$$> eq1 := delay = \frac{1}{c} \cdot \sqrt{x^2 + \left(y + \frac{h}{\tan(elevation)}\right)^2 + h^2} - \frac{h}{\sin(elevation)} - y \cdot \cos(elevation)$$

$$eq1 := delay = \frac{\sqrt{x^2 + \left(y + \frac{h}{\tan(elevation)}\right)^2 + h^2}}{c} - \frac{h}{\sin(elevation)} - y\cos(elevation) \qquad \textbf{(1)}$$

$$> eq2 := f\_doppler = -v\_ty \cdot \cos(elevation) - v\_tz \cdot \sin(elevation) + \frac{v\_rx \cdot x + v\_ry \cdot \left(y + \frac{h}{\tan(elevation)}\right) - v\_rz \cdot h}{\sqrt{x^2 + \left(y + \frac{h}{\tan(elevation)}\right)^2 + h^2}}$$

$$eq2 := f\_doppler = -v\_ty\cos(elevation) - v\_tz\sin(elevation) + \frac{v\_rx\, x + v\_ry\left(y + \frac{h}{\tan(elevation)}\right) - v\_rz\, h}{\sqrt{x^2 + \left(y + \frac{h}{\tan(elevation)}\right)^2 + h^2}} \qquad \textbf{(2)}$$

$$> sys := \{eq1, eq2\}$$

$$sys := \left\{delay = \frac{\sqrt{x^2 + \left(y + \frac{h}{\tan(elevation)}\right)^2 + h^2}}{c} - \frac{h}{\sin(elevation)} - y\cos(elevation), f\_doppler = -v\_ty\cos(elevation) - v\_tz\sin(elevation) + \frac{v\_rx\, x + v\_ry\left(y + \frac{h}{\tan(elevation)}\right) - v\_rz\, h}{\sqrt{x^2 + \left(y + \frac{h}{\tan(elevation)}\right)^2 + h^2}}\right\} \qquad \textbf{(3)}$$

$$> sol := solve(sys, \{x, y\}) :$$

$$> sol[1] :$$

$$> all\_x := allvalues(sol[1]) :$$

$$> sol\_x\_1 := simplify(all\_x[1]) :$$

$$> simplify\Bigg(eval\Bigg(all\_x[1], \Bigg\{elevation = \frac{30 \cdot \pi}{180}, v\_ty = 2121, v\_tz = 5, v\_rx = 2210, v\_ry = 7299, v\_rz = 199, h = 500000, f\_doppler = -500, delay = \frac{1}{1.023e6}, c$$

$= 299792458\}\big)\big)\big)$ *# Check solution*

$$x = -1.654743318\ 10^{10} \qquad \textbf{(4)}$$

> *with*(*CodeGeneration*) :

