15_IRKNGL_SIMD_luzea

December 18, 2022

1 15body problem IRNKGL-SIMD

- Newtonian point-mass $(15 + 1)$-body model of the Solar System: the Sun, the eight planets, Pluto and five main bodies of the asteroid belt (Ceres, Pallas, Vesta, Iris and Bamberga)

Loading packages and functions

Definition of the N-body problem

Integrations

Errors in energy

Errors in position

1.1 Loading packages and functions

[1]:
```
using LinearAlgebra
using Plots
using JLD2
using Dates
```

[2]:
```
PATH_SRC="../../src_seq/"

include(string(PATH_SRC,"IRKGL_SEQ.jl"))
using .IRKGL_SEQ


PATH_SRC="../../src_simd/"

include(string(PATH_SRC,"IRKGL_SIMD.jl"))
using .IRKGL_SIMD
```

[3]:
```
run=false
```

[3]:
```
false
```

Back to the top

1.2 Definition of the N-body problem

In Nbody.jl below, the following functions are defined: NbodyEnergy(u,Gm), NbodyODE!(du,u,Gm,t), and NbodyODE1!(du,u,Gm,t), where

$$u = \begin{pmatrix} q_1 & v_1 \\ \vdots & \vdots \\ q_N & v_N \end{pmatrix} \in \mathbb{R}^{2\times 3\times N}, \quad Gm = (G\,m_1, \dots, G\,m_N) \in \mathbb{R}^N.$$

The energy, as a function of the positions $q_1, \dots, q_N \in \mathbb{R}^3$ and the velocities $v_1, \dots, v_N \in \mathbb{R}^3$ of the N bodies is:

$$\mathrm{Energy}(q_1, \dots, q_N, v_1, \dots, v_N) = \frac{1}{2}\sum_{i=1}^{N} m_i \, \|v_i\|^2 - G \sum_{1\le i<j\le N} \frac{m_i m_j}{\|q_i - q_j\|}.$$

The ODE system of the N-body problem, as a function of the positions $q_1, \dots, q_N \in \mathbb{R}^3$ and the velocities $v_1, \dots, v_N \in \mathbb{R}^3$ of the N bodies is:

$$\begin{aligned} \frac{d}{dt} q_i &= v_i, \\ \frac{d}{dt} v_i &= G \sum_{j\neq i} \frac{m_j}{\|q_j - q_i\|^3} \, (q_j - q_i). \end{aligned}$$

This system of ODEs can be writen in compact form as

$$\frac{du}{dt} = f(t, u, Gm)$$

[4]:
```
PATH_ODES="../../ODEProblems/"

include(string(PATH_ODES,"Initial15Body.jl"))
include(string(PATH_ODES,"Nbody.jl"))
```

[4]:
```
NbodyODE! (generic function with 2 methods)
```

1.2.1 Initial value problem: 15-body problem

[5]:
```
u0, Gm, bodylist = Initial15Body(Float64)
q0=u0[:,:,1]
v0=u0[:,:,2]
dim=length(size(u0))

N = length(Gm)

show(bodylist)
E0=NbodyEnergy(u0,Gm)
```

```
["Sun" "Mercury" "Venus" "EMB" "Mars" "Jupiter" "Saturn" "Uranus" "Neptune"
"Pluto" "Ceres" "Pallas" "Vesta" "Iris" "Bamberga"]
```

[5]: -9.831963632201247e-12

[6]:
```julia
t0 = 0.
tF=15000.
tF=5.49*10^6    # ~ 15.000 years

prob = ODEProblem(NbodyODE!, u0, (t0,tF), Gm)

if run==true
   @save "./Data/prob_15body_FT15Y.jld2" prob
else
   JLD2.@load "./Data/prob_15body_FT15Y.jld2" prob
end
```

[6]:
```
1-element Vector{Symbol}:
 :prob
```

Back to the top

1.3 Integrations

1.3.1 IRKNGL integrazioa-0 (exact solution)

[7]:
```julia
now()
```

[7]: 2022-12-18T21:27:55.030

[8]:
```julia
dt = 0.5

nout=1000
m = Int64(ceil((tF-t0)/(nout*dt)))
println("dt = $dt, m=$m")
(tF-t0)/(nout*dt)
```

```
dt = 0.5, m=10980
```

