Modeling and Symbolic Computation

Master 1 International in Computer Science - Introduction to Research 1
Introduction to Robotics : Wire-driven parallel robot

Yves Papegay - INRIA / COPRIN - January 2013, Monday 14.

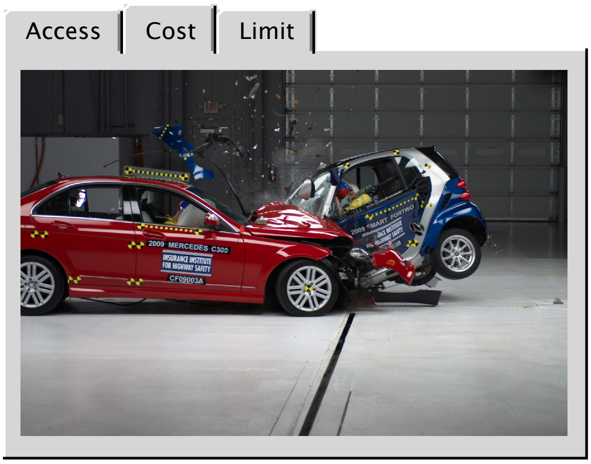
Modeling a (robotic) system is the first and mandatory step to achieve whenever one is interested in the mechanical or dynamical behaviours of such system.
Defining the pertinent physical entities, selecting a formalism to attach variables to them, then writing equations expressing the physical laws produce a first kind of symbolic model.
It then has to be made explicit using numerical schemes or symbolic solvers to become useful for simulation and visualization of results.
All these activities will be explained and demonstrated in the particular and practical case of a 3RPR parallel robot.

Modeling and Simulation Process

Goal of this process is **understanding** and **predicting** some behaviors of a physical system.

▪ Needs

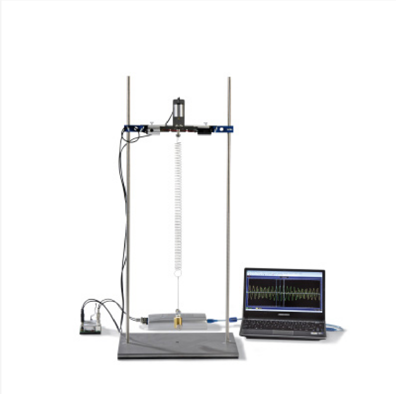
Modeling and simulation process is particularly useful for **replacing experiments**, namely whenever experimental conditions are difficult to obtain, very costly, ore not yet existing.

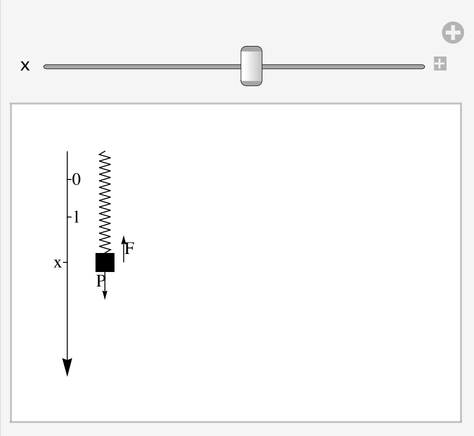


▪ Outline

physical system
↓ engineering
symbolic implicit model
↓ mathematical methods
explicit model
↓ art of programming
simulation code

■ Exercice : Trivial exemple





■ Step 1: From Physical System to Symbolic Implicit Model

Out[54]=

▼ Which behavior am I interested in ?

Motion of the body.

▼ Select a theoretical framework and correspondant formalism...

between several possibilities:

- classical mechanics, cartesian coordinates, forces
- lagrangian mecanics, generalized coordinates, energies

▼ What are the physical quantities involved ?

Length at rest
Spring stiffness
Damping factor
Initial position
Initial speed
Mass
External force
Position
Speed
Weight
Global Force

▼ How to represent them ?

- *Define an inertial reference frame.*
- *Attach a variable to each of the quanities:*

l	*Length at rest*
k	*Spring stiffness*
b	*Damping factor*
x_0	*Initial position*
v_0	*Initial speed*
m	*Mass*
F_{mot}	*External force*
x	*Position*
v	*Speed*
P	*Weight*
F	*Global Force*

▼ How are these quantities linked together ?

