m^-1, or ",
  1 / (RateVariationInDefocus * 0.005), "m"];
```

```
Nominal paraxial defocus is 0. m^-1

Variation in defocus for + 5 mm displacement of F1 = -0.000171129 m^-1, or -5843.55m
```

Hence the relative change in the paraxial defocus is quite large but the absolute change (with displacement of F1) is quite small, about 7km lens for 5mm displacement. At HWS this is not an issue.

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0})];
BeamRAtHWSNom = pOutNom[[1]];
MOut = (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → X1, dz1 → -X1});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamRAtHWSX1 = pOutX1[[1]];
Print["Variation in defocus of Gaussian Beam for +5mm displacement = ",
  1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom, " m^-1", "or lens of = ",
  1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), " m"];
CTestF1OutOfPosition = {1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), "m", 5 , "mm"};
```

```
Variation in defocus of Gaussian Beam for +5mm displacement =
 0.00375971 m^-1or lens of = 265.978 m
```

Hence the Gaussian curvature at HWS does not experience large changes in the radius of curvature that with motion of F1

Test variation in position of F2 - determine the defocus - INSENSITIVE

Set all delta values = 0 except for dz2 and dz3. Set these last two to the same value with opposite sign to model a lens out of position

```
C1A = Simplify[C1 /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz1 → 0}];
NominalDefocus = C1A /. X1 → 0;
Print["Nominal paraxial defocus is ", NominalDefocus, " m^-1"];
RateVariationInDefocus = (D[C1A, X1]) /. X1 → 0;
Print["Variation in defocus for + 5 mm displacement of F2 = ",
  RateVariationInDefocus * 0.005, " m^-1, or ",
  1 / (RateVariationInDefocus * 0.005), "m"];
```

```
Nominal paraxial defocus is -2.22045 × 10^-16 m^-1
```

```
Variation in defocus for + 5 mm displacement of F2 = -0.0000989329 m^-1, or -10107.9m
```

Hence the relative change in the paraxial defocus is quite large but the absolute change (with displacement of F2) is quite small, about 8km lens for 5mm displacement. At HWS this is not an issue.

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0})];
BeamRAtHWSNom = pOutNom[[1]];
MOut = (ITMXtoHWSVar /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz1 → 0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamRAtHWSX1 = pOutX1[[1]];
Print["Variation in defocus of Gaussian Beam for +5mm displacement = ",
  1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom, " m^-1", "or lens of = ",
  1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), " m"];
CTestF2OutOfPosition = {1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), "m", 5 , "mm"};
```

```
Variation in defocus of Gaussian Beam for +5mm displacement =
 0.000760379 m^-1or lens of = 1315.13 m
```

Hence the Gaussian curvature at HWS does not experience large changes in the radius of curvature that with motion of F2

Test variation in F1-F2 distance: determine the defocus - INSENSITIVE

```
C1A = Simplify[C1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz1 → 0}];
NominalDefocus = C1A /. X1 → 0;
Print["Nominal paraxial defocus is ", NominalDefocus, " m^-1"];
RateVariationInDefocus = (D[C1A, X1]) /. X1 → 0;
Print["Variation in defocus for + 5 mm displacement of F2 = ",
  RateVariationInDefocus * 0.005, " m^-1, or ",
  1 / (RateVariationInDefocus * 0.005), "m"];
```

```
Nominal paraxial defocus is -2.22045 × 10^-16 m^-1
```

```
Variation in defocus for + 5 mm displacement of F2 = -0.0000989329 m^-1, or -10107.9m
```

Hence the relative change in the paraxial defocus is quite large but the absolute change (with variation in F1-F2 distance) is quite small, about 8km lens for 5mm displacement. At HWS this is not an issue.

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0})];
BeamRAtHWSNom = pOutNom[[1]];
MOut = (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → -X1, dz1 → 0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamRAtHWSX1 = pOutX1[[1]];
Print["Variation in defocus of Gaussian Beam for +5mm displacement = ",
  1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom, " m^-1", "or lens of = ",
  1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), " m"];
CTestF1F2Distance = {1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), "m", 5 , "mm"};
```

```
Variation in defocus of Gaussian Beam for +5mm displacement =
 0.00102408 m^-1or lens of = 976.484 m
```

Hence the Gaussian curvature at HWS does not experience large changes in the radius of curvature that with variation in F1-F2 distance

Test variation in position of HWS - determine the defocus - INSENSITIVE

Set all delta values = 0 except for dz3. Set these last two to the same value with opposite sign to model a lens out of position

```
C1A = Simplify[C1 /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0}];
NominalDefocus = C1A /. X1 → 0;
Print["Nominal paraxial defocus is ", NominalDefocus, " m^-1"];
RateVariationInDefocus = (D[C1A, X1]) /. X1 → 0;
Print["Variation in defocus for + 5 mm displacement of F1 = ",
  RateVariationInDefocus * 0.005, " m^-1, or ",
  1 / (RateVariationInDefocus * 0.005), "m"];
```

```
Nominal paraxial defocus is 0. m^-1

Power::infy: Infinite expression 1/0. encountered. >>

Variation in defocus for + 5 mm displacement of F1 = 0. m^-1, or ComplexInfinitym
```

Hence the relative change in the paraxial defocus is ZERO

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMXtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0})];
BeamRAtHWSNom = pOutNom[[1]];
MOut = (ITMXtoHWSVar /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dz2 → 0, dz1 → -0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamRAtHWSX1 = pOutX1[[1]];
Print["Variation in defocus of Gaussian Beam for +5mm displacement = ",
  1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom, " m^-1", "or lens of = ",
  1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), " m"];
CTestF2HWSDistance = {1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), "m", 5 , "mm"};
```

```
Variation in defocus of Gaussian Beam for +5mm displacement =
 -0.000261358 m^-1or lens of = -3826.18 m
```

Hence the Gaussian curvature at HWS does not experience large changes in the radius of curvature that with variation in HWS position - expected since beam is approximately collimated here

Difficulty in mode-matching (MMOut = MMIn) - Take Care with Mode Matching

```
InputModeFromHWSWR = InputModeFromHWS;
InputModeFromHWSRW = {InputModeFromHWS[[2]], InputModeFromHWS[[1]]}
OutputMode = OutRW[InputModeFromHWSRW[[2]], InputModeFromHWSRW[[1]], λIN, MHWSv]
```

```
{-4.37031, 0.00725714}
```

```
{4.37031, 0.00725714}
```

10% variation in mode size, same curvature - no worries

```
InputModeFromHWSRW = {InputModeFromHWS[[2]], InputModeFromHWS[[1]] * 1.1}
OutputMode = OutRW[InputModeFromHWSRW[[2]], InputModeFromHWSRW[[1]], λIN, MHWSv]
```

```
{-4.37031, 0.00798286}
```

```
{4.37031, 0.00798286}
```

1.0m lens added to input beam, same mode size

```
InputModeFromHWSRW =
 {(1/1.0 + InputModeFromHWS[[2]]^-1)^-1, InputModeFromHWS[[1]]}
OutputMode = OutRW[InputModeFromHWSRW[[2]], InputModeFromHWSRW[[1]], λIN, MHWSv]
Print["Difference in output curvature = ",
 (1/OutputMode[[1]] + 1/InputModeFromHWSRW[[1]])^-1]
Print["Relative variation in output size with a 1m lens added to input = ",
  (OutputMode[[2]] - InputModeFromHWSRW[[2]])/InputModeFromHWSRW[[2]] * 100, "%"]
```

```
{1.29671, 0.00725714}
```

```
{0.813791, 0.00725714}
```

```
Difference in output curvature = 0.5
```

```
Relative variation in output size with a 1m lens added to input = -5.97592 × 10^-14 %
```

Temperature change required to scatter 0.01% from HWS error

```
Simple model first of temperature sensitivity - assuming common Length and
focal length changes
requirements:
1. consider thermal changes in length and defocus
2. ABCD and Gaussian analysis
```

Look at all the telescopes with defocus and infer maximum acceptable temp.

change

Input Mode is ...

Length sensitive matrices - main telescope between ITMXPO and SR3

```
ITMToITMXPOdX = Simplify[ {{1, ITMXPOtoSR3 + dx}, {0, 1}} . {{1, 0}, {-2/SR3RoC, 1}} . {{1, SR3toITMX}, {0, 1}} ];

ITMXPOToITMdX = Simplify[ {{1, SR3toITMX}, {0, 1}} . {{1, 0}, {-2/SR3RoC, 1}} . {{1, ITMXPOtoSR3 + dx}, {0, 1}} ];


ITMXPOToHWSdx =
  Simplify[{{1, z3 + dx3}, {0, 1}} . {{1, 0}, {-1/f2, 1}} . {{1, z2 + dx2}, {0, 1}} . {{1, 0}, {-1/f1, 1}} . {{1, z1 + dx1}, {0, 1}}] /.
   TestSolutionSubstitution;
HWSToITMXPOdx = Simplify[{{1, z1 + dx1}, {0, 1}} . {{1, 0}, {-1/f1, 1}} . {{1, z2 + dx2}, {0, 1}} .
      {{1, 0}, {-1/f2, 1}} . {{1, z3 + dx3}, {0, 1}}] /. TestSolutionSubstitution;

HWStoHWSdx = ITMXPOToHWSdx.ITMToITMXPOdX.MITM.ITMXPOToITMdX.HWSToITMXPOdx /.
   {dx1 → 0, dx2 → 0, dx3 → 0};
```

Check the matrix

```
HWStoHWS = HWStoHWSdx /. dx → 0;
InputModeFromHWS
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, HWStoHWS];
Reverse[pOut]
```

```
{0.00725714, -4.37031}
```

```
{0.00725714, 4.37031}
```

A1: Get the defocus at ITM for a given displacement between HAM4 and HAM5

To scatter out 0.01% the maximum acceptable defocus is 1.3*10^-6 m^-1;

```
MaxAcceptableDefocusAtITM = MaxDefocusAtITMAll;

dxTest = 0.00025;
dtM = HWStoHWSdx /. dx → dxTest;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (-InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
```

```
Input defocus = -0.228817 m^-1
```

```
Output defocus = 0.228339 m^-1
```

```
dSAtITM = dS / (M2^2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dxTest * 10^6], "um displacement"]
dSdxITMXPOtoSR3 = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
```

```
Defocus at ITM = 1.55861 × 10^-6m^-1 for 250um displacement
```

```
Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

A2: Limit apparent temperature change of the concrete between IMTXPO & SR3

Determine how much the previous dx would need to be scaled by to yield the maximum apparent defocus allowed at the ITM

```
dxScale = MaxAcceptableDefocusAtITM / dSAtITM
```

```
1.34736
```

```
CoefficientThermalExpansionOfConcrete = 1.2 * 10^-5;
dT = dx / (L * CoefficientThermalExpansionOfConcrete) /.
   {L → ITMXPOtoSR3, dx → dxTest * dxScale};
dTITMXPOtoSR3 = {Round[dT * 100] / 100.0, "K"};
Print["Maximum acceptable change in temperature of
   concrete slab between HAM4 and HAM5 = ", Round[dT * 100] * 10, " mK"]
```

```
Maximum acceptable change in temperature of concrete slab between HAM4 and HAM5 = 1850 mK
```

Now we need a secondary set of optics that introduce the same level of thermal defocus and give a small beam at the reference retro-reflector.

B1: Get the defocus at ITM for a given displacement between ITMXPO and F1

```
Print["Magnification between HWS and ITM = ", M2]
```

```
Magnification between HWS and ITM = 17.5
```

```
HWStoHWSdx1 = ITMXPOToHWSdx.ITMToITMXPOdX.MITM.ITMXPOToITMdX.HWSToITMXPOdx /.
   {dx → 0, dx2 → 0, dx3 → 0};
```

```
dxTest = 0.00025;
dtM = HWStoHWSdx1 /. dx1 → dxTest;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (-InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
dSAtITM = dS / (M2^2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dxTest * 10^6], "um displacement"]
dSdx = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
```

```
Input defocus = -0.228817 m^-1

Output defocus = 0.228339 m^-1

Defocus at ITM = 1.55861 × 10^-6 m^-1 for 250um displacement

Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

B2: Limit apparent temperature change of the stainless between ITMXPO & F1

C1: Get the defocus at ITM for a given displacement between F1 and F2

```
HWStoHWSdx2 = ITMXPOToHWSdx.ITMToITMXPOdX.MITM.ITMXPOToITMdX.HWSToITMXPOdx /.
    {dx → 0, dx1 → 0, dx3 → 0};

dxTest = 0.00025;
dtM = HWStoHWSdx2 /. dx2 → dxTest;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (-InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
dSAtITM = dS / (M2^2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dxTest * 10^6], "um displacement"]
dSdx = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
Input defocus = -0.228817 m^-1

Output defocus = 0.228714 m^-1

Defocus at ITM = 3.34865 × 10^-7 m^-1 for 250um displacement

Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

C2: Limit apparent temperature change of the stainless between F1 & F2

Determine how much the previous dx would need to be scaled by to yield the maximum apparent defocus allowed at the ITM

```
dxScale = MaxAcceptableDefocusAtITM / dSAtITM
6.27118

CoefficientThermalExpansionOfStainlessSteel = 1.73 * 10^-5;
dT = dx / (L * CoefficientThermalExpansionOfStainlessSteel) /.
    {L → Z2A, dx → dxTest * dxScale};
dTF1toF2 = {Round[dT * 100] / 100.0, "K"};
Print["Maximum acceptable change in temperature of HAM4 and
    ISCT to affect F1 and F2 distance = ", Round[dT * 100] * 10, " mK"]
Maximum acceptable change in temperature of HAM4 and ISCT to affect F1 and F2 distance =
 32280 mK
```

D1: Get the defocus at ITM for a given displacement between F2 and HWS

```
HWStoHWSdx3 = ITMXPOToHWSdx.ITMToITMXPOdX.MITM.ITMXPOToITMdX.HWSToITMXPOdx /.
    {dx → 0, dx1 → 0, dx2 → 0};

dxTest = 0.00025;
dtM = HWStoHWSdx3 /. dx3 → dxTest;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (-InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
dSAtITM = dS / (M2^2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dxTest * 10^6], "um displacement"]
dSdx = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
```

```
Input defocus = -0.228817 m^-1

Output defocus = 0.22879 m^-1

Defocus at ITM = 8.54339 × 10^-8 m^-1 for 250um displacement

Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

D2: Limit apparent temperature change of the stainless between F2 & HWS

Look at all the telescopes with defocus of lenses and infer maximum acceptable temp. change

Input Mode is ...

```
InputModeFromHWS
```

```
{0.00725714, -4.37031}
```

Length sensitive matrices - main telescope between ITMXPO and SR3

```
ITMToITMXPOdX =
 Simplify[ {{1, ITMXPOtoSR3}, {0, 1}} . {{1, 0}, {-2/(SR3RoC (1+ alphaFS * dTSR3)), 1}} . {{1, SR3toITMX}, {0, 1}} ];
ITMXPOToITMdX = Simplify[
   {{1, SR3toITMX}, {0, 1}} . {{1, 0}, {-2/(SR3RoC (1+ alphaFS * dTSR3)), 1}} . {{1, ITMXPOtoSR3}, {0, 1}} ];

ITMXPOToHWSdx = Simplify[ {{1, z3}, {0, 1}} . {{1, 0}, {-1/(f2 (1 + alphaFS * dTF2)), 1}} . {{1, z2}, {0, 1}} .
      {{1, 0}, {-1/(f1 (1 + alphaFS * dTF1)), 1}} . {{1, z1}, {0, 1}} ] /. TestSolutionSubstitution;
HWSToITMXPOdx = Simplify[ {{1, z1}, {0, 1}} . {{1, 0}, {-1/(f1 (1 + alphaFS * dTF1)), 1}} . {{1, z2}, {0, 1}} .
      {{1, 0}, {-1/(f2 (1 + alphaFS * dTF2)), 1}} . {{1, z3}, {0, 1}} ] /. TestSolutionSubstitution;
```

A1: Get the defocus at ITM for a uniform temperature change of all the optics

To scatter out 0.01% the maximum acceptable defocus is 1.3*10^-6 m^-1;

Fused Silica

Assume that the optics are all fused silica

```
HWStoHWSdT1 = ITMXPOToHWSdx.ITMToITMXPOdX.MITM.ITMXPOToITMdX.HWSToITMXPOdx /.
    {alphaFS → 0.55 * 10^-6, dTSR3 → dT1,  dTF1 → dT1, dTF2 → dT1};

dT1Test = 10.0;
dtM = HWStoHWSdT1 /. dT1 → dT1Test;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (-InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
```

```
Input defocus = -0.228817 m^-1
```

```
Output defocus = 0.229006 m^-1
```

```
dSAtITM = dS / (M2^2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dT1Test], "K temperature change"]
dSdxITMXPOtoSR3 = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
```

```
Defocus at ITM = -6.20123 × 10^-7m^-1 for 10K temperature change
```

```
Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

BK7

Assume that the optics are all BK7, except SR3 which is assumed to stay the same temperature

```
HWStoHWSdT1 = ITMXPOToHWSdx.ITMToITMXPOdX.MITM.ITMXPOToITMdX.HWSToITMXPOdx /.
    {alphaFS → 7.5 * 10^-6, dTSR3 → 0,  dTF1 → dT1, dTF2 → dT1};

MaxAcceptableDefocusAtITM = MaxDefocusAtITMAll;

dT1Test = 10.0;
dtM = HWStoHWSdT1 /. dT1 → dT1Test;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (-InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
```

```
Input defocus = -0.228817 m^-1
```

```
Output defocus = 0.228759 m^-1
```

```
dSAtITM = dS / (M2^2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dT1Test], "K temperature change"]
dSdxITMXPOtoSR3 = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
```

```
Defocus at ITM = 1.86921 × 10^-7 m^-1 for 10K temperature change
```

```
Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

Report on Solution

```
Print["=== REPORT ON H1:ITMX-HWS SOLUTION ==="]
Print[" "];
Print["ITMX apparent Radius of Curvature = ", ITMRoC, "m"];
Print["ITM to SR3 optical distance = ", SR3toITMX, "m"];
Print["BS to SR3 optical distance = ", SR3toBS, "m"];
Print["SR3 Radius of Curvature = ", SR3RoC, "m"];
Print["SR3 to ITMXPO distance = ", ITMXPOtoSR3, "m"];
Print[" "];
Print["ITMXPO to F1 distance = ", Round[(Z1A) * 1000] / 1000.0, "m"];
Print["F1 focal length = ", Round[(F1A) * 1000] / 1000.0, "m"];
Print["F1 to F2 distance = ", Round[(Z2A ) * 1000] / 1000.0, "m"];
Print["F2 focal length = ", Round[(F2A) * 1000] / 1000.0, "m"];
Print["F2 to HWS distance = ", Round[(Z3A ) * 1000] / 1000.0, "m"];
Print[" "];
Print["----"]
Print["TEMPERATURE SENSITIVITY"]
Print["Maximum Acceptable defocus at the ITM = ", MaxAcceptableDefocusAtITM];
Print["ITMXPO to SR3 - maximum acceptable temperature change = ",
 dTITMXPOtoSR3[[1]], dTITMXPOtoSR3[[2]]]
Print["F1 to ITMXPO - maximum acceptable temperature change = ",
```

```
 dTITMXPOtoF1[[1]], dTITMXPOtoF1[[2]]]
Print["F1 to F2 - maximum acceptable temperature change = ",
 dTF1toF2[[1]], dTF1toF2[[2]]]
Print["F2 to HWS - maximum acceptable temperature change = ",
 dTF2toHWS[[1]], dTF2toHWS[[2]]]
Print[" "];
Print["----"]
Print["TOLERANCING"]
Print["Imaging (B=0): F1 out of Position by ", BTestF1OutOfPosition[[3]],
 BTestF1OutOfPosition[[4]], " moves conjugate plane by ",
 BTestF1OutOfPosition[[1]], BTestF1OutOfPosition[[2]]]
Print["Imaging (B=0): F2 out of Position by ", BTestF2OutOfPosition[[3]],
 BTestF2OutOfPosition[[4]], " moves conjugate plane by ",
 BTestF2OutOfPosition[[1]], BTestF2OutOfPosition[[2]]]
Print["Imaging (B=0): F1-F2 distance varies by ", BTestF1F2Distance[[3]],
 BTestF1F2Distance[[4]], " moves conjugate plane by ",
 BTestF1F2Distance[[1]], BTestF1F2Distance[[2]]]
Print["Imaging (B=0): HWS out of Position by ", BTestF2HWSDistance[[3]],
 BTestF2HWSDistance[[4]], " moves conjugate plane by ",
 BTestF2HWSDistance[[1]], BTestF2HWSDistance[[2]]]
Print[" "]
Print["Magnification (A=1/17.5): F1 out of Position by ",
 ATestF1OutOfPosition[[3]], ATestF1OutOfPosition[[4]],
 " varies magnification by ",
 ATestF1OutOfPosition[[1]], ATestF1OutOfPosition[[2]]]
Print["Magnification (A=1/17.5): F2 out of Position by ",
 ATestF2OutOfPosition[[3]], ATestF2OutOfPosition[[4]],
 " varies magnification by ",
 ATestF2OutOfPosition[[1]], ATestF2OutOfPosition[[2]]]
Print["Magnification (A=1/17.5): F1-F2 distance varies by ",
 ATestF1F2Distance[[3]], ATestF1F2Distance[[4]],
 " varies magnification by ", ATestF1F2Distance[[1]], ATestF1F2Distance[[2]]]
Print["Magnification (A=1/17.5): F2-HWS distance varies by ",
 ATestF2HWSDistance[[3]], ATestF2HWSDistance[[4]],
 " varies magnification by ", ATestF2HWSDistance[[1]], ATestF2HWSDistance[[2]]]
Print[" "]
Print["Collimation (C=0): F1 out of Position by ", CTestF1OutOfPosition[[3]],
 CTestF1OutOfPosition[[4]], " looks like an additional lens at HWS with f = ",
 CTestF1OutOfPosition[[1]], CTestF1OutOfPosition[[2]]]
Print["Collimation (C=0): F2 out of Position by ", CTestF2OutOfPosition[[3]],
 CTestF2OutOfPosition[[4]], " looks like an additional lens at HWS with f = ",
 CTestF2OutOfPosition[[1]], CTestF2OutOfPosition[[2]]]
Print["Collimation (C=0): F1-F2 distance varied by ", CTestF1F2Distance[[3]],
 CTestF1F2Distance[[4]], " looks like an additional lens at HWS with f = ",
 CTestF1F2Distance[[1]], CTestF1F2Distance[[2]]]
Print["Collimation (C=0): F2-HWS distance varied by ", CTestF2HWSDistance[[3]],
 CTestF2HWSDistance[[4]], " looks like an additional lens at HWS with f = ",
 CTestF2HWSDistance[[1]], CTestF2HWSDistance[[2]]]

