```
> restart;
```

Generate the ODEs for the 3D spherical double pendulum.

```
> with(linalg) :
>
> x[1] := L[1]·sin(theta[1](t))·cos(phi[1](t));
> y[1] := L[1]·sin(theta[1](t))·sin(phi[1](t));
> z[1] := -L[1]·cos(theta[1](t));
> simplify(x[1]^2 + y[1]^2 + z[1]^2);
>
> x[2] := x[1] + L[2]·sin(theta[2](t))·cos(phi[2](t));
> y[2] := y[1] + L[2]·sin(theta[2](t))·sin(phi[2](t));
> z[2] := z[1] - L[2]·cos(theta[2](t));
> simplify(x[2]^2 + y[2]^2 + z[2]^2);
> PE := g·(m[1]·z[1] + m[2]·z[2]);
```

$$> KE[1] := \frac{m[1]}{2} \cdot simplify\left(diff(x[1], t)^2 + diff(y[1], t)^2 + diff(z[1], t)^2\right);$$

$$> KE[2] := \frac{m[2]}{2} \cdot simplify\left(diff(x[2], t)^2 + diff(y[2], t)^2 + diff(z[2], t)^2\right);$$

```
>
> KE[1];
> KE[2];
```

$$> KEs[1] := \frac{m_1 L_1^2 \left(\sin\left(\theta_1(t)\right)^2 \left(\frac{d}{dt}\phi_1(t)\right)^2 + \left(\frac{d}{dt}\theta_1(t)\right)^2\right)}{2}$$

```
> simplify(KE[1] - KEs[1]);
>
> Lag := simplify(KEs[1] + KE[2] - PE);
>
```

$$> Lags := \frac{L_1^2 \sin\left(\theta_1(t)\right)^2 \left(m_1 + m_2\right) \left(\frac{d}{dt}\phi_1(t)\right)^2}{2} + L_1 \sin\left(\theta_1(t)\right) m_2 L_2 \left(\cos\left(\theta_2(t)\right) \sin\left(\phi_2(t) - \phi_1(t)\right) \left(\frac{d}{dt}\theta_2(t)\right) + \sin\left(\theta_2(t)\right) \left(\frac{d}{dt}\phi_2(t)\right) \cos\left(\phi_2(t) - \phi_1(t)\right)\right) \left(\frac{d}{dt}\phi_1(t)\right) + \frac{L_1^2 \left(m_1 + m_2\right) \left(\frac{d}{dt}\theta_1(t)\right)^2}{2} + L_1 \left(\left(\cos\left(\theta_2(t)\right) \cos\left(\phi_2(t) - \phi_1(t)\right) \cos\left(\theta_1(t)\right) + \sin\left(\theta_1(t)\right) \sin\left(\theta_2(t)\right)\right) \left(\frac{d}{dt}\theta_2(t)\right) - \cos\left(\theta_1(t)\right) \sin\left(\theta_2(t)\right) \left(\frac{d}{dt}\phi_2(t)\right) \sin\left(\phi_2(t) - \phi_1(t)\right)\right) m_2 L_2 \left(\frac{d}{dt}\theta_1(t)\right)$$

$$+\frac{\left(\frac{d}{dt}\theta_2(t)\right)^2 L_2^2 m_2}{2}+\frac{\left(-\cos(\theta_2(t))^2 L_2^2 m_2+L_2^2 m_2\right)\left(\frac{d}{dt}\phi_2(t)\right)^2}{2}+g\left(L_1\left(m_1+m_2\right)\cos(\theta_1(t))+\cos(\theta_2(t))L_2 m_2\right);$$

```
> simplify(Lag − Lags);
>
```

