PHYS 410: Computational Physics: Project 1 Key

My implementation includes the following source code files:

1. `toomre.m`: Main script.
2. `toomre_ct.m`: Function version of `toomre.m` for convergence testing two body orbit.
3. `t_toomre_ct.m`: Driver script for `toomre_ct.m`
4. `nbodyaccn.m`: Function that computes particle accelerations given current core positions.
5. `initcirc.m`: Function that computes initial positions and velocities for stars' circular orbits about cores.

`toomre.m`

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Toomre model of galaxy collisions ...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Script name ...
me = 'toomre';

more off;

% Number of cores ...
nc = 2
% Core masses ...
mc = [1 1]
% Number of stars initially orbiting respective cores ...
ns = [5000 5000]
% Total number of particles ...
np = nc + sum(ns)
% Particle masses ...
m = [mc, ones(1, sum(ns))];
% Masses for nbodyout (star mass = 1) ...
moutc = [1 1];
mout = [moutc, ones(1, sum(ns))];

% Discrete domain parameters ...
tmin = 0
tmax = 10.0
dt = 0.004
t = tmin:dt:tmax;
nt = length(t)
dt = t(2) - t(1)

% Output stride for nbodyout ...
os = 5;

% Tracing frequency ...
trace = 100;

% Initial core positions and velocities ...
% Search and destroy for interesting collision ...
rc0 = [[-0.75, -1.12, 0.0]; [0.75, 1.12, 0.0]; [0.0, 0.0, 0.0]];
vc0 = [[0.75, 0.0, 0.0]; [-0.75, 0.0, 0.0]; [0.0, 0.0, 0.0]];
% Sense of initial rotational velocities of stars:  1, -1 -> CW, CCW ...
vsense0 = [-1, -1];
```

```
% Radial limits for initial star distribution ...
rlim = [[0.05, 0.75]; [0.05, 0.75]; [0.05, 0.75] ];

% Position grid function ...
r = zeros(np, 3, nt);
% Initial velocity grid function ...
v0 = zeros(np, 3);

% First particle index for each group of stars ...
sind = zeros(1, nc+1);
sind(1) = nc + 1;
for ic = 2 : nc + 1
   sind(ic) = sind(ic-1) +  ns(ic-1) ;
end

% Initialize positions, velocities ...
if trace
   fprintf(’%s: Initializing first time step.\n’, me);
end
for ic = 1 : nc
   % Initialize core positions, velocities ...
   r(ic,:,1) = rc0(ic, :);
   v0(ic,:) = vc0(ic, :);
   % Compute star initial positions, velocities ...
   [rs0, vs0] = initcirc(ns(ic), mc(ic), rlim(ic,:), rc0(ic,:), ...
      vc0(ic,:), vsense0(ic));
   % ... and store them ...
   r(sind(ic):sind(ic+1)-1,:,1) = rs0;
   v0(sind(ic):sind(ic+1)-1,:) = vs0;
end
if trace
   fprintf(’%s: Initializing second time step.\n’, me);
end
r(:,:,2) = r(:,:,1) + dt * v0 + 0.5 * dt^2 * nbodyaccn(m, r(:,:,1), nc);

% Evolve system ...
if trace
   fprintf(’%s: Evolution begins.\n’, me);
end
tic
for it = 2 : nt - 1
   r(:,:,it+1) = 2 * r(:,:,it) - r(:,:,it-1) + ...
                 dt^2 * nbodyaccn(m, r(:,:,it),nc);
   if trace && rem(it,trace) == 0
      fprintf(’%s: Step %d of %d\n’, me, it + 1, nt);
   end
end
if trace
   fprintf(’%s: Evolution ends.\n’, me);
end
toc

% Colors for nbodyout ...
rgb = zeros(np, 3);
% Core colors ...
rgbc = [[1.0, 0.0, 0.0]; [0.0, 1.0, 0.0]; [0.0, 0.0, 1.0]];
```

```
% Star colors ...
rgbs = [[1.0, 1.0, 0.0]; [1.0, 0.0, 1.0]; [0.0, 1.0, 1.0]];
for ic = 1 : nc
   rgb(ic, :) = rgbc(ic, :);
   rgb(sind(ic):sind(ic+1)-1, :) = repmat(rgbs(ic, :), ns(ic), 1);
end

