Non-linear Dynamics of a Double Pendulum and Parallel Computing

Rafael O. Soto

April 11, 2014
Phys-440
Illinois Institute of Technology, Chicago, IL 60616

Abstract

In this lab I implemented a simple ODE solver in Mathematica in order to study the chaotic dynamics of a double pendulum. I simulated the pendulum and produced phase space and bifurcation analysis. I also attempted to parallelize my code in order to study the power of parallelized computation.

1. Introduction

Chaotic systems are systems that are very sensitive to initial conditions. Small changes in these conditions yield vastly different behaviors. One such system is the double pendulum; a pendulum attaached to the bottom of a different one.

This system can be modeled using computation by figuring out it's lagrangian and solving for the equations of motions of ther respective pendulums. Much can be learned about the nature of these systems from simulation and this will be explored in this lab. In this lab I will use ODE(Ordinary Differential Equation) solving methods in order to try and produce trajectories of a chaotic double pendulum. I will also attempt to parallelize my code in Mathematica in order to study the strength of parallelization and test some theories behind it.

2. Analysis and Results

2.1. Trajectories

The system was set up in Mathematica and the constant variables were initialized. The 2-D lagrangian was determined, in polar coordinates ans so were the equations of motion. They are labelled on the attached mathematica notebook and can be found in A Survey of Computational Physics [1].

Mathematica's NDSolve was used to calculate a solution with a 40 second timestep. Multiple trajectories were composed and plotted. An animation was also made in order to view the chaotic nature of the pendulum in real-time.

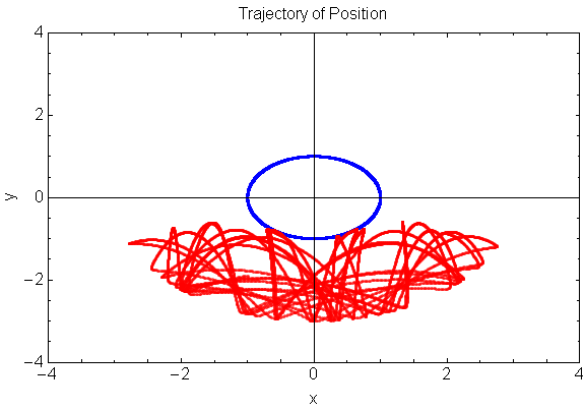


Figure 1: Sample Trajectory 1.

2.2. Phase Space Diagrams

A solution was composed to solve for the phase space trajectories. The trajectory for each pendulum was plotted as the angular velocity versus the angular position.

2.3. Bifurcation

An attempt was made to create a bifurcation diagram of the system. This attempt included readjusting the system so that m[1]=10*m[2] thereby making the system depend on a sole variable. The bifurcation diagram would then be produced by setting a specific m[1] and creating 70 trajectories. These trajectories would be analyzed for an occurence of Θ_1 being less than 1(Degree) and the angular velocity at that moment would be added to a list specific to that value of m[1]. This process would be repeated for 100 different values of

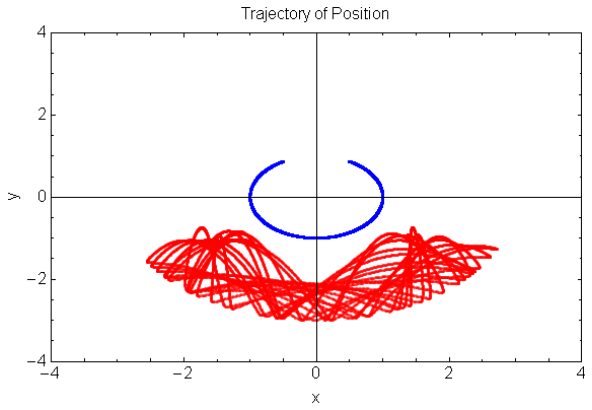


Figure 2: Sample Trajectory 2.

m[1] (1...100 kg). The angular velocities at equilibrium gathered would then all be plotted collectively as a function of m[1]. The pattern is supposed to contain hints of fractal behavior for different values of m[1] and can be used to charactherize the degree of chaos of the system at a certain point. Unfortunately, the bifurcation diagram produced did not show me signs of fractal behavior though I believe that I did see a few different modes of behavior. It also showed that there is a large degree of randomness in the system. Fig. 6 was calculated by varying m[1] from .1 to 40 in steps of .1, while m[2]=10*m[1].

2.4. Serial vs. Parallel Processing

The speed-up gained from running code in parallel using similar processors is estimaded by Amdahl's law as:

$$S_p = \frac{1}{1 - f + f/p} \tag{1}$$

where f is the fraction of the program being parallelized and p is the number of processors running in parallel. Since processes in parallel are indipendent of eachother the bifurcation anaylisis of the code can indeed be parallelized. This was done using ParallelTable in mathematica to generate trajectories for each area of interest. Timing measurements were done using Mathematica's Timing[] function and the result of parallelization were plotted along with the prediction from Amdahl's law. Fig 6 is a plot of Amdahl's law for f=80%, though I am uncertain of the degree of parallelization of mathematica's inner code I assumed it was very high. The data that I received from parallelizing the bifurcation diagram agreed very well with Amdahl's law, and it would be interesting to see the behavior if my computer had more cores.

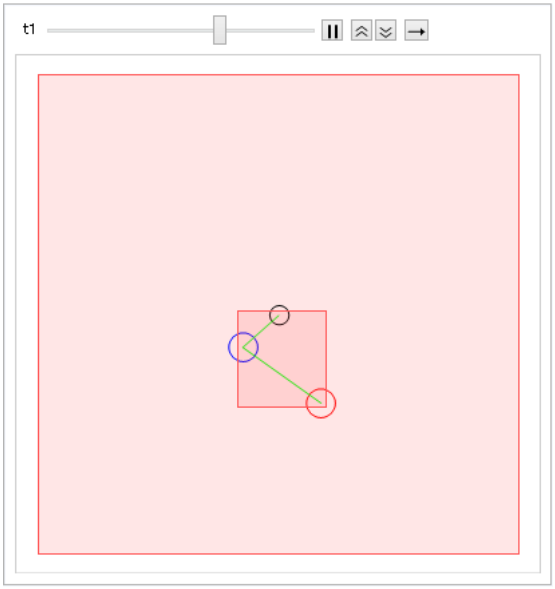


Figure 3: Paused Animation (Couldn't get rid of error-box, but the animation works fine.)

3. Discussion

The results of the lab agreed well with what I expected, except for the bifurcation. While I was able to make really nice plots of my data, and easy solve ODE's with mathematica it was a bit more involved to do other things. The bifurcation, for example, took much longer than I had anticipated and it comes from mathematica's lack of useful error messages. However, the parallelization part was very straightforward and I was surprised by the results. It'd be interesting to see how well Mathematica is at mimicing Amdahl's law on a more powerful computer or cluster.

[1] Rubin, H. Landau, Manuel Jose Paez, Cristian C. Bordeianu *A Survey of Computational Physics* 2012: Princeton University Press.

