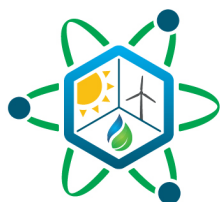




Development of the IES Plug-and-Play Framework

March | 2021

**Konor L Frick
Andrea Alfonsi
Cristian Rabiti
Shannon Bragg-Sitton
Idaho National Laboratory**



IES

Integrated Energy Systems

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

Development of the IES Plug-and-Play Framework

**Konor L Frick
Andrea Alfonsi
Cristian Rabiti
Shannon Bragg-Sitton
*Idaho National Laboratory***

March 2021

**Idaho National Laboratory
Integrated Energy Systems
Idaho Falls, Idaho 83415**

<http://www.ies.inl.gov>

**Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517**

Page intentionally left blank

ABSTRACT

Since early 2013, to accommodate the vast array of possibilities introduced by the concept of integrated energy parks that could incorporate multiple energy generation sources and multiple energy users, Idaho National Laboratory (INL) has been developing a library of high-fidelity process models in the Modelica modeling language. These models are a cornerstone of the analysis and optimization tools developed via the Department of Energy Office of Nuclear Energy (DOE-NE) Integrated Energy Systems (IES) program, led by Idaho National Laboratory (INL). Models are used to create and characterize system inertia, thermal losses, and the efficiency of integrated systems. These physical models help map physical performance into economic performance, allowing for system-level optimization. In addition, the models are used to test innovative system-level control strategies for interconnected thermal generators.

However, for real-world applications, it is not always practical to develop a model or rewrite an existing model in Modelica; instead, interoperability with Legacy FORTRAN, C, Python, or other codes is required. To accomplish this interoperability the IES Program is seeking to modify the existing suite of physical models, currently held in the HYBRID physical modeling repository, to be consistent with a “plug-and-play” approach in Modelica/Dymola models using Functional Mock-Up Units (FMUs), Functional Mock-Up Interfaces (FMIs), and machine-learning techniques. The models developed are held within the HYBRID repository that is part of the IES Framework for Optimization of Resources and Economics ecosystem (FORCE).

This report provides an overview of all the performed activities revolving around the deployment of methods, software infrastructures, guidelines, and a workflow for the construction and usage of models, as encapsulated using the FMI/FMU protocols and standards. The report is organized into three main macro-subjects, all of which are interconnected:

- FMI/FMU adaptors for Modelica models
- The HYBRID repository’s new structure and open-source deployment
- RAVEN FMI/FMU exporting capabilities and artificial-intelligence (AI)-based analysis acceleration.

The first part of the report discusses the FMI/FMU adaptors created within the HYBRID repository to allow users to quickly export models such as FMUs. Several examples are given, highlighting the step-by-step process of converting an existing Modelica model into an FMU for use within the Dymola platform. Simulation results demonstrate that, though minor differences may occur, overall control, trends, and solution integrity are maintained between the standard Modelica simulation and FMU simulation results. However, it is worth noting that, for small systems, the FMU requires a longer simulation time than the Modelica-only simulation. Using this process, a company can provide external entities with models that contain proprietary information, without disclosing any model-related information that could be considered business sensitive. Such an ability would allow institutions to bypass the necessity of having “whitewashed” data.

In the second part of the report, the new structure of the HYBRID repository is discussed, with a major focus on the series of completed updates. These updates include the addition of Modelica system-level regression tests and software quality assurance (SQA) documentation to ensure that modifications to the Modelica models do not alter system-level model results.

The third and final part of the report documents the work performed for deploying methods and workflows to construct RAVEN AI-based models that are compliant with the FMI/FMU standard. Such work is key for deployment of the “flexible ecosystem” concept, since it allows for the replacement of high-fidelity Modelica models (or any other FMI/FMU-compliant model) with RAVEN-generated AI surrogate models.

Overall, extensive work has been completed in regard to developing FMUs and FMIs from existing models, understanding the requirements and limitations of FMUs, and open-sourcing the HYBRID repository with an integrated regression system for use within FORCE.

Page intentionally left blank

CONTENTS

ABSTRACT.....	iii
ACRONYMS.....	xii
1. INTRODUCTION.....	1
2. FUNCTIONAL MOCK-UP INTERFACES AND UNITS.....	3
2.1 Co-simulation.....	4
2.2 Model Exchange.....	5
2.3 Advantages of Each Protocol.....	6
3. MODELICA TO FMU ADAPTATION.....	7
3.1 Adaptors.....	7
Fluid Port Adaptors.....	9
Thermal Port Adaptors.....	13
Electrical Port Adaptors.....	19
3.2 FMI Construction Guide.....	20
Model Preparation.....	21
Adaptors.....	22
Export	23
Import	24
Simulation.....	26
3.3 Turbine Replacement Example.....	28
4. HYBRID REPOSITORY	31
5. DEPLOYMENT OF A RAVEN FMI/FMU DRIVER.....	37
5.1 RAVEN Introduction.....	37
5.2 RAVEN Models.....	38
6. DEPLOYMENT OF RAVEN FMI/FMU EXPORTER FOR AI-BASED ANALYSIS ACCELERATIONS.....	44
6.1 RAVEN AI construction.....	45
6.2 Development of FMI/FMU exporting capabilities for RAVEN AI.....	46
6.3 Future Extension of the FMI/FMU Exporter to the RAVEN Hybrid Model.....	48
7. Integrated Energy Park Demonstration Case.....	49
7.1 FMI/FMU Creation and Use within Dymola.....	50
7.2 Creation of Surrogate Using RAVEN.....	54
7.3 Comparison of Results.....	56
8. CONCLUSION.....	60
9. FUTURE WORK.....	61
10. REFERENCES.....	62
APPENDIX A – HYBRID USER MANUAL.....	65

APPENDIX B – SQA: SOFTWARE QUALITY ASSURANCE PLAN (SQAP).....	113
APPENDIX C – SQA: SOFTWARE DESIGN DESCRIPTION (SDD).....	140
APPENDIX D – SQA: HYBRID SOFTWARE REQUIREMENTS SPECIFICATION AND TRACEABILITY MATRIX (SPC).....	164
APPENDIX E – SQA: HYBRID CONFIGURATION ITEM LIST.....	186

FIGURES

Figure 1. Example IES architecture, illustrating thermal and electrical interconnection to support hydrogen production and chemical conversion.....	1
Figure 2. Plug-and-play framework environment.....	2
Figure 3. Co-simulation FMI/FMU scheme.....	5
Figure 4. Model exchange FMI/FMU scheme.....	6
Figure 5. Fluid ports (note that “ports” are a container method in Modelica used to transfer several pieces of physics-based information within a single “connector”).....	7
Figure 6. Transition from a Modelica physical model into an FMU.....	8
Figure 7. FMU template folder location within the larger Nuclear Hybrid Energy Systems (NHES) folder as part of the HYBRID Repository.....	9
Figure 8. (Left) PressuretoMassFlow adaptor. This adaptor is best connected to a resistance port able to set the output mass flow rate. (Right) MassFlowtoPressure adaptor. This adaptor is best connected to a volume port able to set the output pressure. Note: Both of these adaptors were created by Modelon for use in the INL plug-and-play framework as part of an FMI/FMU course subcontract.....	10
Figure 9. An example (using adaptors) involving two pressure sources using moist air, one of which oscillates in pressure, causing a mass flow reversal. The unit in the red box will become an FMU. (k=Temperature input to pressure flow boundary; freqHz = pressure frequency oscillation placed on pressure_source).....	11
Figure 10. Example of a reversible flow using two pressure sources, moist air, and a model exchange FMU. (k=Temperature input to pressure flow boundary; freqHz = pressure frequency oscillation placed on pressure_source).....	11
Figure 11. Example of a reversible flow using two pressure sources, moist air, and a co-simulation FMU. (k=Temperature input to pressure flow boundary; freqHz = pressure frequency oscillation placed on pressure_source).....	12
Figure 12. Comparison of mass flow to the pressure sink across the original model, model exchange FMU, and co-simulation FMU (timestep = 0.02 seconds).....	13
Figure 13. (Left) GeneralTemperatureToHeatFlow adaptor for use in the INL plug-and-play framework. (Right) GeneralHeatFlowToTemperature adaptor for use in the INL plug-and-play framework. (T=temperature, dT = first derivative of temperature, d2T = second derivative of temperature, Q = heat flow, der(Q) = first derivative of heat flow, der2(Q) = second derivative of heat flow.) Note: only T and Q are required the derivative values are optional for stability.....	14
Figure 14. Example meant to demonstrate the FMU variants available with the thermal FMU adaptors. The upper part demonstrates how to export two heat capacitors and connect them together in a target system. The lower part demonstrates how to export a	

conduction element that only requires temperatures for its conduction law, and connects this conduction law to both heat capacitors in a target system.....	15
Figure 15. Demonstration of an FMU variant example that uses model exchange FMUs for the thermal heat port adaptors.....	16
Figure 16. Collapse of the upper part of Figure 14 into a single FMU for co-simulation. This is required because the frequency between the direct and inverse conduction problem is so fast that a single cut between the two could not be made without instabilities occurring.....	16
Figure 17. Upper model of Figure 14 connected with the combined direct/inverse co-simulation FMU.....	17
Figure 18. Lower model of Figure 14, co-simulation FMU.....	17
Figure 19. Direct/inverse simulation results for the original, model exchange, and co-simulation (communication interval: 0.002 seconds) FMU runs. (Top) Heat flow into the capacitor. (Bottom) Capacitor 3b temperature.....	18
Figure 20. Conduction (lower model) simulation results for the original, model exchange, and co-simulation (communication interval: 0.002 seconds) FMU runs. (Top) Heat flow into the capacitor. (Bottom) Capacitor 3b temperature.....	19
Figure 21. (Left) GeneralFrequencyToPowerFlow adaptor for use in the INL plug-and-play framework. (Right) GeneralPowerFlowToFrequency adaptor for use in the INL plug-and-play framework. (Red circle represents the electrical port, with inputs and outputs equal to the aforementioned variables in the section).....	20
Figure 22. Incorrect level for proper export as FMI/FMU. Control system has not been declared and is replaceable from a higher level within the HYBRID repository.....	21
Figure 23. Correct level from which to begin FMI/FMU preparation. Control system has been selected via the drop-down menu available in the custom parameters section, shown on the left. (Red dots are electrical flow ports).....	22
Figure 24. Preparing a natural gas turbine to be converted into an FMU. The inputs into the system are the peaking demand and the connection points for electricity backflow into the turbine model. The output is the electrical power as a real value.....	23
Figure 25. Export settings from Dymola 2021x.....	24
Figure 26. Importing FMU steps in Dymola 2021x.....	25
Figure 27. Import settings from Dymola 2021x.....	25
Figure 28. Proper import and use of a co-simulation FMU in Dymola.....	26
Figure 29. FMI settings for the natural gas turbine FMI/FMU in co-simulation mode. The communication interval was every 0.12 seconds, with an internal solver tolerance of 1e-6. The internal solver was the Dymola specific DASSL solver.....	27
Figure 30. Comparison of Dymola model results to co-simulation and model exchange FMU results. Communication intervals for co-simulation = 0.12 seconds and 1 second.....	28
Figure 31. Translation of the Modelica turbine generator model into an FMU-ready design.....	29
Figure 32. Transition from a Modelica model to an FMI-based simulation.....	30
Figure 33. Comparison of turbine output results between the original model and model exchange FMU.....	30
Figure 34. New structure of the repository.....	32

Figure 35. An example of tests run in the ROOK regression system.....	35
Figure 36. Status of the required SQA documentation for the HYBRID modeling repository.....	37
Figure 37. RAVEN framework scheme.....	39
Figure 38. External model API.....	40
Figure 39. FMI/FMU model skeleton in RAVEN.....	41
Figure 40. FMI/FMU co-simulation protocol coupled with RAVEN.....	42
Figure 41. FMI/FMU model exchange protocol coupled with RAVEN.....	43
Figure 42. External model FMIFMU example RAVEN input file.....	43
Figure 43. Construction process for surrogate models in RAVEN.....	44
Figure 44. RAVEN ROM cross-validation scheme.....	45
Figure 45. RAVEN AI FMI/FMU exporting process.....	46
Figure 46. Example RAVEN input file to export AI as FMIs/FMUs.....	47
Figure 47. RAVEN's current FMI/FMU exporting capabilities.....	47
Figure 48. RAVEN hybrid model scheme.....	48
Figure 49. Integrated energy park consisting of a nuclear reactor (NPP), Energy Manifold (EM), Balance of Plant (BOP), Switch Yard (SY), Electric Batteries (Battery), Infinite Grid (IG), and a Natural Gas turbine (NG). The natural gas turbine is to be exported as an FMU.....	50
Figure 50. Preparing the natural gas turbine for conversion into an FMU. The inputs into the system are the peaking demand and connection points for electricity backflow into the turbine model. The output is the electrical power as a real value.....	51
Figure 51. Integrated energy park consisting of a nuclear reactor, electric batteries, and a natural gas turbine replaced by a co-simulation FMU.....	52
Figure 52. Top) Five-hour simulation of the natural gas turbine power vs. setpoint demand for the integrated energy park in regard to Modelica-only model, co-simulation FMI, and model exchange FMU. Bottom) Closeup shot of the turbine demand vs. turbine output for the different FMI versions. Note that all agree reasonably well. Co-simulation communication interval = 1 second.....	53
Figure 53. Simplified model of the FMI for RAVEN surrogation.....	54
Figure 54. Comparison of the Modelica/Dymola GTTP.fmu response (P_flow) and the RAVEN AI-based GTTProm.fmu.....	55
Figure 55. Closeup of the comparison of the Modelica/Dymola GTTP.fmu response (P_flow) and the RAVEN AI-based GTTProm.fmu.....	56
Figure 56. Integrated energy park (excluding the turbine) FMI/FMU generated with Dymola.....	57
Figure 57. Integrated energy park FMI/FMU, including the Dymola GTTP model.....	58
Figure 58. Integrated energy park FMI/FMU, replacing the Dymola GTTP model with the RAVEN AI-based FMI/FMU.....	58
Figure 59. Co-Simulation FMI/FMU (Dymola) vs. RAVEN AI-based FMI/FMU for a 5-hour simulation of the turbine power vs. setpoint demand for the integrated energy park....	59

Figure 60. Co-Simulation FMI/FMU (Dymola) vs. RAVEN AI-based FMI/FMU closeup shot of turbine demand vs turbine output.....	59
Figure 61. Proposed master simulator within RAVEN.....	61

TABLES

Table 1. Synopsis of Modelica test cases.....	32
---	----

Page intentionally left blank

ACRONYMS

AI	Artificial Intelligence
API	Application Program Interface
CPU	Central Processing Unit
FMI	Functional Mock-Up Interface
FMU	Functional Mock-Up Unit
FORCE	Framework for Optimization of Resources and Economics ecosystem
FOM	Figure of merit
HTSE	High Temperature Steam Electrolysis
IES	Integrated Energy Systems
INL	Idaho National Laboratory
NHES	Nuclear Hybrid Energy Systems
RAVEN	Risk Analysis and Virtual Environment
ROM	Reduced-order model
SMR	Small Modular Reactor
SQA	Software Quality Assurance
V&V	Validation and Verification

Page intentionally left blank

1. INTRODUCTION

Grid demand variability is an inherent part of the modern dynamic lifestyle. The addition of renewable energy (e.g., wind and solar) technologies introduces variability into the grid supply. As renewable energy integration continues to grow, variability will further increase. The Department of Energy Office of Nuclear Energy (DOE-NE) Integrated Energy Systems (IES) Program, led by Idaho National Laboratory (INL), is researching the effects the impact of increasing variability on grid reliability and generator profitability, and is also investigating the complementary role of non-electric applications of these generators. IES involve the design, integration, and coordinated operation of several complex, traditionally standalone systems. The control algorithms involved are unique to each application and component design. IES architecture can include process steam applications, thermal energy storage, and the presence of intermittent energy sources such as wind and solar, as illustrated in Figure 1.

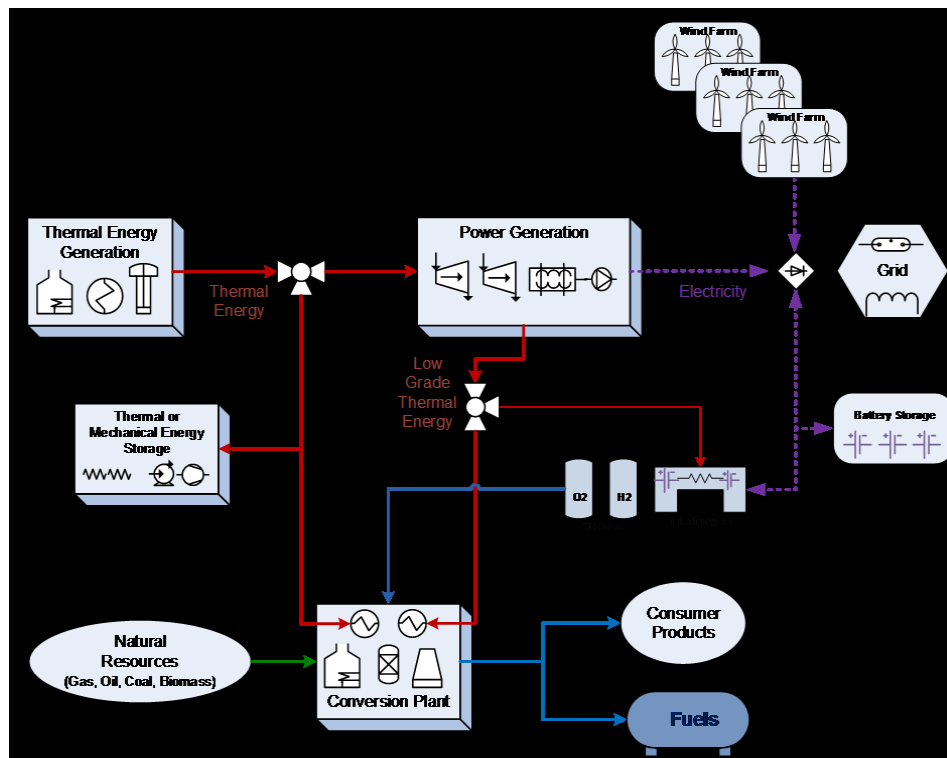


Figure 1. Example IES architecture, illustrating thermal and electrical interconnection to support hydrogen production and chemical conversion.

The goal of these systems is to operate as economically and efficiently as possible. For integrated energy parks that incorporate thermal storage, this means operating thermal generators at full power and storing excess energy during times of low total demand, then discharging that energy during times of high demand.

Since early 2013, to accommodate the vast array of possibilities introduced by integrated energy parks, the IES program team has been developing a library of high-fidelity process models in the Modelica modeling language [1]–[4]. Modelica is a non-proprietary, object-oriented, equation-based language for conveniently modeling complex physical systems. It is inherently time-dependent and enables the swift interconnection of independently developed models. As an equation-based modeling language that employs differential-algebraic equation

solvers, Modelica allows users to focus on the physics of the problem rather than on the solving technique, thus enabling faster model generation and, ultimately, analysis. This feature, alongside system flexibility, has led to widespread use of Modelica for commercial applications throughout the industry. System interconnectivity and the ability to quickly develop novel control strategies while still encompassing overall system physics is why INL chose to develop the IES framework in the Modelica language.

The dynamic physical models created in Modelica are a cornerstone of the IES program. These models are used to create system architectures and characterize the system inertia, thermal losses, and the efficiency of integrated systems. These physical models help map physical performance into economic performance, allowing for system-level optimization. In addition, the models are used to test innovative system-level control strategies for interconnected thermal generators. However, it is noted that, for real-world applications, it is not always practical to rewrite a model in Modelica; instead, interoperability with Legacy FORTRAN, C, *Python*, or other codes may be required.

To accomplish this, the IES Program is seeking to modify HYBRID, the existing physical modeling repository, to be consistent with the “plug-and-play” approach in Modelica/Dymola models using Functional Mock-Up Units (FMUs), Functional Mock-Up Interfaces (FMIs), and machine-learning techniques (see Figure 2). The final product will greatly enhance the physical modeling interoperability within INL’s Framework for Optimization of Resources and Economics ecosystem (FORCE) that is used to solve system/grid level optimization problems [5],[6].

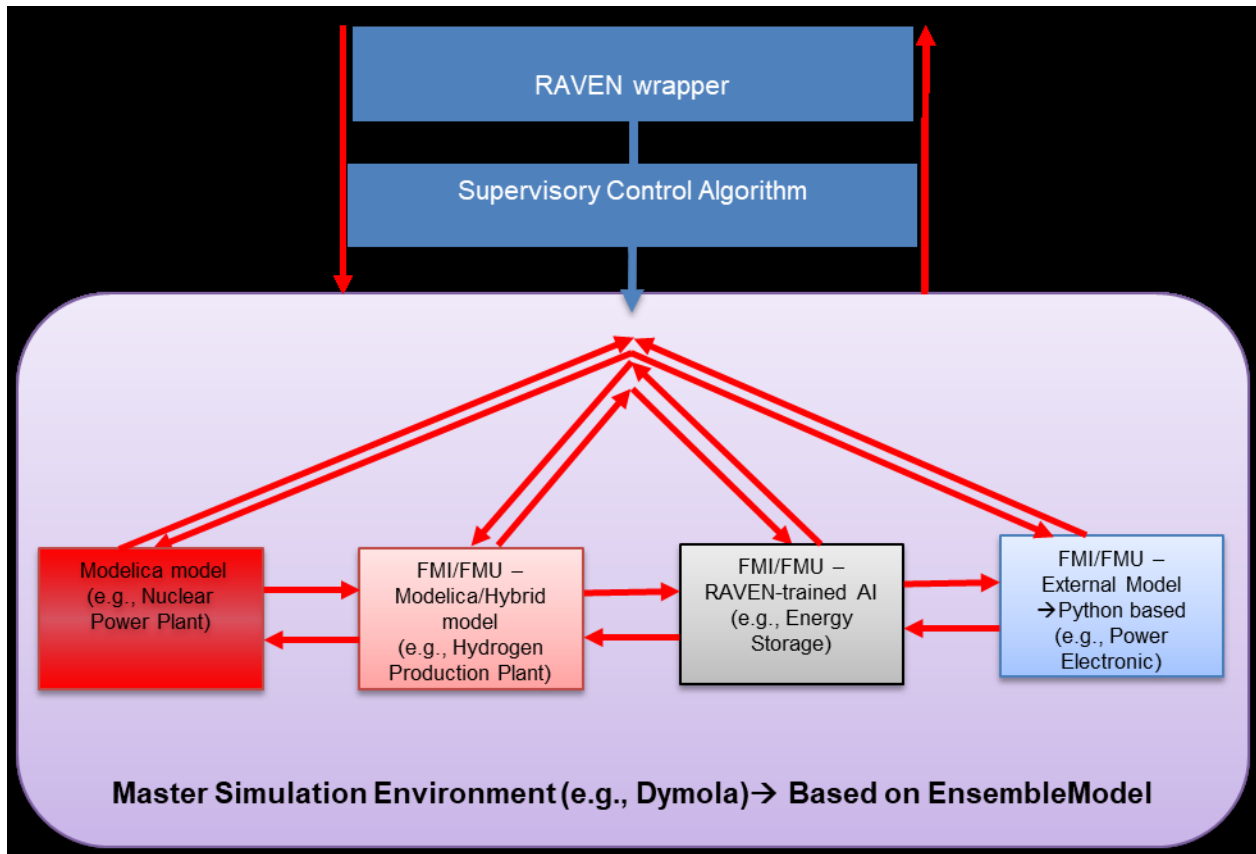


Figure 2. Plug-and-play framework environment.

This report summarizes the fiscal year (FY) 2020 efforts to create a plug-and-play repository of process models using the existing HYBRID repository, FMIs, FMUs [4], and the newly developed capabilities in the Risk Analysis and Virtual Environment (RAVEN) software for exporting artificial intelligence (AI)-based FMI/FMU models. The document characterizes and demonstrates the capabilities and improvements made to the previously-developed HYBRID repository of Modelica models for use as a software-quality-assured (SQA) plug-and-play system within FORCE.

The infrastructure of the GitHub repository that hosts the HYBRID repository was also enhanced. These improvements, described later in full detail, include the development (using the RAVEN-based ROOK regression system) of a Dymola output “differ” script for use with the commercially available Modelica-based modeling and simulation environment (i.e., a Dynamic Modeling Laboratory [Dymola] version 2021 FD01 [7]), inclusion of the Oak Ridge National Laboratory (ORNL) TRANSFORM library as an automatic submodule [8], creation of a user manual [9], and development of component-level regression tests for each Modelica model.

Extensive work was carried out on the deployment of methods for constructing RAVEN AI-based models compliant with the FMI/FMU standard. Such work represents the necessary initial development for deploying the “flexible ecosystem” (plug-and-play) concept, since it allows for replacement of high-fidelity Modelica models (or any other FMI/FMU-compliant model) with RAVEN-generated AI surrogate models. This capability enables the deployment of acceleration schemes for analyzing IES.

The Conclusions section of this report highlights the high flexibility achieved via the plug-and-play framework, possible shortcomings of the approach, and areas for further enhancement.

2. FUNCTIONAL MOCK-UP INTERFACES AND UNITS

This section briefly describes the FMIs and FMUs. As per the Modelon website (<https://www.modelon.com>): “FMI is an open standard for exchanging dynamical simulation models between different tools in a standardized format.”

FMIs were first introduced by Dassault Systems under the name MODELISAR in 2008. FMIs define a standardized interface for use in computer simulations to develop complex cyber-physical systems. Additionally, FMIs/FMUs can be exported as binary files, enabling industry partners to exchange and simulate proprietary information safely and securely, without potential information leakage.

The FMI standard describes an open format for exporting and importing simulation models using a common data exchange nomenclature. In other words, the FMI standard allows the user to retain the same model while selecting the tools best suited for each type of analysis.

In order to be executed, an FMI is always “shipped” with an FMU. An FMU is the executable that implements the FMI. During exportation of an FMU, an FMU archive is generated from a systems model, whereas during an FMU import, a systems model is generated from an FMU archive.

FMUs contain the following:

- ***A model description XML file:*** This file contains information about the model (e.g., variable definitions: type, unit, description, etc.) and other more general model information, such as model name, generation tool, and FMI version.

- **Model equations:** A model can be described using ordinary differential equations, algebraic relations, and discrete equations—including time, state, and step events. These equations can in turn be represented by a small set of *C* functions. The *C* code is then distributed in the FMU in source and/or binary form, and one FMU can contain binaries for more than one platform and/or platform version.
- **Optional resource files:** Other optional files might be included in the FMU, such as documentation files (HTML), model icons (bitmap files), maps and tables, and other libraries or dynamic link libraries (DLLs) used in the model.

The FMI/FMU standard currently specifies two types of protocols:

- FMI/FMU for model exchange (import and export)
- FMI/FMU for co-simulation (master and slave).

The main difference between these two protocols is that, in model exchange, the FMU is simulated using the importing tool's solver, whereas in co-simulation, the FMU is shipped with its own solver.

The FMI for model exchange allows FMUs to be used in offline or online simulation—with several FMUs potentially being connected—or in embedded control systems on microprocessors.

2.1 Co-simulation

Figure 3 shows the information flow and scheme of FMIs/FMUs in a co-simulation configuration. The co-simulation (CS) configuration is characterized by:

- Standalone black-box simulation components
- Data exchange being restricted to discrete communication “checkpoints”
- Between two consecutive communication checkpoints, the system model is solved by its internal solver.

In summary, the goal of a co-simulation operation is to individually compute the solution of time-dependent coupled systems and have them communicate back and forth at predetermined time steps, Δt , known as communication steps (or checkpoints). The simulation is independently performed between all the subsystems, and at each Δt there is a communication and transfer of boundary conditions between subsystems. Because of the independent nature of these subsystems, an FMI for co-simulation is the easiest method to implement. However, due to the different solver types and the need to specify Δt , the scheme between systems becomes fully explicit. Being fully explicit, it is crucial to identify a small enough Δt to ensure system stability. This step size limitation ultimately reduces the simulation speed.

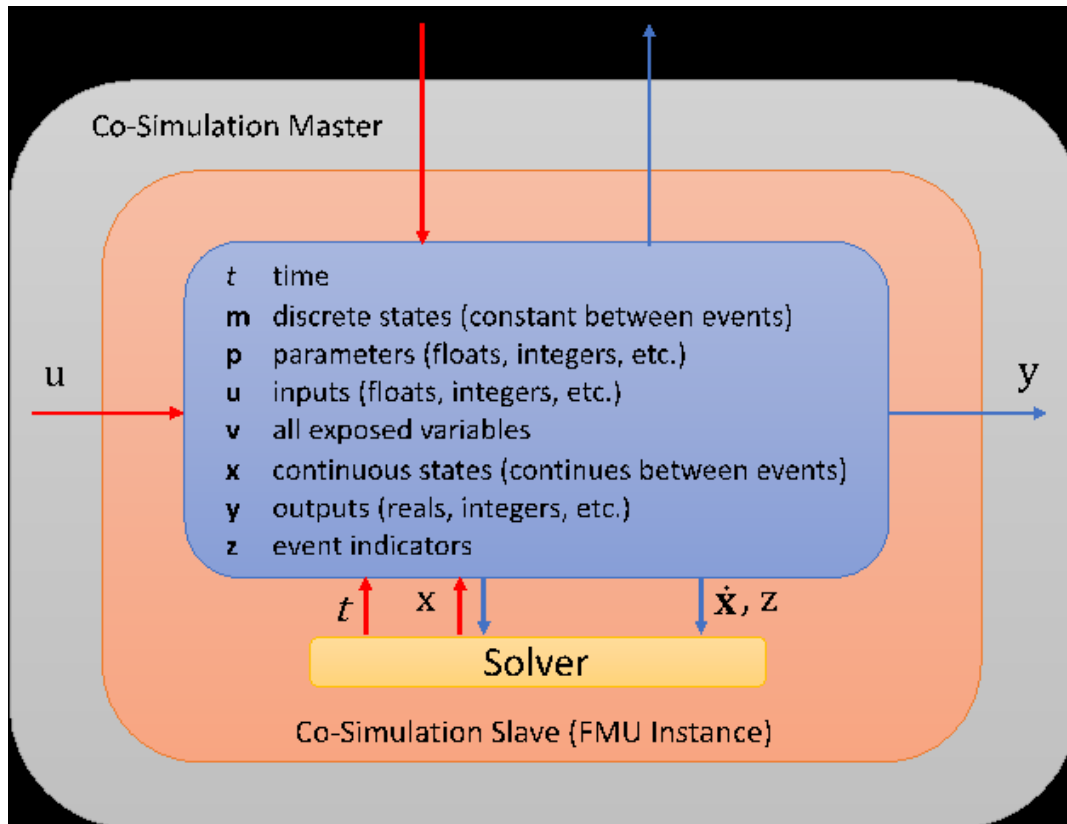


Figure 3. Co-simulation FMI/FMU scheme.

2.2 Model Exchange

Figure 4 shows the information flow and scheme of an FMI/FMU in a model exchange (ME) configuration. As shown in the figure, the model exchange configuration can be described as having the following characteristics:

- Standardized access to model equations
- Models described by algebraic, differential, and discrete equations
- Monitoring of time, state, and step events
- Models that must be solved using solvers provided by the embedding environment.

In summary, in a model exchange FMI/FMU, the numerical solver is supplied by the importing tool. The FMU provides functions to set the state/inputs and compute the state derivatives. The solver in the importing tool will determine what time steps to use and how to compute the state at each subsequent time step.

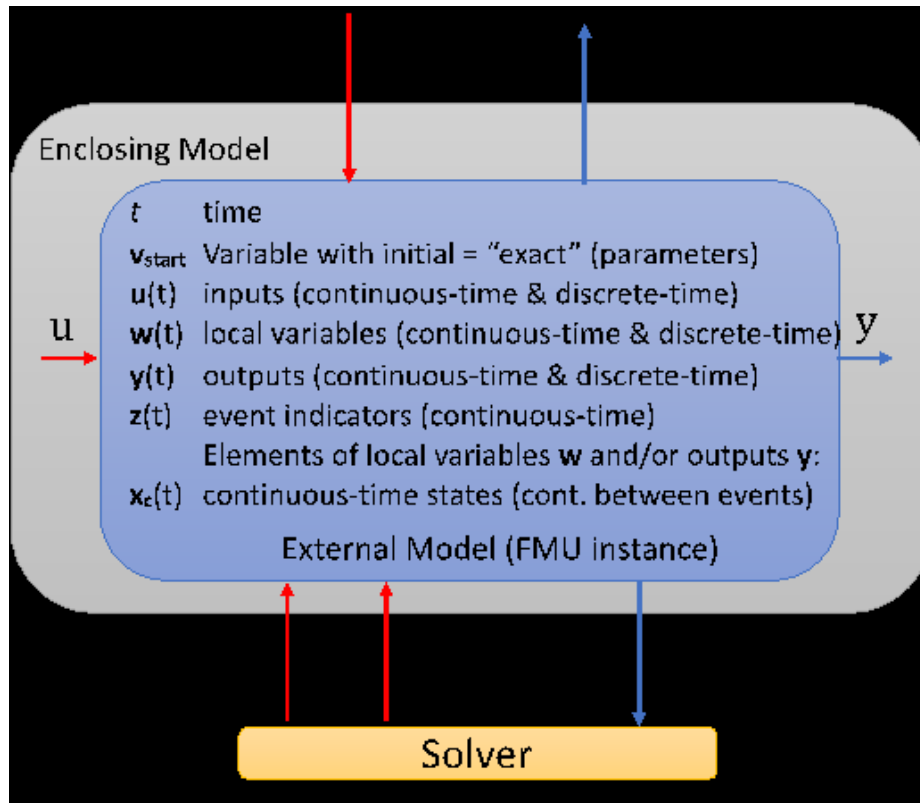


Figure 4. Model exchange FMI/FMU scheme.

