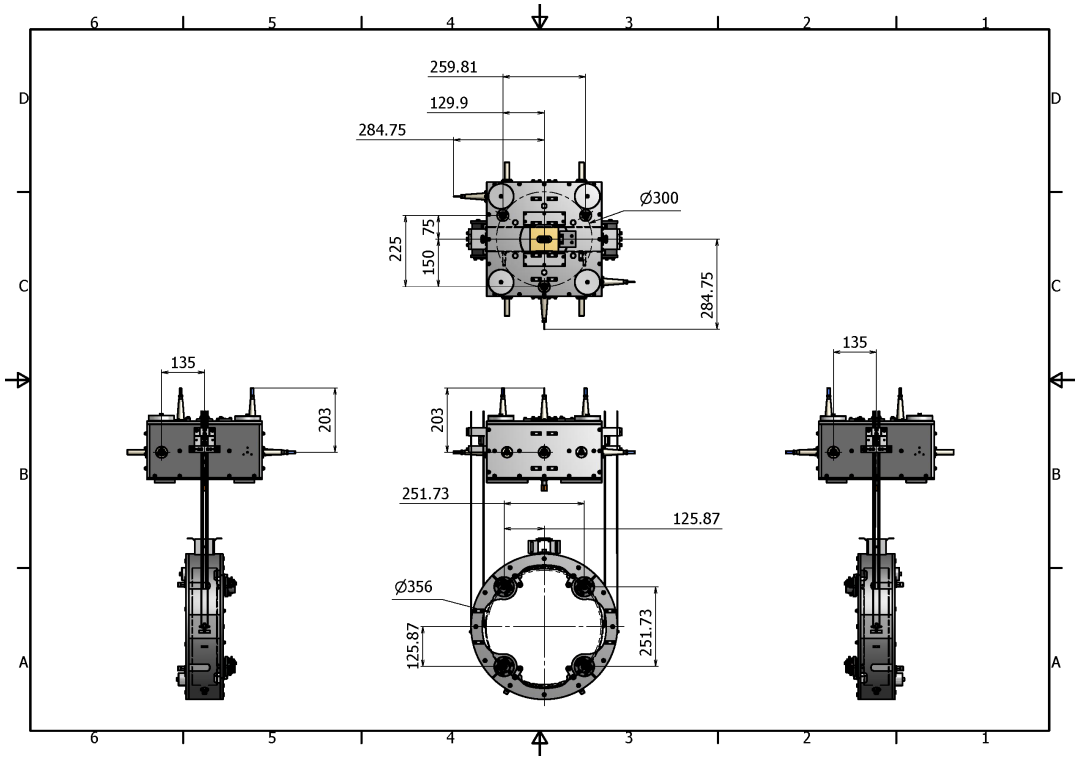
T1707205-v6 - PI Matrix Calc - BS Final Hang

2018-02-27, -v5, Mark Barton: PI items renumbered from 1 (instead of #0).

2018-11-27, -v6, Mark Barton: Improved PI geometry from JGW-D1707077 with separate angles for LVDT sensing and actuation.