[2] Python Software Foundation *Python v2.7.6 documentation* 1990-2014

[3] The People of Stack Exchange, Google, *Internet.*

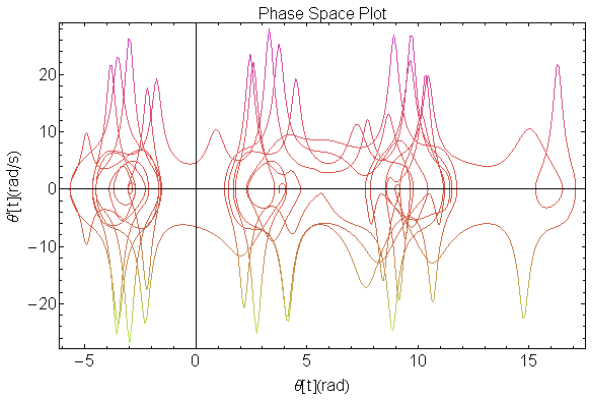


Figure 4: Phase Space Diagram Pendulum 1.

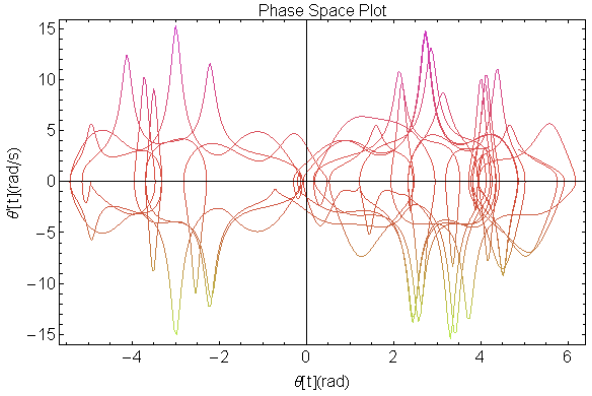


Figure 5: Phase Space Diagram Pendulum 2.

Figure 6: Bifurcation diagram for the double pendulum attached with mathematica documentation at the end.

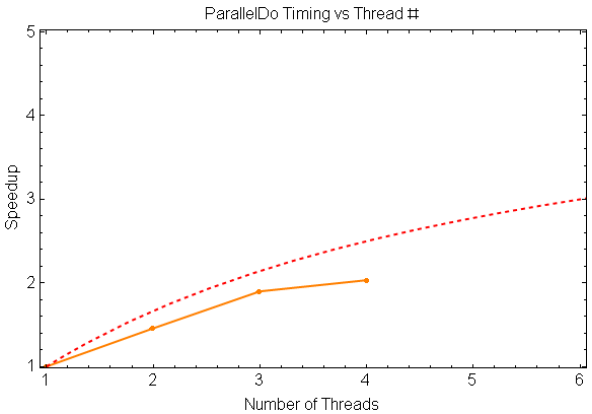


Figure 7: Plot of Amdahl's Law vs Speedup Data .

Double Pendulum

In[21]:=

```
n = 2; tmax = 40; g = -10; MomentInertia = 1000;
Do[(m[v] = 1; length[v] = v; mi[v] = 1), {v, 1, n}];
m[1] = 1;
m[2] = 10;
(*x[n_][t_]:=Sum[length[v]*Sin[θ[1][t]]*Cos[θ[2][t]],{v,1,n}];
y[n_][t_]:=Sum[length[v]*Sin[θ[1][t]]*Sin[θ[2][t]],{v,1,n}];*)
x[n_][t_] := Sum[length[v] * Sin[θ[v][t]], {v, 1, n}];
y[n_][t_] := Sum[length[v] * Cos[θ[v][t]], {v, 1, n}];
L = .5 (m[1] + m[2]) * (length[1]) ^2 * (θ[1]'[t]) ^2 +
     .5 * m[2] * length[2] ^2 * (θ[2]'[t]) ^2 + m[2] * length[1] * length[2] * θ[1]'[t] *
      θ[2]'[t] * Cos[θ[1][t] - θ[2][t]] + (m[1] + m[2]) * g * length[1] * Cos[θ[1][t]] +
     m[2] * g * length[2] * Cos[θ[2][t]] // Simplify;
TexForm[L = .5 (m[1] + m[2]) * (length[1]) ^2 * (θ[1]'[t]) ^2 +
      .5 * m[2] * length[2] ^2 * (θ[2]'[t]) ^2 + m[2] * length[1] * length[2] * θ[1]'[t] *
       θ[2]'[t] * Cos[θ[1][t] - θ[2][t]] + (m[1] + m[2]) * g * length[1] * Cos[θ[1][t]] +
      m[2] * g * length[2] * Cos[θ[2][t]] // Simplify;]
(*eqns1=ParallelTable[D[L,ϕ[v][t]]==D[D[L,ϕ[v]'[t]],t],{v,1,n}]//FullSimplify;*)
eqns2 =
  ParallelTable[D[L, θ[v][t]] == D[D[L, θ[v]'[t]], t], {v, 1, n}] // FullSimplify;
(*vars=Table[ϕ[i][t],{i,1,n}];
ics=Join[Thread[(D[vars,t]/.{t→0})==ParallelTable[π Random[],{n}]],
  Thread[(vars/.{t→0})==ParallelTable[π Random[ ],{n}]]];*)
vars2 = ParallelTable[θ[i][t], {i, 1, n}];
ics2 = Join[Thread[(D[vars2, t] /. {t → 0}) == ParallelTable[π Random[], {n}]],
   Thread[(vars2 /. {t → 0}) == ParallelTable[π Random[ ], {n}]]];
```

Out[28]= `TexForm[Null]`

In[12]:=
```
L // TraditionalForm
```

Out[12]//TraditionalForm=

$$5.5\,(\theta(1)'(t))^2 + 20.\,(\theta(2)'(t))^2 + 20.\,\theta(2)'(t)\,\theta(1)'(t)\cos(\theta(1)(t) - \theta(2)(t)) - 110.\cos(\theta(1)(t)) - 200.\cos(\theta(2)(t))$$

In[32]:=
```
Parameters = Join[eqns2, ics2];
```

In[33]:=
```
Equations = vars2;
```

In[34]:=
```
soln = NDSolve[Parameters, Equations, {t, 1, tmax}, AccuracyGoal → 6,
     MaxSteps → Infinity, Method -> {"EquationSimplification" -> "Solve"}][[1]];
```

