A flexible software environment for steady-state power flow optimization with series FACTS devices

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY (ETH)
ZURICH

for the degree of
Doctor of Technical Sciences

presented by
TINA ORFANOYANNI
Dipl. El.-Ing. NTU Athens, Greece
born February 7, 1973

accepted on the recommendation of
Dr. Rainer Bacher, examiner
Prof. Dr. Göran Andersson, co-examiner

Zurich, 2000
To my parents,
To Sonia,
To Dionysis
Preface

First of all I wish to express my deepest gratitude to my advisor Dr. Rainer Bacher for introducing me to the worlds of software engineering and power system optimization. The motivating remarks and valuable suggestions, his positive attitude, and his encouragement and patience have been a big help for me over the last four years. Further, I would like to express my gratitude to Prof. Hans Glavitsch who, together with Dr. Bacher, offered me the opportunity to work on this project. I thank him also for his insightful suggestions concerning this work. A special thanks goes to Prof. Göran Andersson for consenting to be the co-examiner and for providing a number of valuable comments and suggestions.

I also wish to thank my colleagues from the Power Transmission Lab for making work a pleasure. The coffee-breaks, the entertaining discussions and the group excursions will stay in my memory. A big thank goes to my colleagues from the Automatic Control Lab, and especially to Dr. David Farruggio, with whom it was a lot of fun to share the office with. Last but not least, Cornelius Dorn from the Automatic Control Lab who provided an excellent help with proof-reading and improving this manuscript quite a bit. I would also like to extend a special thanks to Dr. Dennis Jansen and Eva Virag-Jansen for their valuable advice and for being such good friends.

My deepest gratitude goes to my parents, my sister Sonia, my long-time friend Dionysis, and to Cornelius for their encouragement, their support and for putting up with me in general. It is to them that this work is dedicated.
Abstract

The current major restructuring of the electricity market and the advances of new technology add new complexities in the power system application software, particularly optimization. Minimization of the impact of a transaction on unrelated third-party facilities, or maximization of the power transfer between utilities must now be formulated as new constraints or objective functions. The dispatch and control of generating units, the taps of regulating transformers, mechanically switched capacitors and reactors have been the primary means available for controlling power flows over the years. However, certain applications require frequent operations that result in a rapid wearing out of the switching equipment. Other applications require smooth and continuous real-time control that is impossible with the conventional equipment. The diversity and the volume of the new market based transactions push the network elements closer to their limits. The inadequate line flow control may result in overloaded parts of the networks, while other parts are loaded far below their power carrying capacity. Therefore, the demands for flexibility in power flow control are growing continuously.

The FACTS (Flexible Alternating Current Transmission Systems), relying on thyristors, have high speed switching capability that allows for rapid readjustment of line power flows in response to various contingencies. Installation of FACTS devices on key locations in a meshed network changes the effective reactances of the parallel paths and consequently the way that power divides among them.

The motivation for this thesis is the need for flexible and expandable power system application software in the continuously changing electricity market. The focus of the thesis is the design and implementation of a prototype solver for power system optimization problems that considers the concepts mentioned above. The primary design goal was to use robust off-the-shelf research and commercial software tools that minimize the required hand coding and increase the code flexibility and maintainability. An automatic differentiation (AD) package is used to generate code for the first-order derivatives and take care of the sparsity pattern. Code generation relieves the program developer of the tedium and inevitable “bugs” normally associated with hand-coding. A general-purpose nonlinear optimization package requiring only first-order derivatives from the user is used for the optimization studies.

The thesis concentrates mainly on the description of the optimization software package and the steady-state modeling of the series FACTS devices TCSC (Thyristor Controlled Series Capacitor) and UPFC (Unified Power Flow Controller). The FACTS modeling equations illustrate how the flexibility of the software environment in use allows the easy combination of different objective functions, different sets of variables and different formulations of functions. Case studies demonstrate the operating regions of the series FACTS devices and their effectiveness in increasing the MW power transferability of a particular network.
Durch den gegenwärtigen Restrukturierungsprozess der Elektrizitätsmärkte sowie den Einzug neuer Technologien werden neue Anforderungen an die Software für Netzbe-rechnungen, insbesondere an die Optimierung, gestellt.


FACTS-Geräte, die auf Hochleistungshalbleitern aufbauen, besitzen die nötigen Schaltgeschwindigkeiten zur schnellen Anpassung des Leitungsflosses auf sich ändernde Anforderungen. Die Installation von FACTS-Geräten an Schlüsselpositionen in vermaschten Netzwerken kann die effektiven Reaktanzen parallel er Pfade ändern und damit die Verteilung des Lastflusses zwischen ihnen beeinflussen.


Die vorliegende Arbeit beschreibt vor allem den entwickelten Prototypen für den Optimalen Lastfluss sowie die Modellierung der seriellen FACTS Geräte im stationären
TABLE OF CONTENTS

Chapter List of Acronyms and Abbreviations v

Chapter 1 Introduction 1
  1.1 Evolving networks .......................... 1
  1.2 Transmission line loadability ............... 3
  1.3 Limiting factors for power transmission in a meshed network 4
  1.4 Series FACTS devices as measure to enhance transmission capability in meshed systems 6
  1.5 Optimal Power Flow .......................... 7
  1.6 Purpose of the work ........................... 8
  1.7 Overview ..................................... 8

Chapter 2 Automatic differentiation in power system studies 11
  2.1 Computational differentiation ................... 12
  2.2 Derivative code generation .......................... 13
  2.3 The technique of automatic differentiation ............... 18
    2.3.1 The Forward mode .......................... 22
    2.3.2 The Reverse mode .......................... 24
    2.3.3 ADIFOR tool - Hybrid approach .................. 26
  2.4 How ADIFOR works ................................ 29
    2.4.1 The ADIFOR 2.0 preprocessor ...................... 30
    2.4.2 The ADIntrinsics system .......................... 31
    2.4.3 The SparsLinC Library .......................... 32
  2.5 ADIFOR applied to the AC power flow algorithm ............ 33
    2.5.1 Step 1: Specifying Input for ADIFOR ................. 33
    2.5.2 Step 2: Incorporating the generated derivative code to the main power flow program .................. 35
  2.6 Software engineering and simulation results ................ 38
  2.7 Conclusions ................................... 40

Chapter 3 Flexible Optimal Power Flow Software Environment 41
  3.1 Introduction .................................. 41
  3.2 The OPF Problem .................................. 42
    3.2.1 Mathematical formulation ...................... 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2</td>
<td>Optimaliy conditions</td>
<td>43</td>
</tr>
<tr>
<td>3.2.3</td>
<td>The OPF problem solution</td>
<td>46</td>
</tr>
<tr>
<td>3.3</td>
<td>MINOS package</td>
<td>47</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Reduced Gradient method</td>
<td>48</td>
</tr>
<tr>
<td>3.3.2</td>
<td>MINOS: Brief description of the algorithm</td>
<td>50</td>
</tr>
<tr>
<td>3.3.3</td>
<td>The MINOS interface to the OPF problem</td>
<td>54</td>
</tr>
<tr>
<td>3.4</td>
<td>The SW environment structure: MINOS combined with ADIFOR</td>
<td>56</td>
</tr>
<tr>
<td>3.5</td>
<td>Adding features and modifying the structure</td>
<td>60</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Standardized steps to include a new constraint</td>
<td>60</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Standardized steps to include a new independent variable</td>
<td>62</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Standardized steps to include a new objective</td>
<td>62</td>
</tr>
<tr>
<td>3.6</td>
<td>Conclusions</td>
<td>65</td>
</tr>
</tbody>
</table>

**Chapter 4** Series-connected FACTS devices

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Compensation and line loadability</td>
<td>67</td>
</tr>
<tr>
<td>4.2</td>
<td>The Thyristor Controlled Series Capacitor (TCSC)</td>
<td>71</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Introduction</td>
<td>71</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Circuit analysis and operating principles of TCSC</td>
<td>72</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Two-port modeling equations</td>
<td>78</td>
</tr>
<tr>
<td>4.2.4</td>
<td>The two-port model of the TCSC</td>
<td>78</td>
</tr>
<tr>
<td>4.2.5</td>
<td>The two-port model of the TCSC-embedded transmission line</td>
<td>80</td>
</tr>
<tr>
<td>4.2.6</td>
<td>Integration of the TCSC-embedded transmission line two-port model into the MINOS/ADIFOR environment</td>
<td>82</td>
</tr>
<tr>
<td>4.3</td>
<td>The Unified Power Flow Controller (UPFC)</td>
<td>85</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Introduction</td>
<td>85</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Basic operating principles of the UPFC</td>
<td>86</td>
</tr>
<tr>
<td>4.3.3</td>
<td>The two-port model of UPFC</td>
<td>87</td>
</tr>
<tr>
<td>4.3.4</td>
<td>The two-port model of the UPFC-embedded transmission line</td>
<td>88</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Integration of the UPFC-embedded transmission line two-port model into the MINOS/ADIFOR environment</td>
<td>90</td>
</tr>
<tr>
<td>4.4</td>
<td>Conclusions</td>
<td>91</td>
</tr>
</tbody>
</table>

**Chapter 5** Case studies

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Maximization of MW power transfer in a two node network</td>
<td>93</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Maximization of MW power transfer with TCSC in the two-node network</td>
<td>95</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Maximization of MW power transfer with UPFC in the two-node network</td>
<td>98</td>
</tr>
<tr>
<td>5.2</td>
<td>Application of TCSC in a 37-node real-life network</td>
<td>104</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Case A: TCSC on branch #16 (AAA22-BBB22)</td>
<td>109</td>
</tr>
</tbody>
</table>
5.2.2  Case B: TCSC on branch #7 (WWW41-FFF41) . . . . . . . . 112
5.2.3  Case C: TCSC on branch #16 (AAA22-BBB22) and on branch
       #7 (WWW41-FFF41) . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
5.3  Application of UPFC in a 37-node real-life network . . . . . . . . 115
      5.3.1  Case A: UPFC on branch #16 (AAA22-BBB22) . . . . . . . . 115
      5.3.2  Case B: UPFC on branch #7 (WWW41-FFF41) . . . . . . . . 116
      5.3.3  Case C: UPFC on branch #16 (AAA22-BBB22) and on branch
            #7 (WWW41-FFF41) . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
5.4  Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117

Chapter 6  Summary and Conclusions 125

Chapter Bibliography 129
# LIST OF ACRONYMS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Alternating Current</td>
</tr>
<tr>
<td>ACD</td>
<td>Automatic Code Differentiation</td>
</tr>
<tr>
<td>AD</td>
<td>Automatic Differentiation</td>
</tr>
<tr>
<td>ADIFOR</td>
<td>Automatic Differentiation of Fortran</td>
</tr>
<tr>
<td>CSC</td>
<td>Controlled Series Compensation</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>FACTS</td>
<td>Flexible Alternating Current Transmission Systems</td>
</tr>
<tr>
<td>GTO</td>
<td>Gate Turn-Off</td>
</tr>
<tr>
<td>IGBT</td>
<td>Insulated Gate Bipolar Transistor</td>
</tr>
<tr>
<td>MINOS</td>
<td>Modular In Core Optimization System</td>
</tr>
<tr>
<td>SSR</td>
<td>Subsynchronous Resonance</td>
</tr>
<tr>
<td>TCR</td>
<td>Thyristor Controlled Reactor</td>
</tr>
<tr>
<td>TCSC</td>
<td>Thyristor Controlled Series Capacitor</td>
</tr>
<tr>
<td>UPFC</td>
<td>Unified Power Flow Controller</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The Kirchhof laws determine the pattern of the electrical network power flows. The ratings of the various network components and the operating state determine the maximum allowable power carrying capability (loadability) of the network elements. However, the configuration of the network can further limit the overall power transfer capability. Parallel lines or corridors of unequal ratings or impedances induce uneven loadings of the network transmission paths. The advanced power semiconductor technology can be used to modify the “natural” patterns of the network flows and obtain a more uniform loading of the network components.

1.1 Evolving networks

The high voltage transmission grid was originally built to connect generating plants with load centers. Growing demand pushes the electrical high voltage transmission networks to their physical limits, where outages of specific lines could result in the rapid failure of connected facilities. The electric utilities, that traditionally own the transmission system, have had mainly two options: upgrade the transmission facilities or extend the existing system by building new lines, thereby creating a highly meshed and more reliable network. To guarantee reliability the lines are operated at loadings far lower than their ratings. The generous margins represent reserve transmission capacity that makes sure that the system can recover from contingencies, like a generator outage or line tripping.

However, the major changes in the electricity market and the inevitable restructuring of the electric utilities modify or even eliminate some of the traditional aspects of the power industry: The vertical integrated utilities are now "unbundling" their operations and they supply power beyond a given geographical area. On the demand
side, the customers look for cheaper power supply, higher quality or better electrical service beyond their local providers. Independent power producers and other nonutility generation are allowed to sell electric power for any purpose, modifying the generation-demand location pattern in the network’s configuration and creating new transmission paths or transmission corridors.

Considering the volume and diversity of today’s transactions, it becomes obvious that the transmission networks are used for purposes other than bringing power from generators to loads: they permit sharing surplus generating capacity between neighboring utility systems, they provide energy reserves in case of emergency, or just wheel considerable amounts of power to other utilities. The extended use of the networks has the consequence of higher flows through the lines, that overload parts of the networks, while other parts are loaded far below their power carrying capacity. This reflects the inadequate influence of the power system operators on the way the power flows.

![Figure 1.1: Loop flows in transmission networks](image)

The nonuniform loading of the network is partly a consequence of the co-existence of networks of different voltage levels or transmission paths with different configurations. Electrical phenomena like parallel paths and loop flows are taken for granted, since they are a natural consequence of the fundamental laws of electricity (Kirchhoff, Ohm). The current divides along all available transmission circuits in inverse proportion to the relative impedances. In such an “inflexible” system, a simple transaction between the adjacent utilities A and B (Figure 1.1) can overload transmission facilities in C, a utility not involved in this transaction.
In certain electricity market structures of today’s power systems there is an urgent need to control the paths of the transmitted power. The ability to transmit power over defined line corridors without involving other partners or stressing heavily loaded parts of the network can increase the reliability of the system. Better controllability also leads to optimal use of the installed facilities. This is essential, considering that the power industry has significantly fewer options than before to meet the growing energy demand. Environmental concerns, high investment costs and long construction times make the installation of new overhead high voltage transmission lines an unattractive option. That creates the necessity to

- enhance the capability and performance of the existing facilities
- search for solutions that enhance the network flexibility and capability to meet the rapidly changing operating conditions.

1.2 Transmission line loadability

The basic relationship for power transmission is, for simplicity, limited to the two machine model shown in Figure 1.2, where $E_S$ and $X_S$ are the sending end generator internal voltage and reactance, $E_R$ and $X_R$ the receiving end internal generation voltage and reactance, $X_L$ the transmission line’s reactance (losses are neglected) and $\delta$ the transmission angle.

$$P_t = P_{t}^{\text{max}} \sin \delta$$

![Figure 1.2: Power angle curve and steady-state stability margin](image)

Any of the above variables or parameters have a direct impact on the power transmission $P_t$. With the idealized models used for the generators and transmission line, the power transmitted varies as a sine of the transmission angle and the steady-state
maximum is
\[ P_{\text{max}}^t = \frac{E_R \cdot E_S}{X_S + X_L + X_R} \] (1.1)

at a transmission angle of 90 degrees.

However, there are numerous factors limiting the power transfer capability of a transmission line in a meshed network. One obvious limitation is the current creating the maximum allowed temperature. For continuous rating this current value depends on the parameters of the line, ambient temperature, wind velocity etc. However, the transmission lines are usually operated (in continuous operation) at much lower levels than their thermal limits. The most important limiting factors on the loading of a transmission line are:

- The thermal limit
- The voltage drop limit
- The small-signal (steady-state) stability limit
- The transient stability limit

The usual value of the voltage drop limitation is 5\% [1], while the minimum allowable steady-state stability margin is 30\%. The maximum transfer \( P_t \) in Figure 1.2, for a steady-state stability margin 30\%, is 70\%\( P_{\text{max}}^t \) and the maximum allowable transmission angle is then
\[ P_{\text{limit}} = 0.7 P_{\text{max}}^t \Rightarrow \delta_{\text{limit}} \simeq 44^\circ \] (1.2)

For uncompensated lines longer than about 80 km the voltage drop limitation is the limiting factor on the line loadability, while for lines longer than 320 km the limiting factor is the steady-state stability limit [1]. In these cases the steady-state stability limit is far below the thermal limit.

### 1.3 Limiting factors for power transmission in a meshed network

Determining the maximum possible power transfer in a highly interconnected network is a complex task, involving many other aspects apart from the line loadability. Contingency studies determine maximum allowable line flows in normal operation of perhaps 50\% of the thermal limit, whereas during outages some of the elements in the system can be stressed to their thermal or short-time overload rating limit. Therefore, the different security criteria have a large impact on the normal (base case) operating limit [2]. Another factor limiting the transmission capability in normal operation
is the network topology [3]. In the following, a simple example will describe how the existence of parallel paths with unequal ratings can limit the maximum power transfer.

![Diagram showing two parallel transmission lines AB-1 and AB-2 with different ratings and flows.]

**Figure 1.3:** Maximum power transfer from Bus/Area 1 to Bus/Area 2. The two parallel paths have unequal ratings (see Table 1.1).

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Resistance (p.u. of $P_{max}$)</th>
<th>Reactance (p.u.)</th>
<th>Total Line charging (p.u.)</th>
<th>Rating (MVA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB-1</td>
<td>0.0019</td>
<td>0.0256</td>
<td>1.0700</td>
<td>2770</td>
</tr>
<tr>
<td>AB-2</td>
<td>0.0038</td>
<td>0.0304</td>
<td>0.9000</td>
<td>1385</td>
</tr>
</tbody>
</table>

Table 1.1: Per unit (p.u.) parameters for transmission lines AB-1 and AB-2 in Figure 1.3 calculated with $U_B = 500$ kV and $S_B = 100$ MVA

In Figure 1.3 the two transmission lines connecting buses or areas 1 and 2 are modeled with the parameters of Table 1.1. The line AB-1 has twice as high MVA rating for continuous operation as line AB-2. This case could refer to lines of same nominal voltage but with different conductor configurations. The flows along the two circuits are determined from the Kirchhoff and Ohm laws. The MW and MVAR flows as well as the loading in percent of the rating of the uncompensated lines are displayed in Figure 1.3 for both the sending and the receiving end. In this case the weaker path determines the maximum possible power transfer. The high rated circuit is only about 60% loaded at the maximum transfer. The dashed boxes in Figure 1.3 contain the kind of compensation that would increase the power transfer. Obviously, a fixed capacitor placed in series with the higher-rated circuit AB-1 creates a difference in the relative circuit impedances, that increases the flow along that circuit. Alternatively,
a series inductance (or a series voltage source creating an equivalent effect) could be placed in series with the lower-rated circuit AB-2.

1.4 **Series FACTS devices as measure to enhance transmission capability in meshed systems**

The two bus/area network in section 1.3 is a simple example showing how series compensating devices can be installed in order to:

- direct the flow along a desired path
- direct the flow away from a stressed region
- increase the loading of the lines up to their thermal limits

The controllability of power flows in general is a steady-state problem. Series compensation devices with fixed or mechanically switched capacitors or reactors have been used extensively in the past to deal with this problem. However, the conventional series devices could not achieve high levels of compensation. Subsynchronous resonance (SSR) considerations pose a maximum limit on the series compensation of the line impedance with conventional, mechanically switched capacitors. This limit could be as low as 40% of the line reactance. However, even a small percentage thyristor controlled compensation can mitigate SSR, and total series compensation can be increased safely [4, 5].

Advances in power semiconductor technology and power electronic equipment have provided a means for fast and smooth control of the three system parameters that have a direct impact on power flow: the bus voltage, the line impedance and the transmission angle. The Electrical Power Research Institute (EPRI) in collaboration with a number of host utilities and manufacturers has initiated the development of Flexible Alternating Current Transmission Systems (FACTS) [6, 7]. The principal objectives of FACTS are to increase the transmission capability of lines and to provide real-time steady-state control of the power flow. Considering the operating principles, there are mainly two categories of FACTS devices:

- Conventional Thyristor-Controlled Power Flow Controllers
  
  This type of FACTS devices consists of conventional capacitors or reactors banks. Thyristor switches with conventional thyristors with no turn-off capability have replaced the mechanical switches. They control the on and off periods of the fixed capacitor and reactors and thereby vary the capacitive and inductive effective impedance.
• Synchronous Voltage Source (SVS) Controllers

This type of FACTS devices can generate capacitive or inductive reactive power without capacitor or reactor banks [8]. The SVS is implemented by a multiphase inverter using a power semiconductor device with switch-off capability (like IGBT or GTO thyristors). The SVS can generate internally the reactive power necessary for compensation, and with an energy storage device can also negotiate real power with the system for a limited amount of time. The arrangement with two SVS is the Unified Power Flow Controller (UPFC), the most complex and versatile controller of all FACTS devices at present [9].

1.5 Optimal Power Flow

The ever increasing dependence of modern societies on a reliable electrical service makes the efficient and secure operation of the power grids the primary goal of power utilities. The Optimal Power Flow (OPF) software helps utilities to maintain reliable supply of electricity while maximizing profit and minimizing the total cost of operation.

Figure 1.4: Optimal Power Flow program

Figure 1.4 displays the OPF main input and output. In conventional OPF algorithms the objective is minimum cost or minimum MW losses operation. For a certain network the OPF identifies the various quantities as parameters, control variables and state variables. The larger the portion of the control variables is, the higher degree of freedom the optimization problem has. The most common control variables are the generators’ voltages, the generators’ power and the taps of the transformers.

The OPF problem is a nonlinear optimization problem. The objective functions (formulation of losses or generation costs) are nonlinear functions of the node voltages or the MW power of the generators. The largest part of the equality constraints consist of the nonlinear power flow equations and the Kirchhoff equations. The inequality
constraints represent limitations imposed by the operating capabilities of the installed equipment or by various quality criteria (see section 3.2.1). There are different solution methods proposed (see Chapter 3) for solving the OPF problem. After obtaining the OPF solution, the implementation of the optimal control variables will bring the system to the "optimum" state.

Installation of FACTS elements increases the network’s controllability. The TCSC introduces one new control variable, and the UPFC introduces three additional control variables. The new modeling and operating constraints must be formulated and incorporated in the OPF code, whereas the new derivative expressions with respect to the new variables have to be also integrated.

1.6 Purpose of the work

The purpose of this work is

- the development of a robust OPF software tool that is particularly flexible and easily extensible and maintainable, so that the integration of new models and new objective functions requires only modular modifications or additions. The resulting software is expected to be used in environments, such as research groups of universities, where software developers change often and “older” codes or parts of code are reused rather than rewritten. In such an environment the need for transparent and easy-to-comprehend programming is obvious, so that the user/developer can concentrate on modeling. The automatic code differentiation technique will be used to provide code for the first order derivatives.

- the integration of series FACTS devices in the OPF environment. The complete AC steady-state models have nonlinear functional constraints and nonlinearly used control variables that create a challenge to the programmer. This process involves only standardized steps from the part of the programmer, since certain parts of the code will be generated from an automatic code differentiation tool.

- performance of optimization studies with realistic network configurations to evaluate the effectiveness of the series FACTS devices with respect to maximizing power transfer. Different objective functions, sensitivity analysis for optimal siting as well as multiobjective optimization will provide useful numbers for sizing the installed devices.

1.7 Overview

The thesis consists of the following chapters:
• **Chapter 1**: Introduction

**Chapter 2**: Automatic differentiation in power system studies

This chapter describes the technique of automatic differentiation (AD) that generates code for first order derivatives of functions. The traditional hand coding of the derivatives of the power flow equations is compared against the AD technique and the relative merits and shortcomings are presented in detail. This chapter, further, describes the interface between the AD software tool in use and the power flow code.

• **Chapter 3**: Flexible Optimal Power Flow software environment

This chapter first outlines the optimal power flow (OPF) problem: what are the constraints and objective functions, control and state variables. Further, it describes the optimization software used, the solution method and the way the AD technique is integrated. Finally, the quality of the resulting software in terms of flexibility and maintainability is demonstrated by describing the steps required to include a new variable, constraint or objective function in the existing environment.

• **Chapter 4**: Series-connected FACTS devices

In this chapter, two of the main series-connected FACTS devices are described. First, the model of the Thyristor Controlled Series Capacitor is presented in detail. It belongs to the conventional thyristor-controlled FACTS devices. The TCSC model consists of the effective reactance with, in general, nonlinear constraint functions, that represent limitations on the firing angle, the TCSC voltage and current. Next, the steady-state model of Unified Power Flow Controller (UPFC) is presented. The UPFC has three independently controlled variables and can control the line real and reactive power independently.

• **Chapter 5**: Case studies

This chapter compares the TCSC and UPFC controllers’ impact on maximizing the loading of sample networks. More than one objectives can be optimized simultaneously by optimizing a weighted sum of the original objective functions.

• **Chapter 6**: Conclusions

This chapter summarizes the benefits of the optimization code and underlines some conclusions of the simulations results.
Many numerical methods employed in power system studies require the computation of derivatives of functions. In power network studies the Jacobian matrix of the power flow equations is built up and evaluated in every iteration of the Newton-Raphson method. In optimization studies the chosen nonlinear optimization method very often requires the Jacobian or Hessian matrix of functions as well as the gradient vector of the objective function. Sensitivity analysis also requires calculation of first or higher order derivatives.

A substantial part of the computational working load in the above studies is devoted to the numerical evaluation of function derivatives of varying order. The original functions are usually described in a higher level programming language like FORTRAN or C. There are several ways to obtain the first order derivatives of a function with a computer program:

- Hand coding
- Finite-difference approximation
- Symbolic differentiation
- Automatic differentiation

In this chapter the above approaches to computational differentiation will be reviewed. Hand coding of derivatives has been extensively used in the past, especially when the number of functions to be differentiated is small and the functions relatively simple. When the functions are more complex, or when high accuracy of the calculations is not desired the derivatives can be estimated with finite-difference approximation. Both approaches have drawbacks and for this reason alternative techniques were developed. Symbolic differentiation became popular in the mathematical community.
It computes exact derivatives and relieves the programmer from coding the derivative expressions. However the automatic differentiation technique, by making clever use of the chain rule, has proven to be more efficient and will be the core of this chapter.

In power system studies the nonlinearity of the AC power flow equations, the large dimension of the problem, and the (sparse) connections of the network buses force the software developer to handle the problem as a nonlinear large scale sparse problem. The computation of the Jacobian matrix of the power flow equations usually involves two main steps:

1. Obtaining the numerical values of the Jacobian terms at the current iterate
2. Placing the Jacobian terms in the right position in the sparse Jacobian matrix

The accuracy of computations, which can be critical for certain optimization algorithms, excludes the finite-difference approximation of derivatives, and the sparsity of the power system equations rules out some popular general purpose symbolic differentiation packages that do not handle sparsity. As a result, the power system community has been traditionally hand coding the expressions and the sparsity structure of first and second order derivatives.

In this chapter the properties of the different computational differentiation methods will be reviewed in the context of power system calculations and relevant examples. The efficiency of each method will be evaluated as well as the flexibility of the resulted derivative code. Code flexibility and maintainability is critical when the program developer wishes to experiment with a non-static code, i.e. different sets of variables, alternative formulations of functions and new network components. This chapter will focus on how the technique of automatic differentiation works and how it can be used in this environment where development speed is of primary importance.

2.1 Computational differentiation

As pointed out in the introduction, one approach to obtain numerical derivatives of simple functions is to derive the analytical expressions for the first or higher order derivatives by hand and code them in a high level programming language, like FORTRAN or C (hand coding approach). The efficiency of the resulting code depends heavily on the efficiency of the coded derivatives, which is the responsibility of the programmer. Issues like exploitation of common sub-expressions and use of area specific knowledge can have a large effect on memory and time requirements of the executable code. However, an efficient code presumes a large development effort
on the part of the programmer. As a result this approach can be very tedious and error-prone in large and complex code structures.

