AUTOCODING WORKING GROUP

Automatic Code Generation for AOCS Flight SW

DAVIDE ODDENINO

16/10/2018

Special thanks for their valuable contribution to: Carlo Valentini (trainee at ESA) and to the Woking Group ESA members

Outline of the presentation

- Background – WG objectives
- Presentation of ESA Handbook
- Autocoding Process Definition – Proof of equivalence
- Extended WG terms of reference
- Planning
- Conclusions

SAVOIR AUTOCODE
space avionics open interface architecture

WG Objectives

Guidelines for the Automatic Generation of AOCS/GNC Flight SW Handbook

- **Modelling Guidelines**

List of modelling guidelines have been implemented in the draft HB.

They [illegible]

- **Process mapping vs standards**

The full process will be mapped onto the SW development process and AOCS development process as defined in the standards.

The review will be phased according to SW auto-generation steps.

- **Reporting**

The template for automatic reporting generation and the definition of its content from the Matlab toolbox has to be defined and discussed.

RD-01	GMV-AUTOCOGEQ-FR_V1.1	AUTOCOGEQ ((Preparation for the Qualification of Auto-Code Generated from Simulink Models) Final Report
RD-02	ESA-TECSAA-TN-007001	AOCS Flight SW Automatic Code Generation process
RD-03	ESA-TECSAA-TN-011350	AOCS FSW AUTOCODING Verification Process Final Report

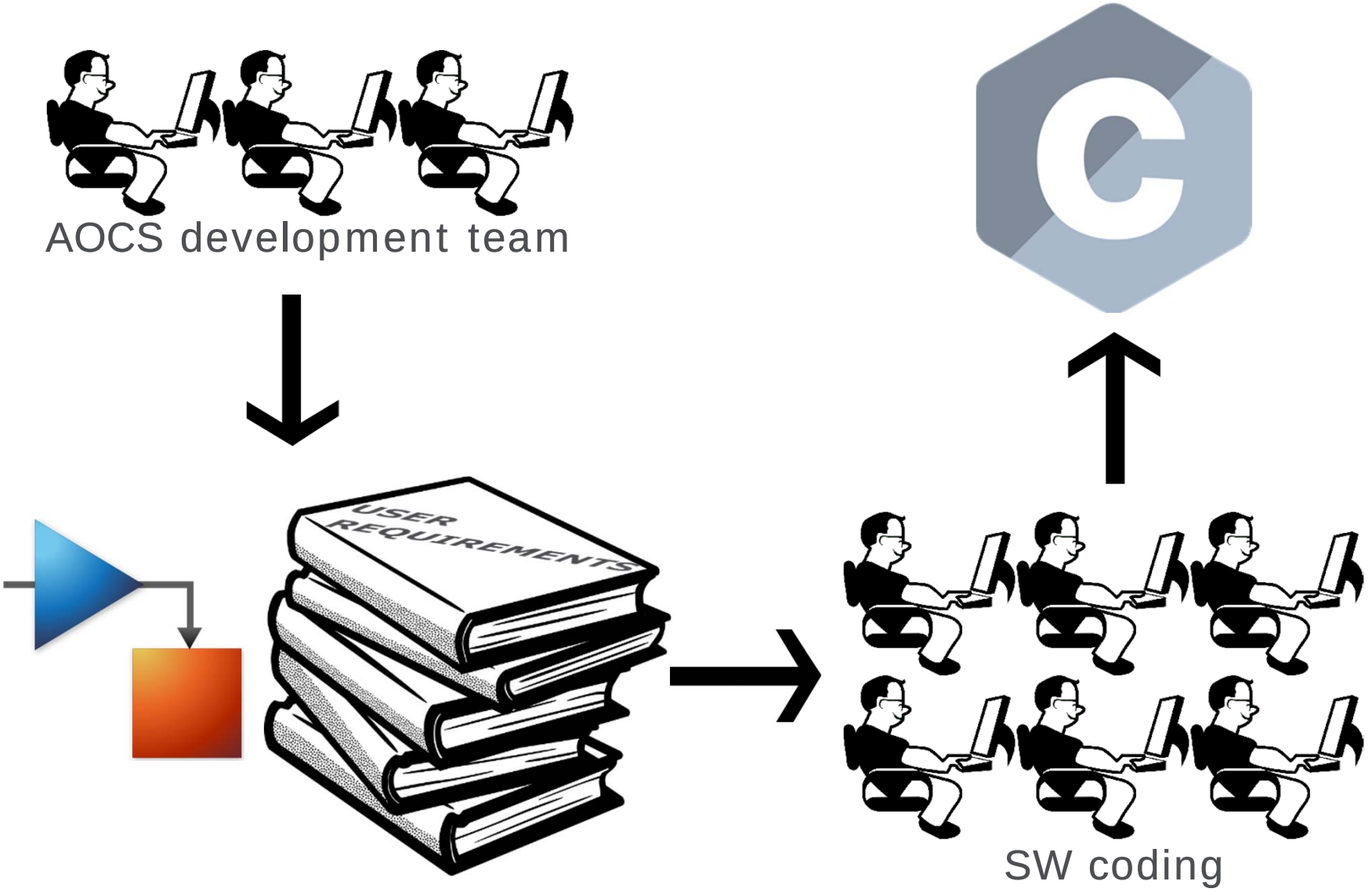
Autocoding Handbook

1. INTRODUCTION
2. APPLICABLE AND REFERENCE DOCUMENTS
3. TERMS, DEFINITION AND ABBREVIATED TERMS
4. INTRODUCTION to AOCS FSW development PROCESS
    a. The classical process: Manual Coding
    b. Introduction to Autocoding
    c. Comparison and key differences
5. MODELLING GUIDELINES FOR CODE GENERATION
    a. Modelling guidelines
        - General modelling guidelines
        - Modelling with Matlab
        - Modelling with Simulink
        - Modelling with Stateflow
    b. Code generation guidelines
        - Coder configuration settings
        - Generated Code structure
        - Reuse of legacy code
6. VERIFICATION & VALIDATION
    a. Automatic model and code verification tools
7. Traceability to ECSS-E-40C, ecss-q-80 FOR AUTOCODING
    a. Traceability to ECSS-E-40C
    b. Traceability to ECSS-Q-80C
8. GENERATION OF REPORTING
    a. Design reports
    b. Test plans
    c. Test reports
    d. Verification control documents
9. CONCLUSIONS
10. ANNEX A: EXAMPLES
    a. Project Examples
    b. Matlab examples
11. ANNEX B: SIMULINK BLOCKS ALLOWED FOR CODE GENERATION

