Symbolic control analysis of cellular systems

Timothy John Akhurst

Dissertation presented for the degree of Doctor of Philosophy
in the Faculty of Science (Biochemistry)
at Stellenbosch University

Promoter: Prof J.M. Rohwer
Co-Promoter: Prof J.-H.S. Hofmeyr
March 2011

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the authorship owner thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: Date:

Abstract

Metabolic Control Analysis (MCA) provides a powerful quantitative framework for understanding and explaining the control and regulation within a cellular system. MCA allows the global control of a steady-state system to be quantified in terms of control coefficients, which we can express in terms of the local properties referred to as elasticity coefficients. MCA relates elasticities to control coefficients through a matrix inversion, thus allowing scientists to predict and quantify how the kinetics of the individual enzymes affect the systemic behaviour of cellular systems. Traditionally we solved this problem numerically, while we used algebraic and symbolic control analysis techniques less frequently. By using symbolic algebraic computation we present a general implementation of the symbolic matrix inversion of MCA, known as SymCA, which requires only the description of any allosteric modifier interactions and the stoichiometry of a cellular system. The algebraic expressions generated allow an in-depth analysis of the distribution of the control within a system and also of the parameters which exhibit the greatest effect on this control distribution. This also applies when the exact values for the elasticities or control coefficients are unknown. We have demonstrated that by quantifying the control patterns, referred to as 'routes of regulation', inherent in all control coefficient expressions, we can gain insight into how perturbations are propagated through a cellular system and which regulatory pathways are favoured under changing conditions.

Opsomming

Metaboliese Kontrole-Analise (MKA) bied 'n kragtige kwantitatiewe raamwerk om die beheer en regulering binne sellulêre sisteme te verstaan en te verduidelik. 'n Sleutelaspek van MKA is dat die globale beheer van 'n sisteem met 'n bestendige toestand gekwantifiseer kan word in terme van kontrole-koëffisiënte en dat hierdie koeffisiënte uitgedruk kan word in terme van die sisteem se lokale eienskappe, genaamd elastisiteitskoëffisiënte. Deur van matriksinversie gebruik te maak kan MKA die verband tussen elastisiteitskoëffisiënte en kontrole-koëffisiënte aflei wat mens in staat stel om te sien hoe die kinetika van die individuele ensiemreaksies die sisteemgedrag op sellulêre vlak beïnvloed. Dié probleem word tradisioneel hoofsaaklik op numeriese wyse bereken terwyl die gebruik van algebraïese en simboliese kontrole-analise minder gereeld gebruik word. In hierdie proefskrif verskaf ons, deur van simboliese algebraïese metodes gebruik te maak, 'n generiese implementasie van die simboliese matriksinversie van MKA, genaamd SymCA, wat slegs 'n beskrywing van 'n sellulêre sisteem se allosetriese interaksies en die stoichiometrie benodig. Die algebraïese uitdrukkings sodanig gegenereer stel mens in staat om 'n in-diepte analise te doen om vas te stel waar die beheer binne 'n sisteem lê, asook watter parameters die grootste effek op die kontrole-verspreiding het. Dit geld selfs in die geval waar die presiese waardes van die elastisiteitskoëffisiënte of kontrole-koëffisiënte onbekend is. Hierdie proefskrif demonstreer hoe die kwantifisering van kontrole-patrone, ook gesien as 'roetes van regulering', wat inherent is aan kontrole-koëffisiënt vergelykings, mens in staat stel om te sien hoe perturbasies in 'n sellulêre sisteem voortplant en watter regulatoriese paaie bevoordeel word onder veranderde kondisies.

Acknowledgements

The completion of this thesis would not have been possible without the influence of many people to whom I will always be grateful. The enthusiasm and passion for the subject shown by my supervisor and co-supervisor have been instrumental during my PhD. I would like to thank a few key people who have contributed to the success in my doctoral studies.

My funders, the National Bioinformatics Network, as without their funding none of this would have been possible.

My supervisor, Johann Rohwer, whose encouragement, guidance, patience and support has been unwavering from the onset.

My co-supervisor, Jannie Hofmeyr, whose support, encouragement and guidance has been instrumental throughout my PhD studies.

My colleagues, Riaan Conradie, Lafras Uys, James Dominy, Athlee Maclear and Brett Olivier for creating a challenging and enthusiastic climate in which to work. I would like to extend special thanks to Brett Olivier for his open-door policy, willingness to be a sounding board, encouragement and most important, our many fruitful discussions.

My parents, for the encouragement and moral support I needed the closer the finishing line became. I would also like to thank them for the sacrifices they made throughout my life to enable me to be where I am today.

The Kantor family, who offered us a quiet place to stay when the arduous task of writing-up began, and for the support and encouragement they have constantly provided.

A special thanks to the two ladies in my life, my wife Nina Carstens, and Jessica, my daughter. I will always be grateful for the sacrifices Nina has made throughout my postgraduate studies, and for her patience and unwavering support. Jessica kept me company for many hours in the study. I will always remember the sound of her footsteps early in the morning as she made her way downstairs to keep me company.

Contents

1 Introduction **1**
- 1.1 The emergence of Systems Biology . . . 1
- 1.2 Metabolism – an introduction . . . 2
- 1.3 Metabolic Control Analysis . . . 4
  - 1.3.1 Local properties . . . 5
  - 1.3.2 Global properties . . . 7
- 1.4 Control properties of MCA . . . 9
  - 1.4.1 Summation Theorem . . . 10
  - 1.4.2 Connectivity Theorem . . . 10
- 1.5 Equations of MCA . . . 11
  - 1.5.1 The control-matrix equation . . . 11
  - 1.5.2 The inverse problem . . . 12
  - 1.5.3 Control-matrix example using a simple model . . . 13
- 1.6 A brief introduction to metabolic regulation . . . 15
- 1.7 Computational simulation and MCA . . . 17
- 1.8 Symbolic computation . . . 19
  - 1.8.1 Symbolic computation and MCA . . . 20
- 1.9 Thesis outline . . . 21

2 SymCA: Proof of concept **23**
- 2.1 Introduction . . . 23
- 2.2 Software requirements . . . 24
  - 2.2.1 Maxima . . . 24
  - 2.2.2 PySCeS . . . 24
  - 2.2.3 Python . . . 24
- 2.3 The humble beginnings of SymCA . . . 25
  - 2.3.1 Maxima interface . . . 26
  - 2.3.2 Symbolic metabolic control analysis . . . 27
- 2.4 Discussion . . . 37

3 Development of SymCA **40**
- 3.1 Introduction . . . 40
- 3.2 SymCA overview . . . 41

3.3 Basic requirements . . . 42
3.4 SymCA classes . . . 43
3.4.1 symca . . . 43
3.4.2 Data . . . 48
3.4.3 PyscesData . . . 49
3.4.4 ProcessedData . . . 50
3.4.5 SymbolicData . . . 51
3.4.6 SymcaObjects . . . 51
3.4.7 SCA . . . 55
3.4.8 Maxima . . . 56
3.4.9 Parser . . . 57
3.4.10 Output . . . 57
3.4.11 Utilities . . . 58
3.4.12 Visualise . . . 58
3.5 Discussion . . . 59

4 SymCA at work **60**
4.1 Introduction . . . 60
4.2 Beginning a SymCA session . . . 60
4.3 Instantiating a model object . . . 62
4.3.1 Symbolic control analysis . . . 62
4.4 Post-symbolic control analysis features . . . 63
4.4.1 Symbolic to PySCeS comparison . . . 64
4.4.2 Summation theorems . . . 64
4.4.3 Elasticity substitution . . . 64
4.4.4 Co-control coefficients . . . 64
4.4.5 Response coefficients . . . 65
4.4.6 Control pattern quantification . . . 65
4.4.7 Post-analysis simplification . . . 66
4.4.8 Pattern scan . . . 66
4.4.9 Graphical representation of elasticity coefficient distribution . . . 66
4.4.10 Graphical representation of control patterns . . . 67
4.4.11 Data output . . . 68
4.5 LaTeX output as generated by SymCA . . . 69
4.5.1 Control coefficient output . . . 69
4.5.2 Co-control coefficient output . . . 71
4.5.3 SymCA vs PySCeS output . . . 72
4.5.4 Summation output . . . 74
4.5.5 Quantified control pattern output . . . 74
4.5.6 Control pattern parameter scan output . . . 75
4.5.7 Graphical representations . . . 77
4.6 Discussion . . . 80

5 Applications of SymCA: Part I **82**
5.1 General introduction . . . 82
5.1.1 Supply-demand analysis . . . 82
5.1.2 Control pattern quantification . . . 83
5.2 Control pattern quantification and supply-demand analysis, a combined strategy . . . 84
5.3 Discussion . . . 91

6 Applications of SymCA: Part II **94**
6.1 Introduction . . . 94
6.2 Regulatory analysis and SymCA . . . 96
6.3 Discussion . . . 105

7 Applications of SymCA: Part III **108**
7.1 Introduction . . . 108
7.2 Regulatory analysis and SymCA . . . 110
7.2.1 Internode 3 . . . 111
7.2.2 Internode 4 . . . 114
7.2.3 Internode 5 . . . 116
7.2.4 Internode 6 . . . 118
7.2.5 Internode 7 . . . 121
7.2.6 Internode 8 . . . 123
7.2.7 Internode 9 . . . 126
7.2.8 Internode 10 . . . 128
7.3 Discussion . . . 131

8 Discussion **137**
8.1 Synopsis . . . 137
8.2 Critical appraisal . . . 138
8.3 Future work and perspectives . . . 141
8.4 Conclusion . . . 143

Bibliography **144**

Appendix **154**

A Detailed description of SymCA classes **154**
A.1 Data . . . 154
A.2 PyscesData . . . 156
A.3 ProcessedData . . . 158
A.4 SymbolicData . . . 160
A.5 SymcaObjects . . . 163
A.6 SCA . . . 167
A.7 Maxima . . . 177

A.8 Parser 179
A.9 Output 180
A.10 Utilities 183
A.11 Visualise 189

List of Figures

1.1 A generic overview of metabolism, adapted from Fell [37]. Of importance is that the complex molecules produced by anabolism are not necessarily the same ones that are broken down by catabolism. 3
1.2 A schematic representation of a simple chemical reaction where a substrate (S) is converted to a product (P) in the presence of an enzyme (1). 6
1.3 A schematic representation of a simple linear pathway consisting of three enzymes (1,2,3), a source (X_0), a sink (X_2), an external effector (X_1) and two variables (metabolites S_1 and S_2). 7
1.4 A schematic representation of a simple metabolic pathway consisting of a branched flux and a moiety-conserved cycle [51, 53]. 13
1.5 A four-enzyme linear pathway with both a feedforward and a feedback loop, as illustrated by Hofmeyr [49]. 16

2.1 Theoretical system used as the initial test model for development of `SymCA`. 28

3.1 Overview of SymCA classes 41
3.2 SymCA workflow 42
3.3 Data classes 49
3.4 PyscesData classes 50
3.5 ProcessedData classes 50
3.6 SymbolicData classes 51
3.7 SCA classes 56
3.8 Maxima classes 56

4.1 Theoretical system to demonstrate `SymCA`, as previously used to describe the control-matrix equation in Section 1.5.3. 60
4.2 Control pattern parameter scan output. 76
4.3 Control coefficient parameter scan output. 77
4.4 The base model generated by `SymCA` 78
4.5 Variable elasticity distribution image 78
4.6 Variable and parameter distribution image 79
4.7 A control pattern image for a pattern found in $C_{R3}^{J_{R6}}$ 79

5.1 A simple representation of the functional organisation of metabolism. . . . 83
5.2 Linear pathway with a feedback loop where X serves as the linking metabolite between the supply block, which consists of three steps, and the demand block, which has one step. 84
5.3 Supply-demand rate characteristic plot for the model shown in Figure 5.2. The values found in the figure legend correspond to the Vmax values for the demand step (Reaction 4). 86
5.4 C. P. 1 found in C_{R4}^{JR4} at steady-state 87
5.5 C. P. 2 found in C_{R4}^{JR4} at steady-state 88
5.6 Control pattern contributions for parameter scan V4 on C_{R4}^{JR4} 89
5.7 Control pattern contributions for parameter scan V4 on C_{R4}^{JR4} over a range from ≈ 1.8 to ≈ 2.2 showing the dramatic flipping of control between the two control patterns. 90
5.8 Flux-control coefficient values for a V4-parameter scan. 91

6.1 Reaction scheme of the kinetic model of fermentation pathways in *Saccharomyces cerevisiae*. Reactions are numbered 1 to 8 and are denoted by squares, metabolites are denoted by circles and any feedback loops are shown as dashed lines. A number enclosed in a box next to a line indicates a stoichiometric coefficient. 95
6.2 Dominant control pattern for C_{v1}^{Jv6} for suspended cells at pH 4.5 & pH 5.5. The perturbed reaction is denoted by a blue square, the elasticities found in the pattern are shown in red with a grey bubble surrounding them and the reaction they affect. All fluxes present in the control pattern are shown as red reaction numbers and the dashed lines indicate feedback loops. 101
6.3 Dominant control pattern for C_{v4}^{Jv6} for immobilised cells at pH 4.5. The perturbed reaction is denoted by a blue square, the elasticities found in the pattern are shown in orange with a grey bubble surrounding them and the reaction they affect. All fluxes present in the control pattern are represented by an orange reaction number and the dashed lines show feedback loops. 103
6.4 Dominant control pattern for C_{v1}^{Jv6} for immobilised cells at pH 5.5. The perturbed reaction is denoted by a blue square, the elasticities found in the pattern are shown in yellow with a grey bubble surrounding them and the reaction they affect. All fluxes present in the control pattern are represented by an yellow reaction number and the dashed lines show feedback loops. 104

7.1 Extended model of sucrose accumulation as illustrated in Uys *et al.* [116]. Variable metabolites are shown in light grey shaded rectangular boxes and fixed in light grey shaded circles. Enzymes are shown in unshaded rectangular boxes. A number 2 next to a line indicates a stoichiometric coefficient. *Abbreviations for enzymes and transport steps:* ALD, Aldolase; FRK, Fructokinase; HK, Hexokinase; NI, Neutral Invertase; PFK, Phosphofructokinase; PFP, Pyrophosphate-dependent PFK; SPase, Sucrose phosphatase; SPS, Sucrose phosphate synthetase; SuSy, Sucrose synthase ; UDPGDH, UDP-Glucose dehydrogenase; VAC, vacuolar sucrose import. *Metabolites:* FbP, Fructose-1,6-bisphosphate; Fru, Fructose; Glc, Glucose; HexP, Hexose phosphate; S6P, Sucrose-6-phosphate; Suc, Sucrose; Trp, triose phosphate; UDPGA, UDP-Glucuronic acid. 109

7.2 Dominant control pattern for $C^{JVAC}_{FRU\_uptake}$ in Internode 3. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations:* 1, GLC_uptake; 2, FRKa; 3, FRKb; 4, VAC; 5, FRU_uptake; 6, Spase; 7, NI; 8, PFK; 9, SPS; 10, SuSya; 11, PFP; 12, SuSyc; 13, SuSyb; 14, UDPGDH; 15, HK_FRU; 16, HK_GLC; 17, ADL. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines. 113

7.3 Control pattern illustrating pathway via NI for $C^{JVAC}_{FRU\_uptake}$ in Internode 4. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines. 115

7.4 Major contributing control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 5. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines. 117

7.5 Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 6. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines. 118

7.6 Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 7. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines. 121

7.7 Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 8. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines. 123
7.8 Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 9. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines. 128
7.9 Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 10. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines. 129

List of Tables

2.1 Independent flux control coefficients generated by `SymCA`. 32
2.2 Independent concentration control coefficients generated by `SymCA`. 33
2.3 Numerical values for independent control coefficients generated numerically with `PySCeS` and via substitution of expressions from `SymCA`. 36

3.1 Description of ControlCoefficient object attributes 52
3.2 Description of CoControlCoefficient object attributes 53
3.3 Denominator object attributes 54
3.4 SBMLelasticity object attributes 59
3.5 SBMLreaction object attributes 59

4.1 Control patterns index 76

5.1 Kinetic parameters for all reactions for the model as shown in Figure 5.2. 85
5.2 Quantified control pattern data for flux control coefficient C_{R4}^{JR4} 87

6.1 Flux control coefficients for Reaction 6 computed by `PySCeS`. 97
6.2 Control patterns per control coefficients, only those with a contribution of over 5% were considered for this study. 97
6.3 Quantified control patterns for all control coefficients in suspended cells. . 100
6.4 Quantified control patterns for all control coefficients in immobilised cells. 102

7.1 Flux control coefficients, with a value greater than 0.1 for a single internode, on J_{VAC} computed by `PySCeS`, bold values indicate the key control coefficients for each internode. 111
7.2 Internode 3 control patterns for $C_{FRU\_uptake}^{JVAC}$ 111
7.3 Internode 4 control patterns for $C_{FRU\_uptake}^{JVAC}$ 114
7.4 Internode 5 control patterns for $C_{HK\_GLC}^{JVAC}$ 116
7.5 Internode 5 control patterns for C_{FRKa}^{JVAC} 116
7.6 Internode 6 control patterns for $C_{HK\_GLC}^{JVAC}$ 119
7.7 Internode 6 control patterns for C_{FRKa}^{JVAC} 119
7.8 Internode 6 control patterns for $C_{FRU\_uptake}^{JVAC}$ 120
7.9 Internode 7 control patterns for $C_{HK\_GLC}^{JVAC}$ 122
7.10 Internode 7 control patterns for C_{FRKa}^{JVAC} 122

7.11 Internode 8 control patterns for $C^{JVAC}_{HK\_GLC}$ . . . 124
7.12 Internode 8 control patterns for $C^{JVAC}_{FRU\_uptake}$ . . . 124
7.13 Internode 8 control patterns for C^{JVAC}_{FRKa} . . . 125
7.14 Internode 9 control patterns for $C^{JVAC}_{HK\_GLC}$ . . . 126
7.15 Internode 9 control patterns for $C^{JVAC}_{FRU\_uptake}$ . . . 127
7.16 Internode 10 control patterns for $C^{JVAC}_{FRU\_uptake}$ . . . 130
7.17 Internode 10 control patterns for $C^{JVAC}_{HK\_GLC}$ . . . 130
7.18 Control pattern contributions for patterns acting via NI. . . . 132
7.19 SuSya control pattern contributions for patterns leading to sucrose accumulation. . . . 133
7.20 SuSyb control pattern contributions for patterns leading to sucrose accumulation. . . . 133
7.21 SuSyc control pattern contributions for patterns leading to sucrose accumulation. . . . 134
7.22 Combined actual % contributions for all control patterns including the SuSy isoforms. . . . 135

A.1 Description of `coefficient` object attributes . . . 163
A.2 `Elasticity` class attributes . . . 163
A.3 `ElasticityExpression` class attributes . . . 164
A.4 `Flux` object attributes . . . 164
A.5 `Species` object attributes . . . 164
A.6 `Matrix` object attributes . . . 165
A.7 `ProcessedRecord` object attributes . . . 165
A.8 `SymbolicRecord` object attributes . . . 166
A.9 `Reaction` object attributes . . . 166
A.10 `PyscesRecord` object attributes . . . 167

Chapter 1

Introduction

1.1 The emergence of Systems Biology

During the 20th century, biology developed into a major interdisciplinary field as a result of numerous discoveries pushing it ahead. An in-depth look into the history of biology is beyond the scope of this study, but there are a few key events which we will highlight. One key event is the elucidation of the organisation of living cells with specific reference to the discovery of enzymes, metabolic pathways, the structure of DNA [120] and the determination of DNA sequences of the genomes of numerous organisms, including that of humans [22, 117]. Several key scientists are associated with the discovery of enzymes. In 1987 Eduard Buchner discovered that alcoholic fermentation could occur in cell-free yeast extracts [24, 29, 31]. In 1926 James Sumner successfully isolated and crystallised urease and proposed that all enzymes are proteins [108]. Leonor Michaelis and Maud Menten established the basic concepts of enzyme inhibition [29], and J.B.S. Haldane developed our current understanding of enzyme catalysis.

The enzyme research outlined above required the development of new techniques, including chromatography, metabolic gas manometry, spectrophotometry, stable and radioactive isotopes for use as tracers for intermediary metabolites. The techniques for purification of proteins and other macromolecules also developed and gave rise to the basic outlines of cell biochemistry [43]. The past biological achievements continue to have a tremendous impact on technology and our understanding of living organisms. Presently, there is a vast amount of knowledge associated with our understanding of nature, and this data increases daily. During the 21st century there has been an explosion in the amount of organism wide data as a result of the 'omics' fields such as genomics, proteomics and metabolomics. This data is stored in extensive databases such as the NIH genetic sequence database GenBank [15], SWISSPROT housing annotated protein sequences [19], the KEGG (Kyoto Encyclopedia of Genes and Genomes) database [69, 70, 71] and the protein data bank (PDB), which is a resource for studying biological macromolecules [17].

In addition to the aforementioned databases, the Human Genome Project has irreversibly altered the practice of biology for studying the elements of a system, either one or a few at a time [11]. As outlined by Hood [59], these changes include the realisation that biology is in essence an informational science requiring the use of computer science, mathematics and statistics to uncover biological complexity. These changes have led to a fresh approach to biology, which has been termed Systems Biology. Systems Biology aims to study the interrelationships of all the elements in a system as opposed to the traditional one-at-a-time approach [59]. Many scientists have suggested that the integration of molecular biology, genetics and cell biology into the interdisciplinary field of Systems Biology will enable biology to advance to the next level [11, 59, 62, 74, 125].

Specifically, metabolism is one field that we have studied traditionally as sets of chemical reactions catalysed by separate enzymes. This approach is unsatisfactory in the study of the nature of life, and to gain an understanding of metabolism, we must treat it as a system. Some scientists have suggested that if metabolism is not studied as a whole, the systemic aspects crucial for the optimal functioning of an organism may go unnoticed due to the fact that a living organism is responsible for making and maintaining itself and its components [32]. Metabolic Control Analysis (MCA) illustrates that the state of a cell is integral in determining the associated control distribution and that the control properties of specific processes concerned with systemic properties are invariably distributed over many processes [48, 65].

In its current form Systems Biology makes use of mathematical models of cellular pathways in an attempt to replicate their *in vivo* behaviour. The aim of these models is to gain a quantitative understanding about how we can explain the properties of biological systems in terms of the characteristics and interactions between their macromolecular components. To aid this process, Systems Biology makes use of computer models to integrate this information and that of molecular interactions. Scientists have proposed that if we link the kinetic models of parts of metabolism together, it may be possible to begin creating kinetic models of larger pieces of metabolism, with the ultimate goal being the creation of a single kinetic model depicting the entire living cell. This single kinetic model is referred to as the 'Silicon Cell' [111, 112, 121].

1.2 Metabolism – an introduction

For all organisms to maintain life, i.e. feed, grow and reproduce, a set of chemical reactions, commonly referred to as metabolism, is required. Metabolism constitutes the flow of molecules and energy through pathways of chemical reactions [38]. Metabolism, metabolic regulation and enzymology are some of the central areas of traditional biochemistry that we have generally regarded as solved. These metabolic processes are generally divided into two groups, anabolic and catabolic. The anabolic processes are involved with the utilisation of energy to generate complex molecules, such as nucleic acids and proteins, from simple ones. On the other hand, the catabolic processes break

down these complex molecules into simple ones with the release of energy, such as glycolysis in which glucose is broken down into pyruvate with the release of high energy compounds in the form of ATP and NADH.

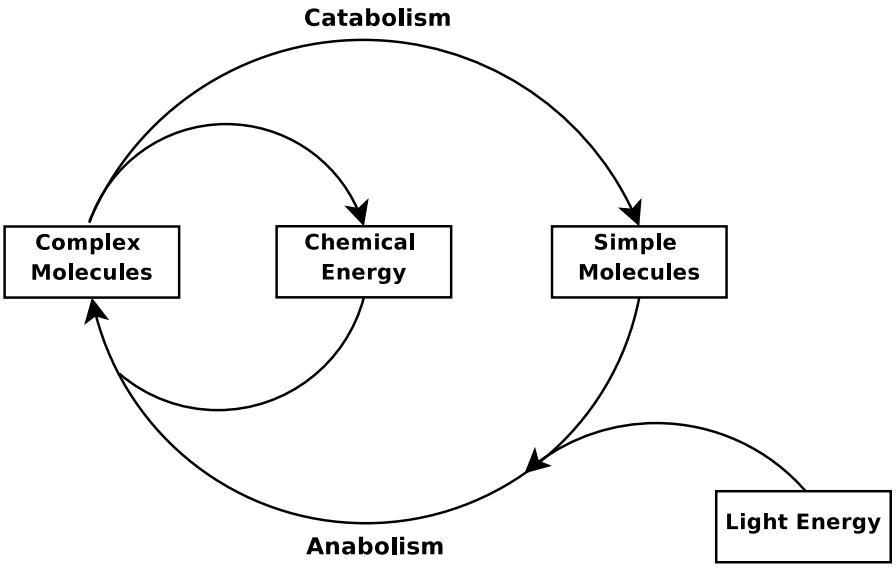

Figure 1.1: A generic overview of metabolism, adapted from Fell [37]. Of importance is that the complex molecules produced by anabolism are not necessarily the same ones that are broken down by catabolism.

Metabolic reactions are organised into sequences known as metabolic pathways in which the reactions are catalysed by enzymes and the product of one reaction serves as the substrate of the next reaction. The full extent of all metabolic pathways forms a complex network of patterns. Some patterns are linear pathways such as the synthesis of tryptophan from chorismate, other patterns are branched pathways, whilst others, such as the Krebs cycle, form closed cycles [78].

Recently problems concerning metabolic control and regulation were tackled either by identifying the molecular details of the underlying mechanisms or by formulating qualitative descriptions of the system's behaviour. Traditionally biochemists discovered pathways through isolating and characterising each step that aided the conversion of a defined substrate to a given product [112]. These traditional approaches led to the notion that for any given pathway there was a single controlling or 'rate-limiting' step; this assumption is still the norm in many current biochemistry textbooks [16, 26, 81]. The assumption gives rise to the implication that if a 'rate-limiting' step exists, then by varying that step alone, a change in the pathway flux will occur, and that by varying any of the other steps no change will occur. A prime example dispelling this notion is phosphofructokinase in yeast, which was believed to be the 'rate-limiting' enzyme of glycolysis. Experiments

have shown that a 3.5 fold increase in this enzyme activity had no significant effect on the anaerobic glycolytic flux [39].

A few researchers remained unconvinced about the notion of a 'rate-limiting' step, and believed that there were underlying flaws with the biological explanations given for metabolic control and regulation. This led to two schools of thought in the early 1970s. Savageau developed Biochemical Systems Theory as a means of quantitatively analysing metabolic behaviour [100]. At the same time two independent groups, Kacser and Burns [65], and Heinrich and Rapoport [48, 47] developed metabolic control analysis, which showed that there is seldom one 'rate-limiting' step, but in fact that the enzymes in a pathway tend to share the control of flux. These biochemists have since been vindicated by the dramatic and rapid advances in biochemistry, which have led to the emergence of fields such as genetic engineering and systems biology.

This thesis will focus on the framework of metabolic control analysis, and will use this framework as the basis for developing a symbolic (or algebraic) analysis in the form of a software package designed for use with `PySCeS` [84] to aid the investigation of metabolic regulation. `PySCeS` is a software package developed by our group for the purposes of performing metabolic control analysis on cellular systems.

1.3 Metabolic Control Analysis

Metabolic Control Analysis (MCA), is a powerful quantitative framework for analysing and quantifying the control and regulation of cellular pathways [37, 38, 79]. MCA enables the quantification of the steady-state properties of a system in terms of its global properties (referred to as control coefficients) and the local properties (referred to as elasticity coefficients). MCA equips biochemists with a robust mathematical and theoretical framework providing a means of quantifying the controls governing cellular processes [126].

Since its inception, a select group of researchers have further advanced, refined and expanded the theory and applications of MCA to formulate the present day theoretical body [126]. A number of formulisms of MCA exist [27, 28, 44, 45, 51, 87]. The work in this thesis stems directly from a dissertation by Hofmeyr [51], in which he presents the algebra underlying MCA. MCA is not limited to the study of metabolic systems, and a number of extensions have been presented to various types of systems, for example, hierarchical or multi-level systems [58, 67], modular systems [23, 92, 106], oscillating systems [18, 36], signal transduction pathways [73] and generalised supply-demand analysis [90].

In its most elementary form, MCA is concerned with the steady-state behaviour of systems of enzymes that link a series of metabolites. For these systems to attain a steady-state they require two or more reservoirs of metabolites with concentrations that

are fixed independently of the enzymes within the system, and are thus referred to as external. These reservoirs include a *source* from which metabolites flow and at least one area to which they flow, the *sink* [30].

The underlying theory of MCA is essentially sensitivity analysis, which is integral to a number of varied fields such as reaction kinetics, air pollution, weather forecasting and economics. Like MCA, all these systems consist of sets of coupled, nonlinear equations that can be in the form of differential, integral or algebraic equations. These systems may contain hundreds of equations, a large number of parameters and an equally large number of output and internal variables [33, 37]. Solutions for these large sets are only possible with the aid of computer simulation for which a number of dedicated packages exist, such as `PySCeS` [84], `Jarnac` [95] and `Copasi` [60]. Once a solution has been obtained, we are still faced with the question of how sensitive the solution is to variations of or inherent uncertainties in the parameters of the equation set.

This concept of sensitivity is central to our understanding of how systems behave. It is critical to know how sensitive output variables (such as fluxes or metabolite concentrations) are either to changes in or uncertainties in the parameters (such as enzyme concentration), and which variables are either sensitive or insensitive to which parameters [33]. The concept of sensitivity relates the magnitude of the effect of small change (perturbation) in a parameter on a metabolic system property and is mathematically related to the properties of the individual components of the system [37].

Metabolic control analysis revolves around the use of 'coefficients' in order to describe metabolic control. Traditionally these coefficients have been ratios of relative changes or fractional changes, which have the advantage of being dimensionless. Two types of coefficients were proposed; local, referring to the properties of the individual enzymes within a system, and global, concerned with the response of the system as a whole, i.e. the systemic effect.

These coefficients were defined with the aim of providing the quantitative means with which to describe the global properties of systems of reactions in terms of the individual catalytic steps [118]. This was achieved in the form of the control, elasticity and response coefficients, and with connectivity and summation theorems. The control and response coefficients describe the global properties of the system and the elasticity coefficients describe the local properties.

1.3.1 Local properties

Elasticity coefficients

The elasticity coefficient ('elasticity') is a major concept of MCA, and is used to describe the response of the isolated rate of an enzyme or transporter to a perturbation in the concentration of a substrate, product or effector. In other words, elasticities are

used to quantify the effect of some form of an effector on the rate of an isolated enzyme under locally specified conditions. The algebraic expression representing an elasticity is normally obtained by differentiation of its rate law, which is then multiplied by the effector concentration and divided by the rate law itself. This results in a dimensionless expression, which is generally expressed in terms of a partial derivative [127]. Elasticities are represented by an epsilon and can be demonstrated with a simple example.

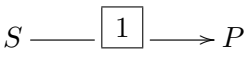

Figure 1.2: A schematic representation of a simple chemical reaction where a substrate (S) is converted to a product (P) in the presence of an enzyme (1).

For the system in Figure 1.2 there are three distinct elasticity coefficients. Two coefficients are concerned with the effects of S and P on the rate of the enzyme reaction, and the third coefficient is related to the effect of the enzyme concentration on the rate of the enzyme reaction. For example, if the substrate and enzyme concentrations are kept constant we can deduce the elasticity coefficient for the product in terms of the reaction rate by varying the product concentration around its steady-state value. This example is expressed by the equation below where p refers to the product and υ refers to the rate of the reaction catalysed by the enzyme labelled 1 in Figure 1.2.

$$\varepsilon_p^{\upsilon} = \left(\frac{\partial \upsilon/\upsilon}{\partial p/p}\right)_{s,e} \tag{1.1}$$

In the same manner we can obtain the elasticity coefficients for the substrate s and the enzyme concentration e and represent them by the following equations.

$$\varepsilon_s^{\upsilon} = \left(\frac{\partial \upsilon/\upsilon}{\partial s/s}\right)_{p,e} \tag{1.2}$$

$$\varepsilon_e^{\upsilon} = \left(\frac{\partial \upsilon/\upsilon}{\partial e/e}\right)_{s,p} \tag{1.3}$$

As mentioned earlier, once we know the rate of a reaction, it is possible to obtain an explicit equation for an elasticity coefficient. We can demonstrate this using the simple reaction in Figure 1.2. The rate can be expressed by the following equation, using reversible Michaelis-Menten kinetics.

$$\upsilon = \frac{\frac{Vf}{Ks}(s - \frac{p}{Keq})}{1 + \frac{s}{Ks} + \frac{p}{Kp}} \tag{1.4}$$

Partial differentiation with respect to s or p and subsequent scaling of the partial derivative with s/v and p/v respectively, results in the elasticity coefficients towards s and p:

$$\varepsilon_s^v = \frac{1}{1 - \frac{\Gamma}{Keq}} - \frac{\frac{s}{Ks}}{1 + \frac{s}{Ks} + \frac{p}{Kp}} \tag{1.5}$$

$$\varepsilon_P^v = \frac{\frac{-\Gamma}{Keq}}{1 - \frac{\Gamma}{Keq}} - \frac{\frac{p}{Ks}}{1 + \frac{s}{Ks} + \frac{p}{Kp}} \tag{1.6}$$

where Γ is defined as the mass action ratio, i.e. the ratio of the product concentrations to substrate concentrations each raised to the exponent equal to the stoichiometry of the species in the reaction, and K_{eq} is the equilibrium constant which is equal to Γ at equilibrium. The above example and equations are presented by Fell in [39].

1.3.2 Global properties

Control and response coefficients

Since all the parameters (such as enzyme concentrations) of a system are responsible for setting the steady-state of a system, it is intuitive that a change in any parameter will have a subsequent effect on the steady-state. However, the theory behind MCA revolves around the idea that all the steps (reactions) within a system contribute to the overall system behaviour. This presents the challenge on how a change in a parameter that effects a specific step (e.g. perturbation of an enzyme rate), can best describe the sensitivity of a steady-state variable towards this parameter change that effects the step directly.

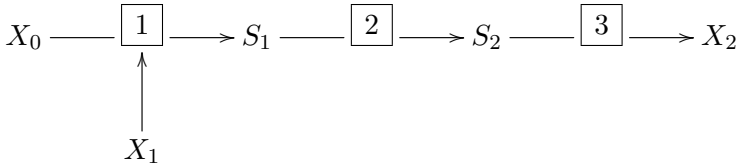

Figure 1.3: A schematic representation of a simple linear pathway consisting of three enzymes (1,2,3), a source (X_0), a sink (X_2), an external effector (X_1) and two variables (metabolites S_1 and S_2).

The global coefficients quantify these systemic responses in response to local perturbations. Control coefficients quantify how a change in an enzyme activity affects either a flux or a metabolite concentration, whereas response coefficients describe the effect of a change of an external parameter, such as the concentration of an extracellular component, on the fluxes and metabolite concentrations [118]. To fully understand the concepts for both response and control coefficients it is best to describe them in terms of a simple

thought experiment, as illustrated in [65].

Consider a steady-state flux J_1, which is the rate of the Step *1* indicated by $\boxed{1}$ at steady-state in Figure 1.3. In Figure 1.3 we can clearly see that X_1 only influences Step *1*, and for this explicit example we will consider it to be an external effector with a concentration of x_1. If a small change (δx_1) is made to parameter X_1, a whole sequence of events unfolds. The change in x_1 results in an increase in the rate of Step *1*, which in turn has an effect on the concentrations of its substrates and products. These changes affect the rates at which the metabolites interact, thus the small change made in x_1 spreads through the system by changing the metabolites linking the reaction network.

The change in Step *1* reverberates through the system, which proceeds through a transient state before the system once again reaches a steady-state. This steady-state will be different from the original steady-state since one of the parameters has changed. Thus the fractional change in X_1 ($\partial x_1/x_1$) results in a fractional change in the steady-state flux J_1 ($\partial J_1/J_1$). We have defined the ratio of these changes as the response coefficient, and it provides a quantitative means with which to describe how a parameter acting on a local step results in a systemic response, in this case a change in steady-state flux. We define a response coefficient mathematically as:

$$R_{x_1}^{J_1} = \frac{\partial J_1/J_1}{\partial x_1/x_1} \tag{1.7}$$

We now have a means of describing the response of a steady-state variable (i.e. J_1) to a perturbation in a parameter, but need to address a quantitative means of describing the response of a steady-state variable (a flux or metabolite concentration) to a change in the rate through a step. The elasticity coefficient provides a means of describing the effect of ∂x_1 on the local rate v_i:

$$\varepsilon_{x_1}^{v_i} = \frac{\partial v_i/v_i}{\partial x_1/x_1} \tag{1.8}$$

If we consider the ratio of the systemic effect of $\partial x_1/x_1$ on J_1 and the local effect of $\partial x_1/x_1$ on v_i we obtain the following expression:

$$\left(\frac{\partial J/J}{\partial x_1/x_1}\right) / \left(\frac{\partial v_i/v_i}{\partial x_1/x_1}\right) = \frac{R_{x_1}^{J_1}}{\varepsilon_{x_1}^{v_i}} \tag{1.9}$$

Since $\partial x_1/x_1$ is found in both the numerator and the denominator, we can cancel it from the expression resulting in:

$$\frac{\partial J_1/J_1}{\partial v_i/v_i} = \frac{R_{x_1}^{J_1}}{\varepsilon_{x_1}^{v_i}} \tag{1.10}$$

Now we can clearly see that the resultant expression is the ratio of the change in a steady-state variable to the change in the local rate of Step *i*. We commonly refer to this ratio as the control coefficient of Step *i*. The following expression represents the general case which applies for concentration and flux-control coefficients, where y represents a steady-state variable:

$$C_i^y = \frac{\partial y/y}{\partial v_i/v_i} \tag{1.11}$$

Therefore there are both flux and concentration control coefficients. From Equations 1.10 and 1.11 we can see that:

$$C_i^y = \frac{R_x^y}{\varepsilon_x^{v_i}} \quad \text{or} \quad R_x^y = C_i^y \varepsilon_x^{v_i} \tag{1.12}$$

In Equation 1.12, x can be any parameter that acts specifically on step i, and it is apparent that the *response coefficient* is always the product of a *control coefficient* and a *elasticity coefficient*. This relationship is known as the combined response relationship [65].

1.4 Control properties of MCA

Now that the local and global (systemic) properties within MCA have been introduced, it is important to highlight that the elasticity and control coefficient values are subject to a variety of constraints and inter-relationships, which are collectively known as the theorems of Metabolic Control Analysis [37]. The first theorem, the Summation Theorem, demonstrates that the enzymes of a pathway can share the control of flux and is independent of the individual kinetic properties of the enzymes. The second theorem, the Connectivity Theorem, provides a means to relate the properties of the individual enzymes to the systemic behaviour.

1.4.1 Summation Theorem

On developing Metabolic Control Analysis, Kacser and Burns [65] found that if all the enzymes affecting a particular metabolic flux are taken into account, and the values of their control coefficients on that flux added up, then the sum is 1. Therefore, for a system of n enzymes:

$$C_1^J + C_2^J ... + C_n^J = 1 \quad \text{or} \quad \sum_{i\,=\,1}^{n} C_i^J = 1 \tag{1.13}$$

Heinrich and Rapoport [48] defined the second summation relationships, the sum of concentration control coefficients. This property is concerned with the steady-state concentrations of a linear system, and states that the s_j-control coefficients for all steps within a metabolic system sum to zero:

$$C_1^{s_j} + C_2^{s_j} ... + C_n^{s_j} = 0 \quad \text{or} \quad \sum_{i\,=\,1}^{n} C_i^{s_j} = 0 \tag{1.14}$$

These two properties demonstrate how the global properties of a system are related. We can see clearly that the summation property of flux-control coefficients is key in terms of its role in dispelling the myths behind a single 'rate-limiting' step [39].

1.4.2 Connectivity Theorem

The Connectivity Theorem, derived by Kacser and Burns in 1973 [39, 65], provides the key to answering the question of how the flux control coefficients of enzymes can be related to the kinetic properties of the enzymes. This theorem is widely regarded as being the most meaningful of the theorems, since it provides a means of understanding how the kinetics of the enzymes affect the values of the flux control coefficients [37].

Within a metabolic system, if we isolated one pathway metabolite (S_1) and found all enzymes interacting with S_1, e.g. a, b and c, the connectivity theorem then states that for each of these enzymes, if one created a term of its flux control coefficient for a particular flux and multiplied it by its elasticity for S_1, then the sum of these terms is 0 [126, 39, 30].

$$C_a^J \varepsilon_{S_1}^a + C_b^J \varepsilon_{S_1}^b + C_c^J \varepsilon_{S_1}^c = 0 \tag{1.15}$$

Like the summation theorem, the connectivity theorem has two distinct general properties. The first property is the connectivity between flux-control coefficients and elasticities:

$$\sum_{i\,=\,1}^{n} C_i^{J_m} \varepsilon_{s_j}^{v_i} = 0 \tag{1.16}$$

The second property is concerned with the connectivity between the concentration-control coefficients and elasticities:

$$\sum_{i=1}^{n} C_i^{s_j} \varepsilon_{s_k}^{v_i} = -\delta_{jk} \tag{1.17}$$

where J_m refers to a specific system flux, S_j indicates the variable metabolite pool and δ_{jk} refers to the Kronecker delta (= 1 if $j = k$; otherwise = 0).

The second connectivity property was defined by Westerhoff and Chen in [122]. Although both theorems were originally defined for linear systems, they have since been extended and developed for both branched and unbranched pathways as well as for networks containing substrate cycles and moiety-conserved cycles [40, 56].

1.5 Equations of MCA

When combined, the summation and connectivity theorems allow the expression of control coefficients in terms of elasticity coefficients. According to Hofmeyr [51], this is arguably the most powerful feature of metabolic control analysis. We refer to the resulting equation as the *control-matrix equation*:

1.5.1 The control-matrix equation

A matrix formulation exists within metabolic control analysis which infers that the elasticity matrix for any system can be multiplied by the corresponding control matrix to yield an identity matrix. This generalised matrix formulation, known as the *control-matrix equation*, was initially derived by Reder in 1988, and emphasises that the structural characterisations and properties of a system are only dependent on the structure of the network and not on the reaction kinetics [87]. This formalism serves as the first concrete mathematical foundation of MCA. Numerous variations of this equation have subsequently been proposed [27, 28, 40, 44, 66, 98, 99, 106, 123, 124]. We will present the variation described in [53, 55] in more detail, as this variation forms the basis of this thesis.

Reder's formalism was developed in unscaled form whereas Hofmeyr *et al.* described a derivation in scaled form [55]:

$$\begin{bmatrix} \mathbf{C^J} \\ \mathbf{C^S} \end{bmatrix} \begin{bmatrix} \mathcal{K} & -\boldsymbol{\varepsilon}_\mathbf{s}\mathcal{L} \end{bmatrix} = \begin{bmatrix} \mathcal{K} & 0 \\ 0 & \mathcal{L} \end{bmatrix} \tag{1.18}$$

where $\mathbf{C^J}$ is an $n \times n$ matrix of scaled flux-control coefficients, $\mathbf{C^s}$ an $m \times n$ matrix of scaled concentration-control coefficients, and $\boldsymbol{\varepsilon}_\mathbf{s}$ an $n \times m$ matrix of scaled elasticity coefficients. n represents the number of reactions and m indicates the number of variable metabolites in the pathway. As per Reder [87], the $\mathcal{K}$ matrix expresses the dependence of the steady-state fluxes on the independent fluxes, and the $\mathcal{L}$ matrix expresses the dependence of the differential equations on the independent differential equations. In this case both the $\mathcal{K}$ and $\mathcal{L}$ matrices have been scaled. The use of the scaled forms of the various matrices for control analysis is generally the preferred method; Hofmeyr and colleagues [51, 52, 55] all provide sound examples of this scaling procedure.

The control-matrix can be further partitioned in terms of independent and dependent variables to produce:

$$\begin{bmatrix} \mathbf{C^{J_i}} \\ \mathbf{C^{J_d}} \\ \mathbf{C^{S_i}} \\ \mathbf{C^{S_d}} \end{bmatrix} \begin{bmatrix} \mathcal{K} & -\varepsilon_s \mathcal{L} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n-r} & 0 \\ \mathbf{K_o} & 0 \\ 0 & \mathbf{I}_r \\ 0 & \mathbf{L_o} \end{bmatrix} \tag{1.19}$$

Hofmeyr [51, 53] has shown that by extracting the equations for the independent variables (J_i and S_i) we can simplify the equation further. The steps involved are:

$$\begin{bmatrix} \mathbf{C^{J_i}} \\ \mathbf{C^{S_i}} \end{bmatrix} \begin{bmatrix} \mathcal{K} & -\varepsilon_s \mathcal{L} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n-r} & 0 \\ 0 & \mathbf{I}_r \end{bmatrix} \tag{1.20}$$

which, if $\mathbf{C^i} = [\mathbf{C^{J_i}} \mathbf{C^{S_i}}]^T$ and $\mathbf{E} = [\mathcal{K} - \varepsilon_s \mathcal{L}]$, can be written as:

$$\mathbf{C^i E} = \mathbf{I}_n \tag{1.21}$$

$\mathbf{C^i}$ is the matrix representing all independent systemic properties and $\mathbf{E}$ is the matrix representing all structural and local properties for the system.

Both $\mathbf{C^i}$ and $\mathbf{E}$ are square invertible $n \times n$ matrices in the absence of moiety-conservation (see Hofmeyr and Cornish-Bowden [53]), which implies that Equation 1.21 can also be written as $\mathbf{E C^i} = \mathbf{I}$. An important property is that $\mathbf{C^i}$ and $\mathbf{E}$ are inverses of each other. The implication is that control coefficients can be calculated from the elasticity coefficients:

$$\mathbf{C^i} = \mathbf{E^{-1}} \tag{1.22}$$

and that the opposite is also true, so that elasticity coefficients can be calculated from control coefficients [51]:

$$\mathbf{E} = (\mathbf{C^i})^{-1} \tag{1.23}$$

1.5.2 The inverse problem

Since $\mathbf{C^i} = \mathbf{E^{-1}}$, the implication is that if the elasticity coefficients have been determined experimentally or by calculation, we can calculate the control coefficients with respect to the independent concentrations and fluxes by inversion of $\mathbf{E}$. We can then calculate the dependent control coefficients using the following relationships [51]:

$$\mathbf{C^{J_d}} = \mathcal{K}_0 \mathbf{C^{J_i}} \tag{1.24}$$

$$\mathbf{C^{S_d}} = \mathcal{L_o} \mathbf{C^{S_i}} \tag{1.25}$$

When we consider the inverse problem, the calculation of elasticity coefficients from experimentally determined control coefficients, we find that for the case where there are no conservation constraints (i.e. moiety conserved cycles), this can be achieved through

inversion of the control matrix $\mathbf{C^i}$. This is because $\mathcal{L} = \mathbf{I}$. However, in the case where these conservation constraints are present ($\mathcal{L} \neq \mathbf{I}$), some elements in the right-hand columns within the resultant $\mathbf{E}$ matrix contain linear functions of elasticity coefficients. To solve for the individual elasticities we require additional information stored in the $\mathbf{T}$ matrix of conservation sums [51].

1.5.3 Control-matrix example using a simple model

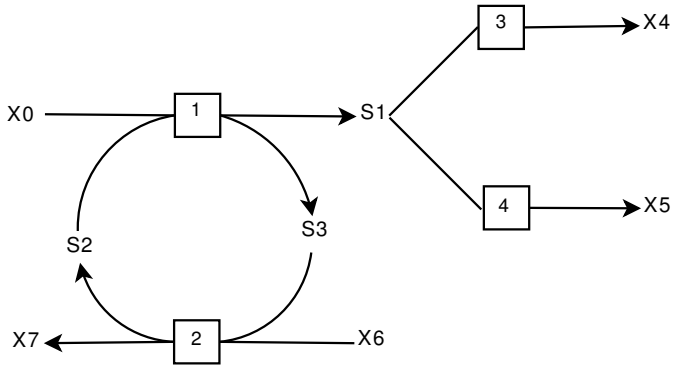

Figure 1.4: A schematic representation of a simple metabolic pathway consisting of a branched flux and a moiety-conserved cycle [51, 53].

Hofmeyr and co-workers [51, 53] provide a detailed explanation of the formulation of the control-matrix equation by means of a simple model, consisting of a branched flux and a moiety-conserved cycle. We will outline this process now as this equation forms the basis of the work undertaken in this thesis.

The $\mathbf{K}$ and $\mathbf{L}$-matrices are constructed from an analysis of the stoichiometric matrix ($\mathbf{N}$), both the $\mathbf{K}$ and the $\mathbf{L}$ matrices are ordered by independent flux or species followed by the independents, i.e. J_3, J_4, J_1, J_2 and s_1, s_2, s_3 for the examples in equations 1.26 and 1.27. The analysis of the $\mathbf{N}$ matrix can be performed by numerous software packages, such as `PySCeS` and `Copasi`. We have incorporated `PySCeS` into this study for the purpose of performing the stoichiometric analysis, and will not discuss the details. We can see an example of the analysis of the $\mathbf{N}$ matrix in Hofmeyr [51]. Examples of $\mathbf{K}$ (1.26) and $\mathbf{L}$ (1.27) matrices for the system illustrated in Figure 1.4 are:

$$\mathbf{K} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{1.26}$$

$$\mathbf{L} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.27}$$

The $\mathbf{K}$ matrix is then scaled to $\mathcal{K} = (\mathbf{D^J})^{-1}\mathbf{K}\mathbf{D^{J_i}}$:

$$\begin{bmatrix} \frac{1}{J_3} & 0 & 0 & 0 \\ 0 & \frac{1}{J_4} & 0 & 0 \\ 0 & 0 & \frac{1}{J_1} & 0 \\ 0 & 0 & 0 & \frac{1}{J_2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} J_3 & 0 \\ 0 & J_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \frac{J_3}{J_1} & \frac{J_4}{J_1} \\ \frac{J_3}{J_2} & \frac{J_4}{J_2} \end{bmatrix} \tag{1.28}$$

$\mathbf{L}$ is scaled to $\mathcal{L} = (\mathbf{D^s})^{-1}\mathbf{L}\mathbf{D^{s_i}}$:

$$\begin{bmatrix} \frac{1}{s_1} & 0 & 0 \\ 0 & \frac{1}{s_2} & 0 \\ 0 & 0 & \frac{1}{s_3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -\frac{s_2}{s_3} \end{bmatrix} \tag{1.29}$$

The next step in the formulation of the control-matrix equation is to calculate the matrix product $-\boldsymbol{\varepsilon}_\mathbf{s}\mathcal{L}$ where $\boldsymbol{\varepsilon}_\mathbf{s}$ is the matrix of all variable elasticity coefficients. The $-\boldsymbol{\varepsilon}_\mathbf{s}\mathcal{L}$ for the example system is formed as shown below:

$$-\begin{bmatrix} \varepsilon_{s_1}^{v_3} & 0 & 0 \\ \varepsilon_{s_1}^{v_4} & 0 & 0 \\ \varepsilon_{s_1}^{v_1} & \varepsilon_{s_2}^{v_1} & \varepsilon_{s_3}^{v_1} \\ 0 & \varepsilon_{s_2}^{v_2} & \varepsilon_{s_3}^{v_2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -\frac{s_2}{s_3} \end{bmatrix} = \begin{bmatrix} -\varepsilon_{s_1}^{v_3} & 0 \\ -\varepsilon_{s_1}^{v_4} & 0 \\ -\varepsilon_{s_1}^{v_1} & (\varepsilon_{s_3}^{v_1}\frac{s_2}{s_3} - \varepsilon_{s_2}^{v_1}) \\ 0 & (\varepsilon_{s_3}^{v_2}\frac{s_2}{s_3} - \varepsilon_{s_2}^{v_2}) \end{bmatrix} \tag{1.30}$$

The $\mathcal{K}$ matrix is then augmented onto the matrix product giving rise to:

$$\mathbf{E} = [\mathcal{K} - \boldsymbol{\varepsilon}_\mathbf{s}\mathcal{L}] = \begin{bmatrix} 1 & 0 & -\varepsilon_{s_1}^{v_3} & 0 \\ 0 & 1 & -\varepsilon_{s_1}^{v_4} & 0 \\ \frac{J_3}{J_1} & \frac{J_4}{J_1} & -\varepsilon_{s_1}^{v_1} & (\varepsilon_{s_3}^{v_1}\frac{s_2}{s_3} - \varepsilon_{s_2}^{v_1}) \\ \frac{J_3}{J_1} & \frac{J_4}{J_1} & 0 & (\varepsilon_{s_3}^{v_2}\frac{s_2}{s_3} - \varepsilon_{s_2}^{v_2}) \end{bmatrix} \tag{1.31}$$

This leads to the control-matrix equation, $\mathbf{C^iE} = \mathbf{I}$, which for the system in question is:

$$\begin{bmatrix} C_3^{J_3} & C_4^{J_3} & C_1^{J_3} & C_2^{J_3} \\ C_3^{J_4} & C_4^{J_4} & C_1^{J_4} & C_2^{J_4} \\ C_3^{s_1} & C_4^{s_1} & C_1^{s_1} & C_2^{s_1} \\ C_3^{s_2} & C_4^{s_2} & C_1^{s_2} & C_2^{s_2} \end{bmatrix} \begin{bmatrix} 1 & 0 & -\varepsilon_{s_1}^{v_3} & 0 \\ 0 & 1 & -\varepsilon_{s_1}^{v_4} & 0 \\ \frac{J_3}{J_1} & \frac{J_4}{J_1} & -\varepsilon_{s_1}^{v_1} & (\varepsilon_{s_3}^{v_1}\frac{s_2}{s_3} - \varepsilon_{s_2}^{v_1}) \\ \frac{J_3}{J_1} & \frac{J_4}{J_1} & 0 & (\varepsilon_{s_3}^{v_2}\frac{s_2}{s_3} - \varepsilon_{s_2}^{v_2}) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.32}$$

1.6 A brief introduction to metabolic regulation

Metabolic control analysis is an important tool for describing how the steady-state behaviour of a system is dependent on the individual enzyme-catalysed reactions within the system. Metabolic regulation aims to take the quantitative data obtained from MCA to another level. Hofmeyr and Cornish-Bowden [52] described metabolic regulation as 'the response of a metabolic steady-state to environmental changes seen as a combination of external and internal regulation'.

