Widrow, B. (1964). A trainable control system: 'The broom balancer.' Standard Electronics Lab Report S4-SEL-64-042, Technical Report 6762-1, App. C, 84, and 85–87.

Williams, R. J.(1988). Toward a theory of reinforcement learning connectionist systems. Technical Report NU-CCS-88-3, Northeastern University, Boston, MA.

17 A Neural Network Baseline Problem for Control of Aircraft Flare and Touchdown

Charles C. Jorgensen and C. Schley

17.1 Introduction

Most commercial aircraft currently have available an optional automatic landing system or "autolander." The first automatic landing was made in England on June 10, 1965, using a Trident aircraft operated by the British airline BEA. Since then, these systems have been significantly enhanced and are now used routinely on a variety of planes including the Lockheed L1011s and Boeing 747s. They are most often activated in clear, calm weather but can also be used in fog or rain if winds are calm. When they are used in place of a pilot it is usually for two reasons. First, automatic systems give a reliably smoother landing, leading to increased passenger comfort and to a reduction of wear on tires and landing gear. Second, their use in calm weather provides an important training function, accustoming pilots to a system that they must understand and trust if it is to be used in adverse conditions in the future.

Automatic landing systems rely on an airport based system, called the Instrument Landing System (ILS), to position an aircraft for the final phases of landing. The ILS has two radio beacons, called the glide-slope and localizer beams, which guide the aircraft into the proper height (elevation) and approach angle (azimuth) relative to the runway threshold. The actual distance to the runway threshold may also be measured. At approximately 50 feet above the runway surface, the flare is initiated to elevate the nose of the aircraft, bleed off airspeed, and cause a soft touchdown on the runway surface. ILS signals are dispensed with before the actual flare because ILS signals are too noisy to provide reliable guidance below about 200 feet altitude. Below that altitude, a radio altimeter or visual data is used as the primary reference. From the flare-initiation point until touchdown, the aircraft follows a control program which decreases both vertical velocity and air speed. Because the full three dimensional dynamics of an aircraft are quite complex, the simplified baseline model developed in this chapter considers movements only in the longitudinal and vertical axes (i.e., horizontal displacements are not considered). Figure 17.1 illustrates the geometry of the resulting two dimensional flare and landing. This simplification permits the display of aircraft movement and controller errors on a standard personal computer screen without elaborate graphics.

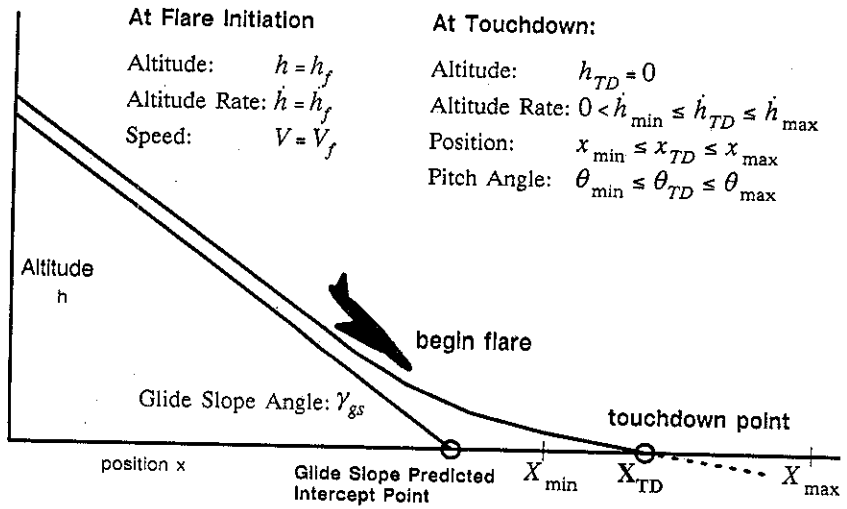


Figure 17.1
Nominal flare geometry.

A typical strategy used by a conventional controller is to decrease the vertical velocity in proportion to the altitude above the runway (i.e., altitude decreases exponentially with time or in space). For greater technical detail, the complete phases of landing and flare are presented in *Automatic Flight Control* (Pallett 1983) and in the *Systems and Control Encyclopedia* (Pelegrin 1987).

It should be noted that existing autolanded systems work reliably only within a carefully specified operational safety envelope. Today, as specified in Federal Aviation Administration regulations, these limitations are a head wind of less than 25 knots (28.75 mph), a crosswind of less than 15 knots (17.25), and a tail wind of less than 10 knots (11.5). Also, the system cannot be operated if strong downdrafts called microbursts are present. Usually, turbulence is not a significant problem for large commercial aircraft, but there are limits. A conventional autoland system may have difficulties in severe turbulence. thus, to increase the safety and smoothness of landing it would be desirable if new technology could expand the operational envelope to include safe responses under a wider range of conditions. Neural networks may provide one method of

exploring alternative control systems capable of dealing with turbulent, possibly nonlinear control conditions. Consequently, it is very useful from a research standpoint to have available a baseline problem within which alternative network configurations can be tried and compared. In the following sections we develop one such system defined in terms of linearized component equations. First, however, it is useful to elaborate on why a neural controller might be useful and how one might go about exploring the autoland problem.

17.2 Neural Network Controller

A trained pilot develops skilled intuitions called *flight sense* which help him to make appropriate responses in new and unforeseen situations. It is unclear exactly what flight sense is operationally except that it appears experienced pilots are able to handle aircraft better than automated controllers in situations outside normal conditions. An example would involve control behaviors during weather related events such as wind shear (varying head or tail winds at different altitudes), unexpected wake vortices, or microbursts (abrupt violent changes in vertical wind speed). One possible use of neural net technology would be to have a network acquire some of these subtle temporal and spatial skills from a human pilot through observation. Current linearized controllers do not do a good job of emulating pilot responses to emergencies or in generating creative solutions to rare phenomena. This is perhaps due to nonlinear or unrecognized linear relationships that pilots may use which are not captured in a conventional controller. Because a neural net is capable in principle of generating a mapping from one large set of variables (e.g., sensor streams) to another (e.g., operational modes or control actions), it can potentially capture critical behaviors implemented in very complex functions and hence may, if provided with the correct training set, be useful for the construction of such functions.