% Output data for subsequent visualization via xfpp3d ...
nbodyout('toomre.dat', t(1:os:nt), r(:,:,1:os:nt), mout, rgb);
```

`t_toomre_ct.m`

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% t_toomre_ct: Driver for toomre_ct ...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tmin = 0;
tmax = 8;

level_min = 10;
level_max = 13;

for level = level_min : level_max
   [t{level} r{level}] = toomre_ct(tmin, tmax, level);
end

% Compute and plot unscaled level-level deviations of x-coordinate
% of first particle ...
figure(1);
clf;
hold on;
for level = level_min : level_max - 1
   dx{level} = r{level+1}(1,1,1:2:end) - r{level}(1,1,:);
   size(dx{level})
   plot(t{level}, reshape(dx{level},1,length(t{level})));
end
title('Convergence of x-coordinate of first particle');
box on;
xlabel('t');
ylabel('x_{l+1} - x_{l}');
legend('l=10', 'l=11', 'l=12', 'Location', 'NorthWest');
print('x-convergence.eps','-depsc');

% Compute and plot scaled level-level deviations of x-coordinate
% of first particle ...
figure(2);
clf;
hold on;
for level = level_min : level_max - 1
   dx{level} = r{level+1}(1,1,1:2:end) - r{level}(1,1,:);
   size(dx{level})
   plot(t{level}, 4^(level - level_min) * reshape(dx{level},1,length(t{level})));
end
title('Scaled convergence of x_1');
box on;
xlabel('t');
ylabel('x_{l+1} - x_{l}');
legend('l=10', 'l=11', 'l=12', 'Location', 'NorthWest');
```

```
print('x-scaled-convergence.eps','-depsc');

% Compute and plot scaled convergence of energy conservation ...
figure(3);
clf;
hold on;
m = [1 0.5];
for level = level_min : level_max
   [T{level} V{level} Etot{level}] = calc_energy(m, t{level}, r{level});
   dEtot = Etot{level} - Etot{level}(1);
   % Exclude first and last points since extrapolation spoils convergence ...
   plot(t{level}(2:end-1), 4^(level - level_min) * dEtot(2:end-1));
end
title('Scaled convergence of \Delta E');
box on;
xlabel('t');
ylabel('E(t) - E(0)');
legend('l=10', 'l=11', 'l=12', 'l=13', 'Location', 'SouthEast');
print('E-scaled-convergence.eps','-depsc');
```

toomre_ct.m

```
function [t r] = toomre_ct(tmin, tmax, level)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% toomre_ct: Version of toomre for convergence testing.
%
% Hardwired for two cores, no particles.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

more off;

% Script name ...
me = 'toomre_ct';

% Tracing frequency ...
trace = 100;

% Discrete domain parameters ...
nt = 2^level + 1
t = linspace(tmin, tmax, nt);
dt = t(2) - t(1)

% Number of particles ...
np = 2;
% Core masses ...
m = [1 0.5];

% Initial conditions for mutual circular orbit ...

% Core separation ...
d = 1.0;
% Initial core positions ...
rc0 = [[-m(2) * d / (m(1) + m(2)), 0.0, 0.0];  ...
       [ m(1) * d / (m(1) + m(2)), 0.0, 0.0]];
% Initial core velocities ...
vc0 = [[0.0,  sqrt(m(2) * abs(rc0(1,1))) / d, 0.0];  ...
       [0.0, -sqrt(m(1) * abs(rc0(2,1))) / d, 0.0]];
```

```
% Position grid function ...
r = zeros(np, 3, nt);
% Initial velocity grid function ...
v0 = zeros(np, 3);

% Initialize positions, velocities ...
if trace
   fprintf('%s: Initializing first time step.\n', me);
end
for ip = 1 : np
   % Initialize core positions, velocities ...
   r(ip,:,1) = rc0(ip, :);
   v0(ip,:) = vc0(ip, :);
end
if trace
   fprintf('%s: Initializing second time step.\n', me);
end
r(:,:,2) = r(:,:,1) + dt * v0 + 0.5 * dt^2 * nbodyaccn(m, r(:,:,1), np);

% Evolve system ...
if trace
   fprintf('%s: Evolution begins.\n', me);
end
tic
for it = 2 : nt - 1
   r(:,:,it+1) = 2 * r(:,:,it) - r(:,:,it-1) + ...
                 dt^2 * nbodyaccn(m, r(:,:,it),np);
   if trace && rem(it,trace) == 0
      fprintf('%s: Step %d of %d\n', me, it + 1, nt);
   end
end
if trace
   fprintf('%s: Evolution ends.\n', me);
end
toc

% Colors for nbodyout ...
rgb = zeros(np, 3);
% Core colors ...
rgbc = [[1.0, 0.0, 0.0]; [0.0, 1.0, 0.0]; [0.0, 0.0, 1.0]];
for ip = 1 : np
   rgb(ip, :) = rgbc(ip, :);
end

% Output data for subsequent visualization via xfpp3d ...
os = 1;
nbodyout('toomre_ct.dat', t(1:os:nt), r(:,:,1:os:nt), m, rgb);

% end function
end
```

