Computer Modelling Applied to the Calvin Cycle

Mark Graham Poolman

A Thesis submitted in partial fulfilment of the requirements of
Oxford Brookes University
for the degree of

Doctor of Philosophy

January 1999

Abstract

This thesis developes computer modelling techniques, and their use in the investigation of biochemical systems, principally the photosynthetic Calvin cycle. A set of metabolic modelling software tools, "Scampi", constructed as part of this project is presented. A unique feature of Scampi is that it allows the user to make a particular model the subject of arbitrary algorithms. This provides a much greater flexibility than is available with other metabolic modelling software, and is necessary for work on models of (or approaching) realistic complexity.

A detailed model of the Calvin cycle is introduced. It differs from previously published models of this system in that all reactions are assigned explicit rate equations (no equilibrium assumptions are made), and it includes the degradation, as well as the synthesis, of starch. The model is later extended to include aspects of the thioredoxin system, and oxidative pentose phosphate pathway. Much of the observed behaviour is consistent with experimental observation. In particular, Metabolic Control Analysis of the model shows that control of assimilation flux is likely to be shared between two enzymes, rubisco and sedoheptulose bisphosphotase (SBPase), and can readily be transferred between them. This appears to offer an explanation of experimental evidence, obtained by genetic manipulation, that both of these enzymes can exert high control over assimilation. A further finding is that the output fluxes from the cycle (to starch and the cytosol), show markedly different patterns of control from assimilation, and from each other.

An novel observation in behaviour of the Calvin cycle model is that, under certain circumstances, particularly at low light levels, the model has two steady-states and can be induced to switch between them. Although this exact behaviour has not been described experimentally, published results show charecteristics suggesting the potential is there *in vivo*.

An explanation of all the observed behaviour is proposed, based upon the topology of the model. If this is correct then it may be concluded that the qualitative behaviour observed in the model is to be expected *in vivo*, although the quantitative detail may vary considerably.

Acknowledgements

People

In addition to those people who have been explicitly cited in this thesis, there are others who have helped, encouraged, and made the undertaking more enjoyable than it otherwise would have been, and to whom I therefore wish to record my thanks.

As well as my supervisor, Dr. David Fell, and co-supervisors, Drs. Simon Thomas and Michael Burrel, I would like to thank former members of the Metabolic Control Analysis group at Oxford Brookes University, Drs. John Woods and J/ uan-Pedro Barrito for stimulating and often challenging discussions, and for infecting me with their enthusiasm for things metabolic.

I would particularly like to thank Dr. Herbert Sauro (also late of the Metabolic Control Analysis group) for letting me have the source code for his metabolic simulation package, SCAMP. This represents the single biggest contribution to the work described here, and I hope I will be forgiven if he thinks that I have been unduly critical of SCAMP in the course of this thesis.

Beyond the circle of my immediate colleagues, I wish thank my undergraduate personal tutor, Ms. Hazel Peperell, for encouraging me to give those areas of computing that were interesting precedence over those that were merely conventional.

No list of acknowledgements could be complete without inclusion of my wife, Karen. She has tolerated the various costs of being married to someone working on a PhD, without necessarily sharing the pleasures of observing unexpected behaviour in a computer model, or achieving a good fit between simulated and real-world data sets.

When engaged in the activity of identifying those who have influenced one's life there is the inevitable temptation, which must be resisted, to extend the line back in time, until arriving at some point shortly after birth. Nonetheless, I must record my gratitude to Senior Sister Iris Holmes, late of the Towler theatre suit at Oxford's Radcliffe Infirmary. By turning a blind eye to the hours spent completing maths exercises, when, officially, I should have been engaged in other duties. I thus was able to gain the qualification which represented an early, and essential, mile stone on the road upon which this thesis represents most recent.

Software

The software used during the course of this project has been entirely free (in the sense of both freedom and lunch). Without the public-spirited attitude of those workers who thus provide such software, the project presented in this thesis would probably have been impossible.

Most programming was undertaken using the (ANSI) 'C' programming language [66] with the Gnu C compiler, "gcc" [3]. The "flex" [1] variant of the lexer generator, "lex" [84], was used to construct the code generator described in section 2.2.1.

Additional processing of results was accomplished with the Gnu variant of the "Awk" [5] programming language, "gawk" [2]. All graphs were plotted using the "Gnuplot" plotting and fitting program [65], and other figures produced with the "xfig" drawing program [4]. All document preparation was carried using the "PasTex" [60] distribution of the LaTeX [79] document preparation system.

Contents

Chapter 1

Introduction

1.1 Aims of the Thesis

The original motivation for the work upon which this thesis is based was three-fold:

1. To identify and characterise the metabolic components of plants that determine the allocation of photosynthetically assimilated carbon between various parts of the plant.

2. To develop the computer modelling skills necessary to build, and analyse, models of realistic biological complexity, and apply these to photosynthesis.

3. To use such results to identify useful investigative techniques that may be applied to other systems, and to draw more general conclusions relating system structure to behaviour.

As the project progressed, the scope of these goals narrowed considerably, and those of this thesis are correspondingly more modest:

1. To describe computer software, constructed as part of the project, that allows large metabolic models to be effectively investigated.

2. To present the results of using this software in conjunction with a detailed model of the Calvin cycle of photosynthesis.

3. To demonstrate that these results are consistent with experimental observation, and that model results and experimental observation can be shown to arise from

from the topology of the Calvin cycle.

Although some room for improvement in the realisations of these goals may remain, it is hoped that all three areas show improvements over the works of those who have gone before, but without which this thesis would not have been possible.

The remainder of this chapter serves to describe in outline, the flow of carbon within plants, traditional and modern views on the control of metabolism, and the principles of computer modelling of biochemical systems, and concludes with an overview of the subsequent chapters.

1.2 Photosynthetic Carbon Metabolism

1.2.1 Photosynthesis

Photosynthesis is the process by which energy recovered from the absorption of light by certain pigments, is used to assimilate CO_2, releasing oxygen, and incorporating carbon into metabolically useful compounds. As CO_2 is the end point of most other metabolic processes, photosynthesis, by recycling the carbon, is essential for sustainable life on earth[1].

Photosynthesis is common amongst the prokaryotes as well as the eukaryotes, but only the latter are considered in this thesis. The radically different intra-cellular organisation of the two means that results obtained from one are not readily applicable to the other.

Eukaryotic photosynthetic organisms may be sub-divided into three categories: C-3, C-4, and crassulacean acid metabolism (CAM) [111]. C-4 photosynthesis and CAM are thought to have evolved in response to arid conditions. The operation of C-4 photosynthesis is dependent upon a specialised leaf anatomy, and CAM upon an unusual diurnal cycle. Both C-4 and CAM photosynthesis are extensions to, and not substitutes for, C-3 photosynthesis, which is therefore present in all photosynthetic eukaryotes. Accordingly, this thesis is restricted to the consideration of C-3 metabolism.

The process of photosynthesis, exclusively confined to the chloroplast, is divided into two, logically and physically separate, components: the energy harvesting light reactions, located on the thylakoid membrane; and the carbon fixing Calvin cycle, located in the

[1] With the possible exception of habitats around deep ocean geo-thermal vents, and similarly exotic locations

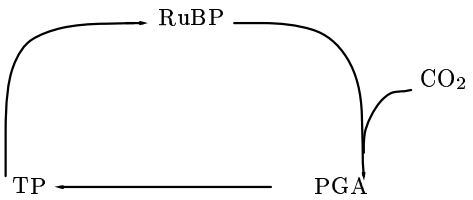

Figure 1.1: General outline of the Calvin cycle , showing the flow of carbon through the three limbs.

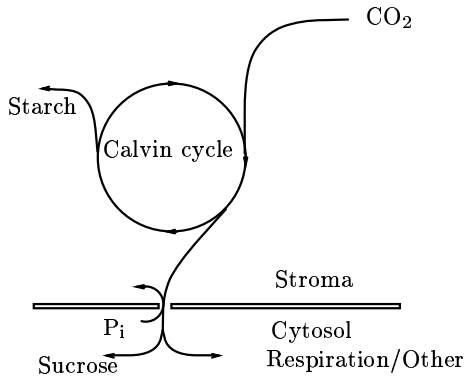

Figure 1.2: Main carbon pathways in photosynthetic plant cells.

chloroplast stroma. It is with the latter that this thesis is predominantly concerned. The light reactions are essential to drive the Calvin cycle, and although the potential for interaction between the light reactions and Calvin cycle is considered in the thesis, the underlying mechanisms of the light reactions are not.

1.2.2 The Calvin cycle

Structure

It is the Calvin cycle, the source of carbon for photoautotrophic organisms, and the ultimate carbon source for heterotrophs, that is the main object of study in this thesis. In its minimal form, the cycle consists of a set of thirteen reactions, comprising a rather complex whole. However, for many purposes, this complexity may be greatly simplified if the Calvin cycle is considered to be comprised of three separate limbs, or branches: the assimilatory, the reductive, and the regenerative, as illustrated in Figure 1.1.

The assimilatory limb is comprised of a single reaction; the formation of two molecules of the three carbon compound phosphoglycerate (PGA) by the addition of CO_2 to the five carbon ribulose bisphosphate (RuBP), catalysed by the enzyme ribulose-bisphosphate carboxylase-oxygenase (rubisco).

The two reactions of the reductive limb are the point at which most of the energy investment from the light reactions is made, reducing the relatively oxidised PGA to three carbon triose phosphate (TP).

The remaining reactions form the most complex component of the cycle, the regen-

erative limb, which have an overall stoichiometry of $5\mathrm{TP} + \mathrm{P_i} \longrightarrow 3\mathrm{RuBP}$. Although an investment of one molecule of ATP for each molecule of RuBP is made, in this context it is more convenient to regard ATP as a carrier of $\mathrm{P_i}$, rather than of energy.

Fate of products

Carbon that is assimilated, but not regenerated to RuBP has two immediate possible fates. It can either enter the starch synthesis pathway, located within the the stroma, or be exported in the form of TP or PGA, from the chloroplast to the cytosol.

The purpose of (chloroplast) starch is to provide a buffer, or reservoir, of carbon against the possibility of carbon demand exceeding assimilation [87,26]; this is obviously the case at night when photosynthesis cannot take place, but has also been reported under experimental conditions in the light [128]. Under normal physiological conditions starch is synthesised during the day and degraded at night. Although it is probable that several routes exist for the degradation of starch, one of these is certainly the simple reversal of the synthesising reaction [94, 12], hence chloroplastic starch metabolism can serve to act as a sink or a source, of Calvin cycle intermediates. As will be shown in subsequent chapters, the presence of starch metabolism has a major effect on the behaviour of the Calvin cycle , and, for the purposes of this thesis, will be regarded as an integral component of it: the storage limb.

Carbon is exported from the chloroplast (at least in the light), in the form of TP and PGA, exclusively mediated by a transport protein, the triose phosphate - phosphate translocator (TPT) [59, 36, 27]. As its name implies, the TPT exports TP in strict counter exchange for $\mathrm{P_i}$, and this is of major physiological significance for several reasons, one of which is that it enables (the concentration of) cytosolic $\mathrm{P_i}$ ($=\mathrm{P_{i_{ext}}}$) to act as a signal for carbon demand. Therefore, the process of export will also be regarded as a limb of the cycle. In higher plants[2], exported carbon has two main fates: the anabolic sucrose synthesis pathway (ultimately exported into the phloem), and the catabolic route to respiration. Although cytosolic respiration is known to occur in the light its physiological significance remains unclear [71].

The major carbon routes, described above, are illustrated in Figure 1.2. The scope of most of this thesis is confined to consideration of the Calvin cycle, including the storage and export branches.

[2]Those with a recognisable vasculature

1.3 The Control of (Photosynthetic Carbon) Flux

1.3.1 Background

The history of biochemistry may reasonably said to have begun at the turn of last century, with the investigations of fermentation, initiated by the Buchner brothers in 1897, and continued by Harden and Young through the first decade of this century. The realisation that hitherto incomprehensible processes took place in small, discrete steps, amenable to isolation and individual investigation, prompted an explosion of activity, with the result that by the end of the 1950s the routes taken by almost all main metabolic processes had been elucidated. It would, of course, be foolish to assert that any particular field of academic inquiry is complete, however, the metabolic map of textbooks of thirty years ago is little different from that of those today.

At the same time that the pathways of metabolism were being mapped out, attention was also turned to the understanding of the individual, enzyme catalysed, reactions that comprise them. This endeavor appears to have been equally successful, resulting in the field of enzymology, and since perhaps the mid 1960s little new fundamental work in this field has appeared, although matters such as experimental design and data handling are still areas of current work. The state of the art is such that it has led one prominent worker in the field to state "*... the analysis of the steady-state behaviour of an individual isolated enzyme may now be regarded as a solved problem*" [21].

1.3.2 Problems in understanding multi-enzyme systems

Despite the formidable body of knowledge representing the individual steps that make up the pathways of metabolism, and the mathematical rigor with which those steps may be characterised, understanding of the behaviour of pathways in terms of their constituent steps remains poor. This state of affairs may be attributed to several causes, acting simultaneously.

Given that rate equations for individual reactions can, in principle, be determined with precision, it might be hoped that the individual rate equations for a given pathway could be combined to yield a rate equation for the whole pathway. Unfortunately, this is not the case: the non-linearity that is inherent in all enzyme rate equations results in such an extreme increase in the complexity of any simultaneous solution to a set of such equations, that this approach is not generally regarded as viable [21]. Under

certain circumstances it may be possible to render the problem tractable by the use of simplifying assumptions: either reducing the number of reactions assumed to be present, or by simplifying the rate equations, or both. The approach can be useful, and is discussed more fully in chapter 6. However, much information is inevitably lost, and anyway begs the questions as to what are reasonable simplifying assumptions, and as to what effects such simplifications have upon the final results.

1.3.3 The quest for rate limiting steps

A second obstacle to developing a general understanding of biochemical systems appears to have arisen as a result of taking the process of simplification to its logical conclusion, and assuming that the system behaviour may be approximated by that of a single enzyme. Such putative crucial enzymes have been variously known as "rate limiting", "regulatory" or "pace-maker" enzymes. The expectation of a rate limiting enzyme tends to be justified with the observation that in any system comprised of a group of enzymes, one will be the slowest, and that the flux through the system is therefore determined by that step, and that by controlling that step the rate through the whole pathway is controlled. This may be compared, by analogy, with a convoy of ships (for example) whose speed is determined by that of the slowest. The earliest formulation of this hypothesis is usually accredited to the work of F. F. Blackman, in the first decade of this century [30,137]. Although Salisbury and Ross [112] credit Blackman with proposing the term "rate-limiting factor", they attribute the concept to the work of Liebig in 1840.

Both Blackman and Liebig were investigating the response of plant growth and photosynthesis to changes in environmental factors, a long time before the mechanisms of photosynthesis were known; they investigated such external factors as the availability of nutrients, which at low concentrations limit growth. It is not clear when the idea of rate limiting factors was applied to the activity of enzymes, but Krebs [70], in 1957 takes the existence of pacemaker enzymes as granted. It is certain that since then, discussion of the control and regulation of metabolic systems has been dominated by the search for, and characterisation of, rate-limiting steps.

Unfortunately, although at first sight the convoy analogy appears obvious, it is quite wrong. The speed will be governed by many, possibly conflicting, factors, including sea and weather conditions, the urgency for cargo to be delivered, the increase in fuel costs by travelling more quickly, and so on. The only statement that can be made with

certainty about the possible contribution of a single ship to the speed at which the whole convoy moves, is that the *maximum* speed at which the convoy can travel is equal to the *maximum* speed that the slowest ship can attain.

In recent years there has been a steady accumulation of theoretical and experimental evidence, that as a general principle, the rate limiting step hypothesis is flawed. The experimental evidence is discussed in some depth by Fell [31], and due to considerations of space will not be discussed further here. This thesis make its contribution to undermining the rate limiting step hypothesis by showing that, in a detailed model of the Calvin cycle, control over a given flux is likely to be shared between at least two steps, can change dramatically in response to environmental change, and that even if one enzyme does attain the characteristics of a rate limiting step, that this degree of control is conferred by the structure of the whole system, and cannot be determined by consideration of that enzyme alone.

Although alternatives to the rate limiting step hypothesis have gained ground, it is probably true to say that, at the time of writing, this hypothesis remains the current orthodoxy, and has led to many workers devoting their time to the detailed study of a single enzyme. One of the more striking examples of this is rubisco, the enzyme catalysing the initial CO_2 fixing reaction in the Calvin cycle This is frequently assigned the central, if not the sole rôle in the control and regulation of photosynthetic carbon assimilation [86,138,139]. Perhaps the most convincing piece of evidence as to the great extent of this *idée fixe*, is that a recent electronic search of literature published since 1990 containing "rubisco" in the title, key-word list, or abstract, yielded some 1200 articles. Repeating this for "Calvin cycle" resulted in only 350.

1.3.4 Concepts of distributed control

The concept of rate limiting steps is qualitative and absolute: an enzyme is either the rate limiting step for a pathway, or it is not; and if not, has no effect upon pathway flux.

The period between the late 1960s and mid 1980s saw the publication of a considerable body of work providing a theoretical framework within which the influence of a given enzyme over pathway flux, or more generally, the influence of any parameter over any variable, can be quantified.

These can be divided into three major schools of thought: biochemical systems theory [119], Newsholme and Crabtree's work [24] called flux oriented theory by some,

and metabolic control analysis. The former two have played no rôle in the work of this thesis; metabolic control analysis appears to have received much more attention, both in terms of theoretical development, and experimental application, and provides the definitions of control used here.

1.3.5 Metabolic control analysis

The fundamental, quantitative, definitions of control, from which the field of metabolic control analysis grew, were independently proposed by Kacser and Burns [63] (recently updated and re-issued [64]), and Heinrich and Rapoport [57]. The authors used different nomenclature, but this was standardised in 1985 by international agreement [17]. The original work considered the effects of enzyme activity upon linear pathways; this was subsequently extended to show its validity for branched pathways [33], for cycles [118], and finally for systems of arbitrary topology [106].

The definitions of metabolic control analysis

With respect to the control of flux, and at its most elementary, metabolic control analysis proposes two characteristics of an enzyme, local and global, and defines relationships between them.

Elasticity, ε_S^{xase}, is defined as the proportional change in the rate of the *isolated* enzyme, xase, brought about a (vanishingly) small proportional change in the concentration of metabolite, S. More formally, this is the scaled first partial derivative of the rate equation with respect to the metabolite:

$$\varepsilon_S^i = \frac{\partial v_i}{\partial S}\frac{S}{v_i} \tag{1.1}$$

where v is the activity of enzyme i, and S is the metabolite under consideration. Potentially, every enzyme can have an elasticity to every metabolite in a system. If an enzyme is entirely unaffected by a metabolite then no term for that metabolite will appear in the rate equation, and equation (1.1) will be equal to zero.

The flux control coefficient, C^J, of an enzyme is defined as the proportional change in pathway flux resulting from a (small) change in the activity of an enzyme: that is,

the scaled first partial derivative of pathway flux with respect to enzyme activity:

$$\mathrm{C}_{\mathrm{i}}^{\mathrm{J_a}} = \frac{\partial J_a}{\partial v_i}\frac{v_i}{J_a} \tag{1.2}$$

where J_a is the flux in pathway a. Equation 1.2 may be re-written as:

$$\mathrm{C}_{\mathrm{i}}^{\mathrm{J_a}} = \frac{\partial \ln v_i}{\partial \ln J_a} \tag{1.3}$$

which has the practical significance that if measurements of v_i and J_a can be made, then $\mathrm{C}_{\mathrm{i}}^{\mathrm{J_a}}$ can be directly determined as the slope of a $\ln - \ln$ plot of J_a *versus* v_i.

Values of C^{J} within a system are related to each other via the *summation theorem*:

$$\sum_{i=1}^{n} \mathrm{C}_{\mathrm{i}}^{\mathrm{J}} = 1 \tag{1.4}$$

and via the *connectivity theorem*, to elasticities:

$$\sum_{i=1}^{n} \mathrm{C}_{\mathrm{i}}^{\mathrm{J}} \varepsilon_S^i = 0 \tag{1.5}$$

Thus, in a linear system of n reactions there will be $n-1$ connectivity relationships and one summation, and, if all elasticities are known, a set of simultaneous equations can be constructed to determine values of C^{J}. An important point to be drawn from this, is that any explicit equation for C^{J}, achieved in such a fashion will necessarily contain terms originating in the kinetic equations of all other enzymes in the system, i.e. control is a system property, not an enzyme property.

In branched or cyclic pathways, the summation and connectivity theorems alone yield a singular set of equations, but additional relationships defined in [33] and [118] allow soluble sets of equations to be found under these conditions. In the case of linear pathways an increase in enzyme activity cannot result in a decrease in flux, and it therefore follows from the summation theorem (eqution 1.4), that $0 \leq \mathrm{C}_i^{\mathrm{J}} \leq 1 (i = 1 \cdots n)$. In branched pathways this is not so: increasing the activity in one limb is likely to reduce flux in the other, leading to values of $\mathrm{C}^{\mathrm{J}} < 0$. Under such circumstances other values of $\mathrm{C}^{\mathrm{J}} > 1$ do not represent a violation of equation 1.4.

A further complication in the interpretation of C^{J} values occurs in reversible pathways: it follows from equation 1.2 that as $J \to 0, \mathrm{C}^{\mathrm{J}} \to \pm\infty$ and changes sign at $J = 0$.

This can lead to an apparently paradoxical situation in the storage limb of the Calvin cycle : despite the fact that an increase in starch synthase activity can only increase flux to starch, under circumstances of net degradation $C_{Starch}^{J_{StSynth}}$ is negative.

It is not the purpose of this thesis to further expound or expand the theory of metabolic control analysis. Values of C^J can be quickly and easily determined using computer modelling techniques (described in chapter 2), and these in turn can be directly compared to experimentally determined values.

1.4 The Use of Computer Modelling

The main tool used in this thesis to investigate the behaviour of the Calvin cycle is computer modelling. The use of computers to model natural systems is by no means new and various approaches exist, but the majority of this thesis is concerned with modelling systems of biochemical reactions as sets of first order ordinary differential equations (ODEs)[3] . This is a well established general technique, dating from the time of Sir Isaac Newton, although it is only relatively recently that it has been applied to biochemical systems, and has yet to become widely established.

1.4.1 Approaches to ODE models of biochemical systems

There are two approaches to using ODE techniques to model biochemical systems. The first is to simplify the system under consideration, so as to render it analytically tractable, extract explicit solutions, and then use a computer program to calculate variables of interest over a range of parameter values. The second is to forgo the possibility of an analytical solution, construct the model in as much detail is as known, and use numerical approximation to determine variables for sets of parameter values. for the purposes of this discussion, the two will be described as simple, and detailed modelling respectively.

Simple models

It is arguable that this approach should be called something other than computer modelling: the modelling activity is performed by the human, and although in the last stage

[3]Since the main input to these models is the kinetic equations of the individual reactions, this is often described as kinetic modelling

of the process the computer alleviates the arithmetic burden, no information is generated that could not have been obtained without a computer.

As described in chapter 6 the approach can yield considerable benefit. By simplifying, it may be possible to show that behaviour observed in the un-simplified system is a general property of all systems sharing some common feature.

However, there are also a number of disadvantages. Firstly, as many features in the original system are removed, the influence of those features on the behaviour of the system cannot be assessed, although they may be known, or at least assumed to be physiologically important. In a similar vein, if some feature is assumed to be unimportant, and omitted from the model, then the possibility of observing a hitherto unsuspected rôle for that feature is eliminated.

Secondly, it is not necessarily clear which components of a complex system should be removed to create a simplified model. For example Giersch [43] considered two simplifications of the Calvin cycle , and demonstrated that not only do both of the resulting models have similar behaviour, but the underlying mathematical structure of the two is also similar, and it is thus not possible to decide which of the two exhibits more realistic behaviour.

A similar problem exists if different simplified models generate contradictory conclusions. Hahn [50] published a simplified model of photosynthesis, and claimed that photorespiration[4] was essential in maintaining stability in the system. However many models (see chapter 6) have been described which neither include photorespiration, nor report lack of stability.

Detailed models

In contrast to simple models, detailed models represent the known biochemical characteristics of the system in as much detail as possible. Clearly the model will provide a representation of the system under consideration more realistic than the simple model, but the price that is paid is the loss of analytic solutions, and hence formal proofs. Despite this intractability, numerical solutions can generally be found for such systems, via the use of standard algorithms for solving (steady-state determination), and integrating

[4]Rubisco can also catalyse a reaction between O_2 and RuBP, generating a two carbon species (glycolate) and a three carbon compound. Glycolate undergoes a series of reactions in the cytoplasm, some carbon is lost as CO_2 , and the rest ultimately converted to PGA. The process, called photorespiration, is well documented, and although its biological purpose is the subject of some controversy, it is not considered further in this thesis.

(time dependent behaviour), such systems. In this approach almost all of the mathematical effort is provided by the computer, leaving the modeler free to concentrate on the physiological relevance of the results thus generated.

Detailed modelling has been criticised on the basis that subsequent model behaviour tends to be nearly as complex as that of the subject, and nearly as hard to explain [44, 50, 14]. Although the observation may be true, the criticism should be rejected: if the *in vivo* behaviour is complex, then so must be that of a realistic model. Modelers have the advantage over their experimental counterparts of being able to alter or measure any aspect of the system, with a speed and precision that is many orders of magnitude greater than could be accomplished by other means.

Thus although, on initial investigation, model behaviour may be complicated, it is reasonable to hope that enough information may be generated to first characterise the behaviour of the model, and then by classic scientific method, test hypotheses to explain it. Most of the modelling work undertaken during this project has followed this strategy. Software requirements, and other technicalities, are discussed in chapter 2.

1.4.2 Elementary modes analysis

A recently developed technique for the analysis of biochemical systems, complementary to kinetic modelling, is based solely on consideration of the stoichiometry matrix, as described by Heinrich and Schuster [58, 125]. The object of the analysis is to identify groups of connected reactions (elementary modes) whose net flux can be altered independently of other fluxes in the system, and without effect on any intermediate concentrations. The primary motivation for the development of elementary modes analysis is the identification of discrete subsystems in very large ($\sim$ 100s) systems of reactions. In this thesis it proved to be particularly valuable in identifying those reactions which must be present in a system to allow any flux, regardless of kinetic parameters, to flow between source and sink metabolites (as described in chapter 4).

Determination of elementary modes requires the use of software designed for this purpose. One such program, *Empath*, has been written by an ex-colleague, John Woods [140], who kindly used it to provide the elementary modes results described elsewhere in this thesis.

1.5 Overview

Chapter 2 describes in detail the facilities offered by the "Scampi" software package. This takes the form of a set of "C" programming language libraries, that allow the user to make a given model the subject of arbitrary algorithms. The interface is the language, and this provides advantages of complete flexibility, cross platform portability, error recovery, and speed, over other software used for similar purposes. In the eyes of many potential users, the necessity of using "C" will present a major obstacle. Although this is a real disadvantage, it is the author's belief that the detailed examination of a model. of the complexity presented here, could not have been achieved with any other currently available software. Furthermore, even relatively simple models have the potential for such unexpectedly complex behaviour, that an interface based on a language, and not the "control panel" paradigm of most modern GUI software, appears to offer a faster and more reliable route to characterising the behaviour of models. Only when behaviour has been characterised, is it possible to offer an explanation for it.

Chapter 3 compares the advantages of stochastic algorithms over their more conventional deterministic counterparts. One particular category of stochastic algorithms, evolution strategy (ES) algorithms are discussed in more detail, and their potential use in biochemical modelling described. The chapter concludes by presenting the results of applying ES to a real-world problem : fitting the parameters of a model to time course data.

Chapter 4 starts by introducing a model of the Calvin cycle , the study of which forms the basis for most of the rest of the thesis. Results of the investigation of the model's behaviour are presented, including response to external factors, control analysis, and dynamics. Some novel behaviour, not previously reported, is described.

Chapter 5 describes the use of ES to optimise the model of the Calvin cycle according to a realistic, if simple, selection pressure: to maximise assimilation by the Calvin cycle , while simultaneously minimising the total amount of protein required for the purpose. The process results in a population[5] of optimised models, and these are characterised statistical. It is shown that the optimisation, unexpectedly, results in models that have dynamic behaviour that is quantitatively closer to experimental observation than the non-evolved model, despite the fact that dynamic behaviour played no part in

[5]In this thesis, in the context of ES, the term *population* carries its biological, and not statistical meaning

optimisation process.

Chapter 6 contains three main sections. Firstly skeleton models of the Calvin cycle cycle are investigated, with the aim of determining a model of minimal complexity, whose behaviour encompasses that of the more complete model. Next, the modelling work in this thesis is compared with other published models of the Calvin cycle . Finally, the modelling work is compared to experimental observations of the Calvin cycle .

Chapter 7, is the concluding chapter, and starts by building on the material in the previous chapter, to propose a common explanation for behaviour of the Calvin cycle as observed in both skeleton and detailed models, and *in vivo*. Possible physiological and implications of this are also discussed. Next the approaches to computer modelling used in the thesis are summarised and appraised. Finally the desirability of extending the Calvin cycle model is discussed, and in particular that of including parts of cytosolic metabolism, that might enable it to be placed within the context of the complete cell.

Chapter 2

Scampi - The Users Perspective

2.1 Introduction

When work on the model (described in chapter 4 onwards) was commenced, the SCAMP modelling package, written by Herbert Sauro [116, 117] was used. It was soon realised that, although SCAMP provides a convenient and natural format to describe the structure of biochemical systems, problems existed with the investigation of large models.

These problems were observed to fall into three separate areas: poor error handling, lack of flexibility in controlling the model once defined, and instability. A further consideration was that the software was only available for a single hardware platform. It was thought that the cause of these problems lay not in the underlying numerical algorithms, or their implementation, but in the flow of control and information between these.

It was also apparent that in order to undertake studies of a model with complex, and, at that time, unknown behaviour, the ability to describe sequences of actions, and take decisions on the basis of the results of such actions was essential. This requirement mandates the use of a language to control the model.

The considerations above led to the development the "Scampi", which formed a large part of the work of this thesis, and made possible the modelling investigations and results described in later chapters.

2.2 Overview

Scampi, as the name implies, is a direct descendent of the SCAMP[1] modelling package, written by Herbert Sauro [116, 117, 115]. It is a package providing a number of utility libraries and associated files, allowing the user to write programs in the C programming language that control models whose structure (i.e. topology, reaction kinetics, and identifier names), has been defined using SCAMP. Four libraries are supplied: Scampi, Evol, Math2, and Lists. The "Evol" library supplies functions concerned with evolution strategy algorithms, and is described in chapter 3. "Lists" and "Math2" provide low-level functionality used by the "Scampi" and "Evol" libraries, and available to the user, but a description of these is not relevant to this thesis, and is therefore omitted.

All of the modelling functionality is provided by the Scampi library. This ranges in complexity from the providing the ability to read the value of a single, named, concentration, through finding steady-state solutions and simulating models, Metabolic Control Analysis (MCA) functions, up to functions (for example) analysing the dynamic properties of a model. In most cases, such actions are performed through a single function call, and the user's knowledge of C, and general programming techniques, need not be great. Data thus generated may either be stored internally for subsequent analysis and/or saved to a file as plain (tab delimited) ASCII.

2.2.1 Generating executables using Scampi

Using Scampi keeps the model definition, and the model control phases of an investigation physically as well as logically separate. The structure of Scampi allows, but does not demand, a similar separation between the model control (data generation) and data analysis phases to be maintained. Thus, the ability to modify the model iteratively, assess the effect, and re-modify accordingly, is a straightforward programming exercise.

It is not (at present) possible to change the structure of a model, but no other restrictions are imposed. If a specific high-level function is not available, the lower-level functions are thought to be sufficiently generalised to allow it to be readily constructed.

Scampi considers that a usable model has a predefined structure, associated with sets of individually, and uniquely, named model variables. The names of the model variables are part of the model structure and are therefore constant. Functions supplied

[1] Familiarity with basic SCAMP features is assumed in this chapter.

by the Scampi library allow the user to control the model by, directly or indirectly, accessing the values of identified model variables. No other access is possible.

The structure of a model is defined in a SCAMP `.cmd` file, and SCAMP is used to process this, producing a corresponding `.lst` file[2] . Scampi converts the `.lst` file to a C source code file, containing only model definition information. The user incorporates this into their own source code which passes (references to) the model to the functions implemented in the Scampi library.

For example, consider the construction of a program called "DoThingsTo", acting on a model named "MyModel". The user must perform the following actions:

1. Define the model structure in a file `MyModel.cmd`.

2. Use SCAMP to generate the corresponding `MyModel.lst` file. The "codegenr" and "runexec" phases of SCAMP modelling can be omitted.

3. Use the Scampi parser, spicg (= Scampi Code Generator), to generate the files `MyModel.c` and `MyModel.h`, containing the C description of the model structure.

4. Write the program source code in `DoThingsTo.c`. As a minimum this source code must:

   - `#include` the header file `Scampi.h` (described in section 2.4).
   - `#include` the file `MyModel.h`.
   - Contain the `main()` function call.

5. Compile the two source files to their corresponding object files, and link these with the relevant libraries.

. With the exception of steps 1 and 4 the whole process can be conveniently described in a single `makefile` (see appendix A.2.3 for an example). Furthermore as it is likely to be necessary to perform many investigations on a model without changing the structure, the process of using Scampi to investigate model tends to be cyclical: write the controlling source code, make and run the executable, assess the results, write more source code.

[2]SCAMP produces a number of other files at this stage of processing, but it is the "`.lst`" file that provides the sole interface between SCAMP and Scampi.

2.3 The modular structure of Scampi

The Scampi library consists of a set of hierarchical and interdependent modules, covering a spectrum of levels of abstraction. Modules providing a high level of abstraction are dependent on those providing a low level of abstraction. Thus the dependencies within a user's program are determined by the highest level of abstraction required. The modules available to the user and the services they provide are, in ascending order of abstraction, as follows:

ScampiConst

(Scampi Constants) A very small module containing some fundamental and essential constants. It is described in section 2.4.

Scampi

This module provides the lowest usable level of abstraction, and, from the user's perspective, is the core module of the Scampi package. It provides mechanisms to identify and control models, and to generate output from them. The functionality supplied by this module allows the user to read and write values of model variables, to determine steady state solutions of models, and to simulate models over a period of time.

Scampi_Ute

(Scampi Utilities) Services provided by Scampi_Ute fall into two categories; The first provides mechanisms to read and write groups of model variables. The second comprises MCA and related functions.

Scampi_Dyn

(Scampi Dynamics) The module provides approaches to determining dynamic behaviour of models. One is determination of eigenvalues of the model at steady-state. The other is based on Fast Fourier Transform (FFT) of simulation results.

In addition to these modules any user program depends upon at least one model module. Model modules define the structure[3] and initial values of a model. As such they represent a specific instance of a type, `ScampiModel_t` declared in `Scampi.h` and implemented by Scampi module. They are generated using SCAMP and spicg as described above, and later in section 2.4.

[3] i.e. the *model* structure, not the *data* structure

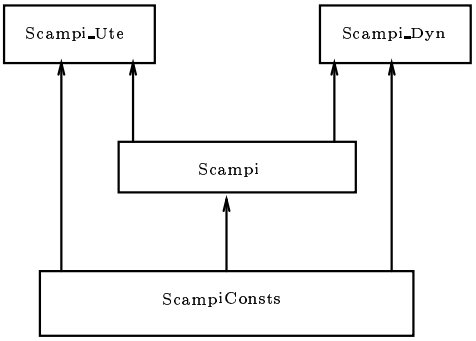

Figure 2.1: Dependencies within Scampi user modules

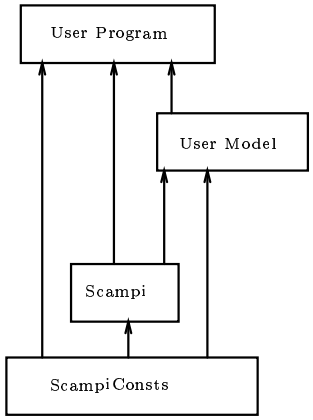

Figure 2.2: Minimal dependencies of a user program

Figures 2.1,2.2 and 2.3 illustrate dependencies within the Scampi library, a minimally, and a maximally dependent user program, respectively.

2.4 The Scampi application programmers interface

2.4.1 Header files

The application programmers interface (API) is defined in a collection of C header files, corresponding to each of the modules described above. The file `ScampiConst.h` is `#include`'d by `Scampi.h`, and may be treated by the programmer as an integral part of the Scampi module. `Scampi_Ute.h` and `Scampi_Dyn.h` do not `#include Scampi.h`, which must therefore be explicitly `#include`'d before them.

The header files contain (external) function declarations, type declarations, and constant definitions. There are no global variables. Macros are used to provide housekeeping for the compiler (e.g. conditional compilation to prevent problems caused by multiple inclusion), but do not form a part of Scampi *per se*. As long as the compiler is able to locate them, the location of the headers in the file system is not important.

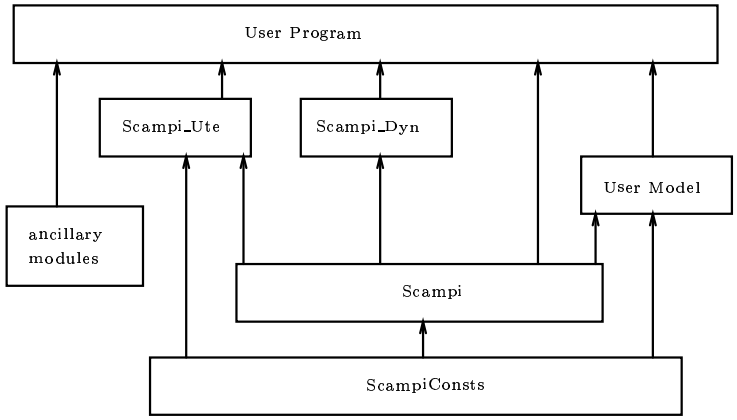

Figure 2.3: Maximum dependencies of a user program

2.4.2 Representation of models

Individual models in a user program are treated as variables of an abstract data type (ADT) via an opaque pointer as declared in `Scampi.h` :

```
typedef struct ScampiModel *ScampiModel_t ;
```

The user has no access to the underlying `struct ScampiModel`; attempts to dereference a variable of type `*ScampiModel_t` are forbidden and, in most cases, will be rejected by the compiler. The two binary operators "`=`" (assignment) and "`==`" (comparison) are legal, but rarely needed. The unary "`&`" (address of) operator is also valid, and hence normal C array manipulation is possible. As pointer semantics apply to this data type, customary care must be taken to avoid "losing" models when using the assignment operator.

Other operations (e.g. `++` etc.) on variables of this type may be syntactically valid, but are always semantically invalid. The effect of such operations is to render the model variable invalid, with respect to functions that take variables of `ScampiModel_t` as a parameter

Because the user is unable to modify the structure of a model, an initialised instance of a variable of type `ScampiModel_t` is supplied in the header file of a specific model module. In the example given in 2.2.1 the file MyModel.h contains nothing more than:

```
extern ScampiModel_t MyModel ;
```

The underlying implementation of `MyModel` is defined in `MyModel.c`, and should be regarded as private by the user.

2.4.3 Model variables

Scampi models contain four sets of model variables: Parameters, Concentrations, Velocities and, a special set containing exactly one value, Time. Thus in order to access a model variable the user must specify both the set to which it belongs, and its name. The sets are identified by use of the enumerated data type defined in `ScampiConsts.h`:

```
enum Component_id {Time, Param, Conc, Vel} ;
```

In most cases it is sufficient to refer to a set using one of the four literal values above, but the user may declare variables of `enum Component_id` if desired.

Model variable names are standard C `NULL` terminated strings of characters. Names of parameters, concentrations, and velocities correspond to those defined by the user in the `.cmd` file, Time has the fixed name "Time".

Certain Scampi functions require that a number of model variable names are specified. In these circumstances names are referred to as a `NULL` terminated array of pointers to `NULL` terminated strings.

Model variable values are declared as type `double`. Values of parameters and concentrations are initialised in the model `.c` file using initial values from the `.cmd` file (to six significant figures). Initial values of velocities are undefined. The internal precision of values is machine dependent.

2.4.4 A Trivial Example

The information presented above provides the foundation for the Scampi API. Almost all Scampi functions take some combination of `ScampiModel_t`, `Component_id`, and name. Maintaining the file names of the example given in 2.2.1, and assuming that MyModel

contains a metabolite called "Met", it becomes possible to construct a complete (if not very useful) program, that reports the initial value of "Met":

```
            /*** DoThingsTo.c - a simple but complete Scampi program ***/

#include <stdio.h>      /* because we will use printf */
#include <Scampi.h>     /* basic Scampi functions     */

#include "MyModel.h"    /* header containing declaration of MyModel */

int main(){

  double MetCon ;       /* will hold value of concentration "Met" */


                       /*  use the Scampi function GetMDval to
                           assign the value of the model variable called
                           "Met", which is a concentration in the model
                           MyModel, to the  local variable MetCon   */

    MetCon = GetMDval(MyModel, Conc, "Met") ;
    printf("Concentration of Met is %e\n", MetCon) ; /* and tell the user */

    exit(0) ;
}
```

2.4.5 Controlling output from Scampi

Although the previous example illustrates a means for communicating internal model values to the external environment, it is rather crude. The user may, for example, desire to record several groups of model variables in different files. Under these circumstances the approach of 2.4.4 will produce source code cluttered with many calls to `GetMDval()`. Furthermore, simply obtaining the value of a single model variable does nothing to assist the user wishing to maintain, in code, the logical separation between the activities of data generation and data analysis. Furthermore, as described later, this approach will not allow the user to record information generated over a simulated period of time. Scampi provides a single solution to these problems: the `OutputDesc_t` (Output Descriptor type).

Like the `ModelDesc_t` the `OutputDesc_t` is an ADT. Unlike the `ModelDesc_t`, it must be explicitly initialised at run time (with the function `InitOPDesc()`). Initialisation of a variable of `OutputDesc_t` specifies three things: The destination(s) of the model data, the source model for the data, and the names of the relevant model variables.

Information can be sent to one, or both, of two destinations: a named file, and an internal store. If a file name is specified `InitOPDesc()` will attempt to open it

for writing and, if successful, uses the specified model variable names to print column headings. Subsequent updating of a variable of `OutputDesc_t` causes the current values of the specified model variables to be printed in the corresponding column of the file. If values are to be stored internally, the same updating function adds current values to the store. The values may later be retrieved, in the order in which they were written.

To allow the possibility of simultaneously updating two or more Output Descriptors, the relevant function, `printOPs()`, takes as its parameters (a pointer to) the first element in an array of initialised `OutputDesc_t`, and the number of elements to be updated. The semantics of C pointer and array operations are such that if a single `OutputDesc_t` is to be updated, an array is not necessary, the user passes a pointer to the `OutputDesc_t` and specifies an array length of one.

2.4.6 Error handling

The Scampi library functions have been designed to ensure that as long as the user fulfils certain defined preconditions, functions always return useful information. The structure of the API is such that the user does not *demand* that a certain operation be performed upon a model, because the possibility may exist that the demand might not be possible (e.g. determining the steady state of a partially closed system). Rather the user *requests* that an action be performed, and the user is informed of the result of the attempt to fulfill the request.

Scampi functions use one of three possible strategies to respond to errors: silent ignorance, success/failure indication, and error description. Silent ignorance is the response when failure has no effect on the model, and success/failure indication when there is only one possible reason for a failure to occur. Functions not falling into these categories use error indication.

Silent ignorance is used in one context only, that of certain functions taking model variable names as their parameters. If such functions are passed a non-existent name, it is simply ignored. The behaviour is silent only in the context of the run-time environment: the function will send a message to `stderr` identifying the function and the offending variable name. This mode of behaviour was adopted because in the applications originally envisaged for Scampi, such problems would only arise as a result of mistakes in the user code. As it is not possible to write code that corrects its own bugs, there is little point in reporting the problem back to the calling code. If the

user program is large (perhaps using Scampi to provide a back-end to a program with a front-end GUI), such behaviour is less acceptable. In this case the simplest solution is test names for existence before passing them to such functions. A Scampi function exists for this purpose (see 2.5.1).

Functions that return a resource allocated by the OS (files or memory) use success/failure indication. Such functions return a pointer to the resource, and, in keeping with C convention, return `NULL` if the allocation fails.

The enumerated error type

All other functions for which the possibility of failure exists, indicate their final status via an enumerated type, defined in `ScampiConsts.h`:

```
enum SPI_err{ OK, NegConc, OutOfTol, SingMat, LsodaError, NoMem, BadFile } ;
```

These functions take as a parameter a pointer to a variable of this enumeration, whose value will be set when the function returns. The initial value of the variable is always ignored. The enumeration is ordered in increasing order of severity, and interpreted as follows:

OK
: The function call succeeded.

NegConc
: The model currently contains one or more negative concentrations. The condition can result from one of the steady-state solution functions (the simplest), or from simulation. It is frequently possible to recover from this condition by selecting a different set of starting concentrations.

OutOfTol
: The steady-state and simulation functions depend on successive approximation algorithms, and are passed a parameter, the tolerance, specifying "how close" the user wishes the approximation to be. OutOfTol is the error condition if this tolerance was not achieved. OutOfTol takes precedence over NegConc, because as far as the numerical algorithms are concerned the model is a set of differential equations whose solutions may involve negative values, although in general this

is not a solution the user wants. In contrast, if the tolerance is not achieved the function genuinely has failed.

SingMat

This is a more serious error condition in steady-state solution functions, referring to the model possessing a singular Jacobian matrix. The most likely explanation for this condition is that the model is structurally insoluble, although it may also result from a bad choice of initial concentrations models in with complex topology.

LsodaError

The name refers to the LSODA algorithm, on which the simulation function is based. LSODA was not implemented as part of this project but derived from the implementation released as part of SCAMP 2.5GA. Although it performs excellently, the structure of the underlying source code is not entirely clear. Hence, it is only possible to supply a general "failure to proceed" error condition. The most likely causes of the condition are making the step size too large when simulating, poor initial concentrations, or a structural problem.

NoMem

If a function needs dynamically allocated memory and cannot obtain it, the error condition is NoMem. If the allocated memory is to be returned to the user, the relevant pointer is set to `NULL`. By the standards of current hardware the memory requirement of Scampi programs is small, $\sim$ 100s of Kb. Hence, the occurrence of this error is most likely to be due to a memory-management problem elsewhere in the users code.

BadFile

Set by functions that attempt to open named files, and fail.

Scampi error messages

Error messages from a program using Scampi functions come from one of three groups: explicit, implicit, and low-level. Explicit error messages are output by functions written by the user that interpret values of the enumerated `SPI_err` type. In order to facilitate this `ScampiConsts.h` supplies an array of strings mapping `SPI_err` values to English:

```
#ifdef WANT_SPI_ERR_MSG

char *SPI_errMsgs[] = {
    "No Error",
    "Negative concentration",
    "Requested tolerance not achieved",
    "Newton-Raphson can't get steady state with these start conditions",
    "Simulation can't proceed from this position",
    "Out of memory",
    "Couldn't open file" } ;

#else

extern char *SPI_errMsgs[] ;

#endif
```

The purpose of the conditional compilation directive is to allow `ScampiConst.h` to be `#included` by more than one file. The user code must not `#define` the macro `WANT_SPI_ERR_MSG`, or the resultant object file will fail to link.

Implicit error reporting is performed by certain Scampi functions and is not controllable by the user. Messages of this type identify the function generating them, and some indication of the problem encountered. Implicit error messages fall into two broad categories: User warnings and debugging messages. User warnings are issued if a condition is encountered that should be brought to the user's attention, despite the fact that the relevant data structures are still intact. This is the behaviour of functions whose error handling strategy is silent ignorance. Debugging messages indicate the detection of an internal inconsistency, and in general result from either incorrect parameters being passed to a function, or previously executed code having corrupted a Scampi internal data structure.

Low level reporting originates in the numerical routines from which the integrator and steady-state solver are constructed. They are recognisable by the inscrutability of their content. In general it is safe for the user to ignore them, if the condition that triggered them represents a genuine problem this will be reported back to the user via the `SPI_err` mechanism. If a piece of user code generates a large stream of such error messages, this is probably in indication of incorrect parameters and/or corrupt memory.

2.4.7 Function specification

In order for the functions in a library to be of use, the user must be informed of their behaviour. This information should be complete, concise, and unambiguous, and not

depend on context or an implicitly assumed level of knowledge of the user. This goal is not easily achieved using "conversational" human language.

The approach used here is to specify functions using pre and post conditions, without using a formal specification language. Instead a notation comprising of commonly used symbols to represent logical and mathematical operations, and English language assertions to describe states not readily representable in terms of the symbolic operators. Although this falls some way short of a formal specification language, it does fulfill the criteria that the user is informed of the conditions that must be ensured before a given function is invoked, and the effects that such an invocation will have, in a form concise enough to be incorporated into the relevant header files, maintaining the human, and machine descriptions in the same place.

Notation used for function specification

All specifications of functions in the Scampi package are given as a comment immediately following the function declaration. The precondition starts with the string "`pre:`" and continues up to the post condition, which commences on a new line starting with "`post:`", and continues to the termination of the comment. The C stdio library function `putchar()` would thus be specified[4]:

```
putchar(char c) ;
    /* pre: TRUE
      post: c is sent to the stdout file stream */
```

The logical and numeric operators used are the C language equivalents, where these exist. Their meanings are given in Table 2.1. If (part of) the precondition of a function is that another function has been previously invoked, that function is given as a clause in the precondition. This implies that its preconditions were met, and that if the function return has a value that may be treated as boolean, that value was TRUE. The parameters for the precondition function are not defined unless they form part of the specification, for example the "`putc()`" function might be specified:

[4]the example is to illustrate the syntax of the specification. The specification itself is a simplification, because it ignores the returned value, and the possibility that the user may have previously closed the standard output.

Table 2.1: Logical and numeric symbols used in function specification

Symbol	Meaning
,	Clause separator
&&	Logical AND
||	Logical OR, Selection of non-booleans
!	Logical inverse, NOT
=>	Implies that
*	Pointer dereference when used as unary operator
* / + -	Standard arithmetic operators
^	exponentiation
< <= > >=	Standard comparison operators
==	Is equal to
!=	Is not equal to
=	Has been assigned to
. .	Range operator (as in Array[0..Max] etc.)

```
putc (char c, FILE *fp) ;
    /* pre: fp = fopen(name, mode), mode is writable
      post: c is sent to fp */
```

In this instance the usual mapping between `NULL` and `FALSE`, and `!NULL` and `TRUE` is assumed. Thus the precondition clause "`fp = fopen(name, mode)`" stipulates that not only must `fp` have been assigned the return value of `fopen()`, but also that the resulting value evaluates `TRUE` (≡ `!NULL`). The name of the file concerned is of no relevance, but it is essential that is when opened the ability to write to it was granted to the user.

If the specification imposes identical restrictions on the values of parameters passed to a function, these are grouped together inside parenthesis, and separated with commas: `(x > 0) && (y > 0)` ≡ `(x,y) > 0`.

2.5 Basic Functions

2.5.1 Determining constant characteristics of a model

The three `Component_id` values `Param`, `Conc`, and `Vel` are each associated with an internal array (or vector) of values. Although the user has no direct access to these, it is sometimes convenient to be able to determine their size. To this end Scampi provides the function:

```
extern int GetMDvecSize(ScampiModel_t md, enum Component_id comp) ;
    /* pre : md valid,
      post : GetMDvecSize(md, comp) = number of elements in md of type comp **/
```

Several Scampi functions take arrays of names as a parameter. If the user desires an array all names of a given `Component_id` they may be conveniently obtained with the function:

```
extern char **GetAllNames(ScampiModel_t md, enum Component_id comp,
                          int *n_names) ;
    /* pre : md Valid,  comp == (Param || Vel || Conc)
      post : (n_names > 0 ) => return is a null terminated array of null
               terminated strings which are all the names of type comp in
               the model.
             ( n == 0 ) => malloc fail
      */
```

The function allocates memory on behalf of the user, but once (a pointer to) it is returned, management of the allocated memory becomes the user's responsibility.

On occasion the user may wish to determine whether or not a component of a particular name exists (e.g. section 2.4.6), `ExistsMDname()` fulfils such a requirement:

```
extern int /* bool */ ExistsMDname(ScampiModel_t model,
                                   enum Component_id comp, char *name) ;
    /* pre : model is valid
      post : (ExistsMDname(model, comp, name)) =>
                model has a component of type comp called name */
```

In this case, in common with all other functions returning a boolean value, the usual C mapping between int and boolean types is assumed. `TRUE` and `FALSE` are conventionally defined in `ScampiConsts.h`.

