Modelling of Soyuz Docking and Radar Systems for Implementation in the IRS Simulator

Modellierung des Docking- und Radarsystems für Soyuz-Raumschiffe im Simulator des IRS

Diplomarbeit von
cand. aer. **Karin Schlottke**

IRS-13-S-118

Betreuer:
Prof. Dr.-Ing. Stefanos Fasoulas
Dipl.-Ing. Manuel Schmitz

Institut für Raumfahrtsysteme, Universität Stuttgart
Dezember 2013

Abstract

The Space Systems Institute at the University of Stuttgart offers its students the unique possibility of flying a model Russian Soyuz spacecraft in the institute's own Soyuz simulator. Within the Soyuz Rendezvous and Docking Seminar, which takes place each summer semester, the participating students first have a few theoretical lectures, then the actual flight training begins. The goal of the present thesis is to increase both the complexity and the realism of the simulator and develop the first steps for a fully automated rendezvous and docking. Focus is put on the implementation of the radar system KURS. During the design of the model, a trade-off often takes place between being realistic and being simple enough for unexperienced pilots such as the training aerospace students. This leads to differences between the implementation and the actual systems, both by omitting details in the model and by adding extra features. Additionally, an existing draft of flight procedures is enhanced and adapted to include the newly implemented systems.

Kurzzusammenfassung

Das Institut für Raumfahrtsysteme an der Universität Stuttgart bietet den Studierenden die einzigartige Möglichkeit, ein Modell des russischen Soyuz Raumschiffs im institutseigenen Soyuz Simulator zu fliegen. Im Rahmen des “Soyuz Rendezvous and Docking” Seminars, welches jedes Jahr im Sommersemster statt findet, erhalten die Teilnehmer zunächst eine theoretische Einführung, bevor es dann zu den eigentlichen Flugstunden geht. Das Ziel der vorliegenden Arbeit ist es, sowohl die Komplexität, als auch die Realitätsnähe des Simulators zu erhöhen und dabei die ersten nötigen Schritte für ein vollautomatisches Rendezvous und Docking zu entwickeln. Der Schwerpunkt liegt hierbei auf der Implementierung des Radar Systems KURS. Bei der Entwicklung des Modells muss die richtige Balance zwischen Realitätsnähe und einfacher Bedienung für unerfahrene Piloten (wie die teilnehmenden Luft- und Raumfahrtstudenten) gefunden werden. Die daraus entstehenden Unterschiede zwischen der Implementierung und dem realen System können sowohl Vereinfachungen, als auch zusätzliche Optionen im Modell sein. Darüber hinaus wurde ein Entwurf der sogenannten “Procedures”, d.h. Verfahrensanweisungen für die Nutzung der Systeme, erweitert und für die neu hinzu gekommenen Systeme weiter entwickelt.

Contents

List of Figures

List of Tables

Nomenclature

BZWK **бортовой цифровой вычислительный комплекс** (**БЦВК**). Soyuz digital computer complex.

DPO **двигатели причаливания и ориентации** (**ДПО**). Soyuz attitude and approach control thrusters.

Insertion phase Flight phase that starts with launch and ends with separation from the launcher.

KDU **комбинированная двигательная установка** (**КДУ**). Combined propulsion system.

Operating mode Current state of radar system, depending on the rendezvous progress.

Orbital phase Flight phase that starts with separation from the launcher and ends with reentry.

Phasing phase This flight phase is part of the orbital phase and begins as soon as the vehicle is in its desired orbit. It ends with the onboard computer propagating the state vector.

(Rendezvous) Far phase This flight phase is part of the orbital phase and begins as soon as the onboard computer starts propagating the state vector.

(Rendezvous) Near phase This flight phase is part of the orbital phase and starts at a relative distance of $400\,\mathrm{m}$ from the station.

Scenario Initial state of a simulation.

Simpit An artificial word madeup by combining “simulator” and “cockpit”. It describes a simulator environment which is designed to replicate a vehicle cockpit.

SKD **сближающе-корректирующий двигатель(СКД)**. Soyuz orbital maneuvering engine.

SNC **сигнал наличия целн (СНЦ)**. Target Signal Acquisition.

Vessel An object (or class) derived from the “VESSEL” class. It is usually a spacecraft (e.g. spaceship, satellite, station) or an aircraft.

View An object (or class) derived from the “VIEW” class. Consists of a set of MFD displays depending on the current MFD settings.

Latin Symbols

ch	$-$	channel number of navigation device
f	MHz	frequency of navigation device
S	$-$	navigation signal strength, arbitrary Orbiter units
r	m	distance between signal transmitter and receiver

Greek Symbols

γ	deg	roll misalignment angle
η	deg	heading attitude
η_{Π}	deg	heading bearing
ϑ	deg	pitch attitude
ϑ_{Π}	deg	pitch bearing
ρ	m	relative range from vehicle to station
ρ'	$\mathrm{m/s}$	relative range rate
Ω	$\mathrm{deg/s}$	line of sight angular rate

Acronyms

$3D$	Three-Dimensional
AP	Autopilot
API	Application Programming Interface
ASA	Active Docking Assembly

$BZWK$	Digital Computer Complex
CM	Command Module
DPO	Approach and Attitude Control Thruster
EAC	European Astronaut Center
ESA	European Space Agency
EVA	Extra-Vehicular Activity
$GCTC$	Gagarin Cosmonaut Training Center
IDE	Integrated Development Environment
IDS	Instrument Docking System
IRS	Institut für Raumfahrtsysteme/Space Systems Institute
ISS	International Space Station
KDU	Combined Propulsion System
$KURS$	Radio Technical Rendezvous System
LOS	Line Of Sight
$LVLH$	Local Vertical Local Horizontal
MCC	Mission Control Center
MFD	Multi-Function Display
MGK	Hatch Sealing Mechanism
MGS	Interface Sealing Mechanism
$MSVC$	Microsoft Visual C++
$MSVS$	Microsoft Visual Studio
$NASA$	National Aeronautics and Space Administration
$OAPI$	Orbiter API
OM	Orbital Module
PC	Personal Computer
PSA	Passive Docking Assembly
SDK	Software Development Kit
SKD	Orbital Maneuvering Engine
SM	Service Module
SNC	Target Signal Acquisition
$SSWP$	Docking and Internal Transfer System
SUD	Motion Control System
$XPDR$	Transponder

Chapter 1

Introduction

1.1 Motivation and objectives

The Space Systems Institute (IRS) at the University of Stuttgart is the largest European research institution in various areas of aerospace science. The institute has a large variety of test stands and laboratories which are used for both research and teaching. One of these facilities is the IRS Soyuz simulator, which includes a simulation cockpit (simpit) resembling the actual Soyuz spacecraft.

The simulator is used as part of the Soyuz Rendezvous and Docking Seminar, which consists of a theoretical and a practical part. In the theoretical part, students learn about the International Space Station (ISS) and its docking ports, the rendezvous and docking maneuvers of the Soyuz vehicle, its modules, control systems and docking mechanism and how all of this is modelled in the simulator. Additionally, stress and human factors in space engineering are discussed [3]. Then, in the practical part, the goal is for the students to learn and experience how to fly and operate a complex space vehicle within a typical mission scenario. They employ their motor skills and use their personal audiovisual perception, while being in a stressful situation. In this way, they gain personal insights and experience, realize and improve their audiovisual perception and motor skills, and learn how to handle stress and increase their performance.

So far, both the radar and the docking system of the Soyuz spacecraft have been simulated using the standard routines from the underlying basis software *Orbiter*. The main objective of the present thesis is to increase the complexity and realism of the simulated radar and docking system model in order to achieve a more realistic instrument based rendezvous and docking of the Soyuz to the ISS. A second goal is to obtain procedures for the operation of the modelled systems and an automated approach to the station.

1.2 Approach, methods and organization

As mentioned above, this thesis aims at modelling the radar and docking systems of the Soyuz spacecraft as realisticly as possible within the IRS Soyuz simulator. This study is conducted in collaboration with the European Astronaut Center (EAC) in Cologne, which belongs to the European Space Agency (ESA), in order to gain a better understanding of the two systems. First, the actual radar and docking systems are explained. A closer look is taken at each of their components and their configuration. Focus is also put on the function and operation of the systems, as well as the course of events during the rendezvous and docking phase. Then, both systems are modelled and implemented into the IRS Soyuz simulator in a highly simplified version. Afterwards, the systems are integrated to the already existing guidance system of the simulated Soyuz spacecraft. Finally, procedures are developed for the operation of the modelled systems and the automated apporoach to the station. They can be used by the students participating in the Soyuz Rendezvous and Docking Seminar.

Chapter 2

Description of the modelling problem

2.1 Soyuz radar and docking systems - A short introduction

2.1.1 Soyuz-TMA vehicle

The Soyuz vehicle family is the longest serving manned spacecraft in the world. It was originally designed for the Soviet Manned Lunar program by the Korolyov Design Bureau in the 1960s. The spacecraft is launched on the Soyuz rocket from the Baikonur Cosmodrome in Kazakhstan (see figure 2.1), and can carry a crew of up to three members. Life support can be provided for 30 days without being docked to a station. Once docked, it can stay at the station up to 180 days. At least one Soyuz spacecraft is docked to the International Space Station at all times as an emergency escape craft.

The first unmanned Soyuz was launched in 1966, the first manned mission (Soyuz 1) on April 23, 1967. Over the course of the years, several development steps have been implemented to constantly improve the vehicle. The one currently in use is the sixth generation, the Soyuz-TMA-M. However, as the IRS simulator is based on the fifth generation, the specifications of the so-called Soyuz-TMA vehicle are described in the following. The Soyuz-TMA vehicle was first launched in 2002, and had its last descent on April 27, 2012. It was used by the Russian Federal Space Agency to carry Russian cosmonauts and also NASA and ESA astronauts to and from the International Space Station. Table 2.1 summarises the spacecraft specifications.

The spacecraft consists of three parts: the orbital module (habitation), the command module (used for reentry) and the service module (with solar panels attached), as shown in figure 2.2. Only the command module is reusable and returns back to Earth with the crew. Both orbital and service module are single-use only. They are jettisoned during the descent phase and burn up in the atmosphere during reentry.

The *orbital module OM* is a spheroid pressurized module and is also called the habitation

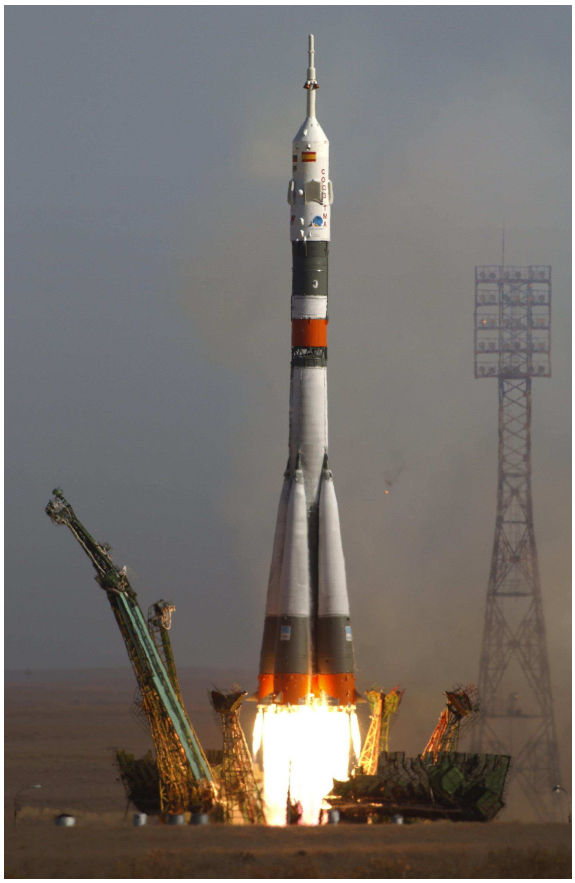


Figure 2.1. Soyuz rocket launch from Baikonour Cosmodrome. Image courtesy of NASA.

Dimensions	Length: $7.48\,\mathrm{m}$
	Max. diameter: $2.72\,\mathrm{m}$
Span (solar array)	$10.70\,\mathrm{m}$
Total mass	$7.2\,\mathrm{t}$
Crew	3
Launch vehicle	Souyz-FG
Landing Systems	Parachute, retro rockets; landing on land
Manufacturer	RKK
Reentry acceleration	$5 - 8\,\mathrm{g}$

Table 2.1. Soyuz-TMA spacecraft specifications [3].

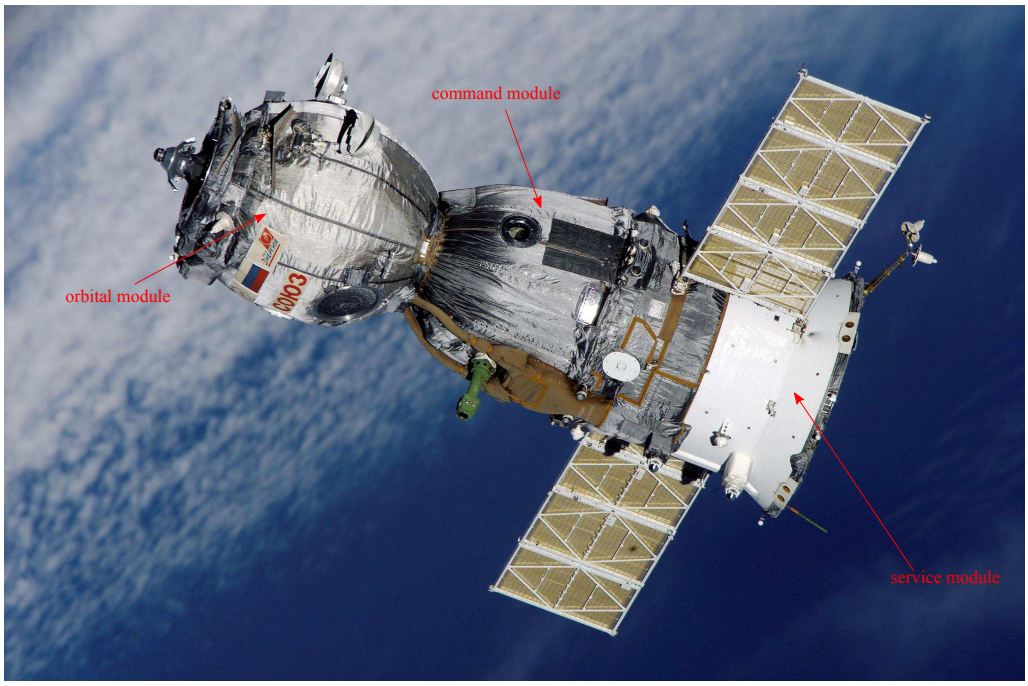


Figure 2.2. Soyuz TMA vehicle with orbital module, command module and service module. Image courtesy of NASA.

section. Figure 2.3 shows the module from the inside. It is mostly used as storage for equipment not needed during reentry, e.g. cameras and experiments. The kitchen and a toilet are also located in the orbital module, as well as the radar system KURS, the life support systems and the docking system. It has a length of $3\,\mathrm{m}$, a diameter of $2.25\,\mathrm{m}$ and a habitable volume of $4.8\,\mathrm{m}^3$. At its far end, it contains the docking port. The hatch at the other end, which connects the orbital and the command module, can be sealed, turning the orbital module into an airlock, in case the crew needs to exit the vehicle when it is not docked to the station. There is an additional hatch on one of the sides, through which the crew exits during an extra-vehicular activity (EVA). This side hatch is also used by the crew to enter the vehicle on the launch pad.