Express Newton's second law of mechanics:

$$F == m.\frac{dv}{dt}$$

■ Step 2: Getting an explicit Model

- Defining inputs and outputs

IO	Symbol	Description	Unit	Default	Mode
I	l	Length at rest	m	0.	C
I	k	stiffness	$N.m^{-1}$	5.	C
I	b	damping	$N.m^{-1}.s$	0.1	C
I	x_0	Initial position	m	0.	S
I	v_0	Initial speed	$m.s^{-1}$	False	S
I	m	Mass	kg	1.	S
I	F_{mot}	Excitation	N	0.	D
O	x	Position	m	□	D
O	v	Speed	$m.s^{-1}$	□	D
O	F	Force	N	□	D

- Defining a model to compute outputs in terms of inputs

$$g =_c 9.81$$

Forces applied on mass : weight, excitation, spring reaction, damping force

$$F =_d P + F_{mot} + F_{spring} + F_{damp}$$

$$P =_d m\, g$$

$$F_{spring} =_d -k\,(x - l)$$

$$F_{damp} =_d -b\, v$$

The application of fundamental law (F=m.γ) leads to a second order differential equation that will be integrated, with an Euler numeric scheme:

$$\{x, v\} =_d \text{Scheme}[\text{NumIter}[\,]]$$

$$x =_i {}^{-\Delta t}x + \Delta t\, {}^{-\Delta t}v$$

$$v =_i {}^{-\Delta t}v + (\Delta t / m)\, {}^{-\Delta t}F$$

$$\text{EndScheme}[\,]$$

$$\{x, v\} =_s \text{Scheme}[\text{NumIter}[\,]]$$

$$x =_i x_0$$

$$v =_i v_0$$

$$\text{EndScheme}[\,]$$

■ Step 3: Implementing a simulation code

- Defining inputs

```
m = 1.;
k = 10.;
l = 1.;
b = 0.2;
x0 = 1.;
v0 = 0.1;
Fmot[t_] := 5 Sin[3 √(k / m) t]
```

- Defining constants

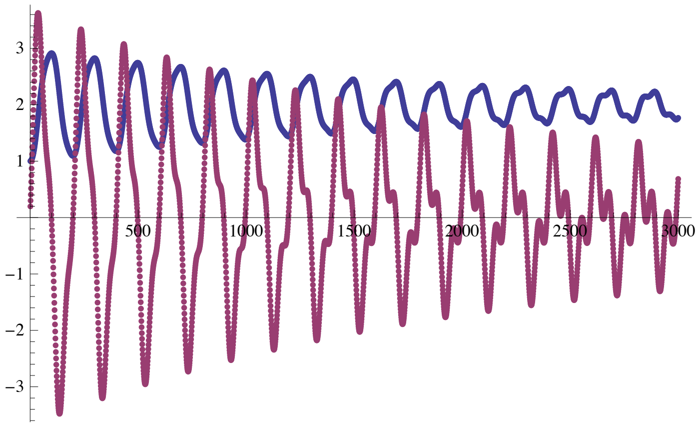
```
g = 9.81`;
Δt = 0.01;
```

- Defining formulas

```
F[n_] := P[n] + Fmot[n Δt] + Fspring[n] + Fdamp[n]
P[n_] := m g
Fspring[n_] := -k (x[n] - l)
Fdamp[n_] := -b v[n]
```

- Iterative numerical scheme

```
NumIter[0] = Module[{v, x}, x = x0; v = v0; {x, v}];
NumIter[n_] := NumIter[n] = Module[{Localv, Localx},
    Localx = x[n - 1] + Δt v[n - 1]; Localv = v[n - 1] + Δt F[n - 1] / m; {Localx, Localv}]
x[n_] := NumIter[n][[1]]
v[n_] := NumIter[n][[2]]
Table[NumIter[i], {i, 1, 10000}];
ListPlot[{Table[x[i], {i, 1, 3000}], Table[v[i], {i, 1, 3000}]}]
```