[8]: 10980.0

[9]:
```julia
if run ==true
   alg=IRKNGL_simd(s=8, initial_interp=1, m=m,myoutputs=true)
   sol0,iters0=solve(prob,alg,dt=dt, adaptive=false)
   @save "./Data/sol0_15body_FT15Y.jld2" sol0
else
   JLD2.@load "./Data/sol0_15body_FT15Y.jld2" sol0
end
```

[9]:
```
1-element Vector{Symbol}:
 :sol0
```

[10]:
```julia
now()
```

[10]: 2022-12-18T21:27:55.626

1.3.2 IRKNGL integrazioa-1

[11]:
```julia
now()
```

[11]: 2022-12-18T21:27:55.626

[12]:
```julia
dt = 3.

nout=1000
m = Int64(ceil((tF-t0)/(nout*dt)))
println("dt = $dt, m=$m")
(tF-t0)/(nout*dt)
```

```
dt = 3.0, m=1830
```

[12]: 1830.0

[13]:
```julia
if run ==true
   alg=IRKNGL_simd(s=8, initial_interp=1, m=m,myoutputs=true)
   sol1,iters1=solve(prob,alg,dt=dt, adaptive=false)
   @save "./Data/sol1_15body_FT15Y.jld2" sol1
else
   JLD2.@load "./Data/sol1_15body_FT15Y.jld2" sol1
end
```

[13]:
```
1-element Vector{Symbol}:
 :sol1
```

[14]:
```julia
now()
```

[14]: 2022-12-18T21:27:55.721

1.3.3 IRKNGL integrazioa-2

[15]:
```julia
now()
```

[15]: 2022-12-18T21:27:55.722

[16]:
```julia
dt = 1.5

nout=1000
m = Int64(ceil((tF-t0)/(nout*dt)))
println("dt = $dt, m=$m")
```

```
(tF-t0)/(nout*dt)
```

```
dt = 1.5, m=3660
```

[16]: 3660.0

[17]:
```
if run ==true
   alg=IRKNGL_simd(s=8, initial_interp=1, m=m,myoutputs=true)
   sol2,iters2=solve(prob,alg,dt=dt, adaptive=false)
   @save "./Data/sol2_15body_FT15Y.jld2" sol2
else
   JLD2.@load "./Data/sol2_15body_FT15Y.jld2" sol2
end
```

[17]:
```
1-element Vector{Symbol}:
 :sol2
```

[18]:
```
now()
```

[18]: 2022-12-18T21:27:55.810

1.3.4 IRKNGL integrazioa-3 (adaptive=true!!)

[19]:
```
now()
```

[19]: 2022-12-18T21:27:55.812

[20]:
```
dt = 3.

nout=1000
m = Int64(ceil((tF-t0)/(nout*dt)))
println("dt = $dt, m=$m")
(tF-t0)/(nout*dt)
```

```
dt = 3.0, m=1830
```

[20]: 1830.0

[21]:
```
if run ==true
   alg=IRKNGL_simd(s=8, initial_interp=1, m=m,myoutputs=true)
   sol3,iters3=solve(prob,alg,dt=dt, adaptive=true)
   @save "./Data/sol3_15body_FT15Y.jld2" sol3
else
   JLD2.@load "./Data/sol3_15body_FT15Y.jld2" sol3
end
```

[21]: 1-element Vector{Symbol}:
 :sol3

[22]:
```
now()
```

[22]: 2022-12-18T21:27:55.826

1.3.5 IRKNGL integrazioa-4 (adaptive=true & saveat=tout!!)

- Kokapen erroreak kalkulatzeko integrazioa
- nout zenbaki osoa eta tout=$[t_1, t_2, \ldots, t_{nout}]$ izanik, $u_{t_k} \approx u(t_k) \quad t_k \in tout$ kalkulatzea

[23]:
```
now()
```

[23]: 2022-12-18T21:27:55.827

[24]:
```
dt = 3.

nout=1000
m = Int64(ceil((tF-t0)/(nout*dt)))
saveat=t0:dt*m:tF
println("dt = $dt, m=$m")
(tF-t0)/(nout*dt)
```

dt = 3.0, m=1830

[24]: 1830.0

[25]:
```
m=1
if run ==true
   alg=IRKNGL_simd(s=8, initial_interp=1, m=m,myoutputs=true)
   sol4,iters4=solve(prob,alg,dt=dt, saveat=saveat[2:end], adaptive=true)
   @save "./Data/sol4_15body_FT15Y.jld2" sol4
else
   JLD2.@load "./Data/sol4_15body_FT15Y.jld2" sol4
end
```

```
 Warning: Backwards compatability support of the new return codes to Symbols
will be deprecated with the Julia v1.9 release. Please see
https://docs.sciml.ai/SciMLBase/stable/interfaces/Solutions/#retcodes for more
information
 @ SciMLBase /home/mikel/.julia/packages/SciMLBase/VKnrY/src/retcodes.jl:360
```