The number of free metabolites in a model may be determined:

```
extern int GetNFree(ScampiModel_t Mod) ;
    /* pre : Mod is valid
      post : returns number of free concentrations in Mod */
```

2.5.2 Functions to read and write values of model variables

Scampi provides three pairs of functions to manipulate model variables: as individuals, as a group, or as a set.

```
extern double GetMDval(ScampiModel_t model, enum Component_id comp,
                       char *name) ;
    /* pre : model is valid
      post : if mod has a variable of type comp and name name
               value of name is returned
             otherwise a warning is sent to stderr */

extern void PutMDval(ScampiModel_t model,
                enum Component_id comp, char *name, double val) ;
    /* pre : model is valid
      post : if model has a variable of type comp and name its value is
                set to val
             otherwise a warning is sent to stderr */
```

These two functions follow the "silent ignorance" error handling strategy, which is arguably poor. The behaviour can be improved without altering the implementation, by restricting the pre-condition and simplifying the post-condition:

```
extern double GetMDval(ScampiModel_t model, enum Component_id comp, char *name) ;
    /* pre : model is valid, and has a variable of type comp and name name
      post : returns value of name */

extern void PutMDval(ScampiModel_t model,
                enum Component_id comp, char *name, double val) ;
    /* pre : model is valid and has a variable of type comp and name name
      post : value of name is set to val */
```

It is frequently convenient to refer to a number of variables as a group, saving the clutter of repeated calls to the previous functions. A pair of analogous functions dealing with subsets of values is supplied:

```
extern double *GetSubSet_md(ScampiModel_t model,
                            enum Component_id val_type, char **names) ;
    /* pre : model is valid, names is NULL terminated array of strings
      post : returns new array of copies of values of names in model
               in order corresponding to names
             NULL if no memory for return array */


extern void PutSubset_md(ScampiModel_t model, enum Component_id val_type,
                            char **Names, double *Vals) ;
    /* pre : model is valid, names is NULL terminated array of n strings,
             vals[>n]
      post : Vals[0..n-1] == GetMDVal(model, val_type, Names[0..n-1]) */
```

These also operate "silent ignorance" error handling and the previous comments on the subject apply equally to these functions.

All values of a given `Component_id` may be referred to by the functions:

```
extern double  *GetMDvec(ScampiModel_t model, enum Component_id comp,
                    enum SPI_err *err) ;
    /* pre : md valid,
      post :    (err == OK)    => (GetMDvec(md, comp) == a copy of the
                                    current comp vector in md)
                (err == NoMem) => no memory available,  returns NULL */


extern void  PutMDvec(ScampiModel_t model, enum Component_id comp, double *vec) ;
    /* pre : md valid, vec[ >= GetMDvecSize(md, comp)]
      post : values in vec copied into corresponding model values  */
```

The functions do not give information as to the size, or the order, of the arrays concerned. In general the order will not be the order in which identifiers were declared in the original SCAMP command file. Although not essential it is generally convenient to use these as a pair with the vector returned from `GetMDvec()` passed back into `PutMDvec()`. Code fragment 2.1 shows how this pair of functions may be used to perform simple, but robust error recovery.

Code Fragment 2.1 Simple error recovery (C language)

```
/****
    Assume the usual #includes and a model called mod
****/

  enum SPI_err err ;              /* error indicator */
  double *OrgConcentrationsns ;   /* where we will keep "back up" values */

                                /* read the set of concentrations in mod */
    OrginalConcentrations = GetMDvec(mod, Conc, &err) ;

    if(err != OK){
          /** panic !! couldn't get memory for OrginalConcentrations !! **/
    }
    else{
        PerformLethalFunctionTo( mod ) ;            /* could be anything */

        PutMDvec(mod, conc, OrginalConcentrations) ;  /* mod is restored */
    }
```

2.5.3 Defining and using output

As described (2.4.5), Scampi provides an ADT whose purpose is to handle output from models, the `OutputDesc_t`. Before variables of this type can be used they must be

initialised. Initialisation of an output descriptor specifies the model with which it is to be associated, the form in which output is to be recorded, and the names of the values to be recorded:

```
extern OutputDesc_t InitOPDesc(char *fname, int /*bool*/ Mem,
                               ScampiModel_t model, int /*bool*/ Time,
                               char **params, char **concs, char **vels) ;
    /* pre : model valid
      post : ( (op = InitOPDesc()) != NULL) =>
             file fname open with header described by Time, params, concs, vels.
             op is valid. ( (Mem == TRUE) iis precondition for "Read*Val()"
             functions) */
```

If the file specified could not be opened for writing `InitOPDesc()` will return `NULL`. Otherwise the function follows silent ignorance as far as names associated with `params, concs,` and `vels` are concerned. If a name is specified but not found the function will send a warning to stderr, and continue with the names that are found. The case of no names being found does not render the resulting output descriptor invalid. The empty string "" is a legal value of fname and results in no file being opened. If, in this case, the `Mem` parameter is `FALSE`, an output descriptor will result whose output goes nowhere.

Although these special-case behaviours may not appear to be useful, the general philosophy of Scampi is "Trust the user". It is not the programmer's responsibility to forbid courses of action to the user on the sole grounds that they do not appear to be useful.

Output is sent to output descriptors using the `printOPs()` function:

```
extern void printOPs(OutputDesc_t *op, int n) ;
    /* pre : n >= 0, op is an array of <= n valid OutputDesc_t
      post : op[0..n-1] updated   */
```

The case of `n == 0` is valid; in this case no descriptors are updated. Although such behaviour might also seem to be of no benefit, Scampi puts it to good use (see 2.5.5).

Two functions exist to access data that has been stored internally:

```
void Read1stValOP(OutputDesc_t op, double *result, enum SPI_err *err ) ;
    /* pre : op valid and initialised with Mem == TRUE,Δ
             result allocated to hold >= number of values declared in
              InitOPDesc()
      post : (err == OK) => result is first set of values recorded in op */
```

```
extern void ReadNextVal0P(OutputDesc_t op, double *result,
                          enum SPI_err *err ) ;
    /* pre : Read1stVal0P(op, result, error), error == OK
      post : (err == OK) => result is the next set of values recorded in op,
             (err != OK) => no more values to read */
```

Note that this is an example of a precondition (for `ReadNextVal0P()`) inheriting that of a previously declared function (`Read1stVal0P()`).

Data is saved in an output descriptor in the order in which it was received. Assuming preconditions to have been met, a typical code fragment to read all the stored data is:

```
    Read1stVal0P(op, MyResults, &err ) ;

    while(err == OK){
        DoSomethingUsefulWith( MyResults ) ;
                    /* we got the first set from Read1stVal0P */

        ReadNextVal0P(op, MyResults, &err ) ;
                    /* now get some more, if there are any */
    }
```

`Read1stVal0P()` may be called repeatedly to reread the stored data. Internally saved data may be erased, and the associated memory freed, with the function:

```
extern void FreeMem0P(OutputDesc_t op) ;
    /* pre : op initialised with MEM == TRUE
      post : internal memory freed, internally stored results destroyed */
```

After calling this function the output descriptor may be used as before. Output descriptors may be completely destroyed, and all associated resources relinquished by:

```
extern void Destroy0PD(OutputDesc_t *op) ;
    /* pre : op = Init0PDesc()
      post : op == NULL, op !valid, associated memory freed */
```

Under certain circumstances the user may wish to determine the number of values being referred to by a given output descriptor. The following is provided to achieve this:

```
extern int Size0f0P(OutputDesc_t op) ;
    /* pre : op = Init0PDesc()
      post : Size0f0P(op) ==  (number of variable names recorded by op) */
```

2.5.4 Steady-state determination

The core algorithm used by Scampi to determine steady-states is based on the Newton-Raphson method [95]. The numerical source code used is that released with the "code-genc" version of SCAMP, with only minor changes, but Scampi employs entirely different strategies for error and memory management (the latter of which is invisible to the user).

Scampi allows the user to apply the Newton-Raphson method to a model via three functions. The first makes one attempt to find a steady state and reports the result. This depends on the system being within the vicinity of a steady-state, and is therefore not very robust. The other two employ different algorithms to search for suitable starting positions for Newton-Raphson.

Direct determination of steady-states

The simplest interface to the Newton-Raphson algorithm is the function `Find_ss()` :

```
extern double Find_ss(  ScampiModel_t model, int MaxIters, double tol,
                        enum SPI_err  *err) ;
    /* pre : model is valid, MaxIters > 1, tol > 0.0
      post : (err == OK)      => model is in steady state to within
                                 tolerance tol, all concentrations >=0.0,
             (err == NegConc) => model is in steady state to within tolerance
                                 tol but contains >= 1 concentrations <0.0,
             (err == OutOfTol)=> failed to get within tol after MaxIters
                                 iterations,
             (err == SingMat) => singular Jacobian => insoluble model,
             (err == NoMem)   => Couldn't allocate memory,
             (OK <= err <= OutOfTol) =>
                                    return value is tolerance achieved */
```

`Find_ss()` performs up to `MaxIters` iterations of Newton-Raphson, stopping as soon as the requested tolerance is achieved. In the cases in which requested tolerance was not reached, the specification gives no information as to the state of the model, which is therefore undefined. This does not mean that the variable of `ScampiModel_t` is no longer valid, rather, the model does not currently hold useful information regarding concentrations and reaction velocities. Such situations may be easily redeemed using the approach suggested in section 2.1.

The specification admits the possibility of `Find_ss()` failing due to a lack of memory. In practice this is extremely unlikely. The necessary memory is allocated to the model the first time it is subjected to Newton-Raphson. Repeated invocations of `Find_ss()` do not result in any further allocation. Several other functions provided by Scampi

depend upon dynamically allocated memory. Such functions use the same internal data structures as `Find_ss()` and memory allocation is performed by the first of these to be invoked. Subsequent invocations will not result in further attempts to allocate memory, hence, if an initial call to `Find_ss()` does not result in a `NoMem` error the user may safely ignore the possibility for later functions.

A known problem with the Newton-Raphson algorithm is the fact that if the system to which it is applied is not sufficiently close to a solution, the algorithm will fail to converge. It is not possible to determine in advance how close a system needs to be to a solution in order for Newton-Raphson to converge (within a given number of iterations) , nor to calculate the distance from a solution (because knowing the distance implies knowing the position of the solution).

A second problem for the biochemical investigator is the possibility that system has multiple solutions, one or more of which contains negative values. Scampi provides two functions to overcome these problems. Both depend upon searching for a more favourable starting point for Newton-Raphson, but differ greatly in their search algorithm. Before starting the search, both functions invoke `Find_ss()`, and only continue if the error flag `!=OK`. There is therefore little point in using `Find_ss()` for routine steady-state determinations. Its intended use is for those users who wish to implement other algorithms that search for favourable starting points.

Simulating to steady state

The two algorithms for seeking suitable starting positions are simulation and Evolution Strategy (ES). The first of these attempts to find a steady-state, and, if this fails, simulates the time course of the system over a period of time. This is repeated over increasing periods of time and temporal resolution until either a solution is found, or a maximum number of attempts (specified by the user) is reached. The final values of the simulation time and number of simulation points are returned to the user, and hence the corresponding parameters are passed by reference:

```
extern void Sim_to_SS(ScampiModel_t model, double *duration, int *n_points,
                      double tol, int iters, int MaxTry, enum SPI_err *err) ;
    /*  pre : model valid, (*time, *n_points, iters, MaxTry) > 0
       post : (err == OK) => model reached steady state to within tol after
                 simulating for duration time and n_points,
                 and <= iters iterations of Newton-Raphson
              (err != OK)      => model unchanged
              (err == NegConc) => SS found, but contained -ve concs
```

```
            (err == OutOfTol)=> couldn't reach tol
            (err == SingMat) => Newton-Raphson suggests structural
                                problem in  model
            (err == LsodaErr)=> Integrator suggests structural problem
            (err == NoMem)   => Couldn't allocate memory
        */
```

Despite the extra parameters resulting in a more complex pre-condition, the post condition of the function is very similar to that of `Find_ss()`. However if the specified tolerance was not achieved `Sim_to_SS()` restores the model to its original state. Because the function exposes the model to the integrator, an additional possibility for failure arrises, as indicated by `(err == LsodaErr)`. Possible interpretations of this are discussed in section 2.5.5. Memory management is the same as for `Find_ss()`.

The total number of points simulated by `Sim_to_SS()` increases exponentially with each iteration of the `Find_ss()` - Integrate cycle. A poor choice of values for the initial duration and number of simulation points can therefore result in spectacularly bad performance from this function. The function also performs (unsurprisingly) badly if the steady-state solution is unstable.

Evolving to steady state

The alternative to simulating to a position from which Newton-Raphson proceed is to evolve to such a position using an Evolution Strategy (ES) algorithm. The basis of this is that a group of randomised starting positions are evaluated, and a proportion of the "best" positions are then used as seeds for a new set. The process continues iteratively until an acceptable solution is found, or a maximum number of iterations have been performed. The algorithm used here is based on that described by Hoffmeister *et al* [61, 10], and discussed in more detail in chapter 3. The function is specified:

```
extern void Ev_to_SS(ScampiModel_t model, double tol, int MaxGens,
                     double MuteSize, double MuteRate, enum SPI_err *err) ;
    /* pre: model valid, (tol, MaxGens, MuteSize) > 0, 0 < MuteRate <= 1.0
      post: (err == OK)          => model at steady-state within tol

            (err == NegConc)     => at steady-state but contains -ve concs
            (err == OutOfTol)    => requested tolerance not achieved
            (err == NoMem)       => couldn't allocate memory
    */
```

The parameters `model`, `tol`, and `err` carry the same meaning as those of the same name in previously described functions. The other parameters control the ES compo-

nent of the function. `MaxGens` specifies maximum number iterations of the randomise-evaluate-reseed cycle that ES performs, `MuteRate` is the mutation rate, defined as the probability of a given concentration being randomised, and `MuteSize` defines the relative size of such a randomisation.

The function returns as soon as a solution is found, and therefore reducing the value of `MaxGens` merely reduces the probability of a successful return. The parameter is nonetheless essential to ensure that the function will eventually return, but the value to which it is assigned should be a function of the speed of the machine upon which it is being used, and the length of real time that the user is prepared to wait for a result.

The `MuteRate` parameter was introduced early in the development of this function. However, subsequent work strongly suggests that there is nothing to be gained from using a value other than 1.0 for this parameter.

At present no method is known by which an *a priori* estimate of the optimum value `MuteRate` can be made. Experience suggests that the optimum value decreases with increasing numbers of concentrations, and increasing reaction elasticities. It is also possible to be reasonably sure that sub-optimal estimates will result in better performance than supra-optimal ones.

Although `Ev_to_SS()` does appear to offer a speed advantage over `Sim_to_SS()` not enough experience has been gained to be certain about this. The major difference between the two is that `Ev_to_SS()` is able to determine solutions in systems exhibiting unstable steady-states. The function does not provide information as to the stability of a solution, but the Scampi library provides the means to achieve this (see 2.6.2).

2.5.5 Simulating time dependent behaviour

Scampi provides a single function to simulate the time course of a model, `Simulate()`:

```
                                                |
extern void Simulate(ScampiModel_t model, double TimeStart, double TimeEnd,
                     int n_points, double tol, OutputDesc_t *op,
                     int n_ops, enum SPI_err *err) ;
    /* pre : 0 <= TimeStart < TimeEnd, (n_points, tol) > 0,
                op[0..n_ops-1] = InitOPDesc()
      post : (err == OK) => model simulated for n_points over
                            TimeStart..TimeEnd with output to op[0 .. n_ops-1]
                            i.e. (n_ops == 0) => no  output,
             (err == OutOfTol) => as (err == OK) but final point was out
                                  of tolerance,
             (err == NegConc)  => as (err == OK) but final point contains
                                  at least 1 -ve concentration,
```

```
            (err == LsodaErr) => Simulation could not proceed from this
                                 starting  point
            (err == NoMem)    => No memory
    */
```

The function attempts to integrate `n_points` between `TimeStart` and `TimeEnd`. At each point integrated output is sent to the array of `n_ops` output descriptors whose first element is referenced by `op`.

The core integration algorithm, Lsoda, is of the predictor-corrector type, which first predicts the value of the next point by extrapolating from previous points, calculates the first derivative of the predicted and previous points, and integrates this to provide a second estimate of the new point. The difference between the prediction and estimate is taken as an error value, which the integrator iteratively seeks to minimise[5]. The tolerance, `tol` is a measure of the maximum value of the error value that the user considers acceptable. It is not uncommon for single points near the turning point of a rapid transient to be out of tolerance, while the rest of the points of the time course are calculated satisfactorily. Hence simulation will continue if isolated points out of tolerance are encountered, although a warning will be issued on `stderr` (as an implicit error message) informing the user of the time at which the simulation was out of tolerance. However, if the last point of a simulation is out of tolerance the state of the model is no longer defined, and the user is informed of this situation by the value `OutOfTol` of the `err` parameter. If a contiguous run of five points out of tolerance is encountered `Simulate()` sets `err` to `LsodaErr` and returns, although repeated out of tolerance errors tend to be indicative of a more serious problem, resulting in premature termination before the limit of five is reached.

`Simulate()` does not check the model for negative concentrations at each step, `err` is set to `NegConc` if the final point contains one or more negative concentrations.

The Lsoda algorithm, and its implementation as used by Scampi, is extremely complex, and can fail for many reasons, as attested by its exotic and varied set of internal error messages. However, in practice, Lsoda is robust and problems tend to fall into one of two categories: a structurally incomplete model, and lack of machine precision.

If the topology or kinetics of a model is such that one or more concentrations increase unrestrainedly over time then failure at some point is inevitable (as would be the case in the real-world system). The user can diagnose this condition by inspection of the

[5] Up to a maximum of 500 iterations. This limit is not currently accessible to the user

concentration trajectories generated up to the point of failure.

The minimum achievable tolerance of Lsoda is governed by the complexity of the model, the requested step-size, and the numerical precision of the hardware upon which it is implemented. Furthermore the minimum step size is also limited by machine precision. Internally the effect of approaching these limits is to generate a detectable inconsistency (e.g. `(x + y) == x` and `y != 0.0`). Under such circumstances Lsoda sends an internal error message to `stderr`, and `Simulate()` sets `err` to the catch-all value of `LsodaErr` and terminates. The options open to the user to correct such problems are limited. The first resort is to reduce the step size, but as this has a definite lower bound, defined by the hardware, this may only achieve the effect of moving the problem from one point in the algorithm to another. The second resort is to increase the tolerance, but the best solution, if possible, is to improve the hardware.

2.6 Advanced Functions

2.6.1 Metabolic control analysis

Determination of control and response coefficients

The objective of Metabolic Control Analysis (MCA) is to relate the properties of isolated components (typically enzymes) of a system (local properties) to the overall behaviour of the intact system (global properties), and quantify such relationships.

MCA recognises two classes of global coefficient: the control coefficient, measuring the effects of small changes in enzyme activity, and the response coefficient, providing information as to the effect of small changes in any other type of system parameter, as described in the previous chapter, section 1.3.5. However, internally, Scampi maintains a single homogenous vector of system parameters, hence the distinction between the two must be left to the user.

A further refinement of the control coefficient is the *group* control coefficient (e.g. see [6,49,16], defined as the sum of effects of simultaneous small changes in more than one enzyme activity, upon defined system variables. Likewise a group response coefficient may be defined for parameters other than enzyme activity.

In order to calculate any of these coefficients Scampi must make three steady-state determinations. Although this may be computationally expensive, the additional effort needed to calculate coefficients of the specified parameters over additional variables is

negligible.

Scampi provides for the determination of coefficients of any group of parameters over any group of variable with the function ScaledSensits():

```
extern double *ScaledSensits(ScampiModel_t md, char **parameters,
                             char **variables, double tol, double perturb,
                             enum SPI_err *err) ;
    /* pre : (tol, perturb) >0.0,
             parameters is NULL terminated array of n names,
             ExistsMDName(md, Param, names[0..n-1]),
             variables is NULL terminated array of n names (all of type Vel)
             || (all of type Conc)
      post : (err == OK) => array of group coeff of params of vars is
                           returned in  the order of found in variables,
             (err == (OutOfTol || NegConc)) =>
                unable to get steady state in  tolerance with +ve concentrations,
             (err == NoMem) => couldn't allocate memory */
```

The syntax and semantics of this function are essentially the same as for previously described functions. The parameter `tol` is the tolerance required for steady-state determination, `perturb` is the relative size of δP, where P is the value of a parameter, used to perform a three point differentiation across P.

Although allocated by Scampi the returned array is in the domain of the user, as is the subsequent memory management. The memory requirement of `ScaledSensits()` over and above that of `Find_ss()` is small, and therefore, so is the chance that `err` will be equal to `NoMem`. However, if this does occur, the possibility exists that the return value will be `NULL`, and the user must check this before using the returned array.

Determination of elasticities

Because of the relatively low computational effort involved in determining elasticities, Scampi provides a single function that returns an array of elasticities of all reaction rates to a specified metabolite:

```
extern double *ElastVec(ScampiModel_t md, char *ConcName, double Pert,
                        enum SPI_err *err) ;
    /* pre : ExistsMDName(md, Conc, Name), Pert > 0.0
      post : (err == OK)    => returns vector of elasticities of all
                               velocities to ConcName in order defined by
                               GetAllNames(md, Vel ..) calculated with
                               perturbation Pert
             (err == NoMem) => Couldn't allocate memory
```

The parameter `Pert` has the same significance as its counterpart in `ScaledSensits()`.

As with `ScaledSensits()` the probability of failure to allocate memory is small, but if it does occur then the returned value may be `NULL`, and the user should check before accessing it.

2.6.2 Analysis of dynamic properties of models

Functions for the investigation of the dynamic properties models fall into two categories, response of the steady state to a (small) perturbation, and direct analysis of time dependent behaviour. The former of these is faster, but only generates useful results if the system has been brought to a stable steady state.

Impulse response

The basis of this is the determination of the eigenvalues of the system. Underlying this is the assumption that the response of a given variable in a system to a perturbation can be defined in terms of the sum of decaying exponential functions:

$$f(t) = \sum_{i=1}^{i=N} k_i e^{t\lambda_i} \tag{2.1}$$

Where N is the number of free metabolites, t is time, and k is a function of the size of the perturbation and the Eigenvector corresponding to the variable under consideration. Eigenvectors are not discussed further here. The variables λ_i are the complex eigenvalues of the system. If the imaginary part of the ith eigenvalue is non-zero, then the system will display oscillations with a period of π imag(λ_i). The real part of the Eigenvalue is τ, the damping factor, and thus in the case of a non-oscillating system with a single variable equation 2.1 simplifies to the familiar:

$$f(t) = ke^{\tau t} \tag{2.2}$$

Clearly, if the real part of any eigenvalue is greater than zero the system is not stable. This may be due to structural problems of the types discussed previously, but may also indicate that the model is in an unstable steady-state. In either case the user will need to study time course data to determine the true behaviour of the system.

In order to allow the examination of eigenvalues, Scampi defines (in `Scampi_Dyn.h`) a data type to represent complex numbers:

```
typedef struct complex{
    double real, imag ; } Complex_t ;
```

And an array of this type is returned by the function:

```
extern Complex_t *GetEigVals_md(ScampiModel_t md, enum SPI_err *err) ;
    /* pre : md at steady state
      post : (err == OK)        => returns array[0..GetNFree(md)-1] of
                                   eigenvalues of md
             (err == OutOfTol)  => returns NULL, couldn't calculate
             (err == NoMem)     => returns NULL couldn't get memory */
```

The underlying algorithm used by `GetEigVals_md()` is that described in [96]. According to [96] determining eigenvalues is potentially error prone, and the possibility exists for the calculation to fail, hence the potential for the `err` to be `OutOfTol`.

Analysis of time course data

Superficially the problem of determining the frequency of an oscillatory component in a time dependent data set would appear to be soluble by simply counting the turning points. Unfortunately the approach is rendered invalid in the presence of strong harmonics resulting in multiple turning points per cycle. Consequently Scampi uses Fast Fourier Transform (FFT) to identify relative amplitudes in a data set, and other functions using the FFT results to further characterise the behaviour. The initial FFT function is `Spectrum_md()`:

```
extern double *Spectrum_md( ScampiModel_t mod, double RunInTime,
                            double SampleTime, int elen, char *VarName,
                            OutputDesc_t *u_op, enum SPI_err *err) ;
    /* pre: (RunInTime, SampleTime, ExpSamplePoints) > 0, u_op not
                initialised, VarName is name of Conc or Vel in mod
      post: (err == OK) =>
        returns array[0..(2^elen)-1] containing spectrum of VarName
        between RunInTime and RunInTime+SampleTime as real-imag pairs,
        op contains the time course data
            (err != OK) => as for Simulate()
 */
```

`Spectrum_md()` simulates over an initial time, `RunInTime` to allow unwanted transients to diminish, and then simulates again over the length of `SampleTime`, determining $2^{\texttt{elen}}$ points [6]. FFT is performed on the resulting data set as described in [97]. Even

[6] FFT demands that the size of data set is a power of 2

valued (real) elements of the returned array contain amplitude information and odd (imaginary) valued elements phase information at frequencies of $\frac{n/2}{\Delta}$ where `n` is an index into the array, the numerator is the result of *integer* division, and Δ the time between sample points. On return the output descriptor, `u_op`, contains the values of time and `VarName` stored internally.

The spectral information returned by `Spectrum_md()` is not particularly convenient for the biochemical modeler, and so Scampi provides a function to extract information and convert it into a more useful form:

```
extern double NiceSpec(double *spec, int elen, double SampleTime,
                       double thresh) ;
    /* pre: spec = Spectrum_md(... SampleTime, elen  ...),
          0.0 < thresh < 1.0
    post: returns fundamental freq in spec, and converts spec to
            freq-amplitude pairs as normalised absolute real values,
            spec[0..1] undefined
    */
```

`NiceSpec()` converts the `spec` array into frequency-amplitude pairs, even elements holding the frequency. The odd (amplitude) elements are normalised against the largest amplitude found at non-zero frequency. The frequency value of the lowest element whose amplitude value is greater than or equal to `thresh` is returned as the fundamental. The first two elements of the spectrum contain zero frequency (i.e. D.C.) information. When a time course contains a significant offset (common in biochemical models) this will swamp the superimposed oscillations. Therefor `NiceSpec()` ignores these elements which are hence undefined on return. A typical code fragment to print the information in a format suitable for display with the software of choice might be:

Code Fragment 2.2 Simple FFT demonstration (C language)

```
                                                          |
  const int elen = 12           /* length of array will be 2^12 = 4096 */
  int n, len ;
  double *spec ;                          /* this will hold our spectrum */

    len = ipow(2, elen) ;
        .
        .
        .
    spec = Spectrum_md(.... elen ....) ;
    NiceSpec(spec, elen, ...) ;
                           /* we'll ignore the return in this example */
    for(n = 2 ; n < len ; n+=2)             /* scan spec in steps of 2 */
        printf("%e\t%e\n",       /* print on stdout, user can redirect */
```

```
            spec[n],                /* even indices give the frequency */
            spec[n+1]) ;                 /* and odd ones the amplitude */
        .
        .
        .
```

The user can further process the information in `u_op` if desired, but a single function, utilising `Spectrum_md()` and `NiceSpec()`, is provided to fully characterise the dynamics of a model exhibiting periodic behaviour:

```
extern void DynCharacter(ScampiModel_t mod, char *VName, double RunInTime,
                  double SampleTime, int elen, double thresh, /*input*/
                  double **spec, double *Freq, double *Ampl, /*output*/
                  double *Offset,double *tau, enum SPI_err *err) ;
    /* pre: mod valid, VName is a variable name in mod,
            (RunInTime,SampleTime,elen) > 0, 0 < thresh < 1.0
       post: (err == OK) => mod simulated over RunInTime to
                 RunInTime + SampleTime,
                 Freq, Ampl, Offset, tau contain fundamental freq,
                (RMS) amplitude, offset and time constant tau, respectively
             spec[2..(2^elen)-1] spectrum as freq-ampl pairs
             (err == LsodaErr) => unable to simulate from starting point
             (err == OutOfTol) => time course to complex for FFT
             (err == NoMem)    => couldn't allocate memory
    */
```

Comparison of impulse response and FFT analysis

Although the `DynCharacter()` performs adequately when exposed to ideal test data, its performance (i.e. accuracy of results) rapidly degrades unless the magnitude of τ is (3 or 4 times) less than the period. Whilst it would certainly possible modify the implementation to gain some improvement in this respect, the use of FFT imposes an absolute limit of 2Δ upon the resolution with which the period may be determined. As the value of the period is the starting point for calculating the other variables, obtaining results by this approach, whose accuracy is comparable to other Scampi functions will require very large ($\sim 10^6$) numbers of points, with a concomitant penalty in execution time and memory requirement.

Even if a relatively low resolution is acceptable, `DynCharacter()` still exacts a considerable cost in terms of execution time. Comparison of the two approaches (FFT and impulse), specifying a sample size of 2^{12} for `DynCharacter()`, resulted in an execution time $\sim$ 100 times greater for `DynCharacter()` than for `Ev_to_SS()` (tol = 10^{-6}) followed by `GetEigVals_md()`.

It would appear therefore that the only justification for using `DynCharacter()` is in cases where the system exhibits unstable, but periodic behaviour.

2.7 Testing and concluding remarks

The software described in this (and the next) chapter has, at time of writing (November 1998), been in regular use for about three years. It has been compiled and used, without modification to the source code, on Commodore Amiga, Sun/Sparc, and PC platforms, using the gcc compiler (Amiga and Sun) and Microsoft C/C++ (PC).

Results from using Scampi on these platforms, and when possible from SCAMP, have always been consistent. In addition to routine testing at the time of construction, Scampi has also been tested by demonstrating that error in evaluating the summation theorem is acceptably low. An example of such a test is given below:

```
/*
MGP 15/1/96
demonstrate summation theorem
over all inputs and outputs from the calvin cycle, over range of external Pi
*/

#include <stdio.h>
#include <string.h>
#include <Scampi.h>         /* basic modelling functions */
#include <Scampi_Ute.h>     /* contains MCA functions    */

#include "calvin0.h"        /* header file for the Calvin0 model */

int main(){

  const int n_params = 10, n_vels = 5, iters = 50 ;                    |
      /* number of parameters, reaction velocities and Newt iterations */
  const double tol = 1E-6, perturb = 1E-6 ;
              /* simulation and SS tol, perturbation for MCA function */
  char  *params[] = { "Rbco_vm", "FBPase_ch_vm", "SBPase_ch_vm",
                      "Ru5Pk_ch_vm", "TP_Piap_vm" ,"LR_vm", "Vstsyn_ch",
                      "EQMult", "" },
                                                    /* Parameter names */
        *vels[] = { "Rubisco", "TP_Pi_apPGA", "TP_Pi_apGAP",
                    "TP_Pi_apDHAP","St_synthase", ""},/* velocity names */
        fname[80] ;                                 /* output file name */

  double *results, Pi_cyt,  timestart = 0.0, timend = 2.0 ;
  enum SPI_err err ;             /* error return from Scampi functions */

  int vel_count, n ;
  OutputDesc_t OP ;
  FILE *fp ;

    strcpy(fname, *argv) ; strcat(fname, ".out") ;
    fp = fopen(fname, "w") ;                 /* open a file for output */
```

```
    timestart = 0.0 ; timend = 2.0 ;
    Simulate(calvin0, timestart, timend, 400, tol, &OP, 0, &err) ;
                                                        /* (no output) */

    for(Pi_cyt = 0.2 ; Pi_cyt < 0.7 ; Pi_cyt += 0.1){
                                                  /* loop over range of */
        PutMDval(calvin0, Param, "Pi_cyt", Pi_cyt) ;    /* external Pi */
        Simulate(calvin0, timestart, timend, 400, tol, &OP, 0, &err) ;
                                             /* pre simulate - no output */
        results = ScaledSensits(calvin0, params, vels, tol, perturb, &err) ;
          /* group fcc of all reactions over all input and output flux */
        if(err) fprintf(stderr, "suspect result for Pi_cyt = %e\n", Pi_cyt) ;
                                               /* warn of problems */
       for (vel_count = 0 ; vel_count < n_vels ; vel_count++ )
            fprintf(fp, "%e\t", 1.0 - results[vel_count]) ;
                                /* send summation theorem errors to file */
        fprintf(fp, "\n") ;
        free(results) ;                          /* relinquish memory */
    }
    exit(0) ;
}
```

This generated the following output:

```
-6.487926e-10   -1.018172e-10   -2.081233e-10   -2.951179e-10   -3.553318e-09
 2.719526e-10   -8.385226e-11    1.340357e-10    2.802697e-10    4.872608e-10
-1.544049e-10   -8.801493e-11   -2.180089e-10   -1.297609e-10   -7.952063e-10
 1.042184e-10   -9.207124e-11    1.412002e-10    9.298351e-11    1.087578e-10
-2.902469e-10   -6.115841e-11   -1.498925e-10   -4.338641e-11    4.081349e-10
 1.454329e-10   -1.967826e-10   -5.577405e-11    6.828738e-11   -4.252267e-10
```

Which has a worst absolute error value $< 3.6 \times 10^{-9}$, and is probably acceptable for most purposes. No problems have been encountered with other testing during the project, and Scampi has proved adequate for all the modelling work described in the rest of this thesis.

Chapter 3

Applications of Evolution Strategy Algorithms to Biochemical Modelling

3.1 Introduction

In this chapter the general features of stochastic search algorithms (SSAs) are introduced by a comparison with the more common iterative successive approximation. Several variants of SSA are outlined, and one, evolution strategy (ES), discussed in more detail.

The API to a library providing ES functions is presented, and this library is used to construct a simple test program demonstrating the ability of ES algorithms to tackle problems posing difficulty for more conventional algorithms.

The relevance of ES algorithms to metabolic modelling is discussed, and illustrated firstly with a description of the Scampi `Ev_to_SS()`[1] function, and then by considering a program using Scampi and ES library functions to fit the parameters of a model to observed real-world data.

[1] See section 2.5.4

3.1.1 Successive approximation algorithms

Many programming problems exist for which an solution cannot be directly computed; instead, the problem is solved by an iterative process of approximation followed by an error correction step to generate a new approximation. This is commonly used if a solution to the problem is (or can be arranged to be) expressed in the form

$$\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X}), |\boldsymbol{Y}| \rightarrow \mathbf{0} \tag{3.1}$$

where $\mathbf{X}$ is an approximate solution to the equation, and $\mathbf{Y}$ contains the error associated with the approximation. In the current context it does not matter whether or not $\mathbf{f}(\mathbf{X})$ has an exact solution, the goal is to minimise $\mathbf{f}(\mathbf{X})$. Assuming this to be the case, an algorithm to realise equation 3.1 may be described in pseudo-code as follows:

Code Fragment 3.1 *General Successive Approximation (Pseudo-code)*

```
1    initialise X
2    Y = f(X)
3    Error = ErrFunc (Y)
4    WHILE ( Error > Tolerable ) AND ( Wish to continue ) DO
5        X = CalcNewX (X, ...)
6        Y = f(X)
7        Error = ErrFunc (Y)
     END WHILE
```

The function ErrFunc () in line 3 converts the error vector, $\mathbf{Y}$, into the scalar value Error, which may then be conveniently compared with the maximum error that the user is prepared to accept, Tolerable. Achieving this goal for the error value is one of two termination criteria expressed in line 4, the second, (Wish to continue) is necessary to ensure that the algorithm will terminate, even if no acceptable solution can be found, and at its simplest need only compare the number of loop iterations with some maximum value.

Distinction between varieties of successive approximation algorithm is made by consideration of the algorithm implemented by CalcNewX () function at line 4. In addition to the current best approximation, $\mathbf{X}$, the function will require other information, characteristic of the particular algorithm, as indicated by the ellipsis parameter. There are two broad classes of strategy for generating an improved approximation: those that (attempt to) bracket a solution and update $\mathbf{X}$ so as to reduce the range of the brackets (e.g.

root finding by bisection), and those that use derivative information to determine the "downhill" direction of $\mathbf{X}$ (e.g. Newton's method, Simplex optimisation[2]). Although well established, all such algorithms suffer from well known problems:

1. Execution time increases rapidly with dimensionality; the rate of increase for algorithms using derivative information is $O(N^2)$, and $O(2^N)$ for those using bracketing, rendering the latter unfeasible for all but very low dimension problems.

2. All are sensitive to initial conditions, resulting in (at least) two possible pathologies:

   - The algorithm fails to progress. If the hyper-plane defined by the starting points of a bracketing algorithm changes monotonically, the algorithm will converge to one of these points. If an unfavourable starting position is selected for Newton's method, the algorithm may fall into a limit cycle, or wander unpredictably in the space defined by $\mathbf{X}$.
   - If $f(\mathbf{X})$ If the search area contains local minima the algorithm may converge to one of these, rather than the "true" global minima.

3.2 Stochastic Search Algorithms

SSAs are a group of algorithms designed to overcome such problems. These algorithms also fit into the framework outlined in code fragment 3.1, the crucial difference between SSAs and more conventional successive approximation algorithms is that the CalcNewX () function (line 5) in an SSA is not purely deterministic, but relies on some form of random sampling of the search volume.

The randomised nature of SSAs means that their ultimate success or failure is much less dependent on initial conditions; the long term trend in $\mathbf{f}(\mathbf{X})$ will be down, although the rate, and in particular the initial rate, of convergence may be slow. The random component of the algorithm also allows the possibility of escape from local minima, although whether or not this is achieved within a useful time depends upon the individual implementations.

[2] Although this algorithm does not use explicitly calculate $\boldsymbol{\partial f(X)/\partial X}$ it nonetheless estimates the size and direction of the slope of $\boldsymbol{f(X)}$ from N independent evaluations of $\boldsymbol{f(X)}$

SSAs proceed by maintaining a population[3] , $\mathbf{P}$ of λ instances of $\mathbf{X}$. Each of these is evaluated, and a sample of μ individuals from $\mathbf{P}$ is taken and used to regenerate a new population[4] . The process generally ends when one individual is found that is considered to be an adequate solution. Bearing these points in mind, the pseudo-code for general successive approximation can be extended to describe a general SSA as follows:

Code Fragment 3.2 *General SSA (Pseudo-code)*

```
1     create P of λX from X_initial
2     Y = f(X_initial)
3     Error = ErrFunc (Y)
4     WHILE ( Error > Tolerable ) AND ( Wish to continue ) DO
5         X = CalcNewX (P, f, ErrFunc, ...)
6         Y = f(X)
7         Error = ErrFunc (Y)
      END WHILE
```

It should be noted that in line 5 the function CalcNewX () requires the function $\mathbf{f}()$ and ErrorFunc () in order to update $\mathbf{P}$. This line may then be further refined:

Code Fragment 3.3 *Refinement of* CalcNewX () *(Pseudo-code)*

```
5.1     select μ fittest survivors, S, from P
5.2     FOR Child = FirstChild TO λ DO
5.3         Select parent(s) from S
5.4         P[Child] = copy of parent(s)
5.5         Mutate P[Child]
        END FOR
5.6     Calculate fitness of all individuals, n, in P as ErrorFunc ( f(P[n]))
5.7     Sort P by fitness
5.8     RETURN( P[0])
```

Each invocation of this function is referred to as a generation. In the form given above, the assumption is made that in the first invocation $\boldsymbol{P}$ is already sorted by fitness.

Various forms of SSA differ primarily in the manner in which lines 5.3 (selection), 5.4 (reproduction), and 5.5 (mutation) are implemented. The calculation of fitness in line 5.6 is not a part of the algorithm *per se.* but is supplied by the user. For the sake of convenience, the function ErrorFunc ($\mathbf{f}(\mathbf{P}$[Child])) will be referred as the *fitness evaluation function*, $F()$.

[3]In this context "Population" carries its biological, not statistal, meaning.

[4]This nomenclature was used in [61] to describe ES algorithms. For the sake of consistancy, it is used here, with the same meaning, for all SSAs.

3.2.1 Simulated Annealing

The earliest SSA to be described is simulated annealing, attributed to the work of Metropolis *et al.* in 1953, by Press *et al.* [98]. As the name implies, the algorithm is intended to model the process observed in materials undergoing liquid-solid phase transition. At the molecular level the arrangement of atoms or molecules is determined by two opposing forces, intra-molecular attraction and thermal energy. The effect of the first of these is to cause atoms (or molecules) to associate in crystalline aggregates, and the latter to cause atoms to be lost from such aggregates. Thus for a given material the maximum attainable size of a crystal (which is equivalent to minimum energy) is determined by temperature. If temperature is rapidly reduced to below freezing point the resulting solid will consist a mass of small crystals. However if temperature is reduced slowly enough all atoms can aggregate into a single crystal, representing the minimum attainable energy of the system.

In implementations of simulated annealing the formation and disruption of aggregates is represented by randomisations of $\mathbf{X}$. The resulting energy of the system is considered to be $\mathbf{f}(\mathbf{X})$. Selection is based on the Boltzmann energy distribution, $Prob(E) \sim e^{-E/kT}$ where E is energy, T temperature, and k Boltzmann's constant. In Metropolis' algorithm a new $\mathbf{X}_n$ is always selected as the seed for generation $n+1$ if it is fitter (i.e. $\mathbf{f}(\mathbf{X})$ is smaller) than $\mathbf{X}_{n-1}$. If $\mathbf{X}_n$ is less fit then it is selected if $e^{-\Delta E/T} > U(0,1)$ where $\Delta E = \mathbf{f}(\mathbf{X}_n) - \mathbf{f}(\mathbf{X}_{n-1})$, and $U(lo, hi)$ is a random real number, uniformly distributed between lo and hi.

Regeneration of $\mathbf{P}$ is by randomisation of $\mathbf{S}$. The precise nature of the randomisation is determined by the problem domain. The major difference between simulated annealing and the algorithms described below is that simulated annealing algorithms must maintain a value for the current temperature, T, and arrange for this to decrease monotonically.

3.2.2 Evolution Strategy and Genetic Algorithms

Just as simulated annealing is based upon a model of a natural physical process, Evolution Strategy (ES) and Genetic Algorithms (GAs) are based upon a biological one: the fittest individuals in a population (are more likely to) survive to produce offspring, and these offspring are generally similar, but not necessarily identical, to their parent(s). The model of evolution is based upon three assumptions:

1. An organism's fitness is determined by the interaction of its phenotype with the environment.

2. The phenotype is determined by interaction of sub-cellular machinery with information encoded in the organism's DNA.

3. An organism's genotype is a mixture of copies of parental genotypes (recombination), but the copies may not always be exact (mutation).

Although this description will be immediately familiar to anyone with even the most limited biological experience, the algorithms described below were developed as a result of their intrinsic interest for theoretical computer scientists, and their potential application to problems mainly within the fields of electrical and electronic engineering. Despite the biological inspiration, a recent search of the "BIDS" database for literature published since 1990 identified a total of some 60 articles describing work using these algorithms, but none of these were applied to a biological problem.

Genotype Coding

In the context of ES/GAs, each individual of a population possesses a *genome*, $\mathbf{G}$ which contains an encoding of the parameter vector $\mathbf{X}$, and one the major perceived differences between ES and GAs lies in the nature of this coding.

In a GA [136] implementation, the genome is considered to be a bitstring of constant size. Inasmuch as the concept of an individual gene exists, each gene is represented by a single bit. The mapping of the genome into $\boldsymbol{X}$ is the responsibility of the user, the genome is used to represent arbitrary data.

In an ES algorithm the genome is considered to be a set of real numbers, of constant size. In this case individual genes each have their own value, which it is assumed, represents some actual quantity in the problem domain. Hoffmeister and Bäck [61] describe this difference as being equivalent to the difference between genotype and phenotype information. While this is a debatable point from the molecular biological view, within the context of this thesis it is more convenient to describe any particular instance of a genome as the *genotype*, and the *phenotype* as $\boldsymbol{f(X)}$, or some property derived therefrom.

Selection

Having evaluated the fitness of every organism in $\boldsymbol{P}$, and identified the fittest μ individuals as the seed population $\boldsymbol{S}$, it is necessary to select parents for each individual that is to be created in the next generation of $\boldsymbol{P}$.

There exist several possible strategies for selecting the parents of the next generation, but all can be broadly categorised as proportional or ranked selection. In a proportional selection scheme the probability of any individual in $\boldsymbol{P}$ being selected for $\boldsymbol{S}$ is proportional to its fitness. If ranked selection is used then a fraction (determined by the user) of the fittest individuals form $\boldsymbol{S}$; by convention individuals are selected from $\boldsymbol{S}$ with uniform probability [10], although other strategies have been described [83, 136].

The identification of $\boldsymbol{S}$, and the selection of parents from $\boldsymbol{S}$ to generate individuals in the next generation is described here as two separate processes, although this distinction may not as clear if a proportional parental selection scheme is used

One of the important distinguishing features between GA and ES algorithms is the way in which this step is implemented. In GA implementations it is mandatory for all individuals to have (at least) two parents. In ES algorithms individuals generally have only one parent.

Reproduction

Once parents for a new individual have been selected, then this offspring must be generated from them. If there is only one parent then the offspring is simply a copy of the parent.

In the case of more than one parent then three general recombination strategies exist:

1. Individual genes in the offspring are selected at random from each parent.

2. Genes are continuously selected from one parent until a low probability random number is generated, at which point genes are selected from the other parent. The second parent contributes genes. Contributing parents continue to swap in this fashion until the new genome is complete.

3. If genes are (or represent) real values then offspring genes may be calculated as an average of the parents' gene values.

All of these strategies may be further modified by introduction of a weighting scheme, based upon the relative fitness of the parents. As the work in this thesis has only used the one parent reproduction, these points are not considered further.

Another important distinction to be made between different implementations of SSA is whether or not $\boldsymbol{S}$ goes forward unmodified to become a part of $\boldsymbol{P\prime}$ (where $\prime$ denotes next generation). The advantage of allowing this is that the overall fitness of $\boldsymbol{S}$ is guaranteed to increase monotonically, leading to more rapid convergence. The counter-argument to allowing this is that it leads to the population representing a less uniform sample of the genotype space, maybe subject to premature convergence, and will perform less well in situations in which the fitness evaluation function is either noisy or variable. Using the ES nomenclature of [10] the former strategy is described $(\mu + \lambda)$, and the latter as (μ, λ).

Mutation

Although it is possible to implement SSAs without recombination, all must have some form of mutation. As with recombination the details of the implementation are dependent on the coding scheme for $\boldsymbol{G}$, but only real valued genotypes are considered here. In this case several possibilities for mutation present themselves, the simplest of which is to add a random number to each gene. By convention this normally distributed, (although Levine [83] allows a variety of other distributions).

$$\mathbf{G}_i' = \mathbf{G}_i + N(0, \sigma), 0 < i < n \tag{3.2}$$

Where $N(x, \sigma)$ denotes a normally distributed random number. A more powerful variant of equation 3.2 allows not only the mutation of individual gene values, but also the size of such mutations (mutability). Under these circumstances the genotype consists of two vectors, the genotype values, $\boldsymbol{X}$, an the size of mutation, $\boldsymbol{\sigma}$, to which each element of $\boldsymbol{X}$ may be subject:

$$\left.\begin{array}{lcll} \boldsymbol{\sigma}_i' & = & \boldsymbol{\sigma}_i . e^{N(0,\Delta\sigma)} & (a) \\ \boldsymbol{G}_i' & = & \boldsymbol{G}_i + N(0, \boldsymbol{\sigma}_i') & (b) \end{array}\right\} 0 < i < n \tag{3.3}$$

Where $\Delta\sigma$ is a global parameter. The form of equation 3.3a has two important characteristics. Firstly (assuming $\boldsymbol{\sigma}_i > 0$ in generation 0) it guarantees $\boldsymbol{\sigma_i'} > 0$, secondly

it ensures that, unless a selection pressure is operating, $\boldsymbol{\sigma_i}$ will remain constant in the long run (because probability of multiplying $\boldsymbol{\sigma_i}$ by x is equal to the probability of multiplying by $1/x$.

The advantage of this scheme over that of equation 3.2 is that mutation sizes $\boldsymbol{\sigma_i}$ can adapt according to the sensitivity of $\boldsymbol{F()}$ toward $\boldsymbol{G_i}$, resulting in faster convergence. The improvement thus obtained is related to the complexity of $\boldsymbol{F()}$ and under some circumstances appears to lead to a qualitative as well as a quantitative improvement in fitness (see Figure 3.2).

3.3 Implementation of ES by Scampi

The Scampi package provides a single module, `Evolve`, implementing $(\mu + \lambda)$ES algorithms. The user interface is defined in the header file Evolve.h . As with the other Scampi modules the interface comprises type definitions and functions acting upon those types. In contrast to the other modules, the `Evolve` interface is not defined in terms of ADTs, consequently allowing much greater public access to data structures. This reflects the relatively underdeveloped nature of this module; at a later stage these structures will be encapsulated, and the possibility of direct access eliminated.

The interface defines a single structure type, representing an organism (a population is an array of these), and two groups of functions acting upon them. The first of these comprises utility functions, used mainly to initialise organisms and populations. The second comprises functions applying $(\mu + \lambda)$ES to populations. The major difference within the latter group lies in the termination criteria, and hence the applications to which they are most suited. Other differences are concerned with the relative degree to which the burden of memory management rests with the user.

3.3.1 Maintaining organisms and populations

The organism type

All ES functions in the `Evolve` module require a population of organisms to act upon. Currently a population is simply defined as an array of organisms (`Organism_t`), where `Organism_t` is defined:

Code Fragment 3.4 `Organism_t` type definition (C language)

```
typedef enum {FitnessHigh, FitnessLow} Fitness_t ;
typedef enum {FixMute, VarMute } StratSet_t ;

typedef struct OrgStruct {
    double      *Genome ;       /* array of gene values */
    double      *MuteSize ;     /* size of mutation of individual genes */
    int         LenGenome ;     /* length of genome */
    StratSet_t  Strat ;         /* fixed or variable size mutations */
    Fitness_t   FitHiOrLo ;     /* high fitness value is more or less fit */
    double      FitnessVal ;    /* fitness value */
    void        *user ;         /* points to whatever the user wants */
} OrganismStruct, *Organism_t;
```

The enumerated `Fitness_t` allows the user to specify whether a high fitness value is regarded as more or less fit than a low value. This in turn depends upon the problem to which ES is being applied[5] . The second enumerated type, `StratSet_t` allows the user to specify fixed or variable mutation strategies, as described by equations and 3.2 and 3.3. If the latter is selected then the `MuteSize` field is used to hold $\boldsymbol{\sigma}$.

Although the user may declare structures as `OrganismStruct` for their own purposes, the functions within `Evolve` all operate upon variables of `Organism_t`, or arrays thereof. Several functions[6] are provided to generate new `Organism_t` of which the most useful is `NewOrg3()`:

```
extern Organism_t NewOrg3(int LenGen, double *Adam, double InitMuteSize,
                  Fitness_t ft, StratSet_t strat) ;
    /* pre : Adam[LenGen], InitMuteSize >0.0
      post : return (!NULL) =>  New org with Genome as a  copy of Adam and
                                MuteSize[n] = Genome[n] * InitMuteSize ;
                  (NULL)  =>  couldn't get dynamic memory          */
```

This initialises an `Organism_t` that is valid as a parameter for any function with formal parameters of this type. The elements the `MuteSize` array (i.e. the individual $\boldsymbol{\sigma_i}$) are initialised using `InitMuteSize` to calculate relative mutation sizes. This is to ensure that in instances in which individual gene values (i.e. elements of `Genome` $\equiv \boldsymbol{G_i}$) cover a wide range, genes with high values are subject to mutations of reasonable size, while maintaining those with low values within a reasonable range. Initialisation of the `MuteSize` array takes place even if the mutation strategy is set to fixed mutations, but its contents are ignored unless the `Strat` field is set to `VarMute`. If desired the user may change the value of this field during the course of an organism's life.

[5]Previous discussion has assumed the goal of minimising $F()$, maximising is achieved by reversing the order of the sort at line 5.7 in code fragment 3.3.

[6]The complete interface, `Evolve.h` may be found in appendix B of which a representative sample is presented here.

Generating populations from organisms

Once an organism has been thus initialised, copies of it may be generated by copying or cloning. The difference between the two is that copying sets the field values of an existing destination organism to those of a source. Cloning generates a new organism whose field values are equal to those a parent. Cloning when copying should have been used results in a memory leak. Copying instead of cloning renders subsequent execution of the program undefined.

```
extern Organism_t CloneOrg(Organism_t Orig) ;
    /* pre : Orig is valid
      post : returns exact copy of Orig else NULL if fail */

extern void CopyOrg(Organism_t src, Organism_t dst) ;
    /* pre : src and dst valid and have equal length genome
      post : dst is a copy of src */
```

It is then a trivial matter to initialise a new population; C code fragment 3.5 shows an example.

