

classical mechanics with maxima

Published: September 27, 2026 | Index ID: #-

Introduction

Classical mechanics is the foundation of modern physics, describing the motion of bodies from everyday objects to celestial bodies. While textbooks provide the mathematical framework, solving real-world problems often requires symbolic manipulation, algebraic simplification, and numerical integration. Maxima, a free and open source computer algebra system (CAS), offers a powerful platform to tackle these tasks. In this guide we explore how to use Maxima for classical mechanics, from basic kinematics to advanced topics like Hamiltonian dynamics and perturbation theory. Whether you are a student, educator, or researcher, this article will show you how to harness Maxima's capabilities to deepen your understanding of classical mechanics and streamline your workflow.

What is Maxima?

Maxima is a computer algebra system that originated from the Macsyma project at MIT. It is written in Common Lisp and provides extensive support for symbolic computation, numerical analysis, and visualization. Key features include:

- Symbolic differentiation, integration, and simplification.
- Linear algebra operations and eigenvalue computations.
- Differential equation solvers.
- Plotting and 3 D visualization.
- Extensible through user defined functions and packages.

Because Maxima is open source, it is freely available on Windows, macOS, Linux, and even mobile platforms via web interfaces. Its scripting language is concise and expressive, making it ideal for automating repetitive calculations common in classical mechanics.

Why Use Maxima for Classical Mechanics?

Classical mechanics involves manipulating equations that can become algebraically heavy. Maxima offers several advantages:

1. Symbolic Power: Derive equations of motion analytically, simplify expressions, and verify identities.
2. Reusability: Write functions once and reuse them across multiple problems, saving time and reducing errors.
3. Educational Clarity: Visualize trajectories and phase space diagrams to aid intuition.
4. Integration with Other Tools: Export results to LaTeX, Python, or MATLAB for further analysis.
5. Cost Effective: No licensing fees, making it accessible for students and institutions.

These strengths make Maxima an excellent companion for exploring classical mechanics with precision and flexibility.

Basic Setup and Syntax

Before diving into mechanics, let's get acquainted with Maxima's environment and syntax. Launch Maxima (e.g., `maxima -w` for a console session) and you'll see a prompt like:

[1]

Maxima uses a Lisp like syntax. Here are a few basic commands:

- `diff(f, x);` – Symbolic derivative of f with respect to x .
- `integrate(f, x);` – Symbolic integral of f with respect to x .
- `solve(eq, x);` – Solve equation eq for x .
- `plot2d(f, [x, a, b]);` – Plot function f over interval $[a, b]$.
- `diff(f, [x, y]);` – Partial derivative of f with respect to y .

Maxima automatically simplifies expressions, but you can force simplification with `ratsimp` or `expand`. Remember that each command ends with a semicolon.

Solving Kinematics Problems

Projectile Motion

Consider a projectile launched with initial speed v_0 at angle θ . The position components are:

$$x(t) := v_0 \cos(\theta) t$$

$$y(t) := v_0 \sin(\theta) t - \frac{1}{2} g t^2$$

To find the time of flight, set $y(t) = 0$ and solve:

$$\text{time_of_flight} := \text{solve}(y(t) = 0, t);$$

Maxima returns two solutions; the positive one is physically relevant. You can then compute horizontal range and maximum height using similar substitutions.

Rotational Kinematics

For a rotating rigid body with angular velocity ω the angular displacement is $\theta = \int \omega dt$. If angular acceleration α is constant, you can derive $\omega = \alpha t$ and $\theta = \frac{1}{2} \alpha t^2$ by integrating twice. Maxima's `diff` and `integrate` functions make this straightforward.

Dynamics with Lagrangian Mechanics

Formulating the Lagrangian

For a system with generalized coordinates $q_1, q_2, \dots$, the Lagrangian is $L = T - V$, where T is kinetic energy and V is potential energy. Maxima can handle symbolic expressions for T and V . Example: a simple pendulum of length L and mass m :

$$\theta(t) := q(t);$$

$$T := 0.5 * m * \text{diff}(\theta(t), t)^2 * L;$$

$$V := m * g * L * (1 - \cos(\theta(t)));$$

$$\text{Lagrangian} := T - V;$$

Euler–Lagrange Equations

Maxima can automatically compute the Euler–Lagrange equation:

$$\text{eq} := \text{diff}(\text{diff}(\text{Lagrangian}, \text{diff}(\theta(t), t)), t) - \text{diff}(\text{Lagrangian}, \theta(t));$$

$$\text{eq_simplified} := \text{ratsimp}(\text{eq});$$

Setting `eq_simplified = 0` yields the equation of motion. For the pendulum, you recover $\theta'' + (g/L) \sin(\theta) = 0$.

