

82220

NUMERICAL METHODS FOR THE SOLUTION
OF
THE NEOCLASSICAL GROWTH MODEL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF SOCIAL SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

BÜLENT DEDEOĞLU

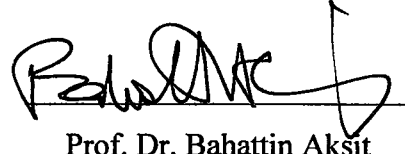
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE
OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF ECONOMICS

T 82220

TC. YÜKSEKÖĞRETİM KURULU
BOLUNANTASYON MERKEZİ

SEPTEMBER 1999

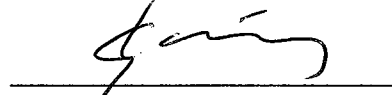
Approval of the Graduate School of Social Sciences



Prof. Dr. Bahattin Akşit

Director

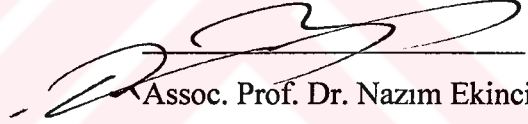
I certify that this thesis satisfies all the requirements as a thesis for the degree of
Master of Science



Prof. Dr. Erol Taymaz

Head of Department

This is to certify that we had read this thesis and that in our opinion it is fully
adequate, in scope and quality, as a thesis for the degree of Master of Science.



Assoc. Prof. Dr. Nazım Ekinci

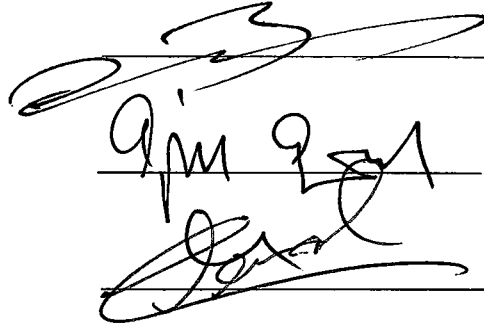
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Nazım Ekinci

Assoc. Prof. Dr. Alper Güzel

Assistant Prof. Dr. Cemal Akyel



ABSTRACT

NUMERICAL METHODS FOR THE SOLUTION OF THE NEOCLASSICAL GROWTH MODEL

Dedeođlu, Bülent

M.S., Department of Economics

Supervisor: Assoc. Prof. Dr. Nazım Ekinci

September 1999, 108 pages

This thesis is an attempt to write a software program which solves dynamic optimisation problems (optimal control problems) specifically for the neoclassical growth model and find the approximate polynomial equation of the saddle path and sketch the graph of this path by using the numerical and simulation techniques.

Keywords: The Neoclassical Growth Model,
Optimum Control Theory.

ÖZ

**NEOKLASİK BÜYÜME MODELLERİNİN ÇÖZÜMÜ İÇİN
NUMERİK METODLAR**

Dedeođlu, Bülent

Yüksek Lisans, İktisat Bölümü

Tez Yöneticisi: Doç. Dr. Nazım Ekinci

Eylül 1999, 108 pages

Bu çalışma, özellikle büyüme modellerinde karşılaşılan, dinamik optimizasyon problemlerini (optimal kontrol problemlerini) numerik metodlar kullanarak çözen ve saddle-path'in polinom denklemini bulan ve grafiđini çizen bir software program yazılımını amaçlamaktadır.

Anahtar Sözcükler: Neoklasik Büyüme Modelleri,
Optimum Kontrol Teori

ACKNOWLEDGMENTS

I express sincere appreciation to Assoc. Prof. Dr. Nazım Ekinçi for his encouragement, guidance and insight throughout the research. I offer sincere thanks to Hakan Yetkiner for his amity and assistance. To my parents, I thank them for their support.



TABLE OF CONTENTS

ABSTRACT.....	iii
ÖZ.....	iv
ACKNOWLEDGEMENTS.....	v
TABLE OF CONTENTS.....	vi
LIST OF TABLES.....	ix
LIST OF FIGURES.....	x
CHAPTER	
1. INTRODUCTION.....	1
2. DYNAMIC OPTIMISATION	
2.1. An Introduction to Dynamic Optimisation.....	3
2.1.1 The Maximum Principle.....	7
2.2. The Typical Problem of Optimal Control Theory.....	9
2.3. Heuristic Derivation of the First-Order Conditions.....	11
2.4. Transversality Condition.....	14
2.5. The Behaviour of the Hamiltonian.....	15
2.6. Sufficient Conditions.....	16
2.6.1 The Mangasarian Sufficiency Theorem.....	16
2.6.2 The Arrow Sufficiency Theorem.....	18
2.7 Infinite Horizon Problem.....	19
2.8 An Economic Interpretation of Optimal Control Theory.....	19
2.8.1 The Basic Equations.....	20

**3. THE NEOCLASSICAL THEORY OF OPTIMAL GROWTH
AND RAMSEY’S ANALYSIS**

3.1 The Neoclassical Growth Model..... 27

3.2 The Maximum Principle..... 31

3.3 Phase Diagram Analysis..... 36

3.4 Transversality Conditions..... 40

3.5 Checking The Saddle Point by Characteristic Roots..... 41

3.6 The Shape of the Stable-Arm..... 43

3.7 Typical Example of Solving The Stable Arm Equation by Using The
Method of Taylor’s Approximation in the Cobb-Douglass Production
Function Case..... 44

**4. NUMERICAL METHODS FOR SOLVING ORDINARY
DIFFERENTIAL EQUATIONS**

4.1 Ordinary Differential Equations and The Lipschitz Condition..... 51

4.2 Euler’s Method..... 52

4.3 The Trapezoidal Rule..... 56

4.4 The Theta Method.....60

4.5 The Adams Method..... 61

4.6 Order and Convergence of Multistep Methods..... 63

4.7 Backward Differentiation Formula.....65

4.8 Taylor’s Theorem and Runge Kutta Methods..... 67

4.9 Gaussian Elimination and Pivoting..... 71

**5. TYPICAL EXAMPLE OF THE NUMERICAL SOLUTION OF THE STABLE
ARM BY USING THE SIMULATION PROGRAM**

5.1 Input Initial Values and Parameters.....75

5.2 Choosing The Numerical Methods and Approximating Polynomial.....77

5.3 Results and Required Options..... 80

5.4 Conclusions..... 83

REFERENCES..... 84

APPENDICES

1. The Simulation Algorithm in Mathematica..... 85
2. The Simulation Algorithm in Pascal..... 87



LIST OF TABLES

1. The iteration results for the theta method for the solution of the non-linear differential equation system..... 79



LIST OF FIGURES

1. Phase Diagram analysis.....	37
2. Phase Diagram analysis.....	38



CHAPTER 1

INTRODUCTION

In the First Phase of the Neoclassical Growth Theory, the most well known approach is developed by Robert Solow (and independently by Trever Swan). The key aspect of the Solow-Swan model is the neoclassical form of the production function, a specification that assumes constant returns to scale, diminishing returns to each input, and some positive constant elasticity of substitution between the inputs. This production function is combined with a constant-saving-rate rule to explain macroeconomic growth dynamics of an economy. The main conclusion of the Solowian approach is as follows: in the absence of continuing (exogenous) improvements in technology, per capita growth must eventually cease. Thus it is known that positive rates of per capita growth persist over a century and these growth rates have no tendency to decline. So, we end up with a model of growth in which long run growth is exogenously determined. David Cass and Tjalling Koopmans brought Ramsey's analysis of consumer optimisation back into the neoclassical growth theory and thereby provided for an endogenous determination of the saving rate. This extension allows for richer transitional dynamics but does not eliminate the dependence of the long run per capita growth rate on exogenous technological progress.

In the Second Phase, research on economic growth has experienced a new boom, beginning with the seminal works of Paul Romer and Robert Lucas since mid-1980s. The incorporation of R&D and imperfect competition, human capital, governmental actions, international trade, financial markets, and other aspects of the economy into the growth framework have enriched growth literature and have been shown that a positive steady-state growth exists.

Most of these studies are presented in dynamical general equilibrium environment and are basically constructed in discrete or continuous time framework. For those in the latter form, a dynamic optimisation problem is solved by using optimal control theory method given that saving rate is endogenously determined. However, as literature grows, it has been observed that optimisation problems becomes highly complicated. Most studies have started to present only steady-state solutions as it becomes almost impossible to solve these problems for their transition periods. Moreover, some models need to be simulated as they are presented in parametric form and parameter values are not known empirically.

This study is an attempt to write a software programme which solves dynamic optimisation problems (optimal control problems) specific to growth problems (that is newly emerging endogenous growth models) and find a polynomial equation of the saddle path and sketch the graph of this path by using numerical simulation techniques. It is known that, after reducing an optimal control theory problem into a system of differential equations, it will be quite easy to obtain dynamics of variables under consideration as well as their steady-state values by means of numerical methods. Thus users will be able to see how a system of variables dynamically move together (time paths of variables) besides their steady-state values by the help of this software programme.

The simulation programme which solves the non-linear differential equations by using numerical techniques is written on the programming language 'PASCAL' and the other programme which solves these equations by linearizing them around the steady-state points with the help of Taylor's expansion is written under the Mathematica background. This second solution is suggested by Ramsey and is very well known. However the first one which involves a polynomial approach to the equation of the saddle path provides a new way of obtaining transitional dynamics.

CHAPTER 2

DYNAMIC OPTIMIZATION

2.1 An Introduction to Dynamic Optimisation

Optimisation is a predominant theme in economic analysis, Chiang (1992). A dynamic optimisation problem poses the question of what is the optimal magnitude of a choice variable in each period of time within the planning period (discrete time case) or at each point of time in a given time interval, say $[0, T]$ (continuous time case). This problem can be represented by the following formula:

$$\begin{array}{ll} \text{Maximize or minimize} & V[y] = \int_0^T F[t, y(t), y'(t)] dt \\ \text{subject to} & y(0) = A \quad (A \text{ given}) \\ \text{and} & y(T) = Z \quad (T, Z \text{ given}) \end{array} \quad (2.1)$$

It is also possible to consider an infinite planning horizon, so that the relevant time interval is $[0, \infty)$. The solution of dynamic optimization problem would thus take the form of an *optimal time path* for every choice variable.

Regardless of whether the variables are discrete or continuous, a simple type of dynamic optimization problem would contain the following basic ingredients:

1. a given *initial point* and a *terminal point*;
2. a set of *admissible paths* from the initial point to the terminal point;
3. a set of *path values* serving as performance indices associated with the various paths;
4. a specified objective -either to maximize or minimize the path value or performance index by choosing an *optimal path*.

The assumption of a given initial point may not be unduly restrictive, because, in the usual problem, the optimizing plan must start from some specific initial position, say, the current position. The terminal position, on the other hand, may very well turn out to be a flexible matter, with no inherent need for it to be predetermined. For instance, we may face only a fixed terminal time, have complete freedom to choose the terminal state. On the other hand, we may also be assigned a rigidly specified terminal state but are free to select the terminal time. In such a case the terminal point becomes a part of the optimal choice.

As the first type of variable terminal point, we may be given a fixed terminal time T , but a free terminal state. In such a problem, the planner has much greater freedom in the choice of the optimal path and, as a consequence, will be able to achieve a better optimal path value than if the terminal point is rigidly specified. This type of problem is commonly referred to in the literature as a *fixed-time-horizon problem*, or *fixed-time problem*, or alternatively *vertical-terminal-line problem* meaning that the terminal time of the problem is fixed rather than free.

The second type of variable-terminal-point problem reverses the roles played by the terminal time and terminal state; now the terminal state is stipulated, but the terminal time is free. Again, there is greater freedom of choice as compared with the case of a fixed terminal point. This type of problem is commonly referred to as a *fixed-endpoint problem*, or alternatively *horizontal-terminal-line problem*.

In the third type of a variable-terminal-point problem, neither the terminal time nor the terminal state is individually preset, but the two are tied together via constraint equation of the form $Z = \phi(T)$. This type of problem is called the *terminal-curve* (or *terminal-surface*) problem.

The common feature of variable-terminal-point problems is that the planner has one or more degree of freedom than in the fixed-terminal-point case. But this fact automatically implies that, in deriving the optimal solution, an extra condition is needed to pinpoint the exact path chosen and can conclusively distinguish the optimal path from the other admissible paths. Such a condition is referred to as a *transversality condition*, because it normally appears as a description of how the optimal path crosses the terminal line or the terminal curve.

An optimal path is, by definition, one that maximizes or minimizes the path value $V[y]$. Inasmuch as any y path must perforce travel through an interval of time, its total value would naturally be a sum. In the discrete-stage framework, the path value is the sum of the values of its component arcs. The continuous-time counterpart of such a sum is a definite integral, $\int_0^T (\text{arcvalue}) dt$. Three pieces of information is needed for defining the “arc value”: (1) the starting stage (time) t , (2) the starting state $y(t)$, and (3) the direction in which the arc proceeds $y'(t) = dy/dt$. It follows that the general expression for arc values is $F[t, y(t), y'(t)]$, and the path-value functional -the sum of arc values- can generally be written as the definite integral:

$$V[y] = \int_0^T F[t, y(t), y'(t)] dt \quad (2.2)$$

The continued study of variational problems has led to the development of the modern method of *optimal control theory*.

In optimal control theory, the dynamic optimization problem is viewed as consisting of three types of variables. Aside from the variable t and the state variable $y(t)$, consideration is given to a control variable $u(t)$. An optimum control problem must contain an equation that relates y to u :

$$\frac{dy}{dt} = f[t, y(t), u(t)]$$

Such an equation, called an *equation of motion* (or *transition equation* or *state equation*), shows how, at any moment of time, given the value of the state variable, the planner's choice of u will drive the state variable over time. Once we have found the optimal control-variable path $u^*(t)$, the equation of motion would make it possible to construct the related optimal state-variable path $y^*(t)$. So this approach can be represented by the formula:

$$\begin{array}{ll} \text{Maximize or minimize} & V[u] = \int_0^T F[t, y(t), u(t)] dt \\ \text{subject to} & y'(t) = f[t, y(t), u(t)] \\ & y(0) = A \quad (A \text{ given}) \\ \text{and} & y(T) = Z \quad (T, Z \text{ given}) \end{array} \quad (2.3)$$

Note that, in (2.3), not only does the objective functional contain u as an argument, but it has also been changed from $V[y]$ to $V[u]$. This reflects the fact that u is now the ultimate instrument of optimization.

The single most significant development in optimal control theory -a first order necessary condition- is known as the *maximum principle*. The powerfulness of that principle lies in its ability to deal directly with certain constraints on the control variable.

Specifically, it allows the study of problems where the admissible values of the control variable u are confined to some closed, bounded convex set U .

2.1.1 The Maximum Principle

In effect we have been led to construct the auxiliary or Hamiltonian function

$$H = u(k, x, t) + \lambda(t)f(k, x, t),$$

to compute its partial derivative with respect to x , and set that partial derivative equal to zero. This construction has substantial economic significance. If we imagine H to be multiplied by Δ , we can see that it is the sum of the total profits earned in the interval Δ plus the accrual of capital during the interval valued at its marginal value. $(H\Delta)$ is thus the total contribution of the activities that go on during the interval Δ , and the value of the capital accumulated during the interval. Naturally, then, the decision variable x during the current interval should be chosen so as to make H as great as possible. It is for this reason that the procedure we are describing is called the maximum principle.

We have also, in effect, computed the partial derivative of H with respect to k and equated that partial derivative to $-\dot{\lambda}$. The common sense of this operation can be seen best from a modified Hamiltonian,

$$\begin{aligned} H^* &= u(k, x, t) + \frac{d}{dt}\lambda k \\ &= u(k, x, t) + \lambda\dot{k} + \dot{\lambda}k. \end{aligned}$$

$(H^* \Delta)$ is the sum of the profits realized during an interval of length Δ and the increase in the value of the capital stock during the interval, or in a sense, the total contribution of activities during the interval to current and future profits. If we maximize H^* formally with respect to x and k we obtain:

$$\frac{\partial u}{\partial x} + \lambda \frac{\partial f}{\partial x} = 0,$$

$$\frac{\partial u}{\partial k} + \lambda \frac{\partial f}{\partial k} + \dot{\lambda} = 0,$$

There is one additional feature that has to be mentioned. This feature is the boundary conditions. To see how the boundary data effect the solution, we should consider how the three basic formulas operate:

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

$$\frac{\partial u}{\partial x} + \lambda \frac{\partial f}{\partial x} = 0,$$

$$\frac{\partial u}{\partial k} + \lambda \frac{\partial f}{\partial k} = -\dot{\lambda}.$$

The three formulas are conveniently written and remembered in terms of the Hamiltonian. In this form they are:

$$\frac{\partial H}{\partial \lambda} = \dot{k}$$

$$\frac{\partial H}{\partial x} = 0$$

$$\frac{\partial H}{\partial k} = -\dot{\lambda}$$

These three formulas jointly determine completely the time paths of the choice variable, the capital stock, and the value of capital. Now look at the second formula written out a bit more explicitly:

$$\frac{\partial}{\partial x} u(k, x, t) + \lambda(t) \frac{\partial}{\partial x} f(k, x, t) = 0.$$

With k and λ known, this formula determines the value of x , the choice variable. Putting this value into the first formula we obtain \dot{k} , the rate at which the value of a unit of capital is changing. Thus we know the capital stock and the value of a unit of capital a short time later. Using these new values, we can repeat our substitutions in the three formulas above and so find, in order, a new value of the change in the capital stock and a new rate for the change in the value of capital. Repeating this cycle over and over again, we can trace through the evolution of all the variables from time zero to time T .

In short, these three formulas working together determine the optimal paths of all the variables starting out from any given initial position. In other sense, then, the problem of the choice of an optimal path has been reduced to a much simpler problem, the problem of choosing an optimal initial value for the value of a unit of capital. This is not by any means an easy problem, but it is obviously a great deal easier than finding an entire optimal path without the aid of these formulas.

Same results above can be deduced from the more familiar method of maximizing subject to a finite number of constraints. The following typical problem of optimal control theory can be given as an example of this method.

2.2 The Typical Problem of Optimal Control Theory

In this part we want to demonstrate how to use Pontryagin's technique. This technique will provide us to solve the various dynamic models. Thus the typical problem explained in Barro and Sala-i-Martin (1995), that we want to solve takes the following form. The agent chooses or controls a number of variables, called *control variables*, so as to maximize an objective function subject to some constraints. These constraints are dynamic in that they describe the evolution of the state of the economy, as represented by a set of *state variables*, over time. The problem is given by:

$$\begin{aligned} \max_{c(t)} V(0) &= \int_0^T v[k(t), c(t), t] \cdot dt, \quad \text{subject to} \\ (a) \quad \dot{k}(t) &= g[k(t), c(t), t], \\ (b) \quad k(0) &= k_0 > 0, \quad \text{given,} \\ (c) \quad k(T) \cdot e^{-\bar{r}(T)T} &\geq 0 \end{aligned} \tag{2.4}$$

where $V(0)$ is the value of objective function as seen from the initial moment 0, $\bar{r}(t)$ is an average discount rate that applies between dates 0 and t , and T is the terminal planning date, which could be finite or infinite. The variable $k(t)$ is the *state variable* and the variable $c(t)$ is the control variable. Each of these variables are functions of time. The objective function in (2.4) is the integral of the functions, $v(\bullet)$, over the interval from 0 to T . These functions depend, in turn, on the state and control variables, $k(t)$ and $c(t)$, and on time, t . The accumulation constraint is a differential equation in $k(t)$; this constraint shows how the choice of the control variable, $c(t)$, translates into a pattern of movement for the state variable, $k(t)$. The equation for $\dot{k}(t)$ is called the *transition equation* or *equation of motion*.

The initial condition in Eq. (2.4) says that the state variable, $k(t)$, begins at a given value, k_0 . The final constraint, in Eq.(2.4), says that the chosen value of the state variable at the end of planning horizon, $k(T)$, discounted at the rate $\bar{r}(T)$ must be nonnegative. For finite values of T , this constraint implies $k(T) \geq 0$, as long as the discount rate $\bar{r}(T)$ is positive and finite.

An economic example of this kind is a growth model in which $v(\bullet)$ is an instantaneous utility function that depends on the level of consumption and is discounted by a time preference factor,

$$v(k, c, t) = e^{-\rho t} \cdot u[c(t)].$$

In this example $v(\bullet)$ does not depend on the capital stock, $k(t)$, and depends directly on time only through the discount factor, $e^{-\rho t}$.

2.3 Heuristic Derivation of the First-Order Conditions

The starting point is the static method for solving nonlinear optimization problems, the Kuhn-Tucker Theorem. This theorem suggests the construction of a Lagrangian of the form,

$$L = \int_0^T v[k(t), c(t), t] \cdot dt + \int_0^T \left\{ \lambda(t) \cdot \left(g[k(t), c(t), t] - \dot{k}(t) \right) \right\} \cdot dt + \lambda_0 \cdot k(T) \cdot e^{-r(T)T}, \quad (2.5)$$

where $\lambda(t)$ is the Lagrange multiplier associated with the constraint in Eq. (2.4a), and λ_0 is the multiplier associated with the constraint in Eq. (2.4c). Since there is a continuum of constraints (a), one for each instant t between 0 and T , there is a corresponding continuum of Lagrange multipliers, $\lambda(t)$.