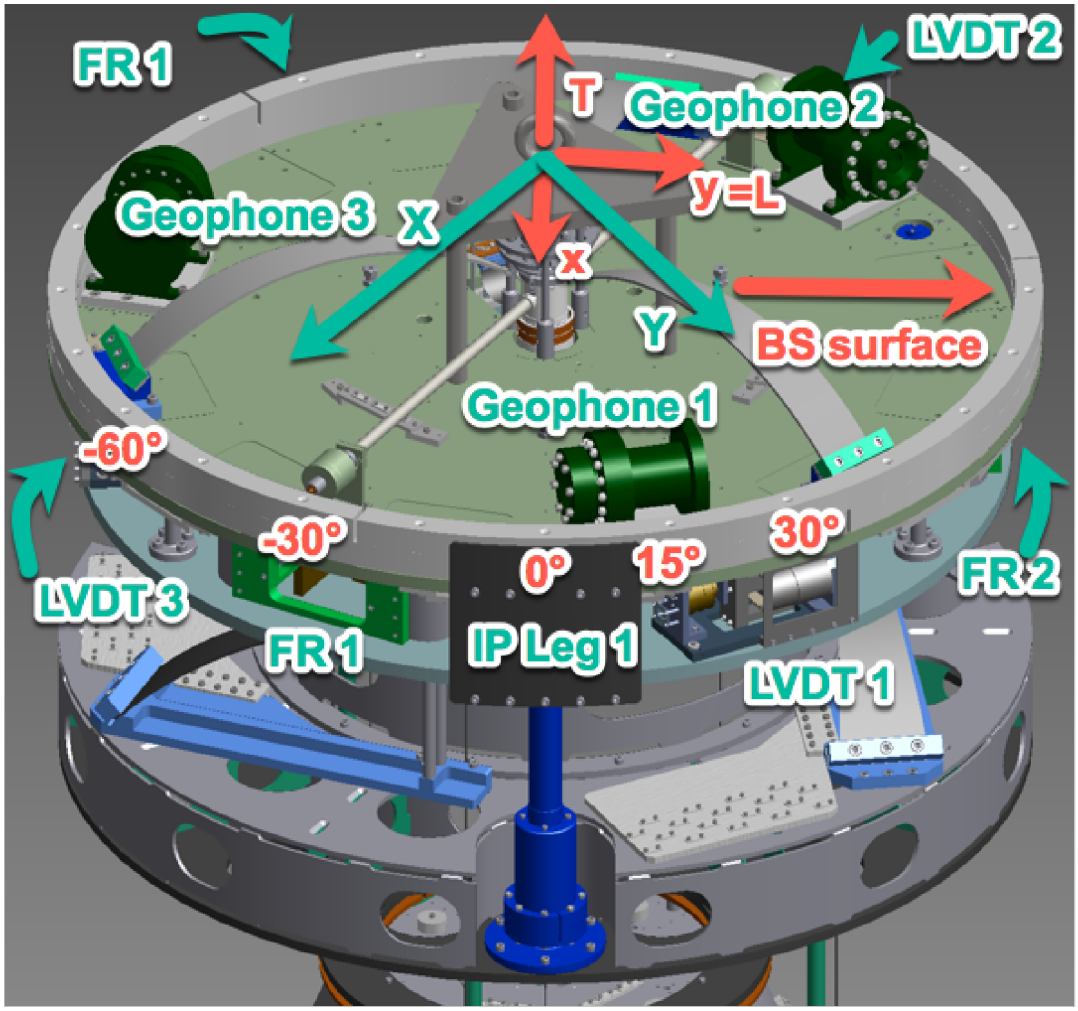
Data

Payload Data



The calculation is for the BS Final Hang arrangement in the tank.
X and Y are the standard interferometer global coordinates.
The BS surface faces midway between +Y and -X, and defines L (longitudinal).
T (transverse) is 90° anticlockwise from L (so as to make right handed LTV with vertical up).
For convenience, so that standard rotation matrices can be used, the calculation is done first in x/y coordinates where y=L and x=-T.
Leg 1 is in the +x direction and angles are measured anticlockwise from it.
FR 1, Geophone 1 and LVDT 1 are offset by -30°, 15° and 30° from Leg 1.
Legs 2 and 3 are at 120° and 240°.

PI Data


FR 1
T
Geophone 2
LVDT 2
y=L
Geophone 3
X
x
Y
BS surface
Geophone 1
-60°
-30°
0°
15°
30°
LVDT 3
FR 1
IP Leg 1
LVDT 1
FR 2