Another potential application is to let a network adaptively capture variable interrelationships not specified by a design engineer through the use of on-line training. Interactive trainability may be possible because a network can operate in a variety of learning modes, one of which is to learn by example. Thus, if presented with example pilot behaviors during selected scenarios, a net may be able to extract critical initiating variables as well as the essence of a correct response without being explicitly told what to do at each step. A word of caution should be voiced for such an approach, however, because human factors studies

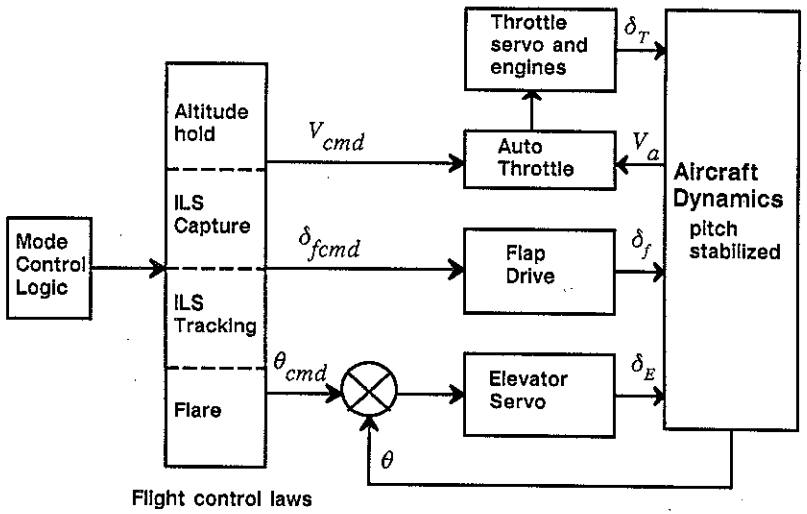


Figure 17.2
Aircraft landing system components.

have tended to indicate that human pilots may use different control strategies depending upon characteristics of the display and local environment. Thus, they may not provide a network with a stable control behavior from which consistent, more complex control principles can be extracted.

Some recent research on neural networks emulating simple classical control problems has shown encouraging results that indicate an extension of such technology to more complicated problems may well be feasible (Ba-83, Gu-87, Ye-88). However, much research remains to be done before valid comparisons between real world control applications and neural network methods can be made. In figure 17.2 we give a block diagram of some of the most important components of a generic aircraft landing control system. Notice that there are a number of operational modes which in turn depend upon the stage of the landing. After several minutes of altitude hold, ILS capture occurs, followed by ILS tracking, and finally flare. Output from these modes are a series of control commands, the most important of which for us is θ_{cmd} which controls the aircraft elevator servomechanism and consequently the pitch up during

landing. Because the aircraft is often flying under reduced power at landing, the throttle and autothrottle have minimum effect. These control variables are then fed into the aircraft dynamic model which in turn is influenced by the environment. Thus, to train a neural net we must first create the mathematical models of a standard controller (simplified and adjusted for our problem), a reduced complexity model of the airframe upon which the controller will act, and some type of environmental disturbance (in our case a wind model which exhibits logarithmic shear or different strengths at different altitudes following a logarithmic variation with altitudes). After these models are generated, a series of experiments can be conducted using variations on neural network architectures, training rules, and performance criteria. Later, we will present the above models in detail. Next, however, we will illustrate how one might use the baseline by briefly describing our current approach to the problem.

17.3 Using the Baseline Equations

In our laboratory, we plan to study the autoland problem in three stages. The first stage is complete and investigated whether a neural network could emulate the performance of a traditional control equation through observation of input/output controller responses. To do this we had to specify reasonable aircraft control requirements and provide landing metrics the network could use to adjust performance. Next, because of the complexity of a full-scale aircraft system, we feel it is important to consider how well the network performance would scale up as the plant model was increased in complexity. The third and final stage will examine the hardware implications of control algorithm computations in an actual parallel environment. Each stage requires increasingly complex models building upon the results and feasibility tests of the previous steps. In the course of our investigations we have already uncovered a number of interesting research problems. We provide them as examples of the kinds of issues which can be explored using the above paradigm.

17.4 Teaching a Neural Network Using a Known Control Law

Our general approach was to apply process control using a *trainable* controller like that represented in figure 17.3. It consisted of a teacher (a human or a linear controller), a trainable neural network (using a

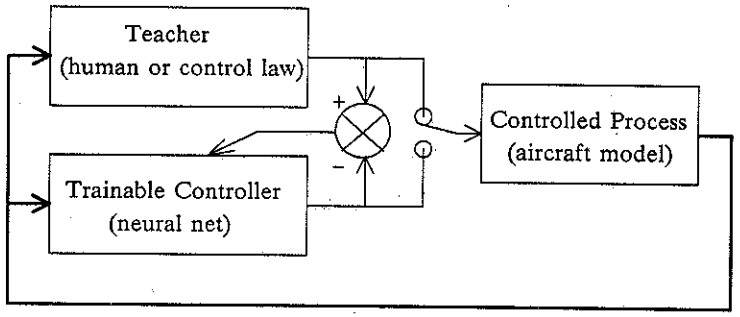


Figure 17.3
Trainable adaptive controller structure.

backpropagation learning algorithm), and the process to be controlled (e.g., keeping an aircraft icon on a predetermined flight path by issuing an attitude command θ_{cmd}). The teacher could have used a linear or nonlinear law, linear in the case of the traditional controller we used and probably nonlinear if humans were to provide landing values. The linear control law we used was derived using conventional control theory methods (Kwakernaak and Sivan 1972). For a human teacher, we would propose using actual input/output time histories obtained through direct experiments using a simple interface and a mouse or a joystick. Supervised learning in the network was used to provide us with the greatest amount of control over the mapping so the learning was not taking place in real time. The true state of the control process was provided to both the teacher and the network. The teacher defined the desired behavior of the controller by providing outputs stored in a data base sufficiently large to provide a detailed sample of the possible input state vectors.