The *command module CM* is also a pressurized module and is the only one returning to Earth at the end of the mission. During ascent, descent and landing, the crew is seated inside the command module. During reentry, the module is protected by an ablative heat shield. First, it is slowed down by the atmosphere. At an altitude of $9\,\mathrm{km}$, a breaking parachute opens and at $7.5\,\mathrm{km}$ altitude, the main parachute slows the vehicle down even further (see figure 2.4). $1\,\mathrm{m}$ above the ground, the solid-propellant breaking engines ignite to ensure a soft landing.

The command module also serves as the cockpit of the Soyuz vehicle and contains control panels, which are depicted in figure 2.5. Additionally, it holds life support systems and an independent guidance, navigation and control system (much simpler than the main one in the service module). These systems are used during the return flight to Earth, after the module has separated from the service module. Moreover, it also contains a

Figure 2.3. Inside view of the orbital module. Image courtesy of NASA.

Figure 2.4. Command module with landing parachute. Image courtesy of NASA.

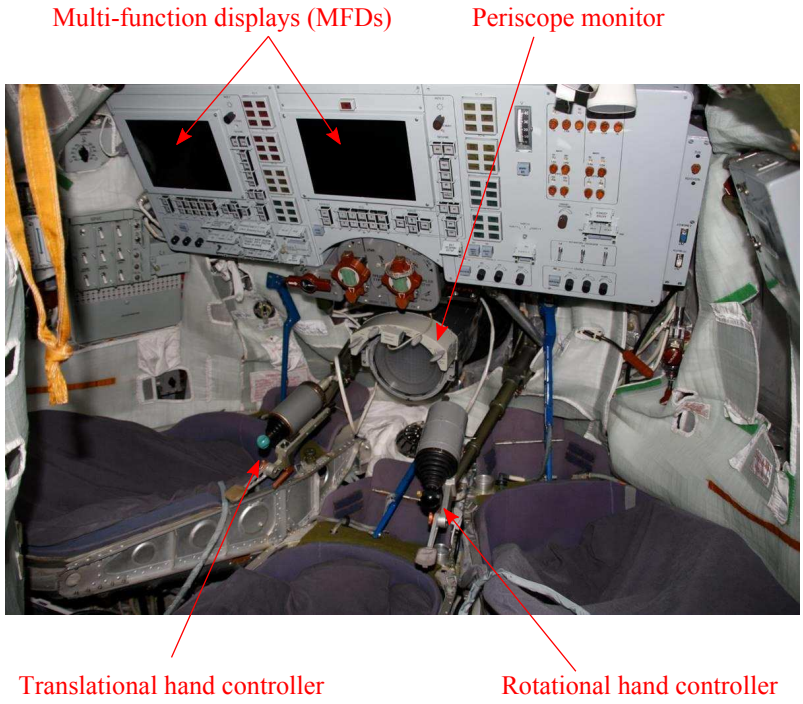


Figure 2.5. Control panel inside the command module. Image courtesy of NASA.

propulsion system for attitude control during reentry. It has a periscope to allow the crew to see the docking target on the station or the Earth below. The module is $2.4\,\mathrm{m}$ long, has a diameter of $2.17\,\mathrm{m}$ and a habitable volume of $3.5\,\mathrm{m}^3$. A payload of up to $50\,\mathrm{kg}$ can be returned to Earth. This value increases to $150\,\mathrm{kg}$ if only two crew members are present. The crew tasks usually contain the surveillance of critical parameters like cabin pressure or oxygen levels. During the automated docking process to the station, the crew members supervise the correct operating sequence by constantly ensuring that all control parameters stay within certain limits and all systems work properly. In case the automated docking fails, the crew can also dock manually, using the hand controllers. Figure 2.6 shows crew members inside the command module together with a printed version of the procedures, which contains a detailed description of all possible crew tasks and which the crew follows step by step at all times. The figure also illustrates how little free space is available in the module.

The *service module SM* is the only non-pressurized module. It contains sytems for thermal control, power supply, radio communications, radio telemetry, as well as instruments for orientation and control. In addition to that, it also contains the combined propulsion system KDU. The module itself has a length of $2.26\,\mathrm{m}$ and is $2.15/2.72\,\mathrm{m}$ in diameter. The solar arrays are also attached to this module. They have a total span of $10.6\,\mathrm{m}$, and with a surface of $10\,\mathrm{m}^2$ they can produce a power of $1\,\mathrm{kW}$. The KDU is a pressure-fed propulsion system, which uses bi-propellant liquid-fuel reactive thrusters. As propellants, an oxidizer (nitrogen tetroxide) and fuel (non-symmetrical dimethylhydrazine) are used and they are stored separately in different tanks. There is an orbital maneuvering engine (SKD), and twenty-eight approach and attitude control thrusters (DPO). Twelve of

Figure 2.6. Crew members inside command module reading procedures. Image courtesy of ESA.

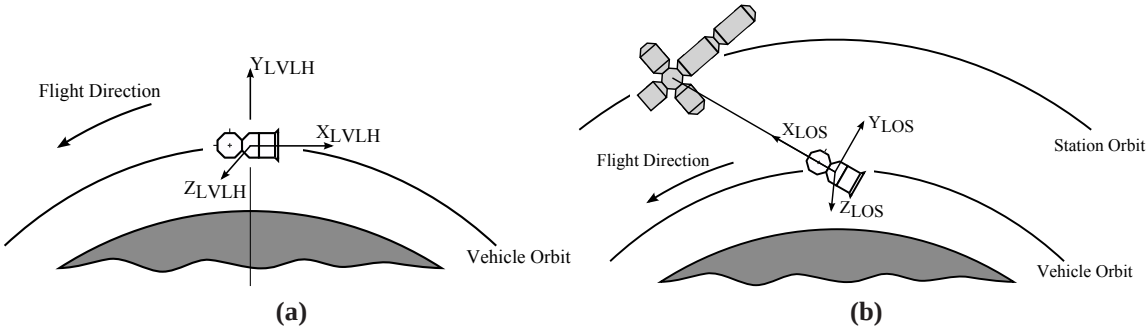


Figure 2.7. Vehicle orienation in space: (a) LVLH; (b) LOS.

the latter are DPO-M (small thrusters) and sixteen are DPO-B (large thursters). They are supplied from two fully redundant manifolds: the first manifold supplies propellant to fourteen of the DPO-B and six of the DPO-M thrusters (DPO-M1); the second manifold supplies two of the DPO-B and six of the DPO-M thrusters (DPO-M2). The small DPO-M1 and DPO-M2 thrusters are only used for vehicle attitude, whereas the large DPO-B thrusters are used for both attitude and translation.

While in orbit, the vehicle can be commanded into different orientations. The two main orientations are called LVLH (Local Vertical, Local Horizontal) and LOS (Line of Sight). Both orientations are shown in figure 2.7. LVLH is acquired during phasing and rendezvous. When the vehicle starts getting closer to the station, LOS is acquired. Only in this orientation, the main radar antennas can measure all the parameters correctly.

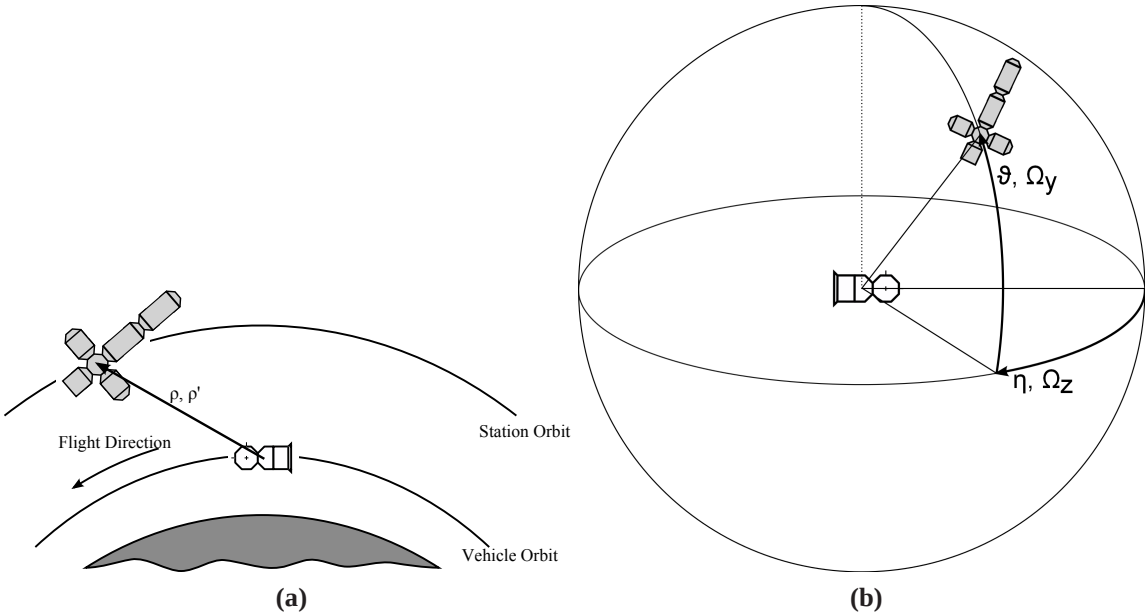


Figure 2.8. Measured relative motion parameters: (a) range ρ and radial closing rate ρ'; (b) heading attitude η, pitch attitude ϑ and line of sight angular rates Ω_y and Ω_z.

2.1.2 Radar system KURS

The radio technical rendezvous system KURS measures relative motion parameters between the Soyuz vehicle and the station during rendezvous, docking and re-docking, thus enabling automated rendezvous and docking to the station. The measured parameters are depicted in figure 2.8 and listed below:

- range ρ
- radial closing rate ρ'
- heading attitude η
- pitch attitude ϑ
- heading bearing η_Π
- pitch bearing ϑ_Π
- roll misalignment angle γ
- line of sight angular rate Ω

The heading and pitch bearing, η_Π and ϑ_Π, determine the chaser (Soyuz) position in the target (station) coordinate system. This means they are the same angles as in figure 2.8b, except with Soyuz and ISS positions swapped. The roll misalignment angle γ is the relative roll angle between the station's docking port and the vehicle. Ω is the LOS angular rate vector and consists of Ω_y, the pitch attitude rate, and Ω_z, the heading attitude rate. The KURS system consists of a total of five antennas (see table 2.2), which are mounted on the orbital and on the service module. Figure 2.9 highlights the antennas on the vehicle.

Depending on the vehicle orientation with respect to the station, the operating range of the KURS system changes. When both the vehicle and the station are not pointed at each other, the range is only $50\,\mathrm{km}$. As soon as the vehicle is pointed at the station, but the

Figure 2.9. KURS antennas on the Soyuz vehicle. Image courtesy of NASA.

station still has an arbitrary orientation, the operating range increases to $200\,\mathrm{km}$. This is the usual case during rendezvous phase. In case both vehicles are pointed at each other, the operating range is as high as $400\,\mathrm{km}$. When the vehicle is pointing at the station during phasing, it is actually not pointed at the desired docking port, but at either one of the station's KURS antennas (XPDR antennas) positioned on each far end of the solar panel of the Zvesda module. Only when the vehicle is close to the station, the LOS orientation is aiming at the KURS antennas of the selected docking port, which are part of the instrument docking system (IDS).

The electronics are made of two identical sets, KURS1 and KURS2. They each contain amongst others a filter, a receiver, a logic unit, and an interface exchange unit, as shown in figure 2.10. When the system is activated, both sets are tested and KURS1 is chosen by default. Both crew and ground control can command the KURS system and observe displayed commands.

There are several operating modes of the KURS system: the *long test* and *short test* mode, which are assumed during the two test phases, the *SNC* mode ("Target Acquired"), assumed after KURS has detected a signal from the station for the first time, *Lock-on* mode starts when the vehicle is in LOS orientation and is auto-tracking the station, and *final approach* is assumed once the vehicle is pointing towards the docking port instead of the KURS antenna positioned on the far end of the solar array.

Antenna	Description
AKR1 / AKR2	receive the signal "Target Acquired", measure ρ and ρ'
2AO	measures η and ϑ, used for LOS orientation, retracted before docking
AKR3	operates together with 2AO
ACΦ1	measures ρ, ρ', η, ϑ, γ, Ω
ACΦ2	measures bearing angles η_Π, ϑ_Π

Table 2.2. Description of KURS antennas.

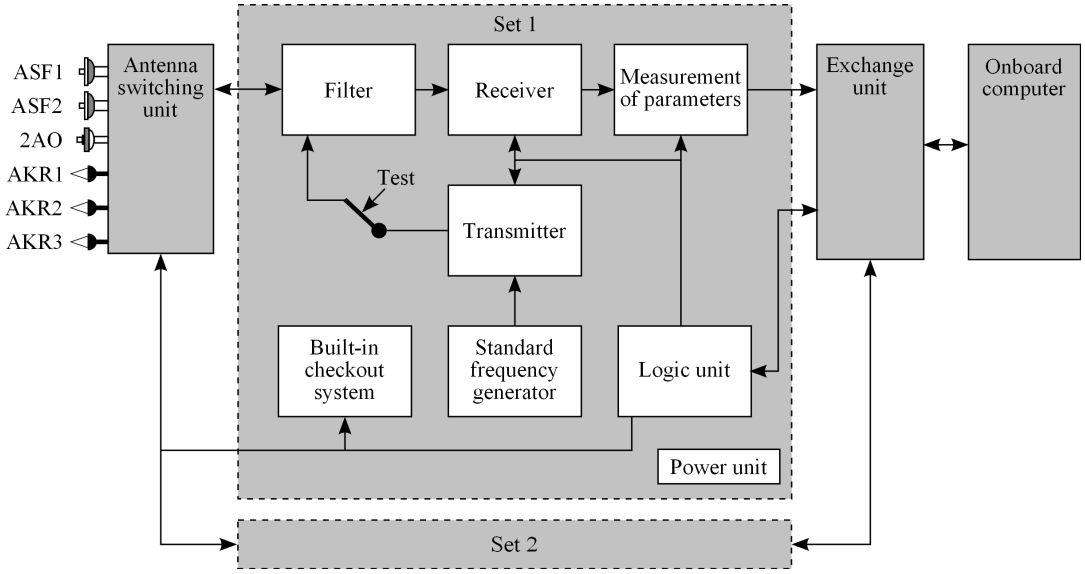


Figure 2.10. KURS system overview

ASA	PSA
docking mechanism	receiving cone
interface sealing mechanism (MGS)	interface sealing mechanism (MGS)
hatch sealing mechanism (MGK)	hatch sealing mechanism (MGK)
electrical connectors	electrical connectors
spring-loaded pushers	spring-loaded pushers
contact sensors	contact sensors

Table 2.3. Major hardware and components of active (ASA) and passive (PSA) docking mechanism.