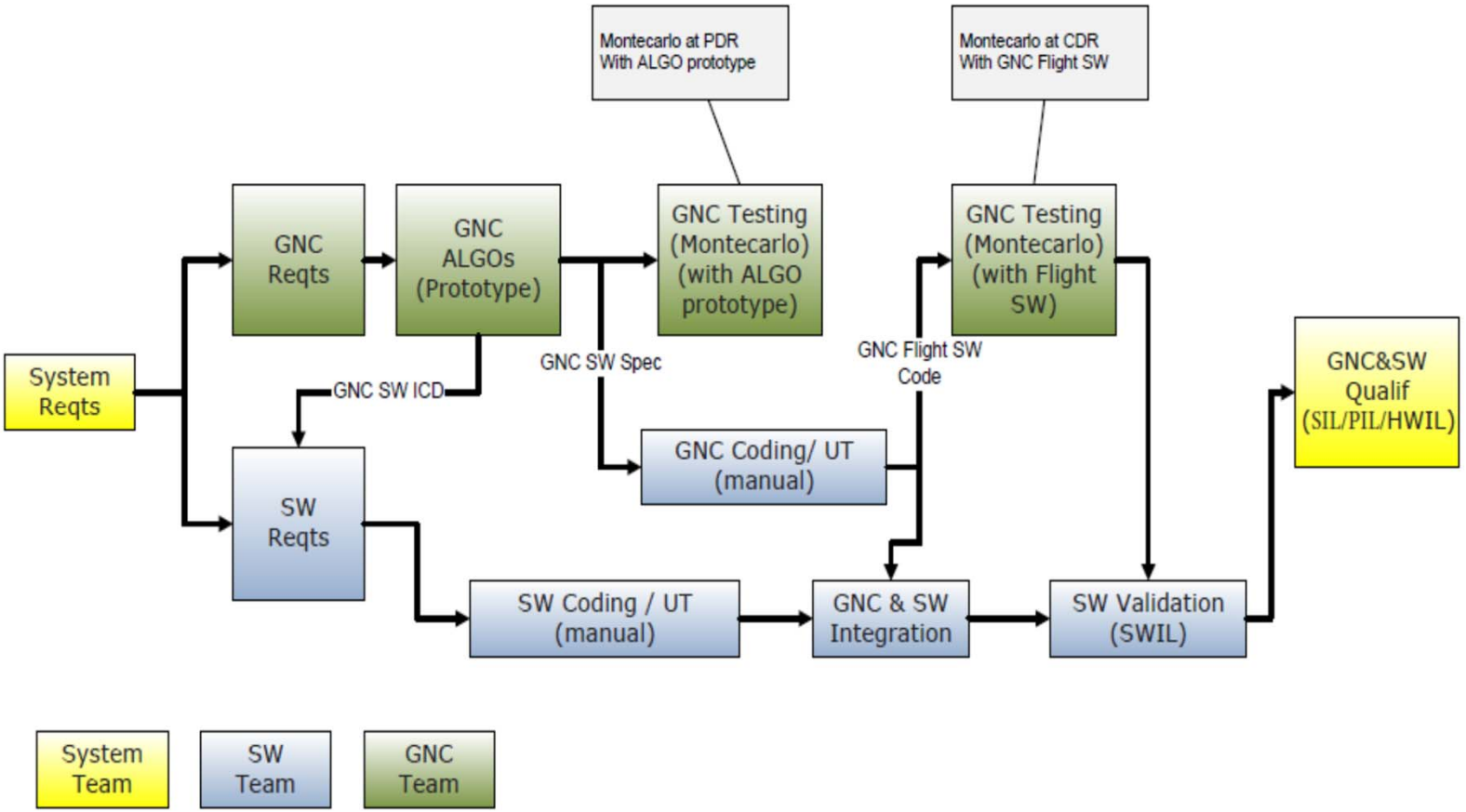
Autocoding Handbook

1. INTRODUCTION
2. APPLICABLE AND REFERENCE DOCUMENTS
3. TERMS, DEFINITION AND ABBREVIATED TERMS
4. **INTRODUCTION to AOCS FSW development PROCESS**
    a. **The classical process: Manual Coding**
    b. **Introduction to Autocoding**
    c. **Comparison and key differences**
5. MODELLING GUIDELINES FOR CODE GENERATION
    a. Modelling guidelines
        - General modelling guidelines
        - Modelling with Matlab
        - Modelling with Simulink
        - Modelling with Stateflow
    b. Code generation guidelines
        - Coder configuration settings
        - Generated Code structure
        - Reuse of legacy code
6. VERIFICATION & VALIDATION
    a. Automatic model and code verification tools
7. Traceability to ECSS-E-40C, ecss-q-80 FOR AUTOCODING
    a. Traceability to ECSS-E-40C
    b. Traceability to ECSS-Q-80C
8. GENERATION OF REPORTING
    a. Design reports
    b. Test plans
    c. Test reports
    d. Verification control documents
9. CONCLUSIONS
10. ANNEX A: EXAMPLES
    a. Project Examples
    b. Matlab examples
11. ANNEX B: SIMULINK BLOCKS ALLOWED FOR CODE GENERATION

Traditional Manual Coding approach of AOCS FSW

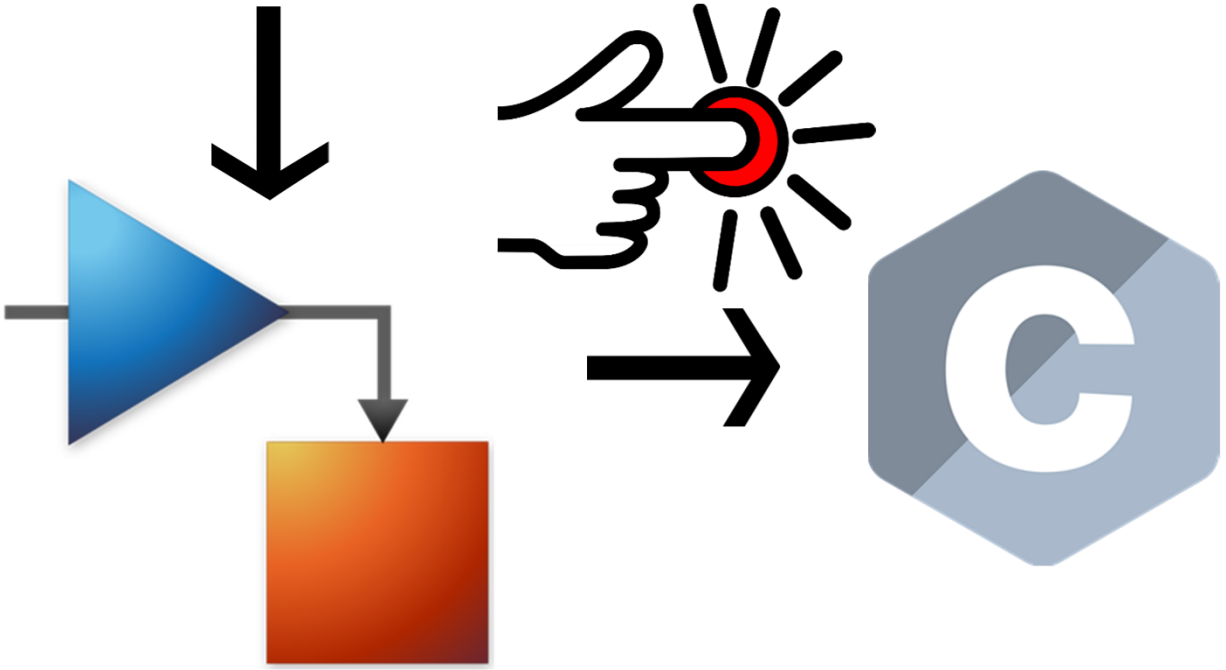


Traditional Manual Coding approach of AOCS FSW

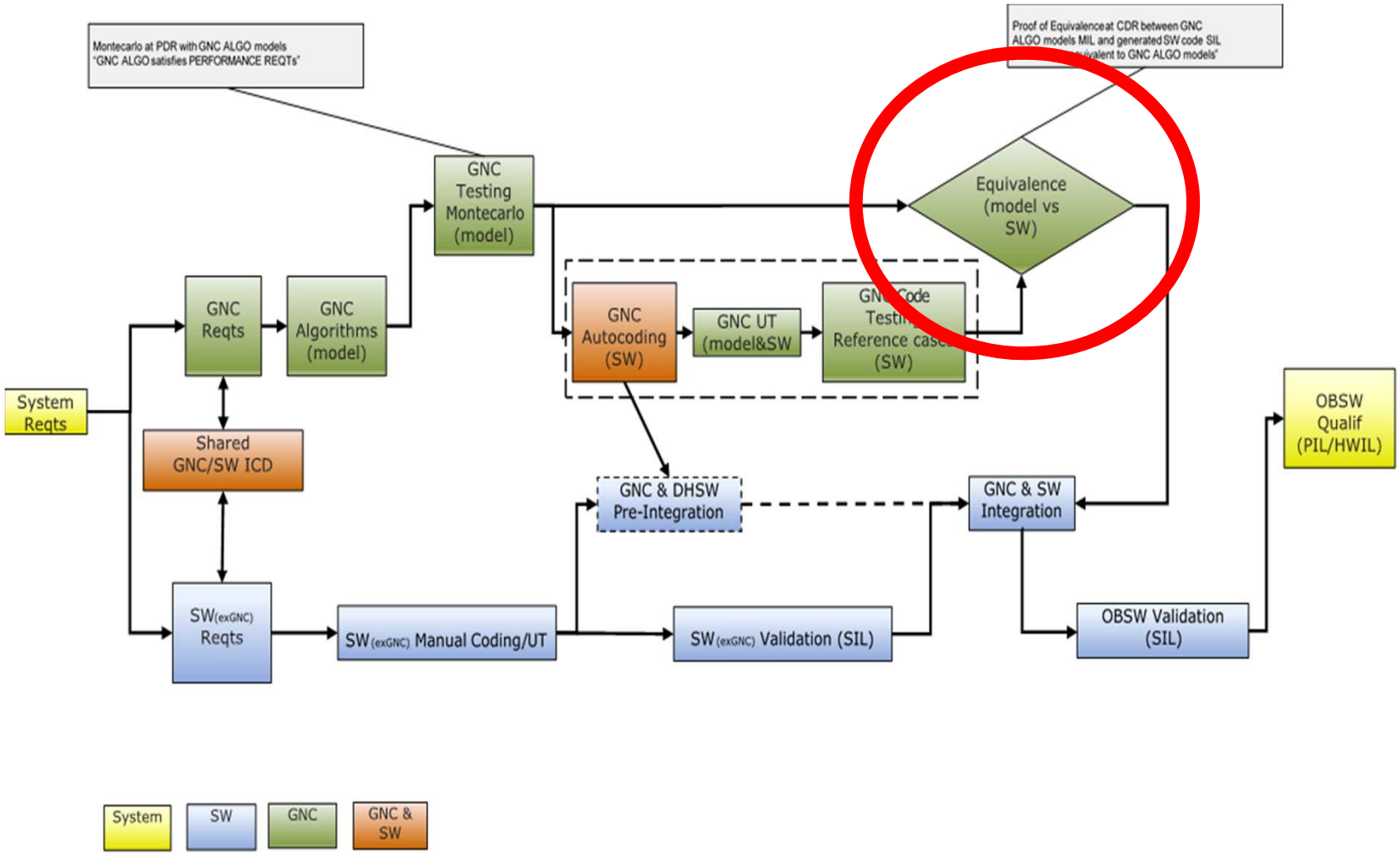


Automatic Code Generation approach of AOCS FSW

AOCS development teams



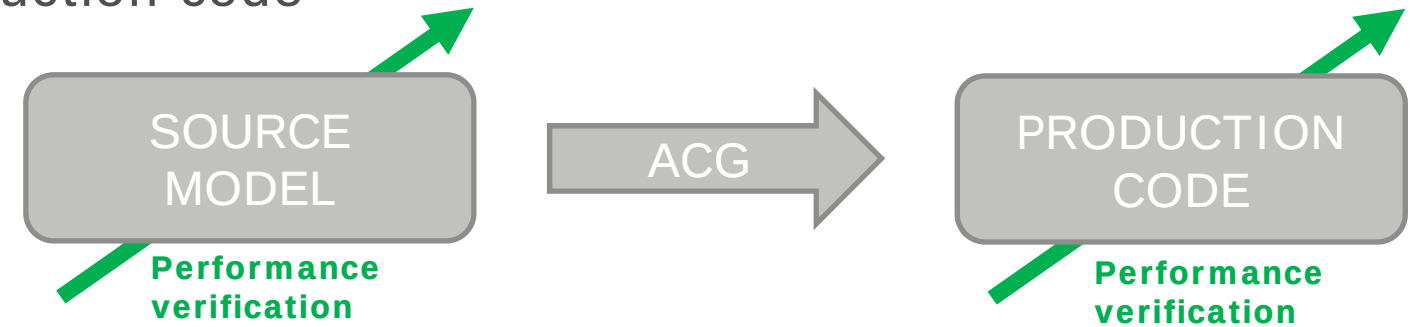
Automatic Code Generation approach of AOCS FSW