Data Summary

```
vals = {
   (* IP stuff *)
   R -> 715. / 1000, (* PI radius *)
   Fradius -> 1268.8 / 1000 / 2, (* from center to FR stepper; 3D CAD *)
   Gradius -> 591.5 / 1000, (* from center to geophone; JGW-D1707077_rev3, p9 *)
   Lradius -> 599.26 / 1000,
   (* from center to LVDT sensing; JGW-D1707077_rev3, p10 *)
   Aradius -> 597.31 / 1000, (* from center to LVDT actuation;
   JGW-D1707077_rev3, p10 *)
   Eradius → 650. / 1000, (* from center to EQ stop hole; guesstimate *)
   Iradius → 650. / 1000, (* from center to leg; guesstimate *)
   Eradius2 → 80. / 1000 / 2, (* radius of EQ stop hole; guesstimate *)
   Iradius2 → 80. / 1000 / 2, (* radius of leg; guesstimate *)
   l -> 200. / 1000, (* arrow length, for diagram *)
   origin -> 0 °, (* 3 o'clock *)
   (* IM stuff *)
   IMV1L → 0.15,
   IMV2L → 0.15 * Sin[Pi / 6],
   IMV3L → 0.15 * Sin[Pi / 6],
   IMV1T → 0,
   IMV2T → 0.15 * Cos[Pi / 6],
   IMV3T → 0.15 * Cos[Pi / 6],
   IMH1L → 0,
   IMH2L → 0.135,
   IMH3L → 0.135,
   (* TM stuff *)
   TMH1V → 0.25173 / 2,
   TMH2V → 0.25173 / 2,
   TMH3V → 0.25173 / 2,
   TMH4V → 0.25173 / 2,
   TMH1T → 0.25173 / 2,
   TMH2T → 0.25173 / 2,
   TMH3T → 0.25173 / 2,
   TMH4T → 0.25173 / 2
  };
```

TM Geometry

- **The test matrix, which gives the change at each sensor for excursions in L, P and Y**

```
(TMtest = Transpose[{
       {-1, -1, -1, -1},
       {-TMH1V, TMH2V, -TMH3V, TMH4V},
       {TMH1T, TMH2T, -TMH3T, -TMH4T}
      }] /. vals
 ) // TableForm
```

```
-1    -0.125865    0.125865
-1    0.125865     0.125865
-1    -0.125865    -0.125865
-1    0.125865     -0.125865
```

- **The diagonalization matrix, which attempts to undo the test matrix. This matrix should be typed as-is into the OSEM2EUL screen and transposed into the EUL2OSEM screen.**

```
(TM = {
      -0.25 * {1, 1, 1, 1},
      0.25 * {-1 / TMH1V, 1 / TMH2V, -1 / TMH3V, 1 / TMH4V},
      0.25 * {1 / TMH1T, 1 / TMH2T, -1 / TMH3T, -1 / TMH4T}
     } /. vals
 ) // TableForm
```

```
-0.25       -0.25      -0.25       -0.25
-1.98626    1.98626    -1.98626    1.98626
1.98626     1.98626    -1.98626    -1.98626
```

- **Multiplying the diagonalization matrix and the test matrix should give the identity matrix.**

```
TM.TMtest // TableForm
```

```
1.    0.    0.
0.    1.    0.
0.    0.    1.
```

IM Geometry

- **The test matrix, which gives the change at each sensor for excursions in L, T, V, R, P and Y**

```
(IMtest = Transpose[{
         {0, 0, 0, -1, 0, 0}, (* L *)
         {0, 0, 0, 0, -1, 1}, (* T *)
         {-1, -1, -1, 0, 0, 0}, (* V *)
         {0, -IMV2T, IMV3T, 0, 0, 0}, (* R *)
         {IMV1L, -IMV2L, -IMV3L, 0, 0, 0}, (* P *)
         {0, 0, 0, 0, -IMH2L, -IMH3L} (* Y *)
        }] /. vals // N
  ) // Transpose // TableForm
```

```
0.      0.           0.          -1.    0.        0.
0.      0.           0.          0.     -1.       1.
-1.     -1.          -1.         0.     0.        0.
0.      -0.129904    0.129904    0.     0.        0.
0.15    -0.075       -0.075      0.     0.        0.
0.      0.           0.          0.     -0.135    -0.135
```