[26]:
```
now()
```

[26]: 2022-12-18T21:28:39.564

1.3.6 IRKNGL integrazioa-5 (adaptive=true & saveat=tout!!)

- Kokapen erroreak kalkulatzeko integrazioa
- nout zenbaki osoa eta tout=$[t_1, t_2, \dots, t_{nout}]$ izanik, $u_{t_k} \approx u(t_k) \quad t_k \in tout$ kalkulatzea

[27]:
```
now()
```

[27]: 2022-12-18T21:28:39.564

[28]:
```
dt = 1.5

nout=1000
m = Int64(ceil((tF-t0)/(nout*dt)))
saveat=t0:dt*m:tF
println("dt = $dt, m=$m")
(tF-t0)/(nout*dt)
```

```
dt = 1.5, m=3660
```

[28]: 3660.0

[29]:
```
m=1
if run ==true
   alg=IRKNGL_simd(s=8, initial_interp=1, m=m,myoutputs=true)
    sol5,iters5=solve(prob,alg,dt=dt, saveat=saveat[2:end], adaptive=true)
   @save "./Data/sol5_15body_FT15Y.jld2" sol5
else
   JLD2.@load "./Data/sol5_15body_FT15Y.jld2" sol5
end
```

```
 Warning: Backwards compatability support of the new return codes to Symbols
will be deprecated with the Julia v1.9 release. Please see
https://docs.sciml.ai/SciMLBase/stable/interfaces/Solutions/#retcodes for more
information
 @ SciMLBase /home/mikel/.julia/packages/SciMLBase/VKnrY/src/retcodes.jl:360
```

[30]:
```
now()
```

[30]: 2022-12-18T21:29:34.733

Back to the top

1.4 Errors in energy

[31]:
```
[length(sol1.t) length(sol2.t) length(sol3.t) length(sol4.t) length(sol5.t)]
```