In cases where the number of the functions to be differentiated is large or the complexity of the functions makes the derivation of the analytical expression by hand difficult, first order derivatives can be estimated with the finite-difference approximation. Finite-difference approximations to first order derivatives are based on the truncated Taylor series and can be calculated using, for example, the forward difference approximation:

$$\frac{\partial f}{\partial x_j} \bigg|_{x^0} = \frac{f(x^0 + h \cdot e_j) - f(x^0)}{h}$$

(2.1)

where $f$ a function of $x \in \mathbb{R}^n$, $x^0$ the point at which the derivatives will be computed, $e_j$ is the $j^{th}$ Cartesian basis vector and $h$ is the perturbation in variable $x_j$. The main advantage of this approach is that it requires only the coding of the original function. For the numerical evaluation of the derivatives, this function is being evaluated at different points and the difference in function value is used as a measure of the function derivative value. This means that the function, however complicated, is treated as a "black box", which can be of high value if the analytical expressions of function derivatives are hard to obtain. However, the difficulties associated with the tradeoff between truncation and roundoff error make this approach unattractive, especially when high accuracy is desired. Another negative aspect of the technique is that the computational cost of obtaining the gradient vector grows with the number of independent variables $x_j$. In fact, if the dimension of vector $x$ is $n$, then the computational cost $q\{\nabla f\}$ of obtaining $\nabla f$ is

$$q\{\nabla f\} = (n + 1) \cdot q\{f\}$$

(2.2)

We define the work ratio $Q\{f\}$ of function $f$ as the ratio of the computational cost for calculating $\nabla f$ and $f$ over the cost of calculating $f$.

$$Q\{f\} = \frac{q\{f, \nabla f\}}{q\{f\}}$$

(2.3)

From (2.1) it is obvious, that for the finite-difference approximation of derivatives the work ratio is $n + 2$, with $n$ being the number of variables.

## 2.2 Derivative code generation

The methods that were described so far were shown to have some serious shortcomings: Hand coded derivatives cannot be coded in a mechanical way and their
efficiency depends heavily on the programmer. Finite-difference derivatives, on the other hand, can be calculated in an "automatic" way since the technique is independent of the underlying function \( f \). However the inaccuracy problem as well as the linear dependence of the computational cost on the number of variables makes this method inefficient. Because of these weak points research moved to the area of code-generation for derivatives. Derivative code generation software packages work as shown in Figure 2.1.

The technique of symbolic differentiation (SD) belongs in this category. The SD packages\(^1\) are large scale general purpose computer algebra packages. The user provides the expressions for the function \( f \) and defines the independent and dependent variables. After obtaining the symbolic expressions for \( \nabla f \) in the SD package the user has two options for obtaining the numerical values:

- evaluate the expressions at certain arguments within the SD environment and export the numerical values to the main program that calls the derivatives
- export the generated code to a high level language (FORTRAN, C) that can be directly compiled and linked to the main code.

In applications that require frequent evaluation of the derivatives (iterative methods, optimization), the most efficient way is to include the generated code in the main program. This requires the conversion of the generated code into the same language as the main program, or into a language that can be linked to it (see \([14, 15]\)).

The main advantage of the SD method is that the derivative code is generated automatically. The user gains the mathematical insight into the derivative functions and obtains exact derivatives up to machine precision. Most SD packages have the capability of exporting these generated formulas to a high level language, so the user

\(^1\) such as Maple \([10]\), Matlab \([11]\), REDUCE \([12]\), Mathematica \([13]\)
need only include the derivative code and recompile. However there is an inefficiency problem when generating code for derivatives with a SD package. The following example will display how the Maple SD package can be used to derive the first order derivatives of function

\[ f(x) = \prod_{i=1}^{n} x_i = x_1 x_2 \cdots x_n \]  

(2.4)

where the vector \( x \in \mathbb{R}^n \) is the vector of independent variables.

\( f \) is a scalar function and can be formulated in Maple with the command

\[ f := \text{product}(x[k], k=1..n) \]

The command \( g := \text{grad}(f, x) \) will provide symbolic expressions for all \( n \) elements of the gradient vector \( g = \nabla f \). Obviously the \( i^{th} \) element of the \( g \) vector is

\[ \frac{\partial f}{\partial x_i} = \prod_{j \neq i} x_j = x_1 \cdots x_{j-1} x_{j+1} \cdots x_n \]  

(2.5)

From (2.4) and (2.5) it is clear that there are common subexpressions in the calculation of function \( f \) and its derivatives. If no advantage is taken of these common expressions, the cost of evaluating each element of \( \nabla f \) is almost the same as the cost of evaluating \( f \) (for this particular function). The \texttt{cost[expression]} command of Maple prints the operation count (multiplications, additions, subscripts, assignment statements) for the numerical evaluation of \texttt{[expression]}. In the following it will be assumed that the measure for the cost \( q\{f\} \) of function \( f \) is the total number of floating point operations plus the fetch-store computational cost. For \( n = 5 \):

\[ q\{f\} = 4 \text{ multiplications} + 5 \text{ subscripts} \]

\[ q\{\nabla f\} = 15 \text{ multiplications} + 25 \text{ subscripts} + 5 \text{ assignments} \]

If the same weight \( w \) is assigned to all operations then (2.3) gives

\[ Q\{f\} = \frac{w(45 + 9)}{w(9)} = 6 \]  

(2.6)

To perform optimization in common expressions in the calculation of \( \nabla f \) the Maple command \texttt{optimize(grad(f,x))} will be used, and for \( n = 5 \):

\[ q\{\nabla f\} = 11 \text{ multiplications} + 17 \text{ subscripts} + 7 \text{ assignments} \]  

(2.7)
and

\[ Q\{f\} = \frac{w(35+9)}{w(9)} = 4.89 \] (2.8)

Figure 2.2 shows the work ratio of function (2.4) as a function of \( n \), the dimension of vector \( x \), without (solid line) and with (dashed line) optimization of common expressions. Without code optimization the work ratio for the calculation of derivatives for

\[ (2.4) \text{ grows linearly with the number of variables. With the optimize option there is a bounded work ratio for the examined range of } n. \text{ The improvement is drastic because there are increasingly more common expressions between } f \text{ and } \nabla f \text{ as the dimension of } x \text{ grows. Failure in considering these common expressions can result in a very inefficient derivative code.} \]

Next, the behaviour of partial separable functions will be examined. These functions have a separable structure and can be formulated as

\[ f(x) = \sum_{i=1}^{m} f_i(\bar{x}_i), \] (2.9)

where \( f_i \) depends on \( p_i = \dim(\bar{x}_i) \ll n \) variables. Partially separable functions arise often in power system applications. Their main property is that they have a sparse Hessian matrix. Assume that the MW losses in an area with \( numTL \) transmission lines have to be calculated (see fig 2.3). One way to calculate the losses is to add up
the MW losses of each transmission line of this area. The equations (2.10) are used for this purpose,

\[
V_1^i \angle \delta_1^i \quad P_{12}^i \quad P_{21}^i \quad V_2^i \angle \delta_2^i
\]

\[P_{\text{LossArea}} = 0\]

for \(i = 1 \ldots \text{numTL}\)

\[
P_{12}^i = \frac{R_i V_1^i V_2^i \cos(\delta_1^i - \delta_2^i) + X_i V_1^i V_2^i \sin(\delta_1^i - \delta_2^i)}{R_i^2 + X_i^2}
\]

\[
P_{21}^i = \frac{R_i V_2^i V_1^i \cos(\delta_1^i - \delta_2^i) - X_i V_1^i V_2^i \sin(\delta_1^i - \delta_2^i)}{R_i^2 + X_i^2}
\]

\[P_{\text{LossArea}} = P_{\text{LossArea}} + P_{12}^i + P_{21}^i\]

end

(2.10)

Figure 2.3: Transmission line (i) with series impedance \(R_i + jX_i\) and no line charging

and they can be a subset of a potentially large set of equations that must be solved. The independent variables are the voltage vectors at both ports of each branch of the network. If only the nodes of the area are considered, the variable vector is

\[x = [V_1^1, V_2^1, \ldots V_1^\text{numTL}, V_2^\text{numTL}, \delta_1^1, \delta_2^1, \ldots, \delta_1^\text{numTL}, \delta_2^\text{numTL}]\]  

(2.11)

and numerical values for \(P_{\text{LossArea}}\) and \(\nabla P_{\text{LossArea}}\) must be obtained. The results of SD with Maple V (Revision 5) will be now examined. After entering functions (2.10) and defining the independent variables (2.11), the expressions for \(\nabla P_{\text{LossArea}} = \frac{\partial P_{\text{LossArea}}}{\partial x}\) can be derived using \(\text{grad}(P_{\text{LossArea}}, x)\). Utilization of common expressions can be achieved with \(\text{optimize}(P_{\text{LossArea}})\) and \(g := \text{grad}(P_{\text{LossArea}}, x); \text{optimize}(g)\).

Figure (2.4) displays the work ratio \(Q\{P_{\text{LossArea}}\}\) without and with optimization for an increasing number of independent variables \(n = 4 \times \text{numTL}\). For this function the work ratio does not grow with \(n\) even without optimization of the expressions. However, code optimization decreases the work ratio. The result can be exported to Fortran code with \(\text{fortran}(P_{\text{LossArea}}, \text{optimized})\) and \(\text{fortran}(g, \text{optimized})\). The result is a sequence of assignment statements in which temporary values are stored in local variables beginning with the letter t. The global names t0, t1, t2, ... are reserved for use by FORTRAN for this purpose.

The output for \(\text{numTL} = 2\) is displayed in Figure (2.5). The number of assignment statements for the evaluation of the scalar function \(P_{\text{LossArea}}\) is 21 and for the
gradient vector 42. For \( numTL = 20 \) transmission lines (i.e. \( n = 80 \) independent variables) the corresponding numbers have increased to 203 and 420.

The above examples show that, although SD provides the analytical expressions for the derivatives, it can be very inefficient. If common expressions that occur in the original and derivative expressions are not utilized, the evaluation of derivatives can require up to \( n \) times the arithmetic operations for evaluating the function itself. This major disadvantage of SD comes from the fact that SD differentiates expressions rather than code. SD cannot handle code structures (branches, loops) and, therefore, obtaining a ready-to-compile derivative code requires additional effort from the side of the programmer.

### 2.3 The technique of automatic differentiation

As pointed out in the previous sections, the most popular traditional techniques of computational differentiation have major disadvantages that make them impractical for many large applications. Ideally, a computational differentiation technique should
\begin{align*}
t_1 &= V_1(1)^2 \\
t_4 &= \text{delta}1(1)-\text{delta}2(1) \\
t_5 &= \cos(t_4) \\
t_7 &= R(1)*V_1(1)*V_2(1)*t_5 \\
t_9 &= \sin(t_4) \\
t_{11} &= X(1)*V_1(1)*V_2(1)*t_9 \\
t_{13} &= R(1)^2 \\
t_{14} &= X(1)^2 \\
t_{16} &= 1/(t_{13}+t_{14}) \\
t_{18} &= V_2(1)^2 \\
t_{22} &= V_1(2)^2 \\
t_{25} &= \text{delta}1(2)-\text{delta}2(2) \\
t_{26} &= \cos(t_{25}) \\
t_{28} &= R(2)*V_1(2)*V_2(2)*t_{26} \\
t_{30} &= \sin(t_{25}) \\
t_{32} &= X(2)*V_1(2)*V_2(2)*t_{30} \\
t_{34} &= R(2)^2 \\
t_{35} &= X(2)^2 \\
t_{37} &= 1/(t_{34}+t_{35}) \\
t_{39} &= V_2(2)^2 \\
t_{43} &= (R(1)*t_1-t_7+t_{11})*t_{16}+(R(1)*t_{18}-t_7-t_{11})*t_{16}+(R(2)*t_{22}-t_{28}+t_{32})*t_{37} \\
# &+ (R(2)*t_{39}-t_{28}-t_{32})*t_{37} \\
t_1 &= R(1)*V_1(1) \\
t_3 &= R(1)*V_2(1) \\
t_4 &= \text{delta}1(1)-\text{delta}2(1) \\
t_5 &= \cos(t_4) \\
t_6 &= t_3*t_5 \\
t_8 &= \sin(t_4) \\
t_9 &= X(1)*V_2(1)*t_8 \\
t_{11} &= R(1)^2 \\
t_{12} &= X(1)^2 \\
t_{14} &= 1/(t_{11}+t_{12}) \\
t_{19} &= R(2)*V_1(2) \\
t_{21} &= R(2)*V_2(2) \\
t_{22} &= \text{delta}1(2)-\text{delta}2(2) \\
t_{23} &= \cos(t_{22}) \\
t_{24} &= t_{21}*t_{23} \\
t_{26} &= \sin(t_{22}) \\
t_{27} &= X(2)*V_2(2)*t_{26} \\
t_{29} &= R(2)^2 \\
t_{30} &= X(2)^2 \\
t_{32} &= 1/(t_{29}+t_{30}) \\
t_{37} &= t_1*t_5 \\
t_{38} &= X(1)*V_1(1) \\
t_{39} &= t_{38}*t_8 \\
t_{46} &= t_{19}*t_{23} \\
t_{47} &= X(2)*V_1(2) \\
t_{48} &= t_{47}*t_{26} \\
t_{56} &= t_{1}*V_2(1)*t_8 \\
t_{58} &= t_{38}*V_2(1)*t_5 \\
t_{59} &= t_{56}*t_{58} \\
t_{61} &= t_{56}-t_{58} \\
t_{65} &= t_{19}*V_2(2)*t_{26} \\
t_{67} &= t_{47}*V_2(2)*t_{23} \\
t_{68} &= t_{65}+t_{67} \\
t_{70} &= t_{65}-t_{67} \\
g(1) &= (2*t_1-t_6+t_{11})*t_{16}+(t_{16}+t_{22})*t_{28}+(R(2)*t_{39}-t_{32})*t_{37} \\
g(2) &= (2*t_{19}-t_{24}+t_{27})*t_{32}+(t_{24}-t_{27})*t_{32} \\
g(3) &= -(t_{37}+t_{39})*t_{14}+(2*t_{3}-t_{37})*t_{39} \\
g(4) &= -(t_{46}+t_{48})*t_{32}+(2*t_{21}-t_{46}-t_{48})*t_{32} \\
g(5) &= t_{59}+t_{61}+t_{14} \\
g(6) &= t_{68}+t_{32}+t_{70}+t_{32} \\
g(7) &= -t_{59}+t_{61}+t_{14} \\
g(8) &= -t_{68}+t_{32}+t_{70}+t_{32} \\
\end{align*}

Figure 2.5: Left: Fortran code created by Maple \texttt{fortran(PLossArea,optimized)} and right: Fortran code created by \texttt{fortran(grad(PLossArea,x),optimized)} for \textit{numTL} = 2
• compute derivatives automatically
• be able to handle arbitrary high-level codes, rather than expressions
• compute exact derivatives (free of truncation errors)
• compute derivatives at a cost independent of the number of variables

Automatic differentiation (AD) is a relatively recent technique [16, 17, 18] that has these four properties. It computes derivatives of functions represented by means of a program written in a high-level language such as FORTRAN or C. The AD technique relies on the principle that the chain rule for calculating derivatives

\[
\frac{\partial f(g(t))}{\partial t} \bigg|_{t=t_0} = \left( \frac{\partial f(s)}{\partial s} \bigg|_{s=g(t_0)} \right) \left( \frac{\partial g(t)}{\partial t} \bigg|_{t=t_0} \right)
\]  

(2.12)
can be applied in a mechanical fashion to compute derivatives of a complicated function. The AD software packages generate code for the derivatives rather than full symbolic expressions with respect to the independent variables.

Consider the function \( y = f(x) \), \( f : \mathbb{R}^n \rightarrow \mathbb{R} \), represented by the following subroutine:

\begin{verbatim}
FUNC(y,x)
    for i = n + 1 \rightarrow n + p
        xi = fi(xj)j ∈ Ji
    end
    y = x_{n+p}
\end{verbatim}

(2.13)

where

\( J_i \subset \{1, 2, \ldots, i - 1\}, \ \forall i = n + 1, n + 2, \ldots, n + p \)

The subroutine (2.13) represents the function \( f(x) \) as a composite of the elementary or library functions \( \{f_i\}_{i=n+1}^{n+p} \), where \( f_i \) is a function of already computed quantities \( x_j, j \in J_i \). In the following, the function of the real part of the terminal current \( I = I_e + iI_f \) of a one-port device (load, generator) of complex power \( S = P + iQ \), voltage \( V = e + if \) and shunt admittance \( Y = G + iB \) (see Figure 2.6) will be used.

The independent variables are

\[
\begin{bmatrix}
    x_1 & x_2 & x_3 & x_4
\end{bmatrix}^T = \begin{bmatrix}
    P & Q & V & \delta
\end{bmatrix}^T
\]  

(2.14)
\[ L = G + iB \quad V = V \angle \delta \]

\[ S = P + iQ \quad I = I_e + iI_f \]

\[ I_e = \frac{1}{\delta} \cdot (P \cos(\delta) + Q \sin(\delta) + GV^2 \cos(\delta) - BV^2 \sin(\delta)) \]

Figure 2.6: One-port device

so that the function in Figure 2.6 can be rewritten as

\[ y = \frac{1}{x_3} \cdot (x_1 \cos(x_4) + x_2 \sin(x_4) + Gx_3^2 \cos(x_4) - Bx_3^2 \sin(x_4)) \] (2.15)

We want to compute the gradient vector of \( I_e = y \)

\[ \nabla I_e = \nabla y = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \frac{\partial y}{\partial x_3} & \frac{\partial y}{\partial x_4} \end{bmatrix}^T \] (2.16)

Figure 2.7: Composition of function 2.15

The function can be brought to the form (2.13) by decomposing it into elementary functions (see Figure 2.7). The variables \( \{x_i\}_{i=5}^{18} \) in Figure 2.7 are the intermediate
variables where the results of the elementary or library functions are stored. Figure 2.8 shows how the composite function can be visualized by means of an acyclic directed graph. Every node represents a variable, independent or intermediate. An arc that runs from node $x_j$ to node $x_i$ represents a dependency between the two variables. The direction $j \rightarrow i$ assumes that variable $x_i$ depends on the already computed variable $x_j$. Therefore, there is a nonzero partial derivative associated with each arc and the graph whose arcs are labeled with their elementary partial derivatives will be referred to as a computational graph (Figure 2.8).

This computational graph in Figure 2.8 displays visually the first step of any AD approach. The complicated expression is broken down into intermediate variables $x_i = f_i(x_j), j \in J_i$ that hold results of unary or binary operations and simple function calls. The partial derivatives $\frac{\partial f_i}{\partial x_j}$ of these simple operations are computed using the simple rules of differentiation (product rule, quotient rule, first derivatives of library functions). Having built the computational graph, it is possible to compute the gradient of $f$ by accumulating the elementary partial derivatives with the chain rule. Traditionally, two different ways of applying the chain rule were developed: In the forward mode derivatives with respect to the independent variables are computed, whereas in the reverse mode derivatives of the final (dependent) variable with respect to the intermediate variables are computed. Both modes have advantages and disadvantages and will be presented in more detail.

### 2.3.1 The Forward mode

The forward mode maintains derivatives of every intermediate variable with respect to the independent variables. The process begins at the independent variables and works its way towards the dependent variables (bottom up approach). Figure 2.9 contains an implementation of the forward mode. With the introduction of $x_i$ the gradient vector $\nabla x_i$ is computed in a source-code transformation fashion. As a consequence, space must be allocated for $n + p$ scalar variables and $n + p$ gradient vectors ($p$ being the number of intermediate variables). All computations are gradient computation loops, so that the computational work for calculating the gradient vector depends on the number of independent variables (see (2.17)).

$$q\{f, \nabla f\} = \sum_{i=n+1}^{p} q\{f_i, \nabla f_i\} + nn_i (\text{multiplications + additions}) \quad (2.17)$$
where $n_i$ a constant depending on the complexity of $f_i$. In the case of differentiation of $y = f(x), f : \mathcal{R}^n \to \mathcal{R}^m$ the forward mode computes

$$
\begin{bmatrix}
\nabla y_1^T \\
\vdots \\
\nabla y_m^T
\end{bmatrix} = \frac{\partial y}{\partial x} \cdot 
\begin{bmatrix}
\nabla x_1^T \\
\vdots \\
\nabla x_n^T
\end{bmatrix}
$$

(2.18)

where

$$
\frac{\partial y}{\partial x} = 
\begin{bmatrix}
\frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n}
\end{bmatrix}
$$

(2.19)
FORWARD($y, \nabla f, x$)
for $i = 1 \rightarrow n$
\[ \nabla x_i = e_i \]
end
for $i = n + 1 \rightarrow n + p$
\[ x_i = f_i(x_j)_{j \in J_i} \]
\[ \nabla x_i = \sum_{j \in J_i} \frac{\partial f_i}{\partial x_j} \nabla x_j \]
end
\[ y = x_{n+p} \]
\[ \nabla f = \nabla x_{n+p} \]

Figure 2.9: Implementation of the forward mode

Note that throughout this chapter the $\nabla(*)$ is a column vector. With suitable initialization of $\nabla x_i$ (2.18) yields a linear combination of the columns of the Jacobian (2.19). In particular, if $\nabla x_i = u_i$ then (2.18) yields the directional derivative
\[ f'(x) \cdot u = \frac{\partial y}{\partial x} \cdot u = \lim_{h \to 0} \frac{f(x + h \cdot u) - f(x)}{h} \] (2.20)
defined as the derivative of variable $x$ along a direction vector $u$.

2.3.2 The Reverse mode

The reverse (top-down or backward) mode starts from the dependent variables and propagates derivatives towards the independent variables accumulating the adjoint scalar quantities $\bar{x}_i = \frac{\partial y}{\partial x_i}$. From the chain rule it holds that
\[ \bar{x}_j = \sum_{i \in I_j} \frac{\partial f_i}{\partial x_j} \bar{x}_i \] (2.21)
with $I_j \equiv \{i \leq n+p : j \in J_i\}$. In Figure 2.10 an implementation of the reverse mode is displayed.

The reverse mode consists of a forward sweep during which the computational graph (variables and elementary derivatives) is built and a reverse sweep for the accumulation of derivatives. For every temporary variable there is now an associated scalar quantity, carrying the derivative information, rather than a vector. Memory should
Chapter 2: Automatic differentiation in power system studies

REVERSE\( (y, \nabla f, x) \)
for \( i = n + 1 \) to \( n + p \)
\[
x_i = f_i(x_j)_{j \in J_i},
\]
\( \overline{x}_i = 0 \)
end
\( y = x_{n+p} \)
\( \overline{x}_{n+p} = 1 \)
for \( i = 1 \) to \( n \)
\( \overline{x}_i = 0 \)
end
for \( i = n + p \) down to \( n + 1 \) by \(-1\)
\[
\overline{x}_j = \overline{x}_j + \frac{\partial f_i}{\partial x_j} \overline{x}_i, \ \forall j \in J_i
\]
end
for \( i = 1 \) to \( n \)
\( \nabla f_i = \overline{x}_i \)
end

Figure 2.10: Implementation of the reverse mode

be allocated for \( n + p \) scalars and all computations are with scalar operands. The work of computing the original function \( f \) and its gradient is

\[
q\{f, \nabla f\} = \sum_{i=n+1}^{n+p} q\{f_i, \nabla f_i\}
\]

which means that the work ratio of \( f \) is bounded above by the worst work ratio of the elementary functions \( f_i \). This upper bound is independent of the number of variables \( n \) and for all practical purposes the value of 5 [18] can be used.

In the case of differentiation of \( y = f(x), f : \mathbb{R}^n \to \mathbb{R}^m \) the reverse mode computes

\[
\begin{bmatrix}
\overline{x}_1 & \cdots & \overline{x}_n
\end{bmatrix} = \begin{bmatrix}
\overline{y}_1 & \cdots & \overline{y}_n
\end{bmatrix} \cdot \frac{\partial y}{\partial x}
\]

(2.23)

where \( \frac{\partial y}{\partial x} \) is defined in (2.19). That is, reverse mode allows the computations of linear combinations of the rows of the Jacobian matrix. If the initial values \( \overline{y}_i = 1, \overline{y}_j \neq i = 0 \) are set, (2.23) yields the Jacobian row \( \frac{\partial y_i}{\partial x} \).

Figure 2.11 on page 27 shows the implementation of calculating the example function (2.15) and its gradient with the forward and the reverse mode. For both approaches the definition of the \( x_i, \ i = 5 \ldots 18 \) variables is the first step. For the estimation of
\( \nabla y \) with the forward mode, space is allocated for 19 gradient vectors, and there are 19 gradient computation loops. With the clean reverse mode of AD space is allocated for 19 scalar variables and, apart from the last gradient computation loop, all operations are scalar. In this relatively simple example with one dependent variable and more than one independent variable the reverse mode proves to be more economical in terms of computational cost and memory requirements. However, the implementation of the reverse mode can be very complicated in the general case, as it requires the ability to access every intermediate result in reverse order. This could increase the memory allocation requirements by an unpredictable factor. In contrast, the storage requirements of the pure forward mode grow with the predictable factor of \( n + 1 \).

### 2.3.3 ADIFOR tool - Hybrid approach

There have been various implementations of AD [19] and many of the available AD tools have similar characteristics, in that they all provide differentiation of algorithms expressed in some general purpose programming language. There are different criteria for dividing all AD tools into classes, and one of those criteria is the differentiation approach (forward or reverse) used to propagate derivatives. As stated in section 2.3 there is no optimal differentiation mode for all purposes. The ADIFOR (Automatic DIfferentiation for FORtran) package \(^2\) is an implementation of AD for FORTRAN programs that employs a hybrid forward/reverse approach: It is mainly based on the forward mode but in every assignment statement there is a local reverse mode step. Since the reverse mode is suitable for obtaining derivatives of scalar functions, it is applied in assignment statements to accumulate the partial derivatives of the left hand side (defined locally as the dependent variable) with respect to each variable or the right hand side (local independent variables). At the end, application of the chain rule in a forward mode sense will propagate the derivatives to the independent variables.

ADIFOR uses mainly the forward mode and therefore (see (2.18)) computes

\[
\mathbf{g}_y = \left( \frac{\partial y}{\partial \mathbf{s}} \cdot \mathbf{g}_x \right)^T
\]

where the notation \( \mathbf{g}_{s \times n} \) denotes the directional gradient vector or derivative object associated with vector \( \mathbf{s} = [s_1, s_2, \ldots, s_n] \) (\( p \) being the number of direction vectors).