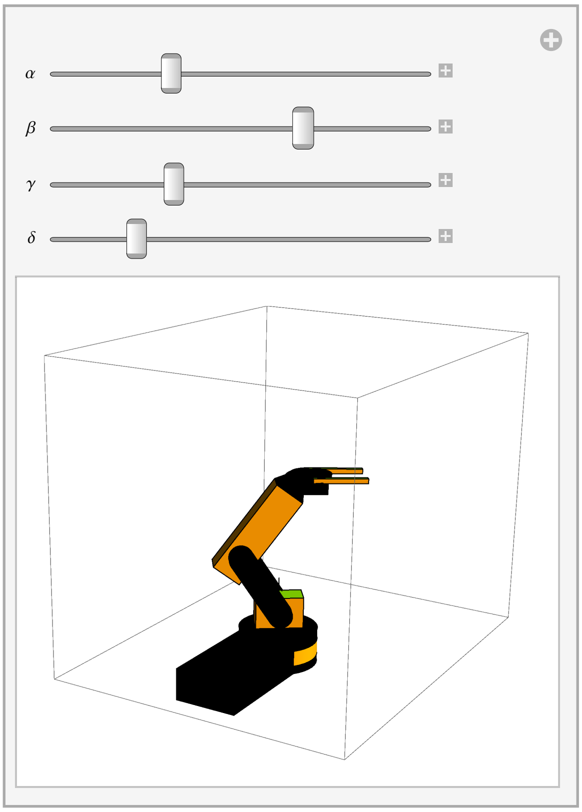


- **Step 3bis: Implementing visualisation code**

Geometrical Models of Robots

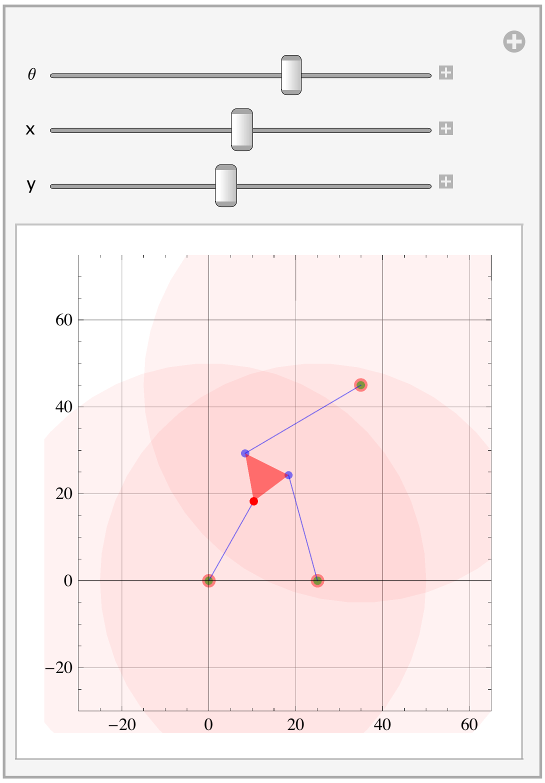
Serial Robot

Out[62]=



Parallel Robot

Out[70]=

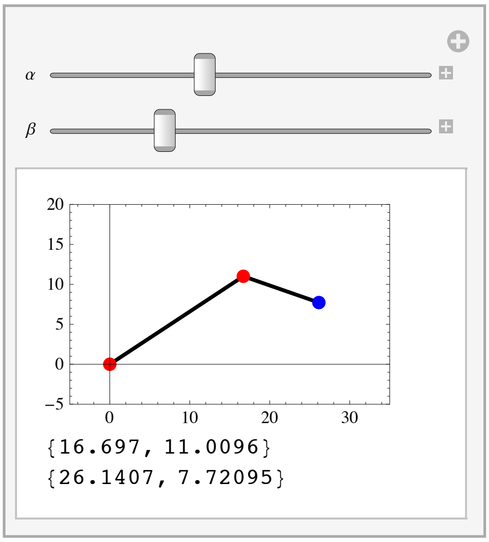


Direct (forward) and Inverse Models

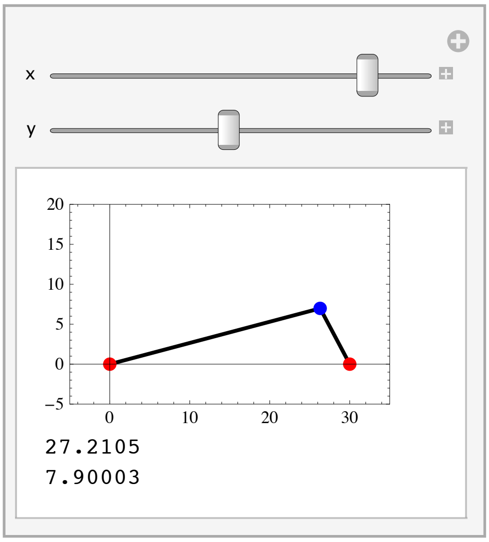
- Find the position and orientation of the end effector in term of the coordinates of the robot.
- Find the coordinates of the robot in term of the position and orientation of the end effector.