2.1.3 Docking system SSWP

The docking and internal transfer system SSWP has the purpose of establishing and maintaining a connection between the Soyuz vehicle and the Russian segment of the station. This connection involves not only a pressurized passageway between the two vehicles, but it also establishes connections of common electrical and hydraulic lines, command and control, and atmosphere exchange. The system consists of an active docking assembly (ASA) and passive docking assembly (PSA). This type of docking system is called the probe and cone, or “Classic”, type and is shown in figure 2.11. The ASA is located on the Soyuz orbital module, the PSA is mounted on the station. Table 2.1 lists all major hardware and components of both ASA and PSA.

The *docking mechanism* is attached to the transfer hatch of the Soyuz vehicle. It corrects inital vehicle misalignments and dampens the impact energy. The probe can be extended and retracted by the docking mechanism drive. The head of the front probe has four latches, which are extended and retracted by the latch drive. The moment when the probe head first touches the cone is called “touchdown”. Once the probe head latches are locked

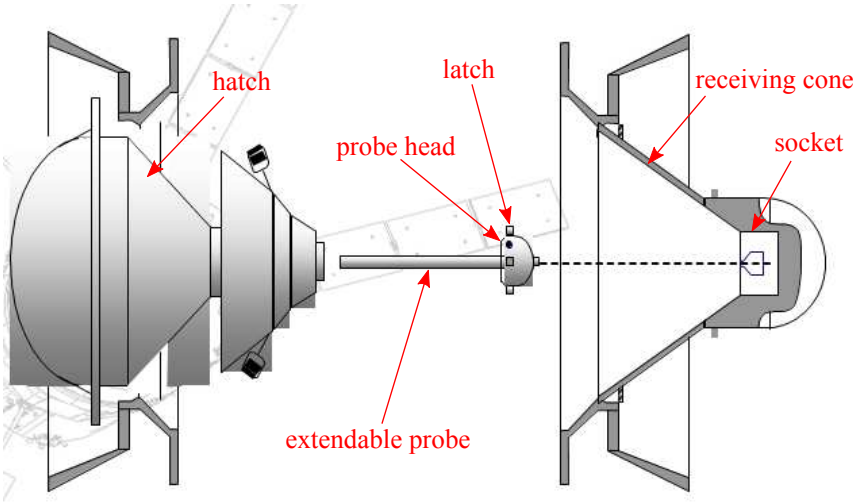


Figure 2.11. Docking mechanism and receiving cone design. Image courtesy of ESA.

in the socket, the docking mechanism draws both vehicles together and mutually aligns them.

The *first mechanical connection* is established once the probe head with extended latches enters the receiving socket of the PSA. When the head enters the socket, the latches overcome the force of the springs, which keep the stops in place. The stops prevent the head from backing out of the cone socket. To break this connection, either the latches need to be retracted or the stops have to be unlocked. During an emergency, pyrotechnics separate the docking mechanism from the hatch cover and the latter remains in the receiving cone socket.

The *interface sealing mechanism MGS* is identical for both the passive and active part. Figure 2.12 shows the MGS on the Soyuz vehicle. It contains an interface sealing device, eight locking mechanisms, two rubber seals, and a braided cable connection. Each locking mechanism has an active and a passive part. Both of them consist of hooks. The active hooks can be controlled by the interface sealing device, the passive hooks are stationary. They are mounted such that an active hook on the station is always facing a passive hook on the Soyuz vehicle, and vice versa. In case of an emergency, pyrotechnic devices can open both types of hooks.

The *second mechanical connection* is established as the hooks close and the docking rings are drawn together. The interface is sealed when the rubber sealing rings are compressed by the docking ring on the PSA. This is also called the “rubber on metal” contact. In order to increase the load-bearing capacity, the crew manually adds several screw clamps.

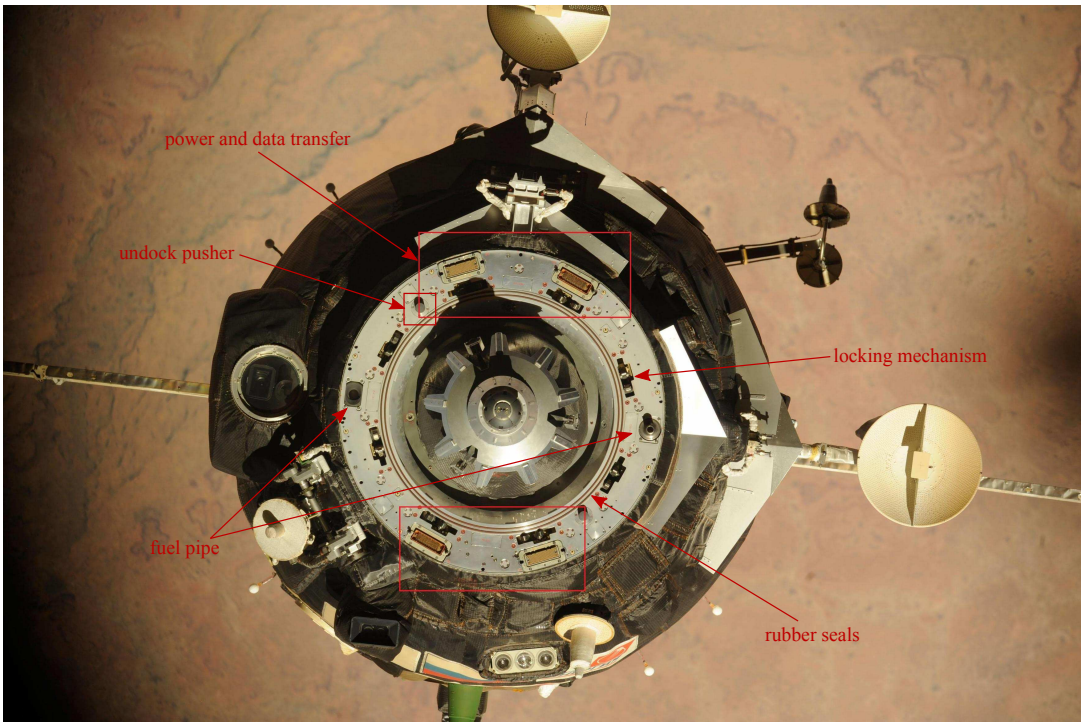


Figure 2.12. Soyuz docking interface. Image courtesy of NASA.

2.1.4 Soyuz rendezvous and docking sequence

2.1.4.1 Overview

After the Soyuz vehicle is launched on board the Soyuz rocket from the Baikonour Cosmodrome, it takes approximately nine minutes until the Soyuz vehicle is separated from the launcher vehicle, indicating the end of the insertion phase and the start of the orbital phase (see figure 2.13). Depending on the chosen approach method (short or long), the phasing phase that follows lasts either a few hours or two days. The critical parameter is the phase angle. This is the angle from the Earth, to the ISS, to the Soyuz, as shown in figure 2.14. As the two vehicles have different orbital periods, the phase angle changes over time. Before initiating the transfer orbit, the phase angle has to be exactly such that when the Soyuz vehicle reaches the Station's orbit, the Station will be at the same position, and hence their phase angle is zero.

Taking the short approach, the phase angle is already comparatively small when the orbital phase starts and it is only allowed to be within a narrow range. Taking the long approach, the phase angle is not quite as constrained, as the two days in the phasing orbit are used to decrease it over a longer period of time. However, no matter which approach is chosen, the rendezvous and docking phase is still identical from a qualitative point of view. Only the time between launch and start of rendezvous varies as well as the time between start of rendezvous and docking.

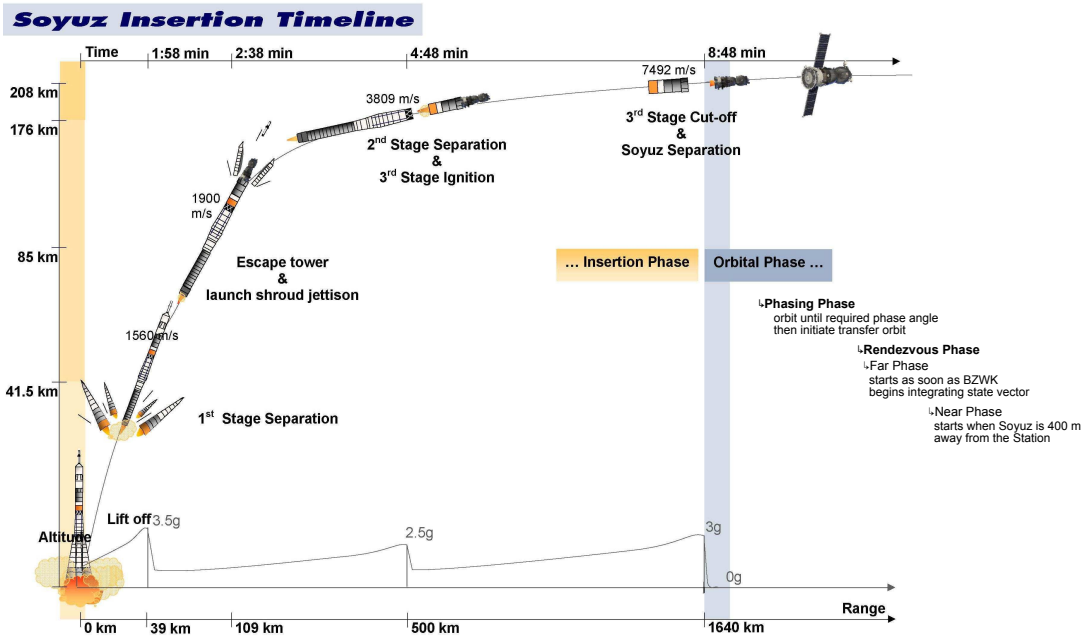


Figure 2.13. Soyuz flight phases. Image courtesy of ESA.

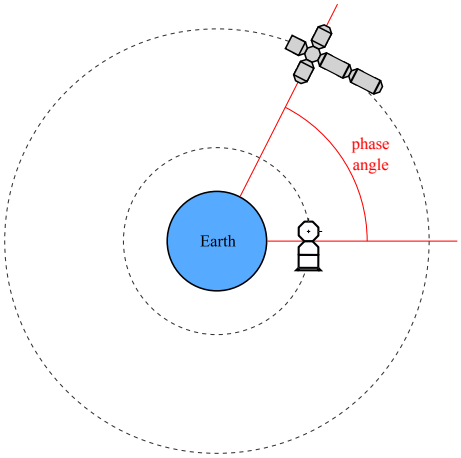


Figure 2.14. Phase angle between Soyuz vehicle and ISS.

2.1.4.2 Timeline

Rendezvous usually starts with the beginning of the first orbit on the respective day (orbit 33 in the long approach). During the phasing phase, ground control (MCC) measures the position and orientation of both the station and the Soyuz spacecraft. In the meantime, the crew onboard the Soyuz starts preparing the rendezvous phase by setting up the onboard computer (BZWK), selecting the thruster set (DPO-M1 or DPO-M2) and closing the hatch to the orbital module. They then check the parameters for the pressure tanks, as well as the remaining fuel and prepare the propulsion system, the optical devices and the hand controllers. The onboard computer is activated, which sends an automatic signal to pressurize the fuel tanks. The crew also activates the accelerometer and the sensors for angular rate. Then, the crew turns on the monitors on the control panel. The next step is the activation of the onboard motion control system (SUD), which then in turn commands the DPO in order for the vehicle to acquire LVLH orientation. Once the vehicle is in LVLH, the infared sensors are activated, which measure Earth's horizon in order to keep the spacecraft at its orientation.

At T_0, the far phase of the rendezvous starts, during which a bi-elliptical transfer from the phasing to the final orbit is performed (see figure 2.15). From this point on, the onboard computer of the vehicle starts integrating the equations of motion in order to determine its current state vector (position and velocity), as well as the current state vector of the station. The calculations are based on measurements from ground control, which the vehicle received shortly before T_0. The crew monitors ρ, ρ', Ω_y and Ω_z. Then, the onboard computer calculates the first correction burn ignition time. The vehicle is rotated such that the SKD thrust vector points in the direction of the rendezvous burn Δv_1. The SKD then completes the first correction burn and the vehicle returns to LVLH orientation afterwards and continues along the internal transfer orbit. For saftey reasons, the rendezvous target is offseted by one kilometer in the station's off-plane direction (see figure 2.16), in order to avoid any possibility of a crash in case the rendezvous phase should fail.

KURS is activated at T_1, between the first and second burn, approximately $800\,\mathrm{km}$ from the station. After the long test "Test D" is completed and both set 2 and set 1 tested, KURS switches to the normal operating mode. The command "Omni-directional search" is issued and antennas AKR1 and AKR2 alternately connect to the receiver and transmitter at a frequency of $1\,\mathrm{kHz}$. By doing so, they can receive signals emitted by the station's KURS antenna and transmit a nonmodulated 3240 or $3245\,\mathrm{MHz}$ homing beacon signal in any direction around the Soyuz.

As soon as one of the antennas receives a reliable signal, the command "SNC" is issued and the search mode is terminated. Whichever antenna generated the "SNC" command, stays connected to the transmitter and receiver. At the same time, the LOS orientation mode and antenna 2AO are activated. 2AO measures the heading and pitch angles η and ϑ and sends them to the control system. When the angular misalignment of each angle is less than $5°$, KURS issues the command "Auto-tracking". As a consequence, antennas AKR1 and 2AO are deactivated and antenna ASF1 is activated and connected to the receiver and transmitter. Now ASF1 measures the angles η and ϑ. The navigation, control and guidance system SUD now uses KURS data for attitude control to keep the vehicle in

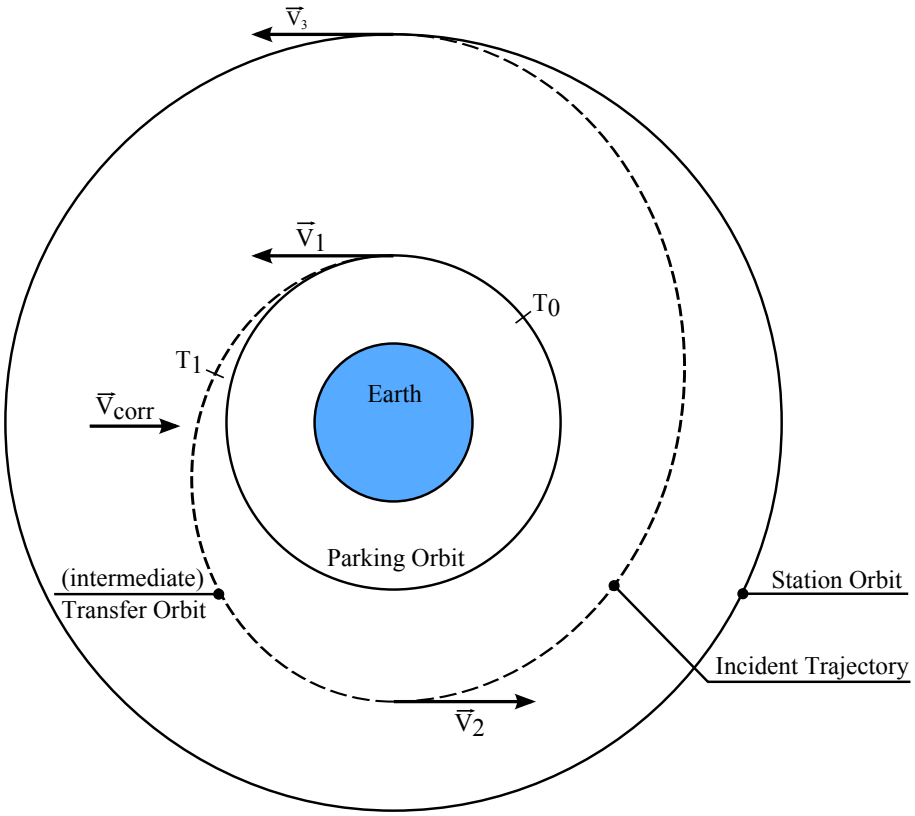


Figure 2.15. Far phase of rendezvous.