Do the Euler-Lagrange calculations.

```
> Lagm := subs({diff(theta[1](t), t) = theta1p}, Lags) :
> Lagm := subs({diff(theta[2](t), t) = theta2p}, Lagm) :
> Lagm := subs({diff(phi[1](t), t) = phi1p}, Lagm) :
> Lagm := subs({diff(phi[2](t), t) = phi2p}, Lagm) :
> Lagm := subs({theta[1](t) = theta[1]}, Lagm) :
> Lagm := subs({theta[2](t) = theta[2]}, Lagm) :
> Lagm := subs({phi[1](t) = phi[1]}, Lagm) :
> Lagm := simplify(subs({phi[2](t) = phi[2]}, Lagm));
>
```

p1 - partial derivative of Lag with respect to angles theta1,theta2,phi1,phi2
p3 - partial derivative of Lag with respect to t derivatives of angles theta1p,theta2p,phi1p,phi2p
p2 - t derivative of p3

d/dt p3 = p1 or p2 - p1 = 0 A system of 4 equations in 4 unknowns.

```
> p1[1] := simplify(diff(Lagm, theta[1]));
> p3[1] := simplify(diff(Lagm, theta1p));
> p1[2] := simplify(diff(Lagm, theta[2]));
> p3[2] := simplify(diff(Lagm, theta2p));
> p1[3] := simplify(diff(Lagm, phi[1]));
> p3[3] := simplify(diff(Lagm, phi1p));
> p1[4] := simplify(diff(Lagm, phi[2]));
> p3[4] := simplify(diff(Lagm, phi2p));
>
> for k from 1 by 1 to 4 do
    p1[k] := subs({theta[1] = theta[1](t)}, p1[k]) :
    p1[k] := subs({theta[2] = theta[2](t)}, p1[k]) :
    p1[k] := subs({phi[1] = phi[1](t)}, p1[k]) :
    p1[k] := subs({phi[2] = phi[2](t)}, p1[k]) :
    p1[k] := subs({theta1p = theta1p(t)}, p1[k]) :
    p1[k] := subs({theta2p = theta2p(t)}, p1[k]) :
    p1[k] := subs({phi1p = phi1p(t)}, p1[k]) :
    p1[k] := subs({phi2p = phi2p(t)}, p1[k]) :
    p3[k] := subs({theta[1] = theta[1](t)}, p3[k]) :
    p3[k] := subs({theta[2] = theta[2](t)}, p3[k]) :
```

```
    p3[k] := subs({phi[1] = phi[1](t)}, p3[k]) :
    p3[k] := subs({phi[2] = phi[2](t)}, p3[k]) :
    p3[k] := subs({theta1p = theta1p(t)}, p3[k]) :
    p3[k] := subs({theta2p = theta2p(t)}, p3[k]) :
    p3[k] := subs({phi1p = phi1p(t)}, p3[k]) :
    p3[k] := subs({phi2p = phi2p(t)}, p3[k]) :
    p2[k] := diff(p3[k], t) :                          # t derivative
    Leqnc[k] := simplify(p1[k] - p2[k]) :              # Lagrange Equation
    Leqnc[k] := subs({diff(theta[1](t), t) = theta1p}, Leqnc[k]);
    Leqnc[k] := subs({diff(theta[2](t), t) = theta2p}, Leqnc[k]);
    Leqnc[k] := subs({diff(phi[1](t), t) = phi1p}, Leqnc[k]);
    Leqnc[k] := subs({diff(phi[2](t), t) = phi2p}, Leqnc[k]);
    Leqnc[k] := subs({diff(theta1p(t), t) = theta1pp}, Leqnc[k]);
    Leqnc[k] := subs({diff(theta2p(t), t) = theta2pp}, Leqnc[k]);
    Leqnc[k] := subs({diff(phi1p(t), t) = phi1pp}, Leqnc[k]);
    Leqnc[k] := subs({diff(phi2p(t), t) = phi2pp}, Leqnc[k]);
    Leqnc[k] := subs({theta[1](t) = theta[1]}, Leqnc[k]);
    Leqnc[k] := subs({theta[2](t) = theta[2]}, Leqnc[k]);
    Leqnc[k] := subs({phi[1](t) = phi[1]}, Leqnc[k]);
    Leqnc[k] := subs({phi[2](t) = phi[2]}, Leqnc[k]);
    Leqnc[k] := subs({theta1p(t) = theta1p}, Leqnc[k]);
    Leqnc[k] := subs({theta2p(t) = theta2p}, Leqnc[k]);
    Leqnc[k] := subs({phi1p(t) = phi1p}, Leqnc[k]);
    Leqnc[k] := subs({phi2p(t) = phi2p}, Leqnc[k]);
  end:
>
> for k from 1 by 1 to 4 do
    Lenqc[k] := simplify(Leqnc[k]); # The 4 equations
  end
>
> for k from 1 by 1 to 4 do
    Leqn[k] := Leqnc[k] :
  end:
> for k from 1 by 1 to 4 do
    Leqn[k] := simplify(Leqnc[k] - coeff(Leqnc[k], theta1pp, 1)·theta1pp - coeff(Leqnc[k],
      theta2pp, 1)·theta2pp - coeff(Leqnc[k], phi1pp, 1)·phi1pp - coeff(Leqnc[k], phi2pp, 1)
      ·phi2pp + A[k, 1]·ans[1] + A[k, 2]·ans[2] + A[k, 3]·ans[3] + A[k, 4]·ans[4]) :
  end:
> for k from 1 by 1 to 4 do
    print(k, simplify(Leqn[k]));
  end
>
```