> *Python*(*sol_x_1*)

```
cg4 = x == ((math.cos(elevation) ** 2 * c * v_ty + c * (v_tz *
math.sin(elevation) + f_doppler) * math.cos(elevation) - v_ry) *
math.sqrt(-2 * ((c * delay * h * (v_ty - v_tz) * (v_ty + v_tz) *
math.cos(elevation) ** 2 + (((c * v_ty - v_ry) * v_tz - v_rz *
(c * v_ry - v_ty)) * h ** 2 + 2 * c * f_doppler * h * v_ty *
delay + c * v_tz * v_ty * delay ** 2) * math.cos(elevation) +
f_doppler * (c * v_tz + v_rz) * h ** 2 + (v_tz ** 2 * c + v_rz *
v_tz + c * (-v_rx ** 2 - v_ry ** 2 + f_doppler ** 2)) * delay *
h + c * delay ** 2 * v_tz * f_doppler) * c * math.sin(elevation)
- 2 * math.cos(elevation) ** 3 * c ** 2 * delay * h * v_ty *
v_tz - 2 * ((v_tz ** 2 * c / 4 + v_rz * v_tz / 2 - c * v_ty ** 2
/ 4 + v_ry * v_ty / 2 + c * (-v_ry ** 2 / 4 + v_rz ** 2 / 4)) *
h ** 2 + c * delay * h * v_tz * f_doppler - c * delay ** 2 *
(v_ty - v_tz) * (v_ty + v_tz) / 4) * c * math.cos(elevation) **
2 + (f_doppler * (c * v_ty - v_ry) * h ** 2 - (-2 * c * v_tz *
v_ty + v_rz * (c * v_ry - v_ty)) * delay * h + c * delay ** 2 *
v_ty * f_doppler) * c * math.cos(elevation) + (c ** 2 * v_tz **
2 / 2 + c * v_rz * v_tz + c ** 2 * f_doppler ** 2 / 2 + (-v_rx *
* 2 / 2 - v_ry ** 2 / 2) * c ** 2 + v_rx ** 2 / 2 + v_rz ** 2 /
2 + v_ry ** 2 / 2) * h ** 2 + 2 * (c * v_tz + v_rz / 2) *
f_doppler * delay * c * h + c ** 2 * delay ** 2 * (-v_rx ** 2 -
v_ry ** 2 + v_tz ** 2 + f_doppler ** 2) / 2) * v_rx ** 2 * math.
sin(elevation) ** 2) + v_rx ** 2 * (c * h * (c * v_ry - v_ty) *
math.cos(elevation) ** 3 - (h * (c * v_rz + v_tz) * math.sin
(elevation) + h * f_doppler + delay * v_tz) * c * math.cos
(elevation) ** 2 - c * (delay * math.sin(elevation) + h) * (c *
v_ry - v_ty) * math.cos(elevation) + ((delay * f_doppler + v_tz
* h) * c + v_rz * h) * math.sin(elevation) + c * (delay * v_tz +
h * f_doppler))) / v_rx / (c ** 2 * (v_ty ** 2 - v_tz ** 2) *
math.cos(elevation) ** 4 + 2 * c ** 2 * v_ty * (v_tz * math.sin
(elevation) + f_doppler) * math.cos(elevation) ** 3 + (2 * c *
f_doppler * math.sin(elevation) * v_tz + (-v_rx ** 2 + v_tz ** 2
+ f_doppler ** 2) * c - 2 * v_ry * v_ty) * c * math.cos
(elevation) ** 2 - 2 * c * v_ry * (v_tz * math.sin(elevation) +
f_doppler) * math.cos(elevation) + v_rx ** 2 + v_ry ** 2) /
math.sin(elevation)
```

> *sol_x_2* := *simplify*(*all_x*[2]) :