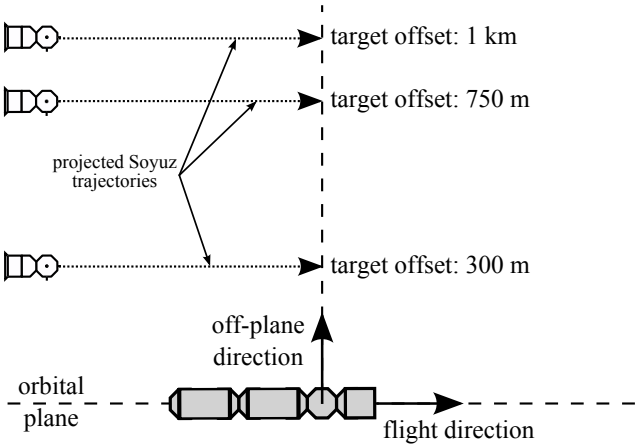


Figure 2.16. Target offset during rendezvous phase.

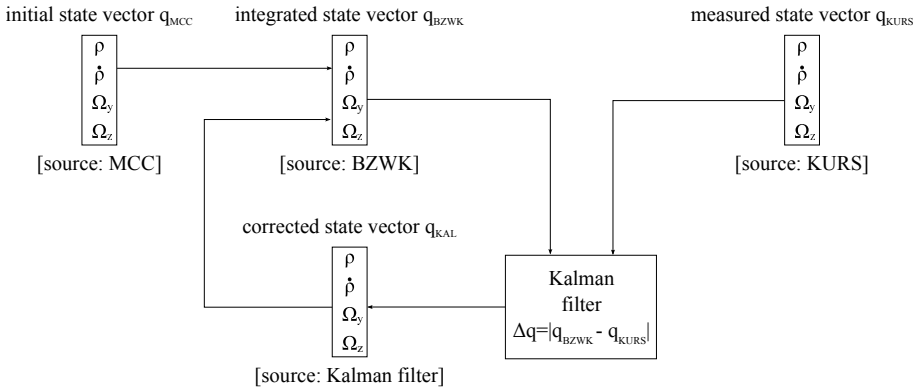


Figure 2.17. Block diagram of Kalman filter and its parameters.

LOS orientation, instead of predictions from the onboard computer. The measured angles correspond to remaining misalignments which SUD needs to correct. At the same time, the measurement channels for the relative distance and closing rate, ρ and ρ', respectively, are activated. Once the vehicle receives measurements for ρ and ρ', the command "Lock-on" is issued. Now the vehicle KURS system is able to measure ρ, ρ', Ω_y and Ω_z. Then, the onboard computer starts a Kalman filter on the integrated state vector to match it with the measured values and correct its prediction, as shown in the block diagram in figure 2.17.

Regardless of whether "Lock-on" has already been issued or not, the motion control system determines the next burn v_{corr}, which usually takes place approximately a quarter orbit after the first large engine burn v_1. As this is only a small correction burn, only the DPO-B engines are employed and the vehicle does not have to change its orientation for the burn. After the correction burn is completed, the vehicle continues along the internal transfer orbit to the offset target.

At the specific time that the onboard computer has calculated for firing the second burn v_2, the vehicle is rotated in the correct attitude and the SKD is fired for the calculated burn duration. Afterwards, LOS orientation is acquired again. The vehicle then continues to the offset target, which is still located one kilometer from the station in off-plane direction. The completion of both of the burns v_{corr} and v_2 marks the time T_2, at which the distance to the station is around $60 - 80\,\mathrm{km}$. If by this time, KURS has not generated the command "Lock-on", a switch is made to the alternate system at $T_2 + 2\,\mathrm{min}$. That is, if KURS 1 was runnning so far, it is switched to KURS 2 (if this one is functional). At at relative distance of $\rho = 15\,\mathrm{km}$, a short test on the "hot" KURS system is performed in order to prevent any errors in relative range measurements during the close approach.

Finally, the onboard computer calculates the firing time for the third engine burn v_3. This rendezvous burn, which cancels out all relative velocity between the vehicle and the target, actually consists of two parts. During the first part, the vehicle is rotated almost by $180\,^\circ$ and the SKD engine is used to rapidly decrease the relative velocity between the two vehicles. The offset target is first reduced to a distance of $750\,\mathrm{m}$ from the station, then to $300\,\mathrm{m}$. The second part of v_3 itself again consists of several smaller burns of the DPO engines. The completion of the third engine burn also marks the end of the far phase of

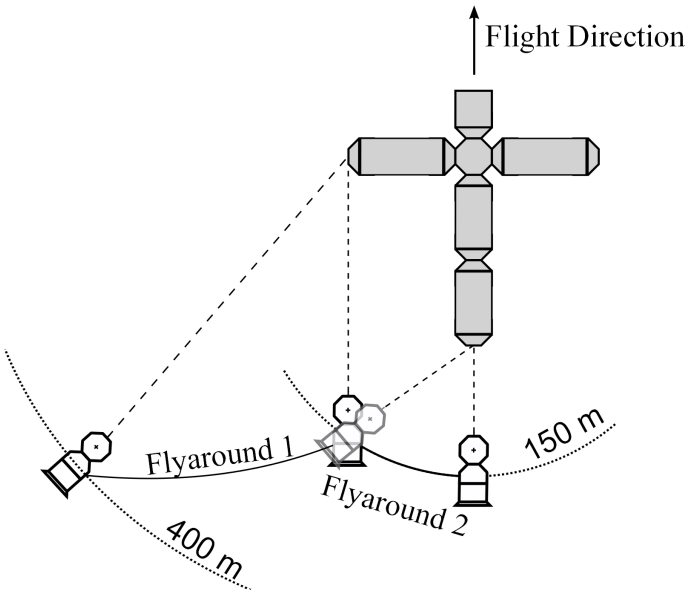


Figure 2.18. Near phase of rendezvous.

the rendezvous.

In order to transition to the near phase, certain requirements need to be fulfilled: The relative distance ρ has to be less than $400\,\mathrm{m}$, the relative velocity ρ' less than $2\,\mathrm{m/s}$ and the relative angular rate has to be smaller than $0.3\,^\circ$ ([2]). The vehicle then performs a flyaround to align the vehicle axis with the station axis (see figure 2.18, while decreasing the relative distance to $\rho = 150\,\mathrm{m}$ and targeting the antenna on the station, whose signal was used as a target for the "Lock-on" command. Once the flyaround is completed, the vehicle shortly enters the station keeping mode, keeping both the relative velocity ρ' and the angular rates Ω_y and Ω_z at zero.

KURS then switches to "Final Approach" mode and shifts from being locked on to the homing beacon antenna of the station to the KURS antenna on the selected station docking port. As a consequence, the vehicle acquires the new LOS orientation. A second flyaround is executed, this time while keeping the relative distance of $150\,\mathrm{m}$. Once the vehicle is aligned with the desired docking port, it performs station keeping again. The crew then issues the final approach command and the vehicle approaches the station while maintaining the LOS orientation relative to the docking port. At a relative distance of $40\,\mathrm{m}$, the 2AO antenna boom automatically retracts, as docking with the antenna still deployed is prohibited due to safety reasons. As soon as the probe head touches the cone, the motion control system enters the "touchdown" mode, which causes the thrusters to push the vehicle forward. Due to the present microgravity, two spacecraft touching each other might lead to the effect of the two vessels pushing themselves away from each other. Therefore, upon touchdown, the thrusters give the Soyuz an extra push, and the two vehicles finally dock.

As soon as the probe head is captured by the station socket, the command "capture" is generated, KURS shuts down and the motion control system enters the "free drift"

mode. This prevents the thrusters from trying to correct the vehicle attitude while the probe head gets retracted. Figure 2.19 shows the Soyuz vehicle and the ISS in the docked configuration. Twenty minutes after the vehicle and station docking mechanisms engage, the motion control system is automatically deactivated. The crew then performs a series of leak checks before they are finally able to open the hatch which connects them to the station, where they are usually greeted by the current crew of the station.

2.2 Soyuz simulator at the Space Systems Institute

2.2.1 Soyuz simulator facilities

The project “Soyuz simulator at the Space Systems Institute” started in 2007 under the direction of Prof. Ernst Messerschmid, who is a former astronaut and was in space in 1985. The very first version of the simulator consisted of two off-the-shelf personal computers and control sticks, which were directly purchased from the Gagarin Cosmonaut Training Center (GCTC) in Russia. In summer 2008, the first training seminar for students had an overwhelming response and over the course of the past years, the simulator has been upgraded step by step.

Today, the simulator includes a model of the Soyuz capsule, which offers the students a semi-realistic cockpit environment, and a simplified ground station for the supervision by the flight instructor. The capsule is an original sized model of the orbital module of the Soyuz spacecraft and was developed at EAC/ESA in Cologne for their own simulator. It has a basic diameter of $2.3\,\mathrm{m}$ and a height of $2\,\mathrm{m}$. For an easy access, the capsule can be opened in the middle, as shown in figure 2.20.

Inside, the capsule features all the important elements of the cockpit, as indicated in figure 2.21: First, there is the integrated operational panel including two multi-function displays (MFD) and several switches. The view through the periscope of the vehicle is simulated on another monitor, allowing to see what is in front of the spacecraft. For motion control around all six degrees of freedom, two control sticks are installed. The left stick controls all translational movements, i.e. forward/backward, right/left and up/down. All rotations around the vehicle’s three axes, i.e. pitch, yaw and roll, are controlled by the right stick.

The crew is seated in three rather narrow seats, with the flight engineer on the left, the mission specialist on the right, and the captain/pilot in the center. However, during the “Soyuz Rendezvous and Docking” seminar, usually only the pilot’s seat is occupied.

Figure 2.22 shows the ground control station with its several personal computers and screens. This is where the simulator framework, the free software “Orbiter Space Flight Simulator”, developed by Dr. Martin Schweiger, is run. For the software to include the control of the cockpit and the Soyuz systems, so called add-ons were implemented at the IRS. These add-ons consist of approximately 25,000 lines of code. A more detailed

Figure 2.19. Soyuz vehicle docked to the International Space Station. Image courtesy of NASA.

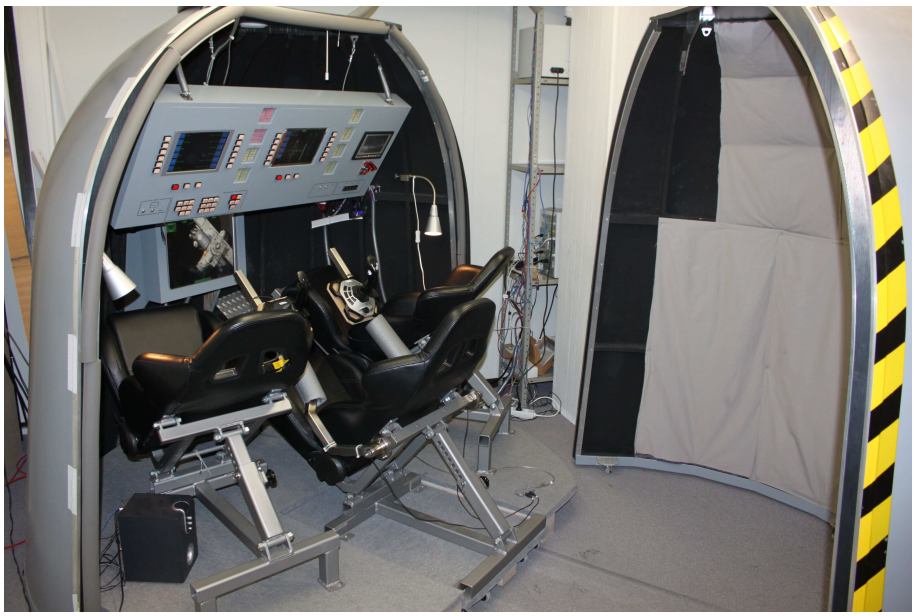


Figure 2.20. Model of Soyuz capsule at the Space Systems Institute. Image courtesy of IRS.

Figure 2.21. Simulator cockpit. Image courtesy of IRS.

Figure 2.22. “Ground station” of the IRS simulator. Image courtesy of IRS.

description of the Orbiter software can be found in section 2.2.2. More information on the implementation at the IRS can be found in [8].

The flight instructors can load different flight scenarios from the ground control station and supervise the practicing student. They can also intervene a running simulation to assist the student from outside the cockpit in case of any deviations from the flight plan, as well as purposely cause a malfunction in a subsystem of the vehicle.

2.2.2 Orbiter space flight simulator

Orbiter is a real-time 3D space flight simulator for Windows PC, developed to simulate space flight using realistic Newtonian physics. Its concept is very similar to traditional flight simulator softwares, however without being limited to atmospheric flight. Orbiter was first released in November 2000, and its latest version was launched in August 2010. Originally, the software was developed by Dr. Martin Schweiger, a senior research fellow at the University College London, who was unsatisfied with space flight simulators lacking in realistic physics-based flight models. It is written in C++ and uses DirectX for 3D rendering.

In Orbiter, the user can experience manned and unmanned space flight missions from a pilot’s point of view. This includes all phases of a mission: Launch, orbital insertion, rendezvous with space stations, deploy and recapture of satellites, reentry and landing on a planetary surface. However, there are no predefined missions to accomplish or opponents to be defeated. Moreover, Orbiter is about learning what is involved in real space flight: What do you need to know when you want to launch into a certain orbit? What is important when trying to rendezvous with a space station? Or what are the difficulties when

flying to another planet?

The Orbiter software itself is basically just a skeleton that defines the physcial model. Included in the core software are a few spacecraft and most of the bodies in our solar system. Even though the program source code is not published, in return there is an extensive Application Programming Interface (API), which allows users to contribute to the software by creating so-called add-ons. A multitude of such add-ons has been developed by the Orbiter community and is largely available on the web: There are additional spacecraft, celestial bodies, enhanced instruments etc.. The Soyuz and ISS models used at the IRS (and among others also further developed as part of the present thesis) are basically also add-ons to the core software.

Chapter 3

Method and tools

3.1 Object-oriented programming

The simulator framework, i.e. the Orbiter space flight simulator, is written in the object-oriented programming language C++. The following section aims at giving a short introduction to both the object-oriented programming paradigm and the C++ programming language, in order to gain a better understanding of the underlying programming concepts of the IRS simulator and the advantages they bring about.

Different approaches to programming have developed over time and the resulting languages are defined by differentiating between paradigms. There are four main paradigms ([5]): imperative, functional, object-oriented and logic programming. Different programming paradigms use different ways to model the information and how it is processed and they have different concepts on how information and processing interact. Some languages are designed to support only one particular paradigm, while other languages can support multiple paradigms. As the Orbiter code is mainly[1] object-oriented, only this paradigm will be explained in more detail hereafter and an elaborate description of the other paradigms is omitted.