To implement this framework we supplemented the generic aircraft model with some supporting structures including a flare controller for the vertical/longitudinal axes (Holley 1975). Once we trained the net using nominal conditions (i.e., no disturbances), we also studied its performance on off-nominal conditions such as various wind disturbances. This was because in order to make the neural net perform robust control, we had first to understand its ability to perform well understood

linear control behaviors and then determine how well the network could generalize to new environmental situations.

We also generated a simple graphic tracking program to provide human input through the interface of a workstation. The neural controller could learn through observation of a human working with elevator angle only. In our initial efforts we did not include variable throttle setting.

17.5 Applications

Regarding application to the real world, we assume that existing trends in pilot usage of autoland systems will continue. Consequently, a neural controller may best be used only as a supplementary enunciator device monitoring pilot landing behavior (i.e., an ILM enhancement). If this operational mode were selected, the pattern recognition capabilities of a neural net would have to be significantly enhanced. Particular attention should be paid to early detection of hazardous external conditions such as wind shear and the monitoring of ongoing pilot control actions. In the case of abnormal weather events, it would be necessary to consider whether a neural network could function adequately in a real-time predictive mode as well, so as to anticipate emerging problems. We have not implemented such a network but the framework would permit investigators to consider this type of problem.

17.6 Baseline Flight and Control Equations

We now present the detailed aircraft model for the glide slope and flare aspects of aircraft flight immediately prior to landing. The model is in the form of equations of motion and control loops and is oriented toward a reader with a basic engineering background. These equations are suitable for a simulation that can be used to obtain training data for neural net based controllers. Their presentation assumes some familiarity with control systems. The casual reader may find them overwhelming; but, they are necessary to specify a well-behaved aircraft plant that, although simple, responds in a realistic way to environment and control parameters. Linearized longitudinal equations of motion are used for the aircraft. We have assumed it is initially trimmed in level flight at low altitude. Data and conventional control system designs were obtained from a variety of published sources to provide a realistic representative problem including a well known commercial aircraft firm. Simplifications

retain only the overall quality of system response (i.e., high frequency hardware and actuator dynamics were neglected).

In addition to linearized equations of motion for the bare airframe, the baseline system has the following components:

1. A pitch autopilot to provide adequate damping and a speed of response to a desired pitch attitude.

2. An autothrottle to hold the aircraft at the requisite speed setting.

3. A glide slope and flare controller to produce pitch commands in response to desired altitude and altitude change rates during descent.

4. Desired altitude and altitude rate commands based on ILS tracking during glide slope and an adaptive ground speed exponential law during flare to provide a smooth descent path.

5. Altitude wind shear and turbulent gusts modeled using a Dryden distribution (a method for generating random gusts of head winds or tail winds) (Neuman and Foster 1970).

The linearized equations of motion define incremental aircraft dynamics in the longitudinal and vertical planes. They constitute the bare airframe velocity components, the pitch rate, the pitch angle, and the altitude. Figure 17.4 illustrates the aircraft coordinate system in which these variables are measured. The figure shows some of the complexities in relating aircraft variables after an aircraft changes pitch attitude. Basically, the aircraft is flying along an initial direction γ_0 at a speed U_0 with its wing chord (fixed to the aircraft body) elevated above γ_0 by an initial angle of attack a_0 in order to generate aerodynamic lift. The initial wing chord is thus at an angle θ_0 with respect to the inertial earth-fixed axes X_E, Z_E. To rotate the aircraft and change its speed, a pitch rate q is obtained through development of aerodynamic forces by elevator surfaces. This causes an incremental angle of attach a to exist along with an incremental pitch angle θ. Also, the aircraft velocity changes and has incremental components u and w directed along rotated axes X, Z. In any case, the main purpose of Figure 17.4 is to illustrate the number of variables which enter into the equations describing aircraft motion. In the equations which follow, the time behavior of these variables and their interrelationships define the effects of changes in aircraft conditions during glide slope descent and flare.

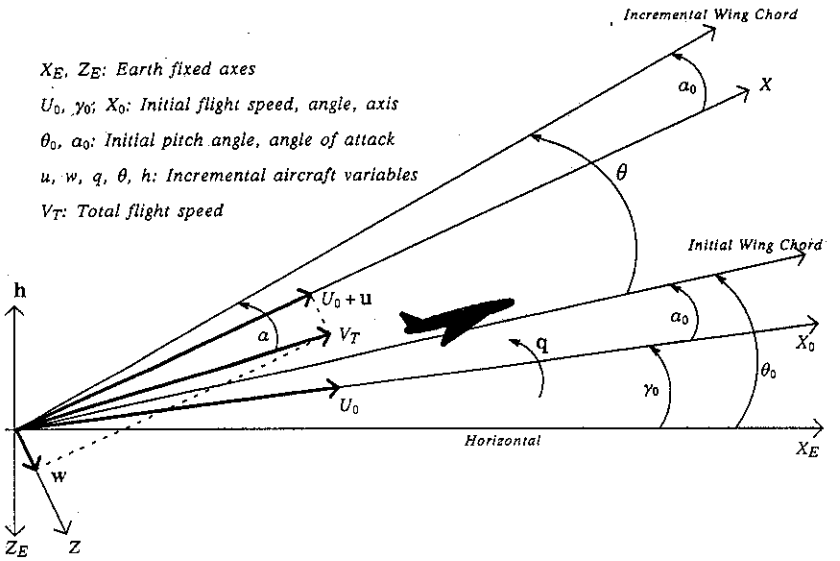


Figure 17.4
Longitudinal aircraft coordinate frame.

We begin by elaborating the incremental aircraft variables. They are developed by linearizing the force balance equations (derived from Newton's law) for aircraft motion. Equation set 17.1 shows the differential equations for incremental response of a generic bare airframe. The relations provide the time behavior of longitudinal and vertical speeds u and w, pitch rate q, pitch angle θ, and altitude h in response to elevator and thrust commands δ_E and δ_T, and horizontal and vertical wind gust speeds u_g and w_g. The values listed below equations 17.1 transform the generic airframe into a specific airframe typical of a large commercial transport.

$$
\begin{aligned}
\dot{u} &= X_u(u-u_g) + X_w(w-w_g) + X_q q - g(\pi/180)\cos\gamma_0\theta \\
&\quad + X_E\delta_E + X_T\delta_T \\
\dot{w} &= Z_u(u-u_g) + Z_w(w-w_g) + (Z_q - [\pi/180]U_0)q + g(\pi/180)\sin\gamma_0\theta \\
&\quad + Z_E\delta_E + Z_T\delta_T \\
\dot{q} &= M_u(u-u_g) + M_w(w-w_g) + M_q q + M_E\delta_E + M_T\delta_T \qquad (17.1) \\
\dot{\theta} &= q
\end{aligned}
$$

$$\dot{h} = -w + [\pi/180]U_0\theta$$

where:

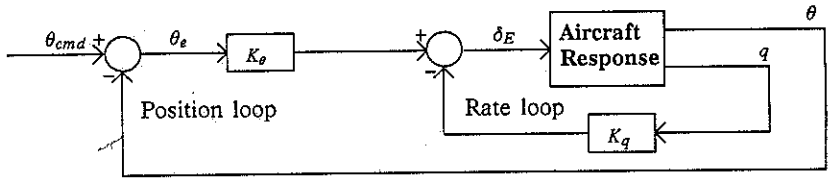
u, w = *incremental velocity components (ft/sec)*
u_g, w_g = *wind gust velocity components (ft/sec)*
q, θ, h = *incremental pitch rate (deg/sec), pitch angle (deg), altitude (ft)*
δ_E, δ_T = *incremental elevator angle (deg), throttle settling (ft/sec)*
$X_u, X_w, X_q, X_E, X_T, Z_u, Z_w, Z_q, Z_E, Z_T, M_u, M_w, M_q, M_E, M_t$ = *stability and control derivatives of aircraft at some normal condition*
U_0), γ_0, g = *nominal speed (235 ft/sec), flight path angle (−3 deg), gravity (32.2 f/s/s)*

typical values :

$X_u = -.038$	$Z_u = .313$	$M_u = -.0211$
$X_w = -.0513$	$Z_w = -.605$	$M_w = .157$
$X_q = .00152$	$Z_q = -.0410$	$M_q = -.612$
$X_E = .00005$	$Z_E = -.146$	$M_E = .459$
$X_T = .158$	$Z_T = .031$	$M_T = .0543$

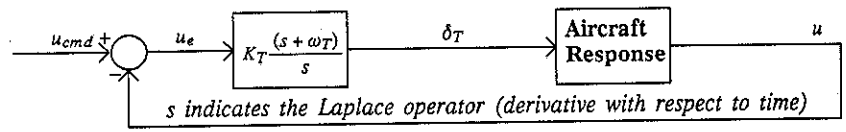
The transient response characteristics of these equations are typical of standard bare airframe behavior. There are two types of response you would notice if you tried to fly such an aircraft without further control systems operative: a low-frequency (about 0.1 radians/sec), very lightly-damped Phugoid response and a higher-frequency (about 1.0 radians/sec), well-damped short-period response. This response pattern is entirely too oscillatory to apply glide slope and flare control laws. Hence, we stabilize the aircraft model with a stability augmentation system. Such an augmented aircraft model is obtained by combining the linearized equations of motion with a pitch autopilot and autothrottle. The pitch autopilot is shown in the diagram of Figure 17.5. The function of a pitch autopilot is to significantly damp the oscillatory bare airframe behavior while providing reasonably fast pitch response to commands. The pitch autopilot consists of proportional plus rate feedback combined with a pitch command to develop the aircraft elevator angle.

The function of the autothrottle is to maintain constant airspeed. Figure 17.6 illustrates the autothrottle which consists of proportional plus integral feedback combined with a commanded incremental speed (normally zero) to develop the aircraft throttle setting.



typical values - Glide Slope: $K_\theta = 2.8$ *Flare:* $K_\theta = 11.5$
$K_q = 2.8$ $K_q = 6.0$

Figure 17.5
Pitch autopilot.



typical values : $K_T = 3.0$
$\omega_T = 0.1$

Figure 17.6
Autothrottle.

Controllers also must be specified for both the glide slope and the flare modes. Other than parameter values and an open-loop pitchup command for flare, both of these controllers were constructed similarly. Controller inputs consisted of altitude and altitude rate commands along with aircraft altitude and altitude rate estimates. Normally, these estimates are obtained by filtering of sensor data (rate gyro, accelerometer, etc.). In our work, these second order effects were neglected. Thus, figure 17.7 illustrates the selected controller architecture. The dotted lines around the figure correspond to the operations which must be performed by a neural network controller.

Glide-slope commands we developed using a constant sink rate along an assumed ILS path. Actually, ground speed is calculated by measuring the distance to the glide slope predicted intercept point (see figure 17.1). Flare commands consist of an adaptive ground speed exponential law (fixed n-space rather than in time). Equations 17.2 and 17.3 show the steps needed to compute the glide slope and flare commands.

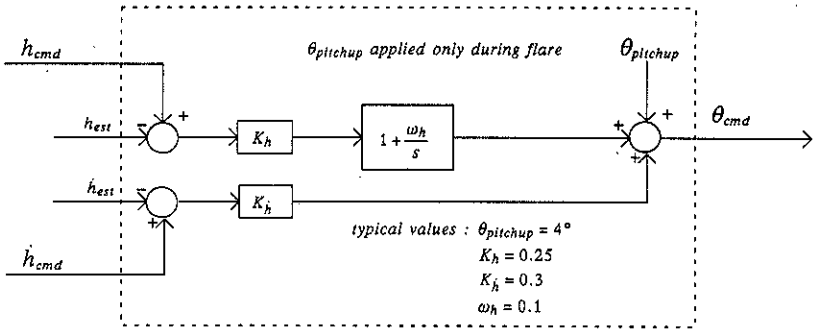


Figure 17.7
Glide-slope and flare controller architecture.

Glide-Slope Altitude and Altitude Rate Commands ($h_{cmd}, \dot{h}_{cmd}$)

$$
\begin{aligned}
V_G &= U_0 \cos(\theta - \alpha) + u_{g_c} \\
\dot{x}_{cmd} &= V_G, \quad x_{cmd}(t_{gs}) = -\frac{h_s}{\tan\gamma_0} \\
h_{cmd} &= -x_{cmd} \tan\gamma_0, \quad \dot{h}_{cmd} = -V_G \tan\gamma_0
\end{aligned} \tag{17.2}
$$

where:

Glide Slope begins at $t = t_{gs}$, $h_s = 500$ ft., $\gamma_0 = 3.0°$

V_G *is the ground speed and U_0 is the nominal aircraft speed in ft./sec.*

θ *is the aircraft pitch and α is its angle of attack in degrees*

u_{gc} *is the constant (altitude shear) component of the horizontal gust velocity in ft./sec.*

x_{cmd} *is the negative of the ground range to the GPIP in ft.*

Flare Altitude and Altitude Rate Commands ($h_{cmd}, \dot{h}_{cmd}$)

$$
\begin{aligned}
h_{cmd} &= h_0 \left(\frac{\dot{h}_0}{(\dot{h}_0 - \dot{h}_{TD})} e^{\frac{-(x_{cmd} - x_{cmd_0})}{\tau_x}} - \frac{\dot{h}_{TD}}{(\dot{h}_0 - \dot{h}_{TD})} \right) \\
& \quad h_{cmd}(t_0) = h_0
\end{aligned}
$$

$$
\begin{aligned}
\dot{h}_{cmd} &= -\frac{h_0}{\tau_x} V_G \left(\frac{\dot{h}_0}{(\dot{h}_0 - \dot{h}_{TD})} e^{\frac{-(x_{cmd} - x_{cmd_0})}{\tau_x}} \right) \\
\dot{h}_{cmd}(t_0) &= \dot{h}_0 \text{ if } \tau_x \text{ is chosen as } \tau_x = -\left(\frac{h_0 V_G}{(\dot{h}_0 - \dot{h}_{TD})} \right) \\
h(t_{TD}) &= 0 \text{ when } (x_{cmd_{TD}} - x_{cmd_0}) = -\tau_x \ln\left(\frac{\dot{h}_{TD}}{\dot{h}_0} \right), \\
& \quad \text{also } \dot{h}(t_{TD}) = \dot{h}_{TD}
\end{aligned} \tag{17.3}
$$

where:

Flare begins at $t = t_0$, $h_0 = 45$ ft.

V_G *is the ground speed, ft./sec. (see equations 17.2)*

x_{cmd} *is the negative of the ground range to the GPIP, ft. (see equations 17.2)*

x_{cmd_0}, $x_{cmd_{TD}}$ *are values of x_{cmd} at beginning of flare and at touchdown*

h_0 *is the altitude rate when flare begins, $\dot{h}_{TD} = -1.5$ ft./sec.*

Next we introduce the influences of the environment. In this baseline example they are confined to winds. Wind disturbances are assumed to have two components: constant velocity and turbulence. The magnitude of the constant velocity component is a function of altitude (wind shear). Turbulence is more complex and is a temporal and spatial function as an aircraft flies through an airspace region. The constant velocity wind component exists only in the horizontal direction (i.e., head wind or tail wind) and its value is given in equation 17.4 as a logarithmic variation with altitude.

$$
u_{g_c} = -u_{wind_{510}} \left[1 + \frac{\ln(h/510)}{\ln(51)} \right] \tag{17.4}
$$

where:

u_{g_c} *is the constant (altitude shear) component of u_g, zero at 10 ft. altitude*

$u_{wind_{510}}$ *is the wind speed at 510 ft. altitude (typical value = 20 ft/sec)*

h *is the aircraft altitude*

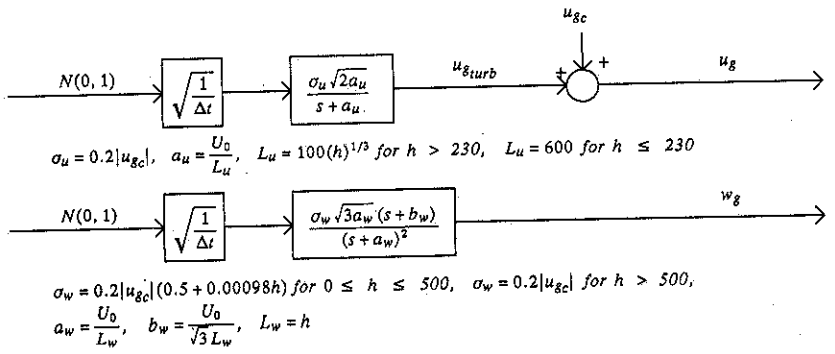


Figure 17.8
Wind disturbance calculation.