> *Python*(*sol_x_2*)

```
cg5 = x == ((-math.cos(elevation) ** 2 * c * v_ty - c * (v_tz *
math.sin(elevation) + f_doppler) * math.cos(elevation) + v_ry) *
math.sqrt(-2 * ((c * delay * h * (v_ty - v_tz) * (v_ty + v_tz) *
math.cos(elevation) ** 2 + (((c * v_ty - v_ry) * v_tz - v_rz *
(c * v_ry - v_ty)) * h ** 2 + 2 * c * f_doppler * h * v_ty *
delay + c * v_tz * v_ty * delay ** 2) * math.cos(elevation) +
f_doppler * (c * v_tz + v_rz) * h ** 2 + (v_tz ** 2 * c + v_rz *
v_tz + c * (-v_rx ** 2 - v_ry ** 2 + f_doppler ** 2)) * delay *
h + c * delay ** 2 * v_tz * f_doppler) * c * math.sin(elevation)
- 2 * math.cos(elevation) ** 3 * c ** 2 * delay * h * v_ty *
v_tz - 2 * ((v_tz ** 2 * c / 4 + v_rz * v_tz / 2 - c * v_ty ** 2
/ 4 + v_ry * v_ty / 2 + c * (-v_ry ** 2 / 4 + v_rz ** 2 / 4)) *
h ** 2 + c * delay * h * v_tz * f_doppler - c * delay ** 2 *
(v_ty - v_tz) * (v_ty + v_tz) / 4) * c * math.cos(elevation) **
2 + (f_doppler * (c * v_ty - v_ry) * h ** 2 - (-2 * c * v_tz *
v_ty + v_rz * (c * v_ry - v_ty)) * delay * h + c * delay ** 2 *
v_ty * f_doppler) * c * math.cos(elevation) + (c ** 2 * v_tz **
2 / 2 + c * v_rz * v_tz + c ** 2 * f_doppler ** 2 / 2 + (-v_rx *
* 2 / 2 - v_ry ** 2 / 2) * c ** 2 + v_rx ** 2 / 2 + v_rz ** 2 /
2 + v_ry ** 2 / 2) * h ** 2 + 2 * (c * v_tz + v_rz / 2) *
f_doppler * delay * c * h + c ** 2 * delay ** 2 * (-v_rx ** 2 -
v_ry ** 2 + v_tz ** 2 + f_doppler ** 2) / 2) * v_rx ** 2 * math.
sin(elevation) ** 2) + v_rx ** 2 * (c * h * (c * v_ry - v_ty) *
math.cos(elevation) ** 3 - (h * (c * v_rz + v_tz) * math.sin
(elevation) + h * f_doppler + delay * v_tz) * c * math.cos
(elevation) ** 2 - c * (delay * math.sin(elevation) + h) * (c *
v_ry - v_ty) * math.cos(elevation) + ((delay * f_doppler + v_tz
* h) * c + v_rz * h) * math.sin(elevation) + c * (delay * v_tz +
h * f_doppler))) / v_rx / (c ** 2 * (v_ty ** 2 - v_tz ** 2) *
math.cos(elevation) ** 4 + 2 * c ** 2 * v_ty * (v_tz * math.sin
(elevation) + f_doppler) * math.cos(elevation) ** 3 + (2 * c *
f_doppler * math.sin(elevation) * v_tz + (-v_rx ** 2 + v_tz ** 2
+ f_doppler ** 2) * c - 2 * v_ry * v_ty) * c * math.cos
(elevation) ** 2 - 2 * c * v_ry * (v_tz * math.sin(elevation) +
f_doppler) * math.cos(elevation) + v_rx ** 2 + v_ry ** 2) /
math.sin(elevation)
```

```
> all_y := allvalues(sol[1]) :
```

```
> sol_y_1 := simplify(all_y[1]) :
```

```
> Python(sol_y_1)
cg6 = x == ((math.cos(elevation) ** 2 * c * v_ty + c * (v_tz *
math.sin(elevation) + f_doppler) * math.cos(elevation) - v_ry) *
math.sqrt(-2 * ((c * delay * h * (v_ty - v_tz) * (v_ty + v_tz) *
math.cos(elevation) ** 2 + (((c * v_ty - v_ry) * v_tz - v_rz *
(c * v_ry - v_ty)) * h ** 2 + 2 * c * f_doppler * h * v_ty *
delay + c * v_tz * v_ty * delay ** 2) * math.cos(elevation) +
f_doppler * (c * v_tz + v_rz) * h ** 2 + (v_tz ** 2 * c + v_rz *
v_tz + c * (-v_rx ** 2 - v_ry ** 2 + f_doppler ** 2)) * delay *
h + c * delay ** 2 * v_tz * f_doppler) * c * math.sin(elevation)
- 2 * math.cos(elevation) ** 3 * c ** 2 * delay * h * v_ty *
v_tz - 2 * ((v_tz ** 2 * c / 4 + v_rz * v_tz / 2 - c * v_ty ** 2
/ 4 + v_ry * v_ty / 2 + c * (-v_ry ** 2 / 4 + v_rz ** 2 / 4)) *
h ** 2 + c * delay * h * v_tz * f_doppler - c * delay ** 2 *
(v_ty - v_tz) * (v_ty + v_tz) / 4) * c * math.cos(elevation) **
2 + (f_doppler * (c * v_ty - v_ry) * h ** 2 - (-2 * c * v_tz *
v_ty + v_rz * (c * v_ry - v_ty)) * delay * h + c * delay ** 2 *
v_ty * f_doppler) * c * math.cos(elevation) + (c ** 2 * v_tz **