External regulation describes the change in steady-state behaviour of a metabolic system in response to a change in an external parameter. The co-control coefficient[1] for either a flux or a concentration was derived to describe this change [52, 55]. Co-control coefficients arose as a means of relating the simultaneous change in two independent steady-state variables on perturbation of a step. Hofmeyr and Cornish-Bowden [53] define a co-control coefficient for steady-state variables x_1 and x_2 with respect to a change in the local activity of Step i as:

$$O_i^{x_1:x_2} = \frac{C_i^{x_1}}{C_i^{x_2}} = \frac{\frac{\partial \ln x_1}{\partial \ln v_i}}{\frac{\partial \ln x_2}{\partial \ln v_i}} = \frac{\partial \ln x_1}{\partial \ln x_2} \tag{1.33}$$

Conversely, internal regulation is concerned with the co-ordinated steady-state response of a metabolic system by sensing the various internal variables of the system. Westerhoff and Kahn [68] proposed the use of partial internal response coefficients, which are in essence the terms of the connectivity equation, as a means of quantifying internal response.

The concept of supply-demand analysis, demonstrated by Hofmeyr and Cornish-Bowden [52, 54], allows us to quantify the behaviour, control and regulation of metabolism in terms of elasticities of supply and demand. In its simplest form a metabolic network consists of a catabolic block, a biosynthetic block, which produces the building block for macromolecular synthesis, and a block responsible for creating and maintaining the cellular structure and enzyme and gene structures. The producing block is referred to as the supply and the consuming block the demand, with either a single common intermediate or a pair of intermediates in the form of a moiety-conserved cycle linking the supply and demand blocks.

Hofmeyr and colleagues [50, 52], suggested that the co-control coefficients used to quantify external regulation are equivalent to the block elasticities under certain conditions. We can use the co-control coefficients to quantify the response of reaction blocks due to perturbations in a parameter, which are in turn brought about due to changes in an intermediary regulatory metabolite. The regulatory metabolites concentration is solely determined by the system parameters and links the supply and demand blocks.

[1] The two references cited [52, 55] use the term 'co-response coefficient', but a more recent document by the same authors uses the term 'co-control coefficient' [53]

The approaches of top-down [20, 21] and modular MCA [23, 102, 106] should not be ignored when assessing regulatory analysis techniques. Both approaches are closely related in that reactions are grouped together into blocks connected via a small number of intermediates. The grouping of reactions enables us to calculate the control coefficients from the fluxes or from the overall elasticities of the blocks to explicit intermediates using the connectivity property, as opposed to traditional MCA whereby we calculate control coefficients using numerous methods such as the effects of specific inhibitors on individual enzymes and enzyme overexpression [20].

Hofmeyr [49] introduced the concept of control-pattern analysis in which he developed a non-algebraic diagrammatic technique that generates the mathematical expressions for both flux and concentration control coefficients in terms of elasticity expressions. Each control-pattern represents how a perturbation in an enzyme activity reverberates through a metabolic pathway, which he refers to as a 'chain of local effects'. Since flux and concentration control coefficients provide a means with which to quantify the regulatory capability of an enzyme, these control-patterns can provide a deeper understanding about how changes in a regulatory enzyme can be transmitted to the rest of the system [49]. A regulatory enzyme would be any enzyme with which either an external or an internal regulator interacts [52]. Control-patterns are best demonstrated by means of a simple model:

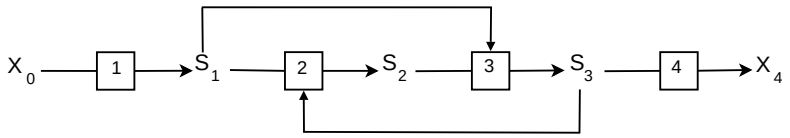

Figure 1.5: A four-enzyme linear pathway with both a feedforward and a feedback loop, as illustrated by Hofmeyr [49].

The model above was taken from [49], and will be used to show the control coefficients as generated through control-pattern analysis. We will not discuss the details of the technique here and these are covered in depth in [49].

The flux-control coefficient expressions are:

$$C_1^J = (\varepsilon_1^2\varepsilon_2^3\varepsilon_3^4 - \varepsilon_1^3\varepsilon_2^2\varepsilon_3^4)/\sum \quad (1.34)$$

$$C_2^J = (-\varepsilon_1^1\varepsilon_2^3\varepsilon_3^4)/\sum \quad (1.35)$$

$$C_3^J = (\varepsilon_1^1\varepsilon_2^2\varepsilon_3^4)/\sum \quad (1.36)$$

$$C_4^J = (-\varepsilon_1^1\varepsilon_2^2\varepsilon_3^3 + \varepsilon_1^1\varepsilon_2^3\varepsilon_3^2)/\sum \quad (1.37)$$

and the concentration-control coefficients with respect to S_1, are:

$$C_1^{S_1} = (\varepsilon_2^3\varepsilon_3^4 - \varepsilon_2^2\varepsilon_3^4 + \varepsilon_2^2\varepsilon_3^3 - \varepsilon_2^3\varepsilon_3^2)/\sum \quad (1.38)$$

$$C_2^{S_1} = (-\varepsilon_2^3\varepsilon_3^4)/\sum \quad (1.39)$$

$$C_3^{S_1} = (\varepsilon_2^2\varepsilon_3^4)/\sum \quad (1.40)$$

$$C_4^{S_1} = (-\varepsilon_2^2\varepsilon_3^3 + \varepsilon_2^3\varepsilon_3^2)/\sum \quad (1.41)$$

where

$$\sum = (\varepsilon_1^2\varepsilon_2^3\varepsilon_3^4 - \varepsilon_1^1\varepsilon_2^3\varepsilon_3^4 + \varepsilon_1^1\varepsilon_2^2\varepsilon_3^4 - \varepsilon_1^1\varepsilon_2^2\varepsilon_3^3 - \varepsilon_1^3\varepsilon_2^2\varepsilon_3^4 + \varepsilon_1^1\varepsilon_2^3\varepsilon_3^2) \quad (1.42)$$

Consider the control-pattern in the numerator term of the flux-control coefficient C_2^J (Equation 1.35), $-\varepsilon_1^1\varepsilon_2^3\varepsilon_3^4$, where the elasticities towards the substrates are positive and the elasticities towards the products are negative, and ignore the phenomena of substrate inhibition and product activation. Then under these conditions, the flux J, is increased by means of this pattern when the activity of Enzyme 2 (E_2) is increased. This change is brought about by the following.

Firstly, the increase in activity of E_2 decreases S_1 which results in the local rate of v_1 increasing. Secondly, the product of E_2 (S_2) increases which in turn increases the local rate v_3 which results in an increase of S_3 and an increase in the local rate of v_4. This example illustrates how a perturbation in an enzyme activity is propagated through a system, and demonstrates the insight provided by control-patterns when used within the framework of MCA. The control coefficient used in this example presents the simplest case consisting of a single control pattern. However, this is not always the case as demonstrated in Equations 1.34, 1.37, 1.38 and 1.41. For a situation with multiple control-patterns within a control coefficient, we can analyse each control-pattern separately as a chain of local events, where each pattern represents one way in which an enzyme perturbation could be propagated through the pathway. The analysis leads to an understanding of the positive or negative nature of the pattern. By combining the knowledge gained we can begin to explain the mechanism of how an enzyme modulation affects a cellular system.

1.7 Computational simulation and MCA

Computer simulation of metabolic pathways has developed since the early 1960s [82, 42, 88] into a key tool for understanding the transient and the steady-state behaviour as well as the control of metabolic pathways. Generally speaking, there are two methods to approach steady-state modelling. The first method is the 'mass action' approach in which

the desired pathway is described in terms of basic first or second order kinetics. However, this approach is compromised in that the values of the kinetic constants are seldom available. The second approach, the 'rate-law' approach, requires the calculation of each enzymatic rate from its associated rate equation. The advantage to the latter approach is that the type of data required is obtained using conventional steady-state analysis [57].

The solutions to problems concerned with metabolic control are seldom intuitive and thus computational simulation provides an important means of tackling these problems. One of the first software packages is `METAMOD`, which was developed by Hofmeyr and van der Merwe in the mid-1980s [57]. This package was initially designed for use on a BBC microcomputer and was capable of calculating the steady-state solution and performing control analysis of a model pathway, as introduced by Kacser and Burns [65] and Heinrich and Rapoport [47, 48]. This software led to the development of `MetaModel` [52], which was designed for use on the IBM PC and compatible computers. `MetaModel` provided a user-friendly framework for calculating steady-state fluxes and metabolite concentrations of metabolic systems. For any steady-state found, we could calculate a matrix of elasticity coefficients at that steady-state, or a matrix of control and response coefficients. Its purpose was thus to offer a simple means of calculating the control structure of a pathway.

In the 1990s and 2000s there was a significant increase in the number of simulation tools available, such as `SCAMP` [96], a general-purpose metabolic and chemical network simulator. `SCAMP` is portable to any operating system supported by an `ANSI C` compiler and is now known as Jarnac [95]. `Gepasi` [80], like `SCAMP`, is a simulator for modelling biochemical and chemical reaction networks consisting of a maximum of 45 metabolites and 45 reactions. This software was only available for use within a Windows environment and was designed as an educational and a research tool. `Gepasi` has since been superceded by `COPASI` [60], which is a stand-alone program that can be executed either by a graphical interface or via command-line. `COPASI` is available for all major operating systems such as Linux, Windows, Mac OS X and Solaris. The user is able to perform operations such as steady-state analysis, control analysis and parameter optimisations.

Of particular importance to this thesis is the software package developed within our research group, i.e. `PySCeS` [84]. `PySCeS`, which stands for `Python` Simulator for Cellular Systems, was designed as an extendable research tool for the purposes of numerical analysis and investigation of cellular systems. `PySCeS` was designed for use on Windows (2000/XP/Vista) and Linux operating systems, and it has been successfully ported to Mac OS X. `PySCeS` is an extremely flexible and user-extensible software package. The interface to `PySCeS` is via command-line, with numerous options available to the user, such as time-course simulations, steady-state analysis, system stability analysis via calculation of eigenvalues and plotting of results.

Due to the increasing importance of computer modelling and simulation in understand-

ing and investigating chemical and biological systems, a number of diverse tools have been developed. In the early 2000s a framework was developed to link a number of these heterogeneous applications thus allowing them to communicate and take advantage of each of their capabilities. Systems Biology Workbench (`SBW`, [97]) was designed and developed with this specific task in mind.

With the growing number of biological and chemical systems being simulated, a need arose for a central store of these models, and to serve this function a few repositories have been developed. Two repositories worth mentioning are BioModels (`http://www.ebi.ac.uk/biomodels-main/`) [77] and JWS online (`http://jjj.biochem.sun.ac.za/`) [85]. BioModels is a database housing curated quantitative models of biochemical and cellular systems that have been published and peer-reviewed. JWS Online serves as a Systems Biology tool to simulate kinetic models from a curated database. JWS Online allows users to view and manipulate curated models using a web browser by means of an easy-to-use interface, and the user can perform steady-state and time-course analysis as well as perform control analysis.

A key question often asked by investigators is how the behaviour of a model is affected by changes in a parameter over a range of values. An example of this would be changing the demand for S_1 in Figure 1.4, by varying the value for either V_{f3} (enzyme 3) or V_{f4} (enzyme 4) over a range such as 0.1–10. This approach involves a series of steady-state calculations over the specified range, and is known as a parameter scan. It is possible to perform single and multi-dimensional parameter scans with most currently available simulation tools, such as `PySCeS` and `Copasi`.

1.8 Symbolic computation

Symbolic computation (or computer-algebra) entails the use of machines such as computers, to manipulate mathematical equations and expressions in symbolic form as opposed to the more common method of manipulating the approximations of specific numerical quantities which are represented by those symbols. The beginnings of computer-algebra date back to 1953 when two programs were written for the purposes of symbolic differentiation [25, 86]. However, it appears that its subsequent use was restricted to the work carried out by specialist research groups, due mainly to the fact that computer-algebra was originally intended for use on large mainframe computers that were designed to handle the extensive memory requirements associated with performing computer-algebra [86].

Today we use computer-algebra more often, and it has applications in many areas of science and technology such as Chemistry, Computational Biology, Computer Science, Education, Engineering, Mathematics and Physics [63]. The increased use of computer-algebra is a direct result of the increasing availability of relatively inexpensive, and ever increasing, powerful personal computers, which can now perform these symbolic compu-

tations [86].

To meet the ever-increasing need for computer-algebra in the domains previously mentioned are numerous Computer Algebra Systems (CAS). These systems began to appear in the early 1970s, and are believed to have evolved from research into Artificial Intelligence [2, 3]. The principle aim of any CAS is to automate tedious and often difficult algebraic tasks, and the uses and capabilities of the various systems available vary greatly from one system to another [129].

A few well-known general purpose computer algebra systems include `Axiom` [1], `GiNaC` [119], `Reduce` [46], `Macsyma` [4], `Maple` [5], `Mathematica` [6] and `Maxima` [7]. Macsyma was designed by Carl Engelman, William Martin and Joel Moses. Its development began in 1968 at the MIT Artificial Intelligence laboratories, and it is regarded as the first general purpose computer algebra program. `Maple` was designed to make computer-algebra more widely accessible and to achieve this, portability and speed were the primary objectives in its design. Development began in 1980 at the University of Waterloo, Canada. Present capabilities include numerical calculation, symbolic integration and differentiation, symbolic equation solving, trigonometric and exponential/logarithm functions, linear algebra, statistics, two- and three-dimensional plotting, animation and interfaces with the `C` and `Fortran` programming language [64, 128].

`Mathematica`, which is arguably today's world leading computer algebra tool, was developed by Stephen Wolfram. `Mathematica` provides users with a comprehensive environment for various mathematical applications such as elementary computations and transformations as well as for large development projects building mathematical models for use in complex engineering problems [64, 128]. `Maxima` is a direct descendant of the `Macsyma` system, and unlike the commercially available `Mathematica`, is a freely available open source system. This system is used to manipulate symbolic and numerical expressions, including differentiation, integration, Taylor series, Laplace transforms, ordinary differential equations, systems of linear equations, polynomials, as well as sets, lists, vectors, matrices and tensors. `Maxima` can also yield high precision numeric results computed by using exact fractions, arbitrary precision integers and variable precision floating point numbers. Plotting of data and functions can also be performed in two and three dimensions [7].

1.8.1 Symbolic computation and MCA

The majority of software packages developed for MCA has revolved around numerical analyses. Computational and experimental MCA can be a time-consuming and error prone process due to the number of and difficulty in determining the values for all the variables within a system. This has led to the development and description of alternative strategies in an attempt to reduce the effort and the possibility of errors, which are commonly associated with performing an analysis on biological systems.

These approaches range from a diagrammatic one as introduced by Hofmeyr with his control-pattern analysis technique, in which he describes a non-algebraic means of expressing control coefficients in terms of elasticity coefficients [49], to graph-theoretical approaches [107], and finally to the numerous matrix-based methods, which can be found in [27, 28, 40, 87, 99, 122, 124].

The use of computer-algebra (symbolic computation) in MCA was first tackled by Schultz [101], in which he describes an algorithm to generate symbolic expressions for control coefficients, and subsequently by Thomas and Fell [114], with their description of the `MetaCon` software package. `MetaCon` was developed as an implementation of the matrix method as illustrated in [40, 110, 99], where all calculations are performed using numeric and symbolic processing such that numerical terms are collected and evaluated. The implication was that in the event of the values of one or more variables being unknown, the control-coefficient data outputted from `MetaCon` would be in the form of polynomials in terms of the unknown or unknowns [114].

`MetaCon` had a number of shortcomings. Firstly, the implementation was based on a variation of MCA in which a reference flux would need to be selected before computation. Secondly, the output generated was by default in text format, with the option of LaTeX expressions as well. The text would then need to be edited into functions for other programs or Excel, thus requiring further post analysis manipulations and additional third party software.

We therefore saw the need for developing a generic symbolic implementation of MCA as an important step in the development of Systems Biology. This is the primary objective of this study, as once a suitable tool exists, it is the intention of this thesis to demonstrate the usefulness of such a tool in further understanding the regulatory behaviour inherent in biological systems. The technique of control-pattern analysis demonstrated by Hofmeyr [49] has provided a solid foundation to understand regulation. However, this technique is performed diagrammatically and does not lend itself to large systems. The generation of control coefficient expressions mirroring these control-patterns would provide these routes of regulation in an automated fashion, as well as for larger systems. A common problem associated with systems modelling is knowing all parameter values for the system under investigation, and the elucidation of the expressions can aid the identification of the key parameters. The chapters that follow describe the tool that will be developed to tackle these challenges.

1.9 Thesis outline

The central idea addressed in this thesis is how we can gain a greater understanding of the mechanisms and chain of events that affect the regulatory aspects of cellular systems. This thesis consists of two parts. The first part is concerned with the development and implementation of the tools required to gain this insight, and the second part

is concerned with demonstrating how we can use symbolic control analysis to describe and gain a clearer understanding into the regulatory mechanisms within cellular systems.

The first part of the thesis consists of the general ideas, development and implementation of the tool required to tackle the problem. **Chapter 2** is concerned with the initial proof of concept of the software tool. An example model illustrates the fundamental aims, and we highlight any shortcomings that we will address in the final development and implementation of the proposed software tool. **Chapter 3** provides an account of the design, development and implementation of the software package, which is later referred to as `SymCA` (Symbolic Control Analysis). We address the choices made and the reasons why the software was implemented in the manner in which it was. **Chapter 4** describes how `SymCA` is used, and introduces the functionality of the software that we use in the remaining chapters of the thesis. For this chapter we created a theoretical model which includes all the common features found in cellular systems, i.e. moiety-conserved cycles, linear pathways and branched pathways.

The second part of the thesis involves the use of `SymCA` as a means of performing an in-depth analysis of various 'real-life' biological systems, to gain a greater insight into the regulatory mechanisms at work. It should be noted that this part of the thesis is concerned with demonstrating the application of `SymCA` by using examples showing how a symbolic analysis can lead to a deeper understanding. The aim of this thesis is not to perform a detailed comprehensive analysis for each of the models mentioned in Chapters 5-7. **Chapter 5** demonstrates the combined strategy of control-pattern quantification, supply-demand analysis and a parameter scan for understanding the regulatory behaviour of a cellular system. **Chapter 6** highlights the role of control patterns, and more importantly the data obtained using a quantified approach to control-patterns. In this chapter, we extend work undertaken by Galazzo and Bailey [41] to demonstrate how quantified control patterns can be used to explain observed experimental outcomes. **Chapter 7** focusses on applying the same techniques of control pattern quantification using the work of Uys *et al.* [116], where they describe a kinetic model of sucrose accumulation in maturing sugar cane as a basis for the investigation. In this study the control patterns are computed for each internode in an attempt to provide evidence of the affects of internode maturity on sucrose accumulation.

The final section of this thesis discusses the findings of this work as well as provides further points which could be considered for future work related to this study.

Chapter 2

SymCA: Proof of concept

2.1 Introduction

This chapter demonstrates the feasibility of an algebraic/symbolic approach to MCA. As outlined in Chapter 1, the field of MCA provides an effective means with which we can explain and quantify the workings of cellular systems. This technique has been applied conventionally by means of numerical simulations, and a number of computational tools are presently available, such as `PySCeS` [84] and `Copasi` [60]. The intention of this study is to implement an equivalent algebraic or symbolic approach to solve this problem, with the ultimate goal the generation of algebraic expressions representing the systemic properties (control coefficients) of cellular systems in terms of the local properties (elasticity coefficients).

This chapter highlights the core algorithm used as well as various external softwares that are required to reach a solution. A theoretical model demonstrates the proof of concept; this model contains all the structural features commonly found in cellular systems, i.e., linear and branched segments and a moiety conserved cycle.

The solution to our problem required the choice of a programming language and a means of performing symbolic computations. Since the desired outcome was to incorporate the symbolic control analysis application into the existing software package developed within our group, `PySCeS`, the logical choice of programming language was `Python`. At the time of development `Python` lacked a comprehensive symbolic computing environment, and so we needed external software. `Sage` has now been developed with a comprehensive mathematics environment providing a `Python`-based interface which combines many open-source mathematics software packages [10]. However, `Sage` requires a separate installation of `Python`, and thus we would have two separate `Python` installations on a single machine, which was undesirable.

One of the mathematical packages which `Sage` had wrapped is `Maxima`. `Maxima` is a powerful, open-source algebraic software which can be run on numerous platforms (Linux,

Mac OS X and Windows), and so is an ideal option to use with `Python`. However, we had to develop our own interface, which will be addressed shortly. The primary objective for the work described in this chapter is to provide a 'proof of concept' for the implementation of the software workflow and the `Python/PySCeS-Maxima` interface, and not to provide any form of mathematical proof in terms of rewriting the connectivity and summation theorems.

2.2 Software requirements

2.2.1 Maxima

`Maxima` is a descendant of `Macsyma`, a computer algebra system that was initially developed at MIT in the 1960s as a component of Project MAC. By the mid-1980s, although versions of current day commercial packages `Mathematica` and `Maple` were gaining market share, `Macsyma` was by far the strongest system. We can trace the beginnings of `Maxima` back to 1982 when William Schelter of the University of Texas at Austin acquired the source code to a version of `Macsyma` (DOE Macsyma), which had been licensed to the US Department of Energy. Schelter adapted the source code for common `Lisp`, and in 1998 released it under the GPL renamed as `GNU Maxima`. `Maxima` is currently under active development, and the team includes some of the original developers of `Macsyma` [9]. There are several advantages for using `Maxima` in this study:

- It is licensed under GPL and thus will always be freely available.
- It is under active development.
- `Maxima` is being developed by its users, and is therefore in tune with its user community.
- In its current form, `Maxima` has excellent calculus, linear algebra and general symbolic capabilities.

2.2.2 PySCeS

`PySCeS` (Python Simulator for Cellular Systems) is an open source modelling tool developed by our group. As the name implies, `PySCeS` is written in the `Python` programming language, runs on both Linux and Microsoft Windows (2000/XP/Vista), and is console-based. This software application can perform stoichiometric analysis, time-course and steady-state calculations as well as numerical MCA of models of a cellular system. However, `PySCeS` lacks symbolic capabilities which this study addresses [84].

2.2.3 Python

The `Python` programming language is best described as an object-oriented scripting language in that its design merges the software engineering features of traditional languages

(Java and C++) with the usability of scripting languages (Perl and Tcl). Numerous features make Python an ideal language for scientific computing.

- Object-oriented language: Its class model supports advanced features such as polymorphism, operator overloading and multiple inheritance. At the same time we can achieve much without having to use these features.
- Open-source: The entire system is freely available over the Internet whilst also being actively maintained and supported.
- Portability: Since Python is written in ANSI C it compiles on virtually every major platform in use today.
- Powerful: Python is a hybrid in terms of its features. The Python tool set places it between traditional scripting languages and systems languages. The simplicity and ease of use of scripting languages combined with more advanced programming tools typically present in systems development languages, is ideally suited for substantial development projects.
- Mixable: Python allows the user to easily 'glue' components written in other languages (e.g. SWIG (http://swig.org) [14] and ctypes (http://sourceforge.net/projects/ctypes/).)

All the above features and many others made Python an obvious choice for this study, combined with the fact that PySCeS is also written in Python. Our intention was for SymCA to be an additional module for PySCeS, and thus all PySCeS typographical conventions were used in SymCA. These included the prefix 'J_' for reactions to denote a flux, the prefixes 'cc', 'ec' for all control and elasticity coefficients and a '_' to delimit the perturbed step from the measured steady-state variable for both control and elasticity coefficients.

2.3 The humble beginnings of SymCA

The *control-matrix equation*, as described in Section 1.5.1, is one of the most powerful features of MCA. The power of the equation lies in that we can compute control coefficients from elasticity coefficients and vice versa, as shown in Equations 1.20, 1.21 and 1.22. Since this matrix method calculates only the independent control coefficients, we required additional relationships to calculate the dependent control coefficients, as shown in Equations 1.24 and 1.25. Although originally derived by Reder [87], numerous variations of this equation have been proposed, with the method described by Hofmeyr and colleagues [51, 53, 55] serving as the basis of the algorithm used to write the control analysis program, SymCA.

We had successfully identified the core idea, an algebraic approach to MCA, and the necessary tools required to prototype a symbolic implementation of MCA, and more

specifically an implementation of the control-matrix equation. But since `Python` lacks symbolic capabilities and has the ability to act as a 'glue' for linking softwares written in other languages, we needed to determine whether we could develop a `Python` interface to `Maxima`[1].

2.3.1 Maxima interface

There are two ways of testing the feasibility of writing a `Python` interface to `Maxima`, that is using:

1. `Pexpect`: `Python` version of the Expect tool, which is a Unix automation and testing tool
2. `subprocess`: a native `Python` module enabling us to spawn new processes and connect to their input/output/error pipes, and thus obtain their return codes.

Pexpect requires the installation of additional libraries, but often encounters difficulties when attempting to port Expect scripts between platforms, especially Windows operating systems[2]. Since we aimed to develop our software for multi-platform use, this was not desirable. Fortunately, `subprocess` comes with all standard installations of `Python` and is easily portable between platforms. This made the choice to use the `subprocess` module simple, as the interface to `Maxima` would not require the installation of additional libraries and our code would be portable between platforms.

The `subprocess` module defines one class called `Popen`, which requires a number of arguments, the majority being optional. The only arguments required to connect and communicate with `Maxima` were:

- *args*: a string, or a sequence of program arguments, with the program to execute normally the first item in the args sequence or string
- *stdin*: executed program's standard input file handle
- *stdout*: executed program's standard output file handle
- *stderr*: executed program's standard error file handle.

Since `Maxima` has a command line interface and a graphical user interface (GUI), it can be launched from a console by typing `maxima`. This served as the `args` string argument, resulting in the execution of `Maxima` from within a `Python` session. The block of `Python` code we used to connect to `Maxima` in the initial stages of development follows:

```
import subprocess as sp
maxima = sp.Popen((‘maxima’), stdin=sp.PIPE, stdout=sp.PIPE,
         stderr=sp.PIPE)
```

[1]Initial development was done on a Linux platform, with the objective being to port the code to Windows platforms once the initial proof of concept had been successfully completed.

[2]This is not an impossible task, but one that requires the installation of additional external libraries.

The `subprocess` module raises two errors. The first error occurs when the application to be executed fails (`OSError`), and the second error occurs when invalid arguments are passed to Popen (`ValueError`). Provided that these exceptions are not raised, a `Maxima` process is spawned and we could begin using `Maxima` from within a `Python` session. To provide the `Maxima` interface with arguments to execute, we wrote a method named `compute` to take the execution statement as an argument. This argument is in string format, equivalent to a command that we would provide if using the command-line interface to `Maxima`. The input argument is passed to `Maxima` using the standard input pipe, and all subsequent output is read from the standard output pipe.

2.3.2 Symbolic metabolic control analysis

This section describes the process of performing a symbolic equivalent of MCA. We used the protocol described by Hofmeyr [51]. For the initial development we used a theoretical model designed to include all common biological features such as a branch, a moiety-conserved cycle and a linear segment.

As mentioned previously, the objective was to develop a symbolic add-on module for the `PySCeS` application, and thus `PySCeS` should be an integral part of `SymCA`. The only requirements for this was that we install `PySCeS` and have a `PySCeS` input file for the cellular system in question. We then loaded the model into `PySCeS` and computed the steady-state. Since `PySCeS` routinely performs stoichiometric analyses in an efficient and optimal manner, we decided to obtain the **K** and **L** matrix data from `PySCeS`, these matrices being easily accessible as model attributes. We extracted the following data from `PySCeS`, converted it into symbolic data equivalents and then used it as inputs for `Maxima`:

- list of all model reaction equations
- list of all model reactions
- list of all model species
- **K** matrix, with columns as independent fluxes and rows consisting of all fluxes ordered first as independents and then as dependents
- **L** matrix, with columns referring to independent species and rows to all species ordered first by independents and then as dependents
- $\mathbf{K_0}$ matrix, describing all dependent flux relationships in terms of the independent fluxes
- $\mathbf{L_0}$ matrix, describing the dependent species relationships expressed in terms of the independent species.

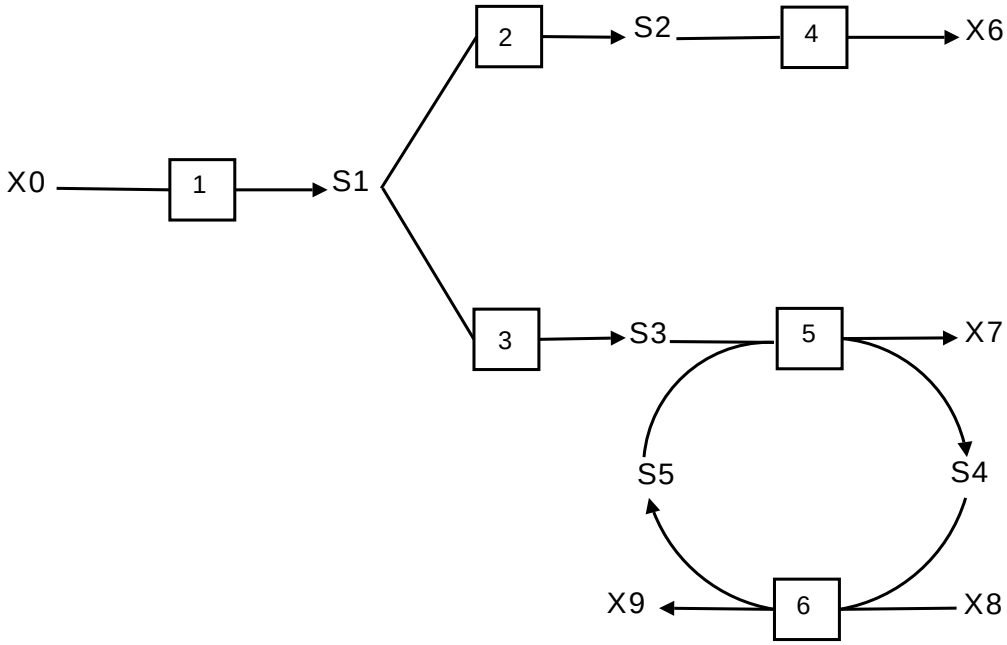

Figure 2.1: Theoretical system used as the initial test model for development of `SymCA`.

After we extracted the required data, the generation of symbolic equivalents to be sent to `Maxima` entailed a number of steps, such as generation of lists for both fluxes and species ordered in terms of independents followed by dependents, the scaling of both the **K** and **L** matrices, and the generation of the **E** matrix as the final step. The **E** matrix was then inverted giving rise to algebraic expressions for the control coefficients expressed in terms of elasticity coefficients.

To scale the **K** and **L** matrices we required the order and the generation of the symbolic flux and species data. We obtained the order of the flux and species from the respective **K** and **L** matrix row model attributes in `PySCeS` where each element in the list extracted represents an index for the corresponding reaction or species label found in the list of reactions or species. For fluxes we prefixed a `'J'` onto each reaction label to denote that it is a flux, and the species retained the same nomenclature as entered in the `PySCeS` input file. For the example model in question (Figure 2.1), the ordered lists for both fluxes and species are:

- JR6*, JR4*, JR5, JR2, JR3, JR1 (where * denotes the independent fluxes)
- S3*, S2*, S1*, S5*, S4 (where * denotes the independent species)

$\mathbf{K}$ is scaled to $\mathcal{K} = (\mathbf{D^J})^{-1}\mathbf{K}\mathbf{D^{J_i}}$ as shown in the following method:

$$\begin{bmatrix} 1/J_6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/J_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/J_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/J_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/J_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/J_1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ `1 & 0 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} J_6 & 0 \\ 0 & J_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \frac{J_6}{J_5} & 0 \\ 0 & \frac{J_4}{J_2} \\ \frac{J_6}{J_3} & 0 \\ \frac{J_6}{J_1} & \frac{J_4}{J_1} \end{bmatrix} \tag{2.1}$$

where $(\mathbf{D^J})^{-1}$ is a diagonal matrix of all inverse fluxes and $\mathbf{D^{J_i}}$ represents the diagonal matrix of all independent fluxes.

$\mathbf{L}$ is scaled to $\mathcal{L} = (\mathbf{D^s})^{-1}\mathbf{L}\mathbf{D^{s_i}}$ as shown in the following method:

$$\begin{bmatrix} 1/S_3 & 0 & 0 & 0 & 0 \\ 0 & 1/S_2 & 0 & 0 & 0 \\ 0 & 0 & 1/S_1 & 0 & 0 \\ 0 & 0 & 0 & 1/S_5 & 0 \\ 0 & 0 & 0 & 0 & 1/S_4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \times \begin{bmatrix} S_3 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_1 & 0 \\ 0 & 0 & 0 & S_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{S_5}{S_4} \end{bmatrix} \tag{2.2}$$

where $\mathbf{D^{s^{-1}}}$ represents a diagonal matrices of all inverse species and $\mathbf{D^{s_i}}$ represents the diagonal matrices of all independent species.

We required a matrix consisting of all variable elasticities, referred to as $\boldsymbol{\varepsilon_s}$, for generation of the final $\mathbf{E}$ matrix in the control-matrix equation. The arrangement of $\boldsymbol{\varepsilon_s}$ shows the columns representing the free species and the rows indicating the reactions, both of which are ordered independents followed by dependents.

We created a dictionary with reaction labels as keys and their corresponding reaction equations using the list of model reaction equations extracted from `PySCeS`. We used this dictionary with the ordered lists of flux and species data to create the $\boldsymbol{\varepsilon_s}$ matrix. The algorithm implemented was a nested for-loop. Firstly, the list of reactions was stepped through, and secondly, for each step in the iteration, i.e. for each reaction, the list of species was stepped through. For every iteration of the species list we checked to see if the species was present in the reaction equation of the reaction being stepped through. If the species was present in the reaction equation, an elasticity for that species towards the reaction was inserted into the matrix. If the species was not present a zero was inserted, with the final $\boldsymbol{\varepsilon_s}$ matrix for the system in Figure 2.1 as follows:

$$\begin{bmatrix} 0 & 0 & 0 & \varepsilon_{S_5}^{R_6} & \varepsilon_{S_4}^{R_6} \\ 0 & \varepsilon_{S_2}^{R_4} & 0 & 0 & 0 \\ \varepsilon_{S_3}^{R_5} & 0 & 0 & \varepsilon_{S_5}^{R_5} & \varepsilon_{S_4}^{R_5} \\ 0 & \varepsilon_{S_2}^{R_2} & \varepsilon_{S_1}^{R_2} & 0 & 0 \\ \varepsilon_{S_3}^{R_3} & 0 & \varepsilon_{S_1}^{R_3} & 0 & 0 \\ 0 & 0 & \varepsilon_{S_1}^{R_1} & 0 & 0 \end{bmatrix} \tag{2.3}$$

The matrix product $-\boldsymbol{\varepsilon_s}\mathcal{L}$ could now be computed,

$$\begin{aligned} -&\begin{bmatrix} 0 & 0 & 0 & \varepsilon_{S_5}^{R_6} & \varepsilon_{S_4}^{R_6} \\ 0 & \varepsilon_{S_2}^{R_4} & 0 & 0 & 0 \\ \varepsilon_{S_3}^{R_5} & 0 & 0 & \varepsilon_{S_5}^{R_5} & \varepsilon_{S_4}^{R_5} \\ 0 & \varepsilon_{S_2}^{R_2} & \varepsilon_{S_1}^{R_2} & 0 & 0 \\ \varepsilon_{S_3}^{R_3} & 0 & \varepsilon_{S_1}^{R_3} & 0 & 0 \\ 0 & 0 & \varepsilon_{S_1}^{R_1} & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{S_5}{S_4} \end{bmatrix} \\ =&\begin{bmatrix} 0 & 0 & 0 & (\varepsilon_{S_4}^{R_6}\frac{S_5}{S_4} - \varepsilon_{S_5}^{R_6}) \\ 0 & -\varepsilon_{S_2}^{R_4} & 0 & 0 \\ -\varepsilon_{S_3}^{R_5} & 0 & 0 & (\varepsilon_{S_4}^{R_5}\frac{S_5}{S_4} - \varepsilon_{S_5}^{R_5}) \\ 0 & -\varepsilon_{S_2}^{R_2} & -\varepsilon_{S_1}^{R_2} & 0 \\ -\varepsilon_{S_3}^{R_3} & 0 & -\varepsilon_{S_1}^{R_3} & 0 \\ 0 & 0 & -\varepsilon_{S_1}^{R_1} & 0 \end{bmatrix} \end{aligned} \tag{2.4}$$

Finally, the **E** matrix was created by augmenting the $\mathcal{K}$ matrix with the newly formed matrix product, $-\boldsymbol{\varepsilon}_{\mathbf{s}}\mathcal{L}$ giving rise to the control-matrix equation:

$$\begin{bmatrix} C_6^{J_6} & C_4^{J_6} & C_5^{J_6} & C_2^{J_6} & C_3^{J_6} & C_1^{J_6} \\ C_6^{J_4} & C_4^{J_4} & C_5^{J_4} & C_2^{J_4} & C_3^{J_4} & C_1^{J_4} \\ C_6^{S_3} & C_4^{S_3} & C_5^{S_3} & C_2^{S_3} & C_3^{S_3} & C_1^{S_3} \\ C_6^{S_2} & C_4^{S_2} & C_5^{S_2} & C_2^{S_2} & C_3^{S_2} & C_1^{S_2} \\ C_6^{S_1} & C_4^{S_1} & C_5^{S_1} & C_2^{S_1} & C_3^{S_1} & C_1^{S_1} \\ C_6^{S_5} & C_4^{S_5} & C_5^{S_5} & C_2^{S_5} & C_3^{S_5} & C_1^{S_5} \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & (\varepsilon_{S_4}^{R_6}\frac{S_5}{S_4} - \varepsilon_{S_5}^{R_6}) \\ 0 & 1 & 0 & -\varepsilon_{S_2}^{R_4} & 0 & 0 \\ \frac{J_6}{J_5} & 0 & -\varepsilon_{S_3}^{R_5} & 0 & 0 & (\varepsilon_{S_4}^{R_5}\frac{S_5}{S_4} - \varepsilon_{S_5}^{R_5}) \\ 0 & \frac{J_4}{J_2} & 0 & -\varepsilon_{S_2}^{R_2} & -\varepsilon_{S_1}^{R_2} & 0 \\ \frac{J_6}{J_3} & 0 & -\varepsilon_{S_3}^{R_3} & 0 & -\varepsilon_{S_1}^{R_3} & 0 \\ \frac{J_6}{J_1} & \frac{J_4}{J_1} & 0 & 0 & -\varepsilon_{S_1}^{R_1} & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

We have generated all the matrices described above in string form to use as `Maxima` input. The computations required to scale both the **K** and **L** matrices were using `Maxima`. Since we had generated the symbolic variant of the **E** matrix, we were in a position to compute the control-matrix equation, and thus generate symbolic control coefficients for the system in question[3].

Computation of symbolic control coefficients

To generate the symbolic control coefficients, we passed the newly formed **E** matrix, in `Maxima` input form to the `Maxima` instance via the `compute` method mentioned previously. We denoted all elasticity coefficients by `ec` in the input argument. We split the input argument for the test model over multiple lines for display purposes as follows:

```
E:matrix([1,0,0,0,0,ecR6_S4*S5/S4-ecR6_S5],[0,1,0,-ecR4_S2,0,0],
[1.0,0,-ecR5_S3,0,0,ecR5_S4*S5/S4-ecR5_S5],[0,1.0,0,-ecR2_S2,
-ecR2_S1,0],[1.0,0,-ecR3_S3,0,-ecR3_S1,0],[JR6/JR1,JR4/JR1,0,0,
-ecR1_S1,0]);
```

We used the `invert` function, available in all standard installations of `Maxima` to invert the **E** matrix. Once successfully inverted, we read the newly computed control coefficients over the standard output file handle, and isolated and removed the common denominator of all control coefficients from each expression in the matrix. To visualise the control coefficients, we created a basic LaTeX output method, and used this in order to validate the initial output.

[3] At this developmental stage we were concerned only with computing the independent control coefficients.

The common denominator ($\sum$) for all control coefficients shown in Tables 2.1 and 2.2 is:

$$\begin{aligned}\sum = & \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} \\ & - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{J_{\mathrm{R1}}\ S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} \\ & + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} \\ & + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} \\ & - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{J_{\mathrm{R1}}} \\ & + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} \\ & + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}\end{aligned} \tag{2.6}$$

Table 2.1: Independent flux control coefficients generated by `SymCA`.

Control coefficient	**Expression**
$C_6^{J_6}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{J_{\mathrm{R1}}\ S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{J_{\mathrm{R1}}} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5})/\sum$
$C_4^{J_6}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}})/\sum$
$C_5^{J_6}$	$(-\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6})/\sum$
$C_2^{J_6}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4})/\sum$
$C_3^{J_6}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6})/\sum$
$C_1^{J_6}$	$(\frac{S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6})/\sum$
$C_6^{J_4}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{J_{\mathrm{R1}}} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{J_{\mathrm{R1}}\ S4})/\sum$

continued on next page

continued from previous page

Control coefficient	Expression
$C_4^{J_4}$	$(-\frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} + \frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{R1}} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5})/\sum$
$C_5^{J_4}$	$(\frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} - \frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{J_{R1}})/\sum$
$C_2^{J_4}$	$(\frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} - \frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{R1}} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5})/\sum$
$C_3^{J_4}$	$(\frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{R1}} - \frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4})/\sum$
$C_1^{J_4}$	$(\frac{S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} - \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5})/\sum$

Table 2.2: Independent concentration control coefficients generated by `SymCA`.

Control coefficient	Expression
$C_6^{S_3}$	$(\frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R5}}{J_{R1}\ S4} - \frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R5}}{J_{R1}\ S4} + \frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R5}}{J_{R1}\ S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R5}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R5}}{S4} - \frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R5}}{J_{R1}} + \frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R5}}{J_{R1}} - \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R5}}{J_{R1}} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R5} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R5})/\sum$
$C_4^{S_3}$	$(\frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} - \frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R5}}{J_{R1}\ S4} - \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R6}}{J_{R1}} + \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R5}}{J_{R1}})/\sum$

continued on next page

continued from previous page

Control coefficient	Expression
$C_5^{S_3}$	$(\frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} - \frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R5}}{J_{R1}\ S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R5}}{S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R5}}{S4} - \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R6}}{J_{R1}} + \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R5}}{J_{R1}} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R5} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R5})/\sum$
$C_2^{S_3}$	$(-\frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} + \frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R5}}{J_{R1}\ S4} + \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R6}}{J_{R1}} - \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R5}}{J_{R1}})/\sum$
$C_3^{S_3}$	$(\frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} - \frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R5}}{J_{R1}\ S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R5}}{S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R5}}{S4} - \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R6}}{J_{R1}} + \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R5}}{J_{R1}} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R5} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R5})/\sum$
$C_1^{S_3}$	$(\frac{S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S4}^{R5}}{S4} + \frac{S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S4}^{R5}}{S4} - \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S5}^{R5} - \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S5}^{R5})/\sum$
$C_6^{S_2}$	$(\frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{J_{R1}} - \frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{J_{R1}\ S4})/\sum$
$C_4^{S_2}$	$(-\frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} - \frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} + \frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} - \frac{J_{R1}\ J_{R4}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{J_{R1}\ S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} + \frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{R1}} + \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{R1}} - \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{J_{R1}} + \frac{J_{R1}\ J_{R4}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{J_{R1}} - \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5})/\sum$
$C_5^{S_2}$	$(\frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} - \frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{J_{R1}})/\sum$
$C_2^{S_2}$	$(\frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} - \frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{R1}} + \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R1}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5})/\sum$
$C_3^{S_2}$	$(\frac{J_{R1}\ J_{R6}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{R1}} - \frac{J_{R1}\ J_{R6}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{R1}\ S4})/\sum$
$C_1^{S_2}$	$(\frac{S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} - \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} + \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} - \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5})/\sum$

continued on next page

continued from previous page

Control coefficient	**Expression**
$C_6^{S_1}$	$(-\frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{J_{\mathrm{R1}}\ S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{J_{\mathrm{R1}}\ S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{J_{\mathrm{R1}}} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{J_{\mathrm{R1}}})/\sum$
$C_4^{S_1}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{J_{\mathrm{R1}}\ S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{J_{\mathrm{R1}}})/\sum$
$C_5^{S_1}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}})/\sum$
$C_2^{S_1}$	$(-\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ S5\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{J_{\mathrm{R1}}\ S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{J_{\mathrm{R1}}})/\sum$
$C_3^{S_1}$	$(-\frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{J_{\mathrm{R1}}})/\sum$
$C_1^{S_1}$	$(\frac{S5\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{S5\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} + \frac{S5\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} - \frac{S5\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} - \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} + \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} + \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} - \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} - \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5} + \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5})/\sum$
$C_6^{S_5}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}}{J_{\mathrm{R1}}} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}}{J_{\mathrm{R1}}} + \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}}{J_{\mathrm{R1}}} - \frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}}{J_{\mathrm{R1}}} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3})/\sum$
$C_4^{S_5}$	$(-\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}}{J_{\mathrm{R1}}})/\sum$
$C_5^{S_5}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}}{J_{\mathrm{R1}}} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3})/\sum$
$C_2^{S_5}$	$(\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}}{J_{\mathrm{R1}}})/\sum$
$C_3^{S_5}$	$(-\frac{J_{\mathrm{R1}}\ J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}}{J_{\mathrm{R1}}} + \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5} - \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5})/\sum$
$C_1^{S_5}$	$(\varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5} - \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5})/\sum$

Comparison of PySCeS and SymCA control coefficients values

We obtained the numerical values by substituting the steady-state fluxes, concentrations and elasticities computed with `PySCeS`, and compared these results with those obtained by performing numerical MCA with `PySCeS`. We found all control coefficient values were identical (see Table 2.3).

Table 2.3: Numerical values for independent control coefficients generated numerically with `PySCeS` and via substitution of expressions from `SymCA`.

Control coefficient	PySCeS	SymCA
C_{R1}^{JR4}	1.01148218028	1.01148218028
C_{R2}^{JR4}	0.429192201278	0.429192201278
C_{R3}^{JR4}	-0.344819448484	-0.344819448484
C_{R4}^{JR4}	0.0429192201276	0.0429192201276
C_{R5}^{JR4}	-0.0691701231971	-0.0691701231971
C_{R6}^{JR4}	-0.069604030007	-0.069604030007
C_{R1}^{JR6}	0.866011875368	0.866011875368
C_{R2}^{JR6}	-0.410880179643	-0.410880179643
C_{R3}^{JR6}	0.417807711196	0.417807711196
C_{R4}^{JR6}	-0.0410880179641	-0.0410880179641
C_{R5}^{JR6}	0.0838114293819	0.0838114293819
C_{R6}^{JR6}	0.0843371816615	0.0843371816615
C_{R1}^{S1}	0.993170655818	0.993170655818
C_{R2}^{S1}	-0.471210787156	-0.471210787156
C_{R3}^{S1}	-0.33857695614	-0.33857695614
C_{R4}^{S1}	-0.0471210787153	-0.0471210787153
C_{R5}^{S1}	-0.0679178911481	-0.0679178911481
C_{R6}^{S1}	-0.0683439426588	-0.0683439426588
C_{R1}^{S2}	0.840931345899	0.840931345899
C_{R2}^{S2}	0.356824057315	0.356824057315
C_{R3}^{S2}	-0.286677796761	-0.286677796761
C_{R4}^{S2}	-0.795702825064	-0.795702825064
C_{R5}^{S2}	-0.0575070188384	-0.0575070188384
C_{R6}^{S2}	-0.0578677625516	-0.0578677625516
C_{R1}^{S3}	1.5636609581	1.5636609581
C_{R2}^{S3}	-0.741880467969	-0.741880467969
C_{R3}^{S3}	0.754388738274	0.754388738274
C_{R4}^{S3}	-0.0741880467965	-0.0741880467965
C_{R5}^{S3}	-0.748642459513	-0.748642459513

continued on next page

continued from previous page

Control coefficient	PySCeS	SymCA
C_{R6}^{S3}	-0.753338722095	-0.753338722095
C_{R1}^{S5}	-0.713421772307	-0.713421772307
C_{R2}^{S5}	0.338483656292	0.338483656292
C_{R3}^{S5}	-0.344190566299	-0.344190566299
C_{R4}^{S5}	0.033848365629	0.033848365629
C_{R5}^{S5}	-0.0690439706312	-0.0690439706312
C_{R6}^{S5}	0.754324287316	0.754324287316

2.4 Discussion

This chapter has illustrated the initial stages of developing a symbolic control analysis tool, in which the process outlined in [51] was used as a basis. Since the software was implemented for use with `PySCeS` [84], `Python` was used as the programming language. This posed a problem because at the time of development, `Python` lacked a suitable symbolic computing component required to develop the tool. Therefore we needed an external application to fill this void. We tested a number of options, namely `GiNaC` [119] via the Swiginac `Python` interface, and the only available `Python` library, `SymPy`. On the surface `GiNaC` seemed a desirable candidate but the Swiginac interface was a major bottleneck. Whilst the `SymPy` library had limited capabilities, it lacked robustness and was inefficient.

Both `GiNaC` and `SymPy` proved to be extremely ineffective in terms of the time taken to perform symbolic inversions of small matrices, and we decided that the best option would be to develop a `Python` interface to `Maxima`, since the SAGE project had successfully done this [10]. Fortunately, `Python` is a useful language when linking disparate tools, and comes with options as part of a standard installation to serve this purpose.

After initially testing both the `Pexpect` and the `subprocess` modules, we used `subprocess` to interact with `Maxima` from within a `Python` session. On developing the interface to `Maxima`, it became apparent that the extraction of data from `Maxima` after computations was problematic. Initially, we read all data via the standard output pipe of `subprocess`. However, as the size of the systems investigated increased, this proved to present problems. The result was a change in implementation to extract data by using the `Maxima stringout` method which writes the data directly to disk. Even with the improvements in analysis time as a result of this change, the magnitude of data generated could still be a bottleneck in the process.

Symbolic control analysis had been successful for our test model. At this stage, and

before moving onto the next stage of development, we tested numerous other test models consisting of a variety of linear pathways of various length, a collection of branched pathways with varying numbers of branches, a series of moiety-conserved pathways with varying numbers of species making up the moieties and a number of biological models.

Here a number of key biological and computational issues came to the fore. The biological issue involves the sign preceding each product of elasticities in the symbolic expressions for control coefficients. Since these symbolic expressions are ratios, it is possible to change the sign preceding any term, as long as the signs preceding all other terms are also changed throughout both numerator and denominator. However, as discussed by Hofmeyr [49], a stable steady-state requires a positive denominator. This implies that the signs preceding each product of elasticities must be such that the overall value of the denominator is positive, taking into account 'normal' values of elasticities ('normal' values for substrate elasticities are positive and for product elasticities are negative). We found that the values of the denominator expressions provided by this initial version of SymCA violated this requirement in certain cases. As an example, consider the first two terms in Equation 2.6:

$$\frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} \tag{2.7}$$

$$-\ \frac{J_{\mathrm{R1}}\ J_{\mathrm{R6}}\ S5\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{J_{\mathrm{R1}}\ S4} \tag{2.8}$$

The first term (2.7) is positive because it contains only fluxes (which are positive by definition), substrate elasticities and is preceded by a plus sign (note that in our usage the value of a term takes its preceding sign into account). The term in 2.8 is preceded by a minus sign, but its overall value is also positive since it contains one (negative) product elasticity and three substrate elasticities. This would not have been the case had Term 2.7 been preceded by a minus sign and Term 2.8 by a plus sign, which could have resulted from a different ordering of rows and columns in the C and E matrices. It therefore became clear that we needed to include a step in our algorithm to check whether matrix inversion produces the correct signs preceding the denominator terms and to apply a correction when these signs were incorrect. The expressions in Equations 2.7 and 2.8 also illustrate the need for recognising the possibility for simplifying individual numerator and denominator terms by cancelling entities such as JR1. If our tool were to be generic, we had to resolve these issues in the next stage.

The second major issue was purely computational. For larger systems it was impossible to read the entire matrix of symbolic control coefficients over the standard output file handle native to `subprocess`. This was critical in that we needed to extract expressions from within `Maxima` to perform further in-depth analyses on them at a later stage. We had successfully shown that the development of a symbolic control analysis tool was viable, and so began the development of a piece of generic software with additional functionality to increase our understanding of the regulatory behaviour of

cellular systems. The ultimate aim was to develop a truly generic application that was portable to different operating systems. We will cover the solutions to these problems as well as their significance in the next chapter, which encompasses the development of the symbolic control analysis module into a generic application.

Chapter 3

SymCA: Development of generic symbolic control analysis software application

3.1 Introduction

The previous chapter provided a proof of concept of symbolic control analysis, (SCA), in terms of developing a software package to perform the computation. This initial study also highlighted a number of key problems which we needed to solve to develop a truly generic piece of software, which was the ultimate goal of this study. We showed that a `Python` interface with `Maxima` is possible, and that `Maxima` is a suitable external application to use to equip `Python` with previously lacking symbolic computing power[1]. In Chapter 3 we will address and optimise the time taken to perform SCA as far as possible, considering the inherent constraints associated with performing symbolic computations.

Since `Python` is inherently object-oriented, the natural progression was to convert the previously developed script into an object-oriented software application. Chapter 3 provides an in-depth view of this process, and also addresses the known issues that arose in the proof of concept phase. This chapter gives an overview of the processes involved in symbolic control analysis as well as an in-depth look at each of the classes making up the `SymCA` application.

[1]Since the development of `SymCA`, there have been significant advances in the development of an external `Python` library (`SymPyCore`) that addresses this deficiency, which will be addressed in the Discussion

3.2 SymCA overview

In its current form `SymCA` consists of 12 classes, each of which plays an integral role. Each class performs specific functions required during the computations, and we will discuss these roles in the following sections. The figure below provides an overview of the relationships between these classes.

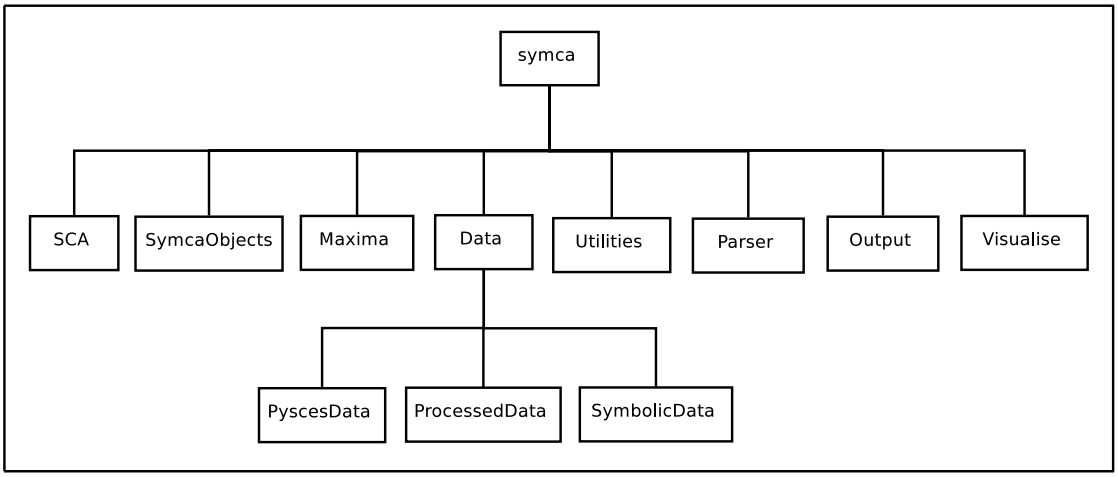

Figure 3.1: Overview of SymCA classes

`SymCA` creates a dedicated directory for the model in a `symca` directory which is located in the designated `PySCeS` output directory. We loaded a correctly formatted input file into a `PySCeS` session creating a `PySCeS` model instance, and performed a stoichiometric analysis. A `PySCeS` input file is required as `SymCA` was designed to be an additional module for `PySCeS`. Of critical importance here is that the input file need not contain any kinetics and that numerical simulation is not required. However, the capability of doing both numerical and symbolic simulations provides a useful cross-checking tool and importantly allows control pattern quantification for a specific steady-state. We then extracted data from the model instance, processed it, and converted it into its symbolic equivalents. We inverted the **E** matrix using `Maxima`, factorised, extracted and processed all control coefficients, and then saved to disk in the form of a `Python` pickle object.