The $\lambda(t)$ and λ_0 are called *costate variables* or *dynamic Lagrange multipliers*. These costate variables can be interpreted as shadow prices: $\lambda(t)$ is the price or value of an extra unit of capital stock at time t in units of utility at time 0. Since each constraints, $g(\bullet) - \dot{k}$, equals 0, each of the products, $\lambda_1(t) \cdot [g(\bullet) - \dot{k}]$, also equals 0. It follows that the sum of all the constraints equals 0;

$$\int_0^T \{ \lambda(t) \cdot (g[k(t), c(t), t] - \dot{k}(t)) \} dt = 0.$$

To find the set of first order necessary conditions in a static problem, we would maximize L with respect to $c(t)$ and $k(t)$ for all t between 0 and T . The problem with this procedure is that we do not know to take the derivative of \dot{k} with respect to k . To avoid this problem, we can rewrite the Lagrangian by integrating the term $\lambda(t) \cdot \dot{k}(t)$ by parts to get:

$$\begin{aligned} L = & \int_0^T (v[k(t), c(t), t] + \lambda(t) \cdot g[k(t), c(t), t]) \cdot dt + \int_0^T \dot{\lambda}(t) \cdot k(t) \cdot dt \\ & + \lambda(0) \cdot k_0 - \lambda(T) \cdot k(T) + \lambda_0 \cdot k(T) \cdot e^{-r(T) \cdot T} \end{aligned} \quad (2.6)$$

The expression inside the first integral is referred to as the *Hamiltonian* function,

$$H(k, c, t, \lambda) \equiv v(k, c, t) + \lambda \cdot g(k, c, t) \quad (2.7)$$

The Hamiltonian function has an economic interpretation. At instant in time, the agent consumes $c(t)$ and owns a stock of capital $k(t)$. These two variables affect utility through two channels. First, the direct contribution of consumption, and perhaps capital, to utility, is captured by the term $v(\bullet)$ in Eq. (2.7). Second, the choice of consumption affects the change in the capital stock in accordance with the transition equation for \dot{k} in Eq. (2.4a).

The value of this change in the capital stock is the term $\lambda \cdot g(k, c, t)$ in Eq. (2.7). Hence, for a given value of the shadow price, λ , the Hamiltonian captures the total contribution to utility from the choice of $c(t)$.

Rewrite the Lagrangian from Eq.(2.6) as:

$$L = \int_0^T \left(H[k(t), c(t), t] + \dot{\lambda}(t) \cdot k(t) \right) dt + \lambda(0) \cdot k(0) - \lambda(T) \cdot k(T) + \lambda_0 \cdot k(T) \cdot e^{-r(T) \cdot T} \quad (2.7)$$

Let $\bar{c}(t)$ and $\bar{k}(t)$ be the optimal paths for the control and state variables, respectively. If we perturb the optimal path $\bar{c}(t)$ by an arbitrary perturbation function, $p_1(t)$, then we can generate a neighboring path for the control variable,

$$c(t) = \bar{c}(t) + \varepsilon \cdot p_1(t)$$

When $c(t)$ is thus perturbed, there must be a corresponding perturbation to $k(t)$ and $k(T)$ so as to satisfy the budget constraint

$$k(t) = \bar{k}(t) + \varepsilon \cdot p_2(t)$$

$$k(T) = \bar{k}(T) + \varepsilon \cdot dk(T).$$

If the initial paths are optimal, then $\partial L / \partial \varepsilon$ should equal zero. Before we compute such a derivative, it will be convenient to rewrite the Lagrangian in terms of ε :

$$\begin{aligned} \bar{L}(\cdot, \varepsilon) = & \int_0^T \left(H[k(\cdot, \varepsilon); c(\cdot, \varepsilon)] + \dot{\lambda}(\bullet) \cdot k(\cdot, \varepsilon) \right) dt \\ & + \lambda(0) \cdot k(0) - \lambda(T) \cdot k(T, \varepsilon) + \lambda_0 \cdot k(T, \varepsilon) \cdot e^{-r(T) \cdot T}. \end{aligned}$$

We now take the derivative of the Lagrangian with respect to ε and set it to zero:

$$\frac{\partial \bar{L}}{\partial \varepsilon} = \int_0^T \left[\left(\frac{\partial H}{\partial \varepsilon} \right) + \dot{\lambda} \cdot \left(\frac{\partial k}{\partial \varepsilon} \right) \right] \cdot dt + \left[\lambda_0 - \lambda(T) \right] \cdot \left(\frac{\partial k(T, \varepsilon)}{\partial \varepsilon} \right) = 0.$$

The chain rule of calculus implies $\frac{\partial H}{\partial \varepsilon} = \left[\frac{\partial H}{\partial c} \right] \cdot p_1(t) + \left[\frac{\partial H}{\partial k} \right] \cdot p_2(t)$ and $\frac{\partial k(T, \varepsilon)}{\partial \varepsilon} = dk(T)$. By using these formulas and rearranging terms in the expression $\frac{\partial \bar{L}}{\partial \varepsilon}$ to get:

$$\begin{aligned} \frac{\partial \bar{L}}{\partial \varepsilon} &= \int_0^T \left\{ \left[\frac{\partial H}{\partial c} \right] \cdot p_1(t) + \left[\frac{\partial H}{\partial k} + \dot{\lambda} \right] \cdot p_2(t) \right\} dt \\ &+ \left[\lambda_0 \cdot e^{-r(T) \cdot T} - \lambda(T) \right] \cdot dk(T) = 0. \end{aligned} \quad (2.8)$$

Equation (2.8) can hold for all perturbation paths, described by $p_1(t)$, $p_2(t)$ and $dk(T)$, only if each of the components in the equation vanishes, that is:

$$\frac{\partial H}{\partial c} = 0, \quad (2.9)$$

$$\frac{\partial H}{\partial k} + \dot{\lambda} = 0, \quad (2.10)$$

$$\lambda_0 \cdot e^{-r(T) \cdot T} = \lambda(T). \quad (2.11)$$

The first order condition with respect to the control variable in Eq.(2.9) says that if $\bar{c}(t)$ and $\bar{k}(t)$ are a solution to the dynamic problem, then the derivative of the Hamiltonian with respect to the control c equals 0 for all t . This result is called the *Maximum Principle*. Equation (2.10) says that the partial derivative of the Hamiltonian with respect to the state variable equals the negative of the derivative of the multiplier, $-\dot{\lambda}$. This result and the transition equation in (2.4a) are often called the Euler equations. Finally equation (2.11) says that the costate variable at the terminal date, λ equals λ_0 , the static Lagrange multiplier associated with the non-negativity constraint on k at the terminal date, discounted at the rate $\bar{r}(T)$.

2.4 Transversality Conditions

In the present problem, there is an inequality constraint that says that the stock of capital left at the end of the planning period, discounted at the rate $\bar{r}(T)$ cannot be negative, $k(T) \cdot e^{-\bar{r}(T)T} \geq 0$. The condition associated with this constraint $\lambda_0 \cdot k(T) \cdot e^{-\bar{r}(T)T} = 0$, with $\lambda_0 \geq 0$. Equation (2.11) implies that we can rewrite this condition as:

$$\lambda(T) \cdot k(T) = 0. \quad (2.12)$$

This boundary condition is often called the *transversality condition*. It says that if the quantity of capital left is positive, $k(T) > 0$, then its price must be zero, $\lambda(T) = 0$.

Alternatively, if capital at the terminal date is has the positive value, $\lambda(T) \geq 0$, then the agent must leave no capital, $k(T) = 0$.

2.5 The Behavior of the Hamiltonian over Time

To see how the optimal value Hamiltonian behaves over time, take the total derivative of H with respect to time to get:

$$dH(k, c, \lambda, t)/dt = [\partial H/\partial k] \cdot \dot{k} + [\partial H/\partial c] \cdot \dot{c} + [\partial H/\partial \lambda] \cdot \dot{\lambda} + \partial H/\partial t$$

The first order condition in Eq.(2.9) implies that at the optimum, $[\partial H/\partial c] = 0$; hence the second term on the right hand side of the above equation equals 0. Equation (2.10) requires $\partial H/\partial k = -\dot{\lambda}$. Since $\partial H/\partial \lambda = \dot{k}$, the first and the third terms on the right hand side of the above equation cancel.

Hence at the optimum, the total derivative of the Hamiltonian with respect to time equals the partial derivative, $\partial H/\partial t$. If the problem is autonomous -that is, if neither the objective function nor the constraints depend directly on time- then the derivative of the Hamiltonian with respect to time is zero. In other words, the Hamiltonian associated with autonomous problems is constant at all points in time.

2.6 Sufficient Conditions

2.6.1 The Mangasarian Sufficiency Theorem:

In static, nonlinear maximization problem, the Kuhn-Tucker necessary conditions are also sufficient when the objective function is concave and the restrictions generate a convex set, Mangasarian (1966) extends this result to dynamic problem and shows that if the functions $v(\bullet)$ and $g(\bullet)$ are both concave and k and c , then necessary conditions are also sufficient, Chiang (1992).

A basic sufficiency Theorem due to O. L. Mangasarian states that for the optimal control problem the necessary conditions of the maximum principle are also sufficient for the global maximization of $V[c]$, if both the $v(\bullet)$ and $g(\bullet)$ functions are differentiable and concave in the variables (k, c) jointly, and in the optimal solution it is true that $\lambda(t) \geq 0$ for all $t \in [0, T]$ if g is nonlinear in k or in c [if g is linear in k and in c , then $\lambda(t)$ needs no sign restriction.]

The optimal control path $c^*(t)$ -along with the associated $k^*(t)$ and $\lambda^*(t)$ paths- must satisfy the maximum principle, so that:

$$H(k, c, t, \lambda) \equiv v(k, c, t) + \lambda \cdot g(k, c, t)$$

$$\left. \frac{\partial H}{\partial c} \right|_{c^*} = v_c(k^*, c^*, t) + \lambda^* g_c(k^*, c^*, t) = 0$$

This implies that $v_c(k^*, c^*, t) = -\lambda^* g_c(k^*, c^*, t)$. Moreover, from the costate equation of motion, $\dot{\lambda} = -\partial H/\partial k$, we should have

$$\begin{aligned} \dot{\lambda}^* &= -v_k(k^*, c^*, t) - \lambda^* g_k(k^*, c^*, t) \\ \Rightarrow v_k(k^*, c^*, t) &= -\dot{\lambda}^* - \lambda^* g_k(k^*, c^*, t) \end{aligned}$$

Finally, assuming for the time being that the problem has a vertical terminal line, i.e; the terminal capital is free which means that $k^*(T) > 0$, which also means that the shadow price $\lambda^*(T)$ equals to zero; then the initial condition and the transversality condition should give us $k_0^* = k_0$ (given) and $\lambda^*(T) = 0$.

Now let both the $v(\bullet)$ and $g(\bullet)$ functions be concave in (k, c) . Then, for two distinct points (k^*, c^*, t) and (k, c, t) in the domain, we have:

$$\begin{aligned} v(k, c, t) - v(k^*, c^*, t) &\leq v_k(k^*, c^*, t)(k - k^*) + v_c(k^*, c^*, t)(c - c^*) \\ g(k, c, t) - g(k^*, c^*, t) &\leq g_k(k^*, c^*, t)(k - k^*) + g_c(k^*, c^*, t)(c - c^*) \end{aligned}$$

Upon integrating both sides of the first inequality above over $[0, T]$, that inequality becomes:

$$\begin{aligned} V - V^* &\leq \int_0^T [v_k(k^*, c^*, t)(k - k^*) + v_c(k^*, c^*, t)(c - c^*)] dt \\ &= \int_0^T [-\dot{\lambda}^*(k - k^*) - \lambda^* g_k(k^*, c^*, t)(k - k^*) - \lambda^* g_c(k^*, c^*, t)(c - c^*)] \cdot dt \end{aligned}$$

The first component of the last integral relating to the expression $-\dot{\lambda}^*(k - k^*)$, can be integrated by parts to yield:

$$\int_0^T -\dot{\lambda}^*(k - k^*) dt = \int_0^T \lambda^* [g(k, c, t) - g(k^*, c^*, t)] \cdot dt$$

This result enables us to write:

$$\begin{aligned} V - V^* &\leq \int_0^T \lambda^* [g(k, c, t) - g(k^*, c^*, t) - g_k(k^*, c^*, t)(k - k^*) - g_c(k^*, c^*, t)(c - c^*)] \cdot dt \\ &\leq 0 \end{aligned}$$

Consequently, the final result is $V \leq V^*$ which establishes V^* to be a (global) maximum. This theorem is based on the $v(\bullet)$ and $g(\bullet)$ functions being concave. If these functions are *strictly* concave, the weak inequalities will become strict inequalities. The maximum principle will then be sufficient for a *unique* global maximum of V .

Although the proof of the theorem has proceeded on the assumption of a vertical terminal line, the theorem is also valid for other problems with a fixed T .

2.6.2 The Arrow Sufficiency Theorem

Another sufficiency theorem, due to Kenneth J. Arrow, uses a weaker condition than Mangasarian's theorem, and can be considered as a generalization of the latter.

At any point of time, given the values of the state and costate variables k and λ , the Hamiltonian function is maximized by a particular c , c^* , which depends on t , k and λ $c^* = c^*(k, \lambda, t)$. When this expression is substituted into the Hamiltonian, we obtain what is referred to as the *maximized Hamiltonian function*

$$H^0(k, t, \lambda) \equiv v(k, t, \lambda^*) + \lambda \cdot g(k, t, \lambda^*)$$

The Arrow theorem states that the conditions of the maximum principle (necessary conditions) are sufficient for the global maximization of V , if the maximized Hamiltonian function $H^0(k, \lambda, t)$ is concave in the variable k for all t in the time interval $[0, T]$, for given λ . Concavity of $v(\bullet)$ and $g(\bullet)$ is sufficient, but not necessary, for the Arrow condition to be satisfied.

2.7 Infinite Horizon Problem

Most of the growth models involve economic agents with infinite planning horizons. The typical problem in Barro and Sala-i-Martin (1995) takes the form:

$$\begin{aligned} \max_{c(t)} V(c) &= \int_0^{\infty} v[k(t), c(t), t] \cdot dt, \quad \text{subject to} \\ (a) \quad \dot{k}(t) &= g[k(t), c(t), t], \\ (b) \quad k(0) &= k_0 > 0, \quad \text{given,} \\ (c) \quad \lim_{t \rightarrow \infty} [k(t) \cdot e^{-r(t)t}] &\geq 0 \end{aligned} \tag{2.13}$$

The first-order conditions for the infinite horizon problem are the same as those for the finite horizon case. The key difference is that the transversality condition, (2.12) applies not to a finite T , but to the limit as T tends to infinity. In other words the transversality condition is now:

$$\lim_{t \rightarrow \infty} [\lambda(t) \cdot k(t)] = 0 \tag{2.14}$$

The intuitive explanation for the new condition is that the value of the capital stock must be asymptotically 0, otherwise something valuable would be left over. If the quantity $k(t)$, remains positive asymptotically, then the price, $\lambda(t)$, must approach 0 asymptotically. If $k(t)$ grows forever at a positive rate then the price $\lambda(t)$ must approach 0 at a faster rate so that the product, $\lambda(T) \cdot k(T)$, goes to zero.

2.8 An Economic Interpretation of Optimal Control Theory

Capital theory is explained as the economics of time by Dorfman. Its task is to explain if, and why, a lasting instrument of production can be expected to contribute more to the value of output during its lifetime than it costs to produce or acquire. All this has changed abruptly in the past decade as a result of a revival of the calculus of variations. In its modern version, the calculus of variations is called optimal control theory. It has become the central tool of capital theory and has given the latter a new lease on life. As a result, capital theory has become so profoundly transformed that it has been rechristened growth theory, and has come to grips with numerous important practical and theoretical issues that previously could not even be formulated.

2.8.1 The Basic Equations

Consider the decision problem of a firm that wishes to maximize its total profits over some period of time. At any date t , this firm will have inherited a certain stock of capital and other conditions from its past behavior. Denote these by $k(t)$. With this stock of capital and other facilities k and at that particular date t , the firm is in a position to take some decisions which might concern rate of output, price of output, or product design. Denote the decisions taken at any date by $x(t)$. From the inherited stock of capital at the specified date together with the specified current decisions the firm derives a certain rate of benefits or net profits per unit of time. Denote this by $u(k(t), x(t), t)$. This function u determines the rate at which profits are being earned at time t as a result of having k and taking decisions x .

Now look at the situation as it appears at the initial date $t = 0$. The total profits that will be earned from then to some terminal date T are given by:

$$W(k_0, \bar{x}) = \int_0^T u(k, x, t) \cdot dt$$

which is simply the sum of the rate at which profit is earned at every instant and added up for all instants. This notation asserts that if the firm starts out with an initial amount of capital k_0 and then follows the decision policy denoted by \bar{x} , it will obtain a total result W , which is the integral of the results obtained at each instant. The firm is able to choose the limits of the time path of the decision variable \bar{x} but it cannot choose independently the amount of capital at each instant. This constraint is expressed by saying that the rate of change of the capital stock at any instant is a function of its present standing, the date, and the decisions taken. Symbolically:

$$\dot{k} = \frac{dk}{dt} = f(k, x, t)$$

Thus the decisions taken at any time have two effects. They influence the rate at which profits are earned at that time and they also influence the rate at which the capital stock that will be available at subsequent instants of time.

These two formulas express the essence of the problem of making decisions in a dynamic context. The problem is to select the time path symbolized by \bar{x} so as to make the total value of the result, W , as great as possible taking into account the effect of the choice of x on both the instantaneous rate of profit and the capital stock to be carried into the future. The strategy of the solution is to reduce the problem which, as it stands, requires us to find an entire time path, to a problem which demands us to determine only a single number. This transformation of the problem can be performed in a number of ways. One way, which dates back to the eighteenth century, leads to the classical calculus of variations. Another way leads to the maximum principle of optimal control theory. First, introduce a formula for the value that can be obtained by the firm starting at an arbitrary decision policy x until the terminal date. It is:

$$W(k_t, \bar{x}, t) = \int_t^T u(k, x, \tau) \cdot d\tau$$

Now break W up into two parts. Think of a short time interval of length Δ beginning at time t . Δ is to be thought of as being so short that the firm would not change x in the course of it even if it could. Then we can write:

$$W(k, \bar{x}, t) = u(k, x_t, t)\Delta + \int_{t+\Delta}^T u[k(t), x, \tau]d\tau.$$

This formula says that if the amount of capital at time t is k and if the policy denoted by \bar{x} is followed from then on, then the value contributed to the total sum from date t on consists of two parts. The first part is the contribution of a short interval that begins at date t . It is the rate at which profits are earned during the interval times the length of the interval. It depends on the current capital stock, the date, and the current value of the decision variable, here denoted by x_t . The second part is an integral of precisely the same form as before but beginning at date $t + \Delta$. It should be noticed that the starting capital stock for this last integral is not $k(t)$ but $k(t + \Delta)$. We can take the advantage of the fact that the same form of integral has returned by writing

$$W(k_t, \bar{x}, t) = u(k, x_t, t)\Delta + W(k_{t+\Delta}, \bar{x}, t + \Delta)$$

where the changes in the subscripts are carefully noted.

If the firm knew the best choice of \bar{x} from date t on, it could just follow it and thereby obtain a certain value. We denote this value, which results from the optimal choice of \bar{x} by V^* , as follows:

$$V^*(k_t, t) = \max W(k_t, \bar{x}, t).$$

Now suppose that the policy designated by x_t is followed in the short time interval from t to $t + \Delta$ and that thereafter the best possible policy is followed. The consequence of this peculiar policy can be written as:

$$V(k_t, x_t, t) = u(k_t, x_t, t)\Delta + V^*(k_{t+\Delta}, t + \Delta).$$

In words, the result of following such a policy are the benefits that accrue during the initial period using the decision x_t , plus the maximum possible profits that can be realized starting from date $t + \Delta$ with capital $k(t + \Delta)$ which results from the decision taken in the initial period.

Now we have arrived at the ordinary calculus problem of finding the best possible value for x_t . If the firm adopts this value, then V of the last formula will be equal to V^* . For the present we assume that the partial derivatives vanishes at the maximum, differentiate $V(k_t, x_t, t)$ with respect to x_t , and obtain

$$\Delta \frac{\partial}{\partial x_t} u(k_t, x_t, t) + \frac{\partial}{\partial x_t} V^*(k(t + \Delta), t + \Delta) = 0 \quad (2.15)$$

where,

$$\frac{\partial V^*}{\partial x_t} = \frac{\partial V^*}{\partial k(t + \Delta)} \frac{\partial k(t + \Delta)}{\partial x_t}.$$

To simplify this expression we can use the approximation

$$k(t + \Delta) = k(t) + \dot{k}\Delta.$$

That is, the amount of capital at $t + \Delta$ is equal to the amount of capital at t plus the rate of change of capital during the interval times the length of the interval.

Remembering the formula which \dot{k} depends on x_t :

$$\dot{k} = \frac{dk}{dt} = f(k, x, t)$$

Thus we can write

$$\frac{\partial k(t + \Delta)}{\partial x_t} = \Delta \frac{\partial f}{\partial x_t}$$

Turn, now, to the first factor, $\partial V^*/\partial k$. This derivative is the rate at which the maximum possible profit flow from time $t + \Delta$ on changes with respect to the amount of capital available at $t + \Delta$. It is, therefore, the marginal value of capital at time $t + \Delta$. We denote this marginal value of capital at time t by $\lambda(t)$, defined by:

$$\lambda(t) = \frac{\partial}{\partial k} V^*(k, t).$$

Inserting these results in formula (2.4), we obtain:

$$\Delta \frac{\partial u}{\partial x_t} + \lambda(t + \Delta) \Delta \frac{\partial f}{\partial x_t} = 0 \quad (2.16)$$

and furthermore, the constant Δ can be canceled out. The marginal value of capital changes over time and so, to a sufficiently good approximation,

$$\lambda(t + \Delta) = \lambda(t) + \dot{\lambda}(t) \Delta.$$

That is, the marginal value of capital at $t + \Delta$ is the marginal value at t plus the rate at which it is changing during the interval multiplied by the length of the interval. Insert this expression in equation (2.5), after canceling the common factor Δ in the equation as written, to obtain:

$$\frac{\partial u}{\partial x_t} + \lambda(t) \frac{\partial f}{\partial x_t} + \dot{\lambda}(t) \Delta \frac{\partial f}{\partial x_t} = 0.$$

Now allow Δ to approach zero. There results:

$$\frac{\partial u}{\partial x_t} + \lambda \frac{\partial f}{\partial x_t} = 0. \quad (2.17)$$

This is our first major result which makes perfectly good sense to an economist. It says that along the optimal path of the decision variable at any time the marginal short-run effect of a change in decision must just counter-balance the effect of that decision on the total value of the capital stock an instant later. We see that because the second term in the equation is the marginal effect of the current decision on the rate of growth of capital with capital valued at its marginal worth, λ . The firm should choose x at every moment so that the marginal immediate gain just equals the marginal long-run cost, which is measured by the value of capital multiplied by the effect of the decision on the accumulation of capital.