`nbodyaccn.m`

```
function [a] = nbodyaccn(m, r, ngrav)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% nbodyaccn: Computes acceleration for gravitational N-body problem
%            with ngrav massive, N - grav test particles.
%
% m:         particle masses
% r:         position array (nparticle x 3)
% ngrav:     number of gravitating particles (cores)
%
% Functions assumes that gravitating particles occupy positions 1:ngrav
% in the r array.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   np = length(m);

   a = nan * ones(np, 3);
   sz = size(r);
   % Check that dimensions of mass and r arrays are compatible ...
   if sz(1) ~= np
      fprintf('nbodyaccn: N(r)=%d ~= N(m)=%d\n', R, sz(1), np);
      return;
   end

   % Preallocate acceleration array ...
   a = zeros(np, 3);
   % For each particle ...
   for i = 1 : np
      % Compute acceleration components ...
      for j = 1 : ngrav
         if j ~= i
            rij = r(j,:) - r(i,:);
            magrij = sqrt(sum(rij .* rij));
            a(i,:) = a(i,:) + m(j) * rij ./ magrij^3;
         end
      end
   end

end
```

`initcirc.m`

```
function [r v] = initcirc(np, m, rlim, r0, v0, vs0)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  initcirc: Determines initial conditions for cicular orbits.
%
%  Inputs:
%
%     np:   Number of stars.
%     m:    Mass of core.
%     rlim: Minimum/maximum radii for stellar orbits [length-2 vector].
%     r0:   Position of core.
%     v0:   Velocity of core.
%     vs0:  Sign of rotation (clockwise/counterclockwise).
```

```
%
%  Ouputs:
%
%     r:    Positions of stars.
%     v:    Velocities of stars.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   r = zeros(np, 3);
   v = zeros(np, 3);
   rinit = linspace(rlim(1), rlim(2), np);
   thinit = 2 * pi * rand(1,np);

   for ip = 1 : np
      r(ip, 1) = r0(1) + rinit(ip) * cos(thinit(ip));
      r(ip, 2) = r0(2) + rinit(ip) * sin(thinit(ip));
      vip = sqrt(m / rinit(ip));
      v(ip, 1) = v0(1) + vs0 * vip * sin(thinit(ip));
      v(ip, 2) = v0(2) - vs0 * vip * cos(thinit(ip));
   end

end
```