In[35]:=
```
trajectory[v_][t1_] := {x[v][t], y[v][t]} /. soln /. {t → t1}
```

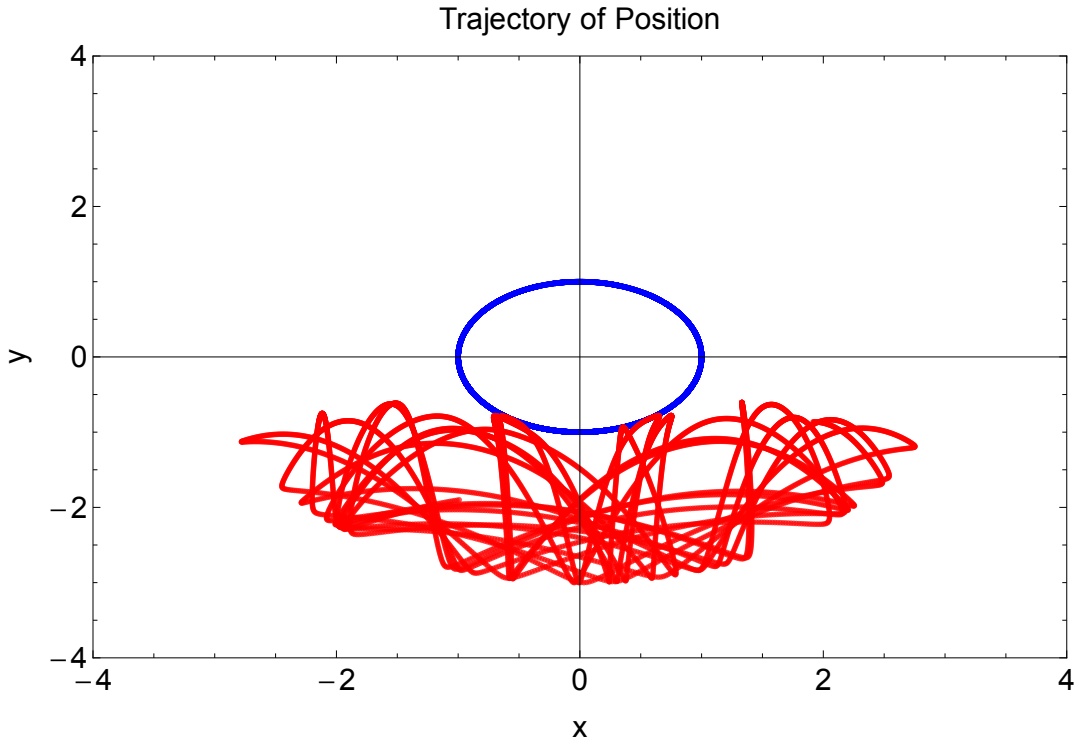
In[36]:=
```
trajectories[t1_] := ParallelTable[{trajectory[i][t1]}, {i, 1, n}]
```

In[37]:=
```
Do[data[v] = Flatten[ParallelTable[{trajectory[v][t]}, {t, 1, tmax, .004}], 1],
  {v, 1, n}];
```

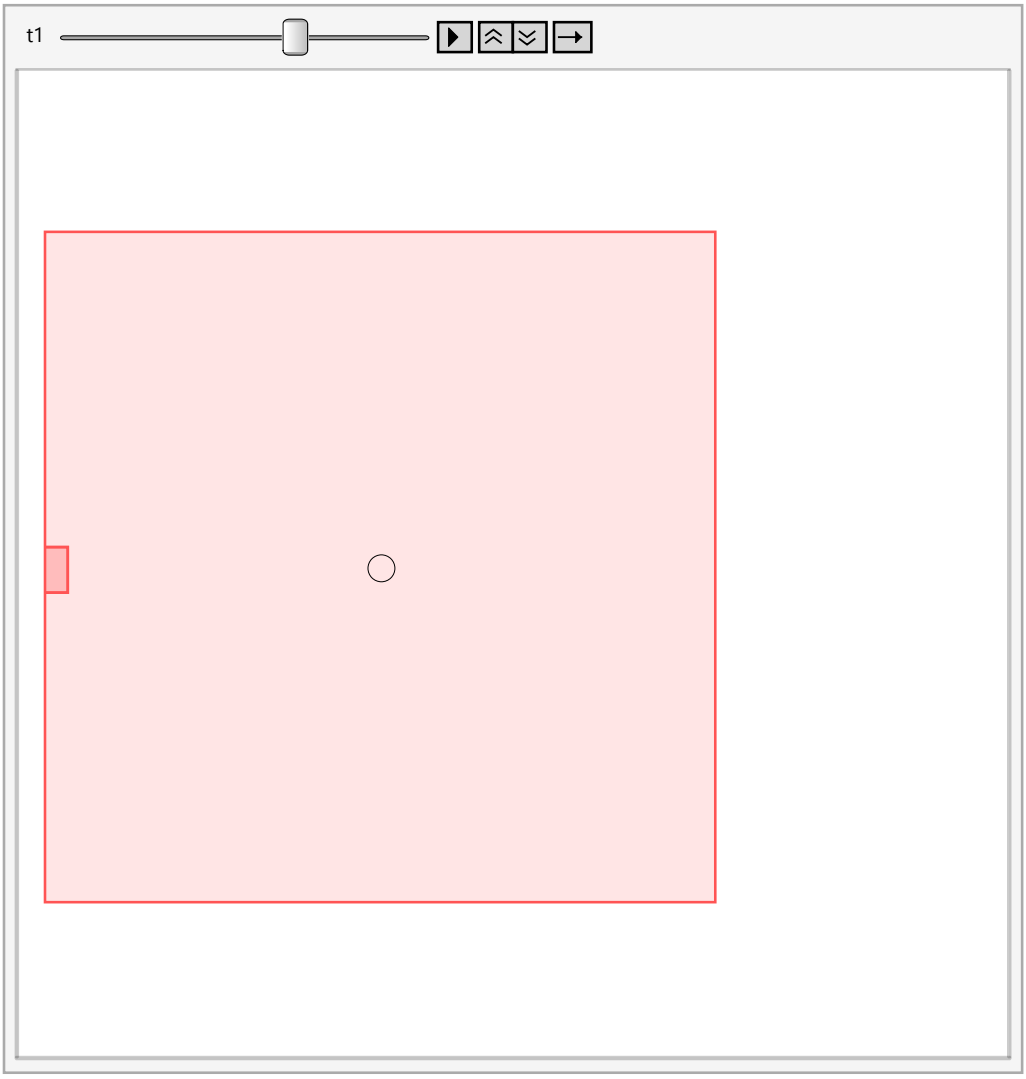
In[38]:=
```
Datas[t1_] := Table[{data[v]}, {v, 1, n}];
```

In[39]:=
```
ListPlot[{data[1], data[2]}, PlotStyle → {{Opacity[1, Blue]},
   {Opacity[.7], Red}, {Opacity[.6], Green}, {Opacity[.8], Purple}}, PlotLabel →
  Style["Trajectory of Position", FontFamily → "Helvetica", FontSize → 16],
 Frame → True, FrameTicks → Automatic, FrameLabel → {"x", "y"},
 LabelStyle → {FontFamily → "Helvetica", FontSize → 16},
 ImageSize → Large, PlotRange → {{-4, 4}, {-4, 4}}]
```

Out[39]=



```
Animate[Graphics[{Black, Circle[{0, 0}, .2], Blue,
   Circle[trajectory[1][t1], .3], Red, Circle[trajectory[2][t1], .3],
   Green, Line[{{0, 0}, trajectory[1][t1], trajectory[2][t1]}, .03]},
  PlotRange → {{-5, 5}, {-5, 5}}], {t1, 0, 10}]
```



Phase Space Diagrams

In[22]:=
```
PhaseEquations = {θ[1][t], θ[1]'[t]};
```

In[21]:=
```
PhaseEquations2 = {θ[2][t], θ[2]'[t]};
```

In[23]:=
```
phasesoln = NDSolve[Parameters, PhaseEquations, {t, 1, tmax}, AccuracyGoal → 10,
     MaxSteps → Infinity, Method -> {"EquationSimplification" -> "Solve"}][[1]];
```

In[24]:=
```
phasesoln2 = NDSolve[Parameters, PhaseEquations2, {t, 1, tmax}, AccuracyGoal → 10,
     MaxSteps → Infinity, Method -> {"EquationSimplification" -> "Solve"}][[1]];
```

In[25]:=
```
phasespace[1][t1_] := {θ[1][t], θ[1]'[t]} /. phasesoln /. {t → t1}
```

In[26]:=
```
phasespace[2][t1_] := {θ[2][t], θ[2]'[t]} /. phasesoln2 /. {t → t1}
```