Concerning the horizontal and vertical wind turbulence velocities, the Dryden spectra (Neuman and Foster 1970) for spatial turbulence distribution were assumed. These spectra involve the specification of a wind disturbance model frozen in time. This is not seriously limiting since the aircraft moves in time through the wind field and thus experiences temporal wind variation. The Dryden spectra are also amenable to simulation and show reasonable agreement with measured data. The generation of these turbulence velocities is effected by applying Gaussian white noise to filters represented as transfer functions. This provides the proper correlation to match the desired spectra. Figure 17.8 summarizes wind shear and turbulence calculations should the user wish to develop the same wind profiles we use. Other winds of a simpler form could be introduced for faster implementation.

To simulate the flight of our aircraft, the five equations for u, w, q, θ, and h given in equations 17.1 must be solved by means of some method for solution of differential equations (e.g., Runge-Kutta). The values for elevator and throttle angle forcing functions for equations 17.1 (i.e., δ_E and δ_T) are obtained by implementing the equations implied by the autopilot and autothrottle diagrams of figure 17.5 and figure 17.6. Here,

the symbol s indicates the Laplace operator (derivative with respect to time). Input to the pitch autopilot θ_{cmd} is calculated by implementing the controller shown in figure 17.7 where flare takes place at about 50 feet of altitude. Inputs to the controller h_{cmd} and $\dot{h}_{cmd}$ are determined by means of equations 17.2 and 17.3. Autothrottle input u_{cmd} is taken to have zero value. Finally, the wind disturbance forcing functions for equations 17.1 (i.e., u_g and w_g) are generated by means of the diagrams of figure 17.8 and the average horizontal speed given by equation 17.4.

A sample scenario for the generation of a neural net training data set involves recording a number of trajectories for descent along a glide slope from an initial altitude of 500 feet followed by flare at 45 feet altitude and then touchdown. Both nominal (disturbance-free) conditions and logarithmic altitude shear turbulent conditions should be considered for completeness.

17.7 Performance Measures

Measures of performance were required to specify the desired landing conditions of the aircraft. Basically, they require that the aircraft must land within a desired envelope of dispersions. The following conditions summarize our touchdown performance measures. We consider four: sink rate (how fast the aircraft descends), longitudinal touchdown position (where the aircraft touches the runway), forward speed, and the attitude of the aircraft.

Sink Rate ($\dot{h}_{TD}$):

$$|\dot{h}_{min}| \leq |\dot{h}_{TD} \leq |\dot{h}_{max}|$$

typical value : $|\dot{h}_{min}| = 1ft/s$, $|\dot{h}_{max}| = 3ft/s$

Longitudinal Touchdown Position (x_{TD} where x_{nom} is nominal touch down point)

$$x_{nom} - x_{min} \leq x_{TD} \leq x_{nom} + x_{max}$$

typical values : $x_{min} = 300ft$, $x_{max} = 1000ft$

Forward Speed (V_{TD}):

$$V_{min} \leq V_{TD} \leq V_{max}$$

Pitch Attitude (θ):

$$\theta_{min} \leq \theta \leq \theta_{max}$$

17.8 Implementing the Baseline Aircraft Model

To facilitate the use of the baseline model we found it useful to construct a simulation environment on a Symbolics computer which provided us with three types of experimental environments.

1. A Process Control environment to simulate the plant to be controlled and to generate trajectories used for display and neural net learning.
2. A Neural Net Fabrication environment to easily construct neural net architectures and train them.
3. A Statistical Analysis environment to study statistical properties of data to be used for neural net training.

Our baseline aircraft simulation was then coded into this environment. Provisions have been made for training data storage and graphical display interfaces to observe performance. Figure 17.9 is a modified screen image of the nominal (no-wind) aircraft response produced by the classical controller during the execution of a simulated landing. The lower graph shows a landing descending from 500 feet after glide-slope capture while the upper graph shows the aircraft vertical velocity ranging between 5 and -15 ft/sec.

At present, alternative neural net input architectures have been explored to address the dynamics inherent in the control problem. Experiments centered around the specification of various input combinations and the issue of including system dynamics. Typically, a control problem requires three elements: past history by means of integration, current values by means of proportional combination, and future predictions by means of differentiation. Consequently, some degree of information recurrence must be used to mimic a conventional control system.

Recurrence within a neural net complicates learning immensely. One approach for recurrent loops is to break them apart and handle them separately. Learning takes place very slowly under these circumstances. Therefore, a scheme was devised to provide the effects of recurrence by defining variables external to the neural net. This stratagem provides as inputs to the neural net not only current values for aircraft variables (i.e., altitude, altitude commands, etc.), but also values at some point in the past. The neural net can thus be expected to have some basis to formulate a model of proportional combination and differentiation of system variables. Additionally, the past value of the neural net output (i.e., itch attitude command) was also back as an input to allow the

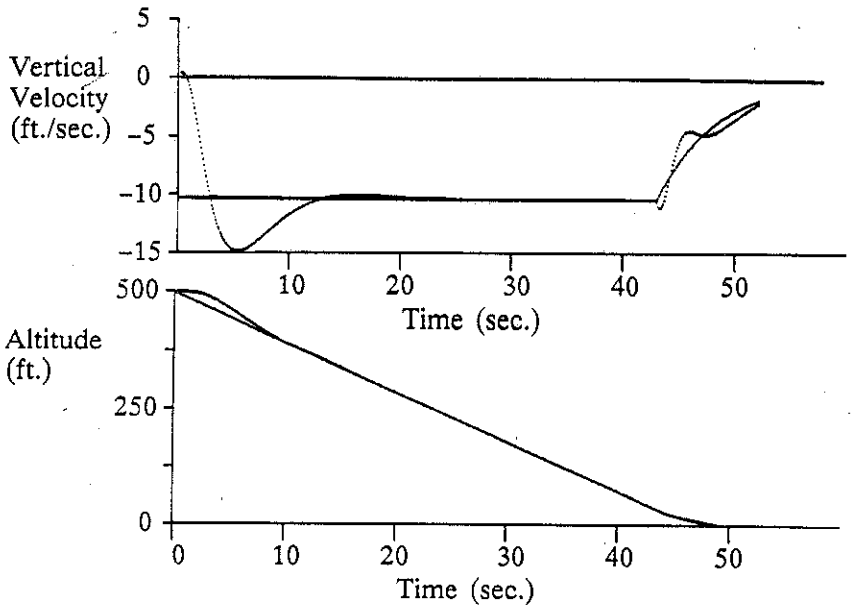


Figure 17.9
Aircraft simulation nominal trajectory in Process Control environment.

neural net to represent internally the integration of system variables. This scheme has a beneficial effect of providing recurrence outside of the neural net. Standard backpropagation can then be used for training. Clearly, there are other approaches that could be explored using the current baseline equations.