Exercice : Trivial exemple

- Serial case



- Parallel case



Geometrical Model of a 3RPR Parallel Robot

Inverse Model

Find the lengthes of the bars in term of the position and orientation of the end effector.

Step 1

- Define a fixed reference frame $R: (O, i, j)$ and a mobile reference frame $R_B: (O_B, i_B, j_B)$
- Attach points (A_1, A_2, A_3) and (B_1, B_2, B_3) to the fixed and mobile ends of each bar
- Define coordinates (a_1, a_2, a_3) of points (A_1, A_2, A_3) in the fixed frame and coordinates (b_1, b_2, b_3) of points (B_1, B_2, B_3) in the mobile frame
- Express position and orientation of R_B with respect to R with the help of angle $\theta = \langle \text{iOi}_B \rangle$ and coordinates of vector $\text{OO}_B: (x, y)$

```
R2RB[v_] := P + RotationMatrix[θ].v
```

- Express each vector $A_i B_i$ in terms of a_i, b_i, θ, x and y

```
AiBi = (R2RB[bi] - ai)
```

```
-ai + P + {{Cos[θ], -Sin[θ]}, {Sin[θ], Cos[θ]}}.bi
```

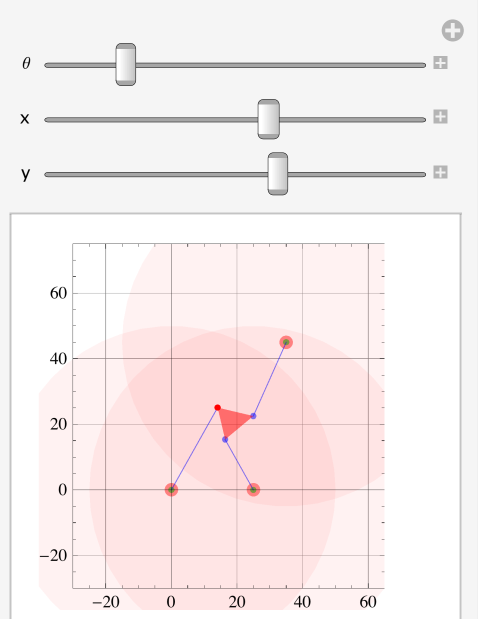
- Express length of each bar (l_i) in terms of a_i, b_i, θ, x and y

```
li^2 == (#[[1]]^2 + #[[2]]^2) &[AiBi /. {ai → {xai, yai}, P → {x, y}, bi → {xbi, ybi}}]
```

$$\text{li}^2 == (\text{y} - \text{yai} + \text{ybi}\,\text{Cos}[\theta] + \text{xbi}\,\text{Sin}[\theta])^2 + (\text{x} - \text{xai} + \text{xbi}\,\text{Cos}[\theta] - \text{ybi}\,\text{Sin}[\theta])^2$$

In[71]:=
```
Manipulate[Column[{Draw3RPR[A, B, {{θ Degree, {x, y}}}], (0.` + x)^2 + (0.` + y)^2,
    (-25.` + x + 10.` Cos[θ Degree])^2 + (0.` + y + 10.` Sin[θ Degree])^2,
    (-35.` + x + 5.` Cos[θ Degree] - 10.` Sin[θ Degree])^2 +
     (-45.` + y + 10.` Cos[θ Degree] + 5.` Sin[θ Degree])^2}],
 {θ, -120, 120}, {x, -10, 30}, {y, 0, 40}]
```

Out[71]=




```
828.82
310.303
605.997
```

```
NumVal = {θ → N[Pi / 4], x → 10., y → 25., l1^2 -> 725.`, l2^2 -> 1091.4213562373097`,
    l3^2 -> 902.5126265847084`, xa1 → 0., ya1 → 0., xa2 → 25., ya2 → 0., xa3 → 35.,
    ya3 → 45., xb1 → 0., yb1 → 0., xb2 → 10., yb2 → 0., xb3 → 5., yb3 → 10.};
```