2.3 Advantages of Each Protocol

Each of the two protocols described in the previous section, namely CS and ME, offer certain advantages.

Co-simulation

1. Not all tools support both protocol types. Support for CS is more common than for ME.
2. The numerics of the model may require a specific solver available in the exporting tool but not in the importing tool.
3. The FMU may represent a sampled data system (e.g., signal processing or control algorithms) not governed by differential equations and therefore more naturally expressed as a co-simulation FMU.
4. The exporting tool may have a more efficient implementation of the solver than the importing tool.

Model exchange

1. An explicit scheme is avoided, since the entire solve is done simultaneously.
2. Dynamic time stepping is allowed.
3. The importing tool could have a more efficient implementation of the solver than the exporting tool.

3. MODELICA TO FMU ADAPTATION

Modelica is a physical modeling language that relies on an acausal (rather than causal) assignment of equations. This means that an equation can only appear once, and that the translator and system solvers will determine the proper way to assign the flow of information. In addition, since Modelica is a physical modeling language, there are the assignments of special variable containers “flow” and “stream” that have an inherent physical representation in the code. Flow variables have a direction and must sum to zero in a “connection.” The “stream” qualifier is used to qualify when a given element in a connection has an intensive property flowing through a connector. These “connectors” include a singular flow variable with several stream variables alongside it. For example, a “fluid port” is a connector that has the mass flow rate as the “flow” variable and enthalpy as the “stream” variable. Mass flow is what physically goes through the connector, while enthalpy is a property of the mass flow. This nuance in variable types is particularly important when considering the translation of Modelica models into FMIs and FMUs. FMIs can only import and export real input/output signals. These signals cannot retain the physical properties seen in Modelica, thus requiring special adaptors to translate them back into physical values for use in other Modelica models.

3.1 Adaptors

For connections between FMIs and other Modelica models within the Dymola platform, a set of standardized variables and adapters are needed to properly transmit energy values among subsystems. This is particularly true if the interconnection is between two physical models, such as a nuclear power plant and a turbine. This is because the physical models contain “ports,” as shown in Figure 5.

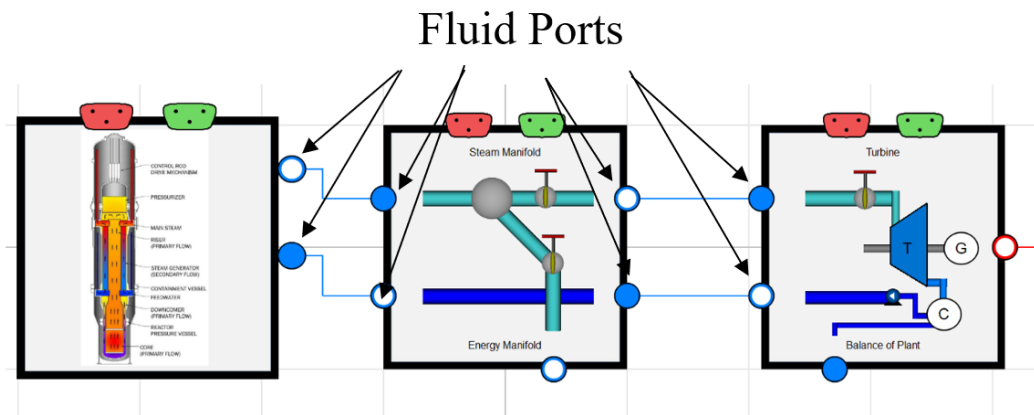


Figure 5. Fluid ports (note that “ports” are a container method in Modelica used to transfer several pieces of physics-based information within a single “connector”).

Each fluid port contains:

- Mass flow (**flow** variable), m_flow
- Conditional enthalpy (**stream** variable), $h_outflow$
- Pressure, P
- Trace substance fraction (**stream** variable), C_i
- Mass fraction (**stream** variable), X_i .

Each electric port contains:

- Power (**flow** variable), W
- Frequency, f

To properly transition from ports to input and output signals, the individual components of the ports must be separated out and assumed to be either an input or an output. This is illustrated in Figure 6, with each fluid port being separated into its five constituent pieces (mass flow, enthalpy, pressure, mass fraction, trace substance fraction), and the electric port being separated into its two constituent parts (power and frequency).

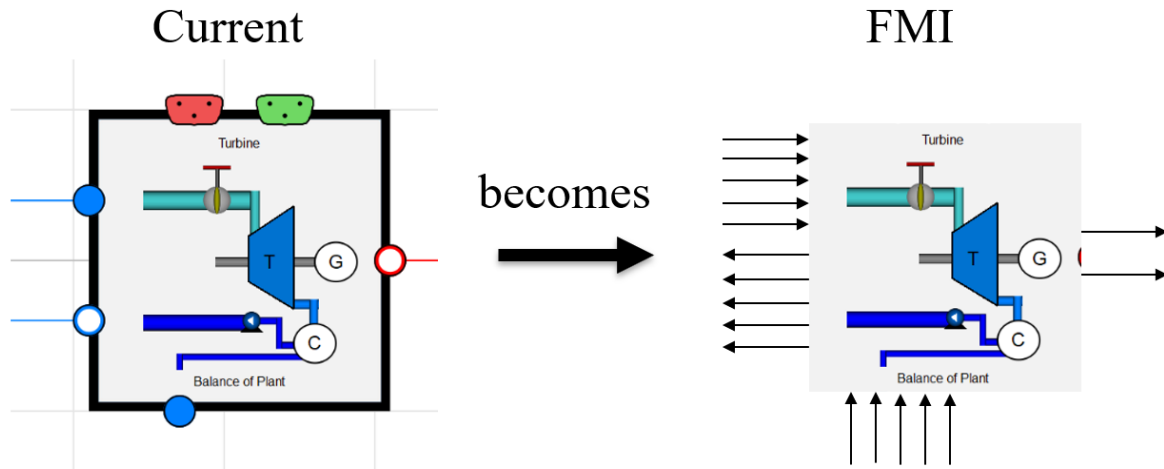


Figure 6. Transition from a Modelica physical model into an FMU.

In the HYBRID repository package structure, a set of adaptors was created and added to the utility folder to enable users to convert an existing Modelica model into a model ready for export as a FMU. The package placement is seen in Figure 7. Further details on each FMI template and interface are outlined in the next section.

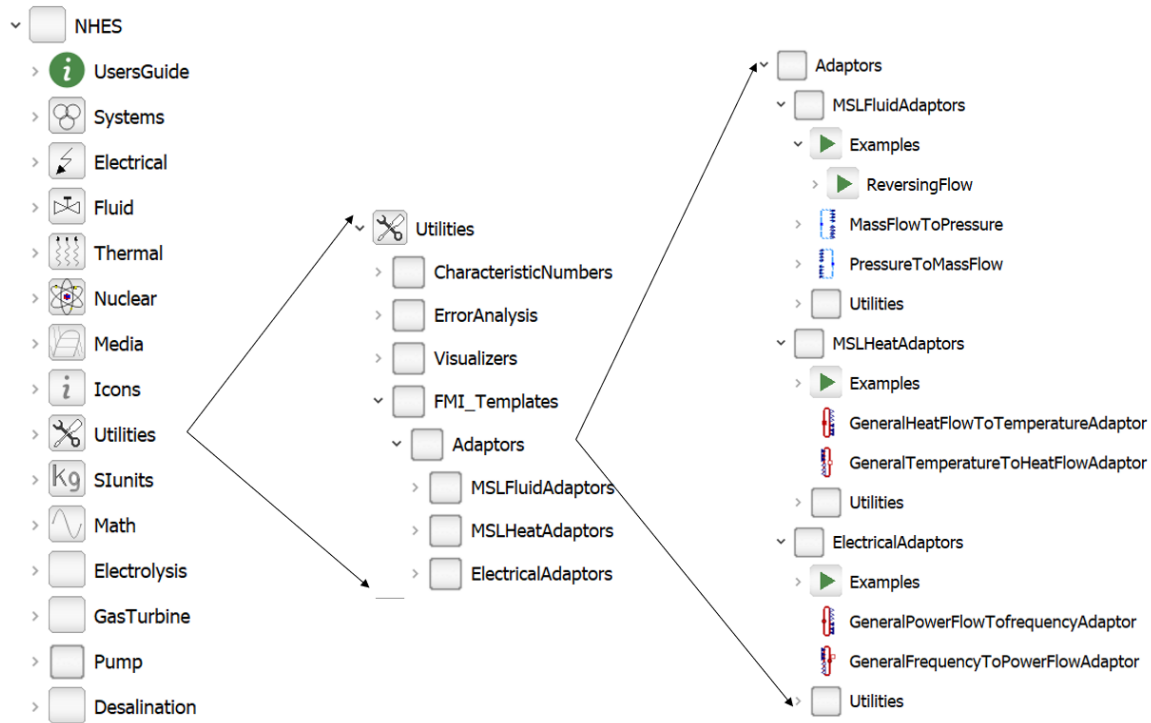


Figure 7. FMU template folder location within the larger Nuclear Hybrid Energy Systems (NHES) folder as part of the HYBRID Repository.

Fluid Port Adaptors

Within the Utility.FMI_Templates folder is an adaptor package created specifically for Modelica standard library fluid adaptors. This package is called MSLFluidAdaptors, and it models acausal to causal adaptors. This folder was created in unison with Modelon. Within this folder are two adaptors, shown in Figure 8. One is a “pressure to mass flow” adaptor, aptly named PressureToMassFlow. This adaptor’s fluid port is best connected to a flow port of some sort (e.g., valves, resistance, pipe model). The inputs to this model are the pressure at the interface, upstream enthalpy from the causal side, upstream mass fraction from the causal side, and upstream trace composition from the causal side. The outputs are the acausal mass-flow rate, upstream enthalpy from the acausal side, upstream mass fraction from the acausal side, and upstream trace composition from the acausal side.

The second adaptor, called the MassFlowToPressure adaptor, is a “mass flow to pressure” adaptor. This adaptor’s fluid port is best connected to a volume port (e.g., pressure sink, tank model). The inputs to this model are the causal mass-flow rate, upstream enthalpy from the causal side, upstream mass fraction from the causal side, and upstream trace composition from the causal side. The outputs are the pressure at the interface, upstream enthalpy from the acausal side, upstream mass fraction from the acausal side, and upstream trace composition from the acausal side.

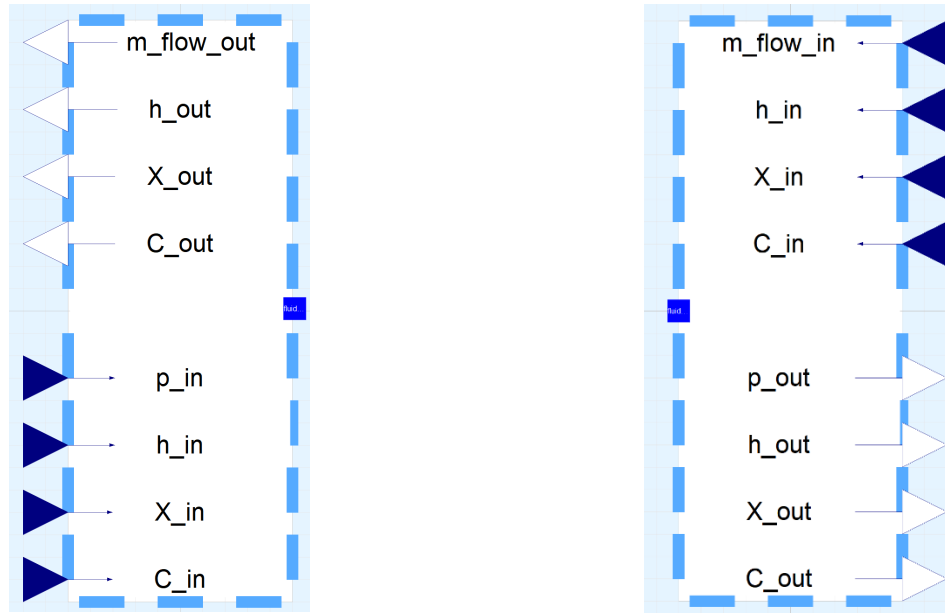


Figure 8. (Left) PressuretoMassFlow adaptor. This adaptor is best connected to a resistance port able to set the output mass flow rate. (Right) MassFlowtoPressure adaptor. This adaptor is best connected to a volume port able to set the output pressure. Note: Both of these adaptors were created by Modelon for use in the INL plug-and-play framework as part of an FMI/FMU course subcontract.

Figure 9 illustrates the usage of the two adaptors on a single model involving reversible flow. The model is of a series of two fully open valves connected to a volume source positioned between them, and a pressure source on either side of the valves. The system fluid is moist air from the Modelica standard library. The pressure source is then subjected to a 1 Hz oscillatory frequency on the pressure system, as would be present in a fast-moving pressure chamber, while the pressure sink remains at a constant pressure. In normal operations, this system will have a reversible flow, as the pressure of the source oscillates about the pressure sink's pressure. Such scenarios have been challenging to meet with FMIs and FMUs, due to the reversible nature of the mass flow. With the new adaptors, this reversible flow issue can be met.

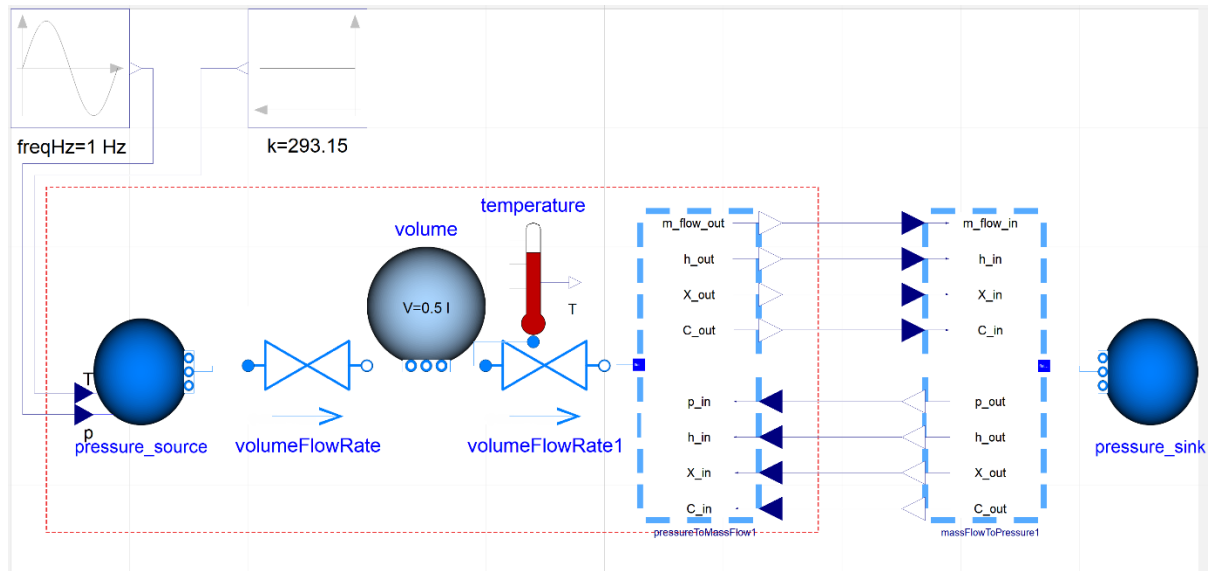


Figure 9. An example (using adaptors) involving two pressure sources using moist air, one of which oscillates in pressure, causing a mass flow reversal. The unit in the red box will become an FMU. (k=Temperature input to pressure flow boundary; freqHz = pressure frequency oscillation placed on pressure_source).

The unit inside the red box in Figure 9 was exported as both a model exchange and co-simulation FMU, as shown in Figure 10 and Figure 11. All systems were then run for 10 seconds of simulation time. The results are depicted in Figure 12.

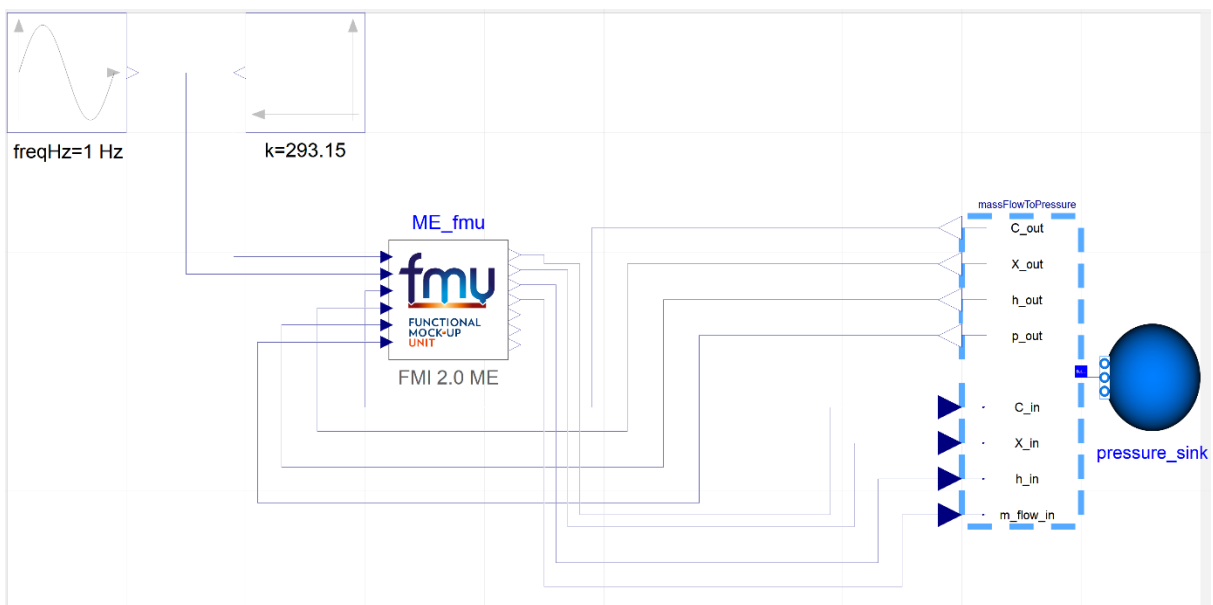


Figure 10. Example of a reversible flow using two pressure sources, moist air, and a model exchange FMU. (k=Temperature input to pressure flow boundary; freqHz = pressure frequency oscillation placed on pressure_source).

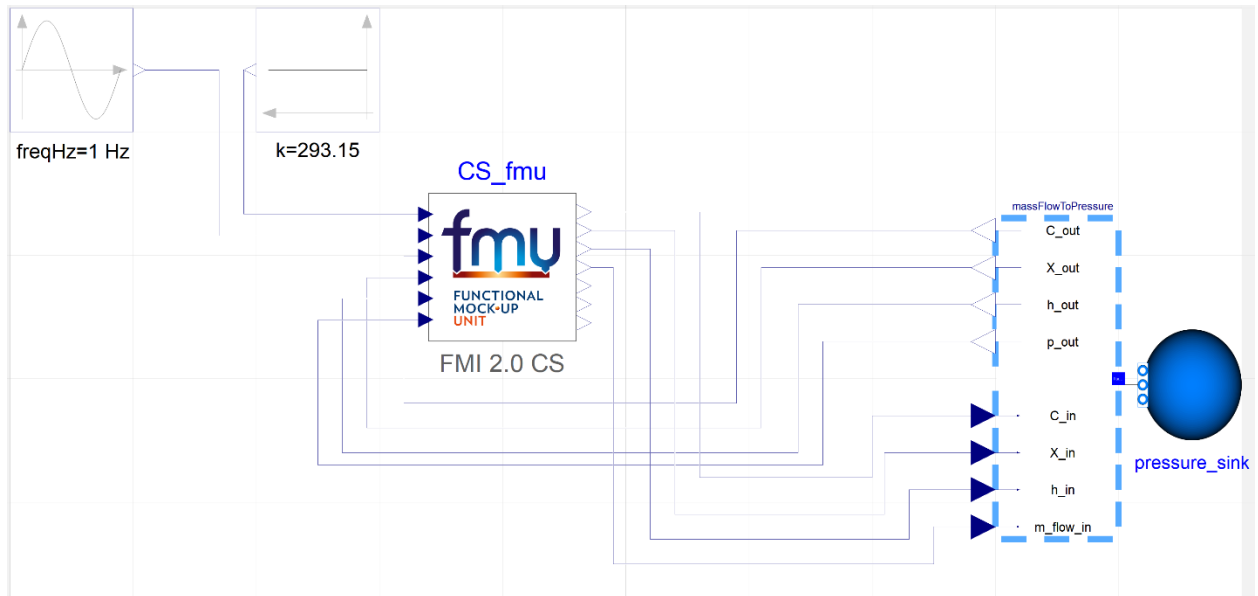


Figure 11. Example of a reversible flow using two pressure sources, moist air, and a co-simulation FMU. (k =Temperature input to pressure flow boundary; freqHz = pressure frequency oscillation placed on pressure_source).

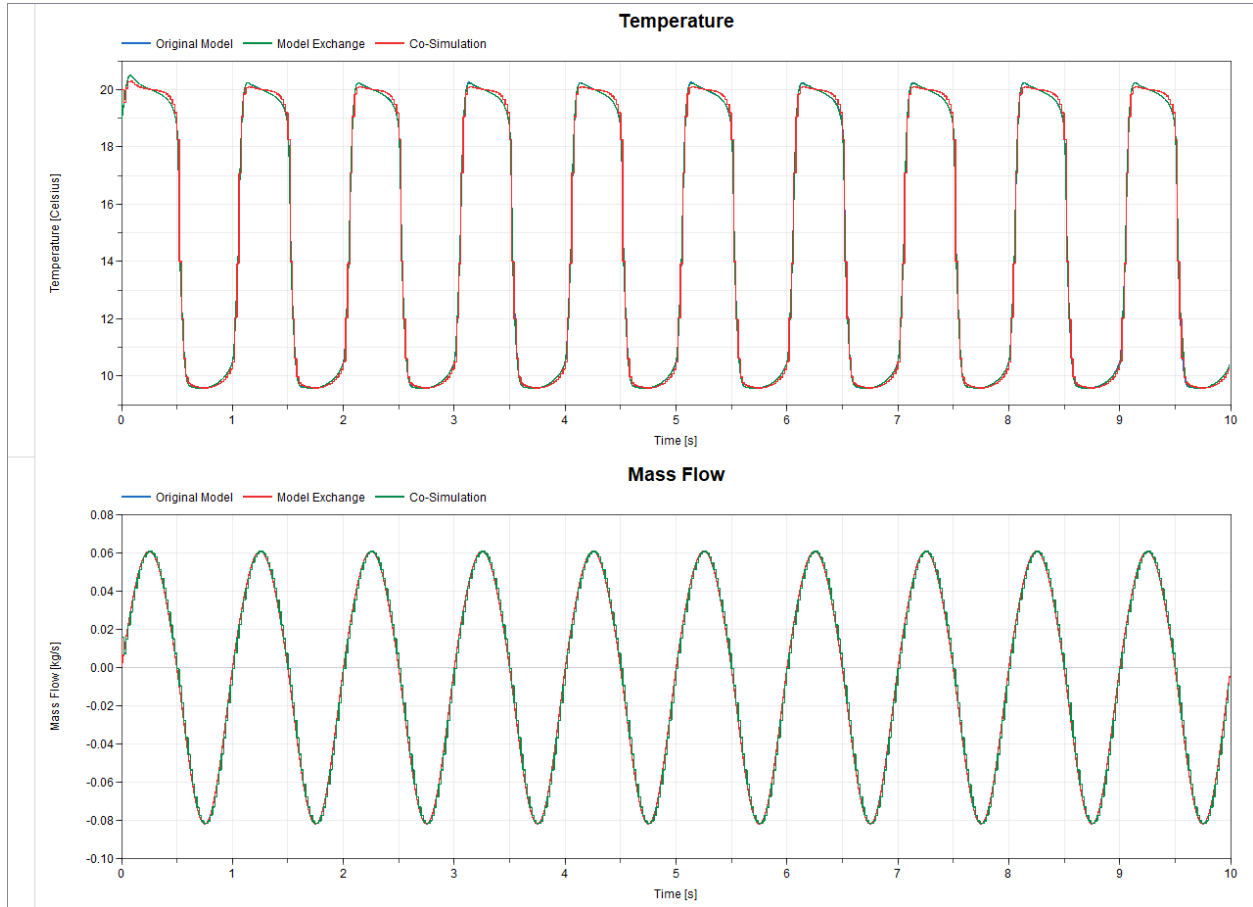


Figure 12. Comparison of mass flow to the pressure sink across the original model, model exchange FMU, and co-simulation FMU (timestep = 0.02 seconds).

The results showcase that, by utilizing the fluid port adaptors, reversible flow is achievable in both the model exchange and co-simulation FMIs/FMUs. However, these capabilities carry additional overhead in regard to central processing unit (CPU) time. Model exchange for this particular model increases the simulation time from 2.364 to 6.749 seconds. Co-simulation with a 0.02-second communication interval took 10.795 seconds. Even so, co-simulation still shows the largest error, due to co-simulation models inherently being an explicit solve. However, given a sufficiently small communication interval, and depending on the dynamics of the model, an acceptable solution can be achieved.

Thermal Port Adaptors

Within the `Utility.FMI_Templates` folder is an adaptor package created specifically for Modelica standard library thermal adaptors. This package is called `MSLHeatAdaptors`, and it models acausal to causal adaptors. These models were initially made available in the Modelica standard library and have been augmented with additional examples and placed within the NHES package for ease of access relative to other FMI adaptors. Two adaptors are included, one being the `GeneralHeatFlowToTemperature` adaptor. The inputs to this adaptor are the acausal heat flow port, causal heat flow, and optional causal first and second derivatives of heat flow. The outputs are the temperature and the optional first and second derivatives of temperature.

The second adaptor is the GeneralTemperaturetoHeatFlow adaptor. The inputs to this adaptor are the acausal heat flow port, causal temperature, and optional causal first and second derivatives of temperature. The outputs are the heat flow and the optional first and second derivatives of heat flow.

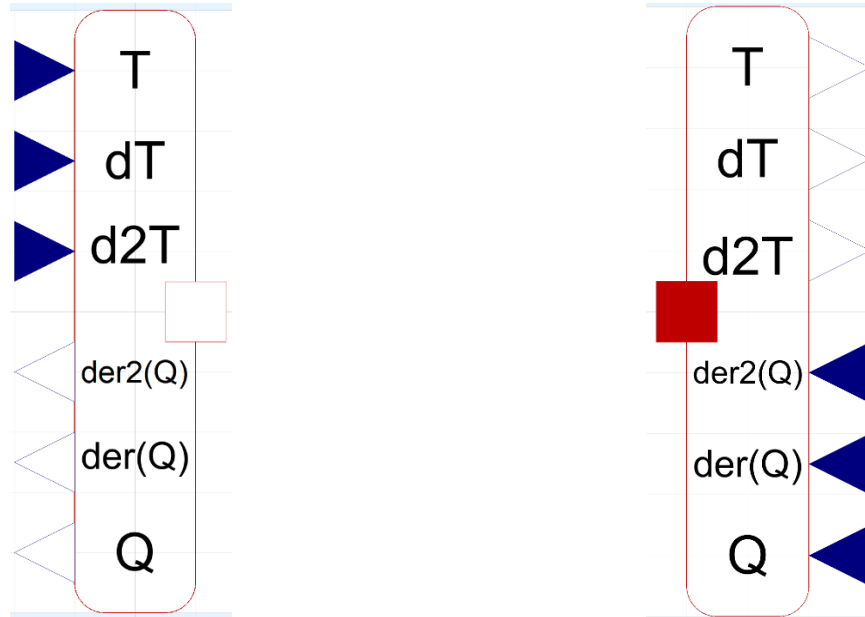


Figure 13. (Left) GeneralTemperatureToHeatFlow adaptor for use in the INL plug-and-play framework. (Right) GeneralHeatFlowToTemperature adaptor for use in the INL plug-and-play framework. (T=temperature, dT = first derivative of temperature, d2T = second derivative of temperature, Q = heat flow, der(Q) = first derivative of heat flow, der2(Q) = second derivative of heat flow.) Note: only T and Q are required the derivative values are optional for stability.

Figure 14 illustrates the usage of the two adaptors in a single model involving two methods of heat port usage. The upper model demonstrates how to export two heat capacitors and connect them together in a target system. This requires that one of the capacitors (here, DirectCapacity) be defined to have states, and that the temperature and derivatives of the temperature are provided in the interface. The other capacitor (here: InverseCapacity) requires a heat flow in accordance with the provided input temperature and derivative of temperature. The lower part demonstrates how to export a conduction element that only requires temperatures for its conduction law, and connects this conduction law to both the heat capacitors in a target system. Both models will be translated into a model exchange and co-simulation model, as shown in Figure 15, Figure 16, Figure 17, and Figure 18. The results are compared in Figure 19 and Figure 20.

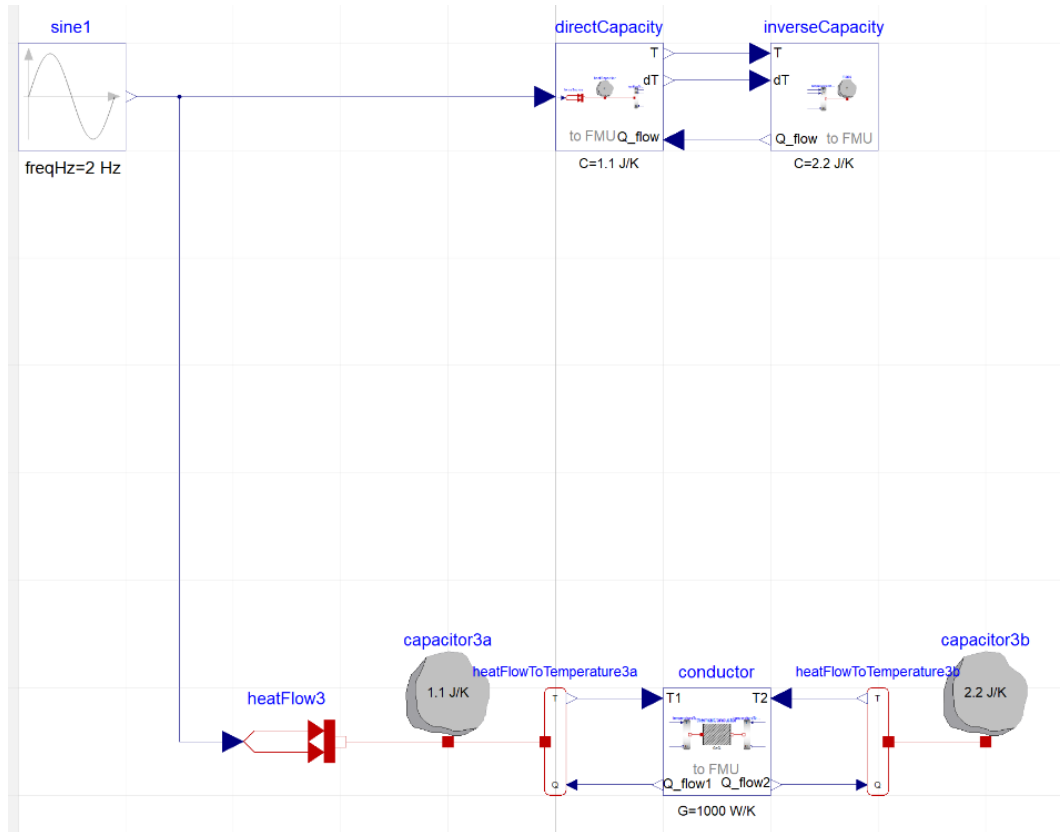


Figure 14. Example meant to demonstrate the FMU variants available with the thermal FMU adaptors. The upper part demonstrates how to export two heat capacitors and connect them together in a target system. The lower part demonstrates how to export a conduction element that only requires temperatures for its conduction law, and connects this conduction law to both heat capacitors in a target system.

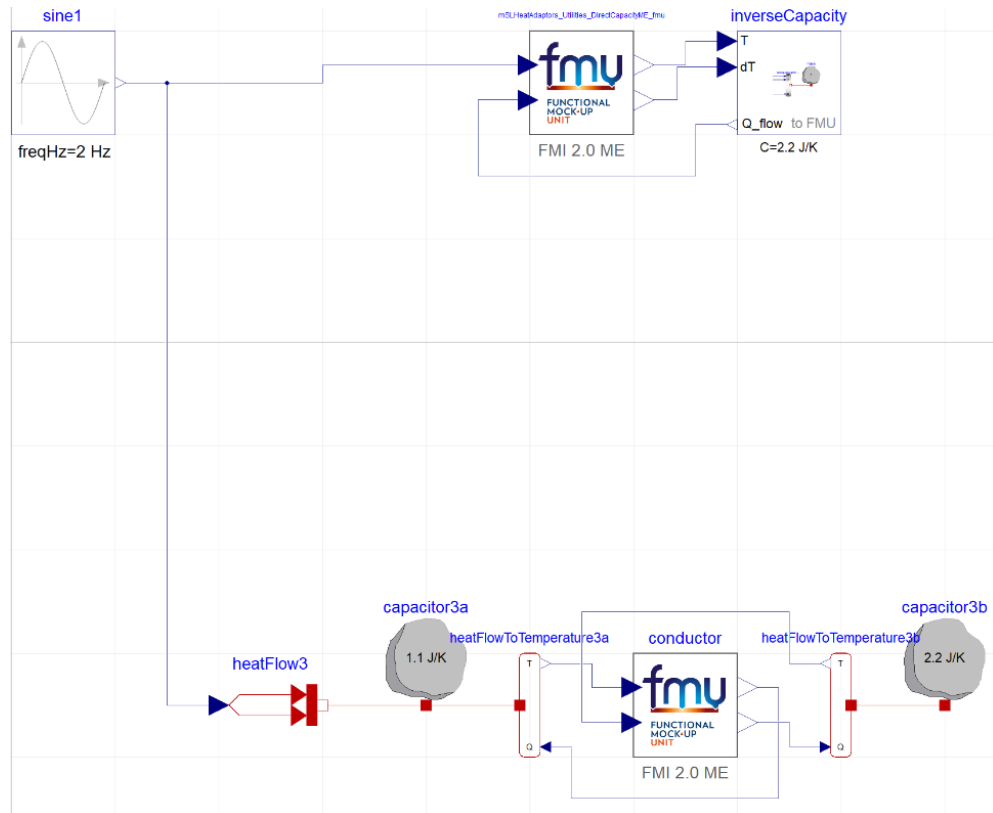


Figure 15. Demonstration of an FMU variant example that uses model exchange FMUs for the thermal heat port adaptors.