[31]:
```
1×5 Matrix{Int64}:
 1001  1001  896  1001  1001
```

[32]:
```julia
eps(1e-25)
```

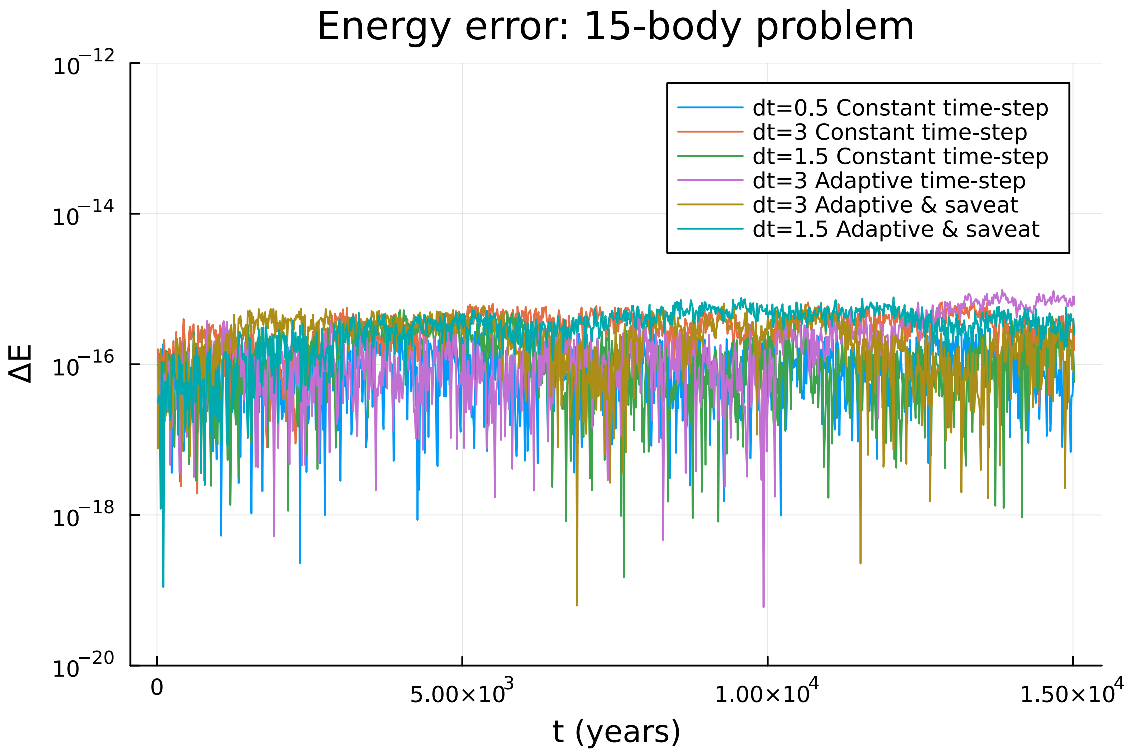
[32]: 1.1479437019748901e-41

[33]:
```julia
year = 365.5

function energy_plot(sol, yrange; label="")
    energies = [NbodyEnergy(BigFloat.(u),Gm) for u in sol.u]
    E0 = energies[1]
    epsilon =eps(1e-25)
    errors = (abs.(energies[2:end]/E0 .- 1)) .+ epsilon
    tt = sol.t[2:end]/year
    pl=plot!(tt,  errors, label=label, legend=:topright) #bottomright
    return pl
end
```

[33]: energy_plot (generic function with 1 method)

[34]:
```julia
yrange = (1e-20,1e-12)

plot(title="Energy error: 15-body problem",ylims=yrange, yscale=:log10,
     xlabel="t (years)", ylabel="ΔE")
pl0 = energy_plot(sol0, yrange,  label="dt=0.5 Constant time-step")
plot!(pl0)
pl1 = energy_plot(sol1, yrange,  label="dt=3 Constant time-step")
plot!(pl1)
pl2 = energy_plot(sol2, yrange,  label="dt=1.5 Constant time-step")
plot!(pl2)
pl3 = energy_plot(sol3, yrange,  label="dt=3 Adaptive time-step")
plot!(pl3)
pl4 = energy_plot(sol4, yrange,  label="dt=3 Adaptive & saveat")
plot!(pl4)
pl5 = energy_plot(sol5, yrange,  label="dt=1.5 Adaptive & saveat")
plot!(pl5)
```

[34]:

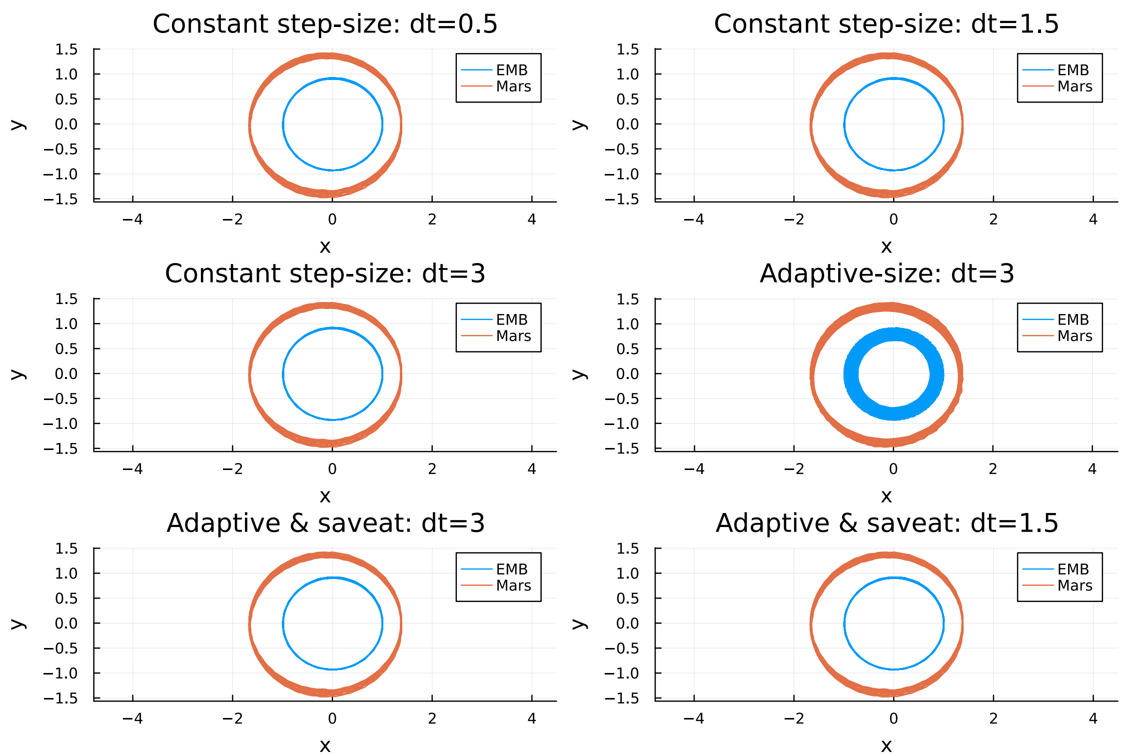


1.4.1 Projection of orbits onto XY plane

[35]:
```
pl0 = plot(title="Constant step-size: dt=0.5",
           xlabel="x", ylabel="y", aspect_ratio=1) #legend=false,
for j = 4:5
   xlist = map(u->u[1,j,1], sol0.u)
   ylist = map(u->u[2,j,1], sol0.u)
   pl0 = plot!(xlist,ylist, label=bodylist[j])
end


pl1 = plot(title="Constant step-size: dt=1.5",
           xlabel="x", ylabel="y", aspect_ratio=1) #legend=false,
for j = 4:5
   xlist = map(u->u[1,j,1], sol1.u)
   ylist = map(u->u[2,j,1], sol1.u)
   pl1 = plot!(xlist,ylist, label=bodylist[j])
end


pl2 = plot(title="Constant step-size: dt=3",
           xlabel="x", ylabel="y",  aspect_ratio=1) #legend=false,
for j = 4:5
```

```
    xlist = map(u->u[1,j,1], sol2.u)
    ylist = map(u->u[2,j,1], sol2.u)
    pl2 = plot!(xlist,ylist, label=bodylist[j])
end

pl3 = plot(title="Adaptive-size: dt=3",
          xlabel="x", ylabel="y", aspect_ratio=1) #legend=false,
for j = 4:5
    xlist = map(u->u[1,j,1], sol3.u)
    ylist = map(u->u[2,j,1], sol3.u)
    pl3 = plot!(xlist,ylist, label=bodylist[j])
end

pl4 = plot(title="Adaptive & saveat: dt=3",
          xlabel="x", ylabel="y", aspect_ratio=1) #legend=false,
for j = 4:5
    xlist = map(u->u[1,j,1], sol4.u)
    ylist = map(u->u[2,j,1], sol4.u)
    pl4 = plot!(xlist,ylist, label=bodylist[j])
end

pl5 = plot(title="Adaptive & saveat: dt=1.5",
          xlabel="x", ylabel="y", aspect_ratio=1) #legend=false,
for j = 4:5
    xlist = map(u->u[1,j,1], sol5.u)
    ylist = map(u->u[2,j,1], sol5.u)
    pl5 = plot!(xlist,ylist, label=bodylist[j])
end


plot(pl0, pl1, pl2,  pl3, pl4, pl5,  layout=(3,2), size=(900,600))
```

[35]:



Back to the top

1.5 Errors in position

[37]:
```
# Check: same initial condition
[norm(sol0.u[1]-sol1.u[1]) norm(sol0.u[1]-sol2.u[1]) norm(sol0.u[1]-sol3.u[1])␣
↪norm(sol0.u[1]-sol4.u[1]) norm(sol0.u[1]-sol5.u[1])]
```

[37]:
```
1×5 Matrix{Float64}:
 0.0  0.0  0.0  0.0  0.0
```

[38]:
```
k=2
[sol0.t[k] sol1.t[k] sol2.t[k] sol3.t[k] sol4.t[k] sol5.t[k]]
```

[38]:
```
1×6 Matrix{Float64}:
 5490.0  5490.0  5490.0  6151.31  5490.0  5490.0
```

1.5.1 Constant step-size

[39]:
```
qe1=Array{Array{Float64,1}}(undef,N)
for i in 1:N
    qe1[i]=map((u0,u1)-> log10(norm(u0[:,i,2]-u1[:,i,2])), sol0.u[2:end],sol1.
↪u[2:end])
```

```julia
end

qe2=Array{Array{Float64,1}}(undef,N)
for i in 1:N
    qe2[i]=map((u0,u2)-> log10(norm(u0[:,i,2]-u2[:,i,2])), sol0.u[2:end],sol2.
 ↪u[2:end])
end
```