Object-oriented programming derives its basic principles from real world processes and objects. These processes are modelled through acting individuals, who perform and assign tasks. In object-oriented programming, those individuals are called objects.

An object is an entity consisting of a data structure in concert with a definition of related operations. All of the used data is distributed among the objects, and additionally, there are no global operations. Each operation directly belongs to an object and can only be engaged by sending a message to the respective object. This technique is called encapsulation. The variables defined locally for an object are called attributes and the local operations are called methods. The description of those local methods is usually done using procedural programming.

[1] The code also contains procedural parts while at the same time lacking some typical object-oriented concepts, such as streams.

An object has a well-defined interface, which describes the properties of the object: the messages or operators, which the object understands and a list of those attributes accessible from outside the object. Ideally, attributes can only be accessed through a method and the object has total control over its data. This technique can rule out inconsistent or unphysical object states. Objects are classified depending on their interface and these interfaces can be inherited by subsidiary classes.

A program can be understood as a system of cooperating objects. These objects have a state, a life span and they exchange messages with each other. While an object is processing a received message, it can change its own state, send messages to other objects (or itself), create or destroy other objects. Objects behave like items in the material world; they are said to have an identity: An object cannot be present at two places at the same time and it can change its state while still staying the same object. Just like a propellant tank can either be full, empty or somewhere in between, and still stay the same propellant tank. This is also the main difference to mathematical objects like numbers and facts. Object-oriented programming models the real world as a *virtual world*. Programs try to reflect as far as possible (or as necessary) that part of reality they are going to treat.

A central thought of object-oriented programming is the separation of the task assignment and the task completion. If one object needs a task to be done, it will look for another object who is capable of performing the task. It then sends a message to the object which has the corresponding method for the task completion. The client object is ignorant of the details on how the executing object performs the task. This principle is also called information hiding. The executing object receives the message and has the corresponding method to respond to it and this is all the client object needs to know. An example for this is the communication between the Soyuz' OM and the KURS system. When the OM wants to know the LOS angles, it asks KURS to compute them and KURS provides the desired answer. The OM only knows that KURS has a method to calculate those angles and it knows what information is required by KURS to do so. It does not know however, what particular calculations are performed.

Another important principle of object-oriented programming is classification and inheritance. A class defines all the methods and properties, which all its objects have in common. Classes can be organized hierarchically. Superior classes only own those properties, that the subsidiary classes/objects have in common. Subsidiary classes inherit properties and methods from the superior classes. Those properties and methods only need to be defined once for the superior class. However, inherited methods can be further adjusted and defined in more detail in the respective class. In Orbiter, for example, the OM, SM and CM are modelled as sub-classes of the "VESSEL3" class, which is itself derived from the "VESSEL" class. Thus, the OM, SM and CM inherit all the methods and properties of the superior "VESSEL" class, and in addition, they each have their own specialized properties.

All in all, object-oriented programming offers an easy way to enhance and reuse existing programs. Due to the application of the messaging system, the connection between a service request (sending of a message) and the method (executable part of the program) only happens during runtime, which makes this a very dynamic programming paradigm.

Objects are easy to extend and specialize using additional attributes and methods, which only lead to a larger interface. However, extended objects still match the old interface, which makes extended objects easy to implement.

3.2 Introduction to C++

The C++ programming language development started in the Bell Labs in Murray, New Jersey, in 1979 and aimed at extending the C programming language by adding object-orientation. It is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language. Note that just because a program is written in C++, this does not automatically imply object-orientation. C++ supports procedural, object-oriented and generic programming.

For the Orbiter space flight simulator, the C++ language holds many advantages ([6]): It offers a simple and safe usage and a high reusability. It enables easy-to-maintain and well-written code, which makes the simulator easy to extend and enhance without great expense. Additionally, C++ is a compiled language, meaning that programs can be distributed to people who do not need to have the respective compiler in order for them to use the program. This makes the simulator a portable program where the end user does not have to be a software developer to be able to use it. But on the other hand, it still enables the user to enhance the program by the so-called add-ons if he desires to do so. How this is done exactly is described in the following section about the Orbiter Software Development Kit (SDK).

3.3 Programming tools

The Orbiter SDK can be downloaded from the same website as the simulator itself [13]. It contains the application programming interface (API) in the form of some libraries, code examples, a few utilities and useful documentation ([12, 10, 11]). The API includes the interface methods; a set of functions for getting and setting general simulation parameters in a running Orbiter simulation session. These methods can be used by all types of plugin modules and their name always starts with “oapi”.

Furthermore, the Orbiter API (OAPI) also contains the properties and methods of the “VESSEL” class and its two derivatives, “VESSEL2” and “VESSEL3”. These classes are the base classes for creating new vessels, e.g. the Soyuz’ orbital module. For creating new multi-function display modes, developers need the “MFD” and “MFD2” classes, which are also part of the API.

In addition to the API provided by Orbiter, there is the so-called “oapiExt”, a function library developed at the IRS [7]. It was developed for modelling the Soyuz spacecraft, but the code itself is generic and can also be used in other vessels. It contains for example an

autopilot (AP) and a timer for counting down the simulation time. More information can be found in [7].

All code developing, debugging and management is done using Microsoft Visual Studio 2010 (MSVS), which is an integrated development environment (IDE). It contains the IDE Microsoft Visual C++ (MSVC), which is designed for C++ programming tasks.

Chapter 4

Model and implementation

4.1 Overview

This chapter describes how the existing simulator code was enhanced in order to implement the Soyuz KURS system. First, the so-called back end of KURS is explained: The internal parts of the systems, the different operating modes, etc.. Second, the KURS front end, i.e. the user interface, is described: The different views on the MFD, the information available to the pilot and the decisions he has to make with respect to KURS. Finally, the procedures are presented. These are the checklists telling the pilots for example, which parameters they need to monitor and which systems they have to activate or deactivate at what time. As stated in chapter 1, originally, also the implementation of the internal docking and transfer system was part of the present thesis. Throughout the project however, it became clear that focus was going to be put on the implementation of the radar system. This seemed the more relevant/interesting system for the aerospace engineering students participating in the Soyuz Seminar to be adressed in this time-constraint thesis.

A general concern while performing the implementation of the systems is the question how close the model should resemble the real system. On the one hand, the simulator aims at providing the students with a realistic environment to experience and understand the challenges and tasks of a pilot as close to reality as possible. On the other hand, the training students should not be overwhelmed by the complexity of the simulator and they should be able to acquire appropriate spacecraft operation skills over the short training period of a few weeks. Finding the right balance between being simple and being realistic, and between what is feasible and what is reasonable, is one of the many challenges of the present thesis.

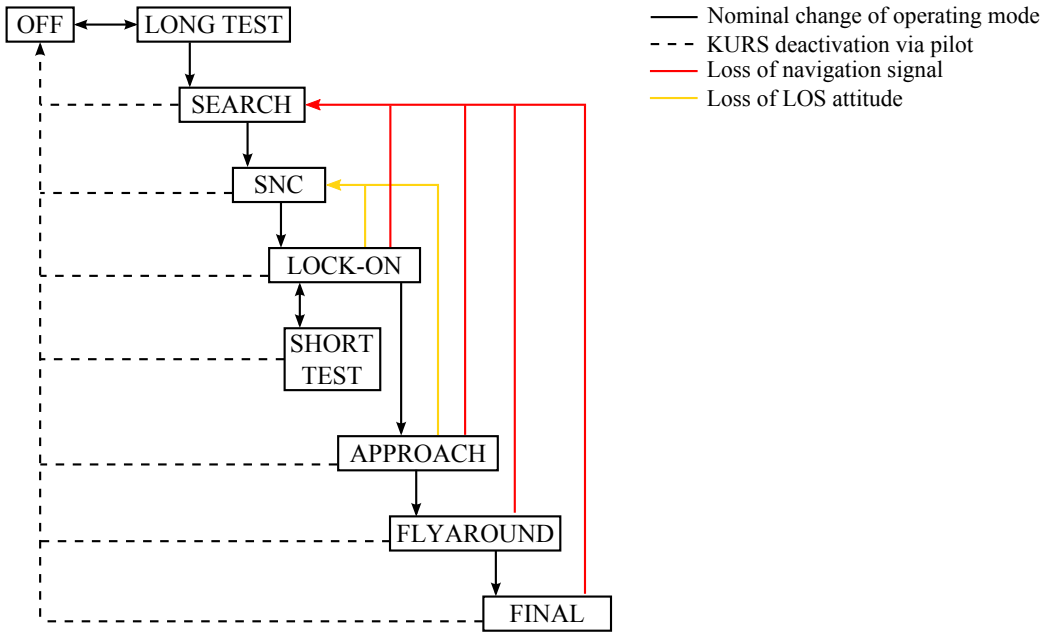


Figure 4.1. Schematic of KURS operating modes.

4.2 KURS back end

As mentioned above, the KURS back end basically consists of everything that the user, i.e. the pilot, cannot see or manipulate directly. The scope of the back end goes from the modelling of the antenna signal range to the calculation of the relative motion parameters up to the commands sent to the autopilot. Depending on the current vehicle state in the rendezvous and docking sequence, KURS has several different operating modes. These modes determine for example which parameters are currently evaluated and what actions have to be taken. As the KURS system usually acquires these modes in a chronological order, from the far phase to near phase to mechanical docking, they will be described in the same order in the following. Figure 4.1 illustrates the relationship between the different modes.

The system is desigend such that it is turned on after the rendezvous far phase has started. This means, that at this point, the vehicle has already left its insertion orbit and is in the transfer orbit. The rendezvous burns are not calculated by KURS, but by the onboard computer BZWK, which is part of the motion control system SUD (see figure 4.2). The BZWK then in turn commands the KDU system to perform the burns. As neither the BZWK nor SUD were part of the present thesis, the automatic calculation and execution of the necessary burns is not implemented in the simulator at this point, but can be added in the future. Until then, the burns have to be performed manually.

When the KURS system is started in the simulator, it automatically assumes that all antennas have been deployed successfully and that the docking probe head is fully extended.

The KURS system is implemented in the simulator as a *C++* class, and the Soyuz OM

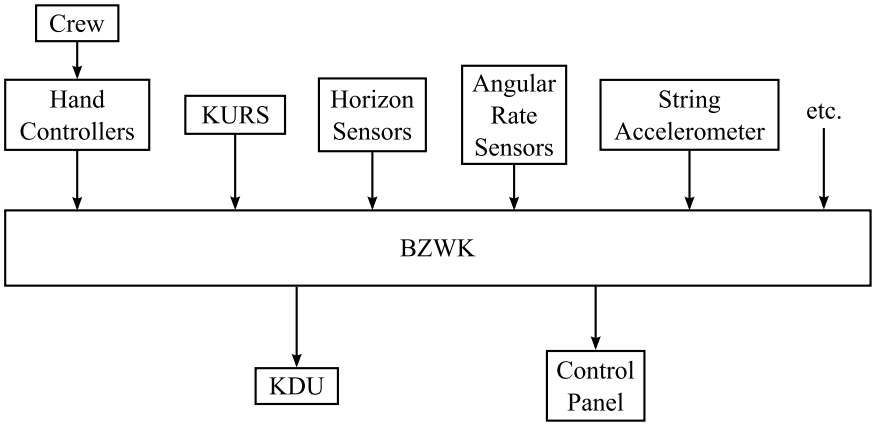


Figure 4.2. Functional schematic of the motion control system SUD.

vessel automatically creates an instance of this class when the simulation is loaded and initialized. Before every time step, the OM checks the current operating mode of KURS using the built-in function of all vessels *clbkPreStep()* from the Orbiter API. Depending on the operating mode, different commands will be executed by the KURS object.

Independent of the current operating mode, the KURS model always determines the current LOS angles (η, ϑ and Ω). In reality, the angles are either calculated by the BZWK or measured by the KURS system. In the implemented simulator model, the values representing BZWK data are calculated from the CM's center of gravity to the position of the received KURS signal transmitter. The "real" measured KURS data is calculated from the Soyuz KURS antenna to the signal source in the model. Additionally, the range and range rate (ρ and ρ', respectively) are calculated similar to the BZWK data, i.e. from the CM's center of gravity to the target transmitter.

The calculation of the LOS angles from the BZWK system takes place as follows. First, the KURS model determines whether a navigation signal is received, and whether it is from the correct transmitter type (XPDR or IDS). Then, the model uses the API function *oapiGetNavPos()* to determine the position of the received signal transmitter. Next, it calculates the position of the CM's center of gravity in global coordinates via the function *GetGlobalPos()*. This function determines the position of a vessel's center of gravity and is also part of the OAPI. By subtracting the two positions, the BZWK line of sight is obtained and can be rotated into local Soyuz coordinates. Now the LOS angles can be determined. The LOS vector is projected into the vehicle's orbital plane and using the scalar product of the projected vector and the vehicle's z-axis (pointing towards the front of the spacecraft), the azimuth angle can be evaluated. The elevation angle is determined using the scalar product of the LOS vector and its projection. Finally, the LOS rates are calculated via the difference of the angles over the previous time step.

The LOS angles representing measured KURS data are calculated similarly to the BZWK values. The only difference is that now instead of the CM's center of gravity, the position of the docking port is used. The latter can be determined via OAPI function *GetDockHandle()*, which returns the position of the docking port in local vessel coordinates.

The KURS data set also contains the pitch and heading bearing angles η_Π and ϑ_Π. A coordinate system is created for the station's docking port and then the line of sight vector is expressed with respect to these coordinates. Afterwards, the pitch and heading bearing angles can be calculated in the same way as the previous angles.

Finally, the range and range rate are also calculated. These values are once again calculated with respect to the CM's center of gravity. After making sure a signal is received from the correct transmitter, the line of sight vector to the station is calculated in local vessel coordinates. The range ρ is simply the length of this vector. The range rate ρ' can be determined using the OAPI function *GetRelativeVel()*, which calculates the relative velocity vector between two vessels. As this vector is with respect to the global reference frame and contains the relative velocity in all three directions, the actual range rate is the result of the scalar product of the relative velocity and the line of sight vector.

4.2.1 "OFF" mode

By default, KURS is in the "OFF" mode. No parameters are calculated and no commands issued. This mode can be acquired while being in any other operating mode. This means that the KURS system can be deactivated throughout the entire rendezvous and docking sequence (see the black dashed line in figure 4.1).

4.2.2 "LONG TEST" mode

Once the system is activated, it automatically goes into the "LONG TEST" mode. During this mode, a long system check of the two KURS systems is simulated by simply staying in this mode for a predefined amount of time ($150\,\mathrm{s}$, [2]) without actually doing anything. For this purpose, the *SIMTIMER* class was developed, which enables the system to count down simulation time.

The "LONG TEST" mode is always acquired after the "OFF" mode. Even if the test has already been performed before, and KURS is deactivated while in another mode, the long test will always be performed again upon (re-)activation of the system.

4.2.3 "SEARCH" mode

In the "SEARCH" mode, the KURS model checks whether the OM's primary navigation device receives a signal from the ISS XPDR. By default, each vessel in the Orbiter simulator has two built-in navigation devices. Besides, the station's XPDR frequency can be set in the scenario configuration file (see [9] for more details on how to edit scenarios). The KURS model is programmed such that by default, the primary navigation device is already tuned to the ISS XPDR frequency. However, its frequency can still be modified by the pilot.