Code Fragment 3.5 Initialising a population for ES (C language)

```
  const int n_orgs = anything /* > 1 */, /* num of orgs in population */
          n_genes = anything /* > 0 */ ;    /* num of genes in genome */

  const double InitRelMuteSize  = anything  /* > 0.0 */ ;
                                        /* initial relative mute size */
  int n ;                                     /* index for population */

  double InitGeneValues[ LenGenome ] ;

  Fitness_t FitnessType ;            /* high or low values are fitter */
  StratSet_t strat ;                   /* fixed or variable mutations */
  Organism_t Population[ n_orgs ] ;

/**
   .
   initialise InitialGeneValues, FitnessType and strat here
   .
**/

    Population[0] =                          /* create first organism */
        NewOrg3(LenGenome, InitGeneValues, InitMuteSize, FitnessType,
                    strat) ;

    for(n = 1 ; n < n_org ; n++)  /* clone the rest of the population */
        Population[n] = CloneOrg(Population[0]) ;
```

Under certain circumstances the user may consider that the characteristics of an organism render it so unfit as to have zero probability of surviving into the next generation, regardless of the state of the rest of the population. To this end the function `KillOrg()` is provided :

```
extern void KillOrg(Organism_t Victim) ;
    /* pre : Victim valid
      post : Victim's fitness val equal deadval according to fitness type */
```

If need be the ES functions described below will reduce the value of μ if this fate has befallen more than λ individuals.

Death, as far as the `Evolve` module is concerned, is not equivalent to destruction: dead organisms remain valid, may be subject to the same functions as live ones, and still have memory allocated to them. In order to release this memory organisms must be destroyed, individually or *en mass*:

```
extern void DestroyOrg(Organism_t Victim) ;
    /* pre : Victim is valid
      post : Victim is not valid, associated memory freed */

extern void DestroyPop(Organism_t *Pop, int n_org) ;
    /* pre : Pop[0 .. n_org-1] are valid
      post : Pop[0 .. n_org-1] not valid, associated memory freed */
```

3.3.2 Evolving populations

In order to subject an initialised population to ES functions, the user must supply the fitness evaluation function, $F()$, with a type specified as:

```
typedef void (*FitnessEval_ft)(Organism_t org) ;
    /* pre : org is valid
      post : org->FitnessVal updated, other fields in org unchanged */
```

The specification presented here has been made as restricted as possible: the only field in the organism's structure that may change is the fitness value. In fact the behaviour of the ES functions remain predictable if any of the other fields, except `FitHiOrLo`, are modified. However, under such circumstances the functions will (probably) no longer be implementing ES, at least in the sense described previously. Of particular note is the behaviour if the fitness evaluation function is allowed to modify the genome in some way, prior to updating `FitnessVal`. Whitley [136] describes this as a hybrid algorithm, and suggests that it is equivalent to Lamarckian inheritance (because the genome is affected by the experience of the organism). Under some circumstances, for example in section 3.4.1, such hybrid algorithms show marked improvement over their purer counterparts. Apart from these considerations, there is no restriction on the behaviour of fitness evaluation functions, beyond those mandated by good programming practice.

`Evolve` provides functions to allow the user to exploit ES to two different ends, loosely defined as exploration, and solution. The first of these is of use in cases in which little is known of the value of the optima of $F()$ (or $\boldsymbol{f()}$), and the goal of the study is to obtain sets of $\boldsymbol{G}$ with favourable characteristics, or to investigate the behaviour of ES algorithms. The second is of use when when one or more optima are known (or assumed) to exist, and the goal of the user is to (attempt to) obtain exactly one $\boldsymbol{G}$ such that $F()$ is equal to or better than a user defined value.

A typical exploration function is `Evolve3Static()`. "Static" in the function name refers to the fact that the user must supply an initialised population (because it does not dynamically allocate one).

```
void Evolve3Static( Organism_t *Population, int PopSize, int n_Survivors,
                    int /* Bool*/ Initialise, int n_gens,
                    FitnessEval_ft FitEval, double DeltaSigma,
                    double MuteRate) ;
 /* pre : Population[0..PopSize-1] valid,
          (PopSize, n_Survivors, n_gens, DeltaSigma, MuteRate) > 0
          n_Survivors < PopSize, MuteRate <= 1.0,
          (Initialise == TRUE) => Population not previously subject to FitEval()
   post : performs n_gens of mu+lambda ES to Population,
          leaves Population sorted in order of fitness according FitHiOrLo */
```

With the exception of the parameters `Initialise` and `MuteRate` the parameters to the function are as previously described. `Initialise` is needed because if the `Population` was generated in the manner of C code fragment 3.5 the `FitnessVal` fields will be undefined. The parameter `MuteRate` is used to specify the probability that a given gene will mutate, the intent in providing it was to restrict the rate at which populations expand in their genotype space. In practice it has been found to be more convenient to fix `MuteRate` as 1.0, and use lower values of `DeltaSigma` as necessary.

The interface to the function is designed to give the user maximum flexibility. Because the population is external to the function, the user may readily examine, report, or even modify, the population or other parameters, between successive invocations.

If the user's intent is solution rather than exploration then `EvolveToTarg()` is more appropriate:

```
void EvolveToTarg(Organism_t *Population, int PopSize, int n_Survivors,
                  int n_gens, FitnessEval_ft FitEval, double DeltaSigma,
                  double MuteRate, double F_Targ) ;
    /* pre : Population[0..PopSize-1] valid
             (PopSize, n_Survivors, n_gens, DeltaSigma, MuteRate) > 0
             n_Survivors < PopSize, MuteRate <= 1.0
```

```
post : performs <= n_gens of mu+lambda ES to Population,
       stops as soon as organism as fit or fitter than F_Targ is found,
       Population[0] is the fittest found */
```

With the exception of `F_Targ`, the fitness target, parameters for this function are the same as for `Evolve3Static()`. The design goal of the function is to minimise the number of fitness evaluations performed. There is no guarantee as to the state of `Population` other than the fact that `Population[0]` is the fittest organism found. In contrast to `Evolve3Static()` the purpose of the `n_gens` parameter is to ensure that the function will (eventually) terminate, if `F_Targ` is not reached.

3.3.3 Example

Specification

As an example of a simple application whose purpose is to illustrate the use of the ES functions described, consider fitting a minimal data set to a rectangular hyperbola:

$$y = \frac{m.x}{k+x} = \text{hyp}(x, m, k) \tag{3.4}$$

where m and k are the parameters whose values we wish to determine.

Design and implementation

This can be achieved with a minimum of two measurements of y for given values of x, hence organisms representing parameter estimates require 2 genes. Assuming two known x, y pairs, (x_1, y_1) and (x_2, y_2) a suitable fitness evaluation function is

$$F(m', k') = |\text{hyp}(x_1, m', k') - y_1| + |\text{hyp}(x_1, m', k') - y_2|$$

where m' and k' represent the current estimates of m and k. Given a function to calculate a rectangular hyperbola, `RecHyp(double x, double m, double k)`, and global constants `X1,Y1, X2,Y2`, the fitness evaluation function can be defined:

```
void F_eval(Organism_t org) {

  double m,k,e1,e2 ;

  m = org->Genome[0] ;
  k = org->Genome[1] ;
```

```
    e1 = RecHyp(X1, m,k) - Y1 ;
    e2 = RecHyp(X2, m,k) - Y2 ;
    org->FitnessVal = fabs(e1) + fabs(e2) ;
}
```

The main statement sequence falls into two logically separate phases: initialising a population, as described above, and subjecting it to the ES function. In order to report the progress of the population the ES function will be called repeatedly over 10 generation intervals, until a solution has been achieved. It is frequently difficult to detect convergence, but in this case, as a solution is known to exist, the program will simply terminate when some low value of $F()$ is attained. Assuming the existence of a trivial function, `PrintIt()` to generate output the main statement sequence is written:

Code Fragment 3.6 Main function to use ES to fit rectangular hyperbola data (C language)

```
int main(){

#define n_Genes 2
/* number of genes */
#define PopSize 60
/* population size */

  const int n_Survive = 10,                    /* number of survivors */
            n_Gens = 10;              /* number of generations for ES */

  const double  MuteProb = 1.0,            /* probability of mutation */
                DeltaSigma = 0.5 ;                      /* mutability */

  int  n, tot_gens = 1 ;                /* counter, total generations */
  double InitGenome[ n_Genes ] = {25, 25} ;         /* initial geneome */
  Organism_t MyPop[ PopSize ] ;                      /* the population */

    MyPop[0] =  NewOrg3(n_Genes, &InitGenome[0], DeltaSigma, FitnessLow,
                          VarMute) ;       /* generate  first organism */
    for(n_org = 1 ; n_org < PopSize ; n_org++)
                                         /* clone rest of population */
        MyPop[n_org] = CloneOrg(MyPop[0]) ;/* from the first organism  */

    Evolve3Static(MyPop, PopSize, n_Survive, TRUE, 1, F_eval, DeltaSigma,
                 MuteProb) ; /* initial first generation (init = TRUE) */

        /***** End of initialisation, start of ES proper *****/

    do{
        PrintIt(tot_gens, MyPop[0]) ;        /* tell the outside world */
        Evolve3Static(MyPop, PopSize,  /* do n_gens worth of evolution */
          n_Survive,  FALSE, n_Gens, F_eval, DeltaSigma, MuteProb) ;
        tot_gens += n_Gens ;                /* count total generations */
    } while (MyPop[0]->FitnessVal > 1e-3) ;
                                 /* until total absolute error < 0.001 */
    PrintIt(tot_gens, MyPop[0]) ;
}
```

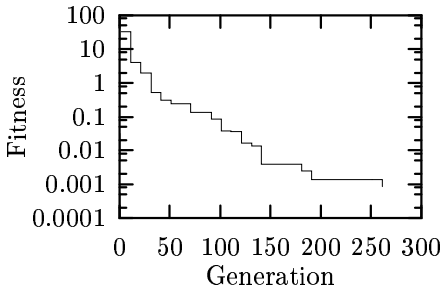

Figure 3.1: Progress curve of program shown in code fragment 3.6; note logarithmic scale of y axis.

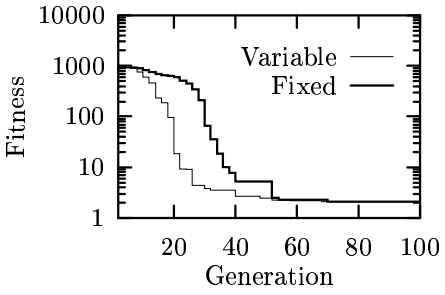

Figure 3.2: Relative rates of convergence for fixed and variable mutation from the progress curve fitting investigation, described in section 3.4.3

Assuming $m = 4$ and $k = 5$ from equation 3.4, the "observed data" `X1,Y1` and `X2,Y2` is set to 1,2/3 and -6,24 respectively, representing one point from each side of the discontinuity.

Results

The program terminated after 260 generations of ES. The fitness progress curve is shown in Figure 3.1. The logarithmic improvement in fitness appears characteristic of ES algorithms, having been observed many times, and reported by other authors [62, 10]. The estimates of m and k in the final generation were 3.998148 and 5.000443 respectively, indicating that a satisfactory convergence had been achieved.

Repeating the investigation with the population initialised as fixed mutation organisms resulted in a failure to converge usefully, after 500 generations. Curve fitting programs based on more conventional algorithms (gnufit, Macfit [65, 105]) failed to fit this set unless the initial values of m and k were brought to within about 20 % of their actual value.

3.4 Application to metabolic modelling

During the course of this project ES functions have been used for both exploration and solution of problems. The `Scampi` module function, `Ev_to_SS()`, introduced in chapter 2 uses `EvolveToTarg()` to seek steady-state solutions of metabolic models. The construction of this is described below.

The exploration approach to ES was used to identify and describe populations of

models of the Calvin cycle optimised in the face of realistic biological constraints, as described in chapter 5. A simpler use of the exploratory approach is described later in this chapter, with the object of demonstrating the efficacy of ES in solving an otherwise difficult problem, and the ease with which a user can integrate the `Evolve` module with the `Scampi` module.

3.4.1 Use of ES to determine steady states

The determination of the steady state of a metabolic model is a search for a set of internal concentration values such that the rates of change of all concentrations is zero. The problem is of the form of equation 3.1 and thus would appear to be suitable for solution by ES, with the genotype representing a concentration vector, $\boldsymbol{C}$, and a fitness evaluation function calculating $|\partial \boldsymbol{C}/\partial t|$. The approach is attractive because the computational demand is $O(N)$ in contrast to $O(N^2)$ for Newton's method.

Unfortunately this simple algorithm fails for systems with conserved moieties: the algorithm converges on $\boldsymbol{C} = \boldsymbol{0}$, where $\boldsymbol{C}$ is the vector of non-conserved metabolites. This appears to be quite a difficult problem, as this is a genuine solution. Attempts to impose lower limits on reaction velocities resulted in the algorithm simply converging to this lowest limit. At the time of writing, no method has been found that enables the potential $O(N)$ behaviour to be realised.

However, a useful algorithm can be developed if the goal is restated as locating suitable initial points from which Newton's method can proceed. In this case the genome is still $\boldsymbol{C}$, but the fitness of the organism is the tolerance achieved by Newton's method in some fixed number of iterations (it will be recalled from the previous chapter, section 2.5.4, that `Find_SS()` returns the tolerance achieved).

This approach was successful, but a significant improvement was gained by copying the the final concentration vector determined by `Find_SS()` back into the genome, an example of Lamarckian inheritance. The fitness evaluation function is written:

Code Fragment 3.7 Fitness evaluation function for `Ev_to_SS()` (C language)

```
void ModelSSFitEval(Organism_t org) {

  enum SPI_err err ;                          /* error from Find_SS() */
  ScampiModel_t mod ;                      /* local copy of the model */
  int n ;                                                /* counter */

    mod = (void *) org->user ;
```

```
                        /* user field points model, convenient copy */
    PutMDvec(mod, Conc, org->Genome) ;    /* load genome to concn vec */

    org->FitnessVal = Find_ss(mod, 20, LowestEvTol, &err) ;
                                                /* fitness = Newts tol */

    if((err != OK) && (err != OutOfTol))
                                    /* if neg conc or other problem */
        KillOrg(org) ;                        /* kill the undesirables */
    else{                           /* else copy final concentrations */
        for(n = 0 ; n < mod->n_Mets ; n++)
            org->Genome[n] = mod->Met_Vals[n] ;  /* back into  genome */
    }
}
```

The implementation of this function is within the Scampi module, and is thus able to dereference fields in the private structure referenced by `ScampiModel_t`. In addition to the field `n_mets` (number of metabolites), and `Met_Vals[]` (the metabolite value vector) the structure alao contains (amongst other things) a field `EvolScrat`. This is (a reference to) a structure maintaining a population and related information for ES algorithms as described in this chapter. Using these fields makes the implementation of `Ev_to_SS()` quite straightforward:

Code Fragment 3.8 Implementation of `Ev_to_SS()` (C language)

```
void Ev_to_SS(ScampiModel_t model, double tol, int MaxGens,
                double MuteSize,  double MuteRate, enum SPI_err *err) {

  int c ;                                                 /* g.p. counter */
  double in_fit ;                                  /* initial fitness val */

    in_fit = Find_ss(model, 20, tol, err) ; /* try for SS w/o evolving */
    if ((*err != OK) && (*err != NoMem))  {                /* got SS ? */
                                                    /* no, but mem OK */
        for( c= 0 ; c < model->n_Mets ; c++)
                                        /* copy concs into first genome */
            (model->EvolScrat->Population[0])->Genome[c] =
                model->Met_Vals[c] ;

        (model->EvolScrat->Population[0])->FitnessVal = in_fit ;
                                          /* we know first fitness val */

        EvolveToTarg2(model->EvolScrat->Population,  /* try to evol to */
                      model->EvolScrat->PopSize,  /* target fitness of */
                      model->EvolScrat->n_Survive,/* user supplied tol */
                      MaxGens, ModelSSFitEval, MuteSize, MuteRate, tol) ;

        if(model->EvolScrat->Population[0]->FitnessVal <= tol)
            *err = OK ;                 /* if successful tell the user */
    }
}
```

3.4.2 Determination of enzyme kinetic parameters from progress curves

A common problem facing the experimental biochemist is the determination of the kinetic parameters of enzymes. The traditional approach to this task has been the use of initial rate methods: for a number of different substrate concentrations the initial rate of conversion of substrate to product is recorded, and an appropriate rate equation is fitted to the curve thus constructed.

Although well established, there are two major inconveniences associated with the approach, if used for any but the simplest enzymes. Firstly the number of observations needed increases exponentially with the number of participating reactants; secondly when more than one reactant is involved, the problem of fitting a curve becomes a problem of fitting a (hyper)surface.

An alternative to initial rate methods is the analysis of reaction progress curves. The enzyme rate equation is a differential equation in terms of metabolite concentration(s) and time, which, if integrated and rearranged yields a function for reactant concentration in terms of time and the enzyme kinetic parameters. Thus, if it is possible to repeatedly measure metabolite concentrations over a period of time, then the problem of determining kinetic parameters becomes one of fitting the integrated rate equation to the observed curve of concentration versus time.

The standard [99] non-linear curve fitting algorithm is the Levenburg-Marquadt algorithm, and this has been incorporated into the "Dynafit" [74] program with the aim of fitting the integrated form of kinetic equations to progress data. However, success with Levenburg-Marquadt is sensitive to initial conditions, and is liable to fail if not started from a 'reasonable' set of estimated parameters.

3.4.3 Investigation of lactate dehydrogenase

This problem was repeatedly encountered by a colleague, Dr. Simon Thomas, who, during an investigation of the control of the glycolytic pathway in cultured fibroblasts, wished to determine the kinetic parameters of the enzyme lactate dehydrogenase (LDH), catalysing the reaction[7]:

$$\mathrm{Pyr} + \mathrm{NADH} \rightleftharpoons \mathrm{Lac} + \mathrm{NAD}^{+} \tag{3.5}$$

[7]Abbreviations: Pyr - Pyruvate, Lac - Lactate

This is a particularly attractive reaction to investigate by analysis of the progress curve, because the concentration of NADH can be determined directly by spectrophotometric means. Furthermore the stoichiometry of reaction 3.5 is such that as long as initial concentrations are known, subsequent values can be determined in terms of NADH concentration.

The experimental method is also relatively simple. A sample of fibroblast culture is homogenised and placed in a cuvette. An aliquot of NADH and pyruvate is added to the cuvette which is then placed in a spectrophotometer. This records the absorbance at 340 nm (the wavelength of the NADH absorbance maximum), at 0.5 s intervals, recording the data as plain ASCII file.

The enzyme has reversible two substrate kinetics [142], and this fact has been used[8] to derive the rate equation:

$$V = \frac{\frac{V_{max}\text{NADH Pyr}}{K_{Pyr}K_{NADH}} - \frac{\text{NAD Lac}}{K_{eq}}}{1 + \frac{\text{NADH}}{K_{NADH}} + \frac{\text{NADH Pyr}}{K_{NADH}K_{Pyr}} + \frac{\text{NAD Lac}}{K_{NAD}K_{Lac}} + \frac{\text{NAD}}{K_{NAD}} + \frac{\text{NAD Pyr}}{K_{NAD}K_{Pyr}}} \tag{3.6}$$

where K_{eq} is the equilibrium constant, and $K_{\text{Metabolite}}$ is the Michaelis constant associated with that metabolite.

Program design

The rate equation and initial concentration values are all that is required to construct a SCAMP/Scampi model of the reaction system. It is then straightforward matter to construct a program that reads the file of experimentally determined concentrations, and invokes an ES function whose genome is the set of kinetic parameters, and whose fitness is determined by the closeness of a simulated progress curve to the observed.

Once the housekeeping needed to read the input file and initialise associated constant global data structures has been put in place, the heart of the program is the fitness evaluation function to be passed to the ES function. Several variants were investigated, but none showed superior behaviour, either in terms of rate of convergence or final goodness of fit, than the simplest, presented in code fragment 3.9.

The function depends upon the existence of global constants, holding the values of the number of data points and time intervals. In addition to these constants, three global data structures are made available:

[8]Simon Thomas - unpublished work

1. The model itself, `ldh`.

2. An `OutputDesc_t`, `glob_op`, connected to `ldh`, and initialised to store TIME and ldh concentration in memory.

3. A list ADT, `glob_Obs_dal`, holding the observed progress curve.

With the exception of the model itself, all global identifiers are prefixed `glob_`.

Code Fragment 3.9 Fitness evaluation for progress curve fitting (C language)

```
void FitEval(Organism_t org){

  double known[2],         /* holds individual observed time/NADH pairs */
         NADH_est ;                        /* estimated NADH from model */
  enum SPI_err err ;                /* errors from the Scampi functions */
  enum dal_err derr ;                   /* errors from the list handler */

    org->FitnessVal = 0.0 ;                          /* we start perfect */

    if(HasNeg(org->Genome, org->LenGenome))       /* discard -ve values */
        KillOrg(org)                                  /* w/o further ado */
    else{
        PutMDvec(ldh, Conc, glob_OrigConcs) ; /* load Initial conc vals */
        PutSubset_md(ldh, Param, GeneNames, org->Genome) ;
                                        /* load a new set of parameters */
        Simulate(ldh, 0.0, glob_Time_End, glob_n_points, 1e-6,
                    &glob_op, 1, &err) ;
      /* simulate a progress curve, for between t = 0 and global time end,
               storing results in the global output descriptor glob_op */

        if(err != OK)      /* if something went wrong kill the organism */
            KillOrg(org)  ;
        else{
            Read1stValOP(glob_op, &NADH_est, &err) ;/* read first point */
                                          /* of simulated progress curve */
            derr = GetTop_dal(glob_Obs_dal, known) ;/*and observed vals */

            while(derr == OK_dal){            /* while we can read data */
                org->FitnessVal += fabs(NADH_est - known[0]);/* add the */
                            /* absolute difference between observed and */
                                       /* calculated to the fitness value */
                ReadNextValOP(glob_op, &NADH_est, &err) ;/*and read more*/
                derr = GetNxt_dal(glob_Obs_dal, known) ; /* data points */
            }
        }
        FreeMemOP(glob_op) ;               /* clear out our sim results */
    }
}
```

Results

Using the parameter set of equation 3.6 as the genome, a population size of 36, with 6 survivors, and code fragment 3.9 for fitness evaluation, the ES algorithm converged as

shown in Figure 3.2. Using the best parameter set found by the algorithm a simulated progress curve may be generated that overlays the experimental observations as shown in Figure 3.3. Although Figure 3.3 shows a good fit, examination of the residuals reveals that there is some systemic error as the residuals are clearly not evenly distributed about zero, with a "saw-tooth" time dependency (Figure 3.4).

Encouragingly, it has been suggested [18, 19] that this pattern is characteristic of a fitting algorithm having converged to within the limit of precision imposed by the input data.

Conclusion

Figure 3.3 clearly demonstrates the ability of ES algorithms to fit simulations to real-world data sets, and thereby determine real-world parameter values. The time taken to achieve convergence in the results shown was of the order of 10 minutes on a slow (20 MHz 68020/68881) computer, and of the order of seconds on a more modern machine (Sun SPARC-Ultra), and so use of the technique is a practical proposition.

The motivation for developing this example was to test the behaviour of ES algorithms in the face of real-world experimental data. A better goodness of fit than might be reasonable to hope for has been shown in Figures 3.3 and 3.4. This is all the more satisfactory when it is considered that software based on the more conventional Levenburg-Marquadt algorithm failed entirely to fit the data set shown.

However a particular problem remains to addressed if the program is to be extended to become a practical tool for routine fitting: the determination of confidence limits. Although the idea of determining these on the basis of an analysis of the whole final population has obvious attractions, this is made (probably fatally) complicated by the fact that each individual in the population does not represent an independent estimate of the parameter set, and populations tend not to be normally distributed.

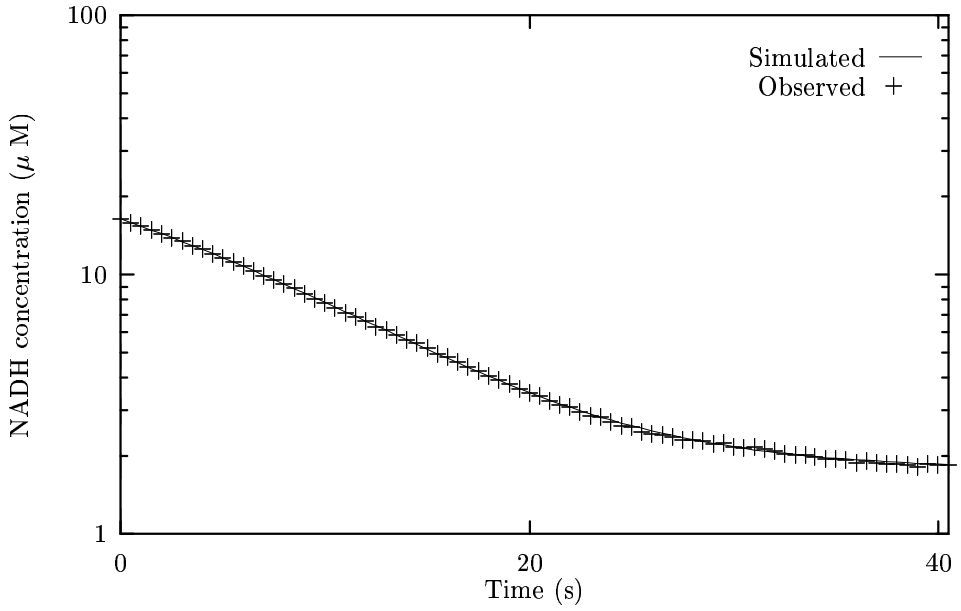

Figure 3.3: Observed and simulated data sets after fitting equation 3.6 using ES algorithm as described in section 3.4.3

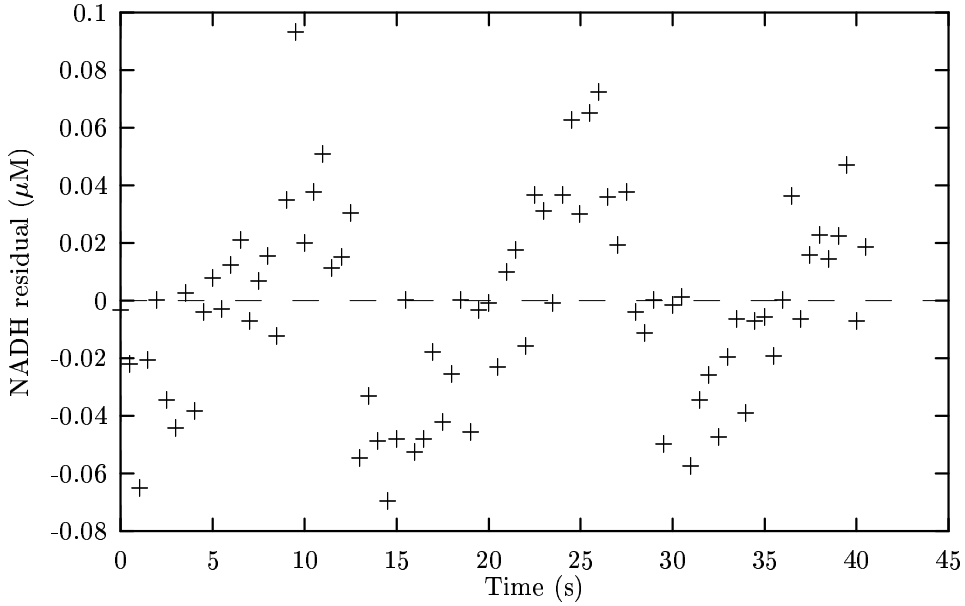

Figure 3.4: Differences between fitted and observed data sets in Figure 3.3

Chapter 4

Investigation of the Behaviour of Computer Models of the Calvin Cycle

4.1 Introduction

As described in Chapter 1 several groups of workers [51–53, 77, 75, 78, 91, 138] have developed and analysed detailed kinetic models of the Calvin cycle and related reactions. All have simplified the system either by using simplified kinetic equations, or by assuming that steady-state concentrations of groups of metabolites are at equilibrium with one another, thus reducing the number of unknown concentrations and simplifying the topology.

The approach used here is to make no attempt to simplify the topology of the system, minimise the kinetic simplifications, and to use the software described in Chapter 2 to define a model with a structure as complete as knowledge of the system allows. Once defined, the behaviour of the model was investigated empirically, the results from one investigation being used to form hypotheses tested by subsequent investigation. Such investigations included alterations of the structure, as well as the parameters, of the model. In this chapter investigations of three refinements of the model of the Calvin cycle , described in Section 4.2, are presented. Discussion of the physiological and theoretical significance of these is deferred until Chapter 6.

The results in this chapter were generated using a conventional strategy for modelling: parameters thought likely to be important were varied over an appropriate range, and the effects of these variations on the behaviour of the model recorded. A major problem with this is the fact that this model (in common with any other of realistic complexity) contains a large number of parameters and the number of potential investigations increases combinatorially with the number of parameters to be investigated. This problem is compounded by the fact that interaction between parameters cannot be readily predicted. Furthermore it is not possible to dismiss individual parameters as being of no importance to the behaviour of the model without investigation. The results in Chapter 5 use the Evolution Strategy algorithm (Chapter 3) to attempt to ameliorate such problems.

4.2 Model Definition

4.2.1 The Petterssons' model

Of the models cited, the most complete description of the Calvin cycle was that of Pettersson and Ryde-Pettersson [91], and this was used as a starting point for development of the model described in this chapter. In common with most others, development of the model in [91] was based on the assumption that substrates and products of fast, reversible, reactions are maintained at equilibrium, and that this assumption may be extended to cover arbitrarily large groups of metabolites interconverted by such reactions. The assumption was used to simplify the system to a set of six reactions.

4.2.2 Removal of equilibrium assumptions

In this chapter the topological simplification is removed by assigning rapid reversible reactions, described as "Fast" in Table 4.1, linear kinetic equations of the form:

$$V = K\left(\prod_{i=1}^{n_S} S_i - \frac{\prod_{j=1}^{n_P} P_j}{k_{eq}}\right) \tag{4.1}$$

where V is the reaction velocity, k_{eq} the equilibrium constant, S and P substrate and product concentration(s) respectively, and K is a rate constant whose dimensions are determined by the molecularity of the reaction. Unless stated otherwise K was set to 5×10^8 for all fast reactions, and values for k_{eq} as described in [91]. Non-equilibrium

reactions, described as "Slow" in Table 4.1, were assigned the same reaction kinetics as in [91].

In common with many other researchers in the field of plant physiology, Pettersson expressed reaction rates in dimensions of μmol.h^{-1}.(mg Chl)$^{-1}$, and these are the units used here. When equation 4.1 is scaled[1] to yield V in units of $M^{-1}S^{-1}$, K (for a bi-molecular reaction) becomes equal to 5.4×10^{7} $M^{-1}S^{-1}$. This compares with substrate-enzyme binding (forward) rate constants, reported by Fersht [34], falling within the range 10^{6} – 10^{8} $M^{-1}S^{-1}$, and well below the diffusion controlled rate limit of 10^{9} $M^{-1}S^{-1}$.

Although equation 4.1 still represents a significant simplifying assumption, the explicit inclusion of these reaction, coupled with the very detailed kinetic equations described in [91], lead to a model of the Calvin cycle thought to be more complete than any previously published.

A second assumption is that the light reactions only regenerate ATP from ADP and P_i, with reaction kinetics:

$$V = \frac{V_{Light}.\mathrm{ADP}.\mathrm{P_i}}{(\mathrm{ADP} + K_{m\mathrm{ADP}}).(\mathrm{P_i} + K_{m\mathrm{P_i}})} \tag{4.2}$$

(Where V_{Light} is taken to be the light reaction activity for the purposes of this model) and that NADPH, $NADP^+$, and H^+ are maintained at constant concentration. Although at first consideration this appears to be a rather drastic simplification, examination of Figure 4.1 suggests that the effect of this, at least in terms of response of the model to light reaction activity, is not likely to be great. The only reaction involving these constant metabolites is v_3 (G3Pdh) and the topology of the Calvin cycle is such that at steady state the flux through this reaction is equal to that through v_2 (PGK). Both of these reaction utilise products of the light reactions as substrates (NADPH and ATP respectively) and both have products ($NADP^+$ and ADP) that are substrates of light reactions. Thus the (local) effect of a reduction in NADPH/$NADP^+$ ratio resulting from decreased light reaction activity would be decreased flux in G3Pdh/PGK. This effect is in any case achieved by the reduction in the ATP/ADP ratio resulting from reduced light reaction activity.

[1] Using Pettersson's estimate of stromal volume as $30\mu L.(mg.Chl)^{-1}$

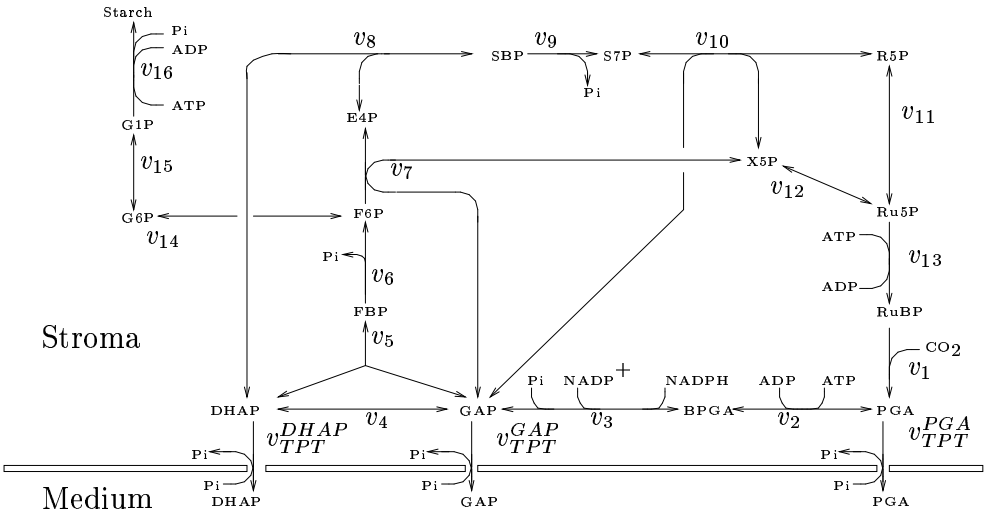

Figure 4.1: Reactions of the Calvin Cycle used in the initial model. Abbreviations are described in Tables 4.1 and 4.2.

4.2.3 Other changes to the model

In the first versions of this model a typing error resulted in V_{Light} being entered as 3500 instead of 350 μmol.h^{-1}.(mg Chl)$^{-1}$. However, as shown in Section 4.4, the model presented here is insensitive to changes in this parameter over most of this range. Furthermore Pettersson [91] points out that experimental evidence exists [13] suggesting that the value of 350 μmol.h^{-1}.(mg Chl)$^{-1}$may be a considerable underestimate of V_{Light}, and hence the results here remain consistent with the assumption in [91] that the model is saturated with respect to light.

The structure of the initial version of the model is shown in Figure 4.1 with additional information in Tables 4.1 and 4.2; the complete SCAMP `.cmd` file for the model is presented in Appendix A.

Various problems were encountered with the initial SCAMP implementation of the model, and eventually it was found that they could be attributed to two ultimate causes. One was the absence of an error handling strategy in SCAMP; this was one of the motivating factors in the development of the Scampi tool-kit described in Chapter 2. The other was the loss of carbon in the form of PGA resulting in an insufficient flux in the regenerative limb to keep the cycle sustainable. This was overcome by inclusion of a factor in the TPT rate equation modulating its rate toward PGA, denoted θ_{PGA}here.

Table 4.1: Reaction subscripts, abreveations, names, and kinetics, used in the Calvin cycle model.

Subscript	Abbreviation	Name	Kinetics
1	Rubisco	Ribulose bisphosphate carboxylase-oxidase	Slow
2	PGK	Phosphoglycerate kinase	Fast
3	G3Pdh	Glyceraldehyde-3-phosphate dehydrogenase	Fast
4	TPI	Triose phosphate isomerase	Fast
5	F.Aldo	Aldolase (FBP reaction)	Fast
6	FBPase	Fructose-1,6-bisphosphatase	Slow
7	F.TKL	Transketolase (F6P reaction)	Fast
8	S.Aldo	Aldolase (SBP reaction)	Fast
9	SBPase	Sedoheptulose bisphosphatase	Slow
10	S.TKL	Transketolase (S7P reaction)	Fast
11	R5Piso	Ribose-5-phosphate isomerase	Fast
12	X5Pepi	Xylose-5-phosphate epimerase	Fast
13	Ru5Pk	Ribulose-5-phosphate kinase	Slow
14	PGI	Phosphoglucose isomerase	Fast
15	PGM	Phosphoglucose mutase	Fast
16	StSyn	Starch Synthase	Slow
TPT	TPT	Triose phosphate translocator (Subscript indicates metabolite)	Slow
-	StPase	Starch Phosphorylase	Slow

Table 4.2: Metabolite abreviations and names used in Fig. 4.1 and elsewhere

Abreviation	Name
PGA	Phosphoglycerate
BPGA	Bisphosphoglycerate
GAP	Glyceraldehyde phosphate
DHAP	Dihydroxy acetonephosphate
FBP	Fructose 1,6-bisphosphate
F6P	Fructose 6-bisphosphate
E4P	Erythrose 4-phosphate
SBP	Sedoheptulose 1,7-bisphosphate
S7P	Sedoheptulose 7-phosphate
X5P	Xylose 5-phosphate
R5P	Ribose 5-phosphate
Ru5P	Ribulose 5-phosphate
G6P	Glucose 6-phosphate
G1P	Glucose 1-phosphate
P_i	Inorganic phosphate
$P_{i_{ext}}$	External P_i

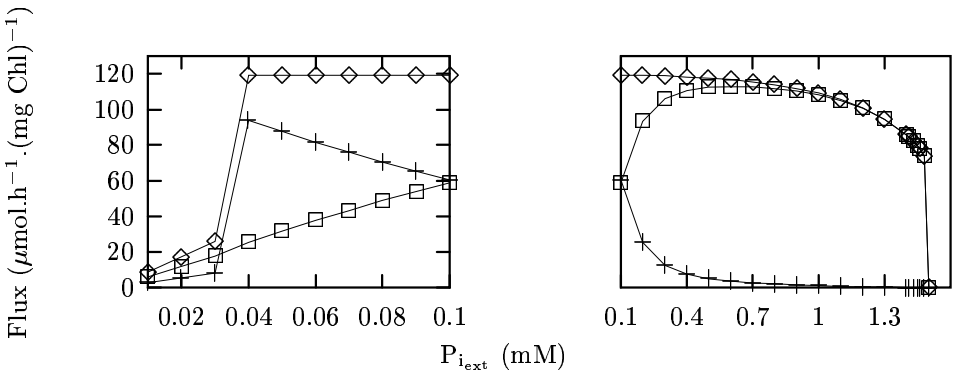

Figure 4.2: Response of input and output fluxes in the Calvin cycle model (without starch phosphorylase) to $P_{i_{ext}}$: ($\diamond$) - rubisco, ($\square$) - TP export, (+) - Starch synthase.

Unless stated otherwise θ_{PGA} was set to 0.1 throughout this thesis.

4.3 Model response to external P_i

4.3.1 Flux response

Carbon flux into and out of the Calvin cycle model as a function of $P_{i_{ext}}$ is shown in Figure 4.2. The response is clearly bi-phasic: at low $P_{i_{ext}}$ ($\leq\sim$ 0.03 mM) assimilation, export, and storage fluxes all increase monotonically with $P_{i_{ext}}$. At some point between 0.03 and 0.04 mM, an abrupt transition occurs, and a region is entered in which assimilation is relatively insensitive, export responds positively, and storage flux negatively to $P_{i_{ext}}$. This behaviour has not been previously described in a Calvin cycle model, however, results presented in sections 4.5, and discussed in chapter 6, demonstrate that such transitions can be induced in the model under a variety of circumstances.

As $P_{i_{ext}}$ is further increased, TP export reaches a maximum at $P_{i_{ext}} \approx 0.6$ mM. After this point all fluxes then fall with increasing rapidity until a catastrophic "overload breakdown" occurs at $P_{i_{ext}} = 1.5$ mM. This breakdown is associated with (and presumably caused by), loss of sugar phosphate intermediates while ATP concentrations remain unaffected (see Figure 4.3.

4.3.2 Metabolite concentrations

The bi-phasic nature of the response to $P_{i_{ext}}$ is also seen in metabolite concentrations in Figure 4.3. Below the transition point, almost all of the conserved P_i of the Calvin cycle

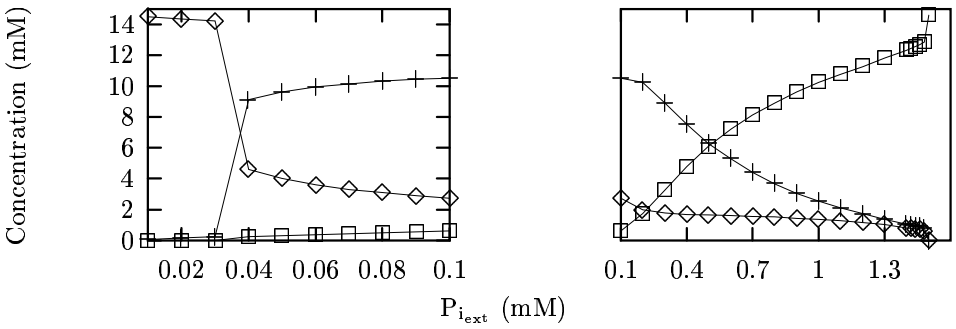

Figure 4.3: Response of metabolite concentrations in the Calvin cycle model (without starch phosphorylase) to $P_{i_{ext}}$: ($\diamond$) - PGA, (+) - total sugar phosphate, ($\square$) - free P_i .

is in the form of PGA, with most of the rest in distributed amongst other phosphorylated intermediates; only $\sim$ 0.02 % of the total is in the form of free P_i . Immediately above the transition point, the concentration of PGA drops and most of the P_i is redistributed amongst the other intermediates. There is also an increase in free P_i concentration, the relative size of which is comparable to that occurring in the non PGA intermediates.

As $P_{i_{ext}}$ is increased beyond the transition point, PGA and total sugar phosphates decrease, and free P_i increases, monotonically. It is interesting to note that none of these curves contain a maxima corresponding to the assimilation flux maxima seen in Figure 4.2. This observation remains true for response curves of individual metabolites (data not shown). The response stays constant until $P_{i_{ext}}$ reaches the breakdown point, when all intermediates, with the exception of P_i fall to zero.

Disequilibrium ratios

The mass action ratios of all the fast reactions with the exception of G3Pdh remained close to their respective K_{eq} ($0.99 < \rho \leq 1.0$) and showed some variation with $P_{i_{ext}}$. As shown in Figure 4.4, the disequilibrium ratio of G3Pdh remains far from 1.0 (the expected value if equilibrium assumptions hold true), and varies considerably as a function of this parameter.

4.3.3 Flux Control

Flux control coefficients over the three external fluxes, (assimilation, export, and starch synthesis) shown in Figure 4.2, over the same range of $P_{i_{ext}}$, are presented in Figure

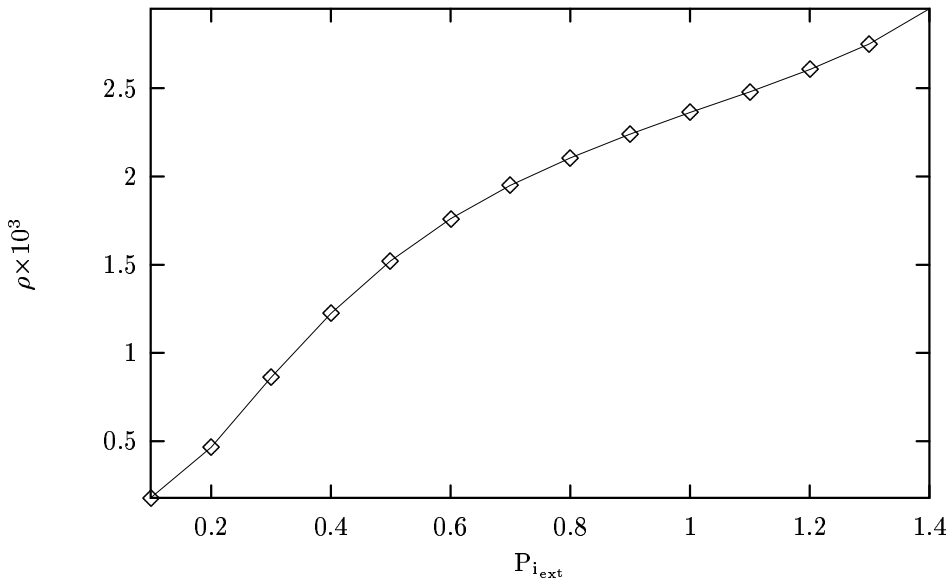

Figure 4.4: Response of ρ_{G3Pdh} to $P_{i_{ext}}$.

4.5. $C^{J_{TPT}}$ is calculated as the control coefficient over the sum of the three component fluxes oof this step.

There are a number of noteworthy features in Figure 4.5. Firstly, despite some similarities, there are major differences in control over the three fluxes. Secondly control changes quite dramatically as a function of $P_{i_{ext}}$.

Control of assimilation flux

The simplest pattern of control is that of assimilation. At low values ($<\sim$ 0.9 mM) of $P_{i_{ext}}$ the only enzyme to have any significant control is SBPase. As the value of $P_{i_{ext}}$ approaches the point at which the cycle breaks down the control of assimilation by the light reactions, fast reactions and FBPase increases dramatically, with little or no change in the control by SBPase. The control exhibited by rubisco, Ru5Pk and StSyn remains negligible, and that of TPT becomes rapidly more negative (i.e. increasing the activity of this step results in a decrease in the assimilation flux).

Control of export flux

At values of $P_{i_{ext}}$ below 3 mM, the TPT has the majority of control over its own flux, while SBPase maintains a negative control over export. The fast, and the light, reactions

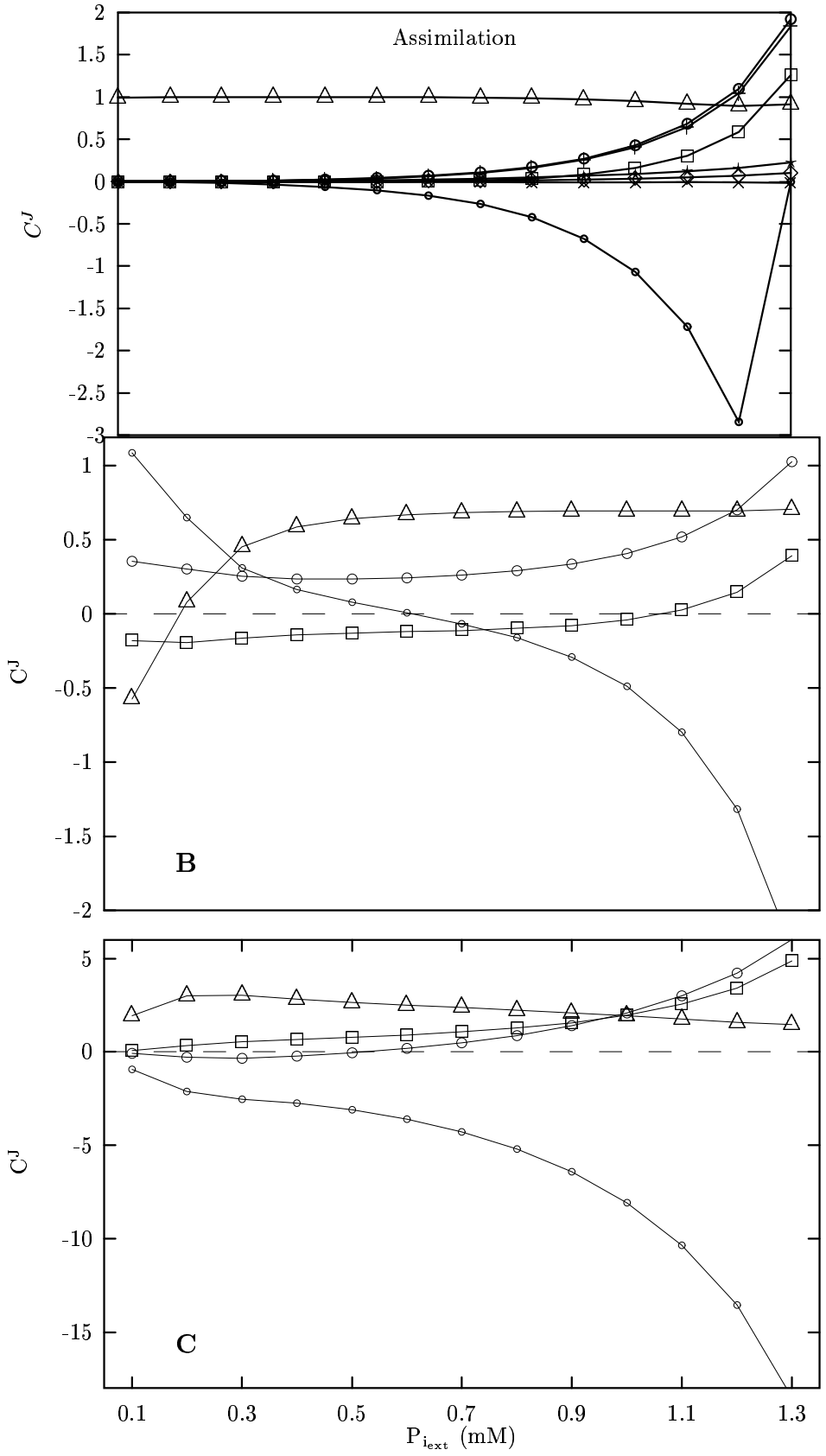

Figure 4.5: Effect of $P_{i_{ext}}$ on control of **A**: assimilation, **B**: TPT, **C**: starch fluxes, by: (□) - FBPase, (△) - SBPase, (∘) - TPT, (○) - light reactions. Other reactions omitted for clarity. rubisco and StSyn had negligible C^J over all fluxes, including their own.

have small positive control coefficients while that of FBPase is small and negative.

As $P_{i_{ext}}$ increases, the positive control exerted by the TPT diminishes rapidly and, at 0.6 mM $P_{i_{ext}}$, becomes negative, exhibiting the counter-intuitive behaviour that an increase in the activity of this step results in a decrease in the flux through it. As the control coefficient of the TPT decreases, that of SBPase increases reaching a plateau of about 0.7 at 0.5 mM $P_{i_{ext}}$. As the breakdown point approaches the control characteristics become similar to those of the assimilation flux, with control by light reactions, fast reactions, TPT, and FBPase increasing rapidly in magnitude, and with relatively little control shown by rubisco, Ru5Pk or starch synthase.

Control of storage flux

Control over the storage flux and export fluxes bear more resemblance to each other than either do to that of assimilation; nonetheless there are important differences between the two. The first point of note is that control coefficients over starch synthesis are much (5 to 10 times) greater than corresponding coefficients over TPT flux.

Control by SBPase and TPT remain positive and negative respectively and do not change signs with increasing $P_{i_{ext}}$@. Over most of the range of $P_{i_{ext}}$ ($>\sim$ 0.2 mM), StSyn maintains C^{J} over its own flux in the range ($0.7 < C^{J_{StSyn}}_{StSyn} < 1.0$). This is the only one of the three external fluxes over which the enzyme has any degree of control; however it should be noted that, over the entire range of $P_{i_{ext}}$, there are always other steps exerting much greater positive and negative control.

4.4 The inclusion of starch phosphorylase

Despite some important differences, the results presented thus far are generally comparable to those reported in other experimental and modelling work (see chapter 6). However, in terms of the interaction of the model with its external environment, the model is not complete as the possibility exists for starch to be a carbon source, due to the presence of starch phosphorylase (StPase) [12], as well as a sink. Furthermore, it would seem unlikely that the overload breakdown described above is a feature of chloroplasts in an intact organism (despite being observed in isolated chloroplasts, see Chapter 6). It was conjectured that the additional carbon source provided by StPase might overcome this problem. Therefore a step was added to the model described in

section 4.2 representing StPase, catalysing the reaction:

$$\text{Starch} + \text{P}_\text{i} \rightarrow \text{G1P} \tag{4.3}$$

At a rate determined by the equation:

$$v = \frac{V_{max}\text{P}_\text{i}}{\text{P}_\text{i} + K_m\left(1 + \frac{\text{G1P}}{k_i}\right)} \tag{4.4}$$

(where V_{max} denotes the maximum velocity, k_m the Michaelis constant, and k_i the inhibition constant due to G1P) based on the following considerations:

1. The reaction is irreversible.

2. StPase will have Michaelis-Menten kinetics with regard to P_i , but will be insensitive to changes in the starch substrate (to which a meaningful concentration cannot be assigned).

3. StPase will be (competitively) product inhibited by G1P.

These local considerations are also compatible with the physiological role that the reaction will play in the context of the Calvin cycle . Under conditions of carbon starvation, P_i concentration must necessarily increase (as a result of the conservation relationship), and that of G1P decrease; thus supply of carbon into the cycle via StPase will increase under the circumstances in which it is most needed. Furthermore the effects of G1P and P_i upon StPase are in the opposite sense to those upon StSyn. V_{max}, K_m, and k_i were assigned values of 40 μmol.h^{-1}.(mg Chl)$^{-1}$, 0.1 mM and 0.05 mM respectively, based on the assumption that the maximum rate of degradation will be comparable to the maximum rate of synthesis, and that, in common with the other irreversible reactions, the K_m values will be similar to the relevant metabolite concentrations.