Show[ITMXPlottedSolutionFromITMX]
Show[ITMXPlottedSolutionFromITMXPO]
```

```
=== REPORT ON H1:ITMX-HWS SOLUTION ===
```

ITMX apparent Radius of Curvature = 1338.41m

ITM to SR3 optical distance = 24.663m

BS to SR3 optical distance = 19.4313m

SR3 Radius of Curvature = 36m

SR3 to ITMXPO distance = 15.1729m

ITMXPO to F1 distance = 2.156m

F1 focal length = -1.699m

F1 to F2 distance = 2.808m

F2 focal length = 1.699m

F2 to HWS distance = 2.706m

TEMPERATURE SENSITIVITY

Maximum Acceptable defocus at the ITM = 2.1×10^{-6}

ITMXPO to SR3 - maximum acceptable temperature change = 1.85K

F1 to ITMXPO - maximum acceptable temperature change = 9.03K

F1 to F2 - maximum acceptable temperature change = 32.28K

F2 to HWS - maximum acceptable temperature change = 131.26K

TOLERANCING

Imaging (B=0): F1 out of Position by 5mm moves conjugate plane by -536.785mm

Imaging (B=0): F2 out of Position by 5mm moves conjugate plane by -993.78mm

Imaging (B=0): F1-F2 distance varies by 5mm moves conjugate plane by 537.47mm

Imaging (B=0): HWS out of Position by 5mm moves conjugate plane by -1531.25mm

Magnification (A=1/17.5): F1 out of Position by 5mm varies magnification by 0.1%

Magnification (A=1/17.5): F2 out of Position by 5mm varies magnification by -0.2%

Magnification (A=1/17.5): F1-F2 distance varies by 5mm varies magnification by -0.1%

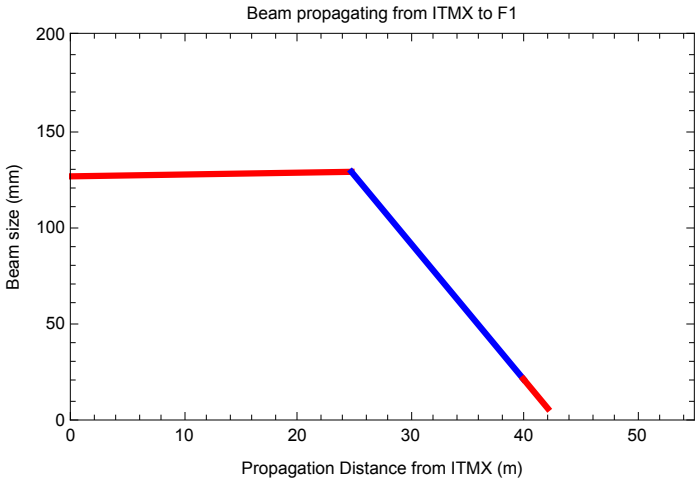
Magnification (A=1/17.5): F2-HWS distance varies by 5mm varies magnification by -0.1%

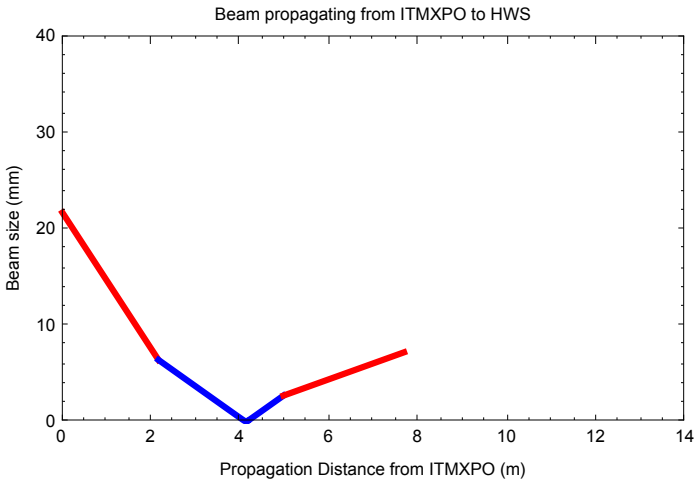
Collimation (C=0): F1 out of Position by 5mm
looks like an additional lens at HWS with f = 265.978m

Collimation (C=0): F2 out of Position by 5mm
looks like an additional lens at HWS with f = 1315.13m

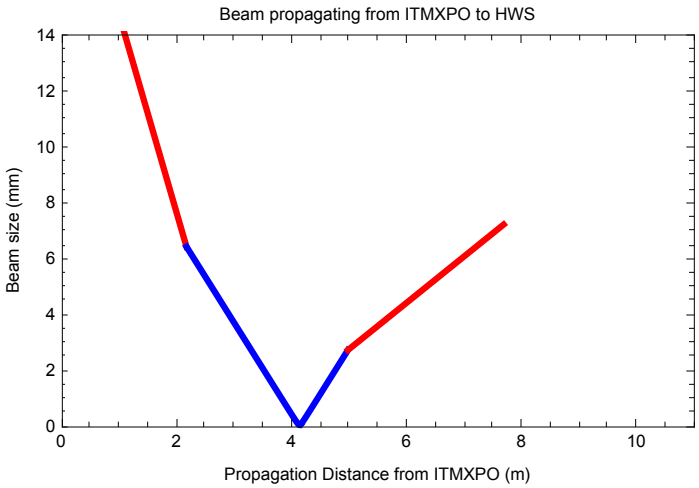
Collimation (C=0): F1-F2 distance varied by 5
mm looks like an additional lens at HWS with f = 976.484m

```
Collimation (C=0): F2-HWS distance varied by 5
 mm looks like an additional lens at HWS with f = -3826.18m
```






```
Show[ITMXPlottedSolutionFromITMXPO, PlotRange → {{0, 11}, {0, 14}}]
```



Beam size at the mirrors

Create a Module to determine the beam size

Import the data on the mirror positions

```
dataIN =
  Import["/Users/aidanbrooks/Documents/LIGO/documents/Advanced_LIGO/TCS/Hartmann
     sensor/modeling/H1HWSX_coords.txt", "TSV"];
totaldistance = 0;
distList = {};
nameList = {};
For[ii = 1, ii ≤ Length[dataIN], ii++, {
   d1 = dataIN[[ii]][[7]] / 1000.0;
   totaldistance = totaldistance + d1;
   dataIN[[ii]] = Append[dataIN[[ii]], totaldistance];
   distList = Append[distList , totaldistance];
   nameList = Append[nameList, dataIN[[ii]][[1]]];
  }];
```

Import::nffil: File not found during Import ≫

```
dataIN
```

```
$Failed
```

Distance List from T1100463-v5

```
distances = {0, 972.7, 1650.0, 2663.7, 2926.6, 4883.2, 5563.0,
    5811.4, 5861.4, 6009.9, 6959.7, 7760.0, 7850.0, 8278.8, 8328.8,
    8757.6, 9172.6, 9922.6, 10084.4, 10234.4, 10634.4, 10721.4} / 1000;
distList = distances;
```

Determine the beam size at each interface

```
beamSizeList = {};
For[kk = 1, kk ≤ Length[distList], kk++, {
   beamsize =
    GetTheBeamSize[{{{1, z1}, {0, 1}}, {{1, 0}, {-1/f1, 1}}, {{1, z2}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /.
      TestSolutionSubstitution, OutModeITMXPO, 840 * 10^-9, distList[[kk]]];

   beamSizeList = Append[beamSizeList, {distList[[kk]], beamsize}];

   If[kk == 3, Print[distList[[kk]]]];
   If[kk == 7, {
     Print[distList[[kk]] - 0.877];
     wVP = GetTheBeamSize[{{{1, z1}, {0, 1}}, {{1, 0}, {-1/f1, 1}},
         {{1, z2}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /. TestSolutionSubstitution,
       OutModeITMXPO, 840 * 10^-9, distList[[kk]] - 0.877];
     wUPM = GetTheBeamSize[{{{1, z1}, {0, 1}}, {{1, 0}, {-1/f1, 1}}, {{1, z2}, {0, 1}},
         {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /. TestSolutionSubstitution,
       OutModeITMXPO, 840 * 10^-9, distList[[kk]]];
    }];
  }];
```

```
1.65
```

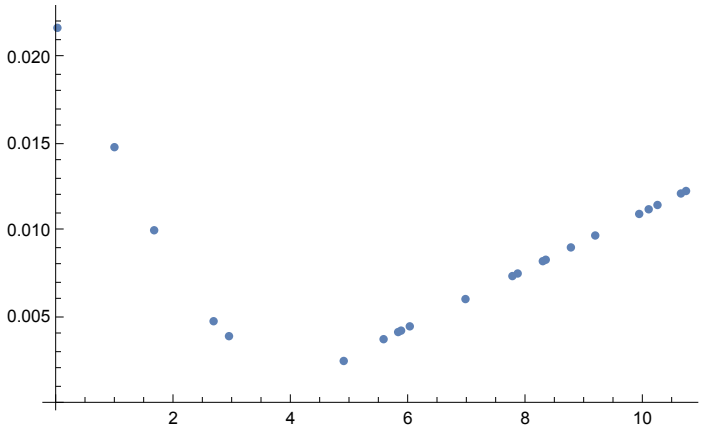
```
4.686
```

```
MtoVP = {{1, 4.0533 - 1.64901}, {0, 1}} . {{1, 0}, {-1/f1, 1}} . {{1, z1}, {0, 1}} . ITMToITMXPO /.
   TestSolutionSubstitution;
w1 = Sqrt[2] * 54.0 * 10^-3;


div = (wVP - wUPM) / 0.877;
w0 = (λIN / (Pi div)) * 1000;
f0 = wVP / div;
```

```
beamSizeList
```

```
{{0, 0.0217538}, {0.9727, 0.0148567}, {1.65, 0.0100542}, {2.6637, 0.00479701},
 {2.9266, 0.00393309}, {4.8832, 0.00249981}, {5.563, 0.00375886},
 {5.8114, 0.0041711}, {5.8614, 0.00425409}, {6.0099, 0.00450058},
 {6.9597, 0.00607741}, {7.76, 0.0074063}, {7.85, 0.00755575},
 {8.2788, 0.00826782}, {8.3288, 0.00835086}, {8.7576, 0.00906296},
 {9.1726, 0.00975215}, {9.9226, 0.0109977}, {10.0844, 0.0112664},
 {10.2344, 0.0115156}, {10.6344, 0.0121799}, {10.7214, 0.0123244}}
```

```
ListPlot[beamSizeList]
```




```
Export["/Users/aidanbrooks/Documents/LIGO/documents/Advanced_LIGO/TCS/Hartmann
    sensor/design/optical layout and
    modeling/H1HWSX_coords_beamsize.txt", beamSizeList, "Table"];
```

```
MatrixForm[beamSizeList]
```

0	0.0217538
0.9727	0.0148567
1.65	0.0100542
2.6637	0.00479701
2.9266	0.00393309
4.8832	0.00249981
5.563	0.00375886
5.8114	0.0041711
5.8614	0.00425409
6.0099	0.00450058
6.9597	0.00607741
7.76	0.0074063
7.85	0.00755575
8.2788	0.00826782
8.3288	0.00835086
8.7576	0.00906296
9.1726	0.00975215
9.9226	0.0109977
10.0844	0.0112664
10.2344	0.0115156
10.6344	0.0121799
10.7214	0.0123244

```
For[ii = 1, ii ≤ Length[beamSizeList], ii++, Print[1000 * beamSizeList[[ii]][[2]]]]
```

```
21.7538
14.8567
10.0542
4.79701
3.93309
2.49981
3.75886
4.1711
4.25409
4.50058
6.07741
7.4063
7.55575
8.26782
8.35086
9.06296
9.75215
10.9977
11.2664
11.5156
12.1799
12.3244
```

Determine the beam size at each interface

```
GouyPhase =
 GetTheGouyPhase[{({{1, z1}, {0, 1}}), ({{1, 0}, {-1/f1, 1}}), ({{1, z2}, {0, 1}}), ({{1, 0}, {-1/f2, 1}}), ({{1, z3}, {0, 1}})} /.
   TestSolutionSubstitution, OutModeITMXPO, 840 * 10^-9, distList[[1]]]
```

```
Solve::ratnz: Solve was unable to solve the system with inexact coefficients
    The answer was obtained by solving a corresponding exact system and numericizing the result >>
-1.56906
```

```
beamSizeList = {};
GouyPhaseList = {};
GouyPhaseListDeg = {};
For[kk = 1, kk ≤ Length[distList], kk++, {
   beamsize =
    GetTheBeamSize[{{1 z1; 0 1}, {1 0; -1/f1 1}, {1 z2; 0 1}, {1 0; -1/f2 1}, {1 z3; 0 1}} /.
      TestSolutionSubstitution, OutModeITMXPO, 840 * 10^-9, distList[[kk]]];

   beamSizeList = Append[beamSizeList, {distList[[kk]], beamsize}];

   GouyPhase =
    GetTheGouyPhase[{{1 z1; 0 1}, {1 0; -1/f1 1}, {1 z2; 0 1}, {1 0; -1/f2 1}, {1 z3; 0 1}} /.
      TestSolutionSubstitution, OutModeITMXPO, 840 * 10^-9, distList[[kk]]];

   GouyPhaseList = Append[GouyPhaseList, {distList[[kk]], GouyPhase}];
   GouyPhaseListDeg =
    Append[GouyPhaseListDeg, {distList[[kk]], GouyPhase * 180 / Pi}];

  }];
```

Solve::ratnz: Solve was unable to solve the system with inexact coefficients
The answer was obtained by solving a corresponding exact system and numericizing the result >>

Solve::ratnz: Solve was unable to solve the system with inexact coefficients
The answer was obtained by solving a corresponding exact system and numericizing the result >>

Power::infy: Infinite expression $\frac{1}{0.}$ encountered >>

Solve::ratnz: Solve was unable to solve the system with inexact coefficients
The answer was obtained by solving a corresponding exact system and numericizing the result >>

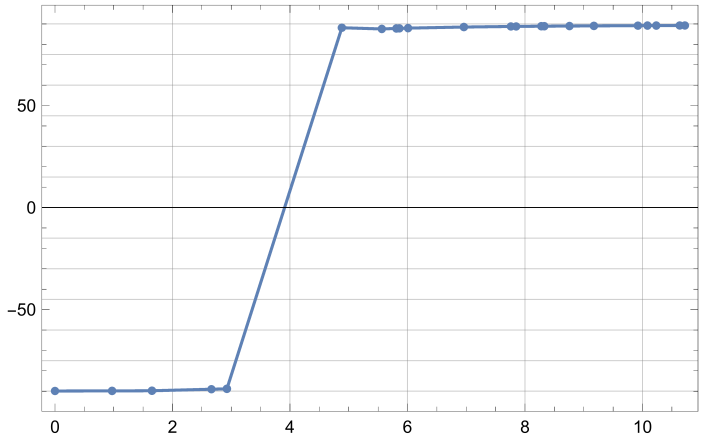
General::stop: Further output of Solve::ratnz will be suppressed during this calculation >>

Power::infy: Infinite expression $\frac{1}{0.}$ encountered >>

Power::infy: Infinite expression $\frac{1}{0.}$ encountered >>

General::stop: Further output of Power::infy will be suppressed during this calculation >>

```
plot1 = ListPlot[GouyPhaseListDeg, Frame → True,
   GridLines → {Automatic, Table[ii, {ii, -90, 90, 15}]}];
plot2 = ListPlot[GouyPhaseListDeg, Frame → True, Joined → True];
Show[plot1, plot2]
```




```
GouyPhaseOptics = {};
For[jj = 1, jj ≤ Length[GouyPhaseListDeg], jj++, {
   GouyPhaseOptics =
     Append[GouyPhaseOptics, {nameList[[jj]], GouyPhaseListDeg[[jj]][[2]]}];
  }];
MatrixForm[GouyPhaseOptics]
```

Part::partw: Part 1 of {} does not exist >>

Part::partw: Part 2 of {} does not exist >>

Part::partw: Part 3 of {} does not exist >>

General::stop: Further output of Part::partw will be suppressed during this calculation >>

$$\begin{pmatrix} \{\}[[1]] & -89.9007 \\ \{\}[[2]] & -89.8546 \\ \{\}[[3]] & -89.7851 \\ \{\}[[4]] & -89.0283 \\ \{\}[[5]] & -88.8148 \\ \{\}[[6]] & 88.1351 \\ \{\}[[7]] & 87.5455 \\ \{\}[[8]] & 87.7882 \\ \{\}[[9]] & 87.8314 \\ \{\}[[10]] & 87.9502 \\ \{\}[[11]] & 88.4822 \\ \{\}[[12]] & 88.7546 \\ \{\}[[13]] & 88.7792 \\ \{\}[[14]] & 88.8843 \\ \{\}[[15]] & 88.8954 \\ \{\}[[16]] & 88.9822 \\ \{\}[[17]] & 89.0542 \\ \{\}[[18]] & 89.1613 \\ \{\}[[19]] & 89.1813 \\ \{\}[[20]] & 89.199 \\ \{\}[[21]] & 89.2427 \\ \{\}[[22]] & 89.2516 \end{pmatrix}$$

```
For[kk = 1, kk ≤ Length[distList], kk++, {
   dataIN[[kk]] = Append[dataIN[[kk]], beamSizeList[[kk]][[2]] * 1000.0];
  }];
```

Part::partd: Partspecification$Failed[[1]] is longerthandepthofobject. >>

Part::pkspec1: The expression21.753760240986814` cannotbeusedas a partspecification. >>

Set::partd: Partspecificationdata IN[[kk]] is longerthandepthofobject. >>

Part::partd: Partspecification$Failed[[2]] is longerthandepthofobject. >>

Part::pkspec1: The expression14.856670812053558` cannotbeusedas a partspecification. >>

Set::partd: Partspecificationdata IN[[kk]] is longerthandepthofobject. >>

Part::partd: Partspecification$Failed[[3]] is longerthandepthofobject. >>

General::stop: Furtheroutputof Part::partdwillbesuppressedduringthiscalculation. >>

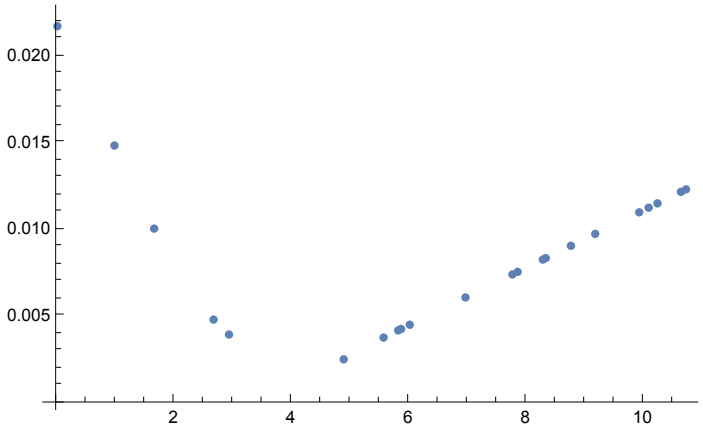
Part::pkspec1: The expression10.0541759915748` cannotbeusedas a partspecification. >>

General::stop: Furtheroutputof Part::pkspec1willbesuppressedduringthiscalculation. >>

Set::partd: Partspecificationdata IN[[kk]] is longerthandepthofobject. >>

General::stop: Furtheroutputof Set::partdwillbesuppressedduringthiscalculation. >>

```
ListPlot[beamSizeList]
```




```
Export["/Users/aidanbrooks/Documents/LIGO/documents/Advanced_LIGO/TCS/Hartmann
    sensor/modeling/H1HWSX_coords_beamsize.txt", dataIN, "Table"];
```

```
MatrixForm[dataIN]
```

$Failed

Astigmatism

```
Theta = 6.5;

AstigmaticRadiiCurvature =
  {(f1 * 2) * Cos[Theta * Pi/180]^2, (f2 * 2)} /. TestSolutionSubstitution
AstigmaticFocalLengthsMM = AstigmaticRadiiCurvature / 2.0
```

```
{-3.35524, 3.3988}
```

```
{-1.67762, 1.6994}
```

Horizontal Direction - work out change in conj. plane position

```
ITMToITMXPO;
ImgMatrix = Simplify[{{1, z3}, {0, 1}}.{{1, 0}, {-1/AstigmaticFocalLengthsMM[[2]], 1}}.
    {{1, z2}, {0, 1}}.{{1, 0}, {-1/(AstigmaticFocalLengthsMM[[1]]), 1}}.{{1, z1}, {0, 1}}];
SR2ARToHWS = ImgMatrix /. TestSolutionSubstitution;
M1 = {{1, z}, {0, 1}}.SR2ARToHWS.ITMToITMXPO;
zsoln = Solve[M1[[1]][[2]] == 0, z]
mag = M1[[2]][[2]] /. zsoln[[1]]
imgHsoln = {z, mag} /. zsoln[[1]]
zSolve = z /. zsoln[[1]]
```

```
{{z → 0.00826309}}
```

```
-17.5909
```

```
{0.00826309, -17.5909}
```

```
0.00826309
```

Horizontal Direction - compensate with second lens

```
ITMToITMXPO;
ImgMatrix = Simplify[{{1, z3}, {0, 1}}.{{1, 0}, {-1/(f2*a1), 1}}.
    {{1, z2}, {0, 1}}.{{1, 0}, {-1/(AstigmaticFocalLengthsMM[[1]]), 1}}.{{1, z1}, {0, 1}}];
SR2ARToHWS = ImgMatrix /. TestSolutionSubstitution;
M1 = SR2ARToHWS.ITMToITMXPO;
asoln = Solve[M1[[1]][[2]] == 0, a1]
AngleV = 180/Pi ArcCos[Sqrt[a1]] /. asoln[[1]];
Print["Angle of Incidence for Lens 2 = ", ToString[AngleV], " degrees"];
M1Vertical = Chop[M1 /. asoln[[1]]];
MatrixForm[M1Vertical]
(f2 /. TestSolutionSubstitution) * 2000 * a1 /. asoln[[1]]
```

Solve:ratnz: Solve was unable to solve the system with inexact coefficients
The answer was obtained by solving a corresponding exact system and numericizing the result >>

```
{{a1 → 0.998092}}

Angle of Incidence for Lens 2 = 2.50346 degrees

{{-0.0566745, 0}, {-0.000122291, -17.6446}}

3392.32
```

Major Optics Misalignments - ABCDEFGH formalism

```
TestSolutionSubstitution
```

```
{f1 → -1.6994, f2 → 1.6994, z1 → 2.15622, z2 → 2.80774, z3 → 2.70621}
```

```
solnMis = {VacLens → -1.699, AirLens → 1.699,
  POtoVacLens → 2.156, VacLenstoAirLens → 2.808, AirLenstoHWS → 2.706}
```

```
{VacLens → -1.699, AirLens → 1.699, POtoVacLens → 2.156,
 VacLenstoAirLens → 2.808, AirLenstoHWS → 2.706}
```

```
MITMmis = {{1, 0, 0}, {-2/-ITMRoC, 1, 0}, {0, 0, 1}};
MAirLensToVacLens = {{1, VacLenstoAirLens, 0}, {0, 1, 0}, {0, 0, 1}}
MVacLensToAirLens = {{1, VacLenstoAirLens, 0}, {0, 1, 0}, {0, 0, 1}}
```