---

\(^2\) A collaborative project between the Mathematics and Computer Science Division at Argonne National Laboratory and the Center for Research on Parallel Computation at Rice University ([20, 21]).
\[ y = \frac{1}{x_3} \cdot (x_1 \cos(x_4) + x_2 \sin(x_4) + Gx_3^2 \cos(x_4) - Bx_3^2 \sin(x_4)) \]

**Forward Mode**

Initialize:
\[ \nabla x_1 = [1, 0, 0, 0]^T, \nabla x_2 = [0, 1, 0, 0]^T, \]
\[ \nabla x_3 = [0, 0, 1, 0]^T, \nabla x_4 = [0, 0, 0, 1]^T, \]
\[ \nabla x_i = 0, \quad i = 5 \ldots 19 \]

\[ \nabla x_5 = 2x_3 \cdot \nabla x_3 \]
\[ \nabla x_6 = -1/x_3^2 \cdot \nabla x_3 \]
\[ \nabla x_7 = -\sin(x_4) \cdot \nabla x_4 \]
\[ \nabla x_8 = \cos(x_4) \cdot \nabla x_4 \]
\[ \nabla x_9 = G \cdot \nabla x_5 \]
\[ \nabla x_{10} = B \cdot \nabla x_5 \]
\[ \nabla x_{11} = x_6 \cdot \nabla x_7 + x_7 \cdot \nabla x_6 \]
\[ \nabla x_{12} = x_8 \cdot \nabla x_6 + x_6 \cdot \nabla x_8 \]
\[ \nabla x_{13} = x_{11} \cdot \nabla x_1 + x_1 \cdot \nabla x_{11} \]
\[ \nabla x_{14} = x_{12} \cdot \nabla x_2 + x_2 \cdot \nabla x_{12} \]
\[ \nabla x_{15} = x_{11} \cdot \nabla x_9 + x_9 \cdot \nabla x_{11} \]
\[ \nabla x_{16} = x_{12} \cdot \nabla x_{10} + x_{10} \cdot \nabla x_{12} \]
\[ \nabla x_{17} = \nabla x_{13} - \nabla x_{14} \]
\[ \nabla x_{18} = \nabla x_{15} - \nabla x_{16} \]
\[ \nabla y = \nabla x_{17} + \nabla x_{14} \]

**Reverse Mode**

Initialize:
\[ \bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4 = 0 \]
\[ \bar{x}_i = 0, \quad i = 1 \ldots 18 \]
\[ \bar{y} = 1 \]

\[ \bar{x}_{18} = 1 \cdot \bar{y} \]
\[ \bar{x}_{17} = 1 \cdot \bar{y} \]
\[ \bar{x}_{15} = 1 \cdot \bar{x}_{18} \]
\[ \bar{x}_{16} = (-1) \cdot \bar{x}_{18} \]
\[ \bar{x}_{14} = 1 \cdot \bar{x}_{17} \]
\[ \bar{x}_{13} = 1 \cdot \bar{x}_{17} \]
\[ \bar{x}_{12} = x_{10} \cdot \bar{x}_{16} \]
\[ \bar{x}_{10} = x_{12} \cdot \bar{x}_{16} \]
\[ \bar{x}_9 = x_{11} \cdot \bar{x}_{15} \]
\[ \bar{x}_{11} = x_9 \cdot \bar{x}_{15} \]
\[ \bar{x}_2 = x_{12} \cdot \bar{x}_{14} \]
\[ \bar{x}_{12} = x_2 \cdot \bar{x}_{14} \]
\[ \bar{x}_1 = x_{11} \cdot \bar{x}_{13} \]
\[ \bar{x}_{11} = x_1 \cdot \bar{x}_{13} \]
\[ \bar{x}_8 = x_6 \cdot \bar{x}_{12} \]
\[ \bar{x}_6 = x_8 \cdot \bar{x}_{12} \]
\[ \bar{x}_7 = x_6 \cdot \bar{x}_{11} \]
\[ \bar{x}_5 = x_7 \cdot \bar{x}_{11} \]
\[ \bar{x}_5 = B \cdot \bar{x}_{10} \]
\[ \bar{x}_5 = G \cdot \bar{x}_9 \]
\[ \bar{x}_4 = \cos(x_4) \cdot \bar{x}_8 \]
\[ \bar{x}_4 = -\sin(x_4) \cdot \bar{x}_7 \]
\[ \bar{x}_3 = -1/x_3^2 \cdot \bar{x}_6 \]
\[ \bar{x}_5 = 2x_3 \cdot \bar{x}_5 \]
\[ \nabla y = [\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4] \]

Figure 2.11: Forward versus Reverse mode in computing \( \nabla y \)
\[ \mathbf{g}_s = [ \nabla s_1 \ldots \nabla s_n ] \] (2.25)

The \( p \) direction vectors (independent variables) are defined with the seed matrix \( \mathbf{g}_s^{T \times p} \). By suitable initialization of the seed matrix, the user determines what derivative information \( \mathbf{g}_y \) contains. If the seed matrix is initialized to the unit matrix, then obviously \( \mathbf{g}_y^T \) is the Jacobian matrix. A directional gradient vector of a scalar variable is the set of directional derivatives of that variable along all directions specified in the seed matrix.

**Input to ADIFOR**

```fortran
do i = 1, num_lp_data
    I_e(i) = (e(i) * P(i) + f(i) * Q(i))/
            + (e(i)**2 + f(i)**2) + G(i) * e(i) -
            B(i) * f(i)
enddo
```

**Output from ADIFOR**

```fortran
do i = 1, num_lp_data
    d8_v = e(i) * e(i)
    d2_p = 2.0d0 * e(i)
    d9_v = f(i) * f(i)
    d1_p = 2.0d0 * f(i)
    d10_v = d8_v + d9_v
    d11_v = (e(i) * plp(i) + f(i) * q1p(i))/d10_v
    d8_b = 1.0d0 / d10_v
    d9_b = (-d11_v)/d10_v
    d4_b = -b(i) + d9_b * d1_p + d8_b * q1p(i)
    d14_b = d8_b * f(i)
    d7_b = g(i) + d9_b * d2_p + d8_b * plp(i)
    d15_b = d8_b * e(i)
    do g_i_ = 1, g_p_
        g_i_e(g_i_, i) = d14_b * g_q1p(g_i_, i) +
                         d4_b * g_f(g_i_, i) + d15_b *
                         g_plp(g_i_, i) + d7_b * g_e(g_i_, i)
    enddo
    i_e(i) = d11_v + g(i) * e(i) - b(i) * f(i)
enddo
```

Figure 2.12: Input and generated ADIFOR code for \( I_e = \frac{e^P + f^Q}{e^2 + f^2} + G \cdot e - B \cdot f \)

Figure 2.12 shows the input FORTRAN code for calculating the \( I_e \) of all one-port devices of a network and the ADIFOR generated FORTRAN code that contains commands for calculating both \( I_e \) and \( \mathbf{g}_s \). The process starts with breaking up the input function, storing intermediate results to temporary variables and calculating the scalar partial derivatives. Comparing the output with the input code, it is obvious that the code structure (outer do-loop) is preserved. The inner do-loop in the generated code is the implementation of (2.18). Therefore, the ADIFOR approach (forward-mode) computes derivatives at a cost of roughly \( \mathbf{g}_p \) times the cost of computing the underlying function, regardless of the sparsity of the Jacobian matrix. But if the Jacobian matrix is sparse, it is possible to exploit sparsity and reduce \( \mathbf{g}_p \). This
requires, though, that the sparsity pattern of the Jacobian is known a priori, and that there are structural orthogonal columns that must be identified by the user or with graph-coloring algorithms [22, 23, 20].

For these simple functions $I_e(i) = f(e(i), f(i), P(i), Q(i))$ the ”full” Jacobian for $n = 4$ oneports [24, 25] is the left hand side matrix in Figure 2.13 (the symbols stand for nonzero entries). It is obvious that there are four groups of structural orthogonal columns, regardless of the number of oneports. With suitable initialization of the seed matrix, a compressed version of the full Jacobian is obtained, and the individual nonzero elements of the full Jacobian can be easily extracted from its compressed version. The compressed Jacobian has in general fewer zero elements and in this particular case the compressed matrix is 100% dense. For this example the sparsity pattern and the partition into structural orthogonal columns is trivial, and therefore the compressed Jacobian approach is efficient. In the general case, however, it is desirable to have a tool that provides the sparsity structure of the derivatives (see section 2.4.3).

2.4 How ADIFOR works

The ADIFOR 2.0 system consists of three main components (see Figure 2.14):

- The ADIFOR 2.0 preprocessor: The ADIFOR 2.0 preprocessor parses the code, determines which variables should have a derivative object associated with them and generates derivative code with templates at call sites of Fortran77 intrinsics and, potentially, calls to SparsLinC routines.
Chapter 2: Automatic differentiation in power system studies

2.4.1 The ADIFOR 2.0 preprocessor

The ADIFOR 2.0 preprocessor is the first phase of the derivative code generation process. In a first stage the input code is essentially rewritten to comply with the source transformation rules imposed by ADIFOR 2.0. This code canonicalization phase consists mainly of the process of breaking up long right-hand-side assignments.

- The ADIntrinsics system: The ADIntrinsics system expands calls to Fortran77 intrinsics to Fortran77 code
- The SparsLinC Library: The SparsLinC Library provides support for sparsity handling in the derivative computations.
and storing the intermediate results to temporary variables. Statement functions are expanded into in-line code and certain modifications make sure that all variables appearing on the right hand side of a statement are of the same type.

In the **variable nomination phase** the *active* variables are detected. Active variables are those whose values depend on the values of the independent variables and influence the values of the dependent variables in the derivative computation. The variables that do not influence the dependent variables in the derivative computations are *passive* variables. In order to locate all active variables ADIFOR performs a dependency analysis: for each subroutine a "local interaction graph" connects each input variable with all output variables that it influences. Next, an interprocedural analysis is employed which detects all possible paths through which an independent variable affects a dependent variable. As ADIFOR 2.0 supports complicated control structures, common blocks and equivalence statements, the variable nomination phase can be a computer intensive process.

After the variable nomination phase has been completed ADIFOR 2.0 associates a derivative object with each active variable. This is the **code generation phase** where code is generated for each statement. The approach used locally is the AD reverse mode, described in section 2.3.2. This approach rewrites the original (input) code and provides formulas for the local derivatives. Certain eliminations (like multiplication with 1.0 or additions with 0.0) take place and calls to Fortran intrinsics are replaced by templates that will be instantiated by the ADIntrinsics system (see 2.4.2). The user can customize certain aspects in the code generation with the options in the script file and the seed matrix initialization.

### 2.4.2 The ADIntrinsics system

The ADIntrinsics system is activated with calls to Fortran intrinsic subroutines. It provides ADIFOR with rules defining the behavior of the generated code for the exception events, cases where the elementary function is evaluated at points where the derivative function is not defined. For example, for the functions $z1 = \text{sign}(x)$ and $z2 = \sqrt{x}$ an exception rises for $x=0.0$. At the code generation phase (see section 2.4.1) the ADIFOR preprocessor generates invocations of "templates" at call sites of Fortran 77 intrinsics. At a later stage a *purse* postprocessor will expand the templates into explicit Fortran code.

ADIFOR defines *classes* of intrinsic exceptions. For functions that are discontinuous at the point of evaluation ( $z1$ at $x=0.0$) the class **JumpVal** is defined and for the events when the derivative approaches infinity ( $z2$ at $x=0.0$) class **InfVal** is defined. The default value that each exception class returns is also determined. In the case
of the sign function at the point of nondifferentiability, JumpVal returns 0 which is the average of the two limits at each side. In any case the exception handler prints a message every time an exceptional condition occurs.

The ADIntrinsics system is mainly designed for the software developer to customize the behavior of the exception-handling mechanism. The gradient values returned from every exception class can be modified by the user as well as the exception report messages. Usually the user needs only to be aware of the occurrence of such exceptions. For that a call to the ehrpt routine as well as the linking of the ADIntrinsics objects and libraries are necessary.

2.4.3 The SparsLinC Library

In section 2.3.3 the Jacobian of the vector $I_e$ was computed as a dense matrix, although it is highly sparse. By exploiting the special orthogonality of the structure a compressed form of the Jacobian was used for the computation. For this particular case the columns of the Jacobian could be partitioned into $p = 4$ orthogonal groups, i.e. $p$ is small and independent of the problem dimension. These kinds of structures can be handled efficiently with dense data structures [23]. In the general case though, the cost for identifying the structural orthogonal columns can be prohibitive, especially if the sparsity pattern is not constant throughout the computation. It can also happen that there are no structural orthogonal columns in the structure. In these cases it is desirable to relieve the programmer from the tedious and error prone task of entering the sparsity structure of matrices. In these cases it is possible to let ADIFOR compute the sparsity pattern by using the SparsLinC Library.

The SparsLinC (Sparse Linear Combinations) Library is a library written in ANSI C for the sparse implementation of the operation linear combination of vectors.

$$w = \sum_{i=1}^{k} \alpha_i u_i$$  \hspace{1cm} (2.26)

Equation (2.26) is the kernel operation in the derivative propagation (chain rule), where the vectors $u_i$ are the derivative objects of the active variables, and the multipliers $\alpha_i$ are the scalar local derivatives.

SparsLinC computes the sparsity pattern of derivatives as a byproduct of the AD process. The user has to specify the SPARSE option before running ADIFOR, and initialize the seed matrix to the identity matrix. On exit the user obtains the Jacobian in a sparse data structure. In the following chapters, the ADIFOR/SparsLinC approach will be used because:
• power system studies can result in large problems, therefore the underlying sparsity in the computations must be considered

• if the programmer wants to experiment with different sets of variables, models, constraints, the sparsity structure of derivatives changes often

• the user should be relieved of the error-prone task of entering the sparsity structure of derivatives for every case

2.5 ADIFOR applied to the AC power flow algorithm

There are various examples of applying an AD tool to a power system’s application [26, 27]. The process of applying ADIFOR to the power flow algorithm [28] is schematically illustrated in Figure 2.15. The process can be divided in two steps:

• Step 1: Specifying input for ADIFOR and running ADIFOR to generate the derivative code

• Step 2: Incorporating the generated code into the user’s main program

2.5.1 Step 1: Specifying Input for ADIFOR

In order to apply ADIFOR for generating the code related to the sparse Jacobian matrix of user-programmed power flow functions, the user must provide ADIFOR with:

1. The source (FORTRAN77) code (powerflow.f) that builds the vector of equations (mismatch). This is the name of the variable used in the subroutine code, which after execution of this compiled and linked routine, will contain all numeric mismatch values. This code part includes all necessary case statements (if-then-else), all necessary “do-endo” loops, mathematical functions such as $\sin$, $\cos$, include statements, common blocks etc. Using ADIFOR, the user is free to use all expression and language capabilities of the standard FORTRAN77 language.
Figure 2.15: Flow diagram of Power Flow with ADIFOR/SPARSLINC approach
2. A script file (ADIFOR_CONTROL.adf) in which all ADIFOR options and specifications are listed. This file is the control file for the ADIFOR executable. The most important parameters are shown below. The mandatory option AD_IVARS contains those variable vectors of the power flow problem, that are (or can be) independent variables, depending on user-set options.

```
AD_PROG = ADIFOR_MAIN.cmp
AD_TOP = powerflow
AD_IVARS = P_slack,Q_slack,PSlackGen,QSlackGen, \
  Pgen,PallSlack,e_node,f_node,V_node,theta_node, \
  Q_pv,Q_pq,QFixAtNode,Qrc_var,Psl_area,Pfl_area,\ 
  PLoss1,PLoss_area,PsLoss,te_reg,tf_reg
AD_OVARS = mismatch AD_FLAVOR = sparse
AD_OUTPUT_DIR = .
AD_DUMP_CALLGRAPH = true
```

The option AD_OVARS contains the dependent variable mismatch. As the option AD_FLAVOR is set to sparse, calls to SparsLinC routines will be invoked and sparse derivative code will be generated.

3. A “dummy” main program (powerflow_dummy.f) that simply contains a call to the top level subroutine powerflow.f.

4. A composition file (ADIFOR_MAIN.cmp) containing only the names of the files of the source FORTRAN77 code to be processed (powerflow_dummy.f and powerflow.f).

Running ADIFOR with the script file as argument

```
Adifor[version number] AD_SCRIPT=ADIFOR_CONTROL.adf
```

the user generates the derivative code of the top-level subroutine (g_powerflow.f).

### 2.5.2 Step 2: Incorporating the generated derivative code to the main power flow program

The only purpose of the dummy main program (powerflow_dummy.f) (see previous section) is to specify the calling conventions of the top level subroutine (powerflow.f). The calling convention of the generated subroutine (g_powerflow.f) has to be used by the main power flow program which first reads the network parameters from a network data file, stores the parameter data into arrays, checks the parameter data for correctness using heuristic rules and initializes values of the unknown variables.

The main power flow program still has to be hand-coded by the power flow software developer. However, it consists mainly of independent, generic code parts related to the used power flow model. The term “generic” indicates that these code sections
have no relation to the actual choice of the power flow equation types, which are coded in `powerflow.f`, or the autogenerated routine `g_powerflow.f` together with the derived Jacobian terms.

The main program includes the following parts, which need to be hand-coded by the power flow code developer:

1. Specification of the order of variables in the Jacobian matrix. This takes place during seed matrix initialization.

2. A call to the autogenerated subroutine (`g_powerflow.f`), that computes both the function values (mismatch) and all non-zero values of the Jacobian matrix. Using a SparsLinC routine, the Jacobian matrix is extracted into the FORTRAN sparse format vector represented by two vectors (index, value).

3. The program developer must then add a small, generic code piece to the main power flow program, which feeds these sparse Jacobian values into a generic linear system sparsity package (UMFPACK 2 [29] has been used in this implementation) together with the mismatch values.

4. After solving the sparse linear system of equations a user-defined code is needed, which updates the unknown variables.

These code pieces allow an executable version of a power flow code by looping over all steps discussed before. The fact that AD inherently keeps track of code dependencies and propagates the derivatives of active variables leads to greater code flexibility. To convey an idea of this flexibility in coding the mismatch equations for new models or features, the example of area loss calculation (page 17) is reconsidered.

The equations (2.10) are in fact all that the user needs to code in the `powerflow.f` for the calculation of the losses. Assuming that $V_i$ and $\delta_i$ (nodal variables instead of branch variables for simplicity) are independent variables, then $P_{12}$ and $P_{21}$ are active variables and, therefore, their derivatives will be computed. After that, $P_{12}$ and $P_{21}$ can be used in any other formulation without expanding to $V_i$ and $\delta_i$. The losses of a branch can be formulated

$$P_{loss}^i = P_{12}^i + P_{21}^i$$  \hspace{1cm} (2.27)

and the real power flow on branch $i$ at port 1 can be fixed to a value with the equation

$$P_{12} - P_{fix} = 0$$  \hspace{1cm} (2.28)

With this technique, it is obvious that the user can now experiment with different formulations of functions. If a model function is more easily described in rectangular
rather than polar coordinates of voltage, whereas $V^i$ and $\delta^i$ are always the independent variables, then the following do-loop

```
do i = 1, num_1p_data
    V_e(i) = V(i) * cos(delta(i))
    V_f(i) = V(i) * sin(delta(i))
endo
```

will trigger code generation for the gradient objects $g_{\text{V}_e}(i)$ and $g_{\text{V}_f}(i)$. After this code part, the user is free to use any of $V(i)$, $\text{delta}(i)$, $\text{V}_e(i)$, $\text{V}_f(i)$ in the same manner.

In contrast to conventional power flow codes with hand-coding derivatives, modifications in the variable set is now trivial. Changing the variables is a major change in a conventional power flow, as it requires:

- the expressions for the derivatives of all equations and their formulation in a high level language,
- positioning of the derivative terms in the sparse Jacobian matrix structure,
- append the new variables in the variable vector $x$ and in the updating $(x = x + dx)$ subroutine.

Introducing a new variable (e.g. the reactance of transmission lines $X_\text{x}$) in the variable set with the ADIFOR/SparsLinC concept requires the following steps:

- appending the new variable in the comma-separated list of independent variables in the ADIFOR\_CONTROL.adff file.
- declaring the gradient object $g_{X_1}$ and including it in the seed matrix
- updating $X_1(i)$ after solving the linear system in one Newton-Raphson iteration

These additions and modifications are totally generic and basically confined to the trivial definition of the new variables, while the rest of the work is taken over by the SparsLinC routines. To summarize, with the ADIFOR/SparsLinC concept, the power flow code developer still has to code the main, generic steps of a Newton-Raphson implementation. Also, he/she is responsible for setting up and calling the main step
of the Newton-Raphson, i.e. the sparse linear system of equations. The correct (and almost trivial) variable update code sections are not generated automatically, so these code sections must also be written by the power flow code developer. However, the power flow code developer is completely relieved of the coding of the formulas for the Jacobian matrix and the positioning of the Jacobian terms. Inserting new features and models is therefore easier, and the standardized, generic steps that are required make the code easily maintainable, flexible and robust.

2.6 Software engineering and simulation results

The concept as described in the previous chapters has been realized for a power flow code with the following features:

- remote/local voltage control
- limits on active, reactive generating power
- regulating transformers (phase shifters)
- MW area interchange control
- switching between polar and rectangular coordinates
- switching between current and power mismatch equations

All functions were formulated in the rectangular-coordinate system. The code for power flow mismatch computation including all features mentioned above has 750 executable statements including output purpose statements. The autogenerated code, which includes both the computation of the mismatches and the computation of all sparse Jacobian terms, has only 1207 executable statements.

The additional memory requirements of the data sections of the autogenerated subroutine is 872 kB static allocated memory (the code was compiled for networks up to 3000 nodes, 3500 lines, 2000 transformers) and 5140 kB dynamically allocated memory for the larger network tested (about 5400 variables). The dynamically allocated memory, which is related to the sparse Jacobian elements, was measured using the SparsLinC `xspmem` command. Note that this number includes all memory needed for the computation of temporary variables for the computation of the Jacobian terms. A small disadvantage of the ADIFOR/SparcLinC approach can be noted: Compared to hand-coded sparse Jacobian matrices of power flow equations, it becomes clear that the data sections of the autogenerated differentiated code use more memory. This disadvantage is, however, offset by the tendency of the computer industry to offer
increasing amounts of memory for less money. Even more important, this negative side-effect of more memory consumption is offset by an extremely flexible and fast (development speed related) handling of code dependencies.

The 2550 node network that was used in the calculations includes 3400 branches and 20 areas (5392 variables and 38131 non-zero Jacobian elements). Several sets of power flow variables (explicit currents, currents eliminated) and associated power flow equations have been programmed. All automatically generated Jacobian codes have performed without code generation errors. For the solution of the sparse linear system the general purpose sparsity package UMFPACK [29] was used.

To convey an idea of the efficiency of the power flow code using the ADIFOR package, it was compared with a power flow computation program (in C code) with hand-coded derivatives (PFLOW [30]). The following table displays the average (per Newton-Raphson iteration) CPU times in seconds for the computation of mismatches and Jacobian terms and for the solution of the linear system (Jacobian $\times dx = -$mismatch). The PFLOW code uses the polar-formulated, power-mismatch equations. In the resulting code the user has the options to choose between Polar and Rectangular for the Voltage variables and between Power and Current for the Mismatch formulation.

<table>
<thead>
<tr>
<th>Voltage, Mismatch</th>
<th>Power Flow Code</th>
<th>Mismatch+ Jacobian</th>
</tr>
</thead>
<tbody>
<tr>
<td>P,P</td>
<td>PFLOW</td>
<td>0.15</td>
</tr>
<tr>
<td>R,C</td>
<td>ADIFOR</td>
<td>0.84</td>
</tr>
<tr>
<td>R,P</td>
<td>ADIFOR</td>
<td>1.41</td>
</tr>
<tr>
<td>P,C</td>
<td>ADIFOR</td>
<td>1.13</td>
</tr>
<tr>
<td>P,P</td>
<td>ADIFOR</td>
<td>1.71</td>
</tr>
</tbody>
</table>

The machine used is a Sun Sparc Server 4000, 128 MB Memory, Solaris 2, UltraSparc based CPU, CPU clock rate 168 MHz, memory clock rate 83 MHz.

The mismatch equations were originally coded in the (R,C) formulation. The changes made to incorporate these two features (switch between polar and rectangular coordinates, switch between power and current mismatches) are described in the example of section 2.5.2, where data dependencies (detected automatically by ADIFOR) are exploited to calculate derivatives. These new dependencies (longer paths from dependent to independent variables) account for the slower computation of derivatives (0.84 sec per Jacobian - mismatch calculation for (R,C) to 1.71 sec for (P,P)). The fastest case is therefore the (R,C) since the equations have been formulated using the independent variables ($V_e, V_f, I_e, I_f$).
2.7 Conclusions

This chapter stated the need for computational differentiation in power system algorithms and focused mainly on the automatic code differentiation. The properties of the software package ADIFOR and its interface to a power flow code were described from the point of view of the software developer. ADIFOR was applied to a FORTRAN power flow code with activated handling of sparse code (calls to the SparsLinC routines). With several examples the resulting code was shown to have the following properties:

- **automatic code generation**
  The power flow code developer specifies standard FORTRAN77 code representing any set of power flow functions using any number of case statements, loops, etc. This set of functions is compiled/interpreted by ADIFOR and a new code is generated which includes the efficient computation both of the user specified functions and associated non-zero Jacobian terms. The programmer can specify which are the dependent and independent variables of the functions, for which derivative code should be computed.

- **sparsity handling**
  ADIFOR includes the capability of inherent sparsity detection of the Jacobian related code parts. It is the combination of Jacobian code generation and inherent sparsity consideration which allows the development of powerful power flow codes. Sparsity is exploited dynamically by use of C pointers (SparsLinC).

- **efficiency**
  The resulting code is fast (CPU time). However it is slower than conventional packages with hand coded Jacobian.

- **flexibility**
  The flexibility and efficiency (development time) in the maintenance and enhancement of the code is obvious: to add a new feature the user needs only to add the new equations in the powerflow.f FORTRAN subroutine, declare the new variables, append them in the script file ADIFOR_CONTROL.adf, and initialize them in the seed matrix related code part.

- **reliability**
  The use of ADIFOR renders the code more reliable by reducing the hand-coded parts and, therefore, the number of “bugs”.
3.1 Introduction

A basic requirement of any modern society is the economical and secure operation of its electrical power system. The secure operation of a power system can be defined as its ability to withstand certain contingencies. Economic operation in this context means that the network at the current state operates at the lowest generation cost or with minimum real power losses. To determine a state that combines the lowest operational cost with the guarantee that emergency conditions can be survived, the power system operators and planners perform security constrained optimization. The generalized term Optimal Power Flow (OPF) Problem is used to describe optimization problems in the power system area. Optimization and security analysis are complex and difficult tasks, and the computational effort tends to rise rapidly with increasing dimension and degree of freedom.

An OPF computation program is a useful tool for the utilities and the power industry. The output of the OPF analysis are the settings of the controls in the system. Implementation of these controls brings the system into the optimum state with regard to specific objectives. There are different modes of operation: in real-time mode the OPF determines the settings of the control variables (see section 3.2) which should be implemented immediately. The OPF code should be fast, reliable and robust. In study-mode, where long-term decisions are made, emphasis is placed on global optimality of the solution and handling of different objectives. In these cases computational speed is of minor importance, so that a more accurate modeling of the network is possible.

In this chapter the OPF problem will first be formulated (section 3.2.1), with emphasis on steady state optimization. The nonlinearity of the power flow equations makes the
OPF problem a non-linearly constrained optimization problem. The theory developed by Karush [31] and Kuhn and Tucker [32] provides the optimality conditions for a non-linearly constrained optimization problem (section 3.2.2) but offers no solution method. Research in the power system field has given rise to a number of techniques for obtaining a solution to this problem (see [33] for an overview). The purpose of this chapter is to present a software tool that solves the OPF problem. In this implementation, a general purpose optimization software package (MINOS) will be used. This chapter further describes the underlying theory of the optimization method used in MINOS (section 3.3.1) and the implementation details to obtain an OPF package (section 3.3.3). The automatic differentiation package ADIFOR will be used to obtain code for the first order derivatives (section 3.4). The interfaces of ADIFOR and MINOS, and their connection, will be described in detail. The properties of the OPF software package that combines these two general purpose software tools will be discussed in detail and with examples. Case studies at the end (section 5) will convey an idea of the performance of the resulting code.

3.2 The OPF Problem

3.2.1 Mathematical formulation

The OPF problem can be mathematically formulated as a non-linearly constrained optimization problem (small letters represent column vectors unless otherwise stated),

\[
\min \quad F(x, u) \\
\text{s.t.} \quad g_E(x, u) = 0, \\
\quad g_O(x, u) \leq 0, \\
\quad g_C(x, u) \leq 0,
\]

where the vector of variables is partitioned into the controllable quantities (control variables) \( u \) and the dependent (state) variables \( x \).

The objective function \( F(x, u) \) is scalar and is a model of the optimization goal that represents economic and security oriented interests of the power utility. The most common objective functions are:

- Minimum operating cost
- Minimum MW losses
- Maintainance of secure operation with minimum deviation of the control settings.
- Minimum or Maximum power transfer to/through an area
• Minimum installed FACTS MVA or MVAR power

The objective $F(x, u)$ is in general a nonlinear function.

• $g_E(x, u) = 0$ are the equality constraints. These are the conventional power flow equations:
  - the Kirchhoff laws
  - modeling equations of network components

• $g_O(x, u) \leq 0$ are the operating constraints. Most network state variables are not allowed to exceed certain lower and upper limits. These limitations are "soft" constraints and correspond to security and power quality based limitations and requirements. Some of the most common operating constraints are limitations on:
  - voltage magnitude at load buses
  - reactive power of PV-generators
  - branch currents, branch MW/MVAR/MVA flows
  - angle/voltage magnitude drop along a line.