The transmitter strength in the Orbiter simulator is modelled such that it drops off with the square of distance to the transmitter [11],

$$S = S_0/r^2, \tag{4.1}$$

where S is the received signal strength at the distance r. S_0 is set by Orbiter to a value such that a receiver will detect a signal strength of 1 when it is just in range of the transmitter.

The signal range of the station's XPDR predefined by Orbiter is longer than the range of the real system. Therefore, the implemented model not only checks whether an XPDR signal is received, but also if it is above the signal strength S_{real} that would be received at the real system's range r_{real}. Both can be done using the OAPI functions *GetNavSource()* and *oapiGetNavSignal()*. To prevent the model from switching back and forth when the received signal is just around the threshold, a hysteresis factor is implemented. Thus, a signal is processed as "received" when the signal strength is slightly above S_{real}. On the other hand, when a signal is currently received, it will be processed as "lost" only when the signal strength is slightly below S_{real}.

Summing up, this means for the "SEARCH" mode: If a signal is received and its strength is above the required threshold, the KURS model switches to the "SNC" ("Target Acquired") mode. If the KURS model is in any later operating mode, and the signal strength drops below the threshold or is lost completely, it always returns to the "SEARCH" mode (see red lines in figure 4.1).

4.2.4 "SNC" mode

The "SNC" mode starts as soon as the received XPDR signal is strong enough, i.e. the vehicle is in the range of the transmitter. This condition is checked for every time step while in "SNC" mode. During this mode, the vehicle is rotated until it points towards the received signal source. This is achieved using the SUD model (its autopilot, respectively), which is part of the oapiExt, the extended OAPI developed at IRS.

First, the SUD model is passed the target vessel (the station) using the function *SetTargetObject()*. SUD also requests a docking port, but at this point in the rendezvous phase, no docking port has been selected yet. However, the function provides an extra option for this flight phase, in which the center of gravity of the target vessel is chosen instead of a physical docking port. Next, a command is issued to the SUD implementation to rotate the vehicle until it is pointed at the selected docking port (in this case the center of gravity). At the same time, as a secondary constraint, SUD is to maintain the vehicle's rotation attitude, thus keep its y-axis pointing upwards.

For every time step, as always, the KURS model evaluates the current azimuth and elevation angles and rates of the line of sight. As soon as both azimuth and elevation angle are below $5°$, which means the vehicle is aligned with the station, KURS switches to the next mode: "LOCK-ON".

4.2.5 “LOCK-ON” mode

During the “LOCK-ON” mode, the Soyuz continues its flight towards the station. In reality, this is usually the mode in which the correction and the second burn take place. As mentioned at the beginning of this chapter, these burns are neither computed nor commanded by the KURS system and therefore are not implemented within the scope of the present thesis. On the other hand, however, this allows the KURS model in the simulator to operate fully independent of whether these burns are performed manually or not. The only thing that matters to KURS is the attitude of the vehicle and its distance/closing speed with respect to the target, but not the way this position was acquired.

Just as in the previous modes, the KURS model checks whether an XPDR signal is received at a sufficient strength before every time step or else returns to the “SEARCH” mode. Additionally, while in the “LOCK-ON” mode, KURS also checks whether the line of sight angles are still below the threshold of $5°$. If this condition fails, KURS returns to the “SNC” mode (see the yellow lines in figure 4.1). As the SUD hasn’t received any commands otherwise and “LOCK-ON” can only be reached from the “SNC” mode, SUD will keep the Soyuz pointed at the station’s center of gravity while getting closer to it.

At a relative distance of $15\,\mathrm{km}$, the KURS model switches from “LOCK-ON” to “SHORT TEST” mode. After the test is completed, the system returns to “LOCK-ON” and proceeds its approach to the station. Once the Soyuz is within $400\ \mathrm{m}$ of the station, the “APPROACH” mode is acquired.

4.2.6 “SHORT TEST” mode

The short test is performed very similarly to the long test, except that it is, as in the name, shorter. Just as in the long test, the KURS model uses the *SIMTIMER* class to count down the duration of the test ($75\,\mathrm{s}$, [2]). This test will be performed again, should the KURS model have to switch back to the “SEARCH” mode in case the transmitter signal is lost. While the test is performed, no KURS data is generated and therefore not available for the crew to monitor.

4.2.7 “APPROACH” mode

The “APPROACH” mode is acquired once the vehicle is within $400\ \mathrm{m}$ of the target. The start of this mode also marks the beginning of the rendezvous near phase. Each time step, as in the previous mode, both the received signal strength and the line of sight attitude are verified and, if necessary, the KURS model switches back to “SEARCH” or “SNC”, respectively.

During the “APPROACH” mode, the vehicle slowly acquires a station keeping position facing the XPDR antenna mounted on the solar array of the Zvesda module at a distance of $150\ \mathrm{m}$. This is achieved by using the motion control system again. First, the maximum

relative velocity is set to $2\ \mathrm{m/s}$ in all directions. Second, SUD is instructed to acquire the station keeping position.

```
AP_COMMAND cmd;
cmd.commandClass = APCMD_TLIMIT;
cmd.local = _V(2.0, 2.0, 2.0);
pAP->ExecuteCommand(&cmd);

cmd.commandClass = APCMD_TPOS;
cmd.local = vRefApproach;
pAP->ExecuteCommand(&cmd);
```

The exact location of the XPDR signal source on the ISS is not known in the Orbiter simulator. Therefore, the coordinates of this position are predefined in the KURS model (*vRefApproach*) with respect to the center of gravity of the target and so far only represent the correct antenna position when docking to the ISS. When approaching another space station of course, where the KURS antenna is positioned somewhere else, these coordinates might be different. During the whole process, the motion control system keeps the vehicle pointed at the station's center of gravity.

At some point during the "APPROACH" mode, the KURS model expects a docking port selection from the user. As soon as a port has been selected, the system switches to the "FLYAROUND" mode, even if the station keeping position in front of the solar array has not been reached yet.

4.2.8 "FLYAROUND" mode

During the "FLYAROUND" mode the vehicle is moved from its station keeping position in front of the XPDR antenna to a station keeping position in front of the selected docking port at a distance of $150\ \mathrm{m}$. As soon as a docking port is selected, the OM's primary navigation device is tuned to the IDS signal of the respective port instead of the XPDR frequency of the station. The KURS model checks whether an IDS signal is recieved and, analogous to the XPDR signal, the signal strength is verified. Just as with the XPDR signal, a hysteresis has been implemented in order to prevent the system of switching back and forth when the vehicle is on the edge of being in range of the transmitter.

Next, the selected docking port is delivered to the implemented motion control system as the target docking port instead of the center of gravity of the station. The KURS model then commands the SUD model to acquire the station keeping position $150\ \mathrm{m}$ away from the docking port. The maximum relative velocity in any direction during the flyaround is set to $1.5\ \mathrm{m/s}$ (*FlyArSpeed*). As an additional constraint, the KURS model commands SUD to align the the vehicle's x-axis with that of the selected port.

```
AP_COMMAND cmd;
cmd.commandClass = APCMD_TPOS;
cmd.local = _V(0.0,0.0,150.0);
pAP->ExecuteCommand(&cmd);
```

```
cmd.commandClass = APCMD_TLIMIT;
cmd.local = _V(FlyArSpeed, FlyArSpeed, FlyArSpeed);
pAP->ExecuteCommand(&cmd);

cmd.commandClass = APCMD_CONSTRAINT;
cmd.local = _V(-1.0, 0.0, 0.0);
cmd.target = APDIR_REFY;
pAP->ExecuteCommand(&cmd);
```

Now that a docking port has been selected, the line of sight is considered to go from the Soyuz vehicle to the docking port instead of the center of gravity of the station. Hence, during the entire maneuver, the vehicle is pointed now to the docking port instead of the center of gravity.

In order to determine the completion of the flyaround, the remaining misalignment and the remaining relative velocity are measured. First, both the current line of sight vector and the approach vector of the docking port are determined. The line of sight vector is the difference between the position of the vehicle and the docking port in global coordinates. The approach direction can directly be accessed through the function *GetDockParams()* (which is part of the Orbiter API). Then, the cross product of both of these vectors is calculated. If the vehicle was perfectly aligned with the docking port, the result would be zero. However, there are several control loops involved in the simulator, which will always lead to a residual error. Therefore, in the implemented model, the flyaround is considered complete when the result of the cross product is less than 0.3 and the relative velocity is less than $0.3\,\mathrm{m/s}$ in all directions. Finally, KURS waits for a user command to switch into the “FINAL APPROACH” mode.

4.2.9 “FINAL APPROACH” mode

In the “FINAL APPROACH” mode, the received signal strength of the IDS antenna is verified in every time step. The implemented KURS model commands the SUD implementation to stay aligned with the docking port and adds the constraint that vehicle’s rotation should also be aligned with the target. Then, the maximum relative velocity is set to $0.15\,\mathrm{m/s}$ in all directions ([1]). Finally, the command is issued to acquire the docked position, or basically to move the vehicle to the origin of the port’s coordinate system.

```
AP_COMMAND cmd;
cmd.commandClass = APCMD_ALIGN;
cmd.local = _V(0.0, 0.0, -1.0);
cmd.target = APDIR_REFZ;
pAP->ExecuteCommand(&cmd);

cmd.commandClass = APCMD_CONSTRAINT;
cmd.local = _V(-1.0, 0.0, 0.0);
cmd.target = APDIR_REFY;
```

```
pAP->ExecuteCommand(&cmd);

cmd.commandClass = APCMD_TLIMIT;
cmd.local = _V(0.15, 0.15, 0.15);
pAP->ExecuteCommand(&cmd);

cmd.commandClass = APCMD_TPOS;
cmd.local = _V(0.0, 0.0, 0.0);
pAP->ExecuteCommand(&cmd);
```

4.3 KURS front end

A front end is an interface between the user and the back end. In the IRS Soyuz simulator, the front end consists of the control panel, the periscope screen and the two control sticks. Concerning the implemented KURS model, however, only the MFDs are part of the front end. The student pilot can interact with KURS through a series of different views. In the real Soyuz spacecraft, these views are called “format”. The implemented views in the simulator are not identical to the formats, but try to find a balance in the issue described at the beginning of this chapter about how realistic versus how simple the displayed data should be.

In the Orbiter simulator, all MFD views belong to the *VIEW* class, and for each system (KDU, SUD, KURS, etc.) a derived subclass is constructed. Each view can have subviews, which also belong to the *VIEW* class. Figure 4.3 shows the super- and subordinate views of the KURS view. It should be noted, that one view can actually consist of different displays. That is, even though there is only a single KURS view, different information and data might be shown on the screen depending on the current KURS operating mode. The latter is always displayed in the top center of the screen when in the KURS view. There is one main view, as shown in figure 4.4a, through which all implemented instruments of the Orbiter simulator can be accessed. This is also the view the MFD turns to when pressing the “SEL” button in the bottom center. After selecting “Soyuz Systems”, the user will be able to choose one of the implemented Soyuz systems. This view is shown in figure 4.4b. In order to be consistent with the previous section on the KURS back end, the different displays will be explained in chronological order in the following.

OFF. While KURS is still turned off, selecting the KURS view from the main menu leads to the screen shown in figure 4.5a. There are only two buttons carrying a label: “ON” and “<–”. The latter is a built-in default button and is inherited from the “VIEW” class. It will always lead the user to the superordinate view, e.g. when pressing this button while figure 4.5a is displayed, it will lead back to the main view. The “ON” button will activate the (modelled) KURS system. During all other modes, the same button will be labelled “OFF” and will deactivate the KURS system at any given point.

LONG TEST. During the long system test, the pilot only receives information about how far the test has proceeded so far via a bar. The only way he can interact with the system

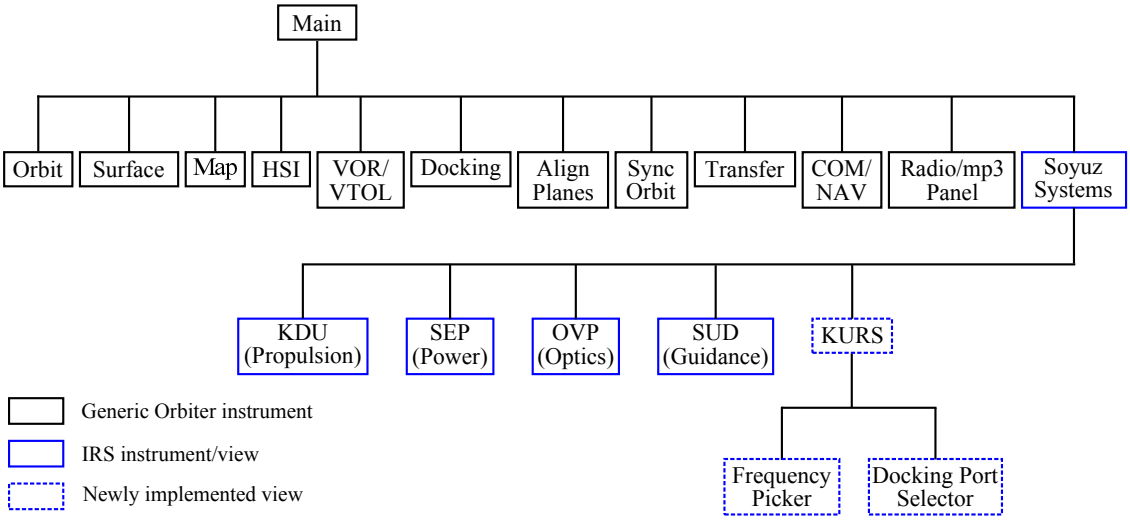
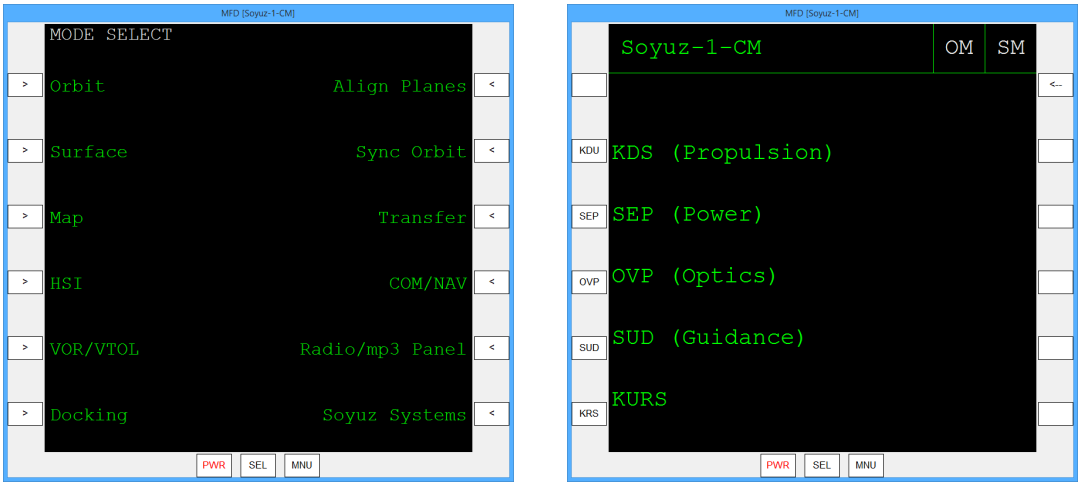
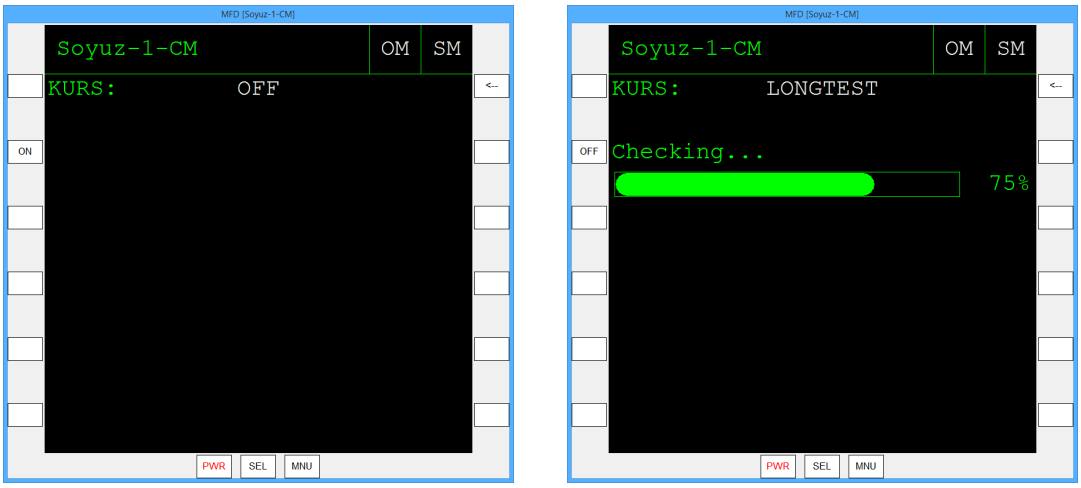


Figure 4.3. View structure.