▪ Direct Model

Find the position and orientation of the end effector in term of the lengthes of the bars.

```
AiBi /. {ai → {xa1, ya1}, P → {x, y}, bi → {xb1, yb1}} // MatrixForm
```

$$\begin{pmatrix} x - xa1 + xb1 \operatorname{Cos}[\theta] - yb1 \operatorname{Sin}[\theta] \\ y - ya1 + yb1 \operatorname{Cos}[\theta] + xb1 \operatorname{Sin}[\theta] \end{pmatrix}$$

```
(#[[1]]^2 + #[[2]]^2) &[AiBi /. {ai → {xa1, ya1}, P → {x, y}, bi → {xb1, yb1}}]
```

$(y - ya1 + yb1 \operatorname{Cos}[\theta] + xb1 \operatorname{Sin}[\theta])^2 + (x - xa1 + xb1 \operatorname{Cos}[\theta] - yb1 \operatorname{Sin}[\theta])^2$

```
eq1 = l1^2 - (#[[1]]^2 + #[[2]]^2) &[AiBi /. {ai → {xa1, ya1}, P → {x, y}, bi → {xb1, yb1}}]
eq2 = l2^2 - (#[[1]]^2 + #[[2]]^2) &[AiBi /. {ai → {xa2, ya2}, P → {x, y}, bi → {xb2, yb2}}]
eq3 = l3^2 - (#[[1]]^2 + #[[2]]^2) &[AiBi /. {ai → {xa3, ya3}, P → {x, y}, bi → {xb3, yb3}}]
```

$l1^2 - (y - ya1 + yb1 \operatorname{Cos}[\theta] + xb1 \operatorname{Sin}[\theta])^2 - (x - xa1 + xb1 \operatorname{Cos}[\theta] - yb1 \operatorname{Sin}[\theta])^2$

$l2^2 - (y - ya2 + yb2 \operatorname{Cos}[\theta] + xb2 \operatorname{Sin}[\theta])^2 - (x - xa2 + xb2 \operatorname{Cos}[\theta] - yb2 \operatorname{Sin}[\theta])^2$