```
{{1, VacLenstoAirLens, 0}, {0, 1, 0}, {0, 0, 1}}

{{1, VacLenstoAirLens, 0}, {0, 1, 0}, {0, 0, 1}}
```

```
MMistoITM = {{1, 0, 0}, {0, 1, 2 * ITMXdtheta}, {0, 0, 1}}.{{1, 0.2 * SR3toITMX, 0}, {0, 1, 0}, {0, 0, 1}}.{{1, 0, 0}, {0, 1, 2 * BSdtheta}, {0, 0, 1}}.
    {{1, 0.8 * SR3toITMX, 0}, {0, 1, 0}, {0, 0, 1}}.{{1, 0, 0}, {-2 / SR3RoC, 1, 0}, {0, 0, 1}}.{{1, 0, 0}, {0, 1, 2 * SR3dtheta}, {0, 0, 1}}.
    {{1, ITMXPOtoSR3, 0}, {0, 1, 0}, {0, 0, 1}}.{{1, POtoVacLens, 0}, {0, 1, 0}, {0, 0, 1}}.{{1, 0, 0}, {-1 / VacLens, 1, 0}, {0, 0, 1}}.
    MAirLensToVacLens.{{1, 0, 0}, {-1 / AirLens, 1, 0}, {0, 0, 1}}.{{1, AirLenstoHWS, 0}, {0, 1, 0}, {0, 0, 1}};

MMisfromITM = {{1, AirLenstoHWS, 0}, {0, 1, 0}, {0, 0, 1}}.{{1, 0, 0}, {-1 / AirLens, 1, 0}, {0, 0, 1}}.
    MVacLensToAirLens.{{1, 0, 0}, {-1 / VacLens, 1, 0}, {0, 0, 1}}.{{1, POtoVacLens, 0}, {0, 1, 0}, {0, 0, 1}}.
    {{1, ITMXPOtoSR3, 0}, {0, 1, 0}, {0, 0, 1}}.{{1, 0, 0}, {-2 / SR3RoC, 1, 0}, {0, 0, 1}}.{{1, 0, 0}, {0, 1, 2 * SR3dtheta}, {0, 0, 1}}.
    {{1, 0.8 * SR3toITMX, 0}, {0, 1, 0}, {0, 0, 1}}.{{1, 0, 0}, {0, 1, 2 * BSdtheta}, {0, 0, 1}}.{{1, 0.2 * SR3toITMX, 0}, {0, 1, 0}, {0, 0, 1}};
Mrt = MMisfromITM.MITMmis.MMistoITM.{y, alpha, 1} /. solnMis;
Mrt /. {alpha → 0, ITMXdtheta → 0, SR3dtheta → 0, BSdtheta → 10^-3}
```

```
{-0.000052043 + 1.00034 y, -0.0702977 + 0.458234 y, 1.}
```

```
17.5 * 2 * 10^-3
```

```
0.035
```

```
0.2 * SR3toITMX * 4 * 10^-3 / 17.5
```

```
0.00112745
```

Beam Splitter Misalignment

```
MBS0 = Mrt /. {alpha → 0, ITMXdtheta → 0, SR3dtheta → 0, BSdtheta → 0 * 10^-3};
MBS1 = Mrt /. {alpha → 0, ITMXdtheta → 0, SR3dtheta → 0, BSdtheta → 1 * 10^-3};
```

```
MBS1 - MBS0
```

```
{-0.000052043, -0.0702977, 0.}
```

SR3 Misalignment

```
MSR30 = Mrt /. {alpha → 0, ITMXdtheta → 0, SR3dtheta → 0, BSdtheta → 0 * 10^-3};
MSR31 = Mrt /. {alpha → 0 * 10^-3,
    ITMXdtheta → 0 * 10^-3, SR3dtheta → 1 * 10^-3, BSdtheta → 0 * 10^-3};
```

```
MSR31 - MSR30
```

```
{-0.0000528074, -0.0713302, 0.}
```

Input change Misalignment

```
Min0 = Mrt /. {alpha → 0, ITMXdtheta → 0, SR3dtheta → 0, BSdtheta → 0 * 10^-3};
Min1 = Mrt /. {alpha → 1 * 10^-3,
    ITMXdtheta → 0 * 10^-3, SR3dtheta → 0 * 10^-3, BSdtheta → 0 * 10^-3};

Min1 - Min0
```

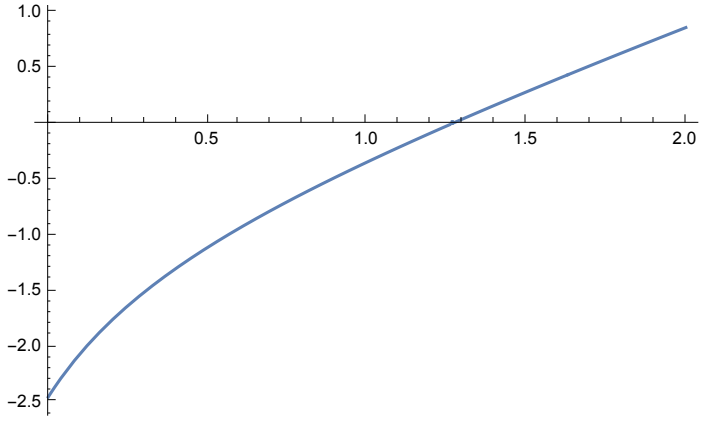
$$\{1.4809 \times 10^{-6}, 0.00100034, 0.\}$$

Retro-reflected beam on the table

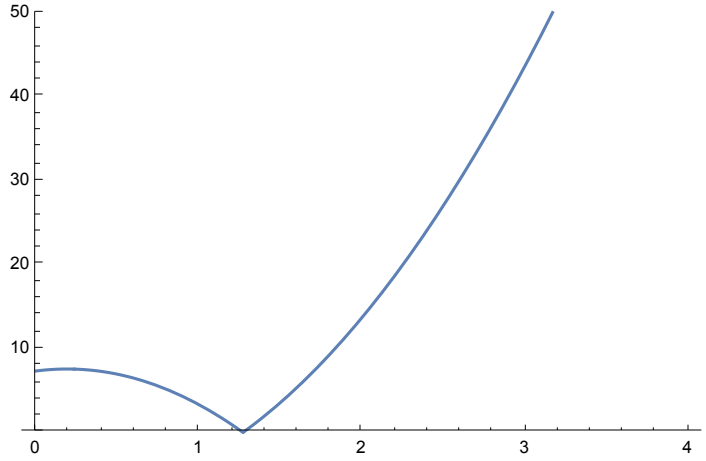
```
RRbeam = Simplify[{{1, AirLenstoHWS}, {0, 1}} . {{1, 0}, {-1/AirLens, 1}} . {{1, L2}, {0, 1}} . {{1, 0}, {1/1.7, 1}} .
      {{1, 0.05}, {0, 1}} . {{1, 0}, {1/1.7, 1}} . {{1, L2}, {0, 1}} . {{1, 0}, {-1/AirLens, 1}} . {{1, AirLenstoHWS}, {0, 1}}] /.
   {VacLens → -1.699, AirLens → 1.699, POtoVacLens → 2.156,
    VacLenstoAirLens → 2.808, AirLenstoHWS → 2.706};
RRbeam = Simplify[{{1, L2}, {0, 1}}] /. {VacLens → -1.699, AirLens → 1.699,
   POtoVacLens → 2.156, VacLenstoAirLens → 2.808, AirLenstoHWS → 2.706}
RRbeam = Simplify[{{1, L2}, {0, 1}} . {{1, 0}, {-1/AirLens, 1}} . {{1, 0.05}, {0, 1}} . {{1, 0}, {-1/AirLens, 1}} . {{1, L2}, {0, 1}}] /.
   {VacLens → -1.699, AirLens → 1.699, POtoVacLens → 2.156,
    VacLenstoAirLens → 2.808, AirLenstoHWS → 2.706};
```

$$\{\{1, L2\}, \{0, 1\}\}$$

```
RWOUT = OutRW[InputModeFromHWSRW[[2]], InputModeFromHWSRW[[1]], λIN, RRbeam];
Plot[RWOUT[[1]], {L2, 0, 2}]
```



```
RWOUT = OutRW[InputModeFromHWSRW[[2]], InputModeFromHWSRW[[1]], λIN, RRbeam];
Plot[RWOUT[[2]] * 10^3, {L2, 0, 4}, PlotRange → {0, 50}]
```



H1-ITMY Vertex Hartmann Sensor Optical Layout: [AirLens = 1.1337m, VacLens = 0.79362m] @ 820nm

30-Apr-2013: This solution was determined on paper after a sign error on SR2 was discovered in -v16. The new solution is analyzed, but not derived from scratch, here. The input beam size was reduced to 127mm based on the availability of collimating lenses.

Requirements

1. Image limiting aperture
2. Magnification = +1/17.5 x
3. Reference beam with own imaging device
4. Determine temperature sensitivity
5. Tolerancing on imaging optics

Get the Zemax Data - Import Units = mm, Convert to m

Optical Layout - Y Arm

Requirements

1. HWS = Conjugate Plane of BS
2. Beam Size at ITM = Sqrt[2]*ModeSize
3. Magnification = 17.5x
4. Put the telescope outside the vacuum??
5. Beam is approximately collimated at the HWS

H1-ITMY Probe Beam

Intialize

Set probe beam size at ITM = Sqrt[2]*17.5*CCD_diameter

```
BeamSizeAtITM = 127 / 1000
```

$$\frac{127}{1000}$$

```
nFusedSilica840 = 1.445;
CPthick = 130 * ConvertToM;
n1 = 1.0;
TotalDeMagnification = +17.5;
```

Create Function to Determine Output Radius and Beam Size

Optical Path - determine solution: AirLens = 1.1337 m, VacLens = 0.79362 m

Given Physical Parameters

Distances

```
D0901179 - Thickness of SR2 = 75mm
```

```
SR2RRT
SR2RRT[[3]] = 0.075
```

```
{0.075, 0.075, 0.0375}
```

```
0.075
```

```
SR3XYZ
```

```
{-0.15685, -19.6148, 0.099061}
```

```
SR2toSR3 = 15.443 (*T1300438*)
SR3toITMY = 24.7836 (*T1300438*)
SR3toBS = 19.4313 (*T1300438*)
```

```
15.443
```

```
24.7836
```

```
19.4313
```

Curvatures - E0900095, T080268, E080518,

```
ITMRoC = 1934 / nFusedSilica840;
SR3RoC = 36;
SR2RoC = 6.43;
```

ABCD Matrices - Not relying on solution

Requirement 2: The mode from the ITM, including the size of Sqrt[2]*Cavity-ModeSize is defined below

Test Gaussian beam in and out

ITM to SR2_AR: From HR surface outside ITM to AR surface outside SR2 [in m]

```
ITMToSR2AR = {{1, 0}, {0, nFusedSilica840/n1}}.{{1, SR2RRT[[3]]}, {0, 1}}.
    {{1, 0}, {(n1 - nFusedSilica840)/(SR2RoC*nFusedSilica840), n1/nFusedSilica840}}.{{1, SR2toSR3}, {0, 1}}.{{1, 0}, {-2/SR3RoC, 1}}.{{1, SR3toITMY}, {0, 1}};
MatrixForm[ITMToSR2AR]
```

$$\begin{pmatrix} 0.138662 & 18.876 \\ -0.0653868 & -1.68928 \end{pmatrix}$$

BS to SR2_AR: ABCD Matrix - From BS to AR

Locate Nearest Conjugate Plane of BS

```
ConjPlaneM = {{1, x}, {0, 1}}.(BSToSR2AR);
soln1 = Solve[ConjPlaneM[[1]][[2]] == 0];
xConj = x /. soln1[[1]];
magnifcationBS1 = ConjPlaneM[[1]][[1]] /. soln1[[1]];
Print["Nearest conjugate plane of BS is ", xConj, "m from AR surface"]
Print["DeMagnification to First BS Conjugate Plane =  ",
 1/magnifcationBS1, " x"]
Print["Residual magnification to still be achieved =  ",
 TotalDeMagnification*magnifcationBS1, " x"]
```

```
Nearest conjugate plane of BS is 13.5397m from AR surface
```

```
DeMagnification to First BS Conjugate Plane =  -1.33931 x
```

```
Residual magnification to still be achieved =  -13.0664 x
```

```
ResidualDeMag = TotalDeMagnification * magnifcationBS1
NearestConjPlane = xConj
```

```
-13.0664
```

```
13.5397
```

Extract Optical Information - Conjugate plane, beam size At SR2

```
solnList = {2.2826, 2.9695, 3.6715, 1.13366, 0.79362}
H1YSoln = solnList;
```

```
{2.2826, 2.9695, 3.6715, 1.13366, 0.79362}
```

Get Output Mode from SR2

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840 * 10^-9;
OutRW[wIN, RIN, λIN, {{1, 100}, {0, 1}}];
pOut = OutRW[wIN, RIN, λIN, ITMToSR2AR];
OutR = pOut[[1]];
OutW = pOut[[2]];
OutModeSR2AR = pOut;
FindNearestWaist[pOut[[2]], pOut[[1]], λIN];
```

```
Solve::ratnz: Solve was unable to solve the system with inexact coefficients
    The answer was obtained by solving a corresponding exact system and numericizing the result >>
```

```
Must create mode - matching optics to produce this mode
Must create imaging optics to image the limiting aperture - the beam split-
ter - correctly
```

```
Print["Probe beam size at AR surface of SR2 = ", OutW * 1000, " mm."]
Print["Curvature at AR surface of SR2 = ", OutR * 1000, " mm."]
```

```
Probe beam size at AR surface of SR2 = 19.4012 mm.
```

```
Curvature at AR surface of SR2 = -2292.09 mm.
```

```
SOutput = OutModeSR2AR[[1]]
```

```
-2.29209
```

Demonstrate curvature sign by propagating a short distance further

```
pOutdz = OutRW[wIN, RIN, λIN, {{1, 0.1}, {0, 1}}.ITMToSR2AR];
OutRdz = pOutdz[[1]];
OutWdz = pOutdz[[2]];
Print["Probe beam size at AR surface of SR2 = ", OutW * 1000, " mm."]
Print["Probe beam size 100mm out of IFO from SR2-AR = ", OutWdz * 1000, " mm."]
```

```
Probe beam size at AR surface of SR2 = 19.4012 mm.
```

```
Probe beam size 100mm out of IFO from SR2-AR = 18.5548 mm.
```

Analytic Solution for imaging a conjugate plane with an arbitrary telescope

Solve for the solution - ITMY: Magnification = +ve!!

```
Just work out the solutions for the ABCD matrices as follows:
1. B = 0
2. C = 0
3. A = 1/17.5

eq1: Collimation requirement - at output in paraxial approximation
eq2: Total magnification requirement
eq3: Imaging requirement
```

```
Use the analytic solution determined above.
Notes:
1. The combination of lenses was chosen by first constraining the parame-
ter space of possible solutions to those where
  a. {z1, z2, z3} >= {3, 5, 0}m
  b. dS/dT of z2 <= 20x smaller than the maximum defocus error of 4E-4 m^-1
2. Once this region was identified the lenses were chosen to have conve-
nient values and to yield a solution close to the minimum Abs[dS/dT]

Hence: f1 = 1.0m, f2 = 3.0m
```

```
Z1A = solnList[[3]];
Z2A = solnList[[2]];
Z3A = solnList[[1]];
F1A = solnList[[5]];
F2A = solnList[[4]];
AllSolns2 = { F1A, F2A, Z1A, Z2A, Z3A}
TestSolutionSubstitution = {f1 → solnList[[5]], f2 → solnList[[4]],
  z1 → solnList[[3]], z2 → solnList[[2]], z3 → solnList[[1]]}
```

```
{0.79362, 1.13366, 3.6715, 2.9695, 2.2826}
```

```
{f1 → 0.79362, f2 → 1.13366, z1 → 3.6715, z2 → 2.9695, z3 → 2.2826}
```

ABCD Matrices - Dependent on solution

SR2_AR to HWS: ABCD matrix - get the imaging matrix part - after SR2

```
ImgMatrix = Simplify[{{1, z3}, {0, 1}}.{{1, 0}, {-1/f2, 1}}.{{1, z2}, {0, 1}}.{{1, 0}, {-1/f1, 1}}.{{1, z1}, {0, 1}}];
SR2ARToHWS = ImgMatrix /. TestSolutionSubstitution;
MatrixForm[SR2ARToHWS]
```

$$\begin{pmatrix} -0.0975183 & -1.08496 \\ 1.15841 & 2.63373 \end{pmatrix}$$

HWS to SR2_AR: ABCD matrix - get the imaging matrix part - after SR2

```
ImgMatrix = Simplify[{{1, z1}, {0, 1}}.{{1, 0}, {-1/f1, 1}}.{{1, z2}, {0, 1}}.{{1, 0}, {-1/f2, 1}}.{{1, z3}, {0, 1}}];
HWSToSR2AR = ImgMatrix /. TestSolutionSubstitution;
MatrixForm[HWStoSR2AR]
```

HWStoSR2AR

ITM to HWS: ABCD matrix coming out of the vacuum - For Gaussian beam mode checking

```
ITMToHWS = SR2ARToHWS.ITMToSR2AR;
MatrixForm[ITMToHWS];
HWSToITM = {{1, SR3toITMY}, {0, 1}}.{{1, 0}, {-2/SR3RoC, 1}}.{{1, SR2toSR3}, {0, 1}}.
    {{1, 0}, {(nFusedSilica840-n1)/(-SR2RoC*n1), nFusedSilica840/n1}}.{{1, SR2RRT[[3]]}, {0, 1}}.{{1, 0}, {0, n1/nFusedSilica840}}.{{1, z1}, {0, 1}}.
    {{1, 0}, {-1/f1, 1}}.{{1, z2}, {0, 1}}.{{1, 0}, {-1/f2, 1}}.{{1, z3}, {0, 1}} /. TestSolutionSubstitution;
MITM = {{1, 0}, {-2/-ITMRoC, 1}};
MHWSv = ITMToHWS.MITM.HWSToITM
```

{{0.999977, -0.000912376}, {0.0498203, 0.999977}}

Alignment Laser?

```
wIN = 0.001
RIN = 1000
OutRW[wIN, RIN, λIN, MHWSv]
0.001 * 17.5
```

0.001

1000

{19.702, 0.000999976}

0.0175

BS to HWS: ABCD matrix - Beam splitter to HWS - Shows Imaging

```
BS2HWS = SR2ARToHWS.BSToSR2AR;
MatrixForm[BS2HWS]
MatrixForm[ITMToHWS]
```

$$\begin{pmatrix} 0.0574202 & -0.315275 \\ -0.0115831 & 17.4791 \end{pmatrix}$$

$$\begin{pmatrix} 0.0574202 & -0.00794556 \\ -0.0115831 & 17.4171 \end{pmatrix}$$

Requirement 1: The Beam Splitter to HWS ABCD matrix has a value of B approximately equal to 0.

Do a Gaussian beam propagation

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOut = OutRW[wIN, RIN, λIN, ITMToHWS]
OutR = pOut[[1]];
OutW = pOut[[2]]
GaussianMag = wIN / OutW
```

```
{40.1439, 0.00729161}
```

```
0.00729161
```

```
17.4173
```

Requirement 3: The beam size at the HWS is 17.5x smaller than at the beam splitter

Define the Input Mode

```
InputMode = {OutW, -OutR}
InputModeFromHWS = InputMode;
```

```
{0.00729161, -40.1439}
```

Propagate modes - from ITM to outside and then back again

Propagate the ITM mode from the ITM to the HWS

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
Print["Probe beam size at ITM = ", wIN * 1000, " mm."]
Print["Curvature at ITM = ", RIN, " m."]
pOut = OutRW[wIN, RIN, λIN, ITMToHWS];
Print["Probe beam size at HWS = ", pOut[[2]] * 1000, " mm."]
Print["Curvature at HWS = ", pOut[[1]] * 1000, " mm."]
Print["ITM_Beam_Size/HWS_Beam_Size = ", wIN / pOut[[2]], " x."]
```

```
Probe beam size at ITM = 127 mm.
```

```
Curvature at ITM = 1338.41 m.
```

```
Probe beam size at HWS = 7.29161 mm.
```

```
Curvature at HWS = 40143.9 mm.
```

```
ITM_Beam_Size/HWS_Beam_Size = 17.4173 x.
```

Propagate the HWS mode to the ITM

```
pOut = OutRW[pOut[[2]], -pOut[[1]], λIN, HWSToITM];
Print["Probe beam size at ITM = ", pOut[[2]] * 1000, " mm."]
Print["Curvature at ITM = ", pOut[[1]], " m."]
```

```
Probe beam size at ITM = 127. mm.

Curvature at ITM = -1338.41 m.
```

Propagate from HWS mode to HWS

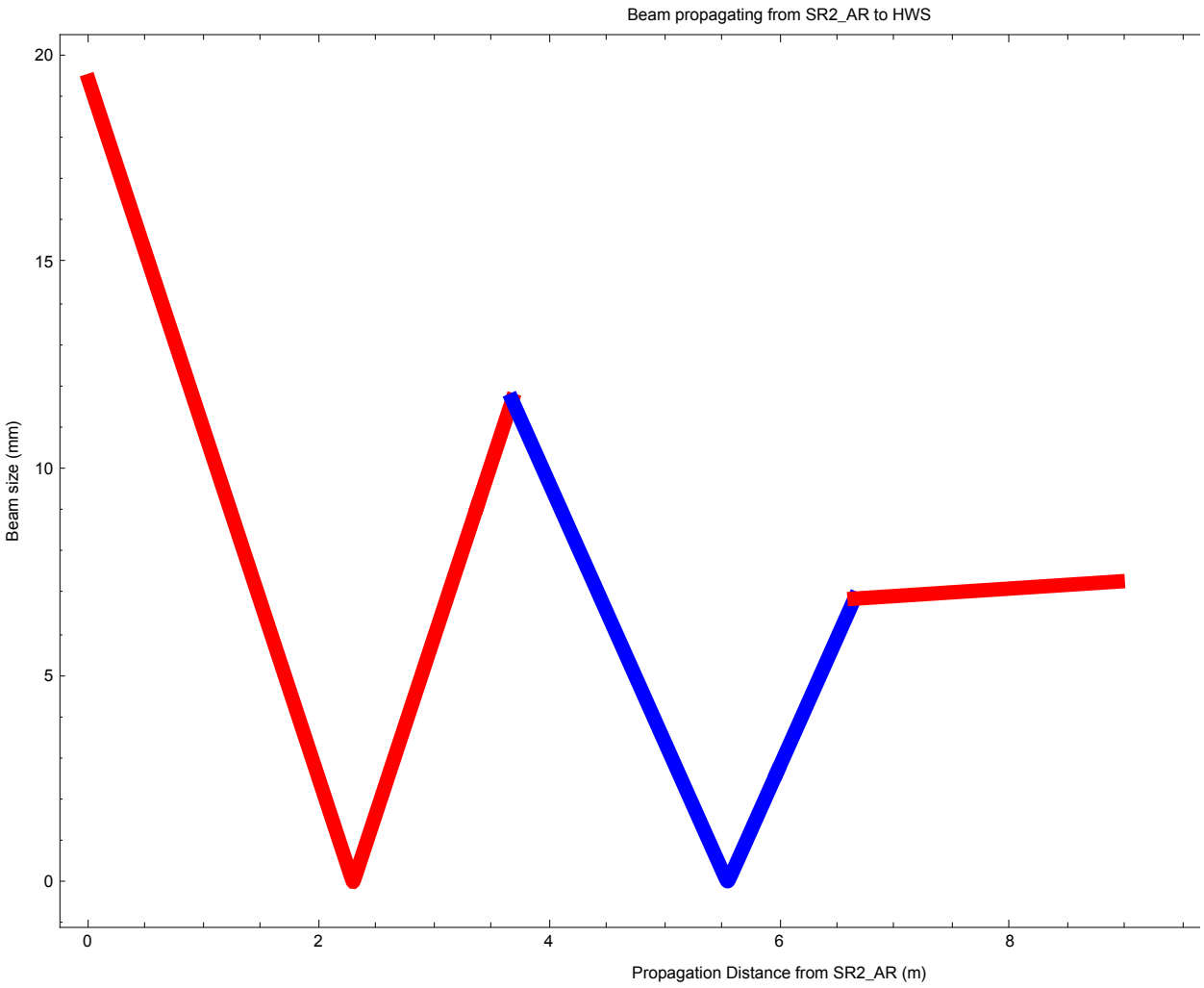
```
Print["Probe beam size at ITM = ", InputMode[[1]] * 1000, " mm."]
Print["Curvature at ITM = ", InputMode[[2]], " m."]
pOut = OutRW[InputMode[[1]], InputMode[[2]], λIN, MHWSv];
Print["Probe beam size at HWS = ", pOut[[2]] * 1000, " mm."]
Print["Curvature at HWS = ", pOut[[1]], " m."]
```

```
Probe beam size at ITM = 7.29161 mm.

Curvature at ITM = -40.1439 m.

Probe beam size at HWS = 7.29161 mm.

Curvature at HWS = 40.1439 m.
```