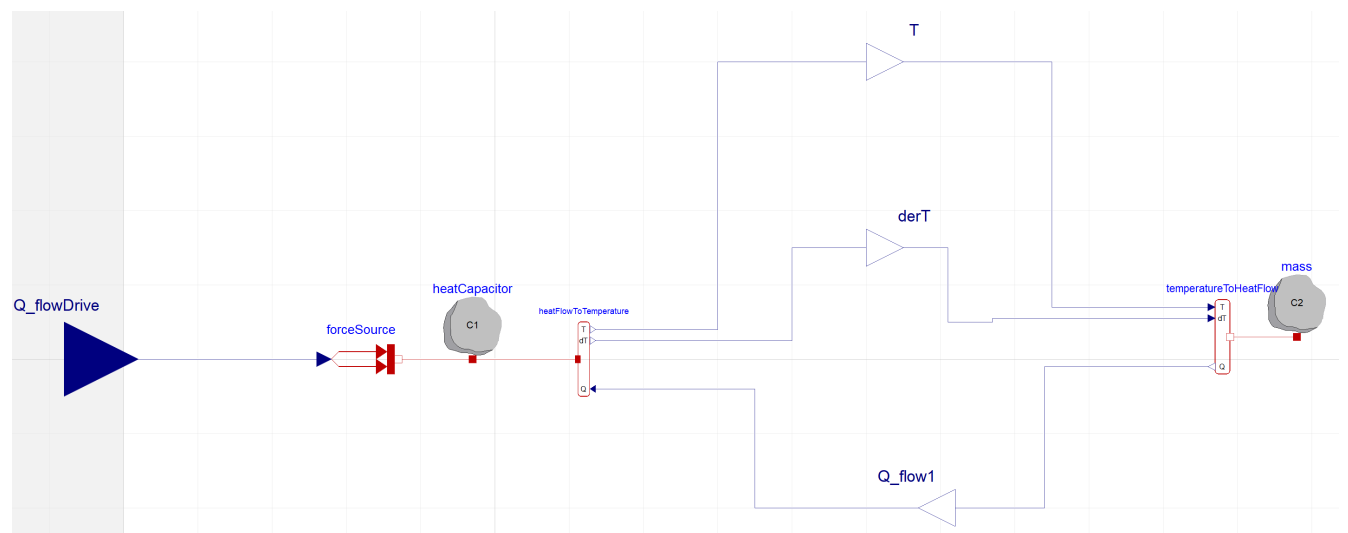


Figure 16. Collapse of the upper part of Figure 14 into a single FMU for co-simulation. This is required because the frequency between the direct and inverse conduction problem is so fast that a single cut between the two could not be made without instabilities occurring.



Figure 17. Upper model of Figure 14 connected with the combined direct/inverse co-simulation FMU.

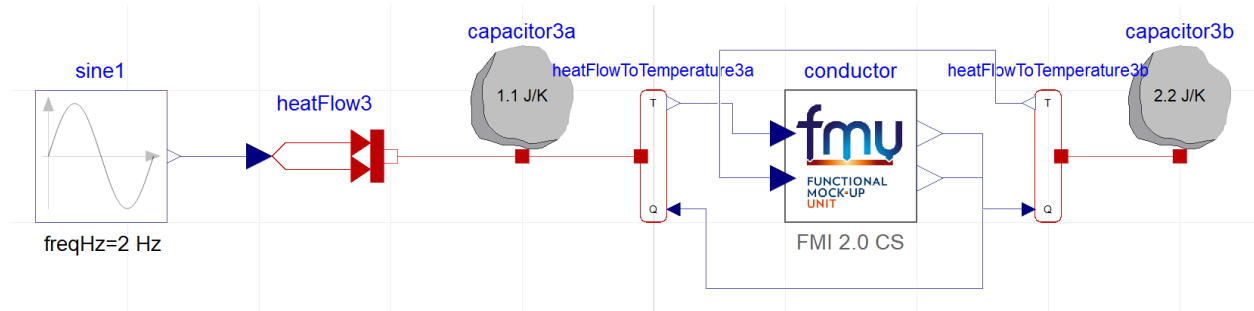


Figure 18. Lower model of Figure 14, co-simulation FMU.

The results showcase that, by utilizing the thermal adaptors, acceptable results in terms of the heat flow between models can be achieved via both model exchange and co-simulation FMIs/FMUs. However, these capabilities carry additional overhead in regard to CPU time, as was the case in the fluid port scenario. The co-simulation mode, though theoretically easier to export to external codes thanks to its inclusion of a solver, required the most augmentation, due to the fast system dynamics. This limitation required the FMU to include both the direct and inverse capacitors within a singular model, as shown in Figure 16, otherwise a divergent solution was acquired. Even with this additional step, the co-simulation solve still showed the largest error, as depicted in Figure 19 and Figure 20. This is because co-simulation models are inherently an explicit solve. However, given a sufficiently small communication interval and depending on the dynamics of the model, an acceptable solution can be achieved.

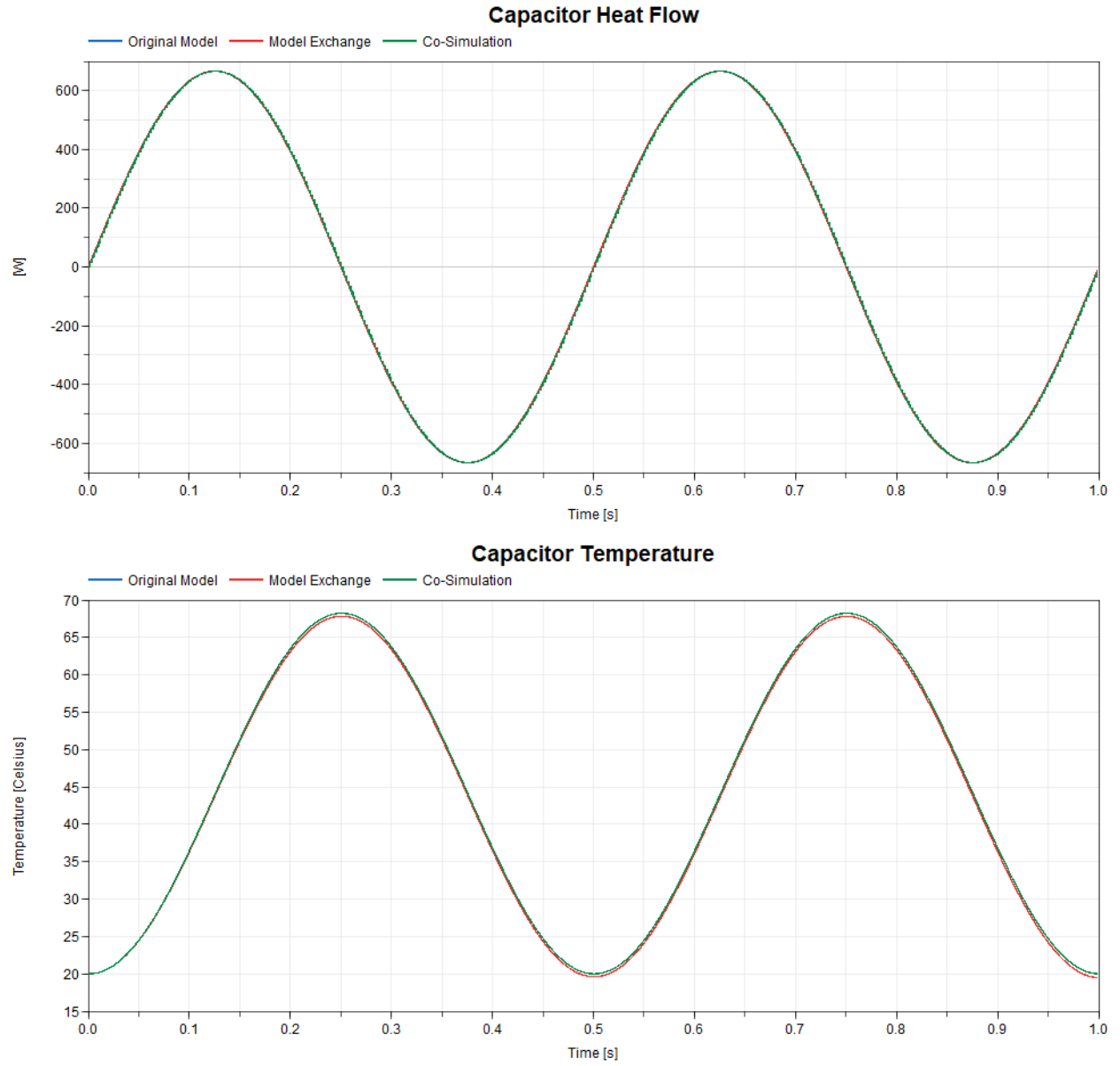


Figure 19. Direct/inverse simulation results for the original, model exchange, and co-simulation (communication interval: 0.002 seconds) FMU runs. (Top) Heat flow into the capacitor. (Bottom) Capacitor 3b temperature.

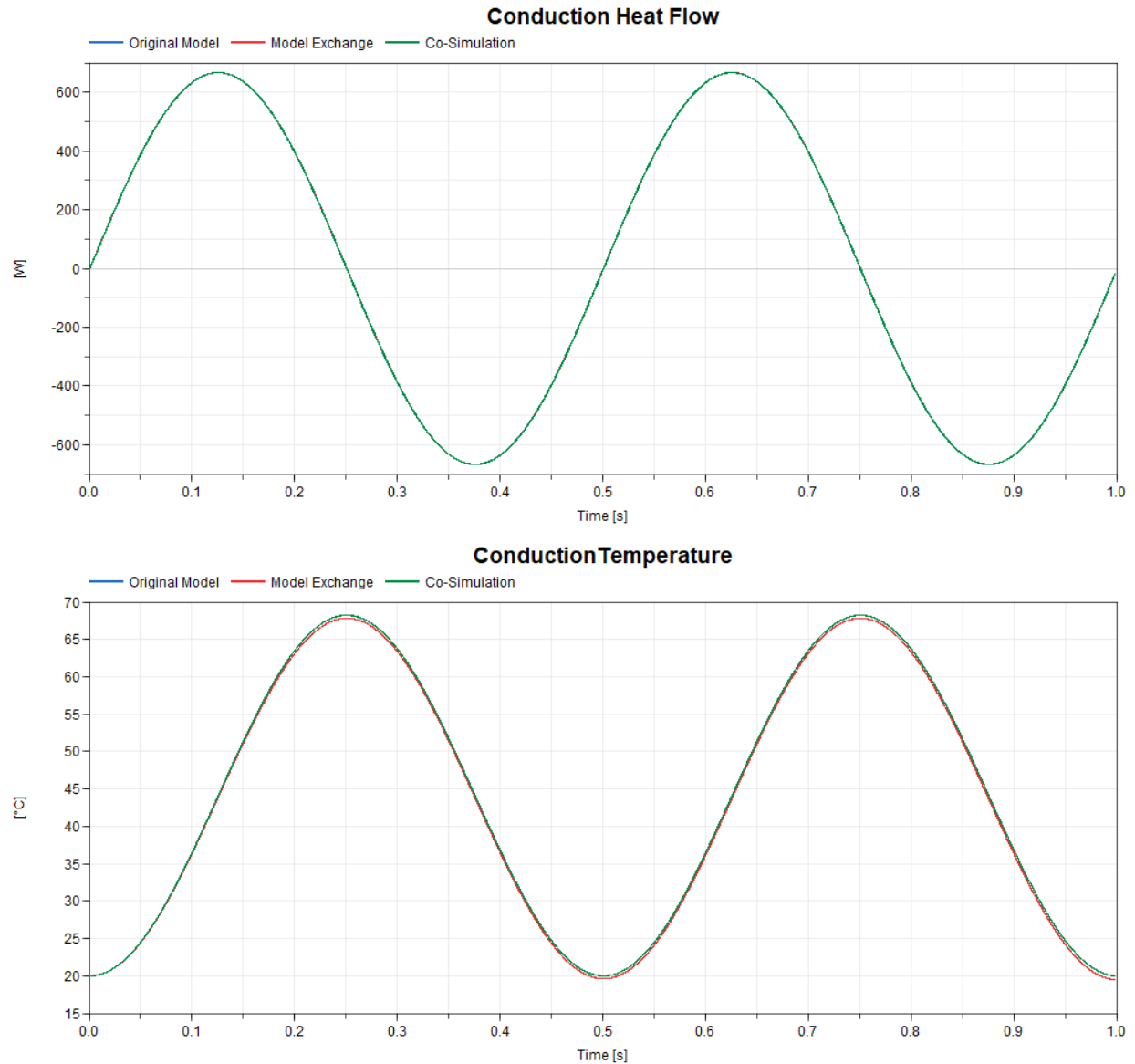


Figure 20. Conduction (lower model) simulation results for the original, model exchange, and co-simulation (communication interval: 0.002 seconds) FMU runs. (Top) Heat flow into the capacitor. (Bottom) Capacitor 3b temperature.

Electrical Port Adaptors

Within the `Utility.FMI_Templates` folder is a package created specifically for electrical adaptors. This package is called `ElectricalAdaptors`, and it models acausal to causal adaptors. Two adaptors are included (see Figure 21), one being the `GeneralPowerFlowToFrequency` adaptor. The inputs to this adaptor are the acausal electrical port, causal power, and optional causal first and second derivatives of power. The outputs are the frequency and the optional first and second derivatives of frequency.

The second adaptor is the `GeneralFrequencyToPowerFlow` adaptor. The inputs to this adaptor are the acausal electrical port, causal frequency, and optional causal first and second derivatives of frequency. The outputs are the power flow and the optional first and second

derivatives of power flow.

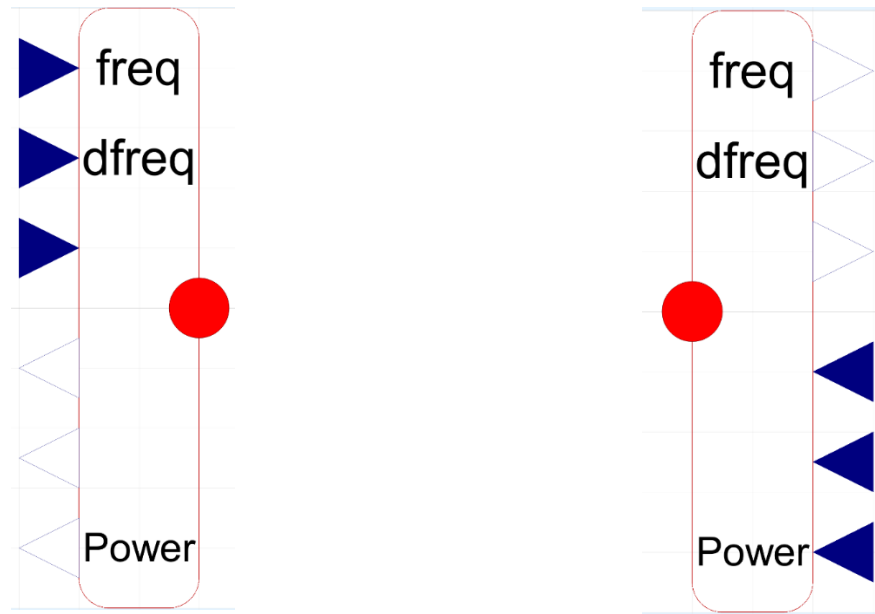


Figure 21. (Left) GeneralFrequencyToPowerFlow adaptor for use in the INL plug-and-play framework. (Right) GeneralPowerFlowToFrequency adaptor for use in the INL plug-and-play framework. (Red circle represents the electrical port, with inputs and outputs equal to the aforementioned variables in the section).

3.2 FMI Construction Guide

To properly create and utilize a model as an FMI/FMU, the following five steps must be accomplished.

1. Model Preparation
2. Adaptors
3. Export
4. Import
5. Simulation

This section seeks to provide step-by-step guidance on how each of these steps can be accomplished.

Model Preparation

For a model to become a usable FMU, it must contain all the required input/output variables within itself, aside from those designated to come from an outside model. This requirement means that, for units featuring interchangeable control systems, a particular control system must be declared via a top-level declaration in an example style file. An example of both an incorrect and a correct file format for a natural gas peaking turbine are shown in Figure 22 and Figure 23, respectively. In Figure 22 the natural gas turbine model includes the basic constituent parts for its simulation (compressor, turbine, combustion chamber, inertial generator shaft, generator, fuel controllers, and geometrical data assumptions). But in Dymola if this model is run it is missing a selected control system for the sensor and actuator bus as this is a “replaceable” component. Meaning the model needs to be imported within a new model to allow us to select the control system.

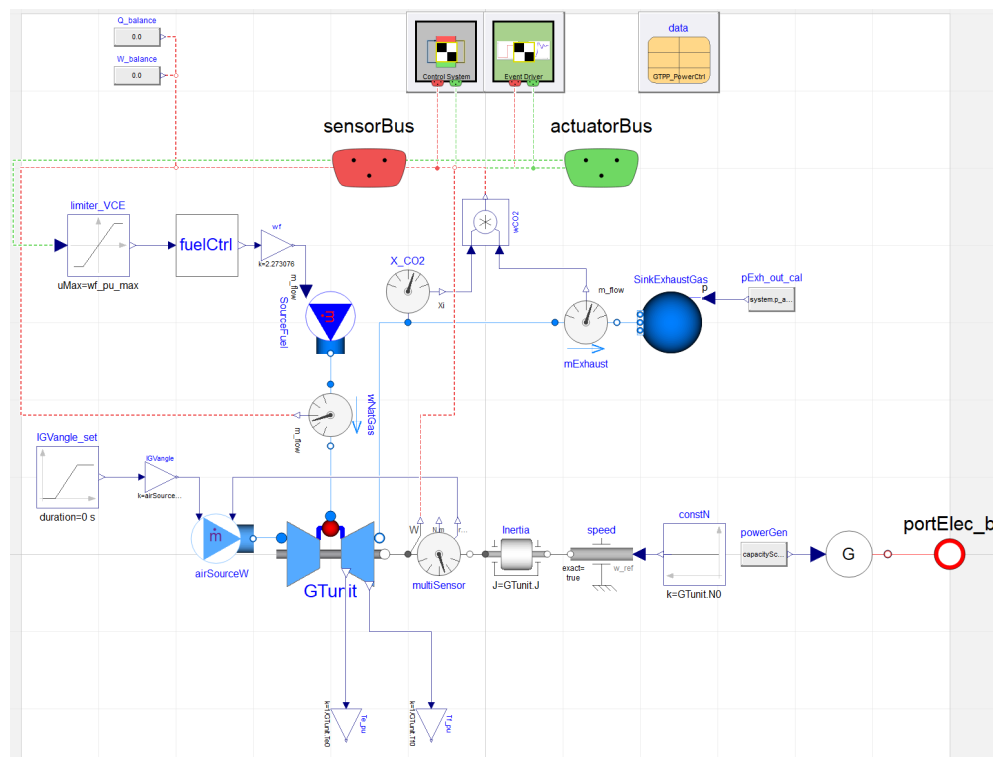


Figure 22. Incorrect level for proper export as FMI/FMU. Control system has not been declared and is replaceable from a higher level within the HYBRID repository.

This placement within a new model is shown in Figure 23. In this case the model from Figure 22 is placed within a new model, the electric port is attached to a frequency boundary condition and if we double click the natural gas turbine icon the table on the left pulls up where the control system can be selected and values can be imported for system size and maximum power output. Once this control system is selected the model is now ready to begin model preparation for FMI/FMU exportation. Note: the control system selected will be the control system exported with the FMI/FMU.

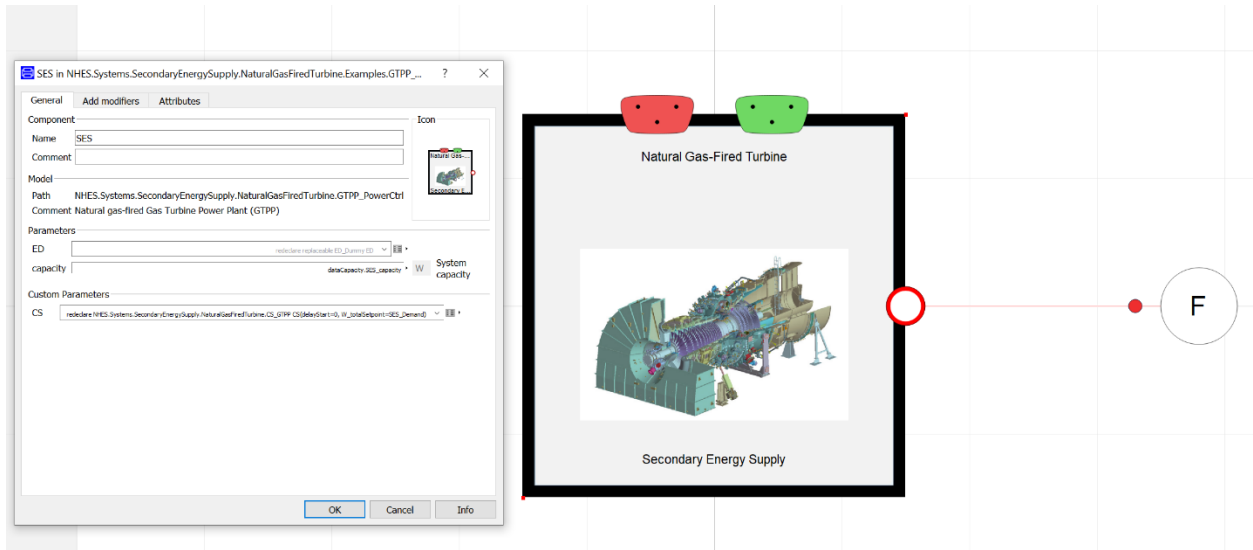


Figure 23. Correct level from which to begin FMI/FMU preparation. Control system has been selected via the drop-down menu available in the custom parameters section, shown on the left. (Red dots are electrical flow ports).

Adaptors

Now that the proper model has been created, the variables designated to come from outside the FMU must be declared as a real “input” or “output” variables, as demonstrated in Figure 6. To accomplish this, the adaptors can be employed in the manner previously outlined. For the natural gas turbine example illustrated in Figure 23, the electric port must be converted into real inputs/outputs using the PowerFlowToFrequency adaptor described in the previous section. In addition, the control system of the natural gas turbine requires a top-level demand signal to communicate the grid demand at each time interval. To implement such communication into the model, an additional real input variable, “SES_Demand,” was created. With the adaptor and new input signal created, the model took the form depicted in Figure 24, and is ready for export as an FMU. This procedure of using an adaptor to transform ports into their real input/output components, and creating additional inputs/outputs for declared variables, works well for simple models and models intended for use in model exchange mode. For complex models planned for simulation in co-simulation mode, use of adaptors may prove challenging if the initialization of the models is not well-defined. This is due to the explicit nature of co-simulation modeling. Further details on this will be given in later sections of this report.

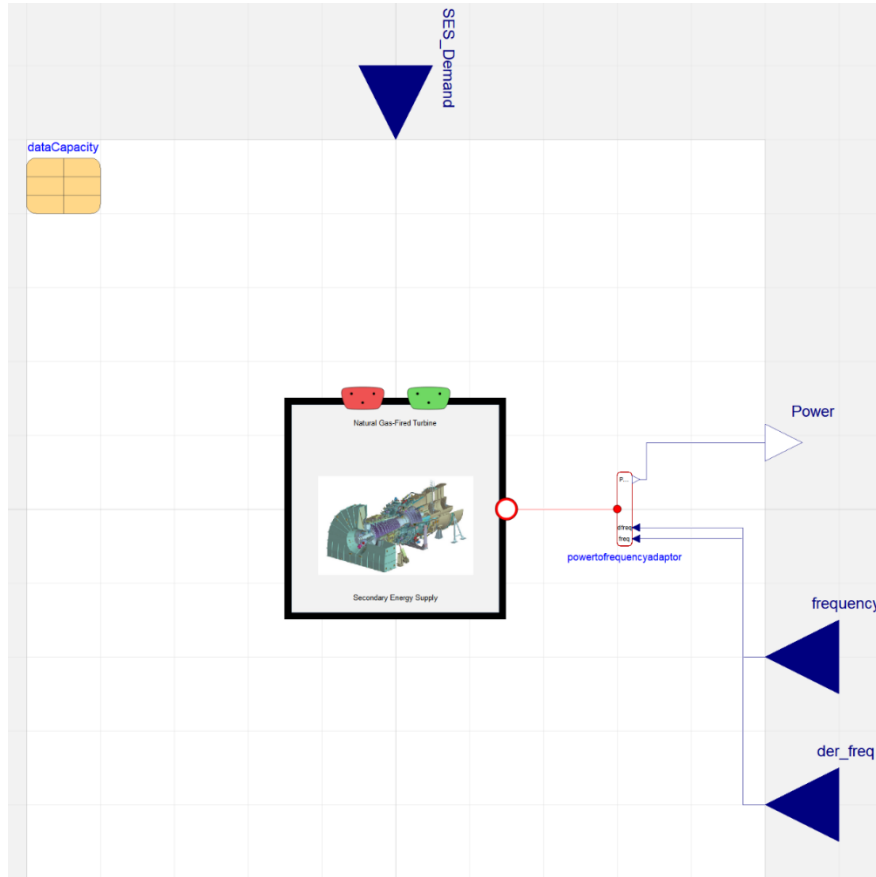


Figure 24. Preparing a natural gas turbine to be converted into an FMU. The inputs into the system are the peaking demand and the connection points for electricity backflow into the turbine model. The output is the electrical power as a real value.

Export

Dymola offers several ways to export a model as an FMU, as shown in Figure 25. The FMU can include three different types of export: model exchange, co-simulation using the CNode solver, and co-simulation using various Dymola solvers.

In model exchange, the component model will be exported without a solver, as it is assumed that the importing tool will provide the solver. For co-simulation models, CNode and Dymola solvers can be exported with the component model for use within other models. In general, CNode solvers are sophisticated enough for most models, and export can be selected in either C-code or binary code, depending on the purchased Dymola license. In the event a particular Dymola solver is required to compile a component model, the co-simulation export can only be accomplished as a binary, thus protecting the proprietary solver information held by Dassault systems. However, binaries are operating-system dependent, so care must be taken to ensure that export of binary FMUs is conducted on the same operating system as the planned importing tool.

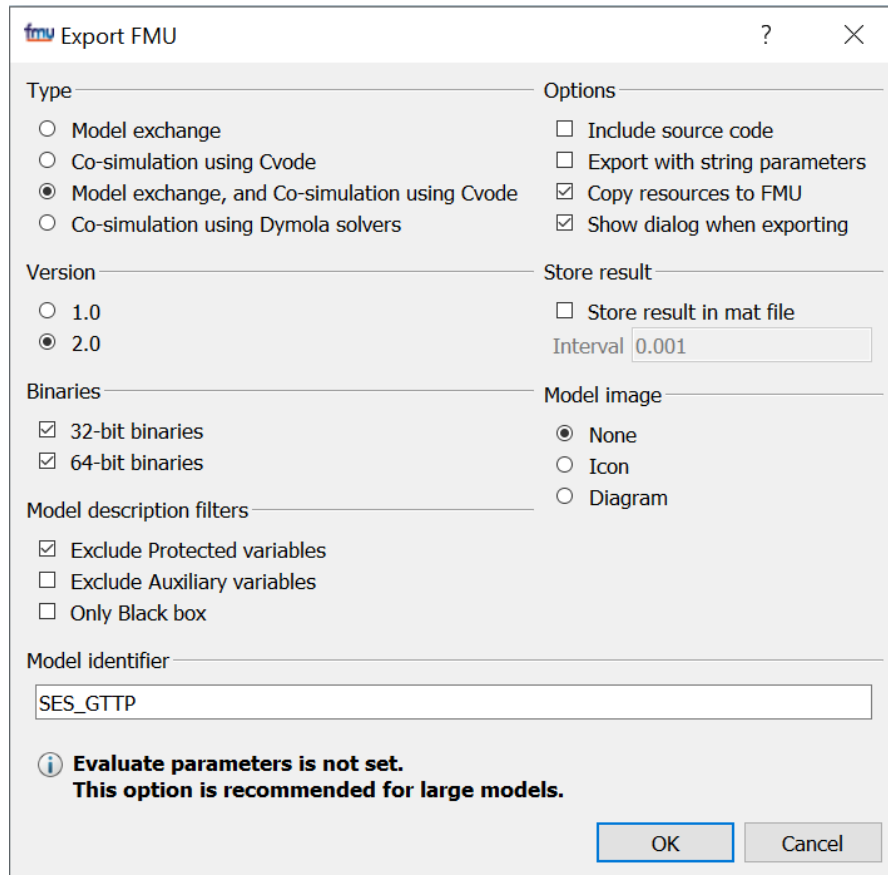


Figure 25. Export settings from Dymola 2021x.

Import

Once the model has been exported and an FMU created, the model will be present as an .fmu file. In the case of the natural gas turbine, it will be called “SES_GTTP.fmu.” To import this file in Dymola, click File → Open → Import FMU, as shown in Figure 26.

The FMU can be imported in either model exchange or co-simulation mode, as per Figure 27. This selection should be consistent with the export options included in the FMU. If the desired import mode is different than the model of the original FMU, the imported FMU will fail.

Including the “structured declaration of variables” option retains the structured file tree of variables that were present in the original model, enabling the user to look inside the FMU as though it were the original Dymola model. If this option is not selected, a single large list featuring all the variables available for access will be made available to the user.

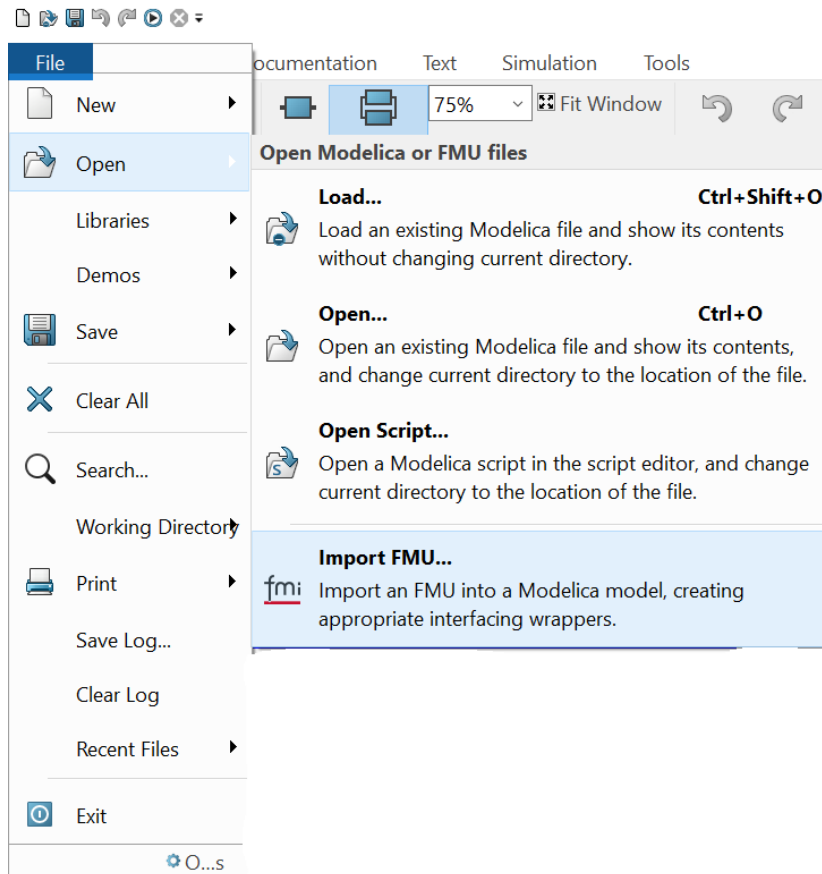


Figure 26. Importing FMU steps in Dymola 2021x.

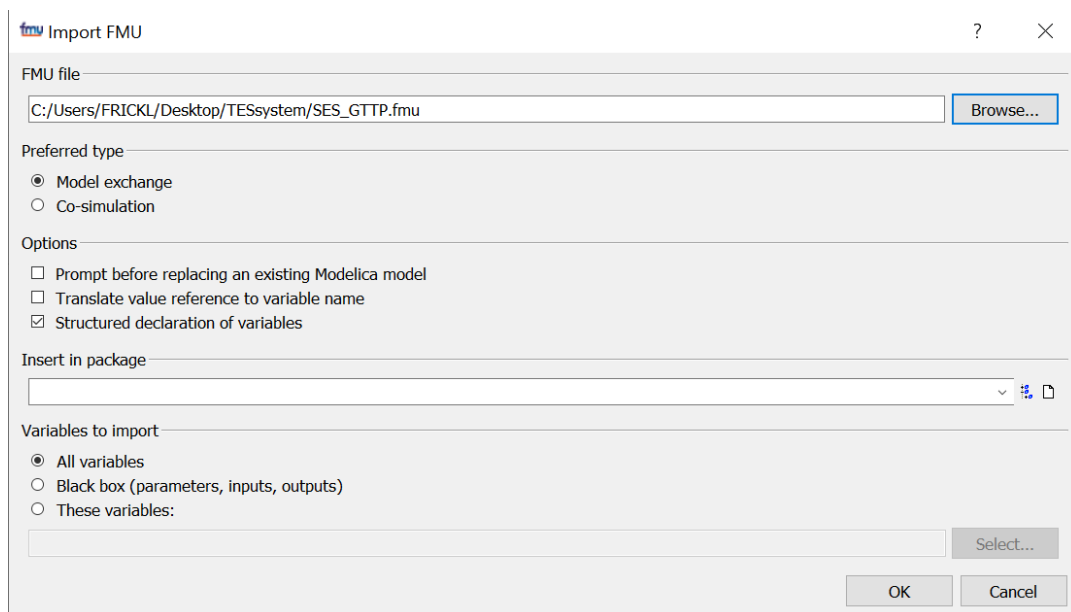


Figure 27. Import settings from Dymola 2021x.

