Using Genetic and Evolutionary Algorithms to Solve Boundary Control Problems in Soil-Water-Plant Interaction

by

David K. Sturgess

A thesis submitted for the degree of Master of Science

Central Queensland University

School of Mathematical & Decision Sciences

March 2002

Abstract

In this thesis we investigate how modern artificial intelligent techniques, namely the genetic algorithm/evolutionary algorithm can be applied to find irrigation strategies for a cropped soil.

We begin by providing an introductory chapter detailing the work that is to be carried out and the results obtained from research.

In Chapter 2 we introduce some basic concepts of soil physics in order to give an understanding of the nature of soil composition and the movement of water within a cropped soil. We then summarise background research undertaken by Terry Janz in his Masters Thesis which shows how an irrigation schedule can be obtained using classical methods to solve the Richards' flow equation with realistic parameters and field data.

Genetic and evolutionary algorithms are introduced in Chapter 3; their algorithmic structure is defined and contrasted with classic search techniques.

In Chapter 4 we apply a genetic algorithm to the problem posed in Chapter 2 to obtain a schedule of irrigation defined as a sequence of irrigation on and irrigation off switches, to control moisture content at specific levels at certain depths within the soil, so that "nutrient uptake" by the root can be maximised. The problem posed is the classical optimal control problem in which the tracking of a desired set of final states is to be achieved.

Finally in Chapter 5, we undertake an initial research study into how an evolutionary algorithm can be applied to solve the tracking problem associated with

boundary control of a parabolic distributed process. The problem is first transformed into a classical optimal control problem with ordinary differential equations as differential constraints, by using the method of semi-discretisation, or method of lines. Our results are compared with classical techniques commonly used to solve this type of problem, including the finite element method which uses full discretisation of both the state and time variables. It is shown that it is feasible to apply evolutionary learning to problems of boundary control which arise in determining realistic irrigation strategies.

List of Publications

1. Stonier, R.J. and Sturgess, D.K., Genetic learning of the irrigation cycle for water flow in cropped soils, *Proceedings of the First Asian-Pacific Conference on Evolutionary Algorithms and Learning (SEAL'96)*, KAIST, Taejon, Korea, 201-208, November 9-12th, 1996.

2. Stonier, R.J. and Sturgess, D.K., Genetic learning of the irrigation cycle for water flow in cropped soils, *Lecture Notes in Artificial Intelligence*, Xin Yao, Jong-Hwan Kim & Takeshi Furuhashi (Eds), Springer Verlag, Vol. 1285, 89-96, 1997.

3. Stonier, R.J., Sturgess, D.K., Smith, S.F. and Drumm, M.J., Optimal boundary control of a tracking problem using evolutionary algorithms, International Conference on Computational Intelligence for Modelling, Control and Automation, ISBN 0858898470, M. Mohammadian (Ed), Las Vegas, 204-214, 2001.

Using Genetic and Evolutionary Algorithms to Solve Boundary Control Problems in Soil-Water-Plant Interaction

by

David K. Sturgess

A thesis submitted for the degree of Master of Science

Central Queensland University

School of Mathematical & Decision Sciences

March 2002

CENTRAL QUEENSLAND
UNIVERSITY LIBRARY

Contents

List of Tables

List of Figures

ACKNOWLEDGMENT

I would like to acknowledge the support of my supervisor: Associate Professor Russel J. Stonier, of the School of Mathematical & Decision Sciences, Central Queensland University, to whom I am eternally indebted, for his boundless knowledge, unwavering guidance, tireless energy and above all his inspiration.

To the staff of Central Queensland University, my colleagues, my friends and my family, especially my parents, thank you for your patience and support.

DECLARATION

This thesis contains no material which has been accepted for the award of another degree. Furthermore, to the best of my knowledge, this thesis does not contain any material previously published or written by another person, except where due reference and acknowledgment is made in the text.

Signature Redacted

Chapter 1

Introduction

This thesis deals with how modern artificial intelligent techniques such as genetic/evolutionary algorithms can be put to use to find irrigation strategies for a cropped soil by finding strategies that will keep moisture content within the soil layers at sufficient quantities to enable nutrient uptake by plants.

Chapter two, based on previously published material from other authors, introduces the theory of water flow in soil. Of particular importance is Equation (2.7). This equation, modified with a sink term and discretised using the Crank-Nicolson method, and given appropriate boundary conditions is employed as the basis of computer simulation of an irrigation period. The information presented in this chapter serves as a necessary foundation to understanding the use of the simulation in conjunction with optimal control techniques.

Chapter 3 introduces the reader to genetic and evolutionary algorithms, their basic

terminology, concepts and mechanics, and how they may be utilised in optimal control problems. This chapter concludes the presentation of necessary background material started in chapter 2 and leads into our first area of research.

Chapter 4 details the research carried out in the application of genetic algorithms to the learning of an irrigation schedule through optimal control. This had the aim of maintaining moisture content around desired values at given depths. We show that genetic algorithms can be used to learn a realistic irrigation schedule for a problem posed with realistic field data.

In Chapter 5 we examine how evolutionary algorithms may be applied to boundary control of a distributed process which is similar to the control problem tackled in Chapter 4 but with a simplified structure compared to the parameter complexity of flow equation used in that simulation. This work is undertaken to show that the problem of chapter 4 may be solved by another technique without having to utilise the classical solution of the flow equations which constantly error checking to maintain accuracy. Here we establish that evolutionary algorithms can be applied to optimal control problems in distributed parameter systems under semi-discretisation with a good measure of success. This particular method has not been applied to the irrigation scheduling problem in Chapter 4 as the time taken to undertake this extra study would have exceeded the maximum completion time for the masters.

Chapter 6 summarises the research carried out over the duration of this masters program and highlights the research results obtained and future directions for research.

Chapter 2

Modelling water flow in soil.

2.1 Introduction

The content of this chapter is a summary taken from previously published material found in [14] [18] [29] and [28].

We first take a look at soil physics, the theory of water flow in soil, Richards' flow equation and evapotranspiration introducing basic concepts and terminology necessary for understanding of the work that is to follow. The reader is invited to reference [14] should a deeper understanding be desired.

Next we review the work carried out in the masters thesis of Terry Janz [18] which introduces the concepts related to irrigation scheduling and the processes and terminology incumbent to the analysis of the state of water flow in soils. The model for simulating the flow of water through the soil is derived and a method for its

solution presented. A clear understanding of this is essential as it forms the basis of the optimisation problem which is presented in later chapters.

Water flow in unsaturated soils is modelled using the Richards' equation in conjunction with an empirical sink term which is used to represent water uptake by plant roots. The sink term introduced by the authors is different from those typically used. It incorporates a notion of an "evaporation front" that encapsulates the effect of drying in the upper layers of the soil. This drying in the upper layers essentially renders water uptake by plant roots useless. A finite difference approach utilising the Crank-Nicolson method is used to solve the Richards' equation numerically and the results produced via computer simulation are compared to real life data to validate the model.

In producing a computer simulation to model the flow of water through a cropped soil, via the developed finite difference equations and sink term, problems are encountered and solutions presented. Among these are the consideration given to the boundary conditions arising at the surface due to both infiltration and evaporation events. Of particular significance is the switching event from infiltration to evaporation and the difficulty this produces in respect to use of the Crank-Nicolson method. This is of enormous importance in a simulation of an irrigation scheduling cycle.

2.2 A brief introduction to soil physics

"Soil Physics" is the branch of soil science dealing with the physical properties of soil, as well as with the description, measurement, and control of the physical processes taking place in the soil such as the state and movement of matter and the fluxes and transformation of energy.

Soil itself is a heterogeneous, polyphasic, particulate, disperse and porous system with a possibly enormously large interfacial area per unit volume. Being a disperse system combined with its related interfacial activity means the several phenomena manifest, for example swelling, dispersion, aggregation, adhesion, adsorption and ion exchange to name a few. Soil actually contains all three natural phases of matter in the form of solid particles, soil water (with dissolved particles) hereto referred to as the soil solution, and soil air.

The *solid matrix* of soil consists of particles differing in chemical and mineralogical composition as well as size, shape and orientation whose mutual organisation determines the characteristics of pore spaces in which water and air are transmitted and/or retained.

Figure 2.1 represents the total mass and volume of the soil in 3 sections which are generally quantitatively unequal.

The terms which are commonly used to express quantitative interrelations of the 3 primary soil constituents are defined using this diagram.

Dry bulk density, which expresses the ratio of the mass of dry particles to the total

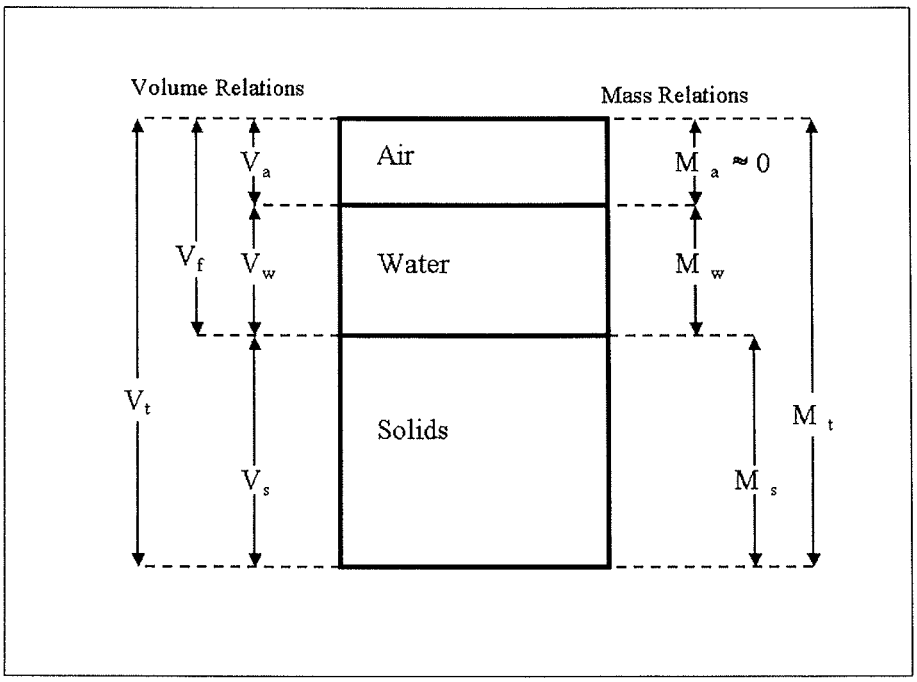


Figure 2.1: The 3 phase soil system

volume of soil (including pores) is defined as:

$$\rho_b = \frac{M_s}{V_t} = \frac{M_s}{V_s + V_a + V_w}.$$

An index of the relative pore volume in the soil, referred to as *porosity*, is expressed by the following equation:

$$f = \frac{V_f}{V_t} = \frac{V_a + V_w}{V_s + V_a + V_w},$$

and is normally in the range of 0.3 to 0.6. Course textured soils tend to be less porous than fine soils. The porosity of clay soils varies greatly as the soil swells, shrinks, aggregates, disperses, compacts and cracks.

- *Volume wetness* θ:

$$\theta = \frac{V_w}{V_t} = \frac{V_w}{V_s + V_f}.$$

Usually called the *volumetric water content*, θ is usually used more often than w since it is more directly adaptable to computation of fluxes and quantities added or subtracted from soils. The typical ranges for θ in saturated soils are, sandy 40% to 50%, medium textured soils approximately 50% and clay type soils approximately 60%. It should be noted that the water volume at saturation for clay type soils may exceed porosity of the dry soil since clay soils swell when wet.

Also

$$\theta = \frac{w\rho_b}{\rho_w},$$

where ρ_w is the density of water. Since, generally, $\rho_b > \rho_w$ it can be seen that $\theta > w$.

- *Degree of saturation* θ_s:

$$\theta_s = \frac{V_w}{V_f} = \frac{V_w}{V_a + V_w}.$$

Often referred to as just *saturation* θ_s represents the volume of water present in a soil relative to the volume of the pores. It ranges from 0% for dry soils to 100% for complete saturation, though 100% is rarely obtained due to trapped air pockets. It is not a good index for soils that swell, such as clay soils.

$$\theta_s = \frac{\theta}{f}.$$

Of the measures defined, the most commonly used in mathematical models of soil - water, and thus the ones of most interest to us are *porosity* f, *bulk density* ρ_b, and *volume wetness* θ.

2.3 The state of water in the soil

In terms of energy distribution the Kinetic energy of water in the soil is negligible whilst it is the Potential energy of water in the soil that is of primary importance.

Differences in potential energy between points in the soil gives rise to water flow from areas of high potential energy to low. Water moves constantly in the direction of decreasing potential. The rate of decrease of potential energy with distance is the moving force causing flow. It is the relative levels of potential energy in different regions that is important.

Soil Water Potential expresses the specific potential energy of soil water relative to that of water in a standard reference state.

N.B. The *standard reference state* is a hypothetical reservoir of pure- free water at atmospheric pressure and at the same temperature as soil water and given constant elevation. Since elevation is arbitrary, potential is not absolute.

The force acting on soil water, directed from a zone of higher to lower potential, is equivalent to the negative *potential gradient* $-d\phi/dx$, the change of energy potential ϕ with distance x. The negative sign indicates that it acts in the direction of

decreasing potential.

The possible values of soil-water potential are continuous excepting for possible abrupt changes at changes of phase.

When soil is saturated, the hydrostatic pressure is greater than atmospheric pressure and water will want to move from the soil to the 'reference reservoir'. This is where soil water potential is positive. The soil water potential is negative when the soil is unsaturated and the hydrostatic pressure is less than atmospheric pressure resulting in water tending to move from the 'reservoir' to the soil.

Soil water potential at any point in the soil depends upon these factors:

- hydrostatic pressure.
- elevation (relative to reference elevation).
- concentration of solutes.
- temperature.

The *total potential of soil water* is defined as the amount of work that must be done per unit quantity of pure water in order to transport reversibly and isothermally an infinitesimal quantity of water from a pool of pure water at a specified elevation at atmospheric pressure to the soil water (at the point under consideration).

In practice it is measured by a related property such as hydrostatic pressure, vapour pressure, elevation etc. The specification of an *infinitesimal amount* is so stated as

to ensure that neither the reference state or soil water potential are changed. Soil water is subject to force fields causing its potential to differ from pure, free water. Force fields are caused by:

- the attraction of the solid matrix for water,
- the presence of solutes,
- the action of external gas pressure, and
- gravitation.

The total potential is the sum of these factors:

$$\phi_t = \phi_g + \phi_p + \phi_o + \dots ,$$

where t is the total, g is gravitational, p is pressure, o is osmotic and ... is to account for additional theoretically possible terms.

Thermodynamically speaking in the classical sense of equilibrium states and reversible processes, equilibrium states occur only rarely in nature and spontaneous processes tend to be irreversible. However, approaching the subject classically, the potential concept depends on 2 laws.

1. The energy conservation law where the heat dQ added to the system is equal to the sum of the change in internal energy of the system dU, the work of

expansion done by the system PdV, and all other work done dW by the system on its surroundings. Expressed mathematically this is:

$$dQ = dU + PdV + dW.$$

2. The direction of change of an isolated system is always toward equilibrium, which rests on the properties of absolute temperature T always being positive and entropy S (the measure of internal disorder or randomness of a system) where:

$$dQ = TdS \text{ for reversible processes,}$$

$$dQ < TdS \text{ for irreversible processes,}$$

and dQ is the heat input into the system. The second law is stated as:

$$dU = TdS - PdV.$$

In a system of variable composition, the total differential of the internal energy can be expressed as a function of S, V and n_i where n_i is the number of moles in a given component.

$$dU = \left(\frac{\partial U}{\partial S}\right)_{V,n_i} dS + \left(\frac{\partial U}{\partial V}\right)_{s,n_i} dV + \left(\frac{\partial U}{\partial n_i}\right)_{S,V,n_j} dn_i.$$

Gibbs free energy of the system is:

$$G = U + PV - TS.$$

Chemical potential of a component is μ_i which is defined as the partial molal Gibbs free energy of that component $\overline{G_i}$:

$$\overline{G_i} = \left(\frac{\partial G}{\partial n_i}\right) T, P, n_j = \mu_i.$$

The Total differential of the chemical potential is:

$$d\mu_i = \left(\frac{\partial \mu_i}{\partial T}\right)_{P,n_i} dT - \left(\frac{\partial \mu_i}{\partial P}\right)_{T,n_i} dP - \left(\frac{\partial \mu_i}{\partial n_i}\right)_{T,P,n_j} dn_i.$$

The chemical potential is an expression of the potential energy state of a component in a mixed system in the absence of external forces, that is, when temperature, pressure and composition are the only effective variables.

The difference in chemical potential between water in the soil and pure free water at the same temperature has been called *moisture potential.*

The *Total Potential* of soil water comprises of the following components, each discussed separately:

Gravitational Potential - Dependent only upon elevation, the gravitational potential at each point in the soil is calculated relative to a reference level positioned such that gravitational potential can always be taken as positive or zero.

At height z above reference, the gravitational potential energy E_g of mass M of water, occupying a volume V is:

$$E_g = Mgz = \rho_w Vgz.$$

Gravitational potential in terms of potential energy per unit mass is:

$$\phi_g = gz,$$

and in terms of potential energy per unit volume is:

$$\phi_g = \rho_w gz.$$

Pressure Potential - Commonly known as *tension* or *suction*, the pressure potential is positive when soil water hydrostatic pressure is greater than atmospheric pressure and negative when the reverse is true.

Positive pressure potential occurring below ground water level is termed *submergence potential.*

The hydrostatic pressure P of water with respect to atmospheric pressure is:

$$P = \rho gh,$$

where h is the submergence depth below the free water surface.

The potential energy of this water is:

$$E = PdV,$$

and the submergence potential or potential energy per unit volume is:

$$\phi_{ps} = P.$$

Negative Pressure Potential, which is also called *matric potential* or *capillary potential*, results from capillary and adsorptive forces due to the soil matrix

which attract and bind water in the soil and lower its potential energy below that of bulk water.

The equation of *capillarity* for an unsaturated, 3-phase, soil system is:

$$p_0 - p_c = P = \gamma \left(\frac{1}{R_1} + \frac{1}{R_2} \right),$$

where p_0 is the atmospheric pressure, usually taken as zero, p_c is the pressure of soil water, which can be less than the atmospheric pressure, P is the pressure deficit or subpressure of soil water, γ is the surface tension of the water and R_1 and R_2 are the principle radii of curvature of a point on the meniscus.

Adsorption is the formation of hydration envelopes over the particle surfaces and is highly important in clayey soils and relatively negligible in sandy soils.

Both capillarity and adsorption are interrelated and changing one affects the other.

Soil may exhibit either matric potential or positive pressure potentials but not both simultaneously. Unsaturated soil has only matric potential which is expressible in negative pressure units.

Dealing only in pressure potential, ϕ_p, both negative and positive, allows use of a continuous potential from a saturated region to the unsaturated region.

Pneumatic Potential which results from the change in atmospheric pressure is negligible in practice.

Osmotic Potential The presence of solutes in the soil affects the thermodynamic

properties of the soil and lowers both the potential energy and the vapour pressure in the soil. Osmotic potential is important in the interaction between plant roots and the soil and in vapour diffusion.

There are three quantitative ways in which soil water potential is expressed. These are:

1. Energy per unit mass L^2T^{-2}.

2. Energy per unit volume, which is in a direct proportion to mass since water is practically incompressible, $ML^{-1}T^{-2}$.

3. Energy per unit weight or *hydraulic head* is the height of a liquid column corresponding to the given pressure. Therefore instead of $\phi = \phi_g + \phi_p$ we could have

$$H = H_g + H_p. \tag{2.1}$$

pF is a logarithmic scale for pressure head in cm such that a pF of 1 = 10cm H_2O and a pF of 3 = 1000cm H_2O etc $\cdots$

To convert between different methods for expressing soil water potential, defining ϕ to designate potential in terms of unit mass, we use

$$\phi = \frac{P}{\rho_w} \quad \text{and}$$

$$H = \frac{P}{\rho_w g} = \frac{\phi}{g}.$$

For a saturated soil at equilibrium, with free water at the same elevation, the hydraulic pressure and suction equal zero.