$l3^2 - (y - ya3 + yb3 \operatorname{Cos}[\theta] + xb3 \operatorname{Sin}[\theta])^2 - (x - xa3 + xb3 \operatorname{Cos}[\theta] - yb3 \operatorname{Sin}[\theta])^2$

```
Solve[{eq1 == 0, eq2 == 0, eq3 == 0}, {x, y, θ}]
```

```
$Aborted
```

```
eq12 = Collect[Simplify[Expand[eq2 - eq1]], {x, y}]
```

```
- l1^2 + l2^2 + xa1^2 - xa2^2 + xb1^2 - xb2^2 + ya1^2 - ya2^2 + yb1^2 - yb2^2 -
 2 xa1 xb1 Cos[θ] + 2 xa2 xb2 Cos[θ] - 2 ya1 yb1 Cos[θ] + 2 ya2 yb2 Cos[θ] -
 2 xb1 ya1 Sin[θ] + 2 xb2 ya2 Sin[θ] + 2 xa1 yb1 Sin[θ] - 2 xa2 yb2 Sin[θ] +
 y (-2 ya1 + 2 ya2 + 2 yb1 Cos[θ] - 2 yb2 Cos[θ] + 2 xb1 Sin[θ] - 2 xb2 Sin[θ]) +
 x (-2 xa1 + 2 xa2 + 2 (xb1 - xb2) Cos[θ] - 2 yb1 Sin[θ] + 2 yb2 Sin[θ])
```

```
eq13 = Collect[Simplify[Expand[eq3 - eq1]], {x, y}]
```

```
- l1^2 + l3^2 + xa1^2 - xa3^2 + xb1^2 - xb3^2 + ya1^2 - ya3^2 + yb1^2 - yb3^2 -
 2 xa1 xb1 Cos[θ] + 2 xa3 xb3 Cos[θ] - 2 ya1 yb1 Cos[θ] + 2 ya3 yb3 Cos[θ] -
 2 xb1 ya1 Sin[θ] + 2 xb3 ya3 Sin[θ] + 2 xa1 yb1 Sin[θ] - 2 xa3 yb3 Sin[θ] +
 y (-2 ya1 + 2 ya3 + 2 yb1 Cos[θ] - 2 yb3 Cos[θ] + 2 xb1 Sin[θ] - 2 xb3 Sin[θ]) +
 x (-2 xa1 + 2 xa3 + 2 (xb1 - xb3) Cos[θ] - 2 yb1 Sin[θ] + 2 yb3 Sin[θ])
```

```
resxy = Solve[{eq12 == 0, eq13 == 0}, {x, y}]
```

```
{{x → - (- (-2 ya1 + 2 ya3 + 2 yb1 Cos[θ] - 2 yb3 Cos[θ] + 2 xb1 Sin[θ] - 2 xb3 Sin[θ])
          (- l1^2 + l2^2 + xa1^2 - xa2^2 + xb1^2 - xb2^2 + ya1^2 - ya2^2 + yb1^2 - yb2^2 -
            2 xa1 xb1 Cos[θ] + 2 xa2 xb2 Cos[θ] - 2 ya1 yb1 Cos[θ] + 2 ya2 yb2 Cos[θ] -
            2 xb1 ya1 Sin[θ] + 2 xb2 ya2 Sin[θ] + 2 xa1 yb1 Sin[θ] - 2 xa2 yb2 Sin[θ]) +
         (-2 ya1 + 2 ya2 + 2 yb1 Cos[θ] - 2 yb2 Cos[θ] + 2 xb1 Sin[θ] - 2 xb2 Sin[θ])
          (- l1^2 + l3^2 + xa1^2 - xa3^2 + xb1^2 - xb3^2 + ya1^2 - ya3^2 + yb1^2 - yb3^2 -
            2 xa1 xb1 Cos[θ] + 2 xa3 xb3 Cos[θ] - 2 ya1 yb1 Cos[θ] + 2 ya3 yb3 Cos[θ] -
            2 xb1 ya1 Sin[θ] + 2 xb3 ya3 Sin[θ] + 2 xa1 yb1 Sin[θ] - 2 xa3 yb3 Sin[θ])) /
      (- (-2 ya1 + 2 ya3 + 2 yb1 Cos[θ] - 2 yb3 Cos[θ] + 2 xb1 Sin[θ] - 2 xb3 Sin[θ])
          (-2 xa1 + 2 xa2 + 2 (xb1 - xb2) Cos[θ] - 2 yb1 Sin[θ] + 2 yb2 Sin[θ]) +
         (-2 ya1 + 2 ya2 + 2 yb1 Cos[θ] - 2 yb2 Cos[θ] + 2 xb1 Sin[θ] - 2 xb2 Sin[θ])
          (-2 xa1 + 2 xa3 + 2 (xb1 - xb3) Cos[θ] - 2 yb1 Sin[θ] + 2 yb3 Sin[θ])),
  y → - (l1^2 - l2^2 - xa1^2 + xa2^2 - xb1^2 + xb2^2 - ya1^2 + ya2^2 - yb1^2 + yb2^2 + 2 xa1 xb1 Cos[θ] -
          2 xa2 xb2 Cos[θ] + 2 ya1 yb1 Cos[θ] - 2 ya2 yb2 Cos[θ] + 2 xb1 ya1 Sin[θ] -
          2 xb2 ya2 Sin[θ] - 2 xa1 yb1 Sin[θ] + 2 xa2 yb2 Sin[θ]) /
       (2 (ya1 - ya2 - yb1 Cos[θ] + yb2 Cos[θ] - xb1 Sin[θ] + xb2 Sin[θ])) +
     ((-2 xa1 + 2 xa2 + 2 (xb1 - xb2) Cos[θ] - 2 yb1 Sin[θ] + 2 yb2 Sin[θ])
        (- (-2 ya1 + 2 ya3 + 2 yb1 Cos[θ] - 2 yb3 Cos[θ] + 2 xb1 Sin[θ] - 2 xb3 Sin[θ])
            (- l1^2 + l2^2 + xa1^2 - xa2^2 + xb1^2 - xb2^2 + ya1^2 - ya2^2 + yb1^2 - yb2^2 -
              2 xa1 xb1 Cos[θ] + 2 xa2 xb2 Cos[θ] - 2 ya1 yb1 Cos[θ] + 2 ya2 yb2 Cos[θ] -
              2 xb1 ya1 Sin[θ] + 2 xb2 ya2 Sin[θ] + 2 xa1 yb1 Sin[θ] - 2 xa2 yb2 Sin[θ]) +
           (-2 ya1 + 2 ya2 + 2 yb1 Cos[θ] - 2 yb2 Cos[θ] + 2 xb1 Sin[θ] - 2 xb2 Sin[θ])
            (- l1^2 + l3^2 + xa1^2 - xa3^2 + xb1^2 - xb3^2 + ya1^2 - ya3^2 + yb1^2 - yb3^2 -
              2 xa1 xb1 Cos[θ] + 2 xa3 xb3 Cos[θ] - 2 ya1 yb1 Cos[θ] + 2 ya3 yb3 Cos[θ] -
              2 xb1 ya1 Sin[θ] + 2 xb3 ya3 Sin[θ] + 2 xa1 yb1 Sin[θ] - 2 xa3 yb3 Sin[θ]))) /
      ((-2 ya1 + 2 ya2 + 2 yb1 Cos[θ] - 2 yb2 Cos[θ] + 2 xb1 Sin[θ] - 2 xb2 Sin[θ])
        (- (-2 ya1 + 2 ya3 + 2 yb1 Cos[θ] - 2 yb3 Cos[θ] + 2 xb1 Sin[θ] - 2 xb3 Sin[θ])
            (-2 xa1 + 2 xa2 + 2 (xb1 - xb2) Cos[θ] - 2 yb1 Sin[θ] + 2 yb2 Sin[θ]) +
           (-2 ya1 + 2 ya2 + 2 yb1 Cos[θ] - 2 yb2 Cos[θ] + 2 xb1 Sin[θ] - 2 xb2 Sin[θ])
            (-2 xa1 + 2 xa3 + 2 (xb1 - xb3) Cos[θ] - 2 yb1 Sin[θ] + 2 yb3 Sin[θ])))}}
```

```
WeierStrass = {Cos[θ] → (1 - t^2)/(1 + t^2), Sin[θ] → (2 t)/(1 + t^2)};
(Solve[{Cos[θ] == (Cos[θ] /. WeierStrass), Sin[θ] == (Sin[θ] /. WeierStrass)}, t] /.
   NumVal)[[1]]
```

```
{t → 0.414214}
```

```
respol = Collect[Numerator[Together[eq1 /. resxy /. WeierStrass][[1]]], t]
```