• $g_C(x, u) \leq 0$ are the control variable constraints. Control variables do not exceed lower and upper limits. These can be "hard" constraints, especially when corresponding to the operating range of physical apparatus. The most common control variable constraints are:
  - tap magnitude or tap angle of regulating transformers
  - active generating power
  - voltage magnitude at PV buses
  - MW interchange transactions
  - shunt reactors or capacitors.

The vector of constraints consists of linear and nonlinear functions. The set of variables chosen determines the dimensions of the problem as well as the number of linear and nonlinear constraints.

### 3.2.2 Optimality conditions

The OPF problem is a nonlinear optimization problem and can take the general form

$$\begin{align*}
\text{min.} & \quad F(x) \\
\text{s.t.} & \quad g_E(x) = 0, \\
& \quad g_I(x) \leq 0,
\end{align*}$$

(3.2)
where $F(x)$ is the objective function, $g_E$ the vector of equality constraints and $g_I$ the vector of inequality constraints. To simplify the notations, the symbol $x$ is used for both control and state variables. It is assumed that all functions are twice differentiable and all variables are continuous.

First, the problem (3.2) with only equality constraints will be considered. The optimality conditions can be expressed in terms of the Lagrangian function

$$
\mathcal{L}(x, \lambda) = F(x) + \lambda^T g_E(x)
$$

where $\lambda$ is the vector of Lagrange multipliers.

The following two theorems state the optimality conditions for a minimum $x_*$ of the nonlinear optimization problem with only equality constraints $g_E(x) = 0$ [34, 35].

**Theorem 1 (Necessary optimality conditions for equality constraints)** Let $x_*$ be a local minimizer of $F$ subject to the constraints $g_E(x) = 0$. Let $Z(x_*)$ be a null-space basis\(^1\) of the Jacobian matrix $\nabla g_E(x_*)^T$. If $x_*$ is a regular point of the constraints\(^2\), then there exists a vector of Lagrange multipliers $\lambda_*$ such that

- $\nabla_x \mathcal{L}(x_*, \lambda_*) = 0$, or equivalently, $Z(x_*)^T \nabla F(x_*) = 0$ (first order optimality condition), and
- $Z(x_*)^T \nabla_{xx} \mathcal{L}(x_*, \lambda_*) Z(x_*)$ is positive semi-definite (second order optimality condition).

**Theorem 2 (Sufficient conditions for equality constraints)** Let $x_*$ be a point satisfying $g_E(x_*) = 0$. Let $Z(x_*)$ be a basis for the null-space matrix of the Jacobian matrix $\nabla g_E(x_*)^T$. If there exists a vector of Lagrange multipliers $\lambda_*$ such that

- $\nabla_x \mathcal{L}(x_*, \lambda_*) = 0$, and
- $Z(x_*)^T \nabla_{xx} \mathcal{L}(x_*, \lambda_*) Z(x_*)$ is positive definite,

then $x_*$ is a strict local minimizer of $f$ s.t. $g_E(x) = 0$.

---

\(^1\) Assume $A$ is a $m \times n$ matrix with $m \leq n$ and full row rank. Null space of $A$: $\mathcal{N}(A) = \{ p \in \mathbb{R}^n, Ap = 0 \}$. Basis $Z$ of $\mathcal{N}(A)$ is any matrix $Z_{n \times (n-m)}$ (not unique) that satisfies $AZ = 0$

\(^2\) the rows of $A$ are linearly independent at $x_*$
For a problem with linear equality constraints $Ax = b$ we have $\nabla g_E = A$ and, therefore, the first order optimality condition yields

$$\nabla f(x^*) = A^T \lambda^*$$  \hspace{1cm} (3.4)

For the nonlinear optimization problem (3.2) with only inequality constraints $g_I(x) \leq 0$ the following two theorems state the optimality conditions for a minimum $x^*$. The optimality conditions are expressed in terms of the Lagrangian function

$$\mathcal{L}(x, \mu) = F(x) + \mu^T g_I(x)$$  \hspace{1cm} (3.5)

where $\mu$ is the vector of Lagrange multipliers.

**Theorem 3 (Necessary conditions for inequality constraints)** Let $x^*$ be a local minimizer of $f$ subject to the constraints $g_I(x) \leq 0$. Let $Z(x^*)$ be a null-space matrix of the Jacobian matrix of the active constraints at $x^*$. If $x^*$ is a regular point of the constraints, then there exists a vector of Lagrange multipliers $\lambda^*$ such that

- $\nabla_x \mathcal{L}(x^*, \mu^*) = 0$, or equivalently, $Z(x^*)^T \nabla f(x^*) = 0$
- $\mu^* \geq 0$
- $\mu^*^T g(x^*) = 0$
- $Z(x^*)^T \nabla_{xx} \mathcal{L}(x^*, \mu^*) Z(x^*)$ is positive semi-definite.

**Theorem 4 (Sufficient conditions for inequality constraints)** Let $x^*$ be a point satisfying $g_I(x^*) \leq 0$. If there exists a vector of Lagrange multipliers $\mu^*$ such that

- $\nabla_x \mathcal{L}(x^*, \mu^*) = 0$,
- $\mu^* \geq 0$,
- $\mu^*^T g(x^*) = 0$,
- $Z_+(x^*)^T \nabla_{xx} \mathcal{L}(x^*, \mu^*) Z_+(x^*)$ is positive definite$^3$,

where $Z_+$ is a basis for the null space of the Jacobian matrix of the nondegenerate constraints (the active constraints with positive Lagrange multipliers) at $x^*$. Then $x^*$ is a strict local minimizer of $f$ s.t. $g(x) \geq 0$.

$^3$Note that the positive definiteness of the projected Hessian of the Lagrangian function, $Z_+(x^*)^T \nabla_{xx} \mathcal{L}(x^*, \mu^*) Z_+(x^*)$, ensures the convexity of $\mathcal{L}$ around the solution point and along certain feasible arcs [34]
The optimality conditions for constrained optimization with equality and inequality constraints are often called the *Karush-Kuhn-Tucker conditions* [31, 32] and can be derived in a straightforward way. They are used to verify that a point is a minimum, but they do not provide a solution method. Furthermore, they require smooth functions and continuous variables.

### 3.2.3 The OPF problem solution

In practice, the OPF problem has some properties that do not meet the prerequisites of the Karush-Kuhn-Tucker conditions, such as:

- certain objective functions are not smooth or not convex (generation cost curves of thermal power plants)
- variables move in discrete steps (transformer taps, conventional shunt reactance)
- practical requirements impose more than one objective, all of which have to be optimized simultaneously

In order to apply the Karush-Kuhn-Tucker conditions non-smooth functions are approximated with smooth models and all variables are considered continuous. After the solution to the continuous problem is found, certain adjustments are made that consider the discrete nature of the variables, or the breakpoints of functions. More than one objectives can be optimized simultaneously by optimizing a scalar objective made up of the weighted sum of the single objectives. The solution to the power flow problem is established and can be exploited to obtain a solution to the optimization problem. Traditionally two different approaches were developed, depending on the extent to which the power flow algorithm was used in the optimization process.

- **Class A**: These algorithms use the power flow algorithm separately from the optimization algorithm. The nonlinear problem is approximated and the optimality conditions of the approximated problem are solved. The starting point is a power flow solution and with an iterative process, the conventional power flow is solved anew in every iteration.

- **Class B**: Here, the power flow equations are integrated in the optimization algorithm as equality constraints. Instead of approximating the nonlinear problem, the optimality conditions of the original problem are solved.

The main advantage of Class A OPF algorithms is that they use stable optimization methods with straightforward techniques and no heuristic rules. The equality
and inequality constraints are linearized and the objective function is replaced with its linear or quadratic approximation. Therefore, Class A algorithms use Linear Programming (LP) or Quadratic Programming (QP) optimization techniques that handle equality and inequality constraints well, in combination with the conventional exact-AC power flow computation that employs the well known and stable Newton-Raphson technique. On the other hand, computational effort must be devoted to the approximation of the nonlinear functions (e.g. quadratic approximation of the total or partial transmission losses). Furthermore, convergence problems can result from a linear approximation of nonlinear functions.

Class B algorithms start with the full formulation of the exact optimality conditions. The goal is to find a solution for the optimality conditions that contain derivatives of the objective function and the constraints. The largest part of the equality constraints are the power flow equations, therefore every iteration of the optimization algorithm approaches both the power flow solution and the optimization solution. Handling inequality constraints is the main problem in solving (3.1). A number of Class B algorithms have been developed \(^4\) that in general result in problems of large dimension and high sparsity.

The OPF software tool described in this chapter uses a general purpose optimization package (see next section). The optimization method used falls into Class B, so that the power flow equations are solved implicitly.

### 3.3 MINOS package

MINOS (Modular In-core Nonlinear Optimization System) [37, 38] is a FORTRAN based general purpose optimization software package. It is designed to solve large-scale optimization problems (linear and nonlinear) in continuous variables. For linear problems, MINOS uses a sparse implementation of the primal simplex method (see section 3.3.2). For nonlinear objective functions and linear constraints, MINOS uses a reduced gradient method with quasi-Newton approximations to the reduced Hessian (see next section). For problems with nonlinear constraints, MINOS uses a projected Lagrangian method and solves a sequence of subproblems in which the constraints are linearized.

MINOS is suitable for the intended flexible OPF software environment since

- It can handle large-scale sparse nonlinear problems
- As a general purpose package has reliable and numerical robust algorithms

\(^4\) like augmented Lagrangian, interior point etc. [36, 34, 35]
• Has features like automatic scaling of linear and nonlinear constraints and warm start capability
• The algorithm used does not require user-coded second-order derivatives

This last feature is particularly beneficial, since ADIFOR, the software package used for providing the FORTRAN code for the first-order derivatives, is not capable - to this day- of providing sparse code for second-order derivatives.

The following sections will give a more detailed description of the algorithms used in MINOS. First, the Reduced Gradient method is applied to a nonlinear optimization problem with only equality constraints. This occurs as a subproblem within the active-set method of the augmented Lagrangian method that MINOS uses to solve optimization problems with nonlinear constraints.

### 3.3.1 Reduced Gradient method

The nonlinear optimization problem (3.6) is considered,

\[
\begin{align*}
\text{min.} & \quad F(x) \\
\text{s.t.} & \quad g(x) = 0 \\
& \quad l \leq x \leq u,
\end{align*}
\]

where \( g : \mathbb{R}^n \rightarrow \mathbb{R}^m \) is the vector of the nonlinear constraints and \( l,u \) are the lower-and upper-bound vectors for the variable vector \( x \). The variables \( x \) in (3.6) also contain \textit{slack} variables in addition to the original \textit{column} variables of (3.2), so that all inequalities are converted to equalities. To simplify the computation, the notation \( x \) is used for both cases.

For the solution of (3.6) the Lagrangian function is formulated

\[
\mathcal{L}(x, \lambda) = F(x) - \lambda^T g(x)
\]

The first-order optimality condition of Theorem 1 is

\[
\nabla \mathcal{L} = 0,
\]

and if Newton’s method is applied to the nonlinear system (3.8) at every iteration we solve the linear system (3.9) for \( \begin{pmatrix} p_k \\ v_k \end{pmatrix} \) and update the current iterate

\[
\begin{pmatrix} \nabla_x^2 \mathcal{L}(x_k, \lambda_k) \end{pmatrix} \begin{pmatrix} p_k \\ v_k \end{pmatrix} = -\nabla \mathcal{L}(x_k, \lambda_k) \\
\begin{pmatrix} \nabla_x \mathcal{L}(x_k, \lambda_k) & -g(x_k)^T \\ \nabla g(x_k) & 0 \end{pmatrix} \begin{pmatrix} p_k \\ v_k \end{pmatrix} = -\begin{pmatrix} \nabla_x \mathcal{L}(x_k, \lambda_k) \\ g(x_k) \end{pmatrix}
\]

(3.9)
\[
\begin{pmatrix}
  x_{k+1} \\
  \lambda_{k+1}
\end{pmatrix}
= \begin{pmatrix}
  x_k \\
  \lambda_k
\end{pmatrix} + \begin{pmatrix}
  p_k \\
  v_k
\end{pmatrix}
\] (3.10)

In the reduced gradient method these formulas are only used to derive that component of the search direction, that lies in the null space of the constraint gradients. Assume \( Z_k \) is a basis matrix for the null space of \( \nabla g(x_k)^T \) and \( Y_k \) a basis matrix for the range space of \( \nabla g(x_k)^5 \). Then

\[
p_k = Z_k p_Z + Y_k p_Y
\] (3.11)

where \( p_Z \) is the solution to the reduced system

\[
Z_k^T \nabla_x \mathcal{L}(x_k, \lambda_k) Z_k p_Z = -Z_k^T \nabla_x \mathcal{L}(x_k, \lambda_k)
\] (3.12)

In general,

\[
Z_k^T \nabla_x \mathcal{L}(x_k, \lambda_k) = Z_k^T \nabla F(x_k)
\] (3.13)

If all constraints are linear, then in addition,

\[
\nabla_x \mathcal{L}(x_k, \lambda_k) = \nabla^2 F(x_k)
\] (3.14)

and (3.12) can be rewritten as

\[
Z_k^T \nabla^2 F(x_k) Z_k p_Z = -Z_k^T \nabla F(x_k)
\] (3.15)

where \( Z_k^T \nabla^2 F(x_k) Z_k \) is the reduced Hessian and \( Z_k^T \nabla F(x_k) \) the reduced gradient of the objective function.

The range-space portion of the search direction will be determined from the condition that the new estimate of the solution must be feasible.

\[
g(x_k + Z_k p_Z + Y_k p_Y) = 0
\] (3.16)

If the constraints are linear then \( p_Y = 0 \).

---

5 Assume \( A \) is a \( m \times n \) matrix with \( m \leq n \):
- Null space of \( A \): \( \mathcal{N}(A) = \{ p \in \mathbb{R}^n, Ap = 0 \} \)
- Range space of \( A^T \): \( \mathcal{R}(A^T) = \{ q \in \mathbb{R}^n, q = A^T \lambda \text{ for some } \lambda \in \mathbb{R}^m \} \)

\( \mathcal{N}(A) \) and \( \mathcal{R}(A^T) \) are orthogonal subspaces whose dimensions sum to \( n \). So, every \( n \)-dimensional vector \( x \) can be written uniquely as the sum of a null-space and range-space component \( x = p + q, p \in \mathcal{N}(A) \) and \( q \in \mathcal{R}(A^T) \).
3.3.2 MINOS: Brief description of the algorithm

MINOS (Modular In-core Nonlinear Optimization System) is a FORTRAN based package designed to solve large-scale optimization problems in continuous variables, expressed in the standard form

\[
\begin{align*}
\min_{x,y} & \quad F(x) + c^T x + d^T y \\
\text{s.t.} & \quad f(x) + A_1 y = b_1 \quad (m_1 \text{ rows}) \\
& \quad A_2 x + A_3 y = b_2 \quad (m_2 \text{ rows}) \\
& \quad l \leq \begin{bmatrix} x \\ y \end{bmatrix} \leq u \quad (n + m \text{ variables})
\end{align*}
\]  

(3.17a) (3.17b) (3.17c) (3.17d)

where the vectors \(c, d, b_1, b_2, l, u\) and the matrices \(A_1, A_2, A_3\) are constant [38]. The standard form (3.17) emphasizes that nonlinearities in the objective function usually involve only a subset of the variables. All functions of \(x\) are considered twice differentiable with bounded Hessians. MINOS distinguishes between nonlinear (3.17b) and linear (3.17c) general constraints and constraints imposing bounds on variables (3.17d). Inequalities on general constraints can be rewritten as equalities with the introduction of logical (slack) variables with appropriate bounds according to the following principle (assume \(g_i(x)\) is a general constraint function):

\[
l_i \leq g_i(x) \leq u_i \quad \rightarrow \quad \begin{cases} 
\quad g_i(x) + s_i = 0 \\
\quad -u_i \leq s_i \leq -l_i
\end{cases}
\]

(3.18)

Slack variables are also introduced to the equality constraints, which are considered as inequalities with \(l_i = u_i\). Problem (3.17) has \(m\) general constraints (\(m_1\) nonlinear and \(m_2\) linear, \(m_1 + m_2 = m\)), \(n\) structural and \(m\) slack variables. The vectors \(x \in \mathbb{R}^{n_1}\) and \(y \in \mathbb{R}^{m_2 + m}\) contain the variables that are nonlinearly and linearly used (nonlinear and linear variables). The last \(m\) components of \(y\) form the vector \(s\) of slack variables and the last \(m\) columns of \(A_1\) and \(A_3\) form the identity matrix \(I\).

Depending on the structure of the problem, MINOS applies different algorithms:

- \(\mathbf{F}(x) = 0, f(x) = 0\) (in (3.17))

In this case the problem assumes the form (we denote the variable vector with \(x\),
instead of $y$):

$$\min_x c^T x$$

s.t. $Ax + Is = 0$

$$l \leq x \leq u$$

(3.19)

For linear programs MINOS employs a primal simplex method. The components of $x$ are partitioned at each vertex into a set of $m$ basic and $n - m$ nonbasic variables. If $x = [x_B^T \ x_N^T]^T$ where $x_B, x_N$ are the vectors of basic and nonbasic variables and the same partition is applied to the constraint matrix $[A \ I]$, the constraint equations are transformed to:

$$Bx_B = -Nx_N$$

(3.20)

where $B$ is a nonsingular square matrix called the basis matrix. The nonbasic (independent) variables are set to a value equal to one of their bounds and the basic variables are determined by solving the linear system (3.20). For the solution of (3.20) MINOS maintains (updates and occasionally recomputes) an $LU$ factorization of $B$. At each iteration the vectors $x_B$ and $x_N$ exchange one component until no further reduction of the objective can be achieved.

- $f(x) = 0, F(x) \neq 0$ (in (3.17))

The optimization problem is expressed in the following standard form:

$$\min_{x,y} F(x) + c^T x + d^T y$$

(3.21a)

s.t. $A \begin{bmatrix} x \\ y \end{bmatrix} = b$

(3.21b)

$$l \leq \begin{bmatrix} x \\ y \end{bmatrix} \leq u$$

(3.21c)

where the matrix $A$ is $m \times n$, $m \leq n$ and the vectors $c, d, b, l, u$ are constant.

MINOS uses an active constraint strategy with a portion of the general constraints and the bound constraints being active at any given time. The general constraints can be written in the form:

$$\begin{bmatrix} B & S & N \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_B \\ x_S \\ x_N \end{bmatrix} = \begin{bmatrix} b \\ b_N \end{bmatrix}$$

(3.22)

where $x_B, x_S, x_N$ are the basic, superbasic and nonbasic variables. $B$ is the nonsingular $m \times m$ basis matrix and $S$ the columns of $A$ corresponding to the superbasic variables.
The second equation of (3.22) indicates that the nonbasic variables are equal to one of their bounds. The basic and nonbasic variables are free to vary between their bounds and at any given time if a basic or superbasic variable encounters one of its bounds this variable is made nonbasic and the superbasic variables are reduced by one. The reduced gradient method is a generalized simplex method for nonlinear objectives.

The constraints (3.22) can be written
\[ \hat{A}\hat{x} = \hat{b} \] (3.23)
with
\[ \hat{A} = \begin{bmatrix} B & S & N \\ 0 & 0 & I \end{bmatrix}, \quad \hat{x} = \begin{bmatrix} x_B \\ x_S \\ x_N \end{bmatrix}, \quad \hat{b} = \begin{bmatrix} b \\ b_N \end{bmatrix} \] (3.24)

The operator
\[ Z = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} \] (3.25)
has full column rank and satisfies
\[ \hat{A}Z = 0 \] (3.26)

Therefore \( Z \) is a basis of the null space of the constraint matrix \( \hat{A} \). MINOS obtains a feasible descent direction (see (3.15)) for the superbasic variables by solving
\[ Z^T HZp_s = -Z^T g \] (3.27)
where \( g \) and \( H \) are the gradient vector and the Hessian of the objective function and \( Z^T g \) and \( Z^T HZ \) the reduced gradient and reduced Hessian. MINOS maintains a dense positive definite approximation of the reduced Hessian in the form \( R^T R \) where \( R \) is an upper triangular matrix. After (3.27) has been solved, the direction \( p \) for all variables will be \( p = Zp_s \) and the new update \( x_{k+1} = x_k + ap_k \) (where \( a \) is determined by a line search algorithm).

The operator \( Z \) and the inverse basis matrix \( B^{-1} \) are never computed explicitly. MINOS maintains a sparse \( LU \) factorization of \( B \) which allows an efficient computation of \( Z^T g \) and \( Zp_s \).

If the gradient vector of the objective function is partitioned using the same partition vector as in the variable vector, and \( \pi \) satisfies
\[ B^T \pi = g_B \] (3.28)
then

\[ Z^T g = -S^T (B^{-1})^T g_B + g_S = g_S - S^T \pi \]  

(3.29)

and

\[ p = Z p_s = \begin{bmatrix} -B^{-1} & S_p \end{bmatrix} \]

(3.30)

By using the \( LU \) factors of \( B \), the equation \( B p_B = -S p_S \) is solved for \( p_B \). The reduced objective has reached an optimum when the reduced gradient is zero. Then we have

\[
\begin{bmatrix}
  B^T & 0 \\
  S^T & 0 \\
  N^T & I
\end{bmatrix}
\begin{bmatrix}
  \pi \\
  \sigma
\end{bmatrix} =
\begin{bmatrix}
  g_B \\
  g_S \\
  g_N
\end{bmatrix}
\]

(3.31)

where the vectors \( \pi \) and \( \sigma \) are the vectors of the exact Lagrange multipliers for the active constraints. The components of \( \sigma \) are the reduced cost coefficients. In practice, optimization will terminate when \( Z^T g \) is sufficiently small, rather than zero. In this case \( \pi \) will simply be an approximation to the Lagrange multipliers for the general constraints.

- \( f(x) \neq 0, F(x) \neq 0 \) (in (3.17))

In this case MINOS uses a projected augmented Lagrangian method [37]. A sequence of major iterations is performed in which the constraints are replaced by their linear approximations (3.32) around the current iterate \( x_k \)

\[ \hat{f}(x) = f(x_k) + J_k (x - x_k) \]  

(3.32)

where

\[ J_k = \frac{\partial f}{\partial x} \bigg|_{x=x_k} \]  

(3.33)

and the objective function is augmented by two terms, the Lagrangian and the penalty term. The subproblem to be solved now takes the form:

\[
\begin{align*}
\min_{x,y} & \quad F(x) + c^T x + d^T y - \lambda_k^T (f(x) - \hat{f}(x)) + \frac{1}{2} \rho (f(x) - \hat{f}(x))^T (f(x) - \hat{f}(x)) \\
\text{s.t.} & \quad \hat{f}(x) + A_1 y = b_1 \\
& \quad A_2 x + A_3 y = b_2 \\
& \quad l \leq \begin{bmatrix} x \\ y \end{bmatrix} \leq u
\end{align*}
\]
where $\lambda_k$ is an estimate of the Lagrange multipliers at the $k$-th major iteration and $\rho$ the penalty parameter. This subproblem is linearly-constrained with nonlinearities in the objective, therefore a sequence of minor iterations of the reduced gradient approach (described above) is used for its solution. The optimum of the linearized subproblem is the optimal solution of the $k$-th major iteration, and provides an estimate for $\lambda_{k+1}, x_{k+1}$.

### 3.3.3 The MINOS interface to the OPF problem

With the subroutine version of MINOS [38] the optimization kernel can be invoked with a simple FORTRAN subroutine call

```fortran
  call minoss(start, m, n, nb, ne,
             nname, nncon, nnobj, nnjac, iobj, objadd, names, a, ha, ka, bl,
             bu, name1, name2, hs, xn, pi, rc, inform, mincor, ns, ninf,
             sinf, obj, z, nwcore)
```

where all problem data is passed as parameters. The most important input parameters are:

- **start**: specifies how a starting point is to be obtained, **m,n**: number of constraints and structural variables, **ne**: number of nonzero entries of the Jacobian matrix of constraints, **nncon**: number of nonlinear constraints, **nnjac, nnobj**: number of nonlinearly used variables (“nonlinear variables”) in the constraints and objective function, **a, ha, ka**: the Jacobian matrix stored in compressed sparse column format.

Beside the problem data, various runtime parameters that influence the solution process can be specified. Any of these (around 80) solver-specific directives assume their default values if not set by the user. Some of these directives are the maximum number of major and minor iterations, the penalty parameter, the feasibility tolerance and scaling. A very important directive for power system problems is the sparse storage of the Jacobian, that is activated with the following MINOS subroutine call

```fortran
  call miopt ('Jacobian Sparse', iprint, isumm, inform)
```

For linear problems the user has to first build up the (constant) Jacobian of the constraints and the gradient vector of the objective function. Then a call to subroutine **minoss** invokes MINOS. For a nonlinear problem at most two additional user-written subroutines are essential (Figure 3.3.3). If nonlinearities in the objective function exist ($F(x) \neq 0$ in (3.17a)), a FORTRAN subroutine **funobj** will calculate the objective function at the iterate $x_k$ and the gradient $\nabla F \big|_{x_k} = \left\{ \frac{\partial F}{\partial x_j} \big|_{x_k} , j = 1 \ldots nnobj \right\}$
of the nonlinear part with respect to the *nnobj* nonlinear variables. Likewise, if the constraints have a nonlinear part \((g(x) \neq 0)\) in (3.17b) then a FORTRAN subroutine *funcon* will compute the constraints at a given point \(x_k\) and the Jacobian submatrix \(\frac{\partial g}{\partial x} \mid_{x_k}\) of the nonlinear part of constraints with respect to the *nncon* nonlinear variables.

The software programmer should note that in a nonlinear problem (such as the OPF problem) a certain partitioning of the constraints should take place, so that the nonlinear constraints are the first *nncon* components of the set of constraints (3.17b). Likewise, the nonlinear variables of the objective (contained in \(F(x)\)) occupy the first *nnobj* places of the variable vector, followed by the *nnjac* nonlinear variables of the constraints (variables contained in vector \(f(x)\)) and the rest are the linear variables. The partitioning of the constraint and variable set is the responsibility of the developer and can be a non trivial task, depending on the dimension and complexity of the problem. MINOS can compute all or some of the derivatives with finite differences, but to achieve efficient code, the user should provide all derivatives.
For nonlinear problems the choice of the initial point is crucial for the success and efficiency of the algorithm, while different initial points can lead to different locally optimal solutions. A basis in this context refers to the pair of vectors that contain a compact representation of the values and states of the variables. The state $h_s(i)$ of a variable $x(i)$ at a point denotes if it is a basic, superbasic or nonbasic variable. Depending on whether a basis exists from a previous run and on whether the user wishes to use it to provide a good starting point for a given optimization problem, the input parameter `start` can have the values: 'Cold': MINOS ignores any information in the $x$, $h_s$ vectors or BASIS files and employs a Crash procedure to choose an initial basis for the problem, 'Warm': MINOS assumes that an initial basis is contained in the vectors $x$, $h_s$, or 'Hot': MINOS uses the basis provided by vectors $x$, $h_s$, as well the reduced Hessian and the basis factors. At the end of a successful run or a run that, although failed, is suspected to have delivered a point near the optimal solution, it is advisable to save the basis and use it to solve the same or a closely related problem. A closely related problem can be a problem with the same dimensions, same order of variables and perhaps different parameters (rhs vectors and coefficients).

### 3.4 The SW environment structure: MINOS combined with ADIFOR

As pointed out before, MINOS expects that the programmer will provide code for the first order derivatives that are contained in the subroutines `funcon` and `funobj`. These subroutines contain FORTRAN user-written code with formulas for the calculation of the functions (constraints, objective) and some or all partial derivatives. For every constraint function the user should supply formulas for all nonzero partial derivatives and the indices pointing to their place in the sparse Jacobian matrix. This is an increasingly difficult task as the code gets larger or if it is maintained by more than one programmer. In the case of integrating a new constraint, the necessary steps in a hand-coded OPF are: the formulation of the constraint function, definition of bounds, calculation of partial derivatives and positioning in the Jacobian matrix. In such a structure of interdepending code sections, propagating errors or simply failure to update some of the necessary parts result in bugs difficult to detect.

To relieve the software programmer of the task of derivative calculation and building-up of the Jacobian structure, the technique of automatic differentiation is used. The software package ADIFOR (section 2.3.3) allows automatic code differentiation for FORTRAN77 programs.