- **The diagonalization matrix, which attempts to undo the test matrix. This matrix should be typed as-is into the OSEM2EUL screen and transposed into the EUL2OSEM screen.**

```
(IM = {
       {0, 0, 0, -1, 0, 0}, (* L *)
       {0, 0, 0, 0, -1, 1} / 2, (* T *)
       {-1, -1, -1, 0, 0, 0} / 3, (* V *)
       {0, -1 / IMV2T, 1 / IMV3T, 0, 0, 0} / 2, (* R *)
       {4 / IMV1L, -1 / IMV2L, -1 / IMV3L, 0, 0, 0} / 6, (* P *)
       {0, 0, 0, 0, -1 / IMH2L, -1 / IMH3L} / 2 (* Y *)
      } /. vals // N
 ) // TableForm
```

```
0.           0.           0.           -1.    0.         0.
0.           0.           0.           0.     -0.5       0.5
-0.333333    -0.333333    -0.333333    0.     0.         0.
0.           -3.849       3.849        0.     0.         0.
4.44444      -2.22222     -2.22222     0.     0.         0.
0.           0.           0.           0.     -3.7037    -3.7037
```

- **Multiplying the diagonalization matrix and the test matrix should give the identity matrix.**

```
IM.IMtest // TableForm
```

```
1.    0.    0.    0.    0.    0.
0.    1.    0.    0.    0.    0.
0.    0.    1.    0.    0.    0.
0.    0.    0.    1.    0.    0.
0.    0.    0.    0.    1.    0.
0.    0.    0.    0.    0.    1.
```

PI Geometry

Calculation

- **90° Rotation matrix for x/y -> L/T.**

```
Ninety = RotationMatrix[90 °]
```

```
{{0, -1}, {1, 0}}
```

- **Angular locations of the various items**

IP legs

```
Iangles = {0 °, 120 °, 240 °};
```

Geophones

```
Gangles = 15 ° + Iangles (* JGW-D1707077_rev3, p9 *)
```

```
{15 °, 135 °, 255 °}
```

LVDTs

```
Langles = 22.4 ° + Iangles (* JGW-D1707077_rev3, p10 *)
```

```
{0.390954, 2.48535, 4.57974}
```

```
Aangles = 36.03 ° + Iangles (* JGW-D1707077_rev3, p10 *)
```

```
{0.628842, 2.72324, 4.81763}
```

Fishing rod steppers

```
Fangles = -30 ° + Iangles
```

```
{-30 °, 90 °, 210 °}
```

EQ stop positions

```
Eangles = -60 ° + Iangles
```

{-60 °, 60 °, 180 °}

- **Positions of the various items as vectors**

These are calculated by rotating the x axis {1,0} by the corresponding angles.

```
Ipositions = Iradius * Table[RotationMatrix[Iangles[[i]]].{1, 0}, {i, 1, 3}]
```

$$\left\{\{\text{Iradius}, 0\}, \left\{-\frac{\text{Iradius}}{2}, \frac{\sqrt{3}\ \text{Iradius}}{2}\right\}, \left\{-\frac{\text{Iradius}}{2}, -\frac{\sqrt{3}\ \text{Iradius}}{2}\right\}\right\}$$

```
Fpositions = Fradius * Table[RotationMatrix[Fangles[[i]]].{1, 0}, {i, 1, 3}]
```

$$\left\{\left\{\frac{\sqrt{3}\ \text{Fradius}}{2}, -\frac{\text{Fradius}}{2}\right\}, \{0, \text{Fradius}\}, \left\{-\frac{\sqrt{3}\ \text{Fradius}}{2}, -\frac{\text{Fradius}}{2}\right\}\right\}$$

$$\left\{\left\{\frac{\sqrt{3}\ \text{Fradius}}{2}, -\frac{\text{Fradius}}{2}\right\}, \{0, \text{Fradius}\}, \left\{-\frac{\sqrt{3}\ \text{Fradius}}{2}, -\frac{\text{Fradius}}{2}\right\}\right\}$$

