The Outer Solar System

Yingbo Ma, Chris Rackauckas

February 23, 2019

0.1 Data

The chosen units are: masses relative to the sun, so that the sun has mass 1. We have taken $m_0 = 1.00000597682$ to take account of the inner planets. Distances are in astronomical units , times in earth days, and the gravitational constant is thus $G = 2.95912208286 \cdot 10^4$.

planet	mass	
Jupiter	$m_1 = 0.000954786104043$	3.5023653 3.8169847 1.55
Saturn	$m_2 = 0.000285583733151$	9.0755314 3.0458353 1.64
Uranus	$m_3 = 0.0000437273164546$	8.3101420 16.2901086 7.25
Neptune	$m_4 = 0.0000517759138449$	11.4707666 25.7294829 10.81
Pluto	$ m_5 = 1/(1.3 \cdot 10^8)$	15.5387357 25.2225594 3.19

The data is taken from the book "Geometric Numerical Integration" by E. Hairer, C. Lubich and G. Wanner.

```
using Plots, OrdinaryDiffEq, DiffEqPhysics, RecursiveArrayTools
gr()

G = 2.95912208286e-4
M = [1.00000597682, 0.000954786104043, 0.000285583733151, 0.0000437273164546,
    0.0000517759138449, 1/1.3e8]
planets = ["Sun", "Jupiter", "Saturn", "Uranus", "Neptune", "Pluto"]

pos_x = [0.0,-3.5023653,9.0755314,8.3101420,11.4707666,-15.5387357]
pos_y = [0.0,-3.8169847,-3.0458353,-16.2901086,-25.7294829,-25.2225594]
pos_z = [0.0,-1.5507963,-1.6483708,-7.2521278,-10.8169456,-3.1902382]
pos = ArrayPartition(pos_x,pos_y,pos_z)

vel_x = [0.0,0.00565429,0.00168318,0.00354178,0.00288930,0.00276725]
vel_y = [0.0,-0.00412490,0.00483525,0.00137102,0.00114527,-0.00170702]
vel_z = [0.0,-0.00190589,0.00192462,0.00055029,0.00039677,-0.00136504]
vel = ArrayPartition(vel_x,vel_y,vel_z)

tspan = (0.,200_000)
```

The N-body problem's Hamiltonian is

$$H(p,q) = \frac{1}{2}\sum_{i=0}^{N}\frac{p_i^T p_i}{m_i} - G\sum_{i=1}^{N}\sum_{j=0}^{i-1}\frac{m_i m_j}{\|q_i - q_j\|} \tag{1}$$

Here, we want to solve for the motion of the five outer planets relative to the sun, namely, Jupiter, Saturn, Uranus, Neptune and Pluto.

```
const ∑ = sum
const N = 6
potential(p, t, x, y, z, M) = -G*∑(i->∑(j->(M[i]*M[j])/sqrt((x[i]-x[j])^2 +
    (y[i]-y[j])^2 + (z[i]-z[j])^2), 1:i-1), 2:N)
```

0.2 Hamiltonian System

`NBodyProblem` constructs a second order ODE problem under the hood. We know that a Hamiltonian system has the form of

$$\dot{p} = -H_q(p, q) \quad \dot{q} = H_p(p, q) \tag{2}$$

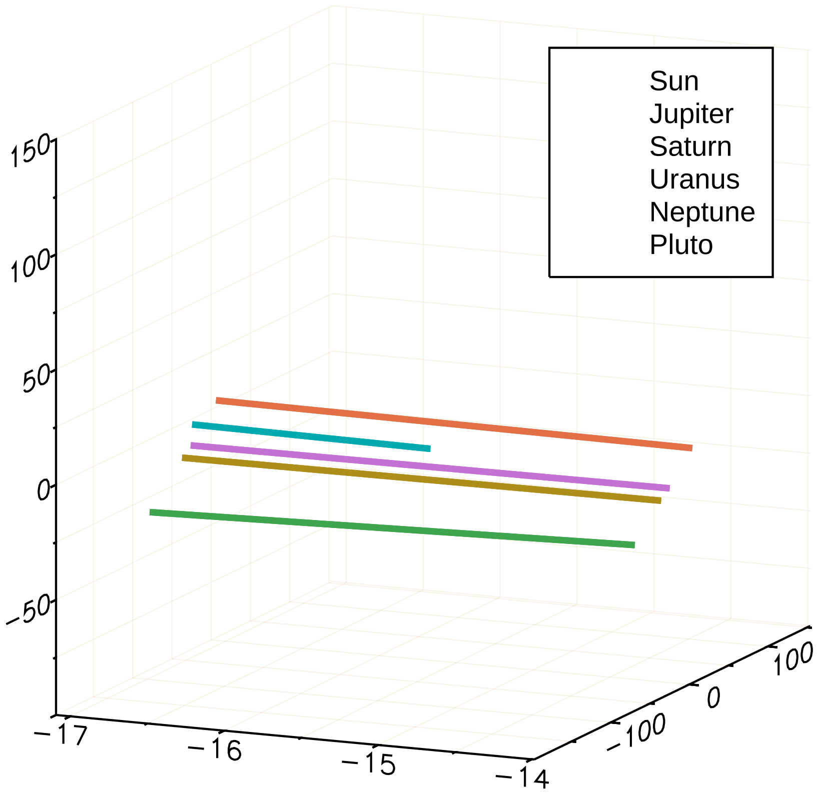
For an N-body system, we can symplify this as:

$$\dot{p} = -\nabla V(q) \quad \dot{q} = M^{-1}p. \tag{3}$$

Thus $\dot{q}$ is defined by the masses. We only need to define $\dot{p}$, and this is done internally by taking the gradient of V. Therefore, we only need to pass the potential function and the rest is taken care of.

```
nprob = NBodyProblem(potential, M, pos, vel, tspan)
sol = solve(nprob,Yoshida6(), dt=100);
```

```
orbitplot(sol,body_names=planets)
```



0.3 Appendix

```
using DiffEqTutorials
DiffEqTutorials.tutorial_footer(WEAVE_ARGS[:folder],WEAVE_ARGS[:file])
```

```
These benchmarks are part of the DiffEqTutorials.jl repository, found at:

https://github.com/JuliaDiffEq/DiffEqTutorials.jl

To locally run this tutorial, do the following commands:

using DiffEqTutorials
DiffEqTutorials.weave_file(".","models/outer_solar_system.jmd")
```

```
Computer Information:

Julia Version 1.1.0
Commit 80516ca202 (2019-01-21 21:24 UTC)
Platform Info:
  OS: Windows (x86_64-w64-mingw32)
  CPU: Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-6.0.1 (ORCJIT, skylake)
Environment:
  JULIA_EDITOR = "C:\Users\accou\AppData\Local\atom\app-1.34.0\atom.exe" -a
  JULIA_NUM_THREADS = 6
```

```
Package Information:

    Status `C:\Users\accou\.julia\environments\v1.1\Project.toml`
  [c52e3926] Atom v0.7.14
  [6e4b80f9] BenchmarkTools v0.4.2
  [336ed68f] CSV v0.4.3
  [be33ccc6] CUDAnative v1.0.1
  [3a865a2d] CuArrays v0.9.1
  [a93c6f00] DataFrames v0.17.1
  [39dd38d3] Dierckx v0.4.1
  [459566f4] DiffEqCallbacks v2.5.2
  [aae7a2af] DiffEqFlux v0.2.0
  [c894b116] DiffEqJump v6.1.0+ [`C:\Users\accou\.julia\dev\DiffEqJump`]
  [1130ab10] DiffEqParamEstim v1.5.1
  [055956cb] DiffEqPhysics v3.1.0
  [225cb15b] DiffEqTutorials v0.0.0 [`C:\Users\accou\.julia\external\DiffEq
Tutorials.jl`]
  [0c46a032] DifferentialEquations v6.3.0
  [587475ba] Flux v0.7.3
  [f6369f11] ForwardDiff v0.10.3+ [`C:\Users\accou\.julia\dev\ForwardDiff`]
  [7073ff75] IJulia v1.17.0
  [c601a237] Interact v0.9.1
  [b6b21f68] Ipopt v0.5.4
  [4076af6c] JuMP v0.18.5
  [e5e0dc1b] Juno v0.5.4
  [76087f3c] NLopt v0.5.1
  [429524aa] Optim v0.17.2
  [1dea7af3] OrdinaryDiffEq v5.1.4+ [`C:\Users\accou\.julia\dev\OrdinaryDif
fEq`]
  [65888b18] ParameterizedFunctions v4.1.0
  [91a5bcdd] Plots v0.23.0
  [71ad9d73] PuMaS v0.0.0 [`C:\Users\accou\.julia\dev\PuMaS`]
  [d330b81b] PyPlot v2.7.0
  [731186ca] RecursiveArrayTools v0.20.0
  [90137ffa] StaticArrays v0.10.2
  [789caeaf] StochasticDiffEq v6.1.1+ [`C:\Users\accou\.julia\dev\Stochasti
cDiffEq`]
  [c3572dad] Sundials v3.0.0
  [44d3d7a6] Weave v0.7.1
```