Once the symbolic control coefficients have been computed and returned by `Maxima`, the user can manipulate and perform various operations with them. There is also an option to write LaTeX output files for the different operations, to allow for a closer inspection of the expressions. The option to display various of the operations in graphical views of the system in question, is also available. In the following sections we will describe these methods as well as others. We will also give an account of the roles and algorithms used in the various classes.

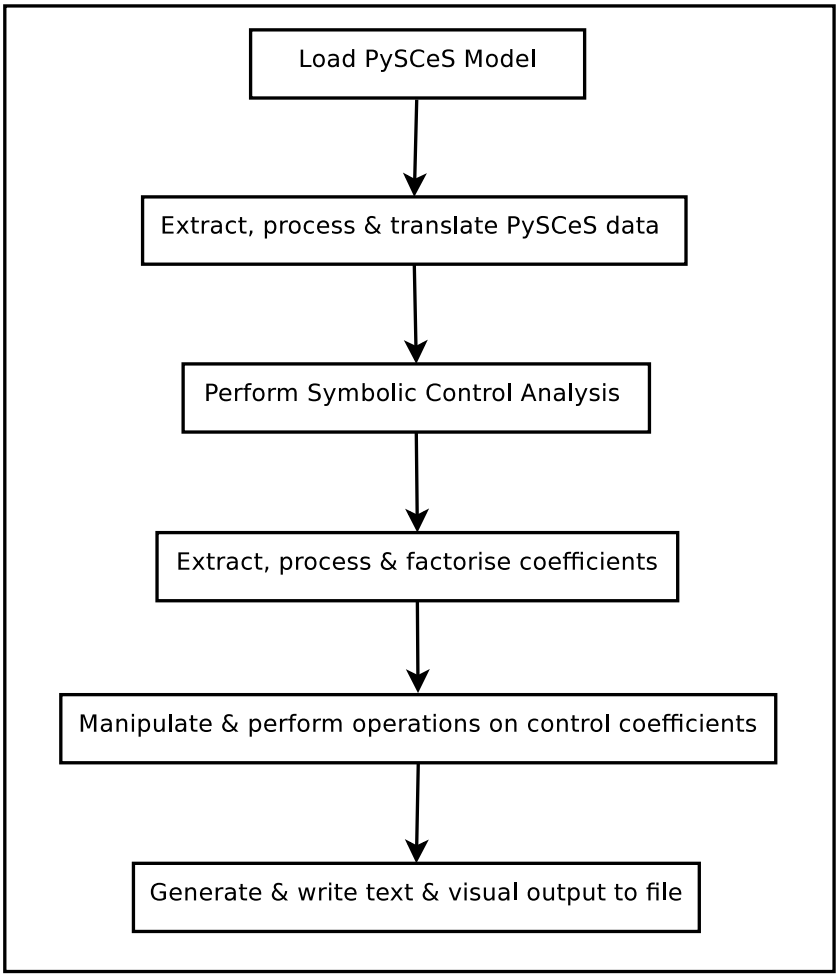

Figure 3.2: SymCA workflow

3.3 Basic requirements

As discussed in Chapter 2, the basic requirements for `SymCA` are a symbolic computing environment, i.e. `Maxima`, a solid programming language, `Python`, and a tool which allows us to perform a stoichiometric analysis of any arbitrary system, whilst also giving access to the data associated with such an analysis, `PySCeS`.

In the previous chapter we have shown that it is possible to incorporate `Maxima` from within a `Python` session, hence the development of a generic implementation of the control-matrix equation was well within our grasp. We had successfully developed a

basic Python interface to Maxima using the subprocess module, and expanded this to provide additional functionality as well as the ability for use on Windows platforms. The initial script we developed (Chapter 2) provided a solid foundation for the development of the generic symbolic control analysis software application, which we refer to as SymCA.

3.4 SymCA classes

SymCA consists of 12 classes with the symca class being the main class (Figure 3.1). This section covers all methods for the symca class, as well as a brief overview of the remaining classes. Appendix A includes a detailed account of all methods for the remaining classes.

3.4.1 symca

Since it is the main class SymCA serves as the 'controller', and has key operations which must be carried out for the successful performance of symbolic control analysis.The symca class can take three arguments on instantiation, of which the first is required and the other two optional. The required argument is the name of the PySCeS input file to be loaded. The first optional argument is the name of the directory in which the PySCeS input file is located in the event that it is not present in the default PySCeS model directory. The second optional argument is set by default to False and governs the LaTeX output generated after all operations have been completed. This argument removes all common substrings found in the labels of reaction names in the input file when set to True, i.e. if each reaction label begins with 'R' or 'rxn', this common substring will be removed from the LaTeX output generated.

Once symca is instantiated, it checks for the presence of the input file in either the default PySCeS model directory or in the directory provided. If the file is absent, a list of files present in the selected directory is displayed and the user is prompted to enter a file from the list. When the file is present a number of directories are created within the PySCeS output directory. The first occurrence SymCA is run will create a symca directory within the PySCeS model output directory; this directory will house all subsequent directories created by SymCA.

For each model run, we created the following directories:

- latex: stores all LaTeX output
- maxima output: stores all output generated from Maxima
- pickle objects: stores all pickled data.

Once SymCA had created all the directories, we loaded the model into PySCeS and extracted all the relevant data by means of the Data class. The Data class is responsible for extracting, processing and creating the symbolic data equivalents, and we will discuss this in more detail in Section 3.4.2.

After successfully creating all symbolic data, we began the process of symbolic control analysis. The `SCA` class is primarily responsible for this operation, and we will address it in Section 3.4.7. Below is a detailed description of the methods associated with the `symca` class:

- `getDir()` - The `PySCeS` model and output directories are obtained. The `Python os.path` module is then used to check if a 'symca' subdirectory exists within the `PySCeS` output directory. In the event of the `symca` directory existing no further action is required, but if absent, it is created.

- `checkModelStatus()` - This method checks to see if the `PySCeS` input file is located either in the native `PySCeS` model directory or in the directory provided on instantiation of the `symca` class. True or False is returned, depending on whether or not the file is located.

- `createAllDir()` - `SymCA` creates a directory for each model within the `symca` subdirectory, which is itself located in the `PySCeS` output directory. A `layout` directory is also created within the `PySCeS` output directory to house all svg and xml model layouts generated via `SBW`. Once a model directory, which uses the model name as its name, has been created within the `symca` directory, a number of subdirectories are created to store the various data generated by `SymCA`. The names of the subdirectories are passed to the method responsible for their creation.

- `createSub()` - This method creates the three subdirectories required per model, i.e. `latex`, `maxima output` and `pickle objects`. A list of the directory names serves as the argument for this method, the list is then stepped through and each directory is created if it is not already present.

- `loadModel()` - Since `PySCeS` is required to perform the stoichiometric analysis as well as computing all steady-state values, the model must be loaded into a `PySCeS` instance. Once loaded, a check is done to detect if any zero fluxes exist at steady-state.

- `checkZeroFlux()` - This method is called by `loadModel()` to check if any steady-state fluxes are equal to zero. In the event of this occurring the `PySCeS elas_zero_flux_fix` variable is set to True.

- `loadData()` - Once a model has been loaded into `PySCeS` all data required must be extracted, processed and the required symbolic data generated. These tasks are handled by the `Data`, `PyscesData`, `ProcessedData` and `SymbolicData` classes, with the Data class being the controlling class throughout all data extractions and manipulations. The data is then returned from the Data class.

- `doSca()` - This is the main method governing the symbolic control analysis procedure, and has a number of optional arguments:

  - `dep`: This argument controls whether or not dependent control coefficients will be computed and by default is set to True.
  - `load`: If symbolic control analysis has been performed previously on a model, there is an option to reload all previously stored data. This argument is set by default to False.
  - `subE`: This option performs elasticity substitutions prior to computation of control coefficients. This argument is set by default to False.
  - `subDict`: In the event of `'subE'` being True, a dictionary of all substitutions to be performed must be provided. The default value is None.

  The first argument checked is `load`. When `load` is False, the method directly responsible for symbolic control analysis is called `symbolicControlAnalysis()`. This then pickles all resulting data for future use. However, if `load` is set to True, the pickled data from a previous session is loaded via `loadSymcaData()`, which returns a tuple of data if the pickled data is found. If the pickled data is not found, False is returned. In the first case when the data is successfully unpickled, the dimensions of the pickled **E** matrix and those of the newly computed **E** matrix are compared via `compareMatrices()`. If True, all the data is reloaded into `Maxima` so that further operations may be performed. However, if False is returned, `symbolicControlAnalysis()` is called as happens when no pickled data is found for the model in question.

- `symbolicControlAnalysis()` - This method performs symbolic control analysis by creating an `SCA` instance and then calling the `SCA` instances `computeControlCoefficients()` method. There are three arguments required by this method `dep`, `subE` and `subDict` (the descriptions for each method appear in the `doSca` section).

- `pickleSymcaData()` - Data must be stored efficiently for future evaluation. This method saves both the `_symcaModelMap` and `_symcaModelOut` dictionaries in the form of a `Python` pickle. The dictionaries are stored in the `$PSCOUT/symca/`*`model name`*`/pickle objects` directory, *model name*_ModelMap and *model name*_ModelOut, respectively.

- `loadSymcaData()` - This method unpickles the data pickled from a previous session to enable access to the data. The unpickled `_symcaModelMap` and `_symcaModelOut` are then returned as a two-membered pickle.

- `loadMaximaData()` - When a session of `SymCA` is ended, the `Maxima` session is also closed and all associated data is lost. This method addresses this shortcoming by recreating the `Maxima` session by inputting all unpickled denominator and control coefficient expressions, so that the user can perform further operations.

- `setElasticityPost()` - This method enables the user to perform elasticity substitutions after all control coefficients have been computed. There are two arguments for this method:

  - `subDict`: This is an elasticity substitution dictionary wherein key value pairs consist of elasticity labels and substitution values.
  - `subType`: Two types of post elasticity substitution exist:
    * 0 - performs all substitutions simultaneously
    * 1 - performs each substitution individually as well simultaneously.

  This method calls the method of the same name in the `SCA` instance which takes the same arguments.

- `symca2Pysces()` - This method performs numeric substitution with steady-state values from `PySCeS` and computes the control coefficient values for all coefficients, and compares them with those computed numerically via `PySCeS`. This method calls the `SymcaToPysces()` method from the `SCA` instance.

- `computeResponse()` - `PySCeS` is used to compute all response coefficients. This methods calls the method of the same name in the `SCA` class.

- `getResponse()` - This method also calls a method by the same name in the `SCA` class which will select response coefficients based on the arguments supplied when the method is called. These options include the arguments `cutoff`, `flux` and `param`. The `cutoff` argument controls the value used to select the response coefficients, `flux` is a list of fluxes or species for which the user would like the response coefficients. If provided, `param` selects all response coefficients for the list of parameters.

- `computeCoControl()` - This method computes symbolic co-control coefficients. It requires a single argument, a list of tuples where each tuple consists of a numerator control coefficient label, denominator control coefficient label and reaction in question.

- `computeSummations()` - The summation theorem is computed by calling an identically named method as found in the `SCA` class.

- `controlPatterns()` - This method takes two optional arguments, both of which are None by default:

  - `ccList`: This list of fluxes or species is used to generate a list of all control coefficients for which quantified control patterns are desired. If no list is provided, the control patterns for all control coefficients are quantified,
  - `tol`: This value selects which control coefficients will be evaluated, i.e. 0.3 indicates that all control coefficients with a value above 0.3 will evaluated. If no value is provided, the default is 0.1.

  This method calls the `SCA` instance `getControlPatterns()` method.

- `patternScan()` - In this method control pattern quantification is performed at every step of a parameter scan. There are four arguments:

  - `cc`: This is a list of control coefficients to be scanned.
  - `param`: This is a list of parameters.
  - `range`: The range of scan is in the region of 0–1 where 1 indicates 100%. The steady-state parameter value is scanned within this range in both directions.
  - `steps`: This is the number of values to be scanned in the designated range.

- `simplify()` - This method simplifies the control coefficient numerators and the common denominator so that all common terms are removed. The `simplify` method of the `SCA` instance is called which performs this function.

- `writePickle()` - This method pickles the data into the desired location provided as the arguments. The reference to the pickled location is then returned.

- `loadPickle()` - This method takes a pickled object as its only argument and attempts to load the pickled object. Once loaded the unpickled data is returned.

- `getControlCoefficients()` - This method returns the dictionary of all Control-Coefficient objects.

- `getCommonDenominator()` - This method returns the Denominator object.

- `getImageCoords()` - In this method the x, y co-ordinates of all fluxes and species are extracted from the layouts generated via `SBW` and returned as a dictionary.

- `SBMLData()` - The generation of model layouts is handled via `SBW`, and all data associated with the fluxes and species is then extracted from the file containing the layout information. This data is then returned as a tuple.

- `viewElas()` - Once `SBW` has successfully generated the layouts required by `SymCA`, the steady-state values can be mapped onto these images. This is achieved via the `Visualise` class, which is an SVG editor. This method has several arguments:

  - `par`: Default is False and if set to True includes the parameter elasticities in the image.
  - `colours`: A default list of ten colours is included. However, if the user prefers custom colours they can provide them via this argument.
  - `colour_codes`: If a list of custom colours is provided a corresponding dictionary with key value pairs, i.e. colour and hexadecimal colour code, is required.
  - `range_max`: This is the maximum value to be used for the elasticity value range.
  - `image_type`: Default is None, otherwise it can be set to ‘eps’, ‘ps’, ‘pdf’ or ‘png’.

- `viewControlPatterns()` - In the same way as elasticity values can be visualised so can the quantified control patterns for any given control coefficient. This allows for the visualisation of regulatory pathways within a control coefficient. A number of options exist for this method, all of which are governed by the following options:
  - `below`: Default is False, and if set to True images for all control patterns falling beneath the `pattern_cutoff` value will be generated.
  - `cc`: This is a list of control coefficients for which control pattern images are desired.
  - `flux_species`: This is a list of fluxes, or species, used to generate a list of control coefficients for which control pattern images are desired.
  - `colours`: This is a custom list of colours.
  - `colour_codes`: This dictionary corresponds to the custom list and key value pairs are colour and hexadecimal colour code.
  - `pattern_cutoff`: This value determines which control patterns are selected, i.e. 0.4 indicates that all control patterns contributing more than 40% will be displayed.
  - `range_max`: This is the maximum value to be used for the colour range.
  - `image_type`: Default is None, otherwise it can be set to 'eps', 'ps', 'pdf' or 'png'.
- `writeData()` - This method handles the output of all data types in LaTeX format, and calls the `SCA` class method of the same name to perform the actual task. This method takes two arguments. The first argument is the type of data to be written and can be one of the following, 'cc', 'sym2psc', 'patterns', 'summation', 'coc' or 'scan'. The second argument, `'file'` is optional and when given indicates the directory to which the data must be written. If emitted the data is written to the latex subdirectory for the model in question, i.e. `$PSCOUT/symca/`*`model name`*`/latex/`.

3.4.2 Data

Before performing SCA, we required the extraction, processing and generation of the symbolic data from the `PySCeS` data, and so will discuss this first. The `Data` class is the master class controlling all processes involved with the extraction and processing of all forms of data. To achieve this there are four separate classes, each dealing with a different aspect of the data manipulation:

- `PyscesData` - Handles all data extraction from `PySCeS`
- `ProcessedData` - Responsible for processing of `PySCeS` data before the creation of the symbolic data required by the `SCA` class

- `SymbolicData` - Generates all symbolic data from the processed `PySCeS` data
- `Utilities` - Contains all methods used by multiple classes

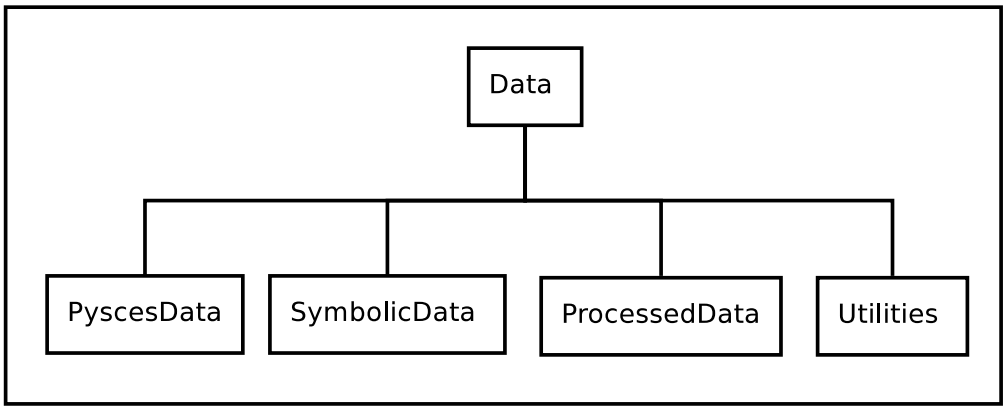

Figure 3.3: Data classes

The `Data` class takes five arguments when instantiated:

- `psc_mod` - The `PySCeS` model name
- `Maxima` - `Maxima` instance
- `pysces` - `PySCeS` instance
- `allDir` - Tuple consisting of the maxima output directory and the maxima stringout directory
- `format` - True to remove common reaction label substring for LaTeX output and False to keep substring; default is False

3.4.3 PyscesData

This class is responsible for extracting the various data objects from `PySCeS` which will be used by the `ProcessedData` class. `PyscesData` requires three arguments on instantiation, i.e. the `PySCeS` model in question, and the `Maxima` and the `PySCeS` instances. There are eight main methods in this class each tasked with extracting specific data from within `PySCeS`. Some of these methods have nested methods which perform specific tasks. `PyscesData` relies on two further classes: `SymcaObjects` and `Utilities` (Figure 3.4).

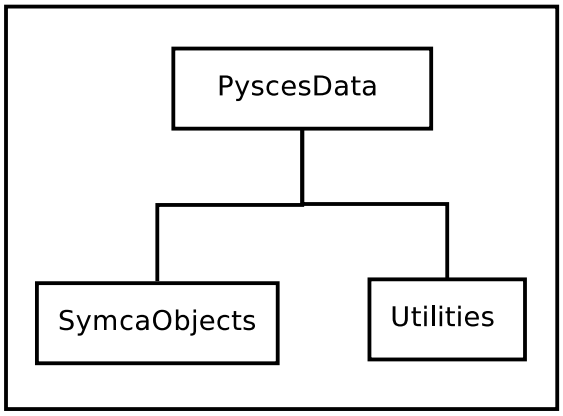

Figure 3.4: PyscesData classes

3.4.4 ProcessedData

The `ProcessedData` class is responsible for taking the newly extracted `PySCeS` data and manipulating it so that it is in the correct format for the generation of the required symbolic data for use in `Maxima`. Four arguments are required on instantiation – the `Maxima` instance, the dictionary of initial `PySCeS` data, the `PySCeS` model output directory and the `Maxima` stringout directory. When instantiated, an instance of the `Utilities` class is created, which contains a number of methods commonly used by many classes. We will discuss these methods in Section 3.4.11. There are five main methods which perform this task, some of which contain varying degrees of nested methods which appear in Appendix A. Like the `PyscesData` class, both `SymcaObjects` and `Utilities` are required to process and store the initial `PySCeS` data.

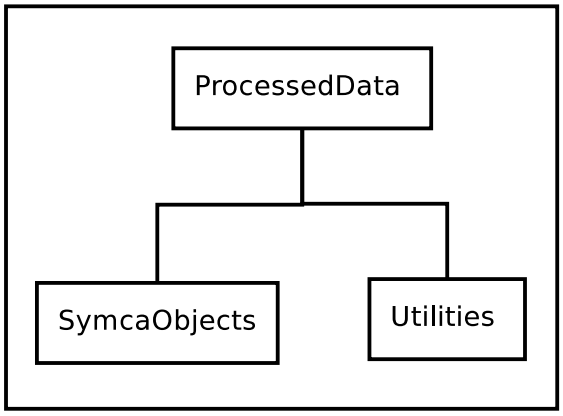

Figure 3.5: ProcessedData classes

3.4.5 SymbolicData

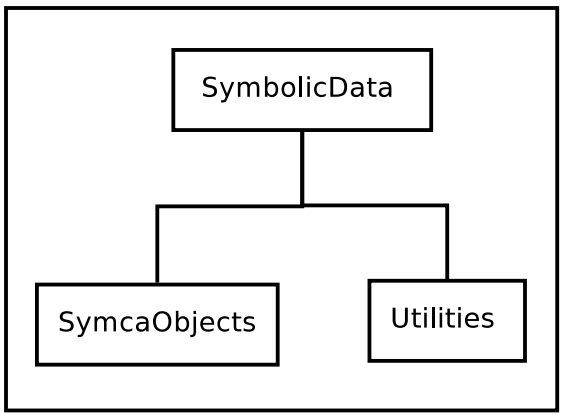

Figure 3.6: SymbolicData classes

To perform symbolic control analysis, we need symbolic equivalents of the initial `PySCeS` data. This class is responsible for the final stage in this process in that it takes the processed `PySCeS` data and generates the symbolic data. To achieve this there are five main methods, some of which have nested methods. `SymbolicData` takes three arguments when instantiated – the `Maxima` instance, the processed data dictionary and True or False depending on whether or not LaTeX formatting is required. Figure 3.6 outlines the classes involved in this process.

3.4.6 SymcaObjects

A number of custom objects were created, many of which are responsible for storing the information required during the initial generation of symbolic data. A few custom objects are used to store the final output generated via `SymCA`. The following section highlights attributes for each of the final output objects, and details any methods which the object may have. The details for all other objects appear in Appendix A.

- `ControlCoefficient` - ControlCoefficient objects contain all information associated with the symbolically generated control coefficients. The numerical value for each control coefficient is computed on instantiation of a ControlCoefficient object. These objects have several methods and attributes (Table 3.1) which are accessible to the end-user.
  - `clear()` - This method resets the following object attributes to None: terms, expression, latex, control_patterns.
  - `controlPatterns()` - This method computes the quantified control patterns for the control coefficients in terms of their value and percentage contribution. All patterns are stored in a dictionary which is then pickled, after which the control_patterns_ref attribute is initialised.

Table 3.1: Description of ControlCoefficient object attributes

Attribute	**Description**
cc_type	either flux or concentration
name	label of the control coefficient
maxima	the `Maxima` label for the control coefficient
mx_pos	index of the coefficient in its respective matrix
pickle_ref	path of pickled control coefficient expression
denom_maxima	`Maxima` label for the control coefficient denominator
elasticities	list of all elasticities found in control coefficient expressions
pysces_values	dictionary of all `PySCeS` steady-state values
latex_equivalents	dictionary of all LaTeX labels
expression	symbolic expression for the control coefficient
terms	a tuple containing a list of control patterns, a list of signs for each control pattern, a list of absolute control patterns and finally the total number of control patterns found
latex	LaTeX representation of expression
ss_value	steady-state control coefficient value
control_pattern_ref	pickle path for pickled control pattern data
control_patterns	dictionary of all quantified control pattern data
substituted	True or False depending on whether or not elasticity substitutions have been performed
substitution_arg	elasticity substitution argument if substituted is True, otherwise None
substituted_cc	dictionary of all resultant ControlCoefficient objects in the event of elasticity substitutions

- `get()` - This loads the pickled control coefficient expression and initialises the expression attribute with the symbolic expression.
- `getControlPatterns()` - This checks to see if the control_pattern_ref is present, and if so the control pattern data is loaded and assigned to the control_patterns attribute.
- `getElas()` - This generates a list of all elasticities present in the expression, and assigns the list to the elasticities attribute.
- `getLatex()` - This method generates the LaTeX form of the symbolic expression for the various output options available.
- `getSubCC()` - The substituted control coefficient values are obtained by calling this method. A dictionary is created in which the keys are the substitution arguments and the values correspond to the substituted coefficient values. The dictionary is returned.
- `getTerms()` - This method takes the control coefficient expression and generates a list of all control patterns. The `getTerms()` method in the Utilities

class is called, which returns the following data: a list of control patterns, a list of signs for each control pattern, a list of absolute control patterns and finally the total number of control patterns found. This data is then assigned to the terms attribute.

- `getValue()` - The numerical value of the control coefficient is calculated from the symbolic expression and the result is assigned to the ss_value attribute.
- `pickleExpression()` - This method pickles the control coefficient expression.

- `CoControlCoefficient` - These objects were developed to store all data relating to the algebraically generated co-control coefficients. Like the ControlCoefficient objects, these objects have numerous attributes as well as methods:

Table 3.2: Description of CoControlCoefficient object attributes

Attribute	Description
name	co-control coefficient label
maxima	`Maxima` input label
numerator_cc	label for numerator control coefficient
denominator_cc	label for denominator control coefficient
reaction	the reaction common to both numerator and denominator control coefficients
pickle_ref	the pickle directory for the expression
ss_value	value of the co-control coefficient at steady-state
pysces_values	`PySCeS` dictionary of all steady-state value
latex_equivalents	LaTeX equivalents dictionary
substituted	True or False depending on whether or not the co-control coefficient has been computed with substituted numerator and denominator control coefficients
substitution_arg	the substitution argument in the event that substituted is True
expression	symbolic co-control coefficient expression
elasticities	list of all elasticities found in the expression
terms	list of all terms in the expression
latex	the LaTeX form of the expression

- `clear()` - This method sets the following attributes to `None`: expression, terms and latex.
- `get()` - This method unpickles the co-control expression and initialises the expression attribute with the unpickled expression.
- `getElas()` - This method determines all elasticities found in the expression and initialises elasticities attribute with the list of all elasticities.
- `getLatex()` - This method generates the LaTeX form of the expression and initialises the latex attribute with the LaTeX string.

  - `getTerms()` - This method determines all terms present in the expression and initialises the terms attribute with this data.
  - `getValue()` - This method computes the value of the co-control coefficient, whereby the ss_value attribute is initialised. This method is called on instantiation of the object.
  - `pickleExpression()` - This method pickles the expression.

- `Denominator` - The denominator represents the common denominator for all control coefficients generated by `SymCA`. Since this expression can sometimes be large, we decided to handle it separately. The denominator is essentially the determinant of the **E** matrix and is computed at the same time as the matrix inversion. Like the ControlCoefficient object, the Denominator has a number of methods and user accessible attributes (Table 3.3):

Table 3.3: Denominator object attributes

Attribute	**Description**
name	denominator label
maxima	`Maxima` denominator label
pickle_ref	path referring to pickle expression location
pysces_values	dictionary of all `PySCeS` steady-state values
latex_equivalents	dictionary of all flux, species, elasticity and control coefficient labels
elasticities	list of all elasticities found in the denominator expression
expression	symbolic common denominator expression
terms	four-membered tuple containing list of control patterns, list of control pattern signs, list of absolute control patterns and the number of control patterns
latex	LaTeX representation of denominator expression
ss_value	steady-state value of denominator
value_label	`Maxima` label for the denominator steady-state value
substituted	True or False, depending on whether or not elasticity substitutions have occurred
substitution_arg	elasticity substitution argument in the event of elasticity substitutions
substituted_denom	dictionary containing denominator objects arising from elasticity substitutions

  - `clear()` - This method resets the following attributes to `None`: terms, expressions and latex.
  - `get()` - This method attempts to load the pickled denominator expression, and if successful the expression attribute is initialised.

- `getElas()` - This method uses the Utilities class `getCCElas()` method which returns a list of all elasticities in the expression. The list is then assigned to the elasticities attribute.
- `getLatex()` - This method uses the `getLatex()` method from the Utilities class to generate the LaTeX representation of the denominator expression.
- `getTerms()` - This method takes the denominator expression and generates a list of all control patterns, a list of signs for each control pattern, a list of absolute control patterns and finally the total number of control patterns found. The `getTerms()` method in the Utilities class generates the data which is assigned to the terms attribute.
- `getValue()` - The value of the denominator is computed on instantiation of the Denominator object, and the value is assigned to the ss_value attribute.
- `pickleExpression` - The denominator expression is stored as a `Python` pickle, and this method is responsible for pickling the expression.

3.4.7 SCA

The `SCA` class performs the symbolic control analysis routine introduced earlier, and drives further operations involving the symbolic control coefficients. This class requires the `Python` interface with `Maxima` as a means of connecting to and performing the subsequent analysis. We will address the `Maxima` interface in Section (3.4.8).
`SCA` requires the following arguments on instantiation:

- `maxima` - `Maxima` instance
- `pysces` - `PySCeS` model instance
- `model` - Current `PySCeS` input file name for model being analysed
- `dep` - True to compute dependents and False to compute only the independents
- `format` - True to remove a common reaction substring (if present), otherwise False
- `modelMap` - A dictionary containing all required data
- `modelOut` - A dictionary to store all data resulting from SCA and its various operations
- `allDir` - A five-membered tuple containing the following directory paths: 0 - main model output directory, 1 - directory storing all `Maxima` generated output, 2 - `Maxima` output directory in the format required by the `Maxima` `stringout` method, 3 - directory housing all pickled `Python` objects and 4 - directory to store all LaTeX output

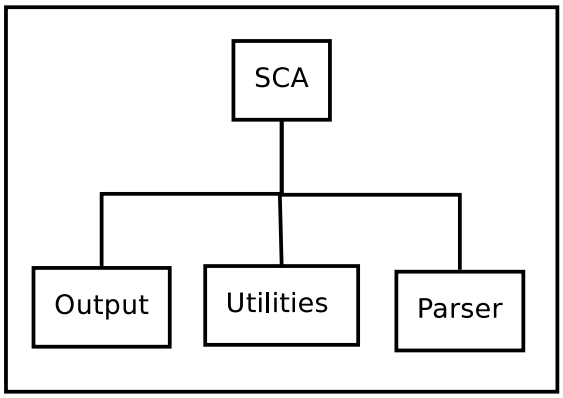

Figure 3.7: SCA classes

The `SCA` class relies on the `Parser`, `Output` and `Utilities` classes (Figure 3.7) to perform the various processing tasks that are required after all symbolic control coefficients have been generated. We will discuss these classes in the following sections. To perform symbolic control analysis we require several methods, each with a specific role. The details of these methods appear in Appendix A.

3.4.8 Maxima

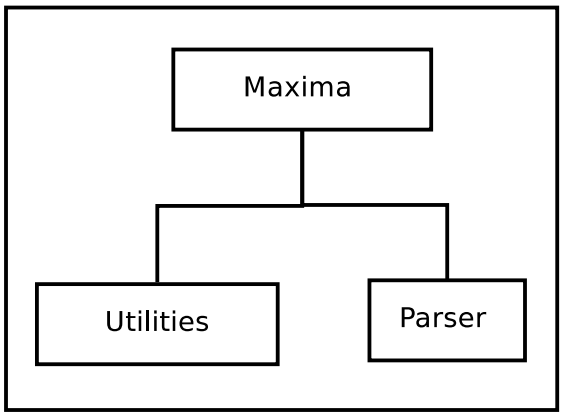

Figure 3.8: Maxima classes

The ability to communicate with the `Maxima` computing environment was crucial to the viability of this study. As we demonstrated in Chapter 2, we were able to use the native `Python subprocess` module to communicate with `Maxima` from within a `Python` session. This section describes how the `Maxima` interface was developed and how it is used to perform all symbolic computations.

The sole purpose of the `Maxima` class is to perform the symbolic computations, and we have written a number of methods to aid this process. This class consists of 17 methods, some of which wrap native `Maxima` methods and others establish the connection and subsequent communication with `Maxima`. An instance of the `Maxima` class is established requiring the `Maxima` stringout directory as the only argument. This is important, as the directory separators for the directory argument, which the `Maxima stringout` function requires, are in standard Linux format where each directory is separated by a '/'. This applies to `Maxima` being run on both Windows and Linux platforms. The additional classes required by this class are shown in Figure 3.8.

3.4.9 Parser

All data generated by `Maxima` and the various operations associated with symbolic control analysis require varying degrees of processing. `Parser`, which is used by the `Maxima`, `Output` and `SCA` classes, contains five methods each of which is responsible for handling a specific data set. The details of these methods appear in Appendix A. This class uses a number of methods from `Utilities` to fulfill all parsing needs.

3.4.10 Output

All LaTeX output generated by `SymCA` is written via this class, which contains 17 methods to generate LaTeX for the various types of data available. Instantiation of the Output class requires nine arguments:

- `max` - `Maxima subprocess` instance
- `data` - A dictionary of all `SymCA` output
- `model_dir` - Main model output directory
- `model` - Name of the model
- `dep`: True to compute both dependent and independent control coefficients, False for only independent control coefficients
- `all_dir`: A tuple containing the paths for the maxima and LaTeX output directories
- `modelMap`: A dictionary of all initial data
- `latex_equivalents`: A dictionary of all variable names and their corresponding LaTeX representation
- `format`: True if one wants any common reaction name substrings to be removed from all LaTeX output, else False to leave in

All LaTeX output is written to the latex subdirectory, $PSCOUT`/symca/model name/latex/`. This class makes use of `Parser` and `Utilities` classes. The details for all these methods appear in Appendix A.

3.4.11 Utilities

This class contains all the methods which are used by a number of the other classes, and is instantiated with three optional arguments – the `Maxima subprocess` instance, a dictionary of all initial data, and a boolean influencing reaction labelling in LaTeX. All arguments are initialised to None when they are not provided. Because of the nature of `Utilities` and essentially being the 'Jack of all trades', it has numerous methods each performing a specific function. There are 48 methods, some used by more than one class, others used by one class only, and a few called internally by `Utilities`; these details appear in Appendix A. A key method in this class is `checkSign()`, which was designed to ensure that the signs associated with the control coefficient equations generated by `Maxima` were correct.

3.4.12 Visualise

We developed the `Visualise` class to generate graphical representations to complement some of the methods and additional features that have been incorporated into `SymCA`. Generating layouts for biochemical networks can be a demanding and challenging exercise, and for this reason we used external tools. The systems biology workbench, (`SBW`), is one tool which we selected for use in this research [35]. Two modules within the `SBW` framework enabled us to draw a network, `DrawNetwork`, which we then rendered into SVG (Scalable Vector Graphics) using the `SBWLayoutModule` module. We then extracted the SVG layout from `SBW` and used it as a starting point for the graphics developed in this thesis.

Since the network layout is exported in the SVG format, it is possible to edit and customise the underlying SVG code, and what we describe here is in essence an SVG editor. The following methods we discuss were required to form our desired end-product, which are intuitive graphics used to display some of the data generated via `SymCA`. At that time, this required an installation of the `SBW` workbench to generate the initial layout, and this could be achieved in the following ways.

Firstly, we could load the SBML [61] file for a specified model manually into the DrawNetwork module where the initial XML layout was generated, and then load the XML layout into the SBWLayout module whereby it is converted and outputted as a SVG file. Secondly, we could perform the above automatically, whereby the layout generated by `SBW` was our choice for all future visualisations. If we used the first option, we could 'tweak' the initial `SBW` generated layout. Once saved, this layout would serve as the basis for all future visualisations, and all that the user would need to do is store both the .xml and .svg files for each model in the desired location within the `PySCeS` output directory, i.e `$PSCOUT/layout/`*`model name`*`/`.

To achieve the level of customisation required for displaying the `SymCA` data, we developed a number of methods within the `visualise` class. Two additional classes within

this class are required to save key data in the form of objects, `SBMLelasticity` and `SBMLreaction`. The attributes for these two classes are summarised in the following tables, and a detailed description of the methods appear in Appendix A.

Table 3.4: SBMLelasticity object attributes

Attribute	**Description**
name	elasticity label in format as used by `SymCA`
species	elasticity species
glpyh_id	SVG species identifier
text_glyph_id	SVG species label identifier
curve_id	curve identifier for elasticity
role	product, substrate or modifier
reaction	elasticity reaction

Table 3.5: SBMLreaction object attributes

Attribute	**Description**
name	reaction label in format as used by `SymCA`
glpyh_id	SVG reaction identifier
text_glyph_id	SVG reaction label identifier
substrates	list of all associated substrates
products	list of all associated products
modifiers	list of all associated modifiers

3.5 Discussion

Chapter 3 gave an overview of the design and workings of `SymCA` as well as a brief introduction to all the classes that make up `SymCA`. This phase of the thesis involved addressing numerous issues which arose during the proof of concept, such as ensuring that the computed expressions had the correct signs ($+$ or $-$), incorporating steps which ensured that any redundancies were removed from the expressions and solving the problem of data extraction from `Maxima`.

During this stage we developed the additional functionality, including methods for control pattern quantification, co-response coefficient calculation, the generation of LaTeX output and the generation and mapping of data onto graphical representations of the system under investigation. Chapter 4 shows example graphics, which will illustrate how `SymCA` works by means of an example system. This chapter will also cover all the features and options mentioned throughout Chapter 3, whilst demonstrating the usability of `SymCA`.

Chapter 4

SymCA at work

4.1 Introduction

The previous chapter revolved around the development and implementation of `SymCA`. This chapter focusses on the use of `SymCA` by means of an example model. We designed the model to incorporate all features common to biological features, i.e. linear segments, branches and moiety conserved cycles.

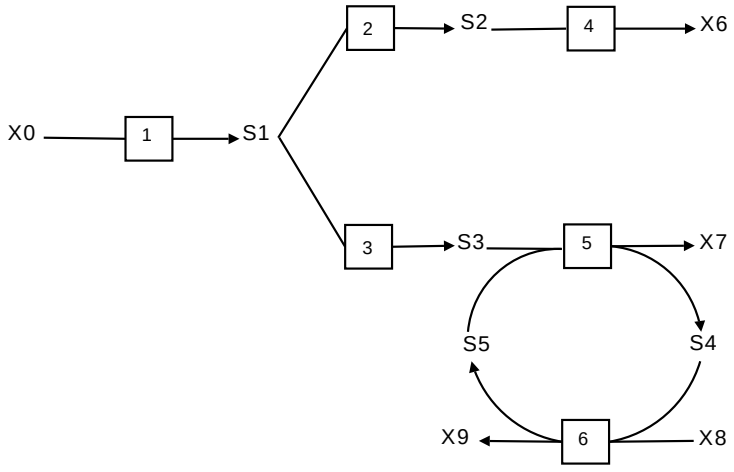

Figure 4.1: Theoretical system to demonstrate `SymCA`, as previously used to describe the control-matrix equation in Section 1.5.3.

4.2 Beginning a SymCA session

`SymCA` can be used in one of two ways, either by command-line interaction or via a `Python` script. We have used the former to describe how `SymCA` works, and included a simple `Python` script to demonstrate certain features. An interactive `Python` or `IPython`

shell is required for command-line interaction and then the `SymCA` module is loaded. The text below shows what to expect on successfully importing `SymCA`:

```
[tim@unknown-00-13-d3-f0-92-59 ~]$ ipython
Python 2.5.2 (r252:60911, Jan  8 2009, 15:49:56)
Type "copyright", "credits" or "license" for more information.

IPython 0.8.4 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object'. ?object also works, ?? prints more.

In [1]: import symca
Using matplotlib for plotting.
pitcon routines available
nleq2 routines available
You are using NumPy with SciPy version: 0.6.0

PySCeS environment
******************
pysces.model_dir = /home/tim/mypysces/pscmodels
pysces.output_dir = /home/tim/mypysces/pscout

*****************************************************************************
* Welcome to PySCeS (0.6.8) - Python Simulator for Cellular Systems   *
*                http://pysces.sourceforge.net                        *
*                  A Brave New World edition                          *
* Copyright(C) B.G. Olivier, J.M. Rohwer, J.-H.S. Hofmeyr, 2004-2008  *
* Triple-J Group for Molecular Cell Physiology                        *
* Stellenbosch University, South Africa                               *
* PySCeS is distributed under the GNU General Public Licence          *
* See README.txt for licence details                                  *
*****************************************************************************

*****************************************************************************
*  Welcome to SymCA (0.2.8.1) - Symbolic Control Analysis PySCeS Module *
*    Copyright(C) T.J. Akhurst, J.M. Rohwer, J.-H.S. Hofmeyr, 2008      *
*          Triple-J Group for Molecular Cell Physiology                 *
*              Stellenbosch University, South Africa                   *
*      SymCA is distributed under the GNU General Public Licence        *
*                See README.txt for licence details                    *
*****************************************************************************
```

SymCA and PySCeS, which is loaded by SymCA since it is used to read the input file and perform the initial stoichiometric analysis of the model, are now loaded into the Python/IPython session and are ready for use. Then all required data is extracted and incorporated into SymCA.

4.3 Instantiating a model object

To perform symbolic control analysis a symca object must be instantiated. This is achieved by typing the following on the command-line, where sym is the SymCA instance and the model to be analysed is 'conceptA.psc':

```
In [2]: sym = symca.symca('conceptA')
Using model directory: /home/tim/mypysces/pscmodels
/home/tim/mypysces/pscmodels/conceptA.psc loading .....
Parsing file: /home/tim/mypysces/pscmodels/conceptA.psc

Calculating L matrix . . . . . .  done.
Calculating K matrix . . . . . .  done.

(hybrd) The solution converged.
```

On instantiation, the user can provide two additional arguments as well as the required model name:

- psc_file - PySCeS input file for model to be analysed.
- model_dir - The first optional argument is provided in the event of the PySCeS input file being located in an alternative directory to the default PySCeS model directory. A string representation of the designated location is thus provided.
- format - The second optional argument governs any LaTeX output that may be generated after analysis. The default value is False, but when designated True, any common substrings found in the reaction labels provided in the PySCeS input file (e.g. 'v', 'r', 'R' or 'rxn') are removed from the final LaTeX output.

The model has been successfully loaded into PySCeS, the structural analysis performed and all symbolic data generated.

4.3.1 Symbolic control analysis

Now that all symbolic data has been generated, the next step is to perform symbolic control analysis (SCA), which is achieved by calling the doSca() command.

```
In [3]: sym.doSca()

=========================================================================
=   Beginning Symbolic Control Analysis for conceptA.                   =
=========================================================================

Maxima matrix inversion complete....
Writing maxima data to file......

=========================================================================
=   SCA data successfully computed for conceptA.                        =
=========================================================================
```

This method has four optional arguments:

- `dep` - This argument determines whether or not dependent control coefficients will be computed because the control-matrix equation computes only the independent control coefficients. There are two relationships which enable the calculation of the dependent control coefficients, Equations 1.24 & 1.25. The argument is set by default to True, implying that dependent control coefficients will be computed. If the argument is set to False, i.e. `dep=False`, only independent control coefficients will be computed.
- `load` - This argument is set by default to False, and can only be used after an initial analysis has been performed on the model. After a model has been analysed this argument can be set to True, which will load data from a previous analysis as opposed to performing the analysis from start.
- `subE` - An option is available to perform elasticity 'knock-out' experiments whereby elasticity coefficients can be assigned pre-determined numerical values, such as 0 or 1. These values are then substituted into the **E** matrix before the inversion and subsequent computation of control coefficients. By setting `subE` to True, the user indicates that this substitution will take place, as the argument is set by default to False.
- `subDict` - This argument is required only when `subE` has been set to True, and is a dictionary of all desired elasticity coefficient substitutions. The dictionary contains key value pairs which include the elasticity coefficient labels in the form `ecR1_s1`, where `R1` and `s1` refer to the reaction and the specie respectively, and the desired value.

4.4 Post-symbolic control analysis features

The user has a number of options ranging from setting the elasticity coefficients after generation of control coefficients in a similar way to that achieved before an analysis, to output of data in LaTeX. We have outlined all these features in the following sections:

4.4.1 Symbolic to PySCeS comparison

The values of symbolic control coefficient expressions can be calculated via substitution of the PySCeS steady-state values for species, fluxes and elasticities into the SymCA expressions. These values are then compared with those obtained numerically using PySCeS. This method can be used as a test for the integrity of the control coefficient expressions as follows:

```
In [4]: sym.symca2Pysces()
(hybrd) The solution converged.
INFO: Steady State evaluation complete.
```

4.4.2 Summation theorems

The user can compute the summation theorems for all control coefficients generated by SymCA. Symbolic expressions are substituted with the steady-state values obtained from PySCeS and the resultant numerical values are used to compute the summations for both flux and concentration control coefficients. This method is accessed as follows:

```
In [5]: sym.computeSummations()
```

4.4.3 Elasticity substitution

This method enables elasticity coefficient substitutions after the initial analysis, which allows the user to assess the effects of the substitutions on the resulting control coefficient expressions. This method has two arguments:

```
In [6]: sym.setElasticityPost(subDict)
```

The only required argument is a dictionary, subDict. The available arguments are described below:

- subDict - This dictionary has all desired elasticity coefficient substitutions. The dictionary is the same as that given as an argument when calling `doSca()`,
- elas_type - This is an optional argument and can be 0 or 1. When 0 is supplied, all elasticity substitutions are performed simultaneously, whereas if the user sets type to 1 the dictionary is stepped through and each elasticity is substituted in isolation with a final substitution of all elasticities.

4.4.4 Co-control coefficients

This method computes all co-control coefficients for the data provided by the user, where a co-control coefficient represents the ratio between control coefficients describing the response of two system variables to a common perturbation [34] (see Section 1.6 for further details). A list of tuples/lists is required as the argument for this method, where each tuple/list (element in the list) represents the data for a single co-control coefficient in the form: (flux,species/metabolite,reaction/enzyme). This method is used as follows:

```
In [7]: coc_data = [('JR2','S2','R1'),('JR4','S3','R3')]
In [8]: sym.computeCoControl(coc_data)
```

where coc_data is the list of tuples, hence represents data for two co-control coefficients.

4.4.5 Response coefficients

There are two methods for computing response coefficients with respect to external parameters, which must be used together. The first method, `computeResponse()`, computes the numerical values of all response coefficients using `PySCeS`. The second method, `getResponse()`, enables the user to select various response coefficients based on specific criteria. After response coefficients have been computed and selected, their corresponding control and elasticity coefficients are identified so that the user can perform symbolic operations with them. The data from both methods is stored in the symcaModelOut dictionary with Response as the key and a dictionary as its values. The `PySCeS` data is stored under the pysces key and the `SymCA` under symca. The following arguments are available for use with the `getResponse()` method:

- cutoff - The default is 0.5, which implies that all response coefficients that are numerically equal to or greater than 0.5 will be extracted.
- flux - This lists all flux or species, i.e. all response coefficients for specified fluxes or species.
- param - This lists the parameters in response coefficients, i.e. all response coefficients for specific parameters.

4.4.6 Control pattern quantification

The value and percentage contribution of all control patterns making up each control coefficient expression can be computed. The control pattern data for each control coefficient is pickled and the pickle reference is stored in the ControlCoefficient control_pattern_ ref attribute. The method is called as follows:

```
In [9]: sym.controlPatterns()
Quantifying control coefficient control patterns....
```

There are three optional arguments which can be set when calling this method:

- ccList - This is a list of all fluxes or species whose control coefficients you wish to analyse.
- tol - The default is 0.1, which is the cutoff value to be used to select coefficients with an absolute value greater than the cutoff.
- type - The default is None, otherwise set to 'coc' to quantify co-control coefficient control patterns.

4.4.7 Post-analysis simplification

This method gives the user the option to perform a further simplification of the generated control coefficients. This is achieved by removing all terms common to both the numerator for each control coefficient and the common denominator.

```
In [10]: sym.simplify()
```

4.4.8 Pattern scan

This allows the user to investigate the effects of perturbing a parameter on the control pattern contributions for each control coefficient. A parameter value is perturbed and the control patterns are quantified for the new parameter value. This gives an insight into how these changes affect the routes of regulation, which are described by the control patterns.

```
In [11]: sym.patternScan()
```

This method has four optional arguments:

- `cc` - The user can specify which control coefficients are to be scanned by providing a list of desired control coefficient labels. If no list is provided all control coefficients are used.
- `param` - Specific parameters can be scanned by providing a list of parameters, otherwise all system parameters are used for the scan.
- `range` - This argument determines the range to be scanned, and is set to 0.1 (10%) if no range is provided. The range scanned represents a relative change around the steady-state value for the parameter in question. For example, if the parameter value is 0.5 and the default range applies, the range to be scanned is between 0.45 and 0.55.
- `steps` - The number of steps within the determined range can be set by assigning a value for this argument. If no value is supplied, a default of 11 steps is used.

4.4.9 Graphical representation of elasticity coefficient distribution

The user has the option to visualise the elasticity coefficient value distribution for a desired network. The user can view either the variable elasticities in isolation, or both variable and parameter elasticities.

```
In [12]: sym.viewSSElas()
```

The user can call this elasticities viewing method with the following optional arguments:

- `par` - This is by default False, which displays only variable elasticities. However, by setting this to True the user can view both variable and parameter elasticities.

- `colours` - By default the visualise class has a set of ten colours in the colour range. The user can provide their own list of colours with this method.
- `colour_codes` - If the user supplies their own colours for the previous argument, they must then supply a dictionary of colours and corresponding hexadecimal colour codes as the key value pairs.
- `range_max` - The user can set maximum value when creating the colour range, otherwise the value is set to 10 by default.

4.4.10 Graphical representation of control patterns

After the user has performed the control pattern quantification, they can visualise the quantified control patterns on a network.

```
In [13]: sym.viewControlPatterns()
In [14]:
```

This method has seven optional arguments:

- `below` - This determines whether all control patterns will be visualised, based on the pattern cutoff value. The default is False implying that only those patterns above the pattern cutoff value will be displayed. When the default is set to True, all patterns are displayed.
- `cc` - This allows the user to provide a list of control coefficients which they wish to visualise, otherwise the control pattern images for all control coefficients are generated.
- `flux_species` - The user can supply a list of fluxes and species. All control coefficients for each entry are determined and the control pattern images for the desired control coefficients are generated.
- `colours` - The user can supply a custom set of 10 colours for the colour range.
- `colour_codes` - If the user supplies a custom set of colours, they must also provide a dictionary of the colours and their hexadecimal representations.
- `pattern_cutoff` - The value set for this argument is used to determine which control pattern images will be generated. If no value is given, the value is set to 0 and all control pattern images are generated.
- `range_max` - This value is used as the maximum for the colour range, and the intervals will be computed based on this value. If no values are given the maximum used is 10.

4.4.11 Data output

A method exists that allows the user to output data in LaTeX format. Each type of data supported has its own specific argument, and all data is written to the `latex` sub-directory which is found in the following path:

`$PSCOUT/symca/`*`model_name`*`/latex/`

where $PSCOUT is the `PySCeS` output directory. See Pages 43 & 43 for more details regarding the directory structure associated with `SymCA`.

```
In [14]: sym.writeData(argument)
```

where

`argument`

can be one of the following:

- `'cc'` - All computed control coefficients; three files are written, one containing the common denominator, one for all flux control coefficients and another for all concentration control coefficients. Additional information found in the output includes whether or not elasticity coefficient substitutions have been performed, in which case the details of the substitutions are included.
- `'sym2psc'` - This table summarises the numerical values for all control coefficients including `PySCeS` and `SymCA` computed values.
- `'patterns'` - The data outputted for control patterns is a table summarising the quantified control pattern data for all control coefficients.
- `'coc'` - This data is outputted in a similar form to that of the native control coefficient, with the major difference that only two files are written, one for flux co-control coefficients and the other for concentration co-control coefficients.
- `'summations'` - This table represents the summation theorem results for the model in question.
- `'scan'` - Two plots are generated, one showing the control pattern quantification % contribution and the other showing the control coefficient value over the parameter range scanned. Since the control pattern expressions can be large, a key in the form of a table describes the annotations used for the former plot. The generation of these plots uses the `PySCeS plt.plot2D` method, which in turn requires the installation of Matplotlib. The .eps format is used for the plots, which are incorporated into a LaTeX output file.

A header is written for all files and includes the time and date when the file was written, the name of the file and the model for which the data applies.

4.5 LaTeX output as generated by SymCA

As previously discussed, numerous data generated by `SymCA` can be written to file as LaTeX. In this section we will illustrate an example for the control coefficient and control pattern output.

4.5.1 Control coefficient output

Data generated for the control coefficients is spread over three files, one each for the flux control coefficients, the concentration control coefficients and the common denominator for all control coefficients. In each file, the information is displayed if any elasticity substitutions have been performed.

There is a separate file for the denominator before ($\sum$) and after ($\sum^{*}$) elasticity substitutions, because the denominator is common to all control coefficients in both situations. If the simplification step is performed, the denominator ($\sum^{**}$) is included with the control coefficient as the denominator which is now unique to the control coefficient, as shown in the example:

```
Output Generated using SymCA On: Sun, 20 Sep 2009 19:40:00
Output File: conceptA_CommonDenom.tex
Pysces Model: conceptA
```

Common denominator

$$\sum = -\frac{J_{\mathrm{R6}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} + \frac{J_{\mathrm{R6}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} - \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} + \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{S5}$$
$$- \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{S5} + \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} - \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} - \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{S5}$$
$$+ \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{S5} + \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{S5} - \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{S5} + \frac{J_{\mathrm{R6}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4}$$
$$- \frac{J_{\mathrm{R6}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} + \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} + \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4}$$
$$- \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} + \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} + \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4} - \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R6}}{S4}$$
$$- \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4} + \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4}$$

After elasticity substitution: ε_{S1}^{R1} , ε_{S4}^{R6} , $\varepsilon_{S2}^{R2} = 0$

$$\overset{*}{\sum} = -\frac{J_{\mathrm{R6}}\ \varepsilon_{S1}^{R3}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} - \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} + \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6}}{S5}$$
$$- \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{S5} + \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4}$$

The control coefficient output below illustrates the additional data included when elasticity substitutions are performed after initial generation of the control coefficients. The simplification step shown can be called at any stage to factorise the numerator and denominator terms to ensure that all common terms are removed.