(a) **(b)**

Figure 4.4. (a) Main view and (b) Soyuz systems view of MFD.



(a) **(b)**

Figure 4.5. (a) Off display and (b) long test display of KURS.

is to turn it off using the “OFF” button. Otherwise, no data is displayed. An example is shown in figure 4.5b.

SEARCH. While the KURS model is in “SEARCH” mode, the user can see the current frequency of the navigation device and how good the reception of the signal is. The green bar visualizes the received signal strength. If it is above the yellow line, the vehicle is in range of the transmitter. In case the pilot needs to tune the navigation device to a different frequency, he can press the “FRQ” button in order to reach the frequency picker subview depicted in figure 4.6b. In this view, the current frequency and channel number of the vehicle’s navigation device are displayed. In the Orbiter simulator, each navigation device has channels ranging from 0 to 639. To convert a channel number *ch* into a frequency, use [11]

$$f = (108.0 + 0.05\,ch)\ \text{MHz}. \tag{4.2}$$

The buttons on the left and right can be used to change the channel number and therefore also the frequency. “+++” increases the channel number by 100, “++” by 10, and “+” by 1. The buttons on the left decrease the channel number by the respective amount. Once the desired frequency has been reached, the pilot can use the “<–” button to return to the previous (in this case: search) view.

SNC. In the “SNC” mode, the display is very similar to the one before in the “SEARCH” mode (see figure 4.7a). The current frequency is shown as well as a bar indicating the received signal strength. Likewise, there is a “FRQ” button on the right in case the frequency needs to be changed. Additionally, the pilot now can also monitor the current line of sight pitch and yaw angles. Note that during the SNC mode, the vehicle is aligning with the line of sight and therefore, the pitch and yaw angles should decrease to zero if the KURS model is working properly.

LOCK-ON. During “LOCK-ON”, even more flight data is available to the pilot: In addition to the previous information (frequency, signal strength, LOS pitch and yaw), now

(a) **(b)**

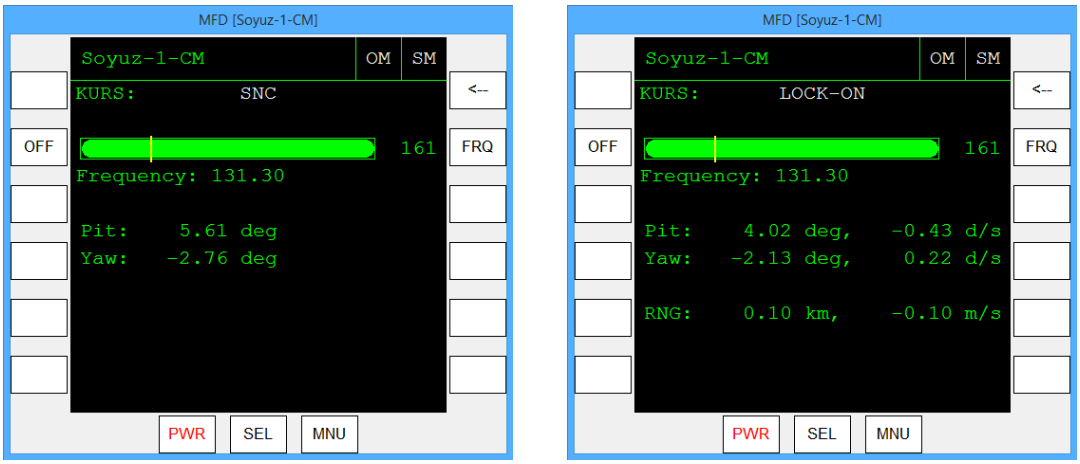
Figure 4.6. (a) Search display and (b) Frequency picker subview of KURS.

also the values for the LOS pitch and yaw rates are available, as well as the range and the range rate. Just as before, the frequency can be adjusted by pressing the “FRQ” button on the right. An example lock-on display is shown in figure 4.7b.

SHORT TEST. As the pilot cannot interact with the KURS system while it is performing the short system test (except to turn it off), the display looks exactly as during the long test at the beginning.

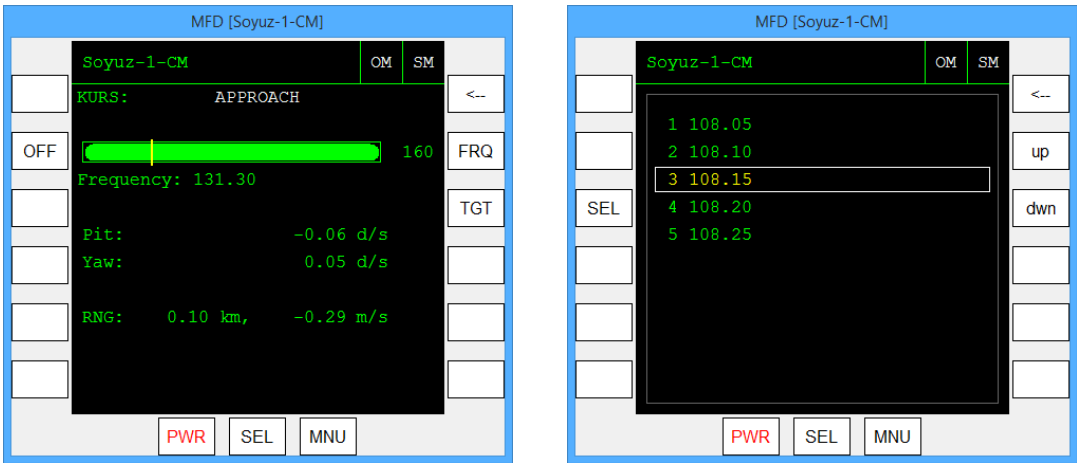
APPROACH. The information available during the “APPROACH” mode is only slightly different from the “LOCK-ON” mode (see figure 4.8a). There are only two differences: First, only the LOS rates are available now, but not the actual values (which should be very small anyway). Second, there is an additional button on the right labelled “TGT”. This button leads the docking port selector. Here, all docking port numbers and their corresponding IDS signal frequency of the target vessel are displayed. Figure 4.8b shows a list of the available docking ports of the ISS and figure 4.9 shows a screenshot of the implemented ISS model with the respective docking ports. With the “up” and “dwn” buttons on the right, the pilot can switch between ports. The “SEL” button on the left selects the port currently displayed in yellow. Upon selection, the view automatically switches to the next display of the KURS view. The docking port selection can be performed any time during the approach, even before the station keeping position is acquired. The Soyuz will then immediately start the flyaround.

FLYAROUND. While the vehicle is performing the flyaround to the selected docking port, the pilot can monitor the line of sight angles, which now also inlcude the measurement of the roll misalignment. Note that now the line of sight has changed and extends from the Soyuz docking port to the selected docking port on the station. Additionally, the bearing angles are displayed. These are the LOS angles as seen from the station. In other words, the LOS angles measured by Soyuz represent the orientation of the vehicle in reference to the selected station docking port, whereas the bearing angles represent the translational position with respect to the latter. Just like before, the range and range rate



(a) **(b)**

Figure 4.7. (a) SNC display and (b) lock-on display of KURS.



(a) **(b)**

Figure 4.8. (a) Approach display and (b) docking port subview of KURS.

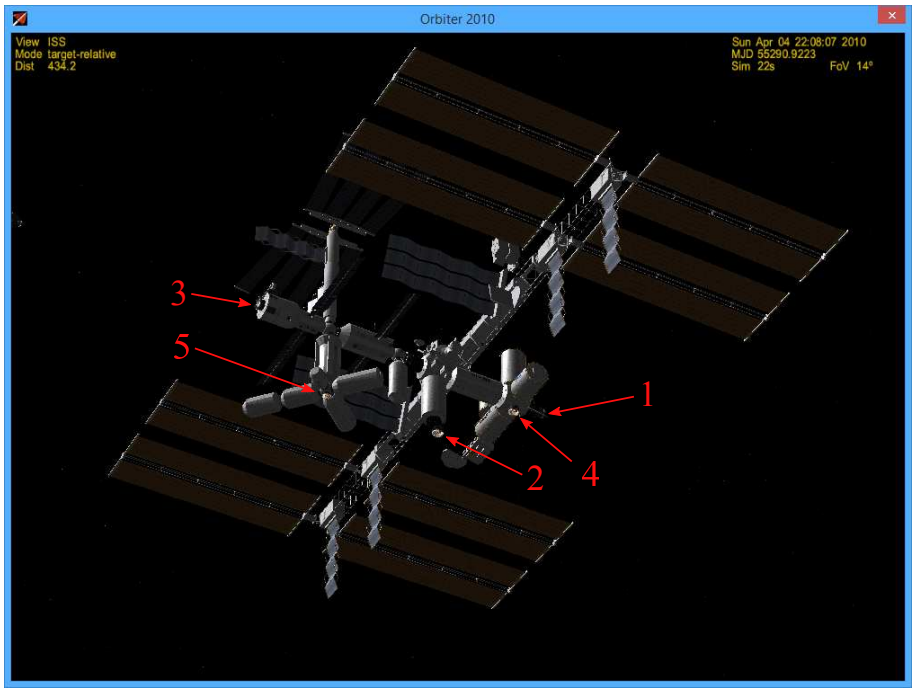
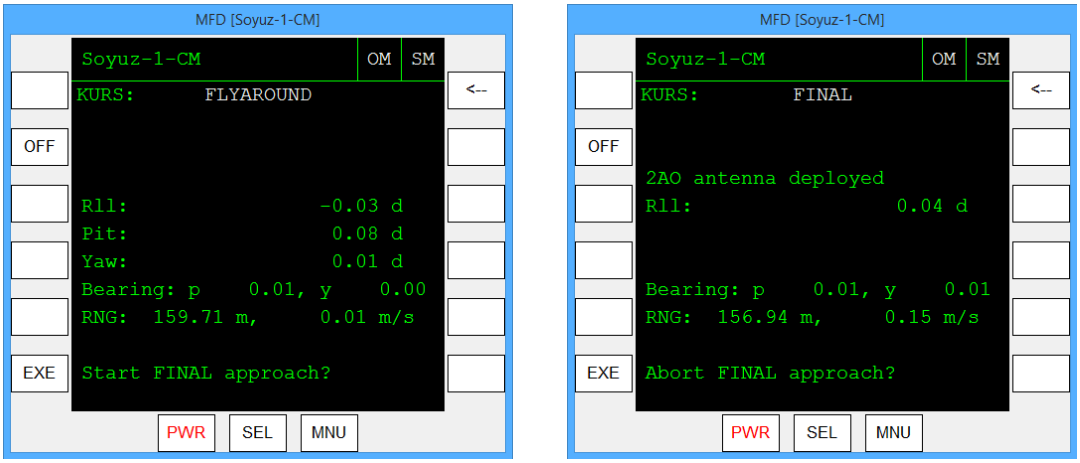


Figure 4.9. Docking ports on the implemented ISS model.

are also displayed. As the vehicle is now fairly close to the station, there is no need to display the current frequency or received signal strength of the navigation device as the Soyuz should not be moving out of its range. Once the station keeping position in front of the docking port has been acquired, an additional line appears on the display saying “Start FINAL approach?”. Figure 4.10a shows the display in this configuration. By pressing the “EXE” button, the pilot commands the Soyuz to start the final approach to the docking port.

FINAL. During the final approach to the docking port, fewer relative motion parameters are available to the pilot than before (see figure 4.10b). Only the roll misalignment is displayed, as well as the bearing angles, the range and range rate. Additionally, the pilot receives information about whether the 2AO antenna boom has been retracted or not. The retraction is usually takes place automatically. Note that docking with the antenna still deployed is prohibited. In the bottom of the display it now says “Abort FINAL approach?”. In case the pilot notices a malfunction, pressing the button “EXE” will abort the final approach and put the KURS system back into the “FLYAROUND” mode. In this way, the motion of the vehicle will be stopped and it will be directed to move back to the station keeping position at $150\,\mathrm{m}$ distance. The only other option for the pilot to interact with the KURS model at this point is to turn it off.



(a) **(b)**

Figure 4.10. (a) Flyaround display and (b) Final approach display of KURS.

4.4 Procedures

In order to increase the authenticity of the IRS Soyuz simulator, flight procedures are developed for the modelled systems. The procedure are used in reality to guide the astronauts through the many highly complex systems of the Soyuz spacecraft. In the simulator, so far no procedures are available to the students. Within the present thesis, an existing procedures draft was enhanced and especially the procedures for an instrument based rendezvous were developed. The simulator procedures try to reproduce the actual ones as close to reality as possible. However, procedures always need to be adapted to those systems, that the training pilot is actually working with. Including many items in the procedures that do not exist in the simulator only confuses the users and contradicts the point of procedures in general, which is to help the pilot with the system usage. The developed simulator procedures draft can be found on the DVD accompanying this thesis.

Chapter 5

Results: Comparison of reality and model

5.1 General remarks

This chapter provides a comparison of the implemented KURS model and the real Soyuz system. The comparison covers a variety of different aspects from the hardware modelling to the differences in the resulting rendezvous sequence up to the differences in the front end, i.e. the MFD simulator views versus the Russian formats. Most of the differences can be related back to the general idea that aerospace engineering students will be using the simulator, and not astronauts training for a real mission. This often leads to simplifications, but can result in some additional features as well, as described in the following.

One major issue, that is not implemented in the IRS simulator in general, is system malfunctions. Of course, during real astronaut training, preparing for malfunctions is what most of the training is about. However, in the IRS simulator, the training is more about getting to know the system in general, without worrying about malfunctions.