If a slight suction is applied to water in a saturated soil no outflow may occur until the suction increases past the critical value at which the largest pore of entry begins to empty. This critical value is called *air-entry suction* and is more distinct in coarse textured soils as the pores are more uniform.

As suction increases, increasingly smaller pores empty, until at very high suction only very narrow pores retain water. Increases in suction also result in decreases in the thickness of hydration envelopes covering particle surfaces and thus a decrease in soil wetness. The amount of water in soil at equilibrium is a function of the sizes and volumes of the water filled pores and hence a function of the matric suction. This function is usually measured experimentally.

Empirical relations are used to predict the matric suction versus wetness relationship from basic soil properties as the complexities prohibit a theoretical model.

Two such empirical equations are the one proposed by Visser [34] in 1966:

$$\psi = \frac{a(f-\theta)^b}{\theta^c},$$

where ψ is the matric suction, f is the porosity and θ is the volumetric wetness. a,b and c are constants which in practice are hard to evaluate.

The other was proposed by Gardner [10] in 1970:

$$\psi = a\theta^{-b},$$

but only fits a small range of the characteristic curve but may be useful in analyzing the processes in which the water content range is narrow, for example redistribution or internal drainage.

Water retention by the soil at low suction is affected by soil structure (the capillary effect and pore size distribution). At high suction it is affected by the texture and specific surface area of the soil material and adsoption is the main factor.

The greater the clay content the greater the water content in the soil at any given suction.

Water in an unsaturated soil, where the soil water is at sub atmospheric pressure, will not tend to seep into the atmosphere. To flow spontaneously out of the soil and into the atmosphere, soil water pressure must exceed atmospheric pressure. Similarly, for a fine textured soil to drain into the large pores of an initially dry coarse textured layer, soil water must be at nearly atmospheric pressure. The slope of the soil moisture characteristic curve (change of water content per unit change of matric potential) is termed the *differential (or specific) water capacity*:

$$c(\theta) = c_\theta = \frac{d\theta}{d\phi_p} \quad \text{or} \quad c(\theta) = c_\theta = -\frac{d\theta}{d\psi}, \tag{2.2}$$

where c_θ depends on wetness range, texture and hysteresis effect.

Hysteresis is an effect observed when the functional relationship between two variables, in this case matric suction and water content, is not generally unique and single-valued. In the cases of desorption, when the soil is drying, the function of

CENTRAL QUEEN LAND

matric suction to water content is a continuous curve. Sorption, when the soil is wetting, also produces a continuous curve, but this curve is not necessarily identical to the one for desorption. Hysteresis is a property common to many physical systems and is commonly observed in magnetism.

2.4 Richards' flow equation

For the case of a soil at saturation point, hydraulic pressure is positive. The gravitational potential reference level is chosen arbitrarily such that gravitational potential is always greater than or equal to zero. The soil water potential gradient is driven by the hydraulic potentials which can be expressed in terms of hydraulic head. Therefore, movement of water within the soil can be defined by the hydraulic head gradient. The discharge rate of water through an area of soil, termed the *flux* q, is proportional to the hydraulic head gradient $\triangle H/L$ where $\triangle H = H_2 - H_1$ in Figure 2.2 below.

Thus the flux is represented as:

$$q = K\left(\frac{\Delta H}{L}\right), \tag{2.3}$$

where K is a constant for a given soil type known as *hydraulic conductivity.* Equation (2.3) is known as Darcy's Law. Application of Darcy's law to the situation of vertical flow where the hydraulic head is composed of both gravitational and pressure heads is straight forward. Figure 2.3 illustrates the situation.

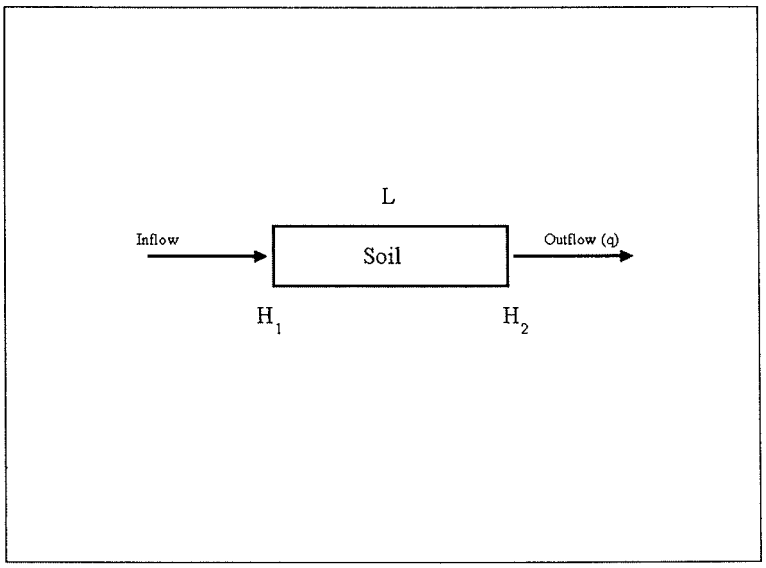


Figure 2.2: Hydraulic Head in Horizontal Flow

For $\triangle H$, the change in pressure head, to be determined, the head at the soil surface and at the base of the soil column must be known. For a column of height L cm the gravitational head is L cm. If there is h cm of water maintained above the column, the pressure head is h cm. Choosing the reference level at the base of the column ensures that both the pressure and gravitational heads at the base are zero. Therefore:

$$\triangle H = H_{\text{surface}} - H_{\text{base}} = (h + L) - 0 = (h + L).$$

Substitution into (2.3) gives flux as:

$$q = K\left(\frac{\triangle H}{L}\right) = K\frac{(h + L)}{L} = K\frac{h}{L} + K.$$

Since Darcy's Equation (2.3) relies upon constancy of flux q and head gradient, which does exist for steady state flow but not for transient flow, a continuity equa-

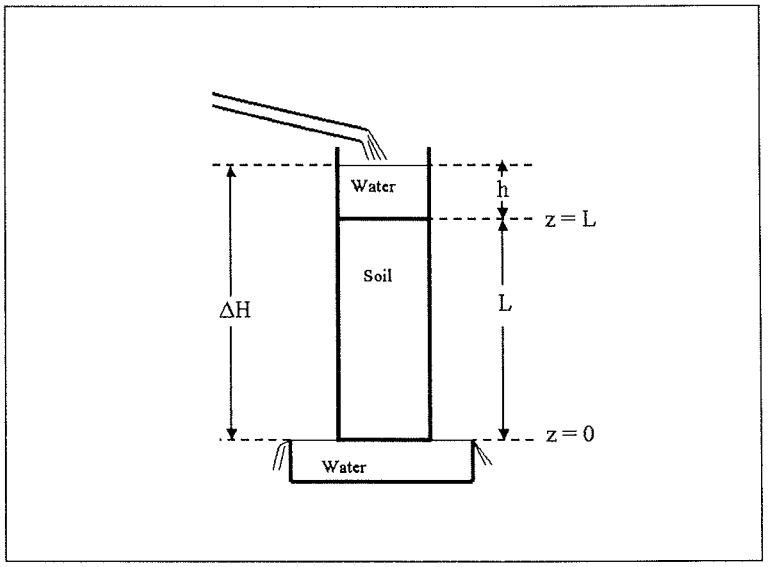


Figure 2.3: Hydraulic Head in Vertical Flow.

tion, embodying the principle of conservation of matter, is introduced to overcome this difficulty of fully describing water movement. For one dimensional flow, the relationship that the rate of change of flux in a particular direction increases as the volume of stored water decreases over time is expressed as:

$$\frac{\partial \theta}{\partial t} = -\frac{\partial q}{\partial z}, \tag{2.4}$$

where θ is the volumetric water content.

The combination of Darcy's Equation (2.3), which written in differential form is:

$$q = -K\frac{\partial H}{\partial z},$$

and Equation (2.4) yields the equation for one dimensional flow in saturated soils:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}\left(K\frac{\partial H}{\partial z}\right).$$

In unsaturated soils, as is normally the case in nature, the driving force of water movement in the soil is due to the negative pressure of *matric suction* ψ. Additionally, hydraulic conductivity, K is not a constant in an unsaturated soil, but is dependent on the value of θ, thus $K(\theta)$. For a given pressure head H_p, matric suction $\psi = -H_p$. Bearing this relationship in mind, from Equations (2.4) and (2.3) we have:

$$\begin{aligned}\frac{\partial \theta}{\partial t} &= -\frac{\partial q}{\partial z} \\ &= -\frac{\partial}{\partial z}\left(-K(\psi)\frac{\partial H}{\partial z}\right).\end{aligned}$$

Now breaking down the hydraulic head H into it's individual components of matric suction ψ and gravitational head z, $H = \psi - z$ in the case of a vertical column, and substituting we have:

$$\begin{aligned}\frac{\partial \theta}{\partial t} &= \frac{\partial}{\partial z}\left(K(\psi)\frac{\partial(\psi - z)}{\partial z}\right) \\ &= \frac{\partial}{\partial z}\left(K(\psi)\frac{\partial \psi}{\partial z}\right) - \frac{\partial}{\partial z}\left(K(\psi)\frac{\partial z}{\partial z}\right) \\ &= \frac{\partial}{\partial z}\left(K(\psi)\frac{\partial \psi}{\partial z}\right) - \frac{\partial K(\psi)}{\partial z}.\end{aligned}$$

This is known as the potential form of *Richards' flow equation* since it is based upon the matric potential ψ.

Another form of this equation exists which is known as the *diffusivity* form, as it parallels known solvable diffusivity differential equations. To arrive at the diffusivity

form of Richards' equation, we exploit the dependence relationship between ψ and θ (assuming no hysteresis[1]). This gives:

$$\begin{aligned} \frac{\partial \theta}{\partial t} &= \frac{\partial}{\partial z}\left(K\frac{\partial \psi}{\partial z} - K\right) \\ &= \frac{\partial}{\partial z}\left(K\frac{\partial \psi}{\partial \theta}.\frac{\partial \theta}{\partial z} - K\right). \end{aligned} \tag{2.5}$$

Recalling Equation (2.2) and defining a soil-water diffusivity function $D(\theta)$ such that:

$$D(\theta) = \frac{K(\theta)}{c(\theta)},$$

substitution into Equation (2.5) gives:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}\left(D(\theta)\frac{\partial \theta}{\partial z} - K(\theta)\right), \tag{2.6}$$

which is the diffusivity form of Richards' equation.

Since θ represents volumetric water content, a volumetric water uptake term $S(\theta, t)$ can be introduced into Equation (2.6) representing water lost due to plant root action. This makes the flow equation:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}\left(D(\theta)\frac{\partial \theta}{\partial z} - K(\theta)\right) - S(\theta, t). \tag{2.7}$$

[1] "The relation of conductivity to volumetric wetness $K(\theta)$ is affected by hysteresis to a much lesser degree than is the $K(\psi)$ function" - Hillel cf [14]

2.5 Evapotranspiration

In a cropped soil, water can be lost through the upper boundary by the action of two processes:

1. Evaporation - loss directly from the upper surface of the soil.

2. Transpiration - loss through evaporation through a plant's leaf surface.

Since these two processes are so closely related, they are encompassed by the term *evapotranspiration.* The exact proportioning of water loss between either evaporation or transpiration is dependant on the stage of growth and density of a crop in the soil.

2.6 Modeling water flow in cropped soils

The aim behind producing models to simulate the movement of water through cropped soils is a simple one. It is, simply put, to provide some deeper understanding of the state of water in the soil and thus guide management practices to aid the conservation of water.

Water content within the soil is governed by either the gain of water to the soil or the loss of water from it. In the area of particular interest to us that is the root zone of a crop, water may be sourced from infiltration at the soil surface or, due to the effect of capillary action, from the rising of water from a water table. Loss of water from

the root zone may be due to water uptake by plant roots, surface evaporation, or by percolation of water deeper into the soil due to gravity. The particular characteristics of the soil determine the rate of change of the water content. Water uptake by roots, which occurs in the upper soil zone where, due to evapotranspiration, drying can occur quickly, is of particular interest in [29].

Richards' equation, also referred to as the classic flow equation, defines the way in which water redistribution occurs within a particular soil. It is a combination of Darcy's law and the continuity equation. Darcy's Law states that the volumetric flux of water is proportional to the driving force set up by the potential gradient. In examining cropped soils it is necessary to enhance the Richards' flow equation for once dimensional flow under gravity with a volumetric sink term:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}[D(\theta)\frac{\partial \theta}{\partial z} - K(\theta)] - S(z,t),$$

where θ is the volumetric water content, z is the vertical distance positive downwards, K is the Conductivity of the soil, D is the Diffusivity of the soil, S is a sink term accounting for water loss with depth and t is time.

Other sinks and sources of water are included in the mathematical formulation of the model as upper and lower boundary conditions.

2.7 Defining the sink term

Models for sink terms have ranged widely and include forms that are wholly empirical such as [21] and many that are fairly mechanistic like those presented in [9], [22], [6], [12], and [13].

The model of Molz and Remson [21] uses as its basis a 40% : 30% : 20% : 10% uptake pattern for plant transpiration requirements with the 40% taking place in the upper soil level, 30% the next level down etc. Based on this format, the sink term presented looks as follows:

$$S(z) = -\frac{1.6T}{v^2}z + \frac{1.8T}{v} \qquad 0 \leq z \leq v, \tag{2.8}$$

where S is the moisture extraction rate per unit volume of soil, T is the transpiration rate per unit area of soil surface and v is the vertical length of the root system.

For most crops in soils where soil moisture is maintained at a high level, greater root development occurs at the surface layers. Thus, this extraction pattern approximates the pattern inherent to most crops.

In [21], Molz and Remson noted that this particular model failed when drying occurred in the upper layers and that the 40% assumption for uptake in the uppermost layer could not be obtained due to insufficient moisture in the zone. To deal with this situation Stonier and Janz [29] introduced an *'evaporation or drying front'* which they describe as "The depth below the surface of the soil, above which, the soil

wetness is less than some critical water content where water uptake is inhibited." The following Figure 2.4 illustrates the idea of the evaporation front e produced by extended evapotranspiration. An assumption is made the water uptake of the root system adjusts to remove the plants entire transpiration demand from the zone $v-e$ containing sufficient moisture.

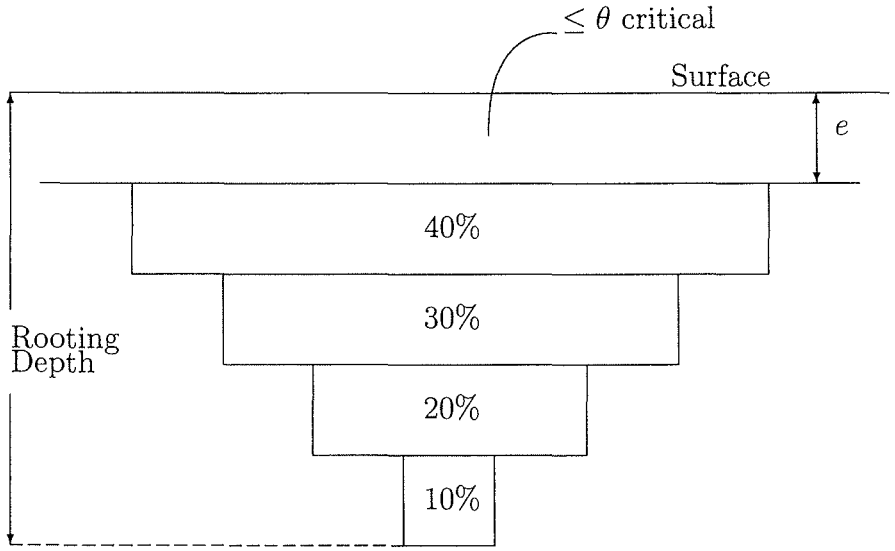


Figure 2.4: Extraction pattern of root system after substantial evapotranspiration

To allow for the roots not being effective in the upper zone, the following sink term was derived which takes into account the evaporation front:

$$S = \frac{-1.6T}{(v-e)^2}z + \frac{(1.8v - 0.2e)T}{(v-e)^2}, \tag{2.9}$$

where e is the evaporation front.

The derivation of this equation can be obtained from [18].

2.8 Numerical model

In order to determine soil water content profiles at any time, the Richards' equation incorporating the above sink term needs to be solved. Whilst an analytical solution would enable the physics of the problem to be well developed as in [15], a solution of this nature would require that the diffusivity function, and initial boundary conditions, have restrictive assumptions made about them. These restrictive assumptions would be most inappropriate given that the model has been developed with the aim of simulating a range of field processes with application to field management practices. Thus a numerical solution to the problem has been pursued.

By use of the method of finite differences, no restrictive assumptions need be made in order to find a solution. However, problems can be encountered in getting the finite difference method to handle the rapid changes that occur at the surface at the start of infiltration events as noted by Hogarth and Watson [15]. A macroscopic approach to water redistribution is taken with the following model which operates with a time frame of one day. Thus, the monitoring of precise surface water content changes is avoided.

To express all equations in finite difference form, the Crank-Nicolson method, the full details of which can be found in [18], was used. It was chosen because it guarantees second order correctness for both the time t and depth z independent variables and also because it is stable for all ratios of $\triangle z$ to $\triangle t$. The depth steps are defined as $z_i = (i - 1/2)\triangle z$ where $i = 1, 2, \cdots, N$, and $t_j = j\triangle t$, $j = 1, 2, \cdots, M$.

An implicit central difference approximation was employed with the derivatives evaluated about the point $z_i, t_{j+1/2}$. The nonlinearity arising from the product of $\partial\theta/\partial z$ and $D(\theta)$ was taken care of by formulating the first derivative of the space derivative at the points $z_{i\pm1/2}, t_{j+1/2}$. Thus, by writing $D(\theta_i^j)$ as D_i^j the finite difference equation is formed:

$$\begin{aligned} &D_{i-1/2}^{j+1/2}\theta_{i-1}^{j+1} - (D_{i-1/2}^{j+1/2} + D_{i+1/2}^{j+1/2} + A)\theta_i^{j+1} + D_{i+1/2}^{j+1/2}\theta_{i+1}^{j+1} = \\ &-D_{i-1/2}^{j+1/2}\theta_{i-1}^{j} + (D_{i-1/2}^{j+1/2} + D_{i+1/2}^{j+1/2} - A)\theta_i^{j} - D_{i+1/2}^{j+1/2}\theta_{i+1}^{j} \\ &+2\triangle z(K_{i+1/2}^{j+1/2} - K_{i-1/2}^{j+1/2}) + 2\triangle z^2 S_i^{j+1/2}, \end{aligned} \tag{2.10}$$

and

$$S_i^j = \frac{-1.6T_j}{(v(j\triangle t) - e_j)^2}(i - 1/2)\triangle z + \frac{(1.8v(j\triangle t) - 0.2e_j)}{(v(j\triangle t) - e_j)^2}T_j, \tag{2.11}$$

where $A = 2\triangle z^2/\triangle t$.

Surface flux is given by Darcy's Law. Thus, evaporation at the soil surface gives the derivative boundary condition:

$$q(t) = -D(\theta)\frac{d\theta}{dz} + K(\theta), \tag{2.12}$$

where q is the volumetric surface flux.

By using the finite difference formulation of Darcy's law for water flux under gravity, the imaginary above-ground points in Equation (2.10) can be eliminated to yield:

$$
\begin{aligned}
&-(D_{3/2}^{j+1/2}+A)\theta_1^{j+1}+D_{3/2}^{j+1/2}\theta_2^{j+1}= \\
&\quad -D_{1/2}^{j+1/2}\theta_0^j+(D_{1/2}^{j+1/2}+D_{3/2}^{j+1/2}-A)\theta_1^j-D_{3/2}^{j+1/2}\theta_2^j \\
&\quad -D_{1/2}^{j+1/2}\triangle z(q_{1/2}^{j+1}-K_{1/2}^{j+1})/D_{1/2}^{j+1}+2\triangle z(K_{3/2}^{j+1/2}-K_{1/2}^{j+1/2}) \\
&\quad +2\triangle z^2 S_1^{j+1/2}, \qquad (2.13)
\end{aligned}
$$

where θ_0 at the $(j+1)$st time step is:

$$
\theta_0^{j+1}=\theta_1^{j+1}+\frac{\triangle z}{D(\theta_{1/2}^{j+1})}(q_{1/2}^{j+1}-K(\theta_{1/2}^{j+1})). \qquad (2.14)
$$

The equation set formed from Equation (2.13), over all depth and time steps, is tridiagonal in form. Solution of this set of equations can be undertaken utilising Thomas' algorithm.

