Research Skills Problem Set 1, Michelle Gurevich

Question 1

(a) The sawtooth wave function is defined to be equal to x for $-\pi < x \leq \pi$, and to be periodic with period 2π. Plot the sawtooth wave function for $-2\pi < x \leq 4\pi$. (Hint: there are a number of ways of defining the sawtooth in Mathematica, but a neat one uses Mathematica's `Mod` function.)

```
In[•]:= SW[x_] := Mod[x - Pi, 2 Pi] - Pi
        Plot[SW[x], {x, -2 * Pi, 4 * Pi}]
```

(b) Use the `Ticks` option to make the ticks on the horizontal axis go from -2π to 4π in steps of π, and those on the vertical axis from $-\pi$ to π in steps of π.

```
In[•]:= Plot[SW[x], {x, -2 Pi, 4 Pi}, Ticks → {Range[-2 Pi, 4 Pi, Pi], Range[-Pi, Pi, Pi]}]
```

(c) Use the `FourierSinSeries` function to find the Fourier sine series for the sawtooth, up to the term in $\sin(5t)$.

```
In[•]:= FSS[t_] = FourierSinSeries[SW[t], t, 5]
```

(d) Plot the sawtooth, and this Fourier series approximation, on the same pair of axes, making the first curve blue and the other red. (The `Evaluate` function may be useful here.)

```
In[•]:= Plot[{Evaluate[FSS[t]], SW[t]}, {t, -2 Pi, 4 Pi},
         PlotStyle → {Red, Blue}, Ticks → {Range[-2 Pi, 4 Pi, Pi], Range[-Pi, Pi, Pi]}]
```

(e) Label your axes, and give your plot a title and legend.

```
In[•]:= Plot[{Evaluate[FSS[t]], SW[t]}, {t, -2 π, 4 π}, PlotStyle → {Red, Blue},
         Ticks → {Range[-2 π, 4 π, π], Range[-π, π, π]}, AxesLabel → {t, y}, PlotLabel →
          Style["Sawtooth function and its Fourier approximation", FontSize → 14],
         PlotLegends → {"Fourier sine series approximation to 5th order",
           "Sawtooth wave function of period 2π"}]
```

(f) Using `Manipulate`, create a graphical demonstration of the way in which the Fourier sine series, up to the term in sin ((2 *n* – 1) *t*), converges as *n* increases.

```
In[•]:= Manipulate[Plot[{Evaluate[FourierSinSeries[SW[t], t, 2 n - 1]], SW[t]},
          {t, -2 π, 4 π}, PlotStyle → {Red, Blue},
          Ticks → {Range[-2 π, 4 π, π], Range[-π, π, π]}, AxesLabel → {t, y}, PlotLabel →
           Style["Sawtooth function and its Fourier approximation", FontSize → 14],
          PlotLegends → {"Fourier sine series approximation to 5th order",
            "Sawtooth wave function of period 2π"}], {n, 1, 6, 1}]
```

Question 2

(a) Use the `ComplexPlot3D` function to create a visualisation of the function $f(z) = (z^2 + 1)/(z^2 - 1)$, for values of z taken from the square whose bottom-left corner is $-2 - 2i$ and whose top-right corner is $2 + 2i$.

```
In[•]:= F[z_] := (z^2 + 1) / (z^2 - 1)
        ComplexPlot3D[F[z], {z, -2 - 2 I, 2 + 2 I}]
```

(b) Repeat part (a), with the `ColorFunction` option set to the value "`CyclicReImLogAbs`" (make sure it's a string).

```
In[•]:= ComplexPlot3D[F[z], {z, -2 - 2 I, 2 + 2 I}, ColorFunction → "CyclicReImLogAbs"]
```

(c) Use the `ComplexContourPlot` function to create a visualisation of the contours of the real and imaginary parts of $f(z)$, on the same pair of axes, for the same set of values of z.

```
In[•]:= ComplexContourPlot[{Re[F[z]], Im[F[z]]}, {z, 2}, PlotLegends → "Expressions"]
```

(d) Use the `ComplexVectorPlot` function to create a visualisation of the complex vector field given by the **conjugate of the derivative** of $f(z)$, for the same set of values of z. Show this on the same pair of axes as your plot from part (c).

```
In[•]:= G[z_] := Conjugate[D[F[z], z]]
        ComplexVectorPlot[G[z], {z, -2 - 2 I, 2 + 2 I}]
```

(e) Use the `ComplexExpand` function to find the real and imaginary parts of $f(x + i\,y)$ in terms of the real variables x and y.

```
In[•]:= ComplexExpand[F[x + I y]]
```

```
In[•]:= ComplexExpand[Re[F[z] /. z → x + I y]]
```

```
In[•]:= ComplexExpand[Im[F[z] /. z → x + I y]]
```

(f) Using `ContourPlot`, create, and show on the same pair of axes, contour plots of the real part (in blue) and the imaginary part (in red), for values of x and y between -2 and 2. **Make sure you turn off contour shading with an appropriate options setting**. (You'll probably need to use the `Show` function here, because the behaviour of `ContourPlot` when you give it a list of functions to plot is not good, for some reason.)

```
In[•]:= CP1 := ContourPlot[Re[F[x + I y]], {x, -2, 2},
            {y, -2, 2}, ContourStyle → Blue, ContourShading → None]
```

```
In[•]:= CP2 := ContourPlot[Im[F[x + I y]], {x, -2, 2},
            {y, -2, 2}, ContourStyle → Red, ContourShading → None]
```

```
In[•]:= Show[CP1, CP2]
```

(g) Using `VectorPlot`, create, and show on the same pair of axes, a contour plot of the vector field $\left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right)$ for values of x and y between -2 and 2, where $u(x, y)$ is the real part of $f(x + i\,y)$. Show it on the same pair of axes as your plot from part (f).

```
In[•]:= U := ComplexExpand[Re[F[z] /. z → x + I y]]
         V := ComplexExpand[Im[F[z] /. z → x + I y]]
```

```
In[•]:= Ux := FullSimplify[D[U, x]]
         Uy := FullSimplify[D[U, y]]
```

```
In[•]:= VectorPlot[Evaluate[{Ux, Uy}], {x, -2, 2}, {y, -2, 2}]
```

```
In[•]:= Show[
          {ContourPlot[U, {x, -2, 2}, {y, -2, 2}, ContourShading → None, ContourStyle → Blue],
           ContourPlot[V, {x, -2, 2}, {y, -2, 2}, ContourShading → None, ContourStyle → Red],
           VectorPlot[Evaluate[{Ux, Uy}], {x, -2, 2}, {y, -2, 2}]}]
```

Question 3

(a) Write a short description of the solid figure displayed when you evaluate the code $\texttt{Plot3D}\left[\left\{\sqrt{1-\left(\sqrt{x^2+y^2}-3\right)^2}, -\sqrt{1-\left(\sqrt{x^2+y^2}-3\right)^2}\right\}, \{x, -4, 4\}, \{y, -4, 4\}, \texttt{BoxRatios} \to \texttt{Automatic}, \texttt{PlotStyle} \to \texttt{White}\right]$

```
In[•]:= Plot3D[{Sqrt[1 - (Sqrt[x^2 + y^2] - 3)^2], -Sqrt[1 - (Sqrt[x^2 + y^2] - 3)^2]},
  {x, -4, 4}, {y, -4, 4}, BoxRatios → Automatic, PlotStyle → White]
```

The figure is a torus shaded in red-blue gradient. Its surface is covered in a grid and it is enclosed in a box that corresponds to the side length ratios of the figure. The x and y axes range from -4 to 4, and the height component ranges from -1 to 1.

(b) Use `ContourPlot3D` to create an image of the exact same solid. (Hint: make the first argument an *equation* of the form `z^2 == ...`)

```
In[•]:= ContourPlot3D[1 - (Sqrt[x^2 + y^2] - 3)^2 - z^2 == 0, {x, -4, 4},
  {y, -4, 4}, {z, -1, 1}, BoxRatios → Automatic, ContourStyle → White]
```

(c) Use the `Tube` function, combined with `Graphics3D`, to create an image of the same solid.

```
In[•]:= Graphics3D[Tube[Table[3 {Cos[t], Sin[t], 0}, {t, 0, 2 Pi, Pi/100}], 1],
  BoxRatios → Automatic, Axes → True]
```

(d) Use `ParametricPlot3D` to create an image of the same solid.

```
In[•]:= ParametricPlot3D[{(3 + Cos[u]) Cos[v], (3 + Cos[u]) Sin[v], Sin[u]}, {u, 0, 2 Pi},
  {v, 0, 2 Pi}, BoxRatios → Automatic, PlotStyle → White, Mesh → Automatic]
```