Plot the Solution

Plot solution from SR2_AR to HWS

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;

p1 = Plot[(OutRW[wIN, RIN, λIN, {{1, x}, {0, 1}}.ITMToSR2AR])[[2]] * 1000,
    {x, 0, Z1A}, PlotStyle → {Red, Thickness[0.01]}];
BS1 = Table[{x, (OutRW[wIN, RIN, λIN, {{1, x}, {0, 1}}.ITMToSR2AR])[[2]] * 1000},
    {x, 0, Z1A, Z1A / 200}];
p2 = Plot[(OutRW[wIN, RIN, λIN, {{1, x - (Z1A)}, {0, 1}}.
          {{1, 0}, {-1 / F1A, 1}}.{{1, (Z1A)}, {0, 1}}.ITMToSR2AR])[[2]] * 1000,
    {x, (Z1A), (Z1A) + Z2A}, PlotStyle → {Blue, Thickness[0.01]}];
BS2 = Table[{x, (OutRW[wIN, RIN, λIN, {{1, x - (Z1A)}, {0, 1}}.{{1, 0}, {-1 / F1A, 1}}.{{1, (Z1A)}, {0, 1}}.
          ITMToSR2AR])[[2]] * 1000}, {x, (Z1A), (Z1A) + Z2A, (Z2A / 200)}];
p3 = Plot[(OutRW[wIN, RIN, λIN, {{1, x - (Z1A) - Z2A}, {0, 1}}.{{1, 0}, {-1 / F2A, 1}}.
          {{1, Z2A}, {0, 1}}.{{1, 0}, {-1 / F1A, 1}}.{{1, (Z1A)}, {0, 1}}.ITMToSR2AR])[[2]] * 1000,
    {x, (Z1A) + Z2A, (Z1A) + Z2A + Z3A}, PlotStyle → {Red, Thickness[0.01]}];
BS3 = Table[{x, (OutRW[wIN, RIN, λIN, {{1, x - (Z1A) - Z2A}, {0, 1}}.{{1, 0}, {-1 / F2A, 1}}.
          {{1, Z2A}, {0, 1}}.{{1, 0}, {-1 / F1A, 1}}.{{1, (Z1A)}, {0, 1}}.ITMToSR2AR])[[2]] * 1000},
    {x, (Z1A) + Z2A, (Z1A) + Z2A + Z3A, Z3A / 200}];
ITMYPlottedSolutionFromSR2AR = Show[p1, p2, p3, Frame → True,
  FrameLabel → {"Propagation Distance from SR2_AR (m)", "Beam size (mm)",
    "Beam propagating from SR2_AR to HWS"}, PlotRange → {{0, 12}, Automatic}]
```




```
BS = BS1;
For[ii = 1, ii ≤ Length[BS2], ii++, {
    BS = Append[BS, BS2[[ii]]];
  }];
For[ii = 1, ii ≤ Length[BS3], ii++, {
    BS = Append[BS, BS3[[ii]]];
  }];
Export["/Advanced_LIGO/TCS/Hartmann sensor/H1_HWSY_beam_size.txt", BS, "Table"];
```

$$\left(\text{OutRW}\left[\text{wIN, RIN, }\lambda\text{IN, }\begin{pmatrix}1 & x\\ 0 & 1\end{pmatrix}.\text{ITMToSR2AR}\right]\right)[[2]] * 1000 /. x \to \text{Z1A}$$

$$\left(\text{OutRW}\left[\text{wIN, RIN, }\lambda\text{IN, }\begin{pmatrix}1 & x-(\text{Z1A})\\ 0 & 1\end{pmatrix}.\begin{pmatrix}1 & 0\\ -1/\text{F1A} & 1\end{pmatrix}.\begin{pmatrix}1 & (\text{Z1A})\\ 0 & 1\end{pmatrix}.\text{ITMToSR2AR}\right]\right)[[2]] * 1000 /. x \to \text{Z1A}$$

```
11.676
```

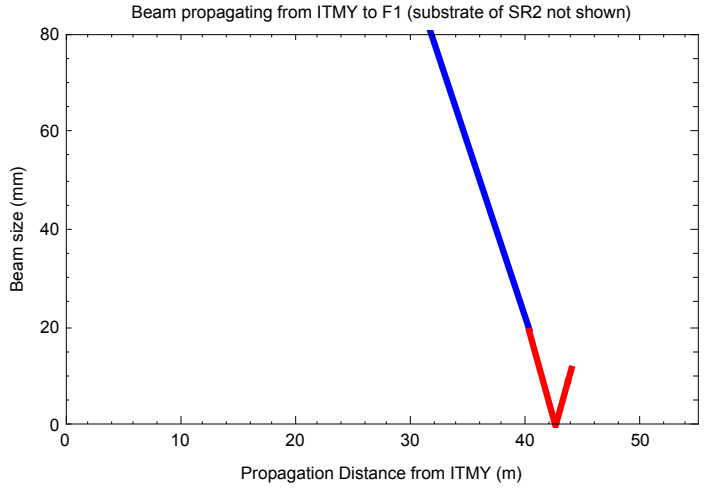
```
11.676
```

```
Z1A
```

```
3.6715
```

Plot from ITMY to SR2

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
plotR = {{0, 55}, {0, 80}};
p1 = Plot[{(OutRW[wIN, RIN, λIN, {{1, x}, {0, 1}}])[[2]] * 1000,
    {x, 0, SR3toITMY}, PlotStyle → {Red, Thickness[0.01]}, PlotRange → plotR};
p2 = Plot[{(OutRW[wIN, RIN, λIN, {{1, x - SR3toITMY}, {0, 1}}] .
          {{1, 0}, {-2 / SR3RoC, 1}} . {{1, SR3toITMY}, {0, 1}})[[2]] * 1000,
    {x, SR3toITMY, SR3toITMY + SR2toSR3}, PlotStyle → {Blue, Thickness[0.01]}};
p3 = Plot[{(OutRW[wIN, RIN, λIN, {{1, x - SR3toITMY - SR2toSR3}, {0, 1}}] .ITMToSR2AR)[[2]] *
     1000, {x, SR3toITMY + SR2toSR3, SR3toITMY + SR2toSR3 + Z1A},
    PlotStyle → {Red, Thickness[0.01]}, PlotRange → plotR};
ITMYPlottedSolutionFromITMY = Show[p1, p2, p3, Frame → True,
  FrameLabel → {"Propagation Distance from ITMY (m)", "Beam size (mm)",
    "Beam propagating from ITMY to F1 (substrate of SR2 not shown)"}]
```



Plot the whole propagation

```
z1 =.;
```

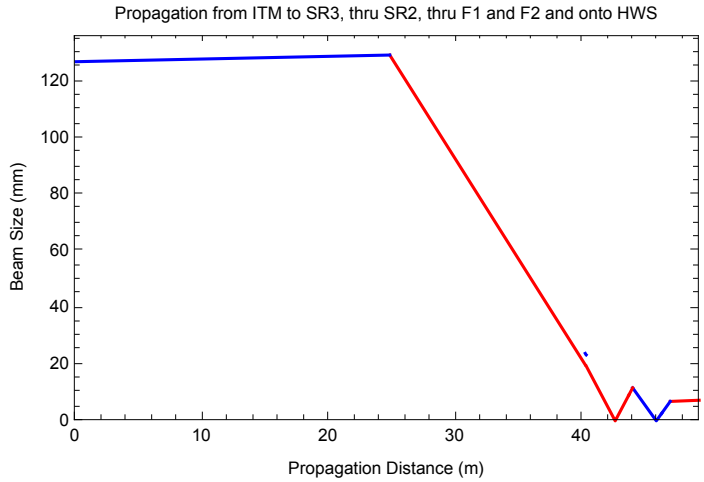
```
TestSolutionSubstitution
```

```
{f1 → 0.79362, f2 → 1.13366, z1 → 3.6715, z2 → 2.9695, z3 → 2.2826}
```

```
xConj
```

```
13.5397
```

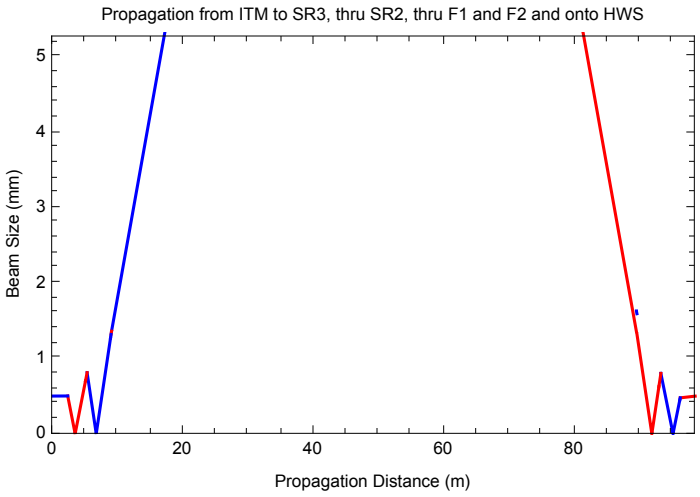
```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
plot1 = PlotTheBeamPropagation[
    {{{1, SR3toITMY}, {0, 1}}, {{1, 0}, {-2/SR3RoC, 1}}, {{1, SR2toSR3}, {0, 1}}, {{1, 0}, {(n1-nFusedSilica840)/(SR2RoC*nFusedSilica840), n1/nFusedSilica840}},
     {{1, SR2RRT[[3]]}, {0, 1}}, {{1, 0}, {0, nFusedSilica840/n1}}, {{1, z1}, {0, 1}}, {{1, 0}, {-1/f1, 1}}, {{1, z2}, {0, 1}},
     {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /. TestSolutionSubstitution, {RIN, wIN}, 840 * 10^-9];
Show[plot1, FrameLabel → {"Propagation Distance (m)", "Beam Size (mm)",
   "Propagation from ITM to SR3, thru SR2, thru F1 and F2 and onto HWS"}]
```




```
MHWSv
RIN = -4.0;
wIN = 0.5 * 10^-3;
nm = 10^-9;
pOut = OutRW[wIN, RIN, 532 nm, MHWSv];
Print["Probe beam size at HWS = ", pOut[[2]] * 1000, " mm."]
Print["Curvature at HWS = ", pOut[[1]] * 1000, " mm."]
Print["ITM_Beam_Size/HWS_Beam_Size = ", wIN / pOut[[2]], " x."]
```

```
{{0.999977, -0.000912376}, {0.0498203, 0.999977}}

Probe beam size at HWS = 0.500103 mm.

Curvature at HWS = -4986.26 mm.

ITM_Beam_Size/HWS_Beam_Size = 0.999794 x.
```

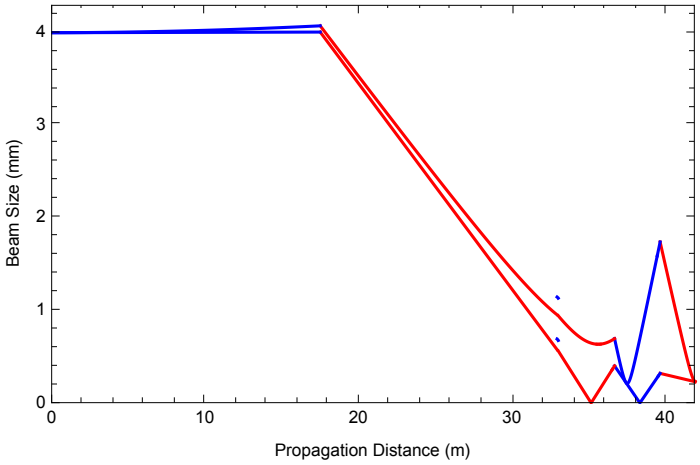
```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
plot1 =
  PlotTheBeamPropagation[{{{1, z3}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, z2}, {0, 1}}, {{1, 0}, {-1/f1, 1}}, {{1, z1}, {0, 1}},
    {{1, SR2RRT[[3]]}, {0, 1}}, {{1, 0}, {0, n1/nFusedSilica840}}, {{1, 0}, {(nFusedSilica840-n1)/(-SR2RoC*n1), nFusedSilica840/n1}},
    {{1, SR2toSR3}, {0, 1}}, {{1, 0}, {-2/SR3RoC, 1}}, {{1, SR3toITMY}, {0, 1}}, MITM, {{1, SR3toITMY}, {0, 1}}, {{1, 0}, {-2/SR3RoC, 1}},
    {{1, SR2toSR3}, {0, 1}}, {{1, 0}, {(n1-nFusedSilica840)/(SR2RoC*nFusedSilica840), n1/nFusedSilica840}}, {{1, SR2RRT[[3]]}, {0, 1}},
    {{1, 0}, {0, nFusedSilica840/n1}}, {{1, z1}, {0, 1}}, {{1, 0}, {-1/f1, 1}}, {{1, z2}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /.
   TestSolutionSubstitution, {1000.0, 5*10^-4}, 0.0532 nm];
A1 = Show[plot1, FrameLabel → {"Propagation Distance (m)", "Beam Size (mm)",
    "Propagation from ITM to SR3, thru SR2, thru F1 and F2 and onto HWS"},
  PlotRange → {0, 5}]
```



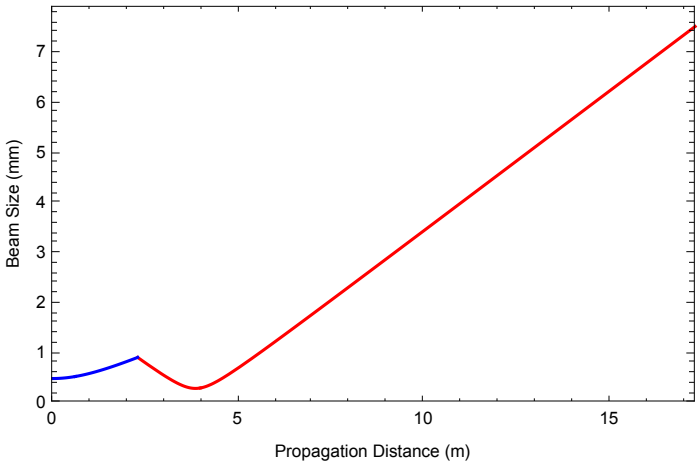

```
Pi * (5 * 10^-4)^2. / (532 nm)
```

```
1.47631
```

```
plot1 = PlotTheBeamPropagation[{{{1, SR3toBS - 2}, {0, 1}}, {{1, 0}, {-2/SR3RoC, 1}},
    {{1, SR2toSR3}, {0, 1}}, {{1, 0}, {(n1 - nFusedSilica840)/(SR2RoC * nFusedSilica840), n1/nFusedSilica840}},
    {{1, SR2RRT[[3]]}, {0, 1}}, {{1, 0}, {0, nFusedSilica840/n1}}, {{1, z1}, {0, 1}},
    {{1, 0}, {-1/f1, 1}}, {{1, z2}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /.
   TestSolutionSubstitution, {10^4, 0.004}, 532 * 10^-9];
plot2 = PlotTheBeamPropagation[{{{1, SR3toBS - 2}, {0, 1}}, {{1, 0}, {-2/SR3RoC, 1}}, {{1, SR2toSR3}, {0, 1}},
    {{1, 0}, {(n1 - nFusedSilica840)/(SR2RoC * nFusedSilica840), n1/nFusedSilica840}},
    {{1, SR2RRT[[3]]}, {0, 1}}, {{1, 0}, {0, nFusedSilica840/n1}}, {{1, z1}, {0, 1}},
    {{1, 0}, {-1/f1, 1}}, {{1, z2}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /.
   TestSolutionSubstitution, {10^4, 0.004}, 532 * 10^-12];
Show[plot1, plot2, PlotRange → All]
```



```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
plot1 = PlotTheBeamPropagation[{{{1, z3}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, 15}, {0, 1}}} /.
    TestSolutionSubstitution, {1000.0, 0.5 * 10^-3}, 532 nm];
Show[plot1, PlotRange → All]
```



Tolerancing on Y arm solution

Get the Variable matrix

```
SR3toHWSVar = {{1, Z3A + dz3}, {0, 1}} . {{1, 0}, {-1 / F2A - (dSf2), 1}} . {{1, Z2A + dZ2A}, {0, 1}} .
    {{1, 0}, {-1 / F1A - (dSf1), 1}} . {{1, Z1A + dz1}, {0, 1}} . {{1, 0}, {nFusedSilica840/n1, 1}} . {{1, SR2RRT[[3]]}, {0, 1}} .
    {{1, 0}, {(n1 - nFusedSilica840)/(SR2RoC * nFusedSilica840), n1/nFusedSilica840}} . {{1, SR2toSR3}, {0, 1}} . {{1, 0}, {-2/SR3RoC, 1}};
BStoHWSVar = SR3toHWSVar . {{1, SR3toBS}, {0, 1}};
ITMYtoHWSVar = SR3toHWSVar . {{1, SR3toITMY}, {0, 1}};
ITMXtoHWSVar = .;
```

Variation in imaging (B = 0) with length changes: (assume that FL variations can be compensated for) - NO ISSUES - Easy to work with!

```
M1 = BStoHWSVar . {{1, x}, {0, 1}};
B1 = M1[[1]][[2]];
```

Test variation in position of F1 - solve for the position of the conjugate plane - VERY SENSITIVE

Set all delta values = 0 except for dZ2A and dz1. Set these last two to the same value with opposite sign to model a lens out of position

```
B1A = Simplify[B1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → X1, dz1 → -X1}];
Solve[B1A == 0, x];
ConjPlaneDisplacement = x /. %[[1]];
NominalConjugatePlane = ConjPlaneDisplacement /. X1 → 0;
Print["Conjugate Plane is nominally out of position by ",
  NominalConjugatePlane, "m"];
RateCPD = (D[ConjPlaneDisplacement, X1]) /. X1 → 0;
Print["Conjugate Plane displacement for +/- 5 mm displacement of F1 = ",
  RateCPD * 1000 * 0.005, "mm"];
BTestF1OutOfPosition = {RateCPD * 1000 * 0.005, "mm", 5, "mm"};
```

```
Conjugate Plane is nominally out of position by 5.49068m
```

```
Conjugate Plane displacement for +/- 5 mm displacement of F1 = -1543.23mm
```

Hence the conjugate plane is VERY dependent on the position of F1: basically see a 10x displacement in the position of the conjugate plane for a displacement of F1 out of position.

Test variation in position of F2 - solve for the position of the conjugate plane - INSENSITIVE to displacement of F2 out of position

Set all delta values = 0 except for dZ2A and dz3. Set these last two to the same value with opposite sign to model a lens out of position

```
B1A = Simplify[B1 /. {dz1 → 0, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz3 → X1}];
Solve[B1A == 0, x];
ConjPlaneDisplacement = x /. %[[1]];
NominalConjugatePlane = ConjPlaneDisplacement /. X1 → 0;
Print["Conjugate Plane is nominally out of position by ",
  NominalConjugatePlane, "m"];
RateCPD = (D[ConjPlaneDisplacement, X1]) /. X1 → 0;
Print["Conjugate Plane displacement for +/- 5 mm displacement of F2 = ",
  RateCPD * 0.005, "m"];
BTestF2OutOfPosition = {RateCPD * 1000 * 0.005, "mm", 5, "mm"};
```

```
Conjugate Plane is nominally out of position by 5.49068m
```

```
Conjugate Plane displacement for +/- 5 mm displacement of F2 = 0.0411556m
```

Test variation in F1-F2 distance - SENSITIVE

```
B1A = Simplify[B1 /. {dz1 → 0, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz3 → 0}];
Solve[B1A == 0, x];
ConjPlaneDisplacement = x /. %[[1]];
NominalConjugatePlane = ConjPlaneDisplacement /. X1 → 0;
Print["Conjugate Plane is nominally out of position by ",
  NominalConjugatePlane, "m"];
RateCPD = (D[ConjPlaneDisplacement, X1]) /. X1 → 0;
Print["Conjugate Plane displacement for +/- 5 mm variation
    in distance between F1 and F2 = ", RateCPD * 0.005, "m"];
BTestF1F2Distance = {RateCPD * 1000 * 0.005, "mm", 5, "mm"};
```

```
Conjugate Plane is nominally out of position by 5.49068m
```

```
Conjugate Plane displacement for +/- 5 mm variation in distance between F1 and F2 =
 1.55765m
```

Hence the conjugate plane is sensitive to the distance between F2 and F1 - but that's due to F1 displacement

Test variation in position of HWS - SENSITIVE = GOOD for tuning. Possibly bad for fringes

Set all delta values = 0 except for dz3.

```
B1A = Simplify[B1 /. {dz1 → 0, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz3 → X1}];
Solve[B1A == 0, x];
ConjPlaneDisplacement = x /. %[[1]];
NominalConjugatePlane = ConjPlaneDisplacement /. X1 → 0;
Print["Conjugate Plane is nominally out of position by ",
  NominalConjugatePlane, "m"];
RateCPD = (D[ConjPlaneDisplacement, X1]) /. X1 → 0;
Print["Conjugate Plane displacement for +/- 5 mm displacement of F2 = ",
  RateCPD * 0.005, "m"];
BTestF2HWSDistance = {RateCPD * 1000 * 0.005, "mm", 5, "mm"};
```

```
Conjugate Plane is nominally out of position by 5.49068m
```

```
Conjugate Plane displacement for +/- 5 mm displacement of F2 = -1.5165m
```

Hence the conjugate plane is totally sensitive to the position of HWS - THIS IS A GOOD THING - means it is easy to tune.