2 / 2 + c * v_rz * v_tz + c ** 2 * f_doppler ** 2 / 2 + (-v_rx *
* 2 / 2 - v_ry ** 2 / 2) * c ** 2 + v_rx ** 2 / 2 + v_rz ** 2 /
2 + v_ry ** 2 / 2) * h ** 2 + 2 * (c * v_tz + v_rz / 2) *
f_doppler * delay * c * h + c ** 2 * delay ** 2 * (-v_rx ** 2 -
v_ry ** 2 + v_tz ** 2 + f_doppler ** 2) / 2) * v_rx ** 2 * math.
sin(elevation) ** 2) + v_rx ** 2 * (c * h * (c * v_ry - v_ty) *
math.cos(elevation) ** 3 - (h * (c * v_rz + v_tz) * math.sin
(elevation) + h * f_doppler + delay * v_tz) * c * math.cos
(elevation) ** 2 - c * (delay * math.sin(elevation) + h) * (c *
v_ry - v_ty) * math.cos(elevation) + ((delay * f_doppler + h *
v_tz) * c + v_rz * h) * math.sin(elevation) + c * (delay * v_tz
+ h * f_doppler))) / v_rx / (c ** 2 * (v_ty ** 2 - v_tz ** 2) *
math.cos(elevation) ** 4 + 2 * c ** 2 * v_ty * (v_tz * math.sin
(elevation) + f_doppler) * math.cos(elevation) ** 3 + (2 * c *
f_doppler * math.sin(elevation) * v_tz + (-v_rx ** 2 + v_tz ** 2
+ f_doppler ** 2) * c - 2 * v_ry * v_ty) * c * math.cos
(elevation) ** 2 - 2 * c * v_ry * (v_tz * math.sin(elevation) +
f_doppler) * math.cos(elevation) + v_rx ** 2 + v_ry ** 2) /
math.sin(elevation)
```

```
> sol_y_2 := simplify(all_y[2]):
```

```
> Python(sol_y_2)
cg7 = x == ((-math.cos(elevation) ** 2 * c * v_ty - c * (v_tz *
math.sin(elevation) + f_doppler) * math.cos(elevation) + v_ry) *
math.sqrt(-2 * ((c * delay * h * (v_ty - v_tz) * (v_ty + v_tz) *
math.cos(elevation) ** 2 + (((c * v_ty - v_ry) * v_tz - v_rz *
(c * v_ry - v_ty)) * h ** 2 + 2 * c * f_doppler * h * v_ty *
delay + c * v_tz * v_ty * delay ** 2) * math.cos(elevation) +
f_doppler * (c * v_tz + v_rz) * h ** 2 + (v_tz ** 2 * c + v_rz *
v_tz + c * (-v_rx ** 2 - v_ry ** 2 + f_doppler ** 2)) * delay *
h + c * delay ** 2 * v_tz * f_doppler) * c * math.sin(elevation)
- 2 * math.cos(elevation) ** 3 * c ** 2 * delay * h * v_ty *
v_tz - 2 * ((v_tz ** 2 * c / 4 + v_rz * v_tz / 2 - c * v_ty ** 2
/ 4 + v_ry * v_ty / 2 + c * (-v_ry ** 2 / 4 + v_rz ** 2 / 4)) *
h ** 2 + c * delay * h * v_tz * f_doppler - c * delay ** 2 *
(v_ty - v_tz) * (v_ty + v_tz) / 4) * c * math.cos(elevation) **
2 + (f_doppler * (c * v_ty - v_ry) * h ** 2 - (-2 * c * v_tz *
v_ty + v_rz * (c * v_ry - v_ty)) * delay * h + c * delay ** 2 *
v_ty * f_doppler) * c * math.cos(elevation) + (c ** 2 * v_tz **
2 / 2 + c * v_rz * v_tz + c ** 2 * f_doppler ** 2 / 2 + (-v_rx *
* 2 / 2 - v_ry ** 2 / 2) * c ** 2 + v_rx ** 2 / 2 + v_rz ** 2 /
2 + v_ry ** 2 / 2) * h ** 2 + 2 * (c * v_tz + v_rz / 2) *
f_doppler * delay * c * h + c ** 2 * delay ** 2 * (-v_rx ** 2 -
v_ry ** 2 + v_tz ** 2 + f_doppler ** 2) / 2) * v_rx ** 2 * math.
sin(elevation) ** 2) + v_rx ** 2 * (c * h * (c * v_ry - v_ty) *
math.cos(elevation) ** 3 - (h * (c * v_rz + v_tz) * math.sin
(elevation) + h * f_doppler + delay * v_tz) * c * math.cos
(elevation) ** 2 - c * (delay * math.sin(elevation) + h) * (c *
v_ry - v_ty) * math.cos(elevation) + ((delay * f_doppler + h *
v_tz) * c + v_rz * h) * math.sin(elevation) + c * (delay * v_tz
+ h * f_doppler))) / v_rx / (c ** 2 * (v_ty ** 2 - v_tz ** 2) *
math.cos(elevation) ** 4 + 2 * c ** 2 * v_ty * (v_tz * math.sin
(elevation) + f_doppler) * math.cos(elevation) ** 3 + (2 * c *
f_doppler * math.sin(elevation) * v_tz + (-v_rx ** 2 + v_tz ** 2
+ f_doppler ** 2) * c - 2 * v_ry * v_ty) * c * math.cos
(elevation) ** 2 - 2 * c * v_ry * (v_tz * math.sin(elevation) +
f_doppler) * math.cos(elevation) + v_rx ** 2 + v_ry ** 2) /
math.sin(elevation)
```

```
>
```