$$\left\{\left\{\frac{\sqrt{3}\ \text{Fradius}}{2}, -\frac{\text{Fradius}}{2}\right\}, \{0, \text{Fradius}\}, \left\{-\frac{\sqrt{3}\ \text{Fradius}}{2}, -\frac{\text{Fradius}}{2}\right\}\right\}$$


```
Gpositions = Gradius * Table[RotationMatrix[Gangles[[i]]].{1, 0}, {i, 1, 3}]
```

$$\left\{\left\{\frac{\left(1+\sqrt{3}\right)\text{Gradius}}{2\sqrt{2}}, \frac{\left(-1+\sqrt{3}\right)\text{Gradius}}{2\sqrt{2}}\right\}, \left\{-\frac{\text{Gradius}}{\sqrt{2}}, \frac{\text{Gradius}}{\sqrt{2}}\right\}, \left\{-\frac{\left(-1+\sqrt{3}\right)\text{Gradius}}{2\sqrt{2}}, -\frac{\left(1+\sqrt{3}\right)\text{Gradius}}{2\sqrt{2}}\right\}\right\}$$

```
Lpositions = Lradius * Table[RotationMatrix[Langles[[i]]].{1, 0}, {i, 1, 3}]
```

{{0.924546 Lradius, 0.38107 Lradius},
{-0.79229 Lradius, 0.610145 Lradius}, {-0.132256 Lradius, -0.991216 Lradius}}

```
Apositions = Aradius * Table[RotationMatrix[Aangles[[i]]].{1, 0}, {i, 1, 3}]
```

{{0.808709 Aradius, 0.588209 Aradius},
{-0.913758 Aradius, 0.406258 Aradius}, {0.105049 Aradius, -0.994467 Aradius}}

```
Epositions = Eradius * Table[RotationMatrix[Eangles[[i]]].{1, 0}, {i, 1, 3}]
```

$$\left\{\left\{\frac{\text{Eradius}}{2}, -\frac{\sqrt{3}\ \text{Eradius}}{2}\right\}, \left\{\frac{\text{Eradius}}{2}, \frac{\sqrt{3}\ \text{Eradius}}{2}\right\}, \{-\text{Eradius}, 0\}\right\}$$

- **Orientations of the various items as unit vectors**

These are calculated by rotating a unit y vector {0,1} (i.e., the tangent vector at {1,0}) by the same set of angles.

```
Fvectors = Table[RotationMatrix[Fangles[[i]]].{0, 1}, {i, 1, 3}]
```

$$\left\{\left\{\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}, \{-1, 0\}, \left\{\frac{1}{2}, -\frac{\sqrt{3}}{2}\right\}\right\}$$

```
Gvectors = Table[RotationMatrix[Gangles[[i]]].{0, 1}, {i, 1, 3}]
```

$$\left\{\left\{-\frac{-1+\sqrt{3}}{2\sqrt{2}}, \frac{1+\sqrt{3}}{2\sqrt{2}}\right\}, \left\{-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right\}, \left\{\frac{1+\sqrt{3}}{2\sqrt{2}}, -\frac{-1+\sqrt{3}}{2\sqrt{2}}\right\}\right\}$$

```
Lvectors = Table[RotationMatrix[Langles[[i]]].{0, 1}, {i, 1, 3}]
```