Variation in magnification (A = 1/17.5) with length changes: (assume that FL variations can be compensated for) - NO ISSUES - Easy to work with!

```
M1 = ITMYtoHWSVar;
A1 = M1[[1]][[1]];
```

Test variation in position of F1 - determine the magnification - INSENSITIVE

Set all delta values = 0 except for dZ2A and dz1. Set these last two to the same value with opposite sign to model a lens out of position

```
A1A = Simplify[A1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → X1, dz1 → -X1}];
NominalMagnification = 1 / A1A /. X1 → 0;
Print["Nominal paraxial magnification is ", NominalMagnification, "m"];
RateVariationInMag = (D[A1A, X1]) /. X1 → 0;
Print["Variation in magnification for +/- 5 mm displacement of F1 = ",
  RateVariationInMag * 0.005, "x"];
```

```
Nominal paraxial magnification is 17.4155m

Variation in magnification for +/- 5 mm displacement of F1 = -0.000348034x
```

Hence the paraxial magnification is VERY invariant to the position of F1

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0})];
BeamWAtHWSNom = pOutNom[[2]];
MOut = (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → X1, dz1 → -X1});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamWAtHWSX1 = pOutX1[[2]];
MagNom = BeamSizeAtITM / BeamWAtHWSNom;
MagX1 = BeamSizeAtITM / BeamWAtHWSX1;
Print["Absolute Variation in magnification for +5mm displacement = ",
  MagX1 - MagNom, "x"];
Print["Relative Variation in magnification for +5mm displacement = ",
  (MagX1 - MagNom) / MagNom * 100, "%"];
ATestF1OutOfPosition = {Round[(MagX1 - MagNom) / MagNom * 100 * 10] / 10.0, "%",  5, "mm"};
```

```
Absolute Variation in magnification for +5mm displacement = 0.0852543x

Relative Variation in magnification for +5mm displacement = 0.489481%
```

Hence the Gaussian magnification is VERY invariant to the position of F1

Test variation in position of F2 - determine the magnification - MARGINALLY SENSITIVE

Set all delta values = 0 except for dL1 and dz3. Set these last two to the same value with opposite sign to model a lens out of position

```
A1A = Simplify[A1 /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz1 → 0}];
NominalMagnification = 1 / A1A /. X1 → 0;
Print["Nominal paraxial magnification is ", NominalMagnification, "m"];
RateVariationInMag = (D[A1A, X1]) /. X1 → 0;
Print["Variation in magnification for +/- 5 mm displacement of F2 = ",
  RateVariationInMag * 0.005, "x"];
```

```
Nominal paraxial magnification is 17.4155m

Variation in magnification for +/- 5 mm displacement of F2 = 0.000258236x
```

Hence the paraxial magnification is VERY invariant to the position of F2

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0})];
BeamWAtHWSNom = pOutNom[[2]];
MOut = (ITMYtoHWSVar /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz1 → 0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamWAtHWSX1 = pOutX1[[2]];
MagNom = BeamSizeAtITM / BeamWAtHWSNom;
MagX1 = BeamSizeAtITM / BeamWAtHWSX1;
Print["Absolute Variation in magnification for +5mm displacement = ",
  MagX1 - MagNom, "x"];
Print["Relative Variation in magnification for +5mm displacement = ",
   (MagX1 - MagNom) / MagNom * 100, "%"];
ATestF2OutOfPosition = {Round[(MagX1 - MagNom) / MagNom * 100 * 10] / 10.0, "%",  5, "mm"};

Absolute Variation in magnification for +5mm displacement = -0.0777756x

Relative Variation in magnification for +5mm displacement = -0.446543%
```

Hence the Gaussian magnification is insensitive to the position of F2

Test variation in F1-F2 distance: determine the magnification - INSENSITIVE

```
A1A = Simplify[A1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz1 → 0}];
NominalMagnification = 1 / A1A /. X1 → 0;
Print["Nominal paraxial magnification is ", NominalMagnification, "m"];
RateVariationInMag = (D[A1A, X1]) /. X1 → 0;
Print["Variation in magnification for +/- 5 mm displacement of F2 = ",
  RateVariationInMag * 0.005, "x"];

Nominal paraxial magnification is 17.4155m

Variation in magnification for +/- 5 mm displacement of F2 = 0.000316152x
```

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0})];
BeamWAtHWSNom = pOutNom[[2]];
MOut = (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz1 → 0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamWAtHWSX1 = pOutX1[[2]];
MagNom = BeamSizeAtITM / BeamWAtHWSNom;
MagX1 = BeamSizeAtITM / BeamWAtHWSX1;
Print[
  "Absolute Variation in mag. for +5mm distance change between F1 and F2 = ",
  MagX1 - MagNom, "x"];
Print["Relative Variation in mag. for +5mm distance change between F1 and F2 = ",
  (MagX1 - MagNom) / MagNom * 100, "%"];
ATestF1F2Distance = {Round[(MagX1 - MagNom) / MagNom * 100 * 10] / 10.0, "%",  5,  "mm"};
```

```
Absolute Variation in mag. for +5mm distance change between F1 and F2 = -0.0752991x

Relative Variation in mag. for +5mm distance change between F1 and F2 = -0.432324%
```

Hence the Gaussian magnification is very insensitive to the distance between F1 and F2

Test variation in position of HWS - determine the magnification - INSENSITIVE to position of HWS: 1% variation per 15mm

Set all delta values = 0 except for dZ2A and dz1. Set these last two to the same value with opposite sign to model a lens out of position

```
A1A = Simplify[A1 /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0}];
NominalMagnification = 1 / A1A /. X1 → 0;
Print["Nominal paraxial magnification is ", NominalMagnification, "m"];
RateVariationInMag = (D[A1A, X1]) /. X1 → 0;
Print["Variation in magnification for +/- 5 mm displacement of HWS = ",
  RateVariationInMag * 0.005, "x"];
```

```
Nominal paraxial magnification is 17.4155m

Variation in magnification for +/- 5 mm displacement of HWS = -0.0000579154x
```

Hence the paraxial magnification is SUPER invariant to the position of F1

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0})];
BeamWAtHWSNom = pOutNom[[2]];
MOut = (ITMYtoHWSVar /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamWAtHWSX1 = pOutX1[[2]];
MagNom = BeamSizeAtITM / BeamWAtHWSNom;
MagX1 = BeamSizeAtITM / BeamWAtHWSX1;
Print["Absolute Variation in magnification for +5mm displacement = ",
  MagX1 - MagNom, "x"];
Print["Relative Variation in magnification for +5mm displacement = ",
  (MagX1 - MagNom) / MagNom * 100, "%"];
ATestF2HWSDistance = {Round[(MagX1 - MagNom) / MagNom * 100 * 10] / 10.0, "%",  5, "mm"};
```

```
Absolute Variation in magnification for +5mm displacement = -0.00216909x

Relative Variation in magnification for +5mm displacement = -0.0124537%
```

Hence the Gaussian magnification is MOSTLY invariant to the position of HWS

Variation in collimation (C = 0) with length changes: (assume that FL variations can be compensated for) -

```
M1 = ITMYtoHWSVar;
C1 = M1[[2]][[1]];
```

Test variation in position of F1 - determine the defocus - INSENSITIVE

Set all delta values = 0 except for dZ2A and dz1. Set these last two to the same value with opposite sign to model a lens out of position

```
C1A = Simplify[C1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → X1, dz1 → -X1}];
NominalDefocus = C1A /. X1 → 0;
Print["Nominal paraxial defocus is ", NominalDefocus, " m^-1"];
RateVariationInDefocus = (D[C1A, X1]) /. X1 → 0;
Print["Variation in defocus for + 5 mm displacement of F1 = ",
  RateVariationInDefocus * 0.005, " m^-1, or ",
  1 / (RateVariationInDefocus * 0.005), "m"];
```

```
Nominal paraxial defocus is -0.0115831 m^-1

Variation in defocus for + 5 mm displacement of F1 = 0.000103557 m^-1, or 9656.54m
```

Hence the relative change in the paraxial defocus is quite large but the absolute change (with displacement of F1) is quite small, about 86km lens for 5mm displacement. At HWS this is not an issue.

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0})];
BeamRAtHWSNom = pOutNom[[1]];
MOut = (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → X1, dz1 → -X1});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamRAtHWSX1 = pOutX1[[1]];
Print["Variation in defocus of Gaussian Beam for +5mm displacement = ",
  1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom, " m^-1", " or lens of = ",
  1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), " m"];
CTestF1OutOfPosition = {1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), "m", 5 , "mm"};
```

```
Variation in defocus of Gaussian Beam for +5mm displacement =
 0.00311344 m^-1 or lens of = 321.188 m
```

Hence the Gaussian curvature at HWS does not experience large changes in the radius of curvature that with motion of F1

Test variation in position of F2 - determine the defocus - INSENSITIVE

Set all delta values = 0 except for dZ2A and dz3. Set these last two to the same value with opposite sign to model a lens out of position

```
C1A = Simplify[C1 /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz1 → 0}];
NominalDefocus = C1A /. X1 → 0;
Print["Nominal paraxial defocus is ", NominalDefocus, " m^-1"];
RateVariationInDefocus = (D[C1A, X1]) /. X1 → 0;
Print["Variation in defocus for + 5 mm displacement of F2 = ",
  RateVariationInDefocus * 0.005, " m^-1, or ",
  1 / (RateVariationInDefocus * 0.005), "m"];
```

```
Nominal paraxial defocus is -0.0115831 m^-1
```

```
Variation in defocus for + 5 mm displacement of F2 = 0.000275168 m^-1, or 3634.14m
```

Hence the relative change in the paraxial defocus is quite large but the absolute change (with displacement of F2) is quite small, about 8km lens for 5mm displacement. At HWS this is not an issue.

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0})];
BeamRAtHWSNom = pOutNom[[1]];
MOut = (ITMYtoHWSVar /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz1 → 0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamRAtHWSX1 = pOutX1[[1]];
Print["Variation in defocus of Gaussian Beam for +5mm displacement = ",
  1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom, " m^-1", "or lens of = ",
  1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), " m"];
CTestF2OutOfPosition = {1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), "m", 5 , "mm"};
```

```
Variation in defocus of Gaussian Beam for +5mm displacement =
 0.00365103 m^-1or lens of = 273.896 m
```

Hence the Gaussian curvature at HWS does not experience large changes in the radius of curvature that with motion of F2

Test variation in F1-F2 distance: determine the defocus - INSENSITIVE

```
C1A = Simplify[C1 /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz1 → 0}];
NominalDefocus = C1A /. X1 → 0;
Print["Nominal paraxial defocus is ", NominalDefocus, " m^-1"];
RateVariationInDefocus = (D[C1A, X1]) /. X1 → 0;
Print["Variation in defocus for + 5 mm displacement of F2 = ",
  RateVariationInDefocus * 0.005, " m^-1, or ",
  1 / (RateVariationInDefocus * 0.005), "m"];
```

```
Nominal paraxial defocus is -0.0115831 m^-1
```

```
Variation in defocus for + 5 mm displacement of F2 = 0.000275168 m^-1, or 3634.14m
```

Hence the relative change in the paraxial defocus is quite large but the absolute change (with variation in F1-F2 distance) is quite small, about 8km lens for 5mm displacement. At HWS this is not an issue.

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0})];
BeamRAtHWSNom = pOutNom[[1]];
MOut = (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → -X1, dz1 → 0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamRAtHWSX1 = pOutX1[[1]];
Print["Variation in defocus of Gaussian Beam for +5mm displacement = ",
  1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom, " m^-1", "or lens of = ",
  1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), " m"];
CTestF1F2Distance = {1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), "m", 5 , "mm"};
```

```
Variation in defocus of Gaussian Beam for +5mm displacement =
 0.00365498 m^-1or lens of = 273.599 m
```

Hence the Gaussian curvature at HWS does not experience large changes in the radius of curvature that with variation in F1-F2 distance

Test variation in position of HWS - determine the defocus - INSENSITIVE

Set all delta values = 0 except for dz3. Set these last two to the same value with opposite sign to model a lens out of position

```
C1A = Simplify[C1 /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0}];
NominalDefocus = C1A /. X1 → 0;
Print["Nominal paraxial defocus is ", NominalDefocus, " m^-1"];
RateVariationInDefocus = (D[C1A, X1]) /. X1 → 0;
Print["Variation in defocus for + 5 mm displacement of F1 = ",
  RateVariationInDefocus * 0.005, " m^-1, or ",
  1 / (RateVariationInDefocus * 0.005), "m"];
```

```
Nominal paraxial defocus is -0.0115831 m^-1
```

```
Power::infy: Infinite expression 1/0. encountered. >>
```

```
Variation in defocus for + 5 mm displacement of F1 = 0. m^-1, or ComplexInfinitym
```

Hence the relative change in the paraxial defocus is ZERO

```
wIN = BeamSizeAtITM;
RIN = ITMRoC;
λIN = 840. * 10^-9;
pOutNom = OutRW[wIN, RIN, λIN,
    (ITMYtoHWSVar /. {dz3 → 0, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0})];
BeamRAtHWSNom = pOutNom[[1]];
MOut = (ITMYtoHWSVar /. {dz3 → X1, dSf2 → 0, dSf1 → 0, dZ2A → 0, dz1 → -0});
pOutX1 = OutRW[wIN, RIN, λIN, (MOut /. {X1 → 0.005})];
BeamRAtHWSX1 = pOutX1[[1]];
Print["Variation in defocus of Gaussian Beam for +5mm displacement = ",
  1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom, " m^-1", "or lens of = ",
  1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), " m"];
CTestF2HWSDistance = {1 / (1 / BeamRAtHWSX1 - 1 / BeamRAtHWSNom), "m", 5 , "mm"};
```

```
Variation in defocus of Gaussian Beam for +5mm displacement =
 -2.97585 × 10^-6 m^-1or lens of = -336039. m
```

Hence the Gaussian curvature at HWS does not experience large changes in the radius of curvature that with variation in HWS position - expected since beam is approximately collimated here

Difficulty in mode-matching (MMOut = MMIn) - Take Care with Mode Matching

```
InputModeFromHWSWR = InputModeFromHWS;
InputModeFromHWSRW = {InputModeFromHWS[[2]], InputModeFromHWS[[1]]}
OutputMode = OutRW[InputModeFromHWSRW[[2]], InputModeFromHWSRW[[1]], λIN, MHWSv]
```

```
{-40.1439, 0.00729161}
```

```
{40.1439, 0.00729161}
```

10% variation in mode size, same curvature - no worries, well inside Rayleigh range ...

```
InputModeFromHWSRW = {InputModeFromHWS[[2]], InputModeFromHWS[[1]] * 1.1}
OutputMode = OutRW[InputModeFromHWSRW[[2]], InputModeFromHWSRW[[1]], λIN, MHWSv]
```

```
{-40.1439, 0.00802077}
```

```
{40.1439, 0.00802077}
```

1.0m lens added to input beam, same mode size

```
InputModeFromHWSRW =
 {(1/1.0 + InputModeFromHWS[[2]]^-1)^-1, InputModeFromHWS[[1]]}
OutputMode = OutRW[InputModeFromHWSRW[[2]], InputModeFromHWSRW[[1]], λIN, MHWSv]
Print["Difference in output curvature = ",
 (1/OutputMode[[1]] + 1/InputModeFromHWSRW[[1]])^-1]
Print["Relative variation in output size with a 1m lens added to input = ",
 (OutputMode[[2]] - InputModeFromHWSRW[[2]])/InputModeFromHWSRW[[2]] * 100, "%"]
```

```
{1.02555, 0.00729161}
```

```
{0.974826, 0.00728495}
```

```
Difference in output curvature = 0.499772
```

```
Relative variation in output size with a 1m lens added to input = -0.0912376%
```

Temperature change required to scatter 0.01% from HWS error

Simple model first of temperature sensitivity - assuming common Length and focal length changes
requirements:
1. beam size at the reference mirror fits on a 3" mirror, maybe 4".
2. the input optics are shared between the probe and reference beams.
3. consider thermal changes in length and defocus
4. ABCD and Gaussian analysis

Look at all the telescopes with defocus and infer maximum acceptable temp. change

Input Mode is ...

```
OutModeSR2AR
```

```
{-2.29209, 0.0194012}
```

Length sensitive matrices - main telescope between SR2 and SR3

```
tempLens = {{1, 0}, {0, nFusedSilica840/n1}}.{{1, SR2RRT[[3]]}, {0, 1}}.{{1, 0}, {(n1 - nFusedSilica840)/(SR2RoC*nFusedSilica840), n1/nFusedSilica840}};
EFLSR2 = -1/tempLens[[2]][[1]];

ITMToSR2ARdX = Simplify[{{1, 0}, {0, nFusedSilica840/n1}}.{{1, SR2RRT[[3]]}, {0, 1}}.
    {{1, 0}, {(n1 - nFusedSilica840)/(SR2RoC*nFusedSilica840), n1/nFusedSilica840}}.{{1, SR2toSR3 + dx}, {0, 1}}.{{1, 0}, {-2/SR3RoC, 1}}.{{1, SR3toITMY}, {0, 1}}];
SR2ToITMARdX = Simplify[{{1, SR3toITMY}, {0, 1}}.{{1, 0}, {-2/SR3RoC, 1}}.{{1, SR2toSR3 + dx}, {0, 1}}.
    {{1, 0}, {(nFusedSilica840 - n1)/((-1*SR2RoC)*n1), nFusedSilica840/n1}}.{{1, SR2RRT[[3]]}, {0, 1}}.{{1, 0}, {0, n1/nFusedSilica840}}];
SR2ARToHWSdx = Simplify[{{1, z3 + dx3}, {0, 1}}.{{1, 0}, {-1/f2, 1}}.{{1, z2 + dx2}, {0, 1}}.
      {{1, 0}, {-1/f1, 1}}.{{1, z1 + dx1}, {0, 1}}] /. TestSolutionSubstitution;
HWSToSR2ARdx = Simplify[{{1, z1 + dx1}, {0, 1}}.{{1, 0}, {-1/f1, 1}}.{{1, z2 + dx2}, {0, 1}}.
      {{1, 0}, {-1/f2, 1}}.{{1, z3 + dx3}, {0, 1}}] /. TestSolutionSubstitution;

MatrixForm[MITM];
SR2toSR2dx = ITMToSR2ARdX.MITM.SR2ToITMARdX;
HWStoHWSdx = SR2ARToHWSdx.ITMToSR2ARdX.MITM.SR2ToITMARdX.HWSToSR2ARdx /.
   {dx1 → 0, dx2 → 0, dx3 → 0};
```

Check the matrices

```
SR2toSR2 = SR2toSR2dx /. dx → 0;
OutModeSR2AR
InModeSR2AR = {-OutModeSR2AR[[1]], OutModeSR2AR[[2]]};
pOut = OutRW[InModeSR2AR[[2]], InModeSR2AR[[1]], λIN, SR2toSR2]
```

```
{-2.29209, 0.0194012}
```

```
{-2.29209, 0.0194012}
```

```
HWStoHWS = HWStoHWSdx /. dx → 0;
InputModeFromHWS
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, HWStoHWS];
Reverse[pOut]
```

```
{0.00729161, -40.1439}
```

```
{0.00729161, 40.1439}
```

A1: Get the defocus at ITM for a given displacement between SR2 and SR3

To scatter out 0.01% the maximum acceptable defocus is MaxDefocusAtITMAll m^-1; - T1000722

```
MaxAcceptableDefocusAtITM = MaxDefocusAtITMAll;
```

```
dxTest = 0.00025;
dtM = HWStoHWSdx /. dx → dxTest;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (-InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
```

```
Input defocus = -0.0249104 m^-1
```

```
Output defocus = 0.0244375 m^-1
```

```
dSAtITM = dS / (M2^2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dxTest * 10^6], "um displacement"]
dSdxSR2toSR3 = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
```

```
Defocus at ITM = 1.55888 × 10^-6m^-1 for 250um displacement
```

```
Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

A2: Limit apparent temperature change of the concrete between SR2 & SR3

Determine how much the previous dx would need to be scaled by to yield the maximum apparent defocus allowed at the ITM

```
dxScale = MaxAcceptableDefocusAtITM / dSAtITM
```

```
1.34712
```

```
CoefficientThermalExpansionOfConcrete = 1.2 * 10^-5;
dT = dx / (L * CoefficientThermalExpansionOfConcrete) /.
    {L → SR2toSR3, dx → dxTest * dxScale};
dTSR2toSR3 = {Round[dT * 100] / 100., "K"};
Print["Maximum acceptable change in temperature = ", Round[dT * 100] * 10, " mK"]
```

```
Maximum acceptable change in temperature = 1820 mK
```

Now we need a secondary set of optics that introduce the same level of

thermal defocus and give a small beam at the reference retro-reflector.

B1: Get the defocus at ITM for a given displacement between SR2 and F1

```
HWStoHWSdx1 = SR2ARToHWSdx.ITMToSR2ARdX.MITM.SR2ToITMARdX.HWSToSR2ARdx /.
    {dx → 0, dx2 → 0, dx3 → 0};

dxTest = 0.00025;
dtM = HWStoHWSdx1 /. dx1 → dxTest;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (- InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
dSAtITM = dS / (M2 ^ 2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dxTest * 10 ^ 6], "um displacement"]
dSdx = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
```

```
Input defocus = -0.0249104 m^-1

Output defocus = 0.0242366 m^-1

Defocus at ITM = 2.22098 × 10^-6 m^-1 for 250um displacement

Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

B2: Limit apparent temperature change of the stainless between SR2 & F1

Determine how much the previous dx would need to be scaled by to yield the maximum apparent defocus allowed at the ITM

```
dxScale = MaxAcceptableDefocusAtITM / dSAtITM
```

```
0.94553
```

```
CoefficientThermalExpansionOfStainlessSteel = 1.73 * 10 ^ -5;
dT = dx / (L * CoefficientThermalExpansionOfStainlessSteel) /.
    {L → Z1A , dx → dxTest * dxScale};
dTSR2toF1 =  {Round[dT * 100] / 100., "K"};
Print["Maximum acceptable change in temperature = ", Round[dT * 100] * 10, " mK"]
```

```
Maximum acceptable change in temperature = 3720 mK
```

Now we need a secondary set of optics that introduce the same level of thermal defocus and give a small beam at the reference retro-reflector.

C1: Get the defocus at ITM for a given displacement between F1 and F2

```
HWStoHWSdx2 = SR2ARToHWSdx.ITMToSR2ARdX.MITM.SR2ToITMARdX.HWSToSR2ARdx /.
    {dx → 0, dx1 → 0, dx3 → 0};
```

```
dxTest = 0.00025;
dtM = HWStoHWSdx2 /. dx2 → dxTest;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (-InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
dSAtITM = dS / (M2^2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dxTest * 10^6], "um displacement"]
dSdx = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
```

```
Input defocus = -0.0249104 m^-1
```

```
Output defocus = 0.0245432 m^-1
```

```
Defocus at ITM = 1.21032 × 10^-6 m^-1 for 250um displacement
```

```
Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

C2: Limit apparent temperature change of the stainless between F1 & F2

Determine how much the previous dx would need to be scaled by to yield the maximum apparent defocus allowed at the ITM

```
dxScale = MaxAcceptableDefocusAtITM / dSAtITM
```

```
1.73508
```

```
CoefficientThermalExpansionOfStainlessSteel = 1.73 * 10^-5;
dT = dx / (L * CoefficientThermalExpansionOfStainlessSteel) /.
    {L → Z2A, dx → dxTest * dxScale};
dTF1toF2 = {Round[dT * 100] / 100., "K"};
Print["Maximum acceptable change in temperature = ", Round[dT * 100] * 10, " mK"]
```

```
Maximum acceptable change in temperature = 8440 mK
```

Now we need a secondary set of optics that introduce the same level of thermal defocus and give a small beam at the reference retro-reflector.

