Particles

A particle is an object of class `particle`, which is a subclass of `point`. Essentially, `particle` adds a mass property to `point`, and methods to compute useful quantities such as the linear and angular momentum or the kinetic energy. This tutorial focuses on the specific aspects of `particle`.

Start by importing the Anakin framework into Matlab. Be sure to include Anakin's base directory before you run this line:

```
import anakin.*
```

Object creation

The default particle has mass 1 and is located at the canonical origin:

```
P = particle% default call
```

```
P = 
Particle with mass:
     1

canonical position vector components:
     0
     0
     0
```

A particle can be created by passing their mass and its position vector or its canonical coordinates:

```
m = 20;
rA = tensor([1;2;3])
```

```
rA = 
Vector with canonical components:
     1
     2
     3
```

```
P = particle(m,rA) % as a vector
```

```
P = 
Particle with mass:
    20

canonical position vector components:
     1
     2
     3
```

```
P = particle(20,[4;5;6]) % as a column array
```

```
P =
Particle with mass:
    20

canonical position vector components:
     4
     5
     6
```

```
P = particle(30,[7,8,9]) % as a row array
```

```
P =
Particle with mass:
    30

canonical position vector components:
     7
     8
     9
```

The components that define the position of the particle can be given in another reference frame. Internally, the point object stores only its coordinates in the canonical reference frame:

```
O1 = point([1;2;3]);
B1 = basis([1;0;0],[0;cos(pi/6);sin(pi/6)],[0;-sin(pi/6);cos(pi/6)]);
S1 = frame(O1,B1);

P = particle(30,[1,1,1],S1)
```

```
P =
Particle with mass:
    30

canonical position vector components:
    2.0000
    2.3660
    4.3660
```

Basic functionality

The linear, angular momentum and the kinetic energy of the particle in a given reference frame are:

```
O1 = point([1;2;3]);
S1 = frame(O1);
P = particle(1,[1,1,1],S1);

% linear momentum
P.p % use default reference frame: canonical frame
```

```
ans =
Vector with canonical components:
     0
     0
```

```
0
```

```
P.p(S1)
```

```
ans =
Vector with canonical components:
     0
     0
     0
```

```
% angular momentum
P.H(O1) % the point about which the angular momentum is computed
```

```
ans =
Vector with canonical components:
     0
     0
     0
```

```
P.H(O1,S1)
```

```
ans =
Vector with canonical components:
     0
     0
     0
```

```
% kinetic energy
P.T
```

```
ans =
Scalar with value:
     0
```

```
P.T(S1)
```

```
ans =
Scalar with value:
     0
```

The tensor of inertia of the particle about a point can be easily computed as:

```
inertia = P.inertia
```

```
inertia =
Second-order tensor with canonical components:
   25.0000   -6.0000   -8.0000
   -6.0000   20.0000  -12.0000
   -8.0000  -12.0000   13.0000
```

Symbolic particles

Particles with symbolic position or mass have additional functionality (running this requires the Symbolic Math Toolbox installed):

```
syms t xi(t) eta(t) zeta(t) phi(t); % declare symbolic variables
assume([in(t, 'real'), in(xi(t), 'real'), in(eta(t), 'real'), in(zeta(t), 'real'), in(phi(t),

P = particle(2,[xi,eta,zeta]) % a symbolic point
```

```
P = 
Particle with mass:
     2

canonical position vector components:
   xi(t)
  eta(t)
 zeta(t)
```

To particularize a symbolic particle at particular values of its parameters, one may use `subs`, by specifying them in a list:

```
P.subs({xi},{6}) % replace xi with 6
```

```
ans = 
Particle with mass:
     2

canonical position vector components:
       6
  eta(t)
 zeta(t)
```