Simulation

After the import step, the model can be used in place of the main component, as shown in Figure 28. In the system, it is important to ensure that all materials, initial conditions, nominal conditions, and parameter setpoints are consistent across the boundaries between the FMU and the rest of the model. This is particularly important because FMUs take real inputs and provide the surrounding model with outputs that have no physical constraint placed upon them. This reduces the number of checkpoints that the underlying application program interface (API)s has in order to ensure a consistent model. This places more onus on the engineers/researchers.

When using model exchange, the model will act similarly to the primary model, as the equation set remains exposed to the underlying import tool solvers. Conversely, in co-simulation mode, a specified “communication step” size must be selected, at which point the models will export results for communication with the surrounding external models. Selecting a small enough communication step to ensure that all the dynamics are captured is critical, but selecting a time-step communication interval that is too small greatly reduces the system’s simulation speed.

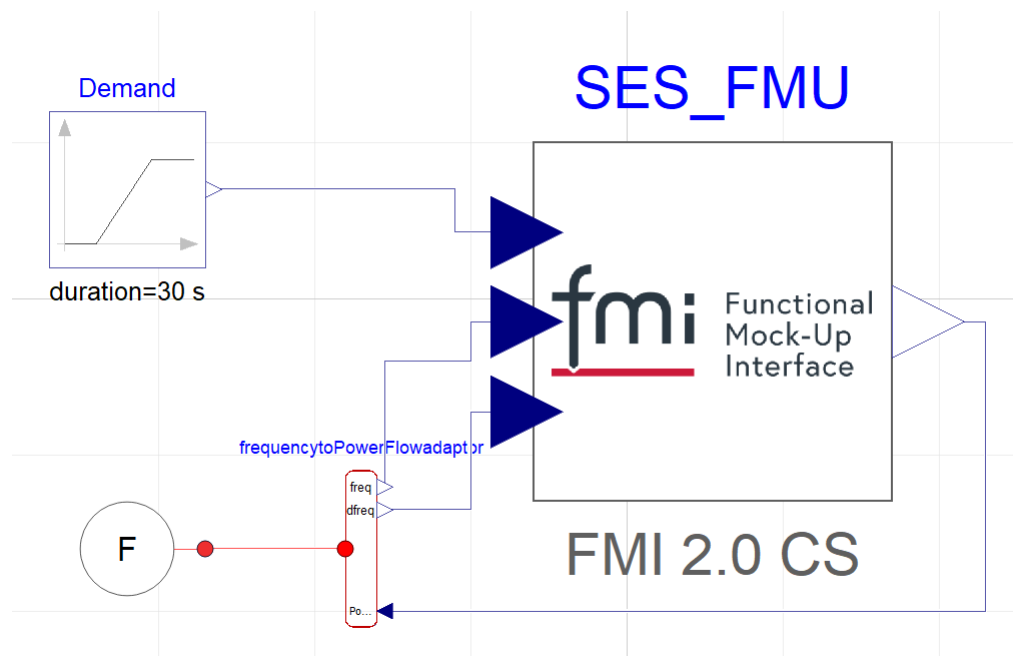


Figure 28. Proper import and use of a co-simulation FMU in Dymola.

SES_FMU in NHES.Systems.SecondaryEnergySupply.NaturalGasFiredTurbine.Examples.GTTP_Test_FMI_e...

General FMI Add modifiers Attributes

Instance name
fmi_instanceName SES_GTTP_fmU

Enable logging
fmi_loggingOn false

Input Handling
fmi_InputTime false Time point of input used when calling doStep.
fmi_UsePreOnInputSignals true

Step time
fmi_StartTime 0.0
fmi_StopTime 60.0
fmi_NumberOfSteps 500
fmi_CommunicationStepSize (fmi_StopTime - fmi_StartTime)/fmi_NumberOfSteps
stepSizeScaleFactor 1 Number of doSteps called between two CommunicationStepSize
fmi_forceShutDownAtStopTime false
fmi_rTol 1e-6 relative tolerance for the internal solver of the fmU

Instantiation
fmi_resourceLocation "file:/// + ModelicaServices.ExternalReferences.loadResource("modelica://SES_GTTP_fmU/Resources/Library/FMU/SES_GTTP/resources")

OK Cancel Info

Figure 29. FMI settings for the natural gas turbine FMI/FMU in co-simulation mode. The communication interval was every 0.12 seconds, with an internal solver tolerance of $1e-6$. The internal solver was the Dymola specific DASSL solver.

To test the FMU, the physical model was run in co-simulation, model exchange, and normal Modelica-only mode. The resulting turbine output is illustrated in Figure 30. For the three aforementioned modes, all the models converged to the same solution over the 60-second simulation time, with real-time simulation speeds of 1.316, 0.147, and 0.064 seconds, respectively. The co-simulation FMI settings are shown in Figure 29. In all cases, the simulation speeds are slower for FMU representations. This can be attributed to the increased overall number of variables that must be simulated due to the need for additional boundary blocks to accommodate real inputs/outputs. In addition, for co-simulation, the limiter on simulation speed is directly impacted by the communication step size and the nonlinearity of the coupled system. For example, increasing the communication step size from every 0.12 seconds to every second reduces simulation time from 1.316 seconds to 0.514 seconds. However, as demonstrated in Figure 30, this comes at the price of accuracy. Therefore, it is essential that, for co-simulation models, the communication step occur at points with slow-moving physics in order to allow the system a larger communication step size.

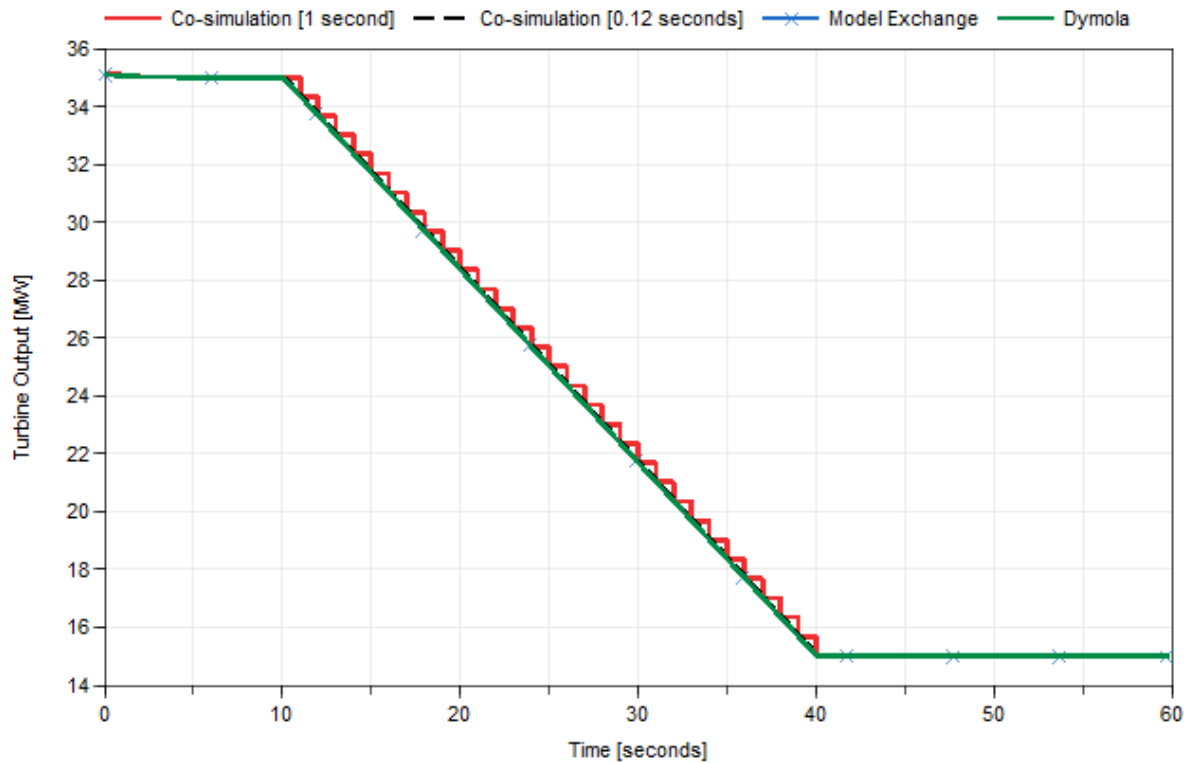


Figure 30. Comparison of Dymola model results to co-simulation and model exchange FMU results. Communication intervals for co-simulation = 0.12 seconds and 1 second.

3.3 Turbine Replacement Example

The creation of FMUs makes it possible to take a model from one coding language and encapsulate it in a standardized format for use within another coding language. To test this functionality with the more complicated fluid equation set of water, a natural circulation small modular reactor (SMR) set was chosen. The modeling set, shown in Figure 32, includes the reactor, energy manifold, turbine generator, and electric grid—all modeled in the Modelica language. The turbine generator set was then converted from a Modelica model into an FMU to ensure that all the proper data were input into and transferred between the models. The initial step was to implement the adaptors (discussed in the previous section) that transform the fluid ports into constituent real outputs, as shown in Figure 31. The progression of translation is shown in Figure 32, going from the Modelica-only model to a model exchange FMU that is then included in the model.

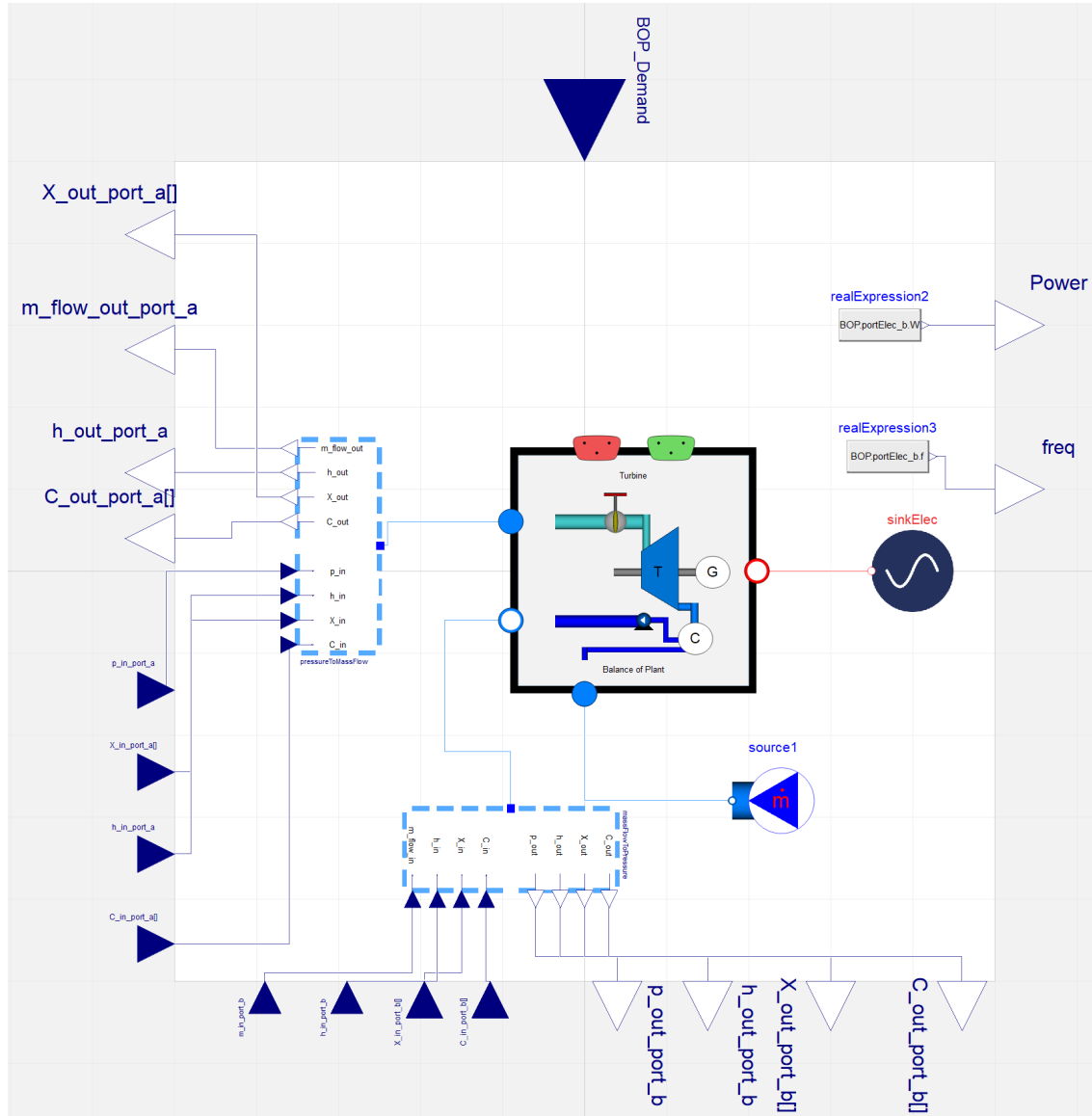


Figure 31. Translation of the Modelica turbine generator model into an FMU-ready design.

The control system within the turbine generator model is maintained through the translation process and can fulfill the desired setpoints within the turbine model. Then, the model is exported into a model exchange FMU and reimported into the Modelica framework. A comparison of the turbine power output is depicted in Figure 33, showing that the different versions of the model are in close agreement with each other. The differences can be attributed to minute variations in initialization subroutines that occur in the initialization phase of the run. The FMU-based results and input-based Modelica results are nearly identical, and both simulations were able to meet the turbine demand setpoints. It is worth noting that a version using co-simulation was attempted, but instabilities arising from the explicit time-stepping scheme could not be overcome; thus, the co-simulation was deemed unsolvable. Such scenarios become more common as the complexity of the models increases. While co-simulation is the easiest version of FMI to implement, instabilities such as these also increase the possibility of simulation roadblocks.

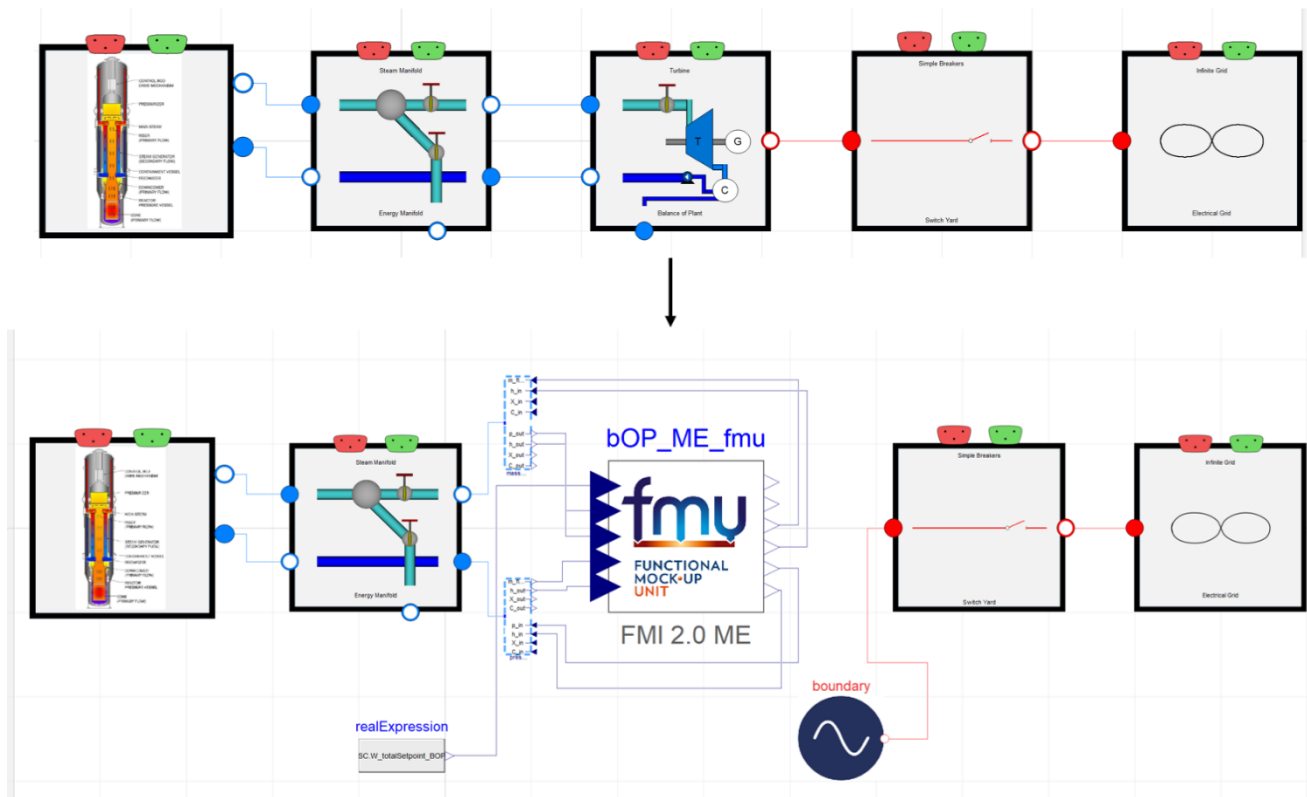


Figure 32. Transition from a Modelica model to an FMI-based simulation.

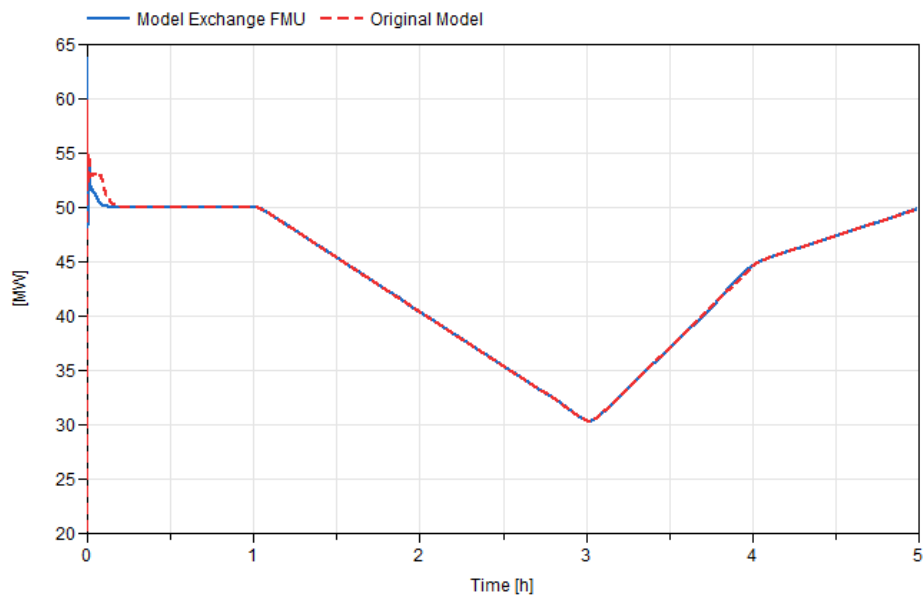


Figure 33. Comparison of turbine output results between the original model and model exchange FMU.

4. HYBRID REPOSITORY

At the beginning of the IES Project, a version control repository was delivered in order to provide a common location for the deployment of system and component models and analyses developed and constructed with Modelica/Dymola and RAVEN. To initiate the construction of a flexible plug-and-play Modelica/RAVEN framework for IES analysis, a restructuring of the version control repository (HYBRID, available at <https://hpcgitlab.inl.gov/hybrid/hybrid> and at the open-source repository location <https://github.com/idaholab/HYBRID>) was performed.

The following main tasks were performed for this specific activity:

- Usage of the RAVEN regression test system (named ROOK) for deployment of a single, integrated testing platform for both Modelica and Dymola models/analysis and RAVEN workflows. The testing system was linked with the automatic continuous integration tool for the automatic testing of the models and analyses when new modifications are added in the repository.
- Folder structure optimization for easier browsing and usage of the version control repository.

Figure 34 shows the new repository structure, with the following main folders identifiable:

- ***Models***: contains the Modelica and Dymola models
- ***archive***: where old examples and analyses (i.e., documents, models, input files, etc.) are archived and stored to guarantee reproducibility of published results
- ***developer_tools***: contains utility scripts, methods, and files required for the automation, deployment, and verification of the tools and software products of the HYBRID repository. This folder contains all the scripts for the automatic generation of software quality assurance (SQA) documentation (e.g., requirements, traceability matrix, etc.).
- ***scripts***: contains scripts for installing the HYBRID repository (e.g., scripts to create the HYBRID configuration file). It also contains specialized classes and scripts for the automatic regression testing system (e.g., output checkers) and *Python*-based launchers for Dymola models (*dymola_launcher*).
- ***tests***: contains all the tests that are automatically executed by the continuous integration system and are locally executable by running the command “run_tests.”
- ***TRANSFORM-library***: submodule of the Oak Ridge National Laboratory based TRANSFORM library that provides base models for many of the integrated energy systems models
- ***raven***: links to the RAVEN repository.

















Name	Last commit	Last update
 .gitlab	issue and MR templates added	3 years ago
 Models	Cleanup of File Tree and removal of broken ...	2 weeks ago
 archive	added licens	1 month ago
 developer_tools	added RTR scripts	3 weeks ago
 doc	Update manual	2 weeks ago
 scripts	added headers	1 month ago
 tests	added rts+srs + modified tests file to add re...	3 weeks ago
 TRANSFORM-Library @ d735ccc3	update of submodules	2 months ago
 raven @ cd67dba1	update of submodules	2 months ago
 .gitignore	Changing the .gitignore file to not ignore .m...	7 months ago
 .gitmodules	added transform submodule	9 months ago
 00README.txt	Civet test dirtectory set-up	4 years ago
 LICENSE.txt	added licens	1 month ago
 NOTICE.txt	added licens	1 month ago
 README.md	added licens	1 month ago
 run_tests	added headers	1 month ago

Figure 34. New structure of the repository.

Furthermore, a series of Modelica tests has been added to test the system-level interactions in the NHES Modelica repository. An example output of the regression system is shown in Figure 35.

Table 1. Synopsis of Modelica test cases.

Test	Description
Bouncing Ball	Simple test that models a bouncing ball hitting the ground.
BOP Boundaries Test A	Balance of plant system based on pressure difference
BOP Boundaries Test B	Balance of plant system based on forced mass-flow rate
Desalination 1 Pass	Single-stage reverse osmosis component check
Desalination 2 Pass	Second stage reverse osmosis component check
Desalination 2 Pass Mixing	Two-stage reverse osmosis with mixing
Desalination Reverse Osmosis	Fully encapsulated two-stage reverse

Module	osmosis with mixing
Desalination NHES Basic	Controlled desalination NHES system
Desalination NHES Complex	Controlled via signal bus NHES RO system with parallel osmosis units
FMI Fluid CS	Test of the fluid adaptors in a small problem in co-simulation mode
FMI Fluid CS	Test of the fluid adaptors in a small problem, using model exchange
FMI Heat CS Capacity	Test of the thermal adaptors in a small problem in co-simulation mode, using a thermal capacitance model
FMI Heat CS Conduction	Test of the thermal adaptors in a small problem in co-simulation mode, using a heat conduction model
FMI Heat ME	Test of the thermal adaptors in a small problem in model exchange (solving both the conduction and capacitance models simultaneously)
Generic Modular PWR	SMR of a NuScale size system with a pump
GTPP_Test	Gas turbine load follow test – 60-second electric demand oscillation
HTSE Power Test	High Temperature Steam Electrolysis (HTSE) NHES system based on power input control
HTSE Steam Test	HTSE NHES system based on steam and power input control
Hydrogen Turbine Test	Hydrogen turbine load follow test – 60-second electric demand oscillation
NSSS_test	Westinghouse-style four loop PWR test – 10,000 seconds at nominal power
Simple_Breakers_Test	Test of electrical breakers on an infinite grid
SMR_4Loop	Test of load following a natural-circulation SMR – 5-hour load follow simulation
SMR Primary Test	Test of the primary loop of a natural-circulation SMR loop
SMR Nominal Test	Addition of nominal power test for a natural-circulation SMR reactor
Step-Down Turbines	Basic set of step-down turbines
Step-Down Turbines Complex	Test of a more complex step-down

	turbine system
Supervisory Control Test	Test of the supervisory control system for receiving input from external files
Test_Battery_Storage	Test of a simple electrical battery system – logical power flow simulation
Test_Thermal_Storage	Test of a Therminol-66 thermal energy storage facility through both charge and discharge cycles
TightlyCoupled_FY18_Battery	Complex system of systems from the 2018 case (including electric battery storage)
Tightly Coupled_FY18_TES	Complex system of systems from the 2018 case (including thermal energy storage [two-tank sensible heat])
Thermocline Cycling Test	Test of the hourly cycling of a single-tank packed-bed thermocline system
Thermocline Insulation Test	Test of the insulation heat loss through the tank walls of a single-tank packed-bed thermocline system

While these tests are not exhaustive of the Modelica repository system, they provide a systems-level understanding of the repository model state. Other tests will be added on an as-needed basis.

```

scripts/testers\DymolaMatDiffer.py:267: RuntimeWarning: invalid value encountered in subtract
  diffMatrix = np.abs(varTrajectories - varTrajectoriesgold)
rook: loading init file "C:/msys64/home/FRICKL/FORCE_Hybrid/hybrid/scripts/rook.ini"
rook: ... loaded setting "add_non_default_run_types = dymola,raven"
rook: ... loaded setting "add_run_types = dymola,raven"
rook: ... loaded setting "test_dir = tests"
rook: ... loaded setting "testers_dirs = scripts/testers,raven/scripts/TestHarness/testers/"
rook: found 34 test dirs under "tests" ...
rook: loading init file "C:/msys64/home/FRICKL/FORCE_Hybrid/hybrid/scripts/rook.ini"
rook: ... loaded setting "add_non_default_run_types = dymola,raven"
rook: ... loaded setting "add_run_types = dymola,raven"
rook: ... loaded setting "test_dir = tests"
rook: ... loaded setting "testers_dirs = scripts/testers,raven/scripts/TestHarness/testers/"
(1/34) Success( 20.32sec) tests\dymola_tests\Bouncing_Ball\
(2/34) Success( 21.43sec) tests\dymola_tests\Desalination_1_pass\
(3/34) Success( 25.44sec) tests\dymola_tests\BOP_L1_Boundaries_a_Test\
(4/34) Success( 26.46sec) tests\dymola_tests\BOP_L1_Boundaries_b_Test\
(5/34) Success( 15.22sec) tests\dymola_tests\Desalination_2_pass\
(6/34) Success( 17.50sec) tests\dymola_tests\Desalination_2pass_mixing\
(7/34) Success( 25.76sec) tests\dymola_tests\Desalination_NHES_basic\
(8/34) Success( 22.70sec) tests\dymola_tests\Desalination_RModule\
(9/34) Success( 26.41sec) tests\dymola_tests\FMI_Fluid_CS\
(10/34) Success( 44.89sec) tests\dymola_tests\Desalination_NHES_complex\
(11/34) Success( 13.93sec) tests\dymola_tests\FMI_heat_CS_capacity\
(12/34) Success( 26.64sec) tests\dymola_tests\FMI_Fluid_ME\
(13/34) Success( 13.90sec) tests\dymola_tests\FMI_heat_CS_conduction\
(14/34) Success( 15.00sec) tests\dymola_tests\FMI_Heat_ME\
(15/34) Success( 18.33sec) tests\dymola_tests\GTPP_Test\
(16/34) Success( 24.72sec) tests\dymola_tests\HTSE_Power_Test\
(17/34) Success( 33.96sec) tests\dymola_tests\Generic_Modular_PWR\
(18/34) Success( 17.87sec) tests\dymola_tests\Hydrogen_Test\
(19/34) Success( 15.92sec) tests\dymola_tests\Simple_Breakers_Test\
(20/34) Success( 39.00sec) tests\dymola_tests\HTSE_Steam_Test\
(21/34) Success( 41.62sec) tests\dymola_tests\NSSS_test\
(22/34) Success( 26.36sec) tests\dymola_tests\SMR_primary_test\
(23/34) Success( 36.63sec) tests\dymola_tests\SMR_Nominal_Test\
(24/34) Success( 16.67sec) tests\dymola_tests\StepDownTurbines\
(25/34) Success( 16.47sec) tests\dymola_tests\StepDownTurbines_complex\
(26/34) Success( 56.81sec) tests\dymola_tests\SMR_4Loop\
(27/34) Success( 14.09sec) tests\dymola_tests\Supervisory_Control_Test\
(28/34) Success( 13.83sec) tests\dymola_tests\Test_Battery_Storage\
" failing
(29/34) Skipped( None! ) tests\dymola_tests\TightlyCoupled_FY18_Battery\
" failing
(30/34) Skipped( None! ) tests\dymola_tests\TightlyCoupled_FY18_TES\
(31/34) Success( 7.77sec) tests\raven_tests\train\TrainArmaOnData
(32/34) Success( 33.35sec) tests\dymola_tests\Test_Thermal_Storage\
(33/34) Success( 27.62sec) tests\dymola_tests\Thermocline_Insulation\
(34/34) Success( 39.62sec) tests\dymola_tests\Thermocline_Cycling\

PASSED: 32
SKIPPED: 2
FAILED: 0
(bases)

```

Figure 35. An example of tests run in the ROOK regression system.

Other capabilities besides tests were added to the regression system in order to allow for smoother cross-platform and cross-machine compatibility. These capabilities were necessary because the commercial platform Dymola by Dassault systems has a series of settings that control the type of outputs sent to the final solution file. Ensuring that every user has the same flags turned on/off is unrealistic, since some of the flags are global settings turned on for every simulation loaded into their particular instantiation of Dymola. To get around this, the ROOK testing system added the capability to only look at those time steps or time intervals guaranteed to be included in each simulation of the model, regardless of the flags automatically loaded by Dymola. To accomplish this, an extra option (either “numberOfIntervals” or “OutputInterval”) is required in the simulateModel command in the regression system. The option numberOfIntervals tells Dymola how many output intervals to make, whereas OutputInterval tells Dymola at what time-step interval an output should be present for comparison. These can be selected in the Simulation Setup tab of the Dymola graphical user interface (GUI).

Further, a restart file loading capability was added to the Modelica regression system. This was included because, for complex models, the initialization phase of a simulation can require the Modelica solvers to spend a significant amount of time finding an initialization point. This is due to the highly nonlinear nature of the underlying physical equations. One way to avoid such situations is to provide a restart file to bypass the initialization phase of the simulation. A restart file is automatically created at the end of each simulation; this is the dsfin.txt file created in the

folder from which the simulation was run. This file includes the final values of the previous simulation, from which the new model can restart. Moving this file to the tests/reference folder and loading it into the regression system can save a substantial amount of time in regression testing and provide a consistent starting point for each test, rather than relying on the same initialization point being found during each regression testing cycle. Full details on how to utilize and create new regression tests can be found on the HYBRID wiki at <https://hpcgitlab.inl.gov/hybrid/hybrid/wiki> or at its open-source location, <https://github.com/idaholab/HYBRID/wiki>.

The work covered in this report was propaedeutic for releasing the modeling framework in the open-source community. Several activities were deployed for open-sourcing of the software:

- User documentation:
 - Development of an extensive user manual [9], providing a detailed description of the models (Modelica and Dymola) and instructions on how to execute them
- SQA documentation (see Figure 36), available both in the INL internal Electronic Document Management System (EDMS) and the GITHUB website under “./doc/sqa/”. Such documentation is aimed at collecting the following information:
 - Project planning information
 - High-level overview touching on our entire project and software development activities.
 - Roles and responsibilities
 - Merge request workflow (e.g. code change requests)
 - Workflow diagram
 - Software development plan
 - Documentation of references to other relevant plans and procedures
 - Information about the software safety and quality level determinations
 - Definitions of software validation and verification
 - Methods and procedures for software validation and verification.

And it is composed of the following set of documents:

- HYBRID Software Quality Assurance Plan (PLN-6274) (detailing the SQA procedures adopted for the development and lifecycle of the HYBRID software framework)
- HYBRID Software Configuration Management Plan (PLN-6274)
- HYBRID Software Test Plan (PLN-6274)
- HYBRID IT Asset Maintenance Plan (PLN-6274)
- HYBRID Verification and Validation Plan (PLN-6274)
- HYBRID Software Design Description (SDD-561)
- HYBRID Software Requirement Specification (SPC-2990)
- HYBRID Traceability Matrix (SPC-2990)

- HYBRID Configuration Item List (LST-1296).