Figure 17.10 illustrates the construction of one of our neural net structures. As mentioned, inputs consisted of current altitude and altitude rate error values along with values at the previous simulation time step (taken as 0.025 for our purposes). Also provided was the value of the pitch attitude command at the previous time step, providing exterior (or broken) recurrence.

The architecture of the neural net shown in figure 17.10 consists of an input layer with nine inputs, a hidden layer with four neurons and an output layer with one neuron. Operation of the neural net was as follows. The input values were scaled so that the input neurons have activity values from -1 to $+1$. Then, each input neuron was connected

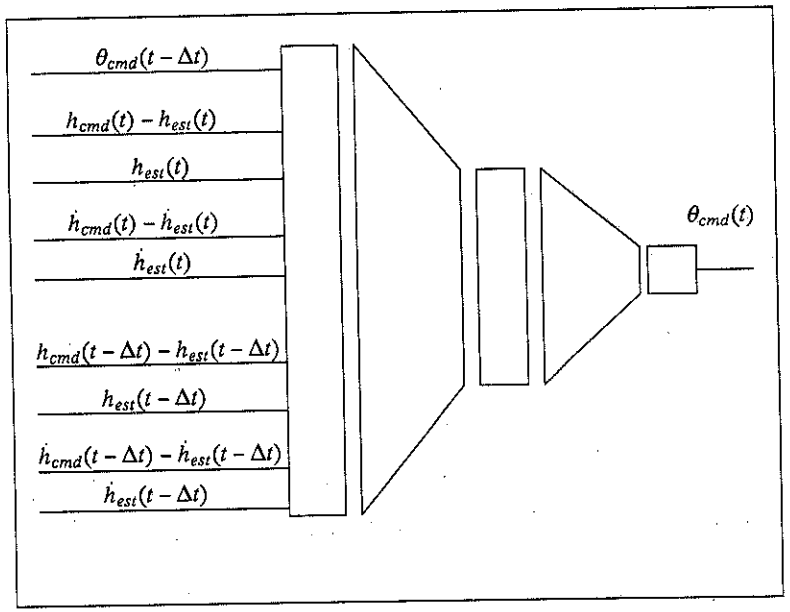


Figure 17.10
Neural net used for glide slope and flare control.

to each hidden layer neuron by a weight value adjusted during learning. Hidden layer neuron activity values are computed by summing the input neuron activity values multiplied by the weight value. This was followed by application of a standard nonlinear (sigmoidal) squashing function which provided hidden layer neuron activity values from -1 to $+1$. Similar weight connections, summation of products and squashing was used to determine the activity value of the output neuron. Finally, the output neuron activity was scaled to provide the requisite range for pitch attitude commands.

17.9 Training the Neural Network

Our net has been trained using a version of the backpropagation algorithm under a variety of wind conditions. Trajectories involving no wind,

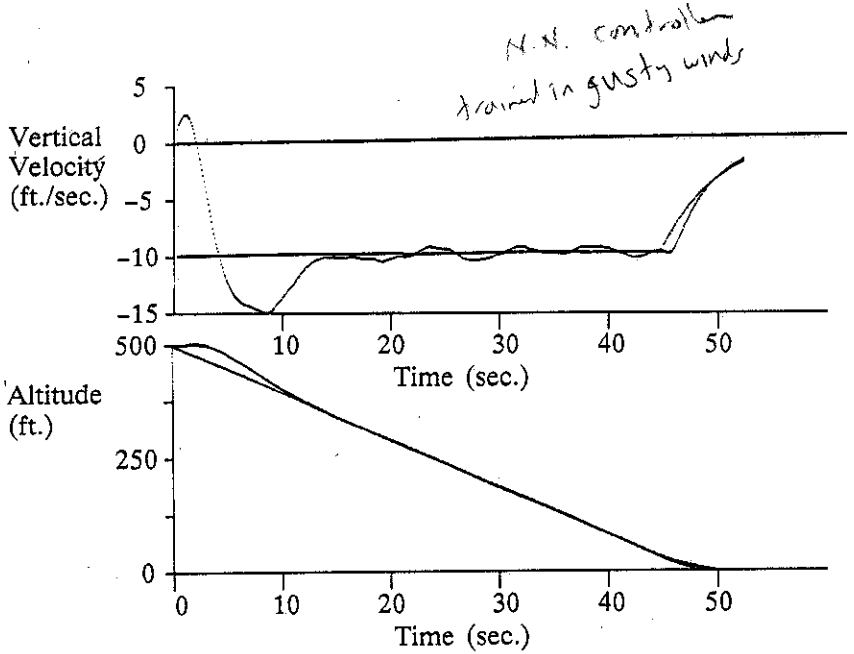


Figure 17.11
Gusty wind trajectory of neural net controlled aircraft.

constant wind, and logarithmic wind shear have been generated. Also, turbulent gusts have been added. Figure 17.11 illustrates the result of controlling the aircraft using a neural net trained in gusty wind conditions. The upper graph of figure 17.11 shows the altitude rate of change while the lower graph shows the altitude. These initial indications imply that the neural net results provide good comparisons with conventional controllers when exposed to conditions similar to the training set. Note that the conventional controller response for these wind conditions is very similar to the neural controller response shown in figure 17.11.