Solve for the ODEs using the Euler-Lagrange equations using Maple.

```
> answer := solve({Leqn[1], Leqn[2], Leqn[3], Leqn[4]}, {ans[1], ans[2], ans[3], ans[4]}) :
>
```

```
> for k from 1 by 1 to 4 do
    ODE[k] := simplify(rhs(answer[k])) :
  end:
>
```

Build the 4 x 4 matrix A. Calculate its determinant. Check that A is symmetric.

```
> for k from 1 by 1 to 4 do
     A[k, 1] := simplify(coeff(Leqnc[k], theta1pp, 1));
     A[k, 2] := simplify(coeff(Leqnc[k], theta2pp, 1));
     A[k, 3] := simplify(coeff(Leqnc[k], phi1pp, 1));
     A[k, 4] := simplify(coeff(Leqnc[k], phi2pp, 1));
  end
> A := [[A[1, 1], A[1, 2], A[1, 3], A[1, 4]], [A[2, 1], A[2, 2], A[2, 3], A[2, 4]], [A[3, 1], A[3,
       2], A[3, 3], A[3, 4]], [A[4, 1], A[4, 2], A[4, 3], A[4, 4]]];
> matrix(A);
> dA := simplify(det(A));
```

> $dAs := -\left(m_2 \sin\left(\theta_1\right)^2 \sin\left(\theta_2\right)^2 \cos\left(-\phi_2 + \phi_1\right)^2 + 2\cos\left(\theta_2\right)\cos\left(-\phi_2 + \phi_1\right)\cos\left(\theta_1\right)\sin\left(\theta_1\right)\sin\left(\theta_2\right) m_2 + \cos\left(\theta_2\right)^2 \cos\left(\theta_1\right)^2 m_2 - m_1 - m_2\right) L_1^4 \sin\left(\theta_1\right)^2 m_1 m_2^2 L_2^4 \sin\left(\theta_2\right)^2;$

```
> simplify(dA - dAs);
>
> sort(collect(dAs, m[1]), m[1]);
> sort(collect(dAs, m[2]), m[2]);
>
```

Check that A is symmetric.

```
> for k from 1 by 1 to 4 do
    for j from k by 1 to 4 do
       print(A[k, j] - A[j, k]) :
  end
  end
>
```

Get the B. Remember to put in the minus sign when solving Ax = B.

```
> for k from 1 by 1 to 4 do
     B[k] := Leqnc[k] - A[k, 1]·theta1pp - A[k, 2]·theta2pp - A[k, 3]·phi1pp - A[k, 4]
        ·phi2pp;
  end
>
> b := [-simplify(B[1]), -simplify(B[2]), -simplify(B[3]), -simplify(B[4])];
>
```

Solve the linear system to get the ODEs using Maple.

```
> ans := linsolve(A, b) :
>
```