An assumption is now made that the water contents on the lower boundary are known and are not represented by a flux condition. Thus at $i=N+1/2$ the values of the water contents are denoted by θ_L. By evaluating Equation(2.10) at $i=N$, θ_{N+1} is eliminated to obtain:

$$
\begin{aligned}
&D_{N-1/2}^{j+1/2}\theta_{N-1}^{j+1}-(D_{N-1/2}^{j+1/2}+2D_{N+1/2}^{j+1/2}+A)\theta_N^{j+1}= \\
&\quad -D_{N-1/2}^{j+1/2}\theta_{N-1}^j+(D_{N-1/2}^{j+1/2}+D_{N+1/2}^{j+1/2}-A)\theta_N^j-D_{N+1/2}^{j+1/2}\theta_{N+1}^j \\
&\quad -2D_{N+1/2}^{j+1/2}\theta_L^{j+1}+2\triangle z(K_{N+1/2}^{j+1/2}-K_{N-1/2}^{j+1/2})+2\triangle z^2 S_N^{j+1/2}, \qquad (2.15)
\end{aligned}
$$

where

$$\theta_{N+1}^{j} = 2\theta_{L}^{j} - \theta_{N}^{j}. \tag{2.16}$$

It can be seen in Equation (2.15) that Diffusivity D and Conductivity K need to be evaluated at points $z_i, t_{j+1/2}$. This is a consequence of the Crank-Nicolson requirements. To do this $\theta_i^{j+1/2}$ must first be found and D and K calculated using the following empirical relationships, [4]:

$$\begin{aligned}
\Psi(\theta) &= \Psi_s(\theta/\theta_s)^{-b}, && \text{for } 0 \leq \theta \leq \theta_s, \\
K(\theta) &= K_s(\theta/\theta_s)^{2b+3}, \\
D(\theta) &= -K(\theta)d\Psi/d\theta &&= b\Psi_s K_s(\theta_s)^{-(b+3)}\theta^{b+2},
\end{aligned}$$

where b,θ_s,Ψ_s and K_s are parameters particular to unique soil types. The s subscript signifies that the values are at a saturated water content.

A two term Taylor series is employed to project the water contents a half time step ahead:

$$\theta_i^{j+1/2} = {\theta_i}^j + \left(\frac{\partial\theta}{\partial t}\right)_i^j (\triangle t/2),$$

with an approximation for $(\partial\theta/\partial t)_i^j$ using $\frac{\partial}{\partial z}[D(\theta)\frac{\partial\theta}{\partial z} - K(\theta)] - S(z,t)$ evaluated at z_i, t_j.

What follows is the algorithm for determining the moisture content profile in the soil:

- Initialise the system by:

- loading the preliminary soil moisture content profile.
- loading the evapotranspiration data.
- loading the lower boundary profile.
- setting depth, depthsize and time of simulation.
- setting $e = 0$.

- Repeat until Number of time steps $= N$.
  - Estimate moisture contents one-half time step ahead.
  - Use the estimates to evaluate water contents at all depths and each time step by solving the finite difference formulation with boundary conditions.
  - Update new moisture contents, monitor and update the evaporation front if necessary.

2.9 Comparison with field data

Results from the computer simulation were compared with results obtained from Olsen and Rose [23] which consisted of experimental data comprising an initial soil moisture profile and 18 days of evapotranspiration data. The type of soil described in [23] was approximated for use in the computer simulation by use of appropriate soil parameters of the types given in [4]. The parameters chosen for the soil reflected a uniform silty clay with a 49% fraction of clay and the rooting depth was set as 120cm.

A suitable degree of accuracy for the Crank-Nicolson method was determined by choosing arbitrary initial values and halving them through each successive program completion until an accuracy of two decimal places was achieved in both the time and the depth steps. The values arrived at were 1 day for the time step and 10cm for the depth step.

Results obtained with the depth step set to 4cm instead of 10cm yielded the same results, all be it with slightly more resolution of the contours.

A graph of the results obtained is reproduced below as Figure 2.5.

The variation of the simulation result from the actual profile beginning at around 20 cm was attributed to the nature of the split layered soil in which the original data was gathered. This layered nature of the original soil accounted for the 'humps' in the original profile.

The following were noted as limitations of the simulation model:

- It would be preferred to replace the wholly empirical modified sink term, incorporating the evaporation front, with a mechanistic model if such a model existed for a broad range of crop types and proved to be accurate.

- There are a number of disadvantages inherent in using the diffusivity-form of the flow equation regarding water content:

    1. The relationship between water content and soil potentials is not unique and single valued in many soils.

2. It is stated in [15] that water contents are discontinuous across the layer interface in non homogeneous soils (e.g. layered soils). Since pressure head is continuous in these types of soils it would be preferable to use the soil potential form of the flow equation.

- There is a high degree of variance in the parameters for the empirical functions of D, K and θ. This is the result of these functions being developed as power curves from measured data, [4].

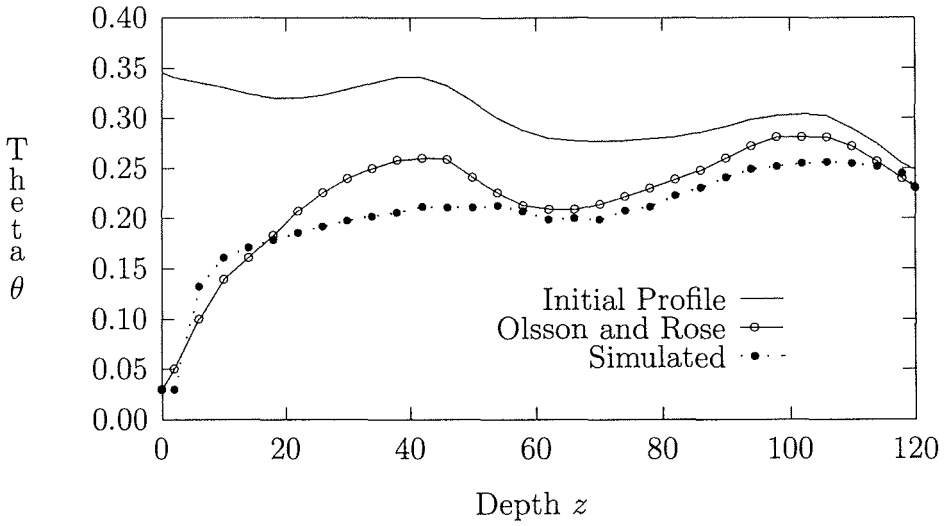


Figure 2.5: Comparison of Data. z in cm. θ in cm^3/cm^3.

2.10 Simulating the irrigation cycle

Being able to model the moisture profile within soil at a particular point in time with respect to the plant root zone would present invaluable assistance in terms of

water conservation. Traditionally, the irrigation cycle has consisted of relatively long periods of evapotranspiration followed by brief periods of infiltration. It is during the transpiration events that the status of water in the soil is monitored in order to determine the timing and amount of the next irrigation.

Taking the model developed above based on the classic Richards' equation and modified sink term (allowing for the evaporation front) some additional work is done to account for a variation in conditions at the upper and lower boundaries that may be encountered in the field. The various conditions that may occur are:

Dirichlet conditions - The dependent variable, in this case θ is specified at all depths initially and at the boundaries.

Neumann conditions - The dependent variable is specified at all depths initially and it's derivative is specified at the boundaries.

Mixed conditions - A mixture of both Dirichlet and Neumann conditions.

In the situation considered for simulation, mixed conditions apply with the upper boundary consisting of Neumann conditions, due to infiltration and evapotranspiration, and the lower boundary having Dirichlet conditions. In the case of the lower boundary it was assumed that the water contents could be measured at any time. Another assumption was that the initial distribution of water at all depths was known.

In the case of evapotranspiration, the nonlinear flux condition is the direct result of

application of Darcy's Law. In the case of infiltration an assumption was made that the upper surface became instantaneously saturated the moment infiltration began. This last assumption whilst clearly not accurate was considered acceptable due to the macroscopic nature of the simulation.

2.11 Modifications to the numerical model

Incorporating the boundary conditions into the finite difference equations resulted in the formation of two sets of equations:

1. Where a derivative boundary condition exists, representative of an evaporation event.

2. Where the boundary condition is a constant, representative of an infiltration event.

The finite difference equations before the boundary conditions are taken into account are given in Equations (2.10) and (2.11).

For $i = 1$ with an Evaporation event:

$$\begin{aligned} -(D_{3/2}^{j+1/2} + A)\theta_1^{j+1} + D_{3/2}^{j+1/2}\theta_2^{j+1} = \\ -D_{1/2}^{j+1/2}\theta_0^j + (D_{1/2}^{j+1/2} + D_{3/2}^{j+1/2} - A)\theta_1^j - D_{3/2}^{j+1/2}\theta_2^j \\ -D_{1/2}^{j+1/2}\triangle z(q_{1/2}^{j+1} - K_{1/2}^{j+1})/D_{1/2}^{j+1} + 2\triangle z(K_{3/2}^{j+1/2} - K_{1/2}^{j+1/2}) \\ + 2\triangle z^2 S_1^{j+1/2}, \end{aligned} \tag{2.17}$$

where θ_0 at the jth time step is as follows:

$$\theta_0^j = \theta_1^j + \frac{\triangle z}{D(\theta_{1/2}^j)}(q_{1/2}^j - K(\theta_{1/2}^j)). \tag{2.18}$$

For $i = 1$ with and Infiltration event:

$$\begin{aligned} &-(2D_{1/2}^{j+1/2} + D_{3/2}^{j+1/2} + A)\theta_1^{j+1} + D_{3/2}^{j+1/2}\theta_2^{j+1} = \\ &\quad (2D_{1/2}^{j+1/2}\theta_0^j + D_{3/2}^{j+1/2} - A)\theta_1^j - D_{3/2}^{j+1/2}\theta_2^j \\ &\quad 2\triangle z(K_{3/2}^{j+1/2} - K_{1/2}^{j+1/2}) - 4D_{1/2}^{j+1/2}\theta_s + 2\triangle z^2 S_1^{j+1/2}. \end{aligned} \tag{2.19}$$

For $i = N$:

$$\begin{aligned} &D_{N+1/2}^{j+1/2}\theta_{N-1/2}^{j+1} - (D_{N-1/2}^{j+1/2} + 2D_{N+1/2}^{j+1/2} + A)\theta_N^{j+1} = \\ &\quad -D_{N-1/2}^{j+1/2}\theta_{N-1}^j + (D_{N-1/2}^{j+1/2} + D_{N+1/2}^{j+1/2} - A)\theta_N^j - D_{N+1/2}^{j+1/2}\theta_{N+1}^j \\ &\quad -2D_{N+1/2}^{j+1/2}\theta_{N+1/2}^{j+1} + 2\triangle z(K_{N+1/2}^{j+1/2} - K_{N-1/2}^{j+1/2}) + 2\triangle z^2 S_N^{j+1/2}, \end{aligned} \tag{2.20}$$

where $\theta_{N+1}^j = 2\theta_{N+1/2}^j - \theta_N^j$.

The method of solution of the equations remains unchanged from that presented earlier. The only difference from this point is in the programming of the computer simulation and the incorporation of the boundary condition equations when switching between drying and wetting.

2.12 Switching between drying and wetting

The approach taken to simulate an entire cropping period was to merge together a series of evaporation events and a series of infiltration events with the choice of whether or not to irrigate being given to the user. On first inspection it seemed that to simply switch between no irrigation and irrigation would simply be a matter of calling the appropriate set of finite difference equations for either drying or wetting.

In developing the full simulation however, problems arose with the Crank-Nicolson method in that, in order to maintain second order correctness in respect to time, it is necessary that approximate water content values one half a time step ahead be known. In the previous simulation a Taylor series expansion, using only the water content values of the prior time step, provides a projection of current water content values.

In using this projection method a major flaw is presented. In switching from one event to another no allowance is made for the surface condition, which impacts on the state of the surface layers, when projecting one half time step forward. Considering the two types of events separately:

Evaporation → Infiltration Since the critical area of the top surface is assumed saturated under an infiltration event, projected values are not required. During the chosen time step the layers immediately underneath the surface layer will be affected by this assumption. Yet, in solving the tridiagonal system the first time following the switch, these upper layers soon adjust to realistic values.

The imposition of the surface condition has virtually no immediate affect of the lower soil layers and the Taylor series projection is performed as usual.

Infiltration → Evaporation Approximating realistic intermediate values in this case represented the major problem, since the effect of the boundary condition was not incorporated until the tridiagonal set was formed. To overcome this difficulty, in the event of such a switch occurring, the half time step projection is not performed. Rather, using the present values of θ (averaged in depth to provide half depth step values) as an approximation of the half time step projections, a tridiagonal set is formed and solved. Thus, a new profile is formed integrating the affects of evaporation, gravity and sink terms, and the surface conditions are enforced. By averaging the known values of θ at j with the provisional j set, a second approximation of the half-step forward projected profile is achieved. The formation and solution at this point of the tridiagonal set provides the foundation for further approximation using the usual method.

2.13 The simulation algorithm

Using the parameters for a silty clay soil with an arbitrary initial wetness profile, a root zone extending to a depth of 120cm and evapotranspiration data in keeping with a soil of this type, the simulation was developed according to the following algorithm:

- Setup initial conditions in the soil and at the boundary.

- Loop the following for N time steps.
  - At the end of each time step irrigation can be switched off or on at the users discretion.
    * If irrigation is implemented (or continued), engage the infiltration boundary conditions.
    * If irrigation changed from on to off, use the estimated θ's to solve for water contents at all depths, using the results to update water contents, then use these values to proceed.
    * If irrigation already off and staying off, project forward values of θ using Taylor's approximation.
  - Use estimates to solve for water content profile.
  - Upgrade water contents.
  - Display profile.

Running this algorithm and simulating 7 days of evaporation the appearance of the evaporation front in contrast to the initial profile is evident in Figure 2.6.
The decision was then made to turn on irrigation at the end of day 7 for three full days. The formation of the wetting front can easily be seen in Figure 2.7.

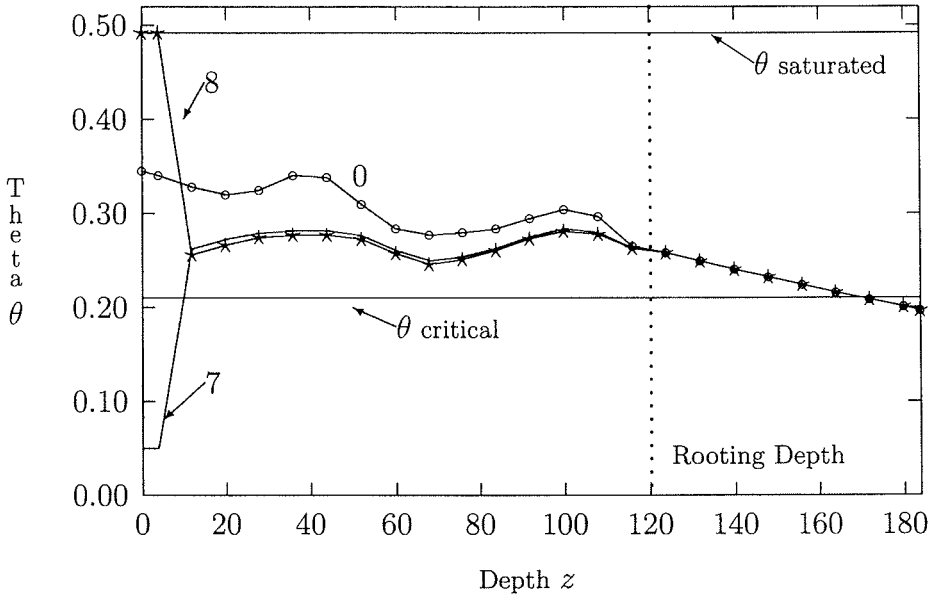


Figure 2.6: Water content profiles for days 0, 7 and 8. z in cm. θ in cm^3/cm^3.

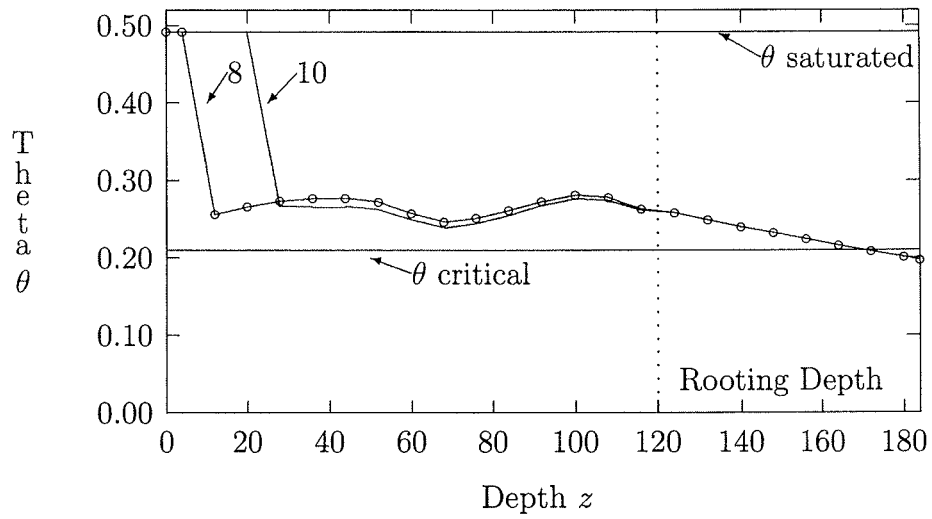


Figure 2.7: Day 10 profile following 3 days of max. infiltration. z in cm. θ in cm^3/cm^3.

Following the termination of irrigation at the end of day 10, the surface begins to dry as the front deepens as shown in Figure 2.8.

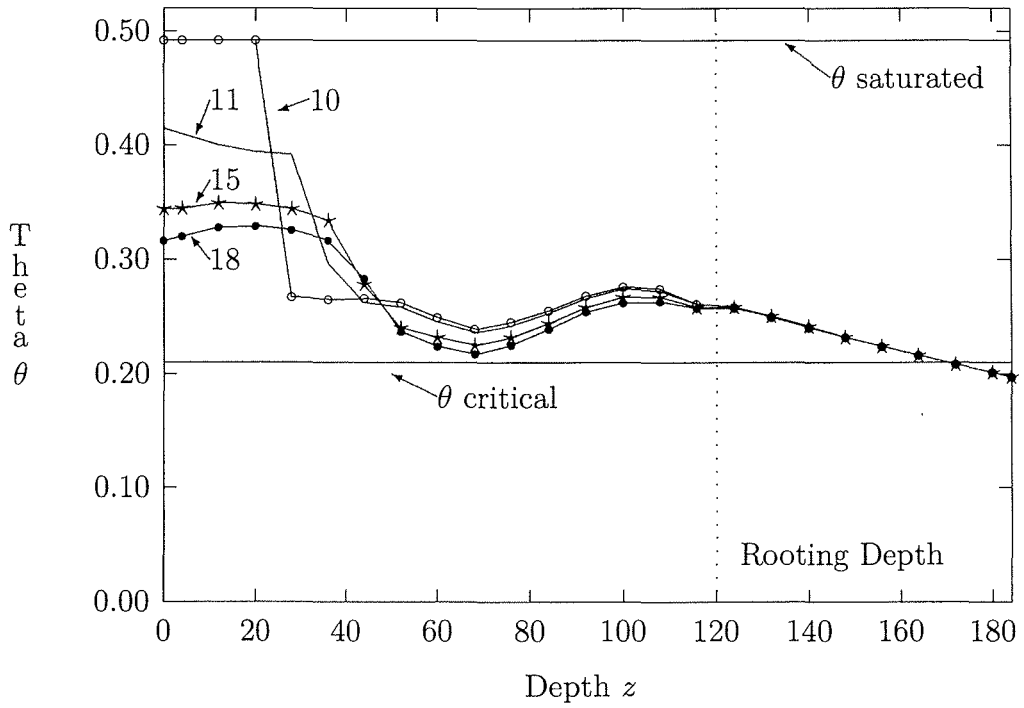


Figure 2.8: Content profiles representing 8 days after day 10. z in cm. θ in cm^3/cm^3.