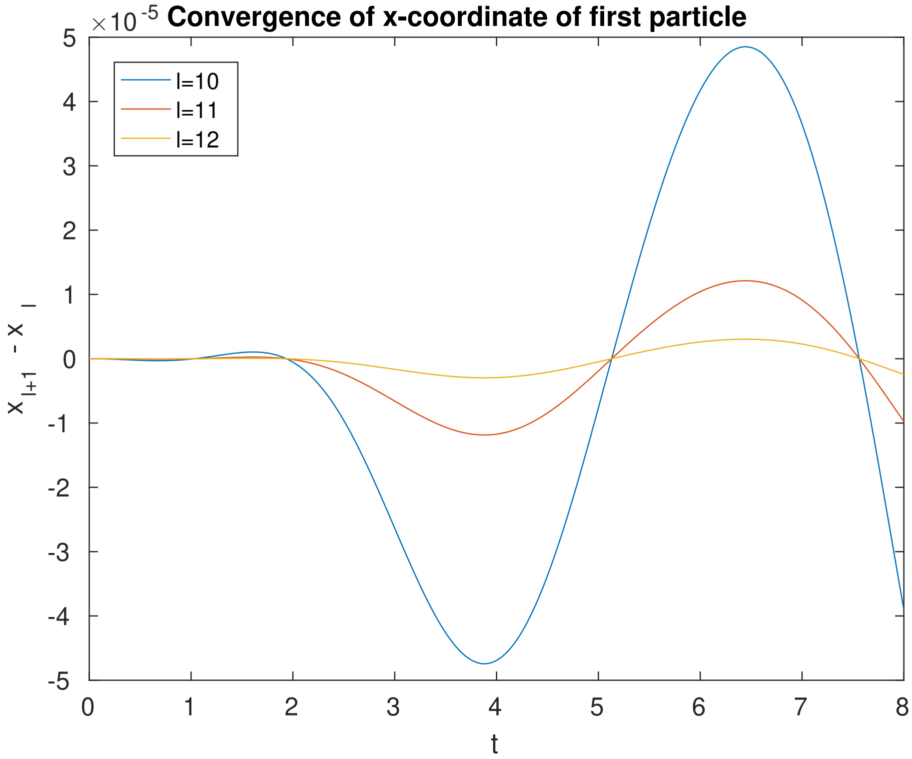

Figure 1: Unscaled convergence plot of x-coordinate of particle 1

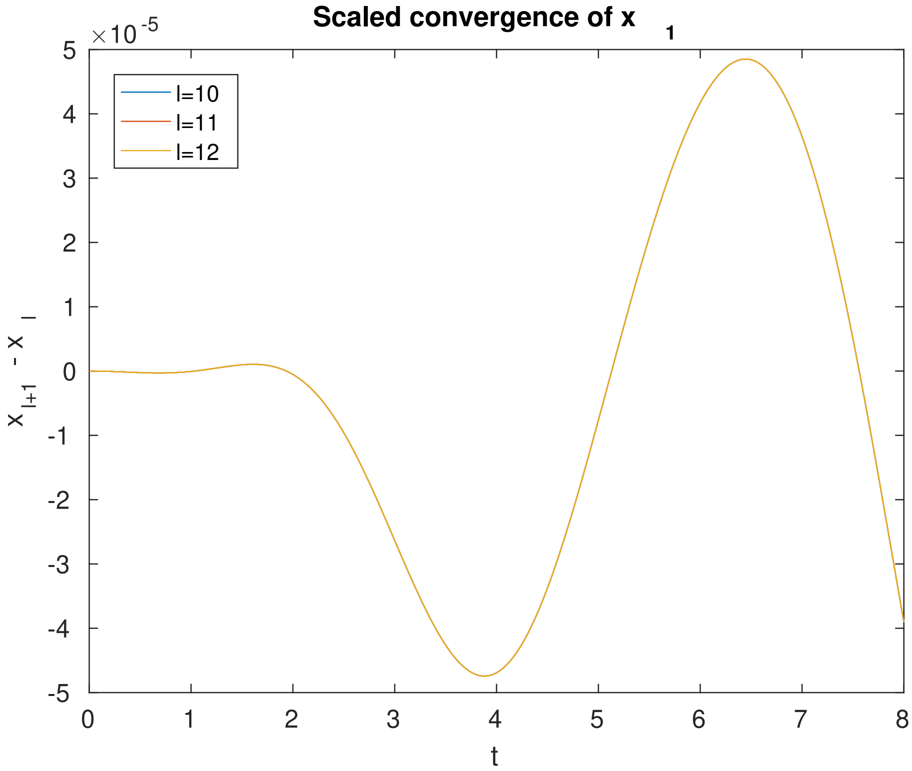

Figure 2: Scaled convergence plot of x-coordinate of particle 1

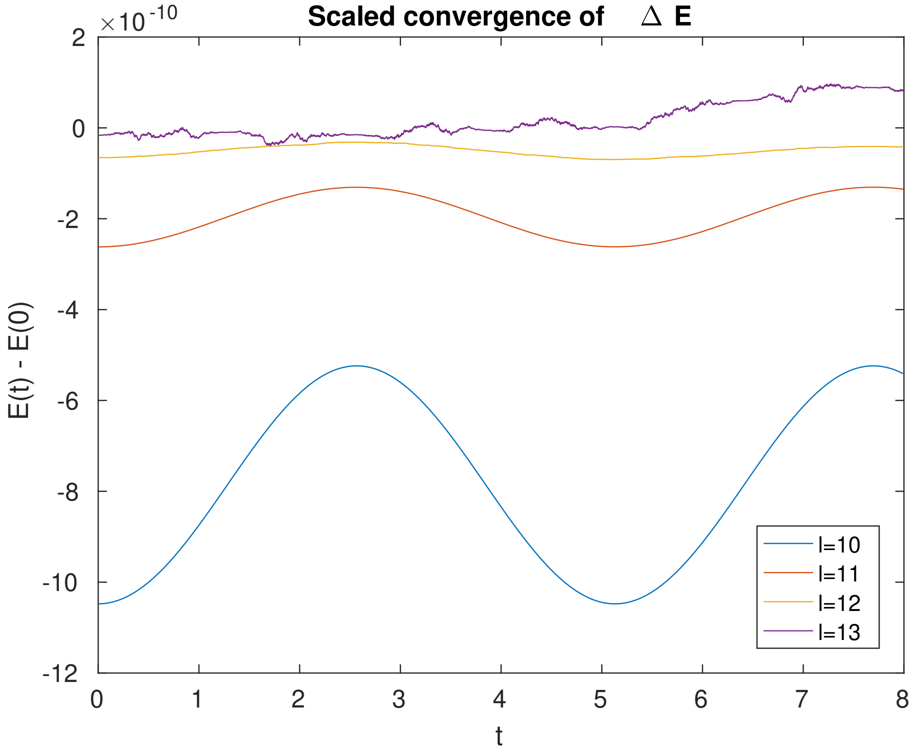

Figure 3: Scaled convergence plot of energy conservation for 2-body orbit