D1: Get the defocus at ITM for a given displacement between F2 and HWS

```
HWStoHWSdx3 = SR2ARToHWSdx.ITMToSR2ARdX.MITM.SR2ToITMARdX.HWSToSR2ARdx /.
    {dx → 0, dx1 → 0, dx2 → 0};
```

```
dxTest = 0.00025;
dtM = HWStoHWSdx3 /. dx3 → dxTest;
InputModeFromHWS;
Print["Input defocus = ", 1 / InputModeFromHWS[[2]], " m^-1"]
pOut = OutRW[InputModeFromHWS[[1]], InputModeFromHWS[[2]], λIN, dtM];
Print["Output defocus = ", 1 / pOut[[1]], " m^-1"]
dS = 1 / (-InputModeFromHWS[[2]]) - 1 / pOut[[1]];
M2 = BeamSizeAtITM / InputModeFromHWS[[1]];
dSAtITM = dS / (M2^2);
Print["Defocus at ITM = ", dSAtITM,
 "m^-1 for ", Round[dxTest * 10^6], "um displacement"]
dSdx = dSAtITM / dxTest;
Print["Max Acceptable defocus at ITM = ", MaxAcceptableDefocusAtITM, " m^-1"]
```

```
Input defocus = -0.0249104 m^-1
```

```
Output defocus = 0.0249101 m^-1
```

```
Defocus at ITM = 9.81063 × 10^-10 m^-1 for 250um displacement
```

```
Max Acceptable defocus at ITM = 2.1 × 10^-6 m^-1
```

D2: Limit apparent temperature change of the stainless between F2 & HWS

Determine how much the previous dx would need to be scaled by to yield the maximum apparent defocus allowed at the ITM

```
dxScale = MaxAcceptableDefocusAtITM / dSAtITM
```

```
2140.54
```

```
CoefficientThermalExpansionOfStainlessSteel = 1.73 * 10^-5;
dT = dx / (L * CoefficientThermalExpansionOfStainlessSteel) /.
    {L → Z3A, dx → dxTest * dxScale};
dTF2toHWS = {Round[dT * 100] / 100., "K"};
Print["Maximum acceptable change in temperature = ", Round[dT * 100] * 10, " mK"]
```

```
Maximum acceptable change in temperature = 13 551 470 mK
```

Now we need a secondary set of optics that introduce the same level of thermal defocus and give a small beam at the reference retro-reflector.

Report on Solution

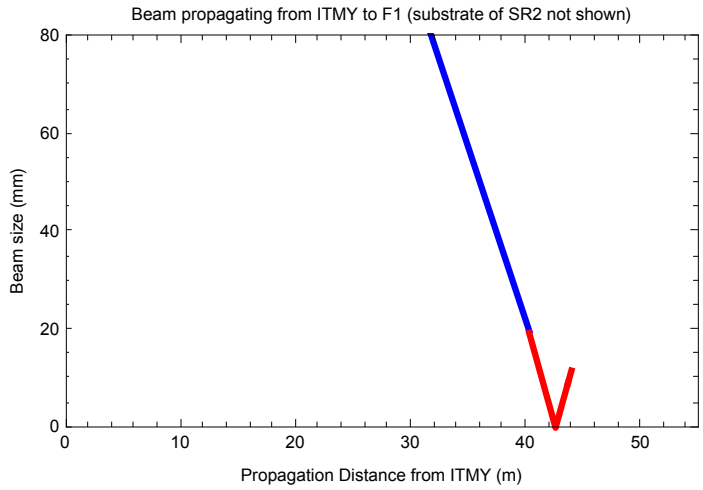
```
Print["=== REPORT ON H1:ITMY-HWS SOLUTION ==="]
Print[" "];
Print["ITMY apparent Radius of Curvature = ", ITMRoC, "m"];
Print["ITM to SR3 optical distance = ", SR3toITMY, "m"];
Print["BS to SR3 optical distance = ", SR3toBS, "m"];
Print["SR3 Radius of Curvature = ", SR3RoC, "m"];
Print["SR3 to SR2 distance = ", SR2toSR3, "m"];
Print["SR2 Thickness = ", SR2RRT[[3]], "m"];
Print["SR2 Effective Focal Length = ", EFLSR2, "m"];
Print["SR2 Radius of Curvature = ", SR2RoC, "m"];
```

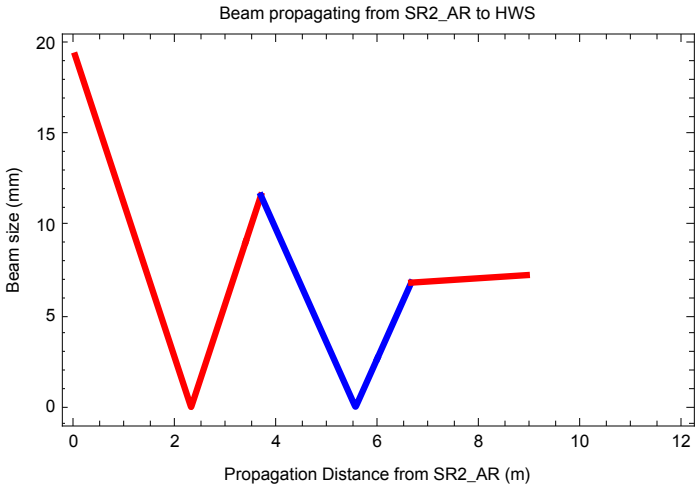
```
Print[" "];
Print["SR2_AR to F1 distance = ", Round[(Z1A) * 1000] / 1000.0, "m"];
Print["F1 focal length = ", Round[(F1A) * 1000] / 1000.0, "m"];
Print["F1 to F2 distance = ", Round[(Z2A) * 1000] / 1000.0, "m"];
Print["F2 focal length = ", Round[(F2A) * 1000] / 1000.0, "m"];
Print["F2 to HWS distance = ", Round[(Z3A) * 1000] / 1000.0, "m"];
Print[" "];
Print["Input beam size from HWS = ", InputModeFromHWS[[1]] * 1000, "mm"];
Print["Input curvature from HWS = ", InputModeFromHWS[[2]], "m"];
Print[" "];
Print["----"]
Print["TEMPERATURE SENSITIVITY"]
Print["Maximum Acceptable defocus at the ITM = ", MaxAcceptableDefocusAtITM];
Print["SR2 to SR3 - maximum acceptable temperature change = ",
 dTSR2toSR3[[1]], dTSR2toSR3[[2]]]
Print["F1 to SR2 - maximum acceptable temperature change = ",
 dTSR2toF1[[1]], dTSR2toF1[[2]]]
Print["F1 to F2 - maximum acceptable temperature change = ",
 dTF1toF2[[1]], dTF1toF2[[2]]]
Print["F2 to HWS - maximum acceptable temperature change = ",
 dTF2toHWS[[1]], dTF2toHWS[[2]]]
Print[" "];
Print["----"]
Print["TOLERANCING"]
Print["Imaging (B=0): F1 out of Position by ", BTestF1OutOfPosition[[3]],
 BTestF1OutOfPosition[[4]], " moves conjugate plane by ",
 BTestF1OutOfPosition[[1]], BTestF1OutOfPosition[[2]]]
Print["Imaging (B=0): F2 out of Position by ", BTestF2OutOfPosition[[3]],
 BTestF2OutOfPosition[[4]], " moves conjugate plane by ",
 BTestF2OutOfPosition[[1]], BTestF2OutOfPosition[[2]]]
Print["Imaging (B=0): F1-F2 distance varies by ", BTestF1F2Distance[[3]],
 BTestF1F2Distance[[4]], " moves conjugate plane by ",
 BTestF1F2Distance[[1]], BTestF1F2Distance[[2]]]
Print["Imaging (B=0): HWS out of Position by ", BTestF2HWSDistance[[3]],
 BTestF2HWSDistance[[4]], " moves conjugate plane by ",
 BTestF2HWSDistance[[1]], BTestF2HWSDistance[[2]]]
Print[" "]
Print["Magnification (A=1/17.5): F1 out of Position by ",
 ATestF1OutOfPosition[[3]], ATestF1OutOfPosition[[4]],
 " varies magnification by ",
 ATestF1OutOfPosition[[1]], ATestF1OutOfPosition[[2]]]
Print["Magnification (A=1/17.5): F2 out of Position by ",
 ATestF2OutOfPosition[[3]], ATestF2OutOfPosition[[4]],
 " varies magnification by ",
 ATestF2OutOfPosition[[1]], ATestF2OutOfPosition[[2]]]
Print["Magnification (A=1/17.5): F1-F2 distance varies by ",
 ATestF1F2Distance[[3]], ATestF1F2Distance[[4]],
 " varies magnification by ", ATestF1F2Distance[[1]], ATestF1F2Distance[[2]]]
Print["Magnification (A=1/17.5): F2-HWS distance varies by ",
 ATestF2HWSDistance[[3]], ATestF2HWSDistance[[4]],
 " varies magnification by ", ATestF2HWSDistance[[1]], ATestF2HWSDistance[[2]]]
Print[" "]
Print["Collimation (C=0): F1 out of Position by ", CTestF1OutOfPosition[[3]],
 CTestF1OutOfPosition[[4]], " looks like an additional lens at HWS with f = ",
 CTestF1OutOfPosition[[1]], CTestF1OutOfPosition[[2]]]
```

```
Print["Collimation (C=0): F2 out of Position by ", CTestF2OutOfPosition[[3]],
 CTestF2OutOfPosition[[4]], " looks like an additional lens at HWS with f = ",
 CTestF2OutOfPosition[[1]], CTestF2OutOfPosition[[2]]]
Print["Collimation (C=0): F1-F2 distance varied by ", CTestF1F2Distance[[3]],
 CTestF1F2Distance[[4]], " looks like an additional lens at HWS with f = ",
 CTestF1F2Distance[[1]], CTestF1F2Distance[[2]]]
Print["Collimation (C=0): F2-HWS distance varied by ", CTestF2HWSDistance[[3]],
 CTestF2HWSDistance[[4]], " looks like an additional lens at HWS with f = ",
 CTestF2HWSDistance[[1]], CTestF2HWSDistance[[2]]]

Show[ITMYPlottedSolutionFromITMY]
Show[ITMYPlottedSolutionFromSR2AR]
```

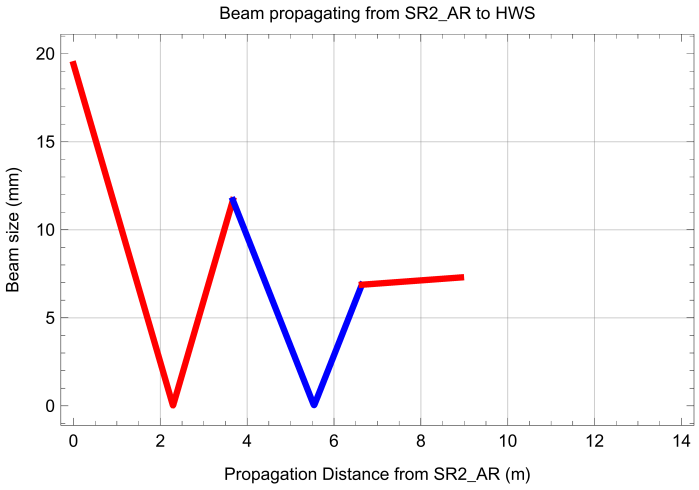
```
=== REPORT ON H1:ITMY-HWS SOLUTION ===


ITMY apparent Radius of Curvature = 1338.41m

ITM to SR3 optical distance = 24.7836m

BS to SR3 optical distance = 19.4313m

SR3 Radius of Curvature = 36m

SR3 to SR2 distance = 15.443m

SR2 Thickness = 0.075m

SR2 Effective Focal Length = 14.4494m

SR2 Radius of Curvature = 6.43m


SR2_AR to F1 distance = 3.672m

F1 focal length = 0.794m

F1 to F2 distance = 2.97m

F2 focal length = 1.134m

F2 to HWS distance = 2.283m


Input beam size from HWS = 7.29161mm

Input curvature from HWS = -40.1439m


----

TEMPERATURE SENSITIVITY

Maximum Acceptable defocus at the ITM = 2.1 × 10^-6

SR2 to SR3 - maximum acceptable temperature change = 1.82K

F1 to SR2 - maximum acceptable temperature change = 3.72K

F1 to F2 - maximum acceptable temperature change = 8.44K

F2 to HWS - maximum acceptable temperature change = 13551.5K
```

```
----

TOLERANCING

Imaging (B=0): F1 out of Position by 5mm moves conjugate plane by -1543.23mm

Imaging (B=0): F2 out of Position by 5mm moves conjugate plane by 41.1556mm

Imaging (B=0): F1-F2 distance varies by 5mm moves conjugate plane by 1557.65mm

Imaging (B=0): HWS out of Position by 5mm moves conjugate plane by -1516.5mm


Magnification (A=1/17.5): F1 out of Position by 5mm varies magnification by 0.5%

Magnification (A=1/17.5): F2 out of Position by 5mm varies magnification by -0.4%

Magnification (A=1/17.5): F1-F2 distance varies by 5mm varies magnification by -0.4%

Magnification (A=1/17.5): F2-HWS distance varies by 5mm varies magnification by 0.%


Collimation (C=0): F1 out of Position by 5mm
  looks like an additional lens at HWS with f = 321.188m

Collimation (C=0): F2 out of Position by 5mm
  looks like an additional lens at HWS with f = 273.896m

Collimation (C=0): F1-F2 distance varied by 5
 mm looks like an additional lens at HWS with f = 273.599m

Collimation (C=0): F2-HWS distance varied by 5
 mm looks like an additional lens at HWS with f = -336 039.m
```

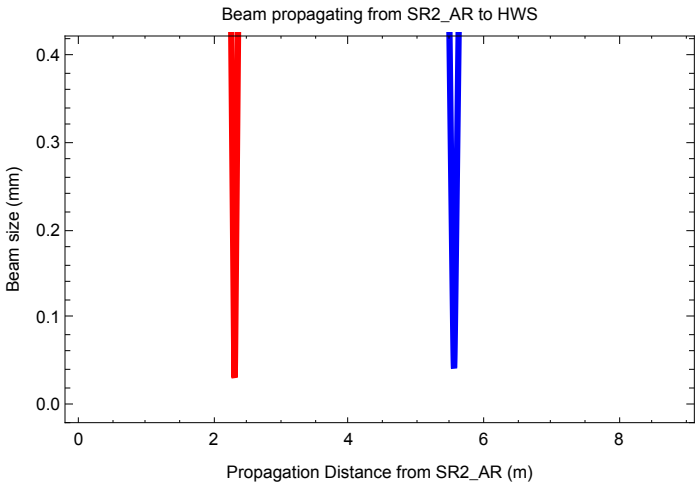





```
Show[ITMYPlottedSolutionFromSR2AR,
 PlotRange → {{0, 14}, {0, 20}}, GridLines → Automatic]
```




```
Show[ITMYPlottedSolutionFromSR2AR, PlotRange → {0, 0.4}]
```



Beam size at the mirrors

Create a Module to determine the beam size

```
GetTheBeamSize[ABCDList_, RWInputMode_, λIN_, zIn_] := Module[{},
   z0qq = 0;
   plotList = {};
   count1 = 0;
   zOut = 0.0;
   For[ii = 1, ii ≤ Length[ABCDList], ii++, {
     M1IN = ABCDList[[ii]];

     plotAll = Plot[0 * x, {x, 0, 0.00001}];
     If[M1IN[[1]][[2]] ≠ 0, {
       MOut = ⎛1  0⎞;
              ⎝0  1⎠
       For[jj = 1, jj ≤ ii - 1, jj++, {
         MOut = ABCDList[[jj]].MOut;
        }];
       MOut = ⎛1  zIn - z0qq⎞.MOut;
              ⎝0       1   ⎠

       eqOut = OutRW[RWInputMode[[2]], RWInputMode[[1]], λIN, MOut];
       If[zIn ≥  z0qq, zOut = eqOut[[2]]];



       count1 = count1 + 1;

       z0qq = ABCDList[[ii]][[1]][[2]] + z0qq;



      }];
    }];
   Return[zOut];
  ];
GetTheGouyPhase[ABCDList_, RWInputMode_, λIN_, zIn_] := Module[{x0, waistRWqq},
   z0qq = 0;
   plotList = {};
   count1 = 0;
   zOut = 0.0;
   For[ii = 1, ii ≤ Length[ABCDList], ii++, {
     M1IN = ABCDList[[ii]];

     plotAll = Plot[0 * x, {x, 0, 0.00001}];
     If[M1IN[[1]][[2]] ≠ 0, {
       MOut = ⎛1  0⎞;
              ⎝0  1⎠
       For[jj = 1, jj ≤ ii - 1, jj++, {
         MOut = ABCDList[[jj]].MOut;
```

```
      }];
      MOut = ( 1  zIn - z0qq
               0      1     ).MOut;

      eqOut = OutRW[RWInputMode[[2]], RWInputMode[[1]], λIN, MOut];
      If[zIn ≥ z0qq, zOut = eqOut];



      count1 = count1 + 1;

      z0qq = ABCDList[[ii]][[1]][[2]] + z0qq;



    }];
   }];

  x0 = FindNearestWaist[zOut[[2]], zOut[[1]], λIN];
  waistRWqq = OutRW[zOut[[2]], zOut[[1]], λIN, ( 1  x0
                                                 0  1 )];
  GouyPhaseqq = ArcTan[ -x0
                        (Pi waistRWqq[[2]]^2
                         λIN                )];
  Return[GouyPhaseqq];
 ];
```

Import the data on the mirror positions

```
dataIN =
  Import["/Users/aidanbrooks/Documents/LIGO/documents/Advanced_LIGO/TCS/Hartmann
     sensor/modeling/H1HWSY_coords.txt", "TSV"];
totaldistance = 0;
distList = {};
nameList = {};
For[ii = 1, ii ≤ Length[dataIN], ii++, {
   d1 = dataIN[[ii]][[7]] / 1000.0;
   totaldistance = totaldistance + d1;
   dataIN[[ii]] = Append[dataIN[[ii]], totaldistance];
   distList = Append[distList , totaldistance];
   nameList = Append[nameList, dataIN[[ii]][[1]]];
  }];
```

```
Import::nffil: File not found during Import. >>
```

```
dataIN
```

```
$Failed
```

```
distList
```

```
{}
```

distances to the mirrors from handoff at SR2_AR

```
distances = {0.0, 893.8, 2096.1, 2516.4, 3970.1, 5793.6, 6655.1,
    7055.1, 9725.6, 10411.7, 10836.7, 10886.7, 11161.7, 11997.4, 12697.8,
    13322.8, 13372.8, 13472.8, 14054.9, 14254.9, 14330.0, 14730.0} / 1000;
distList = distances;
```

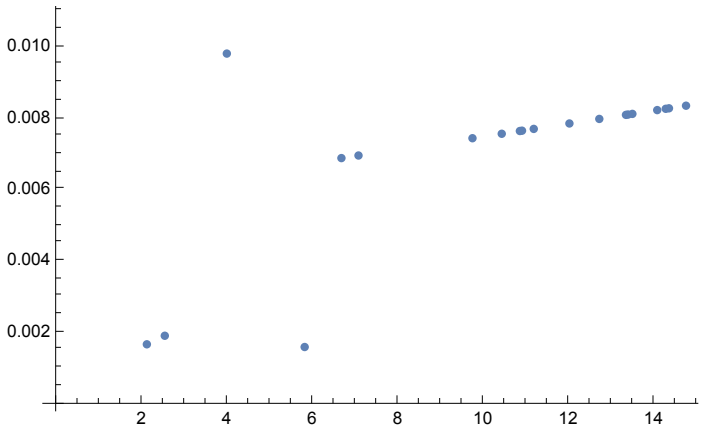
Determine the beam size at each interface

```
beamSizeList = {};
For[kk = 1, kk ≤ Length[distList], kk++, {
   beamsize =
    GetTheBeamSize[{{{1, z1}, {0, 1}}, {{1, 0}, {-1/f1, 1}}, {{1, z2}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /.
      TestSolutionSubstitution, OutModeSR2AR, 840 * 10^-9, distList[[kk]]];

   beamSizeList = Append[beamSizeList, {distList[[kk]], beamsize}];
  }];
```

```
beamSizeList
```

```
{{0., 0.0194012}, {0.8938, 0.0118357}, {2.0961, 0.00165922}, {2.5164, 0.00189894},
 {3.9701, 0.00981034}, {5.7936, 0.00158345}, {6.6551, 0.00688006},
 {7.0551, 0.00695255}, {9.7256, 0.00743734}, {10.4117, 0.0075621},
 {10.8367, 0.00763942}, {10.8867, 0.00764852}, {11.1617, 0.00769856},
 {11.9974, 0.00785073}, {12.6978, 0.00797834}, {13.3228, 0.00809227},
 {13.3728, 0.00810139}, {13.4728, 0.00811962}, {14.0549, 0.00822579},
 {14.2549, 0.00826228}, {14.33, 0.00827598}, {14.73, 0.00834898}}
```

```
ListPlot[beamSizeList]
```




```
Export["/Users/aidanbrooks/Documents/LIGO/documents/Advanced_LIGO/TCS/Hartmann
    sensor/design/optical layout and
    modeling/H1HWSY_coords_beamsize.txt", beamSizeList, "Table"];
```

```
For[ii = 1, ii ≤ Length[beamSizeList], ii++, Print[1000 * beamSizeList[[ii]][[2]]]]
```

```
19.4012

11.8357

1.65922

1.89894

9.81034

1.58345

6.88006

6.95255

7.43734

7.5621

7.63942

7.64852

7.69856

7.85073

7.97834

8.09227

8.10139

8.11962

8.22579

8.26228

8.27598

8.34898
```

Determine the beam size at each interface

```
GouyPhase =
 GetTheGouyPhase[{({{1, z1}, {0, 1}}), ({{1, 0}, {-1/f1, 1}}), ({{1, z2}, {0, 1}}), ({{1, 0}, {-1/f2, 1}}), ({{1, z3}, {0, 1}})} /.
   TestSolutionSubstitution, OutModeSR2AR, 840 * 10^-9, distList[[1]]]
```

```
Solve::ratnz: Solve was unable to solve the system with inexact coefficients
    The answer was obtained by solving a corresponding exact system and numericizing the result. >>

Power::infy: Infinite expression 1/0. encountered. >>

-1.56917
```

```
beamSizeList = {};
GouyPhaseList = {};
GouyPhaseListDeg = {};
For[kk = 1, kk ≤ Length[distList], kk++, {
   beamsize =
    GetTheBeamSize[{{{1, z1}, {0, 1}}, {{1, 0}, {-1/f1, 1}}, {{1, z2}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /.
      TestSolutionSubstitution, OutModeSR2AR, 840 * 10^-9, distList[[kk]]];

   beamSizeList = Append[beamSizeList, {distList[[kk]], beamsize}];

   GouyPhase =
    GetTheGouyPhase[{{{1, z1}, {0, 1}}, {{1, 0}, {-1/f1, 1}}, {{1, z2}, {0, 1}}, {{1, 0}, {-1/f2, 1}}, {{1, z3}, {0, 1}}} /.
      TestSolutionSubstitution, OutModeSR2AR, 840 * 10^-9, distList[[kk]]];

   GouyPhaseList = Append[GouyPhaseList, {distList[[kk]], GouyPhase}];
   GouyPhaseListDeg =
    Append[GouyPhaseListDeg, {distList[[kk]], GouyPhase * 180 / Pi}];

  }];
```