Control coefficient: C_{R3}^{JR6}

Original coefficient

$$C_{R3}^{JR6} = (-\frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} + \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} - \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} + \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} - \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} + \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4})/\sum$$

Cannot be simplified further

After elasticity substitution: ε_{S1}^{R1} , ε_{S4}^{R6} , $\varepsilon_{S2}^{R2} = 0$

$$C_{R3}^{JR6} = (-\frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5})/\sum^{*}$$

Simplified coefficient

$$C_{R3}^{JR6} = (-J_{\mathrm{R4}}\ S4\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6})/\sum^{**}$$

Where

$$\sum^{**} = J_{\mathrm{R4}}\ S5\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5} - J_{\mathrm{R6}}\ S4\ \varepsilon_{S1}^{R3}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} - J_{\mathrm{R4}}\ S4\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} + J_{\mathrm{R4}}\ S4\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R6} - J_{\mathrm{R4}}\ S4\ \varepsilon_{S1}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}$$

4.5.2 Co-control coefficient output

The data below corresponds with co-control coefficients associated with the model in Figure 4.1. As described in Section 1.6, a co-control coefficient represents the ratio between two control coefficients describing the response of two system variables to a common perturbation. In the first example below the two control coefficients are C_{R1}^{JR2} and C_{R1}^{S1} with a perturbation in the rate of reaction R1. In the second example the control coefficients are C_{R3}^{JR5} and C_{R3}^{S3}, and the perturbed reaction rate is R3. The output generated includes the expression for the co-control coefficient as well as its numerical value.

```
Output Generated using SymCA On: Sun, 20 Sep 2009 19:40:03
Output File: conceptA_CoControlCoefficients.tex
Pysces Model: conceptA
```

Co-control coefficient: $O_{R1}^{JR2:S1}$

Elasticity substitutions: None

$$O_{R1}^{JR2:S1} = \frac{1}{\varepsilon_{S1}^{R2}} - \frac{\varepsilon_{S2}^{R2}}{\varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}}$$
$$value = 9.8190e-01$$

Co-control coefficient: $O_{R3}^{JR5:S3}$

Elasticity substitutions: None

$$O_{R3}^{JR5:S3} = \frac{S5\ \varepsilon_{S4}^{R6}}{S5\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}} - S4\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} - \frac{S5\ \varepsilon_{S4}^{R5}}{S5\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}} - S4\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} - \frac{S4\ \varepsilon_{S5}^{R6}}{S5\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}$$
$$- S4\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6} + \frac{S4\ \varepsilon_{S5}^{R5}}{S5\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}} - S4\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}$$
$$value = 1.8056e+00$$

4.5.3 SymCA vs PySCeS output

The data used to compare the control coefficients as determined by substitution of the steady-state values into the `SymCA` expressions as well as the data obtained via numerical computations by `PySCeS` are written to file in the following format:

```
Output Generated using SymCA On: Sun, 20 Sep 2009 19:40:03
Output File: conceptA_symca2pysces.tex
Pysces Model: conceptA
```

SCA v PySCeS

Control coefficient	PySCeS	SymCA
C_{R1}^{JR1}	0.941932256355	0.941932256355
C_{R2}^{JR1}	0.0275502976562	0.0275502976562
C_{R3}^{JR1}	0.0197955907959	0.0197955907959
C_{R4}^{JR1}	0.0027550297656	0.0027550297656
C_{R5}^{JR1}	0.00397095772914	0.00397095772914
C_{R6}^{JR1}	0.00399586769779	0.00399586769779
C_{R1}^{JR2}	1.01148218028	1.01148218028
C_{R2}^{JR2}	0.429192201278	0.429192201278
C_{R3}^{JR2}	-0.344819448484	-0.344819448484
C_{R4}^{JR2}	0.0429192201276	0.0429192201276
C_{R5}^{JR2}	-0.0691701231971	-0.0691701231971
C_{R6}^{JR2}	-0.069604030007	-0.069604030007
C_{R1}^{JR3}	0.866011875368	0.866011875368
C_{R2}^{JR3}	-0.410880179643	-0.410880179643
C_{R3}^{JR3}	0.417807711196	0.417807711196
C_{R4}^{JR3}	-0.0410880179641	-0.0410880179641
C_{R5}^{JR3}	0.0838114293819	0.0838114293819
C_{R6}^{JR3}	0.0843371816615	0.0843371816615
C_{R1}^{JR4}	1.01148218028	1.01148218028
C_{R2}^{JR4}	0.429192201278	0.429192201278
C_{R3}^{JR4}	-0.344819448484	-0.344819448484
C_{R4}^{JR4}	0.0429192201276	0.0429192201276
C_{R5}^{JR4}	-0.0691701231971	-0.0691701231971
C_{R6}^{JR4}	-0.069604030007	-0.069604030007
C_{R1}^{JR5}	0.866011875368	0.866011875368
C_{R2}^{JR5}	-0.410880179643	-0.410880179643
C_{R3}^{JR5}	0.417807711196	0.417807711196
C_{R4}^{JR5}	-0.0410880179641	-0.0410880179641
C_{R5}^{JR5}	0.0838114293819	0.0838114293819
C_{R6}^{JR5}	0.0843371816615	0.0843371816615

continued on next page

continued from previous page

Control coefficient	PySCeS	SymCA
C^{JR6}_{R1}	0.866011875368	0.866011875368
C^{JR6}_{R2}	-0.410880179643	-0.410880179643
C^{JR6}_{R3}	0.417807711196	0.417807711196
C^{JR6}_{R4}	-0.0410880179641	-0.0410880179641
C^{JR6}_{R5}	0.0838114293819	0.0838114293819
C^{JR6}_{R6}	0.0843371816615	0.0843371816615
C^{S1}_{R1}	0.993170655818	0.993170655818
C^{S1}_{R2}	-0.471210787156	-0.471210787156
C^{S1}_{R3}	-0.33857695614	-0.33857695614
C^{S1}_{R4}	-0.0471210787153	-0.0471210787153
C^{S1}_{R5}	-0.0679178911481	-0.0679178911481
C^{S1}_{R6}	-0.0683439426588	-0.0683439426588
C^{S2}_{R1}	0.840931345899	0.840931345899
C^{S2}_{R2}	0.356824057315	0.356824057315
C^{S2}_{R3}	-0.286677796761	-0.286677796761
C^{S2}_{R4}	-0.795702825064	-0.795702825064
C^{S2}_{R5}	-0.0575070188384	-0.0575070188384
C^{S2}_{R6}	-0.0578677625516	-0.0578677625516
C^{S3}_{R1}	1.5636609581	1.5636609581
C^{S3}_{R2}	-0.741880467969	-0.741880467969
C^{S3}_{R3}	0.754388738274	0.754388738274
C^{S3}_{R4}	-0.0741880467965	-0.0741880467965
C^{S3}_{R5}	-0.748642459513	-0.748642459513
C^{S3}_{R6}	-0.753338722095	-0.753338722095
C^{S4}_{R1}	0.709038841096	0.709038841096
C^{S4}_{R2}	-0.3364041703	-0.3364041703
C^{S4}_{R3}	0.342076019709	0.342076019709
C^{S4}_{R4}	-0.0336404170298	-0.0336404170298
C^{S4}_{R5}	0.0686197966214	0.0686197966214
C^{S4}_{R6}	-0.749690070097	-0.749690070097
C^{S5}_{R1}	-0.713421772307	-0.713421772307
C^{S5}_{R2}	0.338483656292	0.338483656292
C^{S5}_{R3}	-0.344190566299	-0.344190566299
C^{S5}_{R4}	0.033848365629	0.033848365629
C^{S5}_{R5}	-0.0690439706312	-0.0690439706312
C^{S5}_{R6}	0.754324287316	0.754324287316

4.5.4 Summation output

The summation theorem data is summarised in two separate tables, one for all fluxes and the other for all species found in the system. The actual and the theoretical values obtained are shown below.

```
Output Generated using SymCA On: Sun, 20 Sep 2009 19:40:03
Output File: conceptA_summations.tex
Pysces Model: conceptA
```

Summation Theorem data for all flux

Flux	Theoretical	SymCA
JR1	1	1.0
JR2	1	1.0
JR3	1	1.0
JR4	1	1.0
JR5	1	1.0
JR6	1	1.0

Summation Theorem data for all species

Species	Theoretical	SymCA
S1	0	0.0
S2	0	0.0
S3	0	-0.0
S4	0	-0.0
S5	0	-0.0

4.5.5 Quantified control pattern output

The data obtained when one performs the control pattern quantification includes the percentage contribution of a control pattern for a particular control coefficient, as well as the value of the control pattern. The following data was obtained by performing the control pattern quantification for control coefficient C_{R3}^{JR6}:

$$(-\frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} + \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} - \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5} + \frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4}$$
$$-\frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4} + \frac{J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4})/\sum$$

which is found in the model depicted in Figure 4.1.

Control pattern data for: C_{R3}^{JR6} **CC value:** $4.1781e-01$

Control pattern	Value	% Contribution
$\frac{+J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5}$	3.4218e-03	0.82
$\frac{+J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4}$	3.4218e-03	0.82
$\frac{-J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5}$	3.4218e-02	8.19
$\frac{+J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4}$	3.4218e-01	81.90
$\frac{-J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5}$	3.4218e-04	0.08
$\frac{-J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4}$	3.4218e-02	8.19

Elasticity Substituted Control Coefficient: C_{R3}^{JR6} **CC Value:** $3.9961e-01$

Elasticity Substitutions: $\varepsilon_{S1}^{R1} = 0$

Control Pattern	Value	% Contribution
$\frac{J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S4}^{R6}}{S4}$	3.6328e-01	90.91
$\frac{-J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R5}\ \varepsilon_{S5}^{R6}}{S5}$	3.6328e-02	9.09

The above example is purely for demonstrating the functionality of performing elasticity substitutions, and it is important to note the selection of the elasticities being substituted is not determined by `SymCA`. The user determines which, if any, elasticities will be substituted. With regards to the example, ε_{S1}^{R1} was selected on the basis that it was the smallest elasticity in the system with a value of -0.0585. Presently, `SymCA` does not cater for using a cut-off value and performing these substitutions at analysis time. However, this may be something to consider. If a system were to be implemented whereby a cut-off value would be used and the substitutions performed at analysis time, the user would need to provide the cut-off value, as ignoring one or more elasticities can have a profound effect on the resultant expressions, thus the value selected is subjective.

4.5.6 Control pattern parameter scan output

The data generated by a control pattern parameter scan is saved to the `$PSCOUT/symca/`*`model_name`*`/latex/` subdirectory in two files, one for the flux control coefficients and the other for the concentration control coefficients. The data is represented by a plot (Figure 4.2) illustrating the changes in the control pattern contributions over a range of parameter values. A second plot (Figure 4.3) corresponds to the change in the control coefficient values over the range of parameter values. Table 4.1 describes the annotations used as labels for the control patterns. The output is generated only for those control coefficients with more than one control pattern.

Table 4.1: Control patterns index

Control pattern ID	Control pattern
C.P.1	$\frac{-J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{S5}$
C.P.2	$\frac{+J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{S5}$
C.P.3	$\frac{-J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S5}^{R5}}{S5}$
C.P.4	$\frac{+J_{\mathrm{R4}}\ \varepsilon_{S1}^{R2}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4}$
C.P.5	$\frac{-J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R4}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4}$
C.P.6	$\frac{+J_{\mathrm{R1}}\ \varepsilon_{S1}^{R1}\ \varepsilon_{S2}^{R2}\ \varepsilon_{S3}^{R3}\ \varepsilon_{S4}^{R5}}{S4}$

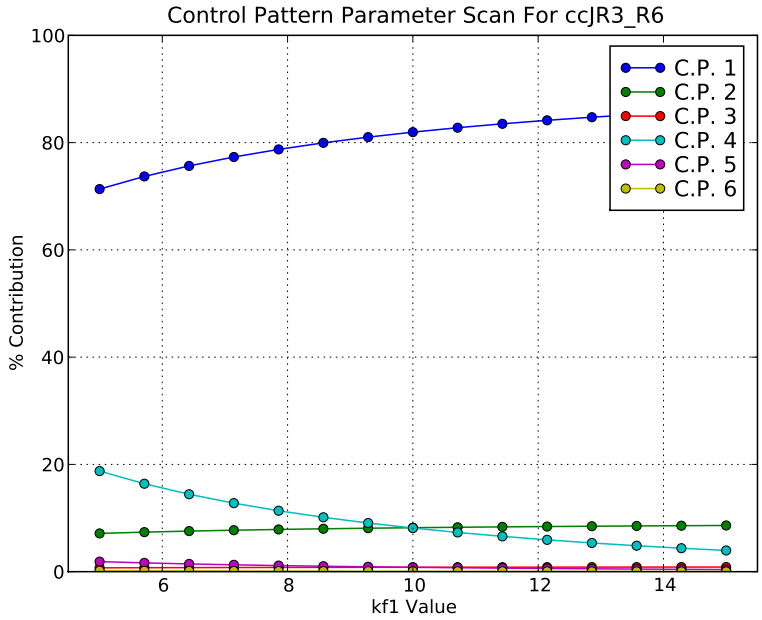

Figure 4.2: Control pattern parameter scan output.

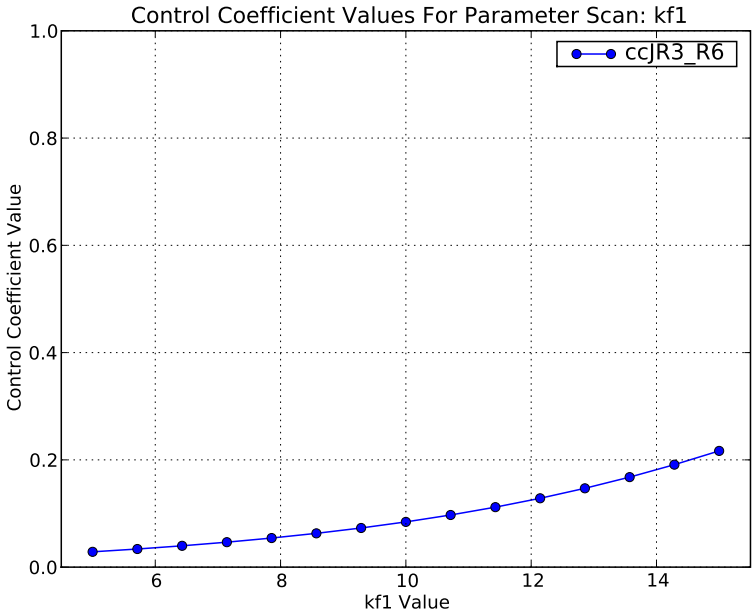

Figure 4.3: Control coefficient parameter scan output.

4.5.7 Graphical representations

As described in Section 3.4.12, the user can generate graphics for the biochemical network under investigation. These graphics contain information which complements numerous methods and additional features of `SymCA`, primarily focussing on illustrating the control patterns within the system.

This section includes examples of the following images, base (Figure 4.4), which is used for all subsequent graphics, two depicting the elasticity distributions (Figures 4.5 & 4.6) and a representation of a control pattern (Figure 4.7). We will use the model from Figure 4.1 for continuity.

In the control pattern figure (Figure 4.7), the modulated enzyme is identified by the square reaction node, whereas all others remain as the standard circle. The colour associated with the pattern gives an indication of the percentage contribution of the control pattern being visualised. In the case of a concentration control coefficient control pattern, the affected species is identified by underlining it. All fluxes found in a control pattern are also identified by the colour of the reaction number corresponding to the colour of the pattern.

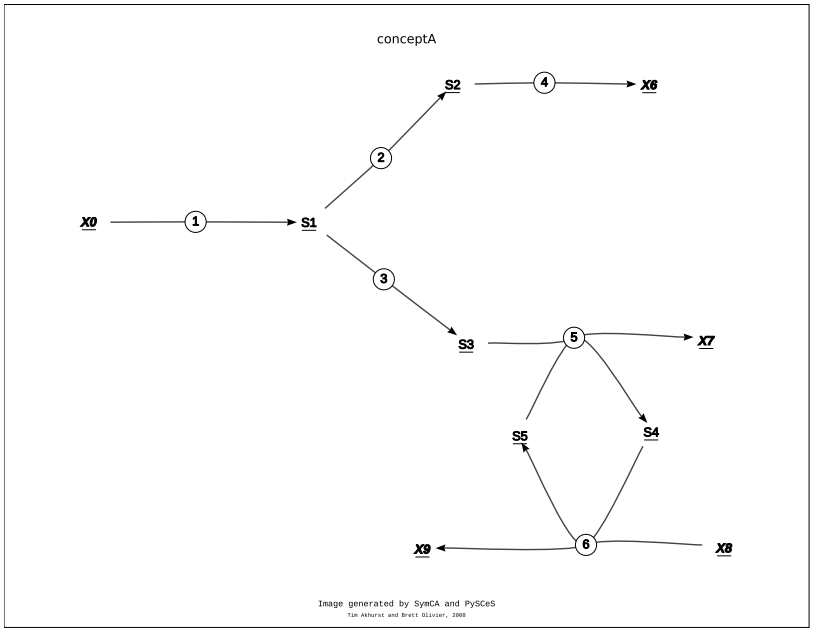

Figure 4.4: The base model generated by `SymCA`

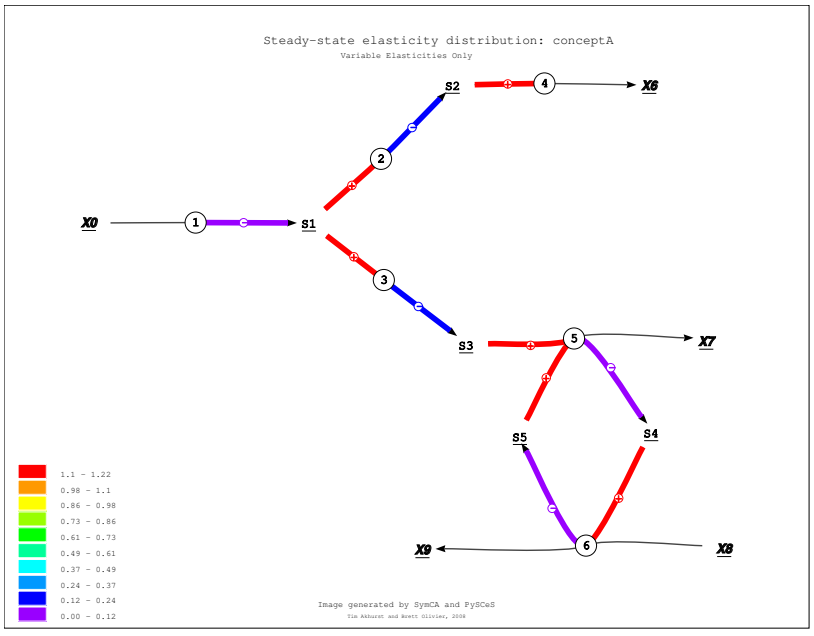

Figure 4.5: Variable elasticity distribution image

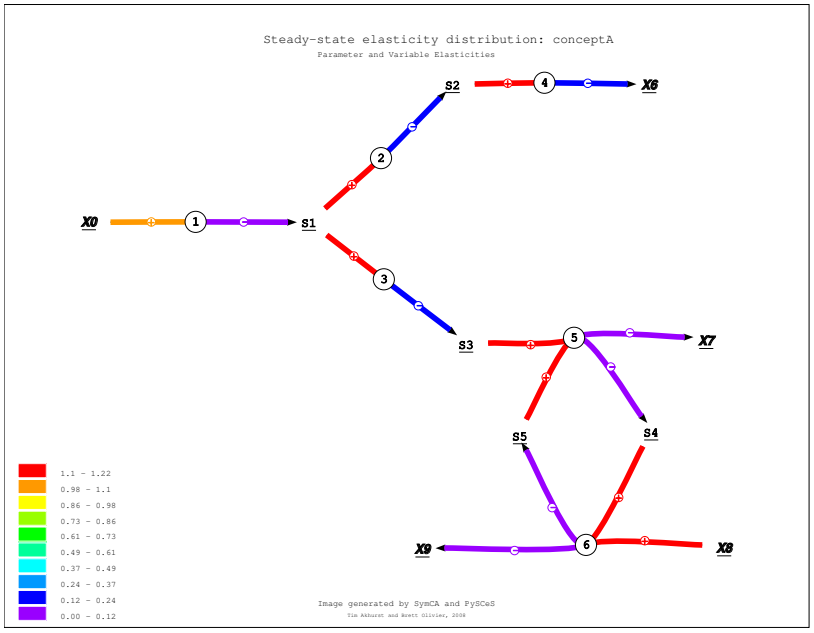

Figure 4.6: Variable and parameter distribution image

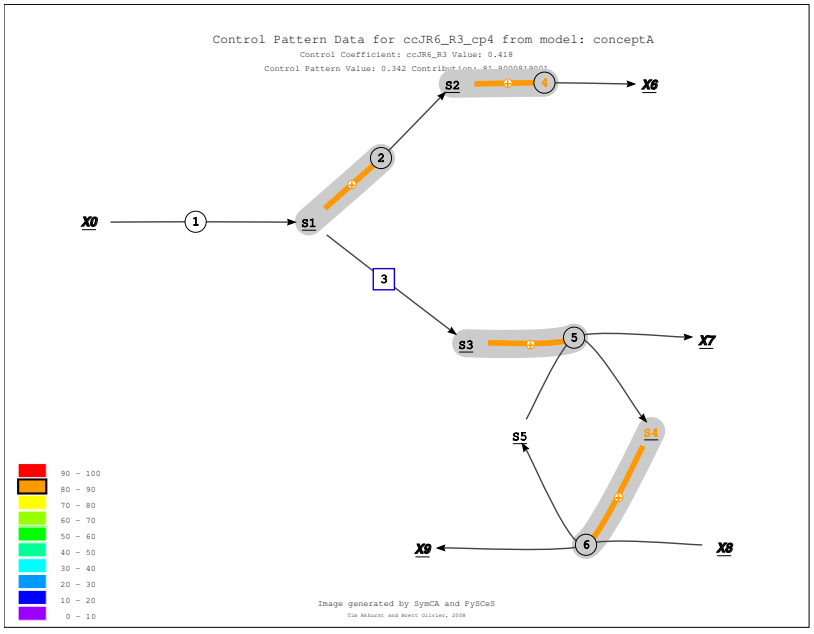

Figure 4.7: A control pattern image for a pattern found in $C_{R3}^{J_{R6}}$

We have now demonstrated the operability and functionality of SymCA by command-line interaction. We have also demonstrated all variations of output. Below is the script which was used to generate all output:

```
import symca
sym = symca.symca('conceptA')
sym.doSca()
sym.setElasticityPost(subDict={'ecR1_S1':0,'ecR2_S2':0,'ecR6_S4':0})
sym.simplify()
sym.symca2Pysces()
sym.computeSummations()
sym.controlPatterns()
sym.computeCoControl([('S1','JR2','R1'),('S3','JR5','R3')])
sym.patternScan(cc=['ccJR3_R6'],param=['kf1'],range=0.5,steps=15)
sym.viewElas(image_type='eps')
sym.viewElas(par=True,image_type='eps')
sym.viewControlPatterns(cc=['ccJR6_R3'],image_type='eps')
sym.writeData('cc')
sym.writeData('sym2psc')
sym.writeData('summation')
sym.writeData('patterns')
sym.writeData('coc')
sym.writeData('scan')
```

In the script a model named 'conceptA.psc' is loaded into SymCA, and then symbolic control analysis is performed and additional methods are called. The LaTeX for all control coefficients as well as the data generated by the various other methods is then written to file.

4.6 Discussion

This chapter described in detail all the methods associated with SymCA. The explanations include the options for each method as well as excerpts of code demonstrating the commands the user would use to carry out the various methods. We gave examples of the types of output generated and discussed a number of the methods that were designed for later use in investigating the regulatory behaviour of cellular systems. We placed particular emphasis on control-patterns, which were introduced by Hofmeyr [49], as these highlight the 'chain of local effects' that exist within cellular systems as a result of enzyme perturbations.

We presented the generation and subsequent manipulations of the control coefficient expressions so that each expression consists of a varying numbers of control patterns. We implemented several additional methods to maximise this feature. Of particular mention is the technique of quantifying the contribution of the control patterns found

within each control coefficient expression, whose results show exactly how a perturbation is manifested within the system. Adding to the knowledge gained via this technique is the ability to combine this approach with that of a parameter scan, where a parameter value is varied over a range and the control patterns are quantified for each steady-state arising from the change in parameter values. This enables us to understand how the regulatory pathways may vary under different conditions.

To fully appreciate these quantified control patterns, we included the ability to generate images of the underlying system within `SymCA`. Since the process of developing the software to generate these images is extremely complex, we used a tool included in the Systems Biology Workbench, (`SBW`) [97] to perform this function. After we had generated the structure of the system via `SBW`, the image was customised and data generated from `SymCA` mapped onto it. This allowed us to visualise exactly how the effects of a perturbation are propagated through the system.

In Section 4.5.5 we gave an example of the data generated by performing quantified control pattern analysis for a control coefficient (C_{R6}^{JR3}) before and after elasticity substitutions had been performed. With respect to the elasticity substitution made, the specific elasticity was selected as it was the smallest in the system, thus resulting in a negligible change of the control coefficient value in question. The substitution was made as a means to illustrate the functionality of elasticity substitutions with `SymCA`. The substitution of numerical values for selected elasticities, needs specific attention as any substitution represents a simplification of the system, resulting in reduced complexity of the expressions. The implication is that one has to make assumptions about the elasticity values. Importantly, elasticity substitutions can be made even without kinetic data for a model, e.g. if it is known that a reaction is saturated with substrate or that it is an irreversible reaction the elasticity can be set to 0; similarly if a reaction is first-order in substrate the elasticity can be set to 1.

We are now in a position to apply `SymCA` to biological models, which leads to the second part of this thesis, the application of `SymCA` to investigate the regulatory behaviour of biological systems in greater detail. The work in the following chapter demonstrates what we can achieve with `SymCA` when using a combined strategy of supply-demand analysis and control pattern quantification.

Chapter 5

Applications of SymCA: Part I – Control pattern quantification & supply-demand analysis

5.1 General introduction

We briefly introduced the theory of supply-demand analysis and control pattern analysis in Section 1.6. The work which follows uses both types of analysis together with the control pattern quantification method developed as part of `SymCA`. Before focussing on the example system and the demonstration of this approach, we will re-introduce the techniques used in this study.

5.1.1 Supply-demand analysis

Our knowledge of the organisation of metabolic networks is central to understanding metabolic function. Intermediary metabolism consists of a catabolic block, a synthetic block making the building blocks for macromolecular synthesis, and a 'growth' block which makes and maintains the gene and enzyme machinery and the cellular structure. A common intermediate or pair of common intermediates provides the coupling mechanism for these blocks. Figure 5.1, on the following page, provides a graphical representation for the organisation of metabolism.

Metabolic pathways, which constitute the living process, are essentially a molecular economy in which the producing block is referred to as the *supply* and the consuming block, the *demand* [54]. The technique of supply-demand analysis provides a clear and quantitative description of metabolic regulation and function. The technique relies on both rate characteristics [50] and metabolic control analysis as its main tools [54]. In its simplest form, supply-demand analysis provides a quantitative description of the regulatory nature of a cellular system and also identifies parameter values which correlate to optimal homeostatic maintenance.

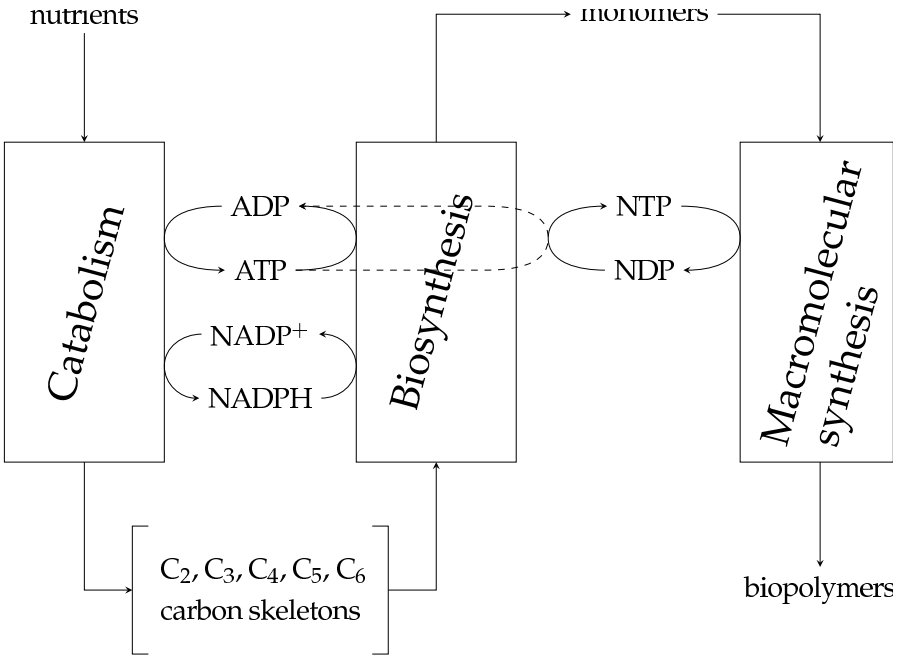

Figure 5.1: A simple representation of the functional organisation of metabolism.

In this chapter we examine the biological question about how the control distribution in a simple supply-demand system with a feedback loop changes when the demand load is varied. We use `SymCA` to generate quantified control patterns in order to understand the cause of this shift in control distribution and how the routes of regulation in the pathway change under different conditions.

5.1.2 Control pattern quantification

As introduced earlier, control pattern analysis is a non-algebraic technique developed by Hofmeyr [49], which enables the expression of control coefficients in terms of elasticity coefficients. The resultant control coefficient expression consists of numerous products of elasticity coefficients, each of which is a control pattern. Hofmeyr described these control patterns as a 'chain of local effects' in which each control pattern demonstrates how a small perturbation in an enzyme activity affects either a flux or metabolite pool. The control patterns give insight into the behavioural patterns inherent in metabolic pathways, and subsequently lead to a greater understanding of the relationships between the local and systemic properties of metabolic pathways.

`SymCA` has been implemented so that all control coefficients are generated in a form reminiscent of the control patterns that Hofmeyr [49] described. To quantify the individual control patterns which are present in the control coefficient expressions, we

developed a method within `SymCA` that isolates all control patterns within an expression and calculates the percentage contribution of the control pattern to the overall control coefficient and the value of the control pattern (see Section 4.4.6 for further details).

We developed a method within `SymCA` to evaluate the effects of varying a parameter value over a defined range. This method uses the parameter scanning functionality of `PySCeS` and computes the quantified control pattern data for each parameter value in the scan. We generated a plot of all control patterns in a control coefficient expression to illustrate the changing roles of the patterns over the parameter space. The section below investigates the effectiveness of this combined strategy.

5.2 Control pattern quantification and supply-demand analysis, a combined strategy

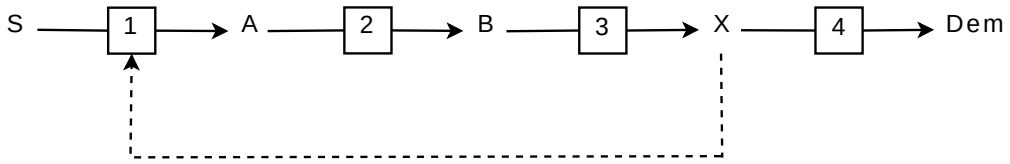

Figure 5.2: Linear pathway with a feedback loop where X serves as the linking metabolite between the supply block, which consists of three steps, and the demand block, which has one step.

The following work focusses on a simple theoretical model (Figure 5.2) in which there is a feedback mechanism between the metabolite linking both the *supply* and the *demand*, and the first reaction in the chain. Olivier used this model in 2005 [83]. Reaction 1 uses the reversible Hill equation which allows for the effects of cooperative enzymes and the effects of allosteric feedback inhibition. Reactions 2 and 3 use the reversible Michaelis-Menten equation and Reaction 4 uses the irreversible Michaelis-Menten equation. Table 5.1 summarises the kinetic parameters for each reaction.

The first step in this analysis is the generation of rate characteristic plots which illustrate the effects of different rates associated with the demand step. For this system the linking metabolite is labelled 'X'; for the purposes of supply-demand analysis, 'X' must be fixed. When 'X' is fixed it becomes a parameter. To assess the effects of varying degrees of demand on the steady-state, we set V_4 for the demand step (Reaction 4 in Figure 5.2) to values corresponding to the following levels of demand: low, medium-low, medium-high and high. For each level of demand we varied the value for 'X' over a range of 0.001 to 30 000, and calculate the steady-state fluxes of the supply and the demand blocks at every step in this range. The results are shown in the Figure 5.3.

Table 5.1: Kinetic parameters for all reactions for the model as shown in Figure 5.2.

Reaction	Parameter	Description	Value
1: S ↔ A	K_{eq1}	Equilibrium constant	400.0
	V_1	V_{max}	200.0
	$S_{0.5}$	Half-saturation constant	1.0
	$A_{0.5}$	Half-saturation constant	10000.0
	$X_{0.5}$	Half-saturation constant	1.0
	h	Hill coefficient	4.0
	α	Inhibition constant	0.01
2: A ↔ B	V_2	V_{max}	100.0
	K_{eq2}	Equilibrium constant	10.0
	K_{2A}	K_{m}	1.0
	K_{2B}	K_{m}	1.0
3: B ↔ X	V_3	V_{max}	100.0
	K_{eq3}	Equilibrium constant	10.0
	K_{3B}	K_{m}	1.0
	K_{3X}	K_{m}	1.0
4: X → Dem	V_4	V_{max}	100.0
	K_{4X}	K_{m}	0.1

We deduced from this plot that the area between V_4 values of 2.5 (red) and 75 (turquoise), corresponds to the area when the concentration of 'X' can be maintained at a relatively constant level, which implies that V_4 values within this range coincide with the area of optimal homeostatic control with regard to metabolite 'X'. When V_4 is between 1.5 (green) and 2.5 (red) the range for 'X' is near equilibirum, which indicates a lack of kinetic regulation with a corresponding increase in role of the thermodynamic term, which now dominates the supply elasticity[1]. When the Vmax rises above 75 (turquoise), the supply block is unable to meet the varying demand for 'X' and we can also see large variations in the concentration of 'X'. For values below 1.5 (green) there is another area of homeostasis regarding the concentration of 'X', and this represents homeostatic control at near equilibrium conditions.

This is undesirable as a supply pathway must fulfill two key functions: to cope with low demand conditions so that its products and intermediate metabolites do not tend towards their equilibrium concentrations, and to meet increasing demand for its product [54].

[1]Generally each elasticity coefficient represents the sum of a kinetic term determined by the enzymes binding properties and a thermodynamic term dependent only on Γ/K_{eq}. In a supply elasticity the thermodynamic term nears 0 far from equilibrium and $-\infty$ near equilibrium, where it totally dominates the kinetic term [54].

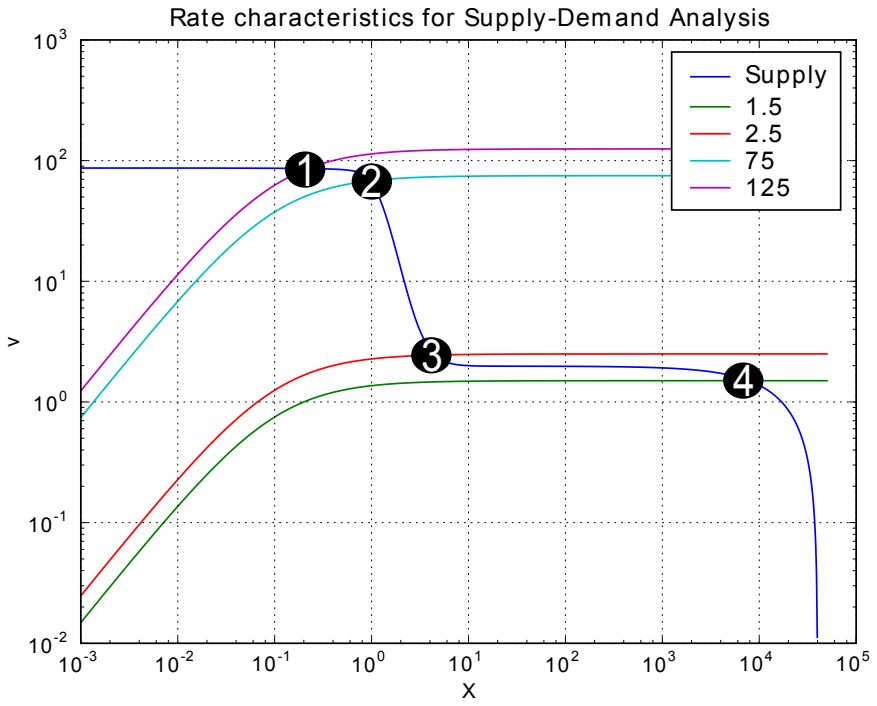

Figure 5.3: Supply-demand rate characteristic plot for the model shown in Figure 5.2. The values found in the figure legend correspond to the Vmax values for the demand step (Reaction 4).

For a system lacking a feedback mechanism, an area of homeostasis near equilibrium would be more likely than homeostasis far from equilibrium. The implication is that the feedback mechanism is responsible for homeostatic maintenance of 'X' at a concentration far from equilibrium. We can demonstrate this influence on the system using a combined control pattern analysis and parameter scan approach. For this investigation, the metabolite 'X' must be freed as opposed to performing the initial supply-demand analysis. The next stage in this investigation is to identify the control patterns involved over a range of Vmax values for Reaction 4.

We can use numerical MCA to identify a step with a substantial flux-control coefficient. At this stage we set the Vmax to 100, which represents the area midway between 1 and 2 to the left of Figure 5.3. In this case C_{R4}^{JR4} has a value of 0.302, and we continue using this as an example. We can now use `SymCA` to analyse this control coefficient in more depth. We calculated the following algebraic equation for C_{R4}^{JR4}:

$(-\varepsilon_A^{R1}\ \varepsilon_B^{R2}\ \varepsilon_X^{R3} - \varepsilon_A^{R2}\ \varepsilon_B^{R3}\ \varepsilon_X^{R1})/\sum$.

By performing a symbolic analysis of the system using `SymCA`, we can investigate the mechanisms involved with the control coefficient on the flux for a step with a substantial flux-control coefficient. The results of numerical MCA show that the greatest control for this step is found in C_{R4}^{JR4}, which has a value of ≈ 0.302. C_{R4}^{JR4} contains two control patterns, the first concerns the effects via the main chain, and the second the effects due to the feedback of 'X' on the first step. By quantifying these control patterns with respect to the control coefficient, we can determine the distribution of the control between the two control patterns (Table 5.2).

Table 5.2: Quantified control pattern data for flux control coefficient C_{R4}^{JR4}

ID	Control Pattern	Value	% Contribution
C.P.1	$-\varepsilon_A^{R1}\ \varepsilon_B^{R2}\ \varepsilon_X^{R3}$	7.8761e-02	26.10
C.P.2	$-\varepsilon_A^{R2}\ \varepsilon_B^{R3}\ \varepsilon_X^{R1}$	2.2297e-01	73.90

The data in Table 5.2 represents the quantified control pattern at steady-state where the Vmax for Reaction 4 was set to 100, which is in an area of relatively high demand. The images, Figures 5.4 and 5.5, represent the two control patterns found in C_{R4}^{JR4}.

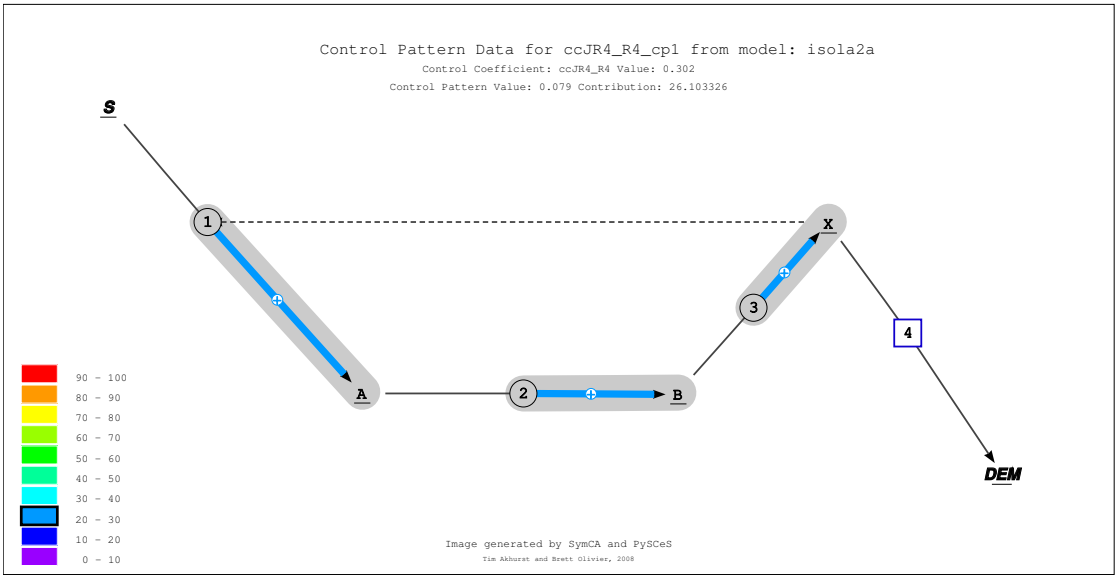

Figure 5.4: C. P. 1 found in C_{R4}^{JR4} at steady-state

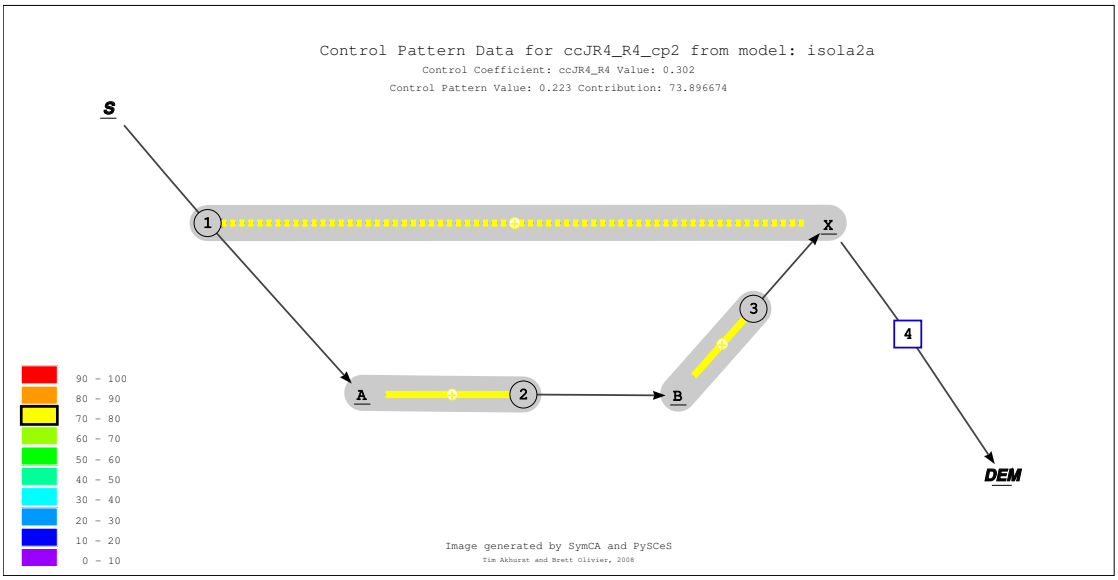

Figure 5.5: C. P. 2 found in C_{R4}^{JR4} at steady-state

By combining the technique of control pattern quantification with a parameter scan, we can gain an insight into the role and influence of each control pattern over a range of Vmax values for Reaction 4, which in this case will correspond to varying levels of demand. We set the Vmax for Reaction 4 to 100 in the `PySCeS` input file, performed symbolic control analysis and scanned the parameters over a range of values from 0.01 to 199; the control patterns were quantified at every step of the scan. We used the `patternScan()` method included in `SymCA` for this procedure (see Page 46 for more details regarding this method).

We can clearly see the effects of the parameter scan on the contributions of the two control patterns in Figure 5.6 where C.P. 1 represents the regulatory pathway along the main chain (Figure 5.4) and C.P. 2 represents the regulatory pathway involving the feedback mechanism (Figure 5.5). At very low demand, we can briefly see C.P. 1 as the dominant control pattern, which links with the area of the supply-demand figure where the homeostatic maintenance of 'X' occurs close to equilibrium. We can then see C.P 1 decline dramatically to 0% contribution which corresponds with the sharp increase in the contribution of C.P. 2 to 100% (Figure 5.7).

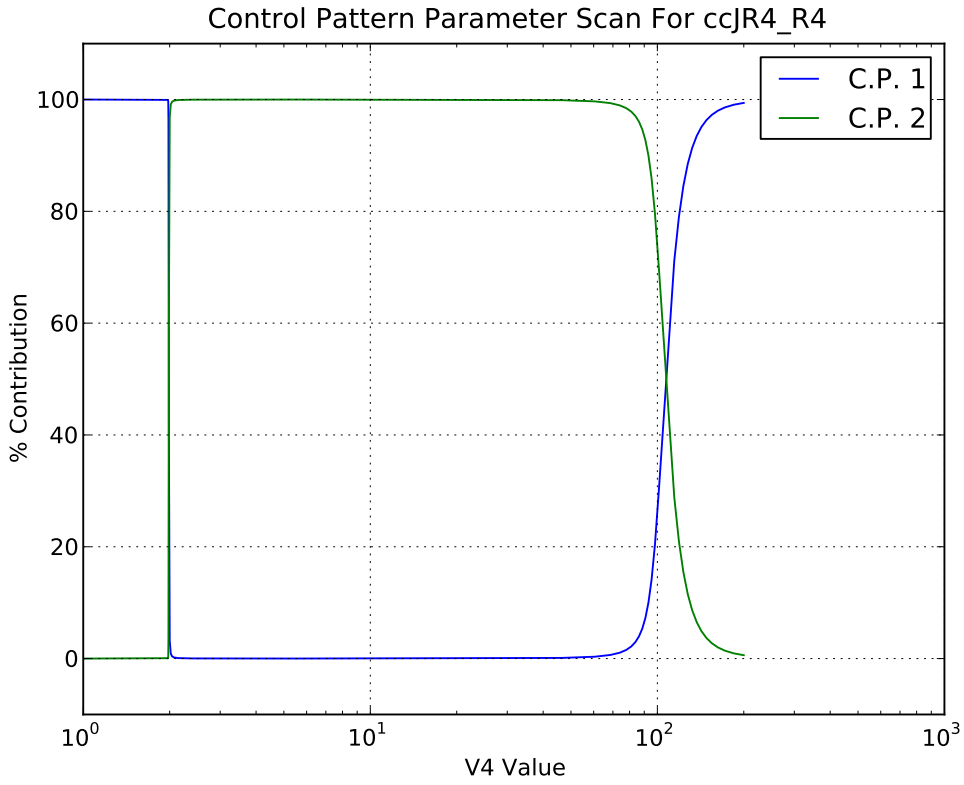

Figure 5.6: Control pattern contributions for parameter scan V4 on C_{R4}^{JR4}

The contribution of C.P. 2 is maintained at 100% between V4 values of ≈ 2 to ≈ 80, which corresponds to the region of optimal homeostasis of 'X'. This shows that the feedback mechanism is responsible for this homeostatic maintenance far from equilibrium. As the value of V4 increases beyond this point, the contribution of C.P. 2 decreases sharply with a subsequent increase in the contribution of C.P. 1. At a value of about 110, both control patterns share control over the demand flux. As the value continues to rise indicating an increase in the rate of demand, all control shifts to C.P. 1, with C.P. 2 decreasing to 0%. This coincides with the value for C_{R4}^{JR4} approaching zero after the value of V4 increasing above 110, indicative of the control shifting to the supply block.

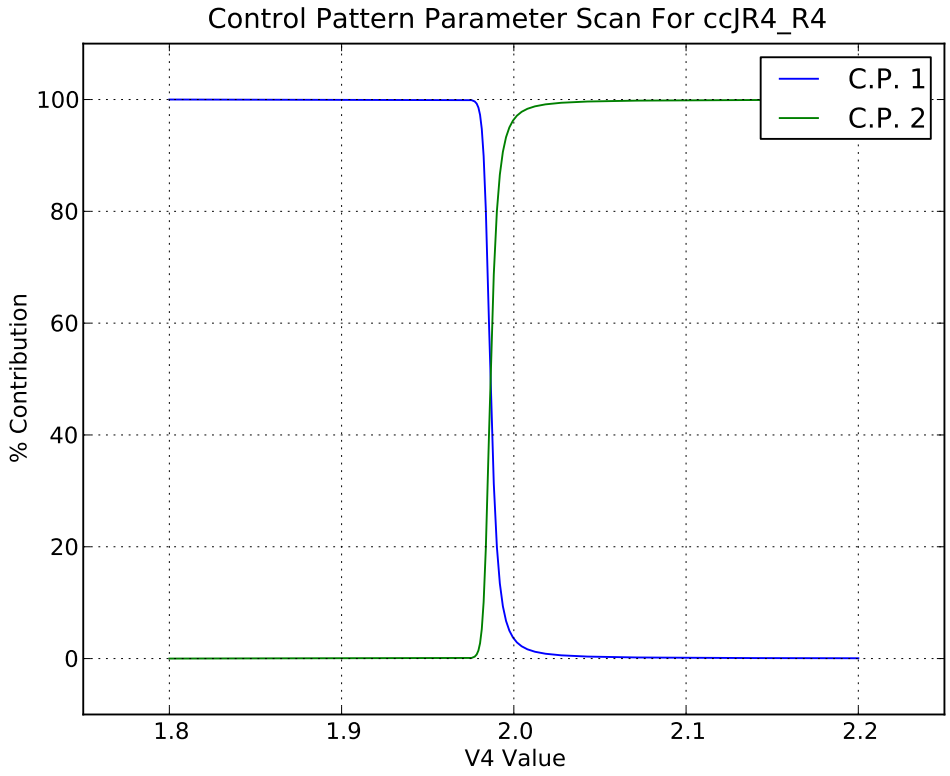

Figure 5.7: Control pattern contributions for parameter scan V4 on C_{R4}^{JR4} over a range from ≈ 1.8 to ≈ 2.2 showing the dramatic flipping of control between the two control patterns.

If we plot all flux-control coefficient values (Figure 5.8), we can see that at low V4-values all control lies in C_{R4}^{JR4}, which is the demand flux. As found for the control patterns (Figures 5.6 and 5.7), when V4 ≈ 2 we see a sharp decrease in the the demand flux control coupled with a sharp increase in the supply flux control. As V4 increases above 2 the demand flux control dramatically increases again and the supply flux control decreases sharply. This phenomena can be attributed to the demand rate characteristic dropping down to such an extent that it crosses the supply rate characteristic. At this stage both the demand and the supply slopes are very flat, resulting in a small V4-range where both supply and demand will have flux control. The shape of the curve for C_{R4}^{JR4} follows the same trends as the contribution curve for control pattern C.P. 2, which suggests that the feedback mechanism is essential for maintaining the control of the flux in the demand block. At high V4 values, which coincide with a rise in the contribution of control pattern C.P. 1 to 100%, we can see that the flux control coefficients in the supply block rise with a decrease in the demand flux control coefficient to near 0.

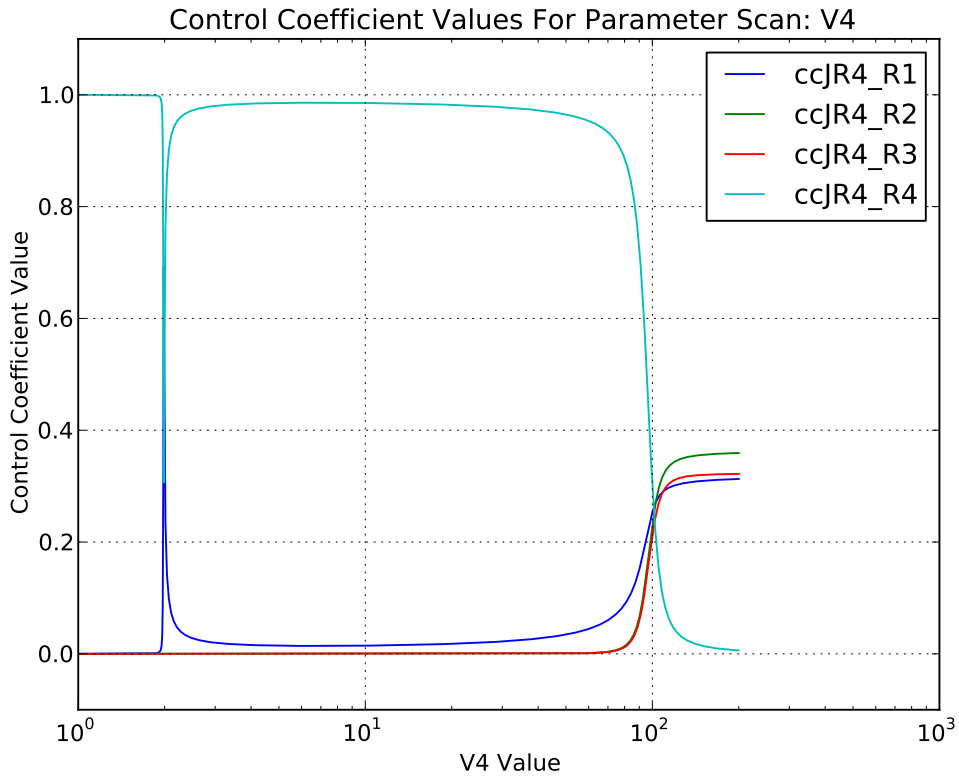

Figure 5.8: Flux-control coefficient values for a V4-parameter scan.

5.3 Discussion

The combined technique of control pattern quantification and of a parameter scan gave us a clearer picture of the role of the feedback mechanism in the homeostatic maintenance of the metabolite linking the supply and demand blocks. From the plots generated by this approach, it was clear that the feedback plays an active role in the homeostasis of the linking metabolite only under conditions which are far from equilibrium. This was deduced by inspection of the supply-demand and the control pattern parameter scan data.

Control pattern quantification identified two control patterns present in the system, one acting via the feedback loop and the other acting exclusively via the main chain.

This analysis aids in the understanding of the role of the feedback loop, by illustrating the conditions under which it is active and when it is inactive. This study has shown that under very low and very high demand conditions the feedback loop is inactive, whereas it is active under all other conditions tested. It is also apparent that when the feedback loop is active flux control lies with demand. This is due to the demand becoming saturated with X, resulting in the demand elasticity decreasing to zero. As a stand-alone technique, supply-demand analysis identifies parameter values which correlate to optimal homeostatic maintenance, thus providing a quantitative description of the regulatory nature of a cellular system. However, when used in combination with control pattern quantification we began to visualise and substantiate how this regulation is achieved to maintain optimal homeostasis within a system. By following a few steps we gained this insight quickly:

- We identified a control coefficient which in turn is determined by the biological question being investigated. In this example the demand flux C_{R4}^{JR4} was selected to describe the changing nature of the demand flux control as the rate of the demand step is increased.
- We identified a parameter to scan, and in this example the rate of the demand step was varied to simulate conditions of low to high demand.
- Since we were interested in describing the regulatory pathways associated with the demand flux, we plotted all control patterns found to play a role for this control coefficient. We based our selection of the control patterns to be plotted on a certain value as this allowed for a control pattern to play little or no role under certain conditions, but be important under different conditions. It also removed those control patterns which did not contribute under any of the conditions investigated.