2.14 Summary

The concepts presented and work undertaken which has been covered in this chapter is vital to the understanding of the remainder of this thesis. The computer simulation of an irrigation period, based upon the validated mathematical model with modified sink term, is to be combined with an evolutionary approach to optimisation. This leads us into the next chapter which presents an introduction to evolutionary algorithms.

Chapter 3

Introduction to evolutionary algorithms

3.1 Introduction

In this chapter we introduce the theory of genetic/evolutionary algorithms and explore the implementation of these algorithms in computer simulations. In particular focusing on the theory and developments necessary to our combination of these search algorithms with the problem of irrigation scheduling. In Chapters 4 and 5 we will be using both Genetic Algorithms (GAs) and Evolutionary Algorithms (EAs). The difference between the two being small but important. First, some background and explanation of these algorithms is required.

Genetic algorithms are based on the process of natural selection common to all

branches of biology and, although used for many years by biologists simulating genetic systems, their first inclusion in research literature dealing with artificial problems only took place in the 1960's due to John Holland. The aim of Holland was to develop genetic algorithms in order to imbue programs and machines with an unlimited capacity to adapt to arbitrary environments. In doing so he followed a population approach recognizing the concept of survival of the fittest.

The term *Genetic Algorithm* arose obviously from the fact the algorithms mimic the process of genetic evolution and survival of the fittest displayed by living creatures.

Genetic algorithms these days are primarily used as search methods for optimisation problems although this was not initially their function. They do not, as do classical mathematical search methods, require assumptions of continuity across the search space nor rely on the existence of derivatives of the objective function. They are also computationally simple.

Indeed, compared to traditional search methods, genetic algorithms can be identified by possessing the following characteristics:

- They use a coding of the parameter space rather than the parameter space itself.
- They use a population of points rather than a single point for search problems. They therefore have characteristics of parallel searching.
- They use an objective function only to refine the points in the population rather than derivatives as used in calculus based techniques for optimisation.

- The selection of new points in the search space is probabilistic rather than deterministic.

3.2 Robustness

The biological process of genetics is a very robust one. So too is the genetic algorithm approach to searching for the best individual. This has been proven through numerous studies on the subject and in the major reference "Adaption in Natural and Artificial Systems" by Holland (1975) [16]. Evolutionary algorithms, which will be investigated more fully in a later section, provide robust searches even in a complex space.

The use of a genetic algorithm as the search technique of choice for the research in Chapter 4 of this Masters Thesis came down to it's suitability for the task at hand and its robust performance compared to the following classical search techniques.

- *Hill Climbing Methods* So called because, if seeking a maximum, these searches seek in the direction of the maximum gradient. Being either Direct (solving equations related to the necessary condition that the gradient of the objective function be zero for a minima) or Indirect (seek local optima by searching point-wise in a direction related to the gradient of the objective function) in nature these methods are not robust because they depend on the continuity of the objective function within the search space, the existence of its partial derivative across the search space and because they are local in nature. These

methods are restricted to problems that satisfy optimally necessary conditions and the direct methods suffer greatly from the curse of dimensionality.

- *Dynamic Programming* An enumerative scheme, this method suffers badly from the curse of dimensionality

- *Random Search* A brute force approach, completely non intelligent, inelegant and requiring a great deal of processing power and time. Since this technique is purely random it does not build a picture of the search space over time.

- *Simulated Annealing* This approach uses random processes to explore for minimum energy states and is essentially a modified version of the hill climbing technique which assigns time based probability weightings to random moves within the search space. Since this technique deals with only one move at a time and does not use information from previous moves to guide move selection it can sometime be hard to obtain convergence to the global minimum of the objective function.

3.3 The mechanism of a genetic algorithm

Genetic algorithms, whilst utilising random choice as a tool, do so in conjunction with a search mechanism that is highly exploitative within the coding of the parameter space. They are a heuristic adaptive search technique that maintain a population $P(t) = \{x_1^t, \ldots x_n^t\}$ of individuals from one iteration in time t to the next $t+1$. Each

individual in a given population is a prospective solution to a given problem. As originally formulated the structure of each individual in a population consists of a binary encoding as referenced in Holland [16] and in Goldberg [11]. Michalewicz [19] realises a more modern structure that includes both integer and real encoding. Each individual has associated with it a measure of it's fitness for survival which plays an important part in the generation of the next population $P(t+1)$. To accomplish this generation two genetic operators, crossover and mutation are used. *Crossover* is the process whereby parts of usually more than one parent selected for mating are recombined into a new child. *Mutation* is the process whereby an individual's structure is perturbed, effectively producing a new individual. In line with the principles set down by Charles Darwin of natural selection and survival of the fittest, over many generations the 'best' individuals (i.e. those with the best solution to the problem at hand) make up the bulk of the final population.

Rather than manipulating the parameters of a problem, genetic algorithms deal with a coded parameter space of fixed length. This parameter space consists of a finite 'alphabet' which is typically the binary bits 0 and 1. The fitness of an individual is evaluated by way of an objective function. The algorithm itself is freed from assumptions of continuity and derivative existence and the constraints of requiring auxiliary information as the search domain is transparent to it.

3.3.1 Nomenclature

Because they are modelled on biological systems, the terminology of genetic algorithms parallels with that of the biological world, although every object name has a more mathematical counterpart.

A *chromosome* in biological terms is referred to as an individual or string whilst a biological *structure* signifies a population, or package of individuals. Particular parameter sets, solution alternatives or points in the solution space are derived from the structure. In biology a *phenotype* is the term referring to the combination of chromosomes forming an entire organism and its environment. Each chromosome is composed of *genes* which can take on a number of values called *alleles*. Genes correspond to the bits of a string and the alleles the binary alphabet of 0 and 1. Where individuals, or strings, are concerned, each has associated with it:

- It's decoded parameters being the Phenotype.
- The bit string or artificial chromosomes being the Genotype.
- It's Fitness Values - which under optimisation is the objective function.
- Inherent information on it's parents referred to as Auxiliary Information.

3.3.2 GA algorithm

In it's simplest form, a genetic algorithm can be expressed in pseudo code as follows:

```
begin
  t = 0
  Create random P(t)
  until (finished) repeat
  begin
    Evaluate Fitness of Strings in P(t)
    t = t+1
    Create P(t+1) from P(t)
  end
end
```

The method for creating a new population $P(t+1)$ is comprised of a selection process crossover and mutation, as mentioned earlier. There is a great deal of diversity in the way in which these operators do their job. The methods that are of relevance for the rest of this thesis are covered below, but for a more comprehensive treatment the reader may wish to examine [2] and [3].

3.4 Crossover and mutation

As stated, at crossover the individual members of a population are recombined to form members of the new population. Exactly how to recombine the population members is a matter of no small importance. A number of schemes exist to determine exactly which strings should be combined and in what way they should

be combined or spliced together. In a simple genetic algorithm, it is mostly the fittest individuals of a given population that are used to produce the next population, although the worst individuals also have a small chance. Presented here are some popular techniques for governing this process. In the presented techniques it is important that the size of the initial population is an even number.

Roulette Wheel Selection In Roulette selection, from an initial population of n individuals, a new population, also of n individuals, is selected by performing a linear search as through a roulette wheel with slots weighted in proportion to each individual's fitness. That is, the greater the fitness of an individual, the greater the probability that individual will be selected for crossover. In this way good individuals, that is, those who have better fitness, will probably be selected more than once in a generation and poor ones may not be selected at all.

Tournament Selection Randomly select $p < n$ individuals out of a population of n individuals and having evaluated their fitness, determine the individual x_1 with the greatest fitness. Repeat this process for another selection of p individuals yielding x_2 with the best fitness. These two individuals are now used to produce children through crossover and mutation. With a full replacement policy these two children replace their parents in the next generation. This procedure is repeated until the full population is obtained for the next generation. It should be noted that an excessive amount of computation time will be engendered if the value of p is too large.

Elitism Rather than a completely different technique, this elitist model is more of a variation that can be added to an existing selection technique such as either Tournament or Roulette selection. In this variation it is ensured that the best individual from generation $P(t)$ is brought into generation $P(t+1)$. This is accomplished in one of two ways:

1. Depending upon the size of the population a number of copies of the fittest string, usually two but possibly more, are placed directly into the next generation. The remainder of the population is produced in the usual manner.

2. Alternatively, the next generation may be formed as per the established technique, the fitness values of the new strings calculated, and the weakest individual replaced with the fittest individual from the previous population.

Once the parents are chosen using one of the above techniques, or any other variation, the actual splicing together can take place. As with the selection routines, there are a number of ways in which the splicing can transpire. The most common technique is simply to use a one point crossover operator. In this technique a random mutual crossover point is selected for both parents, and the 'tails' of the two parents are interchanged. As an example of this process consider two parent bit strings x_1 and x_2 shown below:

$$x_1 \quad 10000010$$

$$x_2 \quad 11110001$$

Choosing a crossover point following the fourth bit we have

$$\begin{array}{lll} x_1 & 1000 & 0010 \\ x_2 & 1111 & 0001 \end{array}$$

and the two children produced, call them c_1 and c_2 are:

$$c_1 \quad 10000001$$

$$c_2 \quad 11110010$$

A variation on this simple one point crossover technique would be a two point crossover technique. As can be imagined this involves swapping a section of each parent with the other rather than one end. An example is not provided here as it is mentioned only as an example of variations which may occur at this stage of the crossover process. An alteration to the crossover process is necessary when dealing with the strings of evolutionary algorithms as will be seen later. Crossover may not necessarily be applied to all pairs within the mating with the probability of it occurring being specified by the user. Typically the probability lies within the range of 0.6 to 1.0.

Following the crossover process, a *mutation* operator is applied to the individuals in the new generation. According to a user specified probability, each bit of each individual is changed if it mutates. For each individual bit within a string there exists a probability p_m that it will mutate and a probability $1 - p_m$ that it will

survive the mutation operation. In the case of a simple binary bit stream a 1 is mutated to a 0 and vise versa. In the case of a string composed of real values as used in the evolutionary algorithm approach, the mutation operation becomes more complex and is covered at the end of this chapter in the section on evolutionary algorithms.

The mutation operator produces periodic variations in the population which can have the effect of 'throwing' the string into a new region of the parameter space. This can be important for maintaining the robustness of the search algorithm.

3.5 Genetic algorithms and optimisation

It is worth noting at this point some of the values of the genetic algorithm as an optimisation search technique.

- The problem of single point searches getting stuck at local maxima is avoided due to the fact that the genetic algorithm utilises more than one point in the search for optimal values. Indeed, the sampling function is in effect being optimised at a number of different places simultaneously.

- At crossover, the individuals with the better fitness values stand a much better chance of being in the 'mating pool'. This means that, in the mating process, there is a reasonable chance that the best qualities of two strings will be combined to produce an individual that has the advantages of both.

- The advantages of the mutation operator, which is intended to introduce periodic variations into the population that can move a string into a new area of the parameter space, have been debated in the literature on genetic algorithms.

- The genetic algorithm approach may be applied to any objective function and doesn't require any additional information such as derivative equations.

- Although random choice is used to guide the search, it is a selective approach to randomisation that selects and attempts to improve good individuals in a population.

3.6 Similarity templates

Although not pertinent to the content of later chapters, similarity templates rate a mention. Similarity templates provide a framework in which to compare how a string is representative of other string classes with similarities at certain string positions. They are important to the mathematical foundation of the science of genetic algorithms and deal with the determination of why particular strings all have high fitness values. These templates are also referred to as schemata. They are mention here as a point of interest and a detailed discussion on similarity templates can be found in [11].

3.7 Encoding options

The next question to be faced is how to encode the particular parameter or parameters into strings for use in the genetic or evolutionary algorithm. The exact method will vary depending, obviously, upon the problem at hand and the exact method used in the next two chapters for the encoding of the parameter space will be explained at that stage. As a general overview, the concepts involved in encoding the parameters are examined below in moderate detail.

Two questions present themselves immediately:

1. How do you represent a number range say $[0 \ldots 63]$ or indeed $[-2.57 \ldots +3.38]$?

2. How do you represent more than one parameter?

A simple answer to question two is that one string is encoded for each parameter and the strings are simply concatenated together, but this is not the only representation possible. For example with two variables x and y, let them be represented by the bit strings u_1 and u_2. For x:

$$u_1 = a_{11}a_{12}a_{13}a_{14}\cdots a_{1m_1},$$

and for y

$$u_2 = a_{21}a_{22}a_{23}a_{24}\cdots a_{2m_2},$$

with u_1 and u_2 not necessarily the same length. These two parameters are then represented by the one string:

$$u_1 \oplus u_2 = a_{11}a_{12}\cdots a_{1m_1}a_{21}a_{22}\cdots a_{2m_2},$$

where $\oplus$ is the concatenation operator and the length of the concatenated string is $m_1 + m_2$ long.

This concept of concatenating individual parameter strings together to form an individual is, of course, extensible to systems of more than two parameters.

In answer to question one, let us first examine the integer range $[0 \ldots 63]$. In using a binary bit string to represent this span of numbers, the answer is self evident, because a 6-bit binary representation will represent every integer from 0 to 63. In the case of the second sample number range, $[-2.57 \ldots + 3.38]$, representation of the interval may be achieved by the use of a linear map. For example, to span the interval $[x_{min}, x_{max}]$, we would map the positive integers in the range $[0, 2^m - 1]$ by using the transformation:

$$x = au + b,$$

where $x \in [x_{min}, x_{max}]$ and $u \in [0, 2^m - 1]$ and the constants a and b are determined from the co-ordinate points $(0, x_{min})$ and $(2^m - 1, x_{max})$.

The accuracy of the this transformation is dependent on the distance between points and can be calculated by

$$\frac{(x_{max} - x_{min})}{(2^m - 1)}.$$

According to Goldberg [11], two principles govern the selection of a coding scheme for the parameter(s) of a genetic algorithm. These are:

1. "The user should select a coding so that short, low-order schemata are relevant to the underlying problem and relatively unrelated to schemata over other fixed positions."

   That is, that when crossover is performed, it is important that short, low order, schema are the objects that are switched.

2. "The user should select the smallest alphabet that permits a natural expression of the problem."

   This is important to ensure that the chance of similarities in different strings is increased.

Moving away from the binary encoding scheme to one utilising real numbers is an important step in overcoming the limitation inherent to the binary encoding system. That being the problem of the excessive size of strings when encoding for multidimensional, high precision numerical problems. This is dealt with in the final section of this chapter which explains the concepts and workings of evolutionary algorithms.

3.8 Objective functions

When tackling an optimisation problem with a genetic algorithm, it is not necessarily true that the function to be maximised and the fitness function are the same. This is due to the fact that the probability calculated for use in the selection process must be a positive number, if using Roulette selection.

Let u be a fitness function that takes negative values for some x in it's defined domain that we wish to maximise. A new function f may be formed such that only positive values are taken:

$$f(x) = \begin{cases} u(x) + c, & u(x) + c > 0, \\ 0, & \text{otherwise.} \end{cases}$$

where c is a positive constant which may be determined by mathematical analysis or by choice of a sufficiently large value. A more computationally intensive, but more certain method for the choice of c is to perform a sort of the population at every generation, including generation zero, and set $c = \mid u_{min} \mid$ where u_{min} is the minimum fitness value found.

Considering the problem of minimisation of u rather than maximisation the classical technique of replacing u with $-u$ will not work in the context of the genetic algorithm with Roulette selection. A fitness function that may be considered in this case is:

$$f(x) = \begin{cases} -u(x) + d, & u(x) + d > 0 \\ 0, & \text{otherwise,} \end{cases}$$

where the positive constant d is chosen, for example, as $d = \mid u_{max} \mid$, with u_{max} being the maximum fitness value in the population at each generation.

3.9 Fitness scaling

When using genetic algorithms a common problem experienced is premature convergence of the algorithm to a local maximum, that is, when a population of strings converge to the same string.

Small populations are particularly problematic in the following regard. When an initial population is formed, the existence of some, comparatively, very fit individuals will result in these individuals quickly multiplying and outnumbering any alternative strings.

Also, at the later stages of a run, although there may still be a great deal of diversity in a population, the increase in the average fitness of the population may stay close to the maximum fitness of the best individual. This results in the average and best strings in the population propagating in approximately even numbers.

These two problems are caused by the comparison of relative fitness values between the best individual and the remainder of the population

A simple linear re-scaling of the raw fitness function f can be implemented to solve this problem:

$$f' = af + b,$$

where f' is the scaled fitness and a and b are constants.

It is desired that the average fitness of both f and f' be the same. To accomplish this we choose:

$$f'_{max} = C_{mult} f_{avg}$$

so that f'_{max} is a constant multiple of the average value of f'. For small population sizes in the region of 50 to 100 individuals it has been found that values of 1.2 to 2 for C_{mult} work well. In order to prevent rescaling producing negative fitness values, which can occur when the average fitness value is close to the maximum, C_{mult} may be chosen such that $f'_{min} = 0$ when $f(x) = f_{min}$.

The rescaling process is dynamic in nature and is implemented for each generation only when needed thus preventing the early domination of extraordinary individuals in the population.

3.10 Evolutionary algorithms

There is a basic conceptual difference between what is thought of as a traditional genetic algorithm and what has now come to be termed an evolutionary algorithm. Genetic algorithms, as we have seen, operate on binary strings, requiring an alteration of the problem which they are being used to address in the form of the encoding of the parameter space for potential solutions. Evolutionary algorithms however, modify the chromosome representation of the problem being addressed

whilst still applying appropriate 'genetic' operators leaving the original problem unchanged. To use the quote coined in Michalewicz [19]:

> "If the mountain will not come to Mohammed, then Mohammed will go to the mountain."

So, unlike genetic algorithms which transform the problem into a form appropriate for solution, evolutionary algorithms are themselves a transformed version of the genetic algorithm altered to suit the problem. This approach is termed *Evolutionary Programming*, but the names evolutionary program and evolutionary algorithm have grown to be synonymous. The genesis of the field of evolutionary programming really represents a move away from the realm of biology into a distinct and unique field of algorithmic science. In terms of nomenclature, evolutionary algorithms generally no longer use the biological terminology associated with structures as did genetic algorithms. This is because the structures now comprising evolutionary algorithms bear little resemblance to the biological structures from whence they sprang. For example chromosomes and genes are now exclusively referred to as *strings* and *elements* respectively.

When dealing with the actual mechanics of evolutionary algorithms as compared to their genetic progenitors, much remains unchanged. Listed below are the major points of relevance.

3.10.1 Parameter encoding

The solution to a problem is represented by a solution vector. Thus, in encoding the strings for an evolutionary algorithm, each string is a vector of floating point numbers representing a possible solution to the problem. Also, each floating point element is selected to lie within a certain domain and the crossover and mutation operators are carefully devised to preserve these limitations.

This floating point representation generally possesses a higher order of precision than the binary representation used in genetic algorithms and is capable of symbolising quite large domains without the loss precision that would occur with binary representations.

3.10.2 Crossover

In noting the differences with the crossover technique it is important to point out that the selection techniques prior to the actual occurrence of crossover are completely unchanged. Roulette and tournament selection, along with the implementation of elitism, are still applicable and widely used with tournament selection now the preferred technique.

When dealing with the floating point numbers that now comprise the strings of evolutionary algorithms it is important to note modifications that must be taken into account due to the altered nature of the strings. Due to the parameterisation often used in composing strings, individual strings now represent groupings of vectors.

Thus, while simple one-point, or even multi-point, crossover is still applicable, it must be ensured that the split point only occurs between the vector's groups less a vector be created that lies outside the allowed solution space.