A major advantage of ADIFOR that renders it ideal for power system problems is the sparsity handling. When linked with a given C-language library called SparsLinC, nonzero entries of the Jacobian matrix are detected and calculated during code deriva-
Figure 3.2: Interface between MINOS and ADIFOR
tion time. By the use of pointers to the sparse representation of the variables, the Jacobian matrix can be extracted in a compressed sparse row format.

As presented in (2.5) a power-flow code has been implemented that uses ADIFOR-generated code for the calculation of the Jacobian in every Newton-Raphson iteration. In this environment the source code (FORTRAN77 subroutine powerflow) contains the power flow equations (network modeling equations, Kirchhoff current or power equations) and several features (fix area MW export, regulating transformers, ability to switch between polar and rectangular formulation etc.) that can all be applied to the optimization. The equality constraints of the OPF are the power flow equations, so that the existing source code could be used with only minor modifications as the subroutine constraints.f [39]. This FORTRAN77 subroutine contains the mathematical formulations of all constraint functions and the objective function.

The subroutine call to the ADIFOR generated code

```fortran
  call g_constraints
```

and a subsequent call to the extraction routine of SparsLinC provides the objective value, its gradient vector, the constraints values and the sparse Jacobian values and structure in compressed sparse row format. A format converter is used to convert this to the compressed sparse column format that MINOS needs. An additional part is necessary, since MINOS expects from funcon and funobj the calculation of the nonlinear terms. A for-loop over all calculated derivatives will subtract the linear part.

At this point a drawback of the combination of the two packages should be noted. MINOS wants the calculation of the objective and constraints (and their derivatives) in different subroutines, funcon and funobj. However, ADIFOR cannot separate the generated code into two independent subroutines. The interdependence path and application of the chain rule for derivative calculations are parts of the automatic differentiation process and cannot be detached. The constraints subroutine contains the mathematical formulations of all constraint functions and the objective function. This amounts to an unnecessary overload, since every time MINOS calls funobj for the calculation of the objective function and its gradient, the whole Jacobian is built anew.

For these reasons this optimization tool is slower than a similar tool with hand-coded derivatives in terms of executing (CPU) time. However the flexibility and modularity that has been achieved translates into programming-time speed. To give an insight into the properties of ADIFOR code, the following extract of the constraints subroutine (the FORTRAN subroutine that the user has to hand-code) is studied in more
do i = 1, ntie_data
  ktie = tie_br_indx(i)
  c The flow on the tie line
  c---------------------------------------------
  if (power_mismatch) then
    Ptie = P_2p_1(ktie)
  else
    Ptie = e_2p_1(ktie) * I_2p_1_e(ktie) +
    f_2p_1(ktie) * I_2p_1_f(ktie)
  endif
  c Add the tie flow to the appropriate area
  c---------------------------------------------
  SPexp_area(area_index(tie_metered_areano(i))) =
  & SPexp_area(area_index(tie_metered_areano(i))) +
  & Ptie
  SPexp_area(area_index(tie_nonmet_areano(i))) =
  & SPexp_area(area_index(tie_nonmet_areano(i))) -
  & Ptie
endo

This do-loop calculates the MW export of all areas by adding the flows on the tie lines. e_2p_1,f_2p_1 are the vectors of the real and imaginary parts of the voltage (in rectangular coordinates) of terminal 1 of lines (_2p_ stands for the two-port model of lines), I_2p_1_e, I_2p_1_f the vectors of the real and imaginary parts of the current of terminal 1 of lines, P_2p_1 the vector of real line flow (MW) measured on terminal 1, and SPexp_area the vector of total MW export of areas. If the option power_mismatch is activated (use of the power-mismatch and not current-mismatch equations) then P_2p_1 is as independent variable, otherwise the vector I_2p_1_e is an independent variable.

A constraint or objective function can be formulated by means of the intermediate (neither independent nor end-dependent) variable vector SPexp_area. An example of a constraint function would be to force the import of an area iarea, specified by the user, to a fixed value by inserting the constraint:

\[
\text{NLConstraintVector(nlcon)} = - \text{SPexp_area(iarea)} \\
\text{NLbu(nlcon)} = -\text{ImportValue} \quad \text{NLbl(nlcon)} = -\text{ImportValue}
\]

An objective function could be to minimize or maximize the MW export of area iarea.

```}

c if maximize export of area iarea
  FObjective = - SPexp_area(iarea)
c if minimize export of area iarea
  FObjective = SPexp_area(iarea)
```
The dependence path of the variable SPexp_area back to the independent variables is generated and updated during ADIFOR code generation and derivative information is propagated with the chain rule. This capability allows great flexibility in defining complex formulations as simple functions of intermediate variables. In the following section we will see how a new objective can be inserted with minimal standardized steps in certain parts of the code.

3.5 Adding features and modifying the structure

In this section the process of modifying the existing OPF/MINOS/AD code for inserting a new constraint or objective function or adding new variables will be described. Tables 3.1, 3.2 contain the most important user-specified options. These are not related with the optimization kernel (MINOS options) but determine:

- whether the power- or current-mismatch formulation of functions will be used
- whether the voltage magnitude and angle (polar=.true.) or voltage real and imaginary part are the independent variables.
- whether the MW losses (distributed_slack=.true.) will be distributed among several generators.
- whether the line X_l series reactances will be independent variables within bounds (X_lmin,X_lmax) (this is a simplified model of the TCSC FACTS device).
- whether the constraints for limiting the MVA flows on lines will be formulated (flow_ratings=.true.).

<table>
<thead>
<tr>
<th>Option name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>power_mismatch</td>
<td>logical</td>
</tr>
<tr>
<td>polar</td>
<td>logical</td>
</tr>
<tr>
<td>distributed_slack</td>
<td>logical</td>
</tr>
<tr>
<td>X_l_variable</td>
<td>logical</td>
</tr>
<tr>
<td>X_lmin,X_lmax</td>
<td>double</td>
</tr>
<tr>
<td>flow_ratings</td>
<td>logical</td>
</tr>
</tbody>
</table>

Table 3.1: General options

3.5.1 Standardized steps to include a new constraint

The following FORTRAN code extract is from the constraints subroutine and adds a new constraint posing an upper limit (the line MVA rating) on the MVA flow of lines at port 1

\[ S_{1p}^{2p} = \sqrt{(P_{1p}^{2p})^2 + (Q_{1p}^{2p})^2} \]
We can distinguish the following steps:

1. Mathematical formulation of the constraint as a function of independent and/or intermediate variables. In this case the variable vectors $P_{2p_1}$, $Q_{2p_1}$ are intermediate variables that have been formulated as a function of independent variables higher up in the code.

2. Depending on the degree of the nonlinearity of the function, storage of the constraint function to the corresponding vector. The vectors `NLConstraintVector` and `LNConstraintVector`, for nonlinear and linear constraints, are locally defined and at the end of the code are copied in the right order to the globally defined variable vector `ConstraintVector` (the input parameter to `minoss`).

3. Specification of the upper and lower limits according to (3.18). The vectors `NLbu`, `NLbl`, `LNbu`, `LNbl` are copied in the right order to the input parameters `bu`, `bl`.

4. Automatic calculation of first order derivatives by execution of ADIFOR for `constraints` subroutine and recompilation with the newly generated code.

Inequality constraints posing bounds on independent variables are formulated during initialization of the ADIFOR-seed matrix (see section 3.5.2).

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Type</th>
<th>Weights on objectives</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>System_losses_MW_min</td>
<td>logical</td>
<td>$W_{sysMWlosses}$</td>
<td>double</td>
</tr>
<tr>
<td>System_losses_MVAR_min</td>
<td>logical</td>
<td>$W_{parMWlosses}$</td>
<td>double</td>
</tr>
<tr>
<td>Partial_losses_MW_min</td>
<td>logical</td>
<td>$W_{max_loadability}$</td>
<td>double</td>
</tr>
<tr>
<td>Partial_losses_MVAR_min</td>
<td>logical</td>
<td>$W_{max_transfer_areas}$</td>
<td>double</td>
</tr>
<tr>
<td>HowManyAreasMin</td>
<td>[integer]</td>
<td>$W_{max_area_export}$</td>
<td>double</td>
</tr>
<tr>
<td>Area_losses_min</td>
<td>[list of integers]</td>
<td>$W_{TCSCMVA_total_fix}$</td>
<td>double</td>
</tr>
<tr>
<td>max_loadability</td>
<td>logical</td>
<td>$W_{TCSCMVA_total_min}$</td>
<td>double</td>
</tr>
<tr>
<td>max_transfer_areas</td>
<td>logical</td>
<td>$W_{min_generation_cost}$</td>
<td>double</td>
</tr>
<tr>
<td>max_area_export</td>
<td>logical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>min_generation_cost</td>
<td>logical</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: Options for objective functions

```python
if (flow_ratings) then
  do i = 1, num_2p_data
    NLConstraintVector(nlcon) = P_2p_1(i)**2 + Q_2p_1(i)**2 \\
    NLbu(nlcon) = 0.0 \\
    NLbl(nlcon) = -branch_MVA_rating_1(i)**2 \\
    nlcon = nlcon + 1
  enddo
endif
```
3.5.2 Standardized steps to include a new independent variable

Inserting a new (independent) variable is a relatively easy procedure. The following "do-loop" introduces the variable $X_L$ (line series reactance) in the seed matrix using the \texttt{dspsd} SparsLinC subroutine, and specifies upper and lower bounds.

\begin{verbatim}
    do i = 1, num_lines
        call dspsd(g_X_L(i),ipos,1.d0,1)
        bu(ipos) = X_max(i)
        bl(ipos) = X_min(i)
        ipos = ipos + 1
    enddo
\end{verbatim}

The $X_L$ is a constant parameter in conventional power system calculations and by considering it a variable, the degree of freedom of the optimization is increased. This is a simplified model to the series FACTS devices for control of power flow along a transmission path. From the programmer's point of view, the new partial derivatives of the objective and all constraint functions have to be recalculated and may not be trivial to obtain. With the OPF/MINOS/AD code (OPF/MINOS approach with derivatives calculated with the automatic differentiation technique) the user does not have to worry about Jacobian calculations. All the necessary modifications are restricted to the seed matrix initialization file:

1. Declaration and initialization of the new variable in the seed matrix (using the appropriate SparsLinC library call).
2. Specification of the bounds of the variable.
3. Storage in vector $x$ of the linear and nonlinear variables.
4. Execution of ADIFOR to update the seed matrix and regenerate the derivative code. Compilation of the newly generated code.

The developer should pay attention when storing the variable in vector $x$ (Step 3. above). The separation of variables in linear/nonlinear is not so trivial as in the case of general constraints. A variable which is linearly used with the existing constraints could be nonlinearly used in a newly inserted constraint. As already mentioned, this is the responsibility of the developer, and it is the only task restricting the modularity of the whole process. However, in power system problems, the high nonlinearity in classical compact formulations (with voltages and reactive generation as the main independent variables) relieves the user from this responsibility, since he/she deals with exclusively nonlinear equations.

3.5.3 Standardized steps to include a new objective

In the following, a number of existing objectives will be listed to provide a better insight into the modularity and flexibility of the OPF software tool. All code segments
are contained in the main subroutine constraints and all additions for inserting a new objective are restricted to this file.

• Losses of areas

\[
\begin{align*}
\text{if (SystemLossMW)} & \text{ then} \\
Wcoeff &= W_{\text{sysMWlosses}} \\
FObjective &= FObjective + Wcoeff \times SPsysLoss \\
\text{elseif (PartialLossMW)} & \text{ then} \\
Wcoeff &= W_{\text{parMWlosses}} \\
do i = 1, \text{HowManyAreasMin} & \\
FObjective &= FObjective + Wcoeff \times SPLoss_area(area_vec(AreaLossMin(i))) \\
enddo
\end{align*}
\]

The user has the option of choosing between minimization of losses of the whole network or of specific areas. The corresponding variables SPsysLoss and SPLoss_area have been defined and formulated higher up in the code.

• Generation cost

\[
\begin{align*}
\text{if (min_generation_cost)} & \text{ then} \\
Wcoeff &= W_{\text{min_generation_cost}} \\
addobj &= Wcoeff \times \text{TotGenCost} \\
FObjective &= FObjective + addobj \\
\end{align*}
\]

Higher up in the code the variable TotGenCost has been calculated by adding up the generators’ individual costs.

\[
\begin{align*}
\text{TotGenCost} &= 0.0 \\
do i = 1, \text{num_gen_data} & \\
kgen &= \text{bus_index(gc_genid(i))} \\
pgm &= \text{mvabase} \times (-P_1p(kgen) + P_1l(kgen)) \\
gc(i) &= gc_{fc}(i) \times (gc_{ai}(i) + gc_{bi}(i) \times pgm + \\
& gc_{ci}(i) \times pgm^2 + gc_{di}(i) \times pgm^3) \\
\text{TotGenCost} &= \text{TotGenCost} + gc(i) \\
enddo
\end{align*}
\]

• Power transfer between areas

\[
\begin{align*}
\text{objsum} &= 0. \\
\text{if (max_transfer_areas)} & \text{ then} \\
Wcoeff &= W_{\text{max_transfer_areas}} \\
do jj = 1, \text{num_obj_rec} & \\
itiearea &= \text{area_vec(tiearea(jj))} \\
\text{if (tiecoeff(jj) .eq. 1)} & \text{ then} \\
\text{objsum} &= \text{objsum} + objcoeff(jj) \times \text{SeTieAreaFrom(itiearea)} \\
\text{elseif (tiecoeff(jj) .eq. -1)} & \text{ then}
\end{align*}
\]
In this case we wish to optimize the MW power transfer between two user-specified areas. The variable $\text{tiearea}$ denotes the group of the tie lines that connect these two areas and $\text{SeTieAreaFrom}$ is the sum of real power flowing on the tie lines measured at area $\text{AreaFrom}$. Keeping in mind that MINOS by default minimizes, we set the appropriate sign on the variable $\text{SeTieAreaFrom(tiearea)}$ and store it to $\text{FObjective}$. This part has been easily extended to include optimization of reactive power flow and optimization of more than one pair of areas.

- **Total power import/export of area**

```plaintext
if (max_area_export) then
    Wcoeff = W_max_area_export
    do jj = 1, num_obj_rec
        iarea = area_vec(objarea(jj))
        FObjective = FObjective + Wcoeff * objcoeff(jj) * Pout_area(iarea)
    enddo
endif
```

By specifying one of the options

- $\text{SystemLossMW}$, $\text{PartialLossMW}$, $\text{max_transfer_areas}$, $\text{max_area_export}$

as being true, one of the above functions is activated. Multi-objective optimization can be achieved by activating more than one option, whereas all individual functions are accumulated in $\text{FObjective}$.

Considering the above examples it is easy to assess the benefits of the OPF/MINOS/AD development approach: it is more flexible in alternative formulations of functions (e.g. one may switch from current to power mismatch equations, or from polar- to rectangular-coordinate formulations in the code, regardless of the independent variables). Furthermore, it is easy to extend and maintain the code. Including a new feature requires changes in certain code sections irrespective of the problem structure. For these reasons development speed increases significantly in comparison to the hand-coded OPF. Execution speed suffers mainly because of the approximation of second order derivatives and the unnecessary rebuilding of the Jacobian with every call of $\text{funobj}$.
3.6 Conclusions

The OPF problem is a large-scale, nonlinear sparse optimization problem. To obtain a robust and reliable OPF solution package the user has to provide in general code for the

- Optimization algorithm
- Functions (constraints, objective functions)
- First-order derivatives of functions and sparsity structure
- Second-order derivatives of functions and sparsity structure

Making use of a general purpose optimization package has the main advantage that a robust optimization algorithm is provided and various features (scalability, warm-start capability) are already included. However, a generic optimization algorithm can fail in exploiting the special structure and properties of the power system OPF problem. MINOS, in particular, employs an augmented Lagrangian method combined with the reduced gradients and for a reliable solution requires only first-order derivatives and their sparsity pattern.

Using an AD package, derivatives are generated automatically and the user can concentrate on the modeling of the network and the objective functions. ADIFOR, the AD package used in this implementation, generates FORTRAN code for first-order derivatives of functions. It is currently not possible to generate sparse second order derivatives with ADIFOR and therefore an optimization package is chosen, that can approximate second order derivatives.

Finally, the three implementations of an OPF solution package

1. OPF/HC: OPF algorithm, functions and sparse derivatives hand coded (HC)
2. OPF/MINOS: MINOS optimization algorithm but functions and sparse derivatives hand coded
3. OPF/MINOS/AD: functions hand coded, MINOS optimization algorithm and AD software generates code for sparse derivatives

are compared in terms of development and execution time.

- The OPF/HC code is more flexible in specifying features in the optimization kernel, while in the OPF/MINOS code the algorithm, features and options are confined to the MINOS capabilities.
• The OPF/MINOS code is easier to extend and to maintain. The power flow and network related formulations are the same in both codes, but in the first case the user is relieved from second order derivative calculations (MINOS approximation) and in an extreme case he/she can omit first order derivative calculation as well. However, in large nonlinear problems like the OPF, Jacobian sparse structure and Jacobian terms must be provided.

• Depending on the algorithm chosen in the OPF/HC and the size of the problem, the execution speed varies significantly, and it is likely that with a robust and efficient hand-coded optimization kernel the OPF/HC is faster. The development time, however, to code a robust optimization kernel in comparison to the time to build up the interface to MINOS is significantly larger.

• The OPF/MINOS/AD implementation is the most flexible and efficient in development time. The code parts requiring hand coding are confined to the modeling equations and to the building of interfaces between the different packages. However, the execution time can suffer mainly because of the MINOS algorithm and the unnecessary rebuilding of the Jacobian each time the objective function is evaluated.
Chapter 4

Series-connected FACTS devices

As stated in the introductory chapter there is a large incentive for the utilization of new power semiconductor technology in order to increase the network loadability. The fast reaction of modern semiconductor devices allows for high levels of series compensation and fast control of certain network variables.

Purpose of this chapter is the

1. modeling of series-connected devices employing advanced semiconductor technology
2. description of the process of integrating these models in the existing OPF code (chapter 3).

4.1 Compensation and line loadability

As stated in section 1.2, the thermal limit determines the maximum allowable line current in normal operation. However, the actual maximum current in medium-length and long lines is imposed by voltage drop and steady-state stability margins. Long lines, in particular, are operated at levels much lower than their capacity\(^1\).

In this section, it will be described how compensation and phase angle control can help to increase long distance power transfers. The transmitted power \(P_t\) for an uncompensated line with sending end voltage \(E_S\) and receiving end voltage \(E_R\) (assuming transmission angle \(\delta\) and neglecting losses) is

\[
P_t = \frac{E_S \cdot E_R}{X_L} \sin \delta
\]  

\(^1\)In this context (power) capacity refers to the maximum power transmitted considering only the equipment rating limitations, while capability considers all types of network constraints
Compensation is a well established technology and can be divided into the following three categories:

- **Shunt compensation:** Capacitors and reactors connected in shunt with the transmission line provide shunt compensation. By controlling the current the voltage magnitude at the point of connection can be regulated (see Figure 4.2(a)). Shunt capacitors are installed far away from the strong buses to help with voltage depressions. In Figure (4.1) the shunt compensator is placed at the midpoint of the line and it is assumed that there is no series compensation ($E'_S = E_S$ for $X_C = 0$). Assuming that the shunt compensator can control the midpoint voltage $E_M = E_S$, the line is divided into two independent parts of half the length, and the transmitted power is

$$P_t = 2 \cdot \frac{E_S \cdot E_R}{X_L} \sin \frac{\delta}{2} \quad (4.2)$$

Comparing the $P - \delta$ curves in Figure 4.3 it is obvious that the steady-state maximum has increased.
• **Series reactive compensation** (or *impedance-type, X-type*): Series capacitors provide this type of compensation. The compensating effect results from the voltage drop across the series capacitor caused by the line current (see 4.2(b)). The effective reactance between $E_R$ and $E_S$ is $X_L - X_C$ and the steady-state maximum is

$$P_{t_{\text{max}}} = \frac{E_S \cdot E_R}{X_L - X_C}.$$  \hspace{1cm} (4.3)

Thus, the steady-state maximum increased by $\frac{X_L}{X_L - X_C}$ comparing to the steady-state maximum of the uncompensated line.

• **Phase angle control** ($\phi$-type): Phase shifting transformers are often installed in order to increase the effective phase angle difference (see 4.2(c)) between the sending and receiving end voltage vectors. This principle is demonstrated in Figure 4.3, where the $\phi$-type compensation $P - \delta$ curve is "shifted" to the left, permitting higher MW flows for smaller transmission angles. However, the maximum transmittable power is independent of the transmission angle.

With series and shunt compensation the power systems operator controls all the network variables and parameters that have a direct effect on MW line flow: the bus voltage, the line impedance and the transmission angle.

The compensating effect, as demonstrated in the voltage vector diagrams 4.2(a)...(c), is the same, regardless of whether the inserted capacitor or reactor banks are fixed, mechanically switched or thyristor switched. However, the flexibility of the system is significantly larger with thyristor-controlled components. The following table con-
tains examples of applications for conventional and thyristor controlled series capacitors, that are relevant to maximizing the steady-state power transfer.

<table>
<thead>
<tr>
<th>Conventional</th>
<th>Thyristor-controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increasing stability</td>
<td>Further increase of transient stability</td>
</tr>
<tr>
<td>Increasing long distance</td>
<td>Limiting fault current</td>
</tr>
<tr>
<td>power transfer</td>
<td>Damping power swings</td>
</tr>
<tr>
<td></td>
<td>Damping subsynchronous resonance</td>
</tr>
<tr>
<td></td>
<td>Dynamic control of power flow</td>
</tr>
<tr>
<td></td>
<td>Further increase of long distance power transfer</td>
</tr>
</tbody>
</table>

The modern high-power semiconductor technology has developed devices with response times fast enough to handle successfully dynamic system conditions. These devices allow for narrower security margins, permitting higher MW flows in normal operation. Furthermore, thyristor-controlled series capacitors can help damping subsynchronous resonance [4, 5], increasing significantly the maximum allowed compensation level. Within the framework of the FACTS program\(^2\) a number of series and shunt connected devices have been realized (see [40] for an overview) that provide real-time control of the transmission voltage, impedance, and phase angle.

FACTS controllers can be divided in the following two categories:

- **Thyristor-controlled**: they consist of conventional capacitor or reactor banks, but the traditional switching circuits have been replaced with thyristor switches.

- **Controllers with solid-state voltage-sourced inverters**: they consist of GTO-based synchronous voltage sources (see section 4.3.1) and can internally generate the necessary reactive power, without capacitors and reactors. With an energy storage device they can even exchange real power with the system.

Table 4.1 on page 71 provides a list of the most powerful FACTS controllers categorized with respect to their implementation and the type of compensation they provide.

In this section the steady-state models of two series connected devices will be described: The Thyristor Controlled Series Capacitor (TCSC) belongs to the impedance-type compensators. The Unified Power Flow Controller (UPFC), the most versatile of the FACTS controllers, provides both series and shunt compensation and can provide simultaneously or sequentially all types of compensation.

\(^2\) initiated by EPRI [7] in cooperation with a number of host utilities and manufacturers
4.2 The Thyristor Controlled Series Capacitor (TCSC)

4.2.1 Introduction

The TCSC was the first series FACTS controller to be developed. The main control unit is a thyristor controlled reactor (TCR). The TCR is a static var controller that by means of a power electronic interface provides fast control over the reactive power. The main (controllable) segment is a fixed inductor in series with a bipolar thyristor switch. By means of the firing angle of the thyristors the effective inductive reactance varies and causes rapid reactive power exchange between the TCR and the system. Depending on the system conditions inductive or capacitive vars may be needed. To meet this requirement the variable inductor is usually connected in parallel with a fixed capacitor (see Figure (4.4)). A metal-oxide varistor is also connected in parallel for overvoltage protection.

The basic principle is the fast adjustment of the apparent line impedance so that spe-
cific system performance criteria are met. By controlling the TCSC reactance one can compensate the line impedance and thus allow increased power flow through the line. Conventional series compensation is traditionally accomplished with switched capacitors, however, mechanical switching means slow and non-smooth control. Thyristor control provides rapid and smooth variance of the line impedance, necessary for power swing damping and power flow regulating on adjacent paths. In the fixed-capacitor, thyristor controlled reactor scheme of Figure (4.4), the degree of series compensation is controlled by increasing or decreasing the thyristor conduction period, and thereby the current through the TCR. The firing angle \( \alpha \) of the TCR is defined as the angle in electrical degrees between the positive-going zero-crossing of the voltage across the inductor and the positive going zero-crossing of the current through it. Firing angles below 90° have no control over the inductor current, while firing angles above 180° are not allowed because the two thyristors of the valve must be fired symmetrically.

4.2.2 Circuit analysis and operating principles of TCSC

In the following it will be assumed that the line current in Figure 4.5 is \( i_{\text{Line}}(t) = \cos(\omega t) \) and the voltage drop across the conducting thyristor valve is zero. The angle \( \sigma \) is defined as \( \sigma = \pi - \alpha \). All calculations assume steady-state.

\[
\begin{align*}
i_{\text{Line}}(t) & = i_T(t) + i_C(t) \\
\cos(\omega t) & = i_T(t) + C \frac{\partial u_C(t)}{\partial t} = i_T(t) + LC \frac{\partial^2 i_T(t)}{\partial t^2} \\
\frac{s}{s^2 + \omega^2} & = I_T(s) + LC(s^2 I_T(s) - s i_T(0^+) - i_T'(0^+) )
\end{align*}
\] (4.4)

Figure 4.5: Per-phase TCSC
Defining $\omega_o = 1/\sqrt{LC}$ the equation (4.4) yields:

$$I_T(s) = \frac{s}{s^2 + \omega^2} \frac{\omega_o^2}{s^2 + \omega_o^2} + \frac{s}{s^2 + \omega_o^2} i_T(0+) + \frac{1}{s^2 + \omega_o^2} i_T'(0+) \mathcal{L}^{(-1)}$$

$$i_T(t) = \frac{\omega_o^2}{\omega_o^2 - \omega^2} \cos(\omega t) - \frac{\omega_o^2}{\omega_o^2 - \omega^2} \cos(\omega_o t) +$$

$$i_T(0+) \cos(\omega_o t) + \sin(\omega_o t) i_T'(0+)$$

(4.5)

With $\rho = \omega_o/\omega$ and $A = \omega_o^2/(\omega_o^2 - \omega^2) = \rho^2/\rho^2 - 1$ (4.5) can be rewritten as:

$$i_T(\omega t) = A \cos(\omega t) - A \cos(\omega_o t) + i_T(0+) \cos(\rho \omega t) + \frac{\sin(\rho \omega t)}{\omega_o} i_T'(0+)$$

(4.6)

In steady state the current through the inductor $i_T$ is zero for $\omega t = -\sigma$ and $\omega t = \sigma$. Substituting for $t = -\omega/\sigma$ and $t = \omega/\sigma$ in (4.6) yields:

$$i_T(0+) = A \cos \sigma = \frac{\cos \rho \sigma - \cos \sigma}{\cos \rho \sigma}, \quad i_T'(0+) = 0$$

(4.7)

Substituting for $i_T(0+)$ and $i_T'(0+)$ in (4.5)

$$i_T(\omega t) = A \left( \cos \omega t - \frac{\cos \sigma}{\cos \rho \sigma} \cos \rho \omega t \right)$$

(4.8)

The voltage across the capacitor is:

$$u_C(t) = L \frac{di_T(t)}{dt} = -AX_L \sin \omega t + \frac{Ax_L \cos \sigma}{\cos \rho \sigma} \sin \rho \omega t$$

(4.9)

For $\sigma \leq \omega t \leq \pi - \sigma$ the thyristor valve does not conduct. Therefore:

$$i_{Line}(t) = C \frac{du_C(t)}{dt} \Rightarrow \cos(\omega t + \sigma) = C \frac{du_C(t)}{dt} \mathcal{L}$$

$$U_C(s) = \frac{1}{sC} \frac{\cos \sigma - \omega \sin \sigma}{s^2 + \omega^2} + \frac{1}{s} u_C(\sigma+)$$

(4.10)

By substituting for $u_C(\sigma+)$ and considering the angle shift the inverse Laplace transformation yields:

$$u_C(t) = X_C(\sin \omega t - \sin \sigma) - AX_L(\sin \sigma - \rho \cos \sigma \tan \rho \sigma)$$

(4.11)

Similar calculations for $\pi - \sigma \leq \omega t \leq \pi + \sigma$ yield the analytic expressions for
\[ i_T(\omega t) = \begin{cases} 
A \left( \cos(\omega t) - \frac{\cos \sigma}{\cos \rho \sigma} \cos(\rho \omega t) \right) & \text{for } -\sigma \leq \omega t \leq \sigma \\
0 & \text{for } \sigma \leq \omega t \leq \pi - \sigma \\
A \left( \cos(\omega t) + \frac{\cos \sigma}{\cos \rho \sigma} \cos(\rho(\omega t - \pi)) \right) & \text{for } \pi - \sigma \leq \omega t \leq \pi + \sigma 
\end{cases} \] (4.12)

\[ u_C(\omega t) = \begin{cases} 
-AX_L \sin \omega t + A\rho X_L \cos \sigma \sin \rho \omega t & \text{for } -\sigma \leq \omega t \leq \sigma \\
X_C(\sin \omega t - \sin \sigma) - A\rho X_L \sin(\sigma - \rho \cos \sigma \tan \rho \sigma) & \text{for } \sigma \leq \omega t \leq \pi - \sigma \\
-AX_L \sin \omega t - A\rho X_L \cos \sigma \sin(\rho(\omega t - \pi)) & \text{for } \pi - \sigma \leq \omega t \leq \pi + \sigma 
\end{cases} \] (4.13)

Fourier analysis of (4.13) yields the fundamental component of the capacitor voltage. Since the line current is sinusoidal, it is easy to derive the TCSC fundamental impedance.

\[ X_{TCSC} = X_C - (X_C + X_{LC}) \frac{2\sigma + \sin 2\sigma}{\pi} + \frac{4X_{LC}^2 \cos^2 \sigma}{X_L} \left( \frac{\rho \tan \rho \sigma - \tan \sigma}{\pi} \right) \] (4.14)

where \( X_{LC} = \frac{X_C X_L}{X_C - X_L} \).

In (4.14) the sign of the capacitive impedance is positive. Figure (4.6) displays the analytical result of (4.14) in p.u. of \( X_C \) for two different values of \( \rho \). It is easy to identify the two operating regions, capacitive and inductive, and the single resonance region, where the TCR reactance for certain firing angles becomes almost equal to the capacitor’s reactance. For certain values of \( \rho \) there can be more than one resonance region [41, 42, 43].

Figure (4.7) displays the capacitor voltage and TCR current waveforms for a firing angle \( \alpha \) of 120° (inductive operation), 155° and 170° (capacitive operation). For
$\alpha > 90^\circ$ the TCR current is not pure sinusoidal and for increasing $\alpha$ the amplitude of the current waveform is decreased.

Figure 4.6: Fundamental impedance of the TCSC with two different $\rho = \sqrt{X_C/X_L}$ ratios.

Figure 4.7: Capacitor voltage and thyristor current waveforms for different thyristor firing angles.
Figure (4.8) depicts the steady-state TCSC equivalent reactance versus firing angle characteristic for a ratio $\rho = 2.06$. For continuous gating of the thyristors ($\alpha = 90^\circ$) the TCSC device works in the *thyristor bypassed mode* where the net impedance is almost equal to the inductor’s reactance ($|X_L| \ll |X_C|$). For $\alpha = 180^\circ$ the thyristors are blocked (*thyristor blocked mode*) and the effective reactance is the reactance of the capacitor. The device can be continuously controlled (*vernier control mode*) in the inductive ($90^\circ \leq \alpha \leq 129^\circ$) and capacitive region ($142^\circ \leq \alpha \leq 180^\circ$), avoiding the steady-state resonance region. In the capacitive region the $X_{TCSC}$ can take any value from $1.0X_C$ up to $3.0X_C$ and in the inductive region from $X_{bypass} (< 0)$ down to $-2.0X_C$. The $X_{TCSC}$ of a single-module TCSC unit can not take any value between $0.0$ and $1.0X_C$.

Figure (4.9) depicts the steady-state TCSC equivalent reactance versus line current characteristic, where most of the physical and operating limitations of the TCSC reactance are displayed:

- A, D: resonance region limitation
- B: firing angle limitation ($\alpha = 180^\circ$, thyristor blocked mode)
- C: firing angle limitation ($\alpha = 90^\circ$, thyristor bypassed mode)
- E, F: upper voltage limits for capacitive and inductive operation
- G: maximum allowed current in continuous operation.

A controlled series capacitor installation typically consists of two components (Figure 4.10): because of cost considerations one element will be a mechanically switched conventional series capacitor, the second portion being thyristor-controlled [44, 45].
The thyristor-controlled segment usually consists of a number of TCSC modules connected in series. Each module is controlled independently. The main advantage from breaking up the TCSC reactance in segments is that the total effective TCSC reactance can achieve a wider range of values, since the control gap appearing in the one-module operation can now be covered (see Figure 4.11).

![Figure 4.10: TCSC configuration](image)

![Figure 4.11: X – I TCSC characteristic for increasing number of modules](image)
4.2.3 Two-port modeling equations

A two-port network has four variables: the two port voltages $U_{1}^{2p}, U_{2}^{2p}$ and the two port currents $I_{1}^{2p}, I_{2}^{2p}$ (see Figure 4.12). There are different representations of the two-port modeling equations depending on which variables are considered as independent [46, 24]. In general a two port is described with a $2 \times 2$ matrix with complex elements. The main two representation that will be used in this chapter are:

- The ABCD or chain matrix $A$ representation:

$$
\begin{bmatrix}
U_{1}^{2p} \\
I_{1}^{2p}
\end{bmatrix}
= A \cdot 
\begin{bmatrix}
U_{2}^{2p} \\
I_{2}^{2p}
\end{bmatrix}
= A \cdot 
\begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix}
\cdot 
\begin{bmatrix}
U_{2}^{2p} \\
-I_{2}^{2p}
\end{bmatrix}
$$

(4.15)

- The admittance matrix $Y$ representation:

$$
\begin{bmatrix}
I_{1}^{2p} \\
I_{2}^{2p}
\end{bmatrix}
= Y \cdot 
\begin{bmatrix}
U_{1}^{2p} \\
U_{2}^{2p}
\end{bmatrix}
$$

(4.16)

The sets of equations (4.15) and (4.16) can be derived directly. However, in certain two-port interconnections (like the cascade connection in the next section) it is convenient to calculate first the chain parameters and after performing algebraic calculations on those obtain form (4.16).

4.2.4 The two-port model of the TCSC

In the two-port TCSC model as illustrated in Figure (4.13) the variable capacitor reactance represents the variable effective reactance of the TCSC.

It is assumed that:

- $X_{TCSC} \leq 0$: capacitive region
- $X_{TCSC} > 0$: inductive region
The controllable variable $X_{TCSC}$ can take any value in the region
\[ \mathcal{E} = \{ X_{TCSC}, \]
\[ X_{FA}^A \leq -X_{TCSC} \leq X_{FA}^A, \]  
\[ -I_{2p}^p X_{TCSC} \leq U_{cap}^{max}, \text{ cap. operation}, \]  
\[ I_{2p}^p X_{TCSC} \leq U_{ind}^{max}, \text{ ind. operation}, \]  
\[ I_{2p}^p \leq I_{TCSC}^{max} \} \]  
\[ (4.17) \]
\[ (4.18) \]
\[ (4.19) \]
\[ (4.20) \]

where
- $X_{FA}^{min}, X_{FA}^{max}$: the upper and lower limits for $X_{TCSC}$ caused by firing angle limitations
- $U_{cap}^{max}, U_{ind}^{max}$: the voltage limits for capacitive and inductive operation
- $I_{TCSC}^{max}$: the current limit through the device

Solving the Kirchhoff equations for $I_{1p}^{2p}, I_{2p}^{2p}$ yields:
\[ U_{1p}^{2p} - U_{2p}^{2p} + jX_{TCSC}I_{1p}^{2p} = 0 \]  
\[ (4.21) \]
\[ I_{1p}^{2p} + I_{2p}^{2p} = 0 \]  
\[ (4.22) \]

The equations (4.21), (4.22) can be brought into the admittance form
\[ \begin{bmatrix} I_{1p}^{2p} \\ I_{2p}^{2p} \end{bmatrix} = \begin{bmatrix} \frac{1}{iX_{TCSC}} & \frac{1}{iX_{TCSC}} \\ \frac{1}{iX_{TCSC}} & \frac{1}{iX_{TCSC}} \end{bmatrix} \cdot \begin{bmatrix} U_{1p}^{2p} \\ U_{2p}^{2p} \end{bmatrix} \]  
\[ (4.23) \]
4.2.5 The two-port model of the TCSC-embedded transmission line

Figure 4.14 depicts a TCSC device in series with a transmission line. In this section the two-port equations (4.24) of the overall network TCSC, Line of a TCSC-embedded transmission line will be derived. It will be assumed that the TCSC is positioned at Port 1.

\[
\begin{bmatrix}
U_{1p}^2 \\
I_{1p}^2
\end{bmatrix} = A_{TCSC,Line}^2 \cdot \begin{bmatrix}
U_{2p}^2 \\
I_{2p}^2
\end{bmatrix}
\] (4.24)

The two-ports of the TCSC and transmission line are connected in cascade. The

![One line diagram of TCSC-embedded transmission line](image)

chain matrix of two-ports in cascade is equal to the product of the chain matrices of the individual networks. For this reason, the chain parameters of the overall network will be first calculated, and from these the admittance parameters can be derived with algebraic computations.

The A-matrix set of equations for the TCSC are given in Figure 4.13. The same set of equations for the transmission line are:

\[
\begin{bmatrix}
U_{1p}^2 \\
I_{1p}^2
\end{bmatrix} = A_{Line} \cdot \begin{bmatrix}
U_{2p}^2 \\
I_{2p}^2
\end{bmatrix}
\]

With

\[
A_{Line} = \begin{bmatrix}
a_{11}^{{Line}} & a_{12}^{{Line}} \\
a_{21}^{{Line}} & a_{22}^{{Line}}
\end{bmatrix}
\]

\[
a_{11}^{{Line}} = \frac{G_l + iB_l + iB_c}{G_l + iB_l}
\]

\[
a_{12}^{{Line}} = -1
\]

\[
a_{21}^{{Line}} = \frac{2iG_lB_c - 2B_lB_c - B_c^2}{G_l + iB_l}
\]

\[
a_{22}^{{Line}} = \frac{-G_l - iB_l - iB_c}{G_l + iB_l}
\]
Chapter 4: Series-connected FACTS devices

With

\[ I_C = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]

the \( A_{TCSC,Line}^{2p} \) is (see (4.15))

\[
A_{TCSC,Line}^{2p} = A_{Line} \cdot I_C \cdot A_{TCSC} \cdot I_C = \begin{bmatrix} a_{11}^{Line,TCSC} & a_{12}^{Line,TCSC} \\ a_{21}^{Line,TCSC} & a_{22}^{Line,TCSC} \end{bmatrix}
\]

with

\[
a_{11}^{Line,TCSC} = \frac{G_i + iB_i + iB_c}{G_i + iB_i} + \frac{iX_{TCSC}(2iG_1B_c - 2B_1B_e - B_c^2)}{G_i + iB_i}
\]

\[
a_{12}^{Line,TCSC} = \frac{-1}{G_i + iB_i} + \frac{iX_{TCSC}(-G_i - iB_i - iB_c)}{G_i + iB_i}
\]

\[
a_{21}^{Line,TCSC} = \frac{2iG_1B_c - 2B_1B_e - B_c^2}{G_i + iB_i}
\]

\[
a_{22}^{Line,TCSC} = \frac{-G_i - iB_i - iB_c}{G_i + iB_i}
\]

The algebraic calculations necessary for bringing the modeling equations in the form

\[
\begin{bmatrix} I_C^{2p} \\ I_C^{2p} \end{bmatrix} = Y_{TCSC,Line}^{2p} \begin{bmatrix} U_{C}^{2p} \\ U_{L}^{2p} \end{bmatrix}
\]

are not trivial, therefore the symbolic package Maple was used. Using the fortran command of Maple, the expressions can be extracted in Fortran and can be directly copied to the subroutine.

The limits of the total \( X_{TCSC} \) of a multi-module TCSC installation are as displayed in Figure (4.15) \(^3\) and explained in Table (4.2). Note that \( X_{max}, X_{min}, U_{C \text{ rated}}, U_{L \text{ rated}} > 0 \).

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_{TCSC} )</td>
<td>Nominal TCSC reactance</td>
<td>( \Omega )</td>
</tr>
<tr>
<td>( X_{bypass} )</td>
<td>TCSC reactance for bypass mode</td>
<td>( \Omega )</td>
</tr>
<tr>
<td>( V_{C \text{ rated}} )</td>
<td>Max. voltage for capacitive vernier region</td>
<td>p.u. of ( I_{L \text{ rated}}X_C )</td>
</tr>
<tr>
<td>( V_{L \text{ rated}} )</td>
<td>Max. voltage for inductive vernier region</td>
<td>p.u. of ( I_{L \text{ rated}}X_C )</td>
</tr>
<tr>
<td>( X_{max} )</td>
<td>Upper limit for TCSC reactance</td>
<td>p.u. of ( X_C )</td>
</tr>
<tr>
<td>( X_{min} )</td>
<td>Lower limit for TCSC reactance</td>
<td>p.u. of ( X_C )</td>
</tr>
<tr>
<td>( I_{TCSC}^{max} )</td>
<td>Upper limit for TCSC current</td>
<td>p.u. of ( I_{L \text{ rated}} )</td>
</tr>
</tbody>
</table>

Table 4.2: Typical TCSC data and parameters for steady-state modeling

\(^3\) [47, 48, 44]
4.2.6 Integration of the TCSC-embedded transmission line two-port model into the MINOS/ADIFOR environment

In the following the Controlled Series Compensation (CSC) stands for installation of fixed and thyristor-controlled capacitors. The total compensation reactance is

\[ X_{CSC} = X_{FixC} + X_{TCSC} \]  \hspace{1cm} (4.26)

where \( X_{FixC} \) the fixed compensation and \( X_{TCSC} \) the total TCSC reactance. The network data consists of bus and branch data. The user chooses the line (of given impedance \( Z_{Line} = R_{Line} + jX_{Line} \)) in series to which the TCSC will be installed. Table 4.3 contains a list of the input parameters. The units are as given in Table (4.2):

The nominal TCSC reactance can be calculated with the equation:

\[ X_{TCSC}^{C} = \frac{(X_{MaxC} - X_{FixC})/100}{X_{max}} \cdot X_{Line} \]  \hspace{1cm} (4.27)

Assuming multi-module TCSC the bounds and constraints of the total TCSC reactance are:
Table 4.3: Input data for advanced series compensation device (fixed + TCSC compensation scheme).

<table>
<thead>
<tr>
<th>Description</th>
<th>Input Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Capacitive Compensation at reduced current</td>
<td>$X_{MaxC} \ (%X_{Line})$</td>
</tr>
<tr>
<td>Fixed Capacitive Compensation</td>
<td>$X_{FixC} \ (%X_{Line})$</td>
</tr>
<tr>
<td>Upper limit for TCSC reactance (at reduced current)</td>
<td>$X_{o max}$</td>
</tr>
<tr>
<td>Lower limit for TCSC reactance (at reduced current)</td>
<td>$X_{o min}$</td>
</tr>
<tr>
<td>Max. voltage for capacitive vernier region</td>
<td>$U_{C \text{ rated}}$</td>
</tr>
<tr>
<td>Max. voltage for inductive vernier region</td>
<td>$U_{L \text{ rated}}$</td>
</tr>
<tr>
<td>Upper limit for TCSC current</td>
<td>$I_{max TCSC}$</td>
</tr>
<tr>
<td>Fixed $X_{TCSC}$, Fixed $I_{Line}$</td>
<td>Control mode</td>
</tr>
<tr>
<td>$X_{sch TCSC}$, $I_{sch TCSC}$</td>
<td>Control setting</td>
</tr>
</tbody>
</table>

\[
X_{o max} X_{C}^{TCSC} \leq X_{TCSC} \leq -X_{o min} X_{C}^{TCSC} \quad (4.28)
\]

\[
U_{C \text{ rated}} X_{C}^{TCSC} \leq I_{Line}/I_{L \text{ rated}} X_{TCSC} \leq -U_{L \text{ rated}} X_{C}^{TCSC} \quad (4.29)
\]

\[
0 \leq I_{Line}/I_{L \text{ rated}} \leq I_{max TCSC} \quad (4.30)
\]

If the control mode is constant TCSC current there is an additional constraint:

\[
I_{sch} \leq I_{TCSC} \leq I_{sch} \quad (4.31)
\]

The required steps to include the CSC-embedded line model to the existing OPF code are:

1. Declare the new variable $X_{tcsc}$ and append it to the independent variable list of ADIFOR in the script file ADIFOR\_CONTROL.adf.

2. Initialize the derivative object $g_{X_{tcsc}}$ in the seed matrix and specify the bounds $bu, bl$

   ```fortran
   do i = 1, num_tcsc_devs
       call dspsd(g_X_tcsc(i),ipos,1.d0,1)
       bu(ipos) = -X_min0(i) * Xc_tcsc(i)
       bl(ipos) = X_max0(i) * Xc_tcsc(i)
       ipos = ipos + 1
   enddo
   ```

3. Supply Fortran code for the functional constraints and store them in the vector of the constraints ($tcsc\_contr\_value$ stands for $I_{sch TCSC}^{sch}$):
c voltage constraint
---------------------
I_tcsc = dsqrt(I_2p_1_e(i)**2+I_2p_1_f(i)**2)
I_2p_tcsc_wrtline = I_tcsc/Ilinerat(ktcsc)
LNConstraintVector(lncon) = X_tcsc(ktcsc)

LNbu(lncon) = -(Xcap_fixed(ktcsc) + Vc_rated(ktcsc)*
& Xc_tcsc(ktcsc)/I_2p_tcsc_wrtline)
LNbl(lncon) = -(Xcap_fixed(ktcsc) - Vl_rated(ktcsc)*
& Xc_tcsc(ktcsc)/I_2p_tcsc_wrtline)
lncon = lncon + 1

c I_tcsc regulation at Port 1
-------------------------------
if (control_mode(ktcsc) .eq. 'Ifixed') then)
NLConstraintVector(lncon) = I_tcsc(ktcsc)
NLbu(lncon) = -tcsc_contr_value(ktcsc)
NLbl(lncon) = -tcsc_contr_value(ktcsc)
nlcon = nlcon + 1
endif

4. Append generic pieces of code for updating the values of the new variable

        do i = 1, num_tcsc_devs
           X_tcsc(i) = x(ipos)
           ipos = ipos + 1
        enddo

5. run ADIFOR to generate code for the new derivatives and recompile
4.3 The Unified Power Flow Controller (UPFC)

4.3.1 Introduction

The Unified Power Flow Controller (UPFC) is the most sophisticated and versatile of all FACTS devices at present [49, 50]. It represents an all solid-state implementation of all functions of power flow controllers. The basic functional block of the UPFC is the solid-state Synchronous Voltage Source (SVS). The SVS is implemented by a voltage-sourced multi-pulse inverter using GTO thyristors (Figure 4.16(a)). The reader is referred to [51, 8, 52] for a more detailed description of the implementation.

The SVS is analogous to an ideal synchronous machine (Figure 4.16(b)) which generates a balanced set of three sinusoidal voltages, at the fundamental frequency, with controllable magnitude and phase angle. This ideal machine

- internally generates or absorbs reactive power (without any ac capacitors or reactors)
- can exchange real power (if coupled to an energy source) independent of var generation
- has no mechanical inertia
- has very fast response

![Figure 4.16: Synchronous Voltage Source operated as a series capacitive compensator](image)

The next sections will describe the basic concepts and modeling issues.
4.3.2 Basic operating principles of the UPFC

The basic structure of the UPFC implementation is shown in Figure 4.17(a). It consists of two voltage-sourced converters, one connected in shunt with the line (Inverter 1) through a transformer, and one connected in series with the line (Inverter 2) through a second transformer. The two inverters are operated from a common dc link, provided by a dc storage capacitor. The series connected inverter injects a controlled voltage $U_T$ in series with the line. The phase angle of the phasor $U_T$ can be chosen independently of the line current between 0 and $2\pi$ and the magnitude is variable between zero and a maximum $U_T^{\text{max}}$.

This implies that the series inverter exchanges real and reactive power with the transmission line. The reactive power can be generated independently from the Inverter 2, while in this arrangement the real power has to be supplied from the network. That is the primary function of the shunt-connected Inverter 1, which is controlled in such a way as to provide at its dc terminals the real power needed by Inverter 2. A secondary function of Inverter 1 is to generate or absorb reactive power for regulation of the shunt transformer ac terminal voltage. Figure 4.17(b) displays the UPFC voltage phasor diagram. The operating area becomes a circle with radius $|U_T^{\text{max}}|$ and the center can be shifted by regulating the voltage $U_1$. This arrangement functions as an ac-ac power converter in which the real power can flow in either direction between the ac terminals and each inverter can generate (or absorb) reactive power at its own...
ac terminal.

The primary function of the UPFC is power flow control which is accomplished by series voltage injection.

### 4.3.3 The two-port model of UPFC

![Two-port model of UPFC](image)

Figure 4.18: The two-port model of the UPFC.

In the two-port UPFC model of Figure (4.18) the series voltage source \( U_T \) represents the controllable voltage inserted from the series branch, and the current source \( I_T \) the controllable current injected by the parallel branch. Of the four controllable quantities, namely, the voltage magnitude \( U_T \), the voltage phase angle \( \delta_T = \angle U_T \), the magnitude of the shunt current \( \phi_T = I_T \) and its phase angle \( \angle I_T \), only three can be regulated independently within a region defined by

\[
\mathcal{E} = \{ U_T, \delta_T, I_T, \phi_T; \\
U_T \leq U_T^{\text{max}}, 0 \leq \delta_T \leq 2\pi, \quad (4.32) \\
I_T \leq I_T^{\text{max}}, 0 \leq \phi_T \leq 2\pi, \quad (4.33) \\
P_{\text{exch}}^{\text{tot}} = 0 \}
\]

where \( P_{\text{exch}}^{\text{tot}} \) the net active power exchange of the UPFC device with the system.

The mathematical relations of the UPFC model are

\[
-\frac{U_1^{2p}}{L_1} - \frac{U_T}{L_T} + \frac{U_2^{2p}}{L_2} = 0 \quad (4.35) \\
\frac{I_1^{2p}}{L_1} + \frac{I_T}{L_T} + \frac{I_2^{2p}}{L_2} = 0 \quad (4.36) \\
\mathcal{R}\{U_1^{2p}I_T^{*}\} + \mathcal{R}\{U_TI_2^{2p}\} = 0 \quad (4.37) \\
\mathcal{I}\{U_1^{2p}I_T^{*}\} = 0 \quad (4.38)
\]

From (4.32-4.34) and (4.35-4.38) it follows directly that the UPFC is modeled as a series voltage source with limits on the injected voltage magnitude. The voltage phase angle is free to vary between 0 and \( 2\pi \) (4.32). The shunt connected inverter is modeled as a current source \( I_T \) with limited current magnitude (4.33). Neglecting losses of the UPFC, the net active power interchange of the UPFC with the power system is zero (4.37). If voltage control of the AC terminal is not required, the reactive power exchanged at the terminals of the shunt transformer is zero [53] (4.38).
4.3.4 The two-port model of the UPFC-embedded transmission line

In this section the modeling equations of the two-port model of a UPFC-embedded transmission line will be derived. The reader is referred to [53] for comparison with the UPFC injection model. It will be assumed that the UPFC is positioned at Port 1 (Figure 4.19). The UPFC two-port equations with the chain matrix representation

\[
\begin{bmatrix}
U_2^p \\
I_1^p
\end{bmatrix} = A_{UPFC} \begin{bmatrix}
U_M^p \\
I_M^p
\end{bmatrix} + u_{UPFC}
\]

(4.39)

with (see Figure (4.18))

\[
A_{UPFC} = \begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix}, \quad u_{UPFC} = \begin{bmatrix}
-U_T \\
-I_T
\end{bmatrix}
\]

(4.40)

The chain matrix representation of the transmission line two-port network is

\[
\begin{bmatrix}
U_1^p \\
I_1^p
\end{bmatrix} = A_{Line} \begin{bmatrix}
U_2^p \\
I_2^p
\end{bmatrix}
\]

(4.41)

The elements of \( A_{Line} \) are given in page 80.

The two networks are connected in cascade, therefore:

\[
\begin{bmatrix}
U_1^p \\
I_1^p
\end{bmatrix} = A_{UPFC} A_C \begin{bmatrix}
U_M^p \\
I_M^p
\end{bmatrix} + u_{UPFC} =
\]

\[
= (A_{UPFC} A_C A_{Line}) \begin{bmatrix}
U_2^p \\
I_2^p
\end{bmatrix} + u_{UPFC} =
\]

\[
= A_{Line} \begin{bmatrix}
U_2^p \\
I_2^p
\end{bmatrix} + u_{UPFC}
\]

(4.42)
with

\[ I_C = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]

Using Maple the equations (4.42) can be brought into the admittance form:

\[
\begin{bmatrix} I^{2p}_1 \\ I^{2p}_2 \end{bmatrix} = \begin{bmatrix} a_{11, \text{Line,UPFC}} & a_{12, \text{Line,UPFC}} \\ a_{21, \text{Line,UPFC}} & a_{22, \text{Line,UPFC}} \end{bmatrix} \cdot \begin{bmatrix} U^{2p}_1 \\ U^{2p}_2 \end{bmatrix} + \begin{bmatrix} U^T_1 \\ U^T_2 \end{bmatrix} \cdot \begin{bmatrix} u_{11, \text{Line,UPFC}} & u_{12, \text{Line,UPFC}} \\ u_{21, \text{Line,UPFC}} & u_{22, \text{Line,UPFC}} \end{bmatrix} \cdot \begin{bmatrix} I^T_1 \\ I^T_2 \end{bmatrix}
\]

(4.43)

with

\[
\begin{align*}
a_{11, \text{Line,UPFC}} &= \frac{-B_l^2 + IB_c G_l + G_l^2 - B_c B_l + 2IG_l B_l}{G_l + IB_l} \\
a_{12, \text{Line,UPFC}} &= \frac{B_l^2 - G_l^2 - 2IG_l B_l}{G_l + IB_l} \\
a_{21, \text{Line,UPFC}} &= G_l - IB_l \\
a_{22, \text{Line,UPFC}} &= G_l + IB_l + IB_c \\
u_{11, \text{Line,UPFC}} &= \frac{-B_l^2 + IB_c G_l + G_l^2 - B_c B_l + 2IG_l B_l}{G_l + IB_l} \\
u_{12, \text{Line,UPFC}} &= -1 \\
u_{21, \text{Line,UPFC}} &= -G_l - IB_l \\
u_{22, \text{Line,UPFC}} &= 0
\end{align*}
\]

The most important parameters for the UPFC are listed in Table 4.4.

<table>
<thead>
<tr>
<th>Input Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_T^{\text{max}}$</td>
<td>Maximum series injected voltage</td>
</tr>
<tr>
<td>$U_T^{\text{min}}$</td>
<td>Minimum series injected voltage</td>
</tr>
<tr>
<td>$I_T^{\text{max}}$</td>
<td>Maximum shunt current</td>
</tr>
<tr>
<td>$P_{\text{exch}}^{\text{max}}$</td>
<td>Maximum MW exchange</td>
</tr>
<tr>
<td>Control Mode</td>
<td>Voltage regulation, P- and Q- flow line regulation</td>
</tr>
<tr>
<td>Control Settings</td>
<td>$V_{\text{min}}, V_{\text{max}}, P_{\text{line}}, Q_{\text{line}}^{\text{sch}}$</td>
</tr>
</tbody>
</table>

Table 4.4: Input data for UPFC
4.3.5 Integration of the UPFC-embedded transmission line two-port model into the MINOS/ADIFOR environment

The required steps to include the UPFC-embedded line model to the existing OPF code are:

1. Declare the new variables $U_T, \delta_T, I_T, \phi_T$ and append them to the independent variable list of ADIFOR in the script file ADIFOR_CONTROL.adf.