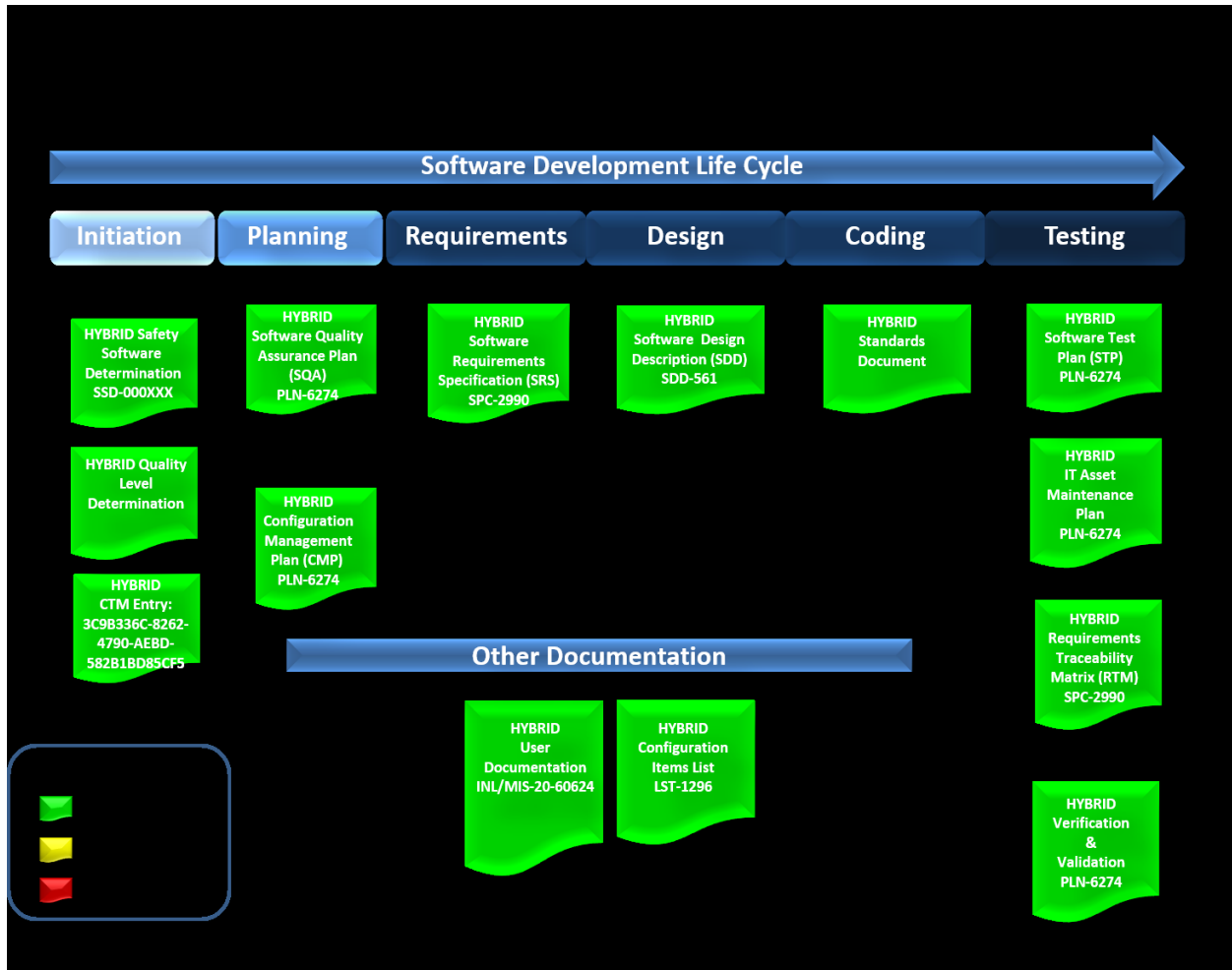


Figure 36. Status of the required SQA documentation for the HYBRID modeling repository.

5. DEPLOYMENT OF A RAVEN FMI/FMU DRIVER

Previous milestone reports [10],[11] demonstrated the successful execution of the FMIs and FMUs using external *Python*-based frameworks (FMPy [12] and PyFMI [13]). Such showcasing provided the basis for implementing the FMI and FMU interfaces within the RAVEN framework. The following sections offer a brief overview of the RAVEN code and the implementation of the driver for FMI/FMU-based models.

5.1 RAVEN Introduction

RAVEN is designed to perform parametric and probabilistic analyses based on the response of complex system codes. RAVEN can be used to investigate the system response—as well as the input space—using Monte Carlo, grid, or Latin hypercube sampling schemes, but its strength lies in the discovery of system features, such as limit surfaces, identifying and separating regions of the input space leading to system failure, and using dynamic supervised learning techniques. RAVEN includes the following major capabilities:

- Sampling of codes for uncertainty quantification and reliability analyses
- Generation and use of reduced-order models (ROMs) (also known as surrogate models)
- Data post-processing (time-dependent and steady-state)
- Time-dependent and steady-state statistical estimation and sensitivity analysis (mean, variance, sensitivity coefficients, etc.).

The RAVEN statistical analysis framework can be employed for several types of applications:

- Uncertainty Quantification
- Sensitivity/Regression Analysis
- Probabilistic Risk and Reliability Analysis
- Data Mining Analysis
- Model Optimization.

RAVEN provides a set of basic and advanced capabilities that range from data generation to data processing and data visualization. Its mission is to provide a framework/container of capabilities that engineers and scientists can use to analyze system responses, physics, and multi-physics by employing advanced numerical techniques and algorithms.

RAVEN was conceived with two major objectives in mind:

- To be as easy and straightforward as possible for scientists and engineers to use
- To allow for straightforward expansion of itself by providing clear and modular APIs (Application Programming Interfaces) to developers.

The RAVEN software is meant to be approachable by any type of user (computational scientists, engineers, or analysts). Every aspect of RAVEN was driven by this singular principle, from the build system to the APIs to the software development cycle and input syntax.

The main idea behind the RAVEN software design remains the creation of a multi-purpose framework characterized by high flexibility with respect to the possible performable types of analyses. The framework must be able to construct the analysis/calculation flow at run-time, interpret the user-defined instructions, and assemble the different analysis tasks following a user-specified scheme.

5.2 RAVEN Models

In RAVEN, coupling of the system to physical models is performed by the model entity API. The model entity represents a “connection pipeline” between the input and output spaces. The RAVEN framework (see Figure 37) provides APIs for the main model categories described below.

- Codes: The *Code* model represents the communication pipe between the RAVEN framework and any system and/or physical code/model. The communication between RAVEN and any driven code is performed through the implementation of interfaces directly operated by the framework. The procedure for coupling a new code/application with RAVEN is a straightforward process. The coupling is performed through a *Python* interface that interprets

the information coming from RAVEN and translates them to the input of the driven code. The coupling procedure does not require modifying RAVEN itself. Instead, the developer creates a new *Python* interface that will be embedded in RAVEN at run-time (no need to introduce hard-coded coupling statements). If the coupled code is parallelized and/or multi-threaded, RAVEN will manage the system in order to optimize the computational resources of both the workstations and High-Performance Computing systems.

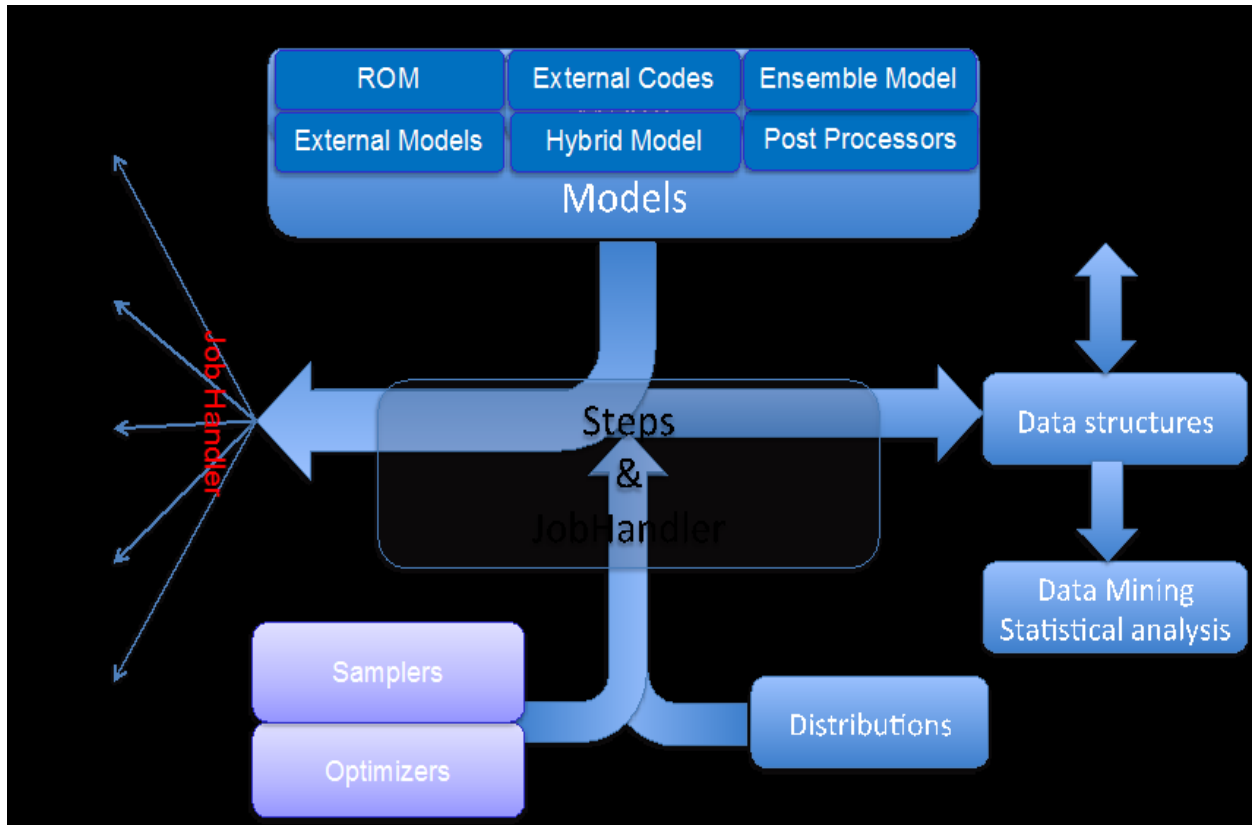


Figure 37. RAVEN framework scheme.

- **Externals:** The *External* model allows the user to create, in a *Python* file (imported at run-time into the RAVEN framework), its own model (e.g., set of equations representing a physical model, connection to another code, and control logic). This model will be interpreted/used by the framework and, at run-time, will become part of RAVEN itself.
- **Reduced Order Models (ROMs):** Reduced order, AI-based surrogate models, are a mathematical representation of a system, used to predict a physical system's selected output space. The "training" is a process that uses sampling of the physical model to improve the ROM's prediction capability (i.e., the capability to predict the status of the system given a realization of the input space). More specifically, in RAVEN, the ROM is trained to emulate a high-fidelity numerical representation (system codes) of the physical system.
- **Hybrid models:** The *HybridModel* can combine ROMs with any other high-fidelity model (e.g., *Code* or *ExternalModel*). The ROM will be "trained" based on the results from the high-fidelity model. The accuracy of the ROM will be evaluated based on the cross-validation scores, and the validity of the ROM will be determined via local validation metrics. After the ROM is trained, the *HybridModel* can decide which model (i.e., the ROMs

or high-fidelity model) to execute, based on the accuracy and validity of the ROMs in a particular operating region.

- Ensemble models: The *EnsembleModel* is used to create a chain of Models whose execution order is determined by the Input/Output relationships among them. If the relationships among the models evolve in a non-linear system, a Picard's Iteration scheme is employed.
- Postprocessors: The Post-Processor model represents the container of all the data analysis capabilities in the RAVEN code. This model is used to process the data (e.g., derived from sampling of a physical code) in order to identify representative Figures of Merit. For example, RAVEN uses Post-Processors to perform statistical and regression/correlation analysis, data mining and clustering, reliability evaluation, topological decomposition, etc.
- RAVEN FMI/FMU Driving System Development.

Development of the FMI/FMU driving system is based on the *ExternalModel* entity in RAVEN. As briefly reported in the previous section, the external model (see Figure 38) enables developers to create, in a *Python* module or platform, a direct coupling with a model coded in *Python* (e.g., a set of equations representing a physical model, connection to another code, and control logic). Once the external model is constructed, it is interpreted and used by RAVEN, ultimately becoming, at run-time, part of RAVEN itself.

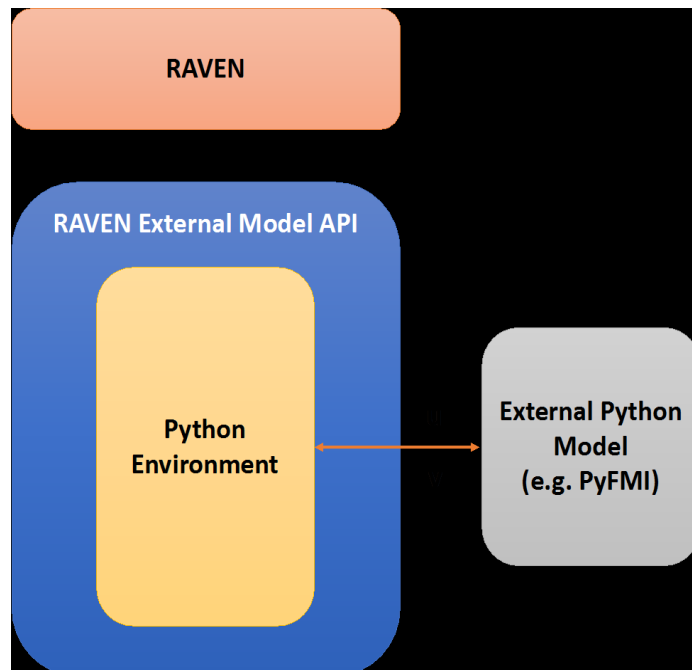


Figure 38. External model API.

<code>class FMIFMU(ExternalModelPluginBase):</code>	
<code>def run(self, container, inputs)</code>	Required
<code>def readExtInput(self, container, xmlNode)</code>	Optional
<code>def initialize(self, container, runInfo, oinputs)</code>	Optional
<code>def createNewInput(self, container, inputs, oinputs, samplerType, **Kwargs)</code>	Optional

Figure 39. FMI/FMU model skeleton in RAVEN.

The *ExternalModel* API (*ExternalModel* plugin) was used to develop, in RAVEN, a native driver for models using the FMI/FMU protocol. Figure 39 shows a snapshot of the “wrapper” that was developed. The “FMIFMU” RAVEN model implements a generalized method—based on the RAVEN API and syntax—to import, execute, and process the results of any model compatible with the FMI/FMU standard. The model consists of the following methods:

- **run**: The run method (the only required method in the API) aims to execute the FMU (FMI) for a given input coordinate (or input perturbation). The **run** method represents the pipeline between RAVEN and the FMI/FMU model. The method both executes and collects the results that will be then stored in the object “container,” ready for processing by RAVEN.
- **readExtInput**: This method is in charge of reading the user-defined input for the FMI/FMU that needs to be driven. It collects the following information (expandable in the future, if needed):
 - **startTime**: The start time of the driven FMU (e.g., 0.0 seconds)
 - **stopTime**: The stop time of the driven FMU (e.g., 60 seconds)
 - **stepSize**: The time step size to use for the calculation (e.g., 1.e-2 seconds)
 - **inputVariables**: A list of the input variables (e.g., demand)
 - **outputVariables**: A list of the output variables (e.g., power level)
 - **fmuFile**: The FMU location (e.g., [/path/to/myFmu.fmu](#))
- **initialize**: This method is invoked right before the model is executed. This method aims to load the FMI/FMU, instantiate the class, and initialize its settings.
This method is also in charge of performing error checking of the user-defined settings/options.
- **createNewInput**: This method, in case of a sampling strategy, is responsible for translating” the RAVEN info (e.g., the values of sampled variables) into the FMI/FMU syntax.

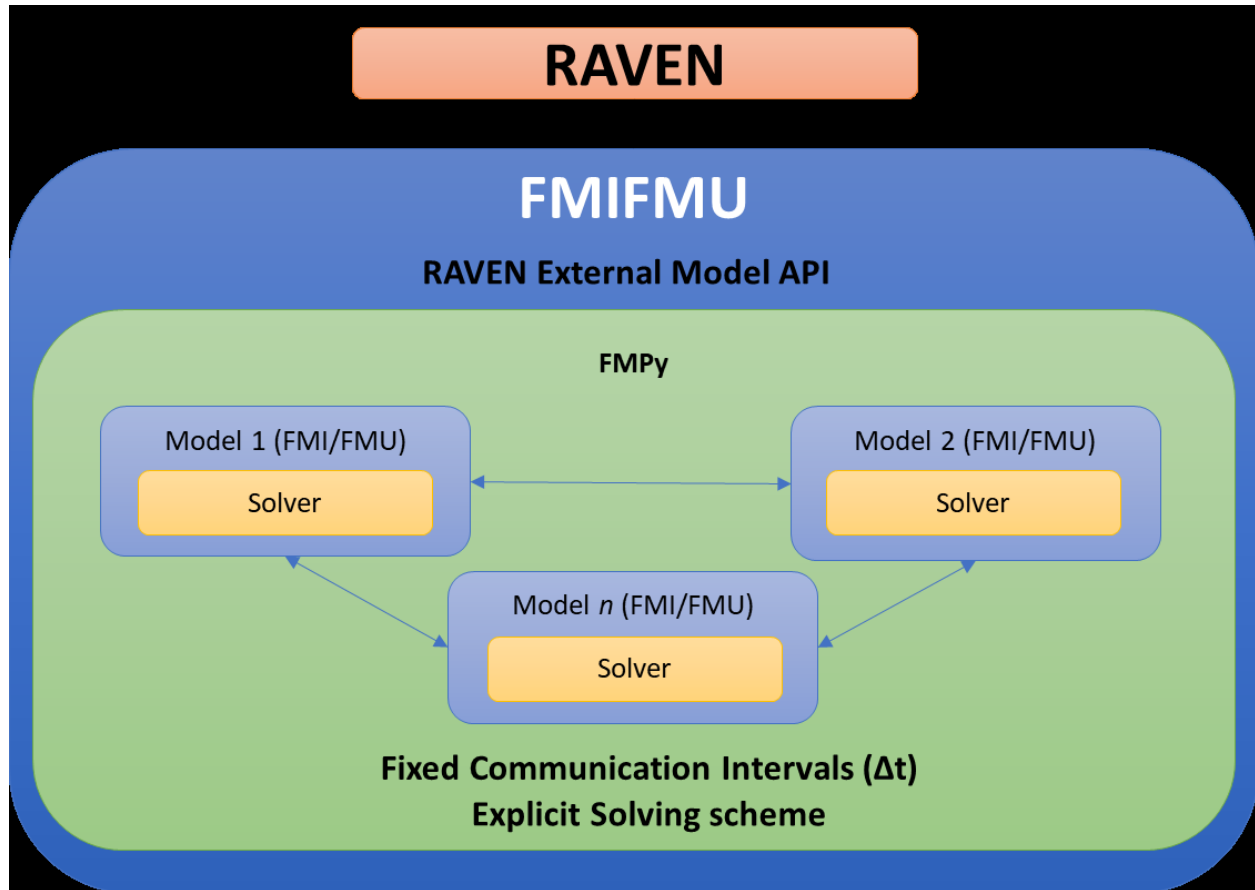


Figure 40. FMI/FMU co-simulation protocol coupled with RAVEN.

Depending on the type of protocol for the FMI or FMU of interest, two coupling schemes in the FMIFMU wrapper were developed. Both schemes are encapsulated in the same wrapper and are executable via the model API in RAVEN.

Figure 40 shows the coupling scheme for FMIs/FMUs when the co-simulation protocol must be used; RAVEN interacts with the different models via the FMIFMU wrapper that uses FMPy to import and interact with the FMUs. In this coupling scheme, RAVEN “perceives” the models imported via FMIs/FMUs just as it would any other external model or code. This protocol is indicated when the models to connect are loosely coupled (multi-physics feedbacks are not strong and/or the physics dynamic of the different models act on different time scales, e.g., seconds vs. hours or days).

On the other end, Figure 41 shows the coupling scheme for FMIs/FMUs when the model exchange protocol is used; in this configuration, RAVEN can directly interact with the universal solver that aims to solve all the models (compatible with the FMI/FMU protocol, in this case Dymola). This coupling scheme is preferable when the models are highly nonlinear and the models are tightly coupled with fast moving dynamics.

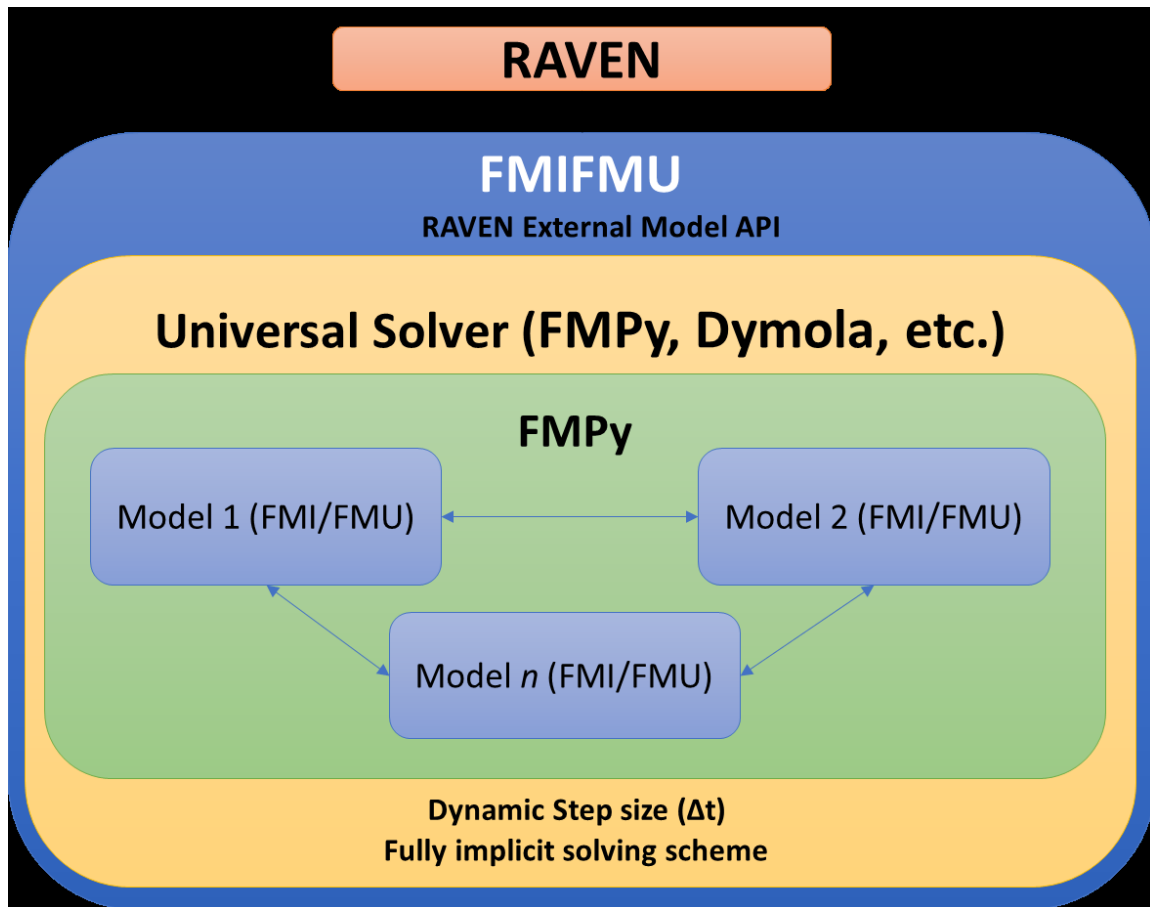


Figure 41. FMI/FMU model exchange protocol coupled with RAVEN.

```

<Models>
  <ExternalModel name="fmu" subType="FMIFMU">
    <variables> demand, time, SES_generator_P_flow</variables>
    <startTime> 0.0 </startTime>
    <stopTime> 14400</stopTime>
    <stepSize> 1.0 </stepSize>
    <inputVariables> demand </inputVariables>
    <outputVariables> SES_generator_P_flow</outputVariables>
    <fmuFile> GTTPfmu.fmu </fmuFile>
  </ExternalModel>
</Models>

```

Figure 42. External model FMIFMU example RAVEN input file.

Figure 42 shows an example of the portion (in XML) of the RAVEN input file required to use the FMIFMU wrapper. This XML block is the one processed by the previously-described method “*readExtInput*.”. Independently on the type of FMI/FMU that the model will import and use, the input file specifications do not change; the FMIFMU wrapper will collect the

information (co-simulation or model exchange) directly from the FMU (i.e., the **<fmuFile>**) after loading.

6. DEPLOYMENT OF RAVEN FMI/FMU EXPORTER FOR AI-BASED ANALYSIS ACCELERATIONS

As described in the previous section, the RAVEN framework provides APIs for different model categories, among which are the ROM, AI-based algorithms. In order to deploy the acceleration of IES analysis, the ROM (AI) entity is key. Indeed, the ROM is aimed at higher fidelity surrogate and system simulator-based models (for specific and limited operational domain) with a set of faster-execution equations that allow for the prediction of Figures of Merit (of interest) in a span of milliseconds.

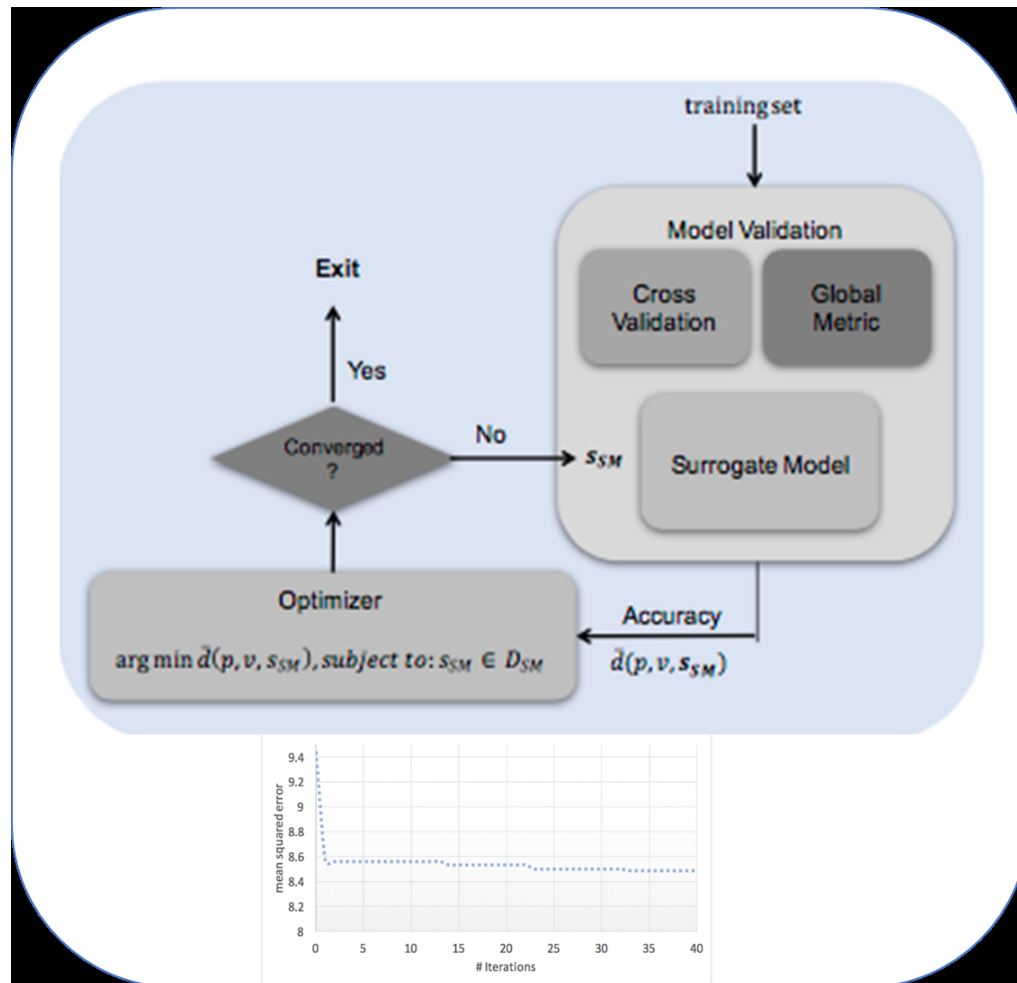


Figure 43. Construction process for surrogate models in RAVEN.

6.1 RAVEN AI construction

Figure 43 illustrates the standard process of constructing (via optimization) RAVEN surrogate (AI) models. The surrogate model of interest is trained on a dataset, and its hyper-parameters (i.e. parameters and characteristics of the surrogate model of interest) are tuned to maximize the accuracy in predicting the figure(s) of merit (FOMs) of interest. As shown in Figure 44, the accuracy is assessed by applying statistical methodologies (i.e., cross-validation), which consists of randomly portioning the dataset into “training” and “testing” datasets. The “training” dataset is used for constructing the surrogate model, and its prediction is compared with the “testing” dataset. The prediction accuracy is then assessed using distance metrics (e.g., R2 score) between the surrogate model and the testing dataset.

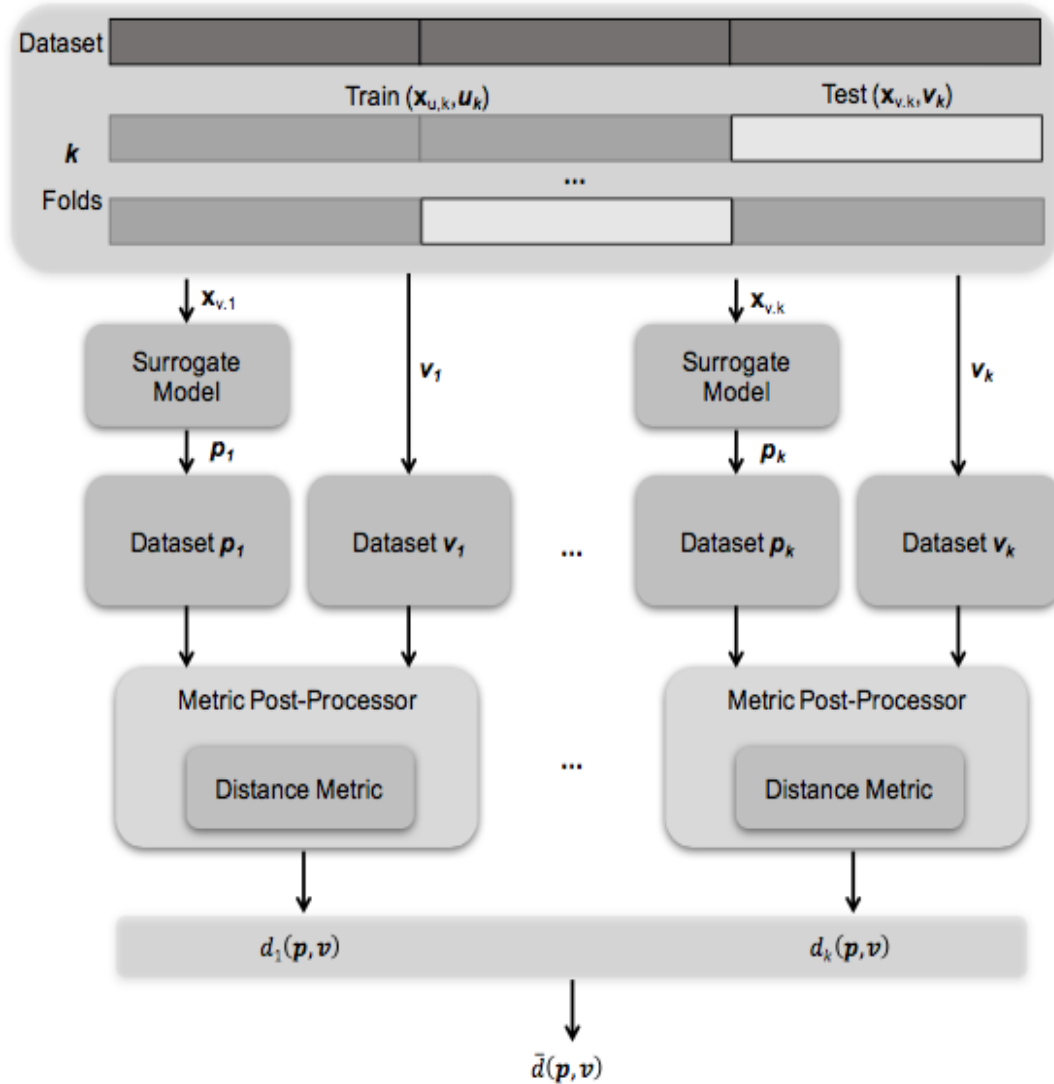


Figure 44. RAVEN ROM cross-validation scheme.

The so-constructed surrogate models allow for fast evaluation of the dynamics (or steady state) of the FOMs of interest. Therefore, such models can accelerate analyses (greatly reduce the computational time), by replacing high-fidelity physical models with a ROM representation.

6.2 Development of FMI/FMU exporting capabilities for RAVEN AI

To exploit RAVEN AI capabilities, a workflow to export trained (constructed) ROMs using the FMI/FMU protocol was developed in RAVEN.

The exporting of RAVEN AI is performed according to the following two steps:

- 1) Exploit the native RAVEN serialization system, which is responsible for serializing (i.e., saving in a binary file) already-trained surrogate models that can be loaded in external (*Python*-based) packages (outside RAVEN).
- 2) Use and extend the PythonFMU library (<https://github.com/NTNU-IHB/PythonFMU>), which is a lightweight framework that enables the packaging of *Python* 3 code as co-simulation FMUs (following FMI version 2.0).

To deploy any model in an FMI/FMU-compatible framework, that model (i.e., ROM) must be able to be inquired at each “time step,” meaning that the model must allow for execution as an integrated model and not as a “black-box simulation”. To achieve this goal, the RAVEN ROM APIs were upgraded by implementing a “method” to solve the surrogate model at each time step. This modification, in conjunction with the two steps reported above, allows for RAVEN ROM models to be exportable as FMI/FMUs.