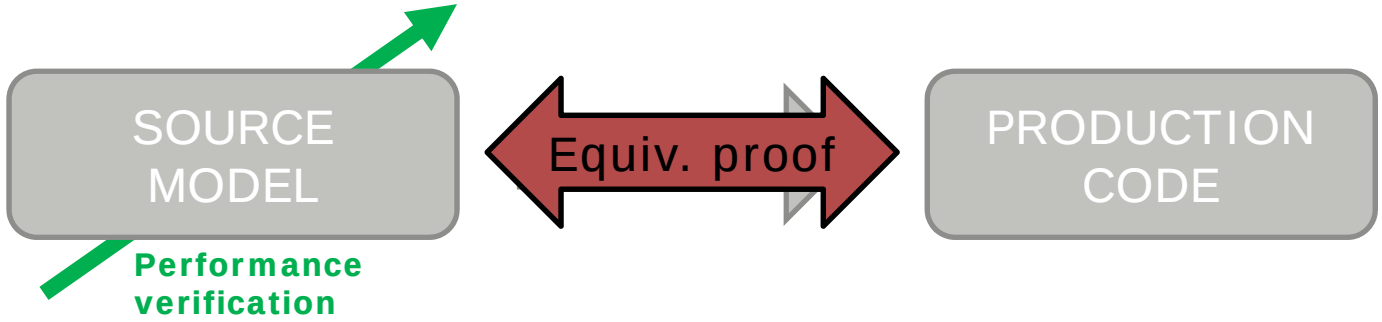
Proof of equivalence

- ~~Certification of the ACG tools~~
- Post-synthesis *certification* of the FSW ✓

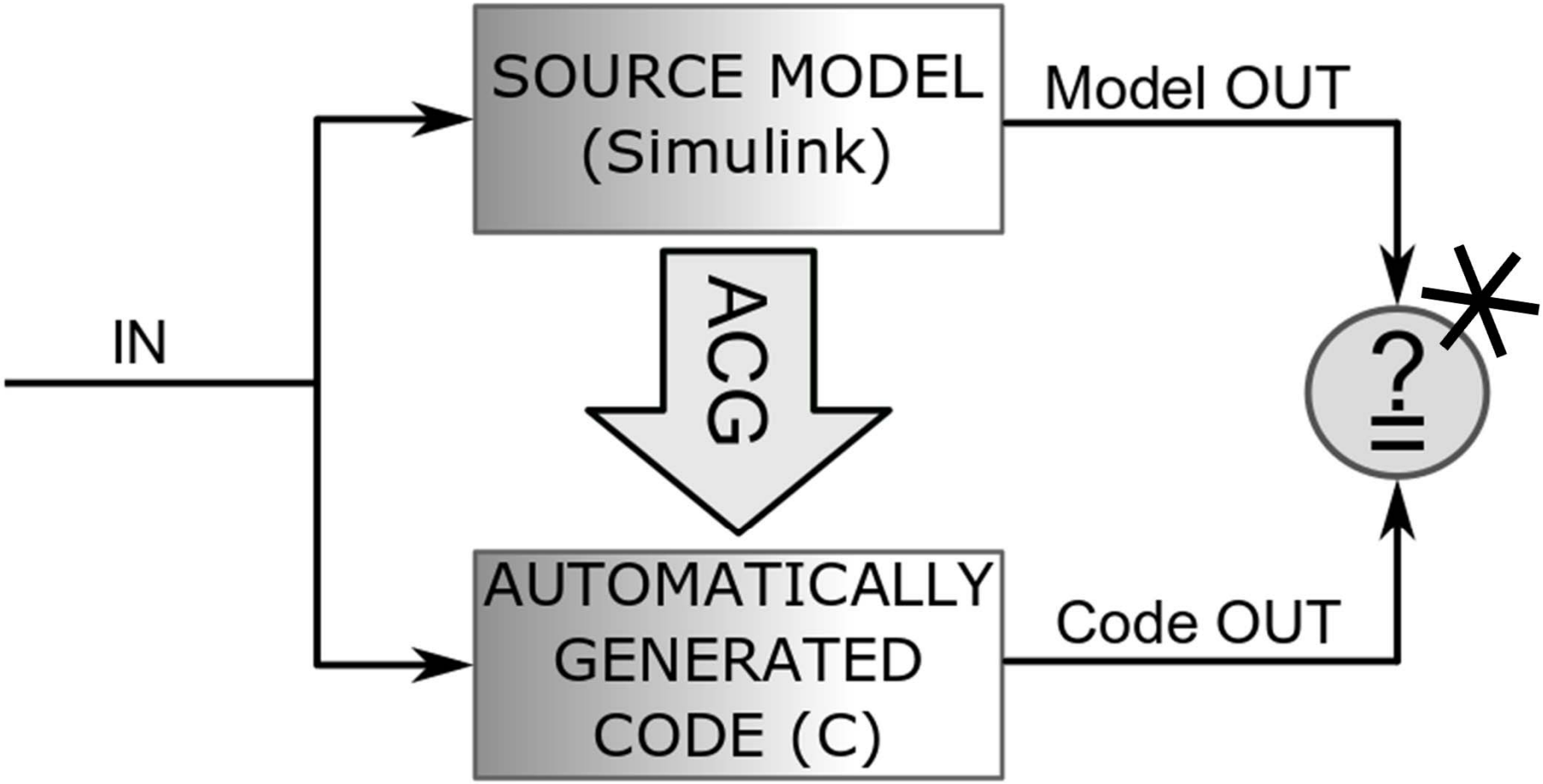
Performance verification on both source model and production code



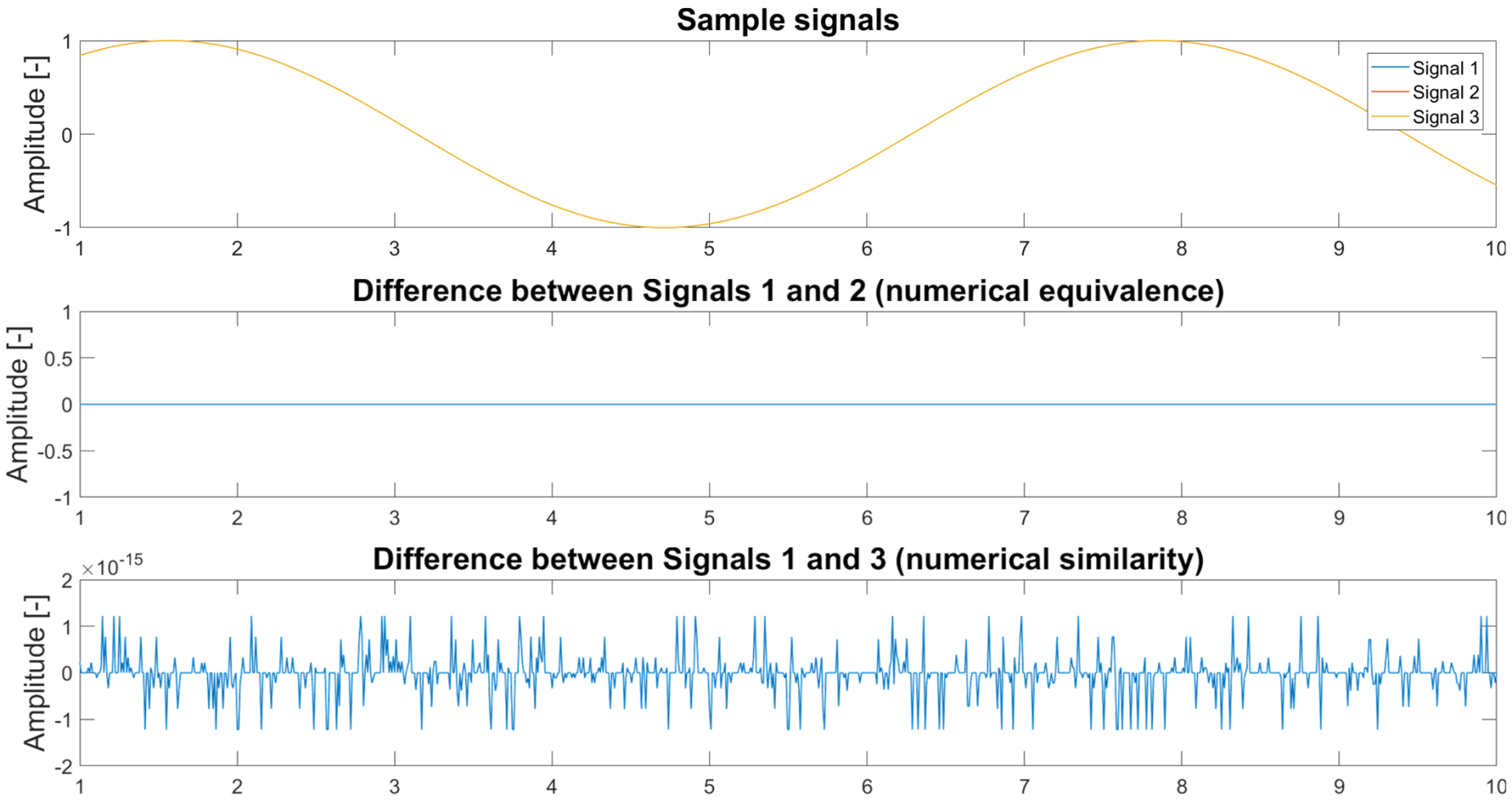
Performance verification on the source model and proof of equivalence with the production code