Solve::ratnz: Solve was unable to solve the system with inexact coefficients
The answer was obtained by solving a corresponding exact system and numericizing the result >>

Power::infy: Infinite expression $\frac{1}{0.}$ encountered >>

Solve::ratnz: Solve was unable to solve the system with inexact coefficients
The answer was obtained by solving a corresponding exact system and numericizing the result >>

Power::infy: Infinite expression $\frac{1}{0.}$ encountered >>

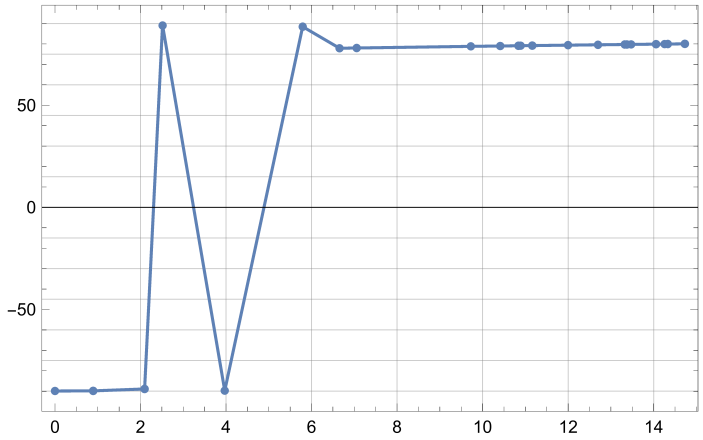
Solve::ratnz: Solve was unable to solve the system with inexact coefficients
The answer was obtained by solving a corresponding exact system and numericizing the result >>

General::stop: Further output of Solve::ratnz will be suppressed during this calculation >>

Power::infy: Infinite expression $\frac{1}{0.}$ encountered >>

General::stop: Further output of Power::infy will be suppressed during this calculation >>

```
plot1 = ListPlot[GouyPhaseListDeg, Frame → True,
   GridLines → {Automatic, Table[ii, {ii, -90, 90, 15}]}];
plot2 = ListPlot[GouyPhaseListDeg, Frame → True, Joined → True];
Show[plot1, plot2]
```




```
GouyPhaseOptics = {};
For[jj = 1, jj ≤ Length[GouyPhaseListDeg], jj++, {
   GouyPhaseOptics =
     Append[GouyPhaseOptics, {nameList[[jj]], GouyPhaseListDeg[[jj]][[2]]}];
  }];
MatrixForm[GouyPhaseOptics]
```

Part::partw: Part 1 of {} does not exist >>

Part::partw: Part 2 of {} does not exist >>

Part::partw: Part 3 of {} does not exist >>

General::stop: Further output of Part::partw will be suppressed during this calculation >>

$$\begin{pmatrix} \{\}[[1]] & -89.9067 \\ \{\}[[2]] & -89.8471 \\ \{\}[[3]] & -88.9091 \\ \{\}[[4]] & 89.0468 \\ \{\}[[5]] & -89.7501 \\ \{\}[[6]] & 88.4513 \\ \{\}[[7]] & 77.8935 \\ \{\}[[8]] & 78.0216 \\ \{\}[[9]] & 78.8129 \\ \{\}[[10]] & 78.9997 \\ \{\}[[11]] & 79.1125 \\ \{\}[[12]] & 79.1256 \\ \{\}[[13]] & 79.1971 \\ \{\}[[14]] & 79.4089 \\ \{\}[[15]] & 79.5802 \\ \{\}[[16]] & 79.7285 \\ \{\}[[17]] & 79.7402 \\ \{\}[[18]] & 79.7635 \\ \{\}[[19]] & 79.897 \\ \{\}[[20]] & 79.9421 \\ \{\}[[21]] & 79.9589 \\ \{\}[[22]] & 80.0476 \end{pmatrix}$$

```
For[kk = 1, kk ≤ Length[distList], kk++, {
   dataIN[[kk]] = Append[dataIN[[kk]], beamSizeList[[kk]][[2]] * 1000.0];
  }];
```

Part::partd: Part specification $Failed[[1]] is longer than depth of object. >>

Part::pkspec1: The expression 19.401205218640996` cannot be used as a part specification. >>

Set::partd: Part specification dataIN[kk] is longer than depth of object. >>

Part::partd: Part specification $Failed[[2]] is longer than depth of object. >>

Part::pkspec1: The expression 11.835727807266018` cannot be used as a part specification. >>

Set::partd: Part specification dataIN[kk] is longer than depth of object. >>

Part::partd: Part specification $Failed[[3]] is longer than depth of object. >>

General::stop: Further output of Part::partd will be suppressed during this calculation. >>

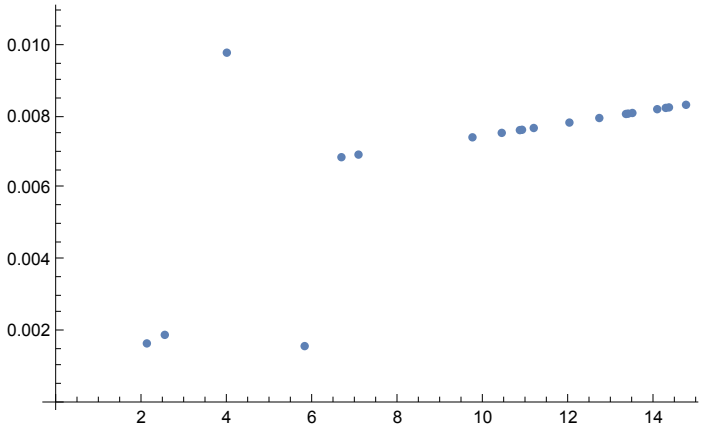
Part::pkspec1: The expression 1.6592205974712422` cannot be used as a part specification. >>

General::stop: Further output of Part::pkspec1 will be suppressed during this calculation. >>

Set::partd: Part specification dataIN[kk] is longer than depth of object. >>

General::stop: Further output of Set::partd will be suppressed during this calculation. >>

```
ListPlot[beamSizeList]
```




```
Export["/Users/aidanbrooks/Documents/LIGO/documents/Advanced_LIGO/TCS/Hartmann
    sensor/modeling/H1HWSY_coords_beamsize.txt", dataIN, "Table"];
```

```
MatrixForm[dataIN]
```

$Failed

Mis - alignment

Alignment

Create an ABCD plotting function

```
PlotABCDMatrixRayTrace[inputBeam_, ABCDList_] := Module[{},
   totalDist = 0;
   currentABCDMatrix = {{1, 0}, {0, 1}};
   PlotCount = 0;

   For[ii = 1, ii ≤ Length[ABCDList], ii++, {
     If[ABCDList[[ii]][[1]][[2]] == 0,
       {currentABCDMatrix = ABCDList[[ii]].currentABCDMatrix};
       , {
        oldABCDMatrix = currentABCDMatrix;
        currentABCDMatrix = {{1, zQQ - totalDist}, {0, 1}}.currentABCDMatrix;

        dzmax = ABCDList[[ii]][[1]][[2]];

        zmin = totalDist;
        zmax = totalDist + dzmax;
        beamOut = currentABCDMatrix.inputBeam;

        plotA = Plot[beamOut[[1]], {zQQ, zmin, zmax},
          PlotRange → {{0, zmax}, {-1, 1}}, Frame → True, PlotStyle → Thick];


        If[PlotCount == 0, {
          plotAll = Show[plotA];
         }, {
          plotAll = Show[plotAll, plotA, PlotRange → {{0, zmax}, Automatic}];
         }];


        PlotCount = PlotCount + 1;
        currentABCDMatrix = ABCDList[[ii]].oldABCDMatrix;
        totalDist = totalDist + dzmax;
       }];

    }];
   Return[plotAll];
  ];
```

Basic Misalignment on Reflection from ITM

Basic Misalignment on Transmission to ITM

Full Transmission

Determine Range and Resolution of the Alignment depending on Actuator Location

Position 1 - Near the Viewport

Nominal Values for the Actuators: CASE 1 (Position 1)

Case 1 : (HWSY STEER M8 and HWSY STEER M10)
dM1dM2 = 0.325
dL1 = 0.83564

```
Case1 = {dM1M2 → 0.325, dL1 → 0.83564}
```

```
{dM1M2 → 0.325, dL1 → 0.83564}
```

```
MOutCase = MOut /. Case1;
InvMout = Inverse[MOutCase]
```

```
{{0.808871, -4.41753}, {0.0732278, 3.40405}}
```

Work Out Resolution

```
Resolution of angles & displacements on HWS
deltaAlphaMin = 0.7 urad
resolution at HWS = Sqrt[C2A^2 + C1A^2]*0.7 urad
```

```
C2A = DdAlpha2 /. Case1;
C1A = DdAlpha1 /. Case1;
resol = 0.7 * 10^-6;
ResolutionAtHWS = Sqrt[C1A^2 + C2A^2] * (resol)
```

$$\left\{1.26875 \times 10^{-6}, 1.84771 \times 10^{-7}\right\}$$

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.177941, 0.0200677}
```

Work Out Ranges

```
Range of angles & displacements on HWS
deltaAlphaMax = 70 umrad
Range at HWS = (Abs[C2A] + Abs[C1A])*70 mrad
```

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.177941, 0.0200677}
```

Angular Range at HWS keeping displacement fixed
dHWS = 0;
Solve dx == 0 and substitute in solution for angles

```
eqSolve1 = deltaxHWS == MOutCase[[1]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaxHWS → 0;
angleOut1 = angleOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[angleOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaxHWS → 0;
angleOut2 = angleOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[angleOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[angleOut2[[2]]] > Abs[angleOut1[[2]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut1[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut1[[1]]];
  }, {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut2[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 77.0578% on Alpha 1

Maximum range on Alpha 1 requires 129.773% on Alpha 2


Maximum range of Angle 1 saturates Angle 2

Angular range is limited by Actuator 2

Maximum range (keeping displacement on HWS constant) = -0.015846 radians
```

Displacement Range at HWS keeping angle fixed
Solve dtheta == 0 and substitute in solution for angles

```
eqSolve1 = deltaThetaHWS == MOutCase[[2]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaThetaHWS → 0;
dxOut1 = dxOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[dxOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaThetaHWS → 0;
dxOut2 = dxOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[dxOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[dxOut2[[2]]] > Abs[dxOut1[[1]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut1[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut2[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 9.05308% on Alpha 1

Maximum range on Alpha 1 requires 1104.6% on Alpha 2


Maximum range of Angle 1 saturates Angle 2

Angular range is limited by Actuator 2

Maximum range (keeping angle on HWS constant) = 86.5404 mm
```

```
MaxRangeCase1 = {MaxRangeDisplacement, MaxRangeAngle}
Case1Result = {ResolutionAtHWS, MaxRangeCase1}
```

```
{0.0865404, 0.015846}

{{1.26875 × 10^-6, 1.84771 × 10^-7}, {0.0865404, 0.015846}}
```

Nominal Values for the Actuators: CASE 2 (Position 1)

Case 2 : (HWSY STEER M9 and HWSY STEER M10)
dM1dM2 = 0.05
dL1 = 0.83564 + 0.275

```
Case2 = {dM1M2 → 0.325 - 0.275, dL1 → 0.83564 + 0.275}
```

```
{dM1M2 → 0.05, dL1 → 1.11064}
```

```
MOutCase = MOut /. Case2
InvMout = Inverse[MOutCase]
```

```
{{1.10632, 1.15699}, {-0.023799, 0.0203059}}
```

```
{{0.406118, -23.1398}, {0.47598, 22.1263}}
```

Work Out Resolution

```
Resolution of angles & displacements on HWS
deltaAlphaMin = 0.7 urad
resolution at HWS = Sqrt[C2A^2 + C1A^2]*0.7 urad
```

```
C2A = DdAlpha2 /. Case2;
C1A = DdAlpha1 /. Case2;
resol = 0.7 * 10^-6;
ResolutionAtHWS = Sqrt[C1A^2 + C2A^2] * (resol)
```

$$\left\{1.12056 \times 10^{-6}, 2.18992 \times 10^{-8}\right\}$$

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.158431, 0.00308735}
```

Work Out Ranges

```
Range of angles & displacements on HWS
deltaAlphaMax = 70 umrad
Range at HWS = (Abs[C2A] + Abs[C1A])*70 mrad
```

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.158431, 0.00308735}
```

```
Angular Range at HWS keeping displacement fixed
dHWS = 0;
Solve dx == 0 and substitute in solution for angles
```

```
eqSolve1 = deltaxHWS == MOutCase[[1]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaxHWS → 0;
angleOut1 = angleOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[angleOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaxHWS → 0;
angleOut2 = angleOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[angleOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[angleOut2[[2]]] > Abs[angleOut1[[2]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut1[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut2[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 95.6202% on Alpha 1

Maximum range on Alpha 1 requires 104.58% on Alpha 2


Maximum range of Angle 1 saturates Angle 2

Angular range is limited by Actuator 2

Maximum range (keeping displacement on HWS constant) = -0.00302509 radians
```

Displacement Range at HWS keeping angle fixed
Solve dtheta == 0 and substitute in solution for angles

```
eqSolve1 = deltaThetaHWS == MOutCase[[2]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaThetaHWS → 0;
dxOut1 = dxOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[dxOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaThetaHWS → 0;
dxOut2 = dxOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[dxOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[dxOut2[[2]]] > Abs[dxOut1[[1]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut1[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut2[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 117.202% on Alpha 1

Maximum range on Alpha 1 requires 85.3225% on Alpha 2


Maximum range of Angle 2 saturates Angle 1

Angular range is limited by Actuator 1

Maximum range (keeping angle on HWS constant) = 147.065 mm
```

```
MaxRangeCase2 = {MaxRangeDisplacement, MaxRangeAngle}
Case2Result = {ResolutionAtHWS, MaxRangeCase2}
```

```
{0.147065, 0.00302509}

{{1.12056 × 10^-6, 2.18992 × 10^-8}, {0.147065, 0.00302509}}
```

Position 2 - Near the Viewport

Nominal Values for the Actuators: CASE 3 (Position 2)

Case 3 : (HWSY STEER M11 and HWSY STEER M12)
dM1dM2 = 0.625
dL1 = 0.7

```
Case3 = {dM1M2 → 0.625, dL1 → 0.7}
```

```
{dM1M2 → 0.625, dL1 → 0.7}
```

```
MOutCase = MOut /. Case3;
InvMout = Inverse[MOutCase]
```

```
{{1.6, -1.53216}, {-1.6, 2.53216}}
```

Work Out Resolution

```
Resolution of angles & displacements on HWS
deltaAlphaMin = 0.7 urad
resolution at HWS = Sqrt[C2A^2 + C1A^2]*0.7 urad
```

```
C2A = DdAlpha2 /. Case3;
C1A = DdAlpha1 /. Case3;
resol = 0.7 * 10^-6;
ResolutionAtHWS = Sqrt[C1A^2 + C2A^2] * (resol)
```

$$\left\{1.29483 \times 10^{-6},\ 9.89949 \times 10^{-7}\right\}$$

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.177814, 0.14}
```

Work Out Ranges

```
Range of angles & displacements on HWS
deltaAlphaMax = 70 umrad
Range at HWS = (Abs[C2A] + Abs[C1A])*70 mrad
```

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.177814, 0.14}
```

```
Angular Range at HWS keeping displacement fixed
dHWS = 0;
Solve dx == 0 and substitute in solution for angles
```

```
eqSolve1 = deltaxHWS == MOutCase[[1]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaxHWS → 0;
angleOut1 = angleOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[angleOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaxHWS → 0;
angleOut2 = angleOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[angleOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[angleOut2[[2]]] > Abs[angleOut1[[2]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut1[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut2[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 165.267% on Alpha 1

Maximum range on Alpha 1 requires 60.508% on Alpha 2


Maximum range of Angle 2 saturates Angle 1

Angular range is limited by Actuator 1

Maximum range (keeping displacement on HWS constant) = 0.0276444 radians
```

Displacement Range at HWS keeping angle fixed
Solve dtheta == 0 and substitute in solution for angles

```
eqSolve1 = deltaThetaHWS == MOutCase[[2]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaThetaHWS → 0;
dxOut1 = dxOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[dxOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaThetaHWS → 0;
dxOut2 = dxOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[dxOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[dxOut2[[2]]] > Abs[dxOut1[[1]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut1[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut2[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 100.% on Alpha 1
```

```
Maximum range on Alpha 1 requires 100.% on Alpha 2
```

```
Maximum range of Angle 1 saturates Angle 2
```

```
Angular range is limited by Actuator 2
```

```
Maximum range (keeping angle on HWS constant) = 43.75 mm
```

```
MaxRangeCase3 = {MaxRangeDisplacement, MaxRangeAngle}
Case3Result = {ResolutionAtHWS, MaxRangeCase3}
```

```
{0.04375, 0.0276444}
```

$$\left\{\left\{1.29483 \times 10^{-6}, 9.89949 \times 10^{-7}\right\}, \{0.04375, 0.0276444\}\right\}$$

Nominal Values for the Actuators: CASE 4 (Position 2)

Case 3 : (HWSY STEER M12 and HWSY STEER M13)
dM1dM2 = 0.625
dL1 = 0.7+0625

```
Case4 = {dM1M2 → 0.05, dL1 → 0.7 + 0.625}
```

```
{dM1M2 → 0.05, dL1 → 1.325}
```

```
MOutCase = MOut /. Case4;
InvMout = Inverse[MOutCase]
```

```
{{20., -18.152}, {-20., 19.152}}
```

Work Out Resolution

```
Resolution of angles & displacements on HWS
deltaAlphaMin = 0.7 urad
resolution at HWS = Sqrt[C2A^2 + C1A^2]*0.7 urad
```

```
C2A = DdAlpha2 /. Case4;
C1A = DdAlpha1 /. Case4;
resol = 0.7 * 10^-6;
ResolutionAtHWS = Sqrt[C1A^2 + C2A^2] * (resol);
```

```
ResolutionCase4 = ResolutionAtHWS;
```

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.130564, 0.14}
```

Work Out Ranges

```
Range of angles & displacements on HWS
deltaAlphaMax = 70 umrad
Range at HWS = (Abs[C2A] + Abs[C1A])*70 mrad
```

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.130564, 0.14}
```

```
Angular Range at HWS keeping displacement fixed
dHWS = 0;
Solve dx == 0 and substitute in solution for angles
```

```
eqSolve1 = deltaxHWS == MOutCase[[1]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaxHWS → 0;
angleOut1 = angleOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[angleOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaxHWS → 0;
angleOut2 = angleOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[angleOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[angleOut2[[2]]] > Abs[angleOut1[[2]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut1[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut2[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 105.509% on Alpha 1

Maximum range on Alpha 1 requires 94.7786% on Alpha 2


Maximum range of Angle 2 saturates Angle 1

Angular range is limited by Actuator 1

Maximum range (keeping displacement on HWS constant) = 0.00365497 radians
```

Displacement Range at HWS keeping angle fixed
Solve dtheta == 0 and substitute in solution for angles

```
eqSolve1 = deltaThetaHWS == MOutCase[[2]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaThetaHWS → 0;
dxOut1 = dxOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[dxOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaThetaHWS → 0;
dxOut2 = dxOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[dxOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[dxOut2[[2]]] > Abs[dxOut1[[1]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut1[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut2[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 100.% on Alpha 1
```

```
Maximum range on Alpha 1 requires 100.% on Alpha 2
```

```
Maximum range of Angle 1 saturates Angle 2
```

```
Angular range is limited by Actuator 2
```

```
Maximum range (keeping angle on HWS constant) = 3.5 mm
```

```
MaxRangeCase4 = {MaxRangeDisplacement, MaxRangeAngle}
Case4Result = {ResolutionAtHWS, MaxRangeCase4}
```

```
{0.0035, 0.00365497}
```

$$\left\{\left\{9.23559 \times 10^{-7}, 9.89949 \times 10^{-7}\right\}, \{0.0035, 0.00365497\}\right\}$$

Nominal Values for the Actuators: CASE 5 (Position 2)

Case 5 : (HWSY STEER M11 and HWSY STEER M12)
dM1dM2 = 1.357
dL1 = 0.7

```
Case5 = {dM1M2 → 0.625 + 0.05 + 0.1 + 0.582, dL1 → 0.7}
```

```
{dM1M2 → 1.357, dL1 → 0.7}
```

```
MOutCase = MOut /. Case5;
InvMout = Inverse[MOutCase]
Det[MOutCase]
```

```
{{0.73692, -0.166249}, {-0.73692, 1.16625}}
```

```
1.357
```

Work Out Resolution

```
Resolution of angles & displacements on HWS
deltaAlphaMin = 0.7 urad
resolution at HWS = Sqrt[C2A^2 + C1A^2]*0.7 urad
```

```
C2A = DdAlpha2 /. Case5
C1A = DdAlpha1 /. Case5
resol = 0.7 * 10^-6;
ResolutionAtHWS = Sqrt[C1A^2 + C2A^2] * (resol)
```

```
{1.5826, 1}
```

```
{0.2256, 1}
```

$$\left\{1.11902 \times 10^{-6},\ 9.89949 \times 10^{-7}\right\}$$

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.126574, 0.14}
```

Work Out Ranges

```
Range of angles & displacements on HWS
deltaAlphaMax = 70 umrad
Range at HWS = (Abs[C2A] + Abs[C1A])*70 mrad
```

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.126574, 0.14}
```

```
Angular Range at HWS keeping displacement fixed
dHWS = 0;
Solve dx == 0 and substitute in solution for angles
```

```
eqSolve1 = deltaxHWS == MOutCase[[1]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaxHWS → 0;
angleOut1 = angleOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[angleOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaxHWS → 0;
angleOut2 = angleOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[angleOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[angleOut2[[2]]] > Abs[angleOut1[[2]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut1[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut2[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 701.507% on Alpha 1

Maximum range on Alpha 1 requires 14.255% on Alpha 2


Maximum range of Angle 2 saturates Angle 1

Angular range is limited by Actuator 1

Maximum range (keeping displacement on HWS constant) = 0.0600215 radians
```

Displacement Range at HWS keeping angle fixed
Solve dtheta == 0 and substitute in solution for angles

```
eqSolve1 = deltaThetaHWS == MOutCase[[2]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaThetaHWS → 0;
dxOut1 = dxOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[dxOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaThetaHWS → 0;
dxOut2 = dxOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[dxOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[dxOut2[[2]]] > Abs[dxOut1[[1]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut1[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut2[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 100.% on Alpha 1

Maximum range on Alpha 1 requires 100.% on Alpha 2


Maximum range of Angle 2 saturates Angle 1

Angular range is limited by Actuator 1

Maximum range (keeping angle on HWS constant) = -94.99 mm
```

```
MaxRangeCase5 = {MaxRangeDisplacement, MaxRangeAngle}
Case5Result = {ResolutionAtHWS, MaxRangeCase5}
```

```
{0.09499, 0.0600215}

{{1.11902 × 10^-6, 9.89949 × 10^-7}, {0.09499, 0.0600215}}
```