Once the RAVEN AI is trained following the standard process reported in section 6.1, it can be finally exported following the steps reported in Figure 45. An example of the RAVEN input blocks is reported in Figure 46, where:

- In the **<Models>** node, the RAVEN ROM (AI) is shown.
- In the **<Files>** node, the output FMI/FMU filename is specified.
- In the **<Steps>** node, the trained ROM (input) is exported as FMI/FMU (output).

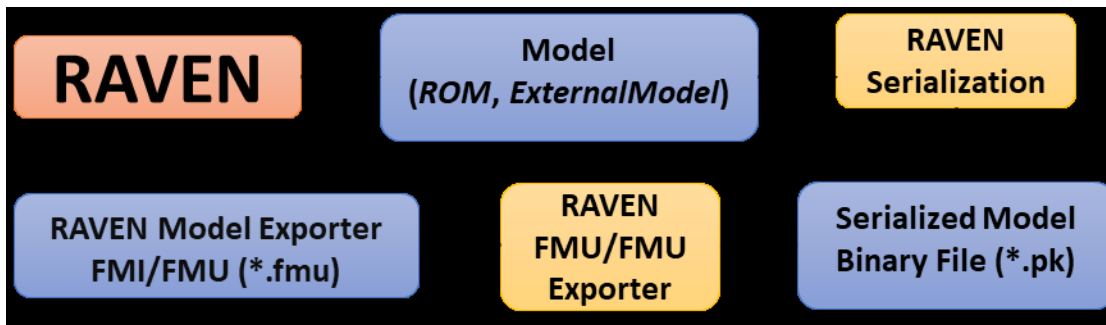


Figure 45. RAVEN AI FMI/FMU exporting process.

```

<Models>
  <ROM name="GTTProm" subType="SciKitLearn">
    ...
  </ROM>
</Models>
<Files>
  <Input name="FMU" type="fmu"> GTTProm.fmu </Input>
</Files>

<Steps>
  <IOStep name="romDump">
    <Input class="Models" type="ROM"> GTTProm </Input>
    <Output class="Files" type="" > FMU </Output>
  </IOStep>
</Steps>

```

Figure 46. Example RAVEN input file to export AI as FMIs/FMUs.

The FMI/FMU exporting capability allows for the deployment of the scheme reported in Figure 47, where the RAVEN models can be used, as FMI/FMUs, in tandem with any Dymola/Modelica (in general) and HYBRID (in particular) physical models.

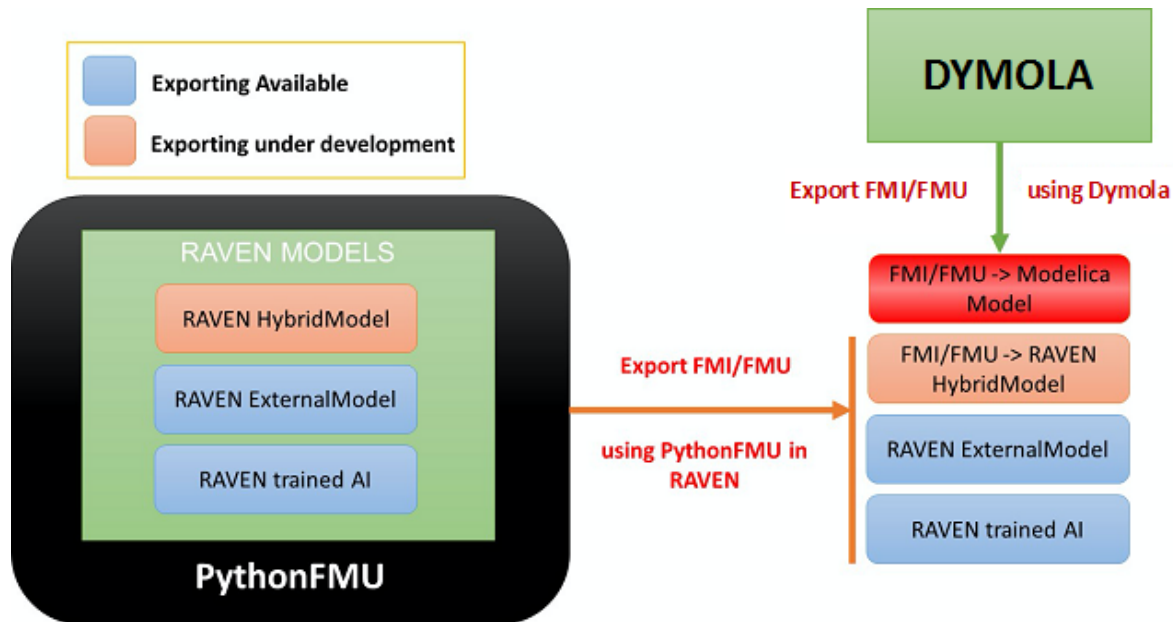


Figure 47. RAVEN's current FMI/FMU exporting capabilities.

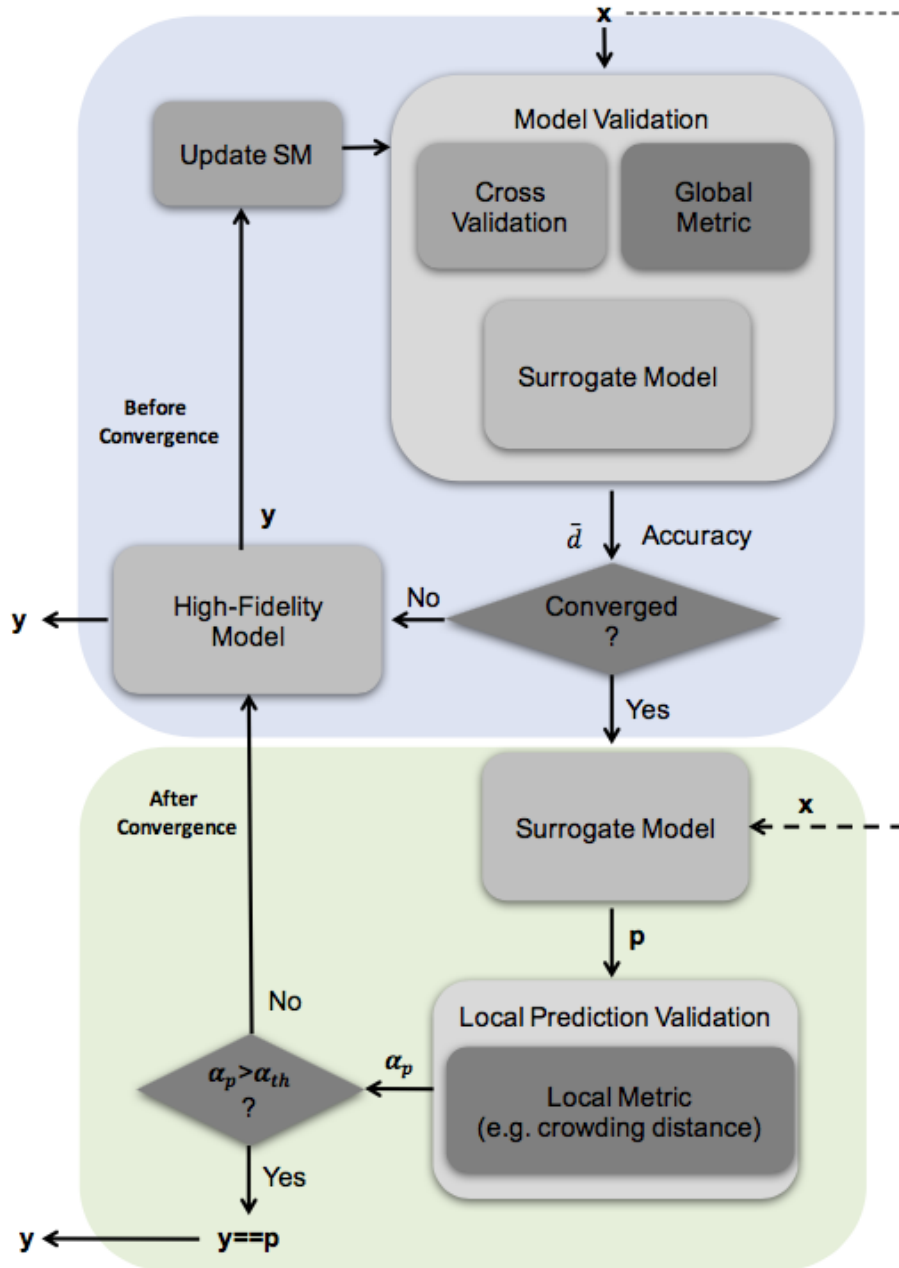


Figure 48. RAVEN hybrid model scheme.

6.3 Future Extension of the FMI/FMU Exporter to the RAVEN Hybrid Model

In previous sections, the creation and export of RAVEN AI was discussed. ROM usage is an approach that can drastically reduce the computational time of analyses and accelerate deployment of models. To obtain the optimal prediction capability, the ROM must be constructed and applied only within the domain of its training set; in other words, the ROM can guarantee valid predictions only within (or slightly outside) the boundaries of its training set. For example, if a ROM is trained by perturbing a temperature between 500 and 600 K, the ROM

should not be used for predicting a system response at 1000 K.

RAVEN includes an advanced capability called the “hybrid model” to tackle this problem. Indeed, this model is a special class of algorithms aimed to couple in-tandem, high-fidelity physical and mathematical models (e.g., FMI/FMU Dymola models) and AI algorithms (e.g., ROM, AI). The AI is trained based on the results from the high-fidelity model. The global accuracy of the AI is evaluated based on cross-validation scores, and the local (e.g., prediction) validity is determined via certain local validation metrics (i.e., metrics aimed to assess the confidence of the AI predictions). Once the AI is trained, the hybrid model can decide which model (i.e., the AI or high-fidelity model) to execute, based on the aforementioned accuracy and validation metrics. Figure 48 shows the scheme behind the hybrid model formulation. Since the predictions of the surrogate model are assessed in terms of accuracy, this algorithm discards ROM predictions if they fall outside its training set boundaries or the response confidence is too low. In such cases, the high-fidelity model is used and the ROM training set updated.

In the next steps for this program, the “hybrid model” capability will be leveraged in tandem with the FMI/FMU exporting protocol in order to accelerate the execution of systems that include multiple FMI/FMUs, allowing for the deployment of models that are able to autonomously switch between RAVEN AI and Dymola models during analyses. Each FMI/FMU will be coupled in a hybrid model configuration, resulting in accurate modeling and CPU time saving.

7. Integrated Energy Park Demonstration Case

To demonstrate the full range of capabilities described in this report, a final test case on an integrated energy park was conducted. The integrated energy park, shown in Figure 49, consists of a nuclear reactor, electric batteries, and a natural gas turbine. The natural gas turbine is the component to be exported as an FMU. The natural gas peaking turbine will then be replaced with its own FMU from three different sources: the Dymola FMU in both model exchange and co-simulation, then a RAVEN-based surrogate using co-simulation mode.

NPP

EM

BOP

SY

IG

Battery

NG

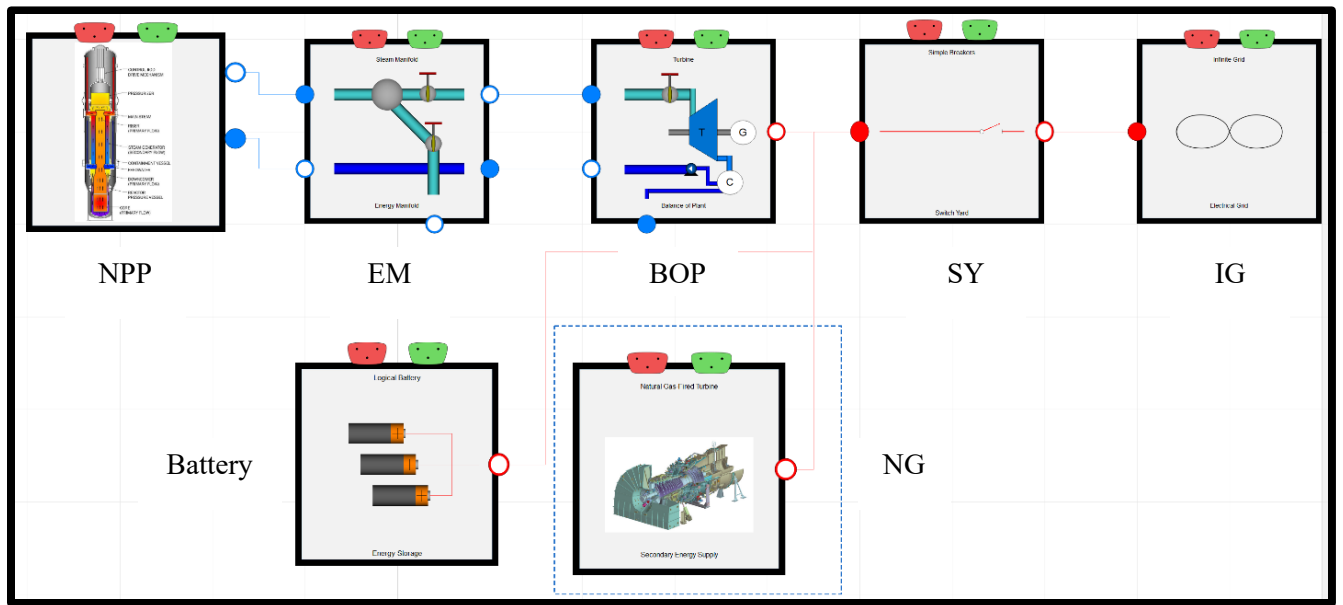


Figure 49. Integrated energy park consisting of a nuclear reactor (NPP), Energy Manifold (EM), Balance of Plant (BOP), Switch Yard (SY), Electric Batteries (Battery), Infinite Grid (IG), and a Natural Gas turbine (NG). The natural gas turbine is to be exported as an FMU.

7.1 FMI/FMU Creation and Use within Dymola

As outlined in earlier sections of this report, the natural gas turbine model needs to be modified with an electric power adaptor and an input demand signal in order to ensure that all the variables contained in the flow ports are realigned into real input/output variables.

The adaptors outlined in the previous sections can be used to accomplish these modifications. For the natural gas turbine example, illustrated in Figure 50, the electric port must be converted into real inputs/outputs using the PowerFlowToFrequency adaptor previously described. In addition, the control system of the natural gas turbine requires a top-level demand signal to communicate the grid demand at each time interval. To implement this communication into the model, an additional real input variable, “SES_Demand,” was created. With the adaptor and the new input signal created, the model is ready to be exported as an FMU.

This procedure of using an adaptor to transform ports into their real components and creating additional inputs/outputs for declared variables works well for simple models and models intended to be used in model exchange mode. For complex models planned for simulation in co-simulation mode, use of adaptors may prove challenging if the initialization of the models is not well-defined. This is due to the explicit nature of co-simulation modeling.

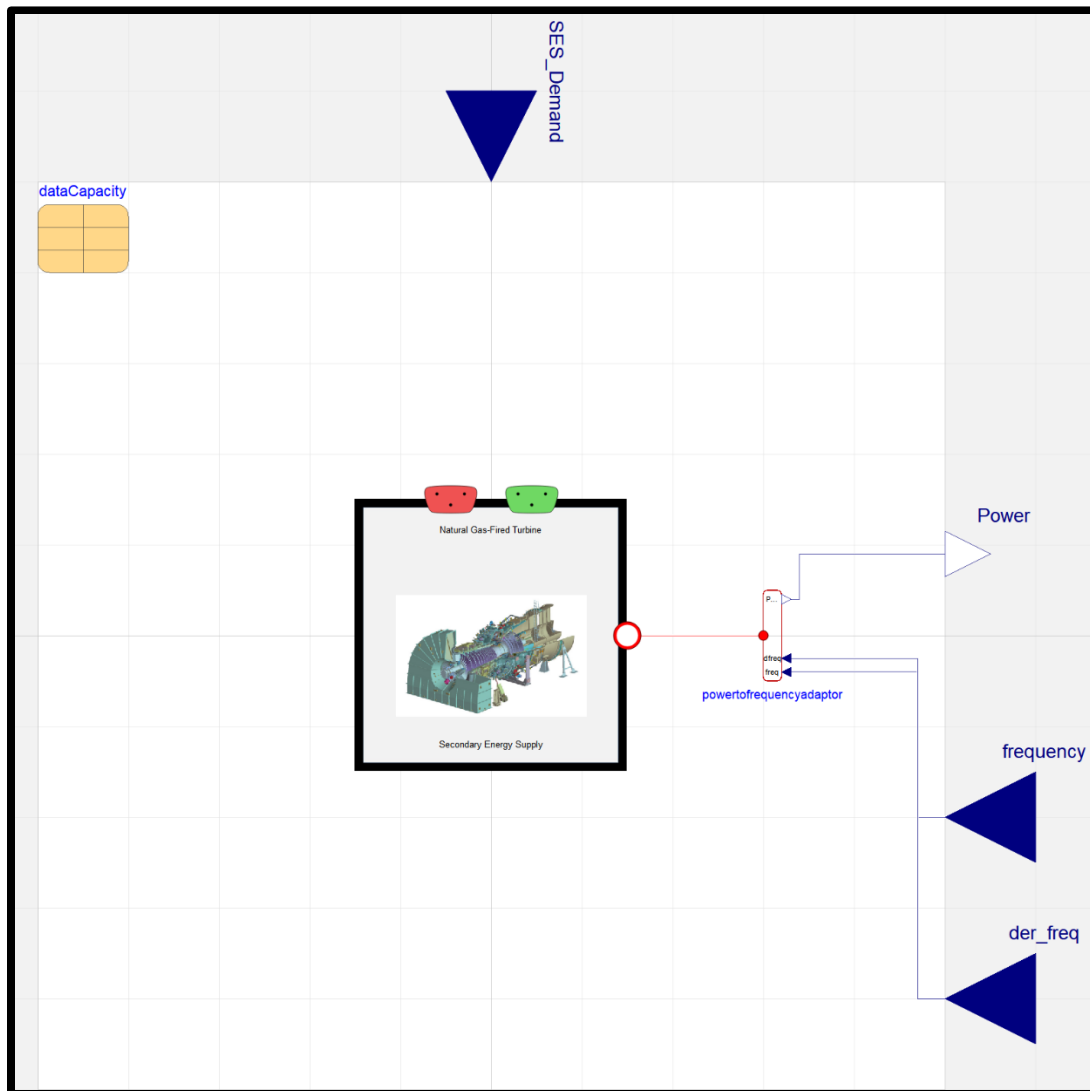


Figure 50. Preparing the natural gas turbine for conversion into an FMU. The inputs into the system are the peaking demand and connection points for electricity backflow into the turbine model. The output is the electrical power as a real value.

Once the model has been exported using the Dymola interface, it can then be re-imported into the program and can replace the natural gas turbine model. Since the FMI consists of three inputs and one output, the three inputs must be specified by the user. To accomplish this, the FrequencytoPowerFlow adaptor was placed in the Modelica model along with a “real” expression to connect the turbine demand to the FMI/FMU, as shown in Figure 51.

Using this version of the FMI/FMU, three separate 5-hour simulations were run: one with Modelica-only input, one with a co-simulation version of the gas turbine, and one in model exchange mode. The results of this simulation set are depicted in Figure 52. Over the course of the full 5-hour simulation, the results are all in near-perfect agreement with the setpoints, with the model exchange and Dymola results being basically identical, and co-simulation being only as accurate as the communication step of 1 second would allow. However, of note is that, while the Dymola and model exchange versions of the model completed in 121.3 and 156 seconds,

respectively, the co-simulation model took far longer to solve (a total of 642 seconds). This increased simulation time can be attributed to the additional communication time between the models as well as the additional initialization routine required by the solvers.

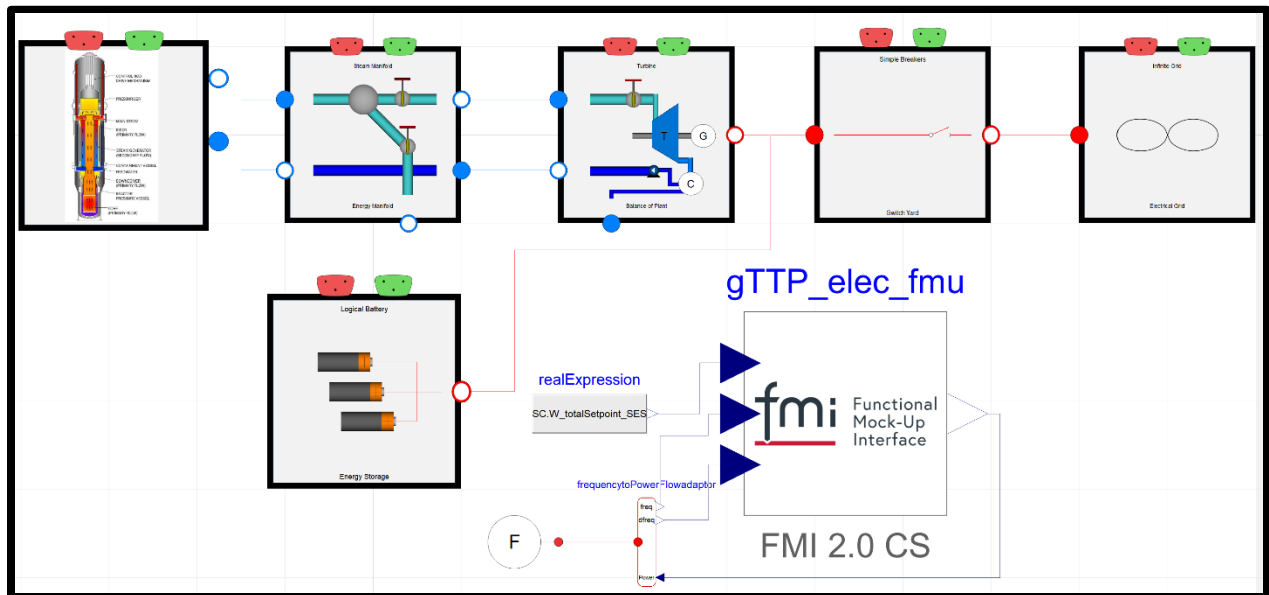


Figure 51. Integrated energy park consisting of a nuclear reactor, electric batteries, and a natural gas turbine replaced by a co-simulation FMU.

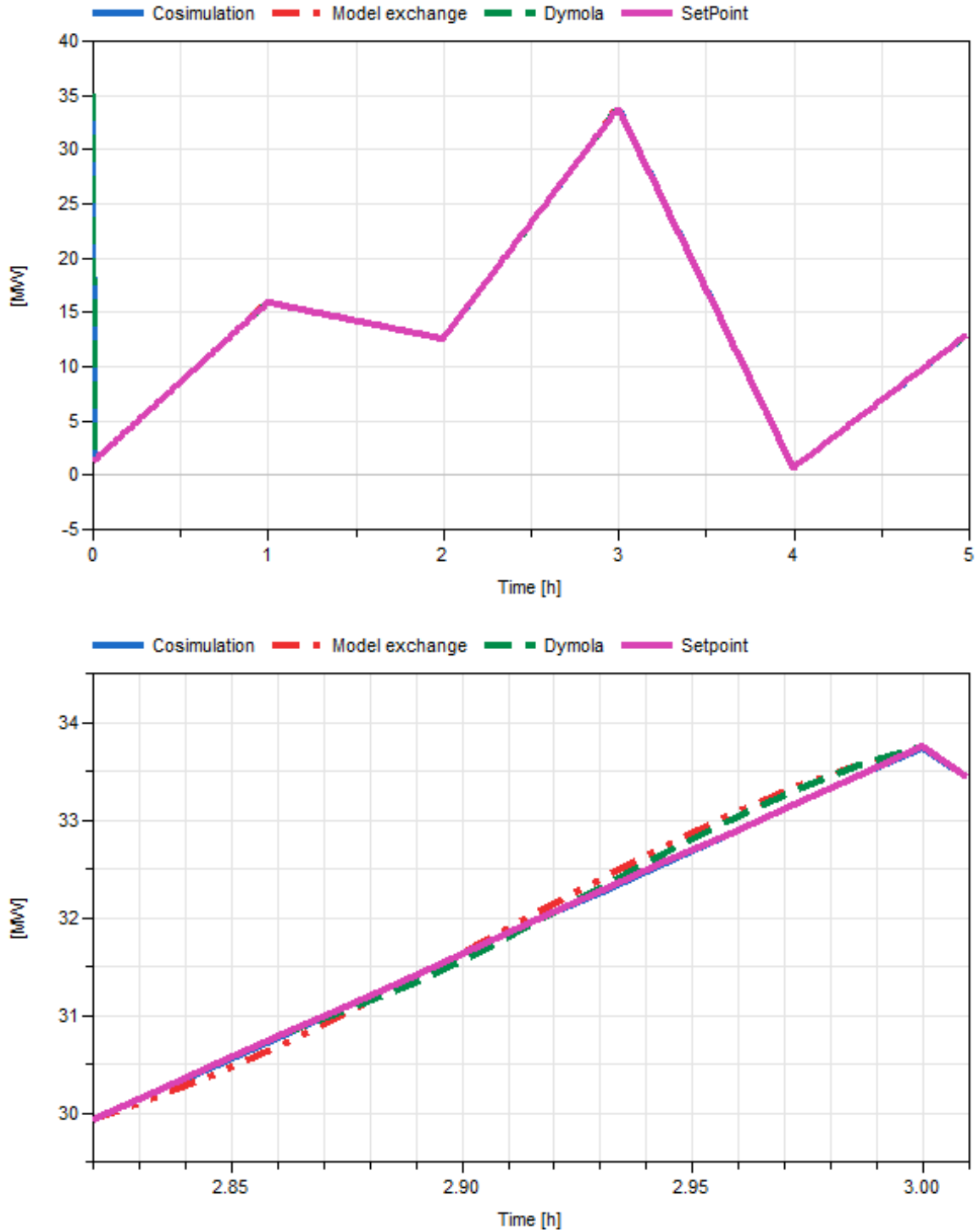


Figure 52. Top) Five-hour simulation of the natural gas turbine power vs. setpoint demand for the integrated energy park in regard to Modelica-only model, co-simulation FMI, and model exchange FMU. Bottom) Closeup shot of the turbine demand vs. turbine output for the different FMI versions. Note that all agree reasonably well. Co-simulation communication interval = 1 second.

7.2 Creation of Surrogate Using RAVEN

Due to the large increase in simulation time, the relative issues with co-simulation initialization routines, and the fact that there is no feedback to the rest of the grid, the FMI created for use in the RAVEN surrogate training was reduced to having only a single input (turbine demand) with no connected outputs. This setup allows the natural gas turbine to keep all the initialization pieces of the “infinite” grid self-contained, thus drastically improving the initialization routine and system robustness. Since the turbine power is a variable given by the FMU, and no feedback is used in other units’ control systems, the turbine power was not required to be an external variable for the initial export. The FMI/FMU (GTTP.fmu) exported to RAVEN is shown in Figure 53.

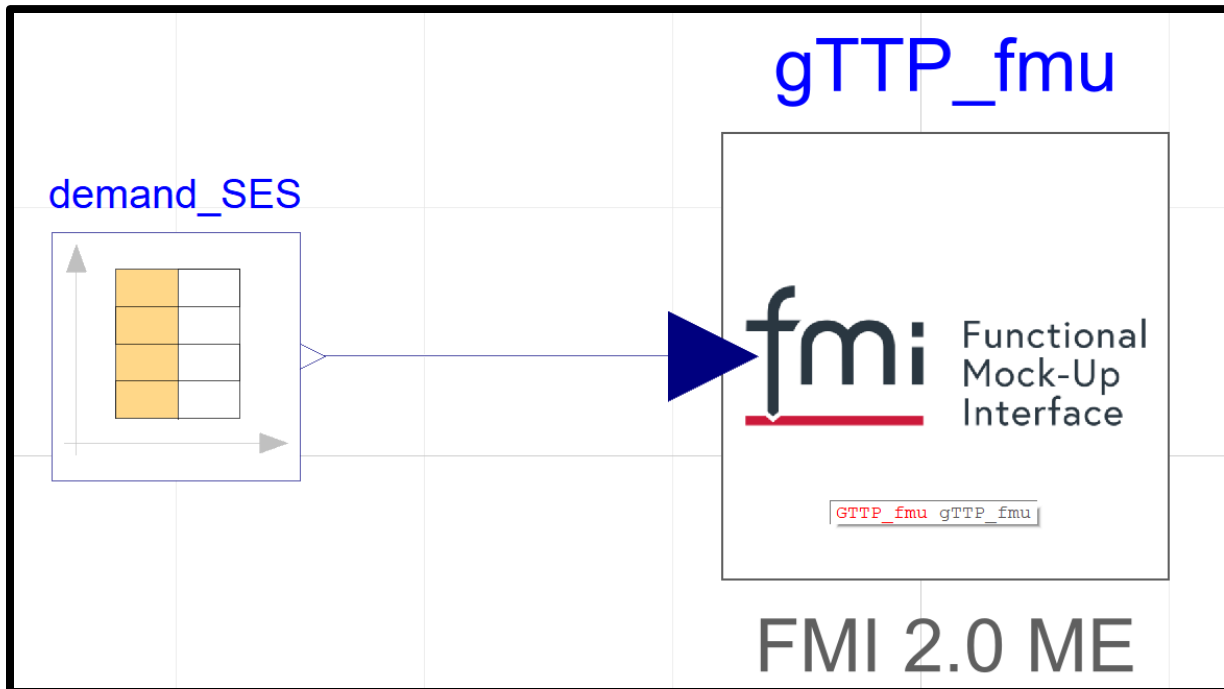


Figure 53. Simplified model of the FMI for RAVEN surrogation.

To construct a RAVEN-based AI to surrogate the response of the turbine component, the FMIFMU RAVEN importer described in Section 4 was used to drive the Dymola-exported FMI/FMU model.

Since the turbine’s response to changes in the demand is very quick (very limited inertia) and almost perfectly linear, a Support Vector Regressor with linear kernel Error! Reference source not found. was selected for surrogating the response. The turbine FMI/FMU GTTP.fmu was loaded via the FMIFMU RAVEN importer and its demand sampled (1,000 Monte Carlo samples) between 0 and 35 MW to capture the model’s full domain of variability. Finally, the Support Vector Regressor was trained (constructed) and exported to a “brand-new” FMI/FMU (GTTProm.fmu) by the RAVEN FMI/FMU exporter (co-simulation), as described in Section 6.2.

To validate the RAVEN AI FMI/FMU, a cross-validation assessment was performed in RAVEN, and, due to the pure linearity of both the turbine and AI models, its average R2 score was >0.99. This is further demonstrated in Figure 54 and Figure 55 which show a comparison of

the Dymola-generated turbine FMI/FMU and the RAVEN AI FMI/FMU, with the models demonstrating good agreement.

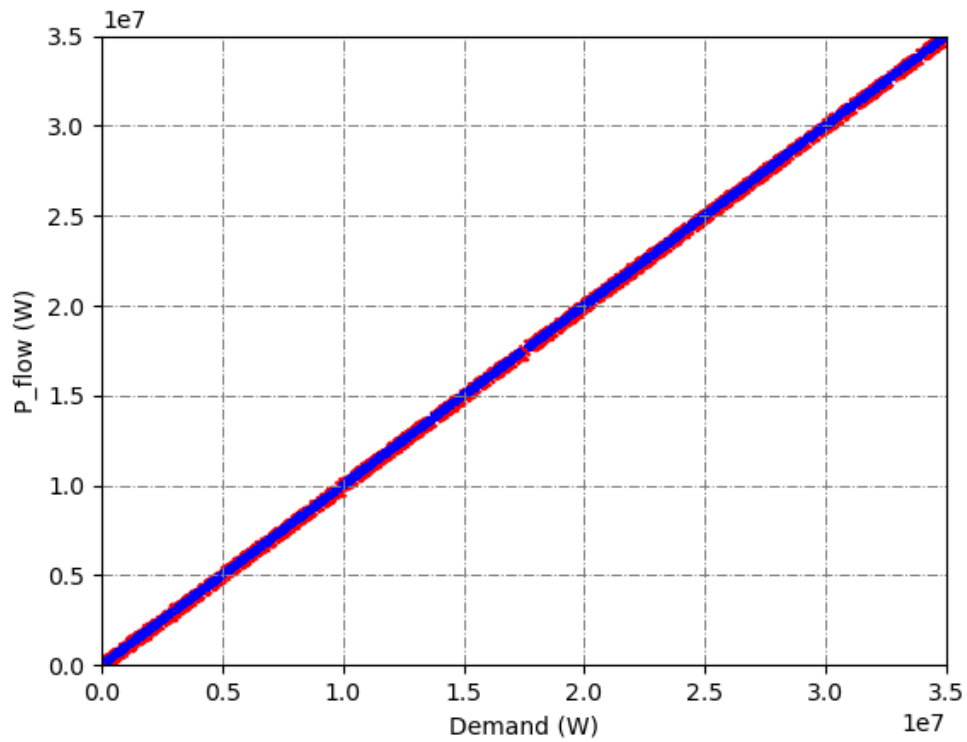


Figure 54. Comparison of the Modelica/Dymola GTTP.fmu response (P_flow) and the RAVEN AI-based GTTProm.fmu.