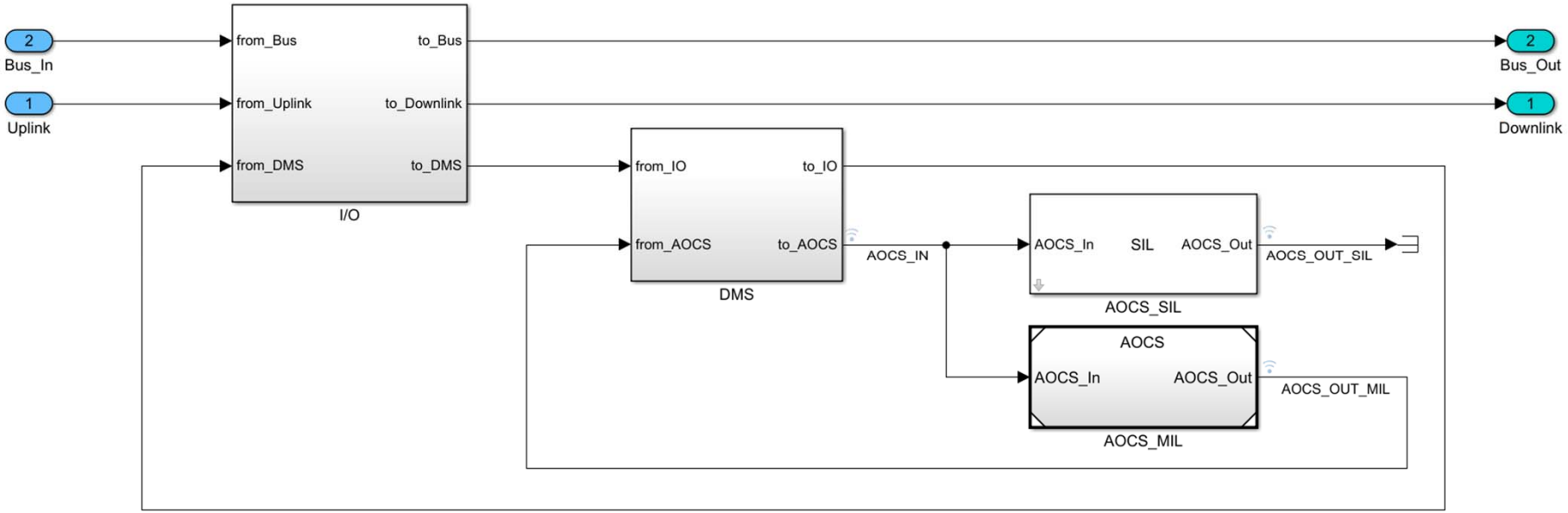
Equivalence between autocoded SW and source