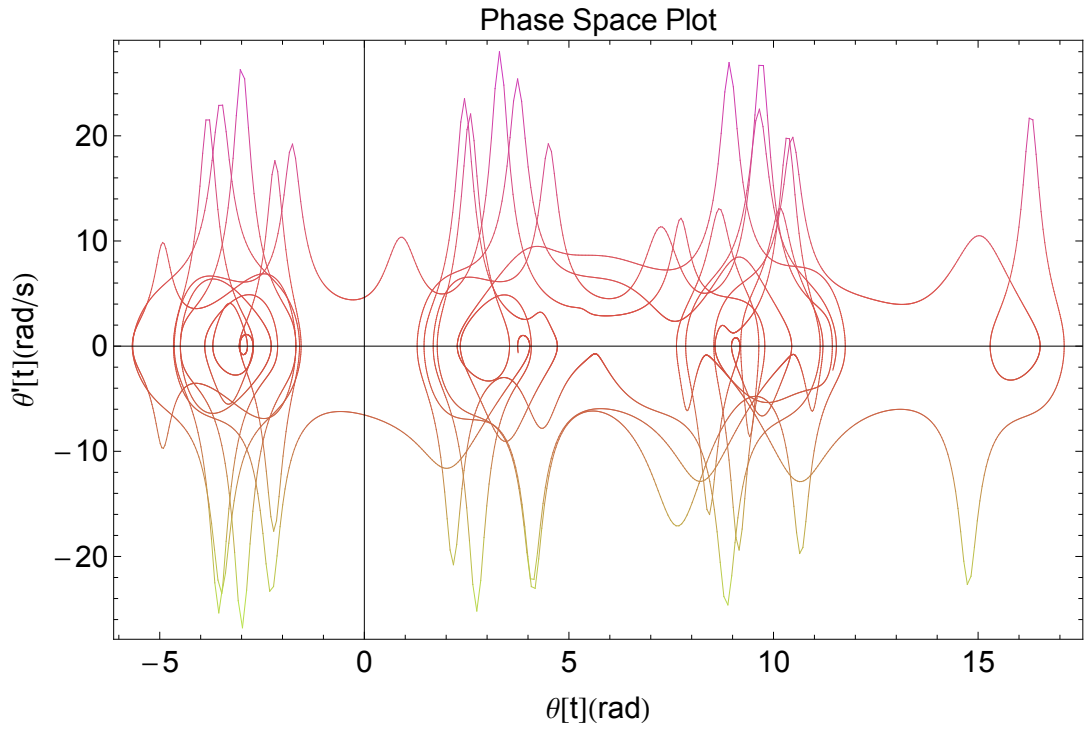
In[27]:=
```
phasedata[1] = Flatten[ParallelTable[{phasespace[1][t]}, {t, 1, tmax, .004}], 1];
```

In[28]:=
```
phasedata[2] = Flatten[ParallelTable[{phasespace[2][t]}, {t, 1, tmax, .004}], 1];
```

In[29]:=
```
PhaseDatas[t1_] := Table[{phasedata[v]}, {v, 1, n}];
```

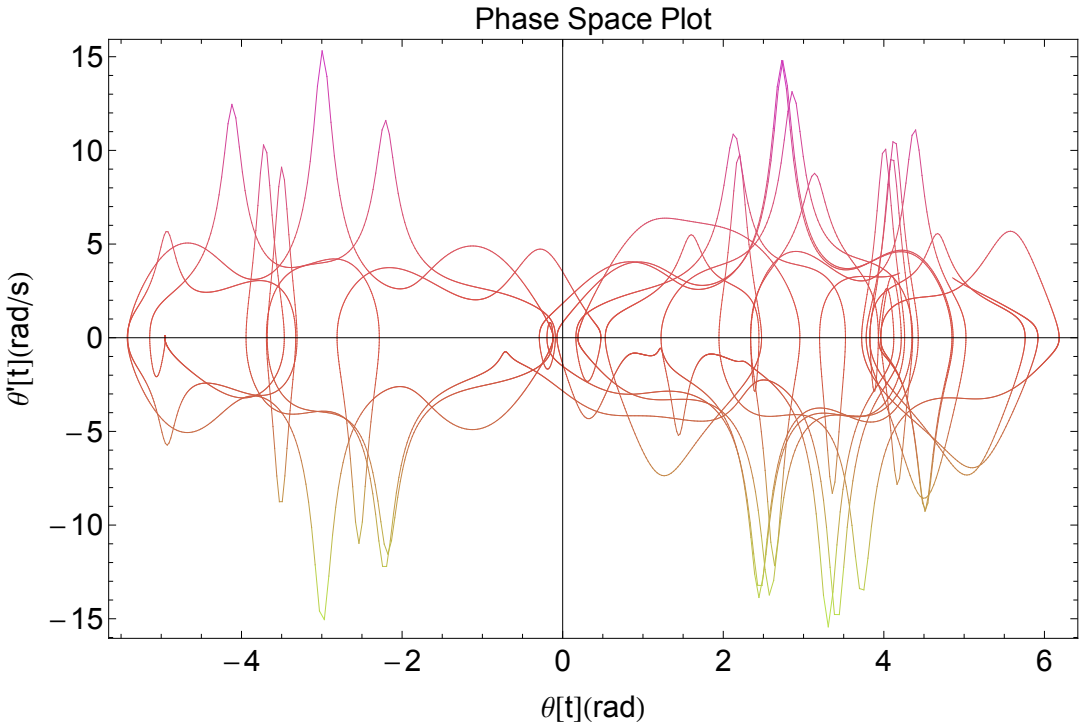
In[30]:=
```
ListPlot[{phasedata[1]},
 PlotStyle → {{Opacity[1, Blue]}}, ImageSize → Large, Joined → True,
 PlotLabel → Style["Phase Space Plot", FontFamily → "Helvetica", FontSize → 16],
 Frame → True, FrameTicks → Automatic, FrameLabel → {"θ[t](rad)", "θ'[t](rad/s)"},
 LabelStyle → {FontFamily → "Helvetica", FontSize → 16},
 ColorFunction → Function[{t, y}, ColorData["NeonColors"][y]]]
```

Out[30]=



In[31]:=
```
ListPlot[{phasedata[2]},
 PlotStyle → {{Opacity[1, Blue]}}, ImageSize → Large, Joined → True,
 PlotLabel → Style["Phase Space Plot", FontFamily → "Helvetica", FontSize → 16],
 Frame → True, FrameTicks → Automatic, FrameLabel → {"θ[t](rad)", "θ'[t](rad/s)"},
 LabelStyle → {FontFamily → "Helvetica", FontSize → 16},
 ColorFunction → Function[{t, y}, ColorData["NeonColors"][y]]]
```