2. Initialize the derivative objects $g_{U_T}, g_{\delta_T}, g_{I_T}, g_{\phi_T}$ in the seed matrix and specify the bounds $bu, bl$ (see below for initialization of $g_{U_T}, g_{\delta_T}$)

   ```fortran
   do i = 1, num_upfc_devs
     call dspsd(g_{U_T}(i),ipos,1.d0,1)
     bu(ipos) = U_{T_max}(i)
     bl(ipos) = U_{T_min}(i)
     ipos = ipos + 1
   enddo
   do i = 1, num_upfc_devs
     call dspsd(g_{\delta_T}(i),ipos,1.d0,1)
     bu(ipos) = infin
     bl(ipos) = -infin
     ipos = ipos + 1
   enddo
   ```

3. Supply Fortran code for the functional constraints. In the following code extract the constraint (4.37) is formulated and stored to the vector of nonlinear constraints (NLConstraintVector)

   ```fortran
   I_{serV_e} = I_{2p_1_e}(i) + I_{q_e}
   I_{serV_f} = I_{2p_1_f}(i) + I_{q_f}
   I_{serV} = dsqrt(I_{serV_e}**2 + I_{serV_f}**2)
   angI = datan2(I_{serV_f},I_{serV_e})
   Vq_magn = dsqrt(V_{lineside1_e}(kupfc)**2 + V_{lineside1_f}(kupfc)**2)
   & P_{exchSer}(kupfc) = U_{T}(kupfc) * I_{serV} * dcos(\delta_T(kupfc) - angI)
   & P_{exchShuV}(kupfc) = - Vq_magn * I_{q_magn} * dcos(\angVq-\angIq)
   & NLConstraintVector(nlcon) = P_{exchSer}(kupfc) - P_{exchShuV}(kupfc)
   NLbu(nlcon) = 0.0
   NLbl(nlcon) = 0.0
   nlcon = nlcon + 1
   ```
4. run ADIFOR to generate code for the new derivatives and recompile

4.4 Conclusions

In this chapter the operating principles and the two-port models of the TCSC and UPFC were described. The TCSC by means of thyristor switches (with traditional thyristors) inserts a rapidly controllable reactance, thereby providing impedance-type series compensation. The UPFC by means of voltage-sourced inverters (with GTO thyristors) can provide simultaneously or sequentially all types of compensation.

The modeling equations were given for both devices in the admittance matrix representation. They represent equality constraints for the optimization problem. The necessary steps for integrating the new models in the existing OPF software tool described in chapter 3 were discussed. Examining these steps it follows directly that the process of expanding the code to include new models is almost entirely modular.
Chapter 5

Case studies

The transfer capability of a network is governed by numerous factors, namely the power flow equations, the mode of operation, the network component ratings, and the various quality criteria that define desired ranges of operation for a number of state variables. Therefore, quantifying the network transfer capability with a simple number is difficult. The following sections contain case studies that evaluate the capability of series FACTS devices in maximizing the steady-state network transfer capability, which will be measured in terms of the MW power drawn from a bus or group of buses (area) in the network.

The series FACTS devices, the TCSC and the UPFC, are capable to vary the transmission characteristics in a flexible manner. By means of a two-node and a realistic network will be demonstrated how the FACTS devices can modify the “natural” flow of electric power in the network and thereby permit a more even loading of the network components.

5.1 Maximization of MW power transfer in a two node network

In this section, the two-node network studied in the introduction will be revisited. The two generic nodes A and B (see Figure 5.1) can represent buses or network areas. Their parameters are given in Table 5.1. In this configuration A is the generator node \((P_A < 0)\) and B the consumption node \((P_B > 0)\). The two generic branches AB-1 and AB-2 operate at 500 kV and have the parameters listed in Table 5.2.

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>(U_{\text{min}})</th>
<th>(U_{\text{max}})</th>
<th>(P_{\text{max}})</th>
<th>(Q_{\text{min,max}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Generation</td>
<td>0.90</td>
<td>1.10</td>
<td>42.0</td>
<td>unlimited</td>
</tr>
<tr>
<td>B</td>
<td>Consumption</td>
<td>0.90</td>
<td>1.10</td>
<td>unlimited</td>
<td>unlimited</td>
</tr>
</tbody>
</table>

Table 5.1: Per unit parameters for the nodes A and B in Figure 5.1 calculated with \(U_B = 500\) kV and \(S_B = 100\) MVA

Figure 5.1 illustrates the two node network and the numerical results of the base case.
Figure 5.1: Maximum power transfer from Bus/Area A to Bus/Area B. The two parallel paths have unequal ratings (see Table 5.2).

Table 5.2: Per unit parameters for transmission lines AB-1 and AB-2 in Figure 5.1 calculated with $U_B = 500$ kV and $S_B = 100$ MVA

The base-case study finds the maximum transferable MW power to consumption bus B, with no FACTS installed. The MW power transferred to bus B is equivalent to the MW power drawn from bus B. Therefore, the base-case objective function is

$$\text{Max. } P_B \iff \text{Min. } -P_B$$

subject to

\footnote{MINOS by default minimizes}
Thus, the base case optimization problem has 8 variables \((U_A, U_B, \delta_A, \delta_B, P_A, P_B, Q_A, Q_B)\), 8 functional constraints (Kirchhoff equations and MVA ratings) and 3 bound constraints (voltage magnitude at both nodes and generating power). MINOS reaches the solution after 13 major iterations and 68 minor iterations and the objective value is \(-29.2824\) p.u., i.e. the maximum load at Node B is 2928.24 MW. At the base-case optimum, as illustrated in Figure 5.1, the lower-rated line operates at the maximum allowable loading, while the high-rated path operates at only about 60% of its rating.

5.1.1 Maximization of MW power transfer with TCSC in the two-node network

![Figure 5.2: The two node system with one TCSC on AB-1](image)

Simulation studies in this section will demonstrate the effect on the network transfer capability of a TCSC device connected in series with the high-rated transmission line of the two-node network (see Figure 5.2). The TCSC embedded transmission line AB-1 is modeled as in section 4.2.5 (TCSC located at the sending end) and has the following parameters (consult Table 4.2 on page 81 for the units).
If the TCSC is dimensioned such that the maximum series compensation at low current is 90% and there is no fixed capacitive compensation ($X_{FixC} = 0.0\%$), the max. transferable power is 3889.00 MW (see Figure 5.3). This optimization problem has one additional variable ($X_{TCSC}$) compared to the base-case problem, one additional functional contraint (voltage limitation in the capacitive operating region) and bound constraints on the new variable (firing angle limitations)\(^2\). MINOS reaches the solution after 30 major and 115 minor iterations. The TCSC capacitor is 0.0077 p.u. and calculated according to (4.27) on page 82. At the optimum the TCSC reactance is 0.0101 p.u. and the TCSC power is 595.81 MVAR.

If the fixed compensation is $X_{FixC} = 30.0\%$ of the line reactance, then the max. load at Bus B is 3972.30 MW (see Figure 5.4) and MINOS converges after 13 major and 108 minor iterations. The TCSC capacitor is 0.0051 p.u. and at the optimum the TCSC reactance is 0.0029 p.u.. The TCSC power and fixed capacitor power at the optimum is 183.68 MVAR and 486.43 MVAR respectively. The series compensation

\(^2\) see section 4.2.5
share between fixed capacitors and TCSC can be determined using results of a number of studies involving mainly subsynchronous resonance [4, 54] and costs considerations.

Table 5.2: Maximum power transfer from Bus/Area A to Bus/Area B with a TCSC

<table>
<thead>
<tr>
<th>TCSC</th>
<th>AB-1</th>
<th>AB-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{TSC}$ = -0.0029</td>
<td>2769.80 MW</td>
<td>1384.89 MW</td>
</tr>
<tr>
<td></td>
<td>33.57 MVAR</td>
<td>17.80 MVAR</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Figure 5.4: Maximum power transfer from Bus/Area A to Bus/Area B with a TCSC and 30% fixed compensation installed on AB-1

Assuming that the $X_{TSC}$ is kept fixed to a value between 0.0 and 0.024 p.u. (absolute value) at increments of 0.001 p.u., and for each point the above optimization problem is solved with the bus voltages and generation as the only variables. For simplicity, there is no fixed-capacitor compensation. Figure 5.5 (left) shows for each optimal point (circle) the MW transfer as well the TCSC MVAR power. It confirms that the MW transfer is maximized with $|X_{TSC}| = 0.0101$ p.u.. Figure 5.5 (right) displays the reactive and active flow at the sending end of the line AB-1. $P_S$ saturates in the $Q_S - P_S$ characteristic due to the MVA constraint.

In order to maximize the power transfer, $P_B$, and in addition minimize the MVAR power of the TCSC, $MVAR_{TSC}$, and the Ohm losses, $P_{loss}$, the following objective function is formulated

$$F_{Obj} = -W_1 \cdot P_B + W_2 \cdot MVAR_{TSC} + W_3 \cdot P_{loss}$$

where $W_1$, $W_2$ and $W_3$ the weighting coefficients and $P_{loss} = -P_A - P_B$. Assigning different values to $W_1$, $W_2$ and $W_3$ a number of optimization cases are solved and the results are summarized in Table 5.3.

In Table 5.3 the $P_{loss}$ is expressed in per cent of the generation $P_A$. Case 1 is shown in Figure 5.3. From the results of Table 5.3 it follows that substantial reduction of the TCSC MVAR power or the losses is coupled with substantial reduction of the transferable power.

In Table 5.3 the $P_{loss}$ is expressed in per cent of the generation $P_A$. Case 1 is shown in Figure 5.3. From the results of Table 5.3 it follows that substantial reduction of the TCSC MVAR power or the losses is coupled with substantial reduction of the transferable power.

3 It is assumed that the cost of the TCSC is proportional to its MVAR power
Chapter 5: Case studies

Figure 5.5: Left: Per unit values for the objective function and the TCSC MVAR vs. $X_{TCSC}$. Right: Reactive flow vs. active flow at the sending end of AB-1.

<table>
<thead>
<tr>
<th>Case</th>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$P_B$ (MW)</th>
<th>MVAR $TCSC$</th>
<th>$P_{loss}^{tot}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3889.00</td>
<td>595.81</td>
<td>4.29</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>3889.00</td>
<td>595.81</td>
<td>4.29</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>3889.00</td>
<td>595.81</td>
<td>4.29</td>
</tr>
<tr>
<td>4</td>
<td>10.0</td>
<td>1.0</td>
<td>1.0</td>
<td>3889.00</td>
<td>595.81</td>
<td>4.29</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>10.0</td>
<td>1.0</td>
<td>2927.98</td>
<td>0.00</td>
<td>3.48</td>
</tr>
<tr>
<td>6</td>
<td>1.0</td>
<td>0.0</td>
<td>10.0</td>
<td>3889.00</td>
<td>595.81</td>
<td>4.29</td>
</tr>
</tbody>
</table>

Table 5.3: MW power transfer, TCSC MVAR power and MW system losses for different weight values in $F_{Obj} = -W_1 \cdot P_B + W_2 \cdot MVAR_{TCSC} + W_3 \cdot P_{loss}^{tot}$

5.1.2 Maximization of MW power transfer with UPFC in the two-node network

Simulation studies in this section will show the effect of the UPFC on the MW power transfer of the two node network in Figure 5.6. As the TCSC in the previous section, the UPFC is installed at the sending end of AB-1. The UPFC input data are listed in Table 5.4.

The total MVA power of the UPFC will be defined in the following as the sum of the MVA power of the series and shunt inverter.
### Table 5.4: Input data (per unit) for the UPFC on AB-1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_T^{max}$</td>
<td>0.5</td>
</tr>
<tr>
<td>$U_T^{min}$</td>
<td>0.0</td>
</tr>
<tr>
<td>$I_T^{max}$</td>
<td>10.0</td>
</tr>
<tr>
<td>$\delta_T^{min}$</td>
<td>0$^\circ$</td>
</tr>
<tr>
<td>$\delta_T^{max}$</td>
<td>360$^\circ$</td>
</tr>
<tr>
<td>$U_S$ control</td>
<td>True/False</td>
</tr>
<tr>
<td>$U_{Smin}, U_{Smax}$</td>
<td>0.9, 1.1</td>
</tr>
</tbody>
</table>

First, a series of optimization problems are solved, with fixed series inserted voltage magnitude $U_T = 0.10, 0.20, 0.25, 0.30$ p.u. and fixed series inserted voltage angle $\delta_T = 0^\circ, 10^\circ, \ldots 360^\circ$. The objective is maximization of the load drawn from Bus B. The constraints are the base case constraints as well as the UPFC constraints (see section 4.3.4). The variations of line flow $P_S + iQ_S$ are illustrated in Figure 5.7. Comparison of the characteristics of line flows in Figures 5.5 (left) and 5.7 proves the superior performance of the UPFC over the TCSC. By adjusting the inserted series voltage magnitude and angle the operator can realize any of the points $(P_S, Q_S)$ on the curves shown in Figure 5.7. Therefore, the UPFC provides simultaneous and independent control of the P and Q line flows, while the TCSC can regulate only one
of the two.

Figure 5.7: Reactive flow vs. active flow (per unit) after the UPFC for different values of series injected voltage

Next, the same optimization problem is solved with the injected voltage free to vary between 0 and $U_T^{\text{max}} = 0.5$ p.u. and the voltage angle between $\delta_T^{\text{min}} = 0^\circ$ and $\delta_T^{\text{max}} = 180^\circ$. The solution is displayed in Figure 5.8. With an injected voltage magnitude of 0.274 p.u. and voltage angle 97.9° the power transfer can be increased to 3917.36 MW and at that power transfer the MVA power of the series element is $MVA_{\text{series}} = 691.52$ and of the shunt element $MVA_{\text{shunt}} = 60.97$.

In order to see how the objective function behaves with variations in the series injected voltage the following optimization problems are solved. First, the voltage magnitude is kept fixed to $U_T = 0.274$ p.u. (the series voltage level at the optimum) and the angle varies between 0° and 360° at increments of 10°. The variation in the objective function (stars) as well as in the $MVA_{\text{UPFC}}$ values (squares) is illustrated in Figure 5.9 (left). The same process is repeated for voltage magnitude of $U_T = 0.200, 0.250, 0.274, 0.300$ p.u. and the optimal points (circles) of Figure 5.9 (right) are obtained. Observing the curves in Figure 5.9 confirms that the maximum power transfer is established with $U_T = 0.274$ and $\delta_T \approx 98^\circ$ as shown in Figure 5.8.

In order to maximize $P_B$ and in addition minimize the MVA power of the UPFC the objective function (5.2) is formulated

$$F_{\text{Obj}} = -W_1 \cdot P_B + W_2 \cdot MVA_{\text{UPFC}} \tag{5.2}$$

where $W_1$ and $W_2$ are the weighting coefficients.
Figure 5.8: Maximum power transfer with UPFC on AB-1, no voltage regulation and no UPFC MVA minimization

Optimizing (5.2) with $W_1 = 10.0, W_2 = 1.0$ achieves a MW transfer level of 3860.73 MW with $MVA_{series} = 510.03$ and $MVA_{shunt} = 99.14$. The line flows and UPFC parameters are shown in Figure 5.10.

Performing a series of similar optimization problems with fixed UPFC voltage magnitude $U_T = 0.219$ provides the curves of Figure 5.11 (left). Figure 5.11 (right) shows the solution points for different UPFC voltage levels and for voltage angles.

Voltage regulation and losses

The studies so far assume that voltage regulation is not desired and the constraint

$$Q_{shunt} = 0$$  \hspace{1cm} (5.3)

allows no MVAR exchange between the shunt inverter and the system. Alternatively, the line side voltage $U_S$ of the UPFC can be regulated so that $U_S^{min} \leq U_S \leq U_S^{max}$ and the constraint 5.3 is omitted.

Using the objective function

$$F_{Obj} = -W_1 \cdot P_B + W_2 \cdot MVA_{UPFC} + W_3 \cdot P_{loss}^{tot}$$  \hspace{1cm} (5.4)

will
102 Chapter 5: Case studies

Figure 5.9: Left: Per unit values for $P_{2}^{\text{max}}$, $MVA_{\text{UPFC}}$ and $F_{\text{Obj}}$ for $U_T = 0.274$ p.u. versus $\delta_T$ (no $MVA_{\text{UPFC}}$ minimization). Right: Per unit objective values for different UPFC voltage levels.

<table>
<thead>
<tr>
<th>Series Inverter</th>
<th>AB-1</th>
<th>AB-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$|U_T</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td>$\delta_T$</td>
<td>71.298</td>
<td>834.69 MVAR</td>
</tr>
<tr>
<td>$P_{\text{exch}}$</td>
<td>99.137</td>
<td>100.00%</td>
</tr>
<tr>
<td>$Q_{\text{exch}}$</td>
<td>500.298</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shunt Inverter</th>
<th>AB-2</th>
<th>AB-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$|I_T</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td>$\angle I_T$</td>
<td>-180.000</td>
<td>17.80 MVAR</td>
</tr>
<tr>
<td>$P_{\text{exch}}$</td>
<td>99.137</td>
<td>100.00%</td>
</tr>
<tr>
<td>$Q_{\text{exch}}$</td>
<td>0.000</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.10: Maximum power transfer with UPFC on AB-1 (no voltage regulation) and UPFC MVA power minimization ($W_1 = 10.0, W_2 = 1.0$)

1. maximize the load $P_B$, i.e. maximize the power transfer from A to B.
2. minimize the UPFC MVA power
3. minimize the MW losses $P_{\text{loss}} = -P_A - P_B$
In Table 5.5 the $P_{loss}^{tot}$ is expressed in per cent of the generation $P_A$. Cases 1 and 4
are the optimization studies illustrated in Figures 5.8 and 5.10. If there is no voltage control, the line side voltage will exceed the upper limit, 1.1 p.u. for certain loadings. Voltage control will, likely, decrease slightly the maximum transferable power and will decrease, in certain cases considerably, the MW flow through the UPFC. Because of the independent control of P- and Q-flows the losses can be controlled more effectively than with the TCSC (see Table 5.3).

The simulation studies in this section demonstrate the unique ability of the UPFC to control three independent variables and they provide a first approximation to the requirements in equipment rating.

5.2 Application of TCSC in a 37-node real-life network

The situation described in this section is derived from a real network. The study network consists of 37 one-ports (will be referred to as buses or nodes) and 105 two-ports (will be referred to as branches or lines). The network modeling parameters are derived from reduction of the real network and converted into the IEEE common format [55]. The network contains 59 equivalent (artificial) branches. Purpose of the studies in this chapter is to maximize the transferable MW power from a group of generating plants or exporting areas to a consumption bus or importing area. Figure (5.12) illustrates the most important buses and branches of the study network. Branch numbers have been added, whereas branches that do not lie on the interesting paths are not included in the diagram for better readability. Further, Figure 5.12 contains only real (not equivalent) branches. The generators are located at the buses AAA81, AAA87, AAA82, AAA86, PPP81. Bus ZZZ41 represents a connection to an exporting area, that injects MW power in the network. It will be assumed in the following studies that there are no limits imposed on the injected power $P_{ZZZ41}$. Bus FFF21 represents the consumption node or area whose power will be maximized. In power flow calculations bus FFF21 is modeled as a PQ-bus or load bus [56, 57]. That implies that the bus voltage is a free complex variable while the variables $P_{FFF21}$ and $Q_{FFF21}$ are forced to the predefined load MW and MVAR values (parameters).

In optimization studies, however, any network parameter can be a variable. In this case the $P_{FFF21}$ will be considered as a free variable. The point of reference for the following studies will be the base case. That is the network state with no installed compensating devices at the solution of the following optimization problem:

Min. $-P_{FFF21}$

subject to

- **voltage magnitude limitations**: The voltage magnitude of all buses is not allowed to exceed the limits of 0.9 p.u. and 1.1 p.u.

- **generating power limitations**: The real and reactive power of generators is
Figure 5.12: Test system illustrating the locations of the generating plants and the main consumption node

limited to the values of the following table. The imported MW power from area ZZZ41 is unlimited as mentioned.

<table>
<thead>
<tr>
<th></th>
<th>$P_{G}^{\text{min}}$ (MW)</th>
<th>$P_{G}^{\text{max}}$ (MW)</th>
<th>$Q_{G}^{\text{min}}$ (MVAR)</th>
<th>$Q_{G}^{\text{max}}$ (MVAR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA81</td>
<td>175.0</td>
<td>190.0</td>
<td>-30.0</td>
<td>120.0</td>
</tr>
<tr>
<td>AAA82</td>
<td>175.0</td>
<td>190.0</td>
<td>-30.0</td>
<td>120.0</td>
</tr>
<tr>
<td>AAA86</td>
<td>175.0</td>
<td>190.0</td>
<td>-30.0</td>
<td>120.0</td>
</tr>
<tr>
<td>AAA87</td>
<td>175.0</td>
<td>190.0</td>
<td>-30.0</td>
<td>120.0</td>
</tr>
<tr>
<td>PPP81</td>
<td>900.0</td>
<td>1030.0</td>
<td>-265.0</td>
<td>405.0</td>
</tr>
</tbody>
</table>

- **branch MVA constraints**: The line currents should not exceed 80% of their thermal limits. The same applies for transformers. The equivalent branches derived from network reductions have no MVA constraints, since there is no available data.

In the base case the transferable MW to bus FFF21 is limited by the maximum allowable loadings (branch MVA constraints) of certain paths (see Figure 5.14). The critical (i.e. highly loaded) paths are the direct 230 kV paths (see Figure 5.13): #15,#23,#25,(#27/#28) and #18,#26,(#27/#28). At the same time the paths #6,#7,#104 and the paths ending at #29,#30 are parallel to the direct overloaded path, and, judging from their base case line loadings, it seems that there are large transfer capacity reserves that can be exploited.
Figure 5.13: Base case loadings of the main paths (dashed lines) connecting the generators or exporting areas with the FFF21: thicker dashed lines denote heavily loaded transmission elements.

In order to find the “best” locations for placing the TCSC device(s) a sensitivity analysis is conducted. A TCSC is installed on every branch that represents a real (not derived from network reduction) transmission line. This set of 37 transmission lines will be represented as $\mathcal{L}_F$. The TCSC is assumed to be connected as in Figure 4.14 and the TCSC-embedded transmission line is modeled as described in Section 4.2.5, however with fixed capacitive part equal to zero ($X_{TCSC}^i = 0.0$). The impedance of the installed TCSCs is fixed to zero by activating the following constraints

$$X_{TCSC}^i = 0, \quad i \in \mathcal{L}_F$$

(5.5)

This optimization problem has the following dimensions:
Figure 5.14: Base case - Maximum power transfer to FFF21. The maximum allowable loading of transmission elements is 80%. 

\[ P_{lp} = -900.000 \text{ } UL \]

\[ P_{lp} = -175.000 \text{ } UL \]

\[ P_{lp} = -175.000 \text{ } UL \]

\[ P_{lp} = -175.000 \text{ } UL \]

\[ P_{lp} = 837.339 \]
The rows (functional constraints) consist of the Kirchhoff equality constraints, the TCSC voltage constraints, the MVA branch constraints and the taps-related constraints. The columns (structural variables) consist of the node voltages and real and reactive power injections, the regulating transformers’ taps and the TCSC reactances. In this case the TCSC voltage constraints are “dummy”, since the $X_{TCSC}(i)$ are forced to zero (see (5.5)). MINOS output vector $\mathbf{rc}$ contains the Lagrange multipliers, $\mu$ for the active constraints. Each of the constraints (5.5) has an associating $\mu$-value

$$
\mu(i) = \frac{\partial F_{obj}}{\partial X_{TCSC}^i} = -\frac{\partial P_{FFF21}}{\partial X_{TCSC}^i} \quad (5.6)
$$

Keeping in mind that in the code the inductive reactance is positive and the capacitive reactance is negative, the following conclusions can be derived

- For branches with $\mu(i) > 0$ an incremental capacitive $\Delta X_{TCSC}^i$ will increase the optimum load of FFF21
- For branches with $\mu(i) < 0$ an incremental inductive $\Delta X_{TCSC}^i$ will increase the optimum load of FFF21
- The optimal solution is insensitive to an incremental change in the numerical value of $X_{TCSC}^i$ on branches with $-\epsilon \leq |\mu(i)| \leq \epsilon$

The $\mu(i)$ represents the change in the optimal objective per unit-change in the $X_{TCSC}^i$. However, the $X_{TCSC}^i$ depends on the characteristics of the associating line $i$ (e.g. the line reactance $X_i^l$). In order to integrate this information in the Lagrange multipliers the following vector is defined

$$
\overline{\mu}(i) = \mu(i) \cdot X_i^l \quad (5.7)
$$

Figure 5.15 displays the $\overline{\mu}$ values for the equality constraints (5.5). The x-axis is the branch number as displayed in Figure 5.12.

As can be seen from Figure 5.15, placement of a TCSC at branch #16 and/or #7 will have the largest effect if the TCSC is operating in the capacitive region. If the TCSC is working in the inductive region then the TCSC(s) at branches #15 and #20,#23,#25 have the largest effect in the objective function. All these branches lie on two parallel paths that connect the generators with FFF21. Branches #15,#23,#25 lie on the direct, “critical” path while #7 and #16 are the underloaded 380 kV lines connecting the importing area ZZZ41 with FFF21.
5.2.1 Case A: TCSC on branch #16 (AAA22-BBB22)

In Case A, a single TCSC device is placed on branch #16. The following table lists the parameters of the TCSC that need to be specified by the user. The maximum possible capacitive compensation (at low current) of the TCSC is assumed to be 90%. Further, as discussed before, the conventional (fixed) capacitive compensation is assumed to be zero during the study. The actual implementation of the optimal TCSC yet to be found will always allow to partially use conventional capacitive compensation later.

\[ X_{TCSC}^i = 0, \quad i \in \mathcal{L}_F \]

A positive \( X_{TCSC}^i \) on #15, #23, #25 would create an equivalent longer path and therefore a portion of the power would be transferred to the other available paths. Alternatively a negative \( X_{TCSC}^i \) on the branches #16 and #7 would virtually shorten the respective unloaded paths and, therefore, would increase their power loadings.

The following studies will display the optimization results with placement of TCSC on the above mentioned branches.

\[ \mu(i) \]
In the optimization problem presented in this subsection, the reactance $X_{\text{TCSC}}^{16}$ is a fixed variable at first, whereas all other control or state variables are optimized (e.g. generation power, bus voltage, line currents). Although it seems natural to let the optimizer also find the optimal value for $X_{\text{TCSC}}^{16}$ in the same optimization run, this is postponed in the beginning in order to facilitate a better understanding of the problem and the influence of $X_{\text{TCSC}}^{16}$ on the achievable optimum.

The MVAR power of the TCSC is

$$MVAR_{\text{TCSC}} = (I_{1}^{16})^2 \cdot X_{\text{TCSC}}^{16}$$

where $I_{1}^{16}$ the current of the TCSC at port 1 (all variables in per unit values). To maximize the power transfer $P_{\text{FFF21}}$ and at the same time minimize the TCSC MVAR power and the system losses, the objective function

$$F_{\text{obj}} = -W_1 \cdot P_{\text{FFF21}} + W_2 \cdot MVAR_{\text{TCSC}} + W_3 \cdot P_{\text{tot}}^{\text{loss}}$$

is formulated. The MW system losses are calculated as the sum of the MW losses of all branches, i.e. the largest part of the objective function (5.9) is nonlinear.

The result will obviously depend on the choice of the weighting coefficients $W_1, W_2$ and $W_3$. Six combinations will be considered in the following:

<table>
<thead>
<tr>
<th>Case</th>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>A2</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>A3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>A4</td>
<td>10.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>A5</td>
<td>1.0</td>
<td>10.0</td>
<td>1.0</td>
</tr>
<tr>
<td>A6</td>
<td>10.0</td>
<td>10.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Here, Case A1 corresponds to the single objective optimization of the power transfer $P_{\text{FFF21}}$. 