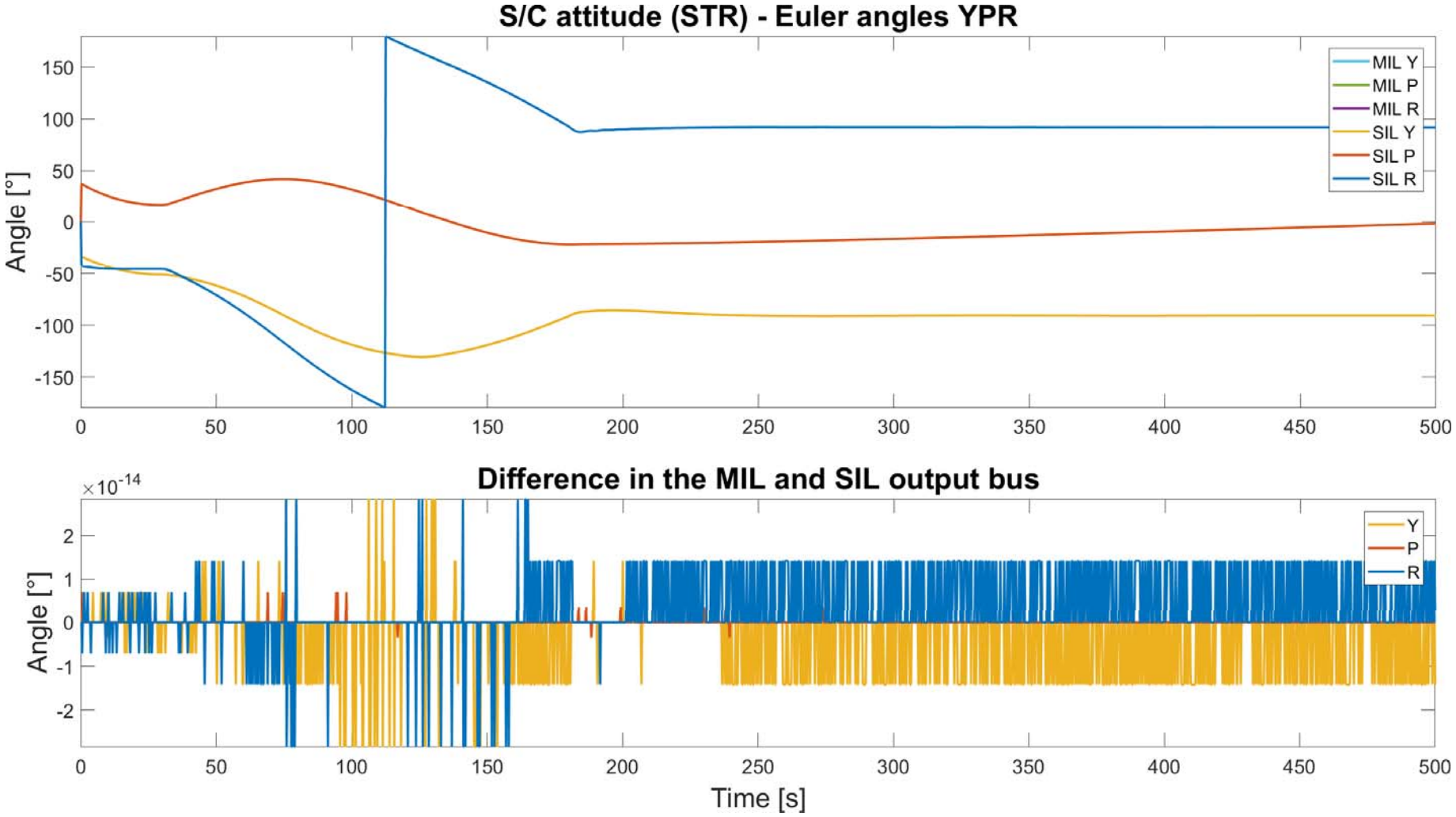
Numerical *equivalence* vs *similarity*



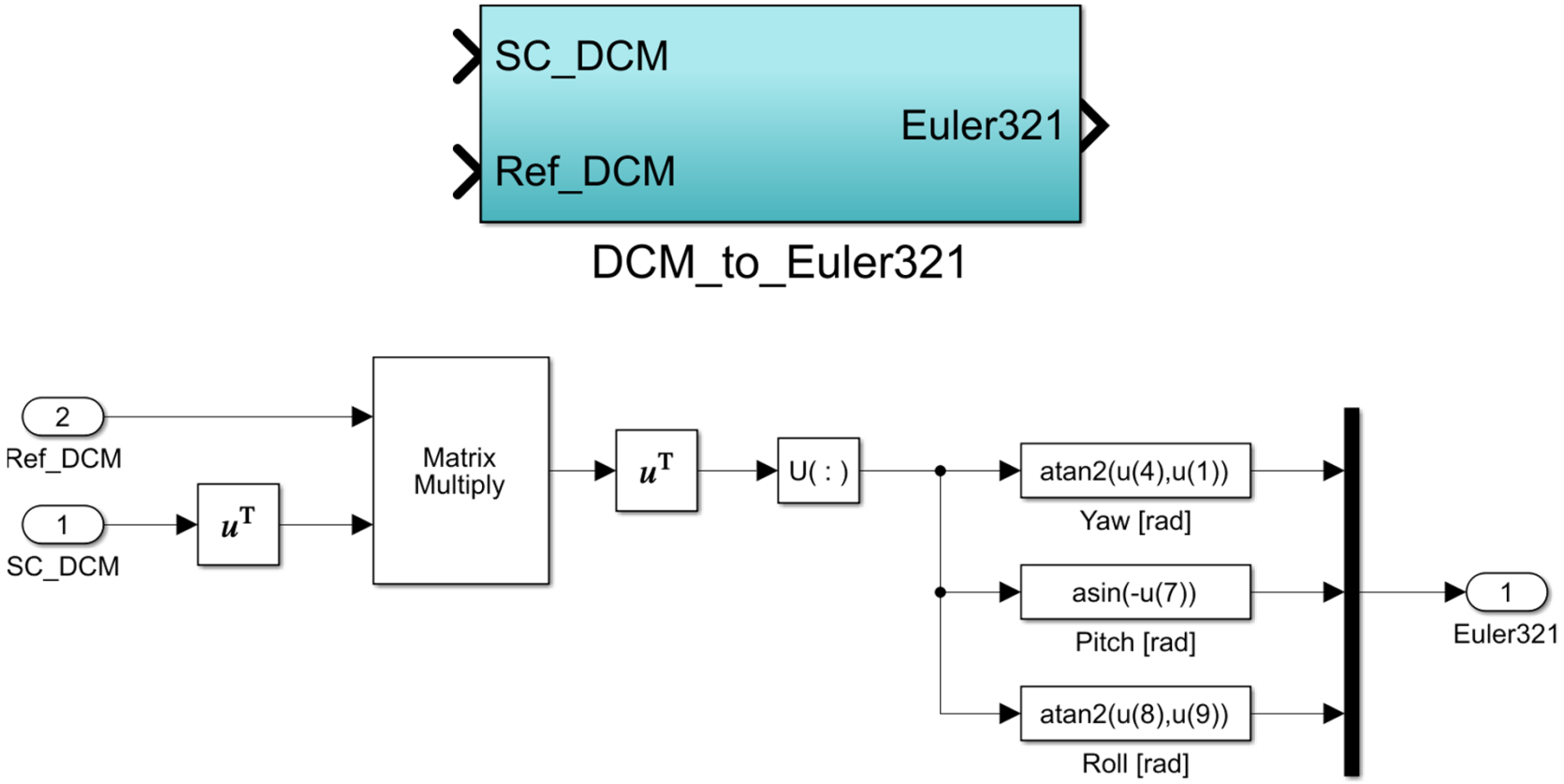
The SIL test-harness



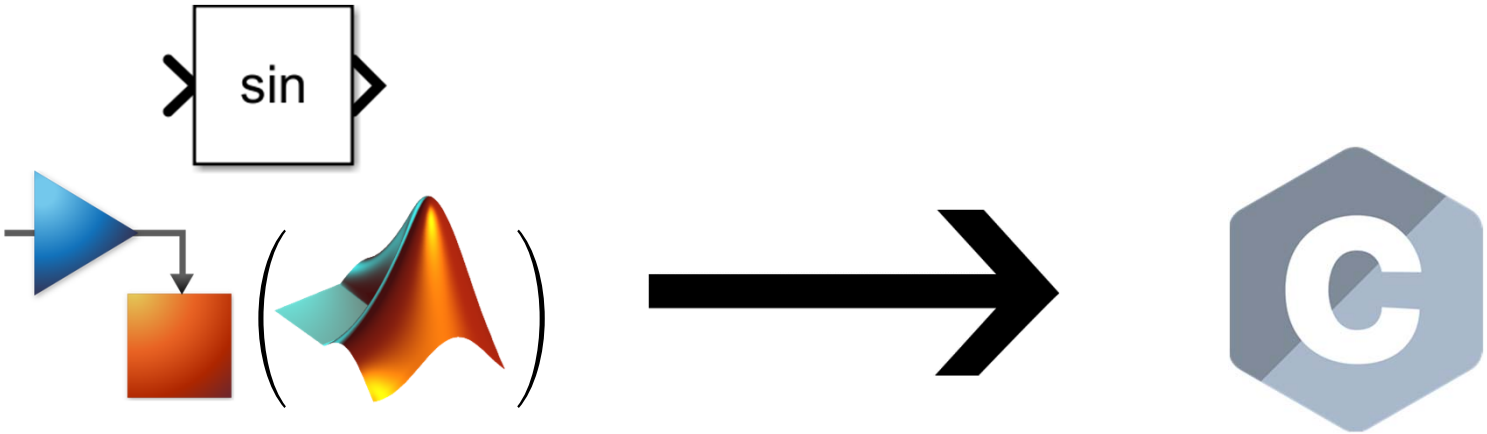
Example of equivalence



Identification of the issue

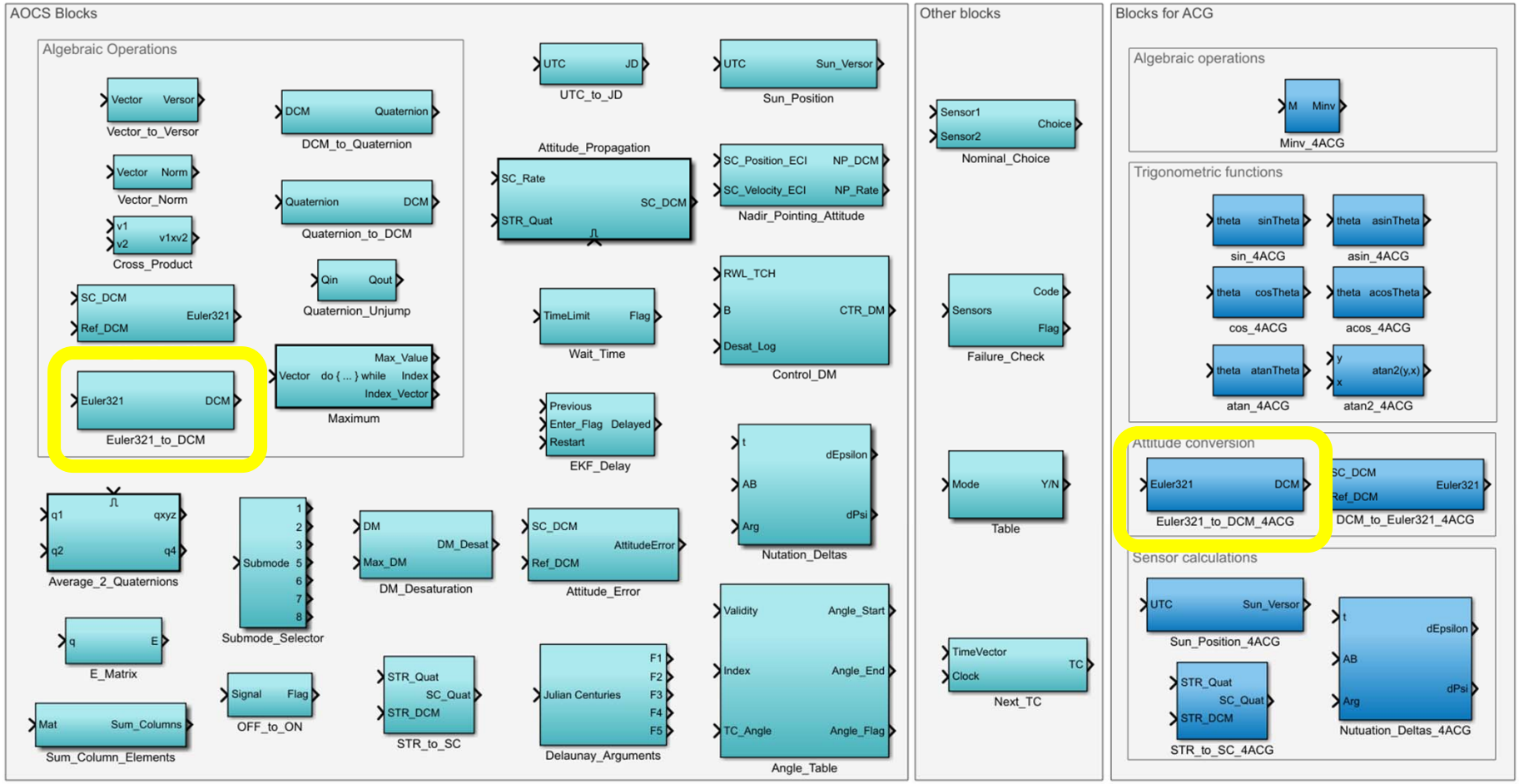


The issue



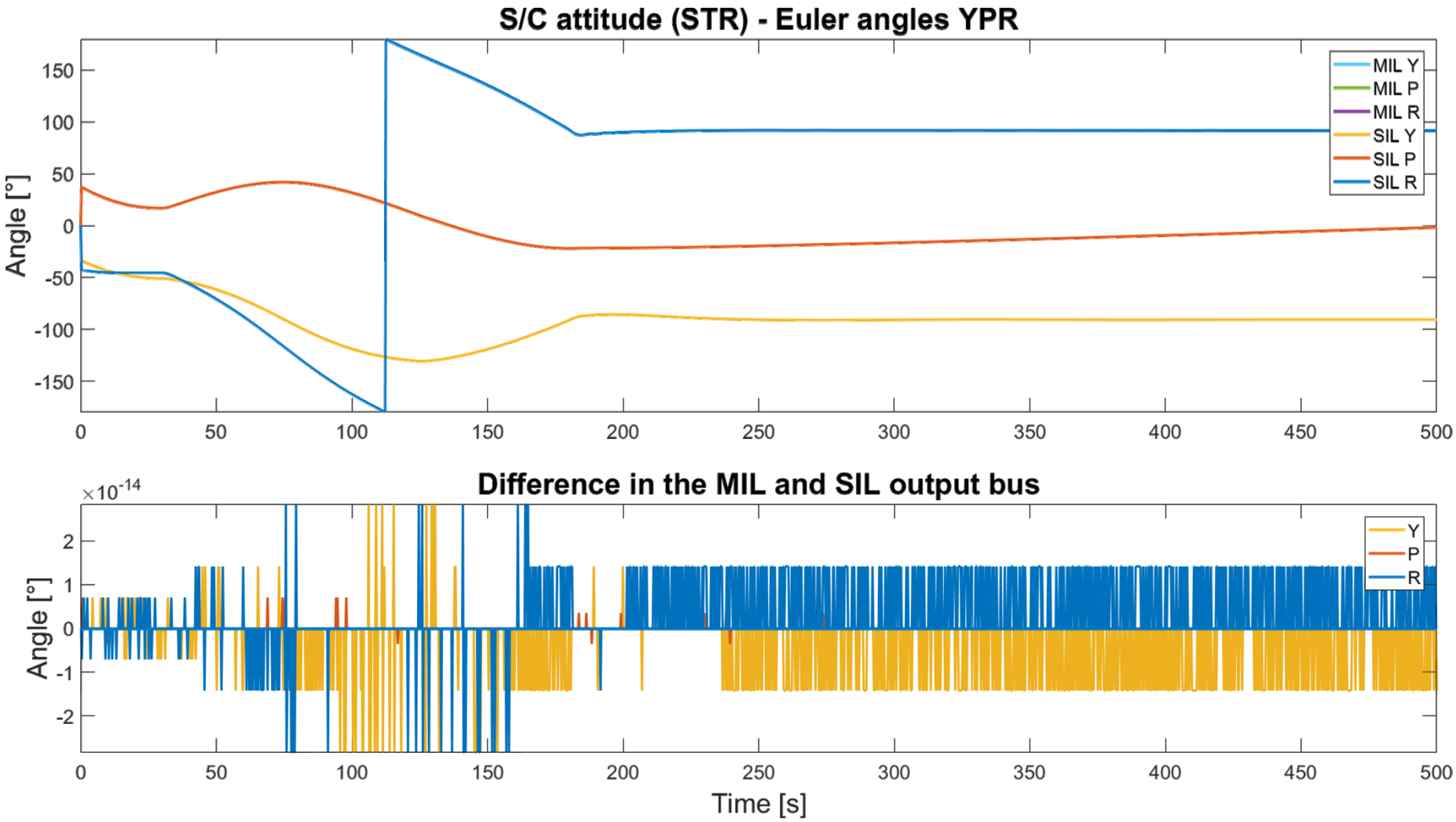
Numerical discrepancy is caused by a different **low-level