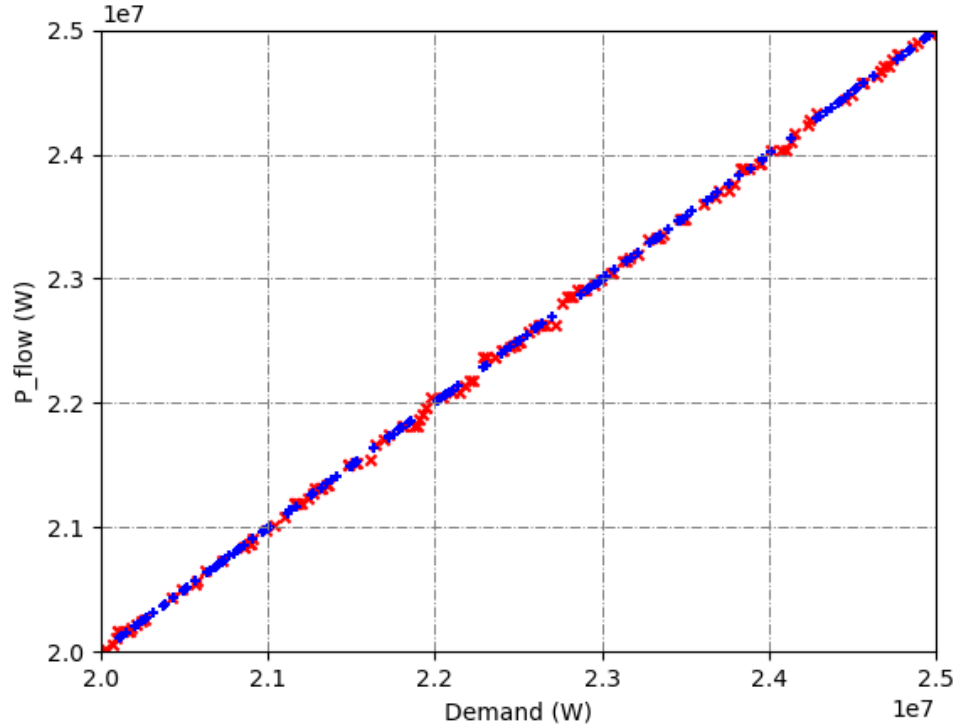


Figure 55. Closeup of the comparison of the Modelica/Dymola GTTP.fmu response (P_{flow}) and the RAVEN AI-based GTTProm.fmu.

7.3 Comparison of Results

Section 7.2 described the process of constructing an AI model exported in an FMI/FMU from RAVEN. To demonstrate the concept of the “plug-and-play” framework, along with the usage of AI for accelerated analysis, the integrated energy park model was simulated, both using the original Dymola model (FMI/FMU) for the gas turbine and using the RAVEN AI-based model.

Figure 56 shows the integrated energy park FMI/FMU exported via Dymola. Among the different variables and outputs is the model fulfillment of the gas turbine model’s demand. Such output represents the link between the IES park, and the turbine chosen for the demonstration.

Figure 57 and Figure 58 show the setup of the integrated energy park along with the detailed Dymola FMI/FMU and the RAVEN AI-based FMI/FMU, respectively. Both models were simulated in an ad-hoc *Python* code (master simulator) using the FMPy package.

Using the above-mentioned FMI/FMU setup, the two 5-hour simulations were run in the master simulator (*Python* code using FMPy). Since the RAVEN AI-based FMI/FMU can be evaluated in mere milliseconds, the simulation of the setup with the AI was much faster (~20%) to complete, making the computation time for the turbine evaluation completely negligible; indeed, the AI FMI/FMU almost zeroed out the CPU time for the turbine simulation, and the totality of the CPU time was used to simulate the remaining systems in the integrated energy park, which were more complex and computationally intensive.

Figure 59 and Figure 60 show a comparison of the turbine responses in the integrated energy park using the Dymola FMI/FMU and the RAVEN AI-based FMI/FMU. Over the course of the full five-hour simulation, all the results were in near-perfect agreement with the setpoints. The results show that the setup using the RAVEN AI FMI/FMU outperformed (in terms of speed) the

Dymola model, with no loss of accuracy.

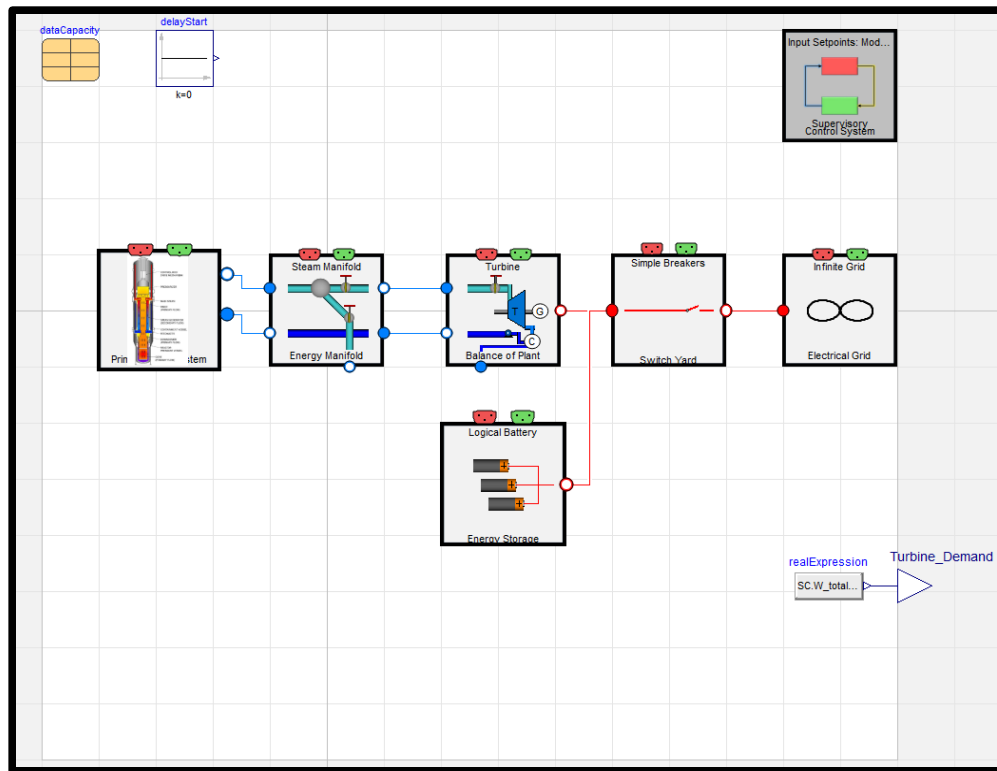


Figure 56. Integrated energy park (excluding the turbine) FMI/FMU generated with Dymola.

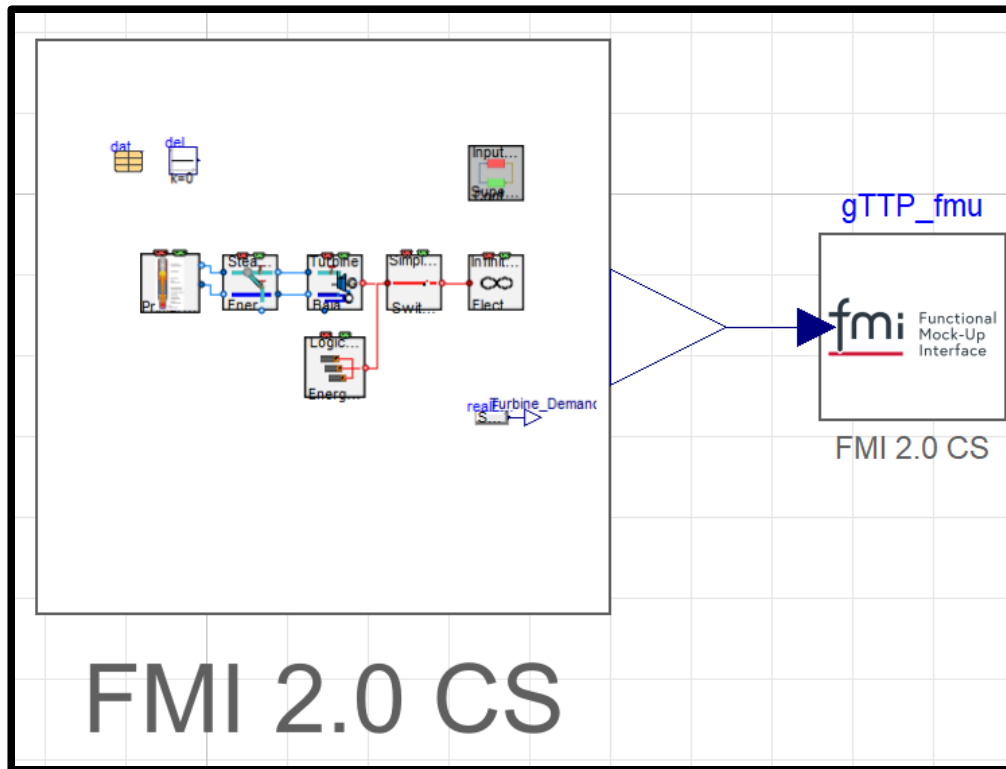


Figure 57. Integrated energy park FMI/FMU, including the Dymola GTTP model.

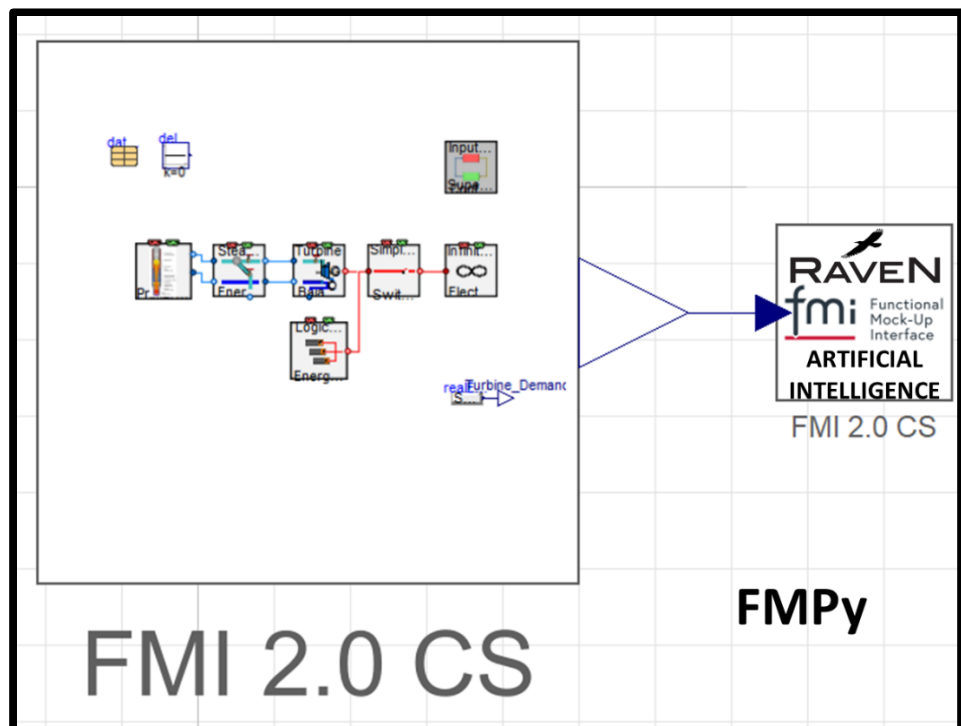


Figure 58. Integrated energy park FMI/FMU, replacing the Dymola GTTP model with the RAVEN AI-based FMI/FMU.

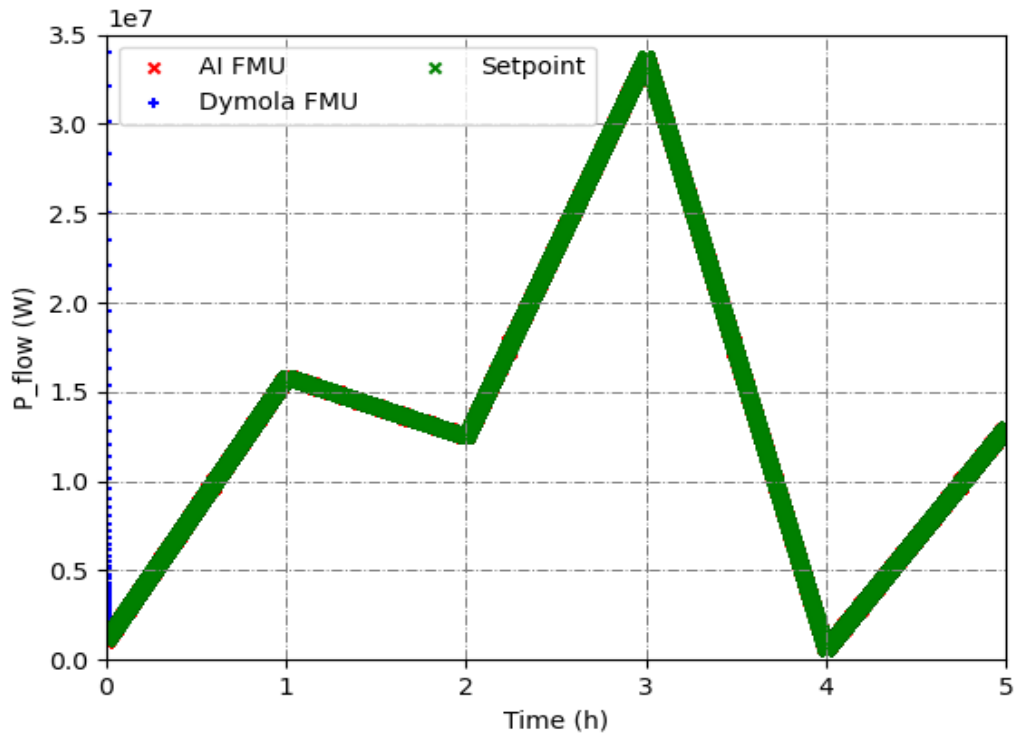


Figure 59. Co-Simulation FMI/FMU (Dymola) vs. RAVEN AI-based FMI/FMU for a 5-hour simulation of the turbine power vs. setpoint demand for the integrated energy park.

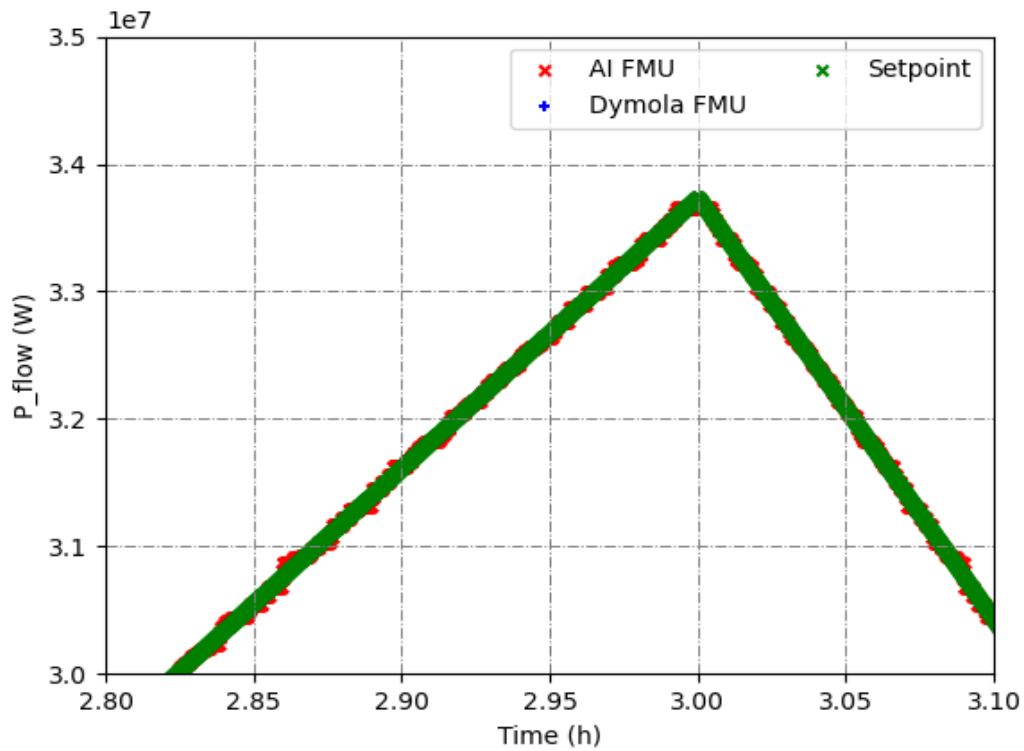


Figure 60. Co-Simulation FMI/FMU (Dymola) vs. RAVEN AI-based FMI/FMU closeup shot of turbine demand vs turbine output.

8. CONCLUSION

This report describes the status of the flexible plug-and-play framework development for design, analysis, and optimization of integrated energy systems. This framework seeks to integrate Modelica and Dymola with RAVEN in terms of both FMI/FMU construction and repository structures intended to simplify model sharing and simulation of complex dynamic systems.

The report provides an in-depth look at the alterations needed to modify existing system-level models for exportation as FMUs. These alterations include modifying specialty “port” variables into their constituent parts as real variables via a new FMI adaptor package added to the existing HYBRID repository. This package includes new adaptors for electrical, fluid, and heat ports for export into the FMIs/FMUs. Examples were included within the FMU adaptor package, illustrating how to properly utilize the system. Several of these examples are discussed in Section 2 of this report.

Simulation results demonstrate that, while minor differences may occur, the overall control, trends, and solution integrity is maintained between the standard Modelica simulation and FMU simulation results. However, it is worth noting that, for small systems, the FMU results have a slower simulation time than the Modelica-only simulation. While this step-by-step process does require several levels of checks, it provides a degree of system flexibility never before experienced. Using this process, a company can provide models that contain proprietary information to separate entities, without disclosing any information about the model that could be considered business sensitive. Such a capability would allow institutions to bypass the necessity of having to “whitewash” data.

In addition to the investigative work being conducted on FMUs and FMIs, a series of updates to the HYBRID repository regression system was completed to ready the repository for open-sourcing. These updates include additional system-level tests for components in the HYBRID repository, as well as increasing the testing level from a mere six tests to 32 and counting. Further, new features have been included in the testing system, such as an initialization subroutine for Dymola models that helps highly nonlinear complex systems initiate their regression test. Additionally, the output keys “numberOfIntervals” and “OutputInterval” were added to the regression system, allowing for consistent comparison points between the reference file and the simulation results between machines. This step is necessary because the commercial Modelica platform Dymola has a series of global output flags that are rarely consistently utilized from one organization to another, yet do not change the trajectories of the solution.

Finally, the work that was deployed to simulate, export, and use FMI/FMU in conjunction with AI algorithms in RAVEN represents a significant step forward in regard to delivering a streamlined process to accelerate simulations and analysis by leveraging RAVEN advanced algorithms. The possibility of using AI exported in FMI/FMU in any FMI/FMU-compatible framework (e.g., FMPy and Dymola) is unique to this framework, posing the basis for deployment of fast simulation, modeling, and analysis accelerations.

Overall, extensive work was completed to develop FMUs and FMIs from existing models and gaining greater understanding of the requirements and limitations of FMI/FMUs.

9. FUTURE WORK

The activities described in this report show the potential of the concept of a “flexible plug-and-play ecosystem” being developed within the IES program and deployed via the creation of FORCE. In order to fulfill the promises of FORCE, several tasks are planned to be carried out in the future of the program:

- 1) Master Simulator development in RAVEN: in order to automate the deployment of models in a system that is compatible with any FMI/FMU interface, an entity (Master Simulator) needs to be developed within RAVEN. Such development will allow for the simulation of FMI/FMU models (AI, Dymola, etc.) directly within the RAVEN framework allowing for the integration of such models in any RAVEN workflow, in general, and in IES technoeconomic analysis, in particular. The Master Simulator in RAVEN will be based on the EnsembleModel entity (see sec. 5.2), in conjunction with the FMPy library. The Master Simulator is shown in Figure 61.

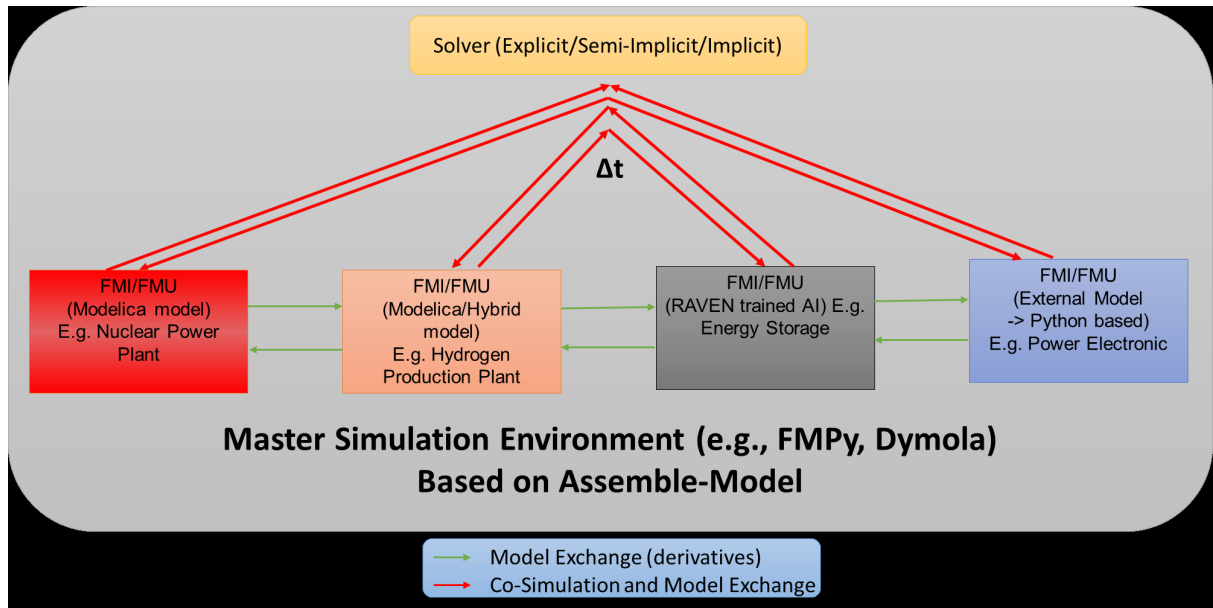


Figure 61. Proposed master simulator within RAVEN.

- 2) Model Exchange for RAVEN-based models: in section 6.2 the deployment of a system for exporting RAVEN-based AI as FMI/FMU leveraging the PythonFMU library has been shown. However, the current library only supports FMI/FMU in co-simulation, useful for loosely coupled models but inadequate for tightly coupled systems. To allow for exporting of nonlinear models (e.g. Nuclear Reactor Balance of Plant, Storage, etc.), the PythonFMU library needs to be upgraded to allow for exporting models in model exchange and, consequentially, leverage the capability of RAVEN AI to provide first and second order derivative information.
- 3) Integration of the FARM supervisory control model: Argonne National Laboratory, in collaboration with Idaho National Laboratory, recently released a RAVEN plugin called Feasible Actuator Range Modifier (FARM) [14],[15]. This plugin oversees deploying supervisory bounding control for dynamic models to ensure physical limitations of the

model are not exceeded. This is an additional layer of control on top of the existing physical modeling control systems. While control is still imposed for each individual process, FARM can identify demand signals that cannot be met within safety limits and augments the demand to meet safety specifications. For the FORCE framework to deploy these supervisory controllers the model needs to be exported as FMI/FMU and integrated into the plug-and-play framework.

- 4) Integration of the HERON plugin: INL has been developing the Holistic Energy Resource Optimization Network (HERON) plugin to construct workflows for solving resource allocation problems inherent to the electrical grid. This plugin oversees the allocation of energy resources within integrated energy systems. The idea of FORCE is to connect HERON with FARM, RAVEN, and HYBRID to solve real world energy allocation problems. With the work completed in FY 2020 the next step is to develop the interconnection between these different platforms and ensure simulation speed is capable of solving real world problems.

Additional investigative work is planned in order to expand the FMU capabilities within the existing HYBRID repository framework.

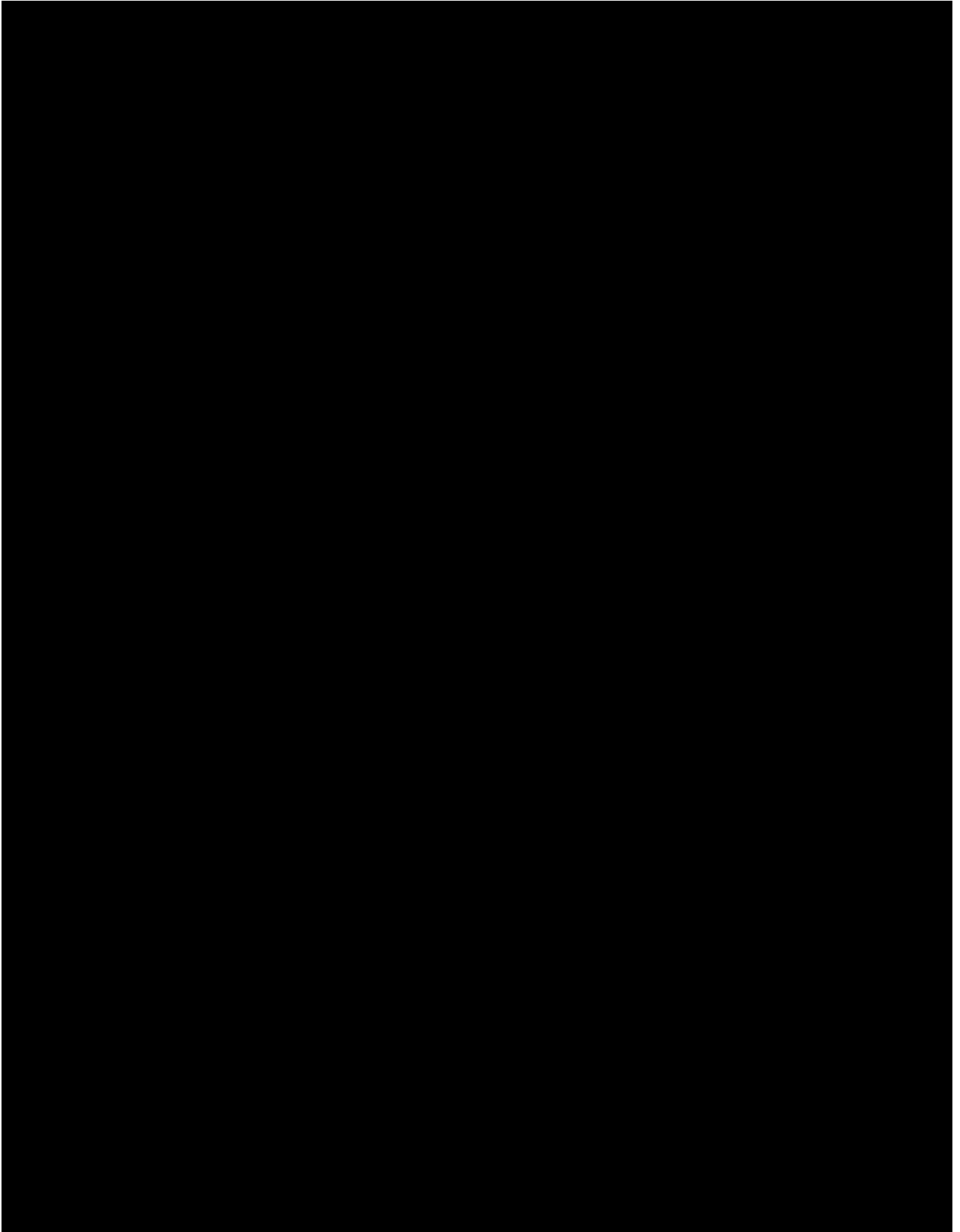
10. REFERENCES

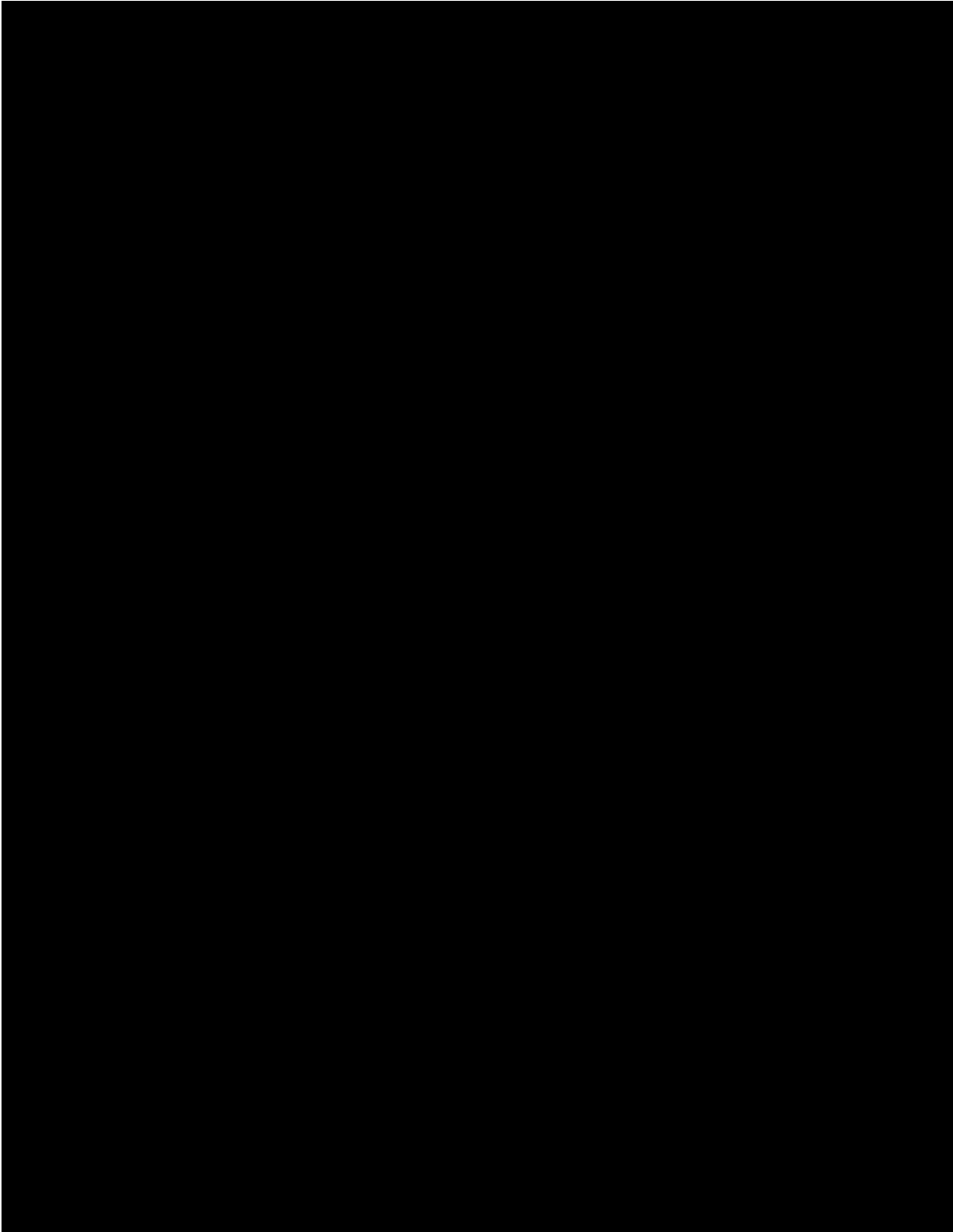
- [1] C. Rabiti, A.S. Epiney, P. Talbot, J.S. Kim, S. Bragg-Sitton, A. Alfonsi, A. Yigitoglu, S. Greenwood, S.M. Cetiner, F. Ganda, G. Maronati. September 2017. "Status Report on Modeling and Simulation Capabilities for Nuclear-Renewable Hybrid Energy Systems." INL/EXT-17-43441, Idaho National Laboratory.
- [2] J.S. Kim, M. McKellar, S. Bragg-Sitton, R. Boardman. October 2016. "Status Report on the Component Models Developed in the Modelica Framework: High-Temperature Steam Electrolysis & Gas Turbine Power Plant." INL/EXT-16-40305, Idaho National Laboratory.
- [3] J.S. Kim, K.L. Frick. May 2018. "Status Report on the Component Models Developed in the Modelica Framework: Reverse Osmosis Desalination Plant & Thermal Energy Storage." INL/EXT-18-45505, Idaho National Laboratory.
- [4] K.L. Frick. August 2019. "Status Report on the NuScale Module Development in the Modelica Framework." INL/EXT-19-55520, Idaho National Laboratory.
- [5] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, C. Wang, P.W. Talbot, D.P. Maljovec, C. Smith. 2016. "RAVEN Theory Manual and User Guide." INL/EXT-16-38178, Idaho National Laboratory.
- [6] C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, S. Sen, C. Wang, J. Chen. 2017. "RAVEN User Manual." INL/EXT-15-34123, Idaho National Laboratory.
- [7] Dassault Systems. "DYMOLA Systems Engineering: Multi-Engineering Modeling and Simulation Based on Modelica and FMI." Accessed July 24, 2020. <https://www.3ds.com/products-services/catia/products/dymola/>.
- [8] M.S. Greenwood: TRANSFORM - TRANSient Simulation Framework of Reconfigurable Models. Computer Software. <https://github.com/ORNLModelica/TRANSFORM-Library>. 07 Nov. 2017. Web. Oak Ridge National Laboratory. doi:10.11578/dc.20171109.1. Available: <https://github.com/ORNLModelica/TRANSFORM-Library>.
- [9] K. Frick, A. Alfonsi, C. Rabiti. 2020. "Hybrid User Manual." INL/MIS-20-60624, Idaho National Laboratory.