Out[31]=



Bifurcation Analysis

In[1]:=
```
CloseKernels[];
LaunchKernels[4];
```

In[3]:=
```
Do[(result[v] = {}), {v, 1, 10}]
```

In[4]:=
```
Timing[Do[n = 2; tmax = 30; g = -10;
  m[1] = z * .1;
  m[2] = 10 * m[1];
  length[1] = 1;
  length[2] = 1;
  x[n_][t_] := Sum[length[v] * Sin[θ[v][t]], {v, 1, n}];
  y[n_][t_] := Sum[length[v] * Cos[θ[v][t]], {v, 1, n}];
  L = .5 (m[1] + m[2]) * (length[1])^2 * (θ[1]'[t])^2 +
      .5 * m[2] * length[2]^2 * (θ[2]'[t])^2 + m[2] * length[1] * length[2] * θ[1]'[t] *
       θ[2]'[t] * Cos[θ[1][t] - θ[2][t]] + (m[1] + m[2]) * g * length[1] * Cos[θ[1][t]] +
      m[2] * g * length[2] * Cos[θ[2][t]] // Simplify;
  eqns = Table[D[L, θ[v][t]] == D[D[L, θ[v]'[t]], t], {v, 1, n}] // FullSimplify;
  vars = Table[θ[i][t], {i, 1, n}];
  Equations2 = vars;

 PhaseEquations2 = {θ[2][t], θ[2]'[t]};
  Do[
   ics = Join[Thread[(D[vars, t] /. {t → 0}) == Table[π (2 * Random[] - 1), {n}]],
     Thread[(vars /. {t → 0}) == Table[π (2 * Random[ ] - 1), {n}]]];
    Parameters = Join[eqns, ics];
   phasesoln2 = NDSolve[Parameters, PhaseEquations2, {t, 1, tmax}, AccuracyGoal → 10,
      MaxSteps → Infinity, Method -> {"EquationSimplification" -> "Solve"}][[1]];
 bifurcate[t1_] := {θ[2][t], θ[2]'[t]} /. phasesoln2 /. {t → t1};
 bifurcatetable[w] = Flatten[Table[{bifurcate[t]}, {t, 1, tmax, .1}], 1];
   bitranspose[w] = Transpose[bifurcatetable[w]][[1]];
 index[w] = Position[bitranspose[w], _?(Abs[#] < 3 &)];
 If[TrueQ[index[w] == {}], Print["Noindex"], AppendTo[result[z],
     Transpose[bifurcatetable[w]][[2]][[Flatten[index[w][[1]]]]]]];
   Clear[phasesoln2];, {w, 1, 70}], {z, 1, 200}]]
```

Out[4]=
```
{1271.500000, Null}
```

```
Flatten[Reap[For[i = 1, i < 140, i++, Sow[{1}]; Sow[result[1][[2 ;;]][[i + 1]]];]][[
    2, 1]]]~Partition~2;
```

In[8]:=
```
s = Table[Flatten[
    Reap[For[i = 1, i < 63, i++, Sow[{x}]; Sow[result[x][[2 ;;]][[i + 1]]];]][[
     2, 1]]] ~ Partition ~ 2, {x, 1, 199}];
```

Part::partw : Part 62 of
{{−1.53026}, {−0.643518}, {−3.09313}, {−2.47001}, {5.19067}, {−3.85635}, ≪39≫, {−1.25126}, {−6.54208}, {1.30263}, {0.823417}, {−4.51541}, ≪11≫} does not exist. ≫

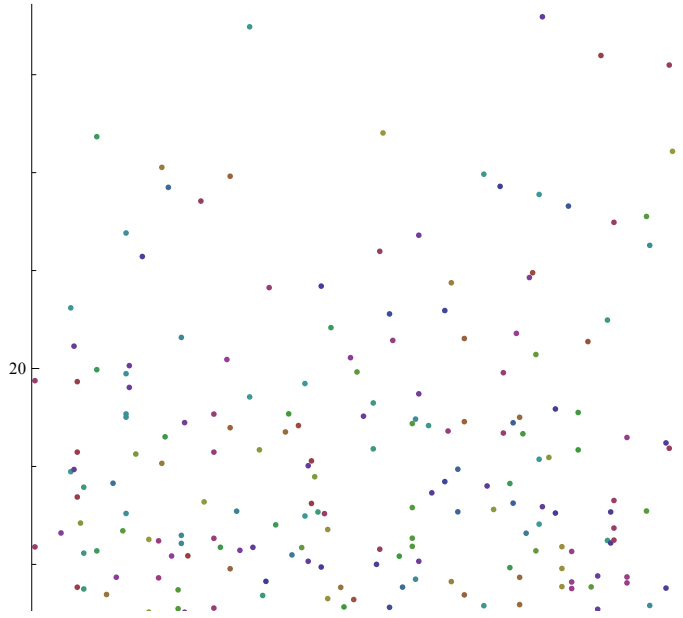
Part::partw : Part 63 of
{{−1.53026}, {−0.643518}, {−3.09313}, {−2.47001}, {5.19067}, {−3.85635}, ≪39≫, {−1.25126}, {−6.54208}, {1.30263}, {0.823417}, {−4.51541}, ≪11≫} does not exist. ≫

Part::partw : Part 63 of
result[{−4.57178}, {3.66335}, {−5.70765}, {4.44668}, {−1.21239}, {1.56353}, ≪40≫, {−4.7524}, {3.95264}, {−7.42343}, {6.5565}, ≪12≫] does not exist. ≫

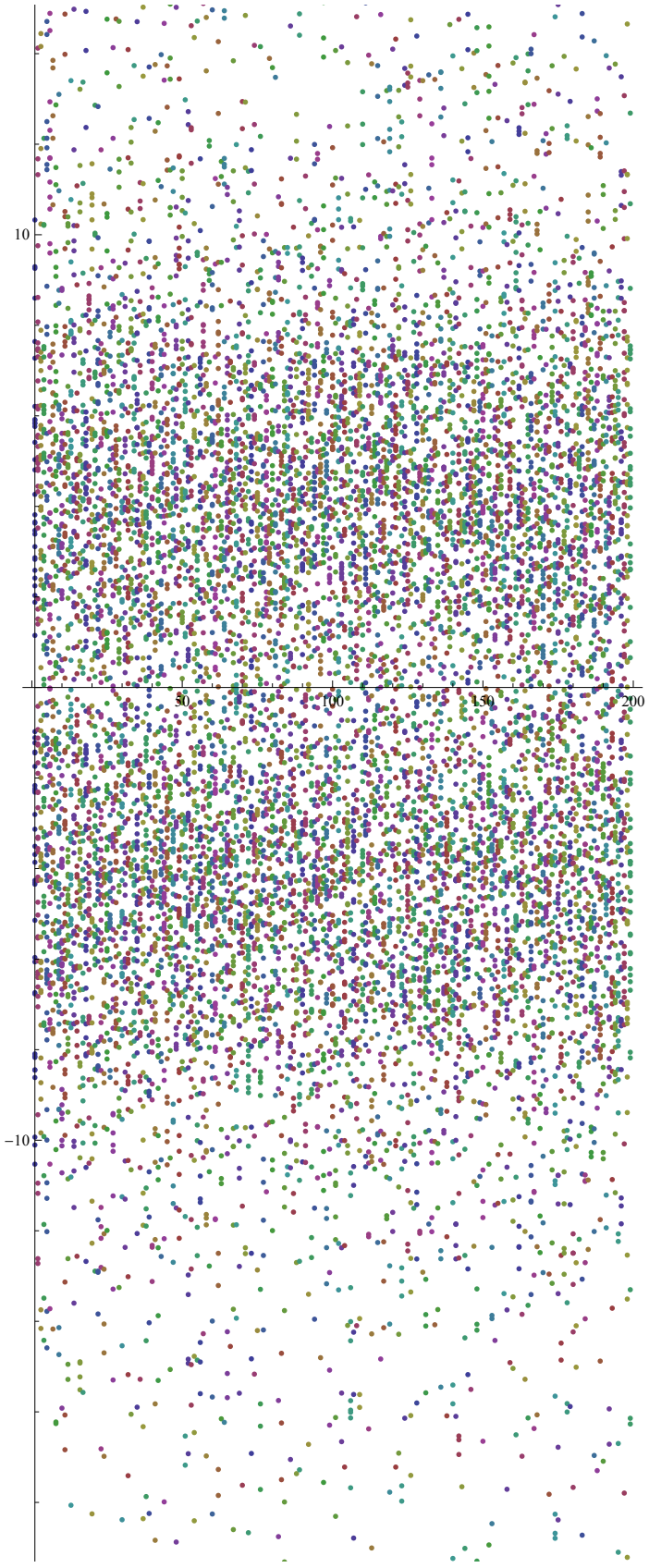
General::stop : Further output of Part::partw will be suppressed during this calculation. ≫