Importantly, this chapter demonstrates one of the key features of `SymCA`, that is, the generation of Figures 5.4 and 5.5. SBW is used to generate the initial system layout which is then optimised by `SymCA` and the control pattern data is then mapped on the images. The initial system layout can be generated 'on the fly' by SBW or be manually created using SBW. This is a once off operation as the base layout is stored within a `layout` directory for each model analysed. By executing a few simple commands the images referred to previously can be generated:

```
import symca
sym = symca.symca('isola2a')
sym.doSca()
sym.controlPatterns()
sym.viewControlPatterns(cc=['ccJR4_R4'],image_type='eps')
```

The method responsible for the generation of the images is `viewControlPatterns`, which we discuss in more detail in Section 4.4.10. However, a requirement is that we quantify the control patterns before generating the control pattern images.

This combined strategy has quantified the role of the feedback loop in homeostatic maintenance under conditions which are far from equilibrium. The next chapter illustrates how we can apply the technique of control pattern analysis to explain changes in regulatory behaviour due to environmental manipulations.

Chapter 6

Applications of SymCA: Part II – Control pattern quantification for regulatory behaviour analysis

6.1 Introduction

In 1990, Galazzo and Bailey published work investigating the effects of suspended and alginate-entrapped *Saccharomyces cerevisiae* cells at pH 4.5 and pH 5.5 [41]. Their primary focus was the rate of production of a cell metabolite, which they stated as 'often being the most important property of a bioprocess'. This production rate is determined by the intracellular reaction rates of the pathway, in which the rates are dependent on the levels of participating enzymes and the concentrations of the metabolites. These in turn are directly influenced by a cell's environment and its genetic composition.

Galazzo and Bailey believed that to maximise metabolite production by changing the expression of certain genes and the environment, they needed to understand firstly, the flux control within the system, and secondly, how this control was affected in response to genetic and environmental changes. To achieve this, they built kinetic models describing the fermentation pathway in *Saccharomyces cerevisiae* for the following environments:

- Suspended *Saccharomyces cerevisiae* cells at pH 4.5
- Suspended *Saccharomyces cerevisiae* cells at pH 5.5
- Alginate-entrapped *Saccharomyces cerevisiae* cells at pH 4.5
- Alginate-entrapped *Saccharomyces cerevisiae* cells at pH 5.5

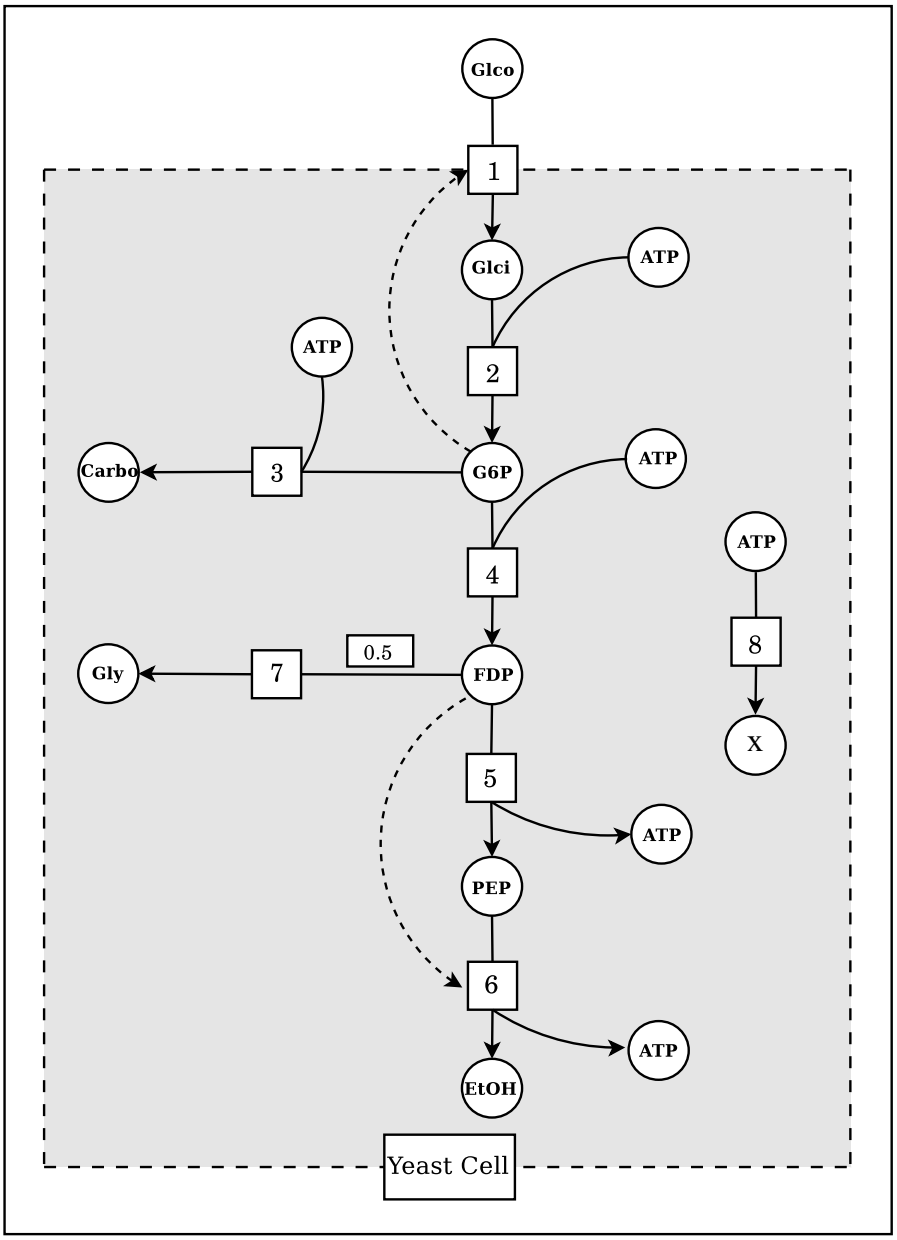

Figure 6.1: Reaction scheme of the kinetic model of fermentation pathways in *Saccharomyces cerevisiae*. Reactions are numbered 1 to 8 and are denoted by squares, metabolites are denoted by circles and any feedback loops are shown as dashed lines. A number enclosed in a box next to a line indicates a stoichiometric coefficient.

Galazzo and Bailey obtained the data required to build their models by measuring the rates of glucose uptake and of ethanol and glycerol formation in combination with *a priori* knowledge of *Saccharomyces cerevisiae* metabolic pathways for all four condi-

tions. They obtained the intracellular concentrations of substrates and effectors for most key pathway enzymes by means of *in vivo* phosphorus-31 nuclear magnetic resonance (NMR) measurements. They used MCA, which they refer as to Metabolic Control Theory (MCT), to identify the control structures for the different environments investigated.

Galazzo and Bailey found that when *Saccharomyces cerevisiae* cells are entrapped in alginate, the glucose uptake rate increased and the step with the most influence over ethanol production shifted from glucose uptake, Reaction 1, to phosphofructokinase, Reaction 4. They also showed that the rate of ATP utilisation limits ethanol production at pH 5.5 but is relatively insignificant at pH 4.5. The work described in this chapter shows how we use `SymCA` to perform control pattern quantification for the four different environments. By identifying the control pattern profiles for these environments, this study gives a quantified explanation for the reasons why the observed changes occur.

The work described in this chapter shows how we use SymCA to perform control pattern quantification for the four different environments. The biological questions we are addressing is what causes the shift in control distribution, and why the regulatory routes are changing under the different conditions. To answer these questions we identify and quantify the control pattern profiles for each of these environments.

6.2 Regulatory analysis and SymCA

When we compute the value and percentage contributions of all control patterns contained within a control coefficient, the technique of control pattern quantification enables us to understand the different underlying regulatory behaviour that arises in the situations investigated. Since the focus is on ethanol production, this investigation is concerned only with the flux-control coefficients on Reaction 6, the step catalysed by pyruvate kinase.

The data in Table 6.1 summarises the control distribution for the control coefficients associated with Reaction 6 for the different conditions and pH values investigated. We can see from this data that for most conditions there are three reactions sharing most of the control for J_{v6}, except for the immobilised cells at pH 5.5 where there are only two control coefficients. The control distribution fluctuates between the various conditions investigated, and this study gives additional insight into the changes in regulatory behaviour associated with the conditions tested.

Table 6.2 summarises all control patterns found in the three control coefficients for Reaction 6. In order to derive the number of control patterns found in each control coefficient a cut-off of 5% was used. If a control pattern was found to contribute more than 5% for any of the conditions investigated then it was included in the study.

Table 6.1: Flux control coefficients for Reaction 6 computed by `PySCeS`.

Control coefficient	**Suspended cells** pH^{ex}**5.5**	**Suspended cells** pH^{ex}**4.5**	**Immobilised cells** pH^{ex}**5.5**	**Immobilised cells** pH^{ex}**4.5**
C_{v1}^{Jv6}	**0.662**	**0.867**	**0.463**	**0.353**
C_{v2}^{Jv6}	0.000	0.000	0.000	0.000
C_{v3}^{Jv6}	0.000	0.000	-0.018	-0.031
C_{v4}^{Jv6}	**0.136**	**0.121**	**0.272**	**0.517**
C_{v5}^{Jv6}	0.000	0.000	0.000	0.000
C_{v6}^{Jv6}	0.032	0.005	0.010	0.033
C_{v7}^{Jv6}	-0.032	-0.005	-0.010	-0.033
C_{v8}^{Jv6}	**0.202**	0.014	**0.284**	**0.161**

Table 6.2: Control patterns per control coefficients, only those with a contribution of over 5% were considered for this study.

	Number of contributing control patterns			
Control coefficient	**Suspended pH 5.5**	**Suspended pH 4.5**	**Immobilised pH 5.5**	**Immobilised pH 4.5**
C_{v1}^{Jv6}	1	1	2	2
C_{v4}^{Jv6}	2	1	2	2
C_{v8}^{Jv6}	1	–	2	2

At pH 4.5 and pH 5.5 for the suspended cells, all responses caused by external perturbations are propagated via a single path for each control coefficient. However, when the cells are immobilised we can see that there are two pathways along which perturbations are propagated for both pH 4.5 and pH 5.5. This suggests a change in the regulatory behaviour and nature as a result of the immobilisation process. By analysing the underlying control patterns involved, we can gain a clearer understanding about ethanol production.

For the suspended cells at pH 4.5, control of the main pathway flux lies with Reaction 1. If we consider flux control coefficient C_{v1}^{Jv6}, we find a single control pattern which

propagates the effects of changes in Reaction 1 along the main chain. We also see that Reaction 4 is a minor controlling step for the flux through Reaction 6, and again a single control pattern is dominant. The effects of perturbations in Reaction 4 are also manifested along the main chain. However, we can see that the feedback of G6P on Reaction 1 is also involved. The control coefficients expressions as generated by `SymCA` for the three key flux-control coefficients are as follows:

$$
\begin{aligned}
&C_{v1}^{Jv6} = \\
&(-4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} - 4\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} \\
&+ 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} \\
&+ 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} - 4\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} - 4\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} \\
&+ 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} \\
&- 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} \\
&- 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} \\
&- 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5})/\sum
\end{aligned}
$$

$$
\begin{aligned}
&C_{v4}^{Jv6} = \\
&(-4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&+ 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} - 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&+ 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} - 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v8}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&+ 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} - 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v7} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v7} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&+ 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} - 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&+ 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} - 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&+ 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v2}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} + 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} \\
&- 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} - 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v2}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&- 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 8\ J_{\mathrm{v1}}\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5})/\sum
\end{aligned}
$$

$$
\begin{aligned}
&C_{v8}^{Jv6} = \\
&(-2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} + 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v7} \\
&- 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} + 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v7} \\
&+ 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} - 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v7} \\
&+ 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} - 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v7} \\
&+ 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} - 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v7} \\
&- 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v7} + 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v7} \\
&+ 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} - 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&+ 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} - 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&- 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&- 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&- 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&+ 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} - 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v7}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&- 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&- 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&+ 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} - 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&+ 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} - 2\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&+ 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} - 2\ J_{\mathrm{v3}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&- 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 2\ J_{\mathrm{v1}}\ J_{\mathrm{v5}}\ J_{\mathrm{v7}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v7}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&- 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} + 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v2}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} + 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} \\
&- 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} - 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} \\
&+ 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v5}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} - 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} \\
&+ 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v6} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v5}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v6} + 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} \\
&- 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v3}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v2}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v4}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&- 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&+ 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} - 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v4}\ \varepsilon_{FDP}^{v6}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&+ 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} - 4\ J_{\mathrm{v3}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v3}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5} \\
&- 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v1}\ \varepsilon_{Glci}^{v2}\ \varepsilon_{PEP}^{v5} + 4\ J_{\mathrm{v1}}\ J_{\mathrm{v4}}\ J_{\mathrm{v5}}\ J_{\mathrm{v8}}\ \varepsilon_{ATP}^{v6}\ \varepsilon_{FDP}^{v4}\ \varepsilon_{G6P}^{v2}\ \varepsilon_{Glci}^{v1}\ \varepsilon_{PEP}^{v5})/\sum
\end{aligned}
$$

Table 6.3: Quantified control patterns for all control coefficients in suspended cells.

		% Contribution	
Control coefficient	**Control pattern**	**Suspended pH 5.5**	**Suspended pH 4.5**
C_{v1}^{Jv6}	-4.0 J_1 J_4 J_8 ε_{ATP}^{8} ε_{FDP}^{5} ε_{G6P}^{4} ε_{Glci}^{2} ε_{PEP}^{6}	98.69	99.99
C_{v4}^{Jv6}	4.0 J_1 J_4 J_8 ε_{ATP}^{8} ε_{FDP}^{5} ε_{G6P}^{1} ε_{Glci}^{2} ε_{PEP}^{6}	94.28	99.71
	-18.0 J_3 J_4 J_8 ε_{ATP}^{8} ε_{FDP}^{5} ε_{G6P}^{3} ε_{Glci}^{2} ε_{PEP}^{6}	5.72	0.00
C_{v8}^{Jv6}	-4.0 J_1 J_4 J_8 ε_{ATP}^{4} ε_{FDP}^{5} ε_{G6P}^{1} ε_{Glci}^{2} ε_{PEP}^{6}	94.28	–
	18.0 J_3 J_4 J_8 ε_{ATP}^{4} ε_{FDP}^{5} ε_{G6P}^{3} ε_{Glci}^{2} ε_{PEP}^{6}	5.72	–

With an increase in external pH from 4.5 to 5.5, we can see that the control distribution with respect to J_{v6} is shared between Reactions 1, 4 and 8, with Reaction 1 having most control. The control pattern profile for these flux control coefficients is similar to that at pH 4.5 in that a single control pattern is dominant for each control coefficient. C_{v1}^{Jv6} and C_{v4}^{Jv6} have the same control pattern as at pH 4.5, and the effects of changes at Reaction 8 are also propagated via the main chain for C_{v8}^{Jv6}. The net change in the external pH for suspended cells is thus a shift in the control distribution, with control over the flux for Reaction 6 involving C_{v1}^{Jv6} and C_{v4}^{Jv6} at pH 4.5, and C_{v1}^{Jv6}, C_{v4}^{Jv6} and C_{v8}^{Jv6} at pH 5.5. Table 6.3 shows the data describing the control pattern contribution profiles for all relevant flux control coefficients for the suspended cells.

When *Saccharomyces cerevisiae* is immobilised in alginate beads, the appearance of the control coefficient distribution changes. This changing profile coincides with a difference in the control pattern distribution associated with the various control coefficients involved. Although the distribution still involves the same three flux control coefficients as found for suspended cells at pH 5.5, there are now two contributing control patterns for each flux control coefficient at pH 4.5 and pH 5.5 and the balance in distribution has shifted. This is seen in Table 6.4.

At pH 4.5, the control is distributed between C_{v4}^{Jv6}, C_{v1}^{Jv6} and C_{v8}^{Jv6} with control coefficient values of 0.517, 0.35 and 0.16 respectively. All three flux control coefficients now contain two contributing control patterns as opposed to the single control pattern present in the suspended cells at pH 4.5. For C_{v4}^{Jv6}, we now find that the dominant control pattern no longer takes into account the feedback of G6P, but acts via the main chain and the branch point at Reaction 3.

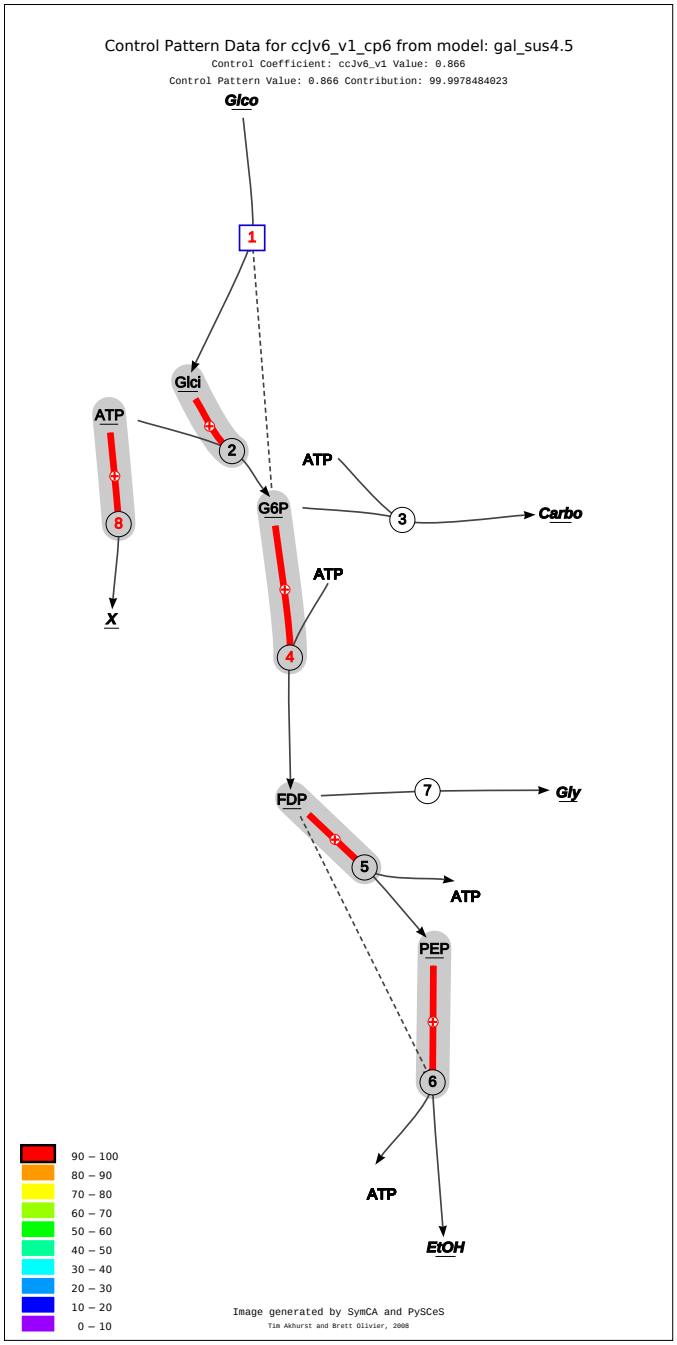

Figure 6.2: Dominant control pattern for C_{v1}^{Jv6} for suspended cells at pH 4.5 & pH 5.5. The perturbed reaction is denoted by a blue square, the elasticities found in the pattern are shown in red with a grey bubble surrounding them and the reaction they affect. All fluxes present in the control pattern are shown as red reaction numbers and the dashed lines indicate feedback loops.

We would thus expect an increase in the rates of Reactions 3 and 6 on a perturbation of Reaction 4. C_{v1}^{Jv6} has the second highest degree of control, and like C_{v4}^{Jv6}, there are now two contributing control patterns, one affecting only the main chain and the other affecting the main chain and the branch point at Reaction 3, with contributions of 58.56% and 41.44% respectively. The situation is similar for C_{v8}^{Jv6} in that there are two contributing control patterns. One effects only the main chain and contributes 85.47%, while the other control pattern and contribution affects the feedback of G6P on Reaction 1 and the main chain.

Table 6.4: Quantified control patterns for all control coefficients in immobilised cells.

		% Contribution	
Control coefficients	**Control patterns**	**Immobilised pH 5.5**	**Immobilised pH 4.5**
C_{v1}^{Jv6}	-4.0 J_1 J_4 J_8 ε_{ATP}^{8} ε_{FDP}^{5} ε_{G6P}^{4} ε_{Glci}^{2} ε_{PEP}^{6}	26.89	58.56
	16.0 J_1 J_3 J_8 ε_{ATP}^{4} ε_{FDP}^{5} ε_{G6P}^{3} ε_{Glci}^{2} ε_{PEP}^{6}	73.11	41.44
C_{v4}^{Jv6}	4.0 J_1 J_4 J_8 ε_{ATP}^{8} ε_{FDP}^{5} ε_{G6P}^{1} ε_{Glci}^{2} ε_{PEP}^{6}	13.27	14.53
	-18.0 J_3 J_4 J_8 ε_{ATP}^{8} ε_{FDP}^{5} ε_{G6P}^{3} ε_{Glci}^{2} ε_{PEP}^{6}	86.73	85.47
C_{v8}^{Jv6}	-4.0 J_1 J_4 J_8 ε_{ATP}^{4} ε_{FDP}^{5} ε_{G6P}^{1} ε_{Glci}^{2} ε_{PEP}^{6}	94.28	14.53
	18.0 J_3 J_4 J_8 ε_{ATP}^{4} ε_{FDP}^{5} ε_{G6P}^{3} ε_{Glci}^{2} ε_{PEP}^{6}	5.72	85.47

When the pH is increased to 5.5 for the immobilised cells, C_{v1}^{Jv6}, C_{v4}^{Jv6} and C_{v8}^{Jv6} again share the control, and each control coefficient has two associated control patterns. C_{v1}^{Jv6} has the highest degree of control, 0.46, with the remainder of the control shared between C_{v4}^{Jv6} and C_{v8}^{Jv6}. The control patterns which contribute to the control of C_{v1}^{Jv6} are the same two that are active at pH 4.5. However, unlike the patterns at pH 4.5, the pattern affecting both the main chain flux and the flux through Reaction 3 is now the dominant control pattern and contributes 73.11%. The control pattern affecting only the main chain flux now contributes 26.89%.

With regard to C_{v4}^{Jv6}, the contributions of the control patterns remain relatively unchanged when compared with the data from pH 4.5. The balance of control lies within the control pattern affecting the branch point at Reaction 3 as well as the main chain, and the remaining pattern affecting the main chain via the feedback of G6P. However, for C_{v8}^{Jv6} the control pattern acting via the main chain and the feedback of G6P can now be seen to be dominant.

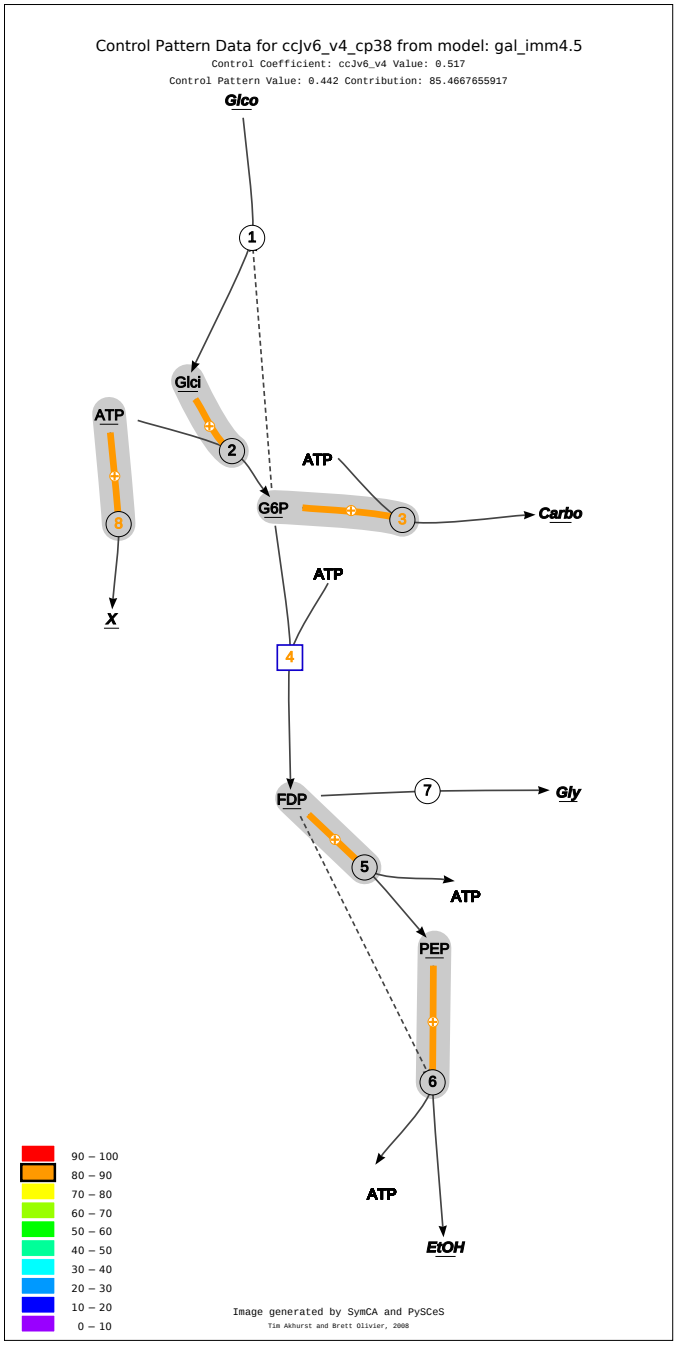

Figure 6.3: Dominant control pattern for C_{v4}^{Jv6} for immobilised cells at pH 4.5. The perturbed reaction is denoted by a blue square, the elasticities found in the pattern are shown in orange with a grey bubble surrounding them and the reaction they affect. All fluxes present in the control pattern are represented by an orange reaction number and the dashed lines show feedback loops.

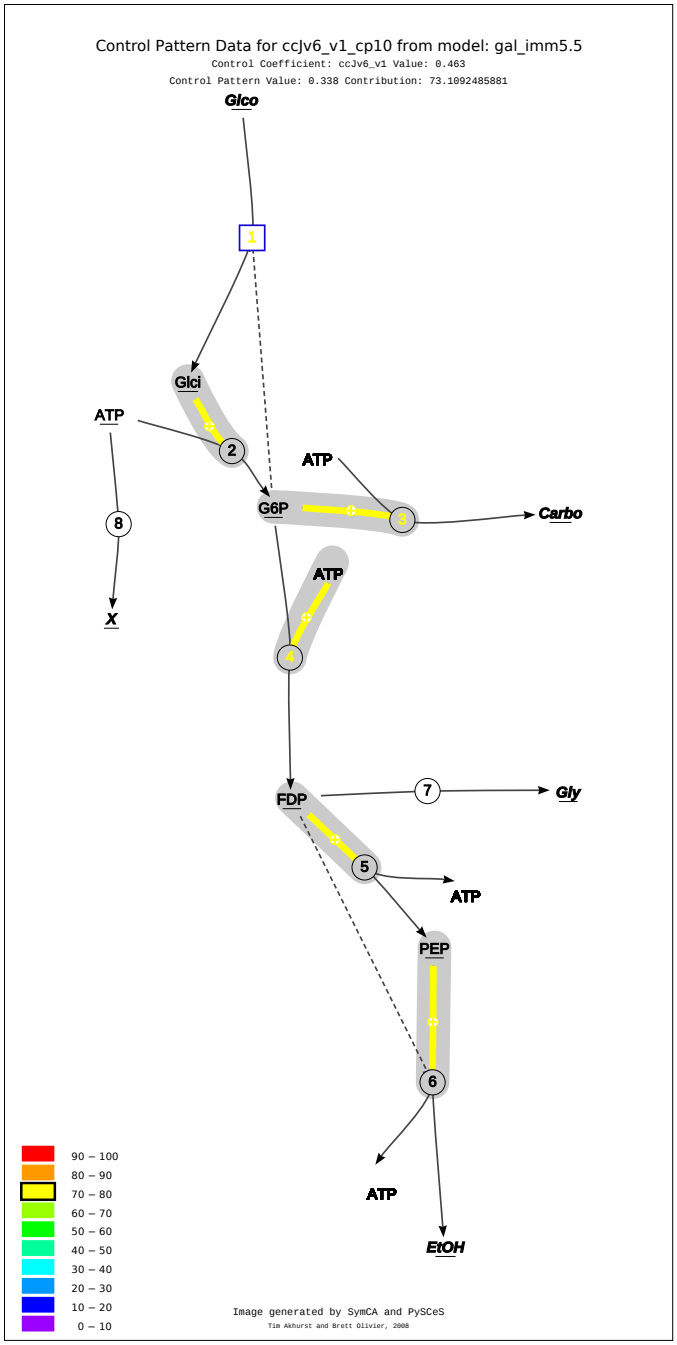

Figure 6.4: Dominant control pattern for C_{v1}^{Jv6} for immobilised cells at pH 5.5. The perturbed reaction is denoted by a blue square, the elasticities found in the pattern are shown in yellow with a grey bubble surrounding them and the reaction they affect. All fluxes present in the control pattern are represented by an yellow reaction number and the dashed lines show feedback loops.

6.3 Discussion

The technique of control pattern quantification gave us a means to have a deeper look into the workings of a cellular system by identifying the key regulatory routes. When applied to the kinetic models described by Galazzo and Bailey [41], we could identify the key regulatory routes under different environmental conditions and thus provide quantified descriptions for their biological observations. From the analysis, it became clear that when the cells are in suspension the bulk of the effects are mediated along the main chain, with a lesser contribution arising from the feedback of G6P on Reaction 1, coupled with the affects via the main chain. This was so for pH 4.5 and pH 5.5.

The immobilisation of the cells in alginate beads resulted in changes in the control distribution profile at pH 4.5 and pH 5.5. At pH 4.5 the control of the main pathway flux shifted from Reaction 1 to Reaction 4 when compared with the suspended cells at pH 4.5. As mentioned previously, all control exerted via Reaction 1 was mediated via the main chain for suspended cells at pH 4.5. However, the decrease in control by this step coincided with an increase in contribution of the control pattern influencing both the main chain and the branch point at Reaction 3, from 0% to 41.44%. C_{v4}^{Jv6} followed a similar trend in that the dominant control pattern for the immobilised cells also affected the system via the main chain and the branch point, with a minor contribution via the G6P feedback and the main chain. C_{v8}^{Jv6} had far less control than both C_{v1}^{Jv6} and C_{v4}^{Jv6}, but we observed the same control pattern behaviour as C_{v4}^{Jv6}.

The increase in pH to 5.5 resulted in a further shift of the control distribution. However, the same general trends as found at pH 4.5 were present with the dominant control pattern for each control coefficient affecting both the branch point at Reaction 3 and the main chain. The major change as a result of the change in pH was that of the flux control distribution between the flux control coefficients. The different nature of the control pattern contributions gave an insight into why polysaccharide storages increase due to the immobilisation process. We saw that the dominant control pattern affected the branch point at Reaction 3, which resulted in an increase in polysaccharides. We observed this for the dominant control coefficient at pH 4.5 and pH 5.5, where C_{v4}^{Jv6} had a control coefficient value of 0.57 and this pattern accounts for 85.47% at pH 4.5 and 86.73% at pH 5.5. C_{v1}^{Jv6} had a control coefficient value of 0.46, with the pattern contributing 73.11% at pH 5.5. Of further interest was the change in control pattern contributions for C_{v1}^{Jv6} at pH 4.5 and pH 5.5, where the dominant pattern affected the main chain at pH 4.5 and the dominant pattern at pH 5.5 affected the main chain and the branch point.

The observed increase of polysaccharides, which was noticed when *Saccharomyces cerevisiae* cells were immobilised, clearly coincided with the emergence of the control pattern affecting the flux for the branch leading to polysaccharide accumulation as the dominant pattern. Control pattern quantification therefore explained observed outcomes from the conditions under which this system was analysed.

As observed by Galazzo and Bailey, the glucose uptake step (Reaction 1) contained most of the flux control at pH 4.5 and pH 5.5. At pH 5.5 ATPase (Reaction 8) also exerted control over the pathway flux, and this was possibly due to the decrease in Vmax values. At pH 4.5 the Vmax was 35 whereas at pH 5.5 it decreases to 12.1, implying that less substrate could be converted to product which may explain why this step contributed to the flux control at that stage. The immobilised cells demonstrated that PFK (Reaction 4) exhibited a higher degree of flux control within the system, which may be partly responsible for the observed increase in the flux through the branch leading to polysaccharide formation. The observed increase in polysaccharide storage for the immobilised cells was mainly due to the increase in glucose uptake which increases from ≈17 for suspended cells to ≈38. This change resulted in an increase in the flow into glycolysis, which in turn led to an increase in the flux in all the branch points further down the pathway (ethanol, glycerol and polysaccharide production). This was confirmed by the emergence of the control patterns leading to polysaccharide production on cell immobilisation at pH 4.5 and pH 5.5.

While the quantified control patterns provide tangible evidence that the pathway leading to an increase in polysaccharide formation becomes activated under immobilised conditions, it is equally important to look at the underlying system properties that give rise to this behaviour. Here we will focus on the control patterns for C_{v1}^{Jv6}, which we will discuss in detail by way of an example to illustrate the possibilities when using the symbolic approach. C_{v1}^{Jv6} is the dominant control coefficient at both pH 4.5 and 5.5 for the suspended cells, as well as for the immobilised cells at pH 4.5. For the suspended cells there is a single control pattern for C_{v1}^{Jv6}: -4.0$J_1 J_4 J_8 \varepsilon_{ATP}^{8} \varepsilon_{FDP}^{5} \varepsilon_{G6P}^{4} \varepsilon_{Glci}^{2} \varepsilon_{PEP}^{6}$. This pattern contains the following main chain elasticities ε_{Glci}^{2}, ε_{G6P}^{4}, ε_{FDP}^{5} and ε_{PEP}^{6} with values of 0.80006, 0.86348, 0.71156 and 0.65464, respectively. Also present in the control pattern are three fluxes, J_1, J_4 and J_8, with values of 17.59, 11.727 and 20.839 for pH 4.5 and 16.37, 10.898 and 19.346 for pH 5.5, respectively.

If we now look at the control patterns associated with C_{v1}^{Jv6} for the immobilised cells, we find that there are two contributing control patterns. The first is the same as the one illustrated previously and the second, 16.0$J_1 J_3 J_8 \varepsilon_{ATP}^{4} \varepsilon_{FDP}^{5} \varepsilon_{G6P}^{3} \varepsilon_{Glci}^{2} \varepsilon_{PEP}^{6}$, includes the elasticity for the branch point reaction leading to polysaccharide formation, ε_{G6P}^{3}. If we focus on the main chain elasticities, ε_{Glci}^{2}, ε_{G6P}^{4}, ε_{FDP}^{5} and ε_{PEP}^{6}, we can see that they have all decreased on immobilisation of the cells, with values of 0.59225, 0.47376, 0.57433 and 0.39170, respectively. We can see the same trend for the branch point elasticity, ε_{G6P}^{3}, which decreases from 8.7813 to 5.985 when one compares the suspended with the immobilised cells. Of equal importance are the fluxes present in the control patterns as these can have a profound effect on the contribution of a control pattern. We can see that the second control pattern that emerges for the immobilised cells includes the flux J_3, increases from 9.6842e-05 at pH 4.5 and 5.1811e-03 at pH 5.5 for the suspended cells to 1.4635 at pH 4.5 and 2.2093 at pH 5.5 for the immobilised cells. This increase in flux

provides sound evidence as to why this second control pattern is now emerging, and thus why an increase in polysaccharide formation is observed.

Additionally, immobilisation of the cells results in the Vmax for the glucose uptake step increasing from 19.7 to 45.6, resulting in a increased rate of glucose uptake, which in turn leads to an increase in the concentration of G6P from 5.7021e-01 at pH 4.5 and 8.999e-01 at pH 5.5 for the suspended cells to 1.8115 at pH 4.5 and 1.9554 at pH 5.5 for the immobilised cells. The decrease in the main chain elasticities may suggest that they are becoming increasingly saturated with substrate, thus leading to an increase in the concentration of G6P. Coupled with this, the increase in the flux for the branch point reaction provide evidence as to why the increase in polysaccharide formation occurs, which is confirmed by the emergence of the control pattern leading to this phenomenon.

This investigation has demonstrated how we can use control pattern quantification to give additional insight and to quantify observed outcomes when a system is subjected to changing genetic and environmental conditions. These changes resulted in changes to metabolite production, which then focussed on ethanol formation. By using control patterns we could quantify how these different conditions affected the regulation within the system. In particular we could explain the observed glucose uptake rate on immobilisation of the cells by 'turning on' part of the system which led to an increase in carbohydrate formation. This has implications in biotechnology where the production of certain metabolites within a system is desirable. By determining the key control patterns we can find out which parts of the system are active under the conditions tested. We can optimise the conditions accordingly to maximise the activity of the pathways leading to desired metabolite formation and minimising the activity of the pathways that are undesirable. The control patterns can then be used to quantify the experimental outcomes.

Control pattern quantification proved to be a useful tool to explain the outcomes of a system subjected to environmental and genetic changes. In the following chapter this technique is used to expand on work done by Uys *et al.* [116], in which they extended a previous model of sucrose accumulation in sugar cane developed by Rohwer and Botha [89]. This work focusses on the final step leading to sucrose accumulation, and explains the observed changes for all internodes within the sugar cane plant.

Chapter 7

Applications of SymCA: Part III – Symbolic control analysis of sucrose accumulation

7.1 Introduction

Insufficient knowledge surrounding factors affecting sugar accumulation in sugar cane (*Saccharum officinarum*) and no clear pattern detailing which enzyme activities were important controlling steps led Rohwer and Botha [89] to develop a kinetic model of sugar accumulation in developing sugar cane. They based their approach on pathway analysis [103, 104] and kinetic modelling [13, 91] to investigate the control of futile cycling (synthesis and breakdown) and sucrose accumulation in sugar cane.

Rohwer and Botha enumerated all possible routes of futile cycling in the system by using the concept of elementary flux modes [105]. They then built a kinetic model of sucrose accumulation in sugar cane culm tissue by collecting available kinetic data for the pathway enzymes. They found that the model was in agreement with independent experimental data, thus validating it. They then used the model to establish various enhancement strategies for increasing sugar accumulation.

Rohwer and Botha calculated the control coefficient of each enzyme in the system on futile cycling of sucrose. They then identified the numerically largest control coefficients and varied the activities of those enzymes over a five-fold range to investigate the affect on the degree of futile cycling, the net sucrose accumulation and the net conversion efficiency from hexoses into sucrose. Their findings suggested that the most promising targets for genetic manipulation are overexpression of the fructose or glucose transporter or the vacuolar sucrose import protein, as well as reduction of cytosolic neutral invertase levels.

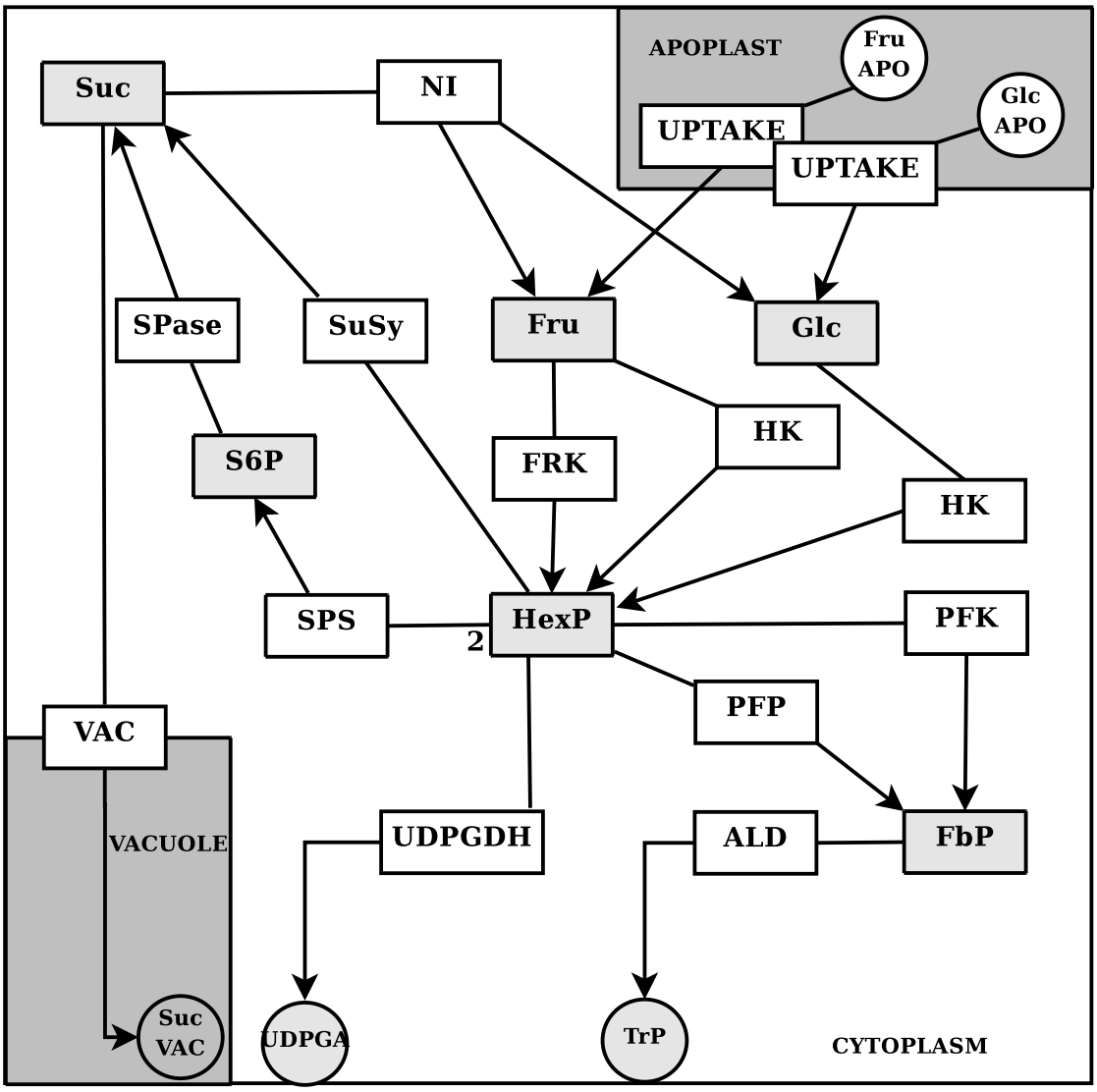

Figure 7.1: Extended model of sucrose accumulation as illustrated in Uys *et al.* [116]. Variable metabolites are shown in light grey shaded rectangular boxes and fixed in light grey shaded circles. Enzymes are shown in unshaded rectangular boxes. A number 2 next to a line indicates a stoichiometric coefficient. *Abbreviations for enzymes and transport steps:* ALD, Aldolase; FRK, Fructokinase; HK, Hexokinase; NI, Neutral Invertase; PFK, Phosphofructokinase; PFP, Pyrophosphate-dependent PFK; SPase, Sucrose phosphatase; SPS, Sucrose phosphate synthetase; SuSy, Sucrose synthase ; UDPGDH, UDP-Glucose dehydrogenase; VAC, vacuolar sucrose import. *Metabolites:* FbP, Fructose-1,6-bisphosphate; Fru, Fructose; Glc, Glucose; HexP, Hexose phosphate; S6P, Sucrose-6-phosphate; Suc, Sucrose; Trp, triose phosphate; UDPGA, UDP-Glucuronic acid.

Uys and colleagues [116] were interested in understanding more about why or how commercial varieties of sugar cane are able to accumulate sucrose in high concentrations. They proposed that kinetic modelling may aid determination of the factors controlling sucrose accumulation or lead the design of experimental optimisation strategies. Their work focussed on extending the model assembled by Rohwer and Botha, by accounting for isoforms of sucrose synthase (SuSy A, SuSy B & SuSy C) and fructokinase (FRK A & FRK B), the glycolytic enzymes phosphofructokinase (PFK), pyrophosphate-dependent PFK and aldolase, and carbon partitioning towards fibre formation. In addition, they included maximal activity data of the enzymes measured in the different Internodes, thus creating a growth model describing the metabolic behaviour as sugar cane parenchymal tissue matures from Internodes 3-10.

Uys *et al.* [116] found that the extended model supported a hypothesis of vacuolar sucrose accumulation against a concentration gradient. Futile cycling of sucrose appeared to decrease as the maturity of the sugar cane internodes increased, coinciding with an increase in sucrose accumulation. Metabolic control analysis revealed that each isoform of sucrose synthase had a unique control profile, and they suggested that the synthesis and storage of sucrose is mostly dependent on SuSy C.

This section details further analysis based on the extended model of sucrose accumulation shown in Figure 7.1. The focus is on the final step leading to sucrose accumulation in the vacuole using symbolic control analysis and control pattern quantification for all internodes as described by Uys *et al.*. The aim of this investigation is to explain the observations of Uys *et al.* using control pattern quantification.

The following work describes how `SymCA` was used to perform control pattern quantification for the maturing sugar cane internodes, with the objective to provide additional understanding how or why the maturity of sugar cane nodes impacts on its ability to accumulate sucrose, and what regulatory routes are responsible for this phenomenon. In order to answer this question we identify and quantify the control pattern profiles for each internode as described by Uys *et al.*

7.2 Regulatory analysis and SymCA

We computed the values for all control coefficients on the flux to sucrose accumulation to determine the distribution of control for this step (Table 7.1). The data in Table 7.1 shows that the control for reaction VAC is distributed between three control coefficients for all internodes, C^{JVAC}_{FRKa}, $C^{JVAC}_{FRU\_uptake}$ and $C^{JVAC}_{HK\_GLC}$. The quantified control patterns for these control coefficients are illustrated and discussed for each internode.

Table 7.1: Flux control coefficients, with a value greater than 0.1 for a single internode, on J_{VAC} computed by `PySCeS`, bold values indicate the key control coefficients for each internode.

Control	**Internode**							
coefficient	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
C_{FRKa}^{JVAC}	**-0.285**	-0.117	**0.279**	**0.287**	**0.294**	**0.224**	0.121	0.088
$C_{FRU\_uptake}^{JVAC}$	**0.989**	**0.820**	0.151	**0.224**	0.124	**0.249**	**0.268**	**0.362**
$C_{HK\_GLC}^{JVAC}$	0.153	0.166	**0.487**	**0.379**	**0.502**	**0.382**	**0.417**	**0.279**
$C_{GLC\_uptake}^{JVAC}$	0.097	0.099	0.112	0.093	0.071	0.095	0.118	0.119
C_{SuSyc}^{JVAC}	0.025	0.034	0.021	0.037	0.037	0.056	0.068	0.111
C_{VAC}^{JVAC}	0.074	0.081	0.085	0.081	0.065	0.087	0.099	0.112

7.2.1 Internode 3

We can see that the control for Internode 3 lies predominantly with $C_{FRU\_uptake}^{JVAC}$ with a value of 0.989. The control pattern quantification data in Table 7.2 shows that there are eight control patterns responsible for ≈75% of the control. Of these eight, three (3c−3e) affect both the pathway resulting in sucrose accumulation in the vacuole and the production of Glc via the step NI, which forms part of the futile cycling. These three patterns account for 22.92% of the control.

Table 7.2: Internode 3 control patterns for $C_{FRU\_uptake}^{JVAC}$

ID	**Control pattern**	**% Contribution**
3a	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{PFK}}\ J_{\mathrm{SuSyc}}$	8.93
	$\varepsilon_{FBP}^{ALD}\ \varepsilon_{FRUcyt}^{SuSyc}\ \varepsilon_{uptake\_GLCcyt}^{GLC}\ \varepsilon_{HEXP}^{PFK}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	
3b	$\mathbf{+2.0\ J_{ALD}\ J_{FRU\_uptake}\ J_{GLC\_uptake}\ J_{PFK}\ J_{SuSya}}$	**19.90**
	$\boldsymbol{\varepsilon_{FBP}^{ALD}\ \varepsilon_{FRUcyt}^{SuSya}\ \varepsilon_{uptake\_GLCcyt}^{GLC}\ \varepsilon_{HEXP}^{PFK}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}}$	
3c	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{NI}}\ J_{\mathrm{PFK}}\ J_{\mathrm{SuSya}}$	11.38
	$\varepsilon_{FBP}^{ALD}\ \varepsilon_{FRUcyt}^{SuSya}\ \varepsilon_{GLCcyt}^{NI}\ \varepsilon_{HEXP}^{PFK}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	

continued on next page

continued from previous page

ID	Control Pattern	% Contribution
3d	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{NI}}\ J_{\text{PFK}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.11
3e	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{NI}}\ J_{\text{SPS}}\ J_{\text{SuSya}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSya}_{FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.43
3f	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.05
3g	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}\ J_{\text{SuSya}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSya}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	11.25
3h	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFK}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.57

The remaining five control patterns (3a, 3b & 3f-3h) all manifest their control via the Fru → SuSy (A,B,C) → Suc → VAC → Suc VAC pathway, with the smallest percentage via SuSy C (3a & 3f), which has been suggested to be the isozyme responsible for the greatest accumulation of sucrose in the vacuole, and the largest contribution via SuSy A (3b & 3f). The presence of futile cycling can be seen by the negative control of C^{JVAC}_{FRKa} with a value of -0.285. The patterns associated with C^{JVAC}_{FRKa} include the pathway via NI (3c, 3d & 3e) and accounted for 36.85% of the total contribution (data not shown).

Figure 7.2 represents the major control pattern found for $C^{JVAC}_{FRU\_uptake}$: $+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFK}}\ J_{\text{SuSya}}\ \varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSya}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$, which is bold in Table 7.2 and accounts for 19.90% of the control for $C^{JVAC}_{FRU\_uptake}$.

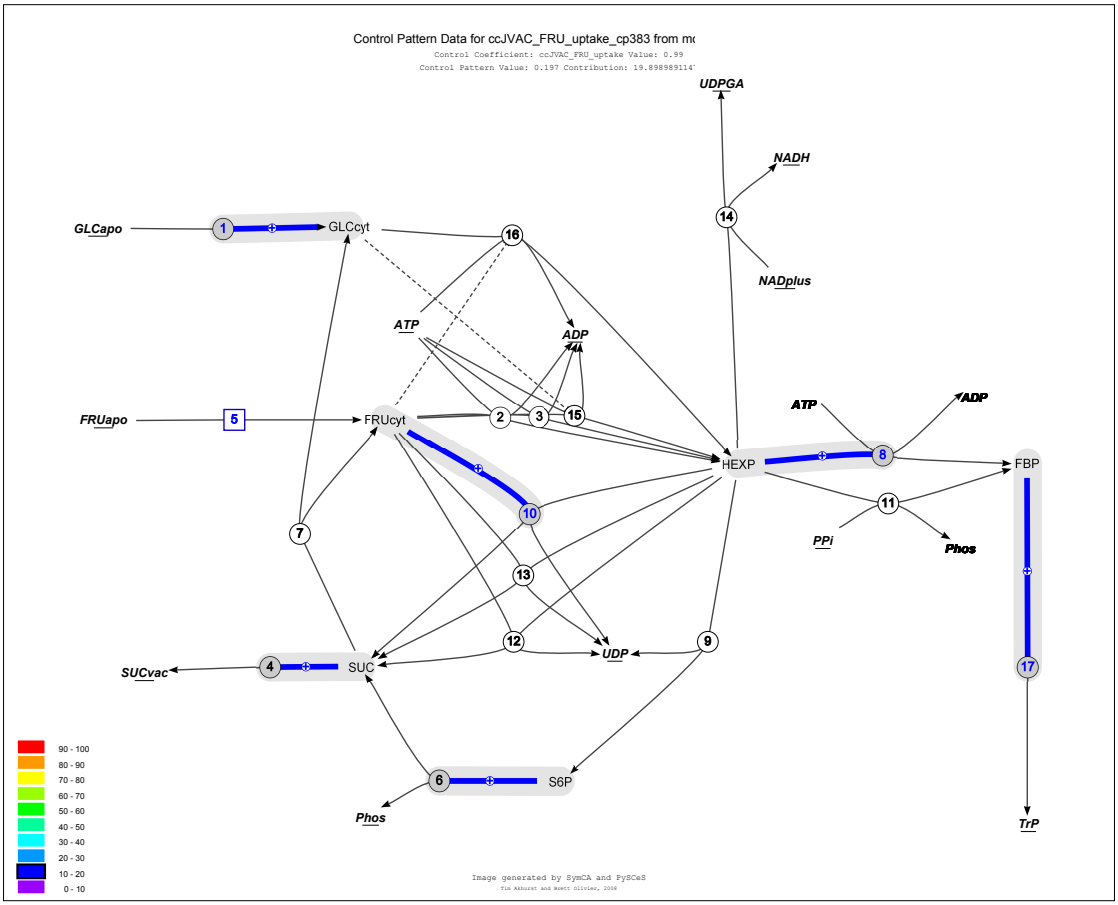

Figure 7.2: Dominant control pattern for $C^{JVAC}_{FRU\_uptake}$ in Internode 3. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations:* 1, GLC_uptake; 2, FRKa; 3, FRKb; 4, VAC; 5, FRU_uptake; 6, Spase; 7, NI; 8, PFK; 9, SPS; 10, SuSya; 11, PFP; 12, SuSyc; 13, SuSyb; 14, UDPGDH; 15, HK_FRU; 16, HK_GLC; 17, ADL. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines.

7.2.2 Internode 4

The data for Internode 4 shows FRU_uptake as the controlling reaction, since $C^{JVAC}_{FRU\_uptake}$ has a value of 0.82. The quantified control patterns in Table 7.3 show that there is no dominant pattern, and that all patterns involved show an almost equal contribution.

Table 7.3: Internode 4 control patterns for $C^{JVAC}_{FRU\_uptake}$

ID	Control pattern	% Contribution
4a	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{PFK}}\ J_{\mathrm{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	9.65
4b	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{PFK}}\ J_{\mathrm{SuSya}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSya}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	10.45
4c	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{NI}}\ J_{\mathrm{SPS}}\ J_{\mathrm{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.75
4d	$\mathbf{+2.0\ J_{ALD}\ J_{FRU\_uptake}\ J_{NI}\ J_{PFK}\ J_{SuSya}}$ $\boldsymbol{\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSya}_{FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}}$	**6.90**
4e	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{SPS}}\ J_{\mathrm{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.69
4f	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{NI}}\ J_{\mathrm{PFK}}\ J_{\mathrm{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.37
4g	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{NI}}\ J_{\mathrm{SPS}}\ J_{\mathrm{SuSya}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSya}_{FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.23
4h	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{SPS}}\ J_{\mathrm{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	8.71
4i	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{SPS}}\ J_{\mathrm{SuSya}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSya}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	9.43
4j	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{PFK}}\ J_{\mathrm{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.30

As for Internode 3, there are several control patterns which affect the pathway via NI (4c, 4d, 4f & 4g) and account for 25.25% of the contributions, whereas the remaining contribution acts via the SuSy (A,B,C) isozyme. The pathway acting via SuSy C (4a & 4h) accounts for about 17% of the control, whereas the pathway acting via SuSy B (4e & 4j) accounts for about 20%. Figure 7.3 represents control pattern 4d in Table 7.3, which shows the futile cycling present.