In addition, the present project only implemented that part of the automated docking process that is provided by the KURS radar system and takes place in proximity of the station. However, what is beyond the scope of this thesis is that part of the motion control system calculating and executing the required rendezvous and correction burns to get the Soyuz vehicle from its phasing orbit to the station. At this point of time, a fully automated rendezvous and docking is not yet possible. Only when the vehicle is within $400\,\mathrm{m}$ of the station, the implemented motion control system takes over and docks the vehicle automatically. Approaching the station up to this distance so far has to be achieved manually.

5.2 KURS antennas and electronics

In the design process of the hardware component models, not all components are modelled as separate objects in the implemented KURS system. First of all, the different KURS antennas on the Soyuz vehicle are not each modelled as separate objects. Moreover, only a single built-in navigation device of the Orbiter simulator is used. No differentiation is made between which antenna currently receives a signal or measures a relative motion parameter. Instead, the implemented model uses the built-in navigation device to receive a signal in general. Besides, as this received signal does not contain any information about the relative motion parameters, the latter are determined using built-in functions of the Orbiter API.

The modelling of the different KURS antennas on the Soyuz vehicle is omitted for several reasons: As mentioned in chapter 3.1, one principle of object-oriented programming is to create objects that resemble reality only as far as necessary. It is not the primary goal for the aerospace students training in the simulator to learn about radio signalling, but about the rendezvous flight path and the operation of a complex vehicle. One argument supporting the modelling of different antennas is that this would allow for the simulation of malfunctions of single antennas. However, as mentioned above, malfunctions are not part of the IRS Soyuz simulator. Therefore, the modelling of separate KURS antennas and their respective received and transmitted signals was considered unnecessary.

In the real Soyuz, the radar system not only consists of several antennas, but also of a set of electronics (filter, receiver, transmitter, etc.). All these electronics were also not modelled in the IRS simulator for very similar reasons as the implementation of separate antennas was omitted. The implemented KURS system itself does not contain any other components in general, but is treated as a single object that measures/calculates most of the things on its own. Besides, the real KURS system consists of two of these electronics sets for redundancy. So far, only one KURS model is implemented in the simulator as a second system would only be interesting for simulating malfunctions. If this is desired at a later point of time, a second KURS system can be implemented by creating another instance of the KURS class and assigning one boolean parameter indicating which model is currently in use and another boolean parameter indicating whether the model is malfunctioning or not.

5.3 KURS operating modes and other software differences

In reality, the KURS system switches directly from the “LOCK-ON” mode to the “FINAL APPROACH” mode. The implemented KURS model has additional operating modes. The “APPROACH” and “FLYAROUND” modes were added at the end of “LOCK-on” and at the beginning of “FINAL APPROACH”. This decision was made for the sake of convenience in order to simplify the implementation of the different flight phases and the corresponding displays.

The two test modes “LONG TEST” and “SHORT TEST” are basically implemented as

timers, where the KURS model just waits for a certain amount of time before proceeding with its operation. In reality, during the long test, both KURS sets are tested and during the short test, the "hot" set is tested again. However, as mentioned above, only one KURS model was implemented and malfunctions are not implemented in general. For these reasons the implemented model cannot check any parameters and just has to "pause" for a given time. Another possibility would have been to simply ignore these two test modes. Nevertheless, these test modes were still desired in the implementation in order to give the students a more realistic experience for example when there is no KURS data available during the short test while being fairly close ($15\,\mathrm{km}$) to the station.

Another, if yet smaller, difference between the mode implementation of the model and the real system occurs at the end of the "APPROACH" mode. In reality, the vehicle is aligned with the station's axis in its station keeping position. In the implementation however, the vehicle acquires the same position but is pointed to the station's center of gravity. This is due to the limitations of the IRS motion control system implementation(its autopilot, respectively).

In reality, the rendezvous far phase begins at T_0 with the onboard computer integrating the equations of motions to determine the current state vector of the vehicle. This integration leads to some errors in the propagated vector later on, which are corrected during the "LOCK-ON" phase using a Kalman filter. As the simulator does not propagate the state vector, but calculates it again every time step, there is no need to correct the state vector. Therefore, no Kalman filter model is implemented in the simulator.

5.4 Crew operations

Most of the differences in crew operations either result from the fact that aerospace engineering students will be operating the simulator (instead of real astronauts). Another great deal of variations are a consequence of another, previous, difference (e.g. in the determination of the relative motion paramters). First, the differences in the crew actions are described, later the MFD "views" are compared to the equivalent "formats".

After the KURS system is started and tested, it will look for a signal from the station's KURS antennas. In reality, the KURS antennas on the Soyuz are tuned to pre-set frequencies and the crew cannot change them. In the simulator, a frequency picker view has been implemented. This allows the crew to vary the frequency of the built-in navigation device of the vehicle. In this way, the students can get a better idea of the operation of the radio system.

Later on, during the near phase of the rendezvous, the crew has to select a docking port in the simulator. In reality, the crew does not select the docking port, yet they do know which port they are supposed to dock to. The actual selection of the docking port is made by ground control ([1]). However, to make the simulator students independent of ground control, the docking port selection was implemented as a crew task.

5.4.1 Displays

In order to differentiate between reality and model, the different MFD displays are called either “format” in the real system or “view” and “display” in the IRS simulation. In the real system, the selection of the different formats can be done both automatically by the BZWK or manually by the crew. There are two main formats the crew uses during rendezvous and docking procedures, which can be seen in figure 5.1.

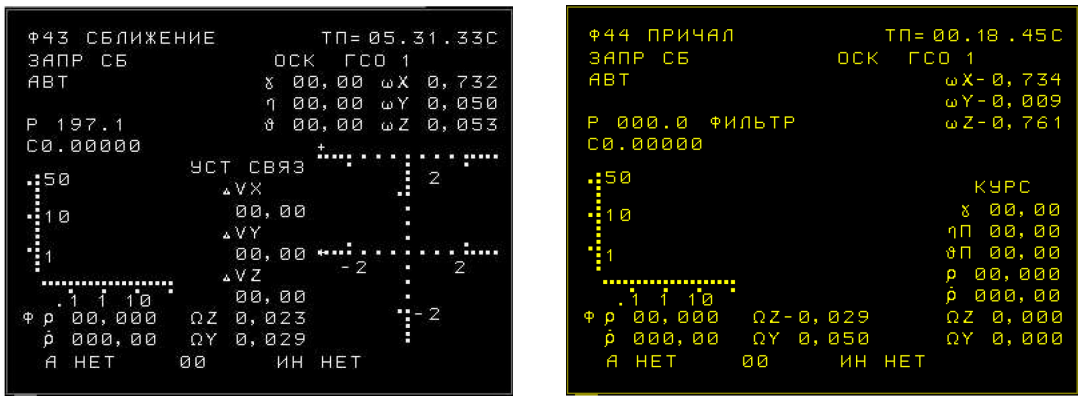
Figure 5.1a shows the “rendezvous” format Φ43, which is used for most of the rendezvous phase. In the top right corner, the roll, yaw and pitch angle (γ, η, ϑ) measured by KURS are displayed (if available), as well as the respective angular rates (ω_x, ω_y, ω_z). In the center, the components of the next required burn are displayed with respect to the vehicle’s center of mass (ΔVX, ΔVY, ΔVZ). In the bottom left, the relative range, the range rate (ρ, ρ') and the LOS angular rates (ΩZ, ΩY) are displayed.

So far, these displayed values are similar to the ones implemented in the KURS view. It should be noted, that not all of these values are available in the real system on any occasion. The roll angle, for example, is not measured by KURS during the entire rendezvous sequence. In reality, of course only those values are displayed, that were actually measured/calculated. In the implementation, the calculation of these values is possible throughout the rendezvous phase. Therefore, the above mentioned additional operating modes (“APPROACH” and “FLYAROUND”) were introduced. These extra modes makes it easier to distinguish between parameters that are currently available and those that are not.

The main difference between this format and the simulator implementation is the lack of diagrams in the simulator. The diagram on the left depicts the relationship of ρ to ρ'. along the horizontal axis, the relative range is plotted in logarithmic scale and along the vertical axis, the rendezvous range rate is plotted. On the right hand side, there is an indicator for the line of sight angular position relative to the vehicle’s local coordinate system. Along the upper horizontal axis, the mutual roll angle is displayed, on the lower horizontal axis the LOS yaw deviation and along the vertical axis the LOS pitch deviation. The implementation of these charts was mainly omitted, because the view model in the IRS simulator cannot display that many characters. However, a similar chart as the right one displaying the LOS angles can be reached through the built-in “Docking” instrument from the main view (yet this docking view does not display the data of the KURS model).

Figure 5.1b shows the “Final Approach” format Φ44. The crew always has to switch to this format manually, as the borderline between rendezvous and final approach is not always obvious. The displayed data is very similar to the one in the previous format Φ43. The major differences are that there are no more ΔV parameters in the center and that the right chart displaying the LOS angles has disappeared as well. The KURS angles that used to be displayed at the top right have disappeared, too. Instead, additional KURS data is now displayed on the right hand side. What is new is the display of the bearing angles η_Π and ϑ_Π, the range and range rate ρ and ρ', and the LOS rates ΩZ and ΩY. The pilots now have the possibility to compare the calculated relative motion paramters (in the bottom left) to the ones provided by KURS (in the lower right).

As the implemented KURS model does not differentiate between data received from



(a) **(b)**

Figure 5.1. (a) Format Φ43 “Rendezvous”; (b) format Φ44 ”Final Approach“; both from [4].

KURS antennas and data calculated by the onboard computer, there is no need to display the range, the range rate and the LOS angular rates twice in the simulator.

Chapter 6

Summary, conclusions and outlook

6.1 Summary and conclusions

The goal of this thesis was to implement a realistic model of the Soyuz spacecraft's radar and docking systems in the IRS Soyuz simulator. In order to obtain a deeper insight into the real systems, the present study was carried out in collaboration with EAC/ESA, where the author spent four weeks at their facilities in Cologne.

When modelling real systems, it is always important to find a well-balanced solution to attain a simple, yet realistic implementation. Usually, the implementation starts off as a rough representation of reality and gets more realistic and refined over multiple development cycles. Very often, decisions have to be made on prioritizing some systems (or their components) over others.

Within the present thesis, these decisions included for example the modelling of the KURS antennas. A separate class was initialized for the KURS system itself, but its single antennas were not implemented as extra objects.

The IRS simulator holds an additional concept, which has to be kept in mind when developing models. That is the fact that the simulator is not intended to train and prepare astronauts for an actual space mission. Instead, it aims at giving aerospace students an opportunity to get a first-hand impression of what flying a spacecraft feels like (except without microgravity, of course). In this way, the students have the possibility to receive a practical experience for example of the consequences of a diminishing drag while performing attitude control. In addition, they also get a deeper insight in orbital mechanics when performing rendezvous maneuvers.

Taking the above into account did not just result for example in leaving out malfunctions in order to keep the vehicle handling simple. Indeed, some extra features were added to the implemented model for the training to-be engineers. It is possible in the IRS simulator to select a docking port before performing the flyaround in proximity of the station. In reality, the docking port is preselected by ground control. When choosing the desired docking port in the simulator, not only the docking port number is displayed, but also

its respective antenna frequency. In this way the students see and learn what is behind selecting a docking port: It means setting the vehicle's navigational device to a specific frequency. Following the received signal will then guide the spacecraft to the desired docking port.

When it comes to developing the procedures, in general, the same issues are taken into consideration as when implementing the models. Of course, procedures can only be written for systems respresented in the simulator. In addition, it has to be ensured that all of the procedures are manageable by a single person. In reality, there is a crew of three, but in the simulator, the students usually practice one at a time.

6.2 Outlook

The implementation of such a highly complex system such as the radar and docking system (not even to mention the Soyuz vehicle in general) always leaves room for extensions and improvements. The implementation model could be refined with regards to modelling each antenna individually or implementing two seperate electronics sets. In general, these kind of enhancements will be most interesting if they also involve the implementation of malfunctions.

One option would also be to implement a model of the onboard computer including its state vector propagation. This propagation could be implemented in the simulator as an actual integration of the state vector. Another (maybe simpler) approach would be to model the propagation by taking the actual correct state vector and randomly tamper with the values to simulate the integration errors. Thus, the relative motion parameters provided from the onboard computer would differ from those provided by the KURS model. In this way, the importance of a Kalman filter could be demonstrated to the students. The Kalman filter itself can be implmented as an algorithm slowly decreasing the scope of how much the correct state vector is tempered with.

An additional feature would be procedures concerning the mechanical docking. Even though the single steps of the mechanical docking process cannot be implemented in the simulator due to software restrictions, the leak checks following the docking, for example, could be modelled and included in the procedures.

In order to be able to perform a fully automated docking, the motion control system and its onboard computer need to be expanded. So far, the rendezvous burns have to be performed manually or can are pre-set in the scenario file. For an automated docking, the motion control system needs to determine the desired interception point and the resulting maneuver time to meet up with the station. The maneuver time is the time the ISS needs from the maneuver starting point to the interception point. From there, the resulting transfer orbits of the bi-elliptic transfer can be calculated.

Finally, special KURS training scenarios should be developed for the Soyuz Rendezvous and Docking Seminar. This would offer the students training environments, in which they

can get to know the KURS system and its operating modes, as well as familiarize themselves with the according procedures. Within the present thesis, only one exemplary scenario was developed to demonstrated the full capabilities of the KURS system. However, the development of such scenarios ranging from the orbit insertion to docking is complicated, as so far there is no automated calculation of the required main engine burns.

References

[1] D. Churkin. Personal communication, 2013. Astronaut Training Division, EAC/ESA.

[2] S. Fasoulas. Personal communication, 2013. Professor at Space Systems Institute.

[3] A. Fink and M. Schmitz. Soyuz rendevous and docking simulator training. University Lecture Notes, 2013. University of Stuttgart.

[4] Gagarin Russian State Scientific Research Testing Center for Cosmonaut Training. Soyuz TMA Crew Operations Manual, 2006.

[5] J. Liberty. *Teach Yourself C++ in 21 days*. Sams Publishing, Indianapolis, Indiana, 2001.

[6] D. Louis. *C/C++ Die praktische Referenz*. Markt+Technik Verlag, Munich, Germany, 2007.

[7] M. Schmitz. RTT Wiki -> Soyuz Simulator. URL http://phobos.irs.uni-stuttgart.de/~wiki/mediawiki/index.php5/Simulator.

[8] M. Schmitz. A simpit framework based on the orbiter space simulator and implementation of a soyuz-style simpit. Studienarbeit, 2010. Sapce Systems Institute, University of Stuttgart.

[9] M. Schweiger. Orbiter Scenario Editor. Orbiter 2010 Edition SDK, 2006.

[10] M. Schweiger. Orbiter Programmer's Guide. Orbiter 2010 Edition SDK, 2009.

[11] M. Schweiger. Orbiter API Reference Manual. Orbiter 2010 Edition SDK, 2010.

[12] M. Schweiger. Orbiter User Manual Spaceflight Simulator. Orbiter 2010 Edition SDK, 2010.

[13] M. Schweiger. Orbiter Spaceflight Simulator, July 2013. URL http://orbit.medphys.ucl.ac.uk/index.html.