Out[9]=
```
{{100, 6.29985}, {100, 1.47871}, {100, 3.76868}, {100, 7.18192}, {100, 1.14167},
 {100, -5.90108}, {100, -7.7835}, {100, -3.0741}, {100, -6.26377}, {100, 3.09381},
 {100, 10.2141}, {100, 9.87641}, {100, 3.93093}, {100, 2.10086}, {100, -4.09499},
 {100, -6.77014}, {100, 4.76139}, {100, 8.17746}, {100, 5.26546}, {100, 6.62347},
 {100, -4.41956}, {100, 15.2819}, {100, 5.45177}, {100, 7.78153}, {100, -2.11846},
 {100, -5.75901}, {100, 5.43238}, {100, 5.78707}, {100, -7.02146}, {100, 5.59564},
 {100, 11.0995}, {100, -6.58928}, {100, 1.46833}, {100, -6.99246}, {100, 5.97964},
 {100, -9.2501}, {100, -5.17018}, {100, -3.51141}, {100, -2.28285},
 {100, -2.24335}, {100, -3.74617}, {100, 3.80707}, {100, 8.28921},
 {100, -5.03327}, {100, -0.41226}, {100, -0.0364799}, {100, 11.9234},
 {100, -4.0426}, {100, -7.00474}, {100, 6.83363}, {100, -3.85125}, {100, 5.32012},
 {100, 6.64908}, {100, -6.76512}, {100, 6.90769}, {100, -4.2929}, {100, -6.99556},
 {100, -6.6189}, {100, 5.2551}, {100, 5.27894}, {100, -9.47888}, {100, -8.16667}}
```

In[10]:=
```
ListPlot[s, ImageSize → Large, PlotRange → All, AspectRatio → 4]
```



Out[10]=



Parallelization

In[7]:=
```
Do[Pause[1]; f[i], {i, 4}] // AbsoluteTiming
ParallelDo[Pause[1]; f[i], {i, 4}] // AbsoluteTiming
```

Out[7]= {4.005672, Null}

Out[8]= {2.952973, Null}

In[23]:=
```
Do[n = 2; tmax = 30; g = -10;
  m[1] = z * .1;
  m[2] = 10 * m[1];
  length[1] = 1;
  length[2] = 1;
  x[n_][t_] := Sum[length[v] * Sin[θ[v][t]], {v, 1, n}];
  y[n_][t_] := Sum[length[v] * Cos[θ[v][t]], {v, 1, n}];
  L = .5 (m[1] + m[2]) * (length[1])^2 * (θ[1]'[t])^2 +
      .5 * m[2] * length[2]^2 * (θ[2]'[t])^2 + m[2] * length[1] * length[2] * θ[1]'[t] *
       θ[2]'[t] * Cos[θ[1][t] - θ[2][t]] + (m[1] + m[2]) * g * length[1] * Cos[θ[1][t]] +
      m[2] * g * length[2] * Cos[θ[2][t]] // Simplify;
  eqns = Table[D[L, θ[v][t]] == D[D[L, θ[v]'[t]], t], {v, 1, n}] // FullSimplify;
  vars = Table[θ[i][t], {i, 1, n}];
  Equations2 = vars;

 PhaseEquations2 = {θ[2][t], θ[2]'[t]};
  Do[
   ics = Join[Thread[(D[vars, t] /. {t → 0}) == Table[π (2 * Random[] - 1), {n}]],
     Thread[(vars /. {t → 0}) == Table[π (2 * Random[ ] - 1), {n}]]];
    Parameters = Join[eqns, ics];
   phasesoln2 = NDSolve[Parameters, PhaseEquations2, {t, 1, tmax}, AccuracyGoal → 10,
      MaxSteps → Infinity, Method -> {"EquationSimplification" -> "Solve"}][[1]];
bifurcate[t1_] := {θ[2][t], θ[2]'[t]} /. phasesoln2 /. {t → t1};
bifurcatetable[w] = Flatten[Table[{bifurcate[t]}, {t, 1, tmax, .1}], 1];
   bitranspose[w] = Transpose[bifurcatetable[w]][[1]];
index[w] = Position[bitranspose[w], _?(Abs[#] < 3 &)];
If[TrueQ[index[w] == {}], Print["Noindex"], AppendTo[result[z],
     Transpose[bifurcatetable[w]][[2]][[Flatten[index[w][[1]]]]]]];
   Clear[phasesoln2];, {w, 1, 70}], {z, 1, 8}] // AbsoluteTiming
```

Out[23]=
```
{54.632450, Null}
```

In[24]:=
```
CloseKernels[];
LaunchKernels[2];
```

In[26]:=
```
ParallelDo[n = 2; tmax = 30; g = -10;
  m[1] = z * .1;
  m[2] = 10 * m[1];
  length[1] = 1;
  length[2] = 1;
  x[n_][t_] := Sum[length[v] * Sin[θ[v][t]], {v, 1, n}];
  y[n_][t_] := Sum[length[v] * Cos[θ[v][t]], {v, 1, n}];
  L = .5 (m[1] + m[2]) * (length[1])^2 * (θ[1]'[t])^2 +
      .5 * m[2] * length[2]^2 * (θ[2]'[t])^2 + m[2] * length[1] * length[2] * θ[1]'[t] *
       θ[2]'[t] * Cos[θ[1][t] - θ[2][t]] + (m[1] + m[2]) * g * length[1] * Cos[θ[1][t]] +
      m[2] * g * length[2] * Cos[θ[2][t]] // Simplify;
  eqns = Table[D[L, θ[v][t]] == D[D[L, θ[v]'[t]], t], {v, 1, n}] // FullSimplify;
  vars = Table[θ[i][t], {i, 1, n}];
  Equations2 = vars;

 PhaseEquations2 = {θ[2][t], θ[2]'[t]};
  Do[
   ics = Join[Thread[(D[vars, t] /. {t → 0}) == Table[π (2 * Random[] - 1), {n}]],
     Thread[(vars /. {t → 0}) == Table[π (2 * Random[ ] - 1), {n}]]];
    Parameters = Join[eqns, ics];
   phasesoln2 = NDSolve[Parameters, PhaseEquations2, {t, 1, tmax}, AccuracyGoal → 10,
      MaxSteps → Infinity, Method -> {"EquationSimplification" -> "Solve"}][[1]];
bifurcate[t1_] := {θ[2][t], θ[2]'[t]} /. phasesoln2 /. {t → t1};
bifurcatetable[w] = Flatten[Table[{bifurcate[t]}, {t, 1, tmax, .1}], 1];
   bitranspose[w] = Transpose[bifurcatetable[w]][[1]];
index[w] = Position[bitranspose[w], _?(Abs[#] < 3 &)];
If[TrueQ[index[w] == {}], Print["Noindex"], AppendTo[result[z],
     Transpose[bifurcatetable[w]][[2]][[Flatten[index[w][[1]]]]]]];
   Clear[phasesoln2];, {w, 1, 70}], {z, 1, 8}] // AbsoluteTiming

1 {, Null}
```