Now suppose that x_t is determined so as to satisfy equation (2.17). On the assumption that this procedure discovers the optimal value of x_t , $V(k_t, x_t, t)$ will then be equal to its maximum possible value or $V^*(k, t)$. Thus:

$$V^*(k, t) = u(k, x_t, t) \Delta + V^*(k(t + \Delta), t + \Delta).$$

Now differentiate this expression with respect to k . The derivative of the left-hand side is by definition $\lambda(t)$. The differentiation of the right-hand side is very similar to the work that we have already done and goes as follows:

$$\begin{aligned}
\lambda(t) &= \Delta \frac{\partial u}{\partial k} + \frac{\partial}{\partial k} V^*(k(t+\Delta), t+\Delta) \\
&= \Delta \frac{\partial u}{\partial k} + \frac{\partial k(t+\Delta)}{\partial k} \lambda(t+\Delta) \\
&= \Delta \frac{\partial u}{\partial k} + \left(1 + \Delta \frac{\partial f}{\partial k}\right) (\lambda + \dot{\lambda} \Delta) \\
&= \Delta \frac{\partial u}{\partial k} + \lambda + \Delta \lambda \frac{\partial f}{\partial k} + \Delta \dot{\lambda} + \dot{\lambda} \frac{\partial f}{\partial k} \Delta^2.
\end{aligned}$$

We can ignore the term in Δ^2 and make the obvious cancellations to obtain:

$$-\dot{\lambda} = \frac{\partial u}{\partial k} + \lambda \frac{\partial f}{\partial k}. \quad (2.18)$$

This is the second major formula of the maximum principle and possesses an illuminating economic interpretation.

To an economist, $\dot{\lambda}$ is the rate at which the capital is appreciating. $-\dot{\lambda}(t)$ is therefore the rate at which a unit of capital depreciates at time t . Accordingly the formula asserts that when the optimal time path of capital accumulation is followed, the decrease in value of a unit of capital in a short interval of time is the sum of its contribution to the profits realized during the interval and its contribution to enhancing the value of the capital stock at the end of the interval. In other words, a unit of capital loses value or depreciates as time passes at the rate at which its potential contribution to profits becomes its past contribution.

CHAPTER 3

THE NEOCLASSICAL THEORY OF OPTIMAL GROWTH AND RAMSEY'S ANALYSIS

3.1 The Neoclassical Growth Model

In this chapter, according to Barro and Sala-i-Martin (1995), we assume that the path of consumption and hence the saving rate are determined by optimizing households and firms that interact on competitive markets in closed economy. We deal, in particular, with infinitely-lived households that choose consumption and saving to maximize their dynastic utility, subject to an intertemporal budget constraint. This specification of consumer behavior is a key element in the Ramsey growth model as constructed by Ramsey(1928) and refined by Cass and Koopmans (1965).

The central question addressed by Ramsey is that how much of the national output at any point of time should be for current consumption to yield current utility, and how much should be saved (and invested) so as to enhance future production and consumption, and hence yield future utility? This important issue of intertemporal resource allocation has exerted a strong influence on economic thinking, although this influence did not come into being until after World War II, when that model rediscovered by growth theorists. In a more recent development, this basic issue is formulated as a problem of optimal control. Moreover, this new treatment -labeled as “the neoclassical theory of optimal growth”- extends the Ramsey model in two major respects:

(1) The labor force (identified with the population) is assumed to be growing at an exogenous constant rate $n > 0$ (the Ramsey model has $n = 0$), and (2) the social utility is assumed to be subject to time discounting at a constant rate $\rho > 0$ (the Ramsey model has $\rho = 0$).

The households in the economy provide labor services in exchange for wages, receive interest income on assets, purchase goods for consumption, and save by accumulating additional assets. Each household contains one or more adults, working members of the current generation. In making plans, these adults take account of the welfare and resources of their actual or prospective descendants. We model this intergenerational interaction by imagining that the current generation maximizes utility and incorporates a budget constraint over an infinite horizon. That is, although individuals have finite lives, we consider an immortal extended family. The current adults expect the size of their extended family to grow at the rate n because of the net influences of fertility and mortality. But at this point, we continue to simplify by treating n as exogenous (given) and constant. We also neglect the migration of persons. Thus under these assumptions we may formulate the family size at time t (which corresponds to the adult population) is :

$$L(t) = L(0).e^{nt}$$

If we normalize the number of adults at time 0 to unity, then the family size becomes $L(t) = e^{nt}$. If $C(t)$ is total consumption at time t , then $c(t) \equiv C(t) / L(t)$ is consumption per adult person. Thus if we denote the utility enjoyed by a person consuming at rate $c(t)$ by $u[c(t)]$, We assume that the social utility index function $u[c(t)]$ is assumed to possess the following properties:

$$u'[c(t)] > 0 \quad u''[c(t)] < 0 \quad \text{for all } c > 0$$

$$\lim_{c \rightarrow 0} u'[c(t)] = \infty \quad \text{and} \quad \lim_{c \rightarrow \infty} u'[c(t)] = 0$$

The total utility enjoyed by all the persons alive at time t with per capita consumption at rate $c(t)$ is $e^{nt} \cdot u[c(t)]$.

Let $\rho > 0$ be the social rate of time preference. Then the importance at time 0 of the consumption achieved at time t is $e^{-\rho t} \cdot e^{nt} \cdot u[c(t)]$. A positive value of ρ means that utils are valued less the later they are received. The other reason is that utils far in the future correspond to consumption of later generations. Suppose that, starting from a point at which the levels of consumption per person in each generation are the same, parents prefer a unit of their own consumption to their children's consumption. This parental "selfishness" corresponds to $\rho > 0$. Ramsey assumed $\rho = 0$. He then interpreted the optimizing agent as a social planner, rather than a competitive household, who choose consumption and saving for today's generation as well as future generations. The discounting of utility for future generations ($\rho > 0$) was, according to Ramsey, "ethically indefensible". It is also plausible that parents would have diminishing utility with respect to the number of children. We could model this effect by allowing the rate of time preference, ρ , to increase with the population growth rate, n .

Then the defensible objective for a society with infinite time horizon is to maximize:

$$U = \int_0^{\infty} u[c(t)] \cdot e^{nt} \cdot e^{-\rho t} \cdot dt \quad (3.1)$$

This formulation assumes that the household's utility at time 0 is a weighted sum of all future flows of utility, $u[c(t)]$. We also assume $\rho > n$, which implies that U is bounded if $c(t)$ is constant over time.

The analytic framework of the neoclassical theory revolves around the neoclassical production function $Y(t) = Y[K(t), L(t)]$, assumed to be characterized by constant returns to scale, positive marginal products, and diminishing returns to each input.

Such a production function, being linearly homogenous, can be rewritten in per capita terms. Let $k(t) = K(t)/L(t)$ denote capital per capita and $y(t) = Y(t)/L(t)$ denote the output per capita. We can express the production function by:

$$y(t) = f(k(t)),$$

$$f(k(t)) > 0, \quad f'(k(t)) > 0, \quad f''(k(t)) < 0 \text{ for } k > 0,$$

$$\lim_{k \rightarrow 0} f'(k(t)) = \infty, \quad \lim_{k \rightarrow \infty} f'(k(t)) = 0,$$

By virtue of constant returns to scale, we can write the production function of the economy as:

$$Y(t) = L(t) \cdot f(k(t)) = e^{\mu t} \cdot f(k(t))$$

Gross investment equals output minus consumption, or $Y(t) - L(t) \cdot c(t)$. Net investment equals gross investment minus physical depreciation. Suppose that physical capital deteriorates the rate δ so that the total rate of decay of the physical stock, when it is $K(t)$, is $\delta \cdot K(t)$. Then net capital accumulation is:

$$\begin{aligned} \dot{K}(t) &= Y(t) - L(t) \cdot c(t) - \delta \cdot K(t) = L(t)[f(k(t)) - c(t)] - \delta \cdot K(t) \\ &= L(t)[f(k(t)) - c(t)] - \delta \cdot L(t) \cdot k(t) \\ &= L(t)[f(k(t)) - c(t) - \delta \cdot k(t)]. \end{aligned} \tag{I}$$

Finally, eliminate $\dot{K}(t)$ by noticing:

$$\begin{aligned} \dot{k}(t) &= \frac{d}{dt} \frac{K(t)}{L(t)} = \frac{K(t)}{L(t)} \left(\frac{\dot{K}(t)}{K(t)} - \frac{\dot{L}(t)}{L(t)} \right) \\ &= k(t) \left(\frac{\dot{K}(t)}{L(t) \cdot k(t)} - n \right) \end{aligned}$$

$$\begin{aligned} \Rightarrow \dot{k}(t) + k(t).n &= \frac{\dot{K}(t)}{L(t)} \\ \Rightarrow \dot{K}(t) &= L(t).[\dot{k}(t) + k(t).n] \end{aligned} \quad (II)$$

So by equating the two equations (I) and (II), we obtain:

$$\dot{k}(t) = f(k(t)) - c(t) - (n + \delta).k(t) \quad (3.2)$$

The problem of optimal growth is thus simply to

$$\begin{aligned} \text{Maximize} \quad U &= \int_0^{\infty} u[c(t)] \cdot e^{(n-\rho)t} \cdot dt \\ \text{subject to} \quad \dot{k}(t) &= f(k(t)) - c(t) - (n + \delta).k(t) \\ k(0) &= k_0 \\ \text{and} \quad 0 &\leq c(t) \leq f[k(t)] \end{aligned} \quad (3.3)$$

3.2 The Maximum Principle

Hamiltonian for this problem is,

$$H = u[c(t)].e^{(n-\rho)t} + \lambda.[f(k(c(t))) - c(t) - (n + \delta).k(t)] \quad (3.4)$$

We can accordingly find the maximum of H by setting

$$\frac{\partial H}{\partial c} = u'(c).e^{(n-\rho)t} - \lambda = 0 \quad (I)$$

$$\frac{\partial H}{\partial k} = \lambda.[f'(k(c)) - (n + \delta)] = -\dot{\lambda} \quad (II)$$

$$\frac{\partial H}{\partial \lambda} = f(k(c)) - c - (n + \delta).k(c) = \dot{k} \quad (III)$$

This three equations, in principle enable us to solve for the three variables c , λ , and k .

From the equation (I), we obtain the condition:

$$u'[c(t)] = \lambda \cdot e^{(\rho-n)t}$$

which states that, optimally, the marginal utility per capita consumption should be equal to the shadow price of capital amplified by the exponential form. In other words, the value of a unit of capital at time t is the marginal utility of consumption at that time, adjusted for population growth and the social rate of time preference.

Now take the logarithm of both sides of equation (I) with respect to base e :

$$\begin{aligned} \ln e^{(n-\rho)t} + \ln u'(c) &= \ln \lambda \\ \Rightarrow (n-\rho)t + \ln u'(c) &= \ln \lambda, \quad \text{taking the derivative of both sides with respect to } t: \\ \Rightarrow (n-\rho) + \frac{u''(c)}{u'(c)} \cdot \dot{c} &= \frac{\dot{\lambda}}{\lambda}, \end{aligned}$$

From equation (II), we obtain $\frac{\dot{\lambda}}{\lambda} = (n+\delta) - f'(k(c))$, so by equating this equation with the result of the above derivation, we can easily get the following equation:

$$f'(k(c)) = \rho + \delta - \frac{u''(c)}{u'(c)} \cdot \dot{c}$$

which asserts that along the optimal path the rate of consumption at each moment must be chosen so that the marginal productivity of capital is the sum of three components:

- (1) ρ , the social rate of time preference,
- (2) δ , the rate of physical deterioration of capital, and
- (3) the rather formidable looking third term which, however, is simply the percentage rate at which the psychic cost of saving at any time is $u'(c)$, its time rate of change is $u''(c) \cdot \dot{c}$, and its percentage time rate of change is the negative of the third term in the sum.

In other words, along the optimum path of accumulation the marginal contribution of a unit of capital to output during any short interval of time must be just sufficient to cover the three components of the social cost of possessing that a unit of capital, namely, the social rate of time preference, the rate of physical deterioration of capital, and the additional psychic cost of saving a unit at the beginning of the interval rather than at the end. All of these are expressed as percents per unit of time, which is also the dimension of the marginal productivity of capital.

Rewriting the same equation,

$$\dot{c} = -\frac{u'(c)}{u''(c)} \cdot [f'(k(c)) - (\rho + \delta)] \quad (3.5)$$

which is a differential equation in the variable c . Consequently, we now can work with the differential equation system

$$\begin{aligned} \dot{k}(t) &= f(k(t)) - c(t) - (n + \delta) \cdot k(t) \\ \dot{c}(t) &= -\frac{u'(c(t))}{u''(c(t))} \cdot [f'(k(t)) - (\rho + \delta)] \end{aligned} \quad (3.6)$$

If we divide both sides of the equality in the second equation above by c , then we get the following equation:

$$\frac{\dot{c}(t)}{c(t)} = -\frac{1}{c(t)} \cdot \frac{u'(c(t))}{u''(c(t))} \cdot [f'(k(t)) - (\rho + \delta)]$$

The term, $-\frac{1}{c(t)} \cdot \frac{u'(c(t))}{u''(c(t))}$, on the right-hand side of the above equation is called the elasticity of intertemporal substitution. The intertemporal substitution between consumption at times t_1 and t_2 is given by the reciprocal of the magnitude of the slope of an indifference curve in response to a proportionate change in the ratio $c(t_1) / c(t_2)$. If we denote this elasticity by σ , then we get

$$\sigma = \left[\frac{c(t_1) / c(t_2)}{-u'[c(t_1)] / u'[c(t_2)]} \cdot \frac{d\{u'[c(t_1)] / u'[c(t_2)]\}}{d[c(t_1) / c(t_2)]} \right]^{-1}$$

where $-u'[c(t_1)] / u'[c(t_2)]$ is the magnitude of the indifference curve. If we let t_2 approach t_1 , then we get the instantaneous elasticity,

$$\sigma = -u'[c(t)] / [c(t) \cdot u''[c(t)]]$$

which is the inverse of the magnitude of the elasticity of marginal utility. We therefore follow the assumption of the functional form

$$u[c(t)] = \frac{c(t)^{(1-\theta)} - 1}{(1-\theta)},$$

where $\theta > 0$, so that the elasticity of marginal utility equals the constant $-\theta$. The elasticity of substitution for this utility function is constant $\sigma = 1/\theta$. Hence this form is called the *constant intertemporal elasticity of substitution* (CIES) utility function. The higher θ , the more rapid is the proportionate decline in $u'[c(t)]$ in response to increases in c and, hence, the less willing households are to accept deviations from a uniform pattern of c over time.

As θ approaches 0, the utility function approaches a linear form in c . So that we can write our second equation in (3.6) as:

$$\frac{\dot{c}(t)}{c(t)} = \frac{1}{\theta} \cdot [f'(k(t)) - (\rho + \delta)] \quad (3.7)$$

However, our analysis has not finished yet because now in the above equation we look the term $f'(k(t))$ under microscope such as this term explains the marginal productivity of the representative firm.

The representative firm's flow of net receipts or profit at any point in time is given by

$$profit = F(K, L) - (r + \delta) \cdot K - w \cdot L$$

where r denotes the rental price of the capital and w denotes the wages of the workers. We assume that the firm seeks to maximize the present value of profits. Because the firm rents capital and labor services and has no adjustment costs, there are no intertemporal elements in the firm's maximization problem. That is, the problem of maximizing the present value of profits reduces here to a problem of maximizing profits in each period without regard to the outcomes in other periods.

A competitive firm, which takes r and w as given, maximizes profit by setting

$$f'(k(t)) = r + \delta$$

That is, the firm chooses the ratio of capital to effective labor to equate the marginal product of capital to the rental price.

This analysis makes us to conclude that the equation (3.6) can be written as:

$$\frac{\dot{c}(t)}{c(t)} = \frac{1}{\theta} \cdot [f'(k(t)) - (\rho + \delta)] = \frac{1}{\theta} \cdot (r - \rho).$$

Therefore, the relation between r and ρ determines whether households choose a pattern of per capita consumption that rises over time, stays constant, or falls over time. A lower willingness to substitute intertemporally (a higher value of θ) implies a smaller responsiveness of \dot{c}/c to the gap r and ρ .

3.3 Phase Diagram Analysis

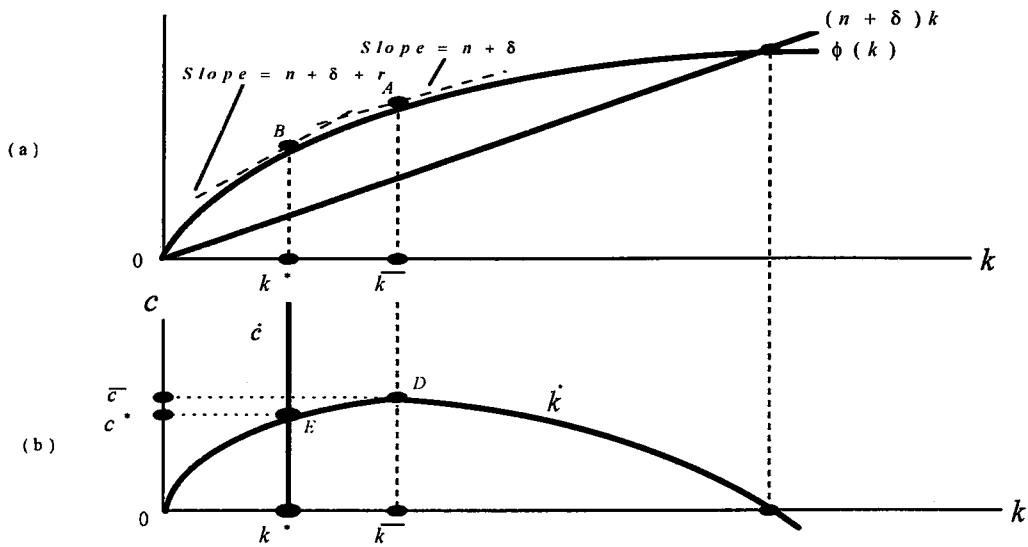
To construct the phase diagram, Chiang (1992), we first draw the $\dot{k} = 0$ curve and the $\dot{c} = 0$ curve. These are defined by the two equations

$$(I) \quad c(t) = f(k(t)) - (n + \delta) \cdot k(t) \quad [Equation \text{ for } \dot{k} = 0 \text{ curve}]$$

$$(II) \quad f'(k(t)) = \rho + \delta \quad [Equation \text{ for } \dot{c} = 0 \text{ curve}]$$

The $\dot{k} = 0$ curve shows up in the kc space as a concave curve, as illustrated in Fig. (3.1b). As the equation (I) above indicates, this curve expresses c as the difference between two functions of k : $f(k(t))$ and $(n + \delta) \cdot k(t)$. Plotting the difference between these two curves then yields the desired $\dot{k} = 0$ curve in Fig.(3.1b). It has the shape drawn because of the conventional assumptions that the marginal productivity of capital is positive but diminishing, and the very plausible assumption that for every low levels of capital per worker, $f'(k(t)) > n + \delta$. We also assume that no output is possible without some capital, i.e., $f(0) = 0$. As to the $\dot{c} = 0$ curve the equation (II) above requires that the slope of the $f(k(t))$ curve is the specific value $\rho + \delta$. The $f(k(t))$ curve being monotonic, this requirement can be satisfied only at a single point on that curve, point B, which corresponds to a unique k value k^* . Thus the $\dot{c} = 0$ curve must plot as a vertical straight line in Fig.(3.1b), with horizontal intercept k^* .

Fig.(3.1)



At the same time, from the equation (I) above, we can obtain the highest possible steady-state c is attained such that:

$$\frac{\partial c}{\partial k} = f'(k(t)) - (n + \delta) = 0$$

that is, when

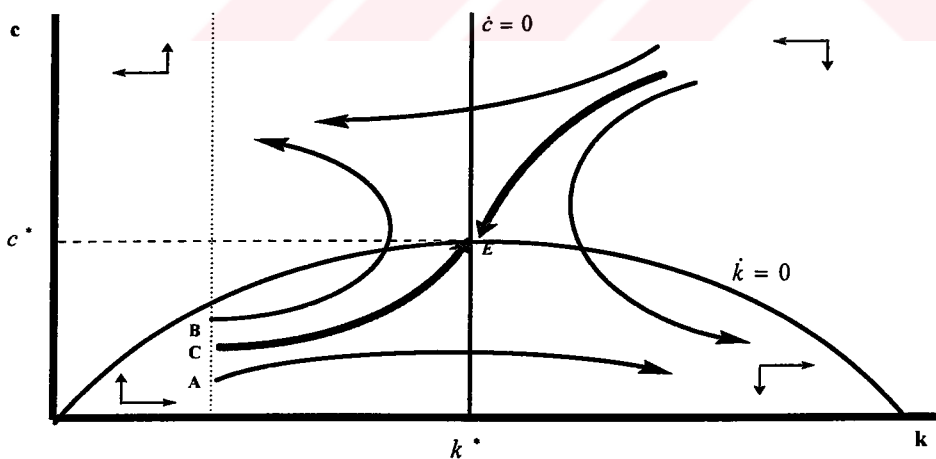
$$f'(\bar{k}(t)) = (n + \delta).$$

This given condition is known as the *golden rule of capital accumulation*. Since $f(k(t))$ is monotonic, there is only one value of k that will satisfy this condition. Let that particular value of k be denoted by \bar{k} , and let the corresponding *golden-rule* value of c denoted by \bar{c} . Then $\bar{c} = f(\bar{k}(t)) - (n + \delta)\bar{k}$. This value of c represents the maximum sustainable per-capita consumption stream, since it is derived from the maximization condition $dc/dk=0$.

The intersection of the two curves in Fig.(3.1b), at point E determines the steady-state values of k and c . Denoted by k^* and c^* respectively, these values are referred to in the literature as the *modified golden rule* values of capital-labor ratio and per-capita consumption, as distinct from the golden rule values \bar{k} and \bar{c} . Since k^* is defined by $f'(k^*) = n + \rho$, which involves a larger slope of $f(k(t))$ than the golden-rule case. Since the golden rule value is the maximum attainable consumption value, then the other values for consumption must be less than this value and from the assumption that the marginal productivity of capital is positive but diminishing, and the very plausible assumption that for every low levels of capital per worker, $f'(k(t)) > n + \delta$, the discount rate should be greater than the depreciation rate, i.e., $\rho > \delta$.

Redrawing the fig.(3.1b) more specifically we can analyze the saddle-path more easily. To prepare the analysis of the phase diagram, we draw the streamlines. These streamlines must cross the $\dot{k} = 0$ curve with infinite slope and cross the $\dot{c} = 0$ curve with zero slope.

Fig.(3.2)



If consumption per capita is less than the rate on the locus, capital per capita increases ($\dot{k} > 0$). Above the locus ($\dot{k} < 0$). If we accept the usual assumptions of positive but diminishing marginal utility $u'[c(t)] > 0$, $u''[c(t)] < 0$. Then $\dot{c} > 0$, i.e., per capita consumption grows, to the left to this line. The reason is that the low levels of capital per capita the amount of depreciation is small and the amount of capital needed to equip the increment in population with the current level of capital is also small. For clues about the general directions of the streamlines, we partially differentiate the two differential equations in (3.6), we find that

$$\frac{\partial \dot{k}}{\partial c} = -1 < 0$$

$$\frac{\partial \dot{c}}{\partial k} = -\frac{u'(c(t))}{u''(c(t))} \cdot f''(k(t)) < 0$$

According to the first equation above, as c increases (going northward), \dot{k} should follow the (+,0,-) sign sequence. So, the k -arrowheads must point eastward below the $\dot{k} = 0$ curve, and westward above it. Similarly the second equation indicates that \dot{c} should follow the (+,0,-) sign sequence as k increases (going eastward). Hence, the c -arrowheads should point upward to the left of the $\dot{c} = 0$ curve, and downward to the right of it.

These considerations enable us to depict qualitatively the laws of motion of the system. The entire evolution of the system is determined by the choice of the initial level of per capita consumption. If a low initial level is chosen, such as point A in the Fig.(3.2), both consumption and capital per capita will increase for some time, following the curved arrow that emanates from point A . But when the level of capital per capita reaches the critical value, consumption per capita will start to fall though capital per capita will continue to increase. This is a policy of initial generosity in consumption followed by increasing abstemiousness intended, presumably to attain some desired level of capital per capita.

Similarly, the path emanating from point B represents a policy of continually increasing consumption per capita, with capital initially being accumulated and eventually being consumed.

The path originating at point C is of particular interest of this theses. It leads to the intersection of the two critical loci, the steady state of the system in which neither per capita consumption nor per capita income changes. Once at this point all the absolute values grow exponentially at the common rate n .

It is now seen that if the initial capital per capita is given, the entire course of the economy is determined by the choice of the initial level of per capita consumption. This choice determines, among other things, the amount of capital per capita at any specified date. If the conditions of the problem prescribe a particular amount of capital at some date, the initial c must be the one with a path that leads to the specified point. If there is no such prescription for capital accumulation, the initial c will be the one that causes the capital stock to be exhausted at the terminal date under consideration. So that the only possible solution is the path that originates at point C and terminates at the point where $\dot{c} = \dot{k} = 0$.

3.4 Transversality Conditions

To select a stable branch from the family of streamlines is tantamount to choosing a particular solution from a family of general solutions by definitizing an arbitrary constants. This is to be done as explained above, with the help of some boundary conditions. The requirement that we choose a specific initial c_0 such that the ordered pair (k_0, c_0) sits on the stable branch is one way of doing it. An alternative is to look at an appropriate transversality condition.

One transversality condition we may expect the steady state solution to satisfy is that

$$\lambda \rightarrow 0 \quad \text{as} \quad t \rightarrow \infty$$

This is because the objective functional does contain a discount factor and the terminal the terminal state is free. Since the solution path for λ in the maximum principle is

$$\lambda^* = u'[c^*(t)].e^{(n-\rho)t}$$

and since the limit of $u'[c^*(t)]$ is finite as $t \rightarrow \infty$, this expression satisfy the transversality condition..

Another condition we may expect is that in the solution,

$$H \rightarrow 0 \quad \text{as} \quad t \rightarrow \infty$$

For the present problem, the solution path for H takes the form

$$H^* = u[c^*(t)].e^{(n-\rho)t} + \lambda^*.[f(k^*(t) - c^*(t) - (n + \delta)k^*(t))]$$

Since $u[c^*(t)]$ is finite as $t \rightarrow \infty$, the exponential term $u[c^*(t)].e^{(n-\rho)t}$ tends to zero as t becomes infinite. In the remaining term, we already know $\lambda^* \rightarrow 0$ as $t \rightarrow \infty$. Therefore, this second transversality condition is also satisfied.

3.5 Checking The Saddle Point by Characteristic Roots

On the basis of the two-equation system

$$\dot{k}(t) = f(k(t)) - c(t) - (n + \delta).k(t)$$

$$\dot{c}(t) = -\frac{u'(c(t))}{u''(c(t))} \cdot [f'(k(t)) - (\rho + \delta)]$$

we first form the Jacobian matrix and evaluate it at the steady-state point E .

$$J_E = \begin{bmatrix} \frac{\partial \dot{k}}{\partial k} & \frac{\partial \dot{k}}{\partial c} \\ \frac{\partial \dot{c}}{\partial k} & \frac{\partial \dot{c}}{\partial c} \end{bmatrix}_{(k^*, c^*)}$$

The four partial derivatives, when evaluated at E , where $f'(k^*) = \delta + \rho$, turn out to be

$$\left. \frac{\partial \dot{k}}{\partial k} \right|_E = f'(k^*(t)) - (n + \delta) > 0$$

$$\left. \frac{\partial \dot{k}}{\partial c} \right|_E = -1 < 0$$

$$\left. \frac{\partial \dot{c}}{\partial k} \right|_E = -\frac{u'(c(t))}{u''(c(t))} \cdot f''(k^*(t)) < 0$$

$$\left. \frac{\partial \dot{c}}{\partial c} \right|_E = \frac{-[u''(c^*(t))]^2 + u'''(c^*(t)) \cdot u'(c^*(t))}{[u''(c^*(t))]^2} [f'(k^*(t)) - (\delta + \rho)] = 0$$

It follows that the Jacobian matrix takes the form

$$J_E = \begin{bmatrix} \rho - n & -1 \\ -\frac{u'(c^*)}{u''(c^*)} \cdot f''(k^*) & 0 \end{bmatrix}$$

The qualitative information we need about the characteristic roots r_1 and r_2 to confirm the saddle point is conveyed by the result is that

$$r_1 \cdot r_2 = |J_E| = -\frac{u'(c^*(t))}{u''(c^*(t))} \cdot f''(k^*(t)) < 0$$

This implies that the two roots have opposite signs, which establishes the steady state to be locally a saddle point. The saddle path property can also be verified by linearizing the system of dynamic equations around the steady state and noting that the determinant of the characteristic matrix is negative. This sign for the determinant implies that the two eigenvalues have opposite signs, an indication that the system is locally saddle-path stable.

3.6 The Shape of the Stable Arm

The stable arm expresses the equilibrium c as a function of k . This relation is known in dynamic programming as a *policy function*: it relates the optimal value of a control variable, c , to the state variable k . This policy function is an upward sloping curve that goes through the origin and the steady state position. Its exact shape depends on the parameters of the model.

Consider, as an example, the effects of the parameter θ on the shape of the stable arm. Suppose that the economy begins with $k_0 < k^*$, so that future values of c will exceed c_0 . High values of θ indicate that households have a strong preference for smoothing consumption over time; hence, they will try hard to shift consumption from the future to the present. Therefore, when θ is high, the stable arm will lie close to the $\dot{k} = 0$ curve.

Conversely, if θ is low, then households are more willing to postpone consumption in response to high rates of return. The stable arm in this case is flat and close to the horizontal axis for low values of k .

3.7 Typical Example of Solving The Stable Arm Equation by Using The Method of Taylor's Approximation in the Cobb-Douglas Production Function Case

From the equations (3.6) and (3.7) we can rewrite our differential equation system where the production function is the Cobb-Douglas production function which can be defined as $F[K(t), L(t)] = K(t)^\alpha \cdot L(t)^{1-\alpha}$. When we divide both sides of this equation by $L(t)$, we get the per capita equation $f(k(t)) = k(t)^\alpha$. Then our system will be as follows:

$$\dot{k} = f(k, c) = k^\alpha - (n + \delta)k - c \quad (3.8)$$

$$\dot{c} = g(k, c) = \frac{c}{\theta} (\alpha \cdot k^{\alpha-1} - \rho - \delta) \quad (3.9)$$

The steady-state values for c and k are determined by setting the expressions in equations (3.8) and (3.9) to zero. The equilibrium steady state values for c and k are

$$k^* = \left(\frac{\alpha}{\rho + \alpha} \right)^{\frac{1}{1-\alpha}} \quad (3.10a)$$

$$c^* = \left(\frac{\rho + \delta}{\alpha} - (n + \delta) \right) \cdot \left(\frac{\alpha}{\rho + \delta} \right)^{\frac{1}{1-\alpha}} \quad (3.10b)$$

The system defined in equations (3.8) and (3.9) is highly nonlinear. Therefore, it is very difficult to solve this system. However one of the suggestions made in the literature is to linearize the system of dynamic equations by Taylor's expansion around the steady state

Then we can expand the equation (3.8) around the equilibrium points as follows:

$$f(k, c) = f(k^*, c^*) + f_k(k^*, c^*)(k - k^*) + f_c(k^*, c^*)(c - c^*) \quad (3.11)$$

$$\begin{aligned}
\text{(I)} \Rightarrow f(k^*, c^*) &= \left(\frac{\alpha}{\rho+\delta}\right)^{\frac{\alpha}{1-\alpha}} - (n+\rho)\left(\frac{\alpha}{\rho+\delta}\right)^{\frac{1}{1-\alpha}} - \left[\frac{\rho+\delta}{\alpha} - (n+\delta)\right] \left[\frac{\alpha}{\rho+\alpha}\right]^{\frac{1}{1-\alpha}} \\
&= \left(\frac{\alpha}{\rho+\delta}\right)^{\frac{1}{1-\alpha}} \left\{ \left(\frac{\alpha}{\rho+\delta}\right)^{\frac{2\alpha-1}{1-\alpha}} - \left(\frac{\alpha}{\rho+\alpha}\right)^{-1} \right\}
\end{aligned}$$

$$\begin{aligned}
\text{(II)} \Rightarrow f_k(k^*, c^*) &= \alpha \cdot (k^*)^{\alpha-1} - (n+\delta) \\
&= \left[\alpha \left(\frac{\alpha}{\rho+\delta}\right)^{-1} - (n+\delta) \right] = \rho - n
\end{aligned}$$

$$\text{(III)} \Rightarrow f_c(k^*, c^*) = -1$$

by substituting the above equations (I), (II), and (III) into equation (3.11); we obtain:

$$\begin{aligned}
f(k, c) &= \left[\frac{\alpha}{\rho+\delta}\right]^{\frac{1}{1-\alpha}} \left\{ \left(\frac{\alpha}{\rho+\delta}\right)^{\frac{2\alpha-1}{1-\alpha}} - \left(\frac{\rho+\delta}{\alpha}\right) \right\} + (\rho-n) \left[k - \left(\frac{\alpha}{\rho+\delta}\right)^{\frac{1}{1-\alpha}} \right] - \left\{ c - \left[\frac{\rho+\delta}{\alpha} - (n+\delta)\right] \cdot \left[\frac{\alpha}{\rho+\delta}\right]^{\frac{1}{1-\alpha}} \right\} \\
&= \left[\frac{\alpha}{\rho+\delta}\right]^{\frac{2\alpha}{1-\alpha}} - \left[\frac{\alpha}{\rho+\delta}\right]^{\frac{\alpha}{1-\alpha}} + (\rho-n) \cdot k - (\rho-n) \left[\frac{\alpha}{\rho+\delta}\right]^{\frac{1}{1-\alpha}} - c + \left[\frac{\rho+\delta}{\alpha} - (n+\delta)\right] \cdot \left[\frac{\alpha}{\rho+\delta}\right]^{\frac{1}{1-\alpha}} \\
\Rightarrow \dot{k} &= \left\{ \left[\frac{\alpha}{\rho+\delta}\right]^{\frac{2\alpha}{1-\alpha}} - \left[\frac{\alpha}{\rho+\delta}\right]^{\frac{\alpha}{1-\alpha}} + \left[\frac{1-\alpha}{\alpha} \cdot (\rho+\delta)\right] \left[\frac{\alpha}{\rho+\delta}\right]^{\frac{1}{1-\alpha}} \right\} + (\rho-n) \cdot k - c
\end{aligned}$$

So, we have linearized the first differential equation of our system. We can write the above equation in simpler form as:

$$\dot{k} = A_0 + (\rho - n) \cdot k - c \quad \text{where } A_0 \text{ is a constant term}$$

Now we are going to apply the same method to the second differential equation (3.9) of our system:

$$g(k, c) = g(k^*, c^*) + g_k(k^*, c^*)(k - k^*) + g_c(k^*, c^*)(c - c^*) \quad (3.12)$$

$$\begin{aligned} \text{(I)} \Rightarrow g(k^*, c^*) &= \frac{1}{\theta} \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] \cdot \left[\frac{\alpha}{\rho + \delta} \right]^{\frac{1}{1-\alpha}} \cdot \left\{ \alpha \cdot \left[\frac{\alpha}{\rho + \delta} \right]^{-1} - (\rho + \delta) \right\} \\ &= \frac{1}{\theta} \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] \cdot \left[\frac{\alpha}{\rho + \delta} \right]^{\frac{1}{1-\alpha}} \cdot \{0\} = 0 \end{aligned}$$

$$\text{(II)} \Rightarrow g_k(k^*, c^*) = \frac{c^*}{\theta} \left[\alpha \cdot (\alpha - 1) \cdot (k^*)^{\alpha-2} \right]$$

$$\begin{aligned} &= \frac{1}{\theta} \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] \cdot \left[\frac{\alpha}{\rho + \delta} \right]^{\frac{1}{1-\alpha}} \cdot \left[\alpha(\alpha - 1) \left(\frac{\alpha}{\rho + \delta} \right)^{\frac{\alpha-2}{1-\alpha}} \right] \\ &= \frac{\alpha - 1}{\theta} (\rho + \delta) \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] \end{aligned}$$

$$\text{(III)} \Rightarrow g_c(k^*, c^*) = \frac{1}{\theta} \cdot \left[\alpha \cdot (k^*)^{\alpha-1} - (\rho + \delta) \right] = \frac{1}{\theta} \cdot \left[\alpha \cdot \frac{\rho + \delta}{\alpha} - (\rho + \delta) \right] = 0$$

By substituting equations (I), (II) and (III) into the equation (3.12), we obtain:

$$g(k, c) = \frac{\alpha - 1}{\theta} \cdot (\rho + \delta) \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] \cdot (k - k^*)$$

$$\Rightarrow \dot{c} = \frac{\alpha - 1}{\theta} \cdot (\rho + \delta) \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] \cdot (k - k^*)$$

$$\dot{c} = \left[\frac{\alpha - 1}{\alpha\theta} (\rho + \delta)^2 - \frac{\alpha - 1}{\theta} (\rho + \delta)(n + \delta) \right] \cdot k - B_0 \quad \text{where } B_0 \text{ is constant.}$$

So we have the following differential equation system which is now linear:

$$\dot{k} = A_0 + (\rho - n) \cdot k - c$$

$$\dot{c} = \left[\frac{\alpha - 1}{\alpha\theta} (\rho + \delta)^2 - \frac{\alpha - 1}{\theta} (\rho + \delta)(n + \delta) \right] \cdot k - B_0$$

For homogenous equation solutions (complimentary solutions), we set A_0 and B_0 equal to zero.

$$\dot{k} = (\rho - n) \cdot k - c$$

$$\dot{c} = \left[\frac{\alpha - 1}{\alpha\theta} (\rho + \delta)^2 - \frac{\alpha - 1}{\theta} (\rho + \delta)(n + \delta) \right] \cdot k$$

or, in the vectorial form, these equations can be shown as:

$$\begin{bmatrix} \dot{k} \\ \dot{c} \end{bmatrix} = \begin{bmatrix} \rho - n & -1 \\ \frac{\alpha - 1}{\theta} (\rho + \delta) \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] & 0 \end{bmatrix} \begin{bmatrix} k \\ c \end{bmatrix}$$

i.e;

$$\bar{y} = A \cdot \bar{x} \quad \text{where } A \text{ is a } 2 \times 2 \text{ matrix.}$$

$$\det A = \frac{\alpha - 1}{\theta} \cdot (\rho + \delta) \cdot \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] < 0$$

detA is negative, because

(i) $0 < \alpha < 1 \Rightarrow \alpha - 1 < 0$ and $\theta > 0$

(ii) $\rho + \delta > 0$,

(iii) $\frac{\rho + \delta}{\alpha} > n + \delta$

so we have concluded that roots have opposite signs.

$$\Rightarrow \det(A - \lambda \cdot I) = \begin{vmatrix} \rho - n - \lambda & -1 \\ \frac{\alpha - 1}{\theta}(\rho + \delta) \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] & -\lambda \end{vmatrix}$$

$$= -\lambda \cdot (\rho - n - \lambda) + \frac{\alpha - 1}{\theta}(\rho + \delta) \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] = 0$$

$$= \lambda^2 - (\rho - n)\lambda + \frac{\alpha - 1}{\theta}(\rho + \delta) \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] = 0$$

$$\Rightarrow \lambda_{1,2} = \frac{\rho - n \mp \sqrt{(\rho - n)^2 - \frac{4(\alpha - 1)}{\theta} \left[(\rho + \delta) \left(\frac{\rho + \delta}{\alpha} - (n + \delta) \right) \right]}}{2 \cdot (\rho - n)}$$

One of the roots will be positive and the other one will be negative (saddle path stable). Without loss of generality lets assume λ_1 and λ_2 be the roots of the characteristic equation. So for negative root, we can find characteristic vector, say \bar{v}_2 ,

where, $\bar{v}_2 = \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix}$ so the solution will be :

$$\begin{bmatrix} k(t) \\ c(t) \end{bmatrix} = a_1 \cdot e^{\lambda_2 t} \cdot \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} + \begin{bmatrix} k^* \\ c^* \end{bmatrix}$$

or ;

$$k(t) = a_1 \cdot e^{\lambda_2 t} \cdot v_{21} + k^*$$

$$c(t) = a_1 \cdot e^{\lambda_2 t} \cdot v_{22} + c^*$$

Now let's try to solve eigenvector which means saddle-path in our sense:

$$(A - \lambda_2 I) \cdot \bar{v}_2 = \begin{bmatrix} \rho - n - \lambda_2 & -1 \\ \frac{\alpha - 1}{\theta} (\rho + \delta) \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] & -\lambda_2 \end{bmatrix} \cdot \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix}$$

Then,

$$\begin{aligned} (\rho - n - \lambda_2) \cdot v_{21} - v_{22} &= 0 \\ \frac{\alpha - 1}{\theta} \cdot (\rho + \delta) \cdot \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] \cdot v_{21} - \lambda_2 \cdot v_{22} &= 0 \end{aligned}$$

If we define v_{22} in the first equation above in terms of v_{21} and substitute this in to the second equation then we obtain the following critical equation:

$$\frac{\alpha - 1}{\theta} \cdot (\rho + \delta) \cdot \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] - \lambda_2 \cdot (\rho - n - \lambda_2) = 0 \quad (3.13)$$

$$\Rightarrow \frac{\alpha - 1}{\theta} \cdot (\rho + \delta) \cdot \left[\frac{\rho + \delta}{\alpha} - (n + \delta) \right] - (\rho - n) \cdot \lambda_2 + \lambda_2^2 = 0$$

What we have obtained is the determinant of the characteristic equation when λ is replaced by the root λ_2 . So for $v_{22} = l$, we obtain $v_{21} = \frac{l}{\rho - n - \lambda_2}$ and the eigenvector or direction vector of the saddle-path is

$$\vec{v}_2 = \begin{bmatrix} l \\ \rho - n - \lambda_2 \\ l \end{bmatrix}$$

so we can write the equations of capital and consumption functions with respect to t as:

$$k(t) = a_1 \cdot e^{\lambda_2 t} \cdot \frac{l}{\rho - n - \lambda_2} + k^*$$

$$c(t) = a_1 \cdot e^{\lambda_2 t} \cdot l + c^*$$

The computer solution of this typical problem which is written in the language Mathematica is at the appendix 1.



CHAPTER 4

NUMERICAL METHODS FOR SOLVING ORDINARY DIFFERENTIAL EQUATIONS

4.1 Ordinary Differential Equations and the Lipschitz Condition

We commence our exposition of the computational aspects of differential equations by a close examination of numerical methods for *ordinary differential equations* (ODEs), Arieh (1996).

Our goal is to approximate the solution of the problem

$$y' = f(t, y), \quad t \geq t_0, \quad y(t_0) = y_0 \quad (4.1)$$

Here f is a sufficiently well-behaved function that maps $[t_0, \infty) \times R^d$ to R^d and the initial condition $y_0 \in R^d$ is a given vector. R^d denotes the d -dimensional real Euclidian space.

The ‘niceness’ of f may span a whole range of desirable attributes. At the very least, we insist on f obeying, in a given vector norm $\|\cdot\|$, the *Lipschitz condition*.

$$\|f(t, x) - f(t, y)\| \leq \lambda \|x - y\| \quad \text{for all } x, y \in R^d, \quad t \geq t_0 \quad (4.2)$$

Here $\lambda > 0$ is a real constant that is independent of the choice of x and y . Subject to (4.2), it is possible to prove that the *ODE* system (4.1) possesses a unique solution. Taking a stronger requirement, we may stipulate that f is analytic function. So the Taylor series of f about every $(t, y_0) \in [0, \infty) \times R^d$ has a positive radius of convergence. It is then possible to prove that the solution y itself is analytic.

4.2 Euler's Method

Let us ponder briefly the meaning of the *ODE* (4.1). We possess two items of information, we know the value y at a single point $t = t_0$ and, given any function value $y \in R^d$ and time $t \geq t_0$, we can tell the slope from the differential equation. The purpose of the exercise being to guess the value of y at a new point, the most elementary approach is to use a linear interpolant. In other words we estimate $y(t)$ by making the approximation $f(t, y(t)) \approx f(t_0, y(t_0))$ for $t \in [t_0, t_0 + h]$, where $h > 0$ is sufficiently small, Integrating (4.1),

$$y(t) = y(t_0) + \int_{t_0}^t f(\tau, y(\tau)) d\tau \approx y_0 + (t - t_0) \cdot f(t_0, y_0) \quad (4.3)$$

Given a sequence $t_0, t_1 = t_0 + h, t_2 = t_0 + 2h, \dots$ where $h > 0$ is the *time step*, we denote the numerical estimate of the exact solution $y(t_n)$ by y_n , $n = 0, 1, \dots$

Motivated by (4.3) gives

$$\begin{aligned} y_1 &= y_0 + h \cdot f(t_0, y_0) \\ y_2 &= y_1 + h \cdot f(t_1, y_1) \\ &\vdots \\ y_n &= y_{n-1} + h \cdot f(t_{n-1}, y_{n-1}) \end{aligned}$$

In general, we obtain the recursive scheme

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) \quad n = 0, 1, \dots \quad h = t_n - t_{n-1} \quad (4.4)$$

Euler's method can be easily extended to cater for variable steps. Thus for a general monotone sequence $t_0 < t_1 < t_2 < \dots$ we approximate as follows

$$y(t_{n+1}) \approx y_{n+1} = y_n + h_n \cdot f(t_n, y_n)$$

where $h_n = t_{n+1} - t_n$, $n = 0, 1, \dots$. However, for the time being we restrict ourselves to constant steps.

How good is Euler's method in approximating (4.1). Suppose that we wish to compute a numerical solution of (4.1) in the compact interval $[t_0, t_0 + t^*]$ with some time stepping numerical method, not necessarily Euler's scheme. In other words, we cover the interval by an equidistant grid and employ the time stepping procedure to produce a numerical solution. Each grid is associated with a different numerical sequence and the critical question is whether, as $h \rightarrow 0$ and the grid is being refined, the numerical solution tends to the exact solution of (4.1). More formally, we express the dependence of the numerical solution upon the step size by the notation $y_n = y_{n,h}$, $n = 0, 1, \dots, [t^*/h]$. A method is said to be *convergent* if, for every ODE (4.1) with a Lipschitz function f and every $t^* > 0$, it is true that

$$\lim_{h \rightarrow 0} \max_{n=0,1,\dots,[t^*/h]} \|y_{n,h} - y(t_n)\| = 0 \quad (4.5)$$

Hence convergence means that, for every Lipschitz function, the numerical solution tends to the true solution as the grid becomes increasingly fine. Unless it converges, a numerical method is useless.

Theorem 4.1 Euler's method is convergent.

Proof: We prove this theorem subject to the extra assumption that the function f (and therefore also y) is analytic.

Given $h \rightarrow 0$ and $y_n = y_{n,h}$, $n = 0, 1, \dots, \lceil t^*/h \rceil$, we let $e_{n,h} = y_{n,h} - y(t_n)$ denote the numerical error. Thus we wish to prove that $\lim_{h \rightarrow 0^+} \max_n \|e_{n,h}\| = 0$.

By Taylor's theorem and the differential equation (4.1),

$$y(t_{n+1}) = y(t_n) + h \cdot y'(t_n) + o(h^2) = y(t_n) + h \cdot f(t_n, y(t_n)) + o(h^2)$$

and, y being continuously differentiable, the $o(h^2)$ term can be bounded uniformly for all $h > 0$ and $n \leq \lceil t^*/h \rceil$ by a term of the form ch^2 , where $c > 0$ is a constant.

$$\begin{aligned} \Rightarrow y_{n+1} - y(t_{n+1}) &= y_n + h \cdot f(t_n, y_n) - y(t_n) - h \cdot f(t_n, y(t_n)) - o(h^2) \\ \Rightarrow e_{n+1,h} &= e_{n,h} + h \left[f(t_n, y(t_n) + e_{n,h}) - f(t_n, y(t_n)) \right] - o(h^2) \end{aligned}$$

Thus it follows by the triangle inequality from the Lipschitz condition:

$$\begin{aligned} \Rightarrow \|e_{n+1,h}\| &\leq \|e_{n,h}\| + h \cdot \|f(t_n, y(t_n) + e_{n,h}) - f(t_n, y(t_n))\| + ch^2 \\ \Rightarrow &\leq (1 + h\lambda) \|e_{n,h}\| + ch^2, \quad n = 0, 1, \dots, \lceil t^*/h \rceil - 1 \end{aligned}$$

We now claim that:

$$\|e_{n,h}\| \leq \frac{c}{\lambda} \cdot h \cdot \left[(1 + h\lambda)^n - 1 \right] \quad n = 0, 1, \dots$$

The proof is by induction on n . When $n = 0$, we need to prove that $\|e_{0,h}\| \leq 0$ and hence that $e_{0,h} = 0$. This is certainly true, since at t_0 the numerical solution matches the initial condition and the error is zero.

For general $n \geq 0$, we assume that the inequality is true up to n and use this to argue that:

$$\begin{aligned} \|e_{n+1,h}\| &\leq (1+h\lambda) \cdot \frac{c}{\lambda} \cdot h[(1+h\lambda)^n - 1] + ch^2 \\ &\leq \frac{c}{\lambda} \cdot h(1+h\lambda)^{n+1} - \frac{c}{\lambda} \cdot h(1+h\lambda) + ch^2 \\ &\leq \frac{c}{\lambda} \cdot h[(1+h\lambda)^{n+1} - 1] \end{aligned}$$

The index n is allowed to range in $\{0, 1, \dots, \lceil t^*/h \rceil\}$, hence $(1+h\lambda)^n < e^{[t^*/h]h\lambda} \leq e^{t^*\lambda}$.

$$\|e_{n,h}\| \leq \frac{c}{\lambda} \cdot h \cdot [e^{t^*\lambda} - 1], \quad n = 0, 1, \dots, \lceil t^*/h \rceil$$

Since $c \cdot (e^{t^*\lambda} - 1) / \lambda$ is independent of h , it follows that

$$\lim_{\substack{h \rightarrow 0 \\ 0 \leq nh \leq t^*}} \|e_{n,h}\| = 0$$

In other words, Euler's method is convergent. However the error bound from this proof must not be used in practical estimation of numerical error.

Euler's method can be rewritten in the form $y_{n+1} - [y_n + hf(t_n, y_n)] = 0$. Replacing y_m by the exact solution $y(t_k)$, $k = n, n+1$, and expanding into the first few terms of the Taylor series about $t = t_0 + nh$, we obtain

$$y(t_{n+1}) - [y(t_n) + hf(t_n, y(t_n))] = [y(t_n) + hy'(t_n) + o(h^2)] - [y(t_n) + hy'(t_n)] = o(h^2).$$

We say the Euler's method is of order one. In general given an arbitrary time-stepping method

$$y_{n+1} = Y_n(f, h, y_0, y_1, \dots, y_n), \quad n = 0, 1, \dots$$

for the ODE (4.1), we say that it is of order p if

$$y(t_{n+1}) - Y_n(f, h, y_0, y_1, \dots, y_n) = o(h^{p+1})$$

for every analytic f and $n = 0, 1, \dots$

As far as Euler's method is concerned theorem (4.1) demonstrates that all is well and the error indeed decays as $o(h)$.

4.3 The Trapezoidal Rule

Euler's method approximates the derivative by a constant in $[t_n, t_{n+1}]$, namely by its value at t_n . Clearly, this approximation is not very good and it makes more sense to make the constant approximation of the derivative equal to the average of its values at the end-points. Bearing in mind that derivatives are given by the differential equation, we thus obtain an expression similar to (4.3).

$$\begin{aligned} y(t) &= y(t_n) + \int_{t_n}^t f(\tau, y(\tau)) d\tau \\ &\approx \frac{1}{2}(t - t_n) \cdot \{f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))\} \end{aligned} \quad (4.6)$$

This is the motivation behind the *trapezoidal rule*.

$$y_{n+1} = y_n + \frac{1}{2}h.[f(t_n, y_n) + f(t_{n+1}, y_{n+1})] \quad (4.7)$$

to obtain the order of (4.7), we substitute the exact solution,

$$\begin{aligned} y(t_{n+1}) - \left\{ y(t_n) + \frac{1}{2}h.[f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))] \right\} \\ = \left[y(t_n) + h.y'(t_n) + \frac{1}{2}h^2.y''(t_n) + o(h^3) \right] \\ - \left(y(t_n) + \frac{1}{2}h. \{ y'(t_n) + [y'(t_n) + h.y''(t_n) + o(h^2)] \} \right) \\ = o(h^3) \end{aligned}$$

Therefore the trapezoidal rule is of order two. This means that error decays globally as $o(h^2)$.

Theorem 4.2 The trapezoidal rule is convergent.

Proof: Subtracting

$$y(t_{n+1}) = y(t_n) + \frac{1}{2}h.[f(t_n, y(t_n)) + f(t_{n+1}, y_{n+1})] + o(h^3)$$

from (4.7), we obtain

$$\begin{aligned} e_{n+1,h} &= e_{n,h} + \frac{1}{2}h. \{ [f(t_n, y_n) - f(t_n, y(t_n))] + [f(t_{n+1}, y_{n+1}) - f(t_{n+1}, y(t_{n+1}))] \} \\ &= o(h^3). \end{aligned}$$

For analytic f we may bound the $o(h^3)$ term by ch^3 for some $c > 0$, and this upper bound is valid uniformly throughout $[t_0, t_0 + t^*]$. Therefore, it follows from the Lipschitz condition and the triangle inequality that

$$\|e_{n+1,h}\| \leq \|e_{n,h}\| + \frac{1}{2}h\lambda \left\{ \|e_{n,h}\| + \|e_{n+1,h}\| \right\} + ch^3.$$

Since we are ultimately interested in letting $h \rightarrow 0$, there is no harm in assuming that $h\lambda < 2$, and we can thus deduce that

$$\|e_{n+1,h}\| \leq \left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda} \right) \|e_{n,h}\| + \left(\frac{c}{1 - \frac{1}{2}h\lambda} \right) \cdot h^3.$$

We thus argue that

$$\|e_{n,h}\| \leq \frac{c}{\lambda} \left[\left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda} \right)^n - 1 \right] \cdot h^2$$

This follows by induction on n . Since $0 < h\lambda < 2$, it is true that

$$\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda} = 1 + \frac{h\lambda}{1 - \frac{1}{2}h\lambda} \leq \sum_{l=0}^{\infty} \frac{1}{l!} \left(\frac{h\lambda}{1 - \frac{1}{2}h\lambda} \right)^l = \exp \left(\frac{h\lambda}{1 - \frac{1}{2}h\lambda} \right)$$

Consequently, this derivation yields

$$\|e_{n,h}\| \leq \frac{ch^2}{\lambda} \left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda} \right)^n \leq \frac{ch^2}{\lambda} \exp\left(\frac{nh\lambda}{1 - \frac{1}{2}h\lambda} \right)$$

This bound is true for every nonnegative integer n such that $nh \leq t^*$. Therefore

$$\|e_{n,h}\| \leq \frac{ch^2}{\lambda} \exp\left(\frac{t^*\lambda}{1 - \frac{1}{2}h\lambda} \right)$$

and we deduce that

$$\lim_{\substack{h \rightarrow 0 \\ 0 \leq nh \leq t^*}} \|e_{n,h}\| = 0.$$

In other words, the trapezoidal rule converges.

The number $ch^2 \exp\left[\frac{t^*\lambda}{1 - (1/2)h\lambda} \right] / \lambda$ is again, of absolutely no use in practical error bounds.

Most important one of the differences between the trapezoidal rule and Euler's method is that Euler's method can be executed explicitly; knowing y_n we can produce y_{n+1} by computing a value of f and making a few arithmetic operations, this is not the case with the trapezoidal rule. The vector $v = y_n + \frac{1}{2}h \cdot f(t_n, y_n)$ can be evaluated from known data, but that leaves us in each step with the task of finding y_{n+1} as the solution of the system of algebraic equations. $v = y_{n+1} - \frac{1}{2}h \cdot f(t_{n+1}, y_{n+1})$.

The trapezoidal rule is thus said to be *implicit*, to distinguish it from the *explicit* Euler's method.

Two assumptions have led us to the trapezoidal rule. Firstly, for sufficiently small h , it is a good idea to approximate the derivative by a constant and, secondly, in choosing the constant we should not discriminate between the endpoints hence the average is a sensible choice. Similar reasoning leads, to an alternative approximant,

$$y'(t) \approx f\left(t_n + \frac{1}{2}h, \frac{1}{2}(y_n + y_{n+1})\right), \quad t \in [t_n, t_{n+1}]$$

and to the *implicit midpoint rule*

$$y_{n+1} = y_n + h \cdot f\left(t_n + \frac{1}{2}h, \frac{1}{2}(y_n + y_{n+1})\right).$$

The implicit midpoint rule is a special case of the *Runge Kutta* method.

4.4 The Theta Method

Both Euler's method and the trapezoidal rule fit the general pattern

$$y_{n+1} = y_n + h \cdot \left[\theta \cdot f(t_n, y_n) + (1-\theta) \cdot f(t_{n+1}, y_{n+1}) \right] \quad n = 0, 1, \dots$$

with $\theta = 1$ and $\theta = \frac{1}{2}$ respectively. We may contemplate using the above equation for any fixed value of $\theta \in [0,1]$ and this, appropriately enough, is called a *theta method*. It is explicit for $\theta = 1$. Also this method is of order two for $\theta = \frac{1}{2}$ and otherwise of order one and is convergent for every $\theta \in [0,1]$.

4.5 The Adams Method

Why discard a potentially valuable vector y_0 ? With greater generality, why not make the solution depend on several past values, provided that these values are available?

Let us thus suppose again that y_n is the numerical solution at $t_n = t_0 + nh$, where $h > 0$ is the step-size, and let us attempt to derive an algorithm that intelligently exploits past values. To that end, we assume that

$$y_m = y(t_m) + o(h^{s+1}), \quad m = 0, 1, \dots, n + s - 1 \quad (4.7)$$

where $s \geq 1$ is a given integer. Our wish being to advance the solution from t_{n+s-1} , to t_{n+s} , we commence from the trivial identity

$$\begin{aligned} y(t_{n+s}) &= y(t_{n+s-1}) + \int_{t_{n+s-1}}^{t_{n+s}} y'(\tau) d\tau \\ &= y(t_{n+s-1}) + \int_{t_{n+s-1}}^{t_{n+s}} f(\tau, y(\tau)) d\tau \end{aligned} \quad (4.8)$$

We note that the integral on the right incorporates y not just at the grid points where approximants are available but throughout the interval $[t_{n+s-1}, t_{n+s}]$. The main idea of an *Adams method* is to use past values of the solution to approximate y' in the interval of integration. Thus let p be an interpolation polynomial that matches $f(t_m, y_m)$ for $m = n, n + 1, \dots, n + s - 1$. Explicitly,

$$\begin{aligned} p(t) &= \sum_{m=0}^{s-1} p_m(t) \cdot f(t_{n+m}, y_{n+m}) \\ p_m(t) &= \prod_{\substack{l=0 \\ l \neq m}}^{s-1} \frac{t - t_{n+l}}{t_{n+m} - t_{n+l}} = \frac{(-1)^{s-1-m}}{m!(s-1-m)!} \prod_{\substack{l=0 \\ l \neq m}}^{s-1} \left(\frac{t - t_n}{h} - l \right) \end{aligned}$$

for every $m = n, n+1, \dots, s-1$, are *Lagrange interpolation polynomials*. We can easily verify that $p(t_m) = f(t_m, y_m)$ for all $m = n, n+1, \dots, n+s-1$. Hence (4.7) implies that $p(t_m) = y'(t_m) + o(h^s)$ for this range of m . We now use *Interpolation Theory* to argue that, y being sufficiently smooth.

$$p(t) = y'(t) + o(h^s), \quad t \in [t_{n+s-1}, t_{n+s}].$$

We next substitute p in the integrand of (4.8), replace $y(t_{n+s-1})$ by y_{n+s-1} there and, having integrated along an interval of length h , we incur an error of $o(h^{s+1})$. In other words, the method

$$y_{n+s} = y_{n+s-1} + h \sum_{m=0}^{s-1} b_m \cdot f(t_{n+m}, y_{n+m}) \quad (4.9)$$

where,

$$b_m = h^{-1} \int_{t_{n+s-1}}^{t_{n+s}} p_m(\tau) d\tau = h^{-1} \int_0^h p_m(t_{n+s-1} + \tau) d\tau, \quad m = n, n+1, \dots, s-1$$

is of order s . The scheme (4.9) is called the *s-step Adams-Bashforth method*. For $s = 1$ we encounter the Euler's method, whereas $s = 2$ gives:

$$y_{n+2} = y_{n+1} + h \left[\frac{3}{2} \cdot f(t_{n+1}, y_{n+1}) - \frac{1}{2} \cdot f(t_n, y_n) \right] \quad (4.10)$$

and $s = 3$ gives

$$y_{n+3} = y_{n+2} + h \left[\frac{23}{12} \cdot f(t_{n+2}, y_{n+2}) - \frac{4}{3} \cdot f(t_{n+1}, y_{n+1}) + \frac{5}{12} \cdot f(t_n, y_n) \right] \quad (4.11)$$

Euler's method error decreases linearly, the error of the *two-step Adams-Bashforth method* (4.10) decays quadratically and the *three-step Adams-Bashforth method* displays cubic decay. Generally, the order of the *s-step Adams-Bashforth method* is after all, s and the global error decays as $o(h^s)$.

4.6 Order and Convergence of Multistep Methods

We write a general *s-step* method in the form

$$\sum_{m=0}^s a_m \cdot y_{n+m} = h \cdot \sum_{m=0}^s b_m \cdot f(t_{n+m}, y_{n+m}), \quad n = 0, 1, \dots \quad (4.12)$$

where $a_m, b_m, m = 0, 1, \dots, s$ are given constants, independent of h, n . When $b_s = 0$ and $a_s = 1$ the method is said to be explicit; otherwise it's implicit.

We note that the method (4.12) is of order $p \geq 1$ if and only if

$$\Psi(t, y) := \sum_{m=0}^s a_m \cdot y(t + m \cdot h) - h \cdot \sum_{m=0}^s b_m \cdot y'(t + m \cdot h) = o(h^{p+1}), \quad h \rightarrow 0$$

The method (4.12) can be characterized in terms of the polynomials

$$\rho(w) = \sum_{m=0}^s a_m \cdot w^m \quad \text{and} \quad \sigma(w) = \sum_{m=0}^s b_m \cdot w^m.$$

Theorem 4.3 The multistep method (4.12) is of order $p \geq 1$ if and only if there exists $c \neq 0$ such that

$$\rho(w) - \sigma(w) \cdot \ln(w) = c \cdot (w-1)^{p+1} + o(|w-1|^{p+2}), \quad w \rightarrow 1 \quad (4.13)$$

proof: We assume that y is analytic and that its radius of convergence exceeds $s.h$. Expanding in Taylor series and changing the order of summation,

$$\begin{aligned}\psi(t, y) &= \sum_{m=0}^s a_m \sum_{k=0}^{\infty} \frac{1}{k!} y^{(k)}(t) \cdot m^k \cdot h^k - h \sum_m b_m \sum_{k=0}^{\infty} \frac{1}{k!} y^{(k+1)}(t) \cdot m^k \cdot h^k \\ &= \left(\sum_{m=0}^s a_m \right) y(t) + \sum_{k=1}^{\infty} \frac{1}{k!} \left(\sum_{m=0}^s m^k \cdot a_m - k \cdot \sum_{m=0}^s m^{k-1} \cdot b_m \right) \cdot h^k y^{(k)}(t).\end{aligned}$$

Thus to obtain order p it is necessary and sufficient that

$$\begin{aligned}\sum_{m=0}^s a_m &= 0, & \sum_{m=0}^s m^k \cdot a_m &= k \cdot \sum_{m=0}^s m^{k-1} \cdot b_m, & k &= 1, 2, \dots, p \\ \sum_{m=0}^s m^{p+1} \cdot a_m &\neq (p+1) \cdot \sum_{m=0}^s m^p \cdot b_m.\end{aligned}\tag{4.14}$$

Let $w = e^z$; therefore $w \rightarrow 1$ corresponds to $z \rightarrow 0$. Expanding again in a Taylor series,

$$\begin{aligned}\rho(e^z) - z \cdot \sigma(e^z) &= \sum_{m=0}^s a_m \cdot e^{mz} - z \cdot \sum_{m=0}^s b_m \cdot e^{mz}, & k &= 1, 2, \dots, p \\ &= \sum_{m=0}^s a_m \left(\sum_{k=0}^{\infty} \frac{1}{k!} m^k z^k \right) - z \cdot \sum_{m=0}^s b_m \left(\sum_{k=0}^{\infty} \frac{1}{k!} m^k z^k \right) \\ &= \sum_{k=0}^{\infty} \frac{1}{k!} \left(\sum_{m=0}^s m^k a_m \right) \cdot z^k - \sum_{k=1}^{\infty} \frac{1}{(k-1)!} \left(\sum_{m=0}^s m^{k-1} b_m \right) \cdot z^k\end{aligned}$$

Therefore

$$\rho(e^z) - z \cdot \sigma(e^z) = c \cdot h^{p+1} + o(h^{p+2})$$

for some $c \neq 0$ if and only if (4.14) is true. The theorem follows by restoring $w = e^z$.

Root Condition We say that a polynomial obeys the *root condition* if all its zeros reside in the closed complex unit disc and all its zeros of unit modulus are simple.

Theorem 4.5 (The Dahlquist Equivalence theorem) Suppose that the error in the starting values y_1, y_2, \dots, y_{s-1} tends to zero as $h \rightarrow 0^+$. The multistep method (4.12) is convergent if and only if it is of order $p \geq 1$ and the polynomial ρ obeys the root condition.

Note that *Adams-Bashforth methods* are safe for all $s \geq 1$ according to the root condition, since $\rho(w) = w^{s-1}(w-1)$.

The multistep method (4.12) has $2s+1$ parameters. Had order been the sole consideration, we could have utilized all the available degrees of freedom to maximize it. The outcome, an (implicit) *s-step* method of order $2s$, is unfortunately not convergent for $s \geq 3$. In general, the maximal order of a convergent *s-step* method (4.12) is at most $2[(s+2)/2]$ for implicit schemes and just s for explicit ones; this is known as the *Dahlquist first barrier*.

4.7 Backward Differentiation Formula

Certain multistep methods are significantly better than other schemes of the type (4.12). These are the backward differentiation formula (*BDF*).

An s -order, s -step method is said to be a *BDF* if $\sigma(w) = \beta \cdot w^s$ for some $\beta \in R - \{0\}$.

Lemma For a BDF we have

$$\beta = \left(\sum_{m=1}^s \frac{1}{m} \right)^{-1} \quad \text{and} \quad \rho(w) = \beta \cdot \sum_{m=1}^s \frac{1}{m} \cdot w^{s-m} (w-1)^m. \quad (4.15)$$

The simplest BDF is the backward Euler method which can be obtained by substituting one for s . So when $s = 1$ we have the following formula

$$s = 1 \Rightarrow \beta = 1 \Rightarrow \rho(w) = w - 1,$$

So,

$$\begin{aligned} w - 1 &= h \cdot f(t_{n+1}, y_{n+1}) \\ y_{n+1} - y_n &= h \cdot f(t_{n+1}, y_{n+1}) \end{aligned} \quad n = 0, 1, \dots$$

$$\Rightarrow y_{n+1} = y_n + h \cdot f(t_{n+1}, y_{n+1}) \quad \text{“Backward Euler’s method”}$$

The next two BDF’s are:

$$\begin{aligned} s = 2 \Rightarrow \beta &= \frac{1}{1 + \frac{1}{2}} = \frac{2}{3} \Rightarrow \rho(w) = \frac{2}{3} \left(\sum_{m=1}^2 \frac{1}{m} \cdot w^{2-m} (w-1)^m \right) \\ &\Rightarrow = \frac{2}{3} \left[\frac{3}{2} w^2 - 2w + \frac{1}{2} \right] \end{aligned}$$

So,

$$\begin{aligned} w^2 - \frac{4}{3}w + \frac{1}{3} &= \frac{2}{3} h \cdot f(t_{n+2}, y_{n+2}) \\ \Rightarrow y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n &= \frac{2}{3} h \cdot f(t_{n+2}, y_{n+2}) \quad \text{“Two-step BDF”} \end{aligned}$$

$$s = 3 \Rightarrow \beta = \frac{1}{1 + \frac{1}{2} + \frac{1}{3}} = \frac{6}{11} \Rightarrow \rho(w) = \frac{6}{11} \left[w^3 - \frac{18}{11}w^2 + \frac{9}{11}w - \frac{2}{11} \right]$$

$$\Rightarrow y_{n+3} - \frac{18}{11}y_{n+2} - \frac{9}{11}y_{n+1} - \frac{2}{11}y_n = \frac{6}{11}h \cdot f(t_{n+3}, y_{n+3}) \quad \text{“Three-step BDF”}$$

Theorem 4.6 The polynomial (4.15) obeys the root condition and the underlying BDF method is convergent if and only if $1 \leq s \leq 6$.

4.8 Taylor’s Theorem and Runge Kutta Methods, Mathews (1992)

Theorem 4.7 Assume that $y(t) \in C^{N+1}[t_0, b]$ and that $y(t)$ has a Taylor series expansion of order N about the fixed value $t = t_n \in [t_0, b]$:

$$y(t_n + h) = y(t_n) + h \cdot T_N(t_n, y(t_n)) + o(h^{N+1}), \quad (4.16a)$$

where

$$T_N(t_n, y(t_n)) = \sum_{j=1}^N \frac{y^{(j)}(t_n)}{j!} \cdot h^{j-1} \quad (4.16b)$$

and $y^{(j)}(t) = f^{(j-1)}(t, y(t))$ denotes the $(j-1)$ st total derivative of the function f with respect to t . The formulas for the derivatives can be computed recursively:

$$\begin{aligned} y'(t) &= f, \\ y''(t) &= f_t + f_y \cdot y' = f_t + f_y \cdot f, \\ y'''(t) &= f_{tt} + 2f_{ty} \cdot y' + f_{yy} \cdot [y']^2 \\ &= f_{tt} + 2f_{ty} \cdot f + f_{yy} \cdot f^2 + f_y (f_t + f_y \cdot f), \end{aligned}$$

and in general,

$$y^{(N)}(t) = P^{(N-1)} \cdot f(t, y(t)), \quad (4.17)$$

where P is the derivative operator

$$P = \left(\frac{\partial}{\partial x} + f \frac{\partial}{\partial y} \right)$$

The approximate numerical solution to the initial value problem $y'(t) = f(t, y)$ over $[t_0, t_M]$ is derived by using the formula (4.16a) on each subinterval $[t_n, t_{n+1}]$. The general step for Taylor's method of order N is

$$y_{n+1} = y_n + d_1 h + \frac{d_2 h^2}{2!} + \frac{d_3 h^3}{3!} + \dots + \frac{d_N h^N}{N!} \quad (4.18)$$

where $d_j = y^{(j)}(t_n)$ for $j = 1, 2, \dots, N$ at each step $n = 0, 1, 2, \dots, M - 1$.

The Taylor method of order N has the property that the final global error is of order $o(h^N)$; hence N can be chosen as large as necessary to make this error as small as desired. However, the shortcoming of the Taylor methods is the a priori determination of N and the computation of the higher derivatives, which can be very complicated. Each Runge-Kutta method is derived from an appropriate Taylor method in such a way that the final global error is of order $o(h^N)$. A trade-off is made to perform several function evaluations at each step and eliminate the necessity to compute the higher derivatives. These methods can be constructed for any order N . The Runge-Kutta method of order $N = 4$ is most popular. It is a good choice for common purposes because it is quite accurate, stable, and easy to program.

The fourth-order Runge-Kutta method simulates the accuracy of the Taylor series method of order $N = 4$. The method is based on computing y_{n+1} as follows:

$$y_{n+1} = y_n + w_1 \cdot k_1 + w_2 \cdot k_2 + w_3 \cdot k_3 + w_4 \cdot k_4 \quad (4.19)$$

where k_1, k_2, k_3 , and k_4 have the form

$$k_1 = h \cdot f(t_n, y_n),$$

$$k_2 = h \cdot f(t_n + a_1 \cdot h, y_n + b_1 \cdot k_1),$$

$$k_3 = h \cdot f(t_n + a_2 \cdot h, y_n + b_2 \cdot k_1 + b_3 \cdot k_2),$$

$$k_4 = h \cdot f(t_n + a_3 \cdot h, y_n + b_4 \cdot k_1 + b_5 \cdot k_2 + b_6 \cdot k_3).$$

By matching coefficients with those of the Taylor series method of order $N = 4$ so that the local truncation error is of order $o(h^5)$, Runge and Kutta were able to obtain the following system of equations:

$$b_1 = a_1, \quad b_2 + b_3 = a_2, \quad b_4 + b_5 + b_6 = a_3, \quad w_1 + w_2 + w_3 + w_4 = 1,$$

$$w_2 \cdot a_1 + w_3 \cdot a_2 + w_4 \cdot a_3 = \frac{1}{2},$$

$$w_2 \cdot a_1^2 + w_3 \cdot a_2^2 + w_4 \cdot a_3^2 = \frac{1}{3},$$

$$w_2 \cdot a_1^3 + w_3 \cdot a_2^3 + w_4 \cdot a_3^3 = \frac{1}{4},$$

$$w_3 \cdot a_1 \cdot b_3 + w_4 \cdot (a_1 \cdot b_5 + a_2 \cdot b_6) = \frac{1}{6},$$

$$w_3 \cdot a_1 \cdot a_2 \cdot b_3 + w_4 \cdot a_3 \cdot (a_1 \cdot b_5 + a_2 \cdot b_6) = \frac{1}{8},$$

$$w_3 \cdot a_1^2 \cdot b_3 + w_4 \cdot (a_1^2 \cdot b_5 + a_2^2 \cdot b_6) = \frac{1}{12},$$

$$w_4 \cdot a_1 \cdot b_3 \cdot b_6 = \frac{1}{24}.$$

The system involves 11 equations in 13 unknowns. Two additional conditions must be supplied to solve the system. The most useful choice is

$$a_1 = \frac{1}{2} \quad \text{and} \quad b_2 = 0.$$

then the solution for the remaining variables are

$$a_2 = \frac{1}{2}, \quad a_3 = 1, \quad b_1 = \frac{1}{2}, \quad b_3 = \frac{1}{2}, \quad b_4 = 0, \quad b_5 = 0, \quad b_6 = 1,$$

$$w_1 = \frac{1}{6}, \quad w_2 = \frac{1}{3}, \quad w_3 = \frac{1}{3}, \quad \text{and} \quad w_4 = \frac{1}{6}.$$

Therefore; starting from the initial point (t_0, y_0) , the Runge-Kutta method of order $N = 4$ can be stated as follows:

$$y_{n+1} = y_n + \frac{h.(f_1 + 2f_2 + 2f_3 + f_4)}{6}$$

where

$$f_1 = f(t_n, y_n),$$

$$f_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}f_1\right),$$

$$f_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}f_2\right),$$

$$f_4 = f(t_n + h, y_n + hf_3)$$

4.9 Gaussian Elimination and Pivoting

In this section we state an efficient scheme for solving a general system $AX = B$ of N equations and N unknowns. The crucial step is to construct an equivalent upper-triangular system $UX = Y$.

Two linear systems of dimension N by N are said to be equivalent provided that their solution sets are the same. Theorems from linear algebra show that when certain transformations are applied to a given system, the solution sets do not change.

Theorem 4.8 Suppose that A is an $N \times N$ nonsingular matrix; there exists an equivalent system $UX = Y$ where U is an upper-triangular matrix with $u_{k,k} \neq 0$. After U and Y are constructed, back substitution can be used to solve $UX = Y$ for X .

Proof:

$$AX = \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} \\ a_{2,1}^{(1)} & a_{2,2}^{(1)} & \cdots & a_{2,n}^{(1)} \\ \vdots & \vdots & & \vdots \\ a_{n,1}^{(1)} & a_{n,2}^{(1)} & \cdots & a_{n,n}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,n+1}^{(1)} \\ a_{2,n+1}^{(1)} \\ \vdots \\ a_{n,n+1}^{(1)} \end{bmatrix}$$

Then we will construct an equivalent upper-triangular system $UX = Y$:

$$UX = \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} \\ 0 & a_{2,2}^{(2)} & \cdots & a_{2,n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{n,n}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,n+1}^{(1)} \\ a_{2,n+1}^{(2)} \\ \vdots \\ a_{n,n+1}^{(n)} \end{bmatrix}$$

Step 1. Store the coefficients in the array. The superscript on $a_{r,c}^{(1)}$ means that this is the first time a number is stored in location (r,c) .

$$\begin{array}{cccc}
a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} & a_{1,n+1}^{(1)} \\
a_{2,1}^{(1)} & a_{2,2}^{(1)} & \cdots & a_{2,n}^{(1)} & a_{2,n+1}^{(1)} \\
\vdots & \vdots & & \vdots & \vdots \\
a_{n,1}^{(1)} & a_{n,2}^{(1)} & \cdots & a_{n,n}^{(1)} & a_{n,n+1}^{(1)}
\end{array}$$

Step 2. If necessary, switch rows so that $a_{1,1}^{(1)} \neq 0$; then eliminate x_1 in rows 2 through N . The new elements are written $a_{r,c}^{(2)}$ to indicate that this is the second time that a number has been stored in the array at location (r,c) . The result after step 2 is

$$\begin{array}{cccc}
a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} & a_{1,n+1}^{(1)} \\
0 & a_{2,2}^{(2)} & \cdots & a_{2,n}^{(2)} & a_{2,n+1}^{(2)} \\
\vdots & \vdots & & \vdots & \vdots \\
0 & a_{n,2}^{(2)} & \cdots & a_{n,n}^{(2)} & a_{n,n+1}^{(2)}
\end{array}$$

Step p+1. This is the general step. If necessary, switch row p with some row beneath it so that $a_{p,p}^{(p)} \neq 0$, then eliminate x_p in rows $p+1$ through N . The final result after x_{N-1} has been eliminated from row N is

$$\begin{array}{cccc}
a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} & a_{1,n+1}^{(1)} \\
0 & a_{2,2}^{(2)} & \cdots & a_{2,n}^{(2)} & a_{2,n+1}^{(2)} \\
\vdots & \vdots & & \vdots & \vdots \\
0 & 0 & \cdots & a_{n,n}^{(n)} & a_{n,n+1}^{(n)}
\end{array}$$

The upper-triangularization process is now complete.

Since A is nonsingular, when row operations are performed the successive matrices are also nonsingular. This guarantees that $a_{k,k}^{(k)} \neq 0$ for all k in the construction process. Hence back substitution can be used to solve $UX = Y$ for X .

The number $a_{p,p}$ in position (p,p) that is used to eliminate x_p in rows $p+1, p+2, \dots, N$ is called the p th *pivotal element* and the p th row is called the *pivotal row*. If $a_{p,p}^{(p)} = 0$, row p cannot be used to eliminate the elements in column p below the diagonal. It is necessary to find row k , where $a_{k,p}^{(p)} \neq 0$ and $k > p$ and then interchange row p and row k so that a nonzero pivot element is obtained. This process is called *pivoting*, and the criterion for deciding which row to choose is called a *pivoting strategy*. The trivial pivoting strategy is as follows. If $a_{p,p}^{(p)} \neq 0$, do not switch rows. If $a_{p,p}^{(p)} = 0$, locate the first row below row p in which $a_{k,p}^{(p)} \neq 0$ and switch rows k and p . This will result in a new element $a_{p,p}^{(p)} \neq 0$, which is a nonzero pivot element.

If there is more than one nonzero element in column p that lies on or below the diagonal, there is a choice to determine which rows to interchange. Because the computer uses fixed-precision arithmetic, it is possible that a small error is introduced each time an arithmetic operation is performed. To reduce the propagation of error, it is suggested that one check the magnitude of all the elements in column p that lie on or below the diagonal, and locate row k in which the element has the largest absolute value, that is,

$$|a_{k,p}| = \max \{ |a_{p,p}|, |a_{p+1,p}|, \dots, |a_{N-1,p}|, |a_{N,p}| \},$$

and then switch row p with row k if $k > p$. Usually, the larger pivot element will result in a smaller error being propagated.

A matrix A is called *ill-conditioned* if there exists a vector B for which small perturbation in the coefficients of A or B will produce large changes in $X = A^{-1}B$. The system $AX = B$ is ill-conditioned when A is ill-conditioned. In this case, numerical methods for computing an approximate solution are prone to have more error.

One circumstance involving ill-conditioning occurs when A is 'nearly singular' and the determinant of A is close to zero. Ill-conditioning can also occur in systems of two equations when two lines are nearly parallel (or, in three equations when three planes are nearly parallel).

The computer solution of finding the saddle path equation by using these numerical simulation techniques is written in the language Pascal is at the appendix2.



CHAPTER 5

TYPICAL EXAMPLE OF THE NUMERICAL SOLUTION OF THE STABLE ARM BY USING THE SIMULATION PROGRAM

5.1 Input Initial Values and Parameters

As we have already mentioned in the third chapter our model is written from (3.8) and (3.9) as:

$$\begin{aligned} \dot{k} &= k^\alpha - (n + \delta)k - c \\ \dot{c} &= \frac{c}{\theta} (\alpha \cdot k^{\alpha-1} - \rho - \delta) \end{aligned} \quad (5.1)$$

When the simulation program is executed it will ask for the initial values and the input parameters. Firstly, the user should give the following initial values and input parameters and after that he or she can continue to the following menu which one of the numerical methods the program is going to use should be chosen by the user.

- Initial value for capital (k_0).
- Initial value for consumption (c_0).
- The step size for time increment (h).
- The value of the power of the Cobb-Douglas Production function (α). Note that this simulation program can be modified for the other production function forms but for the time being we use the most well-known one.
- The population growth rate (n).
- The value of social time preference rate (ρ). We also noted that this rate must be positive and greater than the population growth rate (the reason is explained at the page 29).
- The absolute value of the elasticity of marginal utility (θ).
- The value of the depreciation rate of the capital (δ).
- The number of iterations (m). This number is chosen by the user according to the simulation results.

Now let us assume the initial values as:

$$k_0 = 1, \quad c_0 = 0.65, \quad h = 0.01, \quad \alpha = 0.5, \quad n = 0.05, \\ \rho = 0.3, \quad \theta = 0.33680, \quad \delta = 0.07, \quad m = 33.$$

After substituting these values into the differential equation system (5.1) we get the following :

$$\dot{k} = k^{0.5} - (0.05 + 0.07)k - c \\ \dot{c} = \frac{c}{0.33680} (0.5 \cdot k^{0.4} - 0.3 - 0.07) \quad (5.2)$$

5.2 Choosing the Numerical Methods and Approximating Polynomial

After giving the initial values and pressing the enter button the user should choose the numerical method that he or she wants to execute. The numerical methods used by this program are as follows (Of course other methods can be added to the following list as required):

- 1..Theta Method
- 2..Two step Adams Baschforth
- 3..Three step Adams Baschforth
- 4..Two step Backward Differentiation
- 5..Runge Kutta

Now let us assume that our user choose the first one i.e., the theta method then the program is going to ask him or her to give the value of the theta which is used as the weighted average in the theta method as it is explained at the page 59. Let us also assume that our user enter the value 0.5 for the value of the theta then the program now ask for the order of the polynomial which is approximated for the saddle-path. For the time being we can assume that the order of the polynomial to be entered as 2, however it can be changed as desired by choosing the option 1 at the following menu.

So the theta method which is going to be used for our example can be formulated as follows:

$$y_{n+1} = y_n + (0.01) \cdot \left[\frac{1}{2} \cdot f(t_n, y_n) + \frac{1}{2} f(t_{n+1}, y_{n+1}) \right] \quad n = 0, 1, \dots$$

where;
$$y_{n+1} = \begin{bmatrix} k_{n+1} \\ c_{n+1} \end{bmatrix} \quad y_n = \begin{bmatrix} k_n \\ c_n \end{bmatrix}$$

and
$$f(t_n, y_n) = \begin{bmatrix} \dot{k}_n \\ \dot{c}_n \end{bmatrix} \quad f(t_{n+1}, y_{n+1}) = \begin{bmatrix} \dot{k}_{n+1} \\ \dot{c}_{n+1} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} k_{n+1} \\ c_{n+1} \end{bmatrix} = \begin{bmatrix} k_n \\ c_n \end{bmatrix} + \left(\frac{0.01}{2}\right) \cdot \left\{ \begin{bmatrix} \dot{k}_n \\ \dot{c}_n \end{bmatrix} + \begin{bmatrix} \dot{k}_{n+1} \\ \dot{c}_{n+1} \end{bmatrix} \right\}$$

$$\Rightarrow \begin{bmatrix} k_{n+1} \\ c_{n+1} \end{bmatrix} = \begin{bmatrix} k_n \\ c_n \end{bmatrix} + \left(\frac{0.01}{2}\right) \cdot \left\{ \begin{array}{l} k_{n+1}^{0.5} + k_n^{0.5} - (0.12) \cdot (k_{n+1} + k_n) - (c_{n+1} + c_n) \\ \left(\frac{0.5}{0.33680}\right) \cdot (k_{n+1}^{0.4} + k_n^{0.4}) - \left(\frac{0.37}{0.33680}\right) \cdot (c_{n+1} + c_n) \end{array} \right\}$$

$$n = 0, 1, \dots, 33$$

After giving these values the iteration is going to be started and when it is ended successfully the following menu can be seen. If the execution is not ended successfully then the program will give the warning as ‘WARNING!!! This is an ill-conditioning system or diagonal has a zero element’. The meaning of this message is explained at the page 71. In this situation the user can be increase the number of iterations to remove this warning.

For our example the iteration will end successfully and we get the following iteration results:

Table 5.1 The iteration results of the theta method for the solution of the non-linear differential equation system

<i>Iterations</i>	<i>Input Data (k)</i>	<i>Input Data (c)</i>	<i>Calculated Data</i>
1	1.00	0.65	0.6525
2	1.01	0.65	0.6550
3	1.01	0.66	0.6574
4	1.02	0.66	0.6599
5	1.02	0.66	0.6623
6	1.03	0.66	0.6648
7	1.03	0.67	0.6672
8	1.04	0.67	0.6696
9	1.04	0.67	0.6720
10	1.05	0.67	0.6743
11	1.05	0.68	0.6767
12	1.06	0.68	0.6791
13	1.06	0.68	0.6814
14	1.07	0.68	0.6837
15	1.07	0.69	0.6860
16	1.08	0.69	0.6883
17	1.08	0.69	0.6906
18	1.09	0.69	0.6928
19	1.09	0.70	0.6951
20	1.09	0.70	0.6973
21	1.10	0.70	0.6995
22	1.10	0.70	0.7017
23	1.11	0.70	0.7039
24	1.11	0.71	0.7061
25	1.12	0.71	0.7082
26	1.12	0.71	0.7104
27	1.13	0.71	0.7125
28	1.13	0.71	0.7146
29	1.14	0.72	0.7167
30	1.14	0.72	0.7188
31	1.15	0.72	0.7209
32	1.15	0.72	0.7230
33	1.16	0.72	0.7250

5.3 Results and Required Options

Now after obtaining a successful execution the required results options can be chosen from the following menu:

- 1..to change order of approximating polynomial
- 2..to see input data, calculated data and differences
- 3..to see approximating polynomial
- 4..to draw polynomial
- 5..to see equilibrium point
- 6..to calculate the polynomial at a point
- 7..to add-delete-change data points
- 8..to draw tangent to graph at a point
- 9..to change graphic settings
- 10..to start
- 11..to exit

For our example the equilibrium points where $\dot{k} = 0$ and $\dot{c} = 0$ can be seen by choosing option 5 as $k^* = 1.16$ and $c^* = 0.72$ for our input values, when the option 2 is chosen the calculated values from the execution can be seen as $k_{33} = 1.16$ and $c_{33} = 0.72$. It can be seen that the calculated values are fitted with the equilibrium values so our approximated polynomial of order two which is obtained by using the Gaussian Elimination Method which is explained in chapter 4 page 69 can be seen by choosing the option 3 as follows:

$$c(k) = -0.3745.k^2 + 1.2830.k - 0.2586$$

The graph of this equation can be seen from the option 4. If the user requires to find the value of the consumption at any point of capital then the option 6 should be chosen.

If it is also desired to change the order of the approximating polynomial then the option 1 must be chosen. By using option 7, more data points can be added or some of them can be deleted or changed. Option 8 can be chosen for drawing a tangent to the graph at any point and at last by using option 9 the graphic settings can be changed. Options 10 and 11 can be chosen whether to start from the beginning or to exit from the program.

From the first menu if the two step Adams Baschforth method is chosen then the approximating polynomial of order 2 will be as:

$$c(k) = -0.3737k^2 + 1.2751.k - 0.2514$$

If three step Adams Baschforth method is chosen then the approximating polynomial of order 2 will be as:

$$c(k) = -0.3723.k^2 + 1.2620.k - 0.2397$$

If two step backward differentiation method is chosen with the number of iterations to be 17 and the elasticity of marginal utility to be 0.03840, then the calculated values from the execution will be $k_{17} = 1.16$ and $c_{17} = 0.72$ and the approximating polynomial of order 2 will be:

$$c(k) = -0.3691.k^2 + 1.2659.k - 0.2468$$

Other approximating polynomials can be obtained by changing the order of the approximating polynomial by choosing option 1 from the last menu for the above numerical method options which are be able to chosen from the previous menu. For our example when the theta method is chosen and when the order of the approximating polynomial is chosen from 1 to 9 we obtain the following polynomials:

for order 1: $c(k) = 0.4733.k + 0.1783$

for order 2: $c(k) = -0.3745.k^2 + 1.2830.k - 0.2586$

for order3: $c(k) = -0.1043.k^3 - 0.0362.k^2 + 0.9176.k - 0.1272$

for order4: $c(k) = 0.1818.k^4 - 0.8907.k^3 + 1.2383.k^2 + 0.0004.k - 0.1202$

for order5: $c(k) = 0.0808.k^5 - 0.2549.k^4 + 0.0532.k^3 + 0.2189.k^2 + 0.5505.k + 0.0015$

for order6: $c(k) = 0.1476.k^6 - 0.8775.k^5 + 2.3351.k^4 - 3.6789.k^3 + 3.2420.k^2 - 0.7550.k + 0.2363$

for order7: $c(k) = 0.2249.k^7 - 1.4590.k^6 + 4.0223.k^5 - 5.9301.k^4 + 4.6459.k^3 - 1.7607.k^2 + 0.9046.k + 0.0021$

for order8: $c(k) = -0.0229.k^8 + 0.3198.k^7 - 1.7532.k^6 + 5.2210.k^5 - 9.2602.k^4 + 9.8239.k^3 - 6.2417.k^2 + 2.9430.k - 0.3798$

for order9: $c(k) = -0.2082.k^9 + 0.9636.k^8 - 0.9563.k^7 - 2.4969.k^6 + 7.4236.k^5 - 7.2676.k^4 + 1.6181.k^3 + 2.2250.k^2 - 0.9783.k + 0.3272$

It can be checked easily that for $k = 1.08$, all of the above polynomials give the result $c(1.08) = 0.69$. Other iteration points can also be checked in a similar way. So we can conclude from here that:

By using any of the polynomials above, not only the equilibrium values of the capital and consumption variables of our neoclassical growth model but also the transition period values of these variables can be obtained. This is our critical result for this theses.

5.4 Conclusions

As stated in the first chapter, by using this software program dynamic optimization problems (optimal control problems) specifically for growth problems can be easily solved with the numerical methods and the approximating polynomial equation at any desired order of the saddle path can be obtained and the graph of this path can be sketched.

This program can be extended to not only the Cobb-Dougllass production function but also other forms of the production functions which are obeyed the Inada conditions. Also the program can be modified such as to help the development of the new models in growth theory but of course the increase of the number of differential equations to be solved in these models make to apply numerical techniques more difficult.

For the further studies this model and techniques can be used not only in growth models but for other dynamic optimization problems in any field. _

REFERENCES

Arieh Iserless, "A First Course In The Numerical Analysis of Differential Equations", Cambridge University, 1996.

Barro J. Robert, Sala-i-Martin Xavier, "Economic Growth", McGraw-Hill, Inc., 1995.

Cass David, "Optimum Growth in an Aggregate Model of Capital Accumulation", *Review of Economic Studies*, 32, pp233-240, 1965.

Chiang C. Alpha, "Elements of Dynamic Optimization", McGraw-Hill, Inc., 1992.

Robert Dorfman, "An Economic Interpretation of Optimal Control Theory", *The American Economic Review*, 59, 5, December, pp817-831.

Koffman B. Elliot, "Turbo Pascal", Borland International, 1992.

Mathews H. John, "Numerical Methods for Mathematics, Science and Engineering" Printice Hall International Editions, Second Edition, 1992.

APPENDIX 1

ALGORITHM FOR TAYLOR'S APPROXIMATION METHOD

```
Clear[k,c]
Util = Input["Write the utility function: "]
kapf = Input["Write the kapital function: "]
depr = Input["Write the deprecitation rate: "]
prfr = Input["Write the rate of time preference: "]
popn = Input["Write the population parameter: "]
depop = depr + popn
deprf = depr + prfr
f[k[t]_,c[t]_] = kapf-(depop*k[t])-c[t]
csabit = (-1)*(D[Util,{c[t],1}]/(D[Util,{c[t],2}]))
g[k[t]_,c[t]_] = csabit*(D[kapf,{k[t],1}]-deprf)
sol = Solve[{f[k[t]_,c[t]_] == 0,g[k[t]_,c[t]_] == 0,c[t] != 0},{k[t],c[t]}]
f1[k[t]_,c[t]_] = f[k[t]_,c[t]_] /. sol
g1[k[t]_,c[t]_] = g[k[t]_,c[t]_] /. sol
f2[k[t]_,c[t]_] = (D[f[k[t]_,c[t]_],{k[t],1}] /. sol)*(k[t]-(k[t]/.sol))
g2[k[t]_,c[t]_] = (D[g[k[t]_,c[t]_],{k[t],1}] /. sol)*(k[t]-(k[t]/.sol))
f3[k[t]_,c[t]_] = (D[f[k[t]_,c[t]_],{c[t],1}] /. sol)*(c[t]-(c[t]/.sol))
g3[k[t]_,c[t]_] = (D[g[k[t]_,c[t]_],{c[t],1}] /. sol)*(c[t]-(c[t]/.sol))
flin[k[t]_,c[t]_] = f1[k[t]_,c[t]_] + f2[k[t]_,c[t]_] + f3[k[t]_,c[t]_] // Simplify
glin[k[t]_,c[t]_] = g1[k[t]_,c[t]_] + g2[k[t]_,c[t]_] + g3[k[t]_,c[t]_] // Simplify
```

```

a11 = D[flin[k[t]_,c[t]_],{k[t],1}]
a12 = D[flin[k[t]_,c[t]_],{c[t],1}]
a21 = D[glin[k[t]_,c[t]_],{k[t],1}]
a22 = D[glin[k[t]_,c[t]_],{c[t],1}]
matrixA={{a11,a12},{a21,a22}}
detA = (a11*a22)-(a12*a21)
Print[detA]
detmatA=Solve[(((a11-z)*(a22-z))-(a12*a21))==0,z] // Simplify
listz = z /. detmatA
expz = N[E^(listz[[1]]*t)]
matrAz = {{(a11-listz[[1]]),a12},{a21,(a22-listz[[1]])}}
Clear[v1,v2]
eigvect = {v1[t],v2[t]}
dot1[v1[t]_,v2[t]_] = eigvect. {(a11-listz[[1]]),a12}
dot2[v1[t]_,v2[t]_] = eigvect. {a21,(a22-listz[[1]])}
solEgve = NSolve[{dot1[v1[t]_,v2[t]_] == 0,dot2[v1[t]_,v2[t]_] ==
0,v1[t] != 0,(v2[t] == 1)},{v1[t]}]//Simplify
k[t_] = (c1*expz)(v1[t]/.solEgve) + (k[t]/.sol)
solc1 = Solve[{{(% /.t->0) == 0},{c1}}]
k[t_] = ((c1/.solc1)*expz)(v1[t]/.solEgve) + (k[t]/.sol)
Plot[%,{t,0,10},AxesLabel -> {"t", "k(t)"}, PlotRange -> All]
c[t_] = (c2*expz) + (c[t]/.sol)
solc2 = Solve[{{(% /.t->0) == 0},{c2}}]
c[t_] = ((c2/.solc2)*expz) + (c[t]/.sol)
Plot[%,{t,0,10},AxesLabel -> {"t", "c(t)"}, PlotRange -> All]
Plot[{{{(c1/.solc1)*expz)(v1[t]/.solEgve) + (k[t]/.sol),((c2/.solc2)*expz) + (c[t]/.sol)}
,{t,0,100},AxesLabel -> {"c", "t"}, PlotRange -> All]

```


APPENDIX 2

THE SIMULATION ALGORITHM

```
program LinSysEq;
{$N+}
uses crt,graph;

type arraytype1=array [1..70] of extended;
   arraytype2=array [0..25,0..25] of extended;
   stringty=String[5];
const e=1e-10;

var
  K1, KStar, C1, CStar, K2, C2, D1K, D2K, D3K, D4K, D1C, D2C, D3C, D4C,
  pg, p, dr, t, el, h, pw, w           :Real;
  i,j,k,m,n,o,c,gd,gm,mx,my,xx,y1,cc,mc,RC,op  :integer;
  Kin,Cin ,su                          :arraytype1;
  A,L,U,Heq,x,b,bi,d                  :arraytype2;
  mm,kl,ma,f,x0,y0,tt,slope,fx,fy,lx,ly,
  stepx,stepy,gx,gy,ff,aa,bb,data      :extended;
  satr,sat                             :string[10];
  ch                                     :char;
  ffile                                 :file of extended;
  filename                              :string[20];

procedure Initialize;
begin
  gd:=detect;
  initgraph(gd,gm,' ');
  mc:=getmaxcolor;
  setbkcolor(white);
  setcolor(red);
  rectangle(40,40,getmaxx-40,getmaxy-40);
  rectangle(45,45,getmaxx-45,getmaxy-45);
```

```

settextstyle(1,0,2);
setcolor(blue);
outtextxy(80,120,'      THIS PROGRAM DRAWS GRAPH OF DATA
OBTAINED');
outtextxy(80,160,'      FROM      ');
outtextxy(80,200,'      THE NUMERICAL SOLUTION      ');
outtextxy(80,240,'      OF      ');
outtextxy(80,280,'      THE RAMSEY GROWTH MODEL      ');
settextstyle(0,0,0);
setcolor(red);
outtextxy(140,350,'Copyright (c) ,1999, Ankara, BÜLENT DEDEOĐLU');
setcolor(lightblue);
outtextxy(80,420,'***** < Press a key to continue, press ENTER to help > ***** ');
ch:=readkey;
closegraph;
if ch=#13 then
begin
  clrscr;
  gotoxy(15,10);writeln('File helpDG.doc does not found !!!');
  repeat until keypressed;
end;
end;

```

```

procedure InputDatas;
begin
  textcolor(blue);
  textbackground(white);
  clrscr;gotoxy(2,2);
  textcolor(red);Write('1..');
  textcolor(blue);
  Write('Enter the initial values for capital      : ');
  textcolor(black);ReadLn(K1);Writeln;gotoxy(2,4);
  textcolor(red);Write('2..');
  textcolor(blue);
  Write('Enter the initial values for consumption      : ');
  textcolor(black);ReadLn(C1);Writeln;gotoxy(2,6);
  textcolor(red);Write('3..');
  textcolor(blue);
  Write('Enter the stepsize for time increase      : ');
  textcolor(black);ReadLn(h);Writeln;gotoxy(2,8);
  textcolor(red);Write('4..');
  textcolor(blue);
  Write('Enter the value of the power of the Cobb-Douglass : ');
  textcolor(black);ReadLn(pw);Writeln;gotoxy(2,10);
  textcolor(red);Write('5..');
  textcolor(blue);
  Write('Enter the value of population growth-rate      : ');

```

```

textcolor(black);ReadLn(pg);Writeln;gotoxy(2,12);
textcolor(red);Write('6..');
textcolor(blue);
Write('Enter the value of time-preference-rate      : ');
textcolor(black);ReadLn(p);Writeln;gotoxy(2,14);
if p < pg then begin
  textcolor(red);
  writeln(' WARNING!!! Time preference rate must be greater ');
  writeln('      than the population growth rate');Writeln;
  Write(' ..');textcolor(blue);
  write('Enter the value of time-preference-rate again  : ');
  textcolor(black);readln(p);Writeln;gotoxy(2,19);
  Rc:= 99;
  end;
textcolor(red);Write('7..');
textcolor(blue);
Write('Enter the value of elasticity of marginal utility : ');
textcolor(black);ReadLn(el);Writeln;
if rc = 99 then begin gotoxy(2,21); end
else begin gotoxy(2,16); end;
textcolor(red);Write('8..');
textcolor(blue);
Write('Enter the value of depreciation rate of the kapital: ');
textcolor(black);ReadLn(dr);Writeln;
if rc = 99 then begin gotoxy(2,23); end
else begin gotoxy(2,18); end;
textcolor(red);Write('9..');
textcolor(blue);
write('Enter the number of iterations (m)          : ');
textcolor(black);readln(mm);
m:=trunc(mm);
end;

```

```

procedure StarValues;
begin
  RC := 0;
  KStar := Exp((1/1-pw)*Ln(pw/(p+dr))) ;
  CStar := (((p+dr)/pw)-(pg+dr)) * Exp((1/1-pw)*Ln(pw/(p+dr)));
end;

```

```

procedure PrintStarVal;
begin
  clrscr;gotoxy(1,10);
  textcolor(blue) ;Write('Equilibrium point');
  textcolor(blue) ;Write(' (K,C) : ');
  textcolor(red) ;Write('(');Write(KStar:5:2);

```

```

Write(' ');Write(CStar:5:2);Write(' ');
repeat until keypressed;
end;

```

```

procedure ThetaMethodAplication;

```

```

begin
writeln('m: ', m);
for i:=1 to m do
begin
K1 := K1 + ( (w*h) * (Exp(pw * Ln(K1)) + ((pg + dr)*K1) - C1) );
C1 := C1 + ( (w*h) * (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr)) );
K1 := K1 + ( ((1-w)*h) * ( Exp( pw * Ln(K1) ) + ((pg + dr)*K1) - C1) );
C1 := C1 + ( ((1-w)*h) * (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr)) );
Writeln('K1: ', K1:5:2, ' / C1: ', C1:5:2 );
kin[i]:=K1;
cin[i]:=C1;
writeln(kin[i], ' / ', cin[i]);
end; {for}
Writeln('KStar: ', KStar:5:2, ' / CStar: ', CStar:5:2 );
end;

```

```

procedure AdamsBasch2Step;

```

```

begin
writeln('m: ', m);
for i:=1 to m do
begin
K1 := K1 + ( ((-1)*(1/2))*h * (Exp(pw * Ln(K1)) + ((pg + dr)*K1) - C1) );
C1 := C1 + ( ((-1)*(1/2))*h * (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr))
);
K1 := K1 + ( (3/2)*h * ( Exp( pw * Ln(K1) ) + ((pg + dr)*K1) - C1) );
C1 := C1 + ( (3/2)*h * (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr)) );
Writeln('K1: ', K1:5:2, ' / C1: ', C1:5:2 );
kin[i]:=K1;
cin[i]:=C1;
writeln(kin[i], ' / ', cin[i]);
end; {for}
Writeln('KStar: ', KStar:5:2, ' / CStar: ', CStar:5:2 );
end;

```

```

procedure AdamsBasch3Step;

```

```

begin
writeln('m: ', m);
for i:=1 to m do
begin
K1 := K1 + ( (5/12)*h * (Exp(pw * Ln(K1)) + ((pg + dr)*K1) - C1) );
C1 := C1 + ( (5/12)*h * (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr)) );

```

```

K1 := K1 + (((-1)*(4/3))*h * ( Exp( pw * Ln(K1) ) + ((pg + dr)*K1) - C1) );
C1 := C1 + (((-1)*(4/3))*h * (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr))
);
K1 := K1 + ( (23/12)*h * ( Exp( pw * Ln(K1) ) + ((pg + dr)*K1) - C1) );
C1 := C1 + ( (23/12)*h * (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr)) );
WriteLn('K1: ', K1:5:2, ' / C1: ', C1:5:2);
kin[i]:=K1;
cin[i]:=C1;
writeln(kin[i], ' / ', cin[i]);
end; {for}
WriteLn('KStar: ', KStar:5:2, ' / CStar: ', CStar:5:2);
end;

```

```

procedure Backward2Step;

```

```

begin
  writeln('m: ', m);
  for i:=1 to m do
  begin
    K2 := K1/3;
    C2 := C1/3;
    K1 := K1 + ( h * (Exp(pw * Ln(K1)) + ((pg + dr)*K1) - C1) );
    C1 := C1 + ( h * (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr)) );
    K1 := (4/3)*K1;
    C1 := (4/3)*C1;
    K1 := K1 - K2 + ( (2/3)*h * ( Exp( pw * Ln(K1) ) + ((pg + dr)*K1) - C1) );
    C1 := C1 - C2 + ( (2/3)*h * (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr)) );
    WriteLn('K1: ', K1:5:2, ' / C1: ', C1:5:2);
    kin[i]:=K1;
    cin[i]:=C1;
    writeln(kin[i], ' / ', cin[i]);
  end; {for}
  WriteLn('KStar: ', KStar:5:2, ' / CStar: ', CStar:5:2);
end;

```

```

procedure RungeKutta;

```

```

begin
  writeln('m: ', m);
  for i:=1 to m do
  begin
    D1K := (Exp(pw * Ln(K1)) + ((pg + dr)*K1) - C1);
    D1C := (C1/el) * (pw * (Exp((pw-1) * Ln(K1))) - (p + dr));

    D1K := (h/2)*D1K;
    D1C := (h/2)*D1C;
    D2K := (Exp(pw * Ln(D1K)) + ((pg + dr)*D1K) - D1C);
    D2C := (D1C/el) * (pw * (Exp((pw-1) * Ln(D1K))) - (p + dr));

```

```

D2K := (h/2)*D2K;
D2C := (h/2)*D2C;
D3K := (Exp(pw * Ln(D2K)) + ((pg + dr)*D2K) - D2C);
D3C := (D2C/el) * (pw * (Exp((pw-1) * Ln(D2K))) - (p + dr));

D3K := h * D3K;
D3C := h * D3C;
D4K := (Exp(pw * Ln(D3K)) + ((pg + dr)*D3K) - D3C);
D4C := (D3C/el) * (pw * (Exp((pw-1) * Ln(D3K))) - (p + dr));

K1 := K1 + ( (h/6) * (D1K + (2 * D2K) + (2 * D3K) + D4K) );
C1 := C1 + ( (h/6) * (D1C + (2 * D2C) + (2 * D3C) + D4C) );

WriteLn('K1: ', K1:5:2 , ' / C1: ', C1:5:2 );
kin[i]:=K1;
cin[i]:=C1;
writeln(kin[i] , ' / ', cin[i]);
end; {for}
WriteLn('KStar: ', KStar:5:2 , ' / CStar: ', CStar:5:2 );
end;

procedure CreatFile;
{creates a file to store the kapital and consumption values}
begin
filename:='grpoints.DG#';
assign(ffile,filename);
rewrite(ffile);
write(ffile,mm);
for i:=1 to m do
begin
data:=kin[i];
write(ffile,data);
data:=cin[i];
write(ffile,data);
end;
close(ffile);
end;

function POW(var base:extended;power:integer):extended;
{returns power of a number}
var p:extended;
s:integer;
begin
p:=1;
for s:=1 to power do
P:=P*base;
POW:=p;
end;

```

```

procedure Init(var Ma:arraytype2);
{initiates the elements of a matrix to zero}
var ii,jj:integer;
begin
for ii:=1 to 2*n do
for jj:=1 to 2*n do
  Ma[ii,jj]:=0;
end;

```

```

procedure FileInput;
{gets the data from a file initially stored}
begin
  filename:='grpoints.DG#';
  assign(ffile,filename);
  {$I-} reset(ffile);read(ffile,mm){$I+};
  if ioresult=0 then
  begin
  m:=trunc(mm);
  for i:=1 to m do
  begin
  read(ffile,data);
  kin[i]:=data;
  read(ffile,data);
  cin[i]:=data;
  end;
  close(ffile);
  end
  else
  begin
  writeln(' File not found ,try again...');
  delay(3000);
  end;
end;

```

```

procedure printout;forward;

```

```

procedure ChangeInput;
begin
  clrscr;gotoxy(1,10);
  textcolor(red);writeln(' Choose one of the following: ');
  Writeln;
  textcolor(red);write(' 1..');textcolor(blue);
  Writeln('adding new data point');
  textcolor(red);write(' 2..');textcolor(blue);
  Writeln('delete a data point');
  textcolor(red);write(' 3..');textcolor(blue);

```

```

Writeln('change a data point');
textcolor(black);
ch:=readkey;
if (ch='1') or (ch='2') or (ch='3') then
begin
printout;
if ch='1' then
begin
mm:=mm+1;
m:=m+1;
textcolor(red);write('Enter data point Kin['m,'] to be added : ');
textcolor(black);{$I-}readln(Kin[m]);{$I+}
if iorresult=0 then
begin
textcolor(red);write('Enter data Cin['m,'] at the point Kin['m,'] : ');
textcolor(black);readln(Cin[m]);
assign(ffile,filename);
rewrite(ffile);
write(ffile,mm);
for i:=1 to m+1 do
begin
data:=kin[i];
write(ffile,data);
data:=cin[i];
write(ffile,data);
end;
close(ffile);
end;
end
else if ch='2' then
begin
textcolor(red);write(' Enter no of data to be deleted : ');
textcolor(black);readln(j);
mm:=mm-1;
m:=m-1;
assign(ffile,filename);
rewrite(ffile);
write(ffile,mm);
for i:=1 to m+1 do
if i<>j then
begin
data:=kin[i];
write(ffile,data);
data:=cin[i];
write(ffile,data);
end;
end;

```



```

    close(ffile);
end
else if ch='3' then
begin
    textcolor(red);write(' Enter no. of the data to be changed:');
    textcolor(black);read(i);
    textcolor(red);write(' Enter data point Kin['',i,'] to be changed : ');
    textcolor(black);readln(Kin[i]);
    textcolor(red);write(' Enter data Cin['',i,'] at the point Kin['',i,'] : ');
    textcolor(black);readln(Cin[i]);
    assign(ffile,filename);
    rewrite(ffile);
    write(ffile,mm);
    for i:=1 to m+1 do
    begin
        data:=kin[i];
        write(ffile,data);
        data:=cin[i];
        write(ffile,data);
    end;
    close(ffile);
end;
assign(ffile,filename);
reset(ffile);
read(ffile,mm);
m:=trunc(mm);
for i:=1 to m do
    begin
        read(ffile,data);
        kin[i]:=data;
        read(ffile,data);
        cin[i]:=data;
    end;
close(ffile);
end;
end;

procedure FindMatrixForm;
{Finds solution matrices A and b with respect to polinomial order}
begin
    clrscr;gotoxy(2,10); textcolor(red);
    write(' Enter order of the aproximating polynomial (1-9) : ');
    {$I-} textcolor(black);readln(n); {$I+}
    if (ioresult= 0) and (n>0) and (n<10) then begin
        clrscr;gotoxy(30,10);write('Please Wait !');

```

```

n:=n+1;
init(A);
for i:=0 to n do
begin
for j:=0 to n do
for k:=1 to m do
A[i+1,j+1]:=A[i+1,j+1]+POW(Kin[k],(j+i));
end;
init(b);
for i:=0 to n do
for j:=1 to m do
b[i+1,1]:=b[i+1,1]+POW(Kin[j],i)*Cin[j];
end
else findmatrixform;
end;

```

```

procedure Printout2;
{finds and prints polynomial at a value onto the screen}
var ss,ye:extended;
begin
clrscr;gotoxy(1,10);
write(' Enter k value to be calculated: ');
{$I-} textcolor(black);readln(ye); {$I+}
if (ioresult= 0) then begin
ss:=0;
for i:=0 to n-1 do ss:=ss+x[i+1,1]*POW(ye,i);
su[j]:=ss;
writeln( ye:10:4,su[j]:20:4);
writeln(' ');
repeat until keypressed;end
else printout2;
end;

```

```

function cal2(var mm:extended):extended;forward;
function cal(var mm:extended):extended;forward;

```

```

procedure Diff;
begin
for i:=n downto 2 do
begin
d[i-1,1]:=x[i,1]*(i-1);
end;
end;
end;

```

```

procedure PrintPoly;
{prints approximating polynomial onto screen}
begin
clrscr;gotoxy(1,10);
writeln('C(K) = ');
for i:=n downto 2 do
writeln('  ',x[i,1]:10:4,' K^',i-1);
writeln('  ',x[1,1]:10:4);
repeat until keypressed;
end;

```

```

procedure Printout;
{prints data on the screen}
var ss,Cf:extended;
begin
cf:=0;
clrscr;gotoxy(1,10);
Writeln(' ');
textcolor(brown);writeln(' Order of Approximating Polynomial ....:',(n-1):3);
textcolor(red);writeln(' Input Data Point   Input Data   Calculated Data
Difference');
textcolor(black);writeln(' -----   -----   -----   -----
-----');
for j:=1 to m do
begin
ss:=0;
for i:=0 to n-1 do ss:=ss+x[i+1,1]*POW(Kin[j],i);
su[j]:=ss;
textcolor(blue);writeln(j:2,kin[j]:15:2,cin[j]:20:2,su[j]:20:4,(cin[j]-su[j]):20:4);
Cf:=Cf+sqr(Cin[j]-su[j]);
end;
end;

```

```

function cal(var mm:extended):extended;
var ss :extended;
begin
ss:=0;
for i:=0 to n-1 do ss:=ss+x[i+1,1]*POW(mm,i);
cal:=ss;
end;
function cal2(var mm:extended):extended;
var ss :extended;
begin
ss:=0;
for i:=0 to n-2 do ss:=ss+d[i+1,1]*POW(mm,i);
cal2:=ss;
end;

```

```

procedure readsettings;forward;

procedure drawing;
begin
readsettings;
gd:=detect;
initgraph(gd,gm,' ');
mx:=getmaxx;my:=getmaxy;
gx:=(mx-150)/(lx-fx);
gy:=(my-180)/(ly-fy);
rectangle(0,40,mx-10,my-50);
ff:=fx;
f:=fy;
for i:=0 to 3 do
begin
if ff>0 then ff:=0;
if f>0 then f:=0;
line((59+i-round(ff*gx)),60,(59+i-round(ff*gx)),my-95);
line(50,(my-102+round(f*gy)+i),mx-50,(my-102+round(f*gy)+i));
end;
ff:=fx;aa:=fx;if aa>0 then aa:=0;
f:=0;bb:=fy;if bb>0 then bb:=0;
settextstyle(2,horizdir,4);
outtextxy(round(50-fx*gx),round(my-95+fy*gy),'0');
repeat
if ((lx-fx)/stepx)>=10 then cc:=0
else if ((lx-fx)/stepx)>=1 then cc:=1
else if ((lx-fx)/stepx)>=0 then cc:=2;
str(ff:(cc+2):cc,satr);
if ff<>0 then outtextxy(round(50+f*gx),(my-85+round(bb*gy)),satr);
line(round(60+f*gx),my-104+round(bb*gy),round(60+f*gx),my-
96+round(bb*gy));
ff:=ff+(lx-fx)/stepx;
f:=f+(lx-fx)/stepx;
until ff>lx;
ff:=0;
f:=fy;
repeat
if ((lx-fx)/stepy)>=10 then cc:=0
else if ((lx-fx)/stepy)>=1 then cc:=1
else if ((lx-fx)/stepy)>=0 then cc:=2;
str(f:5:cc,satr);
if f<>0 then outtextxy(20-round(aa*gx),round(my-107-ff*gy),satr);
line(56-round(aa*gx),round(my-100-ff*gy),64-round(aa*gx),round(my-100-
ff*gy));

```

```

ff:=ff+(ly-fy)/stepy;
f:=f+(ly-fy)/stepy;
until f>ly;
settextstyle(defaultfont,horizdir,1);
ff:=0;
f:=fx;
setcolor(round(2+getmaxcolor / 2));
outtextxy(50,450,'Wait !..      ');
repeat
  f:=0.02+f;
  ff:=0.02+ff;
  xx:=round(60+ff*gx);
  y1:=round(my-100-(cal(f)-fy)*gy);
  if (xx>10) and (xx<mx-10) and (y1<my-50) and (y1>40) then
    circle(xx,y1,0);
until f>=lx;
f:=40/gx;outtextxy(round(60+f*gx),round(my-120-cal(f)*gy),'f(x)');
setcolor(getmaxcolor+1);
outtextxy(50,450,'Wait !..      ');
setcolor(5+round(getmaxcolor/2));
if c<>8 then
begin
  outtextxy(50,450,'Press any key to return....      ');
  repeat until keypressed;
  closegraph;
end;
end;

```

```

{AFTER HERE PROCEDURES ARE USED TO SOLVE }
{ THE SYSTEM OF EQUATIONS }

```

```

procedure Factorization;

```

```

var Ar:arraytype2;ch:char;
{$M 25384,0,655360}
begin
  for i:=1 to n do
  for j:=1 to n do Ar[i,j]:=A[i,j];
  init(U);
  init(L);
  for i:= 1 to n do
  begin
  for j:=i to n do U[i,j]:=Ar[i,j];
  for k:=i to n do

```

```

begin
  if Ar[i,i]=0 then
    begin
      clrscr;
      Writeln(' A pivot reached to zero.Cannot pivoting!!!');
      repeat until keypressed;
        textcolor(white);
        textbackground(black);
        clrscr;
        halt(0);
    end
  else L[k,i]:=Ar[k,i]/Ar[i,i];
end;
for o:=(i+1) to n do
begin
  ma:=Ar[o,i]/Ar[i,i];
  for k:=i to n do
  begin
    kl:=Ar[i,k]*(-ma)+Ar[o,k];
    ar[o,k]:=kl;
  end;
end;
end;
end;

```

```

procedure Inverse(var inv:arraytype2);
var Ar:arraytype2;
begin
  for i:=1 to n do
  for j:=1 to 2*n do Ar[i,j]:=L[i,j];
  init(inv);
  for j:=1 to n do Ar[j,j+n]:=1;
  for i:= 1 to n do
  begin
    for j:=i to 2*n do Inv[i,j]:=Ar[i,j];
    for o:=(i+1) to n do
    begin
      ma:=Ar[o,i]/Ar[i,i];
      for k:=i to 2*n do
      begin
        kl:=Ar[i,k]*(-ma)+Ar[o,k];
        ar[o,k]:=kl;
      end;
    end;
  end;
end;
end;

```

```

for i:=1 to n do
for j:=1 to 2*n do Ar[i,j]:=Inv[i,j];
for i:=n downto 2 do
begin
for j:=i to n do Inv[i,j]:=ar[i,j];
for o:=i-1 downto 1 do
begin
ma:=Ar[o,i]/Ar[i,i];
for k:=i to 2*n do
begin
kl:=Ar[i,k]*(-ma)+Ar[o,k];
ar[o,k]:=kl;
end;
end;
end;
for i:=1 to n do
for j:=1 to n do
Inv[i,j]:=Ar[i,n+j]/Ar[i,i];
end;

```

```

procedure product(var A1,A2,P:arraytype2);
begin
init(p);
for i:=1 to n do
for j:=1 to n do
for k:=1 to n do
P[i,j]:=P[i,j]+A1[i,k]*A2[k,j];
end;

```

```

procedure PivotalCheck;
var Sum,p:extended;
begin
if n>2 then
begin
Sum:=0;
for i:=0 to n-1 do
begin
p:=1;
for j:=1 to n do
begin
if (i+j)>n then k:=i+j-n
else k:=i+j;
p:=p*(A[j,k]);
end;
Sum:=Sum+p;
end;
end;

```

**T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**

```

for i:=1 to n do
begin
p:=1;
for j:=n downto 1 do
begin
if (n+i-j)>n then k:=i-j
else k:=n+i-j;
p:=p*(A[k,j]);
end;
Sum:=Sum-p;
end;
end
else sum:=A[1,1]*a[2,2]-A[1,2]*A[2,1];
for i:=1 to n do if (A[i,i]=0) or (abs(Sum)<=1e-6) then
begin
clrscr;gotoxy(1,10);textcolor(red);
Writeln(' WARNING!!! This is an ill-conditioning system');
writeln(' or diagonal has a zero element. ');
repeat until keypressed;
textcolor(white);
textbackground(black);
clrscr;
halt(0);
end;
end;

function cal3(var mm:extended):extended;
begin
slope:=cal2(tt);
Y0:=cal(tt)-(slope*tt);
cal3:=slope*mm+y0;
end;

procedure tangent;
begin
drawing;
outtextxy(50,20,'Enter the point x where the tangential to be drawn..');
writeln(' ');
write(' ');
{$I-} readln(tt); {$I+}
if (ioresult= 0) and (tt<lx) and (tt>fx) then begin
outtextxy(50,450,'Wait !.. ');
fillellipse(round(60+(tt-fx)*gx),round(my-100-(cal(tt)-fy)*gy),3,3);
setcolor(round(5+getmaxcolor/2));
f:=fx;
ff:=0;

```



```

repeat
  f:=0.02+f;
  ff:=0.02+ff;
  xx:=round(60+ff*gx);
  y1:=round(my-100-(cal3(f)-fy)*gy);
  if (xx>15) and (xx<mx-20) and (y1<my-60) and (y1>45) then
    circle(xx,y1,0);
  if (f>0) and (f<0.03) then outtextxy(xx-10,y1,'A');
  if (cal3(f)>-0.02) and (cal3(f)<0.04) then outtextxy(xx,y1+10,'B');
  until ((f>lx+20/gx) );
f:=tt+20/gx;
outtextxy(round(60+f*gx),round(my-120-cal3(f)*gy),'T(x)');
setcolor(getmaxcolor+1);
outtextxy(50,450,'Wait !.. ');
setcolor(round(10+getmaxcolor/2));
ff:=0;
str(cal3(ff):(cc+2):cc,satr);
outtextxy(mx-110,60,'A=(0.0;'+satr+'');
ff:=-y0/slope;
str(ff:(cc+2):cc,satr);
outtextxy(mx-110,80,'B=('+satr+';0.0)');
str(cal(tt):(cc+2):cc,satr);
str(tt:(cc+2):cc,sat);
outtextxy(mx-110,100,'T=('+satr+';'+satr+'');
rectangle(mx-125,40,mx-10,125);
outtextxy(50,450,'Press a key to return... ');
repeat until keypressed;
end else c:=0;
closegraph;
end;

```

```

procedure BackSubs;
var sum:extended;
begin
  product(Heq,b,bi);
  x[n,1]:=bi[n,1]/u[n,n];
  for i:=n-1 downto 1 do
    begin
      sum:=0;
      for k:=i+1 to n do Sum:=Sum+u[i,k]*x[k,1];
      x[i,1:]=(bi[i,1]-Sum)/u[i,i];
    end;
  end;
end;

```

```

procedure WriteSetting;
var ffile:file of extended;
    filename:string[20];
    data:extended;
begin
i:=0;
assign(ffile,'settings.dat');
rewrite(ffile);
write(ffile,fx);
write(ffile,lx);
write(ffile,fy);
write(ffile,ly);
write(ffile,stepx);
write(ffile,stepy);
close(ffile);
end;

```

```

procedure ReadSettings;
var ffile:file of extended;
    filename:string[20];
    data:extended;
begin
clrscr;gotoxy(1,10);
assign(ffile,'settings.dat');
{$I-} reset(ffile);{$I+};
if ioresult=0 then
begin
read(ffile,fx);
read(ffile,lx);
read(ffile,fy);
read(ffile,ly);
read(ffile,stepx);
read(ffile,stepy);
close(ffile);
end
else
begin
fx:=-100;
lx:=100;
stepx:=10;
stepy:=10;
fy:=-100;
ly:=100;
WriteSetting;
end;
end;

```

```

procedure readdata(var value :extended);
begin
  satr:=ch;write(ch);
  repeat
  ch:=readkey;
  if ch<>#13 then
  begin
    satr:=satr+ch;
    write(ch);
  end;
  until ch=#13;
  writeln(' ');
  val(satr,value,i);
end;

```

```

procedure ChangeSettings;
begin
  clrscr;gotoxy(1,10);
  Writeln(' 1 .. AUTO');
  writeln(' 2 .. MANUAL');Writeln;
  Write(' To Change Graphic Settings Choose One :');
  ch:=readkey;
  if ch='1' then
  begin
    fx:=-20;
    lx:=20;
    stepx:=5;
    stepy:=5;
    fy:=-20;
    ly:=20;
    writesetting;
  end
  else if ch='2' then
  begin
    repeat
    clrscr;gotoxy(1,10);
    if i=-1 then writeln(' Input Error !!!');
    Writeln(' Write New Settings ');
    writeln(' (press ENTER if no change)');
    writeln(' ');
    {$I-}
    i:=0;
    write(' Min X =',fx:15:3,'--->');
    ch:=readkey;if ch=#13 then writeln(fx:4:3) else readdata(fx);
    write(' Max X =',lx:15:3,'--->');
    ch:=readkey;if ch<>#13 then readdata(lx) else writeln(lx:4:3);

```

```

write('  Min Y    =',fy:15:3,'--->');
ch:=readkey;if ch<>#13 then readdata(fy) else writeln(fy:4:3);
write('  Max Y    =',ly:15:3,'--->');
ch:=readkey;if ch<>#13 then readdata(ly) else writeln(ly:4:3);
write('Step for X axis =',stepx:15:0,'--->');
ch:=readkey;if ch<>#13 then readdata(stepx) else writeln(stepx:1:0);
write('Step for Y axis =',stepy:15:0,'--->');
ch:=readkey;if ch<>#13 then readdata(stepy) else writeln(stepy:1:0);
{$I+}
if i<>0 then i:=-1;
if (ioresult= 0) and (fx<lx) and (fy<ly) and (stepx<>0) and (stepy<>0) and (i<>-1)
    then writesetting else i:=-1;
until i<>-1;
end;
end;
end;
procedure Main;
begin {main}
  InputDatas;
  StarValues;
  if RC = 0 then
  begin
    clrscr;gotoxy(1,10);
    textcolor(red);writeln(' Choose one of the following methods: ');
    Writeln;textcolor(red);write('  1..');textcolor(blue);
    Writeln('Theta Method');
    textcolor(red);write('  2..');textcolor(blue);
    Writeln('Two step Adams Baschforth');
    textcolor(red);write('  3..');textcolor(blue);
    Writeln('Three step Adams Baschforth');
    textcolor(red);write('  4..');textcolor(blue);
    Writeln('Two step Backward Differentiation');
    textcolor(red);write('  5..');textcolor(blue);
    Writeln('Runge Kutta');
    {$I-} textcolor(black);readln(op);textcolor(blue); {$I+}
    if op = 1 then
    begin
      clrscr;gotoxy(2,10);
      textcolor(red);Write(' Enter the value of the theta : ');
      {$I-} textcolor(black);readln(w);textcolor(blue); {$I+};
      ThetaMethodAplication;
    end
  else if op = 2 then begin AdamsBasch2Step; end
  else if op = 3 then begin AdamsBasch3Step; end
  else if op = 4 then begin Backward2Step; end
  else if op = 5 then begin RungeKutta; end;
  CreatFile;
end;

```

```

repeat
  FileInput;
  repeat;
  FindMatrixForm;
  Factorization;
  Inverse(Heq);
  PivotalCheck;
  BackSubs;
  diff;
  readsettings;
  repeat
    clrscr;gotoxy(1,10);
    textcolor(red);write(' 1');textcolor(blue);writeln('..to change order of
approximating polynomial');
    textcolor(red);write(' 2');textcolor(blue);writeln('..to see input data, calculated data
and differences ');
    textcolor(red);write(' 3');textcolor(blue);writeln('..to see approximating
polynomial');
    textcolor(red);write(' 4');textcolor(blue);writeln('..to draw polynomial');
    textcolor(red);write(' 5');textcolor(blue);writeln('..to see equilibrium point');
    textcolor(red);write(' 6');textcolor(blue);writeln('..to calculate the polynomial at a
point');
    textcolor(red);write(' 7');textcolor(blue);writeln('..to add-delete-change data
points');
    textcolor(red);write(' 8');textcolor(blue);writeln('..to draw tangent to graph at a
point');
    textcolor(red);write(' 9');textcolor(blue);writeln('..to change graphic settings');
    textcolor(red);write(' 10');textcolor(blue);writeln('..to start');
    textcolor(red);write(' 11');textcolor(blue);writeln('..to exit');

    writeln(' ');
    write('      Make a choice :');textcolor(red+blink);write(' ');
    {$I-} readln(c);textcolor(blue); {$I+}
    if ioresult=0 then
    begin
    if c=2 then begin Printout;repeat until keypressed;end
    else if c=3 then PrintPoly
    else if c=4 then drawing
    else if c=5 then PrintStarVal
    else if c=6 then Printout2
    else if c=7 then ChangeInput
    else if c=8 then Tangent
    else if c=9 then ChangeSettings
    else if c=10 then Main;
    end
    else c:=0;

```

```
until (c=1) or (c=11) or ((c=7) and ((ch='1') or (ch='2') or (ch='3')));  
until (c=11) or (c=5);  
clrscr;  
until(c=11);  
textcolor(white);  
textbackground(black);  
normvideo;  
clrscr;  
end;  
begin  
Initialize;  
Main  
end.
```



**T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**