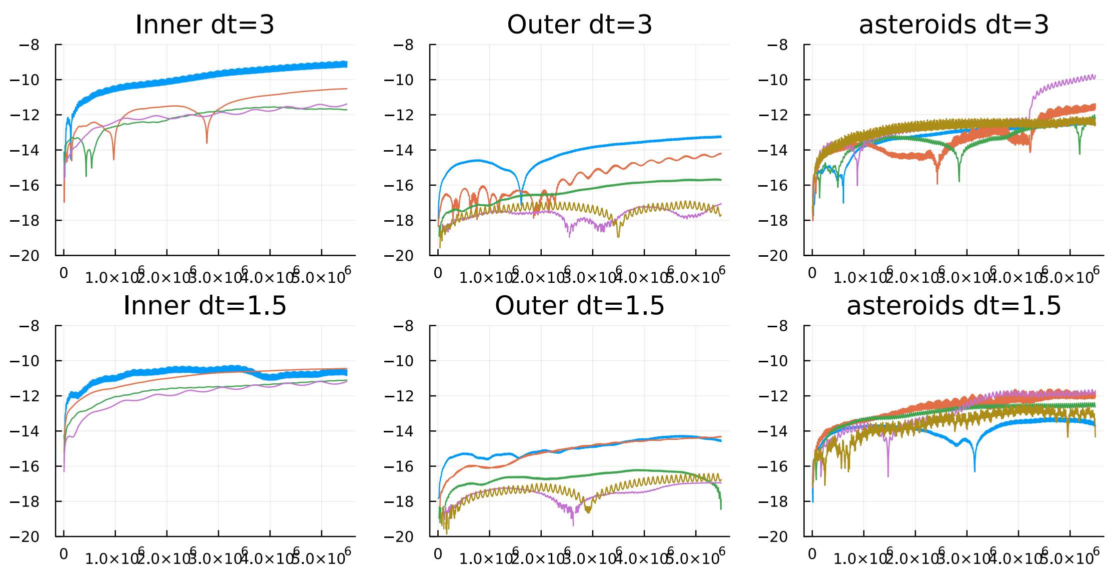
[40]:
```julia
[bodylist[1:5] bodylist[6:10] bodylist[11:15]]
```

[40]:
```
5×3 Matrix{String}:
 "Sun"      "Jupiter"  "Ceres"
 "Mercury"  "Saturn"   "Pallas"
 "Venus"    "Uranus"   "Vesta"
 "EMB"      "Neptune"  "Iris"
 "Mars"     "Pluto"    "Bamberga"
```

[41]:
```julia
MinE=-20
MaxE=-8


pe1=plot(sol0.t[2:end], qe1[2:5],
#      title ="Constant time-step dt=3 (inner bodies)",
     title="Inner dt=3",
     ylims=(MinE,MaxE),
     legend=false)

pe2=plot(sol0.t[2:end], qe1[6:10],
#      title ="Constant time-step dt=3 (outer bodies)",
     title="Outer dt=3",
     ylims=(MinE,MaxE),
     legend=false)

pe3=plot(sol0.t[2:end], qe1[11:15],
#      title ="Constant time-step dt=3 (asteroids)",
     title="asteroids dt=3",
     ylims=(MinE,MaxE),
     legend=false)

pe11=plot(sol0.t[2:end], qe2[2:5],
#      title ="Constant time-step dt=3 (inner bodies)",
     title="Inner dt=1.5",
     ylims=(MinE,MaxE),
     legend=false)

pe12=plot(sol0.t[2:end], qe2[6:10],
#      title ="Constant time-step dt=3 (outer bodies)",
```

```
     title="Outer dt=1.5",
     ylims=(MinE,MaxE),
     legend=false)

pe13=plot(sol0.t[2:end], qe2[11:15],
#      title ="Constant time-step dt=3 (asteroids)",
     title="asteroids dt=1.5",
     ylims=(MinE,MaxE),
     legend=false)


plot(pe1, pe2, pe3,
     pe11, pe12, pe13,
     layout=(2,3), size=(900,450))
```

[41]:



1.5.2 Adaptive step-size

[42]:

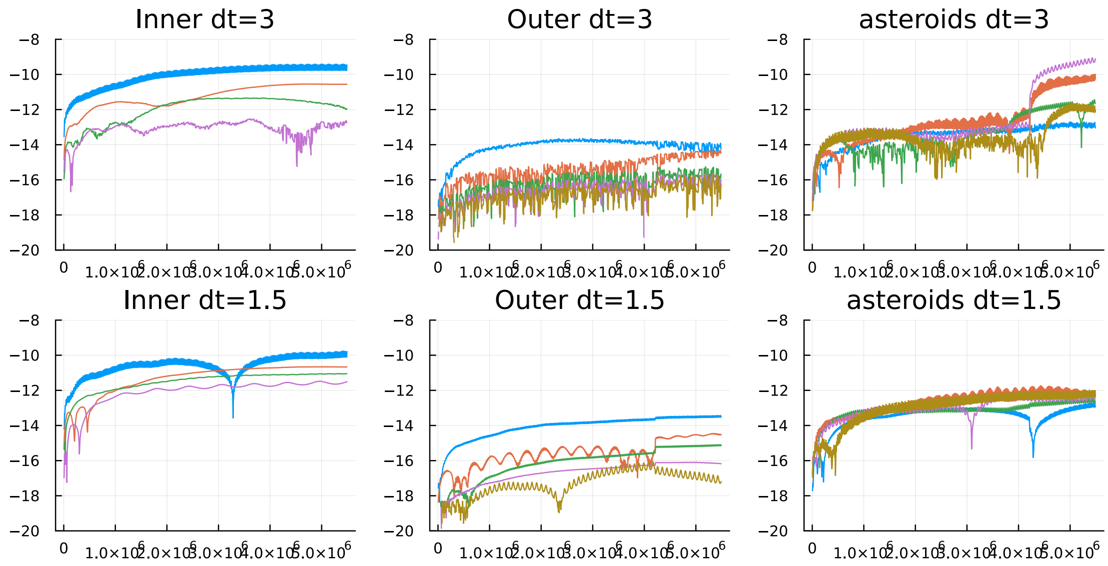
```
qe4=Array{Array{Float64,1}}(undef,N)
for i in 1:N
    qe4[i]=map((u0,u4)-> log10(norm(u0[:,i,2]-u4[:,i,2])), sol0.u[2:end],sol4.
 ↪u[2:end])
end


qe5=Array{Array{Float64,1}}(undef,N)
for i in 1:N
    qe5[i]=map((u0,u5)-> log10(norm(u0[:,i,2]-u5[:,i,2])), sol0.u[2:end],sol5.
 ↪u[2:end])
```

```
end
```

[46]:

```
MinE=-20
MaxE=-8


pe41=plot(sol0.t[2:end], qe4[2:5],
#     title ="Constant time-step dt=3 (inner bodies)",
     title="Inner dt=3",
     ylims=(MinE,MaxE),
     legend=false)

pe42=plot(sol0.t[2:end], qe4[6:10],
#     title ="Constant time-step dt=3 (outer bodies)",
     title="Outer dt=3",
     ylims=(MinE,MaxE),
     legend=false)

pe43=plot(sol0.t[2:end], qe4[11:15],
#     title ="Constant time-step dt=3 (asteroids)",
     title="asteroids dt=3",
     ylims=(MinE,MaxE),
     legend=false)

pe51=plot(sol0.t[2:end], qe5[2:5],
#     title ="Constant time-step dt=3 (inner bodies)",
     title="Inner dt=1.5",
     ylims=(MinE,MaxE),
     legend=false)

pe52=plot(sol0.t[2:end], qe5[6:10],
#     title ="Constant time-step dt=3 (outer bodies)",
     title="Outer dt=1.5",
     ylims=(MinE,MaxE),
     legend=false)

pe53=plot(sol0.t[2:end], qe5[11:15],
#     title ="Constant time-step dt=3 (asteroids)",
     title="asteroids dt=1.5",
     ylims=(MinE,MaxE),
     legend=false)

plot(pe41, pe42, pe43,
     pe51, pe52, pe53,
     layout=(2,3), size=(900,450))
```

[46]:



Inner dt=3
Outer dt=3
asteroids dt=3
Inner dt=1.5
Outer dt=1.5
asteroids dt=1.5
−8
−10
−12
−14
−16
−18
−20
0
1.0×10^6
2.0×10^6
3.0×10^6
4.0×10^6
5.0×10^6

[ ]: