Background on Neural Networks

Dr. Rashmi N Ugarakhod

Department of ECE, PESU

RR Campus

Introduction

- The uses of NN in closed-loop control are dramatically distinct from their uses in open-loop applications, which are mainly in digital signal processing (DSP).
- DSP: classification, pattern recognition, and approximation of nondynamic functions (e.g. with no integrators or time delays).
- NN usage is backed up by a body of knowledge developed over the years that shows how to choose network topologies and select the weights to yield guaranteed performance.
- The issues associated with weight training algorithms are well understood.
- Closed-loop control of dynamical systems, most applications have been ad hoc, with open-loop techniques (e.g. backpropagation weight tuning) employed in a naive yet hopeful manner to solve problems associated with dynamic neural net evolution within a feedback loop, where the NN must provide stabilizing controls for the system as well as maintain all its weights bounded.
- Most published papers have consisted of some loose discussion followed by some simulation examples. By now, several researchers have begun to provide rigorous mathematical analyses of NN in closed-loop control applications.
- The background for these efforts was provided by Narendra and co-workers in several seminal works.
- It has generally been discovered that standard open-loop weight tuning algorithms such as backpropagation or Hebbian tuning must be modified to provide guaranteed stability and tracking in feedback control systems.

Neural network Topologies and Recall

- Neural networks are closely modeled on biological processes for information processing, including specifically the **nervous system** and its basic unit, the **neuron**.
- Signals are propagated in the form of potential differences between the inside and outside of cells.

Neuron Anatomy

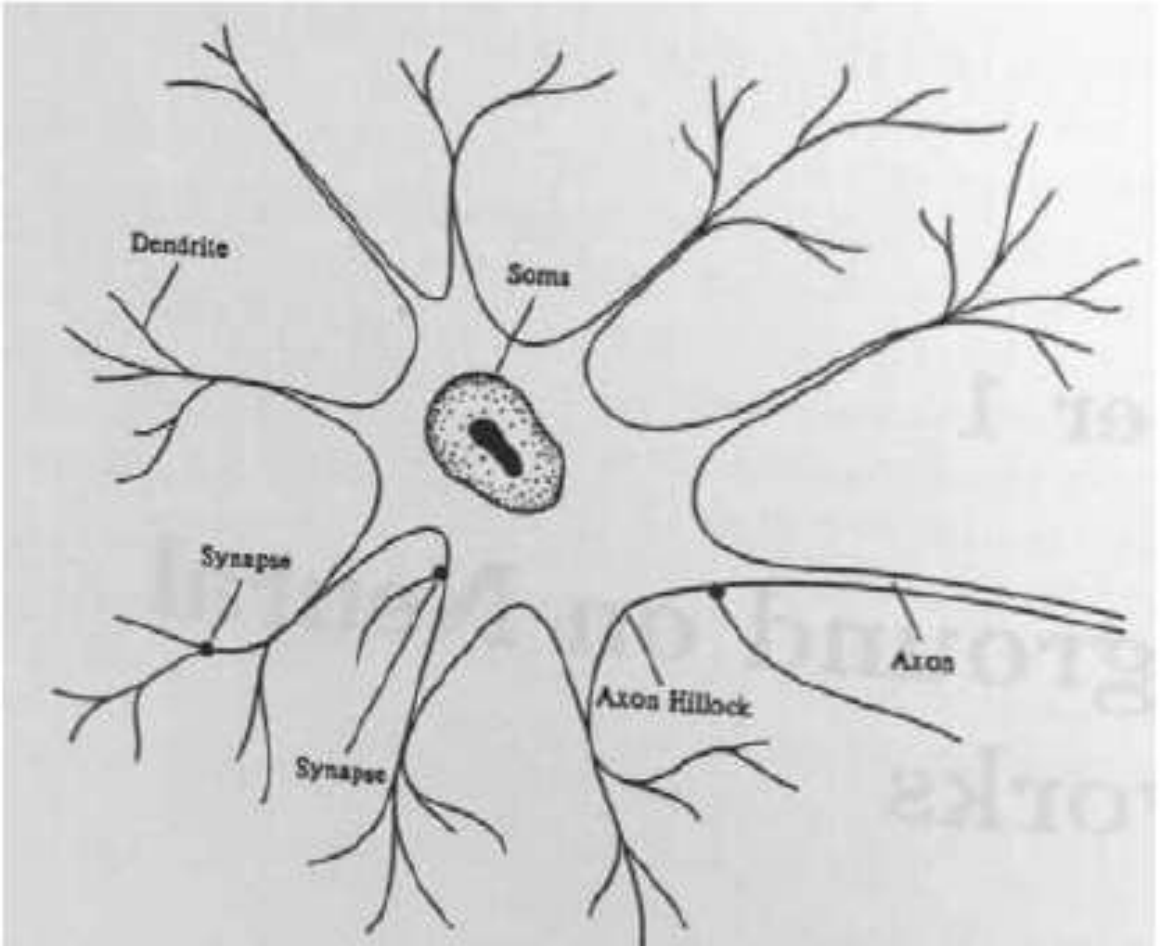
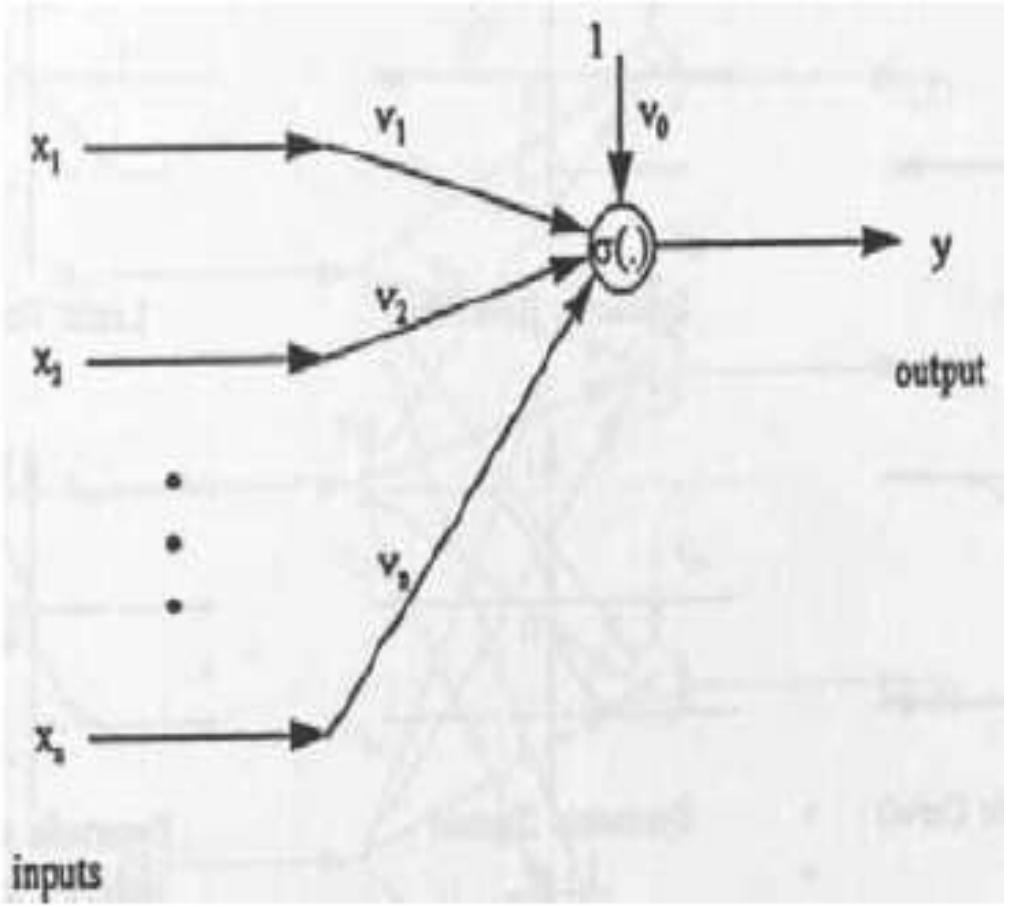


Fig. 1.1.1

- **Dendrites** bring signals from other neurons into the cell body or soma, possibly multiplying each incoming signal by a transfer weighting coefficient.
- In the **Soma**, cell capacitance integrates the signals which collect in the **Axon hillock**.
- The **Axon** connects through synapses with the dendrites of subsequent neurons.
- The **Synapses** operate through the discharge of neurotransmitter chemicals across intercellular gaps, and can be either excitatory (tending to fire the next neuron) or inhibitory (tending to prevent firing of the next neuron).

Neuron Mathematical Model



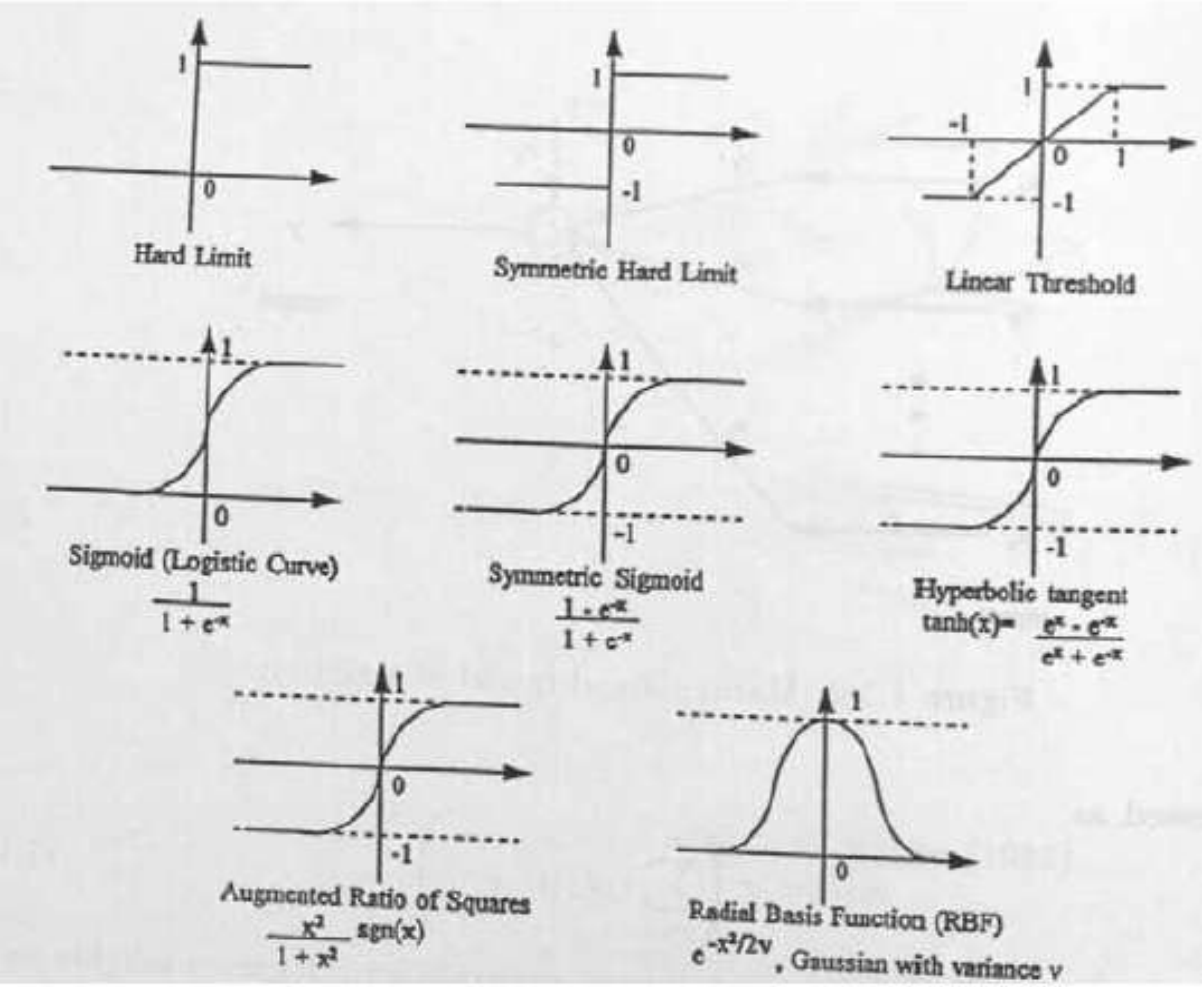
$$y(t) = \sigma\left(\sum_{j=1}^{n} v_j x_j(t) + v_0\right)$$

Positive weights, v_j: Excitatory synapses

Negative weights, v_j: Inhibitory synapses

Perceptron by Rosenblatt in 1959 (Haykin 1994)

Activation Function



- $\sigma(.)$ represents activation function.
- Model the behavior of the cell where there is no output below a certain value of the argument of $\sigma(\cdot)$
- Output takes a specified magnitude above that value of the argument.
- A general class of monotonically nondecreasing functions taking on bounded values at $-\infty$ and $+\infty$ is known as the sigmoid functions.
- Change in threshold or bias v_0 → activation functions shift left or right.
- Derivative of $\sigma(\cdot)$ is needed so that the activation function selected must be differentiable.

- The expression for the neuron output $y(t)$ can be streamlined by defining the column vector of the input signals $\bar{x}(t) \in \mathcal{R}^n$ and the column vector of NN weights $\bar{v}(t) \in \mathcal{R}^n$ as

$$\bar{x}(t) = [x_1\ x_2 \ldots x_n]^T, \bar{v}(t) = [v_1\ v_2 \ldots v_n]^T$$

- In matrix notation,

$$y = \sigma(\bar{v}^T \bar{x} + v_0)$$

- Defining the augmented input column vector $x(t) \in \mathcal{R}^{n+1}$ and NN weight column vector $v(t) \in \mathcal{R}^{n+1}$ as

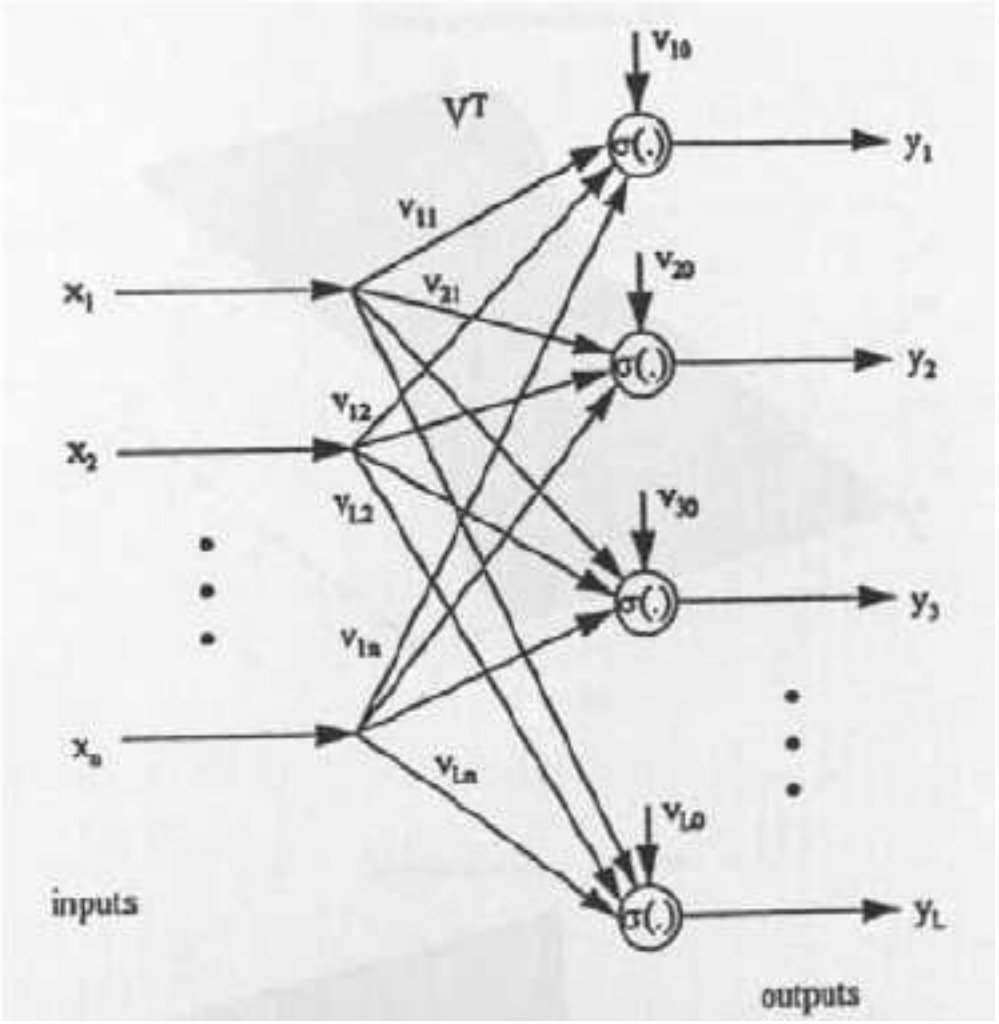
$$x(t) = [1\ \ \bar{x}^T]^T = [1\ \ x_1\ x_2 \ldots x_n]^T$$
$$v(t) = [v_0\ \ \bar{v}^T]^T = [v_0\ v_1\ v_2 \ldots v_n]^T$$

- Then, the output can be written as:

$$y = \sigma(v^T x)$$

- These expressions for the neuron output *y*(*t*) are referred to as the ***cell recall mechanism***.
- They describe how the output is reconstructed from the input signals and the values of the cell parameters.

One-layer neural network



- In one-layer Neural network (NN), it consists of *L* cells, all fed by the same input signals $x_j(t)$ and producing one output $y_l(t)$ per neuron.
- The recall equation for this network is given by

$$y_l = \sigma\left(\sum_{j=1}^{n} v_{lj}x_j + v_{l0}\right); l = 1,2,\dots,L$$

- By defining the matrix of weights and the vector of thresholds as:

$$\bar{v}^T \equiv \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & & \vdots \\ v_{L1} & v_{L2} & \cdots & v_{Ln} \end{bmatrix}, b_v \equiv \begin{bmatrix} v_{10} \\ v_{20} \\ \vdots \\ v_{L0} \end{bmatrix}$$

- The output vector $y = [y_1\ y_2\ \dots y_L]^T$ as

$$y = \bar{\sigma}(\bar{v}^T \bar{x} + b_v)$$

- The vector activation function is defined for a vector $w \equiv [w_1\ w_2\ \dots w_L]^T$ as

$$\bar{\sigma}(w) \equiv [\ \sigma(w_1)\ \ \sigma(w_2) \dots \sigma(w_L)]^T$$

- Defining the augmented matrix of weights as:

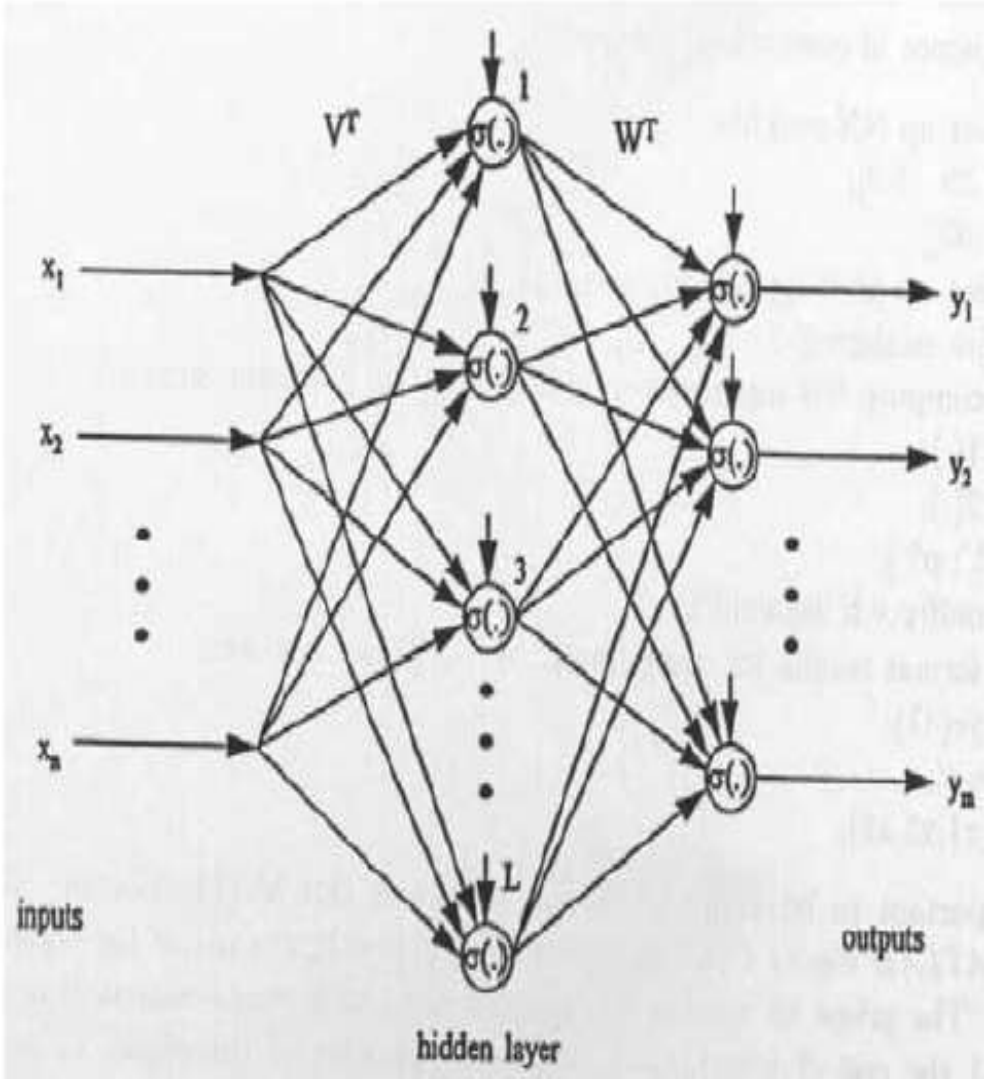
$$V^T \equiv \begin{bmatrix} v_{10} & v_{11} & v_{12} & \cdots & v_{1n} \\ v_{20} & v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & & \vdots & & \vdots \\ v_{L0} & v_{L1} & v_{L2} & \cdots & v_{Ln} \end{bmatrix}$$

- Then the NN outputs can be expressed as follows:

$$y = \overline{\sigma}(V^T x)$$

Simulation

Multilayer Perceptron



- The output of the two-layer NN is given by the recall equation:

$$y_i = \sigma\left(\sum_{l=1}^{L} w_{il}\,\sigma\left(\sum_{j=1}^{n} v_{lj}x_j + v_{l0}\right) + w_{i0}\right);$$
$$i = 1,2,\ldots\ldots,m$$

- Defining the hidden layer outputs z_l;

$$z_l = \sigma\left(\sum_{j=1}^{n} v_{lj}x_j + v_{l0}\right); l = 1,2,\ldots,L$$

$$y_i = \sigma\left(\sum_{l=1}^{L} w_{il}z_l + w_{i0}\right); i = 1,2,\ldots,m$$

Two-layer neural network

- $\bar{W}^T \equiv \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1L} \\ w_{21} & w_{22} & \cdots & w_{2L} \\ \vdots & \vdots & & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mL} \end{bmatrix} \qquad b_w \equiv \begin{bmatrix} w_{10} \\ w_{20} \\ \vdots \\ w_{m0} \end{bmatrix}$
- $W^T \equiv \begin{bmatrix} w_{10} & w_{11} & w_{12} & \cdots & w_{1L} \\ w_{20} & w_{21} & w_{22} & \cdots & w_{2L} \\ \vdots & & \vdots & & \vdots \\ w_{m0} & w_{m1} & w_{m2} & \cdots & w_{mL} \end{bmatrix}$

- Therefore the NN output can be written as:

$$y = \bar{\sigma}(\bar{W}^T \bar{\sigma}(\bar{V}^T \bar{x} + b_v) + b_w) = \bar{\sigma}\left(W^T \sigma(V^T x)\right)$$

$$y = W^T \sigma(V^T x)$$

Simulation

Gaussian or Radial Basis Function(RBF) Networks

- A NN activation function often used is the gaussian or radial basis function (RBF) (Sanner and Slotine 1991) given when *x* is a scalar as

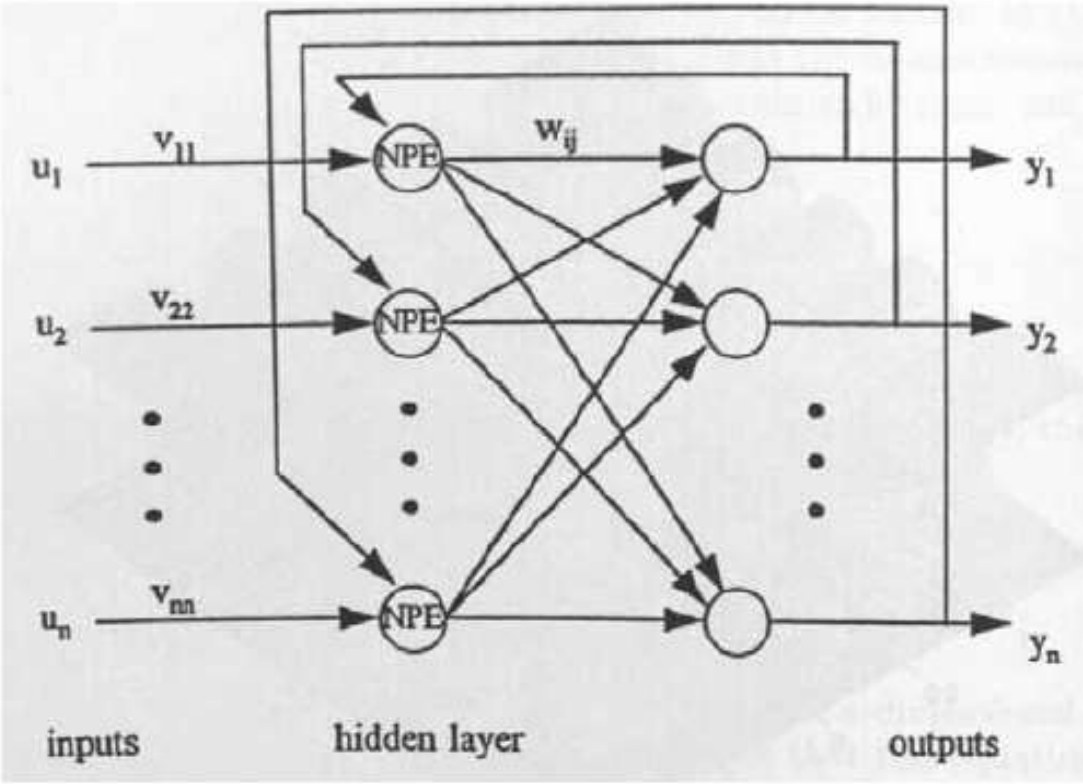
$$\sigma(x) = e^{-(x-\mu)^2/2p},$$

where μ is the mean and p is the variance.

- The RBF variances p_{jk} and offsets μ_{jk} are usually selected in designing the RBF NN and left fixed.
- The output-layer weights W^T are generally tuned. Therefore, the RBF NN is a special sort of FLNN.

Dynamic Neural Networks

- Static NN – no memory, no integrators or time-delay elements.
- Hopfield net: special form of a two-layer NN where the output y_i is fed back into the hidden layer neurons.
- First layer weight matrix V is the identity matrix I
- Second layer weight matrix W is square
- Output layer activation function is linear.



Dynamic Neural Networks

- Hidden layer neurons have increased processing power in the form of a memory.
- Neuronal Processing Elements (NPE) : Neurons with internal signal processing
- Continuous-time Hopfield net
- Discrete-time Hopfield net

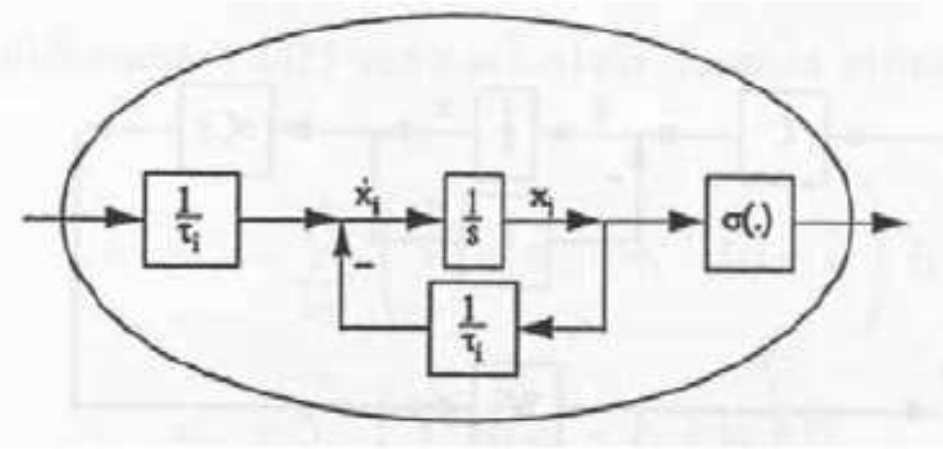
Continuous-Time Hopfield net

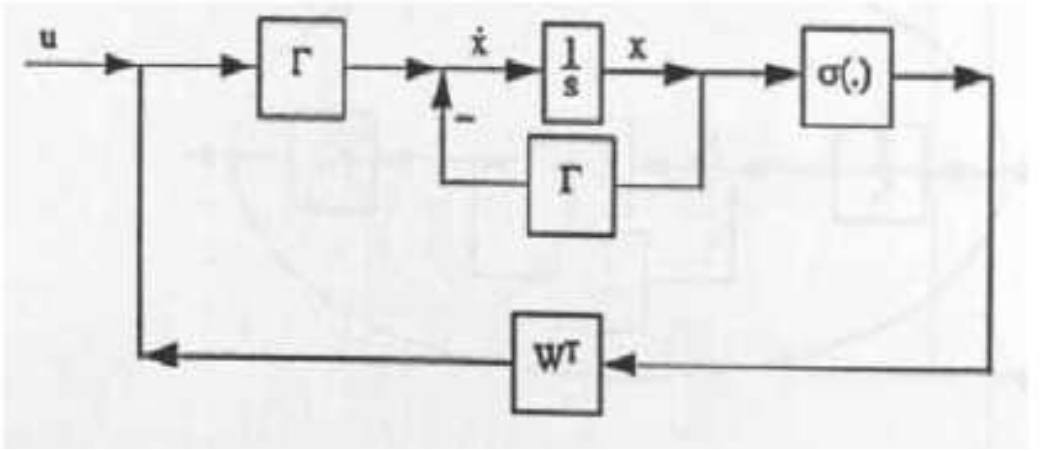
- NPE contains an integrator $\frac{1}{s}$ and a time-constant τ_i in addition to the usual nonlinear activation function σ(·).
- The internal state of the NPE is described by the signal $x_i(t)$.
- The continuous-time Hopfield net is described by the ordinary differential equation

$$\tau_i \dot{x}_i = -x_i + \sum_{j=1}^{n} w_{ij}\sigma_j(x_j) + u_i$$

with output equation

$$y_i = \sum_{j=1}^{n} w_{ij}\sigma_j(x_j)$$

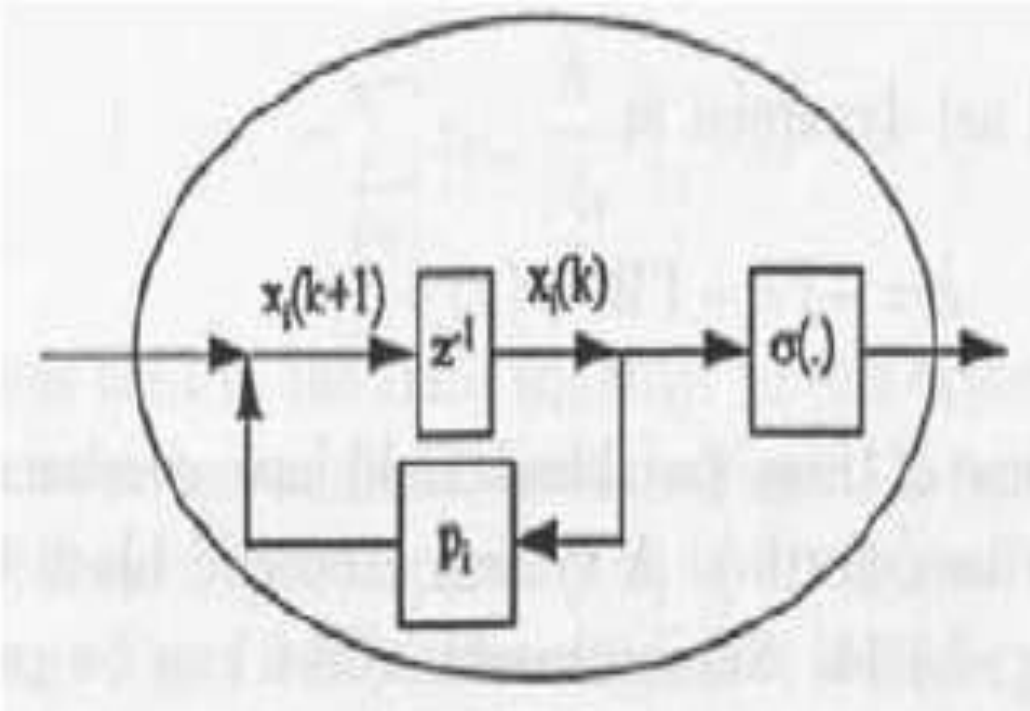




Discrete-Time Hopfield net

- In the discrete-time case, the internal dynamics of each hidden-layer NPE contains a time delay instead of an integrator.
- The NN is now described by the difference equation:

$$x_i(k+1) = p_i x_i(k) + \sum_{j=1}^{n} w_{ij}\sigma_j\left(x_j(k)\right) + u_i(k)$$



Simulation

Background on Dynamic Systems

Continuous-time systems

$$\dot{x} = F(x, u) \ - \text{State Equation}$$

$$y = H(x, u) - \text{Measurement Equation}$$

where $x(t) \in \mathbb{R}^n$ is the internal state vector

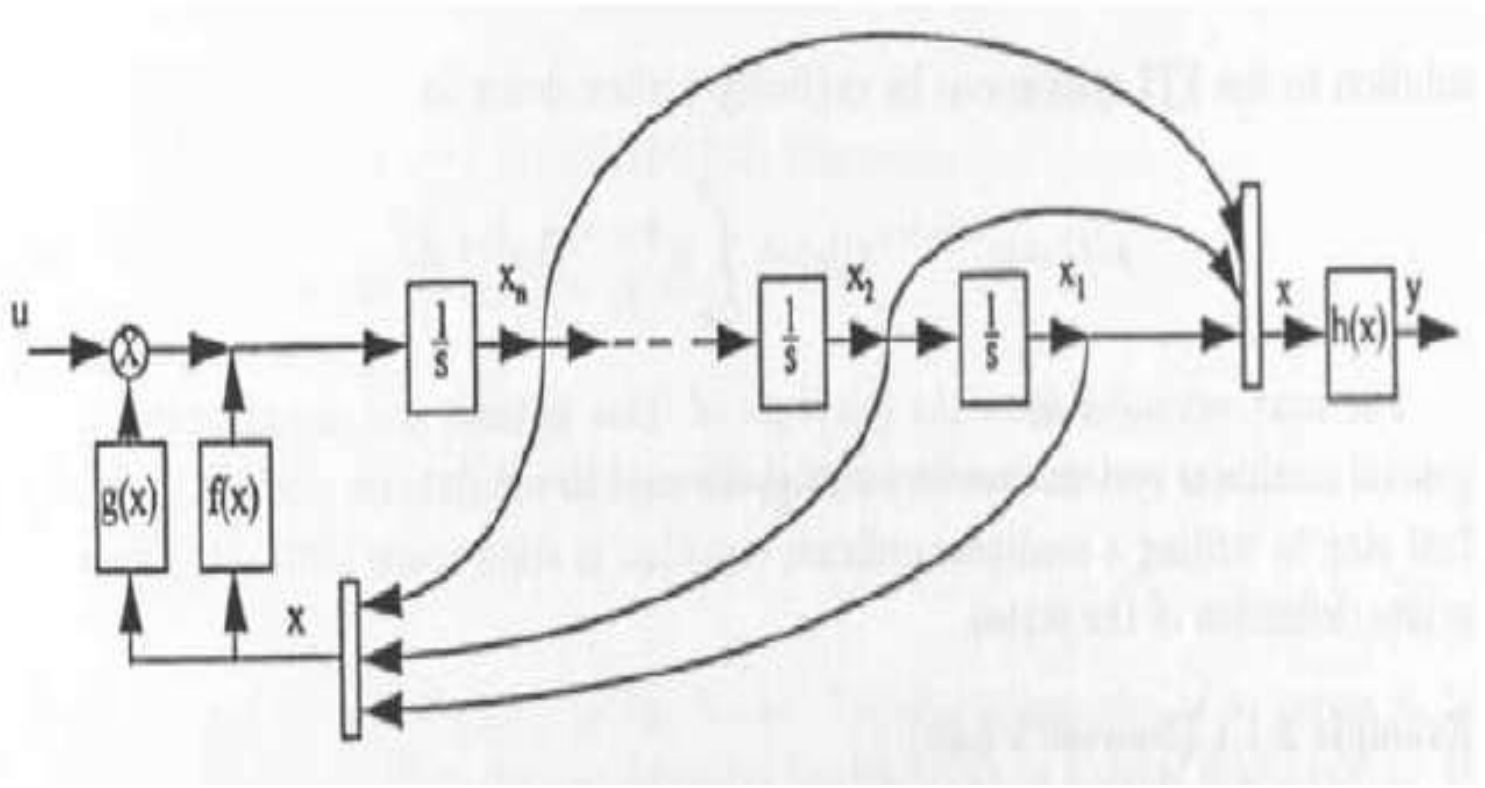
u$(t) \in \mathbb{R}^m$ is the control input

$y(t) \in \mathbb{R}^P$ is the measured system output

Brunovsky Canonical form

Let $x = [x_1, x_2, \dots, x_n]^T$

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = x_3$$
$$\vdots$$
$$\dot{x}_n = f(x) + g(x)u$$
$$y = h(x)$$



- The Brunovsky canonical form may equivalently be written as

$$\dot{x} = Ax + bf(x) + bg(x)u$$

Where

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & \vdots & & \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}; b = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

Linear Time Invariant Systems

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

Solution to the LTI system:

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau$$

Examples

- Newton's law:

$$F = ma$$

It can be written as:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = u$$

States are defined as x_1=position, x_2=velocity and the control input is $u = \frac{F}{m}$, where m is the mass and F is the input force.

Van der Pol Oscillator

Dynamics: $$\ddot{y} + \alpha(y^2 - 1)\dot{y} + y = u$$

Brunovsky canonical form:

Defining the states as $x_1 = position, x_2 = velocity$, then

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \alpha(1 - x_1^2)x_2 - x_1 + u$$

Simulation

Discrete Time systems

- A general class of discrete-time systems can be described by the nonlinear ordinary difference equation in *discrete state-space form:*

$$x(k+1) = F\big(x(k), u(k)\big)$$

$$y(k) = H(x(k), u(k))$$

- Brunovsky canonical form:

$$x_1(k+1) = x_2(k)$$

$$x_2(k+1) = x_3(k)$$

$$\vdots$$

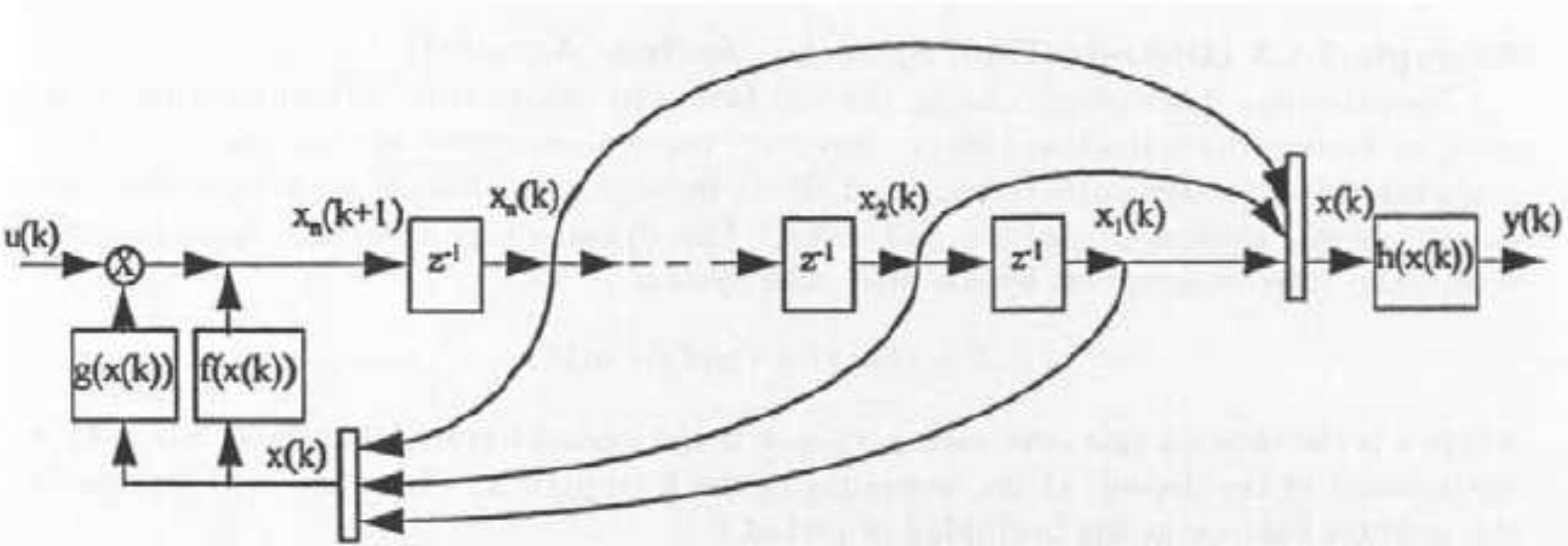
$$x_n(k+1) = f\big(x(k)\big) + g\big(x(k)\big)u(k)$$

$$y(k) = h(x(k))$$

- The discrete Brunovsky canonical form may equivalently be written as

$$x(k+1) = Ax(k) + bf\big(x(k)\big) + bg\big(x(k)\big)u(k)$$

Discrete-time single-input Brunovsky form



Linear Systems

- A special and important class of dynamical systems are the *discrete-time linear time-invariant (LTI) systems:*

$$x(k+1) = Ax(k) + Bu(k)$$

$$y(k) = Cx(k)$$

- Given an initial state $x(0)$ the solution to the LTI system can be explicitly written as:

$$x(k) = A^k x(0) + \sum_{j=0}^{k-1} A^{k-j-1} Bu(j)$$

Example-Savings Account

- The dynamics of a savings account using compound interest are given by the first-order system:

$$x(k+1) = (1+i)x(k) + u(k)$$

where,

- i is the interest rate over each period

-k is the period iteration number

-$u(k)$ is the amount of the deposit at the beginning of the k-th period

-$x(k)$ is the account balance at the beginning of period k.

Simulation

Properties of Dynamical systems

- Consider the dynamical system

$$\dot{x} = f(x, t)$$

Let the initial time be t_0 and the initial condition be $x_0 \equiv x(t_0)$.

- This system is said to be non-autonomous since the time t appears explicitly.
- If t does not appear explicitly in $f(.)$, the system is said to autonomous.
- A primary cause of explicit time dependence in control systems is the presence of time-dependent disturbances d(t).

- A state x_e is an equilibrium point of the system if $f(x_e, t) = 0, t \geq t_0$.
- If $x_0 = x_e$, so that the system starts out in the equilibrium state, then it will forever remain there.
- For linear systems, the only possible equilibrium point is $x_e = 0$.
- For nonlinear systems, x_e may be nonzero. In fact, there may be an equilibrium set, such as a limit cycle.

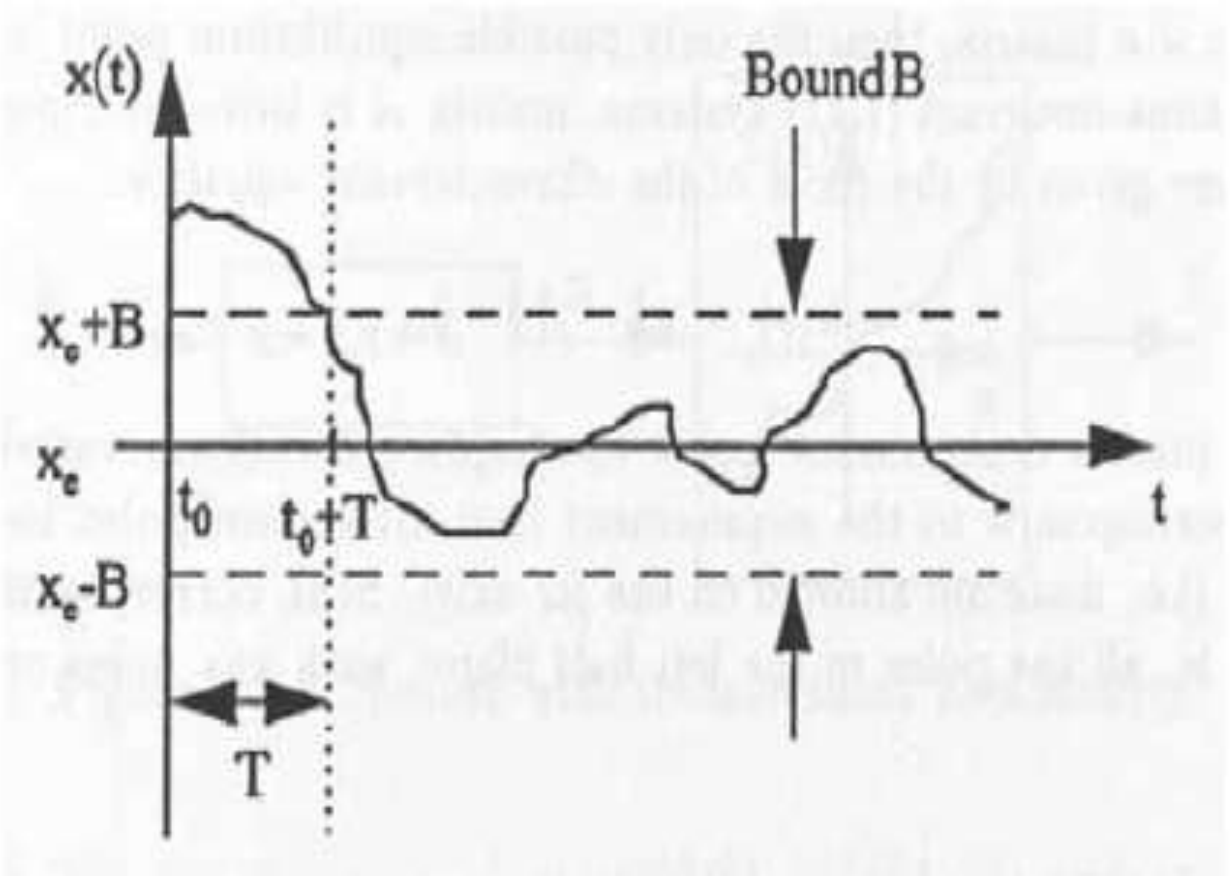
Asymptotic stability

- An equilibrium point x_e is locally asymptotically stable (AS) at t_0 if there exists a compact set S $\subset$ n such that, for every initial condition $x_0 \in$ S, one has $x(t) - x_e \to 0$ as $t \to \infty$. That is, the state $x(t)$ converges to x_e.
- If S = n so that $x(t) \to x_e$ for all $x(t_0)$, then x_e is said to globally asymptotically stable (GAS) at t_0.
- If the conditions hold for all t_0, the stability is said to uniform (e.g. UAS, GUAS).
- Asymptotic stability is a very strong property that is generally extremely difficult to achieve in closed-loop systems, even using advanced feedback controller design techniques.
- The primary reason is the presence of unknown but bounded system disturbances.
- A milder requirement is provided as follows.

Lyapunov Stability

- An equilibrium point x_e is stable in the sense of Lyapunov (SISL) at t_0 if for every $\varepsilon > 0$ there exists a $\delta(\varepsilon, t_0) > 0$ such that $\|x_0 - x_e\| < \delta(\varepsilon, t_0)$ implies that $\|x(t) - x_e\| < \varepsilon$ for $t \geq t_0$.
- The stability is said to be uniform (e.g. uniformly SISL) if $\delta(\cdot)$ is independent of t_0; that is, the system is SISL for all t_0.
- It is extremely interesting to compare these definitions to those of function continuity and uniform continuity.
- SISL is a notion of continuity for dynamical systems. Note that for SISL there is a requirement that the state x(t) be kept arbitrarily close to x_e by starting sufficiently close to it. This is still too strong requirement for closed-loop control in the presence of unknown disturbances.
- Therefore, a practical definition of stability to be used as a performance objective for feedback controller design is as follows.

Boundedness



- The equilibrium point x_e is said to be uniformly ultimately bounded (UUB) if there exists a compact set $S \subset n$ so that for all $x_0 \in S$ there exists a bound B and a time $T(B, x_0)$ such that $x(t) - x_e \leq B$ for all $t \geq t_0 + T$.
- The intent here is to capture the notion that for all initial states in the compact set S, the system trajectory eventually reaches, after a lapsed time of T, a bounded neighborhood of x_e.

- The difference between UUB and SISL is that in UUB the bound B cannot be made arbitrarily small by starting closer to x_e.
- In practical closed-loop systems, B depends on the disturbance magnitudes and other factors.
- If the controller is suitably designed, however, B will be small enough for practical purposes.
- The term uniform indicates that T does not depend on t_0.
- The term ultimate indicates that the boundedness property holds after a time lapse T.
- If $S = \mathbb{R}^n$ the system is said to be globally UUB (GUUB).

Observability

- Observability properties refer to the suitability of the measurements taken in a system; that is, the suitability of the choice of the measurement function h(·) in the output equation.
- A system with zero input u(t) = 0 is (locally) observable at an initial state x_0 if there exists a neighborhood S of x_0 such that, given any other state $x_1 \in S$, the output over an interval $[t_0, T]$corresponding to initial condition $x(t_0) = x_0$ is different from the output corresponding to initial condition $x(t_0) = x_1$.
- Then, the initial state can be reconstructed from output measurements over a time interval $[t_0, T]$.

- Consider the time-varying system

$$\dot{x} = A(t)x$$

$$y = C(t)x$$

And define the state-transition matrix $\varphi(t, t_0) \in \mathbb{R}^{n \times n}$ by

$$\frac{d}{dt}\varphi(t, t_0) = A(t)\,\varphi(t, t_0), \varphi(t_0, t_0) = I$$

- The key object in observability analysis is the observability gramian given by:

$$N(t_0, T) = \int_{t_0}^{T} \varphi^T(\tau, t_0) C^T C \varphi(\tau, t_0) d\tau$$

- The system is said to be uniformly completely observable(UCO) if there exist positive constants $\delta, \alpha_1, \alpha_2$ such that

$$\alpha_1 I \leq N(t_0, t_0 + \delta) \leq \alpha_2 I \text{ for all } t_0 \geq 0$$

- The state transition matrix for linear systems is given by:

$$\varphi(t, t_0) = e^{A(t-t_0)}$$

- Observability gramian is:

$$N(t_0, T) = \int_{t_0}^{T} e^{A^T(\tau-t_0)} C^T C e^{A(\tau-t_0)} d\tau$$

- The system is observable if, and only if, $N(t_0, T)$ has full rank n.
- This observability condition can be shown equivalent to the requirement that the observability matrix

$$V = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^n \end{bmatrix}$$ has full rank n.

- Matrix V is of full rank n if, and only if, the discrete-time observability gramian

$G_o = V^T V$ is nonsingular.

- If the system is observable and the input u(t) is zero, the initial state can be reconstructed from the output y(t) measured over the interval $[t_0, T]$ using the functional operator

$$x(t_0) = N^{-1}(t_0, T) \int_{t_0}^{T} e^{A^T(\tau - t_0)} C^T y(\tau) d\tau$$

Controllability

- Controllability properties refer to the suitability of the control inputs selected for a system; that is, the suitability of the choice of the input function g(·) in the state equation.
- A system is (locally) controllable at an equilibrium state x_e if there exists a neighborhood S of x_e such that, given any initial state $x(t_0) \in S$, there exists a final time T and a control input $u(t)$ on $[0, T]$ that drives the state from $x(t_0)$ to x_e .
- A system is (locally) reachable at a given initial state $x(t_0)$ if there exists a neighborhood S of $x(t_0)$ such that, given any prescribed final state $x_d(T) \in S$, there exists a final time T and a control input $u(t)$ on $[0, T]$ that drives the state from $x(t_0)$ to $x_d(T)$.