Position 3 - Around F1

Nominal Values for the Actuators: CASE 6 (Position 3)

Case 6: (HWSY STEER M11 and HWSY STEER M12)
dL1 = 0.83564
dL2 = 0.700

```
Case6 = {dL1 → 0.835, dL2 → 0.7}
```

```
{dL1 → 0.835, dL2 → 0.7}
```

```
MOutCase = MOut /. Case6;
InvMout = Inverse[MOutCase]
```

```
{{0.980956, -1.55246}, {-0.258431, 1.40899}}
```

Work Out Resolution

```
Resolution of angles & displacements on HWS
deltaAlphaMin = 0.7 urad
resolution at HWS = Sqrt[C2A^2 + C1A^2]*0.7 urad
```

```
C2A = DdAlpha2 /. Case6;
C1A = DdAlpha1 /. Case6;
resol = 0.7 * 10^-6;
ResolutionAtHWS = Sqrt[C1A^2 + C2A^2] * (resol)
```

$$\left\{1.49605 \times 10^{-6},\ 7.23884 \times 10^{-7}\right\}$$

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.211326, 0.0884413}
```

Work Out Ranges

```
Range of angles & displacements on HWS
deltaAlphaMax = 70 umrad
Range at HWS = (Abs[C2A] + Abs[C1A])*70 mrad
```

```
mrad = 0.001;
RangeAct = 70 mrad;
RangeAtHWS = (Abs[C2A] + Abs[C1A]) * RangeAct
```

```
{0.211326, 0.0884413}
```

```
Angular Range at HWS keeping displacement fixed
dHWS = 0;
Solve dx == 0 and substitute in solution for angles
```

```
eqSolve1 = deltaxHWS == MOutCase[[1]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaxHWS → 0;
angleOut1 = angleOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[angleOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
angleOut =
  ({MOutCase[[2]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaxHWS → 0;
angleOut2 = angleOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[angleOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[angleOut2[[2]]] > Abs[angleOut1[[2]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut1[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping displacement on HWS constant) = " <>
     ToString[angleOut2[[1]]] <> " radians"];
   MaxRangeAngle = Abs[angleOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 90.7586% on Alpha 1

Maximum range on Alpha 1 requires 110.182% on Alpha 2


Maximum range of Angle 1 saturates Angle 2

Angular range is limited by Actuator 2

Maximum range (keeping displacement on HWS constant) = -0.0450897 radians
```

Displacement Range at HWS keeping angle fixed
Solve dtheta == 0 and substitute in solution for angles

```
eqSolve1 = deltaThetaHWS == MOutCase[[2]].{dAlpha2, dAlpha1};
soln1 = Solve[eqSolve1, dAlpha1][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha1} /. soln1) /. deltaThetaHWS → 0;
dxOut1 = dxOut /. dAlpha2 → RangeAct;
Print["Maximum range on Alpha 2 requires " <>
   ToString[100 * Abs[dxOut1[[2]]] / RangeAct] <> "% on Alpha 1"];

soln2 = Solve[eqSolve1, dAlpha2][[1]];
dxOut =
  ({MOutCase[[1]].{dAlpha2, dAlpha1}, dAlpha2} /. soln2) /. deltaThetaHWS → 0;
dxOut2 = dxOut /. dAlpha1 → RangeAct;
Print["Maximum range on Alpha 1 requires " <>
   ToString[100 * Abs[dxOut2[[2]]] / RangeAct] <> "% on Alpha 2"];
Print[" "];

If[Abs[dxOut2[[2]]] > Abs[dxOut1[[1]]], {
   Print["Maximum range of Angle 1 saturates Angle 2"];
   Print["Angular range is limited by Actuator 2"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut1[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut1[[1]]];
  } , {
   Print["Maximum range of Angle 2 saturates Angle 1"];
   Print["Angular range is limited by Actuator 1"];
   Print["Maximum range (keeping angle on HWS constant) = " <>
     ToString[dxOut2[[1]] * 1000] <> " mm"];
   MaxRangeDisplacement = Abs[dxOut2[[1]]];
  }
 ];
```

```
Maximum range on Alpha 2 requires 26.3448% on Alpha 1

Maximum range on Alpha 1 requires 379.582% on Alpha 2


Maximum range of Angle 1 saturates Angle 2

Angular range is limited by Actuator 2

Maximum range (keeping angle on HWS constant) = 71.3589 mm
```

```
MaxRangeCase6 = {MaxRangeDisplacement, MaxRangeAngle}
Case6Result = {ResolutionAtHWS, MaxRangeCase3}
```

```
{0.0713589, 0.0450897}

{{1.49605 × 10^-6, 7.23884 × 10^-7}, {0.04375, 0.0276444}}
```

Results

```
Case1Result * 1000
Case2Result * 1000
Case3Result * 1000
Case4Result * 1000
Case5Result * 1000
Case6Result * 1000
```

```
{{0.00126875, 0.000184771}, {86.5404, 15.846}}
```

```
{{0.00112056, 0.0000218992}, {147.065, 3.02509}}
```

```
{{0.00129483, 0.000989949}, {43.75, 27.6444}}
```

```
{{0.000923559, 0.000989949}, {3.5, 3.65497}}
```

```
{{0.00111902, 0.000989949}, {94.99, 60.0215}}
```

```
{{0.00149605, 0.000723884}, {43.75, 27.6444}}
```

```
4.5 / 17.5 * 180 / Pi mrad
```

```
0.0147332
```

Final results

H1X

```
MatrixForm[H1XSoln]
```

```
( 2.70621  2.80774  2.15622  1.6994  -1.6994  -17.5 AirLenstoHWS, VacLenstoAirLens,  PC
```

```
Print["POtoVacLens = ", H1XSoln[[1]][[3]]]
Print["VacLens = ", H1XSoln[[1]][[5]]]
Print["VacLenstoAirLens = ", H1XSoln[[1]][[2]]]
Print["AirLens = ", H1XSoln[[1]][[4]]]
Print["AirLenstoHWS = ", H1XSoln[[1]][[1]]]
```

```
POtoVacLens = 2.15622
```

```
VacLens = -1.6994
```

```
VacLenstoAirLens = 2.80774
```

```
AirLens = 1.6994
```

```
AirLenstoHWS = 2.70621
```

$$\begin{pmatrix} 1 & 2.706 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1/1.6994 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2.808 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 1/1.6994 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2.156 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 15.247 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1/18 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 24.666 \\ 0 & 1 \end{pmatrix}$$

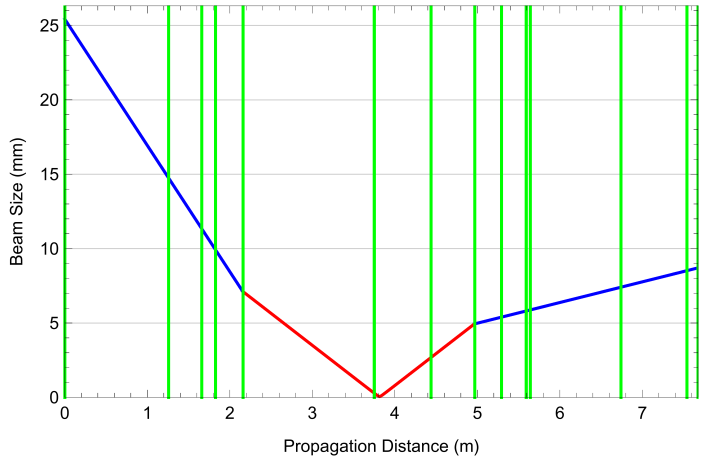
```
{{-0.0572249, 0.00140279}, {0.00399318, -17.475}}
```

```
inputbeam =
  OutRW[ 152 / 1000, 1330, 820 * 10^-9, (1  15.247) . (1  0) . (1  24.666)];
                                        (0     1  )   (-1/18 1)   (0     1  )
```

OutRW[wIN_, RIN_, λIN_, ABCD_]

PlotTheBeamPropagation[ABCDList_, RWInputMode_, λIN_]

```
DistancesFromSR3 = {15241.4, 16498.9, 16901.3, 17067.6, 17402.1, 18992.4, 19680.8,
    20210.2, 20534.3, 20834.3, 20884.3, 21982.5, 22782.5, 22916.1} / 1000;
DistancesFromPO = DistancesFromSR3 - DistancesFromSR3[[1]];

ABCDList =
  Reverse[ {(1  2.706) , (1  0) , (1  2.808) , (1  0) , (1  2.156) }];
           (0    1  )   (-1/1.6994 1)   (0    1  )   (1/1.6994 1)   (0    1  )

LambdaIn = 820 * 10^-9;

For[ii = 1, ii ≤ Length[DistancesFromPO], ii++, {
   If[ii == 1, plotL = ListPlot[{{DistancesFromPO[[ii]], 0},
        {DistancesFromPO[[ii]], 30}}, Joined → True, PlotStyle → Green], {
      plotL1 = ListPlot[{{DistancesFromPO[[ii]], 0},
         {DistancesFromPO[[ii]], 30}}, Joined → True, PlotStyle → Green];
      plotL = Show[plotL, plotL1];
     }];
  }];

plot1 = PlotTheBeamPropagation[ABCDList, inputbeam, LambdaIn];
Show[plot1, plotL, GridLines → {None, Automatic}, PlotRange → {0, 25}]
Export["/Advanced_LIGO/TCS/Hartmann sensor/H1_HWSX_beam_size.txt",
  BStempList, "Table"];
Export["/Advanced_LIGO/TCS/Hartmann sensor/H1_HWSX_beam_size_mirror_posns.txt",
  DistancesFromPO, "Table"];
```



Picomotors

LPM

```
MLPM = {{1, 2.808 - 509.6 / 1000}, {0, 1}}.{{1, 0}, {1 / 1.6994, 1}}.
    {{1, 2.156}, {0, 1}}.{{1, 15.247}, {0, 1}}.{{1, 0}, {-1 / 18, 1}}.{{1, 24.666}, {0, 1}};
MLPMRev = Inverse[MLPM]
```

```
{{10.3517, -42.0135}, {0.0360389, -0.0496651}}
```

```
DLPM = MLPMRev.{0, alpha1}
```

```
{0. - 42.0135 alpha1, 0. - 0.0496651 alpha1}
```

UPM

```
MUPM = {{1, 2.808 - 509.6 / 1000 - 630 / 1000}, {0, 1}}.
   {{1, 0}, {1 / 1.6994, 1}}.{{1, 2.156}, {0, 1}}.{{1, 15.247}, {0, 1}}.{{1, 0}, {-1 / 18, 1}}.{{1, 24.666}, {0, 1}}
MUPMRev = Inverse[MUPM];
DUPM = MUPMRev.{0, alpha2}
```

```
{{-0.0269606, 35.4919}, {-0.0360389, 10.3517}}
```

```
{0. - 35.4919 alpha2, 0. - 0.0269606 alpha2}
```

Hold alpha constant

```
Solve[DLPM[[2]] + DUPM[[2]] == 0, alpha1]
DLPM[[1]] + DUPM[[1]] /. %[[1]]
```

```
{{alpha1 → -20.1349 (0. + 0.0269606 alpha2)}}
```

```
0. + 845.937 (0. + 0.0269606 alpha2) - 35.4919 alpha2
```

H1Y

```
MatrixForm[H1YSoln]
```

$$\begin{pmatrix} 2.2826 \\ 2.9695 \\ 3.6715 \\ 1.13366 \\ 0.79362 \end{pmatrix}$$

```
Print["SR2ARtoVacLens = ", H1YSoln[[3]]]
Print["VacLens = ", H1YSoln[[5]]]
Print["VacLenstoAirLens = ", H1YSoln[[2]]]
Print["AirLens = ", H1YSoln[[4]]]
Print["AirLenstoHWS = ", H1YSoln[[1]]]
```

```
SR2ARtoVacLens = 3.6715

VacLens = 0.79362

VacLenstoAirLens = 2.9695

AirLens = 1.13366

AirLenstoHWS = 2.2826
```

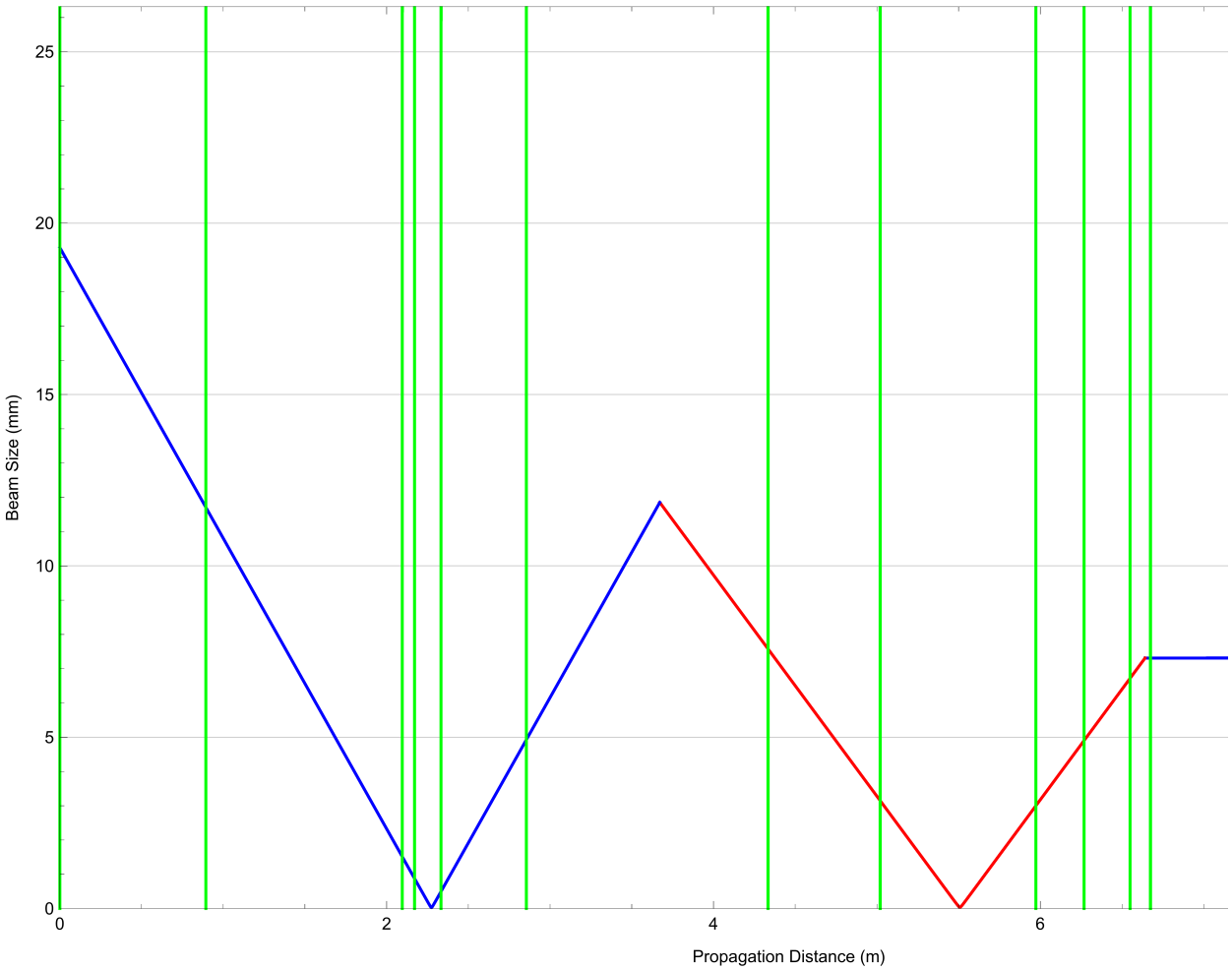
$$\begin{pmatrix} 1 & 4.005 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1/2.2659 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 3.640 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 1/1.6994 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2.333 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1.453 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0.075 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -0.04849 & 0.6882 \end{pmatrix} \cdot \begin{pmatrix} 1 & 15.461 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1/18 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 24.792 \\ 0 & 1 \end{pmatrix};$$

```
inputbeam = OutRW[ 127 / 1000, 1330, 820 * 10^-9,
```

$$\begin{pmatrix} 1 & 0 \\ 0 & 1.453 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0.075 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -0.04849 & 0.6882 \end{pmatrix} \cdot \begin{pmatrix} 1 & 15.461 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1/18 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 24.792 \\ 0 & 1 \end{pmatrix}]$$

```
{-2.27411, 0.019287}

DistancesFromSR2 = {0, 893.5, 2095.7, 2170.7, 2333.0, 2855.1, 4334.2, 5020.3,
    5973.0, 6267.8, 6549.4, 6674.4, 7756.0, 8281.0, 8478.1, 8794.0} / 1000.0;
```

```
ABCDList = Reverse[ {( 1  H1YSoln[[1]] ), (        1          0 ),
                     ( 0        1       )  ( -1 / H1YSoln[[4]]  1 ),
    ( 1  H1YSoln[[2]] ), (        1          0 ), ( 1  H1YSoln[[3]] )}];
    ( 0        1       )  ( -1 / H1YSoln[[5]]  1 ), ( 0        1       )
LambdaIn = 820 * 10^-9;

For[ii = 1, ii ≤ Length[DistancesFromSR2], ii++, {
   If[ii == 1, plotL = ListPlot[{{DistancesFromSR2[[ii]], 0},
         {DistancesFromSR2[[ii]], 30}}, Joined → True, PlotStyle → Green], {
      plotL1 = ListPlot[{{DistancesFromSR2[[ii]], 0},
         {DistancesFromSR2[[ii]], 30}}, Joined → True, PlotStyle → Green];
      plotL = Show[plotL, plotL1];
     }];
  }];

plot1 = PlotTheBeamPropagation[ABCDList, inputbeam, LambdaIn];
Export["/Advanced_LIGO/TCS/Hartmann sensor/H1_HWSY_beam_size.txt",
  BStempList, "Table"];
Export["/Advanced_LIGO/TCS/Hartmann sensor/H1_HWSY_beam_size_mirror_posns.txt",
  DistancesFromSR2, "Table"];
Show[plot1, plotL, GridLines → {None, Automatic}, PlotRange → {0, 25}]
```



Periscope

UPM

```
MUPM = ( 1  3.640 - (637 + 50 + 180 + 136.4 + 381.7 + 125 + 60) / 1000 ) .
       ( 0                                  1                         )
    (     1      0 ) . ( 1  2.333 ) . ( 1    0 ) . ( 1  0.075 ) .
    ( 1 / 1.6994 1 )   ( 0    1   )   ( 0  1.453 )   ( 0    1 )
    (     1        0 ) . ( 1  15.461 ) . (    1    0 ) . ( 1  24.792 );
    ( -0.04849  0.6882 )   ( 0     1   )   ( -1 / 18 1 )   ( 0     1 )
MUPMRev = Inverse[MUPM]
MUPMRev.{0, alpha}
```

```
{{7.03929, -29.4444}, {0.0743895, -0.169096}}
```

```
{0. - 29.4444 alpha, 0. - 0.169096 alpha}
```

LPM

```
MLPM = ( 1  3.640 - ( 50 + 180 + 136.4 + 381.7 + 125 + 60) / 1000 ) .
       ( 0                                  1                      )
    (     1      0 ) . ( 1  2.333 ) . ( 1    0 ) . ( 1  0.075 ) .
    ( 1 / 1.6994 1 )   ( 0    1   )   ( 0  1.453 )   ( 0    1 )
    (     1        0 ) . ( 1  15.461 ) . (    1    0 ) . ( 1  24.792 );
    ( -0.04849  0.6882 )   ( 0     1   )   ( -1 / 18 1 )   ( 0     1 )
MLPMRev = Inverse[MLPM]
MLPMRev.{0, alpha}
```

```
{{7.03929, -33.9285}, {0.0743895, -0.216482}}
```

```
{0. - 33.9285 alpha, 0. - 0.216482 alpha}
```

Alignment Matrices

```
VacLenstoUPM = (637 + 50 + 180 + 136.4 + 381.7 + 125 + 60) / 1000;
VacLenstoLPM = (50 + 180 + 136.4 + 381.7 + 125 + 60) / 1000;
HWStoBS = 38.5 * 25.4 / 1000;
BStoLens1 = 18 * 25.4 / 1000;
BStoLens2 = 21 * 25.4 / 1000;
LensQPD = 200 / 1000;
LensQPDtoQPD1 = 9 * 25.4 / 1000;
LensQPDtoQPD2 = 5 * 25.4 / 1000;

alignMat1 = Simplify[
    ( 1  LensQPDtoQPD ) . (      1      0 ) . ( 1  H1YSoln[[1]] - HWStoBS + BStoLens ) .
    ( 0       1       )   ( -1 / LensQPD 1 )   ( 0                 1               )
     (        1          0 ) . ( 1  H1YSoln[[2]] - VacLenstoPM ) ];
     ( -1 / H1YSoln[[4]] 1 )   ( 0                 1           )
```

UPM

```
alignMatUPM1 = alignMat1 /. {LensQPDtoQPD → LensQPDtoQPD1,
    BStoLens → BStoLens1, VacLenstoPM → VacLenstoUPM};
alignMatUPM2 = alignMat1 /. {LensQPDtoQPD → LensQPDtoQPD2,
    BStoLens → BStoLens2, VacLenstoPM → VacLenstoUPM};
dxUPM1 = alignMatUPM1[[1]][[2]]
dxUPM2 = alignMatUPM2[[1]][[2]]
```

```
-0.19464
```

```
0.323745
```

LPM

```
alignMatUPM1 = alignMat1 /. {LensQPDtoQPD → LensQPDtoQPD1,
    BStoLens → BStoLens1, VacLenstoPM → VacLenstoLPM};
alignMatUPM2 = alignMat1 /. {LensQPDtoQPD → LensQPDtoQPD2,
    BStoLens → BStoLens2, VacLenstoPM → VacLenstoLPM};
dxLPM1 = alignMatUPM1[[1]][[2]]
dxLPM2 = alignMatUPM2[[1]][[2]]
```

```
-0.27261
```

```
0.107908
```

MatrixPeriscope

```
periM = {{dxUPM1, dxUPM2}, {dxLPM1, dxLPM2}}
periMInv = Inverse[periM]
```

```
{{-0.19464, 0.323745}, {-0.27261, 0.107908}}
```

```
{{1.60452, -4.81385}, {4.05352, -2.89416}}
```

```
MatrixForm[periM]
MatrixForm[periMInv]
```

$$\begin{pmatrix} -0.19464 & 0.323745 \\ -0.27261 & 0.107908 \end{pmatrix}$$

$$\begin{pmatrix} 1.60452 & -4.81385 \\ 4.05352 & -2.89416 \end{pmatrix}$$

```
Det[periM]
```

```
0.0672527
```

```
Det[IdentityMatrix[2]]
```

```
1
```