In[33]:=
```
CloseKernels[];
LaunchKernels[4];
```

In[37]:=
```
ParallelDo[n = 2; tmax = 30; g = -10;
  m[1] = z * .1;
  m[2] = 10 * m[1];
  length[1] = 1;
  length[2] = 1;
  x[n_][t_] := Sum[length[v] * Sin[θ[v][t]], {v, 1, n}];
  y[n_][t_] := Sum[length[v] * Cos[θ[v][t]], {v, 1, n}];
  L = .5 (m[1] + m[2]) * (length[1])^2 * (θ[1]'[t])^2 +
      .5 * m[2] * length[2]^2 * (θ[2]'[t])^2 + m[2] * length[1] * length[2] * θ[1]'[t] *
       θ[2]'[t] * Cos[θ[1][t] - θ[2][t]] + (m[1] + m[2]) * g * length[1] * Cos[θ[1][t]] +
      m[2] * g * length[2] * Cos[θ[2][t]] // Simplify;
  eqns = Table[D[L, θ[v][t]] == D[D[L, θ[v]'[t]], t], {v, 1, n}] // FullSimplify;
  vars = Table[θ[i][t], {i, 1, n}];
  Equations2 = vars;

 PhaseEquations2 = {θ[2][t], θ[2]'[t]};
  Do[
   ics = Join[Thread[(D[vars, t] /. {t → 0}) == Table[π (2 * Random[] - 1), {n}]],
     Thread[(vars /. {t → 0}) == Table[π (2 * Random[ ] - 1), {n}]]];
    Parameters = Join[eqns, ics];
   phasesoln2 = NDSolve[Parameters, PhaseEquations2, {t, 1, tmax}, AccuracyGoal → 10,
      MaxSteps → Infinity, Method -> {"EquationSimplification" -> "Solve"}][[1]];
 bifurcate[t1_] := {θ[2][t], θ[2]'[t]} /. phasesoln2 /. {t → t1};
 bifurcatetable[w] = Flatten[Table[{bifurcate[t]}, {t, 1, tmax, .1}], 1];
   bitranspose[w] = Transpose[bifurcatetable[w]][[1]];
 index[w] = Position[bitranspose[w], _?(Abs[#] < 3 &)];
 If[TrueQ[index[w] == {}], Print["Noindex"], AppendTo[result[z],
     Transpose[bifurcatetable[w]][[2]][[Flatten[index[w][[1]]]]]]];
   Clear[phasesoln2];, {w, 1, 70}], {z, 1, 8}] // AbsoluteTiming
```

Out[37]=
```
{22.684136, Null}
```

In[41]:=
```
CloseKernels[];
LaunchKernels[8];
```

```
ParallelDo[n = 2; tmax = 30; g = -10;
  m[1] = z * .1;
  m[2] = 10 * m[1];
  length[1] = 1;
  length[2] = 1;
  x[n_][t_] := Sum[length[v] * Sin[θ[v][t]], {v, 1, n}];
  y[n_][t_] := Sum[length[v] * Cos[θ[v][t]], {v, 1, n}];
  L = .5 (m[1] + m[2]) * (length[1])^2 * (θ[1]'[t])^2 +
      .5 * m[2] * length[2]^2 * (θ[2]'[t])^2 + m[2] * length[1] * length[2] * θ[1]'[t] *
       θ[2]'[t] * Cos[θ[1][t] - θ[2][t]] + (m[1] + m[2]) * g * length[1] * Cos[θ[1][t]] +
      m[2] * g * length[2] * Cos[θ[2][t]] // Simplify;
  eqns = Table[D[L, θ[v][t]] == D[D[L, θ[v]'[t]], t], {v, 1, n}] // FullSimplify;
  vars = Table[θ[i][t], {i, 1, n}];
  Equations2 = vars;

 PhaseEquations2 = {θ[2][t], θ[2]'[t]};
  Do[
   ics = Join[Thread[(D[vars, t] /. {t → 0}) == Table[π (2 * Random[] - 1), {n}]],
     Thread[(vars /. {t → 0}) == Table[π (2 * Random[ ] - 1), {n}]]];
    Parameters = Join[eqns, ics];
   phasesoln2 = NDSolve[Parameters, PhaseEquations2, {t, 1, tmax}, AccuracyGoal → 10,
      MaxSteps → Infinity, Method -> {"EquationSimplification" -> "Solve"}][[1]];
bifurcate[t1_] := {θ[2][t], θ[2]'[t]} /. phasesoln2 /. {t → t1};
bifurcatetable[w] = Flatten[Table[{bifurcate[t]}, {t, 1, tmax, .1}], 1];
   bitranspose[w] = Transpose[bifurcatetable[w]][[1]];
index[w] = Position[bitranspose[w], _?(Abs[#] < 3 &)];
If[TrueQ[index[w] == {}], Print["Noindex"], AppendTo[result[z],
     Transpose[bifurcatetable[w]][[2]][[Flatten[index[w][[1]]]]]]];
   Clear[phasesoln2];, {w, 1, 70}], {z, 1, 8}] // AbsoluteTiming
```

```
{122.82122340000000093596099759452044963837`7.378938835641728, Null}
```

```
maxKernel = 4;
data = Table[CloseKernels[];
    LaunchKernels[n];
    ParallelDo[n = 2; tmax = 30; g = -10;
      m[1] = z * .1;
      m[2] = 10 * m[1];
      length[1] = 1;
      length[2] = 1;
      x[n_][t_] := Sum[length[v] * Sin[θ[v][t]], {v, 1, n}];
      y[n_][t_] := Sum[length[v] * Cos[θ[v][t]], {v, 1, n}];
      L = .5 (m[1] + m[2]) * (length[1])^2 * (θ[1]'[t])^2 +
          .5 * m[2] * length[2]^2 * (θ[2]'[t])^2 + m[2] * length[1] * length[2] *
           θ[1]'[t] * θ[2]'[t] * Cos[θ[1][t] - θ[2][t]] + (m[1] + m[2]) * g * length[1] *
           Cos[θ[1][t]] + m[2] * g * length[2] * Cos[θ[2][t]] // Simplify;
      eqns = Table[D[L, θ[v][t]] == D[D[L, θ[v]'[t]], t], {v, 1, n}] // FullSimplify;
      vars = Table[θ[i][t], {i, 1, n}];
      Equations2 = vars;

  PhaseEquations2 = {θ[2][t], θ[2]'[t]};
      Do[
       ics = Join[Thread[(D[vars, t] /. {t → 0}) == Table[π (2 * Random[] - 1), {n}]],
         Thread[(vars /. {t → 0}) == Table[π (2 * Random[ ] - 1), {n}]]];
        Parameters = Join[eqns, ics];
       phasesoln2 = NDSolve[Parameters, PhaseEquations2,
          {t, 1, tmax}, AccuracyGoal → 10, MaxSteps → Infinity,
          Method -> {"EquationSimplification" -> "Solve"}][[1]];
bifurcate[t1_] := {θ[2][t], θ[2]'[t]} /. phasesoln2 /. {t → t1};
bifurcatetable[w] = Flatten[Table[{bifurcate[t]}, {t, 1, tmax, .1}], 1];
       bitranspose[w] = Transpose[bifurcatetable[w]][[1]];
index[w] = Position[bitranspose[w], _?(Abs[#] < 3 &)];
If[TrueQ[index[w] == {}], None, AppendTo[result[z],
         Transpose[bifurcatetable[w]][[2]][[Flatten[index[w][[1]]]]]]];
       Clear[phasesoln2];, {w, 1, 70}], {z, 1, 8}] // AbsoluteTiming; //
    AbsoluteTiming // First, {n, 0, maxKernel}]
```