A new type of crossover mechanism is also available, that being *arithmetic crossover.* It is defined as a linear combination of two vectors. For example, let $\underset{\sim}{v}_1^t$ and $\underset{\sim}{v}_2^t$ be vectors contained within a mating pair of strings. The children produced would possess the vectors

$$\underset{\sim}{v}_1^{t+1} = a\underset{\sim}{v}_2^t + (1-a)\underset{\sim}{v}_1^t,$$

and

$$\underset{\sim}{v}_2^{t+1} = a\underset{\sim}{v}_1^t + (1-a)\underset{\sim}{v}_2^t,$$

where a is either a constant or a random variable with $a \in [0, 1]$. In the case where a is a constant, the process is referred to as *uniform arithmetic crossover.* Where a is a variable, its value may be chosen appropriate to the age of the population or just randomly, and it is referred to as *non-uniform arithmetic crossover.* Variations on the theme of arithmetic crossover also exist in that the operation may be applied to the whole string of vectors or just selected vectors within the strings.

3.10.3 Mutation

The mutation operators that exist for the space of real numbers signify the greatest departure from what we are already familiar with from the binary representation of genetic algorithms. They are generally divided into two classes, as with the crossover

operators discussed above, being uniform and non-uniform mutation. Once again, the distinguishing factor between the two types is usually whether or not their action is dependent upon the population age.

Uniform operators are similar to what we are already familiar with in that each element of a string has an equally likely chance of being mutated. The actual mutation that takes place alters the particular element to a random value bounded by the domain of the corresponding parameter space.

Non-uniform operators are often utilised in fine tuning the search. The means by which this is accomplished is best explained mathematically. Let $\underset{\sim 1}{\mathrm{v}}^t = (e_1, e_2, \ldots, e_n)$ be a string composed of n elements. If element k is chosen for mutation where $l_k \leq e_k \leq u_k$ defines the range of element k. The mutated element $e'_k = e_k \pm \Delta(t, u_k - l_k)$ where the probability of addition or subtraction is 0.5. The function $\Delta(t, y)$ is chosen such that the value returned by it has a higher probability of being closer to zero, the larger the value of t. This effectively helps to restricts the search from uniform to very local as time or the generation number increases.

As an alternative to only mutating one element of string, the non-uniform mutation operator is sometimes applied to every element of a string.

3.11 Summary

This chapter on genetic and evolutionary algorithms has now presented the reader with enough information and background to enable an understanding of the applica-

tions of genetic and evolutionary algorithms as applied to the optimisation problems associated with water flow in soils as presented in the next two chapters. In conjunction with the theory of water flow in soil presented in Chapter one, and the mathematical modelling of such in a cropped soil as explored by Terry Janz and examined in Chapter two, this concludes the background material necessary to discuss the remaining research.

Chapter 4

Application of genetic algorithms to the model for water flow in soil

4.1 Introduction

The work of Terry Janz in modelling water flow through soils covered in Chapter 2 is merged with the material presented in Chapter 3 in order to find an optimal irrigation strategy. A genetic algorithm is defined to learn the irrigation cycle for the simulated water flow in cropped soils. It is shown that genetic learning provides an appropriate method for defining irrigation on and irrigation off switching to maintain a desired moisture content at predetermined depths in the soil.

An irrigation cycle traditionally consists of a brief period of infiltration followed by a period of evapotranspiration. During the drying period, the moisture content of

the soil is monitored to ascertain the appropriate time and quantity of the next irrigation event. By utilising the research of Terry Janz, soil water content profiles with respect to a growing root are determined numerically by computer simulation. By the unique use of a genetic algorithm and appropriate choice of cost and fitness functions, the decision of whether or not to irrigate is handled by the computer.

4.2 Water flow model

For the benefit of the reader, a summary of the relevant concepts and equations covered in Chapter 2 are reproduced in the next few sections. We have already seen the classic Richards' flow equation for one dimensional flow under gravity in cropped soils which is expressed as:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}[D(\theta)\frac{\partial \theta}{\partial z} - K(\theta)] - S(z),$$

where θ is the volumetric water content, z is the vertical distance positive downwards, D is the Diffusivity of the soil, K is the Conductivity of the soil, and S is a sink term accounting for water loss with depth.

The sink term, incorporating the concept of the evaporation front developed by Stonier and Janz and discussed in Chapter 2 is expressed as:

$$S = \frac{-1.6T}{(v-e)^2}z + \frac{(1.8v - 0.2e)T}{(v-e)^2},$$

where S is the moisture extraction rate per unit volume of soil, v is the vertical length of the root system, T is the transpiration rate per unit area of soil surface, and where e is the evaporation front, see [29].

4.3 Numerical model

Based on the following assumptions:

- That the initial water profile within the soil and at the lower boundary is known,
- That evaporation at the upper boundary is represented by a non-linear flux condition,
- That, for an infiltration event, the upper surface of the soil is held instantly at saturated water content,

it is possible to express all equations in finite difference form having all finite differences written half way between known and unknown time steps at any depth about $z_i, t_{j+1/2}$. The finite difference values are resolved by use of the Crank Nicolson method.

The boundary conditions give rise to the formation of two sets of equations:

1. Where evaporation is occurring at the surface and a derivative boundary condition is set for this.

2. Where infiltration is taking place and the boundary condition is a constant.

The set of equations across all depth steps x_i, formed at each time step t_j, are tridiagonal in form. The finite difference equations, which have been reproduced below, are solved using the Thomas' Algorithm where the depth steps are defined as $z_i = (i - 1/2)\triangle z$ with $i = 1, 2, \ldots, N$.

For $2 \leq i \leq (N-1)$:

$$\begin{aligned} & D_{i-1/2}^{j+1/2}\theta_{i-1}^{j+1} - (D_{i-1/2}^{j+1/2} + D_{i+1/2}^{j+1/2} + A)\theta_i^{j+1} + D_{i+1/2}^{j+1/2}\theta_{i+1}^{j+1} = \\ & \quad -D_{i-1/2}^{j+1/2}\theta_{i-1}^{j} + (D_{i-1/2}^{j+1/2} + D_{i+1/2}^{j+1/2} - A)\theta_i^{j} - D_{i+1/2}^{j+1/2}\theta_{i+1}^{j} \\ & \quad +2\triangle z(K_{i+1/2}^{j+1/2} - K_{i-1/2}^{j+1/2}) + 2\triangle z^2 S_i^{j+1/2}, \end{aligned} \tag{4.1}$$

where $A = 2\triangle z^2/\triangle t$ and the finite discrete form of the sink term is given by:

$$S_i^j = \frac{-1.6T_j}{(v(j\triangle t) - e_j)^2}(i - 1/2)\triangle z + \frac{(1.8v(j\triangle t) - 0.2e_j)}{(v(j\triangle t) - e_j)^2}T_j. \tag{4.2}$$

For $i = 1$ with an Evaporation event:

$$\begin{aligned} & -(D_{3/2}^{j+1/2} + A)\theta_1^{j+1} + D_{3/2}^{j+1/2}\theta_2^{j+1} = \\ & \quad -D_{1/2}^{j+1/2}\theta_0^{j} + (D_{1/2}^{j+1/2} + D_{3/2}^{j+1/2} - A)\theta_1^{j} - D_{3/2}^{j+1/2}\theta_2^{j} \\ & \quad -D_{1/2}^{j+1/2}\triangle z(q_{1/2}^{j+1} - K_{1/2}^{j+1})/D_{1/2}^{j+1} + 2\triangle z(K_{3/2}^{j+1/2} - K_{1/2}^{j+1/2}) \\ & \quad + 2\triangle z^2 S_1^{j+1/2}, \end{aligned} \tag{4.3}$$

where θ_0 at the jth time step is as follows:

$$\theta_0^j = \theta_1^j + \frac{\triangle z}{D(\theta_{1/2}^j)}(q_{1/2}^j - K(\theta_{1/2}^j)). \tag{4.4}$$

For $i = 1$ with an Infiltration event:

$$-(2D_{1/2}^{j+1/2} + D_{3/2}^{j+1/2} + A)\theta_1^{j+1} + D_{3/2}^{j+1/2}\theta_2^{j+1} =$$
$$(2D_{1/2}^{j+1/2}\theta_0^j + D_{3/2}^{j+1/2} - A)\theta_1^j - D_{3/2}^{j+1/2}\theta_2^j$$
$$2\triangle z(K_{3/2}^{j+1/2} - K_{1/2}^{j+1/2}) - 4D_{1/2}^{j+1/2}\theta_s + 2\triangle z^2 S_1^{j+1/2}. \quad (4.5)$$

For $i = N$:

$$D_{N+1/2}^{j+1/2}\theta_{N-1/2}^{j+1} - (D_{N-1/2}^{j+1/2} + 2D_{N+1/2}^{j+1/2} + A)\theta_N^{j+1} =$$
$$-D_{N-1/2}^{j+1/2}\theta_{N-1}^j + (D_{N-1/2}^{j+1/2} + D_{N+1/2}^{j+1/2} - A)\theta_N^j - D_{N+1/2}^{j+1/2}\theta_{N+1}^j$$
$$-2D_{N+1/2}^{j+1/2}\theta_{N+1/2}^{j+1} + 2\triangle z(K_{N+1/2}^{j+1/2} - K_{N-1/2}^{j+1/2}) + 2\triangle z^2 S_N^{j+1/2}, \quad (4.6)$$

where $\theta_{N+1}^j = 2\theta_{N+1/2}^j - \theta_N^j$.

Root depth was set at 120cm, that of a mature plant, and the initial water content profile of the soil was randomly set. The diffusivity D and conductivity K parameters were calculated empirically cf. [4] for the chosen silty clay soil type with evapotranspiration data also corresponding to the soil type. The θ_C and θ_S parameters representing critical moisture content and saturated moisture content respectively were chosen as $\theta_C = 0.21$ and $\theta_S = 0.5$.

4.4 Switching between drying and wetting

As covered previously, there is an issue with the change from an Infiltration event to an Evaporation event due to the surface condition and calculation of values nec-

essary for the solution of the tridiagonal system of equations. For a more detailed explanation of this problem and its solution, please refer to Chapter 2, Section 8.

4.5 Genetic learning of the irrigation schedule.

As stated in [28], "Consideration of movement of water in the soil profile as illustrated in this simulation leads to an idea that it is not so much the volume of water that is important but rather where the water is with respect to the roots." Thus, the goal of simulating an irrigation schedule is to maintain the water content in the soil at a desired value at a chosen depth within the soil. It is at this point that we seek to apply our knowledge of genetic algorithms to the problem.

As stated earlier, at it's most basic level, the logical operation of a genetic algorithm is:

```
begin
  t = 0
  Create random P(t)
  until (finished) repeat
  begin
    Evaluate Fitness of Strings in P(t)
    t = t+1
    Create P(t+1) from P(t)
  end
```

```
end
```

In parameterising this particular problem, we have chosen that each individual $\underset{\sim}{x}$ in the population P at time t is a linear array of binary bits, with each bit representing the action taken on a particular day of the simulation. A bit value of 0 represents a day of evaporation and a 1 represents a day of infiltration. Therefore, given a 120 day time period over which the simulation is run, each string $\underset{\sim}{x}$ consisting of 120 bits represents a potential solution to the problem.

As is the case with any genetic algorithm, a fitness function is required to determine which are the better strings to be used for spawning the next generation. In this case the fitness f_k of the kth individual is determined by passing the individual to the water flow simulation algorithm for processing and having the algorithm return a fitness value for the string. The fitness function chosen is a variance or least squares measure:

$$f_k = \sum_{j=1}^{120} \left(\frac{\theta_j(d) - \theta_d}{\theta_d} \right)^2 .$$

where $\theta_j(d)$ is the moisture content on the jth day at depth $d = 65$ cm and θ_d is the desired moisture content level which was chosen as 0.3. It is this measure of variance about the desired moisture content level that we seek to minimise.

Representing the entire simulation in pseudo code as it stands as a subsystem of the genetic algorithm makes it easier to comprehend the exact operation. Therefore, each time the fitness evaluation is to take place within the genetic algorithm, the following process transpires:

- Present irrigation schedule $\underset{\sim}{x}^j$.
- Initialise and perform boundary setup.
- Repeat for $k = 1$ to $N = 120$.
  - If $\underset{\sim}{x}^j(k) = 1$, implement the infiltration boundary conditions.
  - If $\underset{\sim}{x}^j(k) = 0$, and $\underset{\sim}{x}^j(k-1) = 1$, then solve for water contents at all depths by using first estimate thetas (not using Taylor's approximation), provisionally update water contents, then use these second estimate water contents.
  - If $\underset{\sim}{x}^j(k) = 0$, and $\underset{\sim}{x}^j(k-1) = 0$, then use Taylor's approximation to find projected forward thetas.
  - Use estimates to solve for water content profile θ.
  - Upgrade water contents θ_k.
  - Calculate $((\theta_j(d) - \theta_d)/\theta_d)^2$.
- Calculate jth individual fitness f_j.

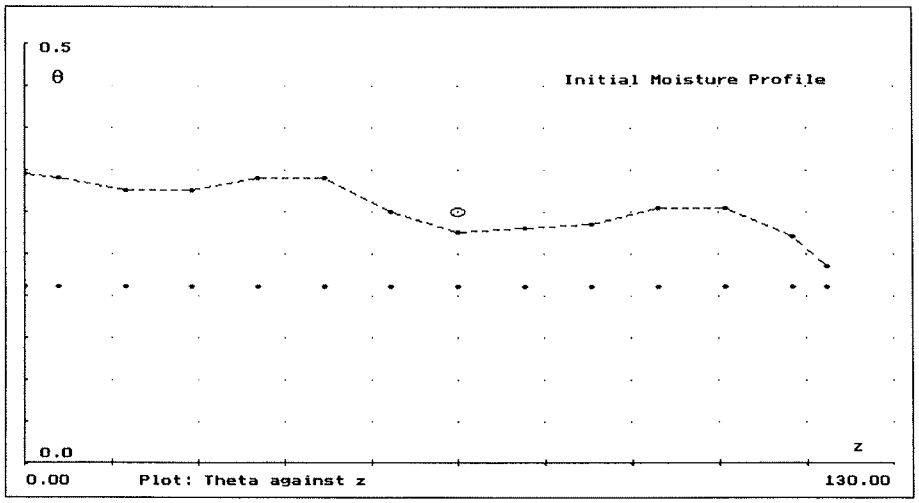
Once in possession of the fitness values for each individual in the population it is now possible to perform the crossover and mutation processes to obtain the next generation $P(t+1)$ of individuals. In this particular case, proportional selection is used to determine which pairs of parents will be chosen for mating. A standard single point crossover with random swap point and probability of 0.6 combined with the standard binary mutation operator, utilizing a probability of 0.001, was used in

the generation of the next generation. The value of 0.001 for mutation was chosen after some experimentation. It was found that the 0.001 value resulted in better convergence than higher values of 0.1 and 0.01.

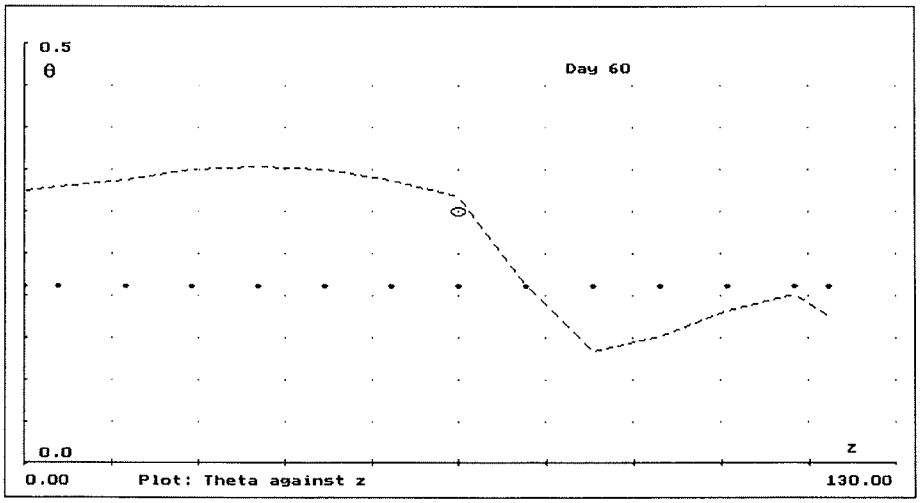
Breaking away from the standard crossover and mutation approach, one concession was made in the generation of new individuals. The field surface data available for use in the simulation did not exceed 18 consecutive days of evaporation. Hence, an addition was made to the algorithm for crossover which, following the mutation process, scanned the newly formed string for a sequence of 19 consecutive 0's and, if found, set the 19th zero to a one. This is certainly not the only way to ensure no more than 18 days of continuous evaporation events, but is simple and effective. Had more field data been available for continuous evaporation, it would of course have been preferable not to have this limitation in place.

The population size was set at 40 individuals and elitism and prescaling, as covered in Chapter 3, were used to improve convergence.

In Figures, 4.1 and 4.2, show the water content profiles within the soil at days 1, 60, 90 and 120. The point depicted in the centre of each diagram by a circle represents the chosen reference point of $d = 65$cm and $\theta = 0.3$. This, as the reader will recall, is the point θ_d from the chosen fitness function.

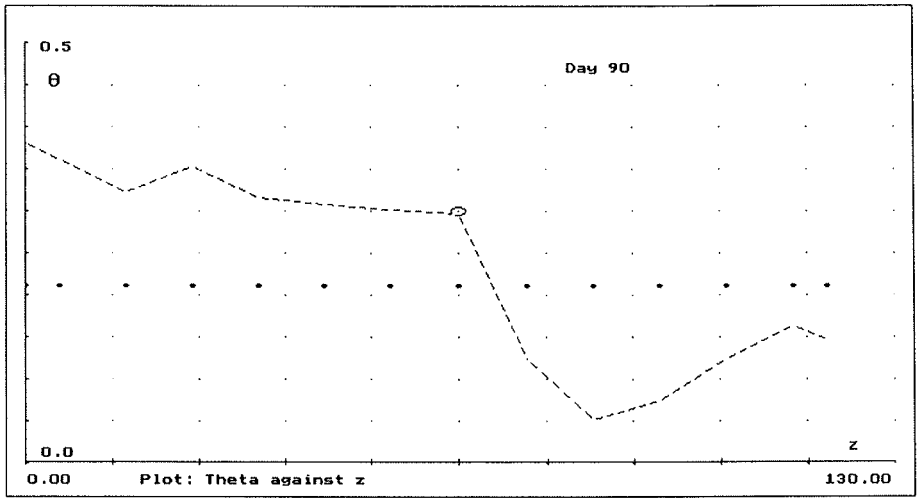


(a)

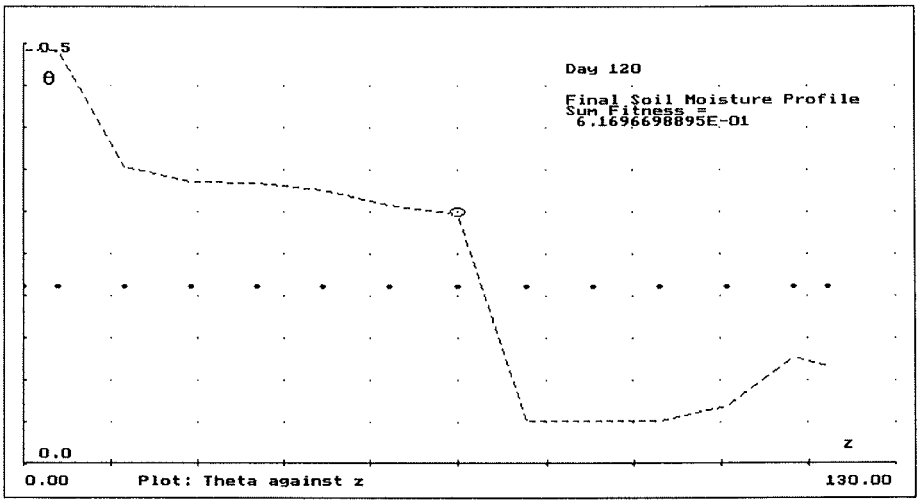


(b)

Figure 4.1: (a) Initial Moisture Profile (b) Moisture Profile at day 60 - One reference point



(a)



(b)

Figure 4.2: (a) Moisture Content Profile at day 90 (b) Moisture Content Profile at day 120 - One reference point

The irrigation cycle depicted in these illustrations has the following binary representation:

$$1000110110110100010100001010101100010101001101001001100101001$$
$$0011000100001011010100010001011010100100100100111000010100 1.$$

It can be noted that the irrigation cycle arrived at by the genetic algorithm seeks to maintain sufficient moisture content at the upper levels of the soil to precisely maintain the moisture content at depth d. Also obvious from the results is the distinct diminishment of the soil water content below depth d. This dropping off of the moisture content within the soil below *wilting point*[1], θ_c, represented by the horizontal dotted line on the diagrams, signifies that for a substantial period of time the end of the root zone does not receive any water. This is obviously not an acceptable solution in a real world situation and an alternative fitness function must be established in order to overcome this limitation. As a proof of concept though, the genetic algorithm has proven a most effective means of finding an optimal irrigation schedule in this test case. By taking an average of 10 simulation runs, utilizing different random seeds to generate the genetic algorithm initial populations, Figure 4.3 has been generated to show convergence of the genetic algorithm and graphs the minimum, average and maximum fitness values at each generation.