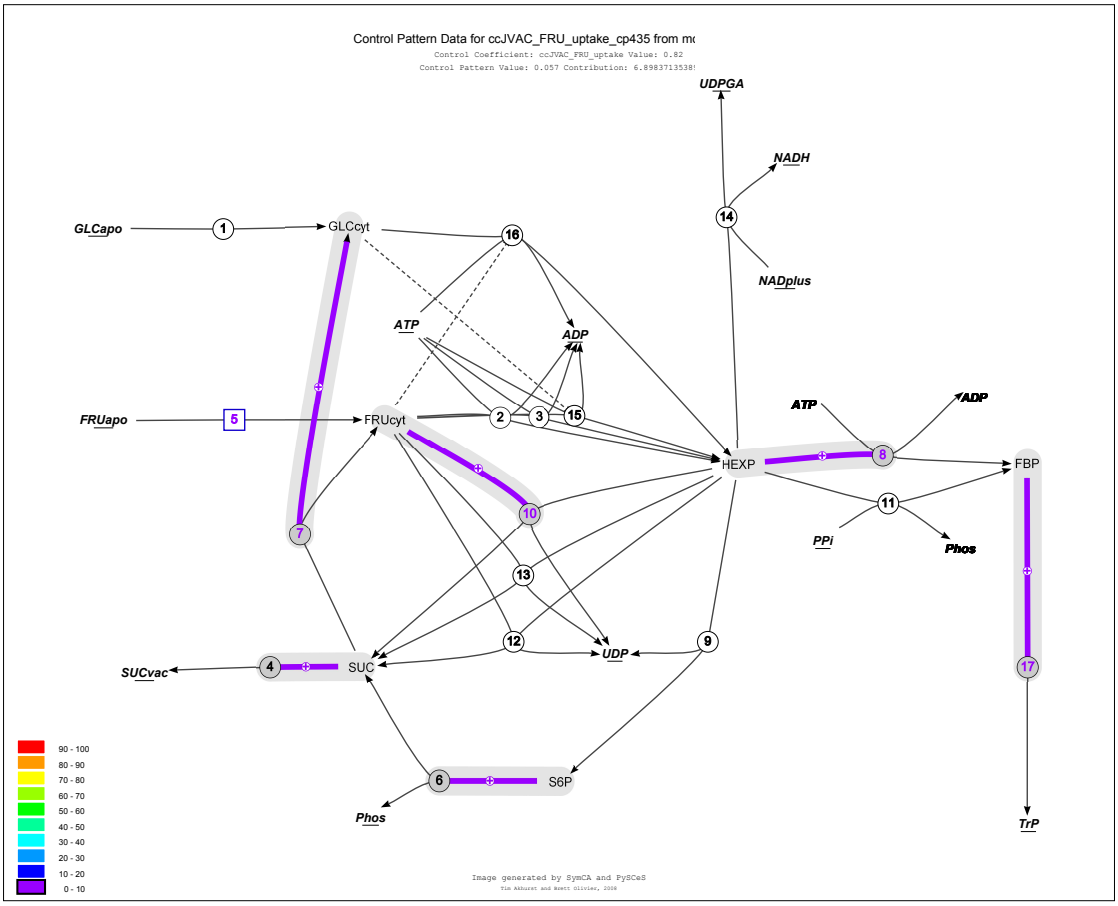

Figure 7.3: Control pattern illustrating pathway via NI for $C^{JVAC}_{FRU\_uptake}$ in Internode 4. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines.

7.2.3 Internode 5

The situation changes slightly for Internode 5 in that there are now two control coefficients involved in controlling J_{VAC}. These are $C^{JVAC}_{HK\_GLC}$ and C^{JVAC}_{FRKa} with values of 0.472 and 0.279 respectively. Tables 7.4 and 7.5 show this control pattern data.

Table 7.4: Internode 5 control patterns for $C^{JVAC}_{HK\_GLC}$

ID	Control pattern	% Contribution
5.1a	$-2.0\ J_{\text{ALD}}\ J_{\text{GLC\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSyb}}$	29.53
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
5.1b	$-2.0\ J_{\text{ALD}}\ J_{\text{GLC\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSya}}$	24.33
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSya}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
5.1c	$\mathbf{-2.0\ J_{ALD}\ J_{GLC\_uptake}\ J_{GLC\_uptake}\ J_{HK\_GLC}\ J_{SuSyc}}$	**37.33**
	$\boldsymbol{\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}}$	

Table 7.5: Internode 5 control patterns for C^{JVAC}_{FRKa}

ID	Control pattern	% Contribution
5.2a	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{GLC\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSya}}$	22.04
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSya}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
5.2b	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{GLC\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyc}}$	8.86
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
5.2c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{GLC\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSya}}$	5.78
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SuSya}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
5.2d	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{GLC\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSyc}}$	33.83
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
5.2e	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{GLC\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSyb}}$	26.76
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
5.2f	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{GLC\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyb}}$	7.01
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	

For $C^{JVAC}_{HK\_GLC}$ we find three dominant control patterns all manifesting their control via the SuSy (A,B,C) isozyme, with the highest contribution via the SuSy C enzyme (5.1c) This pattern is in bold and is shown in Figure 7.4. In contrast there are six key control patterns for C^{JVAC}_{FRKa}, three of which influence the pathway via NI (5.2b, 5.2c & 5.2f). The remaining three control patterns (5.2a, 5.2d & 5.2e) all affect the pathway via SuSy (A,B,C) and are by far the largest contributors to the control of this step with a combined contribution of 82.63% as opposed to the contribution of the patterns including step NI, which is 21.65. Thus, for Internode 5 the preference is for the pathways leading to vacuolar sucrose accumulation.

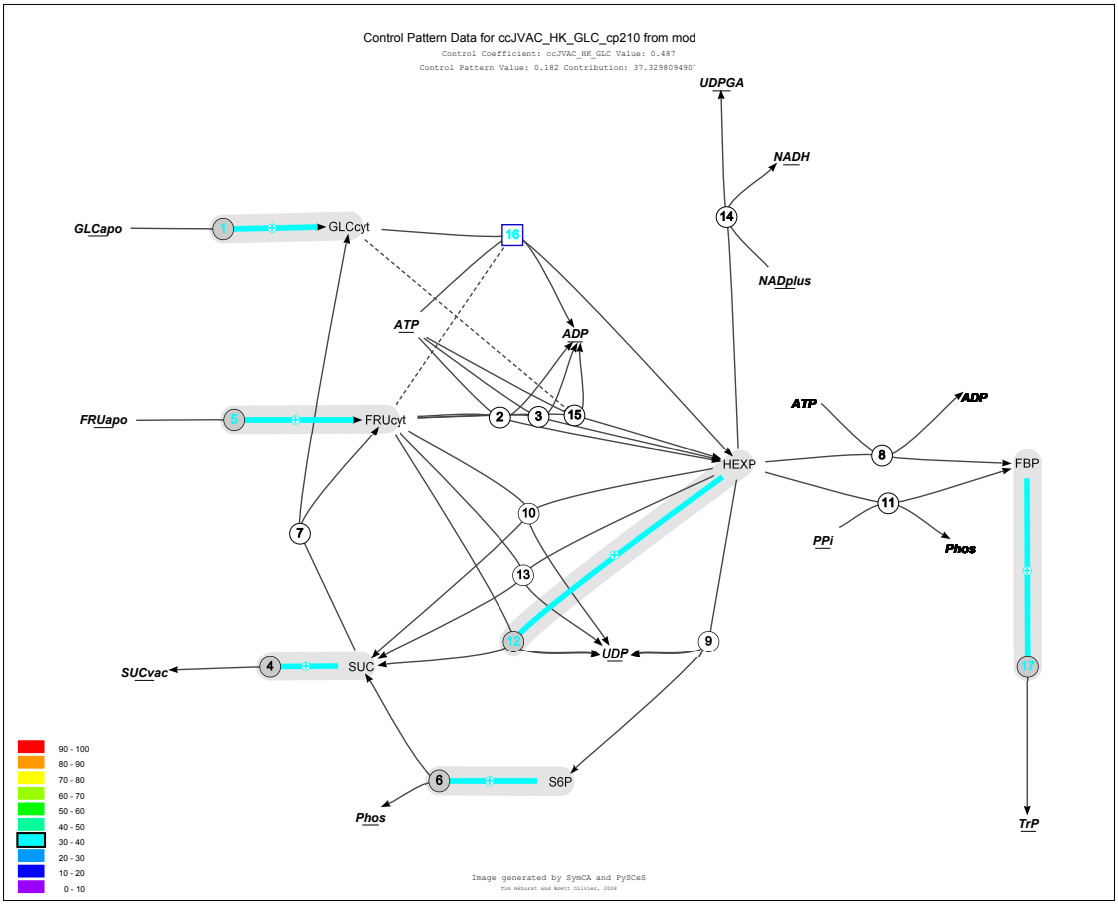

Figure 7.4: Major contributing control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 5. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines.

7.2.4 Internode 6

There are three control coefficients involved for Internode 6, $C^{JVAC}_{HK\_GLC}$, C^{JVAC}_{FRKa} and $C^{JVAC}_{FRU\_uptake}$ with values of 0.379, 0.287 and 0.224 respectively. These control coefficients demonstrate an almost even distribution between all three control coefficients. $C^{JVAC}_{HK\_GLC}$ is the largest control coefficient and contains four major contributing control patterns. We can see that three of these control patterns manifest their control via the SuSy isozymes, with the major contributing pattern via SuSy C (6.1d), which is in bold text in Table 7.6 and is shown in Figure 7.5.

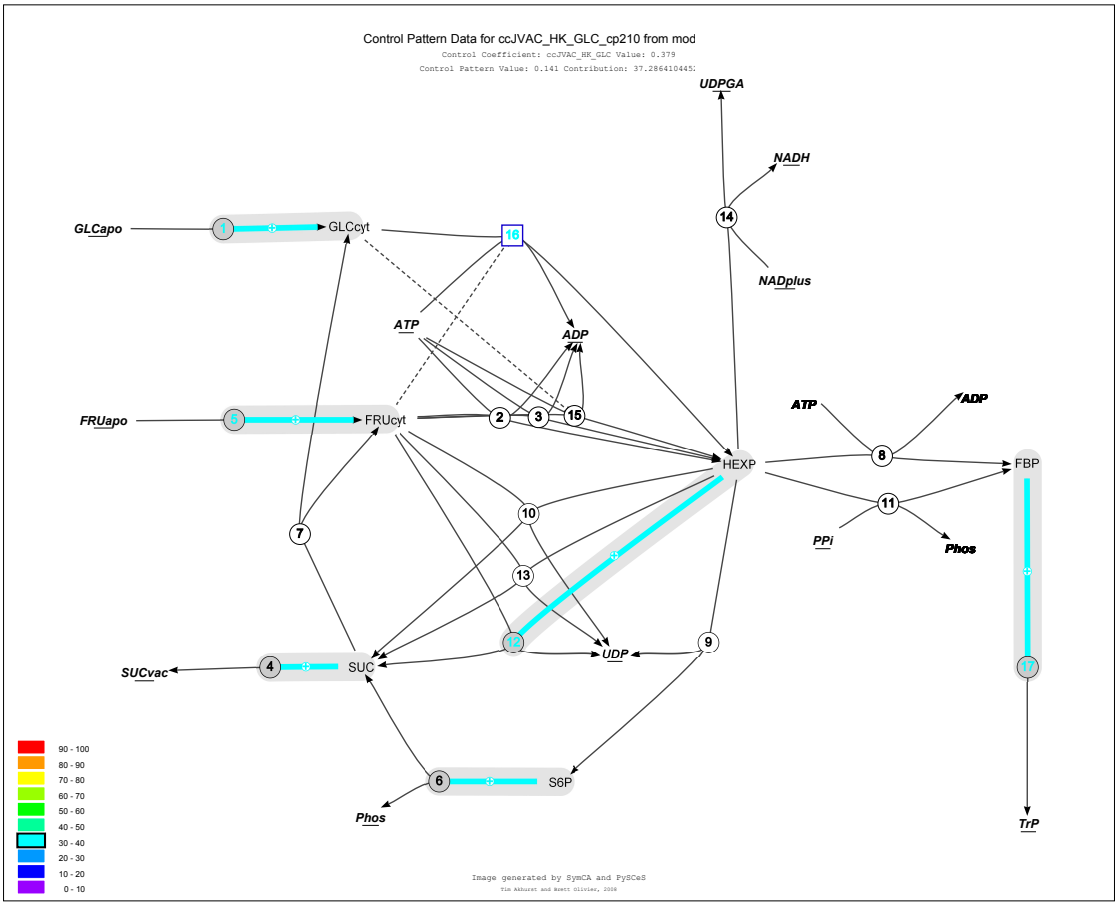

Figure 7.5: Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 6. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines.

Table 7.6: Internode 6 control patterns for $C^{JVAC}_{HK\_GLC}$

ID	Control pattern	% Contribution
6.1a	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}$	7.34
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
6.1b	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSyb}}$	27.13
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
6.1c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSya}}$	17.69
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSya}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
6.1d	$\mathbf{-2.0\ J_{ALD}\ J_{FRU\_uptake}\ J_{GLC\_uptake}\ J_{HK\_GLC}\ J_{SuSyc}}$	**37.29**
	$\boldsymbol{\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}}$	

For C^{JVAC}_{FRKa} we find six major contributing control patterns with the majority mediating their control via the same patterns as seen for $C^{JVAC}_{HK\_GLC}$, and again the pattern includes SuSy C as the major contributor (6.2d). Here we find that there are two patterns which act via NI (6.2b & 6.2f), resulting in diversion from sucrose accumulation and accounting for 16.52% of the control (see Table 7.7).

Table 7.7: Internode 6 control patterns for C^{JVAC}_{FRKa}

ID	Control pattern	% Contribution
6.2a	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSya}}$	17.38
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSya}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
6.2b	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyc}}$	9.56
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
6.2c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}$	7.21
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
6.2d	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSyc}}$	36.63
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	

continued on next page

continued from previous page		
ID	**Control pattern**	**% Contribution**
6.2e	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSyb}}$ $\varepsilon_{FBP}^{ALD}\ \varepsilon_{uptake\_FRUcyt}^{FRU}\ \varepsilon_{uptake\_GLCcyt}^{GLC}\ \varepsilon_{HEXP}^{SuSyb}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	26.66
6.2f	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyb}}$ $\varepsilon_{FBP}^{ALD}\ \varepsilon_{uptake\_FRUcyt}^{FRU}\ \varepsilon_{GLCcyt}^{NI}\ \varepsilon_{HEXP}^{SuSyb}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	6.96

The final contributing control coefficient contains four major control patterns acting via the SuSy isozyme and SPS pathway (6.3a, 6.3d, 6.3e & 6.3f), and the two via the NI pathway (6.3b & 6.3c). As per C_{FRKa}^{JVAC}, the control patterns acting via the NI are the minor contributors with a combined control of 15.72%.

Table 7.8: Internode 6 control patterns for $C_{FRU\_uptake}^{JVAC}$

ID	**Control pattern**	**% Contribution**
6.3a	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFP}}\ J_{\text{SuSyb}}$ $\varepsilon_{FBP}^{ALD}\ \varepsilon_{FRUcyt}^{SuSyb}\ \varepsilon_{uptake\_GLCcyt}^{GLC}\ \varepsilon_{HEXP}^{PFP}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	5.45
6.3b	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyc}}$ $\varepsilon_{FBP}^{ALD}\ \varepsilon_{FRUcyt}^{NI}\ \varepsilon_{uptake\_GLCcyt}^{GLC}\ \varepsilon_{HEXP}^{SuSyc}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	9.10
6.3c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyb}}$ $\varepsilon_{FBP}^{ALD}\ \varepsilon_{FRUcyt}^{NI}\ \varepsilon_{uptake\_GLCcyt}^{GLC}\ \varepsilon_{HEXP}^{SuSyb}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	6.62
6.3d	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}\ J_{\text{SuSyb}}$ $\varepsilon_{FBP}^{ALD}\ \varepsilon_{FRUcyt}^{SuSyb}\ \varepsilon_{uptake\_GLCcyt}^{GLC}\ \varepsilon_{HEXP}^{SPS}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	16.88
6.3e	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}\ J_{\text{SuSyc}}$ $\varepsilon_{FBP}^{ALD}\ \varepsilon_{FRUcyt}^{SuSyc}\ \varepsilon_{uptake\_GLCcyt}^{GLC}\ \varepsilon_{HEXP}^{SPS}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	15.26
6.3f	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}\ J_{\text{SuSya}}$ $\varepsilon_{FBP}^{ALD}\ \varepsilon_{FRUcyt}^{SuSya}\ \varepsilon_{uptake\_GLCcyt}^{GLC}\ \varepsilon_{HEXP}^{SPS}\ \varepsilon_{S6P}^{SPase}\ \varepsilon_{SUC}^{VAC}$	7.90

7.2.5 Internode 7

The data for Internode 7 shows that there are two control coefficients to consider, $C^{JVAC}_{HK\_GLC}$ and C^{JVAC}_{FRKa} with values of 0.502 and 0.294 respectively. For $C^{JVAC}_{HK\_GLC}$, there are four contributing control patterns, with the three largest patterns acting via the SuSy isozyme (7.1b, 7.1c & 7.1d). SuSy B and SuSy C are found to account for most of the control, with the greatest control via SuSy C (7.1d). This pattern can be seen in Figure 7.6 and is in bold text in the Table 7.9.

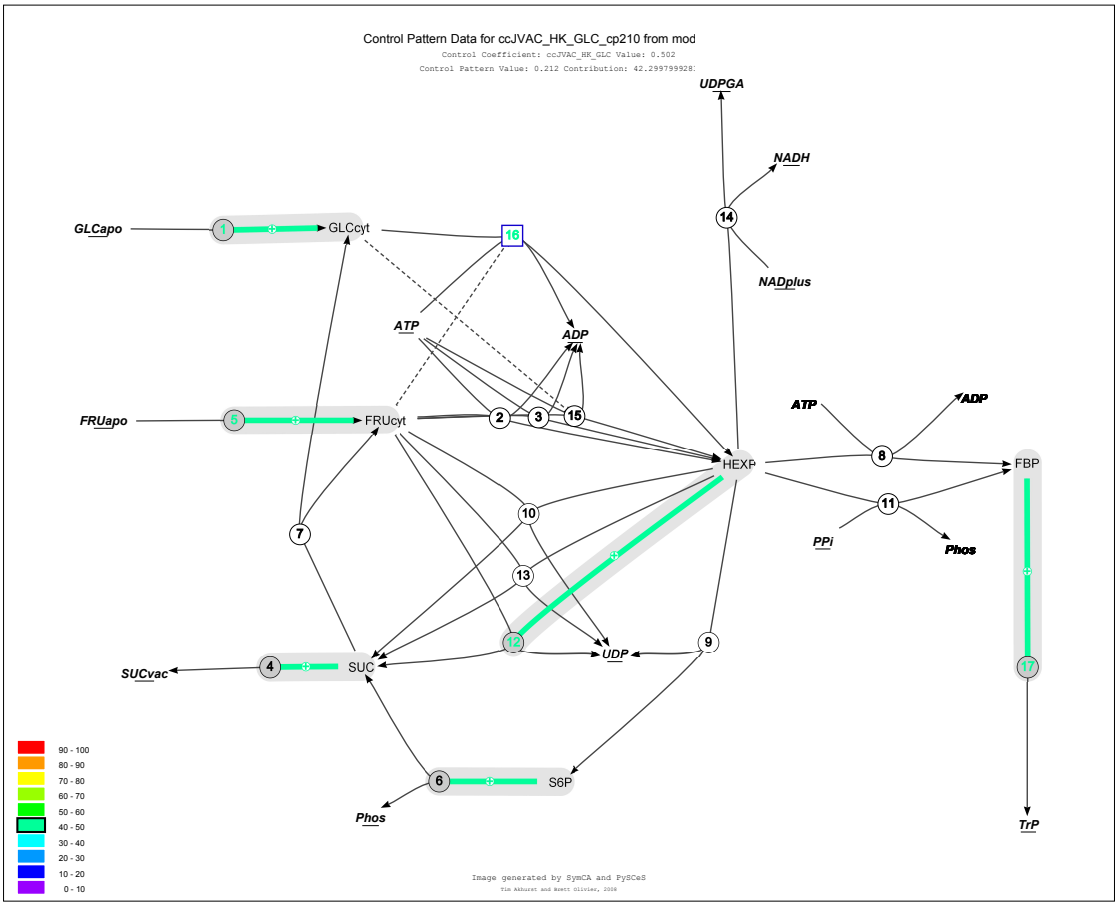

Figure 7.6: Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 7. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines.

Table 7.9: Internode 7 control patterns for $C^{JVAC}_{HK\_GLC}$

ID	Control pattern	% Contribution
7.1a	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}$	5.37
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
7.1b	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSyb}}$	34.72
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
7.1c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSya}}$	11.73
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSya}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
7.1d	$\mathbf{-2.0\ J_{ALD}\ J_{FRU\_uptake}\ J_{GLC\_uptake}\ J_{HK\_GLC}\ J_{SuSyc}}$	**42.30**
	$\boldsymbol{\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}}$	

The control pattern data for C^{JVAC}_{FRKa} is similar to that of $C^{JVAC}_{HK\_GLC}$ with the highest contribution via SuSy C (7.2d). We can also see that a small percentage of the control involves the pathway via NI (7.2b), thus leading to futile cycling.

Table 7.10: Internode 7 control patterns for C^{JVAC}_{FRKa}

ID	Control pattern	% Contribution
7.2a	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSya}}$	11.42
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSya}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
7.2b	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyc}}$	5.89
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
7.2c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}$	5.23
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
7.2d	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSyc}}$	41.18
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
7.2e	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSyb}}$	33.80
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	

7.2.6 Internode 8

The data for Internode 8 shows three control coefficients involved in the control of J_{VAC}, namely $C^{JVAC}_{HK\_GLC}$, $C^{JVAC}_{FRU\_uptake}$ and C^{JVAC}_{FRKa}. For $C^{JVAC}_{HK\_GLC}$ we find two major contributing control patterns which affect the pathway leading to sucrose accumulation by the SuSy isozyme, in particular isozymes B and C, with the control pattern via SuSy C (8.1f, highlighted bold text in Table 7.11) accounting for 40.84% and the other control pattern via SuSy B (8.1c) accounting for 30.63%. The remainder of the control is distributed between the pathways involving HEXP → SuSy A, S6P → SPase and HEXP → SPase.

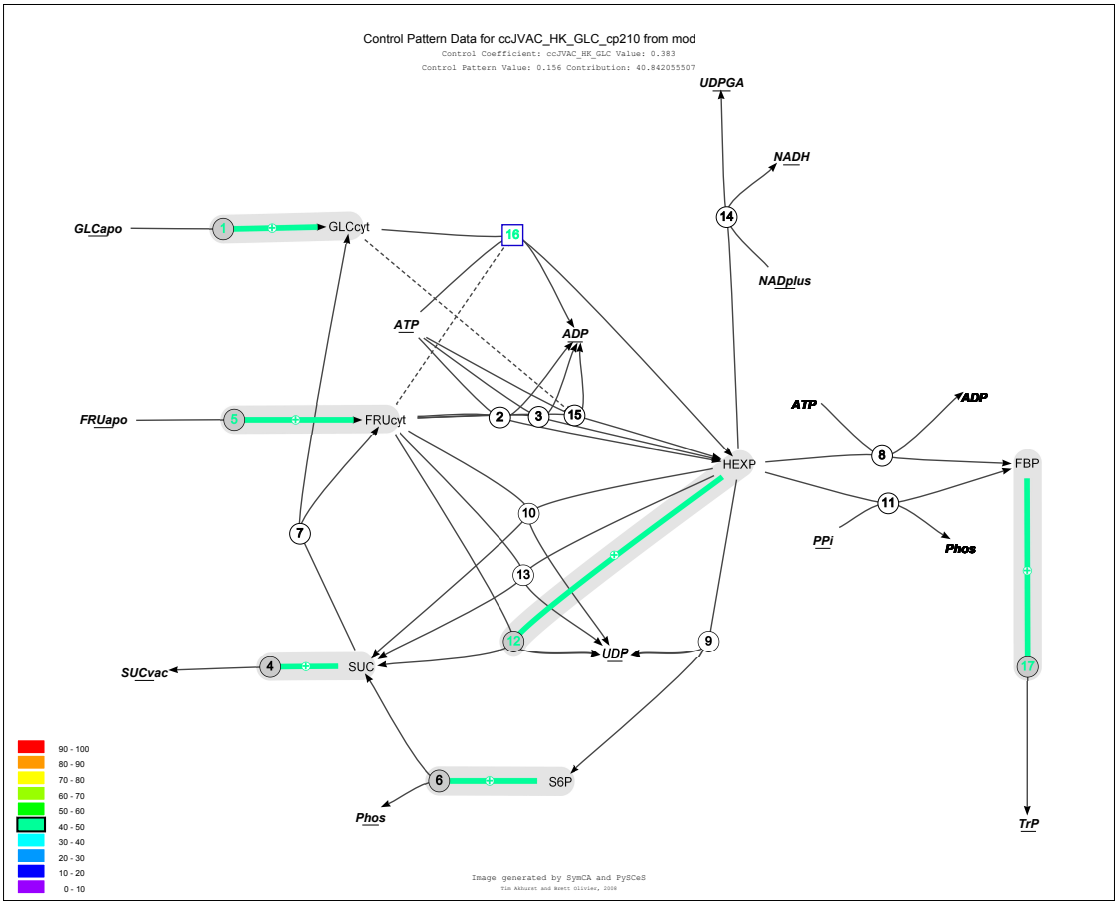

Figure 7.7: Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 8. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines.

Table 7.11: Internode 8 control patterns for $C^{JVAC}_{HK\_GLC}$

ID	Control pattern	% Contribution
8.1a	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}$	9.37
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
8.1b	$+2.0\ J_{\text{ALD}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}\ J_{\text{SuSyb}}$	6.32
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
8.1c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSyb}}$	30.63
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
8.1d	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSya}}$	7.89
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSya}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
8.1e	$+2.0\ J_{\text{ALD}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}\ J_{\text{SuSyc}}$	5.30
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
8.1f	$\mathbf{-2.0\ J_{ALD}\ J_{FRU\_uptake}\ J_{GLC\_uptake}\ J_{HK\_GLC}\ J_{SuSyc}}$	**40.84**
	$\boldsymbol{\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}}$	

When we consider $C^{JVAC}_{FRU\_uptake}$ the major control patterns involve the pathways via SuSy B (8.2a, 8.2f & 8.2h) and SuSy C (8.2c, 8.2d & 8.2g). However, here the major patterns are those concerned with SuSy B. The remaining control patterns include the same pathways as for $C^{JVAC}_{HK\_GLC}$ with the addition of two patterns involving step NI (8.2b & 8.2e), and contribute 14.85%. This is shown in Table 7.12.

Table 7.12: Internode 8 control patterns for $C^{JVAC}_{FRU\_uptake}$

ID	Control pattern	% Contribution
8.2a	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFP}}\ J_{\text{SuSyb}}$	6.36
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFP}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	
8.2b	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyc}}$	8.49
	$\varepsilon^{ALD}_{FBP}\ \varepsilon^{NI}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	

continued on next page

continued from previous page		
ID	**Control pattern**	**% Contribution**
8.2c	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFK}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.24
8.2d	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFP}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFP}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.34
8.2e	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{NI}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.36
8.2f	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	17.70
8.2g	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	14.85
8.2h	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFK}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.25

The third control coefficient for Internode 8 in Table 7.13, C^{JVAC}_{FRKa} for Internode 8 presents a similar mix of control patterns as demonstrated for $C^{JVAC}_{FRU\_uptake}$. However, we can see that the contribution for the control patterns affecting the pathway via step NI (8.3b & 8.3f) increases to a combined total of 18.62%. Of the major contributing control patterns, the two patterns involved in mediating their response by SuSy B (8.3a) and SuSy C (8.3d), are again the dominant ones with 43.25% contributed via SuSy C and 32.44% contributed via SuSy B.

Table 7.13: Internode 8 control patterns for C^{JVAC}_{FRKa}

ID	**Control pattern**	**% Contribution**
8.3a	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSya}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSya}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	8.36
8.3b	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	10.64

continued on next page

continued from previous page		
ID	**Control pattern**	**% Contribution**
8.3c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	9.93
8.3d	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	43.25
8.3e	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	32.44
8.3f	$-2.0\ J_{\text{ALD}}\ J_{\text{FRKa}}\ J_{\text{FRU\_uptake}}\ J_{\text{NI}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	7.98

7.2.7 Internode 9

There are two dominant control coefficients in Internode 9, $C^{JVAC}_{HK\_GLC}$ and $C^{JVAC}_{FRU\_uptake}$ with values of 0.417 and 0.266 respectively. The quantified control patterns for $C^{JVAC}_{HK\_GLC}$ contain only those control patterns which result in sucrose accumulation, with the two dominant patterns acting via SuSy B (9.1c) and SuSy C (9.1e, shown in Figure 7.8 and highlighted in bold text in Table 7.14). Again SuSy C can be seen to be the greatest contributor with 39.28% compared with 30.21% via SuSyb.

Table 7.14: Internode 9 control patterns for $C^{JVAC}_{HK\_GLC}$

ID	**Control pattern**	**% Contribution**
9.1a	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	11.99
9.1b	$+2.0\ J_{\text{ALD}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	9.00
9.1c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	30.21

continued on next page

continued from previous page		
ID	**Control pattern**	**% Contribution**
9.1d	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{HK\_GLC}}\ J_{\mathrm{SPS}}\ J_{\mathrm{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.83
9.1e	$\mathbf{-2.0\ J_{ALD}\ J_{FRU\_uptake}\ J_{GLC\_uptake}\ J_{HK\_GLC}\ J_{SuSyc}}$ $\boldsymbol{\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}}$	**39.28**

Table 7.15: Internode 9 control patterns for $C^{JVAC}_{FRU\_uptake}$

ID	Control pattern	% Contribution
9.2a	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{PFP}}\ J_{\mathrm{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFP}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.84
9.2b	$-2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{NI}}\ J_{\mathrm{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{NI}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	7.82
9.2c	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{PFK}}\ J_{\mathrm{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.26
9.2d	$-2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{NI}}\ J_{\mathrm{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{NI}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.01
9.2e	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{SPS}}\ J_{\mathrm{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	18.16
9.2f	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{SPS}}\ J_{\mathrm{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	13.78
9.2g	$+2.0\ J_{\mathrm{ALD}}\ J_{\mathrm{FRU\_uptake}}\ J_{\mathrm{GLC\_uptake}}\ J_{\mathrm{PFK}}\ J_{\mathrm{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	8.25

The data for $C^{JVAC}_{FRU\_uptake}$ contains the same patterns as found for $C^{JVAC}_{HK\_GLC}$ with the addition of two patterns affecting step NI (9.2b & 9.2d), which represent a combined total of 13.85%. We can also see from Table 7.15 that out of the two dominant control patterns the one acting via SuSy B (9.2e) is now the major contributor with 18.16%, whereas the second highest contributor acts via SuSy C (9.2f) with 13.78%.

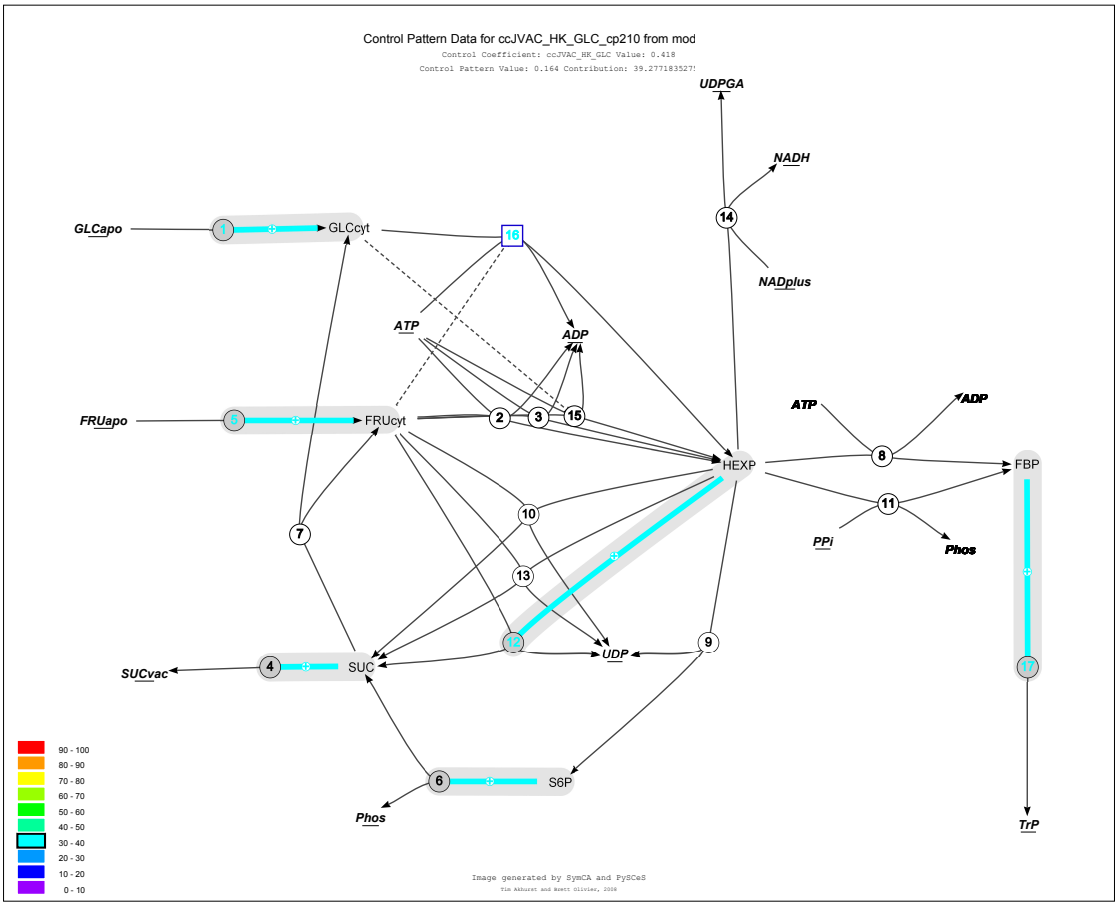

Figure 7.8: Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 9. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines.

7.2.8 Internode 10

For Internode 10 the largest control coefficient is $C^{JVAC}_{FRU\_uptake}$ with a value of 0.362. This control coefficient contains three dominant control patterns, with the highest contributions from the two patterns acting via SuSy B (10.1d & 10.1e), 16.85% and 12.58%. The third pattern involves SuSy C (10.1g) with 12.58%. There are also two control patterns which include the affects on the pathway via step NI (10.1c & 10.1f), with contributions of 6.07% and 5.26% respectively. Control pattern 10.1d exhibits the highest control and is highlighted in bold text in Table 7.16 and shown in Figure 7.9.

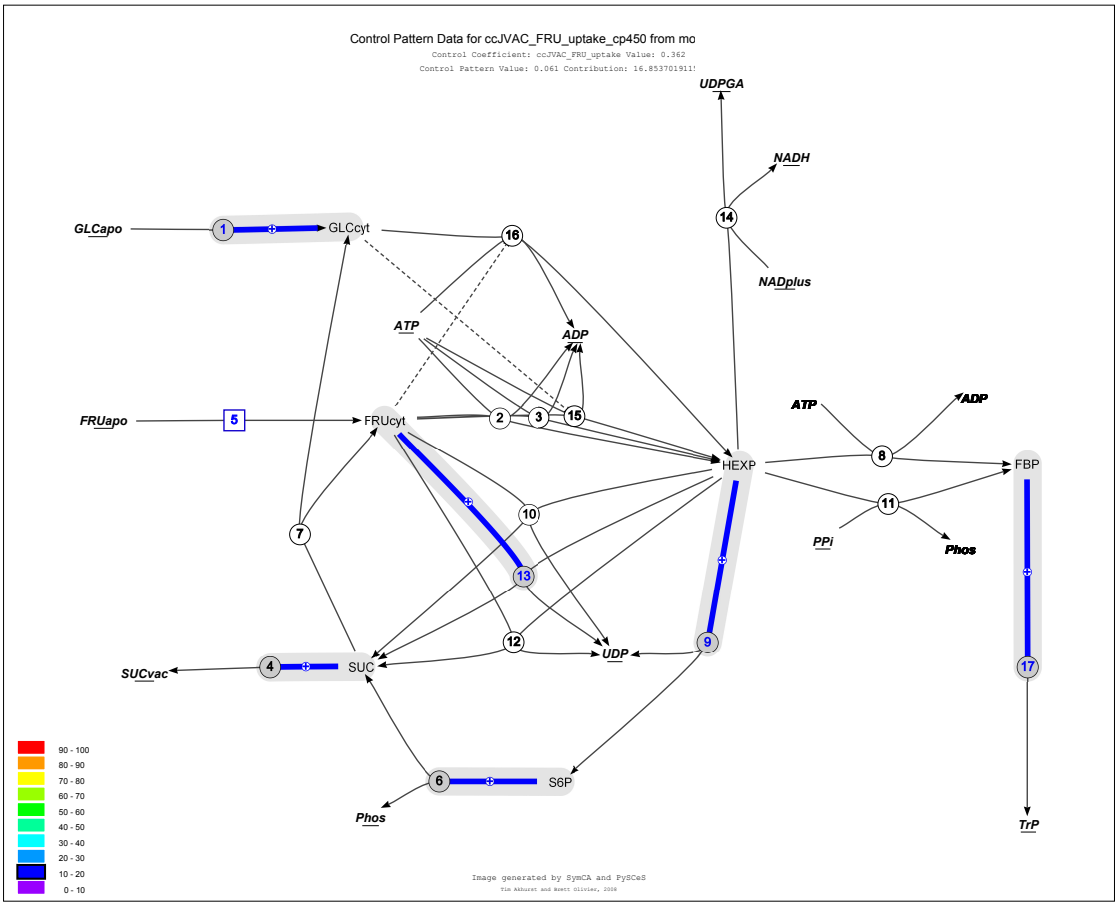

Figure 7.9: Dominant control pattern for $C^{JVAC}_{HK\_GLC}$ in Internode 10. Enzymes are numbered in circles, with the modulated enzyme enclosed in a square. *Number associations* are as in Figure 7.2. Elasticities and fluxes in the control pattern are coloured, based on percentage contribution, and are enclosed in a grey balloon, feedback loops are represented by dashed lines.

The remaining dominant control pattern (10.2a) acts via the HEXP $\rightarrow$ S6P $\rightarrow$ SPase pathway with a contribution of 24.69%. All three patterns ultimately result in sucrose accumulation. Again we find a small percentage of the contribution acting via step NI (10.2d & 10.2g), and diverting from sucrose accumulation as shown in Table 7.17.

$C^{JVAC}_{HK\_GLC}$ has the second-largest value (0.279) for Internode 10, and this control coefficient contains seven major contributing control patterns. Unlike $C^{JVAC}_{FRU\_uptake}$, the major contributing control pattern (10.2f) acts via SuSy C (32.95%) whereas the second highest (10.2c) acts via SuSy B (25.49%).

Table 7.16: Internode 10 control patterns for $C^{JVAC}_{FRU\_uptake}$

ID	Control pattern	% Contribution
10.1a	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFP}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFP}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.48
10.1b	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFK}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	9.39
10.1c	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{NI}}\ J_{\text{SPS}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	5.26
10.1d	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}\ J_{\text{SuSyb}}$ $\varepsilon_{ALD}FBP\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	**16.85**
10.1e	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{SPS}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	12.58
10.1f	$-4.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{NI}}\ J_{\text{SPS}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{NI}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	6.07
10.1g	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{PFK}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	12.58

Table 7.17: Internode 10 control patterns for $C^{JVAC}_{HK\_GLC}$

ID	Control pattern	% Contribution
10.2a	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	24.69
10.2b	$+2.0\ J_{\text{ALD}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyb}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	15.85
10.2c	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSyb}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyb}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	25.49

continued on next page

continued from previous page

ID	Control pattern	% Contribution
10.2d	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{NI}}\ J_{\text{PFK}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{PFK}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	-5.75
10.2e	$+2.0\ J_{\text{ALD}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SPS}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{SuSyc}_{FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	11.83
10.2f	$-2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{GLC\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{SuSyc}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{GLC}_{uptake\_GLCcyt}\ \varepsilon^{SuSyc}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	32.95
10.2g	$+2.0\ J_{\text{ALD}}\ J_{\text{FRU\_uptake}}\ J_{\text{HK\_GLC}}\ J_{\text{NI}}\ J_{\text{SPS}}$ $\varepsilon^{ALD}_{FBP}\ \varepsilon^{FRU}_{uptake\_FRUcyt}\ \varepsilon^{NI}_{GLCcyt}\ \varepsilon^{SPS}_{HEXP}\ \varepsilon^{SPase}_{S6P}\ \varepsilon^{VAC}_{SUC}$	-7.71

7.3 Discussion

The analysis provides quantitative evidence to support the observations of Uys *et al.* [116] that there is a decrease in futile cycling as the internodes mature, which corresponds to an increase in vacuolar sucrose accumulation. We could see this by the diminishing control observed for control patterns acting via NI as the internodes mature. Of further interest is that we observed the suggested role of SuSy C in the accumulation of sucrose, although the quantified control pattern data suggests that SuSy B also plays a role in this phenomenon.

One of the conditions known to influence the accumulation of sucrose is futile cycling, which results in sucrose being broken down by NI to form fructose and glucose. By analysing the control patterns and the underlying kinetics for all the models representing the differing levels of sugar cane internode maturity we can begin to understand why this phenomenon diminishes as the maturity of the internodes increases. For Internodes 3 and 4 we found $C^{JVAC}_{FRU\_uptake}$ to be the dominant control coefficient. The resultant control patterns contained a proportion including NI, namely control patterns 3c, 3d and 3e for Internode 3 contributing a combined 22.92% to the control coefficient and 4c, 4d, 4f and 4g for Internode 4 with a combined contribution of 25.25%. As the internodes mature from 5 to 9, we find that the dominant control coefficient $C^{JVAC}_{HK\_GLC}$, contains no patterns acting via NI, thereby reducing the effects of futile cycling. What is apparent for these internodes is that the additional control coefficients do contain control patterns affecting NI. However, when we calculate the actual contribution of these terms it be-

comes clear that the overall percentage of control acting via NI is in fact decreasing as the internodes mature. This was calculated by summing the percentages of each control pattern acting via NI for the concerned control coefficient and then multiplying this total by the value of the control coefficient to get an actual percentage of control. The details of this can be seen in Table 7.18 below:

Table 7.18: Control pattern contributions for patterns acting via NI.

Control Pattern	**Combined % Contribution**	**Control Coefficient**	**Control Coefficient Value**	**Actual % Contribution**
5.2b, c, f	21.65	C_{FRKa}^{JVAC}	0.279	6.04
6.2b, f	16.52	C_{FRKa}^{JVAC}	0.287	4.74
6.3b, c	15.72	$C_{FRU\_uptake}^{JVAC}$	0.224	3.52
7.2b	5.89	C_{FRKa}^{JVAC}	0.294	1.73
8.2b, e	14.85	$C_{FRU\_uptake}^{JVAC}$	0.249	3.70
8.3b, f	18.62	C_{FRKa}^{JVAC}	0.224	4.17
9.2b, d	13.83	C_{FRKa}^{JVAC}	0.268	3.71

If we now look at the steady-state data for each internode, we can see that the flux for NI decreases (0.052, 0.050, 0.019, 0.017, 0.009, 0.018, 0.024 and 0.027) and that elasticity $\varepsilon_{FRUcyt}^{NI}$ increases (-0.062, -0.070, -0.308, -0.233, -0.356, -0.248, -0.249 and -0.262) as the internodes mature. Note that the elasticity is in itself negative, indicative of a product elasticity, which suggests that with internode maturity increasing so too is the product inhibition of FRUcyt on NI. This may explain why we can observe a decrease in futile cycling as the internodes mature, which also explains why there are still control patterns present including this step, but with a decreased level of control over the step leading to sucrose accumulation.

It is equally important to analyse further the role of the SuSy enzymes in sucrose accumulation. Note that there are three isoforms of SuSy (SuSya, SuSyb and SuSyc) within the system and that Uys *et al.* found that SuSyc was more significant regarding sucrose accumulation. For Internodes 3 and 4, we find that the major control patterns, 3b and 3g for Internode 3 and 4b and 4i for Internode 4, all act via SuSya contributing a combined 31.15% and 19.88%, respectively. The control patterns acting via SuSyb contribute 5.57% (3h) and 11.99% (4e and 4j) for Internodes 3 and 4, and those acting via SuSyc contribute 13.98% (3a and 3f) for Internode 3 and 18.36% (4a and 4h) for Internode 4. The data for the remaining internodes are slightly more complicated due to the control of sucrose accumulation being shared between two or three control coefficients. The data for each isoform is summarised in the following Tables, where the

control pattern identifier refers to the control pattern IDs as illustrated in the quantified control pattern tables for each internode.

Table 7.19: SuSya control pattern contributions for patterns leading to sucrose accumulation.

Control Pattern	**Combined % Contribution**	**Control Coefficient**	**Control Coefficient Value**	**Actual % Contribution**
5.1b	24.33	$C^{JVAC}_{HK\_GLC}$	0.487	11.85
5.2a	22.04	C^{JVAC}_{FRKa}	0.279	6.15
6.1a	17.69	$C^{JVAC}_{HK\_GLC}$	0.379	6.70
6.2a	17.38	$C^{JVAC}_{FRU\_uptake}$	0.287	4.99
6.3f	7.90	C^{JVAC}_{FRKa}	0.224	1.77
7.1c	11.73	$C^{JVAC}_{HK\_GLC}$	0.502	5.89
7.2a	11.42	C^{JVAC}_{FRKa}	0.294	3.36
8.1d	7.89	$C^{JVAC}_{HK\_GLC}$	0.382	3.01
8.3a	8.36	C^{JVAC}_{FRKa}	0.224	1.87

Table 7.20: SuSyb control pattern contributions for patterns leading to sucrose accumulation.

Control Pattern	**Combined % Contribution**	**Control Coefficient**	**Control Coefficient Value**	**Actual % Contribution**
5.1a	29.53	$C^{JVAC}_{HK\_GLC}$	0.487	14.38
5.2e	26.76	C^{JVAC}_{FRKa}	0.279	7.47
6.1b	27.13	$C^{JVAC}_{HK\_GLC}$	0.379	10.28
6.2e	26.66	$C^{JVAC}_{FRU\_uptake}$	0.287	7.65
6.3a, d	22.33	C^{JVAC}_{FRKa}	0.224	5.00
7.1b	34.72	$C^{JVAC}_{HK\_GLC}$	0.502	17.43
7.2b	33.80	C^{JVAC}_{FRKa}	0.294	9.94

continued on next page

continued from previous page				
Control Pattern	**Combined % Contribution**	**Control Coefficient**	**Control Coefficient Value**	**Actual % Contribution**
8.1b, c	36.95	$C^{JVAC}_{HK\_GLC}$	0.382	14.11
8.2a, f, h	30.31	$C^{JVAC}_{FRU\_uptake}$	0.249	7.55
8.3e	32.44	C^{JVAC}_{FRKa}	0.224	7.27
9.1b, c	39.21	$C^{JVAC}_{HK\_GLC}$	0.417	16.35
9.2a, e, g	32.25	$C^{JVAC}_{FRU\_uptake}$	0.268	8.64
10.1a, d, g	34.91	$C^{JVAC}_{FRU\_uptake}$	0.362	12.64
10.2b, c	41.34	$C^{JVAC}_{HK\_GLC}$	0.279	11.53

Table 7.21: SuSyc control pattern contributions for patterns leading to sucrose accumulation.

Control Pattern	**Combined % Contribution**	**Control Coefficient**	**Control Coefficient Value**	**Actual % Contribution**
5.1c	37.33	$C^{JVAC}_{HK\_GLC}$	0.487	18.18
5.2d	33.83	C^{JVAC}_{FRKa}	0.279	9.44
6.1d	37.29	$C^{JVAC}_{HK\_GLC}$	0.379	14.13
6.2d	36.63	C^{JVAC}_{FRKa}	0.287	10.51
6.3e, d	15.26	$C^{JVAC}_{FRU\_uptake}$	0.224	3.42
7.1d	42.30	$C^{JVAC}_{HK\_GLC}$	0.502	21.23
7.2d	41.18	C^{JVAC}_{FRKa}	0.294	12.11
8.1e, f	46.14	$C^{JVAC}_{HK\_GLC}$	0.382	17.63
8.2c, d, g	25.43	$C^{JVAC}_{FRU\_uptake}$	0.249	6.33
8.3d	43.25	C^{JVAC}_{FRkA}	0.224	9.69
9.1d, e	46.11	$C^{JVAC}_{HK\_GLC}$	0.417	19.23
9.2c, f	20.04	C^{JVAC}_{FRKa}	0.268	5.37
continued on next page				

continued from previous page

Control Pattern	**Combined % Contribution**	**Control Coefficient**	**Control Coefficient Value**	**Actual % Contribution**
10.1b, e	21.97	C^{JVAC}_{FRKa}	0.362	7.95
10.2e, f	44.78	$C^{JVAC}_{HK\_GLC}$	0.279	12.50

By summing all the actual control pattern % contributions for each SuSy isoform and for each internode (see Table 7.22), we can see that as the internodes increase in maturity so does the contribution of the control patterns acting via SuSya. By contrast, the control patterns acting via both SuSyb and SuSyc, are fairly consistent in their contributions to the overall control of sucrose accumulation. The contributions via SuSyc are generally greater than those of SuSyb, although both appear to be important with regards to sucrose accumulation. The main cause of this observation lies in the flux through each isoform. The flux though SuSya decreases with an increase in internode maturity (0.088, 0.063, 0.031, 0.023, 0.013, 0.010, 0.005 and 0.000), whereas the flux for both SuSyb and SuSyc increases as the internodes mature, with values of 0.024, 0.038, 0.046, 0.049, 0.047, 0.056, 0.066 and 0.060 for J_{SuSyb}, and 0.049, 0.072, 0.060, 0.068, 0.060, 0.075, 0.086 and 0.078 for J_{SuSyc}. This would explain why the control involving SuSya disappears at more mature internodes and why SuSyb and SuSyc are always present, possibly aiding the accumulation of sucrose as the internodes mature.

Table 7.22: Combined actual % contributions for all control patterns including the SuSy isoforms.

Internode	**SuSya Combined % Contribution**	**SuSyb Combined % Contribution**	**SuSyc Combined % Contribution**
3	30.81	5.51	13.92
4	16.3	9.83	15.10
5	18.00	21.85	27.62
6	13.46	22.93	28.06
7	9.25	27.37	33.34
8	4.88	28.93	33.65
9	-	24.99	24.60
10	-	24.17	20.45

It would be beneficial to perform additional theoretical and laboratory experiments to further elucidate the role played by the SuSy isozymes. This would provide further information regarding sucrose accumulation as this study has highlighted the roles of both SuSyb and SuSyc, whereas Uys *et al.* found that SuSyc was more significant regarding sucrose accumulation.

In addition, it would be advantageous to perform theoretical 'knock out' experiments with substrate elasticity ε_{Suc}^{NI} and thereby eliminate futile cycling. This should result in an increase in sucrose accumulation, but may lead to further implications which could be illustrated by means of control pattern quantification. Rossouw *et al.* [93], have described similar experiments which may give additional insight when combined with symbolic control analysis.

The final chapter gives an overview on what we have achieved in this study, whilst presenting its merits in terms of contribution to the field of Systems Biology.

Chapter 8

Discussion

8.1 Synopsis

The central focus of this thesis addressed a perceived shortcoming in conventional MCA techniques because all available softwares offer only the numerical solution for any given system. The primary objective of this study was therefore to provide an algebraic solution for systems, with an emphasis on its uses in understanding the regulatory behaviour of cellular systems. This thesis describes the development of `SymCA`, a software implementation of the symbolic matrix inversion of MCA, in a generalised way and leads to the generation of analytical expressions for the control coefficients of a pathway in terms of the elasticities. From a systems biological perspective this provides a number of uses:

- The ever-increasing use of computational analysis of biochemical pathways in the expanding field of Computational Systems Biology has led to the number of available models growing monthly. This is evident on an inspection of on-line model repositories such as JWS Online ([85], `http://jjj.biochem.sun.ac.za`) or BioModels ([77], `http://www.ebi.ac.uk/biomodels`), which show an increase in the number of models, and also in the size and complexity of the models. Thus, MCA is becoming increasingly important in elucidating where the control in a pathway lies and in identifying the factors determining this control. With particular reference to `SymCA`, we can use the expressions generated to determine key elasticities responsible for a large or small control coefficient value. We can also evaluate which parameters have the largest effect on an observed behaviour, and more importantly how this effect is transmitted in the system.

- The analysis of the three test cases, i.e. a theoretical model with a feedback mechanism, the fermentation pathways in *Saccharomyces cerevisiae* [41] and sugar accumulation in sugar cane [116]), show how we can use the generated expressions to explain and quantify observed behaviours.

- MCA expressions can become unwieldy as the model size increases beyond a few reactions as demonstrated in the control coefficients generated for the fermentation pathways in *Saccharomyces cerevisiae* (see Chapter 6). In these cases we

can introduce biological assumptions resulting in simplified expressions. Common substitutions include situations where we know that a particular reaction is either insensitive to a product or saturated with substrate under cellular conditions, so here we may set the elasticity to zero. Alternatively, if a reaction is operating in the first-order range because the K_{m} is higher than the substrate concentration, we can set the elasticity to one. We may also investigate how the control distribution within a network is affected by setting the value of certain elasticities to vary within bounds. The use of symbolic MCA is integral to these types of analysis.

- A common hindrance in the development of kinetic models is not knowing all the kinetic parameters. An important and significant aspect of symbolic MCA is that it is valid in general, thus it is not dependent on the availability of particular parameter values. The only requirement is the description of allosteric modifier interactions and the stoichiometry. Thus even if we do not know all the kinetic parameter details, we can infer general conclusions about the control structure of a pathway.

- The technique of control pattern analysis, as demonstrated by Hofmeyr in 1989 [49] whereby the individual terms of a control coefficient expression can be visualised on a network, can be achieved by symbolic control analysis. The importance of this is due to these 'control patterns' essentially describing a 'chain of local events' which corresponds to a particular route of regulation. Symbolic control analysis can aid the identification of these routes of regulation in complex networks and quantify their relative importance.