Further experimentation with learning is needed to show that this or alternative neural nets can indeed fuse the various modes of flight near landing (e.g., glide-slope, flare). To accomplish this fusion, networks need to be trained with a wide variation of environmental conditions. Also, statistical comparisons among trained controllers must be made with respect to performance measures such as those listed earlier.

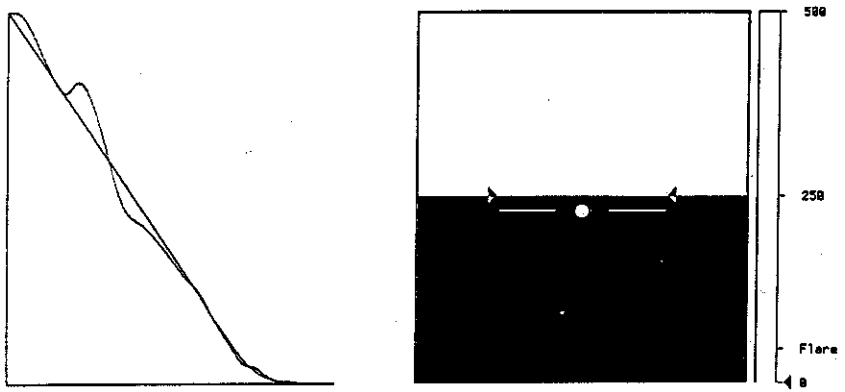


Figure 17.12
Flight manager interface.

17.10 Training the Neural Network via a Human Interface

Learning experiments could also be conducted using human-derived training data. To this end, we have constructed a simple aircraft navigation interface. Currently, the system uses a mouse and does not run in real time. This system is not intended to be realistic autolander simulator. Rather, it is intended to experiment with training selected neural nets with other than conventional linear controllers.

The human interface includes a "Flight Manager," similar to that used in real aircraft, which displays the angle of inclination and height of the aircraft along with additional information. A "Preview" indicator also displays a desired flight-path profile. Figure 17.12 illustrates the interface. The Flight Manager has the following components:

1. A plane symbol, whose height above/below the horizon indicates the aircraft's degree of inclination from horizontal. Its position is determined by θ in the plane state vector.

2. A pair of triangular guides, located just outside of the aircraft symbol's wingtips, which approximates the correct angle of inclination at the current moment for proper flight-path navigation. This angle is computed as a "theoretical θ_{cmd}," as computed from the controller equations.

3. A circular guide that allows the pilot to direct the aircraft's angle of inclination with the use of a mouse input device. The pilot's positioning of the guide above/below the horizon determines the θ_{cmd} put to the aircraft.

4. An altitude bar, which monitors the current height of the aircraft as a rising/falling horizontal line in the bar. The height is taken from h in the plane state vector.

5. A triangular guide adjacent to the altitude bar indicates the correct altitude at the current time for proper flight-path navigation. This value is computed as a "theoretical h_{cmd}," as computed from the controller equations.

6. A constant, computed at initiation of the flight manager, relates overall window size to the number of pixels per degree of inclination. This constant is used to draw the aircraft and angle of inclination guides at consistent distances along the vertical axis.

7. The desired flight path is displayed on an x-y cartesian graph. Height is represented along the vertical axis and distance on the horizontal. The plane must then "fly" the indicated line, with the current position of the plane represented as the leading dot of a series of dots, which mark the plane's actual path.

8. Upon initiation, the entire flight path is shown. At a predetermined altitude, the flight path is redrawn at a larger scale, to assist the pilot during crucial flare and touchdown phases of the flight.

17.11 Summary and Conclusions

In this chapter we have presented a simplified model of a realistic aircraft which potentially provides a baseline model and control relationships against which alternative neural network controllers can be evaluated. We have mentioned several ways in which the problem could be approached including some methods currently being explored in our facility. These efforts are only in their early stages and the general area of comparison with classical control approaches for aviation problem areas

should provide a rich topical source for further work. Of particular interest are the relationships between human and machine controllers, how displays generate conditions that influence human control strategies, and the generation of mappings between classical control laws and the outputs that may be generated by a neural controller as it extracts control regularities from data. As a result of our experiments, we have observed the necessity for refinements in a number of neural network concepts when faced with a complex application. Among them are the ability of a neural net to learn discontinuities, the importance of and potential risks associated with grouping training data around discontinuities, the vital need for faster convergence methods, and the advantages of incorporating a priori knowledge into a network to facilitate convergence. Although we have successfully generated networks that land the aircraft in headwinds and tailwinds, we have not found a single network which has been able to capture the performance capabilities of a linear controller throughout the entire domain of its input space. Further research is needed, and we hope the present work may interest other researchers in using this problem for evaluation of their own network designs.

References

Barto, A., Sutton, R., and Anderson, C. Neuron-like adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics* 13(5): 834–836.

Guez, A., and Selinsky, J. (1988). A trainable neuromorphic controller, *Journal of Robotic Systems*, 5(4): 363–388.

Holley, W. E. (1975). *Wind modeling and lateral control for automatic landing.* Stanford University, Ph.D. Dissertation, Stanford, CA.

Kwakernaak, H., and Sivan, R. (1972) *Linear optimal control systems.* New York: Wiley.

Neuman, F., and Foster, J. D. (1970). Investigation of a digital automatic aircraft landing system in turbulence, *NASA Technical Note TN D-6066*, Ames Research Center, Moffet Field, CA, October.

Pallett, E. H. J. (1983). Autopilot logic for flare maneuver of STOL aircraft. In *Automatic flight control*, 2nd ed. London: Grenada.

Pelegrin, M. (1987). *Encyclopedia of systems and control.* New York: Pergamon Press.

Yeaung, D. (1988). Supervised learning of action probabilities in associative reinforcement learning. In *Proceedings of IEEE International Conference on Neural Networks*, (July): 162–171, San Diego, CA.