Multiple Degrees of Freedom

For systems with several coordinates, define a vector of coordinates and use matrix operations. Maxima's matrix and diff functions allow you to compute the kinetic energy matrix and generalized forces efficiently.

Hamiltonian Mechanics

Legendre Transformation

Convert from Lagrangian to Hamiltonian via $H = \sum_i p_i \dot{q}_i - L$, where $p_i = \frac{\partial L}{\partial \dot{q}_i}$. Maxima can symbolically compute momenta and perform the transformation:

```
p_theta := diff(Lagrangian, diff(theta(t), t));
```

```
H := p_theta*diff(theta(t), t) - Lagrangian;
```

Canonical Equations

Hamilton's equations are:

- $\dot{q}_i = \frac{\partial H}{\partial p_i}$
- $\dot{p}_i = -\frac{\partial H}{\partial q_i}$

Use Maxima to differentiate H with respect to each variable, yielding first order differential equations that can be solved numerically with ode2 or desolve.

Phase Space Visualization

Plotting q versus p reveals the system's dynamics. Maxima's plot2d or plot3d can generate phase portraits, helping identify fixed points and stability properties.

Constraint Systems

Holonomic Constraints

Constraints that depend only on coordinates and time can be incorporated by introducing Lagrange multipliers. Maxima can symbolically handle the augmented Lagrangian:

```
constraint := f(q1, q2, t) - 0;
```

```
lambda := 'lambda; // declare symbolic multiplier
```

```
L_aug := Lagrangian + lambda*constraint;
```

Derive Euler–Lagrange equations for both q_i and lambda to solve for constrained motion.

Scleronomous Constraints

When constraints involve velocities, use the method of undetermined multipliers. Maxima's ability to compute partial derivatives with respect to velocities simplifies this process.

Nonholonomic Constraints

Constraints that are nonintegrable (e.g., rolling without slipping) require the use of generalized forces. Maxima can implement these via the Lagrange–d'Alembert principle, adding constraint forces directly to the equations of motion.

Symmetry and Conservation

Noether's Theorem

Symmetries in the Lagrangian lead to conserved quantities. Maxima can symbolically verify conservation laws by checking whether the total derivative of a candidate quantity is zero.

If the Lagrangian does not explicitly depend on coordinate x , momentum p_x is conserved. Verify in Maxima:

```
dp_x_dt := diff(p_x, t);  
simplify(dp_x_dt); // should return 0 if symmetry holds
```

Example: Rotational Symmetry

For rotational invariance, angular momentum is conserved. Define the angular momentum vector and confirm its time derivative vanishes.

Integrals of Motion

Use Maxima to find invariants by solving $\text{diff}(I, t) = 0$ for a function $I(q, p)$. This process can uncover hidden constants of motion in complex systems.

Numerical Integration and Simulation

Solving Ordinary Differential Equations

Maxima's `ode2` solves first order ODEs analytically when possible, while `desolve` handles linear second order equations. For nonlinear systems, use `ode2` with the `num` flag to obtain numerical solutions.

```
eqn := diff(theta(t), t, t) + (g/L)*sin(theta(t)) = 0;  
sol := ode2(eqn, theta(t), t, [theta(0) = theta0, diff(theta(t), t)(0) = omega0]);
```

Runge–Kutta Methods

For stiff or highly nonlinear problems, implement a Runge–Kutta scheme manually or use the `rk4` package. Maxima's array operations make it easy to iterate over time steps.

Visualization of Trajectories

Plotting position and velocity

Baca Juga (Artikel Terkait)

- [class xii chemistry ch 2 solutions](http://www.bbglobalsolutions.com/class-xii-chemistry-ch-2-solutions.pdf)

URL: <http://www.bbglobalsolutions.com/class-xii-chemistry-ch-2-solutions.pdf>

- [class xi maths supplementary material solution](http://www.bbglobalsolutions.com/class-xi-maths-supplementary-material-solution.pdf)

URL: <http://www.bbglobalsolutions.com/class-xi-maths-supplementary-material-solution.pdf>

- [class xi isc total english handbook pdf](http://www.bbglobalsolutions.com/class-xi-isc-total-english-handbook-pdf.pdf)

URL: <http://www.bbglobalsolutions.com/class-xi-isc-total-english-handbook-pdf.pdf>

- [class xi english question and answers](http://www.bbglobalsolutions.com/class-xi-english-question-and-answers.pdf)

URL: <http://www.bbglobalsolutions.com/class-xi-english-question-and-answers.pdf>

Tags: #classical #mechanics #with #maxima

