An Investigation into the Dynamics of a Continuous Time Recurrent Neural Network Node

Candidate No: 52804

December 7, 2006

1 Introduction

Dynamic neural networks are used widely in the field of evolutionary robotics. One form, Continuous Time Recurrent Neural Networks (CTRNN) are commonly used to evolve controllers for robots and have been used extensively at Sussex. Beer [1] states that these networks are both "the simplest non-linear, continuous dynamical neural network model" and "they are universal dynamical approximators". Another advantage of a CTRNN models is that they are claimed to be a neural model with biologically plausibility.

In this paper I will study the behaviour of a CTRNN neuron using numerical integration, dynamical systems analysis and experimentation.

2 Definition of a CTRNN

A CTRNN is typically describe in the form of a differential equation describing how each node responds to change. I will use the equations describing the networks found in [2] and others, as a the basis for this report, which have following general form:

$$\frac{dy}{dt} = \frac{1}{\tau_i}\left(-y_i + \sum_{j=1}^{N} w_{ji}\sigma(g(y_j + \theta_j))\right) + I_i \tag{1}$$

where y_i is the state of each of each neuron, τ_i is its time constant, w_{ji} is the strength of the connection from the j^{th} to the i^{th} neuron, g is a gain, θ_i is the node's bias and I_i represents the current external input to this node, where i varies between 1 and N. N is the number of nodes in this network.

σ is the standard logistic activation function common to many neural networks and is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

3 A minimal CTRNN Node

To begin the analysis of a CTRNN of this type, we will first examine the properties of a single node without a recurrent connection, input, and with $\tau_i = 1$. This simple 'network' is shown in Figure 1.

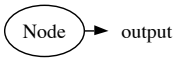

Figure 1: **Minimal CTRNN node**

The derivative of this network now becomes:

$$\frac{dy}{dt} = \frac{1}{1}\left(-y_i + \sum_{j=0}^{0} w_{ji}\sigma(g(y_j + \theta_j))\right) + 0 \tag{3}$$

Simplifying (3) we get:

$$\frac{dy}{dt} = -y_i \tag{4}$$

3.1 Numerical Integration of Node

Numerical integrating using Euler's method, with an initival value of $y(0) = 1$, a timestep $h = 0.01$ and a simulation length of 5, results in Figure 2. We can see from this figure that the system shows exponential decay towards 0 from the initial starting value. The same occurs if $y(0) = -1$. If the initial value is zero then then system remains at zero for all t.

From (4) we can see that that the system has one fixed point, when $\frac{dy}{dt} = 0$, when y = 0. Pertubing the system around this point shows that this is a stable fixed point. For example, using $h = 0.1$ shows the trend to decay back to 0:

y(0) = 0 + 0.01 → y(1h) = 0.009 → y(2h) = 0.008 → y(3h) = 0.007 etc.

y(0) = 0 - 0.01 → y(1h) = -0.009 → y(2h) = -0.008 → y(3h) = -0.007 etc.

∴ the node's basic nature is to decay exponentially to zero from a given starting point.

3.2 Time Constant Effect on a CTRNN Node

The time constant of a CTRNN node, τ, is there to model its membrane resistance time. If we change (4) to include τ_i, we can see how this effects the basic behaviour of the node. We now have a node defined by:

$$\frac{dy}{dt} = \frac{1}{\tau_i}(-y_i) \tag{5}$$

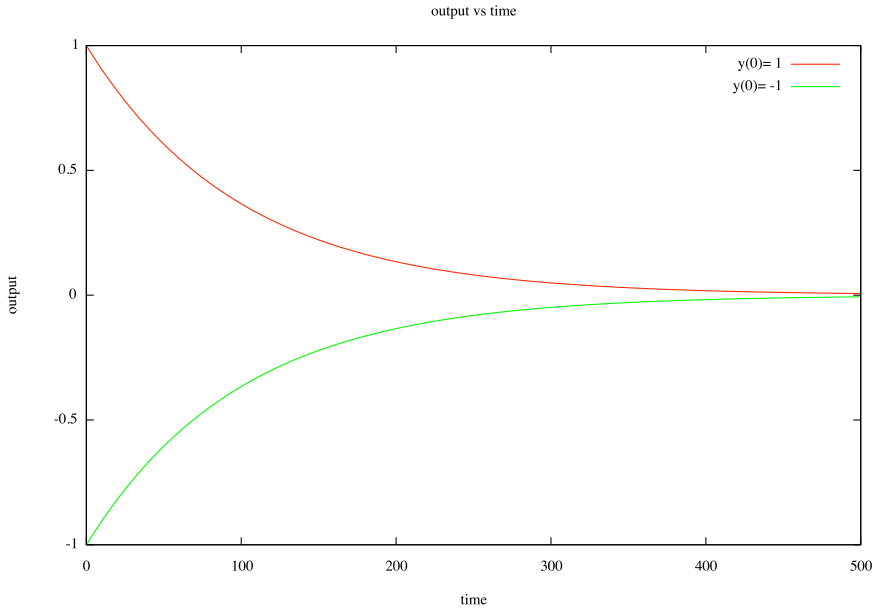

Figure 2: **Output of the CTRNN node over time for y(0) = -1 and y(0)=1**

If $\tau_i < 1$ then it acts as an excitatory component of the node and the initial state decays rapidly, if $\tau_i > 1$ it inhibits change in the node. This can be seen in Figure 3.

τ_i cannot be zero as $1/0$ is undefined and we will not consider $\tau_i < 0$ because it represents a time constant and thus allowing negative times is nonsensical in the context of analysing CTRNNs.

3.2.1 Courant-Friedrichs-Lewy Condition

One thing to pay close attention to during numerical integration is that the time it takes for the system to 'perform' a significant action must be larger than the time step used in the integration. This is known as the **Courant-Friedrichs-Lewy Condition** [5].

If we allow τ_i to approach the timestep, h, so that $\tau_i <= h$, we will breach this condition. Figure (4) shows the system response when $\tau_i = h$ and when $\tau_i = h/2$, for $h = 0.05$ and $y(0) = 1$.

When $\tau_i = h$ the output of the system follows the input, which in this case is zero. So the node 'decays' instantly. If we examine the equation used for Euler integration (6) we can see the term $(h * \frac{1}{\tau_i})$ is reduced to 1, resulting in equation (7).

$$y(t+1) = y(t) + (h * \frac{1}{\tau_i} * (-y(t)) \tag{6}$$

$$y(t+1) = y(t) + -y(t) \tag{7}$$

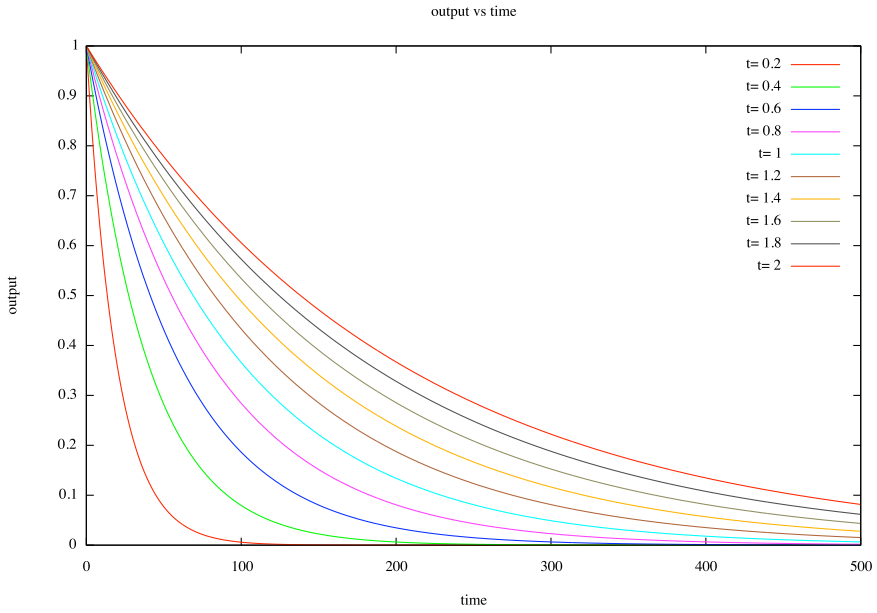

Figure 3: **Output of the CTRNN node over time for changing values of τ**

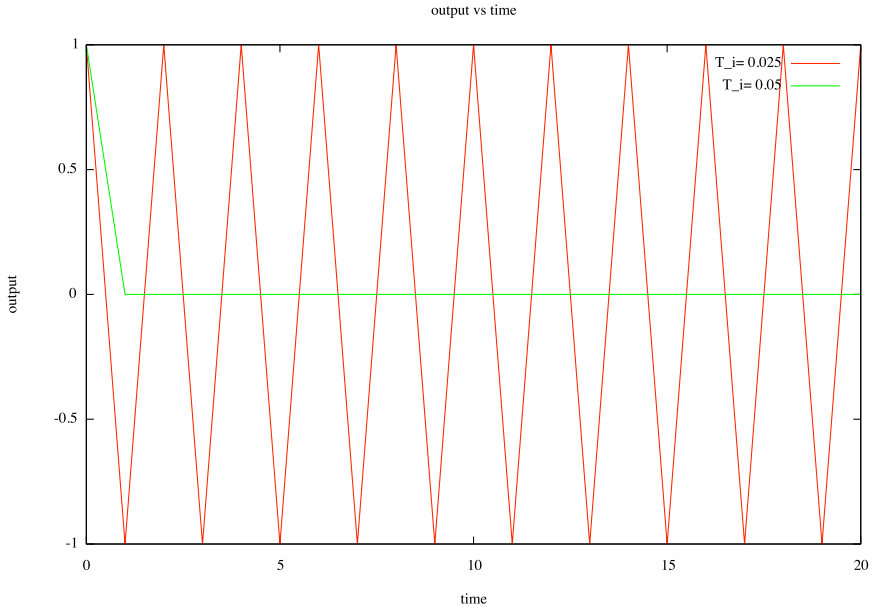

Figure 4: **Output of the CTRNN node over time when τ_i approaches h.**

If the starting value $y(0) = 1$, at the first integration step, $y(h) = 1-1 = 0$ and the system's output will switch to zero and stay there.

When $\tau_i = h/2$ (7) has reduced to (8) and the system now oscillates between -1 and 1 during integration steps. Following integration we see:

y(0) = 1 → y(1h) = 1 - 2 =-1→ y(2h) = -1 + 2 = 1→ y(3h) = 1 -2 = -1 etc.

$$y(t+1) = y(t) + 2(-y(t)) \tag{8}$$

Therefore *the time step must be kept small enough so that information has enough time to propagate through the space discretization* [5].

3.3 Adding input

Adding input to the node (Figure 5) changes the derivative for the network to (9).

$$\frac{dy}{dt} = \frac{1}{\tau_i}(-y_i) + I_i \tag{9}$$

If $\tau_i = 0$, numerical integration of (9) shows that that when $I_i > 0$, then the output of the node starts at $y(0)$ and approaches the value of I_i asymptotically, approaching I_i in the τ_i time. When $I_i < 0$ then the output of the node exponentially approaches I_i.

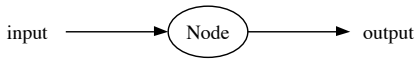

Figure 5: **Minimal CTRNN node with input**

Changing values of τ_i shows similar behaviour to Section 3.2. Figure 6 shows the output of the system when $y(0) = 0$, $h = 0.01$ and a simulation time of 5 for varying values of I_i and τ_i. Input, I_i is governed by (10) and $I_i = 0$ when the simulation time becomes 2.

$$I(t) = \begin{pmatrix} I & t \leq 2 \\ 0 & else \end{pmatrix} \tag{10}$$

Therefore we can see that a CTRNN node with no recurrent connection is exhibits growth to I_i with a time constant τ_i. When input is removed, the system has a fixed point at $y = 0$, which it will return to.

4 A Recurrent Connection

So far the CTRNN node under evaluation has exhibited only simple dynamics for a range of inputs and time constants. Adding a single reccurrent connection enables the node to exhibit much more

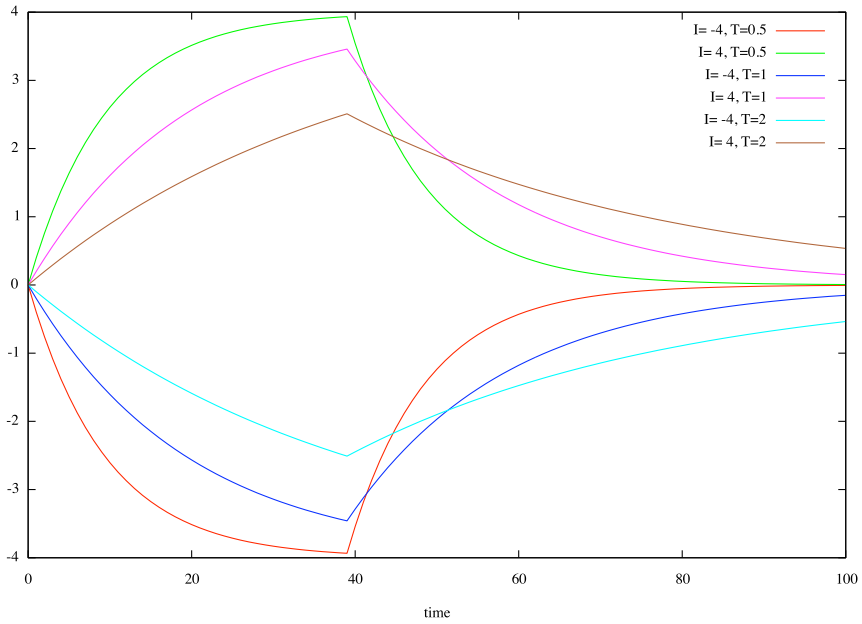

Figure 6: **CTRNN output with varying input and τ_i values**

complicated behaviour. The network under evaluation in this next section is show in Figure 7.

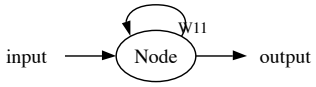

Figure 7: **Minimal CTRNN node with recurrent connection**

The derivative describing the network now becomes:

$$\frac{dy_i}{dt} = \frac{1}{\tau_i}\left(-y_i + w_i\sigma(g(y_i + \theta_i))\right) + I_i \tag{11}$$

4.1 Sigmoid Activation function

Before examining the dynamics of the recurrent node, the response of the sigmoid activation function $\sigma()$, is first considered. $\sigma()$ is defined in (2).

Examining the derivative of σ, σ' gives us further insight into its behaviour. We can rewrite (2) in the form $\sigma(x) = y = (1 - e^{-x})^{-1}$. If we allow $u = (1 + e^{-x})$, y becomes $y = (u)^{-1}$. Calculating the derivatives, $\frac{du}{dx} = -e^{-x}$ and $\frac{dy}{du} = -u^{-2}$, gives:

$$\sigma' = \frac{dy}{dx} = \frac{dy}{du}.\frac{du}{dx} = -u^{-2}.-e^{-x} = \frac{e^{-x}}{(1 + e^{-x})^2} \tag{12}$$

Others have show that the derivative, σ', can be written as in an alternative form as a logistic function (13), which is quick to compute during simulation.

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x)) \tag{13}$$

Plotting the output for $\sigma(x)$ (Figure 8) and the derivative, $\sigma'(x)$ (Figure 9) reveals interesting properties. The sigmoid function shows that its output will range from 0 to 1 as its input ranges from $-\infty$ to ∞. It can also be seen that when $x > 5$ or $x < 5$, the output of the sigmoid function will have saturated

The graph of the derivative σ' shows that $0 < \sigma' < 0.25$. It has its largest values in the activation function's transition range between 0 and 1 and as the connections saturate the derivative of the sigmoid function is almost zero. This shows that any near saturated connections in the CTRNN will reactly slowly to change.

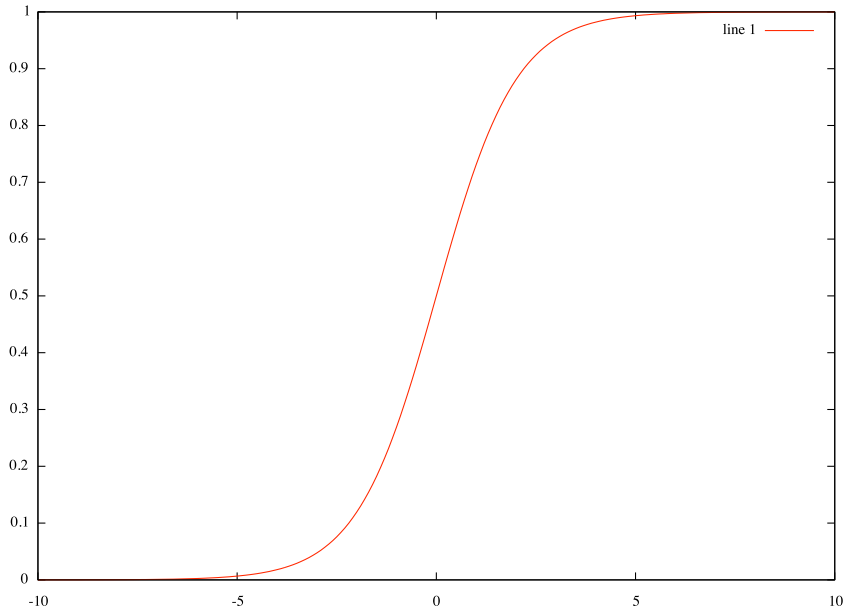

Figure 8: **The Sigmoid Activation Function**

4.2 Basic Behaviour

To examine the behaviour of the network, the derivative (11) can first be simplified by setting $g = 0$, $\tau_i = 1$ and $I_i = 0$, as none effect the fundamental behaviour of the network. Investigating from inside the sigmoid brackets outwards, we will exam the effect of bias, gain and connection weight on the response of the system. All numerical integration results in this section are for $h = 0.01$ and $t = 4$.

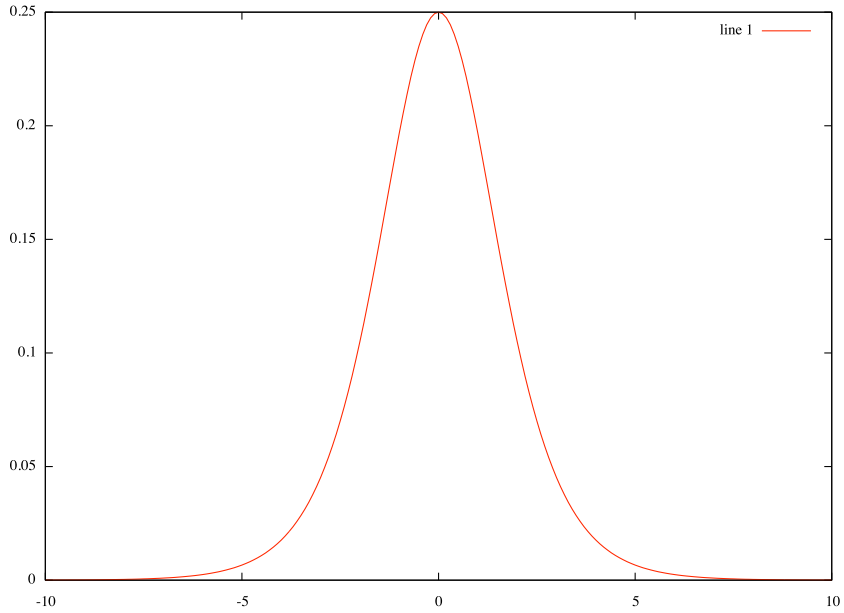

Figure 9: **Plotting the derivative of the Sigmoid function**

4.2.1 Effect of the bias term on the network

We will now add the bias term θ_i but fix the weight of the recurrent connection at $w_i = 1$. For investigation we will considered biases in the range, $\theta_i \in [-5, 5]$ from [2]. From a neural network point of view, the bias value can be considered as a special connection from a node with a constant non-zero activation.

$$\frac{dy_i}{dt} = -y_i + \sigma(y_i + \theta_i) \tag{14}$$

Examining (14) shows that the bias effects the response of the system and of the sigmoid function. This can be seen in Figure 10 and in Figure 11 respectively.

The response of the system when the bias is varied is determined by the squashing of the current state plus the bias, by the sigmoid function. From Figure 11 it can be seen that adding a bias to the node shifts the sigmoid response in the opposite direction. A positive bias moves the transition region to the left, a negative bias moves it to the right. The response is of the system is now centred on $-\theta_i$.

As we saw in Section 4.1, the sigmoid function is responsive when its inputs are in the transition zone. Values outside this range tend to the weight value (in this case 1) or zero (in the case of negative weights). By shifting the the centering of the transition zone, the sigmoid function becomes biased, that is instead of the $x > 0 \to 0$, now $x - \theta_i > 0 \to 0$.

Therefore the θ can be used to 'tune' the contributions of a node's value by biasing the sigmoid's

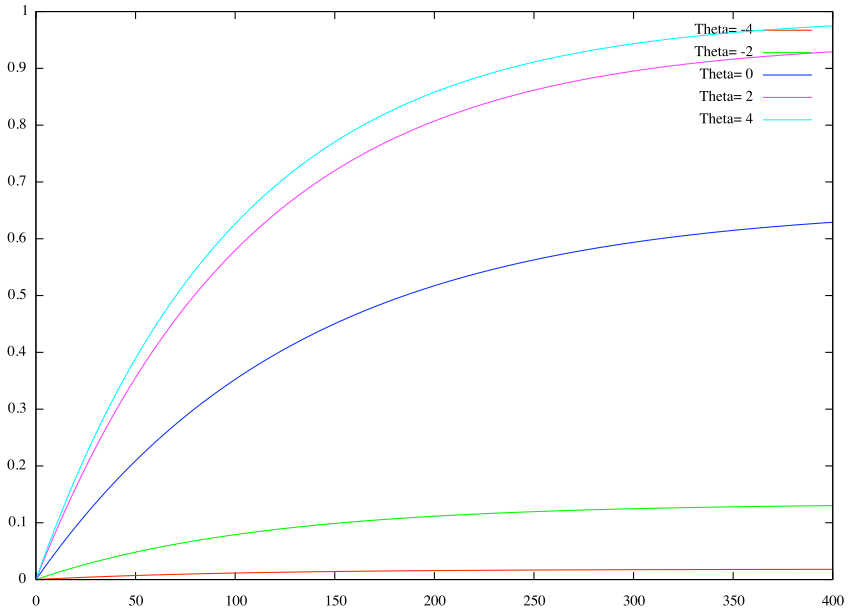

Figure 10: **Changing bias and its effect on system response**

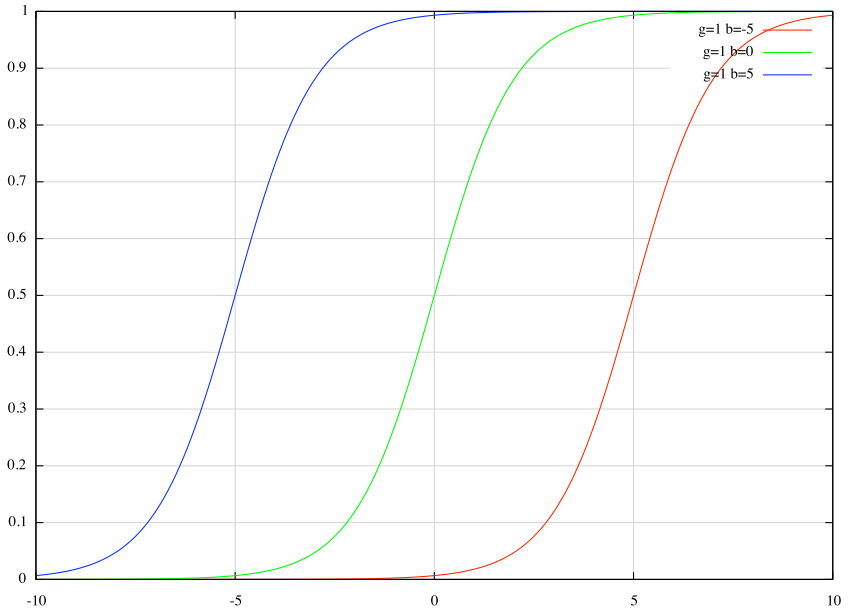

Figure 11: **Changing bias and its effect on the sigmoid function**

response to input.

4.2.2 Effect of the gain term on the network

The gain term is a special parameter used to make certain neurons in a network highly sensitive to their input. Typically, $g \in [1, 5]$ and is only > 1 for neurons connected directly to sensory input or motor output [2]. To examine the effect of the gain term on its own, we will set $w_i = 1$ and $\theta_i = 0$.

$$\frac{dy_i}{dt} = -y_i + \sigma(g(y_i)) \tag{15}$$

Examining (14) shows that the gain, like bias, effects the response of the of the system and of the sigmoid function. This can be seen in Figure 12 and Figure 13 respectively.

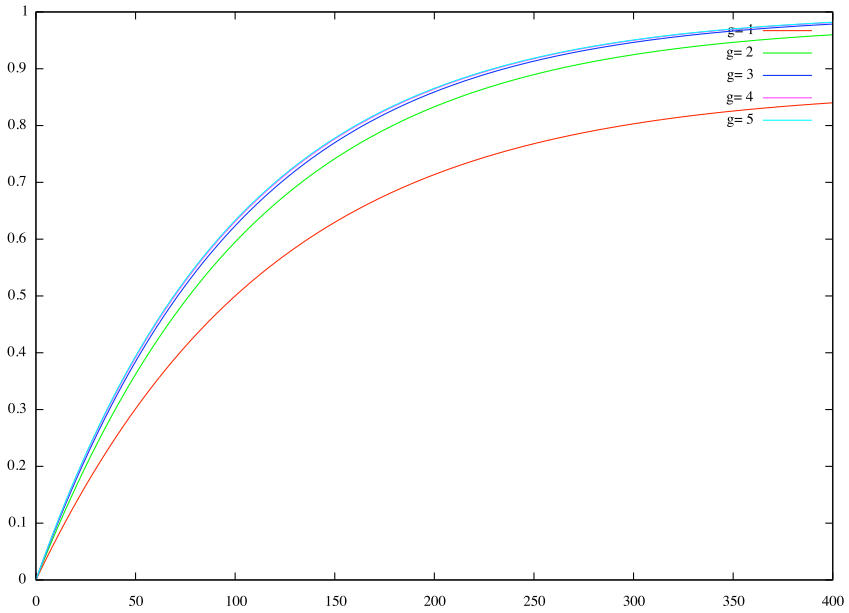

Figure 12: **Changing gain and its effect on system response**

As gain is inside the sigmoid term, it provides another way of tuning the sigmoid response for a range of node values. When $g > 0$, the steeper the transition section is in the activation function and the quicker the function saturates. If $0 < g < 1$, there is a reduction of the gradient of the transition region, increasing the range of inputs the activation function is responsive to.

If $g < 0$ then the response switches around, now values less than the centre of the transition region tend to zero, and values greater than the centre point of the transition region tend to one.

For the remainder of this investigation we will take $g = 1$, as its a parameter present for only a few special case input nodes.

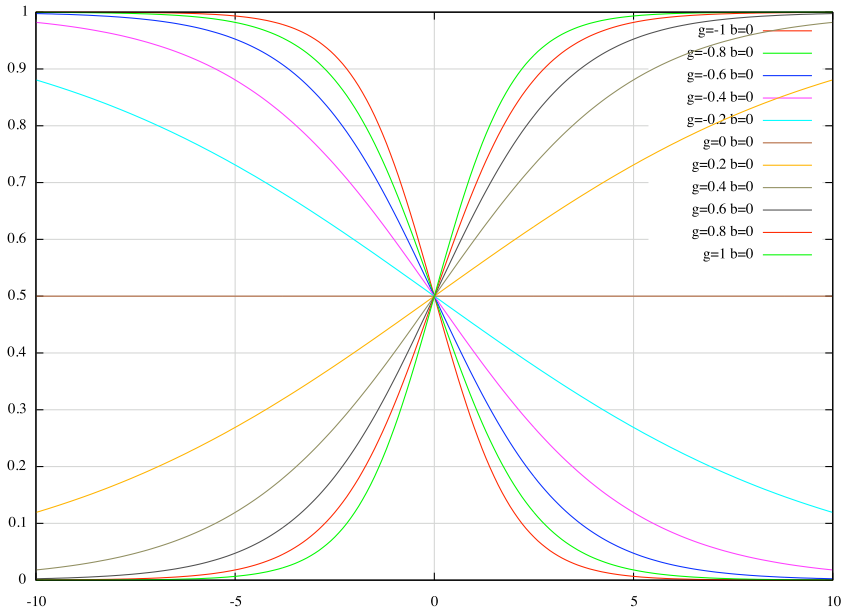

Figure 13: **Changing gain and its effect on the activation function**

4.2.3 Effect of weight changes on the network

Now we will investigate the network behaviour for varying weight values to understand the effect of the weight of connections. For this we will set the bias, $\theta_i = 0$. The the derivative for the network is now:

$$\frac{dy_i}{dt} = -y_i + w_i\sigma(y_i) \tag{16}$$

If we numerical integrate using connection weights $\in [-5, 5]$ from [2] and $y(0) = 0$, we get output as seen in Figure 14.

Examining (16) shows that the weight, w_i, acts a multiplier for the recurrent connection. Starting at $y(0) = 0$ the sigmoid function contributes 0.5 forcing change in the system (see Figure 8).

If the weight is positive, the connection reinforces the the value of y_i, as the sigmoid function for positive values tends to one. The connection will saturate when $\sigma(x) \to 1$ and the change in each timestep to $\approxeq -y_i + (w_i * 1)$, as $y_i \to w_i$ then $\frac{dy}{dx} \approxeq 0 \therefore$ the system asymptopically approaches the weight value.

The graph is asymmetric for the negative weights because of the response of the sigmoid function. From Figure 8 we can see that negative inputs to the sigmoid function tend to zero. Therefore as y_i tries to approach w_i, $\sigma(x) \to 0$, reducing the contribution of the recurrent connection.

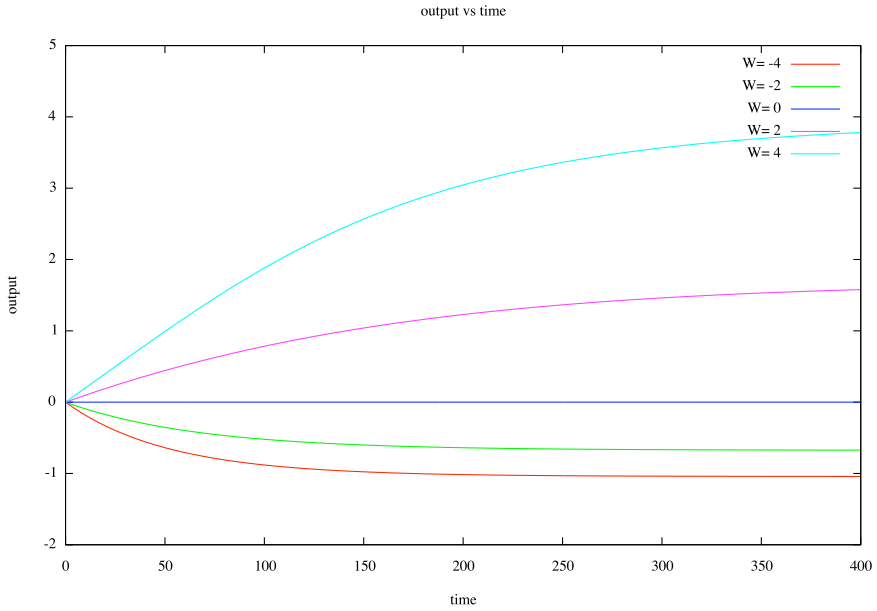

Figure 14: **Changing weights and its effect on system response**

4.2.4 A more complete model

Now we have examined the individual effects of the weight and biases, we will examine a model with both weight, bias and input. The derivative under test now looks like:

$$\frac{dy_i}{dt} = -y_i + w_i\sigma(y_i + \theta_i) + I_i \tag{17}$$

Numerically integrating with $w_i \in [-5, 5]$ and $\theta_i \in [-5, 5]$ with I_i following (10), gives the results in Figure 15.

By examining the phase portraits of y_i we can see the system's trends when moving around $w_i \in [-5 : 5]$ and $\theta_i \in [-5 : 5]$ show in Figure 16. These are direction fields for the extremes of the allowed values of w_i and θ_i. When the bias is in its negative range, it shifts the sigmoid to centre on the bias so values below this tend to zero. When the weight is positive, the trend of the system is to try and approach w_i.

4.3 Analysis of Single CTRNN as a dynamical System

Returning to (18) we can view this equation as:

$$f(y; w_i, \theta_i, I_i) = \frac{dy_i}{dt} = -y_i + w_i\sigma(y_i + \theta_i) + I_i \tag{18}$$

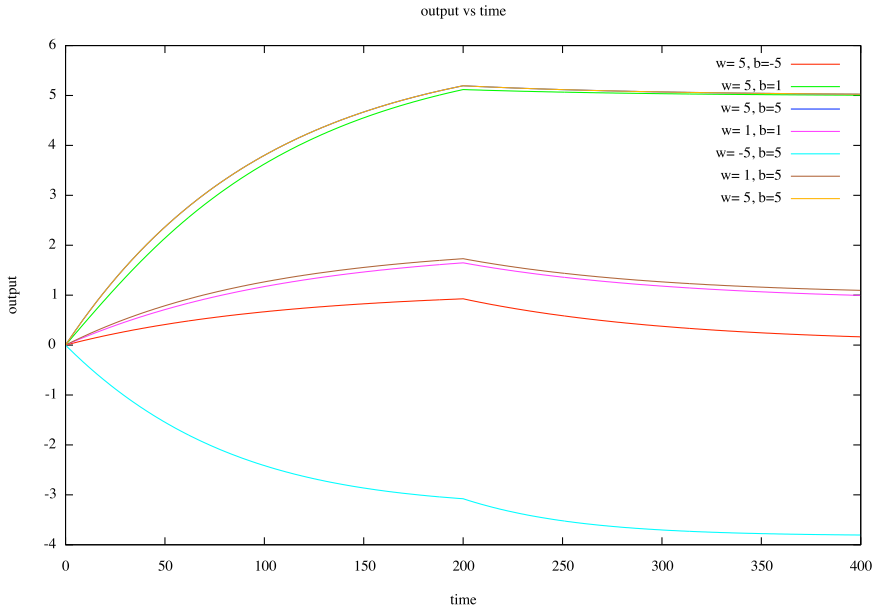

Figure 15: **Changing w_i, θ_i and their effect on system response**

$$\frac{dy_i}{dt} = 0 \tag{19}$$

$$\therefore f(y; w_i, \theta_i, I_i) = 0 \tag{20}$$

$$-y_i + w_i\sigma(y_i + \theta_i) + I_i = 0 \tag{21}$$

To analyse the fixed points of this system we would like to solve (21) which can be rewritten as (23). Unfortunately there is no algebraic solution for (23) [1].

However, we can numerical compute the fix points and analyse their stabilitys by examining the value and sign of derivative f' given in (22).

$$f' = \frac{\partial f}{\partial y} = -1 + w_i\sigma'(y + \theta) \tag{22}$$

$$I_i = y_i - w_i\sigma(y_i + \theta_i) \tag{23}$$

Dynamical systems analysis [3] states if a_0 is a fixed point, then if $f'(a_0) < 0$ then a_0 is a maxima and $f'(a_0) > 0$ the point is a minima. The stability of the fixed point is given by the sign of the derivative. If $|f'(a_0)| > 1$ then the point is unstable and if $|f'(a_0)| < 1$ the point is stable.

Recalling that $0 \leq \sigma' \leq 1/4$, Beer states that:

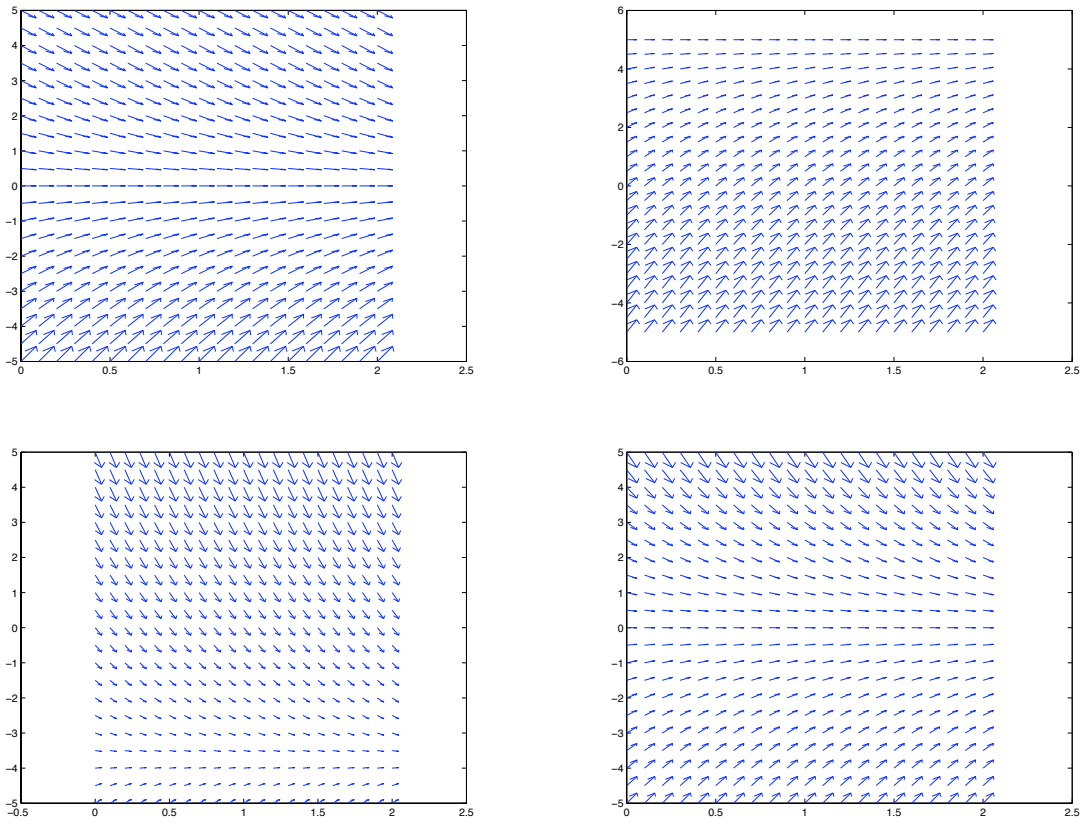

Figure 16: **Direction Fields for** $[w_i = 5, \theta_i = -5]$**,** $[w_i = 5, \theta_i = 5]$**,** $[w_i = -5,\ \theta_i = 5]$ **and** $[w_i = -5, \theta_i = -5]$

> $f' < 0$ when $w < 4$ and thus only stable equilibria are possible in this case. On the other hand, when $w > 4$, the sign of f' depends on y, w and θ. [1]

and we will get marked behaviour differences when w_i passes through 4.

Following Beer's lead, we use (23) to numerically compute the fixed points for $\theta \in [-5, 5]$ using an extended weight range of $w_i \in [-20 : 20]$ to amplify the difference, although the same effects are observed in the typical weight range used in this study. Note that we follow Beer's convention of placing I on the x-axis and y on the y axis.

Figure 17 shows that for $w_i = -20$ there is only one equilibrium point for all I, when $I = -10$. This can be verified by substituting the values in (21):

$$[y = 0, w_i = -20, \theta = 0, I = -10] \Rightarrow 0 + (-20 * \sigma(0)) - 10 = (-20 * 0.5) - 10 = 0 \qquad (24)$$

The absolute value of the derivative, $|f'(0)| = -6$, therefore the point is unstable. Perturbing around the fixed point confirms it is unstable, using $h = 0.1$ shows the trend to move away from the fixed point:

y(0) = 0 + 0.010 → y(1h) = -0.196 → y(2h) = -0.379 → y(3h) = -0.557 etc.

y(0) = 0 - 0.010 → y(1h) = -0.209 → y(2h) = -0.397 → y(3h) = -0.573 etc.

This conflicts with Beer's statement that only stable equilibrium points are present when $w_i < 4$. Beer seems to have overlooked that if $w_i < 0$ then $|f'| \geq 1$ because any negative weight will add on to the -1 in the derivative, -1 + (-weight * sigmoid(y)), so when $w_i < 0$ the fixed point will be unstable. $\therefore$ there is only a fixed single stable point when $0 < w_i < 4$.

Plotting y against f', Figure 18, shows the effects of weight on the value $|f'|$, demonstrating that only when $0 < w < 4$, will we have a single stable equilibria, ie $|f'| < 1$. If $w < 0$ then we will have a single unstable point. When $w > 4$, the plots follow that of $w = 20$ in Figure 17.

When $w_i = 20$, $|f'|$ changes from < 1 to > 1 and back to < 1, as we progress along the plot. From visual inspection it can be seen that there are three equilibria points when $y = -10, 0, 10$ and $I = -10$. It can also be seen that the the first equilibrium is stable, the second is unstable and the final one is stable.

The stability of the equilibria can be confirmed by numeric analysis. Examining f', we see that $|f'(-10)| = 0.999$ which is < 1 so its stable, $|f'(0)| = 4$ which is > 1 so its unstable and $|f'(10)| = 0.999$ so its stable.

Finally we will plot the surface of the equilibrium points as w_i varies, which shows a fold as w_i passes 4. As Beers notes:

> Whenever the values of I or w cross into or out of this fold [the system] undergoes a bifurcation, that is, its dynamical behaviour switches between two qualitively different phase portraits .

these phase portraits can be seen as switching between having three fixed points to having just one and vice versa. Figure 19 shows the output for $f(y, w_i, \theta_i)$.

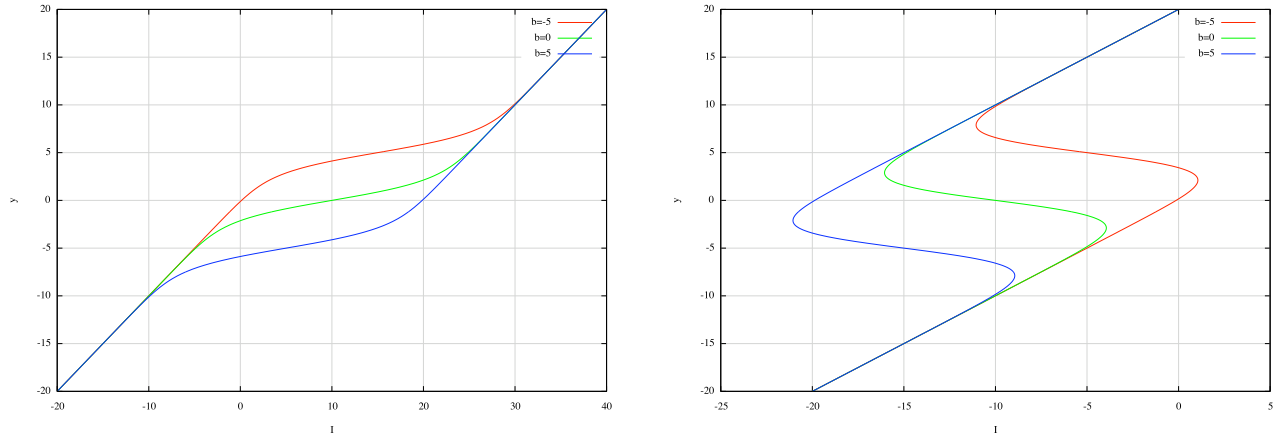

Figure 17: I **plotted against** y **for** $w_i = -20$ **and** $w = 20$

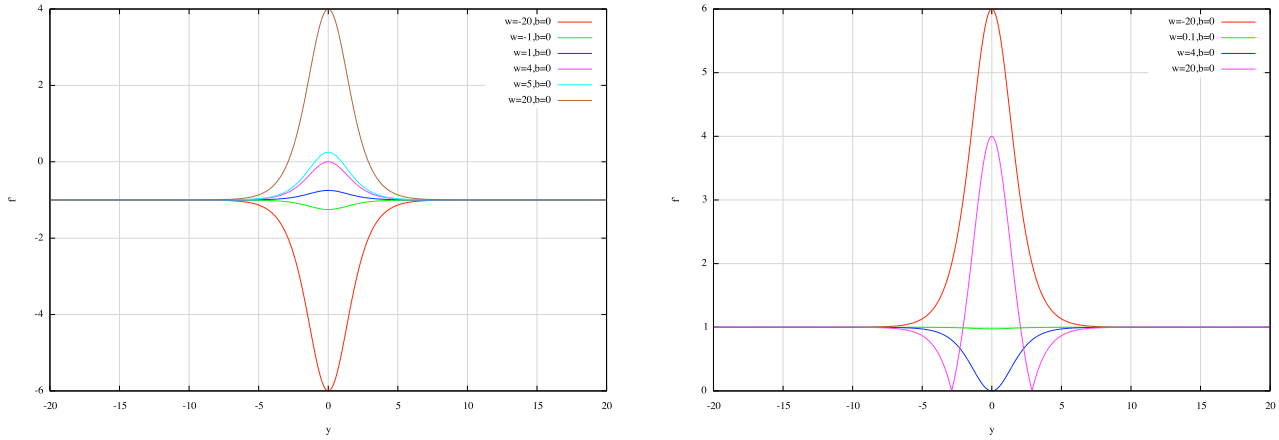

Figure 18: **Plotting** f' **for varying values of** w_i **and** $\theta = 0$. f' is shown on the right. $|f'|$ on the left.

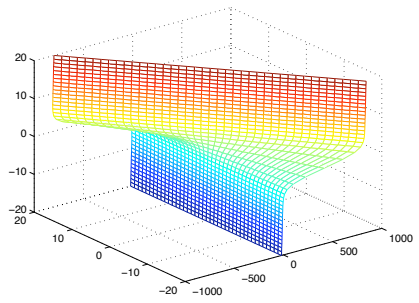

Figure 19: **Changing weights and bias and their effect on system response**

References

[1] Beer, R.D. (1995). *On the Dynamics of Small Continuous-Time Recurrent Neural Networks*. Adaptive Behavior, 3(4), (pp469-509).

[2] Beer, R.D. (1996). *Toward the evolution of dynamical neural networks for minimally cognitive behavior*. In P. Maes, M. Mataric, J. Meyer, J. Pollack and S. Wilson (Eds.), From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (pp. 421-429). MIT Press.

[3] Hale, J. and Kocak, H. *Dynamics and Bifurcations*. Springer-Verlag, 1991.

[4] Mathayomchan, B. and Beer, R.D. (2002). *Center-crossing recurrent neural networks for the evolution of rhythmic behavior*. Neural Computation 14, (pp 2043-2051).

[5] Weisstein, Eric W. "Courant-Friedrichs-Lewy Condition." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Courant-Friedrichs-LewyCondition.html

4.4 Appendix

The following code was used in each section:

3.1 Numerical Integration of a Node

```
%% Use initial value of x=1 and x=-1
%% h is the time step for integration
%% t is the length of simulation
function SimpleNode (h, t)

    NumTimeSteps = t/h;

    x = 1;

    SimulateNode(x, 1, h, t);
    hold on;

    x = -1;
    SimulateNode(x, 1, h, t);
    hold off;
end


%% Use euler to integrate
%% x is the initial value
%% tau is the time constant
%% h is the step size for integration
%% t is the simulation length
function SimulateNode (x, tau, h, t)

    NumTimeSteps = t/h;
    X            = zeros(1, NumTimeSteps);
    oldx         = x;
    X(1)         = oldx;
    tau_value    = 1/tau;

    for TStep = 1:NumTimeSteps
        newx = oldx + (h * (tau_value * -oldx));
        X(TStep+1) = newx;

        oldx = newx;
    end

    % Now display
    t = 0:NumTimeSteps;
    max(X);

    str = sprintf(';x= %g;',x);
    xlabel('time'), ylabel('output'), title('output vs time');
    plot(t, X, str);
end
```

3.2 Effect of Tau on the Node

```
%% Minimal CTRNN - investigates the effect of tau on the system
%% x is the initial value of x
%% h is the time step for integration
%% t is the length of simulation
function MinimalCTRNN (x, h, t)

    NumTimeSteps = t/h;

    for tau = 0.2:0.2:2.0
        X            = zeros(1, NumTimeSteps);
        oldx         = x;
        X(1)         = oldx;
        tau_value    = 1/tau;

        for TStep = 1:NumTimeSteps

            newx = oldx + (h * (tau_value * -oldx));
            X(TStep+1) = newx;

            oldx = newx;
        end

        % Now display
        t = 0:NumTimeSteps;
        max(X);

        str = sprintf(';T_i= %g;', tau);
        xlabel('time'), ylabel('output'), title('output vs time');
        plot(t, X, str);
        hold on;
    end

    hold off;
end
```

3.2.1 Time Constant Effect on a CTRNN Node

```
%% Measures the effect of t -> h
%% h is the time step for integration
%% t is the length of simulation
function MinimalCTRNN2 (x, h, t)

    NumTimeSteps = t/h;

    tau = h/2;
        X            = zeros(1, NumTimeSteps);
        oldx         = x;
        X(1)         = oldx;
        tau_value    = 1/tau;

        for TStep = 1:NumTimeSteps

            newx = oldx + (h * (tau_value * -oldx));
```

```
            X(TStep+1) = newx;

            oldx = newx;
        end

        % Now display
        t = 0:NumTimeSteps;
        max(X);

        str = sprintf(';T_i= %g;', tau);

        xlabel('time'), ylabel('output'), title('output vs time');
        plot(t, X, str);
        hold on;

    tau = h;
    X            = zeros(1, NumTimeSteps);
        oldx         = x;
        X(1)         = oldx;
        tau_value    = 1/tau;

        for TStep = 1:NumTimeSteps

            newx = oldx + (h * (tau_value * -oldx));
            X(TStep+1) = newx;

            oldx = newx;
        end

        % Now display
        t = 0:NumTimeSteps;
        max(X);

        str = sprintf(';T_i= %g;', tau);

        xlabel('time'), ylabel('output'), title('output vs time');
        plot(t, X, str);
        hold on;

    hold off;
end
```