- For continuous LTI systems, reachability and controllability are the same and the key object in analysis is the controllability gramian

$$M(t_0, T) = \int_{t_0}^{T} e^{A(T-\tau)} B B^T e^{A^T (T-\tau)} d\tau$$

- The system is controllable (equivalently reachable) if $M(t_0, T)$ has full rank.
- It can be shown that if $M(t_0, T)$ has full rank for any $T > t_0$, then it has full rank for all $T > t_0$.
- This controllability condition can be shown to reduce to a full-rank condition on the controllability matrix

$$U = [B \quad AB \quad A^2 B \ \cdots \ A^n B]$$

- Matrix U is of full rank n if, and only if, the discrete-time controllability gramian

$G_C = UU^T$ is nonsingular.

- If the system is controllable, the initial state $x(t_0)$ can be driven to any desired final state $x_d(T)$ using the control input computed according to

$$u(t) = B^T e^{A^T(T-t)} M^{-1}(t_0, T)[x_d(T) - e^{AT} x(t_0)]\,, t \in [t_0, T]$$

- In the case of discrete LTI systems a similar analysis holds, but then reachability is stronger than controllability.
- That is, for discrete systems it is easier to drive nonzero initial states to zero than it is to drive them to prescribed nonzero final values.

Feedback Linearization and Control System Design

- Feedback linearization techniques:
  - Input-state feedback linearization
  - Input-output (I/O) feedback linearization.
- The former requires a complex set of mathematical tools including Frobenius Theorem and Lie algebra.
- The control laws derived are often complex due to the need to determine nonlinear state space transformations and their inverses.
- On the other hand, i/o feedback linearization is direct to apply and represents more of an engineering approach to control systems design.
- It is very useful for large classes of nonlinear controls problems including robot manipulators, mechanical systems, and other Lagrangian systems.

I/O Feedback Linearization

- I/O feedback linearization as a controller design technique for systems of the form:

$$\dot{x} = F(x, u)$$
$$y = h(x)$$

- Sample plant and problem specification:

Given the system or plant dynamics

$$\dot{x}_1 = x_1 x_2 + x_3$$
$$\dot{x}_2 = -2x_2 + x_1 u$$
$$\dot{x}_3 = \sin x_1 + 2x_1 x_2 + u$$

- It is desired to design a tracking controller that causes $x_1(t)$ to follow a desired trajectory $y_d(t)$ which is prescribed by the user.

I/O Feedback Linearization Step:

- Select the output

$$y(t) = x_1(t)$$

- Differentiate y(t) repeatedly and substitute state derivatives until the control input u(t) appears.

$$\ddot{y} = f(x) + g(x)u$$

- Define variables as $z_1 \equiv y, z_2 \equiv \dot{y}$ so that

$$\dot{z}_1 = z_2$$
$$\dot{z}_2 = f(x) + g(x)u$$

- This may be converted to a linear system by redefinition of the input as

$$v(t) \equiv f(x) + g(x)u(t)$$

So that

$$u(t) = \frac{1}{g(x)}\left(-f(x) + v(t)\right)$$

- Then

$$\dot{z}_1 = z_2$$
$$\dot{z}_2 = v$$

- It is equivalent to

$$\ddot{y} = v \quad \textbf{(Feedback Linearized System)}$$

- Controller Design Step:

- Standard linear system techniques can now be used to design a tracking controller for the feedback linearized system.

- For instance, one possibility is the proportional-plus-derivative (PD) tracking control.

$$v = \ddot{y}_d + K_d\dot{e} + K_p e$$

- Where the tracking error is defined as

$$e(t) \equiv y_d(t) - y(t)$$

- Closed loop system:

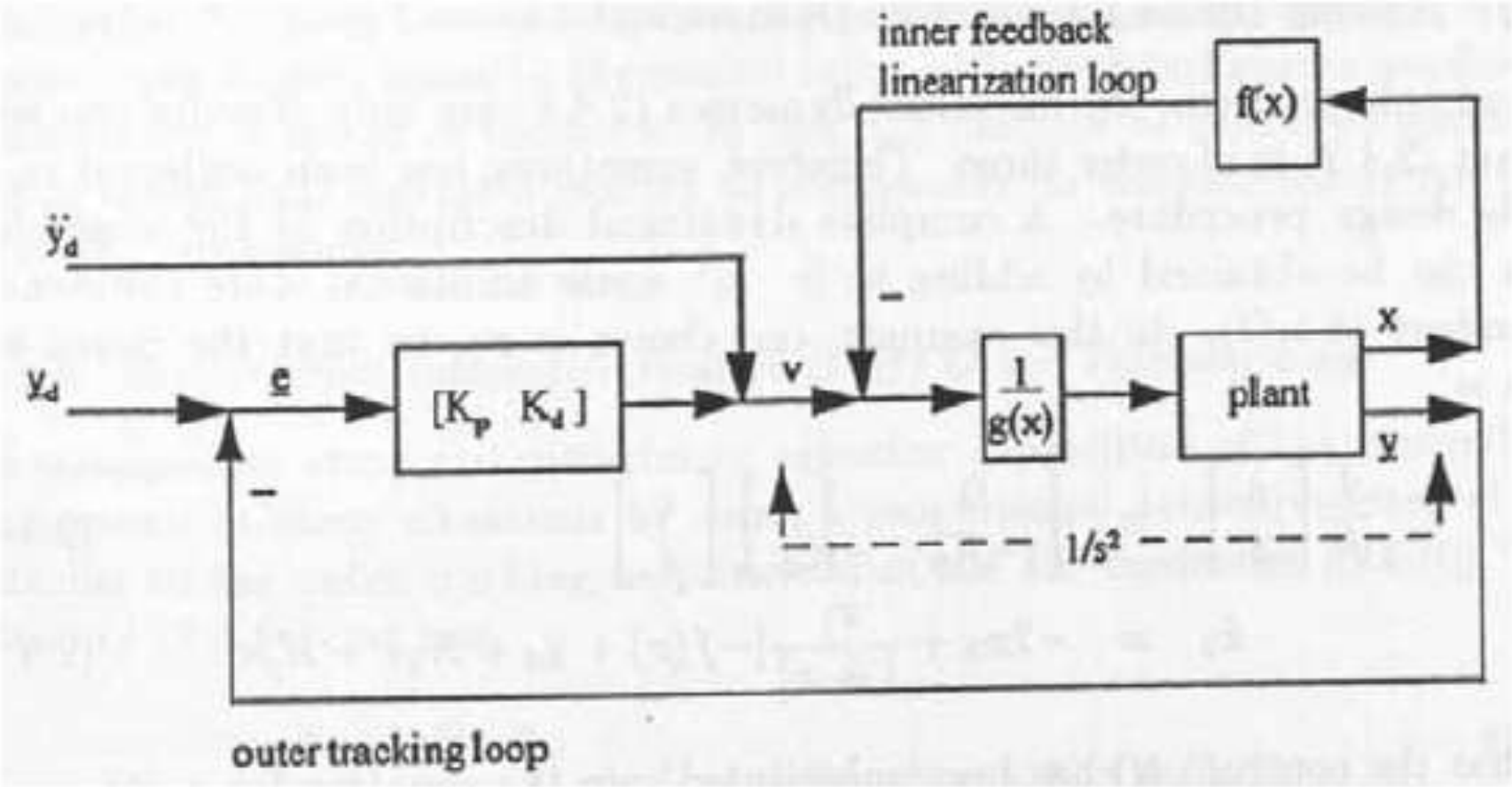
$$\ddot{e} + K_d\dot{e} + K_p e = 0$$

- State-space form:

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -K_p & -K_d \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix}$$

Feedback linearization controller

$$u(t) = \frac{1}{g(x)}[-f(x) + \ddot{y}_d + K_d\dot{e} + K_p e]$$



Proportional-Integral-Derivative (PID) Outer Tracking Loop

$$\dot{\varepsilon} = e$$

$$u(t) = \frac{1}{g(x)}[-f(x) + \ddot{y}_d + K_d\dot{e} + K_p e + K_i\varepsilon$$

The controller now has a state $\varepsilon(t)$ associated with it, so that it has some internal memory.

Simulation

Nonlinear stability analysis and controls design

Lyapunov analysis for autonomous systems:

- Consider the autonomous dynamical system

$$\dot{x} = f(x)$$

representing a closed-loop system after the controller has been designed.

- Assume without loss of generality that the origin is an equilibrium point.
- Let L(x) : n → be a scalar function such that L(0) = 0, and S be a compact subset of n.

Then, L(x) is said to be:

- Locally positive definite if L(x) > 0 when x = 0, for all x ∈ S. (Denoted L(x) > 0.)
- Locally positive semidefinite if L(x) ≥ 0 for all x ∈ S. (Denoted L(x) ≥ 0.)
- Locally negative definite if L(x) < 0 when x = 0, for all x ∈ S. (Denoted L(x) < 0.)
- Locally negative semidefinite if L(x) ≤ 0 for all x ∈ S. (Denoted L(x) ≤ 0.)

- A definite function is allowed to be zero only when x = 0, a semidefinite function may vanish at points where x = 0.
- All these definitions are said to hold globally if S = n.
- A function L(x) : n → with continuous partial derivatives is said to be a Lyapunov function for the system if, for some compact set S ⊂ n, one has locally:
- L(x) is positive definite, L(x) > 0
- L(x) is negative semidefinite, ˙L(x) ≤ 0

where L(x) is evaluated along the trajectories.