Now build the matrices to determine the ODEs using Cramer's rule.

```
> An[1] := [[b[1], A[1, 2], A[1, 3], A[1, 4]], [b[2], A[2, 2], A[2, 3], A[2, 4]], [b[3], A[3, 2],
      A[3, 3], A[3, 4]], [b[4], A[4, 2], A[4, 3], A[4, 4]]] :
> An[2] := [[A[1, 1], b[1], A[1, 3], A[1, 4]], [A[2, 1], b[2], A[2, 3], A[2, 4]], [A[3, 1], b[3],
      A[3, 3], A[3, 4]], [A[4, 1], b[4], A[4, 3], A[4, 4]]] :
> An[3] := [[A[1, 1], A[1, 2], b[1], A[1, 4]], [A[2, 1], A[2, 2], b[2], A[2, 4]], [A[3, 1], A[3,
      2], b[3], A[3, 4]], [A[4, 1], A[4, 2], b[4], A[4, 4]]] :
> An[4] := [[A[1, 1], A[1, 2], A[1, 3], b[1]], [A[2, 1], A[2, 2], A[2, 3], b[2]], [A[3, 1], A[3,
      2], A[3, 3], b[3]], [A[4, 1], A[4, 2], A[4, 3], b[4]]] :
>
> for k from 1 by 1 to 4 do
    dAn[k] := det(An[k]) :
  end:
>
> for k from 1 by 1 to 4 do
    CramerODE[k] := simplify(det(An[k])/det(A)) :
  end:
>
```

Print out the four ODEs for theta_1",theta_2",phi_1" and phi_2"

```
> simplify(ans[1]);
> simplify(ans[2]);
> simplify(ans[3]);
> simplify(ans[4]);
>
> m[1] := 0;
> simplify(ans[1]);
> simplify(ans[2]);
> simplify(ans[3]);
> simplify(ans[4]);
>
> unassign('m');
>
> simplify(limit(ans[1], m[1]=0));
> simplify(limit(ans[2], m[1]=0))
> simplify(limit(ans[3], m[1]=0));
```

```
> simplify(limit(ans[4], m[1]=0));
>
> m[2] := 0;
> simplify(ans[1]);
> simplify(ans[2]);
> simplify(ans[3]);
> simplify(ans[4]);
>
> unassign('m');
>
> phi[1] := ang; phi[2] := ang;
> phi1p := 0; phi2p := 0;
>
> simplify(ans[1]);
> simplify(ans[2]);
> simplify(ans[3]);
> simplify(ans[4]);
```

theta_1' = 0 = theta_2', phi_1 = q = phi_2, phi_1' = v = phi_2' Tracing out two cones

```
>
> theta1p := 0;
> theta2p := 0;
> phi[1] := q;
> phi[2] := q;
> phi1p := v;
> phi2p := v;
>
> simplify(ans[1]);
> simplify(ans[2]);
> simplify(ans[3]);
> simplify(ans[4]);
>
```

Determine the conditions on L so that theta1 and theta2 are constant and phi_1 and phi_2 are increasing

```
> eqn1 := sort(collect(numer(ans[1]), m[1]), m[1]);
> eqn2 := sort(collect(numer(ans[2]), m[1]), m[1]);
>
> simplify(eqn1);
> simplify(eqn2);
> Lans := solve({eqn1, eqn2}, {L[1], L[2]});
```

```
> simplify(Lans[1]);
> simplify(Lans[2]);
>
> subs({L[1]=rhs(Lans[1]), L[2]=rhs(Lans[2])}, ans[1]);
> simplify(subs({L[1]=rhs(Lans[1]), L[2]=rhs(Lans[2])}, ans[1]));
> simplify(subs({L[1]=rhs(Lans[1]), L[2]=rhs(Lans[2])}, ans[2]));
>
> unassign('theta1p','theta2p','phi','phi1p','phi2p');
>
> simplify(limit(ans[1], theta[1]=0));
> simplify(limit(ans[2], theta[1]=0));
> simplify(limit(ans[3], theta[1]=0));
> simplify(limit(ans[4], theta[1]=0));
>
> simplify(limit(ans[1], theta[2]=0));
> simplify(limit(ans[2], theta[2]=0));
> simplify(limit(ans[3], theta[2]=0));
> simplify(limit(ans[4], theta[2]=0));
>
```