---

### Input Data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{\text{MaxC}}$</td>
<td>90</td>
</tr>
<tr>
<td>$X_{\text{FixC}}$</td>
<td>0</td>
</tr>
<tr>
<td>$X_{\text{max}}$</td>
<td>3.0</td>
</tr>
<tr>
<td>$X_{\text{min}}$</td>
<td>2.0</td>
</tr>
<tr>
<td>$U_{\text{C rated}}$</td>
<td>1.15</td>
</tr>
<tr>
<td>$U_{\text{L rated}}$</td>
<td>1.00</td>
</tr>
<tr>
<td>$I_{\text{max TCSC}}$</td>
<td>4.50</td>
</tr>
</tbody>
</table>
For each of the cases A1 to A6, the optimization problem is solved for a number of different values of $X_{TCSC}^{16}$. The results are illustrated in Figure 5.16 for selected cases. Cases A4 and A6 were not included in the Figure, since the high value of the objective would change the y-axis scale, such that the other cases would be very difficult to distinguish.

Figure 5.16: The $F_{Obj}$ as a function of $X_{TCSC}^{16}$ (both in absolute values) for four different sets of weighting coefficients.

In a second step, also the reactance $X_{TCSC}^{16}$ is optimized. For each case A1 to A6, the value of $F_{Obj}$ is given in the following table:

<table>
<thead>
<tr>
<th>Case</th>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$P_{FFF21}$ (MW)</th>
<th>$X_{TCSC}$ (p.u.)</th>
<th>$MVAR$ (TCSC)</th>
<th>$P_{Losses}$ (MW)</th>
<th>$F_{Obj}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1023.00</td>
<td>-0.0212</td>
<td>6.99</td>
<td>48.70</td>
<td>-1023.00</td>
</tr>
<tr>
<td>A2</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1022.23</td>
<td>-0.0202</td>
<td>6.21</td>
<td>48.59</td>
<td>-1016.02</td>
</tr>
<tr>
<td>A3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1020.41</td>
<td>-0.0176</td>
<td>4.47</td>
<td>48.28</td>
<td>-967.66</td>
</tr>
<tr>
<td>A4</td>
<td>10.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1023.00</td>
<td>-0.0212</td>
<td>6.99</td>
<td>48.70</td>
<td>-10174.31</td>
</tr>
<tr>
<td>A5</td>
<td>1.0</td>
<td>10.0</td>
<td>1.0</td>
<td>1015.69</td>
<td>-0.0112</td>
<td>1.92</td>
<td>47.90</td>
<td>-948.59</td>
</tr>
<tr>
<td>A6</td>
<td>10.0</td>
<td>10.0</td>
<td>1.0</td>
<td>1021.09</td>
<td>-0.0186</td>
<td>5.07</td>
<td>48.42</td>
<td>-10111.78</td>
</tr>
</tbody>
</table>

These maximum values of the objective function correspond to the maxima of the curves in Figure 5.16. The influence of the choice of the weighting coefficients is
obvious: The higher the weighting coefficient of the MVAR power of the TCSC, the lower the TCSC MVAR power and the load $P_{FFF21}$ at the optimal solution. In Case A5, the MW transfer to FFF21 is increased by $1015.69 - 837.34 = 178.35$ MW compared to the base case with only 1.92 MVAR TCSC power.

Quite interesting is the comparison of cases A3 and A5 (see Figure 5.16). Both cases optimize the power transfer to FFF21, the TCSC MVAR power and the system losses, however, case A5 uses a weighting factor $W_2$ for the MVAR power of the TCSC ten times larger than that of case A3. This can be seen in the curve corresponding to case A3 in Figure 5.16. For $|X_{TCSC}^{16}| > 0.012$ the curve becomes relatively flat, which means that there is no significant improvement in the objective by increasing the $X_{TCSC}^{16}$ in absolute value. Looking at the table with the optimum values for all cases, we see that the optimum of case A3 lies at $|F_{Obj}| = 967.66$ and the TCSC’s reactance and power is $X_{TCSC}^{16} = -0.0176$ and 4.47, respectively. By increasing the weighting factor of the MVAR to 10.0 (case A5) the curve makes a bend around $X_{TCSC}^{16} = -0.0111$ discouraging the increase of $X_{TCSC}^{16}$ and the MVAR power. Now the optimum lies at $X_{TCSC}^{16} = -0.0112$ and the MVAR power at the optimum is 1.92.

Figure 5.18 on page 119 illustrates the loadings of the branches for Case A2. The loading of branch #16 has increased from 24.36\% (base case) to 42.86\%, while the loading of branch #15, the most heavily loaded branch on the direct path, has decreased from 80.00\% to 75.57\%. These changes in the loading pattern were anticipated. Load has been removed from the heavily loaded path and transferred to the lightly loaded, available parallel paths.

### 5.2.2 Case B: TCSC on branch #7 (WWW41-FFF41)

This section is structured exactly as the last, only this time the TCSC is placed on branch #7. The parameters of the TCSC are listed in the following table:\n
<table>
<thead>
<tr>
<th>Input Data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{MaxC}$ (%$X_{Line}$)</td>
<td>90.0</td>
</tr>
<tr>
<td>$X_{FixC}$ (%$X_{Line}$)</td>
<td>0.0</td>
</tr>
<tr>
<td>$X_{o}$</td>
<td>3.0</td>
</tr>
<tr>
<td>$X_{min}$</td>
<td>2.0</td>
</tr>
<tr>
<td>$U_{C\text{ rated}}$</td>
<td>1.15</td>
</tr>
<tr>
<td>$U_{L\text{ rated}}$</td>
<td>1.00</td>
</tr>
<tr>
<td>$I_{TCSC\text{ rated}}$ (p.u.)</td>
<td>14.00</td>
</tr>
</tbody>
</table>

\[ ^{6} \] The figures with the simulation results are provided at the end of the chapter to simplify comparison of the line flows.

\[ ^{7} \] consult Table 4.2 on page 81 for the units
The MVAR power of the TCSC is

\[ MVAR_{TCSC} = (I_1^7)^2 \cdot X_{TCSC}^7 \]  

(5.10)

where \( I_1^7 \) the current of the TCSC at port 1 (all variables in per unit values). To maximize the power transfer \( P_{F,FF21} \) and at the same time minimize the TCSC MVAR power and the system MW losses, the objective function (5.9) is formulated again.

The same six cases of choosing the weighting coefficients \( W_1, W_2 \) and \( W_3 \) are used. Again, Case B1 corresponds to the single objective optimization of the power transfer \( P_{F,FF21} \).

Solving the optimization problem for each of the cases B1 to B6 for a number of different values of \( X_{TCSC}^7 \) yields the results as illustrated in Figure 5.17. For the same reason, as stated in the previous section, only cases B1, B2, B3 and B5 are displayed.

\[ \begin{array}{c|c|c|c|c}
|X_{TCSC}^7| (p.u.) & B1 & B2 & B3 & B5 \\
\hline
0 & 1000 & 900 & 800 & 700 \\
1 & 900 & 800 & 700 & 600 \\
2 & 800 & 700 & 600 & 500 \\
3 & 700 & 600 & 500 & 400 \\
4 & 600 & 500 & 400 & 300 \\
5 & 500 & 400 & 300 & 200 \\
\end{array} \]

Figure 5.17: The \( F_{Obj} \) as a function of \( X_{TCSC}^7 \) (both in absolute values) for four different sets of weighting coefficients.

Solving the full optimization problem including finding the optimal value of the reactance \( X_{TCSC}^7 \) yields the optima for the cases B1 to B6 given in the following table:
Again, the found optimal values of the objective functions correspond to the maxima of the graphs corresponding to the four cases illustrated in Figure 5.17.

Cases B1 through B6 show that installation of a TCSC on branch #7 can increase the MW transfer by 927.83 – 837.34 = 90.50 MW with 18.80 MVAR power. Assigning different weighting factors to the individual objectives has no effect on the objective value, as it can be seen from the curves in Figure 5.17. Case B5 that assigns a large weight on the MVAR reduction forces the TCSC reactance to zero.

5.2.3 Case C: TCSC on branch #16 (AAA22-BBB22) and on branch #7 (WWW41-FFF41)

In this case two TCSC’s are placed on branch #16 and #7 each. The total TCSC MVAR power is

\[ \text{MVAR}_{\text{TCSC}} = \left(I_{16}^2\right)^2 \cdot X_{\text{TCSC}}^7 + \left(I_{16}^2\right)^2 \cdot X_{\text{TCSC}}^{16}. \]  

Again the objective function (5.9) will be used with the six cases.

The optimization with \( X_{\text{TCSC}}^7 \) and \( X_{\text{TCSC}}^{16} \) as free variables yields the optimal solution for each case C1 to C6 given in the following table:

<table>
<thead>
<tr>
<th>Case</th>
<th>( W_1 )</th>
<th>( W_2 )</th>
<th>( W_3 )</th>
<th>( P_{\text{FFF21}} ) (MW)</th>
<th>( X_{\text{TCSC}}^7 ) MVAR</th>
<th>( X_{\text{TCSC}}^{16} ) MVAR</th>
<th>( P_{\text{loss}} ) (MW)</th>
<th>( F_{\text{Obj}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1042.40 -0.0009</td>
<td>5.27 -0.0212</td>
<td>6.97</td>
<td>48.92</td>
<td>-1042.40</td>
</tr>
<tr>
<td>C2</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1039.17 -0.0010</td>
<td>5.67 -0.0138</td>
<td>2.73</td>
<td>48.25</td>
<td>-1030.77</td>
</tr>
<tr>
<td>C3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1038.77 -0.0010</td>
<td>6.00 -0.0117</td>
<td>2.06</td>
<td>47.96</td>
<td>-982.74</td>
</tr>
<tr>
<td>C4</td>
<td>10.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1042.39 -0.0009</td>
<td>5.26 -0.0212</td>
<td>6.97</td>
<td>48.92</td>
<td>-10362.79</td>
</tr>
<tr>
<td>C5</td>
<td>1.0</td>
<td>10.0</td>
<td>1.0</td>
<td>1015.78 0.0000</td>
<td>0.00 -0.0112</td>
<td>1.93</td>
<td>47.91</td>
<td>-948.61</td>
</tr>
<tr>
<td>C6</td>
<td>10.0</td>
<td>10.0</td>
<td>1.0</td>
<td>1039.06 -0.0010</td>
<td>5.76 -0.0132</td>
<td>2.54</td>
<td>48.17</td>
<td>-10259.46</td>
</tr>
</tbody>
</table>

According to the values on the above table, installation of two TCSCs on branches #7 and #16 can increase the MW transfer up to 1042.40 MW with 5.27 + 6.97 = 12.24
MVAR. Minimization of the TCSC MVAR yields the optimal point C2 with 1039.17 MW and 5.67 + 2.73 = 8.40 MVAR.

In all above cases the fixed capacitive compensation can be increased to reduce the required MVAR TCSC. As mentioned before, the series compensation share between fixed and thyristor-controlled series compensation can be determined by assigning weighting factors in the objective function, that reflect e.g. cost per installed MVAR of TCSC or conventional capacitor, and profit per MW transfer.

5.3 Application of UPFC in a 37-node real-life network

In the following UPFC devices will be placed at the same branches as the TCSC devices and optimization studies with the same objective functions will evaluate their effectiveness. To facilitate direct comparison, the UPFC, as the TCSC, is always placed near port 1 of the transmission line using the model of section 4.3.4.

5.3.1 Case A: UPFC on branch #16 (AAA22-BBB22)

The UPFC is placed at the branch #16 and has the parameters listed in the following table:

<table>
<thead>
<tr>
<th>Input Data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_{max}$</td>
<td>0.5</td>
</tr>
<tr>
<td>$U_{min}$</td>
<td>0.0</td>
</tr>
<tr>
<td>$I_{max}$</td>
<td>10.0</td>
</tr>
<tr>
<td>$\delta_{min}$</td>
<td>0°</td>
</tr>
<tr>
<td>$\delta_{max}$</td>
<td>360°</td>
</tr>
<tr>
<td>$U_S$ control</td>
<td>True/False</td>
</tr>
<tr>
<td>$U_{Smin},U_{Smax}$</td>
<td>0.9, 1.1</td>
</tr>
</tbody>
</table>

Table 5.6: UPFC parameters set by the user (input)

By use of the objective function

$$F_{Obj} = -W_1 \cdot P_B + W_2 \cdot MV A_{UPFC} + W_3 \cdot P_{loss}^{tot}$$

a number of optimization problems are solved with different values for the weight coefficients $W_1, W_2, W_3$ and the results are summarized in Table 5.7. The $MV A_{UPFC}$ is defined in (5.1.2).

Again, Case A1 corresponds to the single objective optimization of the power transfer $P_{FFFF21}$ and at the optimal point the series injected UPFC voltage is $U_T = 0.104$
Table 5.7: Case A: MW power transfer, UPFC MVA power, MW system losses and line side voltage for different weight values in (5.12)

<table>
<thead>
<tr>
<th>Case</th>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$U_s$ control</th>
<th>$P_B$ (MW)</th>
<th>MVA series</th>
<th>MVA shunt</th>
<th>$U_s$ (p.u.)</th>
<th>$P_{\text{loss}}$ (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>$F$</td>
<td>1037.22</td>
<td>33.60</td>
<td>2.28</td>
<td>1.075</td>
<td>52.91</td>
</tr>
<tr>
<td>A2</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>$F$</td>
<td>1033.31</td>
<td>19.90</td>
<td>0.00</td>
<td>1.074</td>
<td>50.88</td>
</tr>
<tr>
<td>A3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>$F$</td>
<td>1031.34</td>
<td>14.85</td>
<td>0.79</td>
<td>1.074</td>
<td>50.11</td>
</tr>
<tr>
<td>A4</td>
<td>10.0</td>
<td>1.0</td>
<td>1.0</td>
<td>$F$</td>
<td>1037.22</td>
<td>33.58</td>
<td>2.29</td>
<td>1.074</td>
<td>52.91</td>
</tr>
<tr>
<td>A5</td>
<td>1.0</td>
<td>10.0</td>
<td>1.0</td>
<td>$F$</td>
<td>1020.62</td>
<td>4.64</td>
<td>0.00</td>
<td>1.088</td>
<td>48.32</td>
</tr>
<tr>
<td>A6</td>
<td>10.0</td>
<td>10.0</td>
<td>1.0</td>
<td>$F$</td>
<td>1032.67</td>
<td>18.13</td>
<td>0.29</td>
<td>1.074</td>
<td>50.61</td>
</tr>
</tbody>
</table>

Figure 5.21 on page 122 shows the loadings of the branches for Case A2. The loading of branch #16 has increased up to 55.83%. In the base case the loading of branch #16 was 24.36% and in the corresponding TCSC case (on page 119) 42.86%.

5.3.2 Case B: UPFC on branch #7 (WWW41-FFF41)

Now the UPFC is placed on branch #7 with the parameters of Table 5.6. Optimization studies with the six combinations of weighting factors and the same objective function (5.12) yield the results of the following table.

<table>
<thead>
<tr>
<th>Case</th>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$U_s$ control</th>
<th>$P_B$ (MW)</th>
<th>MVA series</th>
<th>MVA shunt</th>
<th>$U_s$ (p.u.)</th>
<th>$P_{\text{loss}}$ (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>$F$</td>
<td>928.50</td>
<td>22.06</td>
<td>7.15</td>
<td>1.074</td>
<td>46.04</td>
</tr>
<tr>
<td>B2</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>$F$</td>
<td>928.05</td>
<td>19.18</td>
<td>1.28</td>
<td>1.081</td>
<td>45.91</td>
</tr>
<tr>
<td>B3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>$F$</td>
<td>928.00</td>
<td>19.09</td>
<td>1.02</td>
<td>1.081</td>
<td>45.90</td>
</tr>
<tr>
<td>B4</td>
<td>10.0</td>
<td>1.0</td>
<td>1.0</td>
<td>$F$</td>
<td>928.47</td>
<td>21.20</td>
<td>5.71</td>
<td>1.076</td>
<td>46.00</td>
</tr>
<tr>
<td>B5</td>
<td>1.0</td>
<td>10.0</td>
<td>1.0</td>
<td>$F$</td>
<td>927.76</td>
<td>18.78</td>
<td>0.00</td>
<td>1.082</td>
<td>45.86</td>
</tr>
<tr>
<td>B6</td>
<td>10.0</td>
<td>10.0</td>
<td>1.0</td>
<td>$F$</td>
<td>927.83</td>
<td>18.80</td>
<td>0.00</td>
<td>1.082</td>
<td>45.89</td>
</tr>
</tbody>
</table>

Table 5.8: Case B: MW power transfer, UPFC MVA power, MW system losses and line side voltage for different weight values in (5.12)

Figure 5.19 on page 120 shows the loading of the branches at the optimum. The loading of the transformer #104 is relatively high in the base case. In Case B2 as illustrated in Figure 5.19 the loading of the transformer has reached the upper limit and therefore does not permit any further increase of the transfer from the importing area ZZZ41 to the load FFF21. Observing the loadings of the rest of the network
connecting the generators with the FFF21, it is obvious that the UPFC does not contribute to a more even loading of the available paths.

5.3.3 Case C: UPFC on branch #16 (AAA22-BBB22) and on branch #7 (WWW41-FFF41)

In this case two UPFC’s are placed on branch #16 and #7 each. Again the objective function (5.9) will be used with the six cases. The results of the optimization studies are summarized in the following table.

<table>
<thead>
<tr>
<th>Case</th>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$P_R$ (MW)</th>
<th>UPFC #16 MVA series</th>
<th>UPFC #16 MVA shunt</th>
<th>UPFC #7 MVA series</th>
<th>UPFC #7 MVA shunt</th>
<th>$P_{loss}^{tot}$ (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1058.94</td>
<td>33.48</td>
<td>2.44</td>
<td>1.072</td>
<td>22.59</td>
<td>20.09</td>
</tr>
<tr>
<td>C2</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1052.68</td>
<td>14.74</td>
<td>0.80</td>
<td>1.073</td>
<td>10.79</td>
<td>8.48</td>
</tr>
<tr>
<td>C3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1052.40</td>
<td>13.85</td>
<td>0.93</td>
<td>1.074</td>
<td>10.95</td>
<td>8.48</td>
</tr>
<tr>
<td>C4</td>
<td>10.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1058.90</td>
<td>33.48</td>
<td>2.43</td>
<td>1.072</td>
<td>21.56</td>
<td>19.15</td>
</tr>
<tr>
<td>C5</td>
<td>1.0</td>
<td>10.0</td>
<td>1.0</td>
<td>1042.80</td>
<td>4.36</td>
<td>1.34</td>
<td>1.081</td>
<td>5.46</td>
<td>0.75</td>
</tr>
<tr>
<td>C6</td>
<td>10.0</td>
<td>10.0</td>
<td>1.0</td>
<td>1052.66</td>
<td>14.66</td>
<td>0.82</td>
<td>1.073</td>
<td>10.81</td>
<td>8.36</td>
</tr>
</tbody>
</table>

Table 5.9: Case C: MW power transfer, UPFC MVA power, MW system losses and line side voltage for different weight values in (5.12)

Figure 5.20 on page 121 shows the loadings of the branches for Case C2. Comparing Figures 5.21 and 5.23 on page 124 indicates that the improvement as a result of the installation of the UPFC on #7 (while UPFC on #16 is already installed) is not significant.

5.4 Conclusions

The two-node network is an oversimplified model demonstrating, however, the parallel path problem. It is very useful in verifying the intuitively anticipated results and in illustrating the behaviour of the objective functions with changes in the control variables. The presence of more than one local optimum even in the simplified case of a two-node, two-branch circuit with one UPFC device, requires the use of a more sophisticated global optimization algorithm.

The thirty-seven-node network represents a realistic network, where the MW transfer from generators and generating areas to a certain node is maximized. Optimization based sensitivity analysis for the base case is conducted to determine the best locations for the placement of the TCSC devices. The same locations are used for the UPFC devices. The sensitivities for optimally placing a TCSC working in steady state (capacitive operating region) are large on the underutilized paths. Alternatively, a
TCSC could be placed on the heavily loaded path with fixed line current or with fixed inductive reactance, so that part of the load is transferred to the other available paths.

The main purpose of this chapter was to demonstrate the capabilities of the optimization software in use. The studies provide a rough estimation of the improvement in power transfer by placing series FACTS devices and an approximation of their power capability at the optimum. Contingency studies and cost considerations are required in order to provide a more realistic upper bound on the network MW transfer.
Figure 5.18: Maximum power transfer with one TCSC on branch #16 (Case A2).
Figure 5.19: Maximum power transfer with a TSC on branch #7 (Case B2).
Figure 5.20: Maximum power transfer with two TCSC’s on branches #16 and #7 (Case C2).

\[ P_{lp} = -900.000 \, UL \quad P_{lp} = -175.322 \]

\[ P_{lp} = -1613.339 \quad P_{lp} = -175.313 \]

\[ X_{tcsc} = -0.0010 \quad X_{tcsc} = -0.0138 \]

\[ P_{lp} = -175.000 \, UL \quad P_{lp} = -175.000 \, UL \]

\[ P_{lp} = 1039.170 \]

\[ 179.12\% \quad 80.00\% \]

\[ 21.17\% \quad 29.65\% \]

\[ 34.69\% \quad 79.03\% \]

\[ 25.95\% \quad 17.79\% \]

\[ 49.21\% \quad 18.06\% \]

\[ 80.00\% \quad 80.00\% \]

\[ 73.18\% \quad 8.46\% \quad 6.22\% \quad 39.69\% \]

\[ 58.52\% \quad 18.06\% \quad 18.56\% \quad 42.40\% \]

\[ 70.85\% \quad 23.73\% \quad 5.43\% \]

\[ 38.10\% \quad 25.20\% \]

\[ 34.47\% \quad 36.17\% \]

\[ 34.47\% \quad 34.47\% \]
Figure 5.21: Maximum power transfer with one UPFC on branch #16 (Case A2)

$P_{lp} = -900.000 \ UL \ P_{lp} = -175.367 \ P_{lp} = -175.000 \ UL \ P_{lp} = -175.000 \ UL$

$P_{lp} = 1031.307$
Figure 5.22: Maximum power transfer with one UPFC on branch #7 (Case B2)

\[
P_{ip} = -900.000 \text{ UL} \quad P_{ip} = -175.000 \text{ UL} \quad P_{ip} = -175.000 \text{ UL} \quad P_{ip} = -175.000 \text{ UL}
\]

\[
P_{ip} = -1497.665 \quad 80.00\%
\]

\[
P_{ip} = 928.053
\]
$P_{v_0} = -175.503 \text{ MW}$

$P_{v_1} = -175.553 \text{ MW}$

$P_{v_2} = -900.000 \text{ MW}$

$P_{v_0} = -175.503 \text{ UL}$

Figure 5.23: Maximum power transfer with one UPFC on branch #7 and one UPFC on branch #16 (case C2)
Chapter 6

Summary and Conclusions

The major changes in the electricity market result in an extended use of the transmission network. The diversity of today's transactions places new demands on the power system applications software, primarily on optimization. Minimization of losses and production cost have been the primary objectives for a long time, while the main constraints were the network laws and modeling equations. In a deregulated and market-oriented environment certain solutions for problems or concepts like elimination of loop flows, maximum power transfer between utilities or minimization of the impact of a transaction or a device on an unrelated network area must be considered. The slow response of the conventional network components does not allow the real-time control of the network flows. For this reason the thyristor-controlled FACTS devices are installed, as they provide fast and smooth control of certain network variables. However, they result in a physical change in the system that must be understood and examined.

Focus of this work was the design and implementation of a prototype solver for power system optimization problems that is particularly flexible and easily maintainable. In a working environment where software is developed and used by teams the readability and expandability of the code is of great value. Apart from the conventional network equipment, the optimization package contains the models of the thyristor-controlled FACTS devices. The FACTS devices introduce new control variables and constraints in the optimization problem. In addition, the restructuring in the electricity market calls for new objective functions and new constraints. All the new concepts and network components have to be accommodated in the new design, that in the presence of the continuously changing structure of the competitive market should also allow for easy integration of new functions and features.

One way of ensuring the maintainability and flexibility of the code is to minimize the required hand coding. The technique of automatic differentiation (AD) is used for this reason: It generates code for the first-order derivatives and takes care of the sparsity pattern. The AD technique keeps track of the interdependence of variables in the input code. Therefore, already defined variables can be used to form more complicated functions and the derivatives are propagated accordingly.
There is as yet no AD package that generates code for sparse second-order derivatives “on the fly”. This fact places a first limitation on the choice of the optimization algorithm. On the other hand, the general-purpose optimizers that can handle large-scale nonlinear systems and exploit sparsity are only a few. MINOS, a widely used FORTRAN77 optimization package, uses a projected augmented Lagrangian algorithm for nonlinear constraints and the reduced gradient approach for linear constraints. For reliable computation MINOS requires the user to supply the first-order derivatives of the nonlinear functions, while the second order derivatives are numerically approximated without any user-defined coding. By combining AD with MINOS the user hand coding is confined to the actual interfaces between the various modules, the modeling equations, objective functions and constraints.

In this environment the software integration of the models of the series FACTS devices that have nonlinear functional constraints is shown to require only standardized steps in the existing software environment. Main objective for their integration was the evaluation of their performance in controlling the direction of the line flows. The case studies showed that the series FACTS devices, when placed in key locations in the system, are capable of modifying the natural flow pattern of a meshed network, so that the transferable power increases. The key locations for placing TCSC devices in a real-life 37-node network were identified conducting sensitivity analysis of the optimized base case. The found locations coincide with those intuitively anticipated. TCSC or UPFC placement on these locations result in an increase in the MW power transfer of about 25%. The simulation results provide also a first approximation of the power capability of the installed devices.

The largest part of the simulations consists of multiobjective optimization studies, in which the most important objective is maximizing the MW power delivered to a consumption node. Minimizing the required power of the installed devices and minimization of the network losses are additional, partly conflicting goals. A weighted sum balances the three objectives. The simulation results indicate no significant difference on the maximum achievable power transfer with TCSC or UPFC. However, the ability of the UPFC to control independently the MW and MVAR line flows can be utilized to protect a critical line by regulating its MW and MVAR flows to constant values and exploit the capacity reserves of the other available paths. The weighting factors combinations in the case studies are arbitrary and take no consideration of the relative costs and profits of the individual objectives. However, the results clearly show the trade-off between power capability of the device and effectiveness on maximizing the power transfer.

Suggestions for future work  A straightforward way of extending this research is to expand the package by inserting equipment cost functions for the FACTS devices, the branch flows and the power transfer. Objectives or constraint functions formulating concepts relevant to the electricity market, as the “electronic fence” [58] or prevention of loop flows, can be easily integrated.
Future work could include

- studies considering different contingency scenarios, that would deliver a more realistic upper bound on the achievable maximum MW power transfer. Experimenting with the different control modes and capabilities of the installed devices: Inductive operation of the TCSC, independently controlled MW and MVAR line flow, controlled line current, voltage control etc.

- an algorithm for optimal siting of the FACTS devices. The sensitivities indicate key locations in the network that are optimal with respect to a specific objective around the current operating point. In the case of the TCSC, that has one control variable, combination of sensitivity analysis with experience and good knowledge of the network can help in identifying potential candidates for the placement of the TCSCs. The UPFC, with at least two independent control variables, has a much wider operating range and the sensitivities are much more difficult to compute and to interpret in a straightforward manner, as with the TCSC. Control interactions between the devices must also be considered and examined.

- experimenting with different nonlinear optimization algorithms. The structure of the software allows for easy replacement of the AD and optimization packages. Any new optimizer employing an algorithm for solving large-scale nonlinear problems can be implemented without sacrificing the flexibility of the code. In case that the AD technique allows in the future the efficient generation of sparse second-order derivatives, methods like interior point [59] or unlimited point [60, 61] could be considered.
BIBLIOGRAPHY


Curriculum Vitae

Tina Orfanogianni
born on February 7, 1973
in Athens, Greece

1978 - 1984  3rd Primary school of Kifissia, Athens, Greece
1984 - 1987  2nd Secondary school of Kifissia, Athens, Greece
1987 - 1990  2nd Lyceum of Kifissia, Athens, Greece
1990 - 1995  Student of Electrical Engineering at the National Technical University of Athens (NTUA)
1996 - 2000  Doctorate student and assistant at the Electrical Power Transmission group, Swiss Federal Institute of Technology, Zurich, Switzerland