A very large output was generated. Here is a sample of it:

```
-l2^4 xa1^2 + <<9340>> + t^2 (<<1>> + <<9810>>) +
 t^6 (-l2^4 xa1^2 + 2 l2^2 l3^2 xa1^2 - l3^4 xa1^2 + 2 l1^2 l2^2 xa1 xa2 - 2 l1^2 l3^2 xa1 xa2 -
     2 l2^2 l3^2 xa1 xa2 + 2 l3^4 xa1 xa2 - 2 l2^2 xa1^3 xa2 + 2 l3^2 xa1^3 xa2 - l1^4 xa2^2 +
     2 l1^2 l3^2 xa2^2 - l3^4 xa2^2 + 2 l1^2 xa1^2 xa2^2 + 2 l2^2 xa1^2 xa2^2 - 4 l3^2 xa1^2 xa2^2 - xa1^4 xa2^2 -
     2 l1^2 xa1 xa2^3 + <<9404>> + 2 xa2 xb1 yb3^4 - xb1^2 yb3^4 + 2 xa1 xb2 yb3^4 - 2 xa2 xb2 yb3^4 +
     2 xb1 xb2 yb3^4 - xb2^2 yb3^4 - ya1^2 yb3^4 + 2 ya1 ya2 yb3^4 - ya2^2 yb3^4 - 2 ya1 yb1 yb3^4 +
     2 ya2 yb1 yb3^4 - yb1^2 yb3^4 + 2 ya1 yb2 yb3^4 - 2 ya2 yb2 yb3^4 + 2 yb1 yb2 yb3^4 - yb2^2 yb3^4)
```

Show Less | Show More | Show Full Output | Set Size Limit...

```
Exponent[respol, t]
```

6

```
NSolve[respol /. v_^4 :> (v^2 /. NumVal)^2 /. NumVal, t]
```

```
{{t -> -0.558302 - 0.740072 i}, {t -> -0.558302 + 0.740072 i}, {t -> -0.0707678 - 0.607254 i},
 {t -> -0.0707678 + 0.607254 i}, {t -> 0.414214}, {t -> 0.417393}}
```