(e) (*Tricky*) Use `RegionPlot3D` to create an image of the same solid.

```
In[•]:= RegionPlot3D[(3 - Sqrt[x^2 + y^2])^2 + z^2 <= 1, {x, -4, 4},
  {y, -4, 4}, {z, -1, 1}, BoxRatios → Automatic, ColorFunction → White]
```

Question 4

(a) Write a Mathematica function called `randomHop` that takes as its argument a list containing two coordinates `x` and `y`; your definition should begin

`randomHop[{x_, y_}] :=`

The function should then generate a random integer equal to 1, 2, 3 or 4, and then
- if `r` is equal to 1, return `{x, y}/3`;
- if `r` is 2, return `{x, y}/3 + {2/3, 0}`;
- if `r` is 3, return `RotationMatrix[π/3].{x, y}/3 + {1/3, 0}`;
- if `r` is 4, return `RotationMatrix[-π/3].{x, y}/3 + {1/2, Sqrt[3]/6}`;

Make sure that any variables you use inside the function are locally scoped. (There are implementations that don't require any named variables, but if you do use any, make sure you scope them.)

```
In[•]:= randomHop[{x_, y_}] := Module[{r}, r = RandomInteger[{1, 4}];
          Which[r == 1, {x, y} / 3, r == 2, {x, y} / 3 + {2 / 3, 0},
           r == 3, RotationMatrix[π / 3].{x, y} / 3 + {1 / 3, 0}, r == 4,
           RotationMatrix[-π / 3].{x, y} / 3 + {1 / 2, Sqrt[3] / 6}]]
```

Test your function.

```
In[•]:= randomHopTest[{x_, y_}] := Module[{r}, r = RandomInteger[{1, 4}];
          Print["r = ", r];
          Which[r == 1, {x, y} / 3, r == 2, {x, y} / 3 + {2 / 3, 0},
           r == 3, RotationMatrix[π / 3].{x, y} / 3 + {1 / 3, 0}, r == 4,
           RotationMatrix[-π / 3].{x, y} / 3 + {1 / 2, Sqrt[3] / 6}]]

In[•]:= randomHopTest[{1, 1}]

In[•]:= randomHopTest[{1000, 135}]

In[•]:= randomHopTest[{334, 45}]

In[•]:= randomHopTest[{0, 0}]
```

(b) Using `NestList` (or, if you prefer, `Table` or `Do`), iterate your function 10 times, with the starting value `{0.0, 0.0}`, generating a list of 11 coordinate pairs.

```
In[•]:= NestList[randomHop, {0.0, 0.0}, 10]
```

(c) When you're happy that you've got this working, iterate 10000 times instead. Don't show the output, but plot it using `ListPlot`. Make sure that the scale is the

same on both axes using the `AspectRatio` option.

```
In[•]:= ListPlot[NestList[randomHop, {0.0, 0.0}, 10 000], AspectRatio → Automatic]
```

Question 5

(a) Create a `ComplexPlot3D` plot of $f(z)$ from Question 2(a), over the same set of values of z, with the `Mesh` option set to `10`, and `MeshStyle` set to `Blue`.

```
In[•]:= ComplexPlot3D[F[z], {z, -2 - 2 I, 2 + 2 I}, Mesh → 10, MeshStyle → Blue]
```

(b) Repeat part (b), but this time also set the MeshFunctions option to **a list containing one pure function**, so that the mesh corresponds to contours of the absolute value of $f(z)$; the correct setting is `MeshFunctions → {Abs[#2]&}`.

```
In[•]:= ComplexPlot3D[F[z], {z, -2 - 2 I, 2 + 2 I},
     MeshFunctions → {Abs[#2] &}, Mesh → 10, MeshStyle → Blue]
```

(c) Create two `ComplexPlot3D` plots of f(z) from Question 2; one showing contours of the real part of $f(z)$, in blue, and the other showing contours of the imaginary part of $f(z)$, in red. Show them on the same set of axes.

```
In[•]:= CP3 = ComplexPlot3D[F[z], {z, -2 - 2 I, 2 + 2 I},
       Mesh → 10, MeshStyle → Blue, MeshFunctions → {Re[#2] &}]
```

```
In[•]:= CP4 = ComplexPlot3D[F[z], {z, -2 - 2 I, 2 + 2 I},
       Mesh → 10, MeshStyle → Red, MeshFunctions → {Im[#2] &}]
```

```
In[•]:= Show[CP3, CP4]
```

Question 6

The **Chebyshev polynomials of the first kind** are defined as follows:

$T_0(x) = 1$

$T_1(x) = x$

$T_n(x) = 2\,x\,T_{n-1}(x) - T_{n-2}(x)$

Mathematica has a built-in function called `ChebyshevT` for calculating these:

ChebyshevT[20,x]

(a) The following is some function code that uses a `Do` loop in order to calculate the nth Chebyshev polynomial of the first kind:
chebyshevT1[n_,x_]:=Module[{a=1,b=x},Do[{a,b}={b,Expand[2x b-a]},{n-1}];b]

Test this function.

chebyshevT1[n,x]==ChebyshevT[n,x] results in True except in the case of n=0, where ChebyshevT has the expected output of 1 but chebsyshevT1 incorrectly returns x.

```
In[•]:= chebyshevT1[n_, x_] := Module[{a = 1, b = x}, Do[{a, b} = {b, Expand[2 x b - a]}, {n - 1}];
          b]

In[•]:= chebyshevT1[0, x] == ChebyshevT[0, x]

In[•]:= chebyshevT1[1, x] == ChebyshevT[1, x]

In[•]:= chebyshevT1[15, x] == ChebyshevT[15, x]

In[•]:= chebyshevT1[4, x]
```

(b) Write and test an implementation, `chebyshevT2[n_, x_]`, that instead of a `Do` loop uses `Nest`, preferably with a pure function (though you don't have to use a pure function).

chebyshevT2[n,x]==ChebyshevT[n,x] results in True except in the case of n=0, where ChebyshevT has the expected output of 1 but chebyshevT2 fails because n-1 is out of scope (just as chebyshevT1 did in part a of this question).

```
In[•]:= chebyshevT2[n_, x_] := Nest[Apply[{#2, Expand[2 x #2 - #1]} &], {1, x}, n - 1][[2]]

In[•]:= chebyshevT2[1, x] == ChebyshevT[1, x]

In[•]:= chebyshevT2[15, x] == ChebyshevT[15, x]

In[•]:= chebyshevT2[4, x]

In[•]:= chebyshevT2[0, x]
```

(c) Write and test an implementation, `chebyshevT3[n_, x_]`, that uses a **recursion**: that is, that calls itself. For full marks, write your function in such a way that it reliably executes in reasonable time.

```
In[•]:= chebyshevT3[0, x_] := 1
        chebyshevT3[1, x_] := x
```

```
In[•]:= chebyshevT3[n_, x_] := Collect[2 x chebyshevT3[n - 1, x] - chebyshevT3[n - 2, x], x]

In[•]:= chebyshevT3[1, x] == ChebyshevT[1, x]

In[•]:= chebyshevT3[15, x] == ChebyshevT[15, x]

In[•]:= chebyshevT2[4, x]

In[•]:= chebyshevT1[18, x]
```

(d) Using `RSolve`, show that $T_n(x) = \frac{1}{2}\left(\left(x - \sqrt{x^2-1}\right)^n + \left(x + \sqrt{x^2-1}\right)^n\right)$, and show that for $n = 20$ this is equivalent to the output of the built-in function `ChebyshevT`.

```
In[•]:= Clear[T, n]
        RSolve[T[n] == 2 x T[n - 1] - T[n - 2] && T[0] == 1 && T[1] == x, T[n], n]

In[•]:= SolnT := RSolve[T[n] == 2 x T[n - 1] - T[n - 2] && T[0] == 1 && T[1] == x, T, n]

In[•]:= Expand[T[20] /. SolnT[[1]]]

In[•]:= Expand[T[20] /. SolnT[[1]]] == ChebyshevT[20, x]
```

(e) Calculate the 20th Chebyshev polynomial of the first kind using `LinearRecurrence`.

```
In[•]:= LinearRecurrence[{2 x, -1}, {x, -1 + 2 x^2}, {20}]

In[•]:= ChebyshevT[20, x]
```

Question 7

The **double pendulum** consists of two masses, m_1 and m_2, connected by light rods of lengths a_1 and a_2. The mass m_1 is connected to a fixed smooth hinged joint, and the mass m_2 is connected to a hinged joint that is coincident with the mass m_1; all motion is in the plane.

The following code generates a picture of the machine for $a_1 = a_2 = 1$.

```
In[•]:= Block[{m1 = 1, m2 = 1, a1 = 1, a2 = 1, θ1 = π/6, θ2 = π/3}, Graphics[
    {Thick, Black, Directive[FontSize → 15], Circle[{0, 0}, 0.03 a1 + 0.03 a2], Line[
      {{0, 0}, a1 {Sin[θ1], -Cos[θ1]}, a1 {Sin[θ1], -Cos[θ1]} + a2 {Sin[θ2], -Cos[θ2]}}],
     AbsolutePointSize[Round[10 m1^(1/3)]], Point[a1 {Sin[θ1], -Cos[θ1]}],
     AbsolutePointSize[Round[10 m2^(1/3)]],
     Point[a1 {Sin[θ1], -Cos[θ1]} + a2 {Sin[θ2], -Cos[θ2]}],
     Text["a₁", a1/2 {Sin[θ1], -Cos[θ1]} + {0.1 a1 + 0.1 a2, 0}],
     Text["m₁", a1 {Sin[θ1], -Cos[θ1]} + {0.1 a1 + 0.1 a2, 0}],
     Text["a₂", a1 {Sin[θ1], -Cos[θ1]} + a2/2 {Sin[θ2], -Cos[θ2]} + {0.1 a1 + 0.1 a2, 0}],
     Text["m₂", a1 {Sin[θ1], -Cos[θ1]} + a2 {Sin[θ2], -Cos[θ2]} + {0.1 a1 + 0.1 a2, 0}],
     {Dashed, Line[{a1 {Sin[θ1], -Cos[θ1]}, a1 {Sin[θ1], -Cos[θ1]} + {0, -a2/2}}],
      Line[{{0, 0}, {0, -a1/2}}]}, Circle[{0, 0}, a1/3, {-π/2, -π/2 + θ1}],
     Circle[a1 {Sin[θ1], -Cos[θ1]}, a2/3, {-π/2, -π/2 + θ2}],
     Text["θ₁", a1/2.2 {Sin[θ1/2], -Cos[θ1/2]}],
     Text["θ₂", a1 {Sin[θ1], -Cos[θ1]} + a2/2.2 {Sin[θ2/2], -Cos[θ2/2]}]},
    PlotRange → a1 + a2, ImageSize → 500, Background → Lighter[Lighter[Gray]]]]
```

The state of the system consists of the values of the angles θ_1 and θ_2.
The system's equations of motion are

$$(m_1 + m_2)\, a_1\, \ddot{\theta}_1 + m_2\, a_2\, \ddot{\theta}_2 \cos(\theta_1 - \theta_2) + m_2\, a_2\, \dot{\theta}_2^{\,2} \sin(\theta_1 - \theta_2) + (m_1 + m_2)\, g \sin\theta_1 = 0$$

$$m_2\, a_2\, \ddot{\theta}_2 + m_2\, a_1\, \ddot{\theta}_1 \cos(\theta_1 - \theta_2) - m_2\, a_1\, \dot{\theta}_1^{\,2} \sin(\theta_1 - \theta_2) + m_2\, g \sin\theta_2 = 0.$$

where the dot denotes differentiation with respect to time.

(a) Set $m_1 = m_2 = a_1 = a_2 = 1$, and $g = 9.8$, and solve the system using `NDSolve` for $0 \le t \le 20$, using the initial values $\theta_1 = \theta_2 = \pi/2$, $\dot{\theta}_1 = \dot{\theta}_2 = 0$. Plot θ_1 and θ_2 against t for this range of values.

```
In[•]:= m1 = m2 = a1 = a2 = 1
        g = 9.8

In[•]:= m1

In[•]:= m2

In[•]:= a1

In[•]:= a2

In[•]:= g

In[•]:= Soln = NDSolve[{(m1 + m2) a1 θ1''[t] + m2 a2 θ2''[t] Cos[θ1[t] - θ2[t]] +
            m2 a2 θ2'[t]^2 Sin[θ1[t] - θ2[t]] + (m1 + m2) g Sin[θ1[t]] == 0,
          m2 a2 θ2''[t] + m2 a1 θ1''[t] Cos[θ1[t] - θ2[t]] -
            m2 a1 θ1'[t]^2 Sin[θ1[t] - θ2[t]] + m2 g Sin[θ2[t]] == 0,
          θ1[0] == θ2[0] == Pi / 2, θ2'[0] == θ1'[0] == 0}, {θ1[t], θ2[t]}, {t, 0, 20}]
        Plot[Evaluate[{θ1[t], θ2[t]} /. Soln], {t, 0, 20}, PlotLabel → "Motion of System",
         PlotLegends → {"θ1(t)", "θ2(t)"}, AxesLabel → {t, θ}]
```

(b) Using these same parameter and initial value settings, create a diagram representing the motion of the mass m_2 **in physical space** for $0 \le t \le 100$.

```
In[•]:= ParametricPlot[Evaluate[
          {a1 Sin[θ1[t]] + a2 Sin[θ2[t]], -a1 Cos[θ1[t]] - a2 Cos[θ2[t]]} /. Soln], {t, 0, 20}]
```

```
In[•]:= Soln2 = NDSolve[{(m1 + m2) a1 θ1''[t] + m2 a2 θ2''[t] Cos[θ1[t] - θ2[t]] +
          m2 a2 θ2'[t]^2 Sin[θ1[t] - θ2[t]] + (m1 + m2) g Sin[θ1[t]] == 0,
        m2 a2 θ2''[t] + m2 a1 θ1''[t] Cos[θ1[t] - θ2[t]] -
          m2 a1 θ1'[t]^2 Sin[θ1[t] - θ2[t]] + m2 g Sin[θ2[t]] == 0,
        θ1[0] == θ2[0] == Pi / 2, θ2'[0] == θ1'[0] == 0}, {θ1[t], θ2[t]}, {t, 0, 100}]
      ParametricPlot[Evaluate[{a1 Sin[θ1[t]] + a2 Sin[θ2[t]],
          -a1 Cos[θ1[t]] - a2 Cos[θ2[t]]} /. Soln2], {t, 0, 100}]
```

(c) For small initial values of the variables, the system can be approximated by the linear system

$(m_1 + m_2)\, a_1\, \ddot{\theta}_1 + m_2\, a_2\, \ddot{\theta}_2 + (m_1 + m_2)\, g\, \theta_1 = 0,$

$m_2\, a_2\, \ddot{\theta}_2 + m_2\, a_1\, \ddot{\theta}_1 + m_2\, g \sin\theta_2 = 0.$

Using the parameter values from part (a), solve this linearly approximated system using `DSolve`, for the initial values $\theta_1 = \theta_2 = 0.1$, $\dot{\theta}_1 = \dot{\theta}_2 = 0$. Compare its behaviour to that of the system itself.

```
In[•]:= Soln3 := DSolve[{(m1 + m2) a1 θ1''[t] + m2 a2 θ2''[t] + (m1 + m2) g θ1[t] == 0,
          m2 a2 θ2''[t] + m2 a1 θ1''[t] + m2 g θ2[t] == 0, θ1[0] == θ2[0] == .1,
          θ2'[0] == θ1'[0] == 0}, {θ1[t], θ2[t]}, {t, 0, 100}]
      Plot[Evaluate[{θ1[t], θ2[t]} /. Soln3], {t, 0, 20},
       PlotLabel → "Linear Approximation of System Motion",
       PlotLegends → {"θ1(t)", "θ2(t)"}, AxesLabel → {t, θ}]
```

For the comparison, we consider below the case where initial conditions for the linear approximation match those of the system described in part a. The linear approximation oscillates closer to equilibrium when the angles $\theta 1$ and $\theta 2$ are small. Since large errors would only be introduced when angles were must bigger, linear approximation is reasonable for the case given here.

```
In[•]:= Soln4 := DSolve[{(m1 + m2) a1 θ1''[t] + m2 a2 θ2''[t] + (m1 + m2) g θ1[t] == 0,
          m2 a2 θ2''[t] + m2 a1 θ1''[t] + m2 g θ2[t] == 0, θ1[0] == θ2[0] == Pi / 2,
          θ2'[0] == θ1'[0] == 0}, {θ1[t], θ2[t]}, {t, 0, 100}]
      Plot[Evaluate[{θ1[t], θ2[t]} /. Soln4], {t, 0, 20},
       PlotLabel → "Linear Approximation of System Motion",
       PlotLegends → {"θ1(t)", "θ2(t)"}, AxesLabel → {t, θ}]
```

```
In[•]:=
```

```
In[•]:=
```