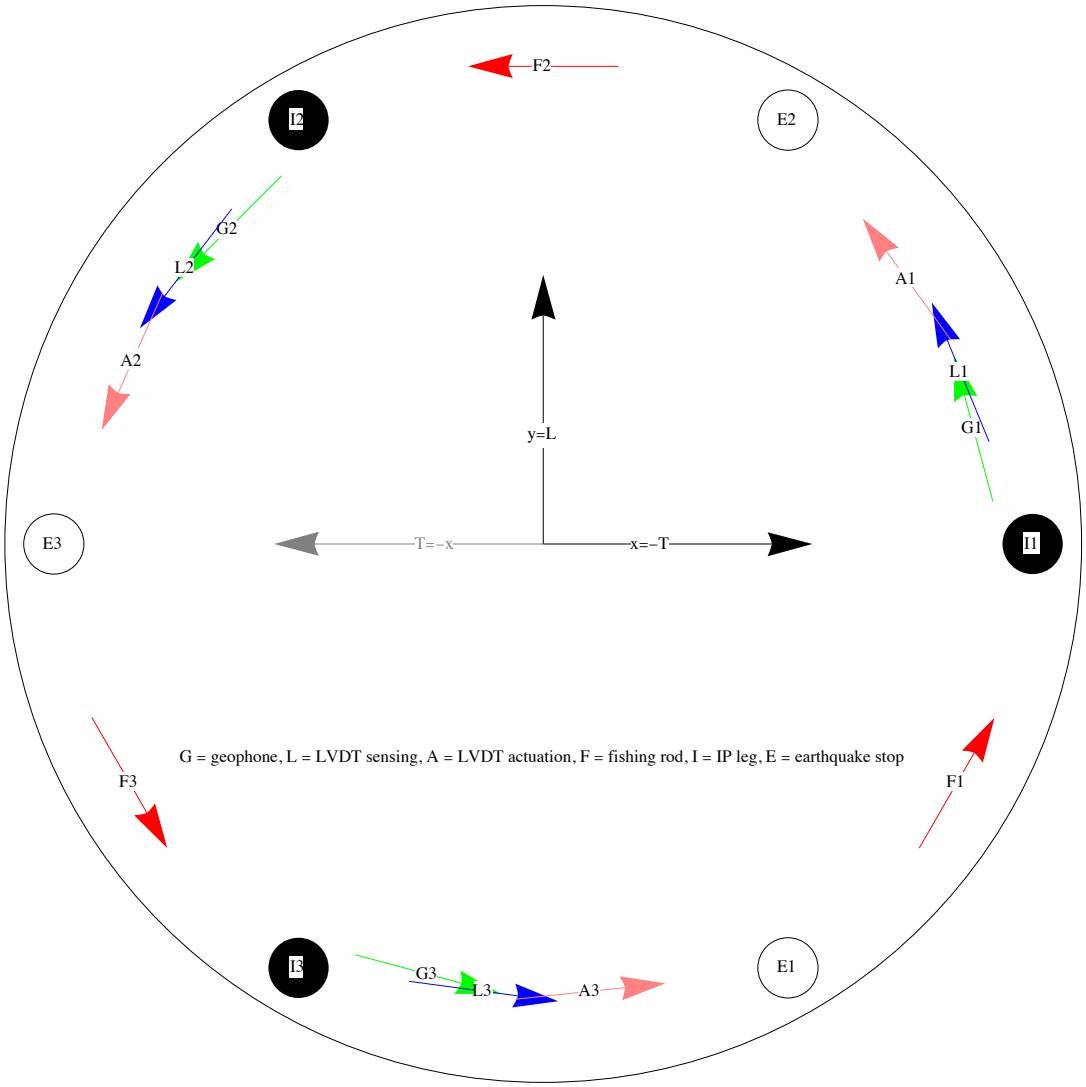
```
{{-0.38107, 0.924546}, {-0.610145, -0.79229}, {0.991216, -0.132256}}
```

```
Avectors = Table[RotationMatrix[Aangles[[i]]].{0, 1}, {i, 1, 3}]
```

```
{{-0.588209, 0.808709}, {-0.406258, -0.913758}, {0.994467, 0.105049}}
```