[1]Wilting point is defined as the moisture content value below which it is impossible for a plant to extract water from the soil.

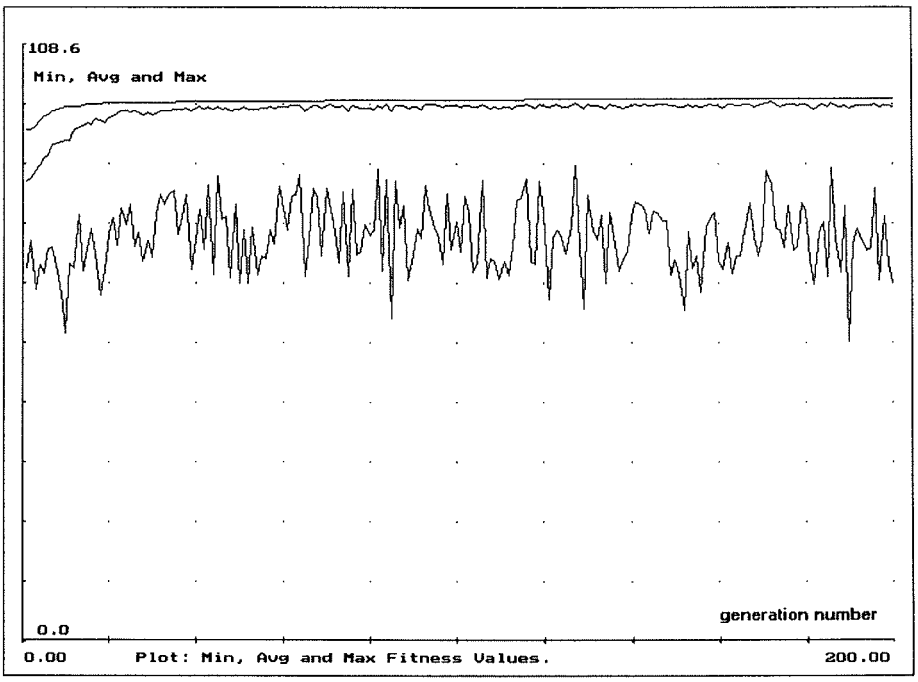


Figure 4.3: Convergence of the genetic algorithm

It is observed that the genetic algorithm converges strongly within 60 generations. The wide variation seen in the fluctuating minimum values is explained by the sensitivity of the system to minor changes. A finer granularity in the enforcement of infiltration and evaporation events would reduce this sensitivity.

In line with the comment made above regarding the moisture content drop off below depth d a new fitness function was developed. This new fitness function is the weighted sum of relative differences at specified depths within the soil. The expected outcome of this new fitness function is that an appropriate volume of water will be maintained throughout the entire root zone.

This new fitness function can be expressed as:

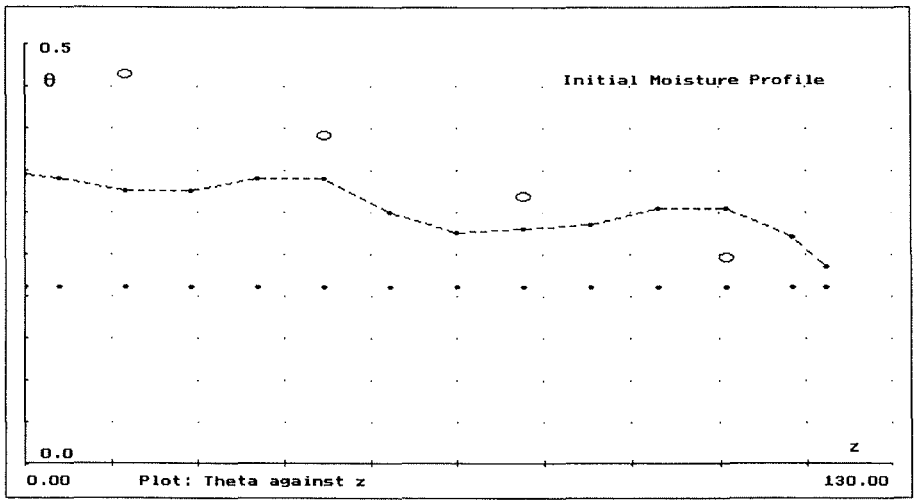
$$f_k = \sum_{j=1}^{120} \left(\frac{(\theta_j(d_1) - \theta_{d_1})^2 + (\theta_j(d_2) - \theta_{d_2})^2 + (\theta_j(d_3) - \theta_{d_3})^2 + (\theta_j(d_4) - \theta_{d_4})^2}{(\theta_{d_1} + \theta_{d_2} + \theta_{d_3} + \theta_{d_4})} \right)^2 .$$

Figures 4.4 and 4.5 display the water content profiles resulting from the computer simulation at days 1, 60, 90 and 120 as before with the weighted sum reference positions show by circles. These reference points are given in Table 4.1.

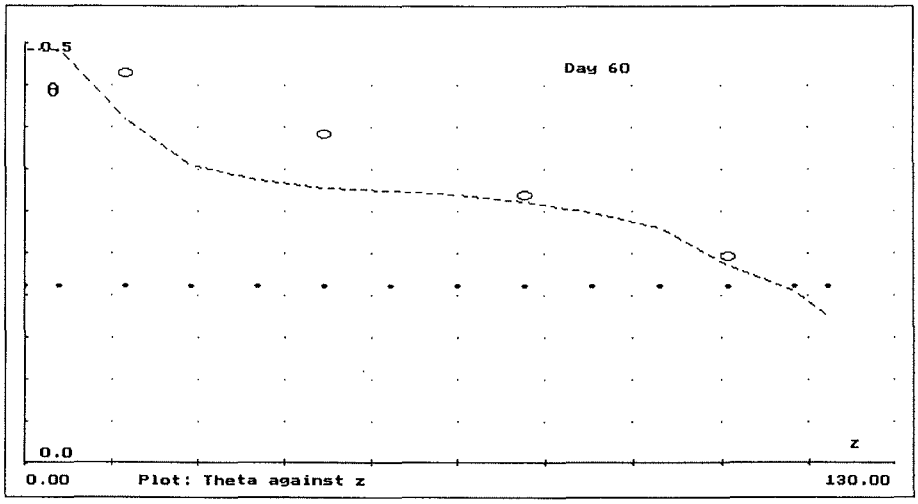
The choice of these particular values for the reference points correlates directly with the sink term of Molz and Remson [21].

depth (cm)	θ
15	0.46375
45	0.39125
75	0.31875
105	0.24625

Table 4.1: Fitness function reference points.

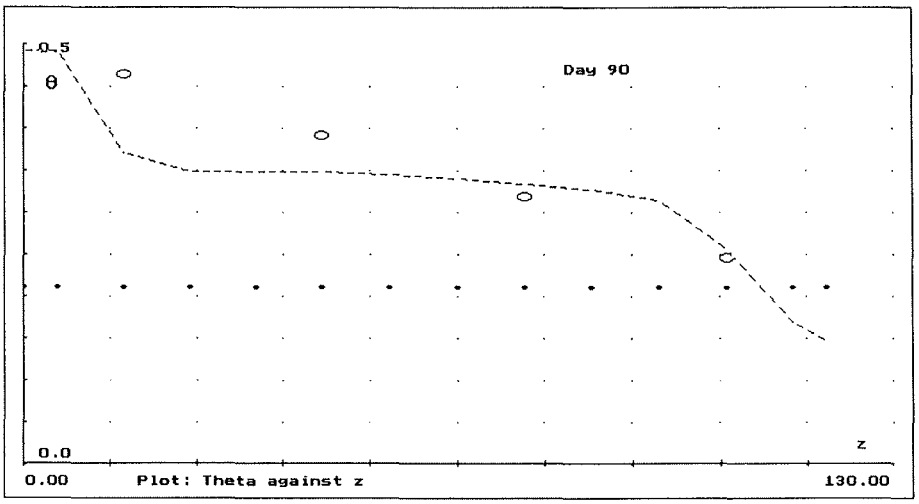


(a)

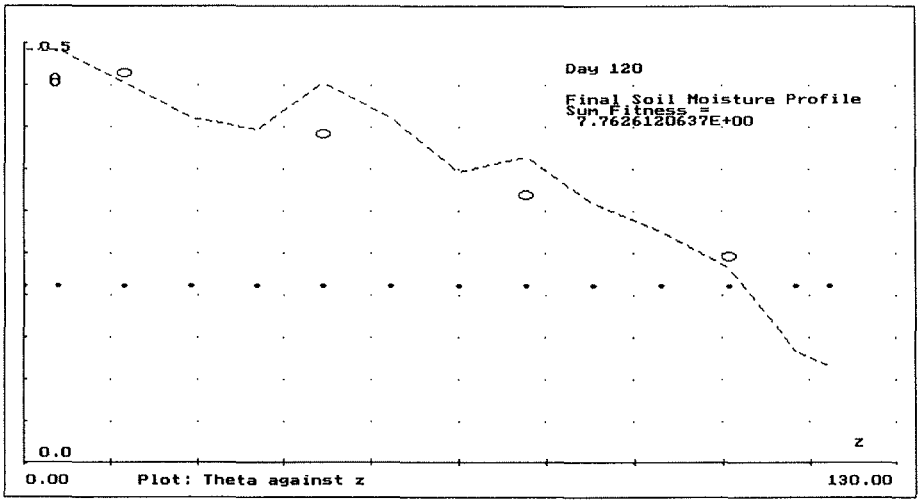


(b)

Figure 4.4: (a) Initial Moisture Profile (b) Moisture Profile at day 60 - Four reference points



(a)



(b)

Figure 4.5: (a) Moisture Content Profile at day 90 (b) Moisture Content Profile at day 120 - Four reference points

As can be seen from the diagrams, the water content profile maintained by the genetic algorithm is now more finely attuned to the requirements of the plant root. Although a sharp decline in the moisture content below θ_C past the deepest reference point in the soil is obvious, as with the last optimisation, this could be solved by the addition of another reference point at the end of the root zone.

The binary representation of this irrigation cycle is:

$$1110110100011000010011110000000111010010100100101100010001 01$$

$$1100001100000000111010001010011001000110100110101000110011 01.$$

When examining both irrigation schedules arrived at by the genetic algorithm it is apparent that while there are periods where day on day off switching occurs, these patterns tend to be infrequent. Rather, it appears as though the genetic algorithm has learned to perform extended drying periods as would be expected in a realistic problem.

Specified drying (or wetting) periods for a crop during its growth may be accomplished by appropriate modifications to the genetic algorithm operators ensuring viable individuals (irrigation schedules) are generated. The convergence of the fitness function selected in this second demonstration was, upon examination, found to be similar to that in Figure 4.3.

4.6 Summary

In this chapter research was conducted into the application of genetic algorithms to the learning of an irrigation schedule with the aim of maintaining moisture content around desired values at given depths. The simulation of the water flow in the soil was one dimensional with water extraction due to a single mature plant root. In the next chapter we examine the application of evolutionary algorithms to the optimal control problem arising from semi-discretisation of a linear parabolic tracking problem with boundary control.

Chapter 5

Optimal boundary control of a tracking problem using evolutionary algorithms

5.1 Introduction

Intelligent control systems, that is systems that self-organise, learn and adapt their control laws or control rules in response to changes in the environment reside in the now well established area of research known as intelligent system research. Two problems present themselves in relation to this approach. These are:

- How do you, for systems that are not precisely described, in a systematic and logical manner, describe the systems and compute control of the evolution?

- How do you optimise the modelling and control of the system?

The forms of the optimisation may be many, ranging from simple optimisation of some objective criteria such as time or fuel consumption to optimisation of the mathematical model itself. Problems may be multi-objective in nature possessing nonlinear constraints or even with constraints not lying within the search space of classical optimal control. The use of evolutionary algorithms to mimic the natural learning of humans through experience is well established in terms of both theory and application. See [3] and [8] for a survey of the evolutionary algorithm and its application to real world problems involving optimisation.

The motivation for this investigation resides in the work performed in the previous chapter. The mathematical modelling of water flow in cropped soils is complex and the solution to optimisation problems in this area by classical numerical methods is difficult due to the requirement of maintaining numerical accuracy at the upper boundary where infiltration and evapotranspiration occur when switching between irrigation on and irrigation off.

The aim of the research in this chapter is to investigate the feasibility of applying an evolutionary algorithm directly, to solve the boundary control problem posed by applying irrigation at the surface boundary and using the Richards' flow equation to model moisture content. This is not a trivial exercise as the problem posed is a nonlinear optimal control problem with differential and boundary constraints.

We begin by undertaking a study of a similar problem, namely, a linear parabolic tracking problem with boundary control for a distributed process whose governing state equation is a parabolic partial differential equation, cf Huntley [17]. It is simple in structure but has inherent computational difficulties that make it a good test example for our investigation.

In Huntley [17] a comparative study was made of five methods for calculating the optimal control function for a linear parabolic tracking problem with boundary control. Both open-loop methods based upon the variational equations and closed-loop methods, cf [5], via the Ricatti equation, were analysed for computational efficiency, accuracy, ease of programming and robustness.

This boundary control problem whose underlying state equations were parabolic partial differential equations, was first converted to a classical optimal control problem with ordinary differential state constraints through a method of semi-discretisation with respect to the state variable (the method of lines), cf [17]. In recent years it has been the practice to tackle these problems using fully discretised difference or finite element methods. Yet it has been suggested [1], that semi-discrete approaches obtained with current methods of solving **stiff** systems of ordinary differential equations might have advantages.

Semi-discretisation is employed to perform conversion of the partial differential equations to classical optimal control formulation. Problems arise with attempting solution of this new formulation such as the discretisation in time when solving the optimal control for a piecewise constant control strategy and the 'curse of dimension-

ality' when such discretisation is made in state variables. The next section discusses the formulation of the classical optimal control problem and the issues surrounding its solution, particularly those relating to the use of evolutionary algorithms.

In the next section we briefly discuss some of the current research issues relating to finding the solution of the classical optimal control problem using evolutionary algorithms.

5.2 Classical optimal control

The optimal control problem can be stated as:

Minimise the performance index

$$I[\underset{\sim}{x}(t_0), t_1] = \phi(\underset{\sim}{x}(t_1), t_1) + \int_{t_0}^{t_1} f_0(\underset{\sim}{x}(t), \underset{\sim}{u}(t), t)dt, \tag{5.1}$$

subject to the differential constraint

$$\frac{d\underset{\sim}{x}}{dt} = \mathbf{f}(\underset{\sim}{x}, \underset{\sim}{u}, t), \tag{5.2}$$

where the state vector $\underset{\sim}{x} \in R^n$ and the control vector $\underset{\sim}{u} \in R^m$ and where $\underset{\sim}{x}(t_0)$, the initial state, is given. The term $\phi(\underset{\sim}{x}(t_1), t_1)$ usually represents a cost or penalty associated with the state at the final time t_1.

This system may be subject to:

- combined state and control.
- equality and inequality constraints.

- integral constraints.

- interior point constraints.

Methods traditionally utilised for solving these types of problems require some form of conversion of the control function $\underset{\sim}{u}$ into an approximately equivalent representation that consists of a weighted combination/amalgamation of simpler functions, referred to as the collocation method. This changes the problem into one of discovering the optimal weights. Examples of these types of solutions are gradient descent methods and differential dynamic programming. As an alternative, the continuous control functions can be partitioned on $[t_0, t_1]$ into N intervals and the control functions replaced with simpler ones such as piecewise control for each u_i in the intervals $[t_i, t_{i+1}]$. Thus the objective becomes one of finding approximations in these local regions to optimise the performance integral.

The application of an evolutionary algorithm to such a problem requires the development of a representation of a control strategy over $[t_0, t_1]$ which can be easily manipulated by the operators that imitate and augment the genetic operators of crossover and mutation. One way in which this can be accomplished is, if the control is considered as a piece-wise constant approximation locally, to regard each real encoded string in the population as an array of N m-vectors. If the approximation is piece-wise linear each real encoded string may be considered as double m-vectors. If the time partition nodes are considered as variable, being able to move, then another row can be included to define the N-time partitions that can 'move' as the

evolutionary learning progresses. To ensure that the operators of the evolutionary algorithm, such as the arithmetic operator, produce strings that belong within the control space, it must be assumed that the control space itself is convex (i.e. the action of the operators is closed).

The formation of a uniform grid and choice of a an admissable control $\underset{\sim}{u}_i(t)$ in $[t_i, t_{i+1}]$ can be performed in order for a fixed time problem to form an initial population that will not converge prematurely. It is only necessary to test $\underset{\sim}{u}$ at the node points for feasibility since there is linearity of the controls and since the control space is assumed convex.

Mutation or perturbation of the string can be done at the local or global level. If t_i is perturbed then it must be restricted to $[t_{i-1}, t_{i+1}]$ and a check made to ensure that the new string is admissable. If it is found not to be then it must be suitably modified. If all strings have the same number of time partitions, then it is clear that a typical arithmetic crossover will, by convexity of the control space, guarantee a valid string in the population.

One-point crossover is more difficult to implement, cf [24].

As with any form of optimisation, constraints are a major problem. In optimal control this is particularly apparent where it may not be possible to explicitly test many constraints. Unlike simple optimisation problems where it is always possible to test if a given state satisfies all constraints before the objective function is evaluated, in optimal control problems this is not the case. Our aim is to solve a functional

optimisation problem for which the solution belongs in control function space and not in the traditional state space. The differential constraints must be integrated to enable testing of interior state constraints and any equality or inequality constraints involving state variables.

Apart from the semi-discretisation process which has already been mentioned, this formulation brings with it the 'curse of dimensionality' when the discretisation is made in the time domain. Achieving good convergence through the use of evolutionary algorithms requires new and enhanced genetic operators to deal with large numbers of real-valued parameters which have to be optimised in the presence of equality and inequality constraints cf [24] and [25]. Indeed, in [24] it was shown that, when compared to classical nonlinear constrained optimisation methods, direct application of evolutionary algorithms to determine accurate solutions of constrained continuous optimal control problems as defined was possible.

5.3 Boundary control of a distributed process

In this section an evolutionary algorithm is applied to the boundary control problem from [17]. The problem is now described. We first consider the state equation

$$\frac{\partial x}{\partial t} = \frac{\partial^2 x}{\partial y^2}, \quad 0 < t < T, \quad 0 < y < L,$$

subject to boundary conditions

$$x(y, 0) = 0,$$

$$\left.\begin{array}{rcll} \dfrac{\partial x}{\partial y} & = & \rho(x-u) & \text{on } y=0 \\ \dfrac{\partial x}{\partial y} & = & 0 & \text{on } y=L \end{array}\right\} \quad 0<t\leq T.$$

where ρ is a constant heat-transfer coefficient, x is temperature, y is depth and t is time.

The process of semi-discretisation may is accomplished by replacing

- $\dfrac{\partial^2 x}{\partial y^2}$ by its central difference approximation $\dfrac{x_{i+1}-2x_i+x_{i-1}}{h^2}$ of local accuracy $O(h^2)$, using $\Delta y = L/(N-1)$,
- $\left[\dfrac{\partial x}{\partial y}\right]_{y=0}$ by $(x_1 - x_{-1})/2h$, and $\left[\dfrac{\partial x}{\partial y}\right]_{y=L}$ by $(x_{N+1} - x_{N-1})/2h$.

Defining $\underset{\sim}{x} = [x_0, \cdots, x_N]^T$, the equations are discretised to

$$\dot{\underset{\sim}{x}} = A\underset{\sim}{x} + Bu, \tag{5.3}$$

where u is the scalar boundary control function, A is constant tri-diagonal matrix, and B is the vector $[2\rho/h\ 0, \cdots, 0]^T$.