A possible cause for concern about the addition of StPase to the model is the introduction of a futile cycle between starch synthesis and degradation. The degree of futile ATP cycling in the data sets presented here is generally less than 1% of the light reaction flux, and always less than 2%. Although in this model starch may be simultaneously synthesised and degraded it is the net flux to starch that is physiologically relevant here. Thus flux to starch reported here is calculated as the flux through PGI: positive values indicate net synthesis, negative net degradation.

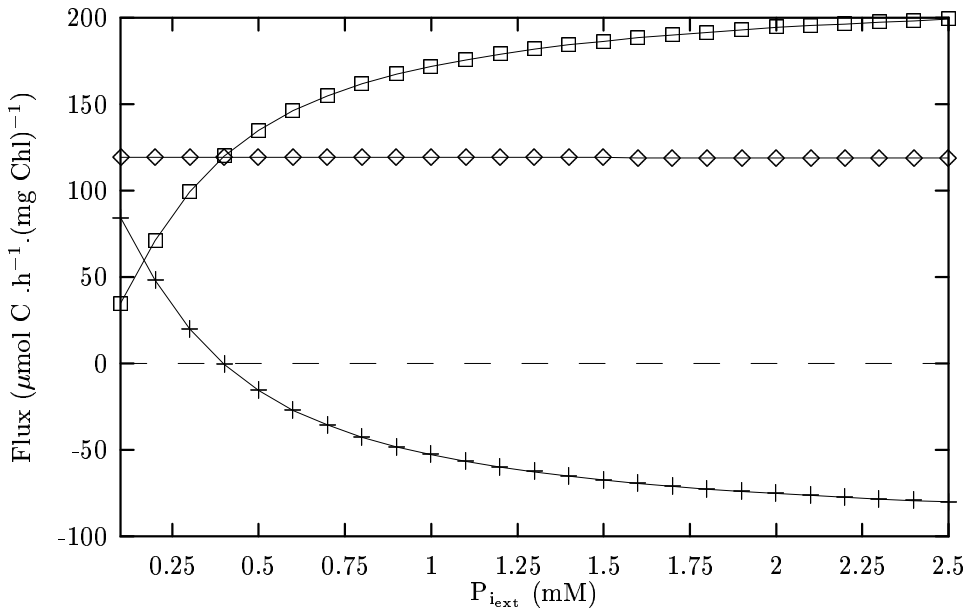

Figure 4.6: Response of rubisco ($\diamond$), TP export ($\square$) and StSyn (+) fluxes to $P_{i_{ext}}$, with StPase included in the model.

4.4.1 Response of external fluxes to $P_{i_{ext}}$

The inclusion of StPase has a marked effect on the response of the Calvin cycle model to $P_{i_{ext}}$. As can be seen in Figure 4.6, the overload breakdown at high $P_{i_{ext}}$ is abolished, and the carbon flux to the cytoplasm can exceed that of assimilation, with the difference being met by degradation of starch, as indicated by the negative flux to starch. However, the overall pattern of response of the two versions of the model is the same: the assimilation flux is insensitive to cytoplasmic demand; changes in demand are met by changes in the rates of starch synthesis. The point at which flux to starch becomes negative is inversely dependent upon the sensitivity of the TPT to PGA.

4.4.2 Response of metabolite concentration to $P_{i_{ext}}$

Metabolite concentrations are not greatly changed by the inclusion of StPase in the model. However, as shown in Figure 4.7, the overall variation in response to $P_{i_{ext}}$ is much more constrained. All metabolites are asymptotic to a fixed concentrations: free P_i increases with increasing $P_{i_{ext}}$, mainly at the expense of PGA. The total concentration

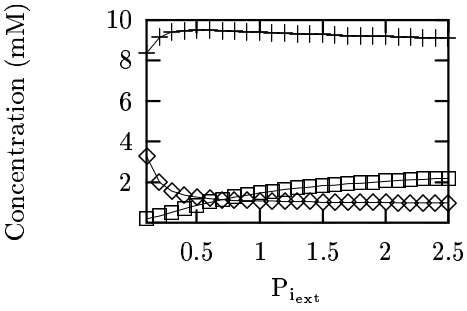

Figure 4.7: Response of metabolite concentrations in the Calvin cycle model (with starch phosphorylase) to $P_{i_{ext}}$: ($\diamond$) - PGA, (+) - total sugar phosphate, ($\Box$) - free P_i .

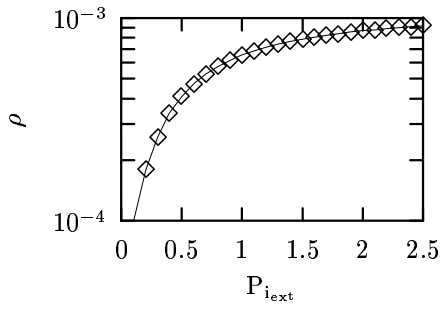

Figure 4.8: Response of ρ_{G3Pdh} to $P_{i_{ext}}$.

of other phosphorylated intermediates increases to a maximum at $P_{i_{ext}} \sim 0.5$, and thereafter decline slightly.

Disequilibrium ratios in the new model were also very similar to the old. All remained very close to (but less than) 1.0, with the exception once more of G3Pdh as shown in Figure 4.8.

4.4.3 C^J responses to $P_{i_{ext}}$

The inclusion of StPase in the model has a strong effect on the control of external fluxes, and their response to $P_{i_{ext}}$ (Figure 4.10). The greatest effect is on the control of the assimilation flux where the only reaction to have any significant control is SBPase, which thus behaves as a classic "rate limiting" step.

Control of flux through the TPT is less simple, with several enzymes exerting both positive and negative control. In contrast to the model with no StPase, the light reactions have no significant control. StSyn and FBPase both have small negative control coefficients that show little variation in response to $P_{i_{ext}}$. The majority of positive control is held between four steps, and this control appears to be exchanged between two pairs as $P_{i_{ext}}$ varies. The two steps having most control are those catalysed by SBPase and TPT: at low $P_{i_{ext}}$ most the control is by TPT itself, but this is lost to SBPase as $P_{i_{ext}}$ increases until high $P_{i_{ext}}$ concentrations are reached, when both of these steps have approximately equal control. In contrast to the version of the model without StPase,

TPT maintains positive control over its own flux as $P_{i_{ext}}$ varies. The group control coefficient of the equilibrium reactions, and that of StPase, is relatively small, not rising much above 0.2. The former decreases, and the latter increases, monotonically with increases in $P_{i_{ext}}$.

Both the size of individual values, and the response (to $P_{i_{ext}}$) of, control coefficients over the flux to starch are, to an extent, the result of a mathematical artifact caused by the original definition of a flux control coefficient (1.2). Because the denominator contains reaction flux as a term, as flux approaches zero, the control coefficient approaches $\pm\infty$. The sign change indicates the change of direction of flux in this pathway. When flux to starch is positive SBPase, FBPase, and StSyn, in decreasing order, have positive control, and StPase, the near equilibrium reactions, and TPT have increasingly negative control. Once again rubisco, Ru5Pk and the light reactions have negligible control.

Effect of altered SBPase activity

Given that SBPase appears to be acting as a true rate-limiting step for assimilation, it is of interest to examine the effect of increasing the activity of this step. The result is somewhat unexpected: as can be seen in Figure 4.9, an increase in the $V_{max}$of SBPase only brings about a propotrional increase in assimilation flux over a limited range, beyond which there is a an abrupt decrease in assimilation flux. The point at which this transition occurs can be increased by increasing $P_{i_{ext}}$ or θ_{PGA}, but assimilation flux remains considerably lower than the limit imposed by the V_{max}value of rubisco.

4.5 Response of the model to light

As noted in section 4.2 the light reactions are not modelled in detail, rather the maximum activity of ATP synthase is taken as an approximation of overall light reaction activity, and hence light intensity. In this section "light", and "light reaction activity" should be treated as being synonymous with maximum ATP synthase activity.

4.5.1 Flux and concentration responses

Figure 4.11 shows the response of the model to changes in light. The model was first evaluated at a light value of 1500 μmol.h^{-1}.(mg Chl)$^{-1}$, and this was decremented in steps of 25 to a minimum of 900 μmol.h^{-1}.(mg Chl)$^{-1}$, at which point the process

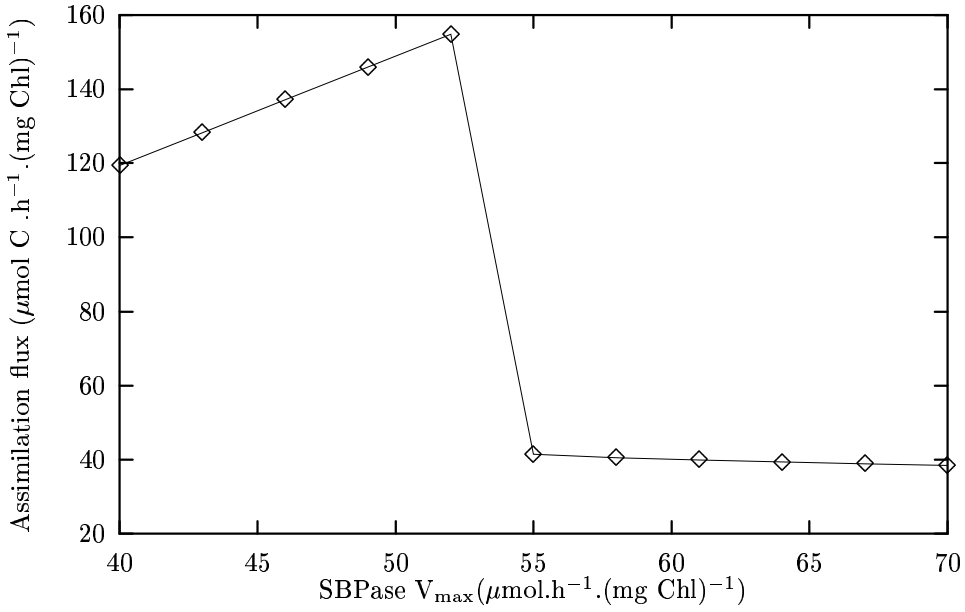

Figure 4.9: Dependence of Assimilation flux upon $V_{max}$of SBPase ($P_{i_{ext}}$ =0.5 mM, θ_{PGA}=0.1).

was reversed, and light incremented until the upper limit was once again reached, as indicated by the arrows on Figure 4.11(A). Figure 4.11(B) shows the response of the output fluxes, and Figure 4.11(C) metabolite concentrations, over the same range of light values. The discontinuities, which appear to represent true switching behaviour, at 1050 and 1175 μmol.h^{-1}.(mg Chl)$^{-1}$observed in all variables.

It can be seen in Figure 4.11(A) that the assimilation flux is insensitive to changes in light until a critical value at about 1050 μmol.h^{-1}.(mg Chl)$^{-1}$is reached, below which point assimilation flux decreases monotonically with light. Although neither the assimilation or light reaction (not shown) fluxes are sensitive to changes in light over the upper range, the model does respond by altering the partitioning between starch synthesis and TP export fluxes. As can be seen in Figure 4.11(B) the system (when in the faster steady state) responds to a decrease in light by increasing the flux to starch, with a concomitant decrease in the export flux. When the system is in the slow steady state, assimilation, TPT, and starch fluxes all respond monotonically, and in the same sense, to changes in light. The flux to starch becomes negative at a point close to, but not coincident with, the fast-slow transition.

The slow steady-state induced by low light levels appears to be very similar to that

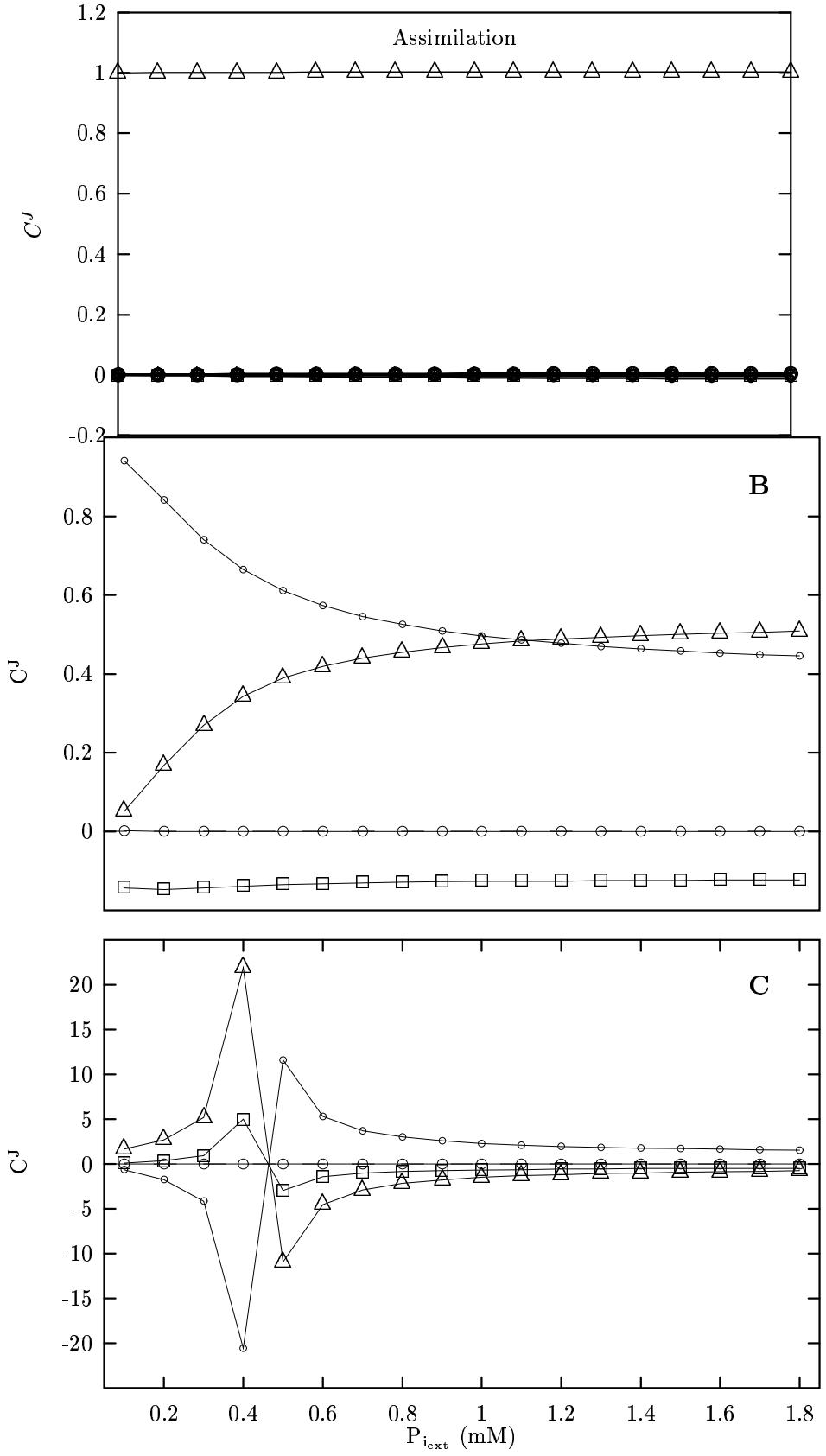

Figure 4.10: Effect of $P_{i_{ext}}$ on **A**: Assimilation, **B**: TPT, **C**: starch fluxes by (□) - FBPase, (△) - SBPase, (○) - TPT, (○) - light reactions. Other reactions omitted for clarity. Rubisco and StSyn had negligible C^J over all fluxes, including their own.

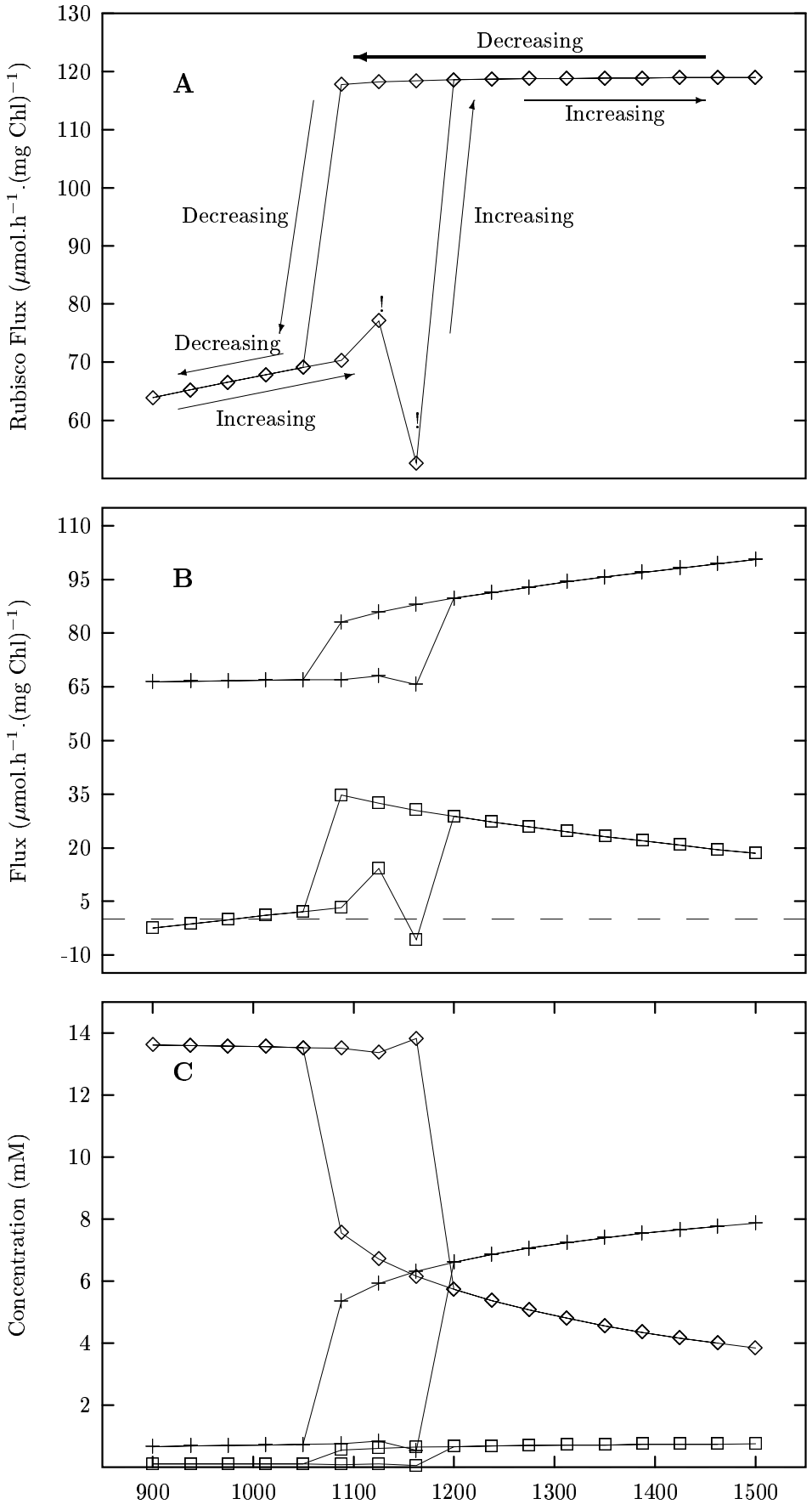

Figure 4.11: Response of the Calvin cycle model to light. Points marked '!' are not true steady state values (See 4.5.3 for further details). **A**: Assimilation Flux, **B** Output fluxes, (+) - TPT and (□) - Starch flux, **C**: Metabolite concentrations, (◇) - PGA, (+) - total sugar phosphate, (□) - P_i .

Table 4.3: Effect of θ_{PGA} on the model switching response to varying light levels.

θ_{PGA} (Dimensionless)	-ve transition (μmol.h^{-1}.(mg Chl)$^{-1}$)	+ve transition (μmol.h^{-1}.(mg Chl)$^{-1}$)	Δ Light	Δ Assimilation (μmol C .h^{-1}.(mg Chl)$^{-1}$)	J_{StSyn} (μmol C .h^{-1}.(mg Chl)$^{-1}$)
0.15	1200	1550	350	65	42
0.2	1050	1175	125	50	36
0.25	925	975	50	40	23
0.5	-	-	0	0	-36

brought about by low external P_i, described in section 4.3.1. Comparison of Figures 4.3 and 4.11(C) shows that in both cases, in the slow steady state, conserved P_i is predominantly in the form of PGA, total sugar phosphates are low, and P_i very low. Flux responses are also very similar. In Figures 4.2 and 4.11(A and B) assimilation, export, and storage fluxes all responded monotonically, and in the same sense, to changes in the relevant parameter.

The light levels at which the switch seen in Figure 4.11 occurs, and indeed whether or not it occurs at all, are sensitive to θ_{PGA} (which modulates the V_{max} of the TPT toward PGA - Section 4.2.3). As shown in Table 4.5.1, the effect of decreasing θ_{PGA} is to increase the light levels at which both positive and negative transitions occur, and the distance between these points, in both the dependent (i.e. light) and independent variables. It should be noted that although switching is absent for $\theta_{\mathrm{PGA}} \geq \sim 0.5$, under these conditions flux in the starch synthesis/degradation branch is negative, even at high light levels, and cannot therefore represent a physiologically sustainable state.

4.5.2 Response of Control Characteristics

Perhaps not surprisingly, the two steady states described above have quite different control characteristics. Figure 4.12 shows flux control coefficients over the input and output fluxes as light changes. These were recorded with the same model parameters as previous figures, but for the sake of clarity only results from decreasing light values are shown.

As described in Section 4.4.3, under conditions of high light control of assimilation flux is dominated by SBPase. At the point at which the system switches from fast to slow steady state there is an abrupt change in the distribution of flux control: SBPase, StPase, and Ru5Pk assume negative flux control coefficients in decreasing order of magnitude,

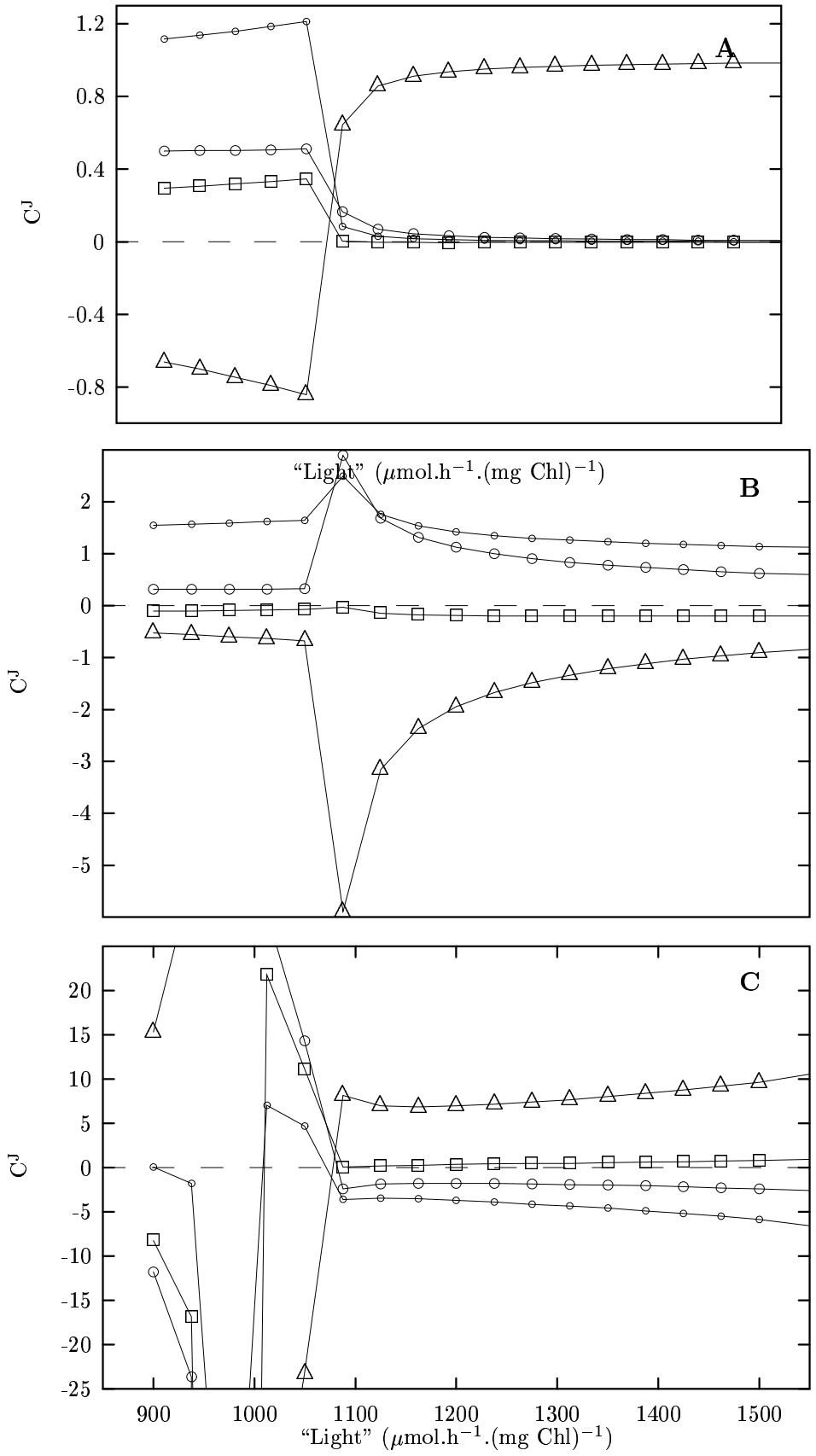

Figure 4.12: Effect of "light" on control characteristics of **A**: Assimilation, **B**: TPT, **C**: starch fluxes by (□) - FBPase, (△) - SBPase, (∘) - TPT, (○) - light reactions. Other reactions omitted for clarity. Rubisco and StSyn had negligible C^J over all fluxes, including their own.

Table 4.4: Control of input and output fluxes at fast and slow steady states

State	Fast			Slow		
Flux	Assimilation	TPT	Starch	Assimilation	TPT	Starch
+ve C^J	SBPase	TPT	SBPase	TPT	TPT	Light
-ve C^J	-	SBPase	TPT	SBPase	SBPase	SBPase

and TPT, the light reactions, StSyn, FBPase, and the fast reactions take significant control, again ordered by decreasing size.

The TPT flux becomes hypersensitive to the activity of individual reactions as the switch is approached, although there is much less difference in control coefficients on either side of the switch than is seen for the assimilation flux. Positive control is more or less evenly shared between TPT, the light reaction, the fast reactions, and, to a lesser extent, StPase. In the slow steady state, control is dominated by TPT, with relatively little control held by StPase and the light reaction, and almost none by the fast reactions. SBPase maintains negative control throughout.

Interpretation of control of the starch flux is made complicated by the fact that the starch flux changes sign from positive to negative just after (i.e. at a slightly lower light value than) the fast-slow transition. The very large peak seen at 975 μmol.h^{-1}.(mg Chl)$^{-1}$is due to the sign change in flux, not the switch. At higher light levels positive control is held almost entirely by SBPase, with negative control distributed amongst other reactions, the greater part of which is held by TPT. At the light-dark transition the magnitude of all control coefficients increase markedly with SBPase, StPase and Ru5Pk (descending order of magnitude) gaining negative control. Most of the positive control is shared between the light reaction, StPase and FBPase, with TPT and the fast reactions playing a relatively minor role. On the low light side of the discontinuity caused by the change in sign of the starch flux the signs of all control coefficients change but the relationships between them remain the same. It is interesting to note that in the slow steady state StSyn and FBPase have almost identical flux control coefficients.

The major negative and positive control coefficients of the input and output fluxes at fast and slow steady states are summarised in Table 4.4

4.5.3 Dynamic Response

Figures 4.13 and 4.14 show the dynamic response of assimilation the Calvin cycle model to changes in light. All other variables follow the same pattern. When the model is in the fast steady-state, transient changes are heavily damped and of short duration. When the initial fast-slow transition occurs there is a burst of moderately damped oscillation of ~ 40 μmol.h^{-1}.(mg Chl)$^{-1}$amplitude and ~ 13 s period. As light is further decreased these oscillations become smaller in initial amplitude, and more heavily damped.

When light is increased this process reverses: oscillations are increasingly sustained and of greater amplitude. The model exhibits continuous oscillation at for the two light values immediately before the slow-fast transition, hence steady-state values were not found for these points. Once the model has returned to the fast steady state the response is as before. As can be seen in Figures 4.13 and 4.14 light values have a major effect on the amplitude and damping time of oscillations in the model, however the period remains unchanged.

This general pattern of dynamic behaviour has been in all cases in which switching in the model has also been observed. The continuous oscillation prior to the slow-fast transition is not always seen: it is more common for the model to exhibit a brief spell of exponentially increasing (i.e. $\tau > 0$) oscillation at the transition point, which then settles on to the fast steady state.

4.6 Conclusions

A detailed computer model of the Calvin cycle has been constructed and investigated. Responses to two physiologically important environmental parameters, external P_i signalling demand for TP, and light, which provides the necessary energy to drive the cycle, were described. It was demonstrated that the inclusion of starch phosphorylase provides a high degree of protection against overload breakdown under conditions of high TP demand.

The response to both of these parameters is bi-phasic: under conditions of low $P_{i_{ext}}$ or light, the Calvin cycle exists in a slow steady-state which contrasts with the fast steady state which is seen when either of the parameters are raised above a certain threshold.

The slow steady-state is characterised by a distribution of metabolite concentrations

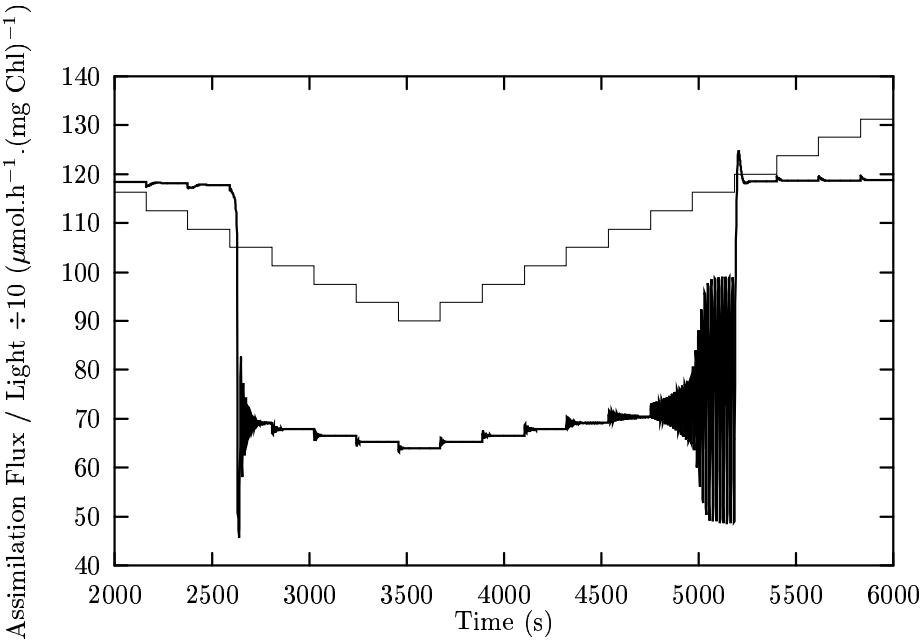

Figure 4.13: Dynamic response of the Calvin cycle model to changes in light: The stepped curve shows the light level (scaled for convenience) as a function of time. The continuous curve is the assimilation flux.

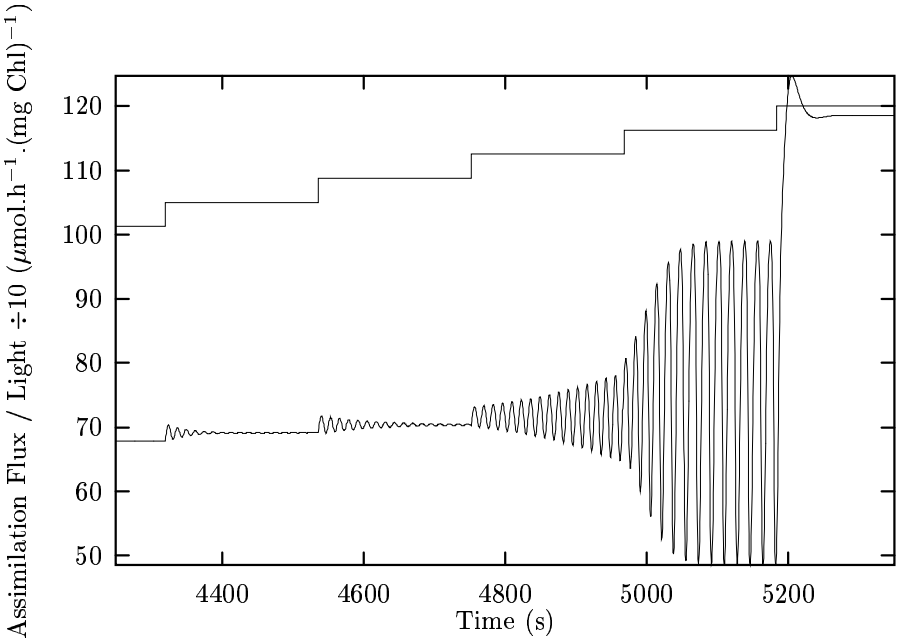

Figure 4.14: Detail from Figure 4.13 showing the slow-fast transition

in which the predominant species is PGA, and in which the total sugar phosphate concentration is considerably greater than that of P_i. Another characteristic of this state is that assimilation, export, and storage fluxes have comparable responses to parameter changes.

In the fast steady-state there is a much more even distribution of metabolites, and responses of external fluxes are qualitatively different: Assimilation flux is, to all intents and purposes, insensitive to parameter changes, and export and storage fluxes respond in strictly opposite sense to one another.

The model was also shown to exhibit damped oscillation in response to changing light levels. In the fast steady-state this is extremely heavily damped, but much less so in the slow steady-state, in which it is possible to observe what appears to be undamped oscillation, under certain values of light reaction activity.

Metabolic control analysis demonstrated a number of surprising features:

1. Significant differences exist in the patterns of control of assimilation, export and storage fluxes.

2. In the fast steady-state, assimilation flux appears to be entirely controlled by SBPase.

3. Control coefficients can vary greatly according to environmental conditions. The transition between the two steady-states is accompanied by a major rearrangement of the distribution of control.

Most of the results in this chapter are reasonably consistent with experimental observation (as will be discussed in chapter 6) and physiological function, although some plant physiologists may be surprised by item 2 in the list above. However, the model has many parameters, which represent observations made on different organisms under different conditions. It is therefore not clear whether the behaviour thus far described represents typical, real, biological behaviour, or is a result of some particular, but unrealistic, combination of parameter values in the model. The next chapter make an attempt to address this issue.

Chapter 5

Use of Evolution Strategy to Investigate a Computer Model of the Calvin cycle

5.1 Introduction

The results presented in the previous chapter were obtained by a conventional approach to modeling, echoing scientific method: changes to parameters were made and the effects recorded, explanations were sought to relate the observations to the model, and these were tested by further changes to parameters. The approach has met with some success, inasmuch as the results, whilst containing some surprises, are capable of being given physiological interpretation, are broadly consistent with experimental observation, and do not violate known physical laws.

However, the validity of such an approach depends upon the quality of the original parameter values used. In the case of Petterssons model [91], kinetic data was drawn from a variety of experimental investigations into different organisms, under different conditions, and by different workers. In addition to concern about this implicit assumption, (that different organisms maintain constant and equal enzyme activities) examination of Petterssons data set suggests that it may not be internally consistent.

Firstly this set assigns V_{max} values for rubisco and SBPase as 340 and 40 μmol.h^{-1}.(mg Chl)$^{-1}$ respectively. However, the stoichiometry of the Calvin cycle is such that the steady-

state assimilation flux is three times the flux through SBPase, thus these values of V_{max}represent an almost three fold "overinvestment" in rubisco. Given the very low catalytic constant of this enzyme and its high molecular weight, it is not obvious that these values represent optimal protein investment.

Furthermore, the various parameters not described by Pettersson were estimated in an entirely *ad hoc* fashion. The V_{max} of the starch phosphorylase was made equal to that of starch synthase on the basis of the assumption that net maximum starch degradation would be approximately equal to the maximum rate of synthesis. The rate constant (initially equal for all) for the fast, reversible reactions was an arbitrary constant, sufficiently large to maintain a flux through the cycle.

Thus, although individual parameter values used in the model of the previous chapter lie within a realistic range, it is not at all clear that the relationships between parameter values are realistic. In an attempt to determine a parameter set with more realistic relationships the evolution strategy (ES) algorithm ([61,10] and chapter 3) was used to optimise the model with the goal of maximising the assimilation flux, whilst simultaneously minimising the protein load required to support the flux. In this chapter results of optimisations are presented, with detailed discussion deferred to chapter 6.

5.2 Implementation

The optimisations were performed using the Scampi and ES software libraries described in chapters 2 and 3. The model used was that described in chapter 4, modified so that the fast reactions had individual rate constants, instead of a single rate constant for all.

Optimisations were carried out using variable mutation sizes, a population size of 600 with the 100 fittest individuals going through to the next generation, an initial mutation size of 0.02, and a mutation rate of 1.0. In this instance high fitness values are considered more fit than low (in contrast to the the curve fitting described in section 3.4.3). The populations were evolved over a period of 400 generations.

5.2.1 Fitness evaluation

As described below, several versions of the fitness evaluation function, `Eval_f()`, were investigated, but all were incremental developments of that shown in Code fragment 5.1. The general goal of the optimisation is to maximise the assimilation flux, whilst

simultaneously minimising the total protein requirement (protein load).

Code Fragment 5.1 Basic fitness evaluation function (C language)

```
void Eval_f(Organism_t org){

  enum SPI_err err ;
  const double tol = 1e-6  ;
  double  dur = 2.0 ;
  const int iters = 20, retries = 4 ;
  int points = 200 ;

    PutSubset_md(mod, Param, ParamNames, org->Genome) ;

    Sim_to_SS(mod, &dur, &points, tol, iters, retries, &err) ;
    if((err == OK)
        org->FitnessVal = GetMDval(mod, Vel, "Rubisco") / ProteinLoad() ;
    else
        org->FitnessVal = 0.0 ;
}
```

In this example the Calvin cycle model has been assigned the globally accessible identifier `mod`, and the names of the parameters represented in the organisms' genomes are the (`NULL` string terminated) global array of strings, `ParamNames`. The function `ProteinLoad()` calculates the current protein load of `mod`, as described below. The other functions used by `Eval_f` are described in detail in chapter 2.

Fitness values, assimilation rates, and protein loads of the surviving population were recorded at increasing intervals (because the most rapid change occurs in the early generations), and after the last generation the complete parameter and concentration vectors were saved (using the `OutputDesc_t` mechanism) for the whole population. Other characteristics of individuals in this final population were then determined by writing small programs that reloaded the parameter and corresponding concentration vector of each individual.

5.2.2 Calculation of protein load

The protein load of an individual was calculated as:

$$L = \sum_{i=1}^{i=N} \frac{v_i^{cur}}{v_i^{Orig}} . l_i \tag{5.1}$$

where L is the total protein load of an individual organism, N the number of reactions, v_i^{cur} and v_i^{Orig} the (slow) activity or (fast) rate constant of the i^{th}reaction, and l_i the load

Table 5.1: Protein costs associated with enzymes and reactions in the Calvin cycle

Enzyme	Cost g_{prot}.min.M^{-1}	Source
Aldolase	1.69	Spinach leaf
G3Pdh	2.27×10^{-2}	"
3PGK	1.65×10^{-2}	"
R5Piso	1.62×10^{-4}	"
Rubisco	4.92×10^{1}	"
StSynth	1.14×10^{1}	"
StPase	4.34×10^{-3}	"
X5Pepi	3.49×10^{-2}	Yeast
TKL	5.75×10^{-1}	"
PGM	9.48×10^{-3}	"
PGI	1.18×10^{-1}	"
TPI	1.24×10^{-3}	"
FBPase	1.37×10^{-2}	"
SBPase	1.37×10^{-2}	"
TPT	4.71	mean value
Ru5Pk	4.71	mean value
Light reactions	10.0	arbitrary

associated with the i^{th}reaction. Scaling the load of an individual against the original load circumvents dimensional problems caused by the mixture of rate constants and V_{max} values in the parameter vector.

Loads for individual reactions were calculated from the k_{cat}, molecular mass, and enzyme concentration values, reported by Albe *et al.* [7], as mass of protein per unit activity (g.min.mole^{-1}). If plant isoforms were not reported then the figure for most closely related available form was used, as detailed in Table 5.1. The assumption here is that yeast is more closely related to plants than are mammals. Although the assumption is arbitrary it was used consistently. Three proteins, SBPase, TPT and Ru5Pk, were not reported in [7]. SBPase was assigned the same load value as FBPase (the two enzymes have a high degree of homology [25]), and TPT and Ru5Pk the arithmetic mean of the others. In the model used here, one enzyme, ATP synthase, serves to represent all light reactions. Thus this one enzyme was arbitrarily assigned a relatively high protein cost, intended to reflect the investment made not only in this enzyme, but in the whole thylakoid apparatus. The information is summarised in Table 5.1.

5.3 Initial Optimisation Attempts

Despite the apparent simplicity of the goal, early optimisations revealed a number of pitfalls, which although unexpected, with the benefit of hindsight proved entirely understandable, but none the less required some effort to remedy. Some of these are described here in order explain the form of `Eval_f()` used to generate the final results.

The first attempt to optimise the model used the `Eval_f()` as shown above, and all parameters were allowed to mutate. The first problem to be encountered was the abrupt (within one generation) extinction of populations that had appeared to be evolving satisfactorily. It transpired that this was due to a lack of isolation between organisms, (due to the communication provided by the concentration vector in the global model): if a single organism possesses a genome which causes all concentrations to fall to zero, subsequent organisms inherit this condition, and the stoichiometry of the Calvin cycle is such that progress is impossible under these conditions. Thus the effects of a lethal mutation in a single organism infect the whole population. The solution is to save an initial (global) concentration vector before the population is evolved, This vector is then loaded into the model before each the fitness evaluation function attempts to determine a steady state.

The next problem to be encountered was the presence of negative values in the genome. This was revealed when a population that had apparently converged showed a sudden improvement in fitness, coinciding with the assimilation flux exceeding rubisco V_{max}. Investigation revealed that this was due to one of the rubisco inhibition constants assuming a negative value. The initial solution was for the fitness evaluation function to kill those organisms with negative valued genes. However, this solution was not entirely satisfactory, as it was found that even after many generations of ES, a large proportion ($\sim 50\%$) of the population was being killed off in this fashion. Thus, in order to maximise the number of viable individuals, a modified mutation function was used, that "reflected" negative gene values back into the positive domain. Although this will change the distribution of mutated values, the algorithm does not depend on new values being drawn from any particular distribution, and the original distribution was chosen for purely pragmatic reasons [61, 10].

With these modifications in place it proved possible to evolve the through many hundreds of generations, maintaining viable and improving populations. However, it was observed in the final generation, that StSyn activity had fallen to near zero: The

ES algorithm was sacrificing the ability to store starch (which does not support the goal of increasing assimilation), in order to reduce the total protein load. This was undesirable for two reasons. Firstly real chloroplasts do synthesise starch; eliminating the ability thus makes the model qualitatively unrealistic. Secondly it appears that much of the model behaviour described in the last chapter stems from the fact that carbon flux through the cycle has two degrees of freedom, and removing one would mean that the evolved model was no longer structurally comparable to the original. The solution was to modify `Eval_f()` such that those organisms maintaining a net starch synthesis flux close to 50% of the assimilation flux are treated as more fit than those with otherwise comparable characteristics maintaining a starch synthesis flux displaced from 50% of assimilation.

Although at this stage, the behaviour of the population under ES appeared satisfactory, the steady state solver `Sim_to_SS()` (section 2.5.4) was reporting numerous failures to achieve steady state, and even more warnings of individual points out of tolerance in the integrator. Although it was not possible to diagnose a cause on the basis of this output alone, the behaviour was thought to be unsatisfactory, and in an attempt to overcome the problem, `Sim_to_SS()` was substituted for `Ev_to_SS()`, which does not depend on integration (see section 2.5.4).

The result of doing this was to greatly reduce the number of failures to attain steady-state. However, subsequent investigation of the dynamic properties of individuals in the final populations revealed that the true long term behaviour of the model was a limit cycle oscillation about an unstable focus (similar in character to that shown in Figure 5.8.C). Although of possible theoretical interest, such behaviour has never been experimental observed, and so `Eval_f()` was modified to eliminate it. This is quite straight-forward, as the state may be identified by examination of the eigenvalues of the jacobian of the system: it is characterised by the presence of complex conjugate values, with positive real parts.

It also transpires that this instability was the cause of the problems with `Sim_to_SS`, as the system in this condition has a static steady state, it will not be possible to simulate onto it. The oscillation contains very rapid transients, and this was the cause of individual points being out of tolerance in the integrator.

Consideration of all of these points points led to a final form of `Eval_f()`:

Code Fragment 5.2 Final fitness evaluation function (C language)

```
void Eval_f(Organism_t org){

  enum SPI_err err ;                              /* Scampi error return */
  const int MaxGens =  10 ;                     /* constants of Ev_toSS() */
  const double tol = 1e-6, MuteSize = 0.01, MuteRate = 1.0 ;
  double Vrub, PGM, StErr;  /* rubisco and PGM fluxes, starch error value */


    KillOrg(org) ;                          /* start by assuming the worst */
    if(!HasNeg(org->Genome, org->LenGenome)){          /* all +ve genes ? */
        PutSubset_md(mod, Param, ParamNames, org->Genome) ;/* load genome */
        PutMDvec(mod, Conc, OrigConcs) ;   /* load good concentration vec */
        Ev_to_SS(mod, tol,MaxGens,  MuteSize, MuteRate,  &err) ;
                                                        /* get steady state */
        PGM = 6.0 * GetMDval(mod, Vel, "PGM_ch") ;    /* C flux to Starch */
        Vrub = GetMDval(mod, Vel, "Rubisco") ;              /* assim flux */
        if((err == OK) && IsStable(mod)){        /* no problems with SS ? */
            StErr = fabs((Vrub/2.0)-PGM) ;/* penalty for Star # 0.5 assim */
            org->FitnessVal = Vrub /(StErr+ProteinLoad()) ;
        }
    }
}
```

which assumes the existence of functions `HasNeg()`, which checks for the presence of negative gene values and returns an appropriate boolean result, `IsStable()` which determines the stability of the model, using the eigenvalue function described in section 2.6.2, and `ProteinLoad()` which calculates the protein load, as described by equation 5.1. Other functions used in `Eval_f()` are described in chapter 2.

5.4 Results

5.4.1 Effect of ES on fitness characteristics of the population

Progress of the population

The progress curves for assimilation, protein load, and fitness of the hundred fittest individuals in the population are presented in Figure 5.1. The extrema of the vertical bars indicate the values associated with the least and most fit individual in this population. In the cases of assimilation and protein load, individuals do not consistently take the highest or lowest values. If these were to be drawn as continuous curves, for the most and least fit individual, the two would continually cross and recross. Furthermore the extrema do not necessarily indicate limits of assimilation or protein load.

As expected, fitness values increase monotonically, and asymptotically, to a value of $\sim$ 1.8 $g_{prot}.min^{-1}.mole^{-1}$, representing a three-fold improvement on the starting position. There is a clear point of inflection in the 25^{th}generation, and a less clearly

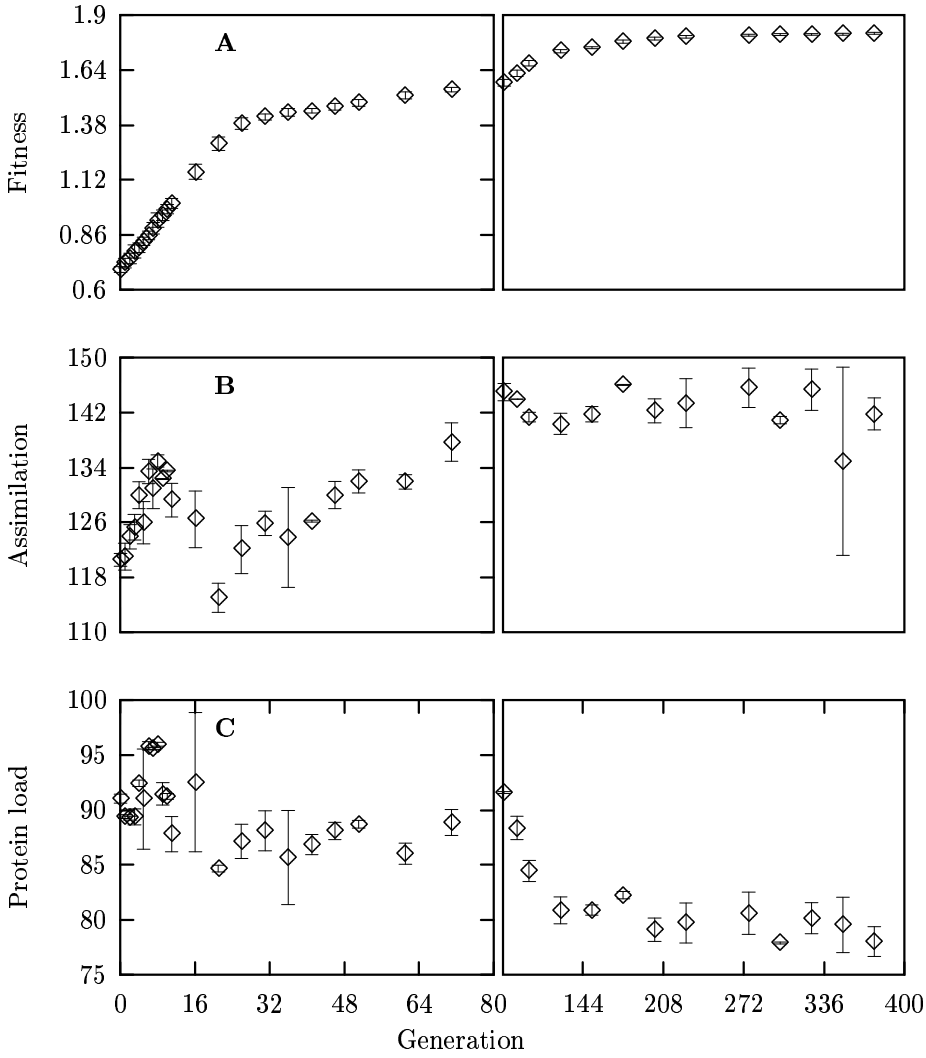

Figure 5.1: Progress curves of a population of Calvin cycle models under ES algorithm: **A** Fitness ($g_{prot}.min^{-1}.mole^{-1}$), **B** Assimilation flux (μmol C $.h^{-1}.$(mg Chl)$^{-1}$), **C** ($g_{prot}.min^{-1}.mole^{-1}$)

Table 5.2: Fitness characteristics in the first and final generation of a population of Calvin cycle models under ES. As distributions are generally non-normal, values are quoted as median and relative inter-quartile range (Rel. IQR)

		Generation 1		Generation 400	
Name	Unit	Median	Rel. IQR	Median	Rel. IQR
Assimilation	μmol C .h^{-1}.(mg Chl)$^{-1}$	119	0.017	143	0.043
Protein load	g_{prot}.min^{-1}.mole^{-1}	90	0.017	70	0.041
Fitness	?dims	0.69	0.013	1.8	0.04

defined on at the 100^{th}generation. In contrast to the fitness progress curve obtained for the fitting of lactate dehydrogenase data in section 3.4.3, there is no initial lag phase.

The progress curves for assimilation flux and protein load (Figures 5.1.B and C), are more complex, although, as may be expected, there is an overall increase in assimilation flux, and decrease in protein load. The assimilation flux rises steadily in early generations, but drops sharply at the 25^{th}generation, coincident with the point of inflection in the fitness progress curve. Thereafter, the assimilation flux increases again, but much more slowly than previously, and becomes stable after about one hundred generations.

The protein load shows no clear pattern of behaviour in the early generations, but undergoes a sharp drop in the 25^{th}, then rises to a peak value which is greater than the starting point, and then declines steadily over the remaining generations.

Characteristics of the final generation

In the final (400^{th}) generation, 327 individuals were viable, Figure 5.2 presents the distributions of fitness, assimilation flux, and protein load, and compares them with those of the fittest 100 individuals from the first generation (i.e. after mutation and fitness evaluation, but without reproduction), the information is summarised in Table 5.2.