implementation** of the trigonometric functions in the two languages

The *extended* library

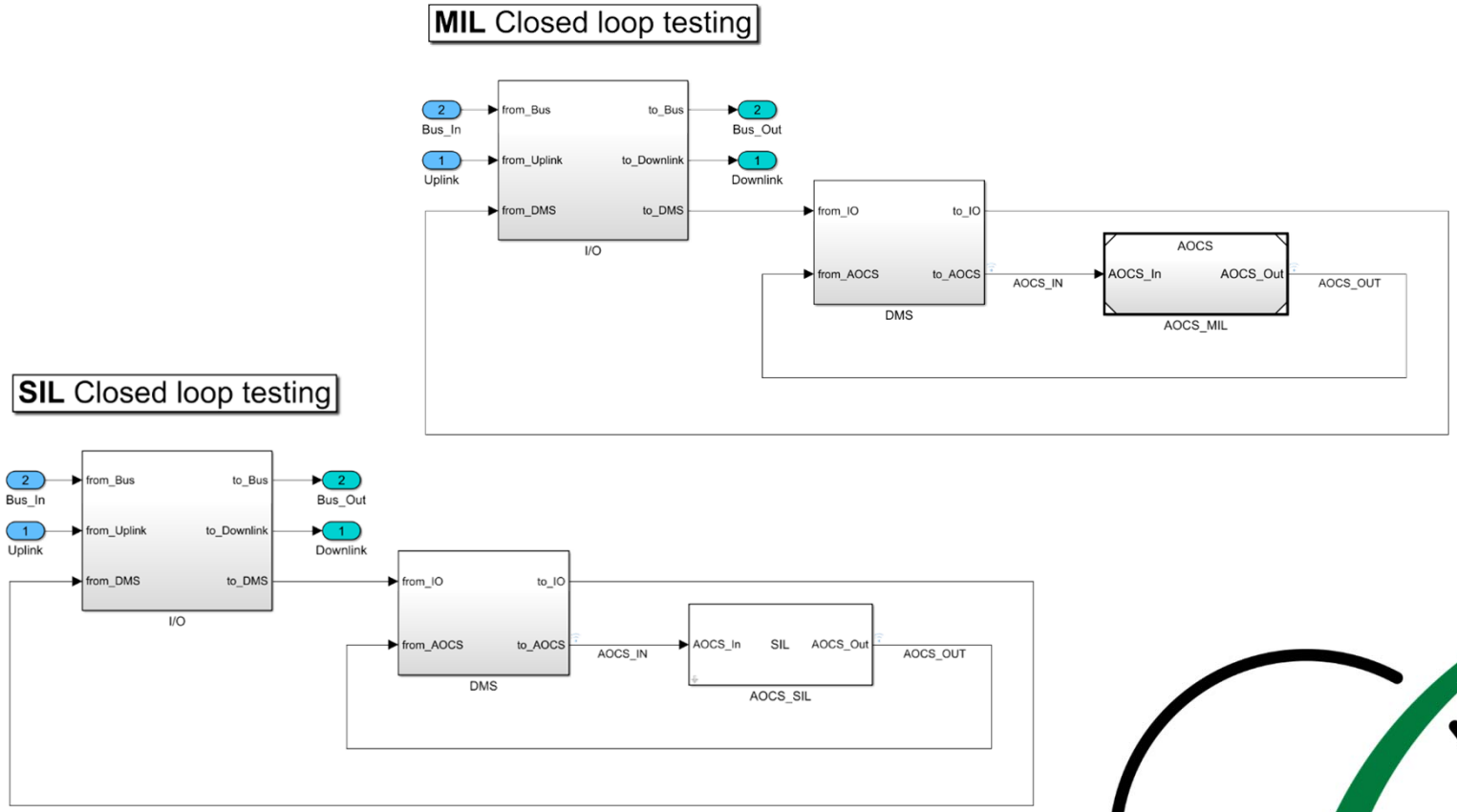


Numerical equivalence

With the original library block

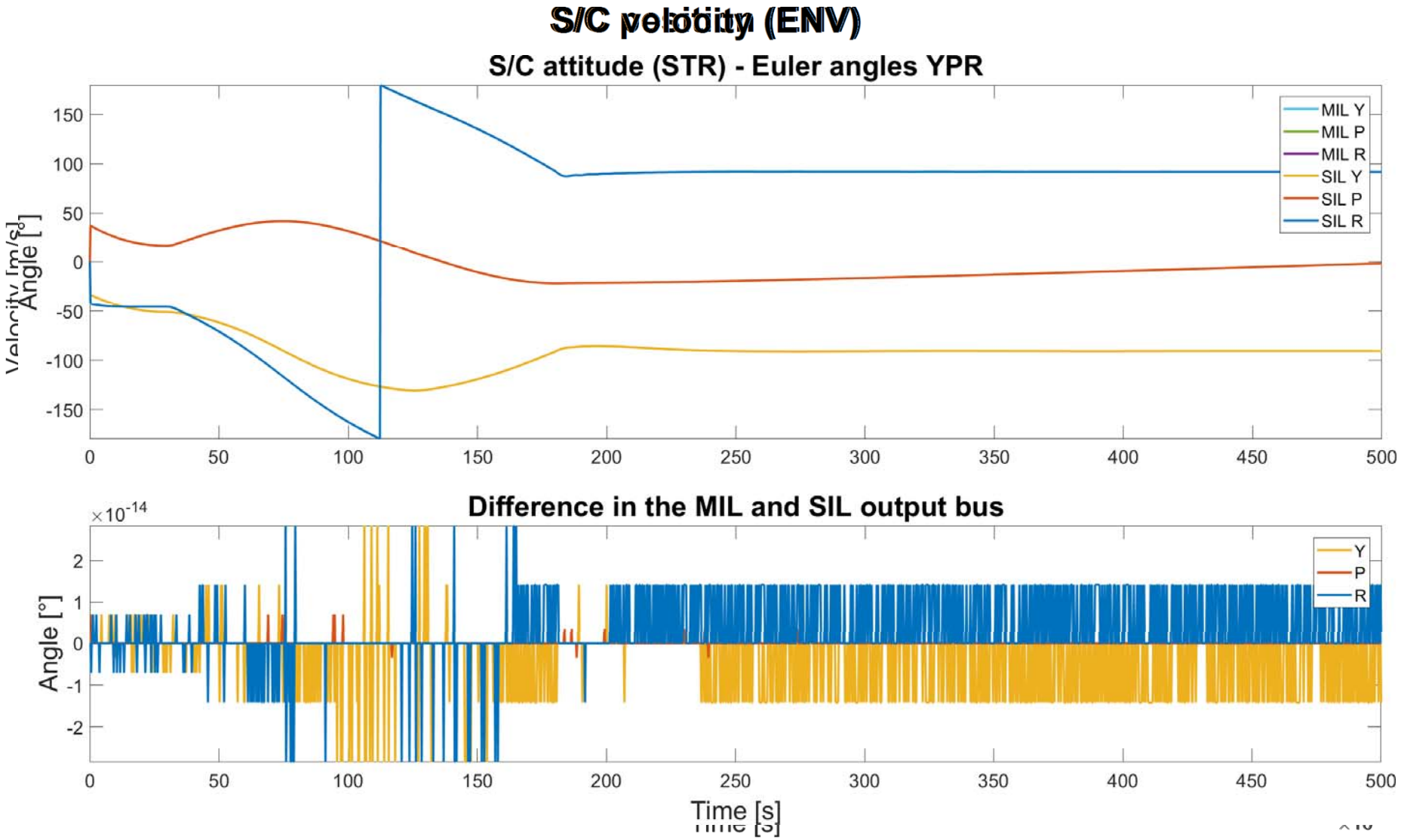


Numerical equivalence – closed loop behavior



- 100 iterations of the MC campaign in SHM
- 100 iterations of the MC campaign in IMM
- *Long* simulations (10 orbital periods)

Difference in CL behavior without *equivalence*



Autocoding Handbook

1. INTRODUCTION
2. APPLICABLE AND REFERENCE DOCUMENTS
3. TERMS, DEFINITION AND ABBREVIATED TERMS
4. INTRODUCTION to AOCS FSW development PROCESS
    a. The classical process: Manual Coding
    b. Introduction to Autocoding
    c. Comparison and key differences
5. MODELLING GUIDELINES FOR CODE GENERATION
    a. Modelling guidelines
        - General modelling guidelines
        - Modelling with Matlab
        - Modelling with Simulink
        - Modelling with Stateflow
    b. Code generation guidelines
        - Coder configuration settings
        - Generated Code structure
        - Reuse of legacy code
6. VERIFICATION & VALIDATION
    a. Automatic model and code verification tools
7. Traceability to ECSS-E-40C, ecss-q-80 FOR AUTOCODING
    a. Traceability to ECSS-E-40C
    b. Traceability to ECSS-Q-80C
8. GENERATION OF REPORTING
    a. Design reports
    b. Test plans
    c. Test reports
    d. Verification control documents
9. CONCLUSIONS
10. ANNEX A: EXAMPLES
    a. Project Examples
    b. Matlab examples
11. ANNEX B: SIMULINK BLOCKS ALLOWED FOR CODE GENERATION

Autocoding Handbook

1. INTRODUCTION
2. APPLICABLE AND REFERENCE DOCUMENTS
3. TERMS, DEFINITION AND ABBREVIATED TERMS
4. INTRODUCTION to AOCS FSW development PROCESS
    a. The classical process: Manual Coding
    b. Introduction to Autocoding
    c. Comparison and key differences
5. **MODELLING GUIDELINES FOR CODE GENERATION**
    a. **Modelling guidelines**
        - **General modelling guidelines**
        - **Modelling with Matlab**
        - **Modelling with Simulink**
        - **Modelling with Stateflow**
    b. **Code generation guidelines**
        - **Coder configuration settings**
        - **Generated Code structure**
        - **Reuse of legacy code**
6. VERIFICATION & VALIDATION
    a. Automatic model and code verification tools
7. Traceability to ECSS-E-40C, ecss-q-80 FOR AUTOCODING
    a. Traceability to ECSS-E-40C
    b. Traceability to ECSS-Q-80C
8. GENERATION OF REPORTING
    a. Design reports
    b. Test plans
    c. Test reports
    d. Verification control documents
9. CONCLUSIONS
10. ANNEX A: EXAMPLES
    a. Project Examples
    b. Matlab examples
11. ANNEX B: SIMULINK BLOCKS ALLOWED FOR CODE GENERATION

Severity of guidelines

Mandatory	Strongly Recommended	Recommended
Definition		
• Guidelines that are absolutely essential. • Guidelines where 100% compliance shall be required.	• Guidelines that are agreed upon to be a good practice, but use of legacy models preclude from being compliant at 100% • Models should conform to these guidelines to the greatest extent possible; however 100% compliance is not required	• Guidelines that are recommended to improve the appearance of the model diagram, but are not critical to running the model • Guidelines where conformance is preferred, but not required
Consequences – If the guideline is violated		
• Essential items are missing • The model might not work properly	• The quality and the appearance deteriorates • There may be an adverse effect on maintainability, portability, and reusability	• The appearance will not conform with other projects
Waiver Policy – If the guideline is intentionally ignored		
• The reasons must be justified and documented	• The reasons must be documented	

Modelling and Coding guidelines

General Modelling Guidelines — Approaches everything that has to do with the environment, in which the user models the system

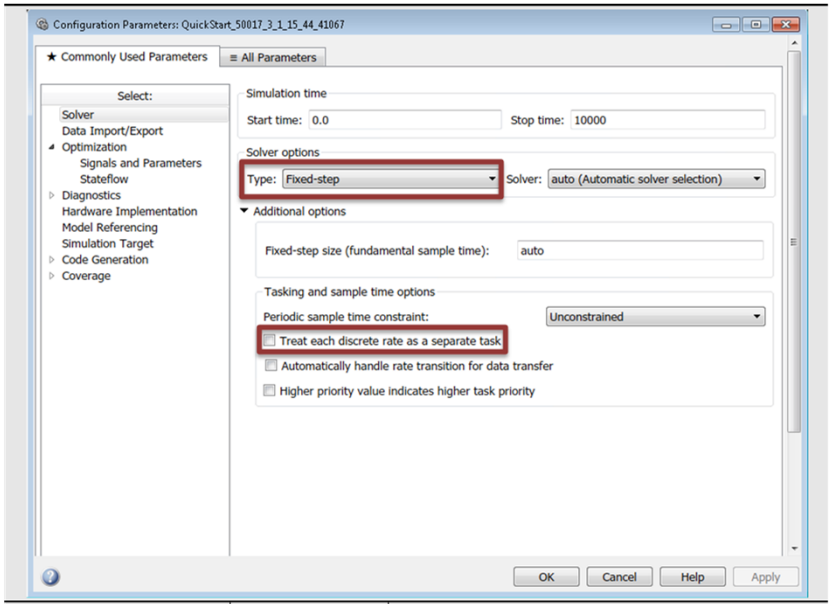
ID	ESA-SY-001
Title	Consistent software environment
Priority	Mandatory
Description	During software development, it is recommended that a consistent software environment is used across the project. Software includes, but is not limited, to: - MATLAB - Simulink - C Compiler (for simulation) - C Compiler (for target hardware) Consistent software environment implies that the same version of the software is used across the full project. The version number applies to any patches or extensions to the software used by a group.
Rationale	If different versions are used there is no guarantee that the features will be compatible and the generated code is the same. This rule ensures the outcome is as expected.

Simulink — Rules regarding the Simulink blocks

ID	ESA-SL-001
Title	Blocks not recommended for C/C++ code production
Priority	Mandatory
Description	The model should not have any kind of blocks that are not suitable for code production. The list of such blocks can be used is in annex **Error! Reference source not found.**. Automatic Testing: mathworks.do178.PCGSupport mathworks.maab.jm_0001 mathworks.maab.hd_0001
Rationale	Using blocks compatible with code generation is essential for the process.

Generated Code Structure — These rules apply to the entire model

ID	ESA
Title	Parameter definition
Priority	Highly Recommended
Description	The parameters should be documented along with the class chosen for the parameter definition. It is recommended that parameters are defined either in the File Scope or in a general file containing all the OBSW parameters. Procedures and options on how to define parameter classes are demarcated in subsection **Error! Reference source not found.**.
Rationale	By defining beforehand how the parameters should be defined, it become predictable in which portion of the code the parameters will be declared and defined.



Parameter	Value	Description
Type:	Fixed-step	Required for code generation
Treat each discrete rate as a separate task	Unselected	Makes sure only one sample time (interruption in generated code) is generated.

Autocoding Handbook

1. INTRODUCTION
2. APPLICABLE AND REFERENCE DOCUMENTS
3. TERMS, DEFINITION AND ABBREVIATED TERMS
4. INTRODUCTION to AOCS FSW development PROCESS
    a. The classical process: Manual Coding
    b. Introduction to Autocoding
    c. Comparison and key differences
5. MODELLING GUIDELINES FOR CODE GENERATION
    a. Modelling guidelines
        - General modelling guidelines
        - Modelling with Matlab
        - Modelling with Simulink
        - Modelling with Stateflow
    b. Code generation guidelines
        - Coder configuration settings
        - Generated Code structure
        - Reuse of legacy code
6. VERIFICATION & VALIDATION
    a. Automatic model and code verification tools
7. Traceability to ECSS-E-40C, ecss-q-80 FOR AUTOCODING
    a. Traceability to ECSS-E-40C
    b. Traceability to ECSS-Q-80C
8. GENERATION OF REPORTING
    a. Design reports
    b. Test plans
    c. Test reports
    d. Verification control documents
9. CONCLUSIONS
10. ANNEX A: EXAMPLES
    a. Project Examples
    b. Matlab examples
11. ANNEX B: SIMULINK BLOCKS ALLOWED FOR CODE GENERATION