- As highlighted previously the usefulness of symbolic MCA when applied to systems with little or no kinetic data, can still allow us to gain insights into the control structures present in a system. When we know the full kinetic data and parameter values, we can cross-check the generated symbolic expressions by substituting steady-state data into the expressions and evaluating them for comparison against data generated from a purely numeric analysis, as provided by `PySCeS`. The combination of both numeric and symbolic analysis also provides the useful technique of quantifying control pattern contributions for a specific steady-state.

8.2 Critical appraisal

MCA has proved to be a key technique for the quantification of biological systems, and has helped to displace the notion that for every system there is a single 'rate-limiting' step. MCA has led to our current understanding of control within cellular systems that the control can be shared between multiple steps. A number of key concepts within MCA have enabled the development of a generic symbolic approach to MCA, in particular the control-matrix equation, $\mathbf{C^i} = \mathbf{E^{-1}}$, described by Hofmeyr in [51]. This equation arose from a combination of the Summation and Connectivity Theorems, and Hofmeyr [51] regards it as the most powerful feature of MCA. The equation provides a means

of expressing the control coefficients for a system in terms of the elasticity coefficients. In essence, all that we require to solve this equation is a symbolic matrix of all local and structural system properties, **E**. Once this matrix has been formed, we need only invert the matrix to generate the control coefficients expressed in terms of the elasticity coefficients.

It is important to note that this study was not the first symbolic MCA implementation. This was first pioneered by Schulz in 1991 [101] and then by Thomas and Fell in 1993 [114]. Thomas and Fell developed `MetaCon` which represents an automation of the matrix method described in [40, 99, 110]. Application of this matrix method resulted in the following matrix method:

$$\mathbf{EC} = \mathbf{M} \tag{8.1}$$

where **E** represents the elasticity matrix, **C** the control-matrix (the control coefficients to be evaluated), and **M** the matrix describing the relationships between the elements of **E** and **C**. The aim of `MetaCon` was to solve the elements of **C**, which was achieved by inverting the elasticity matrix:

$$\mathbf{C} = \mathbf{E^{-1}M} \tag{8.2}$$

A key step in this method for generating control coefficients is the selection of a *reference flux* from the pathway. Thus they calculated only the flux control coefficients for the selected flux (although `MetaCon` provided the option to calculate the control coefficients of all steps on request by the user). The method described in [53, 55] represents a more generic solution to this problem in that there is no requirement to select a *reference flux*. Hofmeyr [51] presented a variation of the control-matrix, $\mathbf{C^i} = \mathbf{E^{-1}}$, which relies on the description of the stoichiometry and any allosteric modifier interactions. Thus this method is generic for all cellular systems. At the time of `MetaCon's` development this method had not been presented, whereas the method described in this thesis and in [51] provides a generic solution to this problem. These perceived shortcomings of the matrix method [40, 99, 110] used by Thomas and Fell in their development of Metacon [114], has led to the implementation of `SymCA` in presenting a generic symbolic control analysis tool.

A further limitation to previous attempts of symbolic MCA implementations was insufficient computing power which is necessary for solving symbolic computations. From the beginning symbolic computing was designed for use on supercomputers. The computing landscape has changed significantly in the past decade and we are now in the fortunate position in which the power available in 'off-the-shelf' computers allows us to perform these tasks.

There are known common issues inherent in symbolic computations, and the most pertinent to this study is the inefficiency in computing the inverse of higher order symbolic matrices. It would therefore be a useful exercise to try to find more efficient methods and softwares which can alleviate this shortcoming as this would allow the analysis of even larger systems. The computer algebra system used in this study, `Maxima`, presents

its own unique set of challenges in that it is a third party software which requires a `Python` interface for its incorporation into `SymCA`. Data generated by `Maxima` must be extracted for further computations, and this can be extremely inefficient as the size of the system under investigation increases.

Of key importance is the size of the model under investigation. In this study the largest system symbolically analysed consisted of 17 reactions (See Chapter 7). This system is fast approaching the size limitations for symbolic analysis, mainly due to inefficiencies in current symbolic matrix inversion routines for high order matrices [8]. As the size of the matrix exceeds $\approx 25 \times 25$, this becomes more apparent. Additionally, the complexity of the system in terms of 'interlinkedness', by way of moieties such ATP/ADP and NADH/NAD which can act on numerous places, adds to the complexity of resultant expressions. We can possibly simplify such systems by clamping the moieties, but this results in the loss of the contribution of the moieties to the control structure of the system. If we know *a priori*, that certain reactions are saturated with substrate, first-order in substrate or irreversible, we can make elasticity substitutions before analysis thereby reducing the complexity of the expressions. These considerations are particularly important for larger systems, but smaller systems are handled efficiently without the need for any simplifications. Although we can analyse smaller systems either by hand or via graphical means [49], the efficiency and additional post-analysis routines provided by `SymCA`, suggests that `SymCA` is well suited for both small and larger models when the symbolic analysis of a system is desired.

The work covered in Chapters 5, 6 & 7 would not have been possible before the development of `SymCA`. These studies have shown how concepts such as control pattern identification introduced by Hofmeyr [49] can be extended to quantify the contribution of each control pattern found within a control coefficient expression. We can then use this data to isolate the key routes of regulation under the experimental conditions in question. This technique provided additional insight into Galazzo and Bailey's work [41] in which they saw pathways activated under cell immobilisation to result in an increase in polysaccharide formation.

This technique was applied again to extend the work of Uys *et al.* [116]. Hence the control patterns were quantified for Internodes 3-10 to provide additional information about the reason for an increase in sucrose accumulation in sugar cane as the plant matures. When used in combination with supply-demand analysis [50] and parameter scans, we can understand how the regulatory pathways vary under changing conditions. Coupled with this are the additional features which have been developed that allow the generation of symbolic control coefficients as well as the visualisations of data generated by `SymCA` by way of SVG images. These features enable us to see which pathways are key for a given control coefficient. There are naturally further possible improvements to `SymCA` and its functionality and we will now discuss future work which can follow this study.

8.3 Future work and perspectives

Since the development of `SymCA`, there have been advances in pure `Python` symbolic libraries which may provide the solution to this problem. SympyCore was developed to address the shortcomings of the SymPy library, with the specific objective to provide an efficient means of computing symbolic calculations. Thus it would be logical to investigate the use of SympyCore as a potential replacement of `Maxima` within `SymCA`. This could be advantageous in a number of ways, such as providing a more efficient matrix inversion algorithm and negating the need to write large volumes of data to disk to make it accessible to `SymCA`.

There are many additional features which should be addressed, such as the ability to perform multi-dimensional parameter scanning with control pattern quantification as opposed to the currently handled combination of single dimension parameter scans and control pattern quantification. Singh and Ghosh [109] demonstrated this in their research which focussed on targeting persistent *tubercule bacilli* to develop anti-tuberculous drugs. They found that the inactivation of isocitrate dehydrogenase 1 and isocitrate dehydrogenase 2 is a potential target for drugs against persistent mycobacteria. If we could use `SymCA` in these studies it may lead to a greater understanding of the regulation within the system, and potentially improve the success rates for identifying novel drug targets.

Since it is often difficult to determine all parameter values for a system, a technique of determining the key parameters would be advantageous. As it stands, the only requirements of `SymCA` are knowledge of the local and structural features of a system to generate the symbolic control coefficient. By developing methods using the response coefficients [52] which are currently calculated, we could determine the importance of the parameters within the system. If we were to link this data with quantified control coefficient data, we could identify exactly how the affect of a parameter is expressed within the system. In addition we have the knowledge that the elasticity coefficients of an enzyme-catalysed reaction are the sum of a regulatory kinetic and a mass-action term [50]. It would be beneficial if these two terms could be distinguished as part of a symbolic analysis as this would prove to be a key feature to aid the quantification of regulation.

Additional work using theoretical models such as showing how control, response and co-response depends on specific elasticities, i.e. positive feedbacks, negative feedbacks and product insensitivities, would also help to further illustrate the power of symbolic computation.

Using experimentally determined steady-state concentrations and rate equations for enzymes in potato tuber glycolysis, Thomas *et al.* [115] were able to calculate most of the elasticity coefficients for this particular state of the metabolic system. They substituted these calculated values for elasticities into the symbolic expressions for control

coefficients obtained with `MetaCon` [114], and replaced the remaining elasticity coefficients with estimated values. They were then able to calculate how sensitive the control coefficients were to variation in the value of a particular elasticity, so allowing them to investigate the variation of important flux-control coefficients with critical elasticities. Although it is already possible to write a Python script that would perform such calculations using the SymCA routines, it would be highly desirable to add in future this functionality to SymCA as an automated procedure.

Thomas *et al.* also utilised a technique of sensitivity analysis in conjunction with the expressions from their software package, `MetaCon`. By incorporating a similar technique, whereby the sensitivity of the control pattern contributions against parameter uncertainties could be deduced, this may provide a summary of which elasticities are important and which are not. This could be achieved by sampling the elasticities and then calculating the control coefficients and the underlying control patterns. This type of analysis could prove to be extremely insightful when used with `SymCA`.

Historically many attempts to increase the yield of biotechnological processes have been at best semi-empirical [72]. In these processes, metabolic or protein engineering is often used to improve biological systems for industrial applications. A key aspect in effectively altering a biological system is understanding the inner workings of the system in question [76]. In the case of fermentation the microorganisms exhibit regulatory mechanisms controlling the production of metabolites. This protects them against overproduction and excretion of primary and secondary metabolites into the environment.

The quest for microbiologists in the field of industrial fermentation is to find a rare overproducing strain in nature. The microorganism is then deregulated so that it overproduces huge quantities of a desired commercially important product such as a metabolite or an enzyme [94]. This has also been highlighted in [113], where Stephanopoulos describes the construction of novel pathways. These metabolic engineering approaches examine biochemical reactions as a system, with the goal being the construction of strains with both a superior productivity and yield.

By using `SymCA` we gain a better understanding into the underlying mechanisms which are producing observed experimental outcomes. By harnessing the power inherent in the control patterns present in all control coefficients, we can begin to design systems to use desirable pathways in a system, and in so doing aid these metabolic or protein engineering approaches. By performing theoretical experiments where the values of elasticity coefficients are varied over a range, we can also design systems to behave in a desirable fashion. Both these features are presently available in `SymCA`. This insight may have significant implications in biotechnological applications as well as in devising rational strategies for target selection in the design of gene therapies or screening candidate drugs [75].

8.4 Conclusion

In this thesis we have presented `SymCA`, a general implementation of the control-matrix equation enabling the generation of control coefficients expressed in terms of elasticities. We achieved this by combining the symbolic computing capabilities of `Maxima` and the programming language of `Python`. We developed a `Python` interface to `Maxima` which allowed access to its symbolic power within `PySCeS` [84]. The end-result is an additional module to use with `PySCeS`, which allowed us to perform symbolic control analysis for cellular systems of any size and complexity, subject to computational constraints. The algebraic expressions generated have been factorised for easy interpretation, and allow an in-depth analysis of where the control lies within the system.

As the field of computational systems grows, so do the number and complexity of models. The net result is that there is an increasing reliance on analysis tools for making sense of the vast amount of model data. The tool of symbolic control analysis provided by `SymCA` contributes to this analysis.

I would like to conclude this discussion with the following quote by R. W. Hammings, which I believe succinctly summarises the motivation behind `SymCA`:

"The purpose of computing is insight, not numbers."
R. W. Hamming (1915–1988)

Bibliography

[1] Axiom. *http://www.axiom-developer.org/*, 9 May 2009.

[2] Computer algebra systems. *http://www.knowledgerush.com/kr/encyclopedia/Computer-algebra-system/*, 10 May 2009.

[3] Computer algebra systems. *http://www.onpedia.com/encyclopedia/Computer-Algebra-System*, 10 May 2009.

[4] Macsyma. *http://www.mathstat.com.au/products/mathematics/macsyma.htm*, 11 February 2009.

[5] Maple. *http://www.maplesoft.com/Products/Maple/*, 11 February 2009.

[6] Mathematica. *http://www.wolfram.com/products/mathematica/index.html*, 11 February 2009.

[7] Maxima. *http://maxima.sourceforge.net*, 11 February 2009.

[8] Maxima manual. *http://maxima.sourceforge.net/docs/manual/en/maxima.html*, 1 June 2009.

[9] Minimal maxima. *http://maxima.sourceforge.net/docs/tutorial/en/minimal-maxima.pdf*, 10 May 2009.

[10] Sage. *http://www.sagemath.org/*, 11 February 2009.

[11] C. Auffray, S. Imbeaud, M. Roux-Rouquié, and L. Hood. From functional genomics to systems biology: concepts and practices. *Comptes Rendus Biologies*, 326:879–892, 2003.

[12] T. Bah. Inkscape: Guide to a vector drawing program. *http://tavmjong.free.fr/INKSCAPE/MANUAL/html/index.php*, 1 June 2009.

[13] B. M. Bakker, P. A. M. Michels, F. R. Opperdoes, and H. V. Westerhoff. Glycolysis in bloodstream form trypanosoma brucei can be understood in terms of the kinetics of the glycolytic enzymes. *Journal of Biological Chemistry*, 272:3207–3215, 1997.

[14] D. M. Beazley. Using swig to control, prototype, and debug c programs with python. In *4th International Python Conference*, 1996.

[15] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. Genbank. *Nucleic Acids Research*, 36:D25–30, 2008.

[16] J. M. Berg, J. L. Tymoczko, and L. Stryer. *Biochemistry.* Freeman, New York, 5th edition, 2002.

[17] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[18] M. Bier, B. Teusink, B. N. Kholodenko, and H. V. Westerhoff. Control analysis of glycolytic oscillations. *Biophysical Chemistry*, 62:15–24, 1996.

[19] B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic Acids Research*, 31:365–370, 2003.

[20] M. D. Brand. Top down metabolic control analysis. *Journal of Theoretical Biology*, 182:351–360, 1996.

[21] G. C. Brown, R. P. Hafner, and M. D. Brand. A 'top-down' approach to the determination of control coefficients in metabolic control theory. *European Journal of Biochemistry*, 188:321–325, 1990.

[22] F. J. Bruggeman. *Of Molecules and Cells: emergent mechanisms.* PhD thesis, Vrije Universiteit, 2005.

[23] F. J. Bruggeman, H. V. Westerhoff, J. B. Hoek, and B. N. Kholodenko. Modular response analysis of cellular regulatory networks. *Journal of Theoretical Biology*, 218:507–520, 2002.

[24] E. Buchner. Alcoholic fermentation without yeast cells. (translation of buchner (1897) by h.c. friedmann.) in: Cornish-bowden, a. (ed.). *New Beer in an Old Bottle: Eduard Buchner and the Growth of Biochemical Knowledge*, pages 25–31, 1997.

[25] J. Calmet. Computer algebra and artificial intelligence. *Mathematics and Computers in Simulation*, 45:73–82, 1998.

[26] M. K. Campbell and S. O. Farrell. *Biochemistry.* Brooks/Cole, 4th edition, 2002.

[27] M. Cascante, R. Franco, and E. I. Canela. Use of implicit methods from general sensitivity theory to develop a systematic approach to metabolic control. i. unbranched pathways. *Mathematical Biosciences*, 94:271–288, 1989.

[28] M. Cascante, R. Franco, and E. I. Canela. Use of implicit methods from general sensitivity theory to develop a systematic approach to metabolic control. ii. complex systems. *Mathematical Biosciences*, 94:289–309, 1989.

[29] A. Cornish-Bowden. Two centuries of catalysis. *Journal of Bioscience*, 23:87–92, 1998.

[30] A. Cornish-Bowden. *Fundamentals of Enzyme Kinetics*. Portland Press, 2004.

[31] A. Cornish-Bowden and M. L. Cárdenas. Self-organization at the origin of life. *Journal of Theoretical Biology*, 252:411–418, 2008. In Memory of Reinhart Heinrich.

[32] A. Cornish-Bowden, M. L. Cárdenas, J.-C. Letelier, J. Soto-Andrade, and F. G. Abarzua. Understanding the parts in terms of the whole. *Biology of the Cell*, 96:713–717, 2004.

[33] R. I. Cukier, H. B. Levine, and K. E. Shuler. Nonlinear sensitivity analysis of multiparameter model systems. *Journal of Computational Physics*, 26:2365–2366, 1978.

[34] A. de la Fuente, P. Brazhnik, and P. Mendes. Linking the genes: inferring quantitative gene networks from microarray data. *Trends in Genetics*, 18:395–398, 2002.

[35] A. Deckard, F. T. Bergmann, and H. M. Sauro. Supporting the sbml layout extension. *Bioinformatics (Oxford, England)*, 22:2966–2967, 2006.

[36] O. V. Demin, H. V. Westerhoff, and B. N. Kholodenko. Control analysis of stationary forced oscillations. *The Journal of Physical Chemistry*, 103:10695–10710, 1999.

[37] D. A. Fell. Metabolic control analysis: a survey of its theoretical and experimental development. *The Biochemical Journal*, 286:313–330, 1992.

[38] D. A. Fell. Systems properties of metabolic networks. In *Proceedings from the international conference on complex systems on Unifying themes in complex systems*, pages 165–177, Nashua, 1997. Perseus Books.

[39] D. A. Fell. *Understanding the Control of Metabolism*. Portland Press, 1997.

[40] D. A. Fell and H. M. Sauro. Metabolic control and its analysis. additional relationships between elasticities and control coefficients. *European Journal of Biochemistry / FEBS*, 148:555–561, 1985.

[41] J. L. Galazzo and J. E. Bailey. Fermentation pathway kinetics and metabolic flux control in suspended and immobilized saccharomyces cerevisiae. *Enzyme and Microbial Technology*, 12:162–172, 1990.

[42] D. Garfinkel and B. Hess. Metabolic control mechanisms. vii. a detailed computer model of the glycolytic pathway in ascites cells. *The Journal of Biological Chemistry*, 239:971–983, 1964.

[43] H. Gest. Landmark discoveries in the trail from chemistry to cellular biochemistry, with particular reference to mileposts in research on bioenergetics. *Biochemical and Molecular Biology Education*, 30:9–13, 2002.

[44] C. Giersch. Control analysis of metabolic networks. 1. homogeneous functions and the summation theorems for control coefficients. *European Journal of Biochemistry / FEBS*, 174:509–513, 1988.

[45] C. Giersch. Control analysis of metabolic networks. 2. total differentials and general formulation of the connectivity relations. *European Journal of Biochemistry / FEBS*, 174:515–519, 1988.

[46] A. C. Hearns. Reduce: The first forty years. *http://reduce-algebra.com/reduce40.pdf*, 11 February 2009.

[47] R. Heinrich and T. A. Rapoport. A linear steady-state treatment of enzymatic chains. critique of the crossover theorem and a general procedure to identify interaction sites with an effector. *European Journal of Biochemistry / FEBS*, 42:97–105, 1974.

[48] R. Heinrich and T. A. Rapoport. A linear steady-state treatment of enzymatic chains. general properties, control and effector strength. *European Journal of Biochemistry / FEBS*, 42:89–95, 1974.

[49] J.-H. S. Hofmeyr. Control-pattern analysis of metabolic pathways. flux and concentration control in linear pathways. *European Journal of Biochemistry / FEBS*, 186:343–354, 1989.

[50] J.-H. S. Hofmeyr. Metabolic regulation: a control analytic perspective. *Journal of Bioenergetics and Biomembranes*, 27:479–490, 1995.

[51] J.-H. S. Hofmeyr. Metabolic control analysis in a nutshell. In T. M. Yi, M. Hucka, M. Morohashi, and H. Kitano, editors, *Proceedings of the 2nd International Conference on Systems Biology*, pages 291–300. Omnipress, 2001.

[52] J.-H. S. Hofmeyr and A. Cornish-Bowden. Quantitative assessment of regulation in metabolic systems. *European Journal of Biochemistry / FEBS*, 200:223–236, 1991.

[53] J.-H. S. Hofmeyr and A. Cornish-Bowden. Co-response analysis: A new experimental strategy for metabolic control analysis*1. *Journal of Theoretical Biology*, 182:371–380, 1996.

[54] J.-H. S. Hofmeyr and A. Cornish-Bowden. Regulating the cellular economy of supply and demand. *FEBS Letters*, 476:47–51, 2000.

[55] J.-H. S. Hofmeyr, A. Cornish-Bowden, and J. M. Rohwer. Taking enzyme kinetics out of control; putting control into regulation. *European Journal of Biochemistry / FEBS*, 212:833–837, 1993.

[56] J.-H. S. Hofmeyr, H. Kacser, and K. J. van der Merwe. Metabolic control analysis of moiety-conserved cycles. *European Journal of Biochemistry / FEBS*, 155:631–641, 1986.

[57] J.-H. S. Hofmeyr and K. J. van der Merwe. Metamod: software for steady-state modelling and control analysis of metabolic pathways on the bbc microcomputer. *Computer Applications In The Biosciences*, 2:243–249, 1986.

[58] J.-H. S. Hofmeyr and H. V. Westerhoff. Building the cellular puzzle: Control in multi-level reaction networks. *Journal of Theoretical Biology*, 208:261–285, 2001.

[59] L. Hood. Systems biology: integrating technology, biology, and computation. *Mechanisms of Ageing and Development*, 124:9–16, 2003.

[60] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. Copasi–a complex pathway simulator. *Bioinformatics (Oxford, England)*, 22:3067–3074, 2006.

[61] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, the rest of the SBML Forum:, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19:524–531, 2003.

[62] T. Ideker, T. Galitski, and L. Hood. A new approach to decoding life: systems biology. *Annual Review of Genomics and Human Genetics*, 2:343–372, 2001.

[63] A. Iglesias. Special section: Computer algebra systems and their applications, casa'2003-06: Selected papers. *Future Generation Computer Systems*, 23:714–715, 2007.

[64] R. B. Jones. Symbolic mathematical computation. *http://www.rbjones.com/rbjpub/cs/ai029.htm*, 11 February 2009, 2000.

[65] H. Kacser and J. A. Burns. The control of flux. *Symposia of the Society for Experimental Biology*, 27:65–104, 1973.

[66] H. Kacser, H. M. Sauro, and L. Acerenza. Enzyme-enzyme interactions and control analysis. 1. the case of non-additivity: monomer-oligomer associations. *European Journal of Biochemistry / FEBS*, 187:481–491, 1990.

[67] D. Kahn and H. V. Westerhoff. Control theory of regulatory cascades. *Journal of Theoretical Biology*, 153:255–285, 1991.

[68] D. Kahn and H. V. Westerhoff. The regulatory strength: How to be precise about regulation and homeostasis. *Acta Biotheoretica*, 41:85–96, 1993.

[69] M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, and Y. Yamanishi. Kegg for linking genomes to life and the environment. *Nucleic Acids Research*, 36:D480–484, 2008.

[70] M. Kanehisa and S. Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28:27–30, 2000.

[71] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Research*, 34:D354–357, 2006.

[72] D. B. Kell and H. V. Westerhoff. Towards a rational approach to the optimization of flux in microbial biotransformations. *Trends in Biotechnology*, 4:137–142, 1986.

[73] B. N. Kholodenko, J. B. Hoek, H. V. Westerhoff, and G. C. Brown. Quantification of information transfer via cellular signal transduction pathways. *FEBS Letters*, 414:430–434, 1997.

[74] H. Kitano. Computational systems biology. *Nature*, 420:206–210, 2002.

[75] M. Koffas, C. Roberge, K. Lee, and G. Stephanopoulos. Metabolic engineering. *Annual Review of Biomedical Engineering*, 1:535–557, 1999.

[76] I. A. Kurnaz. Biochemical modelling tools and applications to metabolic engineering. *Turkish Journal of Biochemistry*, 30:200–207, 2005.

[77] N. Le Novere, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. M. Sauro, M. Schilstra, B. Shapiro, J. L. Snoep, and M. Hucka. Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34:D689–691, 2006.

[78] A. M. Lesk. *Introduction To Bioinformatics.* Oxford University Press, 2005.

[79] J. C. Liao and J. Delgado. Advances in metabolic control analysis. *Biotechnology Progress*, 9:221–233, 1993.

[80] P. Mendes. Gepasi: a software package for modelling the dynamics, steady states and control of biochemical and other systems. *Computer Applications in the Biosciences: CABIOS*, 9:563–571, 1993.

[81] D. L. Nelson and M. M. Cox. *Lehninger Principles of Biochemistry.* Freeman, New York, 4th edition, 2004.

[82] D. Noble. A modification of the Hodgkin-Huxley equations applicable to Purkinje fibre action and pacemaker potentials. *The Journal of Physiology*, 160:317–352, 1962.

[83] B. G. Olivier. *Simulation and database software for Computational Systems Biology: PySCeS and JWS Online*. PhD thesis, University of Stellenbosch, 2005.

[84] B. G. Olivier, J. M. Rohwer, and J.-H. S. Hofmeyr. Modelling cellular systems with pysces. *Bioinformatics*, 21:560–561, 2005.

[85] B. G. Olivier and J. L. Snoep. Web-based kinetic modelling using jws online. *Bioinformatics (Oxford, England)*, 20:2143–2144, 2004.

[86] M. N. Pavlovic. Symbolic computation in structural engineering. *Computers & Structures*, 81:2121–2136, 2003.

[87] C. Reder. Metabolic control theory: a structural approach. *Journal of Theoretical Biology*, 135:175–201, 1988.

[88] D. G. Rhoads, M. J. Achs, L. Peterson, and D. Garfinkel. A method of calculating time-course behavior of multi-enzyme systems from the enzymatic rate equations. *Computers and Biomedical Research*, 2:45–50, 1968.

[89] J. M. Rohwer and F. C. Botha. Analysis of sucrose accumulation in the sugar cane culm on the basis of in vitro kinetic data. *Biochemical Journal*, 358:437–445, 2001.

[90] J. M. Rohwer and J.-H. S. Hofmeyr. Identifying and characterising regulatory metabolites with generalised supply-demand analysis. *Journal of Theoretical Biology*, 252:546–554, 2008.

[91] J. M. Rohwer, N. D. Meadow, S. Roseman, H. V. Westerhoff, and P. W. Postma. Understanding glucose transport by the bacterial phosphoenolpyruvate:glycose phosphotransferase system on the basis of kinetic measurements in vitro. *The Journal of Biological Chemistry*, 275:34909–34921, 2000.

[92] J. M. Rohwer, S. Schuster, and H. V. Westerhoff. How to recognize monofunctional units in a metabolic system. *Journal of Theoretical Biology*, 179:213–228, 1996.

[93] D. Rossouw, S. Bosch, J. Kossmann, F. C. Botha, and J.-H. Groenewald. Down-regulation of neutral invertase activity in sugarcane cell suspension cultures leads to a reduction in respiration and growth and an increase in sucrose accumulation. *Functional Plant Biology*, 34:490–498, 2007.

[94] S. Sanchez and A. L. Demain. Metabolic regulation of fermentation processes. *Enzyme and Microbial Technology*, 31:895–906, 2002.

[95] H. M. Sauro. Jarnac: a system for interactive metabolic analysis. In J.-H. S. Hofmeyr, J. M. Rohwer, and J. L. Snoep, editors, *In BioThermoKinetics 2000: Animating the Cellular Map*, Stellenbosch. Stellenbosch University Press.

[96] H. M. Sauro. Scamp: a general-purpose simulator and metabolic control analysis program. *Computer Applications In The Biosciences*, 9:441–450, 1993.

[97] H. M. Sauro, M. Hucka, A. Finney, C. Wellock, H. Bolouri, J. Doyle, and H. Kitano. Next generation simulation tools: the systems biology workbench and biospice integration. *Omics: A Journal of Integrative Biology*, 7:355–372, 2003.

[98] H. M. Sauro and H. Kacser. Enzyme-enzyme interactions and control analysis. 2. the case of non-independence: heterologous associations. *European Journal of Biochemistry / FEBS*, 187:493–500, 1990.

[99] H. M. Sauro, J. R. Small, and D. A. Fell. Metabolic control and its analysis. extensions to the theory and matrix method. *European Journal of Biochemistry / FEBS*, 165:215–221, 1987.

[100] M. A. Savageau. Concepts relating the behavior of biochemical systems to their underlying molecular properties. *Archives of Biochemistry and Biophysics*, 145:612–621, 1971.

[101] A. R. Schulz. Algorithms for the derivation of flux and concentration control coefficients. *Biochemical Journal*, 278:299–304, 1991.

[102] S. Schuster. Use and limitations of modular metabolic control analysis in medicine and biotechnology. *Metabolic Engineering*, 1:232–242, 1999.

[103] S. Schuster, T. Dandekar, and D. A. Fell. Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *Trends in Biotechnology*, 17:53–60, 1999.

[104] S. Schuster, D. A. Fell, and T. Dandekar. A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nature Biotechnology*, 18:326–332, 2000.

[105] S. Schuster and C. Hilgetag. On elementary flux modes in biochemical reaction systems at steady state. *Journal of Biological Systems*, 2:165–182, 1994.

[106] S. Schuster, D. Kahn, and H. V. Westerhoff. Modular analysis of the control of complex metabolic pathways. *Biophysical Chemistry*, 48:1–17, 1993.

[107] A. K. Sen. Quantitative analysis of metabolic regulation. a graph-theoretic approach using spanning trees. *Biochemical Journal*, 275:253–258, 1991.

[108] R. D. Simoni, R. L. Hill, and M. Vaughan. Urease, the first crystalline enzyme and the proof that enzymes are proteins: the work of james b. sumner. *The Journal of Biological Chemistry*, 277:e1–2, 2002.

[109] V. K. Singh and I. Ghosh. Kinetic modeling of tricarboxylic acid cycle and glyoxylate bypass in mycobacterium tuberculosis, and its application to assessment of drug targets. *Theoretical Biology and Medical Modelling*, 3:27, 2006.

[110] J. R. Small and D. A. Fell. The matrix method of metabolic control analysis: its validity for complex pathway structures. *Journal of Theoretical Biology*, 136:181–197, 1989.

[111] J. L. Snoep. The silicon cell initiative: working towards a detailed kinetic description at the cellular level. *Current Opinion in Biotechnology*, 16:336–343, 2005.

[112] J. L. Snoep, F. J. Bruggeman, B. G. Olivier, and H. V. Westerhoff. Towards building the silicon cell: A modular approach. *Biosystems*, 83:207–216, 2006.

[113] G. Stephanopoulos. Metabolic engineering. *Current Opinion in Biotechnology*, 5:196–200, 1994.

[114] S. Thomas and D. A. Fell. A computer program for the algebraic determination of control coefficients in metabolic control analysis. *The Biochemical Journal*, 292:351–360, 1993.

[115] S. Thomas, P. J. F. Mooney, M. M. Burrell, and D. A. Fell. Metabolic control analysis of glycolysis in tuber tissue of potato (solanum tuberosum): explanation for the low control coefficient of phosphofructokinase over respiratory flux. *Biochemical Journal*, 322:119 – 127, 1997.

[116] L. Uys, F. C. Botha, J.-H. S. Hofmeyr, and J. M. Rohwer. Kinetic model of sucrose accumulation in maturing sugarcane culm tissue. *Phytochemistry*, 68:2375–2392, 2007.

[117] J. C. Venter, M. D. Adams, E. W. Myers, and . others. The sequence of the human genome. *Science*, 291:1304–1351, 2001.

[118] D. Visser and J. J. Heijnen. The mathematics of metabolic control analysis revisited. *Metabolic Engineering*, 4:114–123, 2002.

[119] J. Vollinga. Ginac–symbolic computation with c++. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 559:282–284, 2006. Proceedings of the X International Workshop on Advanced Computing and Analysis Techniques in Physics Research - ACAT 05.

[120] J. D. Watson and F. H. Crick. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*, 171:737–738, 1953.

[121] H. V. Westerhoff. The silicon cell, not dead but live! *Metabolic Engineering*, 3:207–210, 2001.

[122] H. V. Westerhoff and Y.-D. Chen. How do enzyme activities control metabolite concentrations? *European Journal of Biochemistry*, 142:425–430, 1984.

[123] H. V. Westerhoff, J.-H. S. Hofmeyr, and B. N. Kholodenko. Getting to the inside of cells using metabolic control analysis. *Biophysical Chemistry*, 50:273–283, 1994.

[124] H. V. Westerhoff and D. B. Kell. Matrix method for determining steps most rate-limiting to metabolic fluxes in biotechnological processes. *Biotechnology and Bioengineering*, 30:101–107, 1987.

[125] H. V. Westerhoff and B. O. Palsson. The evolution of molecular biology into systems biology. *Nature Biotechnology*, 22:1249–1252, 2004.

[126] M. C. Wildermuth. Metabolic control analysis: biological applications and insights. *Genome Biology*, 1:1031.1–1031.5, 2000.

[127] J. H. Woods and H. M. Sauro. Elasticities in metabolic control analysis: algebraic derivation of simplified expressions. *Computer Applications in the Biosciences: CABIOS*, 13:123–130, 1997.

[128] K. Zotos. Performance comparison of maple and mathematica. *Applied Mathematics and Computation*, 188:1426–1429, 2007.

[129] K. Zotos. Computer algebra systems - new strategies and techniques. *Applied Mathematics and Computation*, 198:123–127, 2008.

Appendix A

Detailed description of SymCA classes

A.1 Data

This class consists of six methods, the first being `initData()`, which creates a dictionary to store all forms of data, i.e. the pysces, processed and symbolic data. The second method `getPyscesData` instantiates the `PyscesData` class which extracts and returns all required data from `PySCeS` and stores it in the data dictionary. The next two methods, `getProcessedData` and `getSymbolicData`, instantiate the `ProcessedData` and the `SymbolicData` classes respectively. The former takes the newly extracted `PySCeS` data and processes it ready for generation of the symbolic data by the latter class.

Once all data has been extracted, processed and generated a model map for all the data associated with the model in question is in a dictionary. The model map referred to as `_symcaModelMap`, is created and populated from within the `populateModelMap` method. The structure of the `_symcaModelMap` is as follows with all keys marked by quote marks and their corresponding values following the -:

- 'Coefficients' - dictionary of all elasticity and control coefficient data.
  - 'Parameter Elasticities' - dictionary of parameter elasticity data where keys are elasticity labels and values are corresponding `Coefficient` objects.
  - 'Variable Elasticities' - as per data structure above, except for variable elasticities.
  - 'Independent control' - with a list of all independent control coefficient data as the value. The list contains the following data:
    - 0 - dictionary of all independent control coefficients with their labels as keys and their corresponding values being `Coefficient` objects.
    - 1 - dictionary of coefficient matrix positions, with the keys being a two-membered tuple with elements representing the `PySCeS` and `Maxima` con-

trol coefficient matrix co-ordinates and the values being the control coefficient label.

* 2 - 2D array of all independent control coefficients.
* 3 - two-membered tuple with elements as follows:
  - 0 - list of all independent flux control coefficients.
  - 1 - list of all independent concentration coefficients.
  - 2 - list of all independent control coefficients.

– 'Dependent control' - as per the data structure described above, except for dependent control coefficients.

- 'Conserved Sums' - list of all conserved sums where each element in the list is a list of species found in each moiety-conserved cycle in the model.
- 'Elasticity Expressions' - dictionary of all elasticities with keys being the elasticity labels and the values being their corresponding symbolic expressions, as computed with `Maxima` by way of partial differentiation.
- 'Fixed Species' - `PyscesRecord` object for all fixed species.
- 'Flux' - `ProcessedRecord` object for all fluxes.
- 'Flux Relations' - dictionary of all flux relations where each flux is a key with its independent flux relations being the values.
- 'Flux Substitutions' - two-membered tuple with the following elements:
  - 0 - True or False, depending on whether or not a flux substitution has been performed.
  - 1 - A list of fluxes substituted into the $\mathcal{K}$ matrix, else None.
- 'Latex' - dictionary of LaTeX representations for all control coefficients, elasticities, fluxes and species, where the normal labels are the keys and the value is the LaTeX equivalent.
- 'Matrices' - dictionary of all symbolic matrices required by `SymCA`, where each matrix label is the key and the value is the corresponding `Matrix` object. The following matrices are present:
  - 'ccD' - dependent control coefficients `Matrix` object.
  - 'ccI' - independent control coefficients `Matrix` object.
  - 'E' - **E** `Matrix` object.
  - 'parElas' -parameter elasticities `Matrix` object.
  - 'varElas' - variable elasticities `Matrix` object.
  - 'k' - unscaled **K** `Matrix` object.

  - 'kdj' - `Matrix` object for diagonal matrix of inverse fluxes.
  - 'kdji' - `Matrix` object for diagonal matrix of independent fluxes.
  - 'ks' - scaled $\mathcal{K}$ `Matrix` object.
  - 'l' - unscaled $\mathbf{L}$ `Matrix` object.
  - 'lds' - `Matrix` object for diagonal matrix of inverse species.
  - 'ldsi' - `Matrix` object for diagonal matrix of independent species.
  - 'ls' - scaled $\mathcal{L}$ `Matrix` object.

- 'Parameters' - List of all model parameters.
- 'Parameter Elasticities' - `Elasticity` object for all parameter elasticities.
- 'Reactions' - `ProcessedRecord` object for all reactions.
- 'Rxn Format' - Common substring for all reaction labels, if one is present.
- 'Species' - `ProcessedRecord` object for all species.
- 'Values' - Dictionary of all steady-state values for fluxes and species, as well as initial parameter values:
  - 'steady state' - dictionary of flux and species steady-state values, with the flux or species labels as keys and their numerical value as the corresponding value,
  - 'parameter' - dictionary of all initial parameter values, where parameter labels are keys with their initial value as the values.
- 'Variable Elasticities' - `Elasticity` object for all variable elasticities.

It should be noted that the various objects referred to above are all custom created for `SymCA` and will be covered in the section on `SymcaObjects`, which will follow immediately after the sections covering the various data classes used by the `Data` class.

A.2 PyscesData

The methods below are involved in extracting and storing all required `PySCeS` data:

- `initObjects()` - This method requires two arguments when called, a dictionary containing the data to be stored and either 0 or 1 as the second argument. 1 indicates that the data to be stored is either the $\mathbf{K}$ or the $\mathbf{L}$, and 0 indicates all other `PySCeS` data to be stored. For each key value pair in the dictionary, a `PyscesRecord` object is created and appended onto a list which is then returned once all dictionary entries have been processed.

- `getConservedSums()` - Conservation data corresponding to moiety-conserved cycles is extracted from `PySCeS`, in the event that no conservation is present `None` is returned. When conservation is present a 2D list is returned where each entry consists of a list of species per conservation sum.

- `getAll()` - All `PySCeS` model data is extracted via this method which uses a number of nested methods.

  - `getModelData()` - All reaction, species, fixed species, parameters, conserved sums, equations as well as the `PySCeS` model data are extracted by this method. The conserved data is extracted via calling the `getConservedSums()` method and all equations are extracted via the `getEquations()`, all other data types mentioned are obtained directly from the `PySCeS` model instance attributes. `PyscesRecord` objects are created for all data by the `initObjects()` method and then loaded into the pyscesData dictionary, with the following key: `'model_data'`.
  - `getModelMatrices()` - All data required for the **K**, **L** and **N** matrices are extracted by this method and each matrix has its own method responsible for obtaining the desired data. Once all matrix data has been obtained it is stored in the pyscesData dictionary with the following key, `'model_mx'`.
    - `getK()` - The data highlighted in the following list is extracted from `PySCeS`, after which `PyscesRecord` objects are created and then stored in a list.
      - 2D `numpy` array of the **K** matrix.
      - tuple representing the reactions making up the columns of the **K** matrix.
      - tuple representing the reactions making up the rows of the **K** matrix.
      - 2D `numpy` array of the $\mathbf{K_0}$ matrix.
      - tuple representing the reactions making up the columns of the $\mathbf{K_0}$ matrix.
      - tuple representing the reactions making up the rows of the $\mathbf{K_0}$ matrix.
    - `getL()` - This method extracts the same data as the `getK()` method, but for the **L** matrix `PyscesRecord` objects are created and appended onto the list containing the **K** matrix data.
      - 2D `numpy` array of the **L** matrix.
      - tuple representing the reactions making up the columns of the **L** matrix.
      - tuple representing the reactions making up the rows of the **L** matrix.
      - 2D `numpy` array of the $\mathbf{L_0}$ matrix.
      - tuple representing the reactions making up the columns of the $\mathbf{L_0}$ matrix.
      - tuple representing the reactions making up the rows of the $\mathbf{L_0}$ matrix.

    * `getN()` - Data pertaining to the stoichiometric **N** matrix is obtained from `PySCeS`, turned into `PyscesRecord` objects and appended onto the list containing the previously extracted **K** and **L** matrix `PyscesRecord` objects.
        · 2D `numpy` array of the **N** matrix.
        · tuple representing the reactions making up the columns of the **N** matrix.
        · tuple representing the reactions making up the rows of the **N** matrix.

- `getParameterValues()` - The initial parameter values for all parameters are obtained and stored in a dictionary with parameter names as keys and their corresponding initial values as the values. This dictionary is stored in the pyscesData dictionary under the key, `'parameter_values'`.

- `getSteadyState()` - The steady-state values for all flux and species are extracted and stored in two dictionaries, one containing all flux data and the other all species data. Each dictionary contains the flux or species labels as the keys and their corresponding steady-state values as the values. These dictionaries are stored in a dictionary, using keys `'flux'` and `'species'`, which are then stored in the pyscesData dictionary under the following key, `'steady state'`.

- `getEquations()` - This method is responsible for extracting the rate equation for each reaction and storing this information in a dictionary where the reaction name serves as the key and its rate equation as the value. Once all reaction rate equations have been processed the dictionary is returned.

- `getData()` - The dictionary populated by all methods in this class is returned.

- `getDataKeys()` - This method returns all the keys associated with the dictionary containing all extracted `PySCeS` data.

A.3 ProcessedData

Below are the methods involved in processing all initial `PySCeS` data:

- `processData()` - The main method processing all `PySCeS` data contains the following nested methods:

    – `K()` - Method which processes all symbolic data concerning the **K** matrix. The data from the `PyscesData` dictionary is extracted and reformatted to create a **K** `Matrix` object which is incorporated into a `ProcessedRecord` object. This object is then stored in the processed data dictionary with `'K'` as the key.
    – `L()` - In a similar way to the method above, this method reformats the `PyscesData` object for the **L** matrix into a `Matrix` object which is incorporated into a `ProcessedRecord` object and stored in the processed data dictionary with `'L'` as the key.

- `equations()` - This method is called by `reactions()` and extracts the dictionary of all reactions and their rate equations returning a list of all reactions and a corresponding list of all rate equations.
- `rxn()` - Two arguments are required for this method, the reaction name and a three-membered list containing the following data: a list of reaction substrates, a list of reaction products and a list of all reaction modifiers. The `process()` method is then called with the same arguments.
  - `process()` This method requires the same arguments as the `rxn()`. The data is then used to generate a list of all free species associated with the reaction, a list of all fixed species and a list of all variable elasticities. This is done for substrates, products and modifiers involved in the given reaction. The data pertaining to the substrates, products and modifiers is then returned to the `rxn()` method which in turn returns it to its calling method, i.e. `reactions()`.
- `reactions()` - This is the main method used to process the initial `PySCeS` reaction data. The `equations()` method is called, returning a list of all reaction names and a corresponding list of all rate equations. The list of reactions is then stepped through and for each reaction the `rxn()` method is called returning a list containing data for the substrates, products and modifiers associated with the reaction. The `deindex()` method is also called which returns a list of all dependent and independent variables for the reaction in question. A list of parameter elasticities is also generated using the rate equations specific to each reaction and a `Reaction` object is created utilising all newly processed data; this is appended onto a list to store all `Reaction` objects. Once all reactions have been processed and an ordered list of all reactions is generated (independents followed by dependents), a `ProcessedRecord` object is created incorporating the list of `Reaction` objects. This is then stored in the processed data dictionary with `'reaction'` as the key.
- `Species()` - The initial `PySCeS` data is extracted and processed such that a Species object is created for each species and then stored in a list. An ordered list of independent followed by dependent species is created, and all data is then stored as a `ProcessedRecord` which is stored in the processed data dictionary under the `'species'` key.
- `speciesRelations()` - This method generates a `ProcessedRecord` for any conserved sums present in the model; this object is stored in the processed data dictionary with the 'conserved' key.
- `parameters()` - A list of all parameters associated with the model is generated and stored in the processed data dictionary with `'parameters'` as the key.
- `Flux()` - This method uses the reactions data to generate flux data, which includes an ordered list of fluxes expressed in terms of independents followed by dependents, as well as Flux objects for each individual flux. The Flux objects are stored in a list which is incorporated into a `ProcessedRecord`

object for all fluxes and then stored with the key `'flux'` in the processed data dictionary.

- `fluxRelations()` - The flux relations are important, since, if any common relationships exist, substitutions may be performed in the $\mathcal{K}$ matrix before building the **E** matrix, and this method is responsible for generating symbolic dictionaries of these relations. For each flux in the system, its relation in terms of independent fluxes is generated. This data is then stored in three dictionaries: one for all independent flux relations, one for all dependent relations and one containing all flux relations. This data is then used to create and store a `ProcessedRecord` object with `'flux relations'` as its key.

- `depIndep()` - Three methods, `Flux()`, `reactions()` and `Species()`, utilise this method during the processing of `PySCeS` data. This method has two arguments, the data to be sorted (lists of flux, reactions or species data) and the type of data being sorted, i.e 'flux', 'rxn' or 'species'. A list of dependent and a list of independent data is then returned.

- `getElasticityExpressions()` - Since elasticity coefficients are partial derivatives of rate equations with respect to a given species, they can be computed from the rate equations. This method computes the symbolic elasticity expressions for all elasticities per reaction using `Maxima` and its in-built `differentiate` method. A list of reactions is stepped through and all associated elasticity expressions are computed by `Maxima` and then written to a file. Once all have been computed and written, the `readElasticityFile()` method is called which returns a list of all elasticity expressions. An ElasticityExpression object is created for each elasticity and these are stored in a list which is then incorporated into a `ProcessedRecord` object and stored in the processed data dictionary with the key `'elasticities'`.

- `readElasticityFile()` - This method takes the name of the file to which the symbolic elasticity expressions have been written by `Maxima`. The expressions are then extracted into a list and returned.

- `getData()` - This method returns the dictionary containing all processed data for use by the `SymbolicData` class.

A.4 SymbolicData

The following methods are required to generate all symbolic data required to perform symbolic control analysis:

- `buildSymbolics()` - In order to generate the symbolic **E** matrix the user needs to symbolically scale both the **K** and **L** matrices to form $\mathcal{K}$ and $\mathcal{L}$, respectively. The **K** matrix requires the formation of a diagonal matrix of inverse fluxes ($(\mathbf{D^J})^{-1}$) and a diagonal matrix of independent fluxes ($\mathbf{D^{J_i}}$), and the **L** matrix requires

the formation of a diagonal matrix of inverse species ($(\mathbf{D^S})^{-1}$) and a diagonal matrix of independent species ($\mathbf{D^{S_i}}$). This method generates these additional four matrices by calling the `buildSymbolicMx()` method from the `Utilities` class, which returns the symbolic form of the matrix, its `Maxima` input argument and the dimensions of the newly formed matrix. `Matrix` objects are created for each matrix generated and these are then used to form `SymbolicRecord` objects, which are stored in a symbolic data dictionary under the following keys: `'kdj'`, `'kdji'`, `'ls'` and `'lsi'`.

- `scaleMatrices()` - The **K** and **L** matrices are scaled according to the algorithm as described by Hofmeyr [51]. **K** is first scaled by `Maxima` and once returned substitutions are performed based on the flux relations. This step is performed by a method housed in the `Utilities` class, which searches for identical relationships amongst the dependent fluxes. The reasoning behind this step is that if any common relations exist, the number of dependent fluxes found in the $\mathcal{K}$ matrix can be reduced, which in turn aids the efficiency in the inversion of the **E** matrix. If any fluxes are substituted into the $\mathcal{K}$ matrix they are stored in a list as they will be required for future factorising of the resultant control coefficients. The **L** matrix is then scaled, and before the creation of `Matrix` objects for both the $\mathcal{K}$ and $\mathcal{L}$ matrices, the dimensions of both are compared with the dimensions of their unscaled counterparts. The `Matrix` objects are only initialised after a successful comparison. The $\mathcal{K}$ and the $\mathcal{L}$ `Matrix` objects are then stored in the symbolic data dictionary with the following keys, `'ks'` and `'ls'`, respectively.
- `buildElasticityMatrix()` - Both symbolic parameter and variable elasticity matrices must be generated and this method in combination with its nested `buildMx()` method was developed to fulfill this role. Elasticity objects representing all parameter and variable elasticities are first created, both of which are then incorporated into `SymbolicRecord` objects which are in turn stored in the symbolic data dictionary as a two-membered tuple with `'elasticities'` as the key. The `buildMx()` method is called twice, once for variable elasticities and once for the parameter elasticities.
  - `buildMx()` - This method is directly responsible for the generation of the desired elasticity matrices, and requires three arguments when called. These arguments are an ordered list of all species or list of parameters depending on which matrix is being built, the `Elasticity` object for the elasticities in question and finally the label to be used for the resultant matrix. The matrices are arranged with the species or parameters as the columns and the ordered list of reactions as the rows. In order to build the matrices, the list of reactions is stepped through, and for each step the list of species/parameters is stepped through and a check is performed identifying whether or not the species/parameter is involved in the reaction. If present, an elasticity is present and the symbol repre-

senting that particular elasticity can be entered into the matrix, when absent a zero is inserted into the matrix. For every elasticity found a `Coefficient` object is created and stored in a list. The symbolic data dictionary is finally populated with the `Elasticity` object list, as well as a `Matrix` object for both parameter and variable elasticities, under the following keys: `'parameter elasticity coefficients'`, `'parameter elasticity matrix'`, `'variable elasticity coefficients'` and `'variable elasticity matrix'`.

- `buildCCMatrix()` - Symbolic representations of both the independent and dependent matrices of control coefficients are generated via this method. For both matrices the ordered list of reactions serve as the columns and the rows are represented by a list of fluxes followed by species. A `Matrix` object is created for each matrix generated, as well as `Coefficient` objects for all control coefficients. `SymbolicRecord` objects are created for independent and dependent control coefficients and stored in the symbolic data dictionary with keys `'independent control coefficients'` and `'dependent control coefficients'`, respectively. A `SymbolicRecord` is created for all control coefficients and this is stored in the symbolic data dictionary with the key, `'control coefficients'`.
- `buildE()` - The final symbolic matrix required by `SymCA` is the **E** matrix which is generated by multiplying the variable elasticity matrix by -1, and then by the $\mathcal{L}$ matrix. The $\mathcal{K}$ matrix is then augmented onto the newly computed matrix product. These computations are performed by `Maxima`, after which a `Matrix` object is created for the **E** matrix which is also stored in the symbolic data dictionary.

- `mergeCC()` - This method takes a tuple with independent and dependent control coefficient data as elements, and combines them so that independent control coefficients are followed by dependent control coefficients. A tuple with all `Coefficient` objects, a full coefficient dictionary and a list of all coefficients is then returned.

- `checkScaledK()` - This method serves as a check to see if the symbolic scaling of the **K** matrix has been performed correctly. The dimensions of the symbolic $\mathcal{K}$ matrix are compared with the `PySCeS` equivalent, and the contents of the matrices checked to ensure that entries are in the correct positions.

- `getLatexEquivalents()` - In order for `SymCA` to generate LaTeX output the LaTeX equivalents of all control coefficient, elasticity coefficient, flux and species labels need to be generated. Any common reaction substring is removed from any labels and this option is controlled by the user. A dictionary is created with the keys being the control coefficient, elasticity coefficient, flux and species labels with their corresponding LaTeX equivalents as the values. This dictionary is then stored in the symbolic data dictionary with `'latex equivalents'` as its key. The common

reaction substring, if present, is also stored in this dictionary under the key, `'rxn format'`.

- `getData()` - This method returns the dictionary containing all symbolic data.

A.5 SymcaObjects

The details for all custom objects involved in storing the initial data before symbolic control analysis can be found below:

- `Coefficient` - These objects were created to store all symbolic control and elasticity coefficients, they possess no methods and the table below summarises the attributes associated with the `Coefficient` objects.

Table A.1: Description of `coefficient` object attributes

Attribute	**Description**
name	label of the coefficient
mx_pos	index of the coefficient in its respective matrix
type	the type of the coefficient e.g. elasticity, control, response
dependency	independent or dependent

- `Elasticity` - The `Elasticity` object was created to store parameter and variable elasticity data. The following attributes are accessible:

Table A.2: `Elasticity` class attributes

Attribute	**Description**
name	either 'variable elasticity' or 'parameter elasticity'
elasticities	list of all elasticities
substrate	list of all substrate elasticities
product	list of all product elasticties
modifier	list of all modifier elasticities

- `ElasticityExpression` - Since the symbolic elasticity expressions are computed by `Maxima`, this class was created to store this information. There are a number of attributes, and no methods in this class.

Table A.3: `ElasticityExpression` class attributes

Attribute	**Description**
name	elasticity label
equation	rate equation associated with the elasticity
reaction	reaction in question
species	species or parameter affecting the reaction
expression	symbolic elasticity expression
parameters	list of all parameters in the expression

- `Flux` - This object stores data for each individual flux, such as its steady-state value and whether it is an independent or dependent flux. There are four attributes for this object.

Table A.4: `Flux` object attributes

Attribute	**Description**
name	the flux label
dependents	True or False
independents	True or False
steadyState	steady-state flux value

- `Species` - Much like the Flux object, the Species object stores data pertaining to each species.

Table A.5: `Species` object attributes

Attribute	**Description**
name	label assigned to the species in the `PySCeS` input file
dependents	True or False
independents	True of False
steadyState	the value of the species at steady-state

- `Matrix` - All matrices generated by `SymCA` are stored as `Matrix` objects, these methods contain no methods and have the following attributes:

Table A.6: `Matrix` object attributes

Attribute	**Description**
name	name of the matrix
data	2D `Python` list representation of the matrix
maxima	matrix in `Maxima` input format
pysces	matrix in `PySCeS` output format
label	the `Maxima` matrix label
size	matrix dimensions
elements	list of all elements in the matrix

- `ProcessedRecord` - Once the initial `PySCeS` data has been processed it is stored as a `ProcessedRecord` object, which has the following attributes:

Table A.7: `ProcessedRecord` object attributes

Attribute	**Description**
name	title assigned to the data being stored
data	the data or data object which is being stored
dependents	list of dependents or None
independents	list of independents or None
ordered	ordered list - independents followed by dependents
network_labels	list of `PySCeS` labels for data else None
fixed	default is None, else list of all fixed items

- `SymbolicRecord` - Once all data was extracted, processed and symbolic data generated, the resultant objects representing this symbolic data are stored as `SymbolicRecords` each possessing the following attributes:

Table A.8: `SymbolicRecord` object attributes

Attribute	**Description**
name	name assigned to the symbolic data
data	symbolic data object

- `Reaction` - Each reaction contains a large amount of data, i.e. products, substrates and parameters associated with it. This object was created to store all data related to a reaction, and has a number of attributes:

Table A.9: `Reaction` object attributes

Attribute	**Description**
name	the reaction label as provided in the `PySCeS` input file
equation	rate equation for the reaction
substrates	list of reaction substrates
products	list of reaction products
modifiers	list of reaction modifiers
parameters	list of parameters in the reaction
fixed_substrates	list of all fixed substrates
fixed_products	list of all fixed products
fixed_modifiers	list of all fixed modifiers
variable_substrate_elasticities	list of reaction variable substrate elasticities
variable_product_elasticities	list of reaction variable product elasticities
variable_modifier_elasticities	list of reaction variable modifier elasticities
variable_elasticities	dictionary of all variable elasticities
parameter_elasticities	dictionary of all parameter elasticities

- `PyscesRecord` - As per the processed data, all initial `PySCeS` data was extracted and then stored as a `PyscesRecord` object. Below are the attributes associated with these objects:

Table A.10: `PyscesRecord` object attributes

Attribute	Description
name	`PySCeS` data label
data	data extracted from `PySCeS`
maxima	`Maxima` input argument for `PySCeS` data
pysces	the `PySCeS` model instance

A.6 SCA

All methods associated with the `SCA` class are outlined below:

- `getData()` - This method serves to return the dictionary of all data generated by `SCA`, which has been dubbed the '_symcaModelOut'.
- `computeControlCoefficients()` - The main method controlling the computation of all symbolic control coefficients, and all processes directly related to it. This method is called with four arguments, two of which are optional and are denoted with a *:
  - `dep` - True to compute dependent control coefficients and False for only independent control coefficients.
  - `subE` - True to perform elasticity substitutions in the **E** before inversion, and False to proceed as normal.
  - `subDict`* - Default is None, but when `subE` is True a dictionary of elasticity substitutions is required.
  - `factor`* - Default is None, else True if numerator and denominator can be further simplified.