- **Diagram**

```
Graphics[{
   Circle[{0, 0}, R],
   Sequence[Table[Disk[Ipositions[[i]], Iradius2],
     {i, 1, 3}
    ]],
   Sequence[Table[Text["I" <> ToString[i], Ipositions[[i]], Background → White],
     {i, 1, 3}]], Sequence[Table[Circle[Epositions[[i]], Eradius2],
     {i, 1, 3}
    ]],
   Sequence[
    Table[Text["E" <> ToString[i], Epositions[[i]], Background → White], {i, 1, 3}]],
   Red,
   Sequence[Table[Arrow[{Fpositions[[i]] - Fvectors[[i]] * l / 2,
       Fpositions[[i]] + Fvectors[[i]] * l / 2}], {i, 1, 3}
    ]],
   Black, Sequence[
    Table[Text["F" <> ToString[i], Fpositions[[i]], Background → White], {i, 1, 3}]],
   Green,
   Sequence[Table[Arrow[{Gpositions[[i]] - Gvectors[[i]] * l / 2,
       Gpositions[[i]] + Gvectors[[i]] * l / 2}], {i, 1, 3}
    ]],
   Black, Sequence[
    Table[Text["G" <> ToString[i], Gpositions[[i]], Background → White], {i, 1, 3}]],
   Blue,
   Sequence[Table[Arrow[{Lpositions[[i]] - Lvectors[[i]] * l / 2,
       Lpositions[[i]] + Lvectors[[i]] * l / 2}], {i, 1, 3}
    ]],
   Black,
   Sequence[
    Table[Text["L" <> ToString[i], Lpositions[[i]], Background → White], {i, 1, 3}]],
   Pink,
   Sequence[Table[Arrow[{Apositions[[i]] - Avectors[[i]] * l / 2,
       Apositions[[i]] + Avectors[[i]] * l / 2}], {i, 1, 3}
    ]],
   Black, Sequence[
    Table[Text["A" <> ToString[i], Apositions[[i]], Background → White], {i, 1, 3}]],
   Black,
   Arrow[{{0, 0}, {R / 2, 0}}], Text["x=-T", {R / 5, 0}, Background → White],
   Arrow[{{0, 0}, {0, R / 2}}], Text["y=L", {0, R / 5}, Background → White],
   Gray,
   Arrow[{{0, 0}, {-R / 2, 0}}], Text["T=-x", {-R / 5, 0}, Background → White],
   Black,
   Text["G = geophone, L = LVDT sensing, A = LVDT actuation, F =
      fishing rod, I = IP leg, E = earthquake stop", {0, -0.4 * R}]
  }] /.
 vals
```



■ Matrices between L/T/Y and LVDTs

Sensing Matrix. Usage: {L0, L1, L2}= LVDTfromLTY.{L, T, Y}

```
(LVDTfromLTY = {
    Table[Lvectors[[i]].Ninety.{1, 0}, {i, 1, 3}],
    Table[Lvectors[[i]].Ninety.{0, 1}, {i, 1, 3}],
    Table[Lradius, {i, 1, 3}]
   }) // TableForm
```

0.924546	−0.79229	−0.132256
0.38107	0.610145	−0.991216
Lradius	Lradius	Lradius

Usage: {L, T, Y}= LTYfromLVDT.{L0, L1, L2}

```
(LTYfromLVDT = Inverse[LVDTfromLTY]) // TableForm
```

0.616364	0.254047	$\frac{0.333333}{\text{Lradius}}$
−0.528193	0.406763	$\frac{0.333333}{\text{Lradius}}$
−0.0881709	−0.66081	$\frac{0.333333}{\text{Lradius}}$

Usage: type the numbers below into LVDT2EUL

```
Transpose[LTYfromLVDT] /. vals // N // TableForm
```

```
0.616364    -0.528193    -0.0881709
0.254047    0.406763     -0.66081
0.556242    0.556242     0.556242
```

Actuation Matrix. Usage: {A0, A1, A2}= LVDTAfromLTY.{L, T, Y}

```
(LVDTAfromLTY = {
    Table[Avectors[[i]].Ninety.{1, 0}, {i, 1, 3}],
    Table[Avectors[[i]].Ninety.{0, 1}, {i, 1, 3}],
    Table[Aradius, {i, 1, 3}]
   }) // TableForm
```

```
0.808709    -0.913758    0.105049
0.588209    0.406258     -0.994467
Aradius     Aradius      Aradius
```

Usage: type the numbers below into EUL2COIL

```
Transpose[LVDTAfromLTY] /. vals // N // TableForm
```

```
0.808709     0.588209     0.59731
-0.913758    0.406258     0.59731
0.105049     -0.994467    0.59731
```