In common with most other attributes, these distributions can be described as non-normal with a high degree of confidence[1] . It is clear from Figure 5.2 that, in addition to changing median values of observed properties, an effect of ES is to change the shape of the distribution. The fitness distribution has a strong positive skew in the first generation, but is strongly negatively skewed in the final generation.

The distribution of assimilation flux in the first generation suggests the possibility

[1]As determined by use of the Kolmogorov-Smirnof (K-S) test, [100]

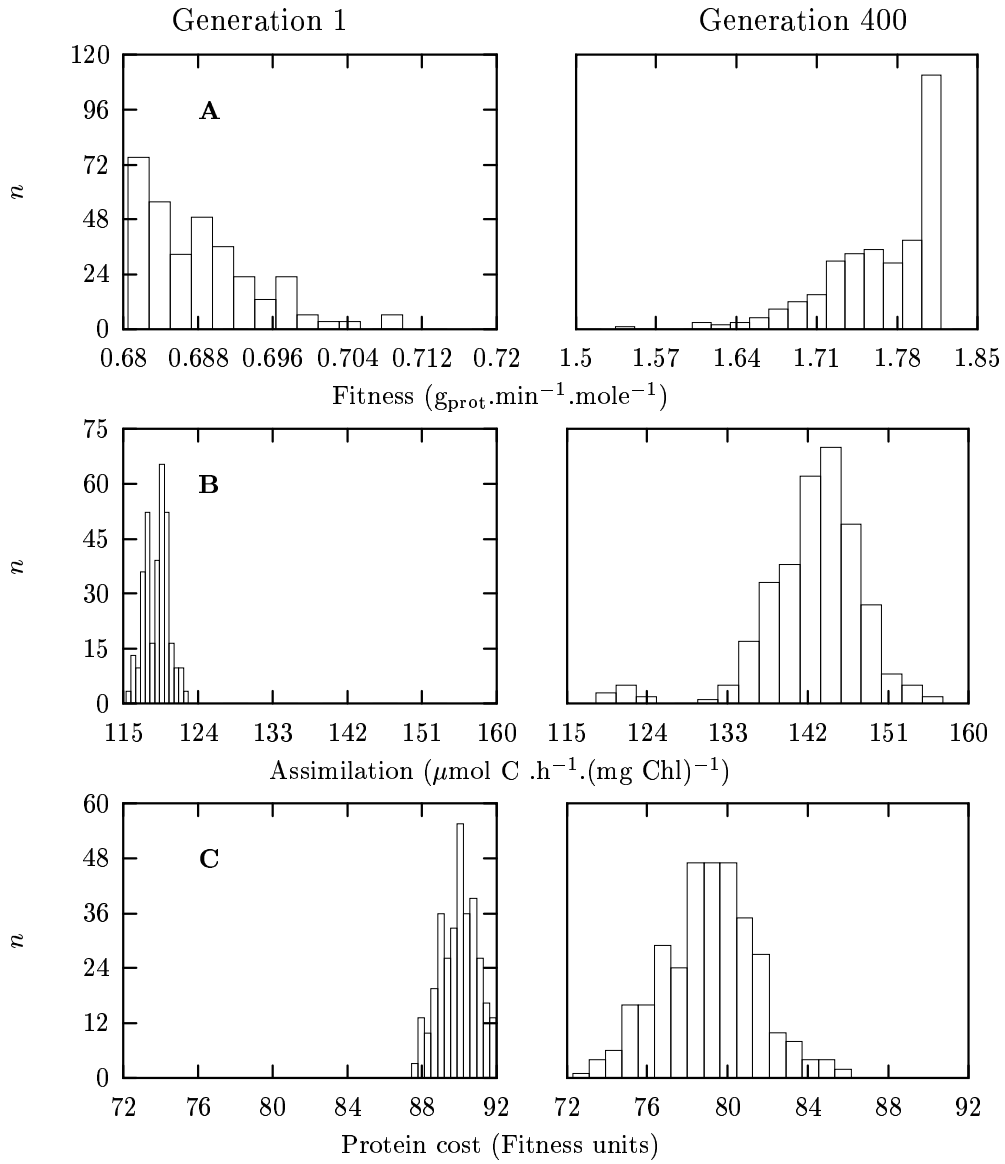

Figure 5.2: Distributions of fitness characteristics in the first and final generation of a population of Calvin cyvle models under ES: **A** Fitness, **B** Assimilation flux (μmol C .h^{-1}.(mg Chl)$^{-1}$), **C** Protein costt ($g_{prot}.min^{-1}.mole^{-1}$)

of two distinct populations of approximately equal size. In the final generation the impression remains, although one sub-population is clearly much smaller than the other, the distance between the two is much greater. Use of the K-S test suggests that it would be unwise to simply dismiss the smaller sub-population as "outliers", as this calculates that the probability of obtaining this distribution from a normally distributed (statistical) population, as 7.4×10^{-3}. The distribution of many other values in the final population also suggest the existence of a small sub-population. Unfortunately time constraints have precluded the separate isolation and characterisation of this sub-population, and its significance remains unclear.

Comparison of initial and final values in Table 5.2 shows that ES achieved a 20% increase in assimilation flux, and a 12% reduction in protein load. Thus most of the improvement in fitness was achieved by minimising the penalty incurred for starch synthesis flux deviating from 50% of assimilation flux.

5.4.2 Model characteristics of the final population

Enzyme activities

As noted previously, normal distributions tend to be the exception, and not the rule, in the data sets generated by this investigation. Of the nineteen model parameters under the influence of ES, only 4 had K-S(normal) > 0.5. Thus Table 5.3 describes values in the final generation as median and Relative inter quartile range Inspection of this table shows that eight of the activities and rate constants[2] increased, and the rest decreased.

The largest overall change was that of StPase, the activity of which was reduced to slightly less than 3% of the starting value. However, as the fitness evaluation function takes no account of the desirability of the possibility of starch degradation, it seems likely that a decreased protein load was achieved at the expense of this enzyme. It is also notable that the Relative inter quartile range of this step is an order of magnitude greater than the others, which may be an indication that the activity is still approaching optimum. However as the activity is so low, further reductions are unlikely to furnish any great improvement in fitness.

With the exception of StPase, changes in activity (relative to initial value), fell within the range 0.2–2.7, and eleven of the nineteen within the range 0.8–1.2, of which

[2]At the risk of sacrificing strict accuracy for convenience, both of these quantities will subsequently reffered to as "activities"

Table 5.3: Values and spread of evolved parameters in the final generation, compared with their initial values. See section 4.2 and table 4.1 for explanation of the abreviations.

Name	Initial value	Median	Rel. IQR	Rel. change
G3Pdh	5×10^{8}	1.3×10^{9}	0.032	2.7
FBPase	2×10^{2}	3.8×10^{2}	0.066	1.9
SBPase	40	57	0.058	1.4
F.TKL	5×10^{8}	5.8×10^{8}	0.051	1.2
PGI	5×10^{8}	6.2×10^{8}	0.081	1.2
TPI	5×10^{8}	6×10^{8}	0.084	1.2
PGK	5×10^{8}	6×10^{8}	0.037	1.2
PGM	5×10^{8}	5.4×10^{8}	0.052	1.09
StSyn	40	39	0.074	0.98
X5Pepi	5×10^{8}	4.8×10^{8}	0.056	0.95
S.Aldo	5×10^{8}	4.4×10^{8}	0.059	0.89
F.Aldo	5×10^{8}	4.2×10^{8}	0.074	0.84
R5Piso	5×10^{8}	4×10^{8}	0.14	0.81
S.TKL	5×10^{8}	2.8×10^{8}	0.13	0.55
"Light"	3.5×10^{3}	1.7×10^{3}	0.056	0.49
rubisco	3.4×10^{2}	1.6×10^{2}	0.042	0.46
TPT	2.5×10^{2}	1.1×10^{2}	0.061	0.44
Ru5Pk	1×10^{4}	1.9×10^{3}	0.15	0.19
StPase	40	0.98	1.3	0.024

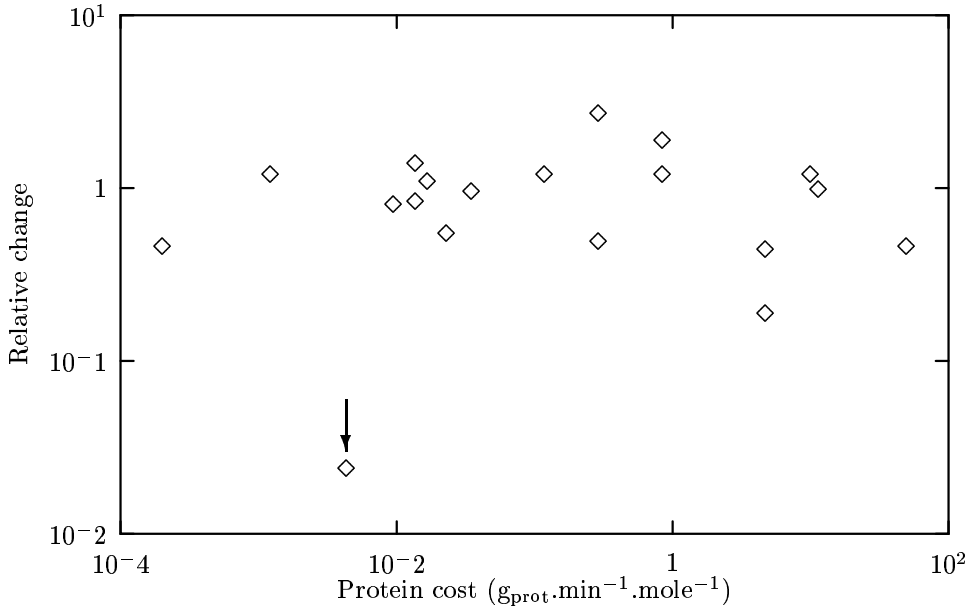

Figure 5.3: Relative change in parameters after 400 generations of ES, plotted as a function of their associated protein cost. The arrow indicates starch phosphorylase, which lies well away from the main trend.

Table 5.4: Median evolved flux control coefficients over input and output flux, and their respective values in the original model ($P_{i_{ext}} = 0.5$ mM)

Name	Assimilation		Export		Storage	
	Evolved	Original	Evolved	Original	Evolved	Original
Rubisco	6.14×10^{-1}	1.70×10^{-3}	-2.06×10^{-1}	1.02×10^{-3}	1.24	-1.68×10^{-2}
FBPase	-1.04×10^{-2}	-2.55×10^{-3}	-9.99×10^{-2}	-1.35×10^{-1}	3.07×10^{-2}	-2.94
SBPase	8.44×10^{-2}	1.00	-1.21	3.89×10^{-1}	2.77×10^{-1}	-1.09×10^{1}
Ru5Pk	1.81×10^{-2}	2.92×10^{-3}	-2.03×10^{-1}	8.48×10^{-3}	5.31×10^{-2}	2.09×10^{-3}
TPT	6.92×10^{-2}	-4.76×10^{-3}	1.44	6.11×10^{-1}	-8.73×10^{-1}	1.16×10^{1}
StSyn	5.89×10^{-2}	8.37×10^{-4}	3.91×10^{-1}	-7.81×10^{-2}	1.26×10^{-1}	-1.46
StPase	-7.08×10^{-6}	1.56×10^{-3}	-4.57×10^{-5}	1.08×10^{-1}	0.00	2.96
G3Pdh	6.09×10^{-2}	-6.56×10^{-4}	3.37×10^{-1}	-6.12×10^{-2}	5.80×10^{-2}	-1.15
Light react	9.94×10^{-2}	8.69×10^{-6}	5.60×10^{-1}	3.94×10^{-4}	9.48×10^{-2}	1.23×10^{-2}

ten were fast reactions. Of the remaining fast reactions S-TKL decreased by a factor of 0.5, and G3Pdh increased by 2.7. With the exception of StPase, this is the greatest change brought about by ES.

Of the slow reactions, FBPase and SBPase both increased their activities by a factor of 1.9 and 1.4 respectively. These had the lowest protein cost of the slow enzymes. Rubisco, ATP synthase, and TPT, all decreased by a factor of ~ 0.4, and Ru5Pk by 0.2. As might be expected, if relative activity change is plotted against protein cost (Figure 5.3), then a reasonably clear negative relationship is seen, although the enzyme with the highest cost (rubisco) was not the most diminished. It is also notable that StPase lies well away from the general trend, providing further evidence that this step has been treated atypically by the ES algorithm.

Metabolic control analysis of the final population

The median values of the flux control coefficients of individual enzymes in the final generation, over assimilation, export and starch fluxes are presented in Table 5.4, along with their respective values in the original model. Once again StPase is an exception, with negligible C^J over input and output fluxes, and will not be discussed further in this section.

Assimilation flux has undergone the greatest change in its control characteristics: the control by SBPase is greatly reduced in the final population, with most of this control having been transferred to rubisco, although the control exerted by the other steps has also increased markedly. The reduction in control by SBPase in the whole population

is not as great as might appear from Table 5.4, as its distribution is both positively skewed and relatively wide. Figure 5.4.2 shows the distributions of $C^{J_{assimilation}}_{Rubisco}$ and $C^{J_{assimilation}}_{SBPase}$. Examination of this figure suggests that control of assimilation flux within the whole population is mainly shared between rubisco and SBPase, with other steps (see Table 5.4) making small, but not negligible contributions.

In contrast to assimilation, the control of export flux has become more, not less, localised in the final generation, with positive and negative control lying mainly with TPT and SBPase respectively. Control of export by StSyn, G3Pdh, and ATP synthase also underwent an increase.

Control over starch synthesis flux[3] has undergone the greatest quantitative change, with the very large values of $C^{J_{starch}}_{SBPase}$, $C^{J_{starch}}_{TPT}$ and , $C^{J_{starch}}_{FBPase}$, being reduced by approximately an order of magnitude, although the sense of these is unchanged. In line with the assimilation control profile, $C^{J_{starch}}_{rubisco}$ has undergone a dramatic increase, apparently at the expense of $C^{J_{starch}}_{SBPase}$.

Examination of control by TPT in Table 5.4 shows that $C^{J_{Assimialtion}}_{TPT}$ is relatively small, and that $C^{J_{Export}}_{TPT}$ and $C^{J_{storage}}_{TPT}$ are of similar magnitude and opposite sign. This is consistent with the model being in the fast steady-state, as described in the previous chapter.

Determination of relationships between parameters and variables in the final generation

Two techniques were used to identify the strongest relationships in the final generation: linear regression, and Spearman's rank correlation test.

Results obtained by linear regression must be viewed with a degree of circumspection for at least two reasons: it cannot be assumed that such relationships as exist will be linear, and the algorithm used ([99] section 3) assumes that data are drawn from a normally distributed (statistical) population. The latter assumption is unsafe in the light of results described previously in this chapter.

Thus the estimated parameters determined by linear regression are unlikely to be

[3] A minor complication in considering the figures in table 5.4 relating to starch synthesis is the fact that the starch flux in the original model was negative (i.e. net degradation), hence steps with positive values of $C^{J_{Storage}}$ represent those that increase the degradation of starch. The opposite if true in the evolved population in which the carbon flux to starch is positive. The term "positive control" is used here to describe those steps for which an increase in activity results in an increase in synthesis, regardless of the the numerical sign of C^{J}.

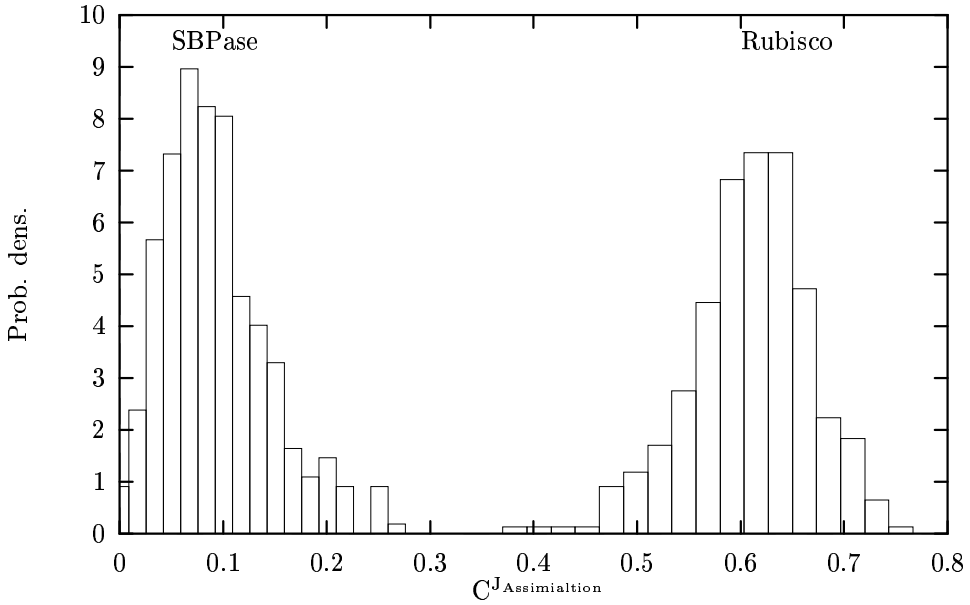

Figure 5.4: Distribution of control by SBPase and Rubisco over assimilation flux, in the final population

of any great use. However the linear regression function used here also determines an estimate of confidence in the calculated parameters, in the form of a standard deviation. Now, if we assume that, if a monotonic trend exists between two data sets, it will then be possible to approximate it (albeit poorly) to a linear function, we may use the standard deviation of the slope to calculate the probability that the slope is equal to zero, or has the opposite sign to that calculated, and use this probability as a measure of the "strength" of the relationship. Stating this in terms of the null hypothesis H_0:

> the sign of the slope of the linear function representing the relationship between two data sets is not known

then the probability of H_0, p_0 is calculated as:

$$p_0 = \int_{-\infty}^{0} \frac{\sigma_r}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-1}{\sigma_r}\right)^2} dx \tag{5.2}$$

Where the right hand side of the equation is the cumulative normal distribution function with a population mean of 1, and the relative standard deviation, $\sigma_r = \sigma/m$, where m and σ are the linear coefficient and associated standard deviation as determined by linear regression.

The numerical calculation of equation 5.2 depends on the use of the incomplete gamma function [101], which is implemented in terms of a successive approximation algorithm, and liable to fail for $\sigma_r <\sim 10^{-2}$, equivalent to a value of $p_0 \sim 10^{-12}$. The error is readily detectable, and such values will be reported as here as $< 10^{-12}$. Calculations of p_0 were made on sets of data sorted by σ_r, hence the relative position of two or more items with $p_0 < 10^{-12}$, in tables below, remains meaningful.

It was found that, under certain circumstances, the results generated from this test could be badly distorted, due to the presence of the small sub-population. The slope of the regression line was that of the line joining the centroids of the two populations, not the slope of the regression line within the main population, (despite the fact that the sub-population is less than a tenth of the size of the main). The Marquardt-Levenberg algorithm suffered from the same problem.

An attractive alternative to regression tests was Spearman's rank correlation as it is not dependent on assumptions about the distributions of the input data, and less likely to generate false positive results than parametric correlation tests [100].

This analysis was undertaken in the expectation that most of the possible relationships would be weak, or non-existent, and that it would therefore be possible to identify a small number of "key" relationships that merited further investigation. In fact this expectation proved false, and most possible relationships proved to be strong. The two statistical tests were consistent in this, both found $\sim 65\%$ of possible relationships to be significant at the 5% confidence level.

Correlation of parameters with fitness value

The exceptions to the rule that all relationships are strong are those involving fitness. Given that fitness is defined as a function of all parameters, such weak relationships would appear, at first sight, to be paradoxical. In fact it is not, and is to be expected if the population has converged to an optimum. If a clear relationship exists between any one parameter and fitness, the ES algorithm would exploit this by moving the population along the gradient, until it no longer existed.

Two parameters did exhibit a strong correlation with fitness: the atypical StPase ($p_0 2.5\times10^{-2}, p_s 7.5\times10^{-3}$), and TPT, ($p_0 \leq 10^{-12}, p_s = 3\times10^{-14}$, Figure 5.5.A), and 5.5.B), but much more representative is that between fitness and FBPase ($p_s = 0.5, p_0 = 0.4$, Figure 5.5.C).

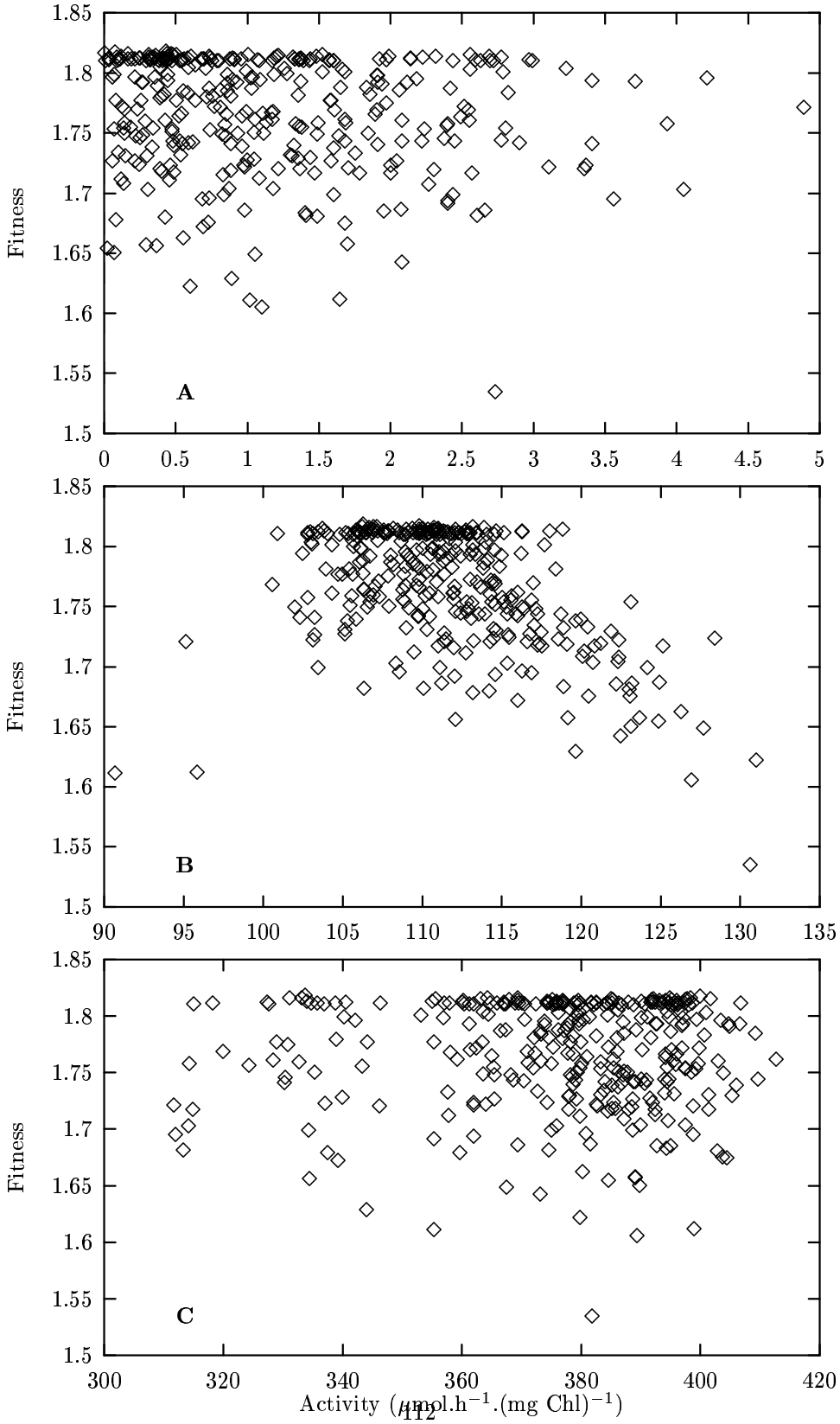

Figure 5.5: Correlations of fitness with various protein activities in a population of Calvin cycle models after 400 generations of ES: **A** StPase, **B** TPT, **C** FBPase

A possible explanation for the correlation of these two parameters with fitness is due to a threshold effect, and to some extent an artifact of the ES algorithm. If, as already suggested, StPase plays no part in increasing assimilation flux, then it may be expected that its optimum activity is zero, because of the associated cost. However, because negative activity values are not allowed, this optimum cannot be approached symmetrically, and it may be that it is this lack of symmetry that causes the corelation. A similar situation exists with TPT. As will be shown later, there is a minimum value of TPT activity, below which the system becomes unstable. As noted above, the fitness evaluation function rejects such individuals, and thus a similar lack of symmetry exists.

Correlation of parameters with protein load

Most parameters show a strong corelation with protein load. This appears to require little explanation, given that the protein load is a simple linear function of all enzyme activities. The exception to this is StPase ($p_0 = 0.16, p_s = 0.2$) shown previously to be atypical. What is more surprising is that six parameters show a strong ($p_0 < 9\times10^{-4}, p_s < 4\times10^{-2}$ for the least significant) *negative* correlation with protein load. All of these are fast enzymes within the regenerative limb of the Calvin cycle . These are not simply the enzymes with the lowest protein cost, so a tentative explanation is that investment in these enzymes enables a disproportionate saving in rubisco protein cost to be made, as these (and only these) enzyme activities have comparably strong negative correlations with rubisco activity. These observations are summarised in Table 5.5.

Table 5.5: Negative correlations between the activity of various fast reactions with protein load, and with Rubisco activity.

	Protein Load		Rubisco Activity	
Activity	Coefficient	p_0	Coefficient	p_0
F Aldo	-1.51×10^{-8}	9.08×10^{-4}	-4.74×10^{5}	4.20×10^{-2}
X5Pepi	-3.54×10^{-8}	2.70×10^{-6}	-7.10×10^{5}	7.48×10^{-6}
G TKL	-2.47×10^{-8}	2.45×10^{-8}	-1.05×10^{6}	8.80×10^{-5}
E Aldo	-3.77×10^{-8}	9.27×10^{-9}	-7.53×10^{5}	3.22×10^{-5}
R5Piso	-1.85×10^{-8}	6.62×10^{-9}	-1.66×10^{6}	8.13×10^{-6}
TPI	-2.90×10^{-8}	$< 10^{-12}$	-2.65×10^{6}	$< 10^{-12}$

Table 5.6: Relationships between assimilation flux and other factors in the final generation, as determined by Spearmans rank correlation test (p_s), and linear regression (p_0)

Activity	p_s	Activity	p_0
Light Reactions	2.56×10^{-24}	FBPase	$< 10^{-12}$
TPT	1.08×10^{-31}	Light Reactions	$< 10^{-12}$
StSyn	7.42×10^{-32}	StSynthase	$< 10^{-12}$
SBPase	3.30×10^{-34}	TPT	$< 10^{-12}$
Rubisco	2.21×10^{-59}	Rubisco	$< 10^{-12}$
Protein Load	1.71×10^{-67}	Protein Load	$< 10^{-12}$

Correlation of parameters with assimilation flux

Both p_0 and p_s identify the same five factors as being within the six strongest correlations with assimilation flux, as shown in Table 5.6. .

The five enzymes with negative correlation to protein load and rubisco also correlate negatively to the assimilation flux. This is to be expected given the strong positive correlation of protein load and rubisco activity factors to assimilation flux. These are the only factors with a negative correlation. More unexpected is the strong correlation of starch synthase with assimilation flux. It is thought that this is be a consequence of the constraint in the fitness evaluation function favouring individuals that maintain a starch synthesis flux at 50% of assimilation flux: if an individual has a higher assimilation flux, then, as long as starch synthase has a positive flux control coefficient over its own flux, we might expect a concomitant increase in starch synthase activity to maintain the balance. However, as described later in section 5.4.2, in the final population, starch synthase does have a relatively large positive flux control coefficient over assimilation, so this explanation is not complete.

The dependence of assimilation flux, upon the factors in Table 5.6 are shown in Figure 5.6. The cluster of ten individuals with relatively low assimilation flux that is clearly seen in these graphs appears to represent a distinct sub-population, as an outlying cluster of ten individuals is frequently seen when other combinations of factors (other than fitness) are plotted against one another.

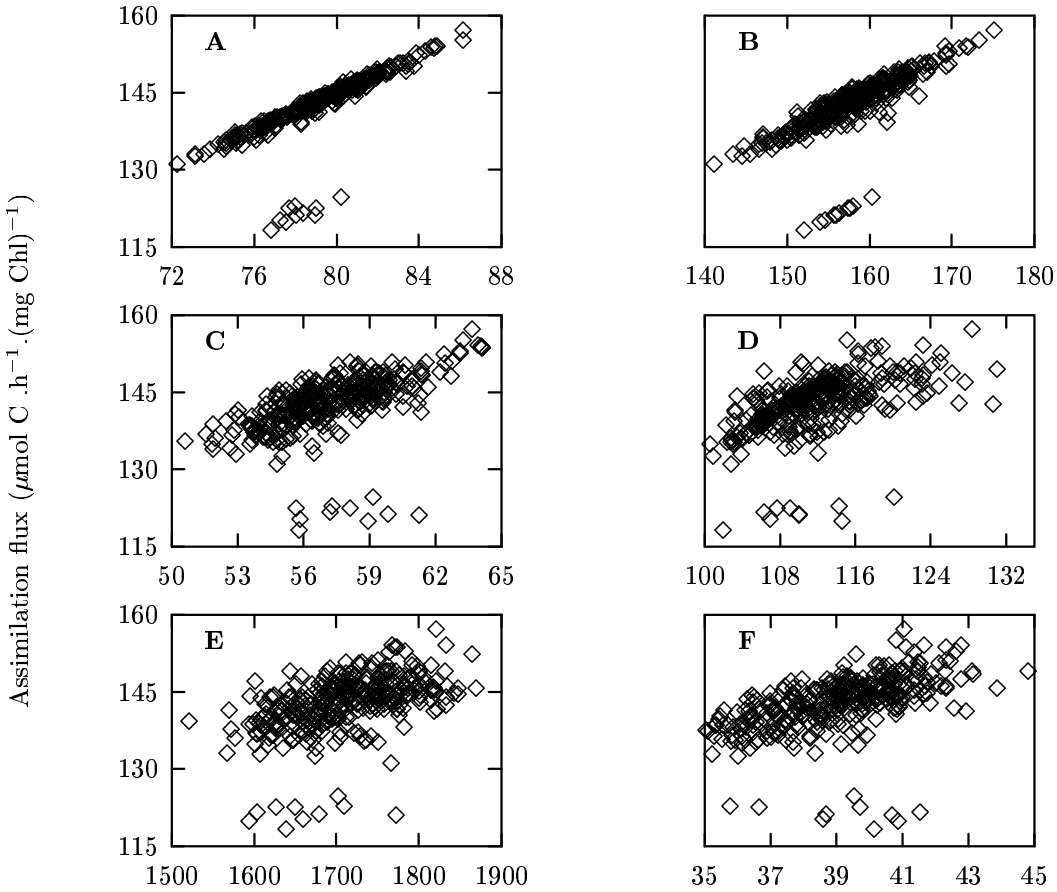

Figure 5.6: Dependence of assimilation flux upon protein load (g_{prot}.min^{-1}.mole^{-1}) and enzyme activities (μmol.h^{-1}.(mg Chl)$^{-1}$) in the evolved population: **A** Protein load, **B** Rubisco, **C** SBPase, **D** TPT, **E** Light Reactions, **F** Starch Synthase

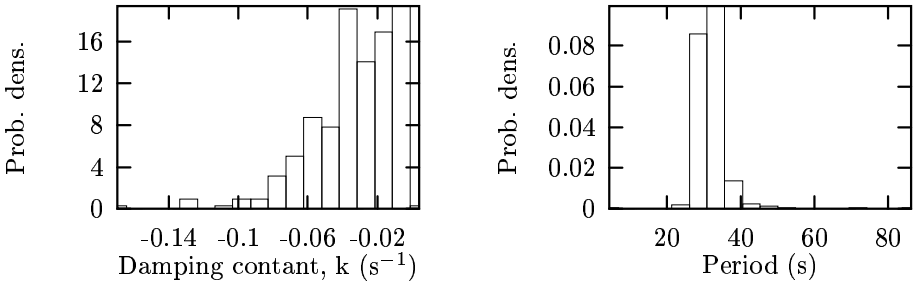

Figure 5.7: Distribution of period and damping constants in the final population, as determined from the eigenvalues of the jacobian matrix of each individual.

Dynamic behaviour

Dynamic behaviour of the final generation was determined by calculation of eigen-values using the Scampi facilities described in Section 2.6.2. Of 328 viable individuals, it proved possible to calculate sets of eigenvalues for 314. All of these had exactly one pair of complex conjugate values. The distributions of (the positive imaginary part of) these, scaled to yield the period of oscillation and damping constant in seconds and inverse seconds respectively, are shown in Figure 5.7. Neither appear normally distributed; K-S(normal)for the real distribution is 5×10^{-3}, and incalculable for the imaginary part. This is presumed to be due the the outliers in the population, as it possible to obtain a reasonable fit to a normal probability density function in the central part of this distribution.

Eigenvalues serve to characterise the response of the system to a (vanishingly) small perturbation. However, in nature, systems are subject to large perturbations, and the response to such perturbations must also be investigated. Figure 5.8 shows the time course of light reaction flux (in the fittest individual), when light reaction activity is positively perturbed for a short time, and then returned to its original value. If the relative size of the perturbation is small ($<\sim 0.09$) the system appears to display classical damped harmonic motion. However if the size of the perturbation is increased beyond a critical point the system exhibits more complex, and undamped, periodic behaviour. Surprisingly, if the perturbed parameter is not returned to its original value, then a much greater perturbation is needed in order to induce the complex oscillation (not shown).

The dynamic behaviour is also sensitive to TPT activity. Figure 5.9 shows the real part of the complex eigenvalue of the fittest individual, as a function of TPT activity.

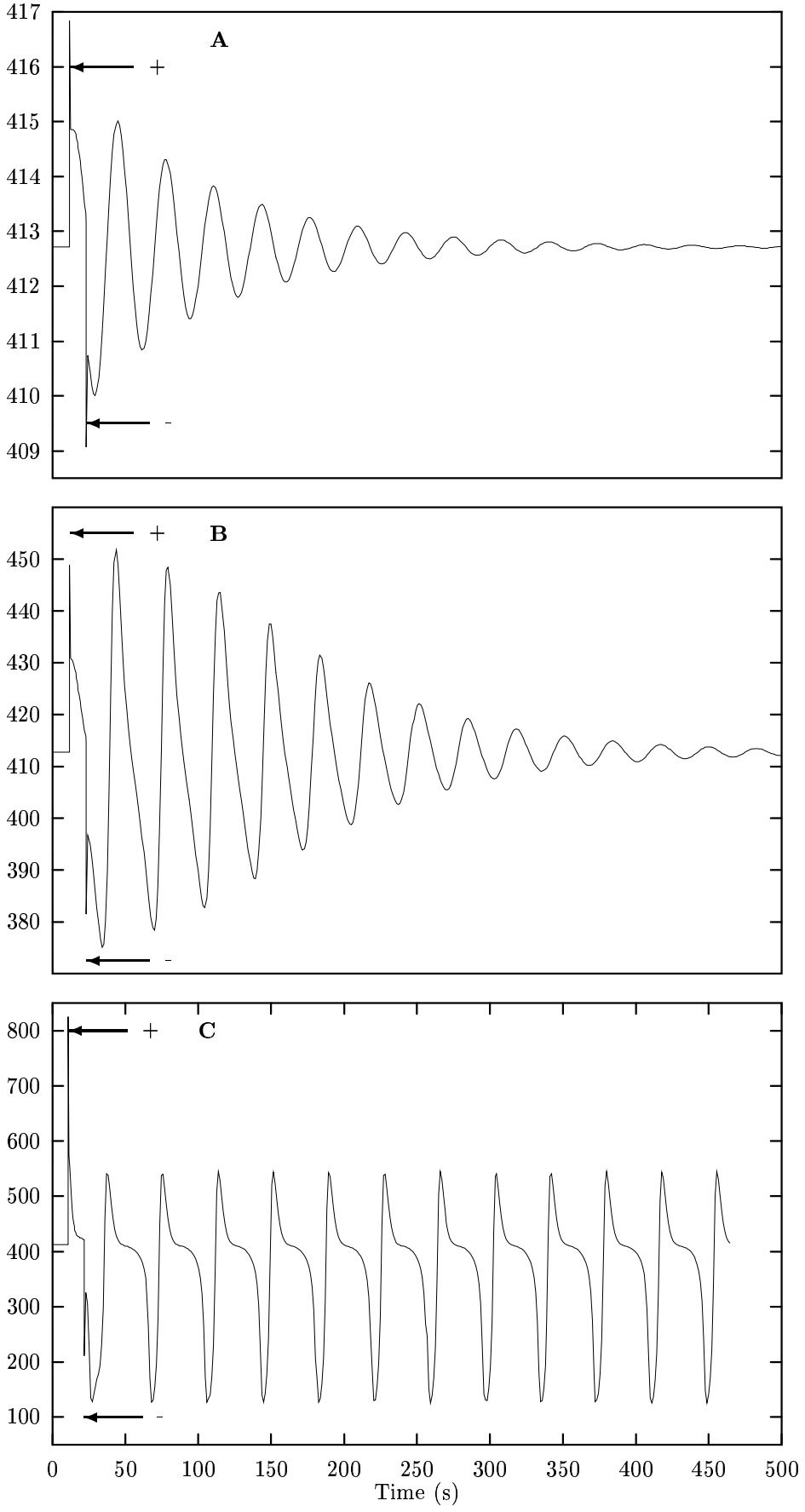

Figure 5.8: Response of light reaction flux in the fittest individual to perturbations (of relative size δ) in light reaction activity: **A** $\delta = 0.01$ damped harmonic motion, **B** $\delta = 0.08$ as a critical value is approached the response is still damped but oscillations can be seen not to be pure sine waves, **C** $\delta = 0.1$ over the critical value the system complex, undamped, oscillation

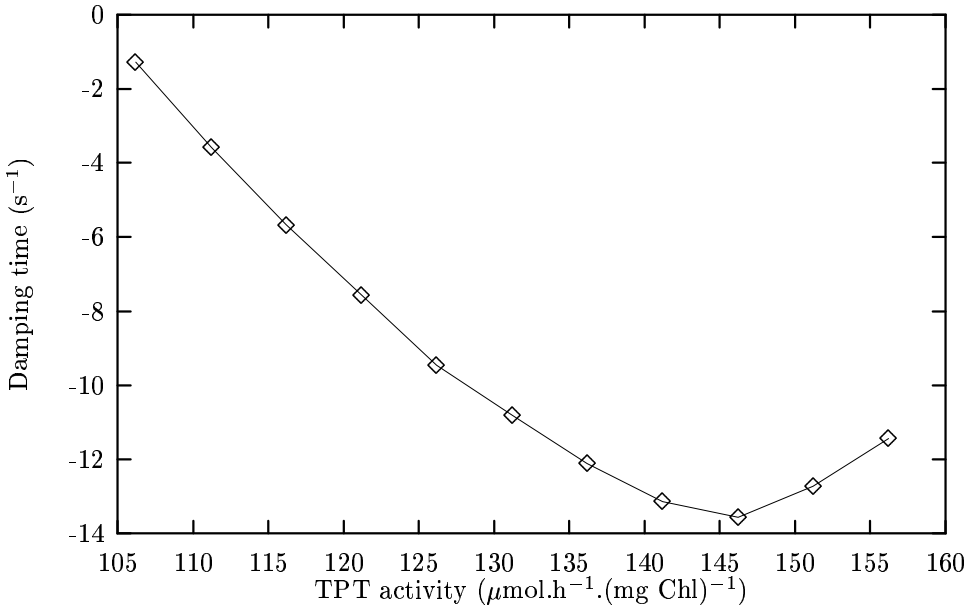

Figure 5.9: Response of damping time to changes in TPT activity in the fittest individual in the final generation of ES

With TPT activity values lower than that shown, the real part of the eigenvalue becomes positive, and the system exhibits sustained, complex oscillation of the type seen in Figure 5.8.C.

5.5 Conclusions

Subjecting a population of Calvin cycle models to a realistic selection pressure (maximising assimilation whilst minimising total protein load), has led to a significant improvement in the population. This improvement has been accompanied by a number of other changes, particularly in the control characteristics. It is notable that in the improved model, control of assimilation flux is no longer dominated by SBPase; the magnitudes of all other $C^{J_{Assimilation}}$ (apart from the atypical StPase) have increased substantially, with rubisco having most control. Comparison of parameter and variable values in the final population allows no clear conclusions as to which individual parameters, or relationships there between, are the most "important". Almost all possible pairs of quantifiable properties show strong correlations. The exceptions to this rule were those involving the

fitness value itself, which is not expected to correlate with parameters if the population has converged, and StPase, which ES appears to have eliminated from the model.

A major change in the dynamic properties has also been observed. The period and damping time have both increased, but more significantly the model becomes progressively unstable as TPT activity (or $P_{i_{ext}}$, results not included) is decreased, and below a threshold value exhibits sustained, non-linear oscillation. It is possible that this behaviour represents the slow steady-state, which has now become an unstable focus, although this has not been fully investigated. If this is the case, then it is predicted that the same behaviour could be induced by a reduction in light levels (as opposed to the short, sharp perturbation used in Figure 5.8).

As there are no experimental reports of this behaviour, the results of this ES optimisation must be treated with a degree of caution. One approach to improving these results would be to modify the fitness evaluation function to calculate total assimilated carbon over a light-dark cycle. As will be described in the next chapter, this would entail structural modification to the model, as well as the fitness evaluation function.

Chapter 6

Discussion

6.1 Toward an explanation of the Calvin cycle model behaviour in terms of skeleton models

A model with the degree of complexity of that described in previous chapters inevitably suffers from two disadvantages: firstly, it is difficult to explain the observed behaviour in terms of the known structure (although not impossible, an explanation is offered later), and secondly, analytic solutions are impossible. In an effort to overcome these problems, skeleton models were investigated, the aim being to identify the simplest structure capable of exhibiting the oscillatory and switching behaviour described in previous chapters.

6.1.1 Model structures

As Giersch [43] points out, the structure of skeleton models may reveal as much about the preconceptions of the investigator constructing them, as the system under consideration itself. Thus, before describing the skeleton models, this author's own preconceptions concerning the Calvin cycle are as follows:

1. The Calvin cycle fixes carbon by the addition of CO_2 to a five carbon compound to produce two molecules of TP[1], which is subsequently consumed by cytosolic metabolism.

[1] for the purposes of this discussion PGA is considered to be a TP

2. The Calvin cycle takes place in the chloroplast stroma, surrounded by a semipermeable membrane, such that TP must be exported in strict counter exchange for P_i .

3. The Calvin cycle is capable of reaching a true steady state, i.e. after infinite time, all intermediate concentrations are finite and positive.

4. No component of the Calvin cycle violates the laws of conservation of mass or energy.

Four models, whose topology is described in figure 6.1, were investigated. Initially reaction kinetics were modelled as first order with respect to substrate, and irreversible (with the exception of starch synthesis, when included). The kinetic equations were realistically non-linear inasmuch as all reactions were assumed to be saturable by substrate and product, with a $K_m$term being included for each substrate and product. As the object of this investigation is to investigate qualitative behaviour, values of concentration and activity are arbitrary.

6.1.2 Determination of model behaviour

Even the simplest of the models, c_0, has twelve parameters, and attempting to determine behaviour by a systematic mapping of parameter spaces is not a practical proposition. The parameter vector of each model was randomised (2000 times), steady state determined (by evolving to steady state as described in section 2.5.4, and starting from a randomised concentration vector), and when this was successful, eigenvalues and C^J of all steps over input and output fluxes determined.

6.1.3 Results

The results of the steady-state and eigenvalue analysis are presented in Table 6.1. Examination of this shows that model c_0 is inherently unstable, while models c_1–c_3 all attain stable steady-states with relative ease. Although the model c_0 does not strictly conform to the pre-conceptions of the Calvin cycle described above, it has been included here to illustrate the benefit obtained by assuming that total phosphate is conserved. A stable steady state could be found for fewer than 4% of the population of c_0 models, in comparison to a value of just over 84% for the next worst case (c_2).

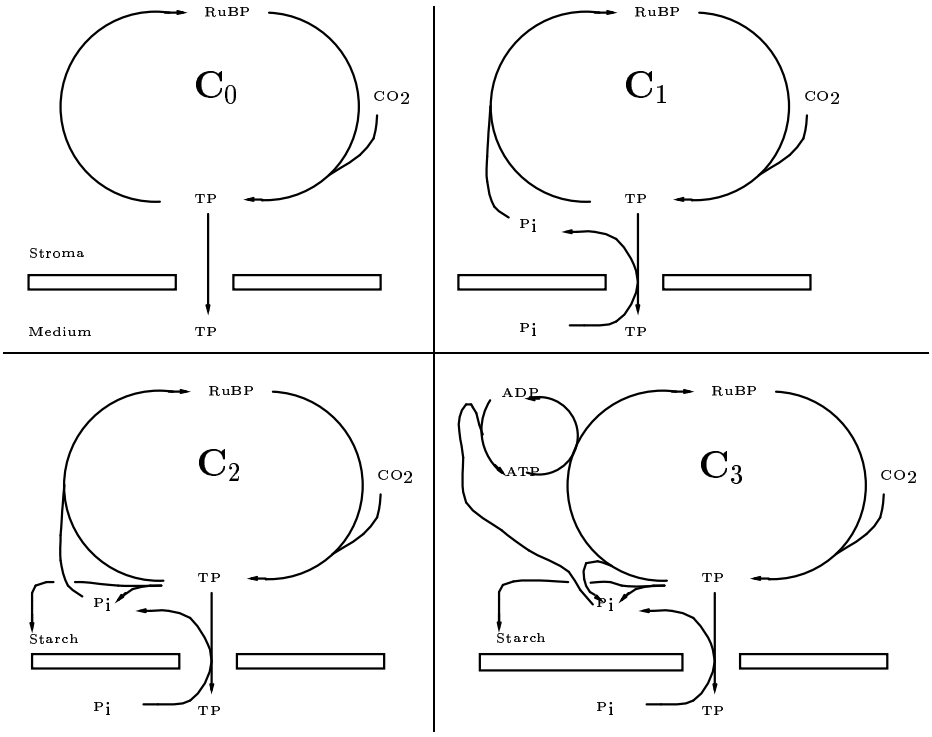

Figure 6.1: Skeleton models of the Calvin cycle

Table 6.1: Steady state behaviour in a sample of 2000 models described in figure 6.1 and determined by inspection of the eigenvalues of the Jacobian of each model at steady state, if attained.

Model	c_0	c_1	c_2	c_3
Total steady states	325	1792	1675	1733
Unstable	258	5	241	359
Damped oscillation	7	12	305	240
Unstable oscillation	3	1	2	21

Table 6.1 shows that although model c_1 is readily able to attain viable steady states, it shows little inclination to oscillate, with less than 1% of the viable parameter sets exhibiting stable (i.e. damped) oscillation. Furthermore, it is possible to argue that the lack of the starch metabolism branch in this version of the model is a more profound simplification than simply lumping reactions together into blocks, as this reduces the degrees of freedom in the net carbon flux from two to one. Thus this version of the model will not be discussed further.

Models c_2 and c_3 appear to be quite likely to oscillate, with 24 and 14% respectively of viable individuals exhibiting damped oscillation. The additional complexity introduced in c_3 thus appears to reduce the likelihood of oscillation. On the basis of these observations it is possible to propose that the "cause" of oscillation in the Calvin cycle model of earlier chapters, and therefore possibly in the Calvin cycle *in vivo* is the combination of the cyclic topology and the additional degree of freedom in the carbon flux engendered by the presence of the starch synthesis/degradation pathway.

Interpretation of the presence of models exhibiting "unstable" steady states or "unstable" oscillations in Table 6.1 requires some care, as the analysis of the eigenvalues of a system is only strictly applicable at the steady state, especially in a system with nonlinear topology [132]. Thus although a system with a positive real eigenvalue will *instantaneously* depart exponentially from the steady state, as the system moves further from the steady state, the trajectory will be first influenced, and then dominated by the non-linear factors in the system. Consequently the safest statement to make about the unstable models in Table 6.1 is that if perturbed they will move to some other state, which may or may not be viable, and cannot be determined without further investigation. Thus, the presence of such individuals can be regarded as evidence, but not proof, that the model can undergo switching behaviour.

Bearing these points in mind, it would thus far appear that models c_2 and c_3 give an equally good imitation of the behaviour of the detailed model. There is however another approach that may allow the two to be distinguished. It was shown in chapter 4 that the complete model of the Calvin cycle could posses two stable steady states, and that one means of characterising them was the response to cytoplasmic TP demand, and in particular the response of the rate of starch synthesis: in the fast steady state $C_{TPT}^{J_{StSynth}}$ was negative, and in the slow, positive. Therefore if the distribution of $C_{TPT}^{J_{StSynth}}$ of one of the models is bi-modal with one negative and one positive peak this can be taken as

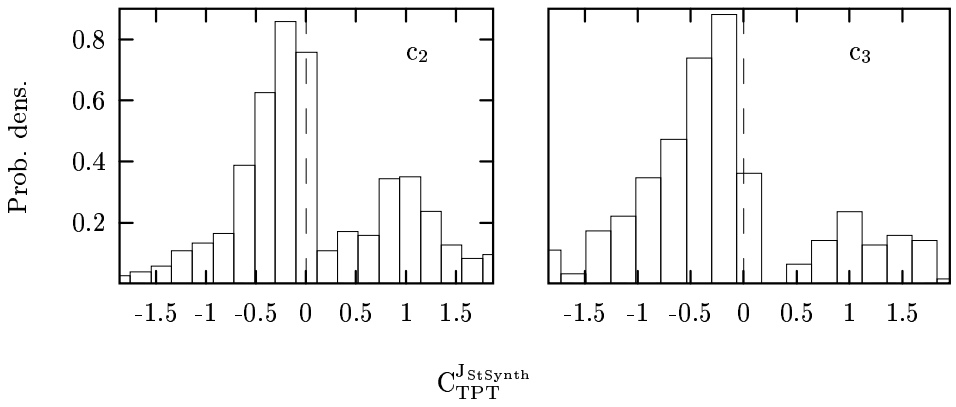

Figure 6.2: Distribution of $C_{TPT}^{J_{StSynth}}$ in viable populations of models c_2 and c_3.

evidence of of the potential for the model to switch. These distributions are presented in Figure 6.2, and it is seen that both models fulfil this criterion. Although it is perhaps arguable that c_3 is "more strongly" bi-modal than c_2, it is hard to avoid the conclusion that the qualitative behaviours of both models equally well represent the behaviour of the complete model, and therefore the simpler of the two, c_2 should be subject to further analysis.

Switching behaviour in skeleton model c_2

That the bi-modal distribution of $C_{TPT}^{J_{StSynth}}$ in Figure 6.2 does reflect the potential for the model to switch is confirmed by two further investigations. Firstly, it will be recalled from Chapter 4, that reducing $P_{i_{ext}}$ to very low levels induced a transition from fast to slow steady-states, and that a characteristic of the slow steady-state is a relatively high concentration of PGA, the immediate product of CO_2 assimilation (Figures 4.2 and 4.3).

Although neither PGA nor $P_{i_{ext}}$ are explicitly represented in these skeleton models, their equivalents may be considered to be TP (as the product of assimilation), and TPT activity (simultaneously modulating TP export and P_i import) respectively. Figure 6.3 shows the response of TP concentration to TPT activity in model c_2. This is qualitatively equivalent to the curve in Figure4.3: At low TPT activity TP is high, reducing with increasing TPT until an abrupt transition is reached, beyond which TP concentration is much reduced. That this represents a true switch, and not simply extreme sensitivity to TPT, may be confirmed by determining steady-state conditions from a

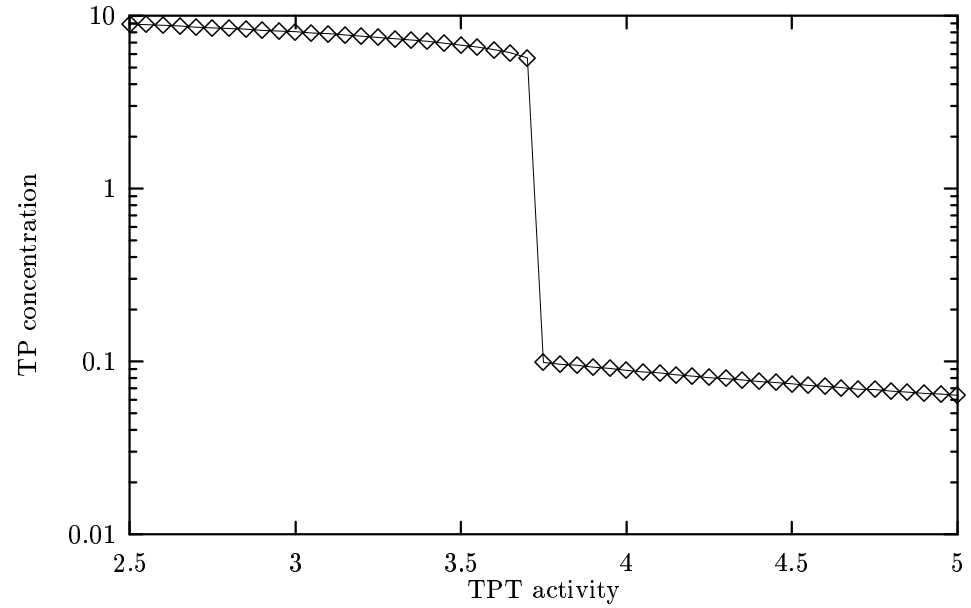

Figure 6.3: Response of TP concentration in model c_2 to TPT activity.

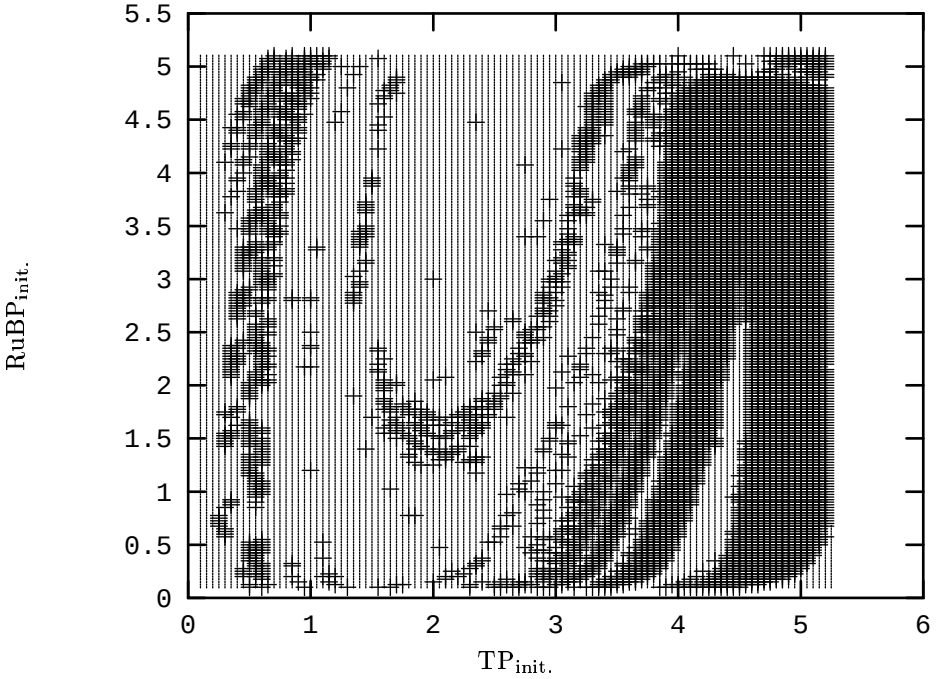

Figure 6.4: Effect of initial TP and RuBP concentrations on steady-state TP concentration in model c_2: (.) low TP ($\sim$ 0.1), (+) high TP ($\sim$ 6).

range of initial concentrations. Figure 6.4 shows the results of simulating to steady-state from a range of initial values of TP and RuBP, and demonstrates unequivocally that two separate steady-states do indeed exist, at least in some regions of parameter space for this model.

Although switching behaviour in model c−2 is established empirically, algebraic analysis proved intractable due to the presence of the non-linear terms in the rate equations. To overcome this problem, the rate equations were re-written as linear equations, first order with respect to each substrate, and with no product terms (other than P_i in the reversible starch synthesis rate equation). With this degree of simplification, a complete solution to the system is possible (see appendix C). The solution is quadratic, but of such a form that only one root results in all concentrations being positive. Thus, this form of the model cannot exhibit the switch between two (physiologically) valid steady-states. It is therefore possible to conclude that the observed switching behaviour is due to the topology of model c_2, in combination with the presence of saturable elements in the rate equations.

6.2 Comparison with other model studies of the Calvin cycle

Of the many previously published models of the Calvin cycle (e.g. [51, 50, 77, 75, 91, 108, 138]) the model presented in this thesis, is thought to be the most complete, both in the sense of using a minimum number of simplifying assumptions (with the possible exception of [75]), and in the variety of behaviour that it has been possible to investigate.

6.2.1 Hahn's model

The earliest model of photosynthate metabolism cited here is that described by Hahn [51,52]. Hahn's model encompasses a larger area of plant metabolism, including cytosolic anabolic reactions from TP up to sucrose synthesis, and was later extended to include the reactions of photorespiration [53]. In common with the model presented in this thesis, starch synthesis and degradation were included as separate reactions.

Hahn makes an original simplification to the regenerative limb of the Calvin cycle by assuming the existence of a free, two carbon intermediate, thiamine pyrophosphate gycolaldehyde (TPGA). Thiamine pyrophosphate is known to be a cofactor for both

aldol and keto transferases, forming respectively a two and three carbon, enzyme-bound intermediate [130]. By treating TPGA generated by the erythrose-aldolase reaction as a free metabolite and further assuming that the two transketolase reactions are in fact (irreversible) transaldolase reactions:

$$\text{S7P} \longrightarrow \text{Ru5P} + \text{TPGA} \tag{6.1}$$

and

$$\text{TP} + \text{TPGA} \longrightarrow \text{Ru5P} \tag{6.2}$$

Hahn greatly simplifies the structure of the regenerative limb of the Calvin cycle .

In addition to this structural simplification, Hahn makes the unusual assumption (as far as detailed biochemical models are concerned) that all reactions have simple first or second order kinetics with respect to substrate, (i.e. no saturation or effector terms), and furthermore, that all reactions are irreversible, with the exception of FBPase which is made reversible under dark, but not light conditions.

Despite the title given to [52], Hahn concentrates his attentions on the dynamic behaviour of the model, and reports damped oscillations of comparable (although rather greater) period and damping times than those reported experimentally (e.g. [134]). Interestingly, Hahn reports four complex conjugate pairs of eigenvalues in his model, which consequently demonstrates quite complex trajectories under certain conditions. This in turn could be taken as a basis for the hypothesis that there is more than one source for the photosynthetic oscillations observed *in vivo*. Such a hypothesis is supported, and implied by Ryde-Pettersson [110, 109], who in an analysis of a model similar to that of Hahn (but omitting photorespiration), describes no less than twenty pairs of metabolites whose interaction could lead to damped oscillation.

Hahn subjected his model to light-dark-light transitions by setting the ATP synthase rate constant to zero to simulate dark conditions, and, in contrast to the results of chapter 4 found it able to metabolise starch in the dark phase and recover the previous light phase behaviour in the absence of the OPPP reactions. This difference in behaviour is due to the simplifications described above, primarily the assumption that the reaction catalysed by FBPase becomes reversible in the dark. Under these conditions a stoichiometrically correct exchange of TP with $P_{i_{ext}}$ through the TPT becomes feasible, utilising starch as the carbon source. Furthermore, because regenerative limb reactions

are assumed to be irreversible, intermediate metabolites do not drain out of the system as was the case in chapter 4. Even if Hahn had made these reactions reversible, the introduction of TPGA, described above, would probably allow recovery from dark conditions, because the simplified stoichiometry of the regenerative limb allows generation of pentose phosphate from hexose phosphate, even if other intermediates are absent.

The major objection to this scheme is, of course, that the phosphorylation of F6P is energetically extremely unfavourable, and is a text book example of a reaction that must be coupled to ATP hydrolysis to proceed, and is catalysed by the enzyme phosphofructokinase (PFK). Stromal isoforms of PFK have not been reported experimentally, but a more serious objection is that Hahn's model does not provide the necessary ATP source under dark conditions, and hence violates the laws of conservation of energy. Therefore, although the model may mimic some of the behaviour observed in experimental systems, the similarity is little more than coincidental, and is unable explain the *in vivo* observations.

6.2.2 The model of Laisk *et al*

Laisk *et al* [77,75,78] present three versions of a model of photosynthesis, varying in the relative detail with which the light reactions, Calvin cycle , and cytosolic reactions are described. In all three cases the regenerative limb of the Calvin cycle is considerably simplified, but in contrast to Hahn (above), reactions are represented with reversible and saturable kinetic equations.

Despite simplifications, the structure of the model described in [75] is very similar to that used here. Although equilibrium approximations are used to simplify the regenerative limb of the Calvin cycle , this is achieved by grouping equilibrated metabolites into *independent* pools, in contrast with some other workers (see below) who group all metabolites connected by reversible reactions into a single pool. The pools specified were triose phosphate (TP), pentosemonophosphate (PMP), hexosemonophosphate (HMP), and mono and bisphosphoglycerate (NPGA). They did not assume (in contrast with others, see below) that NPGA equilibrates with TP. As demonstrated in chapters 4 and 5 BPGA does not appear to equilibrate with GAP. Furthermore, although the model used in this thesis shows that pairs of metabolites within the groups specified by Laisk *et al* maintain highly linear relationships over a wide range of conditions, pairs from different groups do not.

Although Laisk *et al* were primarily interested in aspects of cytosolic metabolism, some of the results in [75] are directly comparable to those in chapter 4, specifically the response of the Calvin cycle to alterations in $P_{i_{ext}}$ and light. Qualitatively their results are very similar: ATP and P_i both increase monotonically with increasing $P_{i_{ext}}$. In contrast to chapter 4, CO_2 assimilation and starch synthesis rates are also seen to increase with $P_{i_{ext}}$, possibly suggesting that their model is in the slow steady state. Laisk *et al* also assumed much higher concentrations of $P_{i_{ext}}$ than here (6–20 mM). However, they also included cytoplasmic TP and PGA in the model, and one might suppose that as this decreases the thermodynamic gradient across TPT, higher levels of $P_{i_{ext}}$ would be needed to in order to obtain comparable effects. This supposition has been tested and vindicated in a model study by Pettersson [92], and thus the much larger values used for $P_{i_{ext}}$ do not amount to a serious incompatibility between the two models.

As in chapter 4, the changing light levels were simulated by modulating the V_{max} value of ATP synthase. It was observed, as in chapter 4 that ATP, TP and P_i increase, and PGA decreases monotonically with increasing light. Assimilation flux is seen to be quite strongly dependent upon ATP synthase V_{max} and from this, an approximate value of $C^{J_{Assim.}}_{ATPSynth}$ of 0.86 can be calculated, comparing with the value of ~ 0.5 obtained in chapter 4 for the slow steady state.

Laisk *et al* also subjected their model to a light-dark transition, with very similar results to those in chapter 4 in the absence of the OPPP reactions: the assimilation flux, ATP, RuBP and TP all drop rapidly, with all but TP approaching zero. The behaviour of TP and PGA in Laisk's model is different from that in chapter 4 in that TP approaches a small but non zero value, and PGA increases to a point where most of the conserved P_i is in this form. In chapter 4 both fall to zero.

The reason for this difference seems to be due to the cytosolic components included by Laisk *et al.* Sucrose synthesis is included as a reversible reaction that is the only reaction capable of generating or consuming sucrose, but which appears none the less to have been included in the model as a free variable. Under dark conditions, the structure of the model appears to allow sucrose degradation with concomitant equilibration of metabolites at least between sucrose and stromal TP. As reactions normally considered irreversible were modeled (rather more realistically) as reversible reactions with large (forward) equilibrium constants, it would also seem likely that the other intermediates

approached equilibrium values, rather than absolute zero. However Laisk *et al* do not publish the results necessary to determine this.

Unfortunately Laisk *et al* do not report results of the complementary dark-light transition. It would be interesting to know whether or not the low, equilibrated concentrations of intermediates presumably present in the regenerative limb of the Calvin cycle are sufficient to allow normal light activity to resume when light is restored, despite the absence of the OPPP.

Another feature of this model was that, despite the best efforts of the investigators, none of the models were observed to oscillate unless an explicit time dependent step was included [77]. Given the propensity of the model in chapters 4 and 5 to oscillate, and that (as has been shown in section 6.1) models of similar, but greatly simplified, topology also oscillate with ease, this is rather surprising. However there is no obvious no explanation for this.

6.2.3 The model of Woodrow

Woodrow [138] presents a model in many ways similar to, but simpler than, that of Pettersson (see chapter 4 and below). Woodrow's model includes the Calvin cycle reactions of assimilation, reduction, regeneration and storage, along with cytosolic sucrose synthesis, represented as a single reaction with cytosolic FBP as a precursor, and sucrose and cytosolic P_i as products. The model does not include light reactions: ATP/ADP and NADPH/NADP are fixed as parameters. Woodrow uses the approximation that substrates and products of fast reactions can be treated as though at equilibrium with some enthusiasm. Not only are all stromal intermediates (with the exception of RuBP) assumed to be related solely in terms of equilibrium constants, but equilibration of TP and P_i across the chloroplast membrane is also assumed, thus removing any influence that may be exerted by either TPT and $P_{i_{ext}}$. As these factors have been shown, by experimental [28, 36, 93] and modelling studies ([75, 91, 92] and chapter 4), to have a considerable impact on the behaviour of the Calvin cycle, results from this model must be treated with caution.

Woodrow's main interest was the determination of flux control coefficients, and the influence of rubisco upon them; to this end used the model to determine values $C^{J_{Assim.}}$ over a range of rubisco and FBPase activities. The curve of rubisco versus $C^{J_{Assim.}}$ bears some qualitative resemblance to that shown previously (chapter 4, figure 6.11), inasmuch

as $C^{J_{Assim.}}_{rubisco}$ was ~ 1.0 at lower values, declining sigmoidally (and more gradually than in figure 6.11), with other enzymes gaining control in a complementary fashion. The major difference is that at high rubisco activity, control is taken up more or less equally by SBPase and Ru5Pk, instead of SBPase alone as seen in figure 6.11. The reason for the difference is probably that the value for Ru5Pk activity (as a proportion of rubisco activity) used by Woodrow was much lower than that used in earlier chapters (~ 1.1 compared with ~ 30 in chapter 4 and an average value ~ 12 in the evolved population of chapter 5).

In addition to the effects of rubisco, Woodrow also determined $C^{J_{Assim.}}$ over a range of FBPase activity values. As FBPase activity was varied between 0.75 and 1.5 times its initial value quite a large exchange of control between FBPase and rubisco was observed. The effects of such changes on the model of chapter 4 were negligible (results not shown).

6.2.4 The Petterssons' model

The model described by Pettersson and Ryde-Pettersson [91] formed the basis of the model used in this thesis and is introduced in chapter 4. The structure of the model is very similar to that described by Woodrow (above), except that the reactions of TPT and ATP regeneration are included, and, like Woodrow's, depend heavily on the assumptions that all fast reactions reach equilibrium.

The approach used to calculate steady state values was to define a system of equations consisting of equilibrium relationships, conservation relationships, kinetic equations of the slow steps, and a set of equations relating the (steady-state) velocities across the slow steps to the assimilation flux. The resulting set of 29 equations were solved simultaneously using "standard algorithms" (which algorithms were not stated). A possible concern with the approach is that although it is clear that the equation set used *will* hold true under steady state conditions, it is not clear from [91] that it will *only* hold true under these circumstances. Although ODEs for individual metabolites were defined, these were not used in the Petterssons' solution of the model.

As noted in chapter 4 there were numerous qualitative and quantitative differences between the results generated by Pettersson's model, and the version used in this thesis. Given that the two had ostensibly identical structure (i.e. topology, rate equations and parameters) and differed only in implementation issues (simplifying assumptions, algorithm for steady state solution etc.) it is clear that at least one of the two sets of results

Table 6.2: Comparison of d/dt of metabolites, and ρ of fast reactions in the Calvin cycle model, as calculated A: from concentrations determined by Pettersson [91] and B: as described in chapter 4. Figures rounded to 3 s.f. but all $\rho < 1.0$ for B

d/dt (μmol.h^{-1}.(mg Chl)$^{-1}$)			Disequilibrium Ratio, ρ		
Metabolite	A	B	Reaction	A	B
RuBP	7.41×10^{2}	-3.19×10^{-10}	PGk	1.54	1.00
ADP	-6.25×10^{7}	1.56×10^{-12}	G3Pdh	2.80×10^{-1}	1.11×10^{-3}
BPGA	-6.25×10^{7}	-1.98×10^{-12}	TPI	1.23	1.00
GAP	4.51×10^{6}	4.43×10^{-13}	F-Aldol	1.25	1.00
DHAP	-1.20×10^{6}	9.74×10^{-12}	PGI	9.97×10^{-1}	1.00
FBP	-3.40×10^{5}	2.36×10^{-12}	PGM	9.95×10^{-1}	1.00
F6P	9.85×10^{5}	-1.25×10^{-9}	F-TKL	1.40	1.00
G6P	-6.54×10^{6}	-2.88×10^{-9}	E-Aldol	9.26×10^{-1}	1.00
PGA	6.25×10^{7}	8.49×10^{-10}	S-TKL	1.28	9.94×10^{-1}
E4P	-3.12×10^{6}	-7.31×10^{-10}	R5Piso	8.33×10^{-1}	1.00
SBP	4.00×10^{5}	-2.94×10^{-9}	X5Pepi	7.46×10^{-1}	1.00
S7P	3.12×10^{5}	2.43×10^{-12}			
R5P	-5.31×10^{6}	4.50×10^{-12}			
Ru5P	1.01×10^{7}	1.80×10^{-12}			
X5P	-8.11×10^{6}	2.69×10^{-12}			
G1P	8.28×10^{6}	-1.67×10^{-10}			

does not accurately reflect the true behaviour of the model. Using the approach to implementation (of a specified model) taken by Pettersson, there are three distinct areas that could give rise to error: incorrect solution of the intermediate equations, incorrect derivation of intermediate equations, and invalid modelling assumptions. Problems arising from the first two of these causes could (but not definitely) be identified by the presence of internal inconsistencies in the results generated, but the third can only be identified by comparing results with those of other implementations of the same model. Examination of the relevant equations in [91] reveals no obvious problem, thus eliminating the second source of error. Table 6.2 furnishes results from which an attempt to identify the other two may be made.

The results in table 6.2 were generated by first calculating (using Scampi) instantaneous values of disequilibrium ratios (ρ) of the fast reactions, and values of d/dt for individual metabolites, from the concentrations and parameters reported in [91]. The model (as described in 4 was then brought to steady state, with a tolerance of $\times10^{-6}$ and the same values recalculated.

If a set of concentrations represents true steady state values, then, by definition, d/dt

for all metabolites must equal zero. Inspection of table 6.2 shows that in the case of the concentrations reported in [91] this is far from the case. The reason for the very high values for most metabolites is the large rate constant used for the reversible reactions (see section 4.2), however, duBP$/dt$ cannot be reasonably said to be $\approx$ zero, despite the fact that this metabolite is neither substrate nor product of any of the fast reactions.

Given that Pettersson assumed that fast reactions were at equilibrium, it follows that ρ should be equal to 1.0 for all fast reactions, regardless of any other considerations. Examination of table 6.2 shows that this is not the case. Furthermore, for several reactions, $\rho > 1.0$, implying that those reactions are running in the reverse direction. Although this appears to be a major inconsistency the possibility that the calculated ρ values are inaccurate, due to the relatively low precision of the concentrations reported in [91], (which in some cases had only one significant figure) cannot be ruled out. Certainly it has proved possible to move values from $\rho < 1.0$ to $\rho > 1.0$ and *vice versa* by altering concentration values by ± 0.5 s.f. However it is not certain that a set of concentrations within ± 0.5 s.f. of those in [91] which simultaneously bring all ρ to 1.0, can be found.

In contrast the results generated by the model in chapter 4 show no such problems. The values of d/dt are consistent with the model having achieved steady state within the requested tolerance. Although no assumption was made that substrates and products of fast reactions would approach equilibrium, all but one do, and for all reaction ρ remains less than 1.0. Although this does not prove an absence of error in the implementation of the model in this thesis, it is clear that this model is much more consistent than Petterssons. On the basis of these comparisons it is possible to suggest that a problem exists with the method used by Pettersson to solve the model. This in turn could be because solutions of the intermediate equations do not uniquely represent steady state conditions, or that there is a programming error in the implementation of the determination of such solutions.

It is also possible to conclude, with rather more certainty, that the assumption that fast reactions achieve equilibrium at steady state is not justified in this model, as despite maintaining internal consistency ρPGK is displaced from 1.0, as described in chapters 4 and 5.

Despite the differences, there remain nonetheless, some similarities between the two models: the response of input and output flux to $P_{i_{ext}}$ are qualitatively the same, as is the response of $C^{J_{Assim.}}$ to $P_{i_{ext}}$, at least at higher values. In particular Pettersson

reports very low values of $C_{rubisco}^{J_{Assim.}}$ for all values of $P_{i_{ext}}$, and large negative values for $C_{TPT}^{J_{Assim.}}$ ($P_{i_{ext}} > 0.15$ mM). The control exerted by SBPase is large and positive ($P_{i_{ext}} > 0.15$ mM), and shares this control with the ATP regenerating steps.

Pettersson also reports that the model can exhibit two possible steady state solutions, but reports that the slower (with regard to carbon assimilation) was dynamically unstable, and that following a small ($\pm 1\%$) perturbation of RuBP the system either returned to the fast steady state, or collapsed. Consequently Pettersson dismissed the slow steady state as being of no relevance. It will be recalled from chapter 4 that it appeared possible to induce switching from the fast to the slow steady state by reducing $P_{i_{ext}}$ to very low levels (section 4.3.1 figure 4.2), and that the signs of $C_{TPT}^{J_{Assim.}}$ and $C_{SBPase}^{J_{Assim.}}$ change at the transition. Interestingly Petterssons results show just such a change occurring at a value of $P_{i_{ext}}$ somewhere between 0.05 and 0.15 mM, although the authors do not comment on it. Surprisingly, despite examining simulated time course data, and the eigenvalues of the Jacobian of the system, Pettersson does not report any oscillatory behaviour.

6.2.5 The models of Giersch

Giersch and others [44,43,108] have reported on the behaviour of two skeleton models in many ways similar to, but with greater analytic success than, those described in section 6.1. The object was to investigate two hypotheses explaining oscillation in the Calvin cycle .

The first of these [44] simplifies the Calvin cycle to three reactions, roughly equivalent to Ru5Pk and PGK both driven by ATP hydrolysis, and ATP synthesis. The model has 5 intermediates: ATP, ADP, P_i, and two lumped metabolites, roughly equivalent to Ru5P and PGA respectively. Although P_i is imported (at a constant rate) there is no concomitant export of TP, with the result that total P_i accumulates. The problem is circumvented by re-arranging the model equations so as to allow ATP, ADP, P_i and Ru5P to reach static positive steady values, with the apparent corollary that PGA (to which none of the reactions are sensitive) accumulates. Giersch points out that this problem does not alter the hypothesis that oscillation is the result of the interdependence between sugar-phosphate and adenylate turnovers. When simulated, the model did produce damped oscillations of appropriate period and damping time, although this was quite sensitive to values of individual parameters: changes in parameter values of

more than 10% resulted in unrealistic behaviour. Damping time was also shown to be sensitive to external P_i, decreasing the rate of P_i import resulted in oscillations of increasing amplitude. Furthermore, parameter changes that led to decreased free P_i also resulted in increased damping times, and *vice versa*.

The second model is very similar to the first, but neglects P_i entirely, and includes more detailed modelling of the reductive limb of the cycle and the light reactions (including the oxidation and reduction of NADP/H as well interconversion of ADP/ATP). The model has the advantage that it is capable of attaining stable steady states, although, as with the previous model, there is no carbon throughput. The behaviour of the model with a number of reasonable simple rate laws for the NADPH and ATP synthesis was investigated, and it was concluded that realistic oscillatory behaviour could only be obtained if it is assumed that the rate of photophosphorylation depends on NADP concentrations as well as ADP, but NADP reduction is dependent only upon the concentration of its substrate.

Taken together the two models may appear to pose more problems than they solve. Firstly, as Giersch points out [44,43], different criteria for simplifying a complex system can lead to skeleton models that are very similar, making it difficult, if not impossible to decide which provides a more satisfactory description of the reality. It would appear reasonable to suggest that this problem in turn is not the result of an inadequacy of the modelling process, but stems from the nature of complex systems. The possibility appears to exist that there is no single cause of phenomena such as oscillation in the Calvin cycle, instead, several "causes" may operate simultaneously.

Secondly assumptions made in the construction of such models may result in fundamental departures from reality. The accumulation of P_i in one of the models has already been described. In the second model Giersch is able to obtain algebraic expressions for flux control coefficients of the (the rate constants of) various reactions, and one of these has $C^J = 0$ which anomalously implies that were the rate constant of this reaction to be set to zero (i.e. the reaction is abolished) it would still carry a flux.

This is not to say that such modelling endeavours are without value. They represent the only method (known to the author) of establishing a link between structure and behaviour of systems. However, the simplifications necessary to obtain a solution mean that the solution represents a substantial departure from reality, and hence conclusions drawn must be treated with some caution.

6.3 Comparison of model behaviour with experimental observation

6.3.1 Metabolite concentrations

Metabolite concentrations observed in the Calvin cycle model of previous chapters are presented in Table 6.3 along with experimental observations. Most of these model results can be regarded as satisfactorily close to observed values. The largest discrepancies between model and experimental values are in the values of P_i and the ATP:ADP ratio. It can be seen that experimental measurement of other stromal metabolites cover quite a large range, if the one experimental report of P_i lies at the high end of a range, then the value determined in the model may not be too unrealistic. A possible explanation for the high ATP:ADP ratio is the relatively low value used for total conserved adenosine of 0.5mM, (Giersch [44,43] uses values of 1.7.and 2.4 mM respectively), it is possible that the light reactions, as modelled, would be unable to maintain such a high ratio if the total pool is greater. Furthermore, not only is the chloroplast stroma the site of a great deal of anabolic activity, but potential shunt mechanisms operate to effectively export ATP from the stroma [28], so it is reasonable to assume that there are other significant sinks which act to lower the *in vivo* ATP:ADP ratio.

6.3.2 Response to environmental factors - 1 - Light and light-dark transitions

The most surprising response in the Calvin cycle model was that of assimilation towards light. Text-book [81,113] reporting of the response of photosynthetic carbon assimilation to light shows saturating response curves, further modifiable by environment. Assuming that the fast steady state (see chapter 4) is the more realistic, the complete insensitivity of the assimilation rate toward light in this steady state appears grossly at variance with experimental observation. An obvious possible explanation to the discrepancy, is that because the light reactions are described in much less detail (section 4.2) than the rest of the model, it is to be expected that the response to light will not be particularly realistic. An alternative explanation may lie in the interpretation of experimental data and its comparison with the model results. Although in the model the assimilation rate does not vary with light, most variables do, in particular TP export does increase (in

Table 6.3: Comparison of modelled and experimentally observed stromal Calvin cycle metabolite concentrations in chloroplast suspensions (†) and whole leaf material(∗). In the model and chloroplast suspension results $P_{i_{ext}} = 0.5$ mM but is unreported in the whole leaf data (the large pool of P_i in vacuole makes measurement of cytosolic P_i impossible). All concentrations are in mM.

	Modelled			Observed					
Metabolite	Original	Evolved	[91]	[42]*	[123]†	[37]†	[104]*	[107]†	[80]
DHAP	2.68×10^{-1}	2.21×10^{-1}	2.70×10^{-1}	5.33×10^{-1}	3.70×10^{-1}	-	-	-	-
Total TP	2.80×10^{-1}	2.31×10^{-1}	2.80×10^{-1}	-	-	1.40×10^{-1}	1.20×10^{-1}	1.50	0.3–0.4
PGA	1.26	4.28	5.90×10^{-1}	8.93	5.70	4.44	4.80×10^{-1}	8.47	3–5
Total PMP	8.12×10^{-3}	1.41×10^{-1}	2.40×10^{-1}	-	1.60×10^{-1}	8.00×10^{-2}	-	-	0.6
RuBP	4.20×10^{-1}	2.17	1.40×10^{-1}	6.10	6.40×10^{-1}	4.60×10^{-1}	3.60×10^{-1}	1.50	0.2–0.6
G6P	3.74	2.79	3.12	1.13	-	-	3.00×10^{-1}	5.70	-
F6P	1.62	1.21	1.36	1.60	-	-	1.80×10^{-1}	-	0.6–1.5
FBP	2.32×10^{-2}	1.57×10^{-2}	2.40×10^{-2}	1.17	5.20×10^{-1}	2.90×10^{-1}	2.50	-	0.1–0.3
S7P	9.58×10^{-4}	3.53×10^{-1}	2.20×10^{-1}	-	-	-	-	-	-
SBP	2.39	7.02×10^{-2}	1.30×10^{-1}	-	2.40×10^{-1}	1.10×10^{-1}	-	-	0.2–1
ADP	2.22×10^{-3}	8.78×10^{-3}	1.10×10^{-1}	-	-	-	6.00×10^{-2}	-	-
ATP	4.97×10^{-1}	4.91×10^{-1}	3.90×10^{-1}	-	-	-	3.60×10^{-1}	-	-
Pi	8.49×10^{-1}	5.40×10^{-1}	8.10	-	4.46	-	-	-	-

the fast steady state) in response to increased light as shown in figure 4.11(B). Thus if the experimental measurements are not of CO_2 assimilation *per se*, but of the rate of accumulation of radiolabeled cytosolic photosynthate, the model and experimental results can be reconciled.

The presence of two steady-states, although reported very rarely in natural systems, has, according to Laisk and Walker [77] (citing a Russian publication [76]), been described in the response of carbon assimilation to light, in lilac leaves. The model showed that this switching behaviour can be modified, or even abolished by varying the capacity of the TPT to transport PGA (section 4.5) and presumably by other parameters. It thus seems likely that the possibility for switching is inherent in the Calvin cycle, but whether or not the possibility is realised depends upon precise environmental conditions. On balance, one would expect the absence of switching to be the norm, for two reasons. Physiologically it is a disadvantage for the organism to have two steady-states for the same set of environmental conditions. If two states exist, then one must be less favourable (if only marginally) than the other, and a selection pressure will exist to eliminate the less favourable. Secondly, as described below, the switching behaviour in the model can be abolished if the model is made more realistic.

The influence of the thioredoxin system

An important mechanism responsible for coordinating changes in Calvin cycle enzyme activity in response to changing light is the thioredoxin system [9, 28]. Thioredoxin is a small mobile protein capable of transporting electrons (originating from ferredoxin in the electron transport apparatus of the light reactions), and thereby reducing disulphide groups in enzymes (with concomitant oxidation of thioredoxin) to break disulphide bridges, inducing a conformational change in the enzyme, and hence alteration in activity. The Calvin cycle enzymes activated by this mechanism are FBPase, SBPase, Ru5Pk, PGK, and probably (via a third protein, rubisco activase) rubisco.

The thioredoxin system has been implemented in the Calvin cycle model by the introduction of a new parameter, included in the rate equations of FBPase, SBPase, Ru5Pk, PGK, rubisco and the light reactions, as a coefficient of the respective V_m parameters. The response of assimilation towards this parameter is almost perfectly linear (not shown), and no evidence of switching is seen. Although this response is just as unrealistic as the response in the absence of thioredoxin, the model representation

of thioredoxin is obviously crude. In reality one would expect that the proportion of reduced thioredoxin as a function of incident light is saturable, and if, as the modeling result suggests, the response of Calvin cycle assimilation to reduced thioredoxin is linear, then the response to light will be saturable, as observed experimentally.

Consideration of the role of thioredoxin also corrects another departure of the model's light response from reality: that of light-dark transitions. In section 4.4 it was shown that the total carbon export (under light conditions) could exceed the assimilation rate, the deficit being made up by the degradation of starch. At first sight it appeared reasonable therefore that the presence of starch phosphorylase would allow a carbon flux from starch to cytosol to be maintained in the absence of assimilation, and therefore in the absence of light (reaction activity). However, when light was reduced to zero the model rapidly collapsed, with all fluxes, and all intermediate concentrations with the exception of hexose phosphate falling to zero, even with the inclusion of the thioredoxin mechanism as described above. Furthermore the collapse was irreversible: once intermediates had fallen to zero the model did not recover when light was restored.

To determine whether this collapse is due to a poor choice of parameter values, or whether the collapse is inevitable regardless of parameter values, is extremely difficult using a kinetic model alone. Use of the evolution strategy algorithm strategy (chapter 3) to search for appropriate parameters values would be possible, but this would not conclusively reveal the absence of a suitable parameter set, should this be the case. However it is problem ideally suited to the technique of elementary modes analysis (described in chapter 1), which depends on knowledge of system topology alone. This revealed that no route exists through the Calvin cycle from starch to TP that does not also involve the phosphorylating reactions, and is thus dependent on light reaction activity, and hence no set of parameter values can exist that will allow the Calvin cycle as previously described to support a carbon flux under dark conditions. It is therefore clear that additional reactions must be involved *in vivo* to allow photosynthetic organisms to survive exposure to dark.

The irreversible nature of the collapse of the model is due to the activity of starch phosphorylase, which under dark conditions causes all free P_i to be sequestered in the form of hexose phosphate. When light conditions are restored there is thus no P_i available for the regeneration of ATP and the cycle cannot restart. The situation is not improved *in vivo* if it is assumed (reasonably) that TPT is reversible. It will still not

be possible to "kick start" the Calvin cycle from cytosolic TP because there will be no stromal P_i available for exchange.

Inclusion of the oxidative pentose phosphate path.

The problem is overcome when it is recalled that in the dark the thioredoxin system not only inhibits the Calvin cycle enzymes previously described, but activates glucose-6-phosphate dehydrogenase (G6Pdh) and transaldolase, both components of the oxidative pentose phosphate pathway (OPPP), known to be present [124] in chloroplast stroma. This branched, acyclic pathway is considered to have G6P as its initial substrate, PGA as its final product [114, 85], and has two limbs: the first three enzymes catalyse the overall reaction

$$\text{G6P} \longrightarrow \text{R5P} + \text{CO}_2 \tag{6.3}$$

and are termed the oxidative limb, whilst the remainder, which are the regenerative Calvin cycle enzymes TKL, X5Pepi, X5Piso plus transaldolase, are the reversible limb.

It was conjectured that the inclusion of the oxidative limb of the OPPP in the Calvin cycle model might be sufficient to maintain the concentrations of intermediates under dark conditions, thus making light-dark transitions recoverable. When the overall reaction for the oxidative limb[2] was included in the model it was found that not only could the model recover from periods of darkness but a small ($\sim 10^{-4}$ of light value) TPT flux was maintained. While it is unlikely that such a flux could make a useful contribution to cytosolic metabolism, it is sufficient to maintain an adequate concentration of stromal P_i to restart ATP synthesis when light is restored. If the transaldolase reaction is also added, then not only can the model recover from the dark, but it is able to sustain a TPT flux of the same order of magnitude as that in the light, with a calculated rate of starch degradation of 10 μmol.h^{-1}.(mg Chl)$^{-1}$ resulting in TP export of 5 μmol.h^{-1}.(mg Chl)$^{-1}$. The starch degradation figure compares with an experimentally measured value of 12 μmol.h^{-1}.(mg Chl)$^{-1}$reported by Stitt and Heldt [128].

Elementary modes analysis of this new system revealed a quite complex cycle, with starch as the initial carbon source, and G6P being regenerated such that for each turn of the cycle one molecule of G6P enters the oxidative limb from starch and two enter from the regenerative limb (see figure 6.5), half a molecule of G6P is lost as CO_2, and

[2] stoichiometry as equation 6.3, Michaelis-Menten kinetics, inactivated by light, V_m = 40 μmol.h^{-1}.(mg Chl)$^{-1}$(= StPase V_m), K_m= 1 mM

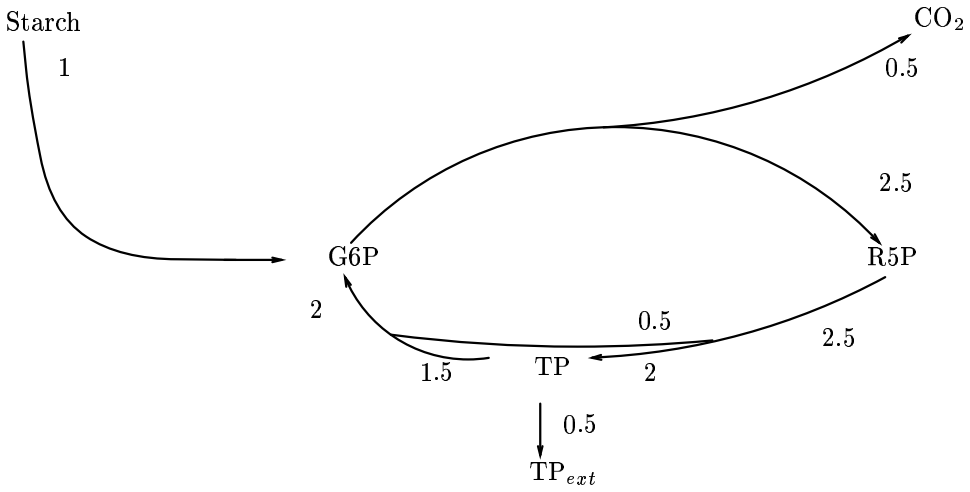

Figure 6.5: Simplified net stoichiometry of carbon flow around OPPP/Calvin cycle under dark conditions, as determined by elementary modes analysis. Figures indicate relative carbon flux under steady state conditions. P_i omitted for clarity.

half exported as TP.

Although this cyclic behaviour in the dark is attractive, there is an additional problem to be solved before it can be proposed as a hypothesis. The oxidative limb of the OPPP reduces NADP to NADPH, so equation 6.3 is more completely represented as:

$$2\text{NADP} + \text{G6P} \longrightarrow 2\text{NADPH} + \text{R5P} + \text{CO}_2 \tag{6.4}$$

resulting in the net reduction of two molecules of NADP for ever one of TP exported to the cytosol.

Although in the model the reduction of NADP by the oxidative limb of the OPPP is not a problem, because NADP(H) is fixed as a parameter, if the cycle as described by Figure 6.5 is feasible *in vivo* then a substantial oxidative consumer of NADPH must exist; there are several possible candidates.

Firstly it seems reasonable to suppose that such sinks do exist. Whether or not the cycle operates as outlined above, the oxidation of G6P to R5P does take place and many anabolic processes depend on the redox potential thus afforded, including nitrogen assimilation, nucleotide and isoprenoid synthesis. The latter is an attractive possibility as this includes synthesis of light harvesting pigments, known to be damaged by light.

A second possibility is that the reducing potential is exported to the cytosol via one or more shuttle mechanisms. Both the oxaloacetate-aspartate (OA) and oxaloacetate-

malate (OM) shuttles are reasonable contenders. Anderson [9] reports that aspartate transaminase is inactivated in the light by the thioredoxin system, thus the OA shuttle may be more promising than OM, as malate dehydrogenase (Mal-dh) is reported to be inactivated in the dark. However, a recent estimate of the (maximal) capacity of the OM as 260 μmol.h^{-1}.(mg Chl)$^{-1}$ [40], compares with the maximal rate of NADPH production by the mechanism proposed here of 240 μmol.h^{-1}.(mg Chl)$^{-1}$ (assuming V_m for StPase and OPPP of 40 μmol.h^{-1}.(mg Chl)$^{-1}$). Furthermore, Anderson also reports that the it is possible for the enzyme to be activated in the dark, at least under anaerobic conditions in maize.

There would appear to be a more radical alternative as it is possible for thioredoxin to be reduced by NADPH [126]. If this is the case then many events will occur. Firstly (in order of relevance, not time), the oxidative limb of the OPPP will become inhibited, forming a simple negative feed-back, reducing the rate of production of NADPH. Secondly, G3Pdh would become active, permitting the operation of the GAP-PGA shuttle to carry redox potential out of the stroma. Thirdly if the other thioredoxin activated Calvin cycle enzymes become active, then a certain amount of CO_2 fixation in the dark will take place. Given that thioredoxin may be reduced by NADPH, it also seems reasonable to propose that the role of the thioredoxin system is not co-ordinating response to light *per se*, but is part of a mechanism maintaining redox poise, which is, of course, greatly affected by light.

The effects of the thioredoxin system upon target enzyme activity do not appear to be absolute, but modulate enzyme activity over a range from a minimum of perhaps 20% of maximal activity [28]. If this is the case, then the scheme for dark metabolism proposed here, and the standard model of the Calvin cycle represent two extremes of a continuum, with cyclic OPPP representing metabolism operating under oxidising conditions (when NADPH:NADP ratio will be low), and Calvin cycle under reducing conditions, with both overlapping under non extreme conditions.

It follows from the foregoing discussion that it is to be expected that abnormal redox poise in the cell will lead to abnormal light assimilation responses. When Elrifi *et al* [29] exposed nitrogen starved *Chlorella* to NH_4^+, the resulting assimilation-light response data much more closely resemble the discontinuous curves shown in chapter 4 than the smooth curve imposed by the authors.

6.3.3 Response to environmental factors - 2 - $P_{i_{ext}}$

There is general consensus [27, 59, 38] that the major, if not the sole route for the export of Calvin cycle and chloroplastic starch metabolism intermediates is via the TPT. The TPT can therefore be considered to be the interface connecting the Calvin cycle to general cytosolic metabolism, thus coupling cytosolic carbon demand with stromal carbon supply, and a considerable body of evidence exists to suggest that this step plays a major rôle in controlling the behaviour of the Calvin cycle in response to changes in concentration of external P_i and TP.

There is also agreement that the export of neutral sugars, arising from the activity of amylase upon chloroplastic starch, is also a significant route by which carbon is exported to the cytosol [12, 128, 133], particularly at night. However, as long as the "concentration" of starch can be regarded as fixed, the two routes are independent from one another. Thus, although the second route is likely to be of physiological importance in the context of overall cellular metabolism, it need not be considered in the following discussion.

The TPT exports TP in strict counter-exchange for P_i [38] and thus effectively catalyses the reversible reaction:

$$TP_{stroma} + P_{i_{ext}} \overset{TPT}{\rightleftharpoons} P_{i\ stroma} + TP_{ext} \tag{6.5}$$

Thus *in vivo* carbon demand is signalled by increased $P_{i_{ext}}$ and/or decreased TP_{ext}. In addition to varying $P_{i_{ext}}$ and TP_{ext} the experimentalist working with isolated chloroplasts can also vary TPT activity by the use of specific inhibitors. The further possibility exists of using GM techniques to alter levels of expression of TPT; this is discussed later.

Flügge *et al* [37], Heldt *et al* [59] and Portis [93] have all investigated the effects of external P_i , TP, and TPT inhibitors upon photosynthesis in isolated chloroplasts. Qualitatively these results are in good agreement with each other, with the Petterssons' model results [91,92], and with the results described in chapter 4. All these results may be summarised as follows :

Except at very low concentrations of $P_{i_{ext}}$, ($<\sim$ 0.2 mM) the rate of carbon assimilation is either unaffected, or *decreases* in response to increased external demand. Rather, the Calvin cycle responds by altering the partitioning of carbon flux between starch synthesis and TP export. If levels of demand are very high, and in the absence of

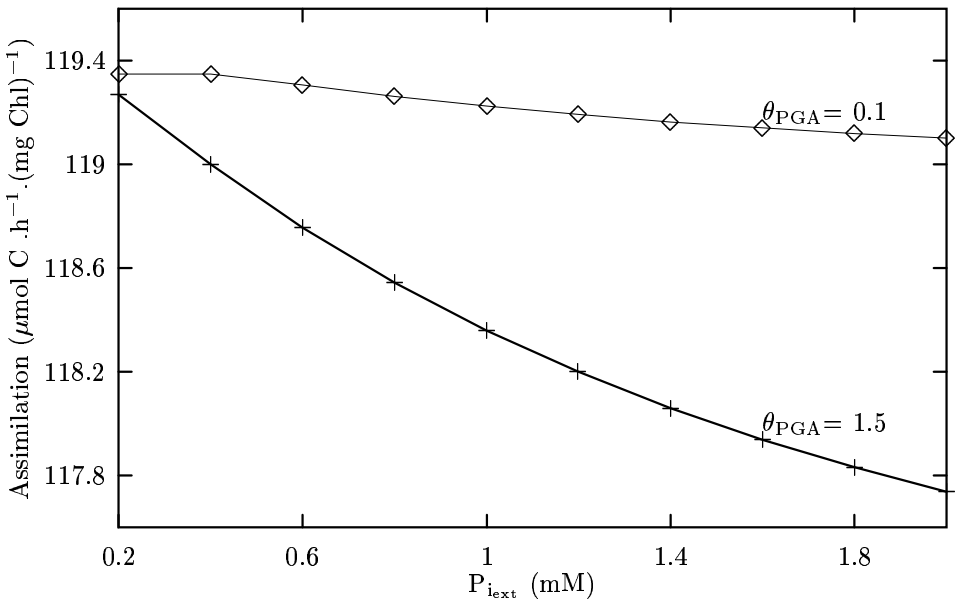

Figure 6.6: Effect of θ_{PGA} on the response of the Calvin cycle model to $P_{i_{ext}}$

(the ability to degrade) starch, the cycle is prone to collapse. At low levels of $P_{i_{ext}}$, the response to increasing $P_{i_{ext}}$ is an increase in assimilation, with concomitant increases both in starch synthesis and TP export.

Although the equation describing the activity of TPT in the model of chapter 4 does not include terms for external metabolites other than P_i, a parameter, θ_{PGA}, was introduced that modulated the sensitivity of TPT toward PGA, thus mimicking, albeit crudely, the effect of external PGA (low values of θ_{PGA} being equivalent to high values of external PGA). Portis [93] investigated the effect of external PGA on the response of photosynthesis to $P_{i_{ext}}$ in isolated chloroplasts. The effect of high external concentrations of PGA in the model is to to cause the assimilation flux to become less sensitive to $P_{i_{ext}}$. This effect can be reproduced in the model of chapter 4, as shown in figure 6.6. However, it should be pointed out that the effect reported by Portis was considerably more pronounced.

Isolated chloroplast suspensions are more similar to the model of chapter 4 than any other experimentally available system. Comparison of model data with the experimental data suggests that the model is sufficient to account for observed responses to changes

in concentrations of external metabolites.

6.3.4 Determination of flux control coefficients by GM

Although interpretation of flux control coefficients determined by GM techniques should be undertaken with a degree of caution (chapter 1), several of the the slow enzymes of the Calvin cycle have been investigated in this fashion, and the results are now briefly described.

Rubisco

Stitt and co-workers [35, 104, 129] investigated the properties of tobacco plants transformed with the antisense gene for the rubisco small sub unit (*rbcS*). The resulting transformants exhibited a rubisco activities ranging from $\sim$ 20%–100% of wild type activity.

Under ambient growth conditions very little impact on photosynthetic flux was observed until rubisco activity fell to $\sim$ 40% wild type (wt). Using the plot of rubisco activity versus assimilation flux Stitt *et al* calculated an approximate value for $C^{J_{Assim.}}_{rubisco}$ of 0.1. This compares with the value of 2×10^{-3} in the model of chapter 4, and the range 0.4–0.8 in the evolved population of chapter 5.

At levels of expression much below 40% wt assimilation flux became linearly dependent on rubisco activity, consistent with a value of $C^{J_{Assim.}}_{rubisco} \sim 1$. However at these lower levels of rubisco activity, a number of other effects were observed, including a reductions of: stromal FBPase activity, chlorophyll content, ATP synthase activity, and a disproportionate decrease in total leaf protein. Thus, although it is reasonable to argue that at high levels of expression *rbcS* antisense mRNA exhibits a high level of control over the photosynthetic activity, it is not possible to be certain that these effects are mediated solely by the changes in rubisco activity.

Photosynthetic assimilation was also recorded in these plants under a range of non-ambient conditions, including high light levels, and saturating CO_2. The latter is particular relevant here, as the levels of CO_2 were high enough to abolish photorespiration. As photorespiration is not included in the model of previous chapters, the structure of the model more closely resembles these plants than those under ambient CO_2 concentrations.

By varying these environmental parameters Stitt *et al* [129] were able to demonstrate

that $C^{J_{Assim.}}_{rubisco}$ can vary considerably. At high light levels, and ambient CO_2 $C^{J_{Assim.}}_{rubisco}$ was estimated as ~ 0.7, increasing to ~ 1.0 at $\sim 20\%$ wt. At high CO_2 and ambient light $C^{J_{Assim.}}_{rubisco}$ remains low (≤ 0.2), again until rubisco has declined to $\sim 30\%$ wt at which point $C^{J_{Assim.}}_{rubisco}$ increases abruptly to a value close to unity. Unfortunately the behaviour of the plants at high light and high CO_2, to which the model would be a still better representation, were not reported.

Ru5P kinase

Gray *et al* [48] used antisense technology to generate tobacco plants with Ru5Pk activities ranging from between 5–100% of wt. No reduction in CO_2 assimilation rates occurred until Ru5Pk activity was reduced to $\sim 15\%$ of wt, at which point chlorophyll concentrations also dropped. The authors also recorded concentrations of ATP, ADP, R5P, Ru5P, RuBP, and PGA. Changes in these concentrations were negligible for Ru5Pk activity $> 50\%$ wt.

In previous chapters C^{J}_{Ru5Pk} has been calculated under a range of model conditions, and has always been small if not negligible. The largest absolute value for C^{J}_{Ru5Pk} was -0.2, over the starch synthesis flux, but other determinations have been much smaller.

On the basis of experimental and modelling evidence it seems reasonable to conclude that, at least under ambient conditions, the modulation of Ru5Pk activity has little rôle in the control of photosynthetic metabolism.

The triose phosphate translocator

Riesmeier *et al* [107] report the behaviour of potato plants in which TPT activity was reduced to between 70 and 100% wt, again by antisense transformation. Plants showed marked growth retardation at four weeks, but there was little difference in the gross phenotypes of mature plants. In particular there was no reduction in tuber yield.

The effect of the transformation upon photosynthetic metabolism was examined in extracted chloroplasts. In contrast to the model studies in earlier chapters, reduction of TPT activity resulted in a decrease in the maximum assimilation rate, suggesting an approximate value for $C^{J_{Assim_{max}}}_{TPT}$ of $>\sim 0.5$, although the wide confidence limits on the data in [107] mean that $C^{J_{Assim_{max}}}_{TPT}$ could be much greater, and that these particular data should be interpreted with some caution. None the less, this result contrasts quite strongly with the very low values for $C^{J_{Assim.}}_{TPT}$ determined in the model.

At ambient conditions Riesmeier's experimental observations become more consistent with the model. Decreased TPT expression resulted in a small (statistically insignificant) increase in CO_2 assimilation, and a large decrease in the rate of starch synthesis, from which a value for $C_{TPT}^{J_{St.synth}}$ of $\sim$ -2.0 can be calculated. This compares with values between $C_{TPT}^{J_{St.synth}} \leq -1.0$ for positive starch flux in the non evolved model, and a range of -0.6 – -1.2 in the evolved population. Riesmeier *et al* did not report TP export flux, however if $C_{TPT}^{J_{Assim.}}$ is zero, and $C_{TPT}^{J_{St.synth}}$ is negative, then it follows from the summation theorem (see chapter 1) that $C_{TPT}^{J_{TPT}}$ must have been positive and of the same absolute value, also consistent with the model results.

It is interesting to note that although the reduced TPT transformants produced considerably more starch during the day in comparison to their wild type counterparts, they were also able to degrade much more rapidly at night. This, coupled with the fact that tuber size was unaffected, is strong evidence that at least one more mechanism, in addition to that proposed in section 6.3.2 must be operative.

Gray *et al* [48] used both sense and antisense transformation on tobacco plants to obtain plants with TPT activity ranging between 20 and 300% wt. Quantitatively the results were much less dramatic than those of Riesmeier. Qualitatively their results are consistent with Riesmeier *et al* and the model in this thesis: $C_{TPT}^{J_{Assim.}}$ is negligible, $C_{TPT}^{J_{TPT}}$ positive and $C_{TPT}^{J_{St.synth}}$ negative.

Sedoheptulose-1,7-bisphosphatase

One of the more unexpected results from the initial modelling work described in chapter 4 was the very high control of assimilation flux exerted by SBPase. Other than acknowledging it as a component of the Calvin cycle , it is an enzyme rarely mentioned in the research literature, and so the domination of assimilation by such an obscure enzyme would appear, at least potentially, to be anomalous.

Recent work by Raines *et al* ([55] and personal communication) provides strong evidence that SBPase does indeed have a high flux control coefficient over CO_2 assimilation. The SBPase gene was antisensed, and introduced into *Nicotiana* to produce plants with levels of SBPase activity ranging from as low as 7%, to 100% of wild type. In common with the antisense work described above, plants with very low levels (in this case $<\sim$ 40% wt) of activity had rather severe phenotypes, typified by low levels of chlorophyll and stunted growth. Much more unusual is the fact that there were

detectable reductions in CO_2 assimilation in plants with a only a modest reduction in SBPase activity.

If, as the model predicts, $C_{SBPase}^{J_{Assim.}}$ really is equal to one, then the plot of assimilation: SBPase will yield a curve of the form $y = m.x + c, m = 3$ (from stoichiometric considerations $J_{\text{Assim.}} = 3.J_{\text{SBPase}}$), $c = 0$. In fact estimates of m and c resulting from fitting a straight line to Raines' data (figure 6.7) appear to differ on both counts, yielding $m = 2.88 \pm 1.55$ (95% confidence limits), $c = 20.2 \pm 12.7$. Although the difference in slopes between the model and experimental data is not significant the value of c is clearly significantly greater than zero ($p < 0.05$).

Metabolic control analysis orthodoxy (e.g. [32]) predicts that, in general, the response curve of flux to enzyme activity will approximate a rectangular hyperbola. Fitting such a function to the data in figure 6.7 results in a sum of squared residuals five times greater than the linear case. Adding an offset to the hyperbolic function (i.e. $f(x) = c + m.x/(k + x)$) made no improvement to the situation (results not shown). It therefore seems reasonable to accept that, in this case, the linear function provides the better description of the experimental data.

Determination of the slope of $\ln(J_{\text{Assim.}})v\ln(\text{SBPase})$ yields $C_{SBPase}^{J_{Assim.}} = 0.55 \pm 0.23$. This compares with the value 1.0 obtained from the initial model and the range of $-8\times10^{-3} - -0.28$ with the evolved parameter set.

The significant offset seen in figure 6.7 is surprising because it implies that, were the organism otherwise viable, an assimilation flux could be sustained with no SBPase activity. The conventional topology of the Calvin cycle as illustrated in chapter 4 (figure 4.1) is such that no assimilation flux can be sustained in the absence of SBPase, and hence Raines' experimental evidence suggests that at least one other reaction is present that acts to "bypass" SBPase.

A good candidate for this extra reaction would appear to be transaldolase as it can utilise E4P as a substrate and produces S7P, which otherwise depend upon SBPase for their consumption and production respectively. Subsequent elementary modes analysis revealed that an assimilation route through the Calvin cycle does indeed exist in the absence of SBPase, if transaldolase is present. Furthermore, although transaldolase, under the influence of the thioredoxin mechanism, is down-regulated in the light [9], its activity is only reduced to $\sim$ 50% of dark activity [8].

Transaldolase was therefore incorporated into the Calvin cycle model with the as-

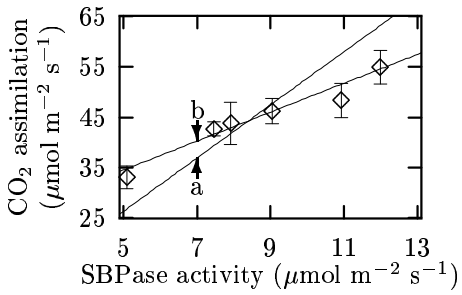

Figure 6.7: CO_2 Assimilation in *N. tabacum* with reduced levels of SBPase (◇) with lines of best fit to (a) $y = m.x$ and (b) $y = m.x + c$. Error bars represent 95% confidence limits. Data from Raines (unpublished)

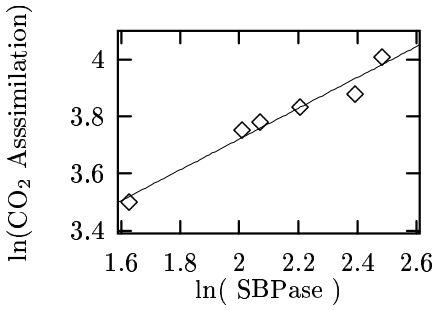

Figure 6.8: Ln-Ln data from 6.7, and line of best fit. The slope of this line, and hence $C_{SBPase}^{J_{Assim}}$, is ≈ 0.5

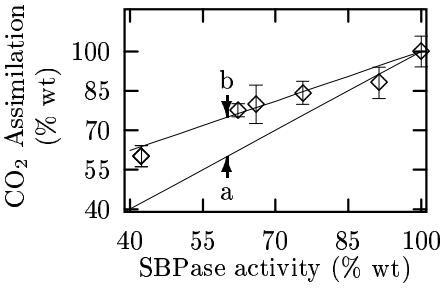

Figure 6.9: Comparison of the experimental data of figures 6.7 and 6.8, and the model of chapter 4 with transaldolase (a) absent, and (b) present.

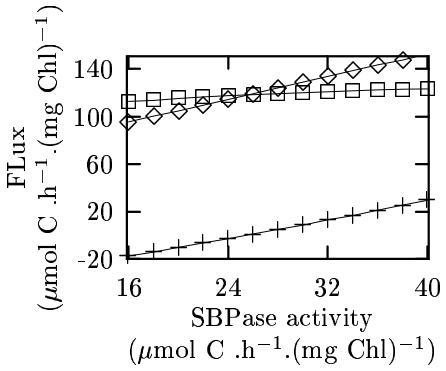

Figure 6.10: Response of external fluxes to changes in SBPase activity in the Calvin cycle model, with transaldolase present : (◇) Assimilation, (□) Export, (+) Storage. Despite the decline in assimilation flux at low SBPase activity, export flux is maintained at the expense of storage.

sumed rate equation:

$$\frac{V_{max}.(E4P.F6P - S7P.GAP)}{K_m + (E4P.F6P - S7P.GAP)} \tag{6.6}$$

Although equation 6.6 is undeniably crude, qualitatively it describes the behaviour of a two substrate, two product, catalysed reaction: at equilibrium (in this case assumed to be 1.0) there is no net conversion, the reaction saturates at V_{max}, and $K_m$indicates the concentrations of metabolites required to bring about 50% saturation. A more rigorous approach to the derivation of a rate equation for this reaction, as demonstrated by Cornish-Bowden [22, 23], leaves the user with a choice either, one of two equivalent equations one of which has nine parameters and eighteen terms, and the other twelve parameters and thirteen terms, or, an eqution of eight parameters and ten terms, depending on the assumed mechanism. As niether the mechanism nor the relavent parameters were known, and the object of the study was to investigate the qualative (or at best semi-quantitative) effect of including the enzyme, it is not thought that use of equation 6.6 will have a particularly detrimental effect on the relavent results.

In addition to the extra burden of complexity, the parameters required are the rate constants for individual substrate and product binding and dissociation, which in themselves do not have the direct physiological interpretation of $K_m$in equation 6.6.

Figure 6.9 shows the effect of the presence of transaldolase (as described by equation 6.6 (V_{max}= 7.5 μmol.h^{-1}.(mg Chl)$^{-1}$, K_m= 0.6 mM) on the original model, and the relationship to experimental data. No existing parameters in the model were altered. The version of the model with transaldolase provides a reasonable fit to the experimental data, and is clearly superior to the model without it. In the former case $C_{SBPase}^{J_{Assim.}} = 0.52$ (comparing with the experimentally determined value of 0.55).

There is a further point of comparison, although less precisely defined, between the model and work of Raines *et al* that transaldolase plays a rôle in CO_2 assimilation. The authors report that total starch at the end of the light period, and hence the flux thereto, underwent a detectable decrease at 71% wt activity, and declined monotonically with reducing SBPase. However, although sucrose concentration also decreased slightly between 71 and 100% wt activity, the relationship between the two was much less clear: at 23% wt SBPase activity sucrose concentration was actually higher than wt, while starch had declined to $\sim$ 50% wt. It was not until SBPase activity was at 15% wt that there was an unequivocal reduction in sucrose concentration.

If the assumption is made that most of the TP exported from the chloroplast is used

in sucrose synthesis, then sucrose concentration is, albeit indirectly, a measure of TPT flux. If this is the case then the responses of starch and sucrose to SBPase activity would appear to be another manifestation of that already observed and commented on several times in this thesis: that the Calvin cycle acts to preserve TPT flux, and thus fulfil the immediate demand of the plant, at the expense of starch synthesis. Figure 6.10 shows that, when SBPase activity is reduced in the model, starch synthesis flux falls more or less in parallel with the assimilation flux, leaving the TPT flux relatively unchanged. However this response is only obtained with the transaldolase step present in the model; in its absence TPT and starch fluxes decline equally (data not shown).

6.3.5 Relationship between rubisco and SBPase activity

It is possible to extend the explanation for the metabolic control analysis results obtained from the work of Raines *et al* which, is attractive because it also explains the results of Stitt *et al* and those of both the initial and evolved model. Furthermore this can be developed (in the next chapter) to provide the possible basis of an explanation that accounts not only for the control analysis behaviour, but also the dynamic and switching behaviour described previously.

Figure 6.11 shows the response of $C^{J_{Assim.}}_{rubisco}$ and $C^{J_{Assim.}}_{SBPase}$ in the (non-evolved) model, with transaldolase as described above, as rubisco activity is varied. There is a clear exchange of control between the two at a value of rubisco activity of about 120 μmol.h^{-1}.(mg Chl)$^{-1}$, three times the SBPase activity. The transition is sharp, and hence small changes in either enzyme in the region of the transition can lead to large changes in $C^{J_{Assim.}}$. An equivalent pair of curves can be generated by varying SBPase activity (not shown), the curves are substantially the same in the absence of transaldolase, the only difference being that the maximum value of $C^{J_{Assim}}_{SBPase}$ is 1.0, rather than $\sim$ 0.9 seen in figure 6.11.

Mark Stitt's results are consistent with wild type rubisco activity being slightly above this point under ambient conditions, but potentially able to gain control under altered environmental conditions. It is interesting to note that rubisco increased control under high light conditions, when it might be predicted that SBPase is more highly activated as a result of the thioredoxin system. As noted previously, Stitt [129] reports an abrupt transition in the curve of $C^{J_{Assim}}_{rubisco}$ v rubisco activity, and furthermore shows that the level at which this occurs coincides with the point at which the potential capacity to

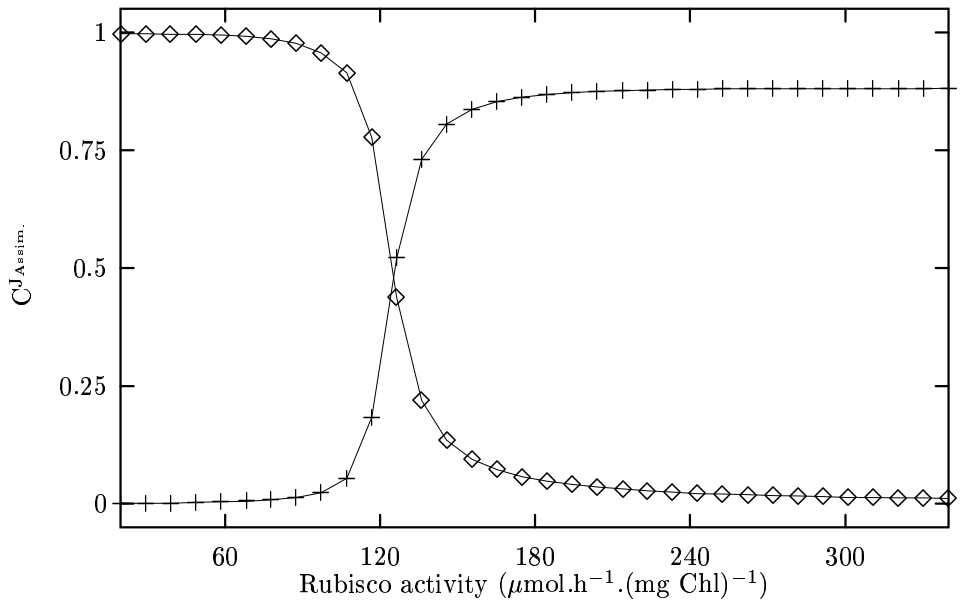

Figure 6.11: Calvin cycle model response of $C^{J_{Assim}}_{rubisco}$ ($\diamond$) and $C^{J_{Assim}}_{SBPase}$ (+) to changes in rubisco activity.

consume RuBP diverges from the rate at which RuBP is actually consumed, consistent with a transfer of control from rubisco to the regenerative limb of the cycle.

Raines' results are also consistent with rubisco activity lying slightly above the transition point. Here SBPase is close to the limiting value of its control coefficient, and therefore flux declines linearly with decreasing SBPase. Both Raines and Stitt used *N. tabacum* as their experimental subject, and so consistency between the two is to be expected.

In the original, unevolved model, rubisco activity was well above the transition (340 μmol.h^{-1}.(mg Chl)$^{-1}$) hence its negligible value of $C^{J_{Assim}}_{,}$ and the dominant value for SBPase (in the absence of transaldolase).

In the evolved model rubisco declined and SBPase increased in activity to median values of 158 and 57.1 μmol.h^{-1}.(mg Chl)$^{-1}$respectively, giving a rubisco:SBPase activity of 2.8:1, below the transition point, leaving rubisco with most of the control.

6.4 Summary

The main points established in this chapter are:

1. The general behaviour of the detailed Calvin cycle model of previous chapters arises from the gross structure (topology + qualitative kinetics) of the system, and not from some fortuitous but unidentifiable combination of factors in a complex model.

2. Although the behaviour of the model differs from that of those previously published, such differences are mainly attributed to differences between the structures of those models and that described here.

3. Consideration of the behaviour of the model under conditions of darkness, in conjunction with the experimental evidence concerning the thioredoxin system, suggests that the Calvin cycle and OPPP can be considered to be complementary components of a single system.

4. Not only are the model results consistent with experimental observation, but the model can be usefully employed in the analysis of such results.

The more general implications and explanations of these points will be considered in the next chapter.

Chapter 7

Conclusions

7.1 Relevance and implications of the Calvin cycle model

Chapters 4–6 demonstrate that one of the original goals of this project has been achieved: it is possible to construct computer models of biochemical systems, of (or at least approaching) realistic complexity, and, by treating such models as experimental entities, gain new insight as to the possible *in vivo* behaviour of the system under investigation.

In terms of both money and effort, the modelling approach used in this project has been shown to yield high returns. The total effort required to generate the experimental data to which the model was compared in chapter 6 is not known, but must have required, at least, decades of researcher-years and millions of pounds. In contrast, this project has involved no capital outlay, beyond the purchase of one computer, and has taken one worker less than four years. Furthermore, although investigating the Calvin cycle model was probably the single most time consuming component of the project, this certainly amounted to less than half of the total time, the balance being taken up with software development, devising strategies for extracting and analysing data from a large model, and other such activities. This ancillary work should be readily applicable to other large models, and it is thought that the time required to undertake a similar study would be less than a year.

The success, such as has been achieved, is attributed to two main factors:

- The model was constructed with a bare minimum of simplifying assumptions, and

is therefore a more realistic representation of the system than previously published work.

- The possibility created by Scampi of placing the model in the context of a programming language, allowed the model to be the subject of algorithms, which then allows the user to rapidly and conveniently investigate a model, with unconstrained flexibility[1].

7.1.1 Comparison of model behaviour with experimental evidence

The model, as described in chapter 6 has been compared with a wide range of experimental data and found to be in good qualitative agreement with such data, and for most of the variables examined, quantitative results lie within the range of experimental observation. The model appears to be able to simultaneously account for observed dynamic (oscillatory) behaviour, response to external (or cytosolic) phosphate concentration, and responses to changes in enzyme levels, as investigated by traditional physiological techniques, and genetic manipulation.

The comparison of model with experimental data is not complete, and in particular the work of [67] and [103] , (who have antisensed FBPase and G3Pdh respectively), has not been addressed. However this lack is the due to time constraints: no effort has been made to simply select those experimental studies reporting results similar to those of the model, whilst ignoring those that are not.

Furthermore, no effort was made to "tune" the model to bring it into line with experimental observation. It would therefore appear that the Petterssons' original choice of parameter values, despite originating from disparate biological sources, was reasonable. It was also striking that when the model was subject to the ES algorithm (itself a simulation of a natural process to which all biological systems are subject) to make the biologically useful improvement of increasing assimilation flux:protein ratio, that the dynamic behaviour, which played no part in the optimisation, became quantitatively more realistic.

The most unexpected aspect of the modelling results was, without doubt, the ability of the model to switch between two steady states. This was first seen as a response

[1] Other than the numerical tolerance limits imposed by the hardware

to changing light levels, but can also be induced by reducing $P_{i_{ext}}$ to very low (and probably unphysiological) concentrations, and by increasing SBPase activity. Although such discontinuous responses might be regarded with some skepticism by biochemists and physiologists long accustomed to seeing responses as smoothly saturating curves, similar behaviour has been observed in other models , in cell free extracts [121], and in suspension cultures [131]. These studies all related to glycolysis, and this behaviour has not been previously seen in a Calvin cycle model, although Laisk *et al* cite a Russian language paper [90] which claims that the response of assimilation to light in lilac leaves more closely resembled a discontinuous line than a rectangular hyperbola.

On the basis of these comparisons it is possible to suggest that a number of characteristics, which, when taken together, serve to describe the general behaviour of the stromal components of the photosynthetic apparatus.

- The system is opportunistic. Carbon is assimilated at the maximum possible rate, and that which is not required for immediate use in cytosolic metabolism is stored as starch.

- The fact that the input flux branches into two main output fluxes (TPT and starch), has a major influence on the nature of control within the system, as knowledge of C^J over one external flux gives no information as to C^J over the other two.

- Control over assimilation flux lies almost exclusively with rubisco and SBPase, the proportion each has is variable, and is likely to range between 0 and 100%. Control of export flux lies predominantly with the TPT, although other enzymes of the regenerative limb of the cycle may also exert some control. Most of the control of starch synthesis flux lies not with the traditional rate limiting step of starch synthase, but with the rest of the system.

- It is clearly possible that the source of experimentally observed photosynthetic oscillations does lie within the Calvin cycle. The investigations of skeleton models in the previous chapter suggests that this is a general property of potentially autocatalytic cyclic systems, and not due to the special properties of any one enzyme.

- The Calvin cycle has the potential to exist in two steady states, and to switch between them, although, on balance it appears that such behaviour is the exception,

and not the rule. Some experimental evidence exists that supports this, but to this author's knowledge, no work has been conducted with the specific intent of inducing and observing such behaviour.

- The Calvin cycle and the OPPP are intimately related, and both can be regarded as components of the same overall system. They are not, as seems to be commonly thought, two independent pathways that, by coincidence, share the same sub-cellular compartment.

- A major rôle of the thioredoxin system is to balance the activities of the Calvin cycle and OPPP. Furthermore it is possible that the real signal to which the thioredoxin system responds may be chloroplast redox status, and not light, although light is obviously likely to be the major factor influencing this.

7.1.2 An explanation of observed behaviour

In the light of the foregoing observations it is possible to propose an explanation relating these to structure of the Calvin cycle The explanation depends upon two uncontentious facts: the system is cyclic, and the regenerative limb contains components that are saturable.

Autocatalytic cyclic systems (biochemical or otherwise) exhibit positive feed-back, and if rate equations are linear, individual variables increase exponentially with time. The true rate equations for real world systems always contain some non-linearity, that, however small, will eventually serve to constrain exponential growth [132].

Calvin cycle behaviour is limited by one of two possible constraints

Examination of skeleton models C_{1--3} in figure 6.1 reveals that even if individual rate equations are linear, that the behaviour of the whole model is constrained (i.e. concentrations and fluxes cannot grow to infinity) because total P_i is constrained. It will be recalled that in model C_0 (which does not have this constraint) few steady states were found, and that the majority of these were unstable (Chapter 6, Table 6.1).

In addition to P_i limitation, a second constraint exists if the rate equations are saturable. If P_i is in plentiful supply then one can expect flux in the system to increase exponentially until one step saturates, and this will represent a new set of limiting[2]

[2]The meaning of the phrases "limiting" or "limiting reaction" etc. in this discussion are

conditions.

Much of the previously described behaviour of the Calvin cycle model can be explained if it is considered that the switching behaviour represents the system moving between these two limiting states.

Behaviour in the fast steady state

If either rubisco or the regenerative limb of the cycle are saturated, then assimilation flux cannot vary. Clearly, changing a parameter (other than those affecting V_{max} of the saturated enzyme) cannot increase the assimilation flux, and because the tendency of the system in this state is to exponential increase, limited only by the saturated step, changes that would otherwise reduce assimilation flux can only do so temporarily, as the system will rapidly return to its limited state.

This saturated state is the fast steady state described in chapter 4, in which assimilation flux was affected only negligibly by parameters other than V_{max} of SBPase (see for example figures 4.6 and 4.10), and is effectively clamped at a rate three times greater than this value. Using the original parameter set it is SBPase that is saturated (SBP concentration is 2 orders of magnitude above the K_m value of SBPase). The stoichiometry of the regenerative limb is such that the steady state assimilation flux is exactly three times the flux carried by SBPase. It follows that if rubisco V_{max} falls to less three times that of SBPase then SBPase cannot be saturated, and it is rubisco that becomes the limiting reaction. This was demonstrated in the previous chapter (see Figure 6.11). In the evolved parameter set it is rubisco that becomes saturated and thus gains control; Table 6.3 in the previous chapter reveals that in the model with the evolved parameter set, SBP was less than the K_m of SBPase, but RuBP is $\sim$ 2 orders of magnitude greater than the K_m of rubisco. Examination of experimentally reported metabolite concentrations in the same table suggests that it is more likely that it is rubisco that is saturated toward RuBP, than SBPase toward SBP, although it may be expected that this might change with environmental conditions.

Now, the kinetics of the TPT respond positively to $P_{i_{ext}}$, but if the assimilation flux is clamped to a constant rate, then increased flux through the TPT is only possible at

subtly different to traditional definitions of "rate limiting steps". Here "limiting" denotes the component of the system that prevents unconstrained behaviour. Quantitative changes to parameters of such reactions may be expected to have a disproportionate effect on the system, and in some respect their behaviour may be similar to traditional "rate limiting steps".

the expense of starch flux. If starch phosphorylase is present, then this not only allows a smooth transition from net starch synthesis to degradation at high TP demand, but also appears to enhance the clamping of assimilation flux, even when there is net starch synthesis. This is then the reason for the shape of the flux response curves to $P_{i_{ext}}$ in chapter 4 (figure 4.6), and for the fact that TPT and Starch synthase have negative C^J over each other's fluxes (assuming flux to starch is positive).

The slow steady state

In order to sustain the cycle in the fast steady state, an adequate supply of free phosphate is required. In skeleton models c_{0-2} P_i is assumed to react directly with Calvin cycle intermediates; in model c_3 and in the complete version it is carried by ATP, in turn regenerated by the light reactions. Comparison of the results from models c_2 and and c_3 suggest that the additional substrate cycle introduced by the inclusion of ATP/ADP does not have any great effect on the overall behaviour. However, in the case of the complete model it means that limitation of P_i assimilation can occur for two (albeit rather closely related) reasons: either the activity of the light reactions is reduced, or P_i concentration becomes low.

Regardless of its source, if the maximum rate at which P_i can be assimilated is not sufficient to drive some part of the cycle to saturation, then different behaviour will apply. The kinetics of the TPT still ensure that TPT flux responds positively to $P_{i_{ext}}$, but this can now be accommodated by increasing assimilation flux. Increasing internal P_i results in increased flux in the reductive and regenerative limbs (unless the light reactions are saturated with respect to P_i) and thus assimilation flux does increase.

Furthermore, starch synthase is able to compete for the extra carbon, now drawn into the cycle, from the increased assimilation, and hence in the slow steady state the response of starch synthesis flux to $P_{i_{ext}}$ is positive. As the effect of increasing TPT activity is also to bring more P_i into the cycle $C^{J_{Starch}}_{Assim}$, $C^{J_{TPT}}_{TPT}$, $C^{J_{Starch}}_{TPT}$ are all positive.

Breakdown in the absence of starch phosphorylase

The foregoing discussion has assumed that starch phosphorylase is present. In the absence of this reaction the discussion remains valid, but an extra factor must be taken into account. As well as P_i in order to maintain a flux the cycle must maintain a supply

of carbon in the form of PGA[3] , and in contrast to P_i, total carbon is not fixed. Thus the possibility exists for all carbon to be lost[4] , resulting in a possible third steady state in which all fluxes are equal to zero. As seen in chapter 4 this state is entered in the absence of starch phosphorylase and high values of $P_{i_{ext}}$, and has been described experimentally in isolated chloroplasts [37,93]. This dead state is approached relatively smoothly (Figures 4.2 and 4.3) and thus at higher values of $P_{i_{ext}}$, the responses of both assimilation and TPT flux toward $P_{i_{ext}}$ become negative.

In the presence of starch phosphorylase this dead state is no longer possible because F6P can now be fed into the cycle. As described in chapter 6 (section 6.3.2) there is no route between F6P and exported TP other than via the assimilation reaction. However, TP is a co-substrate with F6P for three mass-action driven reactions (the E4P aldolase and both the transketolase reactions), and, regardless of demand for TP, F6P appears to be able to reach sufficient concentration to force these in the forward direction, eliminating the possibility of the dead steady-state. As the dead state does not exist, the system cannot approach it, and hence in the fast state response to $P_{i_{ext}}$ is asymptotic and $C^{J_{Assim}}$ values remain constant.

Relevance to dynamic behaviour

This hypothesis of two limiting conditions can also explain some of the dynamic behaviour in the model. In general damped oscillation in a system represents the symmetrical variation around an asymptotically approached steady state. However, in the fast steady state at least one reaction is carrying a flux at, or at least very close to, the absolute maximum, and thus if oscillation is to be symmetric, it must necessarily be of low amplitude. As it is proposed that the fast steady state is approached exponentially, any oscillations in this state will be extremely heavily damped, as seen in Chapter 4 (Figures 4.13 and 4.14).

In the slow steady state such constraints are removed, and hence both amplitude and damping time are greater. At present it is not possible to offer an explanation as to why increased light or P_i should increase the damping time, although from the previous discussion it is to be expected that both should have the same qualitative effect. It is

[3]As cycles have no beginning or end, the following argument holds equally well for any of the other sugar phosphate intermediates

[4]In fact some, mainly hexose phosphate remains. As long as the co substrates of remaining intermediates are zero the cycle cannot proceed

possible to speculate that as light, and/or P_i, increase that the system is more strongly attracted to the fast steady state, and thus takes longer in returning to the slow.

7.1.3 Physiological implications

The traditional view of metabolic systems has been that their control and regulation could be understood by consideration of the kinetics of a single key step. The Calvin cycle and related reactions described in this thesis was no exception, with rubisco commonly being identified as the rate limiting step for photosynthetic carbon assimilation (e.g. [82, 139]).

In a substantial review article ($\sim$ 50 pages), Woodrow and Berry [139] devote little more than one page to the section entitled "Other enzymes", while the remainder concentrates on a detailed discussion of rubisco kinetics. Despite adopting some Metabolic Control Analysis terminology, the authors view of control of the Calvin cycle is dominated by rubisco, and state "All of the enzymes of Calvin cycle ... have the potential to affect rubisco activity and therefore the rate of CO_2 fixation".

Similarly, in discussing plant starch metabolism, Beck and Ziegler [12] assert that "regulation of starch synthesis is centered almost exclusively on ADPG-pyrophosphorylase". Later in the same review the authors discuss likely "rate limiting" reactions in the *degradation* of starch, without noticing the paradox brought about by the fact that the net *in vivo* rate of starch accumulation is the sum of synthesis and degradation[5] .

Even those workers who accept that more than one enzyme may be important almost invariably focus their attention on the detail of individual kinetics, and particularly the rôle played by various effectors, but give little or no consideration to the contribution made by the structure of the system under investigation.

The work in this thesis suggests that although it is possible for enzymes within the Calvin cycle to appear to have the properties of traditional "rate limiting steps", the degree of control they posses, and the much of behaviour of the system, stems not from their kinetic properties, but from the system's topology.

Another notable instance of behaviour arising from topology and not kinetics is that of switching. The traditional view is that sharp transitions in response curves occur as the result of enzymes with strongly sigmoidal kinetics, which are, or become,

[5] In fairness to the authors it should be pointed out that their main interest was in storage starch metabolism, where the processes of synthesis and degradation are likely to be widely separated in time

rate limiting in the low activity state. Neither the full model, nor the skeleton models, contained reactions with such kinetics, and the kinetic equations of the full model were far more complex than those of the skeleton, and yet both switch.

It seems clear that a proper understanding of metabolic systems will not be attained unless at least equal weight is given to the properties of the whole system, and not just those of its components in isolation.

Although it is reasonable to suppose that the complex kinetics exhibited by many enzymes, and conserved across many species, do offer some evolutionary advantage, it appears possible that this has relatively little to do with the control of flux, at least under normal circumstances.

An example of how the consideration of topology can give insight as to the advantage conferred by kinetic features enzymes, is the effective coupling of rubisco, FBPase, SBPase, and G3Pdh activities, via the thioredoxin mechanism. As described in the previous chapter, and for reasons outlined above, it is possible to propose that one of the functions of the thioredoxin mechanism is to prevent the transition to the slow steady state at low light levels, thereby maximising carbon assimilation under such conditions (And not as one anonymous researcher once informed this author "to prevent carbon fixation at night (!)").

7.1.4 Scope for "improving" the Calvin cycle

The goal of improving crop productivity presumably dates back to the first farmers. In recent years much interest has been shown in the potential of genetic manipulation techniques for increasing plant productivity [135]. Most frequently the target for such manipulation has been rubisco (e.g. [11]), but in any case tends to center around the activity of identifying a rate limiting step, and over-expressing the gene for the enzyme catalysing that step.

Material presented in this thesis provides little basis for confidence in this strategy: at least four lines of reasoning can be proposed that suggest that it is unlikely to succeed.

1. It has been shown that a probable effect of increasing one of the rate limiting steps over assimilation is simply to transfer control with little or no net increase in assimilation.

2. If other steps in the Calvin cycle have enough head-room to allow an increase in the limiting step to be translated into a *potential* increase in CO_2 assimilation, but

there is no matching increase in the rate of the light reactions, the Calvin cycle will switch to the low steady state, resulting in a decrease in assimilation.

3. The thioredoxin system is also likely to act to frustrate such efforts. Increasing the rate of assimilation causes the stromal redox potential to become more oxidising, thus causing the thioredoxin system to down-modulate the activities of several enzymes, opposing the increase that might otherwise have been brought about.

4. Even if an increase in assimilation could be brought about, it has been seen that those enzymes which have high $C^{J_{Assim}}$ also tend to have high $C^{J_{Starch}}$ and low, even negative, $C^{J_{TPT}}$. Thus the consequence of increasing assimilation flux might simply be increased levels of leaf starch. This point has also been raised by Geiger and Servaites [41], although on the basis of entirely different reasoning.

Furthermore, these points, and the material in this thesis, relate only to metabolic and enzymatic mechanisms, and neglect the possibility of regulation of gene expression. Evidence exists [68, 120] to suggest that such mechanisms do indeed exist, and that their effect is also likely to oppose any increase that might otherwise be brought about in photosynthetic carbon flux.

Assume that none of these criticisms are valid, that the genetic mechanisms are sophisticated enough to readjust the expression of other enzymes in order to realise the potential increase brought about by some genetic manipulation, and consider the consequences: To fulfil the increased capacity of the Calvin cycle requires increased light reaction activity, which, given that quantum efficiency is high [135] must be achieved by increasing total leaf area. This in turn implies increased rates of transpiration, and must therefore be supported by a larger root and transport system. In short, increased assimilation demands larger plants, and plant size is not under the control of photosynthetic reactions, but plant hormones (e.g. [143]).

There is experimental evidence for this chain of reasoning. Quite small flux reductions (brought about in this case by increased branch-point competition) in reactions involved in plant hormone synthesis led to plants of drastically reduced stature [39] and hence total CO_2 assimilation, although this was not those authors' original intent.

7.2 The methodology used

The general strategy used to investigate the Calvin cycle model, which has met with at least some success, and can hopefully be applied to other models in the future, may be summarised as follows: Firstly, construct the model in as much detail as knowledge of the system allows, placing the resulting burden of complexity upon the computer. Secondly observe the behaviour of as many the model variables as possible, over as wide a range of parameters as possible. The power of modern computers makes the task of browsing the resulting large data sets, in order to separate genuinely novel behaviour from that which is but a variation on previously observed behaviour, relatively light. Having characterised the possible ranges of behaviour, it then becomes possible to search for simplified models of the system that exhibit similar behaviour, and may thus be used as a basis for an explanation.

Use of linearising assumptions

One result of this approach has been to reveal a potential problem arising from the use of linearising assumptions. It is generally accepted (e.g. [132,89] that in the (infinitesimally small) region of a steady-state a system of non-linear ODEs may be treated as linear, and that the steady state behaviour may then be investigated by linearising the algebraic solution to the non-linear system. However, if the approach is then extended by first linearising the individual ODEs, and then determining an analytic steady-state solution, the possibility of obtaining a qualitatively incomplete solution arises.

For example, in the case of the skeleton model c_3 in the previous chapter (see section 6.1), the existence of two steady states in the non-linear version of the model was suggested by the bi-modal distribution of $C_{TPT}^{J_{StSynth}}$ and confirmed by further numerical modelling. However the linearised version of the model showed no evidence of such behaviour, and subsequent algebraic analysis (appendix C) proved that only one solution involving only positive concentrations can exist. Thus the presence of non-linear terms in rate equations can alter the qualitative, as well as quantitative, characteristics of a system.

Use of randomised models

The use of large numbers of randomised parameter sets to characterise the behaviour of model systems (as applied to the skeleton models of the previous chapter) has not,

to this author's knowledge, been described before, although models in which ODEs are combined with a source of random numbers are not uncommon [15]. The approach has proved to be very useful: the four non-linear skeleton models, and two of their linearised counterparts, were all characterised in a matter of hours: in contrast even the linearised version of the most complex model appears to analytically intractable, unless further drastic simplifying measures taken.

In addition to the specific use of these results, further consideration suggests a more general possible conclusion, of concern to experimentalists, and those analysing their results. It will be recalled that the parameter sets supplied to these models were drawn from uniformly distributed sets of random numbers. However, none of the resulting samples of model variables bore any resemblance to a uniform distribution. This in itself is not surprising: non-linear responses tend to be the rule and not the exception. If this is the case then it follows that if, in nature, a measured enzyme activity (for example) is normally distributed, as implied by the almost universal reporting of results as $\bar{x} \pm \sigma$ (or some other measure of spread derived from σ), then system variables such as concentration cannot be. It hardly needs stating that these variables too, are almost invariably reported in the same fashion.

This will be less of a problem if the major source of variation in such measurements arrises from the experimental technique, and this is normally distributed. It is also possible for the impact of the problem to be lessened, if such small samples are taken that their distribution is not distinguishable from normal, although this would presumably result in unnecessarily wide confidence limits. None the less, it would be interesting to see such matters established, and not assumed.

Use of Evolution Strategy algorithms

Use of the ES algorithm has proved to be extremely encouraging, and has certainly shown much better performance (in terms of the likelihood of convergence) than the Marquadt-Levenberg algorithm, as implemented in the Gnufit and Dynafit programs [65, 74], and generally regarded as the standard non-linear fitting algorithm [99]. Since chapter 3 was written the approach used to fit the lactate dehydrogenase progress curve (section 3.4.3) has been used to the same end for the *E. coli* threonine synthesis pathway, in a cell free system. This system consisted of seven reactions, twelve metabolites and some forty parameters. Good fits were obtained to the progress curve, achieving a smaller mean

relative squared residual than the lactate dehydrogenase fitting, although examination of the residuals revealed a small but definite systematic error. It is not known if this represents premature convergence by the ES algorithm, or an inaccurate or incomplete model.

There are however several areas in the understanding and use of this algorithm with that have scope for improvement. Experience with the use of ES suggests (not surprisingly) that the rate at which convergence is achieved, in terms of number of organisms evaluated, depends upon population size, the proportion of survivors in the next generation, mutation size, and the complexity of the model under investigation. It would clearly be useful to have a method to determine, or at least approximate, *a priori* optimal strategy parameters for a given problem.

Even if rate of convergence is not optimal, to be properly useful a fitting algorithm must be able to supply confidence limits for the fitted parameters. As noted in chapter 3, despite the fact that the algorithm maintains a population (in the evolutionary sense) of solutions, it is not obvious as to how this might be used, as the individuals are not independent, and their distribution not normal. Cornish-Bowden [20] and Press *et al* [102] pay considerable attention to such problems, and there are doubtless many strategies waiting to be pursued.

Although resistant, ES has shown itself not to be immune from the twin problems of failing to converge, and converging on sub-optimal solutions, that appear to be universal to all fitting/optimisation algorithms. There would seem to be two or three possible causes for this, and it appears promising that they can be tackled in a piecemeal fashion.

Use of elementary modes analysis

The development and implementation of this form of analysis was not a part of this project, but has shown itself to be extremely useful. It has been of particular benefit in circumstances in which there is no flux at all in a large kinetic model, to be able to prove that there is no possibility of a flux, regardless of parameter values.

A particular example of this was the ability to show that, in the Calvin cycle model, no route exists between starch and TPT export, that does not involve rubisco. Not only is this fact far from obvious by inspection, it is quite counter-intuitive, given the knowledge that starch degradation can allow the export flux to exceed the assimilation flux.

At present the elementary modes in a model are determined by the "Empath" [140] program written in the "Smalltalk" language by a previous colleague, and not readily compatible with Scampi.

However, as the only input to the algorithm is the stoichiometry matrix, which is readily available to Scampi, and a 'C' implementation of the algorithm is also available (by kind donation of Stefan Schuster), this form of analysis will be incorporated into Scampi in the near future.

7.3 Scampi

Overall, Scampi has fulfilled its design remit of providing a flexible, portable, extensible, and reliable system that allows biochemical models to be made the subject of algorithms.

The same source code has been compiled and tested on three different platforms (Amiga, Sun/Sparc, and PC/Win95) with two compilers (gcc and MS-C/C++) without the need for any changes or platform dependent compiler directives. It is reasonable to hope that porting to other platforms in the future will not raise major problems.

Wise programmers should be extremely cautious in asserting that their code is "bug free"; however it is possible to report that none are known at the present time. Over the last two years only two are known to have escaped the testing stage of development (they are now eliminated), and no behaviour has been observed in programs using Scampi to suggest any remain.

7.3.1 Problems with Scampi

Those advantages conferred by Scampi all stem from embedding biochemical models within a general purpose programming language. Unfortunately this inevitably brings two major disadvantages that together comprise a formidable barrier to Scampi becoming a regular tool of the biochemist/biotechnologist.

The first disadvantage is that Scampi demands of the user the ability to construct computer programs. This is a skill that requires more than a text-book, a computer, and enthusiasm to acquire. It requires formal training and no small measure of time and dedication. It is entirely understandable that workers will be reluctant to make a major investment in a technique that, despite having antecedents in the dawn of the computer age, has made but the most mild of impacts upon their chosen field.

The second disadvantage, closely related to the first, is the fact that Scampi possesses no graphical user interface (GUI). Although, as described in chapter 2, this is the result of a deliberate design decision, the fact remains that modern computer users, especially those whose introduction to the subject has been via games or word-processing, tend to regard software without a GUI as being without merit.

7.3.2 A Successor to Scampi

Deplorable though this state of affairs may be, there is little prospect of remedying it. Hence it is hoped to produce a successor to Scampi (using Scampi as a low-level foundation), that includes a simple high level language, encompassing the relavent functionality currently supplied by SCAMP, Scampi and C, and a GUI with sufficient functionality to allow the neophyte user to perform simple modelling tasks without recourse to the language. Since the original Scampi project was started the Java programming language [47, 88], has received much acclaim. Initial impressions are that it does not suffer from the problems of other systems for writing portable GUI programs (described in chapter 2), and would be ideally suited to such a project.

7.4 Directions for future work

7.4.1 Further development of the Calvin cycle model

No piece of academic inquiry can be regarded as complete, any answers obtained inevitably contain within them the seeds of further questions. Furthermore, the comparison of experimental and modelling work in this thesis provides good evidence that the model can account for many experimental observations, and is therefore worth developing. At present there is no funding to continue the Calvin cycle work, and so the points outlined below represent proposals that would be pursued were resources available.

Removal of remaining kinetic assumptions

A common theme in this thesis is that in constructing a computer model of a real and complex system, all simplifying assumptions should be viewed with suspicion. A major assumption remaining is that the reversible reactions can be described by mass action kinetics. This is in contradiction of known facts: all enzyme catalysed reactions are

saturable, and the effect on the model that the resulting non-linearity causes cannot be predicted without removing the assumption.

Furthermore there is experimental evidence that some of the enzymes that in the model maintain disequilibrium ratios (ρ) very close to unity, do not do so *in vivo*. Kruckeberg *et al* [72] predict a value of ρ for PGI in the chloroplast of ~ 0.5, and were able to directly measure ρ_{PGI} as 0.7 in the cytosol, where the isozyme is about twice as active as in the chloroplast.

Lines of reasoning developed over this and the previous chapter, to explain the behaviour of the Calvin cycle model in terms of topology, strongly suggest that no qualitative change in behaviour will result from such a change; but the cost of introducing it is low, and a model of similar complexity, but known to be more realistic, is always to be be preferred to that which is less so. This is especially pertinent as the model has, under certain circumstances, been observed to behave in a fashion almost certainly absent in nature (as described in the next section).

As has been mentioned, the kinetic equations for StPase, G6Pdh, and transaldolase were developed in an extremely *ad hoc* fashion. An effort should be made to replace them with more rigorously derived versions.

Characterisation of the evolved population

The characterisation of the evolved population described in chapter 5 has only been carried out for fixed parameter values. The responses of the population to changes in environmental parameters such as light and $\mathrm{P}_{\mathrm{i}_{\mathrm{ext}}}$ should be determined as they were for the single model in chapter 4.

Attention should also be turned to the unstable oscillations exhibited by the population. They are of interest because oscillations of this type have not, to the author's knowledge, been previously described in a model of the Calvin cycle. Similar behaviour in the glycolitic pathway has, however, been widely reported and discussed by (for example) Goldbeter [45, 46]. Apart from their novelty, these oscillations represent some cause for concern, as nothing like them has appeared in the experimental literature. They should therefore be regarded as an artefact of the model, and eliminated.

Extending the Calvin cycle model

The previous points in this section are essentially "mopping up" operations, and would have been attended to as part of this project, had time allowed. Despite these points, it is thought that characterisation of this model of the Calvin cycle is substantially complete, and attention should be turned to include more of the metabolic processes with which it is known, or suspected, to interact.

The single most important modification that can be made to this model is to make the representation of the light reactions more realistic, by allowing NADP/H to become free variables. In addition to making the model more complete, there are several reasons for doing this:

1. The behaviour of G3Pdh is anomalous amongst the reversible enzymes in the model: It is the only one to maintain a value of ρ substantially displaced from 1.0, and the only one to maintain non negligible C^J values. NADPH, and NADP are substrate and product respectively for this enzyme.

2. The possible influences of stromal redox status have already been discussed. By having NADP/H free it will become possible to investigate this properly. This would also be a first requirement in the implementation of the OPPP and thioredoxin system.

3. It is possible to continuously and non-invasively measure NADPH levels in intact chloroplasts, by spectrophotometric means. It would be extremely interesting to fit such data sets to the Calvin cycle model as was described in chapter 3 for a single enzyme system.

The extensions to the model described above still leave the model restricted to events in the chloroplast. The real reason for investigating the system, by modelling or any other means, is to understand the behaviour of the whole organism, and not just a part. Therefore it would appear logical that the next extension to the model should embrace cytosolic metabolism.

Historically, most effort has, understandably, been directed to the carbon sink-source relationship that exists between cytosol and chloroplast. All of the modelling studies cited in the previous chapter that included cytosolic components, included some representation of this, but not of other possible interactions.

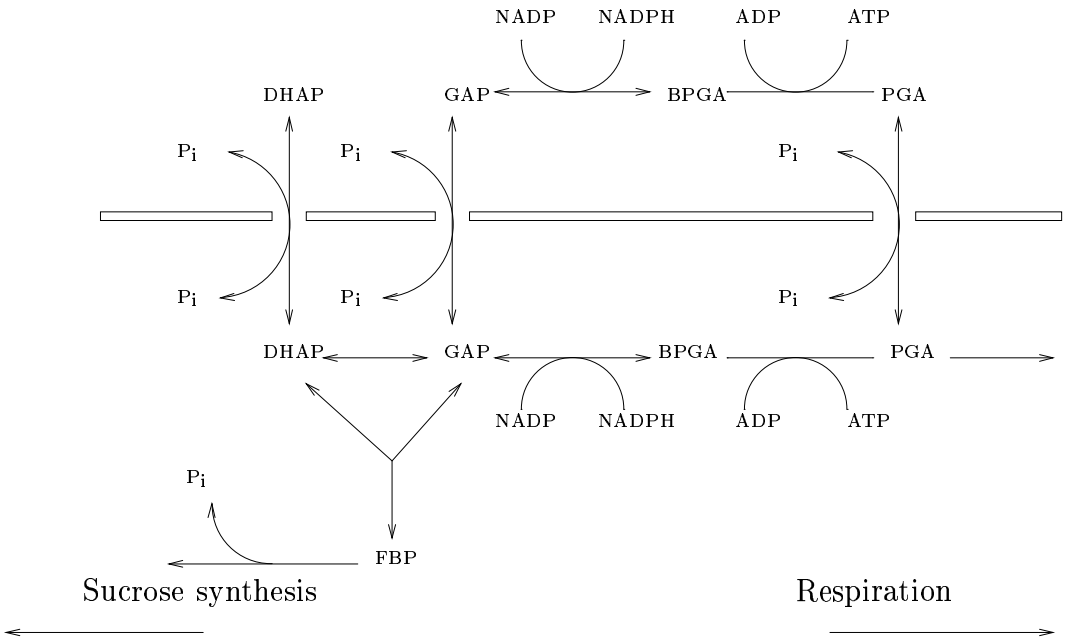

Figure 7.1: Carbon metabolism at the chloroplast/cytoplasm interface

Despite this, there would probably be some merit in investigating a more detailed model of source-sink interaction, as when reactions involving TP in the cytosol are included in the model, quite a complex network is formed, involving reactions on both sides of the chloroplast envelope, as shown in Figure 7.1. Furthermore, these reactions include cytosolic FBPase, which is known to be strongly inhibited by the cytosolic metabolite F-2,6-BP, a metabolite thought to play a pivotal rôle in partioning carbon flux between sucrose synthesis and respiration, but whose function in the context of overall carbon metabolism is still poorly understood [73, 127].

A second, and less thoroughly investigated aspect of chloroplast-cytosol interaction, is that mediated by redox transfer. At least two mechanisms exist to transfer redox potential across the chloroplast envelope, the malate-oxaloacetate (Mal-OAA) shuttle, and the TP-PGA shuttle [56]. Bearing the latter in mind, it can be seen that figure 7.1 can contain hitherto unseen complexity, and can simultaneously support three independent fluxes: TP export as a starting point for sucrose synthesis, PGA flux feeding the lower part of glycolysis and mitochondrial respiration, and a cycle between cytosolic and stromal PGA and GAP, resulting in no net carbon flux, but mediating a redox flux. All of

the stromal enzymes involved in these shuttles are under the influence of the thioredoxin system [8] so as to be activated as the stroma becomes increasingly reduced. Fridyland *et al* [40] have published a model study of the Mal-OAA shuttle, including rate equations and parameters of the enzymes involved, making the addition of this component to the existing Calvin cycle model a straightforward task.

Furthermore, the Mal-OAA is known to exist in plant mitochondria [38, 141], thus allowing the possibility of a network of redox flux, also with a rôle in the control of carbon flux, extending from the mitochondria, through the cytosol, and into the chloroplast, and involving several stromal enzymes under the influence of the thioredoxin mechanism. Ideas similar to this have already been expressed in the published literature: Hanning and Heldt [54] have proposed a network of redox communication involving chloroplast, cytosol, mitochondria and peroxisome; Schiebe [122] suggests a far more dynamic rôle for the thioredoxin system than a simple on/off switch responding to light levels. The complexity of the system suggested makes it unlikely that its potential behaviour could be determined, let alone understood, by intuition unaided. A detailed model, built by extending the Calvin cycle model described in this thesis would certainly be feasible, and would appear to offer the possibility of making reasonable return on any such investment.

Finally no section discussing the extensions to a metabolic model can be complete if the possible influence of deoxyribonucleic acid is entirely ignored. There is ample evidence that co-ordinated changes in levels of photosynthetic gene expression do occur in response to a variety of stimuli. For example all of the antisense investigations, described in the previous chapter, reported that at very low levels of expression of the target gene, chlorophyll content was reduced in addition to the activity of at least one other Calvin cycle enzyme. For most of these studies this coincided with the level reduction in target enzyme activity that was sufficient to bring about a detectable decrease in CO_2 assimilation. Increasing the carbohydrate, by several experimental approaches, to which photosynthetic tissue is exposed, [69, 68, 120] has consistently led to loss of chlorophyll, rubisco, and other Calvin cycle enzymes. At least some of these losses are associated with a reduction in mRNA transcript levels and *de novo* protein synthesis. The mechanisms responsible for such coordinated changes do not appear to known, a fact which will not make modelling them any easier.

Incorporating genetic components into the model may well cause the potential problem of stiffness to become realised. Genetically mediated changes are reported to occur

over a time-scale of days, rather than the minutes and seconds reported here. Whether this will really pose a major problem is not known at present, but it is certainly possible that new additions will have to be made to the software described, if models of this level of completeness are ever to be contemplated.

Bibliography

[1] Flex distribution and documentation. World Wide Web http://www.hensa.ac.uk/mirrors/gnu/flex.

[2] Gawk distribution and documentation. World Wide Web http://www.hensa.ac.uk/mirrors/gnu/gawk.

[3] Gcc distribution and documentation. World Wide Web http://www.hensa.ac.uk/ftp/mirrors/gnu/gcc.

[4] Xfig distribution and documentation. World Wide Web http://www.hens.ac.uk/mirrors/FreeBSD/FreeBSD-current/ports/graphics/xfig.

[5] A. V. Aho, B. W. Kernighan, and P. J. Weinberger. *The Awk Programming Language*. Addison-Wesley, Reading, Mass., 1988.

[6] E. Ainscrow and M. Brand. Top-down control analysis of systems with more than one common intermediate. *Eur. J. Biochem.*, 231:579–586, 1995.

[7] K. R. Albe, M. H. Butler, and B. E. Wright. Cellular concentrations of enzymes and their substrates. *J. Theor. Biol.*, 143:163–195, 1990.

[8] L. E. Anderson. ?? light inhibition of stromal transaldolase ?? *Find out*, Find out:Find out, 1985.

[9] L. E. Anderson. Light/dark modulation of enzyme activity in plants. *Adv. Bot. Res.*, 12:1–46, 1986.

[10] T. Back and F. Hoffmeister. Basic aspects of evolution strategies. *Statistics and Computing*, 4:51–63, 1994.

[11] G. Bainbridge, P. Madgwick, S. Parmar, R. Mithchell, M. Paul, J. Pitts, A. Keys, and M. Parry. Engineering rubisco to change its catalytic properties. *J. Exp. Bot.*, 46:1269–1276, 1995.

[12] E. Beck and P. Ziegler. Biosynthesis and degradation of starch in higher plants. *Annu. Rev. Plant Physiol. Plant Mol. Biol.*, 40:95–117, 1989.

[13] S. Bickel-Sandkötter and H. Strotman. Nucleotide binding and regulation of chloroplast ATP synthase. *FEBS Letters*, 125:188–192, 1981.

[14] D. Brown and P. Rothery. *Models in Biology: Mathematics, Statistics and Computing*, chapter 1. Wiley, Chichester, 1993.

[15] D. Brown and P. Rothery. *Models in Biology: Mathematics, Statistics and Computing*, chapter 4. Wiley, Chichester, 1993.

[16] G. Brown, R. P. Hafner, and D. Brand. A 'top-down' approach to the determination of control coeffecients in metabolic control theory. *Eur. J. Biochem.*, 188:321–325, 1990.

[17] J. A. Burns, A. Cornish-Bowden, A. K. Groen, R. Heinrich, H. Kacser, J. W. Porteous, S. M. Rapoport, T. A. Rapoport, J. W. Stucki, J. M. Tager, R. J. A. Wanders, and H. V. Westerhoff. Control analysis of metabolic systems. *Trends Biochem. Sci.*, 10:16, 1985.

[18] M. L. Cárdenas and A. Cornish-Bowden. Rounding error, an unexpected fault in the output from a recording spectrophotometer: Implications for model discrimination. *Biochem. J.*, 292:37–40, 1993.

[19] A. Cornish-Bowden. *Analysis of Enzyme Kinetic Data*, chapter 3. Oxford University Press, Oxford, 1995.

[20] A. Cornish-Bowden. *Analysis of Enzyme Kinetic Data*, chapter 5. Oxford University Press, Oxford, 1995.

[21] A. Cornish-Bowden. *Fundamentals of Enzyme Kinetics*, chapter 10. Portland Press, London, 1995.

[22] A. Cornish-Bowden. *Fundamentals of Enzyme Kinetics*, chapter 4. Portland Press, London, 1995.

[23] A. Cornish-Bowden. *Fundamentals of Enzyme Kinetics*, chapter 6. Portland Press, London, 1995.

[24] B. Crabtree and E. A. Newsholme. A quantitative approach to metabolic control. *Curr. Top. Cell. Regul.*, 25:21–76, 1985.

[25] R. P. Dunford, M. C. Durrant, M. A. Catley, and T. A. Dyer. Location of redox-active cycteins in chloroplast SBPase idnicates that its allosteric regulation is similar but not identical to that of FBPase. In *Carbohydrate Metabolism in Plants: The Pathways and Their Control*, 1998.

[26] G. Edwards and D. Walker. *C_3, C_4: Mechanisms, and Cellular and Environmental Regulation of Photosynthesis*, chapter 6. Blackwell Scientific Publications, first edition, 1983.

[27] G. Edwards and D. Walker. *C_3, C_4: Mechanisms, and Cellular and Environmental Regulation of Photosynthesis*, chapter 8. Blackwell Scientific Publications, first edition, 1983.

[28] G. Edwards and D. Walker. *C_3, C_4: Mechanisms, and Cellular and Environmental Regulation of Photosynthesis*, chapter 9. Blackwell Scientific Publications, first edition, 1983.

[29] I. R. Elrifi, J. J. Holmes, H. G. Weger, W. P. Mayo, and D. H. Turpin. RuBP limitation of photosynthetic carbon fixation during NH_3 assimilation. *Plant Physiol.*, 87:395–401, 1988.

[30] D. Fell. *Understanding the Control of Metabolism*, chapter 1. Portland Press, London, 1997.

[31] D. Fell. *Understanding the Control of Metabolism*, chapter 4. Portland Press, London, 1997.

[32] D. Fell. *Understanding the Control of Metabolism*, chapter 5. Portland Press, London, 1997.

[33] D. Fell and H. Sauro. Metabolic control and its analysis additional relationships between elasticities and coefficients. *Eur. J. Biochem.*, 148:555–561, 1985.

[34] A. Fersht. *Enzyme Structure and Mechanism.* W. H. Freeman, New York, U.S.A., 2^{nd} edition, 1984.

[35] K. Fichtner, W. P. Quick, E. D. Shulze, H. A. Mooney, S. R. Rodermel, L. Bogorad, and M. Stitt. Decreased rubisco in transgenic tobaco transformed with "antisense" *rbcs* v. relationship between photosynthetic rate, storage strategy, biomass allocation and vegetative plant growth at three different nitrogen supplies. *Planta*, 190:1–9, 1993.

[36] U.-I. Flügge. Phosphate translocation in the regulation of photosynthesis. *J. Exp. Bot.*, 46:1317–1323, 1995.

[37] U.-I. Flügge, M. Freisl, and H.-W. Heldt. Balance between metabolite accumulation and transport in relation to photosynthesis by isolated spinach chloroplasts. *Plant Physiol.*, 65:574–577, 1980.

[38] U.-I. Flügge and H.-W. Heldt. Metabolite translocators of the choroplast envelope. *Annu. Rev. Plant Physiol. Mol. Biol.*, 42:129–144, 1991.

[39] R. G. Fray, A. Wallace, P. D. Fraser, D. Valero, P. Hedden, P. M. Bramley, and D. Grierson. Constitutive expression of a fruit phytoene synthase gene in transgenic tomatoes causes dwarfism by redirecting metabolites from the gibberelin pathway. *Plant J.*, 8:693–701, 1995.

[40] L. E. Fridlyand, J. E. Backhausan, and R. Scheibe. Flux control of the malate valve in leaf cells. *Arch. Biochem. Biophys.*, 349:290–298, 1998.

[41] D. R. Geiger and J. C. Servaites. Diurnal regulation of photosynthetic carbon metabolism in C_3 plants. *Annu. Rev. Plant Physiol. Plant Mol. Biol.*, 45:235–256, 1994.

[42] R. Gerhardt, M. Stitt, and H.-W. Heldt. Subcellular metabolite levels in spinach leaves. *Plant Physiol.*, 83:399–407, 1987.

[43] C. Giersch. Photsynthetic oscillations: Observations and models. *Comments on Theoretical Biology*, 3:339–364, 1994.

[44] C. Giersch, M. N. Sivak, and D. A. Walker. A mathematical skeleton model of photosynthetic oscillations. *Proc. R. Soc. Lond. B*, 245:77–83, 1991.

[45] A. Goldbeter. Models for oscillations and excitability in biochemical systems. In L. A. Segel, editor, *Mathematical Models in Molecular and Cellular Biology.* Cambridge University Press, Cambridge, England, 1980.

[46] A. Goldbeter. *Biochemical Oscillations and Cellular Rhythms.* Cambridge University Press, Cambridge, England, 1996.

[47] J. Gosling, B. Joy, and G. Steele. *The Java*$^{\text{tm}}$ *Language Specification.* Addison-Wesley, Harlow, 1996.

[48] J. C. Gray, M. J. Paul, S. A. Barnes, J. S. Knight, A. Loynes, D. Habash, M. A. J. Parry, and D. W. Lawlor. Manipulation of phosphoribulokinase and phosphate translocator activities in transgenic tobacco plants. *J. Exp. Bot.*, 46:1309–1315, 1995.

[49] R. Hafner, G. Brown, and M. Brand. Analysis of the control of respiration rate, phosphorylation rate, proton leak rate and protonmotive force in isolated mitochondria using the 'top down' approach of metabolic control theory. *Eur. J. Biochem.*, 188:131–319, 1990.

[50] B. Hahn. Photosynthesis and photorespiration: Modelling the essentials. *J. theor. Biol*, 151:123–139, 1991.

[51] B. D. Hahn. A mathematical model of leaf carbon metabolism. *Annals of Botany*, 54:325–339, 1984.

[52] B. D. Hahn. A mathematical model of the calvin cycle: Analysis of the steady state. *Annals of Botany*, 57:639–653, 1986.

[53] B. D. Hahn. A mathematical model of photorespiration and photosynthesis. *Annals of Botany*, 60:157–169, 1987.

[54] I. Hanning and H.-W. Heldt. On the function of mitochondrial metabolism in spinach (*spinacia oleracea* l.) leaves. *Plant Physiol*, 103:1147–1154, 1993.

[55] E. P. Harrison, N. M. Williams, J. C. Lloyd, and C. A. Raines. Reduced sedoheptulose-1,7-bisphosphatase levels in transgenic tobacco lead to decreased photsynthetic capacity and altered carbohydrate accumulation. *Planta*, 204:27–36, 1998.

[56] D. Heineke, B. Riens, H. Große, P. Hofereichter, U. Peter, U.-I. Flügge, and H.-W. Heldt. Redox transfer across the inner chloroplast envelope membrane. *Plant Physiol.*, 95:1131–1137, 1991.

[57] R. Heinrich and T. A. Rapoport. A linear steady–state treatment of enzymatic chains; general properties, control and effector strength. *Eur. J. Biochem.*, 42:89–95, 1974.

[58] R. Heinrich and S. Schuster. *The Regulation of Cellular Systems*, chapter 3. Chapman & Hall, London, England, 1996. Basis for elem modes.

[59] H. W. Heldt, J. C. Chon, D. Maronde, A. Herold, Z. Stankovic, D. Walker, A. Kraminer, M. Kirk, and U. Heber. Role of orthophosphate and other factors in the regulation of starch formation in leaves and isolated chloroplasts. *Plant. Physiol.*, 59:1146–1155, 1977.

[60] G. Heßman. Pastex distribution and documentation. World Wide Web http://www.tex.ac.uk/tex-archive/systems/amiga/.

[61] F. Hoffmeister and T. Back. Genetic algorithms and evolution strategies: Similarities and differences. *Lecture Notes in Comp. Sci.*, 496:455–469, 1991.

[62] J. Hofmeyr and A. Cornish-Bowden. Quantitative assesment of regulation in metabolic systems. *Eur. J. Biochem.*, 200:223–236, 1991.

[63] H. Kacser and J. Burns. The control of flux. *Symp. Soc. Exp. Biol.*, 27:65–104, 1973.

[64] H. Kacser, J. A. Burns, and D. A. Fell. The control of flux. *Biochem. Soc. Trans.*, 23:341–366, 1995.

[65] C. Kelley, T. Williams, and C. Grammes. Gnuplot distribution and documentation. World Wide Web http://www.hens.ac.uk/mirrors/Slackware/contents/gnuplot.

[66] B. Kernighan and D. Ritchie. *The C Programming Language.* Prentice Hall, Englewood Cliffs, New Jersey, U.S.A, 2^{nd} edition, 1988.

[67] J. Kossman, U. Sonnewald, and L. Willmitzer. Reduction of the chloroplastic fructose-1,6-bisphosphatase in transgeneic potato plants impairs photosynthesis and plant growth. *Plant J.*, 6:637–650, 1994.

[68] A. Krapp, B. Hofmann, C. Schafer, and M. Stitt. Regulation of the expression of *rbcs* and other photosynthetic genes by carbohydrates: a mechanism for the 'sink regulation' of photosysnthesis ? *Plant J.*, 3(6):817–828, 1993.

[69] A. Krapp, P. Quick, and M. Stitt. Ribulose-1,5-bisphosphate carboxylase-oxygenase, other calvin cycle enzymes, and chlorophyll decrease when glucose is supplied to mature spinach leaves via the transpiration stream. *Planta*, 186:58–69, 1991.

[70] H. A. Krebs. Control of metabolic processes. *Endeavour*, 16:125–132, 1957.

[71] S. Kromer. Respiration during photosynthesis. *Annu. Rev. Plant. Physiol. Plant Mol. Biol.*, 46:45–70, 1995.

[72] L. Kruckeberg, H. Ekkehard, H. Neuhaus, R. Feil, L. Gottleib, and M. Stitt. Decreased-activity mutants of phosphoglucose isomerase in the cytosol and chloroplast of *clarkia xantiana*. *Biochem. J.*, 261:457–467, 1989.

[73] N. J. Kruger and P. Scott. Integration of cytosolic and plastidic carbon metabolism by fructose-2,6-bisphospphate. *J. Exp. Bot.*, 46:1325–1333, 1995.

[74] P. Kuzmic. Program dynafit for the analysis of enzyme kinetic data: Application to HIV proteinase. *Analytical Biochemistry*, 237:260–273, 1996.

[75] A. Laisk, H. Eichelmann, A. Eatheall, and D. A. Walker. A mathematical model of carbon metabolism in photosynthesis: Difficulties in explaining oscillations by Fructose 2,6-bisphosphate regulation. *Proc. R. Soc. Lond. B*, 237:389–415, 1989.

[76] A. Laisk and V. Oja. Potential rate of photosynthesis is determined by ribulose bisphosphate resynthesis reactions. *Proc. Estonian Acad. Sci. (Ser. Biol.)*, 25:146–150, 1976.

[77] A. Laisk and D. Walker. Control of phosphate turnover as a rate-limiting factor and possible cause of oscillations in photosynthesis: A mathematical model. *Proc. R. Soc. Lond. B*, 227:281–302, 1986.

[78] A. Laisk and D. A. Walker. A mathematical model of electron transport. thermodynamic neccessity for PSII regulation - "light stomata". *Proc. R. Soc. Lond. Ser. B ?? DAF review copy ??*, !!!!!!:!!!!!!, !!!!!!

[79] L. Lamport. *LaTeX User's Guide & Reference Manual.* Addison-Wesley, Reading, MA, U.S.A, 1^{st} edition, 1988.

[80] D. W. Lawlor. *Photosynthesis: Metabolism, Control, and Physiology*, chapter 7, page 141. Longman, New York, first edition, 1987.

[81] D. W. Lawlor. *Photosynthesis: Metabolism, Control, and Physiology*, chapter 11. Longman, New York, first edition, 1987.

[82] A. Lehninger, D. Nelson, and N. Cox. *Principles of biochemistry*, chapter 8. Wiley, New York, 1993.

[83] D. Levine. Users guide to the pgapack parallel genetic algorithm library. Technical report, Argonne National Laboratory, 1996.

[84] J. Levine, T. Mason, and D. Brown. *lex & yacc.* O'Reilly & Associates, 2^{nd} edition, 1992.

[85] H. R. Matthews, R. Freedland, and R. L. Miesfeld. *Biochemistry: A Short Course*, chapter 12. John Wiley, Bognor Regis, 1997.

[86] K. Matthews and K. van Holde. *Biochemistry*, chapter 17. Benjamin/Cummings, Menlo Park, CA, U.S.A., 2^{nd} edition, 1995.

[87] H. E. Neuhaus and N. Schulte. Starch degradation in chloroplasts isolated from c_3 or cam induced *mesembrium crytallinum* l. *Biochem. J.*, 1996.

[88] P. Niemeyer and J. Peck. *Exploring Java.* O Reilly, Sebastopol, CA, U.S.A., 1996.

[89] G. M. Odell. A qualtitative theory of systems of systems of ordinary differential equations. In L. A. Segel, editor, *Mathematical Models in Molecular and Cellular Biology*, pages 649–728. Cambridge University Press, 1980.

[90] V. Oja and A. Laisk. Adaptation of leaf photosynthetic aperatus to light profile in the leaf. *Fiziol. Rast.*, 23:445–451, 1976. Russian Language Publication.

[91] G. Pettersson and U. Ryde-Pettersson. A mathematical model of the Calvin photosynthesis cycle. *Eur. J. Biochem.*, 175:661–672, 1988.

[92] G. Pettersson and U. Ryde-Pettersson. Model studies of the regulation of the Calvin photosynthesis cycle by cytosolic metabolites. *Biochim. Biomed. Acta*, 40:723–732, 1990.

[93] A. R. Portis, Jr. Effects of the relative extrachloroplastic concentrations of inorganic phosphate, 3-phosphoglycerate, and dihydroxyacetone phosphate on the rate of starch synthesis in isolated spinach chloroplasts. *Plant Physiol.*, 70:393–396, 1982.

[94] J. Preiss. Regulation of the biosynthesis and degradtion of starch in higher plants. *Ann. Rev. Plant Physiol.*, 33, 1982.

[95] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipies in C*, chapter 9. Cambridge University Press, Cambridge, 1989. Newts method.

[96] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipies in C*, chapter 11. Cambridge University Press, Cambridge, 1989.

[97] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipies in C*, chapter 12. Cambridge University Press, Cambridge, 1989.

[98] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipies in C*, chapter 10. Cambridge University Press, Cambridge, 1989.

[99] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipies in C*, chapter 14. Cambridge University Press, Cambridge, England, 1989.

[100] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipies in C*, chapter 13. Cambridge University Press, Cambridge, 1989. Newts method.

[101] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipies in C*, chapter 6. Cambridge University Press, Cambridge, 1989. Newts method.

[102] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipies in C*, chapter 15. Cambridge University Press, Cambridge, England, 2^{nd} edition, 1995.

[103] G. D. Price, J. R. Evans, S. von Caemmerer, J. Yu, and M. R. Badger. Specific reduction of chloroplast glyceraldehyde-3-dehydrogenase activity by antisense rna reduces CO_2 assimilation via a reduction in ribulose bisphosphate regeneration in transgenic tobacco plants. *Planta*, 195:369–378, 1995.

[104] W. Quick, U. Schurr, R. Scheibe, E. Shulze, S. Rodermel, L. Bogorad, and M. Stitt. Decreased rubisco in transgenic tobaco transformed with "antisense" rbcs - i impact on photosynthesis in ambient growth conditions. *Planta*, 183:542–554, 1991.

[105] K. Raner. Maccurvefit distribution and documentation. World Wide Web http://mic8.hensa.ac.uk/cgi-bin/browser/mirrors/info-mac/sci/mac-curve-fit.

[106] C. Reder. Metabolic control theory: a structural approach. *J. Theor. Biol.*, 135:175–201, 1988.

[107] J. W. Riesmeier, U.-I. Flügge, B. Schulz, D. Heineke, H.-W. Heldt, L. Willmitzer, and W. B. Frommer. Antisense represion of the chloroplast triose phospphate translocator affects carbon partitioning in transgenic potato plants. *Proc. Natl. Acad. Sci. USA*, 90:6160–6164, 1993.

[108] W. Rovers and C. Giersch. Photosynthetic oscillations and the interdependence of photophosphorylation and electron transport as studied by a mathematical model. *BioSystems*, 35:63–73, 1995.

[109] U. Ryde-Pettersson. Identification of possible two reactant sources of oscillations in Calvin photosynthesis cycle and anciliary pathways. *Eur. J. Biochem.*, 198:613–619, 1991.

[110] U. Ryde-Pettersson. On the mechanistic origin of damped oscillations in biochemical reaction systems. *Eur. J. Biochem.*, 194:431–436, 1991.

[111] F. B. Salisbury and C. W. Ross. *Plant Physiology*, chapter 12. Wadsworth, Belmont, CA, 4th edition, 1992.

[112] F. B. Salisbury and C. W. Ross. *Plant Physiology*, chapter 25. Wadsworth, Belmont, CA, 4th edition, 1992.

[113] F. B. Salisbury and C. W. Ross. *Plant Physiology*, chapter 12. Wadsworth, Belmont, CA, 4th edition, 1992.

[114] F. B. Salisbury and C. W. Ross. *Plant Physiology*, chapter 13. Wadsworth, Belmont, CA, 4th edition, 1992.

[115] H. Sauro. Scamp distribution and documentation. Anonymous ftp - ftp://bmsdarwin.brookes.ac.uk/pub/software/ibmpc/scamp.

[116] H. Sauro. SCAMP: a general-purpose metabolic simulator and metabolic control analysis program. *CABIOS*, 9(4):441–450, 1993.

[117] H. M. Sauro and D. A. Fell. Analysers and simulators: a brief description of two computer programs SCAMP and iMAP. In H. V. Westerhoff, editor, *Bio-ThermoKinetics,*, pages 213–223. Intercept, Andover., 1994.

[118] H. M. Sauro, J. R. Small, and D. A. Fell. Metabolic control and its analysis: Extensions to the theory and matrix method. *Eur. J. Biochem.*, 165:215–221, 1987.

[119] M. A. Savageau. *Biochemical Systems Analysis: a Study of Function and Design in Molecular Biology.* Addison–Wesley, Reading, MA, U.S.A., 1976.

[120] C. Schäfer, H. Simper, and B. Hofman. Glucose feeding results in coordinated changes of chlorophyll content, ribulose-1,5-bisphosphate carxylase-oxygenase activity and photosynthetic potential in photoautotrophic suspension cultured cells of *chenopodium rubrum. Plant, Cell and Environment*, 15:343–350, 1992.

[121] W. Schellenberger, J. Frenzal, and K. Eschrich. Irreversible transitions in the 6-phosphofructo-1-kinase/fructose-1,6-bisphosphotase cycle in cell free extracts of rat liver. In H. Westerhoff, J. Snoep, F. Sluse, J. Wijker, and B. Kholedenko, editors, *Biothermokinetics of the living cell.*, pages 356–360. BioThermoKinetics Press, Amsterdam, 1996.

[122] R. Schiebe. Redox-modulation of choroplast enzymes. *Plant Physiol.*, 96:1–3, 1991.

[123] D. Schimkat, D. Heineke, and H.-W. Heldt. Regulation of sedoheptulose-1,7-bisphosphatase by sedoheptulose7-phosphate and glycerate, and of fructose-1,6-bisphosphatase by glycerate in spinach chloroplasts. *Planta*, 181:97–103, 1990.

[124] C. Schnarrenberger, A. Flechner, and W. Martin. Evidence for a complete oxidative pentose phosphate pathway in chloroplasts and an incomplete pathway in the cytosol of spinach leaves. *Plant Physiol.*, 108:609–614, 1995.

[125] S. Schuster, C. Hilgetag, J. H. Woods, and D. A. Fell. Elementary modes of functioning in biochemical networks. In R. Cuthbertson, M. Holcombe, and R. Paton, editors, *Computation in Cellular and Molecular Biological Systems*, pages 151–165. World Scientific Publishing Co. Pte. Ltd., 1996.

[126] J.-M. Soulié, J. Buc, J. Meunier, J. pradel, and J. Ricard. !!!! find out the title of this paper !!!! *Eur. J. Biochem.*, 119:497–502, 1981.

[127] M. Stitt. Fructose-2,6-bisphosphate as a regulatory molecule in plants. *Annu. Rev. Plant Physiol. Plant Mol. Biol.*, 41:153–185, 1990.

[128] M. Stitt and H.-W. Heldt. Physiological rates of starch breakdown in isolated intact spinach chloroplasts. *Plant Physiol*, 68:455–761, 1981.

[129] M. Stitt, W. P. Quick, U. Schurr, R. Scheibe, E. Shulze, S. Rodermel, and L. Bogorad. Decreased rubisco in transgenic tobaco transformed with "antisense" *rbcs* ii flux control coefficients for photosynthesis in varying light, CO_2, and air humidity. *Planta*, 183:555–566, 1991.

[130] L. Stryer. *Biochemistry*, chapter 19. W. H. Freeman, New York, U.S.A., 3^{rd} edition, 1988.

[131] B. Teusnik, M. C. Walsh, K. van Dam, and H. V. Westerhoff. The danger of metabolic pathways with turbo design. *TIBS*, 23:162 – 168, 1998.

[132] J. M. T. Thomson and H. B. Stewart. *Nonlinear Dynamics and Chaos*, chapter 2. John Wiley and Sons, 1988.

[133] R. Trethewey and T. ap-Rees. A mutant of *arabidopsis thaliana* lacking the ability to transport glucose across the chloroplast envelope. *Biochem. J.*, 301:449–454, 1994.

[134] D. Walker, N. Sivak, R. Prinsley, and CheesbroughJ.K. Simultaneous measurement of oscillations in oxygen evolution and chlorophyll a fluorescence in leaf pieces. *Plant. Physiol.*, 73:542–549, 1983.

[135] D. A. Walker. Manipulating photosynthetic metabolism to improve crops: an inversion of means and ends. *J. Exp. Bot.*, 46:1253–1259, 1995. Special Issue.

[136] D. Whitley. A genetic algorithm tutorial. Technical report, Colorado State University, 1993.

[137] C. P. Whittingham. *The Mechanism of Photosynthesis*, chapter 1. Edward Arnold, London, 1974.

[138] I. Woodrow. Control of the rate of photosynthetic carbon dioxide fixation. *Biochim. Biophy. Acta*, 851:181–192, 1986.

[139] I. E. Woodrow and J. A. Berry. Enzymatic regulation of photosynthetic CO_2 fixation in c_3 plants. *Ann. Rev. Plant Physiol. Plant Mol. Biol.*, 39:533–594, 1988.

[140] J. Woods. Empath distribution. Anonymous ftp - ftp://bmsdarwin.brookes.ac.uk/pub/software/ibmpc/empath.

[141] C. Zoglowek, S. Kromer, and H.-W. Heldt. Oxaloacetate and malate transport by plant mitochondria. *Plant Physiol.*, 87:109–115, 1988.

[142] G. Zubay. *Biochemistry*, chapter 9. Wm. C. Brown, Oxford, 3^{rd} edition, 1993.

[143] G. Zubay. *Biochemistry*, chapter 24. Wm. C. Brown, Oxford, 3^{rd} edition, 1993.

Appendix A

SCAMP/Scampi implementation of the Calvin cycle model

This appendix first presents the SCAMP command file representing the Petterrsons' original description of the Calvin cycle model, and illustrates the development of a "C" program using this with Scampi to determine model response to changes in a parameter.

A.1 Model definition

A.1.1 Naming conventions

Names of reactions and metabolites are as given in Tables and 4.1 and 4.2, with the following differences:

- All species have either the suffix "`_ch`" to indicate chloroplastic, or "`_cyt`", cytoplasmic, species.

- Identifiers relating to the triose phosphate translocator a prefixed "`TP_Piap`" (triose phosphate - P_i antiporter).

The rate constant for the rapid equilibrium reactions K, in equation 4.1 has the identifier "EQMult" and the PGA export modulating constant described in section 4.3, "PGA_xpMult".

A.1.2 The command file

```
Title Calvin1 - Petterssons original + Starch pase;

options auto_conserve ;
    # auto detect conserved cycles

options integrator = lsoda ; # needed for SCAMP, ignored by Scampi ;
Simulate ;                   # ditto ;


    # declare fixed and variable metabolites ;

dec ATP_ch, ADP_ch,  RuBP_ch, PGA_ch, BPGA_ch, $NADPH_ch,
    GAP_ch, DHAP_ch, FBP_ch, F6P_ch, E4P_ch, SBP_ch, S7P_ch,
    G6P_ch,R5P_ch, Ru5P_ch, X5P_ch, G1P_ch, Pi_ch, $NADP_ch,
    $CO2, $Pi_cyt, $PGA_cyt, $DHAP_cyt, $GAP_cyt, $Starch_ch,
    $Proton_ch  ;

reactions

[Rubisco]
    $CO2 + RuBP_ch = 2PGA_ch ;
    Rbco_vm * RuBP_ch/(RuBP_ch + Rbco_km*(1 +
        PGA_ch/Rbco_KiPGA +
        FBP_ch/Rbco_KiFBP +
        SBP_ch/Rbco_KiSBP +
        Pi_ch /Rbco_KiPi  +
        NADPH_ch/Rbco_KiNADPH )) ;

[PhosphoglycerateKinase]
    PGA_ch + ATP_ch = BPGA_ch + ADP_ch  ;
    EQMult  *(PGA_ch * ATP_ch - (BPGA_ch * ADP_ch / q2)) ;

[G3P_dehydrogenase]
    BPGA_ch + $NADPH_ch + $Proton_ch = $NADP_ch + GAP_ch + Pi_ch ;
    EQMult  * (BPGA_ch * NADPH_ch * Proton_ch -
        NADP_ch * GAP_ch * Pi_ch/q3)  ;

[TPI]
    GAP_ch = DHAP_ch ;
    EQMult  *(GAP_ch - DHAP_ch/q4) ;

[Aldolase]
    DHAP_ch + GAP_ch =  FBP_ch  ;
    EQMult  *(DHAP_ch * GAP_ch - FBP_ch/q5) ;

[FBPase]
    FBP_ch = F6P_ch + Pi_ch ;
    FBPase_ch_vm * FBP_ch /(FBP_ch + FBPase_ch_km*( 1 +
        F6P_ch/FBPase_ch_KiF6P +
        Pi_ch/FBPase_ch_KiPi )) ;
```

```
                                                # starch synthesis branch ;
[PGI_ch]
    F6P_ch = G6P_ch ;
    EQMult  *(F6P_ch - G6P_ch /q14) ;

[PGM_ch]
    G6P_ch = G1P_ch ;
    EQMult  *( G6P_ch - G1P_ch/q15) ;

[St_synthase]
    G1P_ch + ATP_ch = ADP_ch + 2Pi_ch + $Starch_ch ;
    Vstsyn_ch * G1P_ch * ATP_ch /
    (( G1P_ch + stsyn_ch_km1) *
    ( 1 + ADP_ch / stsyn_ch_Ki) *
    (ATP_ch + stsyn_ch_km2) +
    stsyn_ch_km2 * Pi_ch /
    (stsyn_ch_ka1 * PGA_ch) +
    (stsyn_ch_ka2 * F6P_ch) +
    (stsyn_ch_ka3 * FBP_ch) ) ;

                        # Starch Pase ;
[StPase]
$Starch_ch + Pi_ch = G1P_ch ;
    StPase_Vm * Pi_ch /(Pi_ch + StPase_km * ( 1 + G1P_ch/StPase_kiG1P)) ;

                                                # end of starch metabolism ;
[transketolase]
    F6P_ch + GAP_ch = E4P_ch + X5P_ch ;
    EQMult  *( F6P_ch * GAP_ch - E4P_ch * X5P_ch / q7 ) ;

[Aldolase2]
    E4P_ch + DHAP_ch = SBP_ch  ;
    EQMult  *(E4P_ch * DHAP_ch - SBP_ch/q8) ;

[seduheptuloseBPase]
    SBP_ch = S7P_ch + Pi_ch ;
    SBPase_ch_vm * SBP_ch/(SBP_ch + SBPase_ch_km*(1 +
    Pi_ch/SBPase_ch_KiPi)) ;

[transketolase2]
    GAP_ch + S7P_ch = X5P_ch + R5P_ch  ;
    EQMult  *(GAP_ch * S7P_ch - X5P_ch * R5P_ch /q10) ;

[R5Pisomerase]
    R5P_ch = Ru5P_ch ;
    EQMult  * (R5P_ch - Ru5P_ch /q11);

[X5Pisomerase]
    X5P_ch = Ru5P_ch ;
    EQMult  *(X5P_ch -  Ru5P_ch/q12);

[Ru5Pkinase]
Ru5P_ch + ATP_ch = RuBP_ch + ADP_ch ;
Ru5Pk_ch_vm * Ru5P_ch * ATP_ch/( (
    Ru5P_ch + Ru5Pk_ch_km1 * (1 +
      PGA_ch / Ru5Pk_ch_KiPGA +
      RuBP_ch / Ru5Pk_ch_KiRuBP +
      Pi_ch / Ru5Pk_ch_KiPi )     )
    *(
```

```
      ATP_ch *( 1 + ADP_ch / Ru5Pk_ch_KiADP1) +
      Ru5Pk_ch_km2 * ( 1 + ADP_ch / Ru5Pk_ch_KiADP2))) ;


                        #TP/Pi antiport 1 PGA ;
[TP_Pi_apPGA]
PGA_ch + $Pi_cyt = Pi_ch + $PGA_cyt ;
    PGA_xpMult * TP_Piap_vm * PGA_ch / (TP_Piap_kPGA_ch * ( 1 +
    ( 1 + TP_Piap_kPi_cyt / Pi_cyt) *
    ( Pi_ch / TP_Piap_kPi_ch +
    PGA_ch / TP_Piap_kPGA_ch  +
    DHAP_ch / TP_Piap_kDHAP_ch +
    GAP_ch / TP_Piap_kGAP_ch ))) ;

                        #       2 GAP ;
[TP_Pi_apGAP]
GAP_ch + $Pi_cyt = Pi_ch + $GAP_cyt ;
    TP_Piap_vm * GAP_ch / (TP_Piap_kGAP_ch * ( 1 +
    ( 1 + TP_Piap_kPi_cyt / Pi_cyt) *
    ( Pi_ch / TP_Piap_kPi_ch +
    PGA_ch / TP_Piap_kPGA_ch  +
    DHAP_ch / TP_Piap_kDHAP_ch +
    GAP_ch / TP_Piap_kGAP_ch))) ;

                        #       3 DHAP   ;
[TP_Pi_apDHAP]
DHAP_ch + $Pi_cyt = Pi_ch + $DHAP_cyt ;
    TP_Piap_vm * DHAP_ch / (TP_Piap_kDHAP_ch * ( 1 +
    ( 1 + TP_Piap_kPi_cyt / Pi_cyt) *
    ( Pi_ch / TP_Piap_kPi_ch +
    PGA_ch / TP_Piap_kPGA_ch  +
    DHAP_ch / TP_Piap_kDHAP_ch +
    GAP_ch / TP_Piap_kGAP_ch))) ;


[Light_react]                    # Light reactions  ;
ADP_ch + Pi_ch = ATP_ch ;
    LR_vm * ADP_ch * Pi_ch/((ADP_ch + LR_kmADP)*(Pi_ch + LR_kmPi)) ;

EOR ;


        ############### Initialise ################ ;

initialise
                            #  kinetic parameters ;
                            # rubisco ;
Rbco_vm = 340 ; Rbco_km = 0.02  ; Rbco_KiFBP = 0.04 ; Rbco_KiPGA = 0.84 ;
 Rbco_KiSBP = 0.075 ; Rbco_KiPi = 0.9 ; Rbco_KiNADPH = 0.07 ;

                            # FBPase ;
FBPase_ch_vm = 200 ; FBPase_ch_km = 0.03 ;  FBPase_ch_KiF6P = 0.7 ;
FBPase_ch_KiPi = 12 ;

                            # SBPase ;
SBPase_ch_vm = 40 ; SBPase_ch_km = 0.013 ; SBPase_ch_KiPi = 12 ;

                            # Ru5Pkinase ;
Ru5Pk_ch_vm = 10000 ; Ru5Pk_ch_km1 = 0.05 ; Ru5Pk_ch_km2 = 0.05 ;
Ru5Pk_ch_KiPGA = 2 ; Ru5Pk_ch_KiRuBP  = 0.7 ; Ru5Pk_ch_KiPi = 4 ;
```

```
Ru5Pk_ch_KiADP1 = 2.5 ; Ru5Pk_ch_KiADP2 = 0.4 ;

                              # TP/Pi ap ;
TP_Piap_vm = 250 ; TP_Piap_kPGA_ch = 0.25 ;
TP_Piap_kGAP_ch = 0.075 ; TP_Piap_kDHAP_ch = 0.077 ;
TP_Piap_kPi_ch =  0.63 ; TP_Piap_kPi_cyt = 0.74 ;

                              # "starch synthase" ;
Vstsyn_ch = 40 ; stsyn_ch_km1 = 0.08 ; stsyn_ch_km2 = 0.08 ; stsyn_ch_Ki = 10 ;
stsyn_ch_ka1 = 0.1 ; stsyn_ch_ka2 = 0.02 ; stsyn_ch_ka3 = 0.02 ;

                              # StPase ;
StPase_Vm = 40  ; StPase_km = 0.1 ;  StPase_kiG1P = 0.05 ;

                              # Light Reactions ;
LR_vm = 3500 ;  LR_kmADP = 0.014 ;  LR_kmPi = 0.3 ;

                              # equil reactions ;
q2 = 3.1E-4 ; q3 = 1.6E7 ; q4 = 22 ; q5 = 7.1; q7 = 0.084 ;
q8 = 13 ; q10 = 0.85 ; q11 = 0.4 ; q12 = 0.67 ; q14 = 2.3 ; q15 = 0.0580 ;

                              # control variables ;
EQMult = 5e8 ; PGA_xpMult = 0.75 ;

                              # initial metaolite values ;
PGA_ch = 3.35479 ; BPGA_ch = 0.14825 ; GAP_ch = 0.01334 ; DHAP_ch = 0.29345 ;
G1P_ch = 0.18206 ; G6P_ch = 3.1396   ; FBP_ch = 0.02776 ; F6P_ch = 1.36481 ;
SBP_ch = 1.56486 ; S7P_ch = 0.00541  ; E4P_ch = 0.41021 ; X5P_ch = 0.00363 ;
R5P_ch = 0.00599 ; Ru5P_ch = 0.00235 ; RuBP_ch = 0.33644;  Pi_ch = 1.5662 ;

                              # initial parameters values ;
Pi_cyt = 0.5 ; DHAP_cyt = 1 ; GAP_cyt = 1 ; PGA_cyt = 1 ; CO2 = 1.0 ;
Proton_ch = 2.512E-5 ; Starch_ch = 1 ; ATP_ch = 0.49806 ; ADP_ch = 0.00149 ;
NADPH_ch = 0.21 ; NADP_ch = 0.29 ;
ei ;


timeend = 2 ;      # remainder needed by SCAMP, ignored by Scampi ;
sim_points = 200 ;
print_sim TIME, [TP_Pi_apDHAP] ;
end ;
```

A.2 Use of Scampi to determine effect of parameter change on $P_{i_{ext}}$ overload point

A.2.1 Problem definition / program specification

In chapter 4 it was shown that the Calvin cycle model, in the absence of starch phosphorlyase, was subject to irrevocable breakdown at high levels of $P_{i_{ext}}$, and that this level was itself appeared to dependent on various parameters. In order to investigate the effect that such parameters may have, it is first neccessary to define a function

to determine the breakdown point. Given an initial pair of bracketing values (`Lo` and `Hi`), this may be readily achieved using a recursive bisection search, as sketched out in Pseudo-code fragment A.1:

Code Fragment A.1 *Recursive bisection search for maximimum sustainable parameter value (Pseudo-code)*

```
1     MidPoint = (Lo + Hi)/2
2     IF (Lo and Hi are nearly equal) THEN
3          breakdown point = MidPoint
      ELSE
4          determine steady state at MidPoint
      ENDIF
5     IF( steady state is viable)
6          search between MidPoint and Hi
      ELSE
7          revive the model
8          search between Lo and MidPoint
      ENDIF
```

In order to write this as a function in a programming language, the user will need to supply some additional information:

- The model under consideration.
- The name of the parameter
- An indication of how close Lo and Hi must be to be "`nearly equal`" in step 2
- A set of initial viable concentrations with which to revive the model in step 6

A.2.2 Implementation

Taking these points into consideration, and assuming the Scampi tools described in chapter 2 are available, it is possible to specify a "C" function `FindMax()`:

```
double FindMax( ScampiModel_t md, char *PName, double Pmax, double Pmin,
                double Pequiv, double *RestoreConcs) ;
    /* pre: md has a parameter called PName,
            Pmin and Pmax bracket the maximum sustainable value of PName,
            0 < Pequiv < 1,
            RestoreConcs = GetMDVec(md, Conc, ) with viable md
      post: Maximum sustainable value of PName is returned with
            relative error +/- (1-Pequiv)/(1+Pequiv) **/
```

With this specification, and using Scampi tools for steady state determinations etc. the pseudo code of A.1 may be translated to "C" without further refinement:

```
  double    Pmid,       /* mid parameter value  */
            rv,         /* return value         */
            duration ;  /* initial sim duration */
  int       sim_points ;/* initial sim points   */
  enum SPI_err err ;    /* scampi error return  */

    duration = 3.0 ; sim_points = 300 ;
    Pmid = (Pmin+Pmax)/2.0 ;                          /* calc mid point */

    if((Pmin/Pmax) >Pequiv)  /* have we reached requested  resolution ? */
        rv = Pmid ;                        /* yes,  we will return Pmid */
    else{
                                        /* no, we have to search again */
        PutMDval(md, Param, PName, Pmid) ;     /* set parameter to Pmid */
        Sim_to_SS(calvin0, &duration, &sim_points, 1e-6, 20, 3, &err) ;
                                                    /* and try for SS */
        if(err == OK)                              /* Good steady state ? */
                                  /* yes, search between Pmid and Pmax */
            rv = FindMax(md, PName, Pmax, Pmid, Pequiv, RestoreConcs) ;
        else{
                                 /* no, limit is between Pmin and Pmid  */
            PutMDvec(md, Conc, RestoreConcs) ;
                                           /* so restore known good concs  */
            rv = FindMax(md, PName, Pmid, Pmin, Pequiv, RestoreConcs) ;
                                                   /* and search there */
        }
    }
    return (rv) ;    /*** Single entry, single exit !,  :-)   ***/
}
```

With this function defined, it becomes a trivial matter to write a `main()` function to using `FindMax()` to determine the breakdown point, and take care of necessary i/o and related matters, as shown overleaf:

```
int main(int argc, char *argv[]){

  char fname[80] ;           /* file name */
  FILE *fp ;                 /* file for output */
  double PGA_xpMult,         /* modulate Vmax of TP-Pi antiport to PGA */
         Pi,                 /* external Pi, */
         MaxPi,              /* and max sustainable external, Pi */
         duration ;          /* treat as magic - or ask the author */
  int   simul_points ;       /* ditto */
  const double
    Xp_hi = 1.25, Xp_lo = 0.1,/* hi and lo limits for PGA_xpMult */
    Pi_hi = 5, Pi_lo = 0.1,  /* and for external Pi */
    discrim = 0.99 ;         /* tol. for  Pi (a/b > dicrim =>  a == b ) */
  double Xp_inc,             /* increment for PGA_xpMult */
         *RescueConcs ;      /* known viable starting concentrations */
  const int n_res = 20 ;     /* number of results we will record */
  int n ;                    /* gp counter */
  enum SPI_err err ;         /* hold error returns from Scampi */

    strcpy(fname, argv[0]) ;              /* sort out a file for output */
    strcat(fname, ".out") ;
    fp = fopen(fname, "w") ;
    fprintf(fp, "PGA_xpMult\tMaxPi\n") ;

    duration = 2.0 ; simul_points = 200 ;       /*  initial external Pi */
    PutMDval(calvin0, Param, "Pi_cyt", Pi_lo) ;
    Sim_to_SS(calvin0, &duration, &simul_points, 1e-6, 20, 3, &err) ;
                                                    /* get initial SS */
    RescueConcs = GetMDvec(calvin0, Conc, &err) ;
                           /* get our own copy of concentration vals */
    Xp_inc = (Xp_hi - Xp_lo)/(double) n_res ;    /* calc increment size */

    PGA_xpMult = Xp_lo ;
    for(n = 0 ; n <= n_res ; n++, PGA_xpMult += Xp_inc){
                                  /* loop through vals of PGA_xpMult */
        PutMDval(calvin0, Param, "PGA_xpMult", PGA_xpMult) ;
                                 /* update model with new PGA_xpMult */
        PutMDval(calvin0, Param, "Pi_cyt", Pi_hi) ;/* set ext Pi to max */

        duration = 2.0 ; simul_points = 200 ;         /* try for new SS */
        Sim_to_SS(calvin0, &duration, &simul_points, 1e-6, 20, 3, &err) ;

        if(err == OK)             /* if model with MaxPi is sustainable */
            MaxPi = Pi_hi ;                              /* don't search */
        else{                                                 /* if not  */
            PutMDvec(calvin0, Conc, RescueConcs) ;
                                          /* ressurect the Calvin cycle */
            MaxPi = FindMax(calvin0, "Pi_cyt", Pi_hi, Pi_lo,
                        discrim, RescueConcs) ;
        }                             /* and search for the max sustainable */
        fprintf(fp, "%e\t%e\n", PGA_xpMult, MaxPi) ;
    }                                    /* record results for posterity */
    free(RescueConcs) ;                       /* collect our own garbage */
    exit(0) ;                                     /* thats all, folks !  */
}
```

A.2.3 Compilation

As with other programs using Scampi described in this thesis, the process of maintaining up to date executables may be conveniently automated using the standard Unix "make" utility (also availble for other platforms). A typical makefile as used in this project is as follows:

```
MODEL = calvin1
# "MODEL" is the name of we wish to investigate
# calvin1 is the Calvin cycle model with starch degradation

CC = gcc
CC_Flags = -m68020 -m68881 -O2
# "go faster" flags for 68020 architectures   e.g. Amiga   ****

LD_FLAGS =  -lscampi -llists -levol -lmath2
# All Scampi progs _MUST_ link with the scampi, lists, evol, and math2 libs

$(TOP): $(TOP).o $(MODEL).o
    $(CC)  -o $(TOP) $(TOP).o $(MODEL).o $(LD_FLAGS)
# "TOP" is the name of the target executable, passed at the command line

$(MODEL).o: $(MODEL).c $(MODEL).h
    $(CC) $(CC_Flags) -c  $(MODEL).c
# compile the C representation of the model

$(MODEL).c: $(MODEL).lst
    spicg $(MODEL).lst
# generate the C representation from the .lst file

$(MODEL).lst: $(MODEL).cmd
    scamp $(MODEL).cmd
# generate the .lst file from the .cmd file

$(TOP).o: $(TOP).c $(MODEL).h
    $(CC) $(CC_Flags) -c  $(TOP).c
# compile the TOP source code
```

As with the rest of Scampi, this makefile is readily portable, the only machine dependent line is that marked "`****` relating to different processor-specific optimisation flags for the compiler. Assuming that the source code file containing the `main()` function definition of section A.2.2 resides in the file `MaxPi.c`, the corresponding executable, `MaxPi` is updated by the command `make TOP=MaxPi`.

Appendix B

Interface to the Evolve library

```
/***
    Evolve.h - interface to ES structures and funtions
***/

#ifndef EVOLVE_H
#define EVOLVE_H

typedef enum {FitnessHigh, FitnessLow} Fitness_t ;
typedef enum {FixMute, VarMute } StratSet_t ;

typedef struct OrgStruct {
    double *Genome ;
    double *MuteSize ;
    int LenGenome,
        n_mute ;
    StratSet_t Strat ;
    Fitness_t FitHiOrLo ;
    double FitnessVal ;
    void *user ;
} OrganismStruct, *Organism_t;

typedef void (*FitnessEval_ft)(Organism_t) ;
                                    /* fitness evaluation function type */
extern Organism_t NewOrg(int LenGen, Fitness_t ft) ;
    /* pre : TRUE
      post : returns valid Organism_t with genome of length LenGen
             (and undefined contents) and fitness type ft
             (returns NULL if fail) */

extern Organism_t NewOrg2(int LenGen, double *Adam, double MuteSize,
                    Fitness_t ft) ;
    /* pre : Adam[LenGen]
      post : as NewOrg but Genome is copy of Adam and
        return->DeltaSigma[n] = Adam[n] * MuteSize ;
      */

extern Organism_t NewOrg3(int LenGen, double *Adam, double MuteSize,
                    Fitness_t ft, StratSet_t strat) ;
```

```
    /* pre : Adam[LenGen]
      post : as NewOrg2 + return->Strat = strat ; */

extern Organism_t CloneOrg(Organism_t Orig) ;
    /* pre : Orig is valid
      post : returns exact copy of Orig else NULL if fail */

extern void CopyOrg(Organism_t src, Organism_t dst) ;
    /* pre : src and dst valid and have equal length genome
      post dst is a copy of src */

extern void RefreshOrg(Organism_t Org, double DeltaSigma) ;
    /* pre : Org = NewOrg*, DeltaSigma > 0.0 ;
      post : Mutation vector reinitialised, Mutation counter = 0 */

extern void KillOrg(Organism_t Victim) ;
    /* pre : Victim valid
      post : Victim's fitness val set to dead val according to fitness type */

extern int /* bool */IsDeadOrg(Organism_t org) ;
    /* pre : org valid
      post : TRUE => org's fitness value == dead value
             FALSE => org's fitness value != dead value */

extern void DestroyOrg(Organism_t Victim) ;
    /* pre : Victim is valid
      post : Victim is not valid, associated memory freed */

extern void DestroyPop(Organism_t *Pop, int n_org) ;
    /* pre : Pop[0..n_org-1] valid
      post : DestroyOrg(Pop[0..n_org-1]) */


extern  Organism_t *Evolve(Organism_t Adam, int PopSize, int n_Survivors,
                           int n_gens, FitnessEval_ft FitEval,
                           double MuteSize, double MuteRate) ;
  /* pre : Adam is valid, 0 < n_Survivors <= PopSize, MuteSize > 0.0,
           0 < MuteRate <= 1.0
    post : returns pointer to fitest Organism
           found using the (mu + lambda)ES algorithm */

extern  Organism_t *Evolve2(Organism_t *Seeds, int n_Seeds, int PopSize,
                            int n_Survivors, int n_gens,
                            FitnessEval_ft FitEval, double MuteSize,
                            double MuteRate) ;
    /* pre : seeds is valid Organism_t[n_Seeds], 0 < n_Seeds < PopSize,
             0 < n_Survivors <= PopSize ;
       post : returns n_Survivors fittest organisms found using
                (mu + lambda)ES algorithm */

extern  Organism_t *Evolve3(Organism_t *Seeds, int n_Seeds, int PopSize,
                            int n_Survivors, int n_gens,
                            FitnessEval_ft FitEval, double DeltaSigma,
                            double MuteRate) ;
    /* pre : seeds is valid Organism_t[n_Seeds], 0 < n_Seeds < PopSize,
             0 < n_Survivors <= PopSize ;
             0.0 < MuteRate <= 1.0, DeltaSigma > 0.0 ;
       post : returns n_Survivors fittest organisms found using
                (mu + lambda)ES algorithm */
```

```
void Evolve3Static(Organism_t *Population, int PopSize, int n_Survivors,
                   int Initialise, int n_gens, FitnessEval_ft FitEval,
                   double DeltaSigma, double MuteRate) ;
    /* pre : Population[0..PopSize-1] = NewOrg(), n_Survivors < PopSize,
             0.0 < MuteRate <= 1.0, DeltaSigma > 0.0 ;
      post : performs n_gens of mu+lamda ES to Population,
             (Initialise == TRUE) => Population[1..PopSize-1] initialised
             from Population[0]  (which will count as a generation) */


void EvolveToTarg(Organism_t *Population, int PopSize, int n_Survivors,
                  int n_gens, FitnessEval_ft FitEval, double DeltaSigma,
                  double MuteRate, double F_Targ) ;
    /* pre : Population[0..PopSize-1] = NewOrg(),
             n_Survivors < PopSize, 0.0 < MuteRate <= 1.0, DeltaSigma > 0.0
      post : performs <= n_gens of mu+lamda ES to Population,
             stops as soon as organism as fit or fitter than F_Targ is found,
             which will be Population[0] ;*/
#endif
```

Appendix C

Analysis of skeleton model C_2

Denoting the metabolites RuBP, TP, and P_i as A, B, and C respectively, and disregarding external metabolites, skeleton model C_2 of chapter 6 can be represented as shown in Figure C.1.

Taking into account the stoichiometries of the Calvin cycle , assuming that all reactions have first order kinetics with respected to any individual metabolite, and that with the exception of the starch phosphorylase reaction V_4, reactions are irreversible, the model of Figure C.1 are represented:

$$A \longrightarrow 2B \quad V_1 = K_1 A \tag{C.1}$$

$$5B + C \longrightarrow 3A \quad V_2 = K_2 BC \tag{C.2}$$

$$B \longrightarrow C \quad V_3 = K_3 B \tag{C.3}$$

$$B \rightleftharpoons C \quad V_4 = K_1(B - \tfrac{C}{q_4}) \tag{C.4}$$

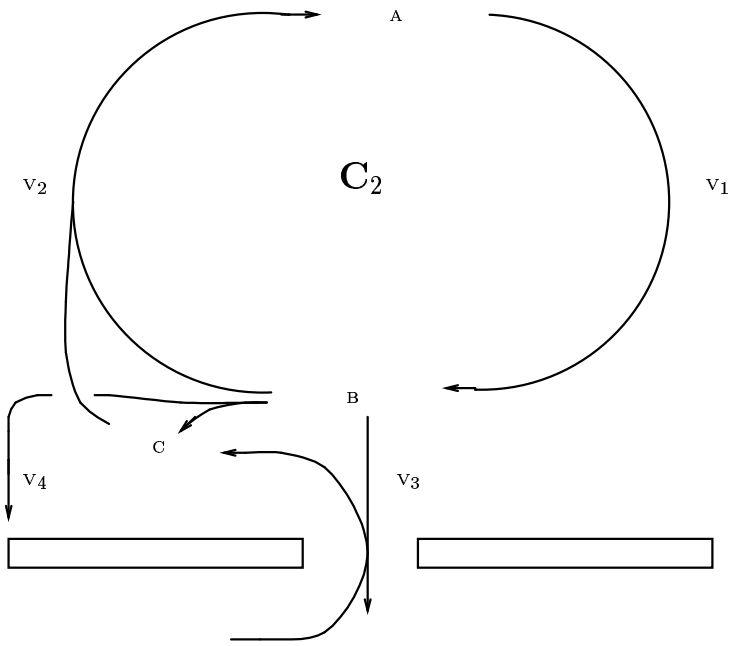

Figure C.1: Metabolites and reactions of the skeleton model C_2 as considered here.

Denoting $\partial x/\partial t$ as x' a set of differential equations is obtained:

$$A' = 3V_2 - V_1 \tag{C.5}$$

$$B' = 2V_1 - 5V_2 - V_3 - V_4 \tag{C.6}$$

$$C' = V_3 + V_4 - V_2 \tag{C.7}$$

but eqn C.6 = 2× eqn C.5 - eqn C.7, therefore to make the system soluble, the conservation relationship is required:

$$2A + B + C = S$$

where S is the conserved sum. Thus

$$A = \frac{S - B - C}{2} \tag{C.8}$$

At steady state eqns C.5...C.7 equate to zero. By substituting eqns C.1, C.2 and C.8

into C.5:

$$3K_2BC = \frac{K_1(S - B - C)}{2}$$

From which C may be obtained:

$$C = \frac{\left(\frac{S}{B} - 1\right)}{6K_2 + \frac{K_1}{B}} \tag{C.9}$$

By Substituting eqns C.2... C.4 into eqn C.7 we obtain:

$$K_3B + K_4\left(B - \frac{C}{q_4}\right) = K_2BC \tag{C.10}$$

Substituting for C from eqn C.9 into eqn C.10 :

$$K_3B + K_4\left(B - \frac{K_1\left(\frac{S}{B} - 1\right)}{q_4\left(6K_2 + \frac{K_1}{B}\right)}\right) = \frac{K_2BK_1\left(\frac{S}{B} - 1\right)}{6K_2 + \frac{K_1}{B}}$$

Which, after expanding brackets, multiplying out denominators, and simplyfying yields:

$$(6K_2(K_3 + K_4) + K_1K_2)B^2 + \left(K_1(K_3 + K_4) + \frac{K_1K_4}{q_4}\right)B - K_1S\left(\frac{K_4}{q_4} + K_2\right) = 0 \tag{C.11}$$

All parameters are positive, hence terms in B^2 and B^1 are positive and B^0 negative, thus only one positive solution can exist.

Unfortunately, the increase in complexity brought about by the introduction of non-linear terms into equations C.1... C.4 is quite spectacular. Even the simplest non-linear assumption, replacing equation C.1 with Michaelis-Menten kinetics, results in a set of equations from which, beyond observing that the solution is at least third order, it has not yet been possible to extract useful information.