ParallelDo::nopar : No parallel kernels available; proceeding with sequential evaluation. ≫

In[52]:=
```
data = {46.400959600000000193631422007456421852 11`7.6871268754007875,
  31.846244800000000907402863958850502967 83`7.52365814245374,
  24.426295499999998384055288624949753284 45`7.408457520003329,
  22.821223400000000935960997594520449638 37`7.378938835641728}
```

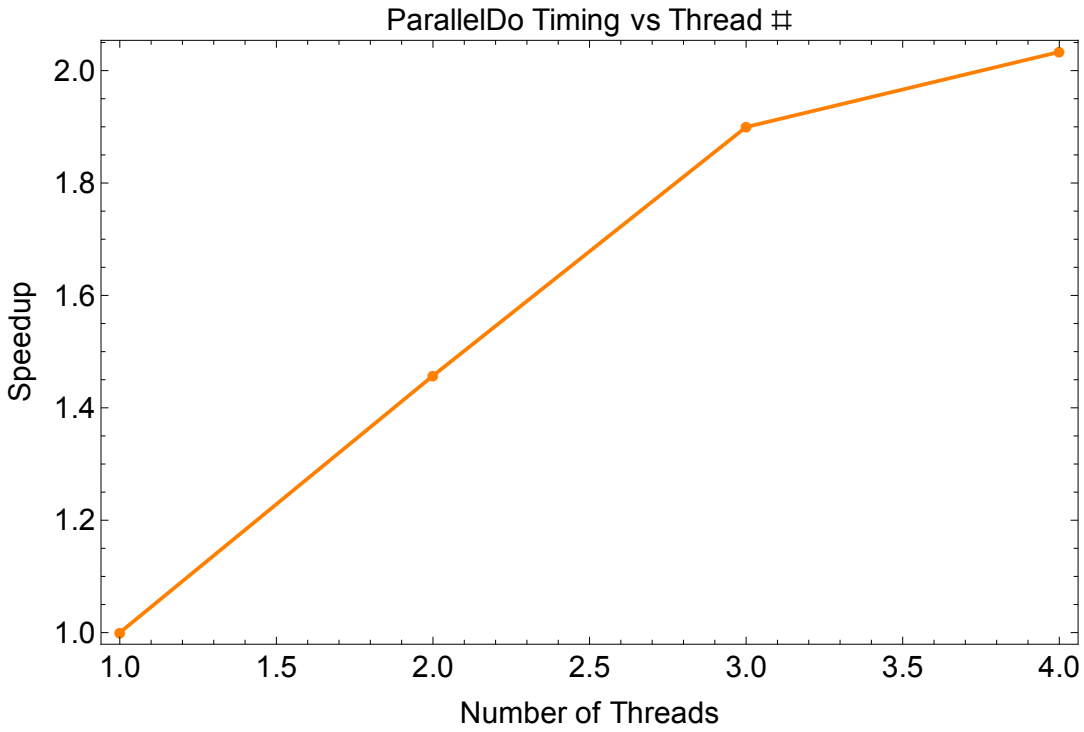
Out[52]= {46.400960, 31.846245, 24.426295, 22.821223}

In[56]:=
```
SpeedupData =
 Function[x, 46.400959600000000193631422007456421852l1`7.6871268754007875 / x] /@
  {46.400959600000000193631422007456421852l1`7.6871268754007875,
   31.846244800000000907402863958850502967 83`7.52365814245374,
   24.426295499999998384055288624949753284 45`7.408457520003329,
   22.821223400000000935960997594520449638 37`7.378938835641728}
```

Out[56]=
```
{1.0000000, 1.4570308, 1.899631, 2.033237}
```

In[87]:=
```
Plot1 = ListPlot[{{1, SpeedupData[[1]]}, {2, SpeedupData[[2]]},
   {3, SpeedupData[[3]]}, {4, SpeedupData[[4]]}}, ImageSize → Large,
  Joined → True, PlotLabel → Style["ParallelDo Timing vs Thread #",
    FontFamily → "Helvetica", FontSize → 16], Frame → True,
  FrameTicks → Automatic, FrameLabel → {"Number of Threads", "Speedup"},
  LabelStyle → {FontFamily → "Helvetica", FontSize → 16},
  PlotStyle -> {Orange, Thick}, PlotMarkers → Automatic]
```

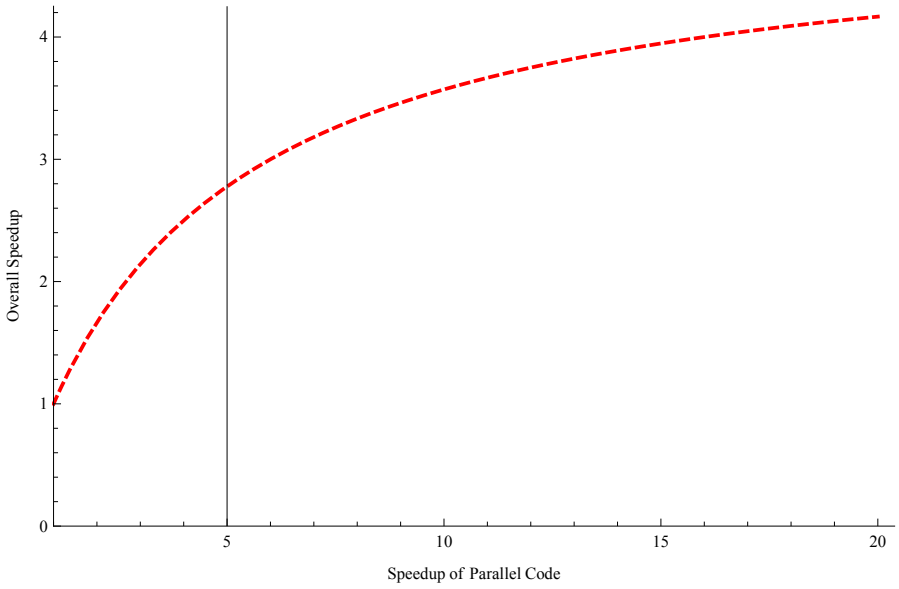
Out[87]=




```
Amdahl[f_, p_] := 1 / ((1 - f) + (f / p))
```

In[89]:=
```
Plot2 = Plot[Amdahl[.8, pro], {pro, 1.00, 20},
  PlotRange → {{1, Automatic}, {0, Automatic}},
  PlotLabel → Text@Style["Amdahl's Law .9 Parallelized", "Label", 14],
  FrameLabel → {"Speedup of Parallel Code", "Overall Speedup"},
  ImageSize → {540, 400}, ImagePadding → {{60, 30}, {35, 50}},
  Frame → {True, True, False, False}, PlotStyle → {Red, Dashed, Thick}]
```

Out[89]=



In[90]:=
```
Show[Plot1, Plot2 , PlotRange → {{1, 6}, {1, 5}}]
```

Out[90]=