That is,

$$\dot{L}(x) = \frac{\partial L}{\partial x}\dot{x} = \frac{\partial L}{\partial x} f(x)$$

- Lyapunov Stability:

If there exists a Lyapunov function for system then the equilibrium point is stable in the sense of Lyapunov (SISL).

- Asymptotic Stability:

If there exists a Lyapunov function L(x) for system with the strengthened condition on its derivative $\dot{L}(x)$ is negative definite, $\dot{L}(x) < 0$ then the equilibrium point is asymptotically stable (AS).

Global Stability:

- Globally SISL: If there exists a Lyapunov function L(x) such that

$$L(x) > 0 \text{ and } \dot{L}(x) \leq 0$$

hold globally and

$$L(x) \to \infty \; as \; \|x\| \to \infty$$

then the equilibrium point is globally SISL.

- Globally AS: If there exists a Lyapunov function L(x) for system (2.5.1) such that

$$L(x) > 0 \text{ and } \dot{L}(x) < 0$$

hold globally and also the unboundedness condition holds, then the equilibrium point is globally AS (GAS).

Robot Dynamics and Control

- The dynamics of robot manipulators with rigid links can be written as

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = \tau$$

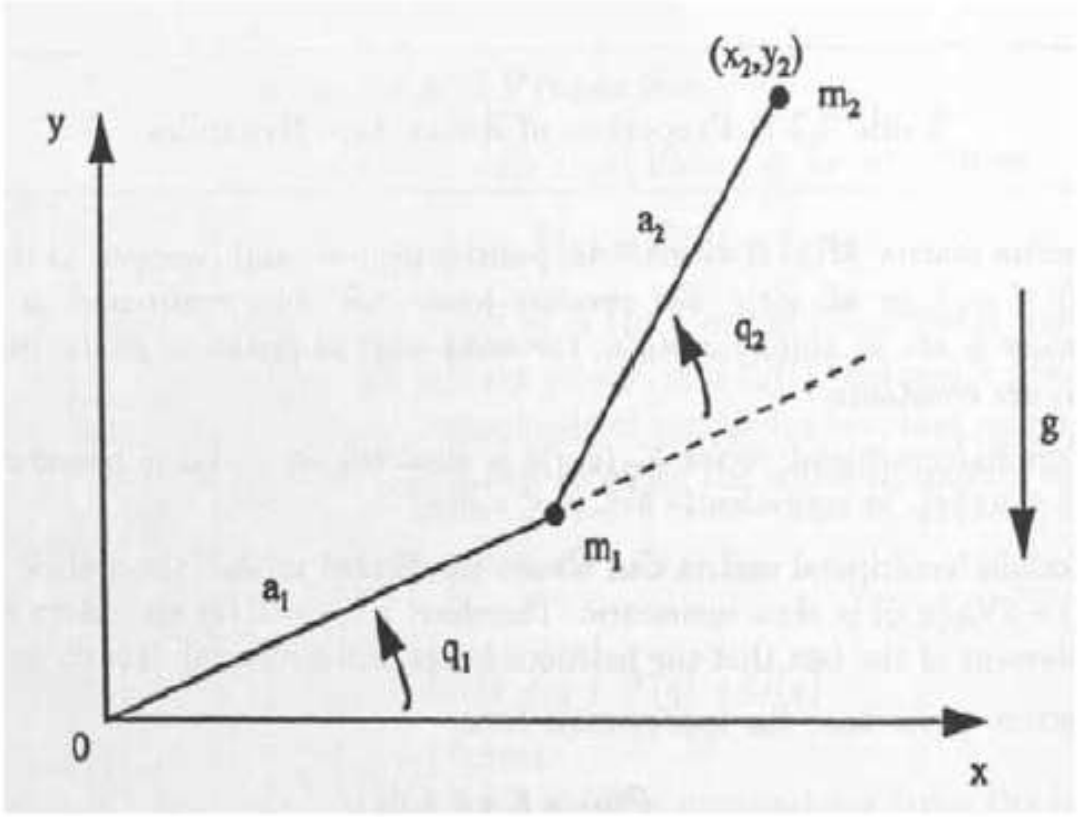
where $M(q)$ is the inertia matrix,

$V_m(q,\dot{q})$ is the Coriolis/centripetal matrix,

$F(\dot{q})$ are the friction terms,

$G(q)$ is the gravity vector, and

$\tau_d(t)$ represents disturbances.



$$M(q)\ddot{q} + N(q,\dot{q}) + \tau_d = \tau$$

where

$$N(q,\dot{q}) = V_m(q,\dot{q})\dot{q} + F(\dot{q}) + G(q)$$

represents a vector of the nonlinear terms.

Properties of Robot Arm Dynamics

P1 The inertia matrix $M(q)$ is symmetric, positive definite, and bounded so that $\mu_1 I \leq M(q) \leq \mu_2 I$ for all $q(t)$. For revolute joints, the only occurrences of the joint variables q_i are as $sin(q_i), cos(q_i)$. For arms with no prismatic joints, the bounds μ_1, μ_2 are constants.

P2 The Coriolis/centripetal vector $V_m(q, \dot{q})\dot{q}$ is quadratic in $\dot{q}$. V_m is bounded so that $\|V_m\| \leq v_B\|\dot{q}\|$, or equivalently $\|V_m\dot{q}\| \leq v_B\|\dot{q}\|^2$.

P3 The Coriolis/centripetal matrix can always be selected so that the matrix $S(q, \dot{q}) \equiv \dot{M}(q) - 2V_m(q, \dot{q})$ is *skew symmetric*. Therefore, $x^T Sx = 0$ for all vectors x. This is a statement of the fact that the fictitious forces in the robot system do no work.

P4 The friction terms have the approximate form

$$F(\dot{q}) = F_v\dot{q} + F_d(\dot{q}),$$

with F_v a diagonal matrix of constant coefficients representing the viscous friction, and $F_d(\cdot)$ a vector with entries like $K_{d_i} sgn(\dot{q}_i)$, with $sgn(\cdot)$ the signum function and K_{d_i} the coefficients of dynamic friction. These friction terms are bounded so that $\|F(\dot{q})\| \leq f_B\|\dot{q}\| + k_B$ for constants f_B, k_B.

P5 The gravity vector is bounded so that $\|G(q)\| \leq g_B$. For revolute joints, the only occurrences of the joint variables q_i are as $sin(q_i), cos(q_i)$. For revolute joint arms the bound g_B is a constant.

P6 The disturbances are bounded so that $\|\tau_d(t)\| \leq d_B$.

Manipulator Dynamics

$$\begin{bmatrix} (m_1+m_2)a_1^2 + m_2a_2^2 + 2m_2a_1a_2cosq_2 & m_2a_2^2 + m_2a_1a_2cosq_2 \\ m_2a_2^2 + m_2a_1a_2cosq_2 & m_2a_2^2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}$$
$$+ \begin{bmatrix} -m_2a_1a_2(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)sinq_2 \\ m_2a_1a_2\dot{q}_1^2 sinq_2 \end{bmatrix}$$
$$+ \begin{bmatrix} (m_1+m_2)ga_1cosq_1 + m_2ga_2\cos(q_1+q_2) \\ m_2ga_2\cos(q_1+q_2) \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

$$\dot{x} = \begin{bmatrix} \dot{q} \\ -M^{-1}(q)N(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}(q) \end{bmatrix} \tau$$

Controller algorithms

- PD Computed Torque(CT) Control:

$$\tau = M(q)\left(\ddot{q}_d + K_v\dot{e} + K_p e\right) + V_m(q,\dot{q})\dot{q} + F(\dot{q}) + G(q)$$

- PID Computed Torque(CT) Control:

$$\dot{\varepsilon} = e$$

$$\tau = M(q)\left(\ddot{q}_d + K_v\dot{e} + K_p e + K_i\varepsilon\right) + V_m(q,\dot{q})\dot{q} + F(\dot{q}) + G(q)$$

- PD Gravity Controller:

$$\tau = K_v\dot{e} + K_p e + G(q)$$

- Classical joint Controller:

$$\dot{\varepsilon} = e$$

$$\tau = K_v\dot{e} + K_p e + K_i\varepsilon$$

Filtered-error approximation-based control

$$e = q_d - q$$
$$r = \dot{e} + \Lambda e$$

where Λ is a positive definite design parameter matrix.

- Computed Torque control variant:

$$\tau = K_v r + M(q)(\ddot{q}_d + \Lambda \dot{e}) + V_m(q, \dot{q})(\dot{q}_d + \Lambda e) + F(\dot{q}) + G(q)$$

- Adaptive Controller:

$$\tau = W(x)\hat{\phi} + K_v r$$
$$\dot{\hat{\phi}} = \Gamma W^T(x) r$$

One-layer FLNN Controller

- Control Input:

$$\tau = \widehat{W}^T \varphi(x) + K_v r$$

- NN weight tuning algorithm:

$$\dot{\widehat{W}} = F\varphi(x) r^T$$