  Once this method has been called the status of the `subE` argument is checked, if True then the `setElasticityPre()` method is called to perform the desired elasticity substitutions in the **E** matrix. If `subE` is False the process proceeds as usual where the `invertMatrix()` method is called. On completion, True or False is returned to indicate the success or failure of the inversion process. If True the common denominator and control coefficient numerators can be generated, and is achieved by calling the following methods `computeDenominator()` and `computeNumerators()`, respectively. Both methods return a boolean as to their execution status.

  When both the common denominator and all control coefficient numerators have been successfully computed a method (`applyRelations()`) is called to perform further computations based on any moiety-conserved cycles found in the system,

as well as any flux substitutions in the $\mathcal{K}$ matrix. The data for both common denominator and numerators is then extracted from `Maxima` and written to file by calling `outputControlCoefficients()`. Initially all output generated by `Maxima` was returned by the standard output pipe, however this proved to be inefficient and in fact impossible in certain instances when the buffering capacity of the output pipe was exceeded. Fortunately `Maxima` has several methods for writing data directly to disk, which is now the method used to return large data sets.

Once all data has been extracted from the `Maxima` environment, the data is then processed and stored as `ControlCoefficient` objects, by the `processControlCoefficient()` method. The data is now ready for output as well performing additional operations which will be addressed in the chapter regarding the `SymCA` user interface.

- `invertMatrix()` - `Maxima` has a predefined function for performing matrix inversions for both numerical and symbolic matrices. This method uses this function to generate the desired symbolic control coefficients. The `invertMatrix()` method requires two arguments, the first being the label of the matrix to invert, which in this case is 'E', the second argument is optional and is used only in the event that one would like to invert a different matrix. The matrix in `Maxima` input format would then be provided for this argument referred to as `mx`. Before the inversion a number of flags are set in the `Maxima` environment, these enable the inversion process to factor out the determinant of the matrix. Once all has been successfully completed True is returned, whereas if an error occurs False is returned.

- `computeCommonDenominator()` - The common denominator for all control coefficients is the determinant of the **E** matrix, and `Maxima` has a method named determinant, which returns the determinant of a specified matrix. The method requires two arguments, with the first being the name of matrix from which the determinant is to be computed, and the second being the label to assign to the extracted determinant. Once the denominator has been computed, True is returned.

- `computeNumerators()` - Since the determinant (denominator) is factored during the inversion routine, the resulting product is a matrix containing all control coefficients divided by the common denominator. Two arguments are required, the label of the matrix from which the control coefficient numerators are to be obtained, and the label for the subsequent matrix of control coefficient numerators. The control coefficient matrix is then obtained by accessing the numerator, using the `Maxima` `num` method. At this stage the dependent control coefficients are computed, if desired, and this computation is handled by calling the `computeDependents()` method.

- `computeDependents()` - Two arguments are required to call this method, a two-membered tuple with the label for the independent flux control coefficient matrix and the label for the independent concentration control coefficient. The second

argument is the label of the matrix of all independent control coefficients. The dependent flux control coefficients are computed as follows, $\mathbf{C^{J_d}} = \mathcal{K}_0.\mathbf{C^{J_i}}$, and the dependent concentration control coefficients in a similar manner, $\mathbf{C^{S_d}} = \mathcal{L}_0.\mathbf{C^{S_i}}$ [51]. Once computed the matrices are augmented onto $\mathbf{C^i}$, with $\mathbf{C^{J_d}}$ added first followed by $\mathbf{C^{S_d}}$. All control coefficients are now computed.

- `applyRelations()` - This method requires two arguments, the `Maxima` denominator label and the `Maxima` label representing the matrix of all control coefficient numerators. The result of this method is that the expressions are rearranged so that elasticity terms are only multiplied (i.e not divided) by fluxes, and the elasticities towards species in conserved moieties are divided by the same species concentrations. This is done by multiplying both the denominator and control coefficient matrix by all fluxes that where substituted into the $\mathcal{K}$ matrix, and secondly by dividing the resultant products by each independent species occurring in the conservation moieties.

- `outputControlCoefficients()` - In order to extract the control coefficient numerators and the common denominator the initial strategy was to use the standard output pipe native to the `subprocess` module, which is used to communicate with `Maxima`. This was found to be efficient for small data sets. However it was not viable for the data sets generated when large systems were analysed. The strategy thus employed uses of a native method to `Maxima` that writes data directly to a file. This process is controlled by the `outputControlCoefficients()` method and uses the `stringout` function from `Maxima`. The denominator is first extracted and its sign checked, as the signs (+ or −) of the expression must be biologically correct (see Equations 2.7 & 2.8 in Section 2.4, Chapter 2 for an example), which will be discussed in `checkDenom()`, the method governing this process. The sign is either right or wrong, and in the case of it being wrong both the denominator and the control coefficient numerators must be corrected by multiplying each by -1. Once the denominator has been extracted, checked and possibly corrected, all numerators are outputted by the `outputNumerators()` method. A tuple containing the following information is returned:

  - 0 - string representation of the denominator.
  - 1 - a dictionary of all control coefficient expression with control coefficient labels as keys and their string representations as the values.
  - 2 - path for denominator `Maxima` output file.
  - 3 - path for control coefficient `Maxima` output file.

- `outputDenom()` - This method takes the `Maxima` denominator label as an argument and uses the `Maxima stringout` function to write the denominator expression to file. The file is stored in the maxima output subdirectory within the model directory, using the `Maxima` label as its title, e.g 'den.txt'. A tuple containing the file path, the denominator label and the `stringout` status is returned.

- `extractDenom()` - Once the denominator has been written to file, it needs to be extracted. This method has one argument, which is the tuple returned by the `outputDenom()`. The denominator expression is read from the output file, parsed by the `Parser` class and then returned in string form.

- `checkDenom()` - This method is integral for ensuring that the expressions computed have the correct sign (+ or −). Since the denominator is the determinant of the **E** and thus contains all elasticities for the cellular system in question, it is used as the check expression. The expression is broken up into terms, which are separated by either a + or −. A term is then isolated and checked by `checkSign()`. Since product elasticities are assumed to have a negative sign and substrate elasticities a positive one, the overall theoretical sign for a term can be determined. If this theoretical sign correlates to the sign computed by `Maxima`, it can be assumed that all the signs for the control coefficients are correct, if the opposite is true, then all signs need to corrected. This is achieved by multiplying the denominator as well as all control coefficient numerators by −1. This method sets the values for two global variables, _term_sign and _sign_change, to either True or False.

- `outputNumerators()` - Once the denominator has been successfully outputted, extracted and checked, the control coefficient numerators can be extracted. The type of data to be outputted is provided as the only required argument, after which each numerator is individually written to file via `Maxima's stringout` function. At this stage the numerator signs are changed if required. All numerator expressions are written to a file within the maxima output directory, with the same naming convention used for the denominator, which in the case of the control coefficients is 'cc.txt'. Once all expressions have been written to file, the `extractNumerators()` method is called which returns a dictionary of all control coefficients expressions computed. This dictionary along with the file path is returned as a tuple.

- `extractNumerators()` - The extraction method takes two arguments, the first of which is the path for the file containing all `Maxima` numerator output, and the second is a tuple containing a list of all control coefficient labels and a list of corresponding matrix positions for the control coefficients. The data from the file is read by the `read_cc_eqns()` method in the `Utilities` class, which returns a list of all expressions. The list is then stepped through and for each expression, the control coefficient label as well as matrix position is extracted from the lists provided as arguments. A dictionary is then created for each control coefficient with keys, 'eqn' and 'mx_pos', and their corresponding values, the symbolic expression and the matrix position, respectively. This dictionary is then stored in a further dictionary with the control coefficient label as its key. Once all control coefficients have been added to the dictionary it is returned.

- `processControlCoefficients()` - The final processing of the denominator and control coefficients, which creates custom `Python` objects for them, is controlled by this method. A tuple of denominator and control coefficient as returned by the

`outputControlCoefficients` method, as well as the `Maxima` denominator label as arguments. This method calls the `processDenominator` and `processNumerators` methods, with the data from each being used to populate the _symcaModelOut dictionary with keys 'Denom' and 'Control' respectively.

- `processDenominator()` - In order to minimise the memory requirements of `SymCA` the denominator expression is stored in the pickle objects directory as a `Python` pickle. A `Denominator` object is then created for the common denominator, which is returned to be stored in the main `SymCA` output dictionary.

- `processNumerators()` - Using the same rationale as above, all control coefficient numerators are pickled and then a `ControlCoefficient` object is created for each control coefficient. The `ControlCoefficient` objects are stored in a dictionary, where keys are control coefficient labels and the values are their corresponding `ControlCoefficient` objects. This dictionary is then returned to be stored in the main `SymCA` output dictionary.

- `setElasticityPre()` - The possibility exists to pre-set elasticity values (e.g. 0, 1, -1, 0.5, 2). This method requires a dictionary of elasticity substitutions to be performed, where the elasticities to be substituted are the keys and the values to be substituted are the values. The substitution takes place before the inversion of the **E** matrix and is performed by `Maxima`. The substitution is done in the $\boldsymbol{\varepsilon}_{\mathbf{s}}$ matrix using a `Maxima` method, `sublis`, which makes multiple parallel substitutions into an expression, after which the **E** matrix is regenerated.

- `setElasticityPost()` - This method allows for the substitution of elasticities in much the same manner as the `setElasticityPre()` method. However, these substitutions are performed after the computation of the control coefficients, and has a number of variations as to how the user would like to perform the substitutions. There are four arguments for this method, of which only the first is required:

  - `elas_dict` - dictionary of all desired elasticity substitutions; keys are elasticity labels and values are their corresponding values to be substituted in.
  - `sub_type` - two variations of substitution can be performed with the default being type 0:
    - 0 - performs all substitutions simultaneously.
    - 1 - performs each substitution individually as well simultaneously.
  - `number` - this number determines the number of elasticities to be used in the generation of elasticity substitution combinations.
  - `subList` - a list of species or fluxes used to generate a list of control coefficients to be substituted, else all control coefficients are used.

  Type 0 and 1, rely on the `subElas()` method which performs the elasticity substitutions as well as extracts the resultant data from `Maxima`, then processes it and finally stores it for later use.

- `subElas()` - The `subElas()` method is used by `setElasticityPost`, to perform elasticity substitutions after the control coefficients have been computed. It is important to note that any substitutions performed in the numerators also need to take place in the denominator. The dictionary of desired elasticity substitutions are converted into the required `Maxima` input for the `sublis` function; if only one substitution is desired the labels for the resulting substituted denominator and control coefficients all have the elasticity name with the addition of '_sub_' as their prefix. When multiple substitutions are performed the 'multiple_sub_' is used as the prefix, the prefix used is then added to a global dictionary with the corresponding `Maxima` substitution argument as its value. After the substitution has been performed the `outputControlCoefficients()` method is called which returns the newly substituted control coefficient data. The denominator is then substituted, outputted and extracted, the latter two tasks being handled by the `outputDenom` and `extractDenom` methods respectively. The `processSubstituted()` method is then called to process the newly computed substituted denominator and control coefficient data.

- `processSubstituted()` - All substituted denominator and control coefficient data is stored in the same manner as the original coefficient data generated before the substitution. Four arguments are required by this method, the substituted numerator data, substituted denominator data, the elasticity substitution dictionary and the substitution label used as a prefix for all coefficients. The substituted denominator is first pickled and stored in the pickle objects subdirectory, after which a new `Denominator` object is created, this object is then added to the substituted_denom dictionary attribute of the original `Denominator` object using the substituted denominator label as its key and the newly created `Denominator` object as the value. All substituted coefficients are then processed in the same manner, with the end result being the creation of a substituted `ControlCoefficient` object. This object is then added to the substituted_cc dictionary attribute of its corresponding original `ControlCoefficient` object, with the substituted control coefficient label as the key and the substituted `ControlCoefficient` object the value.

- `simplify()` - Since numerators and denominators are separated, there may be common terms in both which could be factored out. This method serves to fulfill this task, and if called will factorise each control coefficient numerator divided by the common denominator on an individual basis. For each factorisation performed, the resultant numerator and denominator is extracted and the label assigned stored in a list for use in the method governing the output of these numerator and denominator terms from `Maxima`. In the event of the user previously performing elasticity substitutions on the control coefficients after the inversion routine, all substituted control coefficients are also factorised, via the `simplifySubstituted()` method.

- `outputSimpCC()` - The output of all factorised data resulting from the `simplify()` method is handled via this method, which requires the list of all control coefficients factorised, as well as the list of resulting `Maxima` labels for the numerator and

denominator expressions for each factorised control coefficient as its arguments. All expressions are written to a file named, 'simpCC.txt', which is located in the maxima output subdirectory, in the order in which they appear in the list of all `Maxima` labels. The path for the file containing all outputted data is then used as the argument for the extraction method, `extractSimpSubCC()`, which simply reads in the file and returns a list of all expressions.

- `processSimpCC()` - The newly extracted expressions are then processed by this method, where the list of expressions, a list of all control coefficients and a list of `Maxima` labels are required as arguments. The list of all `Maxima` labels is stepped through, with a step size of 2 starting at position 0. The reason for this is that each factorised control coefficient consists of two parts, a numerator and a denominator, where the denominator label follows immediately after the numerator label. Each expression is parsed via the `Parser` class, the `ControlCoefficient` objects simplified attribute for each control coefficient factorised is set to True, and a `Denominator` and `ControlCoefficient` object created for the factorised denominator and numerator term. The objects are then assigned to the original `ControlCoefficient` objects simp_denom and simp_numer attributes respectively.

- `simplifySubstituted()` - The `simplifySubstituted()` method performs the same task as `simplify()`, except that it factorises all substituted control coefficients with their corresponding substituted denominator. The substituted control coefficients for all control coefficients in the list provided, which serves as the only argument, are factorised. The numerators and denominators from the resultant expression are extracted and all resulting denominator and numerator labels stored in a list.

- `outputSimpSubCC()` - The factorised substituted control coefficients are written to a file named 'simpSubCC.txt', which is located in the maxima output subdirectory, after which they are extracted by the `extractSimpSubCC()` method which returns a list of all expressions. The processing of the newly generated expressions is then handled by calling `processSimpSubCC`.

- `processSimpSubCC()` - This method requires a list of expressions, a list of all control coefficients, a list of all substitution arguments corresponding to the control coefficients list and a list of the `Maxima` labels for the expressions list as its arguments. The data is processed with the method described for the `processSubCC()` method, with the only difference being the location of the Denominator and `ControlCoefficient` objects created for each substituted control coefficient factorised. These objects are stored in the corresponding `ControlCoefficient` objects substituted_cc dictionary attribute. This dictionary has substitution arguments as keys, and as such the list of substitution arguments provides a means of locating the correct `ControlCoefficient` object within this dictionary. The newly created Denominator and `ControlCoefficient` objects are then assigned to the simp_denom and

simp_numer attributes of the substituted ControlCoefficient within the substitution_cc dictionary.

- `writeData()` - There are numerous types of LaTeX output options available, each depending on the preceding methods used. The method requires a label relating to the data to be written, and an optional file name in which to write the data as arguments. The types of options available are:
  - 'cc': all computed control coefficient data.
  - 'sym2psc': data generated by comparing numerical data computed via `PySCeS` with data obtained by numerically substituting the symbolic expressions for all control coefficients.
  - 'patterns': data computed by quantifying the contribution and value for each control pattern within a control coefficient expression.
  - 'summation': data generated from computing the summation theorem.
  - 'coc': co-control coefficient data generated.
  - 'scan': data generated by the quantification of control patterns whilst performing a parameter scan.

  For each type of data selected, the `Output` class generates and writes the LaTeX output which is then saved to files which are located in the latex subdirectory.

- `symcaToPysces()` - This method was built as a check that the symbolic control coefficient expressions are correct. The values for all Denominator and `ControlCoefficient` objects are computed at instantiation and assigned in both cases to the ss_value attribute. The values for the symbolically generated control coefficients are obtained from the `ControlCoefficient` objects and stored in a dictionary with the value obtained from `PySCeS`. This method has already proved useful with respect to the signs associated with the symbolic expressions, and when all signs are correct the values obtained via both means are identical. However, in the event of the expressions having the wrong signs, the absolute values are identical but the results have opposite signs. This has led to the inclusion of this step in the Unittesting routine, which is included in every installation of `SymCA`.
  - `scaCCValues()` - This method obtains all control coefficient values from `PySCeS` and from the values computed from the symbolic expressions. This data is then stored in the `SymCA` output dictionary with the key, 'SymcaToPysces'. As mentioned previously this data is also used as part of the `Unittest` routine, and is stored with the key 'sym2psc', in the `Unittest` dictionary which in turn is stored in the `SymCA` output dictionary with 'Unittest Data', as its key.

- `pyscesValues()` - The steady-state values as computed by `PySCeS` are required for a number of operations within `SymCA`. This method obtains these values for all

data sets required, i.e. flux, species, control coefficient, elasticities and parameters. All values obtained are stored in a dictionary with the following keys:

- 'flux': all steady-state flux values.
- 'species': all steady-state species concentration values.
- 'parameters': all parameter values.
- 'cc': all steady-state control coefficient values.
- 'var_ec': all steady-state variable elasticity values.
- 'par_ec': all steady-state parameter elasticity values.

A list of each of the above mentioned data sets is obtained, after which the `getPyscesValues()` method is called with the list as the only argument. A dictionary is then returned where the labels of the variables are keys and their steady-state values the dictionary values. Once all steady-state value dictionaries have been obtained and placed into a single dictionary, this dictionary is stored under the key, 'Pysces Values', in the `SymCA` output dictionary.

- `getPyscesValues()` - This method requires a list of items for which the steady-state values are required as its only argument. The list is stepped through and each value obtained from `PySCeS` via the `getattr()` method, is then added to a dictionary with the item label and steady-state value as the key value pair. After the values for all items have been obtained and added to the dictionary it is returned.

- `computeSummations()` - The summation theorems [65] are computed via this method. All control coefficient values obtained via substitution of the steady-state flux, species and elasticity values are computed. The summations are then calculated and compared to the theoretical values for both flux and concentration control coefficients.

- `computeResponse()` - When computing symbolic response coefficients, the numerical `PySCeS` equivalents must first be computed. This is achieved by calling this method. The resultant response coefficients are then stored in a dictionary.

- `getResponse()` - This method computes symbolic response coefficients and is used after `computeResponse()` has been called. The user is able to obtain a selection of symbolic response coefficients by providing the necessary data for the three optional arguments, which are as follows:

  - `cutoff` - A value used to select response coefficients on the basis of absolute value.

  - `flux` - A list of flux or species labels, to be used as response coefficient selection criteria.

  - `param` - A list of parameters for which the associated response coefficients are desired.

  When no arguments are provided, a default value of 0.5 is used as the `cutoff`, and all response coefficients are computed that have an absolute value greater than 0.5. Two further methods are required to compute the symbolic response coefficients, both of which are called within this method. The list of flux and species is stepped through and for each entry all associated response coefficients are extracted from `PySCeS` in the form of a dictionary. This dictionary along with the `cutoff` value are passed to `selectCoefficient`, which selects the response coefficients based on the value provided, after which a dictionary is returned containing only those response coefficients selected. A dictionary is created with the following key value pairs, flux or species label and dictionary of all selected response coefficients. Once all flux or species elements in the list have been processed and their associated response coefficients computed and added to the dictionary, it is then passed to the `getResponseElasticity()` method.

  - `selectCoefficient()` - This method takes a cutoff value and dictionary of response coefficients as arguments. The dictionary is stepped through and in the event of the absolute response coefficient value being less than the cutoff value, the item is removed from the dictionary. Once all entries have been processed the dictionary is returned.
  - `getResponseElasticity()` - A dictionary with the following key value pairs, reaction or species label and dictionary of all associated response coefficients, is the only argument. The control coefficient and the elasticity coefficient which when multiplied give the desired response coefficient, are then determined and the data pertaining to a response coefficient is then stored in dictionary. Once all response coefficients have been processed, the dictionary containing all data is stored in a global dictionary.

- `computeCoControl()` - Symbolic co-control coefficients can also be computed, and this is achieved by providing a list of data describing the co-control coefficient as an argument. The data in the list is a three-membered tuple with the following elements: a flux label, a species label and the reaction concerned. This data encodes the two control coefficients involved in the generation of the co-control coefficient. The algebraic co-control coefficient is then obtained by dividing the two control coefficient expressions. The data is then stored as a `CoControlCoefficient` object and each object is stored in a dictionary with the co-control coefficient label as the key. This dictionary is a global attribute.

A.7 Maxima

The details of the methods developed for the `Maxima` interface follow below:

- `__call__()` - On instantiation of the `Maxima` class, this method is called to establish a connection with `Maxima` using the `subprocess` module, which is native to all `Python` distributions. The `getMaxima()` method is tasked with establishing this connection, and once established the `subprocess Maxima` instance is returned. The 'linearalgebra' package is then loaded into `Maxima`, and symbolic computations can now be performed.
- `getMaxima()` - This method is called by `__call__()` on instantiation of the `Maxima` class, and is responsible for the creation the `Maxima subprocess` instance. The first step in this process is to determine which platform is being used, where the platform is either Windows or Linux. This is achieved with the native `Python os` module, once determined the `Maxima` argument is created (in the case of Windows the argument is `'maxima.bat'`, and for Linux it is `'maxima'`). This argument is then used by `subprocess` to connect to `Maxima` from within `Python`. If the connection is successful the `Python subprocess Maxima` instance is returned, but in the event of it failing a `RuntimeError`, is raised. The `RuntimeError` can arise from two situations, the first being that `Maxima` has not been installed, and the second that the system path has not been updated to include the `Maxima` path.
- `loadPackage()` - This method takes a `Maxima` package to be loaded in string format as its only argument and loads the provided package, by using the `Maxima load` function.
- `input()` - In order to send executable arguments to `Maxima`, the `input` method was developed. This method requires the `Maxima` input argument as an argument, and firstly checks the integrity of the provided argument by calling the `maxima()` method. If the argument is correctly formed the `compute()` method is called which inputs the argument into `Maxima`, and extracts any resulting data.
- `maxima()` - The integrity of provided `Maxima` input arguments are checked by this method, which requires the input argument when called. The argument is then passed to `checkInput()`, which ensures the correct format of the input argument. Once the input has been checked a `MaximaInput` object is created and returned. The input is then ready to be evaluated by `Maxima`.
- `compute()` - The `compute` method sends the input argument, provided as an argument, to `Maxima` for computation via the standard input pipe provided by the `Maxima subprocess` instance. Any output generated by the computation is then obtained via the `getMaximaOutput()` method. It should be noted that not all input arguments result in output, since the sign ending the argument determines whether or not the output is suppressed or not. '$' indicates that output is suppressed (None is returned), whereas ';' indicates that any output is accessible via the standard output pipe.

- `getMaximaOutput()` - This method reads any output generated via the standard output pipe, and the data is then passed to `checkOutput`, which checks to see if an error has occurred during any matrix manipulations due to incompatible matrix dimensions. Once checked the output is returned.
- `invert()` - The computation of $\mathbf{C^i} = \mathbf{E^{-1}}$, is performed by this method, which requires an optional argument representing the label of matrix to be inverted. If no argument is provided, the label for the last data entered into `Maxima` is used. The `Maxima` input argument is then created which is passed to `compute` in order to perform the computation. Since the data arising from this inversion can be extremely large the input argument is ended by the '$' symbol, which suppresses the `Maxima` output. The native `Maxima invert` function is used, which computes the inverse by the adjoint method [8].
- `determinant()` - `Maxima` computes the determinant of any given matrix. This method has two optional arguments, the label of the matrix from which the determinant is required and the label to be assigned to the determinant. The `Maxima` input argument is generated and passed to `compute()`, which returns the output.
- `matrixSize()` - This method uses a native `Maxima` function to determine the dimensions of any matrix entered into `Maxima`. An optional argument representing a matrix label is required, and when not entered, the label of the data last entered into `Maxima` is used. A `MaximaOutput` object is returned containing the matrix dimension data.
- `differentiate()` - Since all elasticities are partial derivatives, they can be computed symbolically. `Maxima` has a differentiation function, `diff`, which returns the derivative or differential of an expression with respect to some or all variables in the expression. The label assigned to the elasticity being computed is returned.
- `expand()` - This method wraps the `Maxima expand` function, which expands a given expression in the following manner. Products of sums and exponentiated sums are multiplied out, numerators of rational expressions which are sums are split into their respective terms, and multiplication (commutative and non-commutative) is distributed over addition at all levels of the expression. This method takes the label of the expression to be expanded as an optional argument, otherwise the label of the last entered data is used.
- `fileAppend()` - `Maxima` has the ability to append any data written to a file, this is governed by the value of an internal `Maxima` variable. This method sets the value of this variable to either True of False. The desired status is passed as the only argument required by this method. The `setFlag()` is called to affect the change in the variable value.
- `setFlag()` - There are numerous flags which can be customised within a `Maxima` session. This method takes the maxima flag variable label and desired status as arguments and resets the flags value.

- `string()` - The string form of a `Maxima` expression is computed by calling this method, which in turn calls the `Maxima string` function. The label of the expression, a label to be assigned to the string form and the boolean determining whether or not the output is to be suppressed are provided as arguments. Once `Maxima` has generated the string form of a desired expression, a `MaximaOutput` object is returned.

- `stringout()` - Because the size of the control coefficient expressions often prevents their extraction from `Maxima` via the standard output pipe associated with the `subprocess` instance, all data is written directly to file. `Maxima` contains a method, `stringout`, which writes expressions to a file in the same form the expressions would be typed for input. The name of the file is required as well the label of the data to be written. The full path indicating the written files location is then returned if the data has been successfully written to file.

- `substitute()` - This method wraps the `Maxima subst` function to perform one off substitutions. Four arguments are required with the last two being optional.

  - `in_sub`: item to be substituted into the expression.
  - `out_sub`: item to be substituted out of the expression.
  - `label`: label of the expression to be substituted, if no label is provided the label for the last data entered into `Maxima` is used.
  - `var_label`: label for resulting substituted expression, if none is provided this label is the same as the expression to be substituted.

  The `Maxima` substitution argument is generated and the substitution is performed.

A.8 Parser

The methods within the `Parser` class are outlined below:

- `parseOut()` - The `Maxima` expressions as outputted by the `string` function are handled via this method, which takes the raw data as its only argument. All newline characters are removed, such that the expression is a single continuous string. The mathematical symbols (*, /, +, -, (,)), found in the expressions are all found and padded with white space, this is an important step as these padded 'markers' are necessary in a number of later steps. The parsed string is then returned.

- `parseStringOut()` - The output associated with the `Maxima stringout` function is parsed by this method. The `Maxima stringout` output is the only argument for this method. The `Maxima` output contains an output identifier (e.g. %o3) followed by the path to where the data has been written when `stringout` is successful. The identifier is then stripped off the returned output so that it may be compared with

the directory specified. This match determines whether or not the data has been written to disk.

- `parseMaximaStr()` - When a string is extracted from `Maxima` via the `subprocess` standard output pipe it can contain numerous `Maxima` input and output identifiers, which need to be removed from the string. The identifiers as well as newline and tab delimiter characters are stripped from the string, which is then returned.

- `parseMatrixStr()` - In certain instances the data pertaining to the various matrices involved in `SymCA` are accessed in string form. This method takes the matrix string form as obtained via the standard output pipe, the `Maxima` input and output identifiers are stripped as well as any newline characters. The string is then returned.

- `parseSize()` - When `Maxima` computes the dimensions of a matrix the output is read over the `subprocess` standard output pipe and then passed to this method, whereby the `Maxima` identifiers are stripped and the and the row and column indices isolated. A tuple is then returned, which contains a list where the first element is the number of rows and the second the number of columns.

A.9 Output

Methods involved in the output of data are as follows:

- `getFile()` - This method takes two arguments, a file name or file path, and a type which indicates whether or not the first argument is a filename or a file path. A file handle is then opened to which the LaTeX output is written. True is returned once the file is open and the file handle is assigned to a global variable `__out_file`, in the event of an error occurring on opening the file False is returned.

- `latexHead()` - A standard LaTeX header is required by all files generated, this header contains all the packages required by the LaTeX compiler as well as any custom commands to be used. This method returns this header when called.

- `writeHeader()` - This method takes an optional argument, `'name'`, which is the label for the output file. The LaTeX header from `latexHead()` is obtained and written to the open file, after which the details of the file being written as well as a timestamp are written to the file.

- `writeData()` - All forms of data output are handled via this method, which requires six arguments when called. The first four arguments are mandatory whereas the last two are optional:

  - `data_type`: substring denoting which type of data is to be written:
    - 'cc': symbolic control coefficient expressions.

    * 'sym2psc': `SymCA` derived control coefficient values compared to those obtained with `PySCeS` in a table.
    * 'patterns': quantified control pattern data in table form.
    * 'summation': summation theorem data in table form.
    * 'coc': symbolic co-control coefficient expressions.
    * 'scan': parameter scan quantified control pattern data.
  - `sub_data`: tuple containing all elasticity substitutions performed if necessary else None.
  - `cc_objects`: `SymcaObjects` containing relevant data objects corresponding with data to be written.
  - `all_cc`: list of all control coefficient names.
  - `cc_list`: list of specific control coefficients to be outputted, argument only works for 'cc' and 'controlPatterns' data types.
  - `file`: file to which data is to be written, else data is written to the latex subdirectory.

  This method controls the LaTeX output for all types of data, and calls the respective methods involved based on the type of data output required.

- `writeCCData()` - This method is called for type 'cc' data and is responsible for generating and writing to file all control coefficient expression data. Five arguments are required; elasticity substitution data, common denominator object, control coefficient objects, a list of all control coefficient names and finally a tuple with file names for the common denominator, flux control and concentration control coefficient LaTeX files. The `prepDenom()` and `prepCC()` methods handle the common denominator and control coefficient LaTeX respectively.

  - `prepDenom` - This method takes the `Denominator` object as its only method and generates the LaTeX for the common denominator expression. If any elasticity substitutions have been performed, all substituted denominator expressions are also converted into LaTeX and included in the output. Once the LaTeX denominator string has been created it is returned.
  - `newPrepCC` - The LaTeX output for all control coefficient is handled via this method which takes a dictionary of all control coefficient objects, a 2D list of all control coefficient names and the type of control coefficients being processed, either 'Flux' or 'Concentration', as arguments. The control coefficients are divided into two sections, one for the independent control coefficients and the other for the dependent control coefficients. As per the common denominator, if any elasticity substitutions have been performed the resultant data is also converted into LaTeX and included in the output file. In the same light, if numerator and denominator terms have been factorised together, the resulting simplified expressions are converted into LaTeX via `prepSimplified()`.

Once all LaTeX control coefficient data has been generated it is returned as a string.

* `prepSimplified()` - This method is only called if numerator and denominator terms have been factorised together, and the `ControlCoefficient` object is the only argument required. The LaTeX associated with the simplified control coefficient is created and returned as a string.

- `prepSym2Psc()` - Once a comparison between the numerically substituted symbolic control coefficients and the values obtained by numerical analysis with `PySCeS` has been performed, the resulting dictionary of data is turned into a LaTeX table summarising this data. The dictionary of comparison data is the only argument, with the resulting LaTeX string being returned once generated.

- `writeControlPatternData()` - This method is responsible for generating the LaTeX data displaying all quantified control pattern. Four arguments are required, with the fourth being optional. These are a dictionary of control coefficient objects, a 2D list of all control coefficients, a tuple containing file names for the flux and the concentration control coefficient control patterns and finally the optional list of desired control coefficients. If no list of control coefficients is provided the quantified control pattern data for all computed control coefficients is generated. This method utilises the `prepControlPatterns()` method to generate the LaTeX, which is then returned. The returned LaTeX is then written to file via the `writeFile()` method. This process is repeated twice, once for all flux control coefficient control patterns and once for all concentration control coefficients.

  – `prepControlPatterns()` - This method is called by `writeControlPatterns()`, and requires four arguments. A 2D list of all control coefficient labels, a dictionary of all control coefficient objects, a list of control coefficients to be processed and the type of control coefficient being processed, 'Flux' or 'Conc'. Each control coefficient is processed in turn, and where control patterns have been quantified, `prepObjectPatterns()` is called which returns the LaTeX. Once all control coefficients have been processed the string containing all LaTeX is returned.

    * `prepObjectPatterns()` - A `ControlCoefficient` object is passed, when called the LaTeX table summarising the quantified control pattern data is created and returned.

- `prepSummations()` - All summation data is converted into a LaTeX table for display by this method, which requires a tuple containing the flux control coefficient summation and the concentration control coefficient summation data as its argument. The LaTeX output generated is split over two sections, one for all flux and the other for all concentration control coefficient summation data. The final LaTeX string for all summation data is then returned.

- `prepCoControl()` - This method requires a dictionary of all computed co-control coefficient data and generates the LaTeX for all co-control coefficients, which is returned as a string.

- `prepScanPatterns()` - A dictionary of all control coefficient objects, a 2D array of all control coefficient labels, a tuple with file names for both flux and concentration control coefficients, as well as a list of desired control coefficients for which the output is desired, are the arguments required by this method. If the latter argument is set to None, all control coefficients are handled. The control coefficients are then split into independent and dependent for both flux and concentration control coefficients. The coefficients are then processed one group at a time by the `prepParamPatterns()` method which in turn calls the `prepPatternTable()` method. All desired LaTeX is created and returned as a string, whereby it is written to the designated files.

  - `prepParamPatterns` - This method is called by `prepScanPatterns()` and takes the parameter scan data generated for a single control coefficient, the coefficient name and the `ControlCoefficient` object as its arguments. Two plots are then generated, one for the control pattern quantification data over the parameter range and another for the control coefficient value over this range. These plots are saved to disk and then included in a LaTeX file. A table which serves as a key for the first plot is generated via the `prepPatternsTable()` method, after which the LaTeX string is returned and written to file.

    * `prepPatternsTable()` - This method requires the `ControlCoefficient` object as its only argument and generates the LaTeX for a table depicting the control pattern labels and the actual control pattern expression. The LaTeX string is returned.

- `writeFile()` - This method takes a LaTeX output string and a file name as arguments. The file is opened and the LaTeX header written, after which the LaTeX output is written and the file closed.

A.10 Utilities

This class houses many of the common methods used by a number of the classes in `SymCA`, and its details follow:

- Methods used by more than one class:

  - `loadPickle()` - Takes a pickle file as an argument and loads and returns the data from the pickle.
  - `writePickle()` - Pickles a `Python` data object to a specified file. Three arguments are required, the data to be pickled, the file path for the pickled

data with third (`type`) being optional and determines whether or not the data is pickled in binary format. If `type`='binary', a binary pickled is created, but when no argument is provided the standard ASCII protocol is used.

- `getCCElas()` - Takes a control coefficient expression as an argument and returns a list containing all elasticities found in the expression.
- `getCCList()` - This method takes two arguments, a list of all control coefficient names and a list of flux or species names. The list of flux or species names is stepped through and all control coefficients for the particular flux or species are found and stored in a list. Once the control coefficients have been isolated for flux or species in the list provided, a list is returned.
- `getTerms()` - This method takes a control coefficient expression as an argument and returns a tuple of data pertaining to all control patterns in the expression. The first step is to isolate all control patterns within an expression, this is achieved by using either the + or − to separate individual control patterns. The following data is generated: a list of all control patterns, a list of all control pattern signs (+ or −) and finally a list of corresponding absolute control patterns. This data is passed to `orderTerms()`, which will rearrange each control pattern. Once all control patterns have been rearranged the rearranged lists described earlier are returned.
  - `orderTerms()` - The initial lists of control pattern data are required as the three arguments required by this method. Elasticities are first ordered alphanumerically by species, and second by reaction. All fluxes are placed at the beginning of each control pattern and species are rearranged in ascending alphanumeric order. `orderFlux()` is called when ordering the fluxes present in a control pattern and this method takes a control pattern as its argument. Fluxes are isolated and then rearranged in ascending alphanumeric order, this list is then returned.
- `getLatex()` - The tuple containing the list of all control pattern terms, the corresponding list of all term signs, and the list of all absolute terms is passed to this method. The list of absolute terms is then stepped through and all elasticity, flux or species labels are replaced with their LaTeX equivalents, after which the sign associated with the term is added to the beginning of the term. The LaTeX term is then stored in a list. Once all LaTeX control pattern terms have been generated the list is passed to `buildLatexEqn()`, which builds and returns the LaTeX expression.
  - `buildLatexEqn()` - A list of LaTeX control pattern terms serves as the argument for this method. The full LaTeX expression is generated ensuring that there are no more than 300 characters on any given line. If the number of characters on a line were greater than 300, then part of the expression would run off the page. The LaTeX expression is then returned.
- `getAllElements()` - This method takes a control pattern term as an argument and isolates all elasticities, flux and species found in the term. A

three-membered tuple is then returned containing this data.

- `getRxnCCDict()` - A list of all control coefficient names is given as the argument, and a dictionary containing reaction names as keys and a corresponding list of control coefficients affecting the reaction as its value. This dictionary is returned after all control coefficients have been processed.
- `Array2List()` - This method takes a 2D array of all control coefficient labels, and generates a list of flux control coefficients, a list of concentration control coefficients and a list of all control coefficients, which are returned as a tuple.
- `latexSubArg()` - This method is used to create a LaTeX string representing all elasticity substitutions performed. The dictionary of all elasticity substitutions is the only argument required, and the resulting LaTeX string is returned.

- Methods specific to the `Data` class:
  - `list2Dict()` - Takes a list of `SymcaObjects` as an argument and generates a dictionary where the objects names are the keys and their corresponding objects are the values. A tuple containing a list of all the dictionary keys and the dictionary, is returned.
- Methods specific to the `PyscesData` class:
  - `pysces2Maxima()` - This method takes a `PySCeS` `numpy` data array and a data title as arguments and generates and returns the `Maxima` input argument using the data title as the assigned variable name.
- Methods specific to the `ProcessedData` class:
  - `getObject()` - This method requires a list of objects and the name of the object. The object with the name provided is then extracted from the list and returned.
  - `getIndex()` - This method takes an entry in a list and a list as arguments, and first checks if the entry is found in the list in which case its index in the list is returned.
  - `getElasParams()` - All parameters for a given elasticity expression are returned by searching for parameters present in the expression. The parameters found are stored in list and returned once the search is complete.
  - `pysces2List()` - A 2D `numpy` array is passed as an argument, and is converted into 2D list where each entry is a string as opposed to an integer. The 2D list is then returned.
- Methods specific to the `SymbolicData` class:

  - `buildSymbolicMx()` - Three arguments are required when calling this method, type of matrix being built ('J', '1/J', 'S', '1/S'), list of data for the diagonal and finally a label for the matrix. An initial identity matrix of size number of data entries is created, and populated by calling `populateDiagonal()`. The string form of the matrix, the `Maxima` input argument and the matrix dimensions are returned.
    * `populateDiagonal()` - This method populates the diagonal of a square matrix with data provided. The data is entered in the order that it is found in the list.
  - `fluxRelations()` - This method performs substitutions in the $\mathcal{K}$ matrix based on the flux relations present in the system. A dictionary of all flux relations as well as the label for the matrix to be substituted are the arguments. All fluxes with common relations arising from both linear and branched segments of the system, can be substituted out. All dependent fluxes substituted in are stored, as are all fluxes arising from branches. Once all substitutions have been performed the status (True or False), newly substituted string form of the matrix, the list of dependent fluxes substituted in and a dictionary of all branch fluxes are returned.
  - `compareMxDim()` - This method compares the dimensions of two matrices, which are provided as arguments. True is returned when the dimensions match and False when they do not.
  - `maxima2List()` - Takes a `Maxima` input argument as the only argument, and converts it into a 2D list which is returned.
  - `getElasticities()` - This method takes a list of all `Reaction` objects and generates a tuple containing all variable elasticity data, which contains a list of all variable elasticities, a list of all variable substrate elasticities, a list of all variable product elasticities and a list of variable modifier elasticities, as well as a tuple containing a list of parameter elasticities. The tuples of variable and parameter elasticity data are then returned.

- Methods specific to the `SymcaObjects` class:
  - `subArg2Dict()` - This method takes a `Maxima` argument for multiple substitutions as an argument and returns a dictionary where the keys are the elasticities to be substituted and the values are the numerical values to be substituted in.
  - `getTermValue()` - This method computes the value and percentage contribution for a control pattern with respect to its control coefficient. The control pattern term, `Maxima` substitution argument, control coefficient and denominator labels are required as arguments. The percentage contribution to the overall control coefficient is computed as well as the value for the control pattern. The contribution and control pattern value are returned as floats.

  - `values2Maxima()` - This method generates the `Maxima` substitution argument for all `PySCeS` steady-state values. This data is stored in a dictionary which is passed to this method as an argument. The `Maxima` input is generated and the resulting string is returned.

- Methods specific to the `SCA` class:

  - `addSubCC()` - The two arguments required for this method are a dictionary of all `ControlCoefficient` objects and a dictionary of all `Coefficient` objects resulting from elasticity substitutions. The key value pairs of the substituted dictionary are iterated over, and if the key is present in the first dictionaries keys, the value representing the substituted `ControlCoefficient` object is initialised to the `ControlCoefficient` objects 'substituted' attribute in the main control coefficient dictionary. The main dictionary of all coefficients is returned.
  - `dict2SubArg()` - A dictionary of variable names and their values is provided as an argument, which is used to create the `Maxima` substitution input argument. This string is returned.
  - `getCombinations()` - This method accesses the global variable containing a list of all unique combinations of elasticities, and creates a dictionary where each key value pair represents a list of the elasticities in the combination and the `Maxima` substitution argument for the combination. Once populated this dictionary is returned.
  - `splitCCArgs()` - In order to compute both the dependent flux and concentration control coefficients, the $\mathbf{C^{J_i}}$ and $\mathbf{C^{S_i}}$ matrices must be extracted from $\mathbf{C^i}$, the $\mathbf{I}$ and $\mathcal{K}_0$ matrices from the $\mathcal{K}$ matrix and the $\mathbf{I}$ and $\mathcal{L}_\mathbf{0}$ matrices from the $\mathcal{L}$ matrix. This method is responsible for this process and takes the maxima input argument for the matrix to be split as well as the label for the matrix, e.g. `cc` for the $\mathbf{C^i}$, `ks` for the $\mathcal{K}$ matrix and `ls` for the $\mathcal{L}$ matrix. The submatrices are then extracted and returned in a tuple with the following element pairs: $\mathbf{C^{J_i}}$ & $\mathbf{C^{S_i}}$, $\mathbf{I}$ & $\mathcal{K}_0$, and $\mathbf{I}$ & $\mathcal{L}_\mathbf{0}$ for the $\mathbf{C^i}$, $\mathcal{K}$ and $\mathcal{L}$ matrices respectively.
  - `getSpecies()` - This method isolates all species/metabolites found in any conserved sums present in the model. The conserved sum data generated by `PySCeS` is extracted and passed as the only argument, with the resulting list of all species found in the conserved sums being returned.
  - `checkSign()` - This is a key method, in that, as the name suggests it is responsible for checking the signs associated with the generated control coefficients. A list of all terms found in a control coefficient is required as its only argument. This list is stepped through until a term is found containing elasticities, once found the computed sign as generated by `SymCA` is extracted for the term. The theoretical sign is then determined on the basis that product

elasticities are negative and substrate elasticities are positive. If the computed and theoretical signs are identical, True is returned, whereas False is returned in the event of these being different.

- `assignEntry()` - Since each individual control coefficient is to be extracted from the computed **C** matrix, this method was developed to handle this process. The maxima label for the **C** matrix and a 2D list of all control coefficient labels associated to the **C** matrix are the required arguments. The 2D array is processed element by element, where each element is in turn extracted from the **C** matrix. Once extracted the control coefficient label is appended onto a list and its corresponding index in the **C** is also appended to a list. These lists are then returned in a tuple.
- `dataToArray()` - All control coefficient data in terms of the `PySCeS` numerical value and the value computed from the expressions generated by `SymCA` are stored in a dictionary, which is the only argument for this method. This data is extracted for all control coefficients into three lists: the first being a list of control coefficient labels, the second a list of corresponding `PySCeS` values and the third a list of corresponding `SymCA` values. These lists are returned as a three-membered tuple.

- Methods specific to the `Parser` class:
  - `matchSub()` - This method takes two arguments, the first being a list of all characters or terms to be removed from a string and the second the string from which these items are to be removed. The list of all characters or terms is stepped through and at each iteration the desired element is removed from the string by replacing it with an empty string. The newly edited string is returned once all undesired elements have been removed.

- Methods specific to the `Output` class:
  - `str2Latex()` - In order to generate the LaTeX output, all control coefficient and elasticity coefficient labels need to be in LaTeX. This method requires two arguments, with the first being the coefficient label as a string and the second the LaTeX label for the coefficient, i.e. `ec` for elasticity coefficients and `cc` for control coefficients. Once converted the LaTeX for the specified control coefficient is returned.
  - `getLatexTerm()` - To produce the LaTeX representations for a control coefficient expression, this method was developed. This method takes a term and a dictionary of all LaTeX labels for species, flux and elasticity data as the arguments. Once all species, flux and elasticities present in the term are converted to LaTeX, the full LaTeX term is returned.

A.11 Visualise

The methods involved in all aspects of the visualisation process are as follows:

- `getFiles()` - This method is required to generate the XML and SVG layout files for the model in question, if no layout files can be located. Once generated these files are stored in the `/layout` directory which is located within `pscout` in the `PySCeS` working directory.

- `reactionData()` - Since all reactions are labelled numerically for the `SymCA` generated graphics a HTML page with information pertaining to all reaction labels is created. This method creates the reaction HTML page for the model in question and stores it in the appropriate directory within the `layout` sub-directory. The page is opened in a browser by clicking on any reaction node in the network.

- `SBMLData()` - The initial XML layout file is parsed using the `ElementTree` parser, and all elasticity and reaction is data is obtained. The data for each individual elasticity and reaction are stored in the `SBMLelasticity` and `SBMLreaction` objects (as summarised in Tables 3.4 and 3.5, respectively) and stored in two dictionaries, one for all elasticities where the elasticity labels are keys and the corresponding objects the values. The reaction data is stored in a separate dictionary in the same manner as that of the elasticities. These dictionaries are then stored as class attributes, and True is returned.

- `checkAlias()` - This method was introduced to fix a bug on the SBW SVG rendering module, whereby if one used the auto alias function, the labels for the aliased species were removed from the layout information. Thus this method serves as a check to ensure that all species are labelled. In the event of the species labels being absent a dictionary of all absent species is created containing information about the role of the species, i.e. substrate or product. This dictionary is then initialised to one of the class attributes.

- `getSBMLdetails()` - The XML file is read as a normal text file and all information regarding the species style and text styles are extracted and stored in two separate lists. This data is stored as class attributes for later use.

- `getCoords()` - The co-ordinates for all species are extracted from the SVG layout file and stored in a dictionary where the keys are species labels and the values represent the data for the specific species, i.e. the x and y co-ordinates as well as whether or not the species is fixed or not. This dictionary is returned to the user.

- `checkSVGEncoding()` - The encoding of the SVG layout file needs to be utf-8, and in certain instances this encoding may be utf-16. This method ensures that the encoding is correct, and when incorrect the encoding is changed so that it is always utf-8. Once completed, True is returned.

- `SVGHeader()` - This is a basic method that extracts the XML header from the SVG file, and once extracted it is returned as a string.

- `getElas()` - This method takes a curve identifier and a role as arguments, where the role can be a substrate, product or modifier. The elasticity object matching the curve identifier and the role is then found in the dictionary of all `SBMLelasticity` objects and returned as a key value pair.

- `speciesMidpoint()` - The midpoints for each species found in a network is required for future operations, and this method serves to compute the x and y co-ordinates for a given species. Once computed this data is stored in a class attribute in the form of a dictionary.

- `getPatternCoords()` - As one of the outcomes of this class is to map control pattern data onto the network, we needed to create an elasticity representation for all elasticities such that the control patterns can be visualised. This method is responsible for determining the co-ordinates for these graphics, and the co-ordinates are affected by the nature of the elasticity, i.e substrate, product of modifier. Once the co-ordinates have been computed they are returned as SVG code.

- `elasSignSVG()` - Another feature of this class is to enable one to visualise all elasticities on a given network, and the SVG code associated with displaying the sign ($+$ or $-$) associated with each elasticity is generated by this method.

- `getModelBlurb()` - This method generates the SVG code depicting all model labels that are to be included on the final graphics produced.

- `colourRange()` - A number of the graphics generated make use of a colour key where each colour indicates a specific value range. This method generates a dictionary of ten colours where the range is either from $0 - 100$ or based on a value provided as an argument when calling the method. The colour range dictionary is then returned where colours serve as keys and their respective range is the value.

- `elasColours()` - The mapping of the colours from the colour dictionary to each elasticity, is performed by this method. Each elasticity colour is determined by its numerical value which is then assigned to its corresponding colour range, a dictionary containing all elasticities and their colour is then returned.

- `elasSigns()` - The sign for each elasticity is determined based on its value, this data is stored in a dictionary with key value pairs being the elasticity name and the elasticity sign (+ or −). This dictionary is returned once all elasticity signs have been determined.

- `colourRangeGlyphs()` - In order to display the colour range the appropriate SVG code must be generated, which is handled by this method. Two strings are returned are, one containing the SVG for the actual colours to be displayed and the other for the corresponding numerical range associated to each colour.

- `insertBackground()` - Given a height and a width as arguments, the SVG encoding these dimensions is generated. The SVG file is read and appended onto the newly generated SVG code, and the new SVG is written to file for future use and customisation.

- `xy()` - The `xy` method is used by the `prepAlias` method to compute the x and y co-ordinates for the species that have been affected in the process of performing the auto alias function mentioned previously. The co-ordinates are generated taking into account whether or not a species is a product or a substrate, and once generated the co-ordinates are returned.

- `prepAlias()` - The SVG code for each alias that needs to be added to the SVG layout is generated by this method, and once generated is returned.

- `addAlias()` - Any additional species labels required due to auto aliasing are added to the SVG layer and the file is then saved for future use.

- `customiseSVG()` - This is the main method governing the customisation of the SVG layout as generated by SBW. The SVG containing all additional information required in future visualisation operations is added onto the initial SBW SVG file by this method. Once all information has been added a base layout SVG file is created, which serves as the master template.

- `getSVGelements()` - Like the previous method, this is also a central method in that it parses the SVG layout file, extracts all required SVG data and initialises several class attributes.

- `writeSVG()` - In its simplest form, this method allows one to write the customised SVG layout file to disk.

The following methods are all cosmetic in nature, in that they allow the user to set the colour of the species and reaction labels as well as determine the text to be used for the labels. The user can also manage the signs associated with the elasticities, the control pattern image details, those for the colour range as well as setting the text for the image title and the control coefficient legend.

- `setSpecies()` - The colour of the species label as well as whether or not a species label is to be underlined is determined by this method. A dictionary containing all required information pertaining to the colours is passed as an argument. This dictionary is stepped through and each species present is processed.

- `setSpeciesLabels()` - As the method name suggests, this method is concerned with setting the colour associated with a species label, the species label to be affected and the desired colour are required as arguments.

- `setElasticity()` - Information displayed pertaining to elasticities includes the colour of the elasticity as determined by its absolute numerical value and the sign based on its numerical value. A dictionary containing all this data is required as an argument, which is then stepped through and changes are incorporated into the final image.

- `setPatternElasticity()` - Since one of the outcomes of the `visualise` class is to display quantified control patterns, this method takes a dictionary similar to that used in the previous method, with data for only those elasticities found in the control pattern under investigation. The desired control pattern is then displayed as per the data provided.

- `setReaction()` - This method governs the final colours associated with either the square or circle used to denote a reaction. A dictionary of all reactions is provided and the desired changes are made.

- `setReactionLabels()` - The colour of the text used to denote a specific reaction is set using this method.

- `setControlPatterns()` - The details and colours used to display a quantified control pattern are made via this method. Changes include the change of a reaction node to a square from the standard circle for the modulated reaction, all flux labels for the fluxes present in the control pattern change to the colour of the control pattern, and in the case of a control pattern for a concentration control coefficient the affected species is underlined.

- `setTitle()` - The title for the image is set by providing the desired legend as a string as an argument.

- `setCoeffTitle()` - This legend is set in the same manner as the main legend, and requires the desired text to be provided as a string as an argument.

- `setColourRange()` - The colour range used to differentiate the elasticities based on their absolute values as well the indicator for a control pattern value is set via this method.

It was also desired that the user could export the final images in a variety of formats other than the native SVG. In order for this feature to be available, a local installation of Inkscape (`www.inkscape.org`) [12] (an open-source vector graphics editor) is required. Provided Inkscape is installed one can export images in the following formats: EPS, PDF, PNG and PS. The methods, as shown below, require no arguments to function, however, if one would like to assign a specific name to the file as well as stipulate the directory where the image is to be exported the option is available. In the event of no arguments being provided the files names are of the form *model name_custom.eps/.pdf/.png/.ps*, where the final file extension depends on the format exported. The files are stored within the model sub-directory:
`$PSCOUT/layout/`*`model name`*`/`.

- `exportEPS(svg_file=None,file_out=None,dir=False)`,

- `exportPDF(svg_file=None,file_out=None,dir=False)`,

- `exportPNG(svg_file=None,file_out=None,dir=False)`,

- `exportPS(svg_file=None,file_out=None,dir=False)`.

Here `svg_file` refers to the file to be exported, `file_out` is the file name associated with the newly exported file and `dir` is the directory location for the newly exported file.