The cost function to be minimised is defined by

$$J = \frac{1}{2}\int_0^L [x(y,T) - \eta(y,t)]^2\, dy + \frac{1}{2}\int_0^T ru^2(t)\, dt = J_1 + J_2,$$

where $\eta(y,T) = 0.2$ for $0 < y < L$. This is the target tracking function specified at all depths y for the dependent variable x. The first term J_1 defines a quadratic performance measure of the error from the target profile, and the second term J_2 defines a measure of cost in control.

Even though the definition of this problem is simple in nature, the solution of it is complex. This is due to the stability of the numerical approaches being a complicated function of ρ, r, the number of state steps (in y), and the number of time points (in t). A full discussion of the stability or instability of the numerical methods used to solve this problem can be found in [17]

In [17] the mathematical analysis of finding an open loop control of this system using necessary conditions arising from the application of the Minimum Principle is given. We note at this point that the technique used to do this derives necessary conditions for a solution of the problem and not sufficiency conditions.

For this set of parameters, described in [17], $L = 1$, $T = 0.4$, $r = 0.00001$, and $\rho = 1.0$, Table 5.1 presents a summary of the results of the two methods (Method 3 and Method 4) of the paper along with the results of the finite element and evolutionary algorithm approaches, for $NT = 100$ time steps.

In Method 3 the coupled state and costate equations derived from the Minimum Principle were appropriately numerically integrated with 100 time steps in the interval $[0, T]$. In Method 4, the coupled system of equations was integrated analytically. The control graphs obtained for these two methods are shown in Figure 5.1.

The finite element solution was arrived at by direct application to the boundary control of the partial differential equation using rectangular elements and bilinear shape functions with $N = 10$ and $NT = 100$. The explicit and open loop solution is presented in the table and provides another comparison base. Convergence of the

Station y	Method 3	Method 4	FE	EA
0.0	0.2000	0.1952	0.20003	0.19957
0.1	0.2000	0.2011	0.19995	0.20068
0.2	0.2002	0.1993	0.20010	0.19960
0.3	0.1998	0.2006	0.19999	0.19914
0.4	0.1997	0.1993	0.19964	0.19978
0.5	0.2001	0.2002	0.20003	0.20082
0.6	0.2006	0.2005	0.20054	0.20129
0.7	0.2005	0.2004	0.20053	0.20082
0.8	0.2000	0.2002	0.20001	0.19974
0.9	0.1994	0.1992	0.19938	0.19871
1.0	0.1992	0.1994	0.19911	0.19829

Table 5.1: Comparison of output results - $r = 0.00001$

finite element solution to the target profile is superb.

5.4 Solution by evolutionary algorithm

In the fourth column of the table can be seen the results obtained by the solution of the control problem using an evolutionary algorithm. A Runge-Kutta algorithm was used to integrate the state equations (5.3) with the time interval discretised

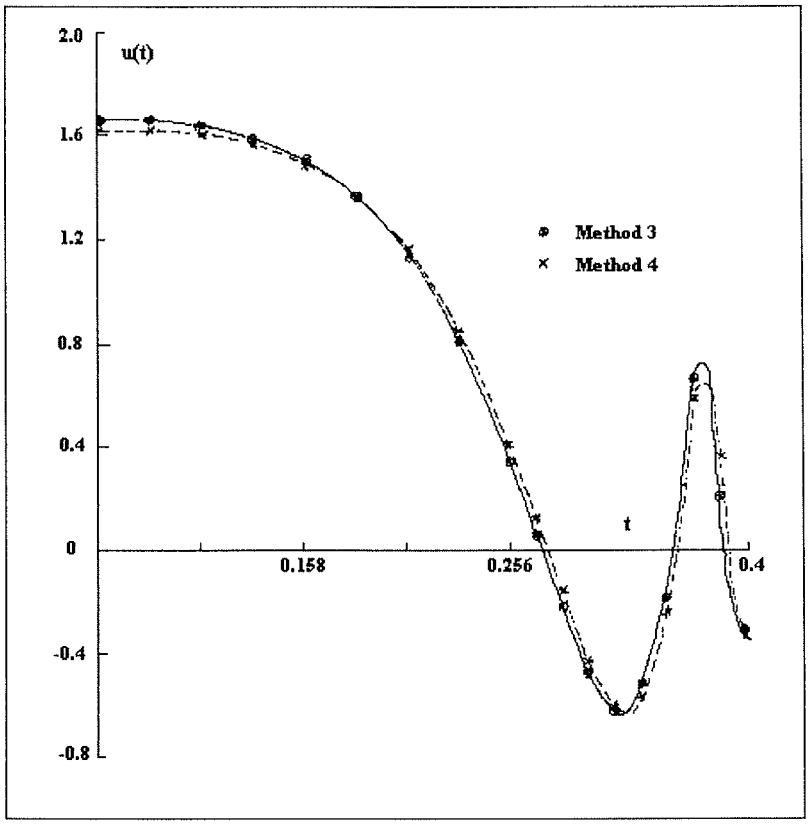


Figure 5.1: Control graphs for Method 3 and Method 4 in [17]

into $NT = 100$ fixed time steps. The Runge-Kutta algorithm requires that the control be evaluated at the midpoint of each time subinterval $[t_i, t_{i+1}]$. A possible solution string for the desired optimising control was constructed as an array $\underset{\sim}{u}$ of 201 constant real elements u_i.

By choosing the u_i^k randomly in the interval $[-1, 2]$ the initial population, $P(0) = \{\underset{\sim}{u}^k \ : \ k = 1, \cdots, M\}$, where M is the number of strings, was determined. The values of the global upper and lower bounds, $U_m = 2$ and $U_\ell = -1$ respectively, were chosen from the control function graphs given in [17]. These bounds, defining a convex region in the control space, can clearly be observed in Figure 5.1.

The mechanisms used in determining successive generations of strings in the evolutionary algorithm were as follows:

- A full replacement policy from generation to generation.
- Tournament selection with size $n_T = 4$.
- Arithmetic crossover with probability p_c.
- Elitism with 4 copies of the best string passed to the next generation.

The mutation operator, occurring with a probability of p_m, approximated small random Gaussian perturbations δu_k and is expressed as:

$$\text{if (mutate) } u_k = u_k + \delta u_k \text{ where } \delta u_k = \alpha(\sum_{k=1}^{12} r_k - 6)/6,$$

where α was a small constant, characteristically around 0.1, and r_k was a random number in the interval $[0, 1]$. The new u_k was capped to lie within the desired interval to ensure that the perturbation resulted in practical values as follows:

$$\text{If } u_k > U_m \text{ then } u_k = U_m, \text{ and if } u_k < U_\ell \text{ then } u_k = U_\ell.$$

In order to measure the fitness of each string, the following fitness function was created:

$$F_{obj} = \beta_1 A(J_1) + \beta_2 A(J_2) + P_1 + P_2 + P_3,$$

where $A(J_k), k = 1, 2$ was either a Trapezoidal approximation or Simpson's approximation to the integrals J_k. $P_k, k = 1, 2, 3$ are the penalty terms which are defined below:

$$\begin{aligned} P_1 &= \alpha_3 u_{201}^2, \qquad P_2 = \alpha_1 \sum_{k=1}^{200} (u_{k+1} - u_k)^2, \\ P_3 &= \alpha_2 \sum_{k=2}^{200} (u_k - u_{k-1})(u_{k+1} - u_k) \ \ (if) \ \ (u_k - u_{k-1})(u_{k+1} - u_k) < 0. \end{aligned}$$

It is assumed that the constants $\beta_k, k = 1, 2$ are positive. The minimum of $\beta_1 A(J_1) + \beta_2 A(J_2)$, it should be noted, is the same as the minimum of $A(J_1) + A(J_2)$. This is due to the fact that both integrals are positive. These constants were set at $\beta_1 = \beta_2 = 1$ for the given set of parameters.

Due to the necessary requirements of the minimum principle requiring that the final value of the control be zero at the final time T, the first penalty term was introduced to make this so. The second penalty term exists in order to provide assurance that the sum of the squares of the differences in the piecewise constant approximation to the control remains small. The third penalty term seeks to remove any spiking in the graph due to gradient changes under mutation. Without these latter two penalty functions the solution string arrived at by the evolutionary algorithm tends towards bang-bang control, cf [5]. For the given set of parameters, $\alpha_k, k = 1, 2$ were chosen as small constants with values typically around 10^{-4} to 10^{-6}, while α_3 was set to 1.0.

The results shown in column 4 of Table 5.1 were obtained by the evolutionary

algorithm around generation 236400. The algorithm was run for the first 50000 generations with a population of 100 and a high mutation probability of $p_m = 1.0$. For the remainder of the generations p_m was set to 0.7. Throughout the entire run the crossover probability, p_c, was set at 0.6 and α at 0.06. To integrate J_1 and J_2 a Trapezoidal approximation was used.

Figure 5.2 shows the control graphs obtained by both the finite element (FE) and evolutionary algorithm (EA) methods where the horizontal axis is time t, the vertical axis is the control $u(t)$ and the FE result is represented by the solid line.

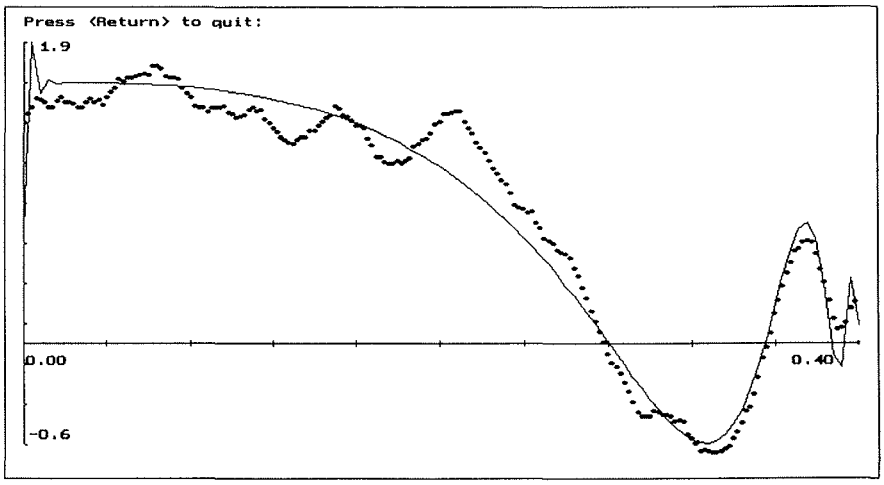


Figure 5.2: Control graphs for methods FE and EA - $r = 0.00001$

5.5 Comparison

It can be seen in Table 5.1 that, when compared with the two methods used for the semi-discretised system reported in [17], the finite element and evolutionary algorithm methods yield comparable results. There is a singularity at $y = 0$ which

was caused by the required jump in solution to $\eta(0,T) = 0.2$ and produced a spike in the initial values of the control for the finite element method but did not appear to be a problem with the evolutionary algorithm method. Figure 5.2 shows that the control obtained via the evolutionary algorithm converges quite well to the solution obtained directly from the finite element method. This is particularly true over the later half of the time interval. The inability of the solution to settle over the first half of the time interval may be the result of the singularity at $y = 0$. Both of the control solutions demonstrate an oscillation at the end of the time interval that is not apparent in the results obtained in [17]. In that paper it was reported that none of the methods they employed yielded convergence to $y(T) = 0$ as was required by the minimum principle and that this was due to the computational nature of the methods. It was to overcome this difficulty specifically that the penalty term P_1 was introduced.

Experimentation revealed that, although the evolutionary algorithm required a re-evaluation of penalty parameters in the algorithm, it appeared to be very robust. Using Simpson's rule for integration rather that the Trapezoidal rule appears to not cause oscillatory problems that were encountered in[17]. The evolutionary algorithm was critically dependent upon a high mutation level and small perturbation factor for reasonable convergence to the desired level of 0.2. By increasing the value of the constant β_1 it was found that the algorithm converged faster and with better accuracy to the desired state.

The mutation operator used by Michalewicz [19] was trialed, but was found to pro-

duce results closer in form to 'bang-bang' control. It was also found that use of the Trapezoidal approximation to evaluate the two integrals produced better convergence than use of the Simpson approximation. By increasing the number of depth steps N and increasing the number of time discretisations NT better convergence could be obtained but at the cost of computational time and complexity.

The definition of the evolutionary algorithm was very basic, consisting of the standard operators of arithmetic crossover and mutation. Special operators used to enhance convergence, such as quadratic operators and others investigated in [25] and [30], were not implemented in this study. Use was not made of a second level structure in each string to incorporate variation in the time node points, however, the implementation of this device may have assisted in gaining a better understanding of the oscillatory variation found in the solution at the end of the time interval. To do this, the learning in the areas where the control solution is changing rapidly could have been enhanced. The major problem with this approach to finding a solution is in determining appropriate penalty factors as is illustrated in the next section.

5.6 Simulation with parameter $r = 0.0005$

In this section the problem is solved with the parameters unchanged from the previous case with the exception of $r = 0.0005$. Table 5.2 below shows the comparison for this new result set. As before, $N = 10$ and $NT = 100$.

Station y	Method 3	Method 4	FE	EA
0.0	0.1985	0.1937	0.19981	0.20002
0.1	0.1979	0.1966	0.20026	0.20009
0.2	0.2034	0.2016	0.19831	0.20006
0.3	0.2064	0.2070	0.19996	0.19998
0.4	0.2045	0.2039	0.20273	0.19956
0.5	0.1984	0.1990	0.20368	0.19989
0.6	0.1901	0.1902	0.20209	0.20050
0.7	0.1816	0.1817	0.19877	0.20061
0.8	0.1743	0.1748	0.19509	0.20010
0.9	0.1695	0.1693	0.19231	0.19942
1.0	0.1678	0.1684	0.19129	0.19911

Table 5.2: Comparison of output results - $r = 0.0005$

Utilising the same set of parameters as defined in the previous test case, the evolutionary algorithm was found to be slow to converge. The control it produced was saturated at both the upper and lower bounds for two non-empty subintervals of the integration.

As a result of this, the following changes were made in order to improve performance:

- Constant β_1 was increased to 100.

- Constant β_2 was set to 1.
- $\alpha_k = 10^{-6}, k = 1, 2.$
- $\alpha_3 = 1.$
- The control bounds were increased such that $u_i^k \in [-2, 4]$.
- In the case of there being too large a jump in control another penalty function was added:

$$P_4 = 100, \text{ if } |(u_{k+1} - u_k)| > 0.3.$$

With these changes, the evolutionary algorithm was found to converge to the tabulated values within 143000 generations. These are excellent results when compared to those of the other methods.

Figure 5.3 displays the control graphs obtained by the Finite Element and Evolutionary Algorithm methods where the horizontal axis is time t, the vertical axis is the control $u(t)$ and the Finite Element result is represented by the solid line.

There exists a significant difference between the two control graphs on the second half of the time interval where the solution of evolutionary algorithm is quite oscillatory. One possible cause for this oscillation is that the approximation to the cost integral was not reduced by the algorithm to those values given in the first simulation where $r = 0.00001$. r was a factor in this integral and in this instance it is now a factor of 50 times greater than before.

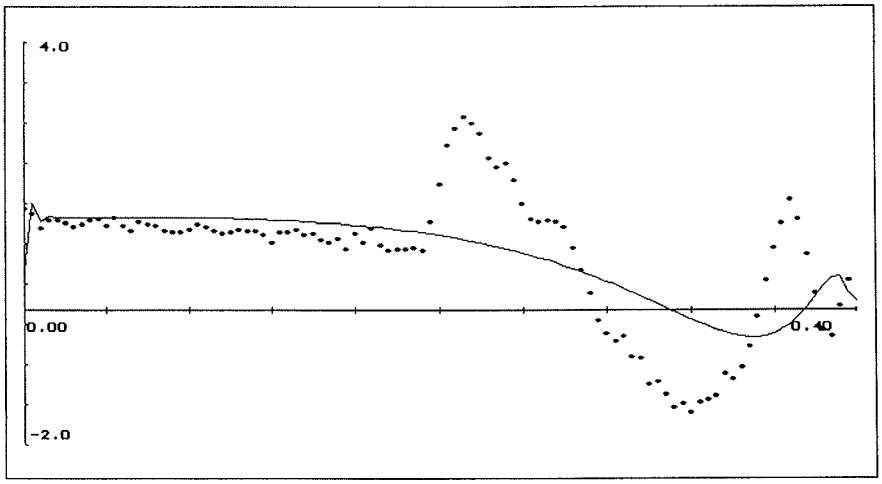


Figure 5.3: Control graphs for methods FE and EA - $r = 0.0005$

5.7 Constant target case $r = 0.00001$ revisited

Our initial discussion concerned the case of a constant target function $\eta = 0.2$ using the parameters: $L = 1$, $T = 0.4$, $r = 0.00001$, and $\rho = 1.0$.

As stated, the results were achieved at around generation 236400, with a population of 100, high mutation $p_m = 1.0$ for the first 50000 generations and then 0.7 thereafter, $p_c = 0.6$ and $\alpha = 0.06$, and were in reasonable agreement with those obtained by the finite element method, see Figure 5.2.

As can be seen, the control obtained via the EA, does approximate well the solution obtained directly by the finite element method (the continuous line on the graph). The number of generations is unacceptably high and the EA depended critically on a high level of mutation and a small perturbation factor to enable tracking to the desired level of 0.2 to be effective.

In order to address these issues, let M(α) define the mutation given above, and take the new mutation operator to be:

```
if (mutate)
 if (flip(0.33))  M(1.0);
 else
    if (flip(0.5)) M(0.1);
    else
      if (flip(0.5))  pd = gen/maxgen;
         else  pd = 0.995;
      pow_ = ((1 - pd) * (1 - pd));
      fact = (1 - power(Random(), pow_));
      if (flip(0.5))
        delta = fact * (lb - u_k);
      else
        delta  = fact * (u - u_k);
      return (u_k + delta);
 else
   return(u_k);
```

If a control value u_k is mutated, then the basic mutation with a large factor $\alpha = 1$, is implemented one third of the time, basic mutation with a small factor $\alpha = 0.1$ one third of the time, and a modification of Michalewicz's mutation [19], is applied the remainder of the time. The standard arithmetic crossover with constant α_c and two parents generating two children was modified as follows: Given two parents p_1

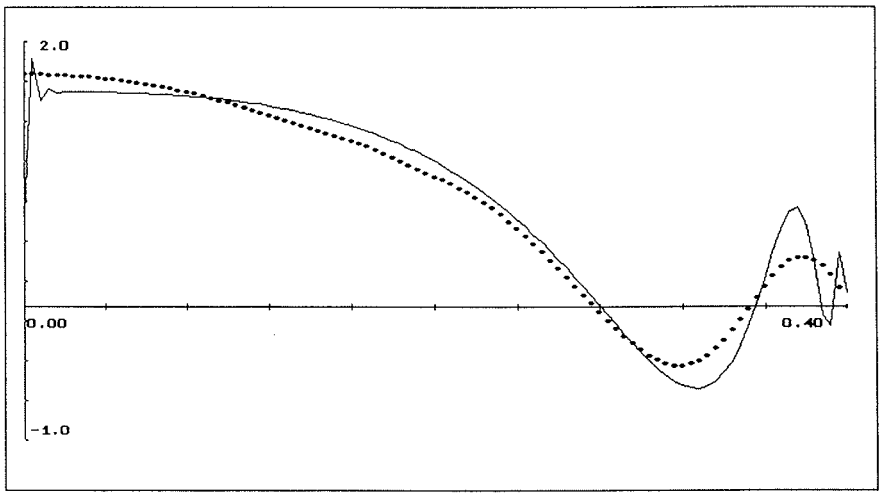


Figure 5.4: Control graphs for methods FE and EA - $r = 0.00001$ revisited

and $\underset{\sim}{p}_2$ selected by tournament a child $\underset{\sim}{c}_1$ is then determined by random $\alpha_c \in [0, 1]$ from the equation

$$\underset{\sim}{c}_1 = \alpha_c \underset{\sim}{p}_1 + (1 - \alpha_c)\underset{\sim}{p}_2.$$

With continued selection of parent pairs, the children are added to the next generation's population until it is complete.