3.3 Adding Input

```
%% Minimal CTRNN3
%% Shows the effect of varying Tau and input for a CTRNN Node
%% h is the time step for integration
%% t is the length of simulation
function MinimalCTRNN3 (x, h, t)

        NumTimeSteps = t/h;                         % Num of integration steps
        HalfTimeStep = 2/h                          % halfway point
        tau = [0.5, 1, 2];
        I   = [-4,  4];
        t = 0:NumTimeSteps;
```

```
        for i=1:3
            for j=1:2
                X            = zeros(1, NumTimeSteps);
                oldx         = x;
                X(1)         = oldx;
                tau_value    = 1 / tau(i);

                for TStep = 1:NumTimeSteps

                    if (TStep < HalfTimeStep)
                        delta_x = tau_value * (-oldx + I(j));
                    else
                        delta_x = tau_value * (-oldx);          %% I == 0
                    end

                    newx = oldx + (h * delta_x);
                    X(TStep+1) = newx;

                    oldx = newx;
                end

                % Now display
                str = sprintf(';I= %g, T=%g;', I(j), tau(i))

                %xlabel('time'), ylabel('output'), title('output vs time');
                plot(t, X, str);
                hold on;
            end
        end
        hold off;
end
```

4.1 Sigmoid Activation Function

Used Sigmoid Plot with bias = 0, and gain =1

```
% Standard logistic activation function
function s = Sigmoid (x)
    s = 1;
    s = s / (1 + exp(-x));
end


%% Plots the response of the Sigmoid function
%% b is the bias
%% g is the gain
function SigmoidPlot (b, g)

    X     = -10:0.1:10;
    dim    = size(X);
    length = dim(1,2);
    S      = zeros(1, length);

    % Calculate Sigmoid
```

```
    for i=1:length
        S(i)  = Sigmoid(g * (X(i) + b));
    end

    str = sprintf(';g=%g b=%g;', g, b);
    plot(X, S, str);
end


% Standard logistic activation function
% Sigmoid
function s = SigmoidDerivative (x)
    s = exp(-x);
    d = (1 + exp(-x)) ^ 2;
    s = s / d;
end


%% Plots the response of the derivative of the
%% Sigmoid function
%% b is the bias
%% g is the gain
function SigmoidDerivativePlot (b, g)

    X     = -10:0.1:10;
    dim    = size(X);
    length = dim(1,2);

    % Calculate change in sigmoid
    dS = zeros(1, length);

    %% Plot derivative
    %hold on;

    for i=1:length
        dS(i) = SigmoidDerivative(g * (X(i) + b));
    end

    plot (X, dS);
end
```

4.2.1 Effect of bias on System

Used SigmoidPlot to show effect of bias on Sigmoid Function.

```
%% Single recurrent CTRNN node with Bias
%% x is the initial value of the node
%% h is the time step for integration
%% t is the length of simulation
function BiasNode (x, h, t)

        NumTimeSteps = t/h;                          % Num of integration steps
        tau = 1;

        for theta = -4:2:4
            X            = zeros(1, NumTimeSteps);
```

```
            oldx         = x;
            X(1)         = oldx;
            I            = 0;           %% no input
            w            = 1;           %% connection is on but no multiplier

            for TStep = 1:NumTimeSteps


                delta_x = -oldx + (w * Sigmoid(oldx + theta)) + I;

                newx = oldx + (h * delta_x);
                X(TStep+1) = newx;

                oldx = newx;
            end

            % Now display
            t = 0:NumTimeSteps;
            str = sprintf(';Theta= %g;', theta);

            %xlabel('time'), ylabel('output'), title('output vs time');
            plot(t, X, str);
            hold on;
        end

        hold off;
end
```

4.2.2 Effect of gain on system

Used SigmoidPlot to show effect of gain on Sigmoid Function.

```
%% Single recurrent CTRNN node with Gain
%% x is the initial value of the node
%% h is the time step for integration
%% t is the length of simulation
function GainNode (x, h, t)

        NumTimeSteps = t/h;         % Num of integration steps
        tau = 1;

        for g = -5:5
            X            = zeros(1, NumTimeSteps);
            oldx         = x;
            X(1)         = oldx;
            I            = 0;           %% no input
            w            = 1;           %% connection is on but no multiplier
            theta        = 1;

            for TStep = 1:NumTimeSteps
                delta_x = -oldx + (w * Sigmoid(g*(oldx + theta))) + I;

                newx = oldx + (h * delta_x);
                X(TStep+1) = newx;
```

```
            oldx = newx;
        end

        % Now display
        t = 0:NumTimeSteps;
        str = sprintf(';g= %g;', g);

        %xlabel('time'), ylabel('output'), title('output vs time');
        plot(t, X, str);
        hold on;
    end

    hold off;
end
```

4.2.3Effect of weight changes on the network

4.2.4 A more complete model

```
%% Minimal CTRNN node with recurrent connection
%% x is the initial value
%% h is the time step for integration
%% t is the length of simulation

function RecurrentNode (x, h, t)

    NumTimeSteps = t/h;                         % Num of integration steps
    tau = 1;

    for w = -4:2:4
        X           = zeros(1, NumTimeSteps);
        oldx        = x;
        X(1)        = oldx;
        I           = 0;                        % No input
        theta       = 0;                        % No bias

        for TStep = 1:NumTimeSteps

            delta_x = -oldx + (w * Sigmoid(oldx + theta)) + I;

            newx = oldx + (h * delta_x);
            X(TStep+1) = newx;

            oldx = newx;
        end

        % Now display
        t = 0:NumTimeSteps;
        str = sprintf(';W= %g;', w);

        %xlabel('time'), ylabel('output'), title('output vs time');
        plot(t, X, str);
```

```
            hold on;
        end

        hold off;
end


%% Shows phase portrait for a given weight and bias
function FullNodePhase (w, bias)

    [T, X] = meshgrid([0:0.1:2], [-5:0.5:5]);
    dT = ones(size(T));
    %dX = -X + (w * ((1-exp(-X)).^-1));

    dX = -X + (w * ((1 + exp(-(X + bias))).^(-1)) );

    quiver(T,X,dT,dX)

end
```

4.3 Analysis of Single CTRNN as a dynamical System

```
% Plots the equilibria for % I = y - w * sigma (y + bias)
% w is the weight of the connection
% b is the bias
function PlotEquilibria (w, b)

    % I = y - w * sigma (y + bias)

    Y=-20:0.1:20;

    % plot for weight and bias
    str = sprintf(';b=%g;', b);
    xlabel("I");
    ylabel("y");
    plot (Y - (w *((1 + exp(-(Y + b))).^(-1)) ), Y, str);
end


% Plots the equilibrium as a function of I
% Will plot for a bias = -5, 0 and 5
% W is the weight of the connection
function PlotEquilibriaRange (w)

    % I = y - w * sigma (y + bias)

    Y=-20:0.1:20;

    % plot for bias = -5
    b = -5;
    PlotEquilibria(w, -5);
    hold on

    % plot for bias = -0
    PlotEquilibria(w, 0);
```

```
    % plot for bias = 5
    PlotEquilibria(w, 5);

    hold off
end


% Plots the equilibrium as a function of I
% where b is the bias
% and w is the weight of the connection
function PlotEquilibria2 (w, bias)

    % I = y - w * sigma (y + bias)
    % y = f(x)

    X=-20:0.1:20;

    % plot for bias
    str = sprintf(';w=%g, b=%g;', w, bias);
    plot (X, X - (w *((1 + exp(-(X + bias))).^(-1)) ), str)

    hold on
    % I = y
    plot(X, X);


    hold off
end


function WeightMesh

    % I = y - w * sigma (y + bias)
    [y, w] = meshgrid(-20:20, -20:20);
    %i = y - (w *((exp(-y) * (((1 + exp(-(y))).^(-1))^2) )));
    i = y - (w * ((1 + exp(-(y + 0))).^(-1)));
    xlabel('i');
    ylabel('w');

    mesh(i, w, y);
end
```