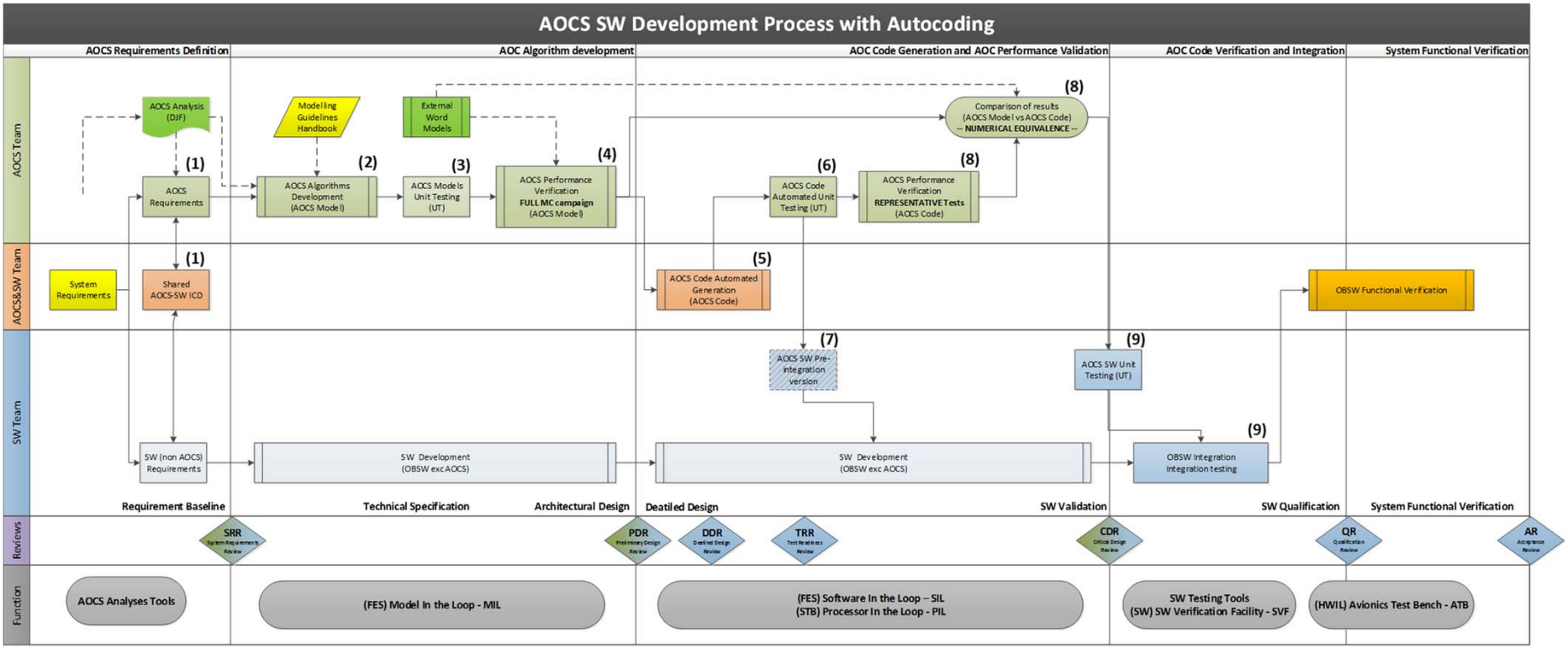
Autocoding Handbook

1. INTRODUCTION
2. APPLICABLE AND REFERENCE DOCUMENTS
3. TERMS, DEFINITION AND ABBREVIATED TERMS
4. INTRODUCTION to AOCS FSW development PROCESS
    a. The classical process: Manual Coding
    b. Introduction to Autocoding
    c. Comparison and key differences
5. MODELLING GUIDELINES FOR CODE GENERATION
    a. Modelling guidelines
        - General modelling guidelines
        - Modelling with Matlab
        - Modelling with Simulink
        - Modelling with Stateflow
    b. Code generation guidelines
        - Coder configuration settings
        - Generated Code structure
        - Reuse of legacy code

6. VERIFICATION & VALIDATION
    a. Automatic model and code verification tools
7. Traceability to ECSS-E-40C, ECSS-Q-80 for Autocoding
    a. Traceability to ECSS-E-40C
    b. Traceability to ECSS-Q-80C

8. GENERATION OF REPORTING
    a. Design reports
    b. Test plans
    c. Test reports
    d. Verification control documents
9. CONCLUSIONS
10. ANNEX A: EXAMPLES
    a. Project Examples
    b. Matlab examples
11. ANNEX B: SIMULINK BLOCKS ALLOWED FOR CODE GENERATION

Development and Verification Process



Traceability vs ECSS E-40 and ECSS Q-80

Clause	Description	Compliance
5.3.2.4	Automatic code generation a. The autocode input models shall be reviewed together with the rest of the software specification, architecture and design. NOTE The autocode input models are integral part of the software specification, architecture and design. EXPECTED OUTPUT: Autocode input model review [MGT, SDP; SRR, PDR]. b. In the case of coexisting autocoded and manually written parts, the software development plan shall include the definition of a clear interface definition and resource allocation (memory, CPU) at PDR. EXPECTED OUTPUT: Autocode interface definition and resource allocation [MGT, SDP; SRR, PDR]. c. The input model management, the code generation process and supporting tools shall be documented in the SDP. EXPECTED OUTPUT: Automatic code generation development process and tools [MGT, SDP; SRR, PDR]. d. The supplier shall define in the SDP the verification and validation strategy for automatic code generation as a result of the trade off between the qualification of the code generation toolchain and the end to end validation strategy of the software item, or any combination thereof, in relation with ECSS-Q-ST-80 clause 6.2.8. EXPECTED OUTPUT: Automatic code generation verification and validation strategy [MGT, SDP; SRR, PDR]. e. The configuration management of the automatic code generation related elements shall be defined in the SCMP. EXPECTED OUTPUT: Automatic code generation configuration management [MGT, SCMP; SRR, PDR].	a. Proposed in this HB: the model is part of the PDR, DDR reviewed by joint GNC/SW teams b. As proposed in this HB. In particular a SW/SW ICD between manual SW/GNC models and autocoded SW shall exist and be submitted to PDR c. This HB provided useful inputs for such Software Development Plan d. Qualification of the code generator is complex. Instead, this HB provide inputs for producing automated “qualifiable” code e. The approach to configuration management of model options, model toolchain shall be described in the SW Configuration Management Plan.

Autocoding Handbook

1. INTRODUCTION
2. APPLICABLE AND REFERENCE DOCUMENTS
3. TERMS, DEFINITION AND ABBREVIATED TERMS
4. INTRODUCTION to AOCS FSW development PROCESS
    a. The classical process: Manual Coding
    b. Introduction to Autocoding
    c. Comparison and key differences
5. MODELLING GUIDELINES FOR CODE GENERATION
    a. Modelling guidelines
        - General modelling guidelines
        - Modelling with Matlab
        - Modelling with Simulink
        - Modelling with Stateflow
    b. Code generation guidelines
        - Coder configuration settings
        - Generated Code structure
        - Reuse of legacy code
6. VERIFICATION & VALIDATION
    a. Automatic model and code verification tools
7. Traceability to ECSS-E-40C, ECSS-Q-80 for Autocoding
    a. Traceability to ECSS-E-40C
    b. Traceability to ECSS-Q-80C
8. GENERATION OF REPORTING
    a. Design reports
    b. Test plans
    c. Test reports
    d. Verification control documents
9. CONCLUSIONS
10. ANNEX A: EXAMPLES
    a. Project Examples
    b. Matlab examples
11. ANNEX B: SIMULINK BLOCKS ALLOWED FOR CODE GENERATION

Autocoding Handbook

1. INTRODUCTION
2. APPLICABLE AND REFERENCE DOCUMENTS
3. TERMS, DEFINITION AND ABBREVIATED TERMS
4. INTRODUCTION to AOCS FSW development PROCESS
    a. The classical process: Manual Coding
    b. Introduction to Autocoding
    c. Comparison and key differences
5. MODELLING GUIDELINES FOR CODE GENERATION
    a. Modelling guidelines
        - General modelling guidelines
        - Modelling with Matlab
        - Modelling with Simulink
        - Modelling with Stateflow
    b. Code generation guidelines
        - Coder configuration settings
        - Generated Code structure
        - Reuse of legacy code
6. VERIFICATION & VALIDATION
    a. Automatic model and code verification tools
7. Traceability to ECSS-E-40C, ECSS-Q-80 for Autocoding
    a. Traceability to ECSS-E-40C
    b. Traceability to ECSS-Q-80C
8. GENERATION OF REPORTING
    a. Design reports
    b. Test plans
    c. Test reports
    d. Verification control documents

TBW

9. CONCLUSIONS
10. ANNEX A: EXAMPLES
    a. Project Examples
    b. Matlab examples
11. ANNEX B: SIMULINK BLOCKS ALLOWED FOR CODE GENERATION

SAVOIR AUTOCODE

space avionics open interface architecture

Extended Working Group

Extended Working Group ToR

The purpose of this Extended Working Group (EWG) is to review and update the draft *ESA Modeling guidelines for Autocoding Handbook* to be used as reference when creating models and generating flight code using modelling and autocoding tools

The intended use of the guidelines are:

- support to projects providing a harmonized ESA position across the Agency.
- Support to R&D technology activities.
- Promotion of the use of this type of methodology across the phases of a development.
- Contribution to the assessment of the quality of the final software product

Extended Working Group Schedule

T_0 (October, 2018)	Kick-off: Distribution to nominated representatives of draft ESA Handbook
December, 2018	Deadline for comments
February/March 2019	Individual meetings
30th March 2019	Second distribution of draft ESA Handbook
(TBD) April 2019	Plenary EWG ACG meeting (ESTEC)
May 2019	Official release of the *Guidelines for the Automatic Code Generation for AOCS/GNC Flight SW Handbook*

MAY2019 → Guidelines for the Automatic Code Generation for AOCS/GNC Flight SW Handbook

Conclusions