Applying these new operators with small mutation $p_m = 0.01$, the evolutionary algorithm yielded the control graph given by the dotted line in Figure 5.4 within 100000 generations. The results are a substantial improvement upon those obtained in [33]. Indeed the resultant control curve is very much as shown in 70000 generations.

5.8 Constant target case $r = 0.001$

Returning to the constant target case $\eta = 0.2$, with the new operators and with small $p_m = 0.01$, the evolutionary algorithm converged quickly within 7000 generations to the control graph shown by the dotted line in Figure 5.5. The value of $J_1 + J_2$ was 0.00021561 for the evolutionary algorithm and 0.0002143 for the finite element method.

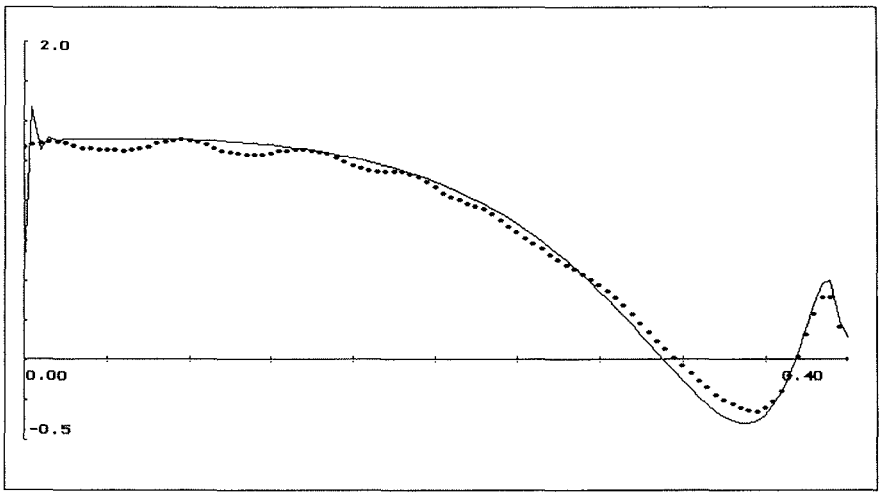


Figure 5.5: Control graphs for methods FE and EA - constant case $r = 0.001$

We now discuss results for two other cases discussed in [17].

5.9 Ramp case

In this case the target function is given by

$$\eta(y,T) = \begin{cases} 0.4(y/L) & 0 \leq y/L < 0.5, \\ 0.2 & 0.5 \leq y/L \leq 1.0, \end{cases}$$

Station y	Method 3	Method 4	FE	EA
0.0	0.2001	0.1953	0.2003	0.1991
0.1	0.1990	0.2004	0.1993	0.1989
0.2	0.1987	0.1980	0.1983	0.1992
0.3	0.2020	0.2020	0.2016	0.2023
0.4	0.2044	0.2040	0.2044	0.2046
0.5	0.2040	0.2041	0.2042	0.2039
0.6	0.2008	0.2009	0.2011	0.2006
0.7	0.1962	0.1962	0.1963	0.1960
0.8	0.1915	0.1918	0.1916	0.1914
0.9	0.1882	0.1882	0.1882	0.1881
1.0	0.1870	0.1874	0.1869	0.1869

Table 5.3: Comparison of output results - constant case $r = 0.001$

and the parameter r is maintained at the same value $r = 0.001$. This case is easier than the constant case as a non-zero target value at $y = 0$ is no longer demanded. In initialising the evolutionary algorithm and for the mutation operator, global bounds on the control were selected as $U_\ell = -7$ and $U_m = 6$.

Results obtained for a longer run of 300000 generations to obtain the control graph shown again by the dotted line in Figure 5.6. The approximation with the solution found by the finite element method is excellent. The value of $J_1 + J_2$ obtained

was 0.00028373 for the evolutionary algorithm and 0.0002732 for the finite element method. A value of $J_1 + J_2$ of 0.000285 was obtained by the evolutionary algorithm within 11000 generations. Table 5.4 shows a comparision of the results of the evo-

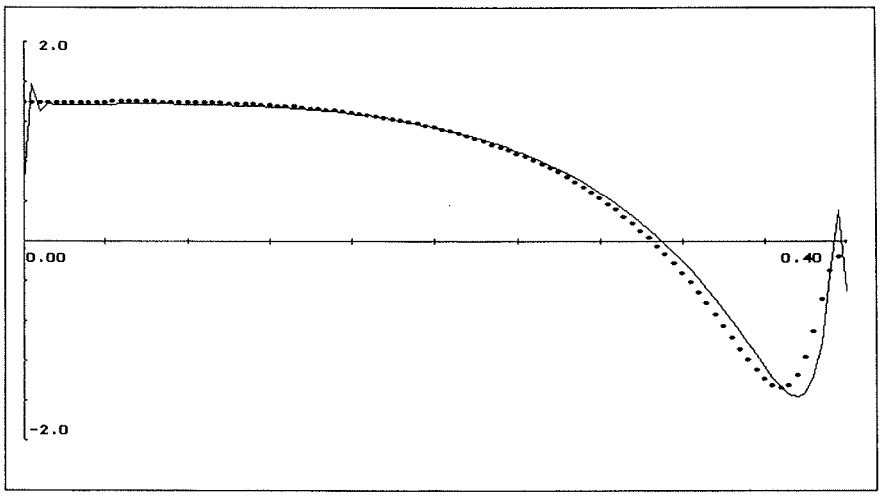


Figure 5.6: Control graphs for methods FE and EA - ramp case

lutionary algorithm and the finite element method with Method 3 of [17] which is a numerical integration of the state and costate partial differential equations arising as necessary conditions from the minimum principle for distributed systems.

5.10 Triangle case

The final case discussed by Huntley is that of a triangular target function with its peak at $y/L = 0.2$, defined by

$$\eta(y,T) = \begin{cases} 0.4(y/L) & 0 \leq y/L < 0.5, \\ 0.4(L-y)/L & 0.5 \leq y/L \leq 1.0, \end{cases}$$

Station y	Ramp	Method 3	FE	EA
0.0	0.00	0.0011	0.0022	0.0028
0.1	0.04	0.0368	0.0367	0.0349
0.2	0.08	0.0846	0.0829	0.0857
0.3	0.12	0.1317	0.1316	0.1324
0.4	0.16	0.1658	0.1666	0.1656
0.5	0.20	0.1853	0.1862	0.1848
0.6	0.20	0.1939	0.1944	0.1935
0.7	0.20	0.1958	0.1959	0.1957
0.8	0.20	0.1947	0.1944	0.1947
0.9	0.20	0.1929	0.1925	0.1931
1.0	0.20	0.1922	0.1917	0.1924

Table 5.4: Comparison of output results - ramp case

In this case we take the parameter $r = 0.0001$ and the global bounds on the control are taken to be those in the ramp case. Noting that boundary control is applied at one end of the spatial dimension only, it is not obvious that this difficult target function might be achieved computationally. Results shown in Figure 5.7 obtained for a run of 300000 generations to obtain the control graph shown again by the dotted line in Figure 5.6. The approximation with the solution found by the finite element method is excellent. The value of $J_1 + J_2$ obtained was 0.00033010 for the

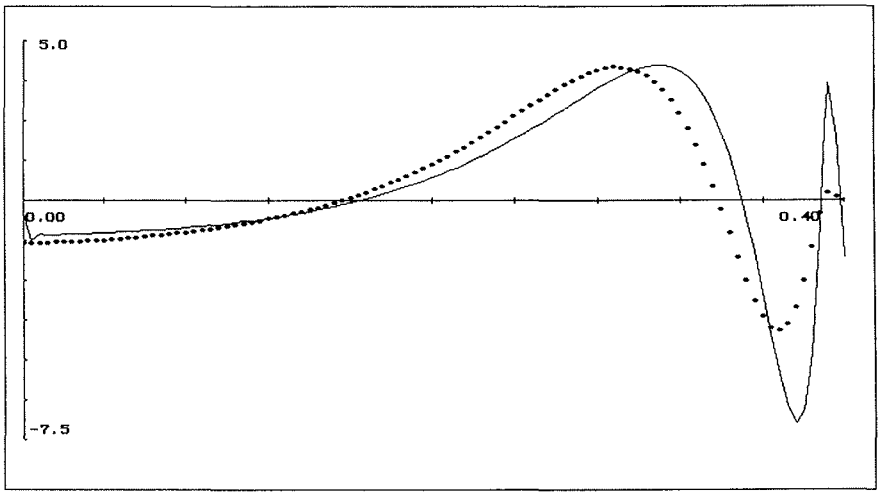


Figure 5.7: Control graphs for methods FE and EA - triangle case

evolutionary algorithm and 0.0002249 for the finite element method.

These target values are consistent with the graphical results for the case depicted in [17]. However the graphs of the control for both the evolutionary algorithm and finite element are not in good agreement. Further they are in disagreement with an extra oscillation in control at the later end of the time interval as described in Figure 5 of [17].

5.11 Summary

In this chapter we have shown that evolutionary algorithms can be applied to optimal control problems in distributed parameter systems under semi-discretisaion with a good measure of success provided care is taken in correctly defining the right blend of operators and parameters.

Station y	Triangle	FE	EA
0.0	0.00	-0.0009	0.0043
0.1	0.04	0.0423	0.0276
0.2	0.08	0.0701	0.0885
0.3	0.12	0.1358	0.1438
0.4	0.16	0.1763	0.1684
0.5	0.20	0.1742	0.1626
0.6	0.16	0.1450	0.1377
0.7	0.12	0.1076	0.1066
0.8	0.08	0.0748	0.0786
0.9	0.04	0.0532	0.0598
1.0	0.00	0.0458	0.0533

Table 5.5: Comparison of output results - triangle case

Future research is now considered feasible by progressing this application of evolutionary algorithms to solve the boundary control for irrigation scheduling using the Richards' flow equation, implementing semi-discretisation to convert the problem into a classical optimal control problem with ordinary differential constraints, [17]. From this position one can then apply the growing number of techniques in the application of evolutionary learning to the resulting constrained optimal control problem and compare the results obtained with those obtained by the genetic algorithm

in [32] and the results in [18]. It needs to be noted that the results obtained in [32] yield a bang-bang control at the surface boundary whereas the results obtained in this chapter yield a boundary control which is continuous.

The solution obtained for this boundary control problem constitutes an open-loop solution to the problem, that is, the solution is a function of time only and does not include any feedback on the current value of the state variable. The closed loop solution for the boundary control problem poses more computational difficulties as reported in Huntley. Yet it should be possible to find a closed loop solution by developing a fuzzy controller using evolutionary algorithms in a manner similar to that described in [30]. This is a topic of future research beyond the scope of this thesis.

Chapter 6

Conclusion

This thesis was an initial investigation of how modern artificial intelligent techniques, namely the genetic algorithm/evolutionary algorithm could be applied to find irrigation strategies for a cropped soil.

The exercise was to find such strategies to keep the moisture content in the soil layers at sufficient levels to enable nutrient uptake by the plant.

We first introduced concepts in soil physics in order to give an understanding of the nature of soil composition and the movement of water within a cropped soil. Previous work undertaken by Janz in his Masters Thesis, was then summarised to show how classical solutions to the Richards' flow equation could be obtained for an irrigation schedule, defining periods of time when the water infiltration at the surface boundary was assumed to be off or assumed to be on.

A brief introduction to genetic and evolutionary algorithms defining their algorith-

mic structure then followed in Chapter 3.

Our research commenced with the novel application of a genetic algorithm to obtain a schedule of irrigation for control of the moisture content to be maintained at specific levels at certain depths within the soil to maximise "nutrient uptake" by the root. Essentially this is a classic tracking problem in optimal control. In this application an individual string of the genetic algorithm, was a binary string composed of 0's, representing a day without irrigation, and 1's, representing 1 day with irrigation. This string represented a possible irrigation schedule. It was of bang-bang control type with typical control off or control on. We showed that it was possible to learn a realistic irrigation schedule for a problem posed with realistic field data.

A problem with applying this technique still lay in using the classical solution of the flow equations as undertaken by Janz, and continually making appropriate correction to ensure accuracy in the transition from irrigation on to irrigation off, when using the Crank-Nicolson method.

Encouraged by the new substantial research by Stephen Smith on the feasible application of evolutionary algorithms to solve difficult non-linear optimal control problems, we sought to investigate this problem further by examining how an evolutionary algorithm could be used to solve the tracking problem that was solved by the genetic algorithm in Chapter 4.

We turned our attention to the boundary control of a distributed process which

is inherently the same as the one posed but with a simplified structure compared with the parameter complexity in the Richards' flow equation. It still however was a difficult problem to solve with inherent stability problems as can be ascertained from a reading of Huntley's paper, [17].

Our objective here was to establish if such a problem could be solved by an evolutionary algorithm approach with reasonably simple operators and structural definition of the individuals in the population.

A continuous open loop control applied at the boundary, was found by evolutionary learning and compared with two methods from Huntley's paper and the more popular finite element method. The results obtained verify that the evolutionary learning approach to find the boundary control compares favourably with the classical techniques commonly used to solve this type of problem, in all cases including the constant, ramp and triangular target final states.

We believe that the research presented in this thesis shows that it is now possible to continue an investigation into the evolutionary learning of an open loop, boundary control solution to find an irrigation schedule which will keep the moisture content in the soil at specified levels in the vicinity of the plant-root structure, using a full mathematical model of Richards' equation with realistic field parameters and data. Such an application would need to account for the non-practical nature of a continuous control, as derived from the evolutionary algorithm, when considering its application to an irrigation scheduling problem. Indeed the aim should be to arrive at some form of discretisation for the control as was present in the genetic algorithm

control solution.

Bibliography

[1] Ames, W.F., Numerical Methods for Partial Differential Equations, Academic Press, New York, 1977.

[2] Baeck, Th., *Evolutionary Algorithms in Theory and Practice*, Oxford University press, NY, 1996.

[3] Baeck, Th. and Schwefel, H.P, *Evolutionary Computation: An Overview*, Proceedings ICEC'96, Nagoya, Japan, 20–29, May 1996.

[4] Clapp, R.B. and Hornberger, G.M., Empirical Equations for Some Soil Hydraulic Properties, *Water Resources Research*, Vol. 14, No. 4, 601-604, 1978.

[5] Dorf, R.C., *Modern Control Systems*, Addison-Wesley 1986.

[6] Feddes, R.A., Bresler, E., and Neumann, S.P., Field Test of a Modified Numerical Model for Water Uptake by Root Systems, *Water Resources Research*, Vol. 10, No. 6, 199-206, 1974.

[7] Fogel, D.B, *Evolutionary computation: Toward a New Philosphy of Machine Intelligence*, IEEE Press, Piscataway, NJ, 1995.

[8] Fogel, D.B. and A. Ghozeil, A., Using Fitness Distributions to Design more Efficient Evolutionary Computations, *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 11-19, 1996.

[9] Gardner, W.R., Relation of Root Distribution to Water Uptake and Availability, *Agronomy Journal*, Vol. 56, 41-45, 1964.

[10] Gardner, W.R., Hillel, D., Benyamini,Y., Post irrigation movement of soil water: I. Redistribution, *Water Resources Res.* 6 (3), 851-861; II. Simultaneous redistribution and evaporation, *Water Resources Res.* 6 (4), 1148-1153. 1970.

[11] Goldberg, D., *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison–Wesley, 1989.

[12] Hayhoe, H.N., Analysis of a Diffusion Model for Plant Root Growth and an Application to Plant Soil Water Soil Uptake, *Soil Sciences*, Vol. 131, No. 6, 334-343, 1981.

[13] Hayhoe, H.N. an De Jong, R., Comparison of Two Soil Water Models for Soybeans, *Canadian Agricultural Engineering*, Vol. 30, No 1, 5-11, 1988.

[14] Hillel, D., *Soil and water : physical principles and processes*, New York : Academic Press, 1971.

[15] Hogarth, W.L. and Watson, K.K., Water Infiltration in Unsaturated Soils: a Review of Some Recent Studies, *Trends in soil Science*, Vol. 1, 277-284, 1991.

[16] Holland, J.H., *Adaptation in natural and artificial systems.* Ann Arbor: The University of Michigan Press. 1975.

[17] Huntley, E., Optimal boundary control of a tracking problem for a parabolic distributed parameter system, *International Journal of Control*, Vol. 42, No. 2, 411-431, 1985.

[18] Janz, T.C., *Mathematical modelling of water flow in cropped soils*, Masters in Applied Science, Central Queensland University, 1992.

[19] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd Edition, Springer Verlag, 1994.

[20] Michalewicz, Z., *Evolutionary Computation: Practical Issues*, Proceedings ICEC'96, Nagoya, Japan, 30-39, May 1996.

[21] Molz, F.J. and Remson, I., Extraction Term Models of Soil Moisture Use by Transpiring Plants, *Water Resources Research* Vol. 6, No. 5, 1346-1356, 1970.

[22] Nimah, M.N. and Hanks, R.J., Model for Estimating Soil Water, Plant and Atmospheric Interrelations: I. Description and Sensitivity, *Proc. Soil Soc. of America*, Vol. 37, 522-527, 1973.

[23] Olsson, K.A, and Rose, C.W., Patterns of Withdrawal Beneath an Irrigated Peach Orchard on a Red-Brown Earth. *Irrigation Science*, Vol. 9, 89-104, 1988.

[24] Smith, S.F. and Stonier, R.J., Applying evolutionary programming to constrained continuous optimal control problems, *Proceedings of IEEE International Conference on Evolutionary Computing*, Nagoya Japan, 285-290, 1996.

[25] Smith, S.F. and Stonier, R.J., Evolutionary Operators for Constrained Continuous Optimal Control and High Dimensional Constrained Optimisation Problems, *Australian Journal of Intelligent Information Processing Systems*, Vol. 4, No. 3/4, 240-248, 1997.

[26] Smith, S.F., The simplex method and evolutionary algorithms, *Proceedings of the International Conference on Evolutionary Computing*, Anchorage, Alaska, 799-804, 1998.

[27] Stonier, R.J., Control Problems in Irrigation Management, *Proceedings of the Second International Workshop on Control Theory applied to Renewable Resource Management*, Lecture Notes in Biomathematics, Springer Verlag, Vol. 72, 195-206, 1987.

[28] Stonier, R.J. and Janz T.C., Modelling Water Flow in Cropped Soils: Simulating the Irrigation Cycle, *Proceedings of the International Congress on Modelling and Simulation*, Perth, WA Vol. 3, 931-936, 1993.

[29] Stonier, R.J. and Janz T.C., Modelling Water Flow in Cropped Soils: Water Uptake by Plant Roots, *Environment International* Vol. 21, No. 1, 705-709, 1995.

[30] Stonier, R.J., Stacey, A., Mohammadian, M. and Smith, S.F., Applications of evolutionary learning in fuzzy logic and optimal control, *Computational Intelligence for Modelling ,Control and Automation, Evolutionary Computation, and Fuzzy Logic for Intelligent Control, Knowledge Acquisition and Information Retrieval*, M. Mohammadian (Ed), IOS Press, 76-85, 1999.

[31] Stonier, R.J. and Sturgess, D.K., Genetic learning of the irrigation cycle for water flow in cropped soils, *Proceedings of the First Asian-Pacific Conference on Evolutionary Algorithms and Learning (SEAL'96)*, KAIST, Taejon, Korea, 201-208, November 9-12th, 1996.

[32] Stonier, R.J. and Sturgess, D.K., Genetic learning of the irrigation cycle for water flow in cropped soils, *Lecture Notes in Artificial Intelligence*, Xin Yao, Jong-Hwan Kim & Takeshi Furuhashi (Eds), Springer Verlag, Vol. 1285, 89-96, 1997.

[33] Stonier, R.J., Sturgess, D.K., Smith, S.F. and Drumm, M.J., Optimal boundary control of a tracking problem using evolutionary algorithms, International Conference on Computational Intelligence for Modelling, Control and Automation, ISBN 0858898470, M. Mohammadian (Ed), Las Vegas, 204-214, 2001.

[34] Visser, W.C., Progress in the knowledge about the effect of soil moisture content on plant production, Inst. Land Water Management, Wageningen, Netherlands, Tech. Bull. 45. 1966.