■ Matrices between L/T/Y and Geophones

Usage: {G0, G1, G2}= GfromLTY.{L, T, Y} (assuming the geophones had actuation)

```
(GfromLTY = {
    Table[Gvectors[[i]].Ninety.{1, 0}, {i, 1, 3}],
    Table[Gvectors[[i]].Ninety.{0, 1}, {i, 1, 3}],
    Table[Gradius, {i, 1, 3}]
   }) // TableForm
```

$$\frac{1+\sqrt{3}}{2\sqrt{2}} \qquad -\frac{1}{\sqrt{2}} \qquad -\frac{-1+\sqrt{3}}{2\sqrt{2}}$$

$$\frac{-1+\sqrt{3}}{2\sqrt{2}} \qquad \frac{1}{\sqrt{2}} \qquad -\frac{1+\sqrt{3}}{2\sqrt{2}}$$

```
Gradius    Gradius    Gradius
```

Usage: you would type these numbers into EUL2ACC if it existed, which it doesn't because geophones don't have actuation.

```
Transpose[GfromLTY] /. vals // N // TableForm
```

```
0.965926     0.258819     0.5915
-0.707107    0.707107     0.5915
-0.258819    -0.965926    0.5915
```

Usage: {L, T, Y}= LTYfromG.{G0, G1, G2}

```
(LTYfromG = Inverse[GfromLTY] // Simplify) // TableForm
```

$$\frac{3+\sqrt{3}}{3\sqrt{6}} \qquad \frac{-1+\sqrt{3}}{3\sqrt{2}} \qquad \frac{1}{3\,\text{Gradius}}$$

$$-\frac{\sqrt{2}}{3} \qquad \frac{\sqrt{2}}{3} \qquad \frac{1}{3\,\text{Gradius}}$$

$$\frac{-3+\sqrt{3}}{3\sqrt{6}} \qquad -\frac{3+\sqrt{3}}{3\sqrt{6}} \qquad \frac{1}{3\,\text{Gradius}}$$

Usage: type the numbers below into ACC2EUL

```
Transpose[LTYfromG] /. vals // N // TableForm
```

```
0.643951    -0.471405    -0.172546
0.172546    0.471405     -0.643951
0.563539    0.563539     0.563539
```

■ Matrices between L/T/Y and FRs

Usage: {F0, F1, F2}= FfromLTY.{L, T, Y}

```
(FfromLTY = {
    Table[Fvectors[[i]].Ninety.{1, 0}, {i, 1, 3}],
    Table[Fvectors[[i]].Ninety.{0, 1}, {i, 1, 3}],
    Table[Fradius, {i, 1, 3}]
   }) // TableForm
```

$$\begin{array}{lll} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ \text{Fradius} & \text{Fradius} & \text{Fradius} \end{array}$$

Usage: write a Python script to take DC force/torque requests in LTY coordinates, multiply by the following matrix and output stepper motor movements in steps.

```
FfromLTY /. vals // N // TableForm
```

```
0.866025    0.          -0.866025
-0.5        1.          -0.5
0.6344      0.6344      0.6344
```

Usage: {L, T, Y}= LTYfromF.{F0, F1, F2}

```
(LTYfromF = Inverse[FfromLTY] // Simplify) // TableForm
```

$$\begin{array}{lll} \frac{1}{\sqrt{3}} & -\frac{1}{3} & \frac{1}{3\,\text{Fradius}} \\ 0 & \frac{2}{3} & \frac{1}{3\,\text{Fradius}} \\ -\frac{1}{\sqrt{3}} & -\frac{1}{3} & \frac{1}{3\,\text{Fradius}} \end{array}$$

Usage: you would type these numbers into FR2EUL, if it existed, which it doesn't because FRs don't have sensing.

```
Transpose[LTYfromF] /. vals // N // TableForm
```

```
0.57735      0.          -0.57735
-0.333333    0.666667    -0.333333
0.525431     0.525431    0.525431
```