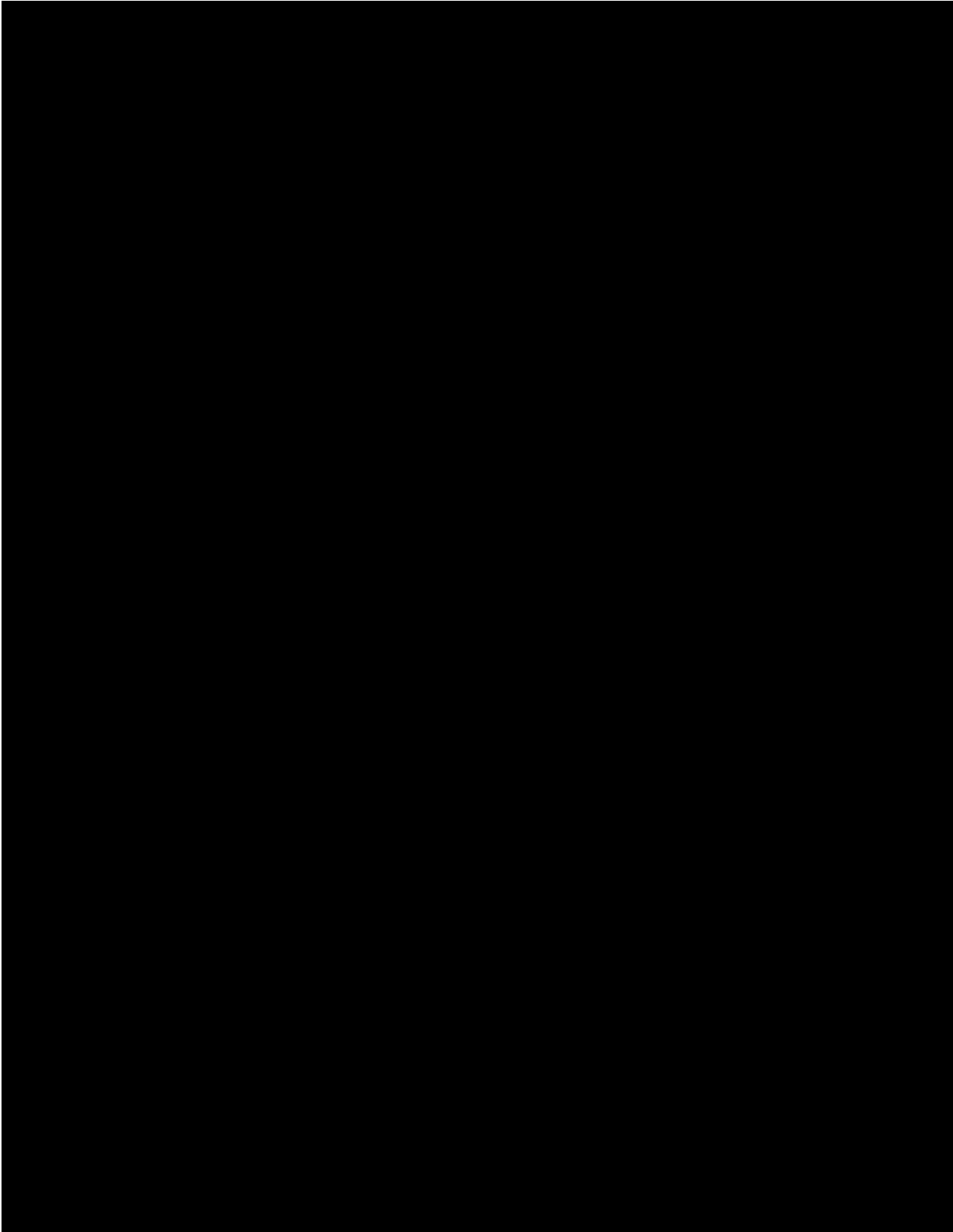
- [10] A. Alfonsi, K. Frick, S. Greenwood, C. Rabiti. 2020. “Status on the Development of the Infrastructure for a Flexible Modelica/RAVEN Framework for IES.” INL/EXT-20-00160, Idaho National Laboratory.
- [11] K. Frick, A. Alfonsi, C. Rabiti. 2020. “Flexible Modelica/RAVEN Framework for IES.” INL/EXT-20-00419, Idaho National Laboratory.
- [12] FMPy. 2020. “FMPy 0.2.21.” Accessed July 8, 2020. <https://pypi.org/project/FMPy/>.
- [13] PyFMI. 2018. “PyFMI 2.5.” Accessed March 25, 2020. <https://pypi.org/project/PyFMI/>.
- [14] H. Wang, R. Ponciroli, A. Alfonsi. Feasible Actuator Range Modifier (FARM). <https://github.com/Argonne-National-Laboratory/FARM>
- [15] H. Wang, R. Ponciroli, R. Vilim, A. Alfonsi, C. Rabiti. “A Recursive Data-Driven Approach to State Variable Selection and Digital Twin Derivation.” in *12th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies. (NPIC&HMIT 2021)*. June 2021.

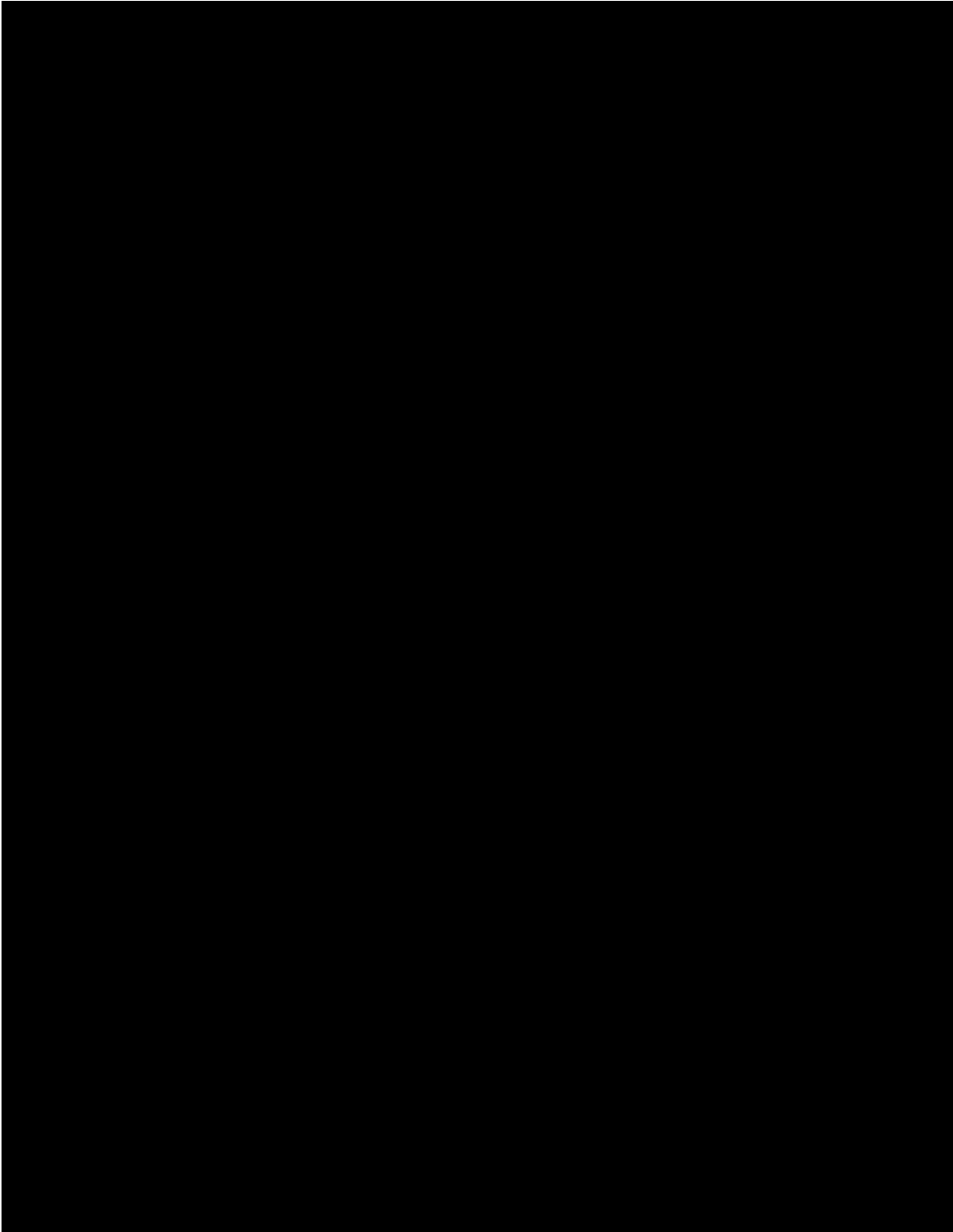
APPENDIX A – HYBRID USER MANUAL

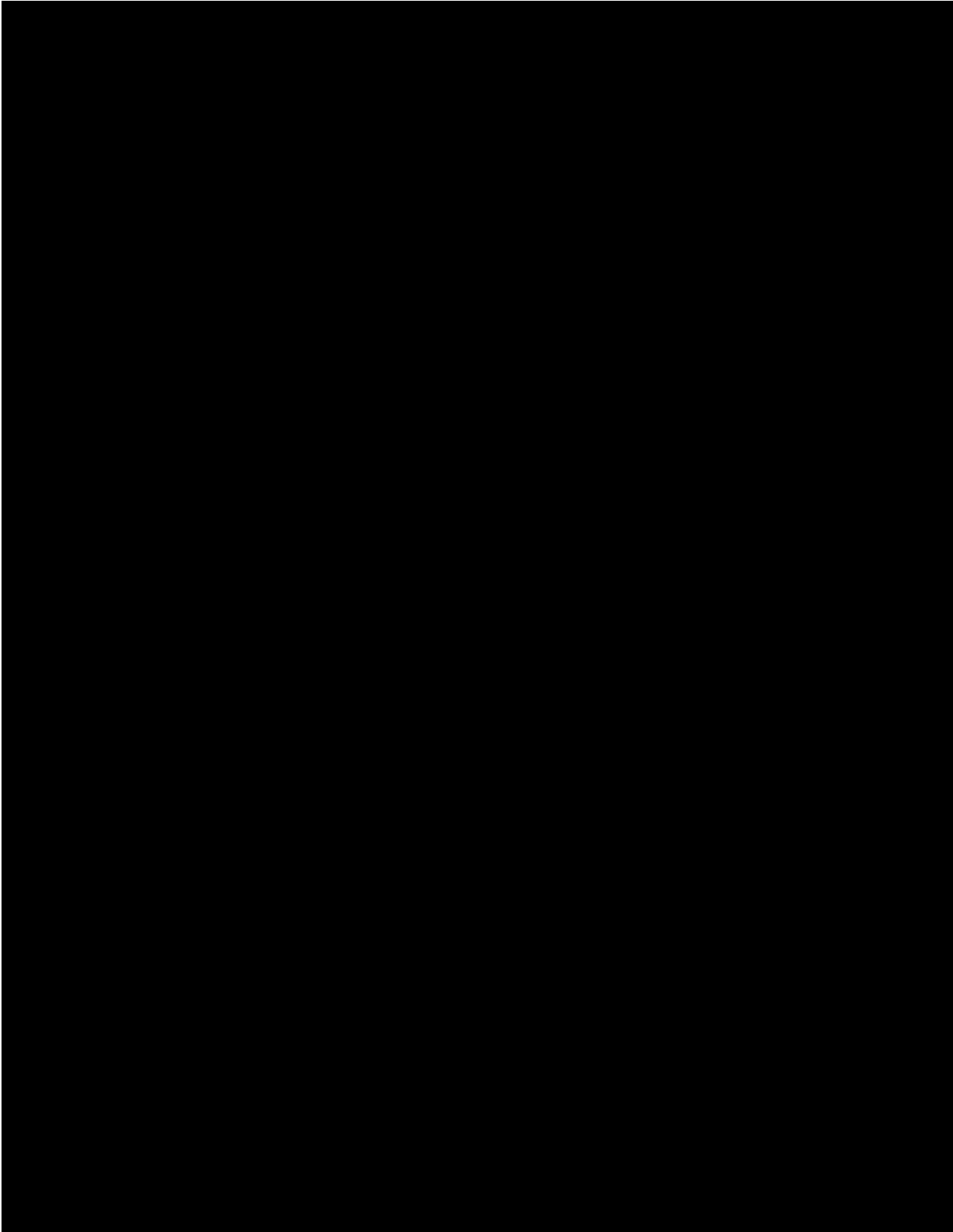


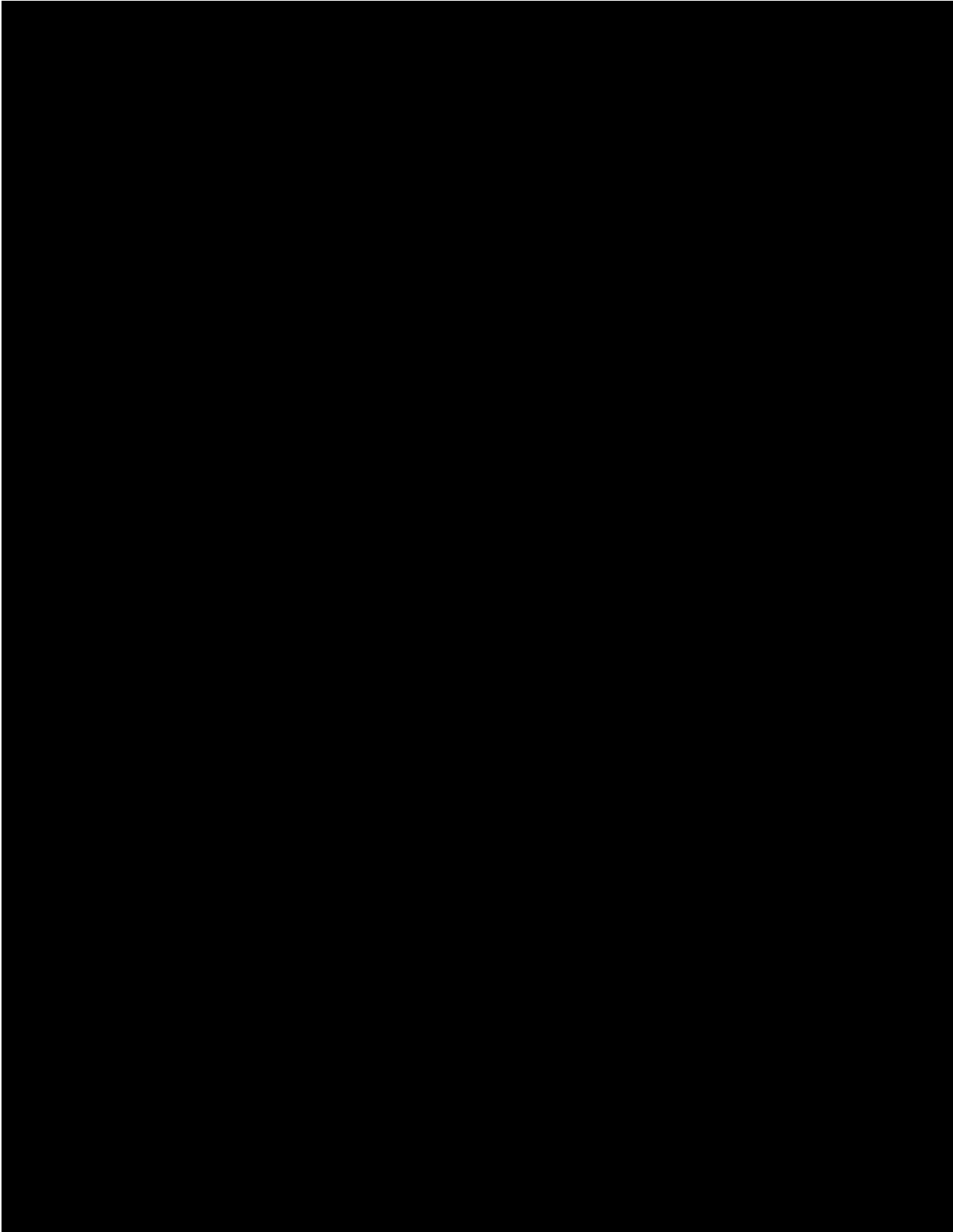


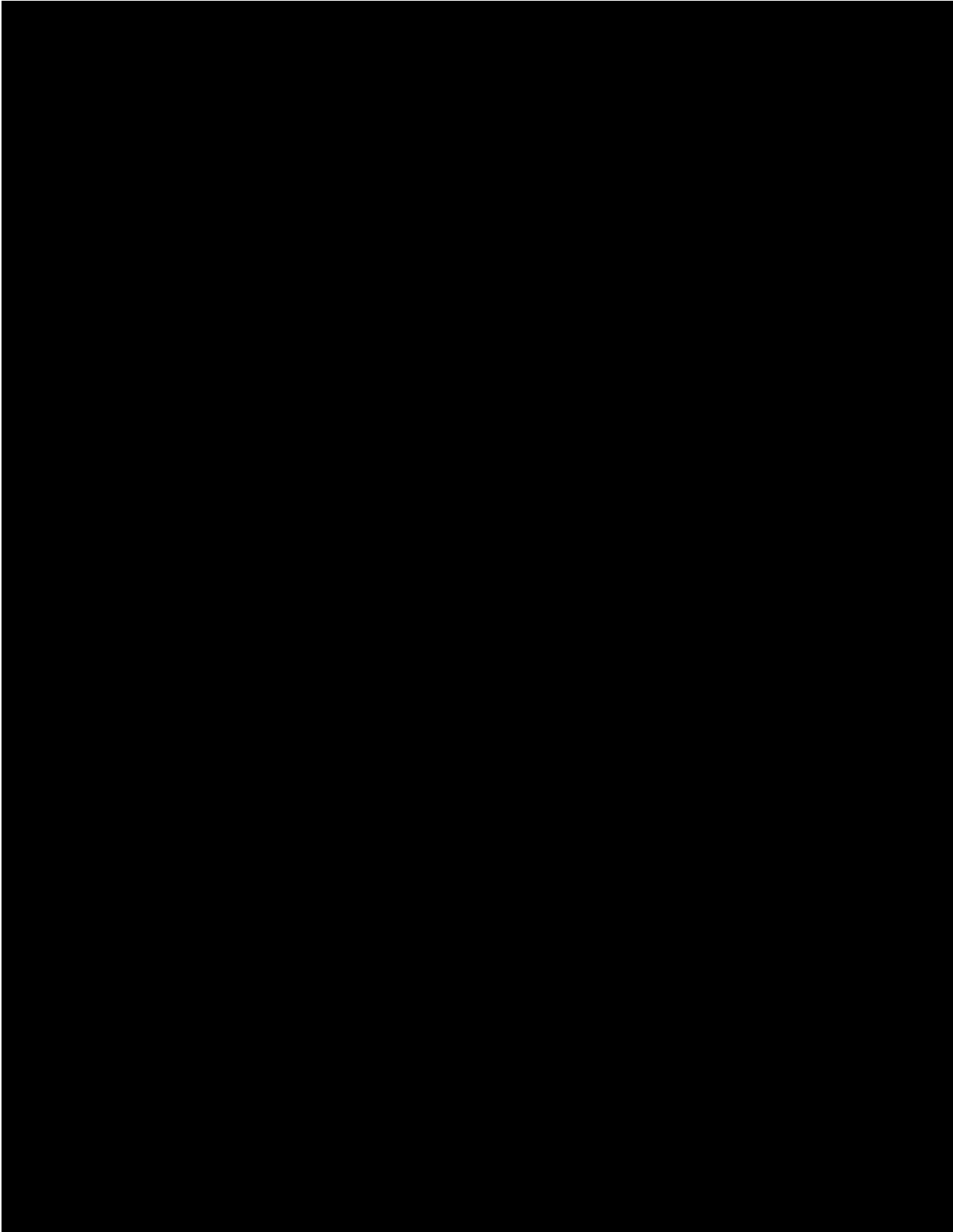


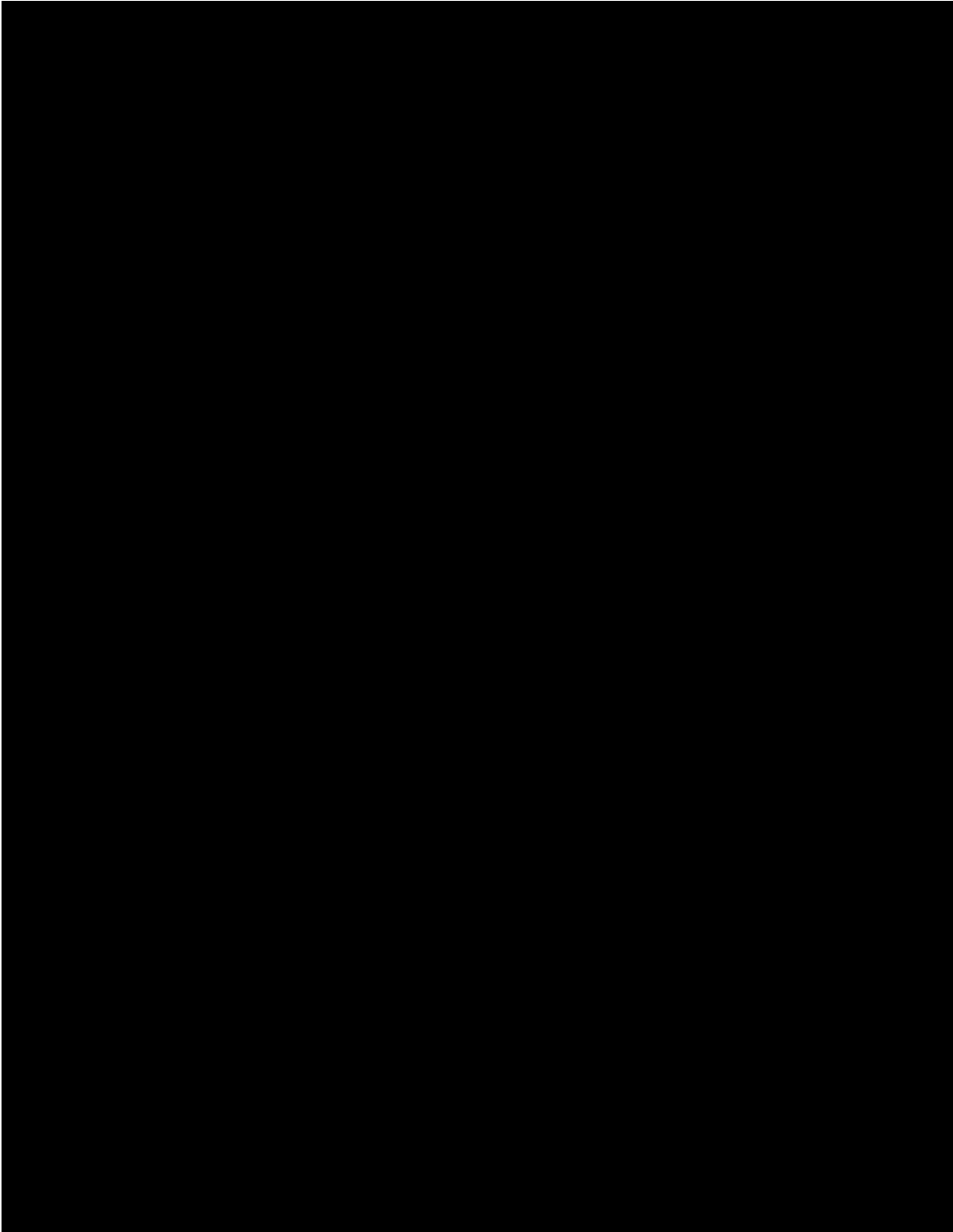


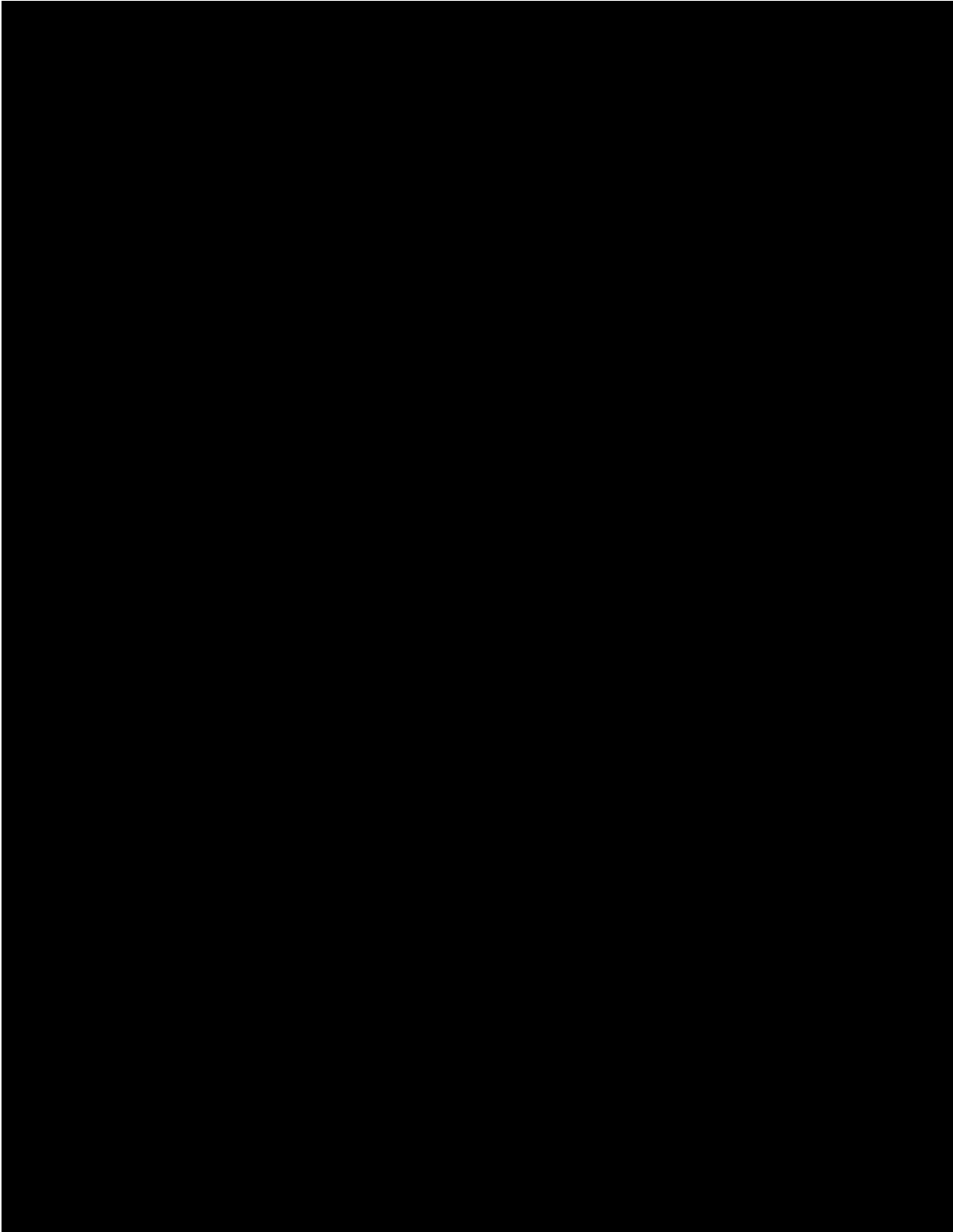


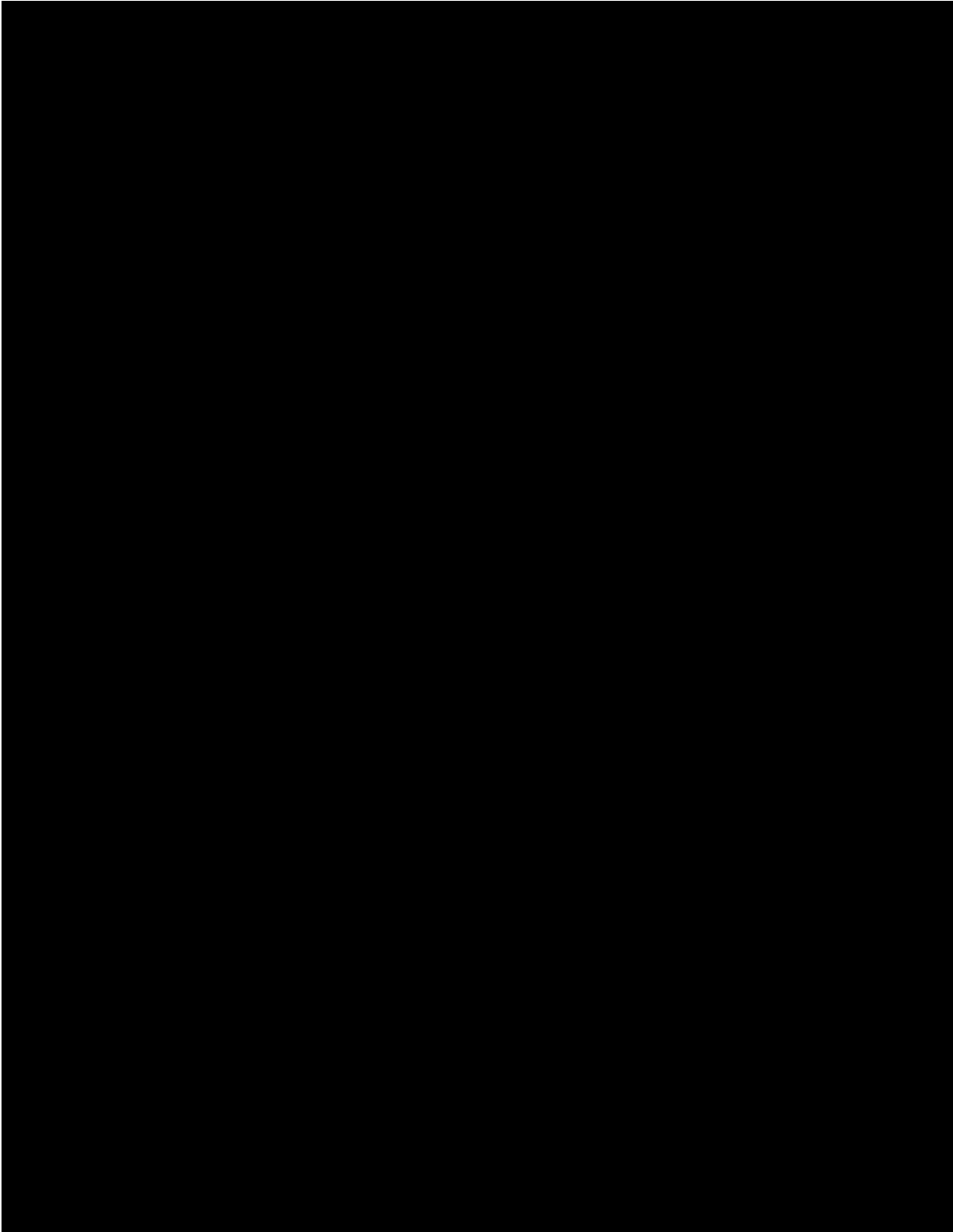


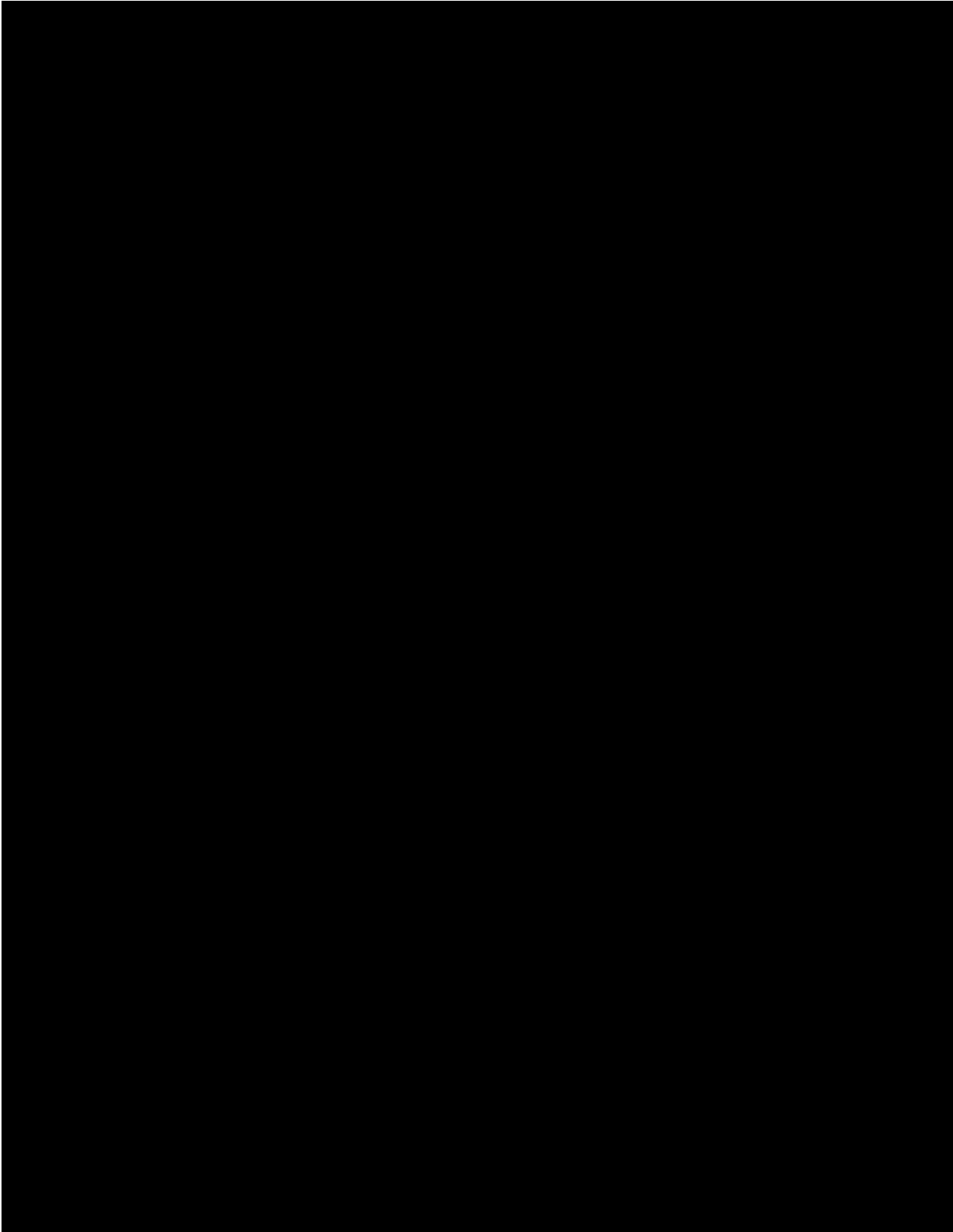


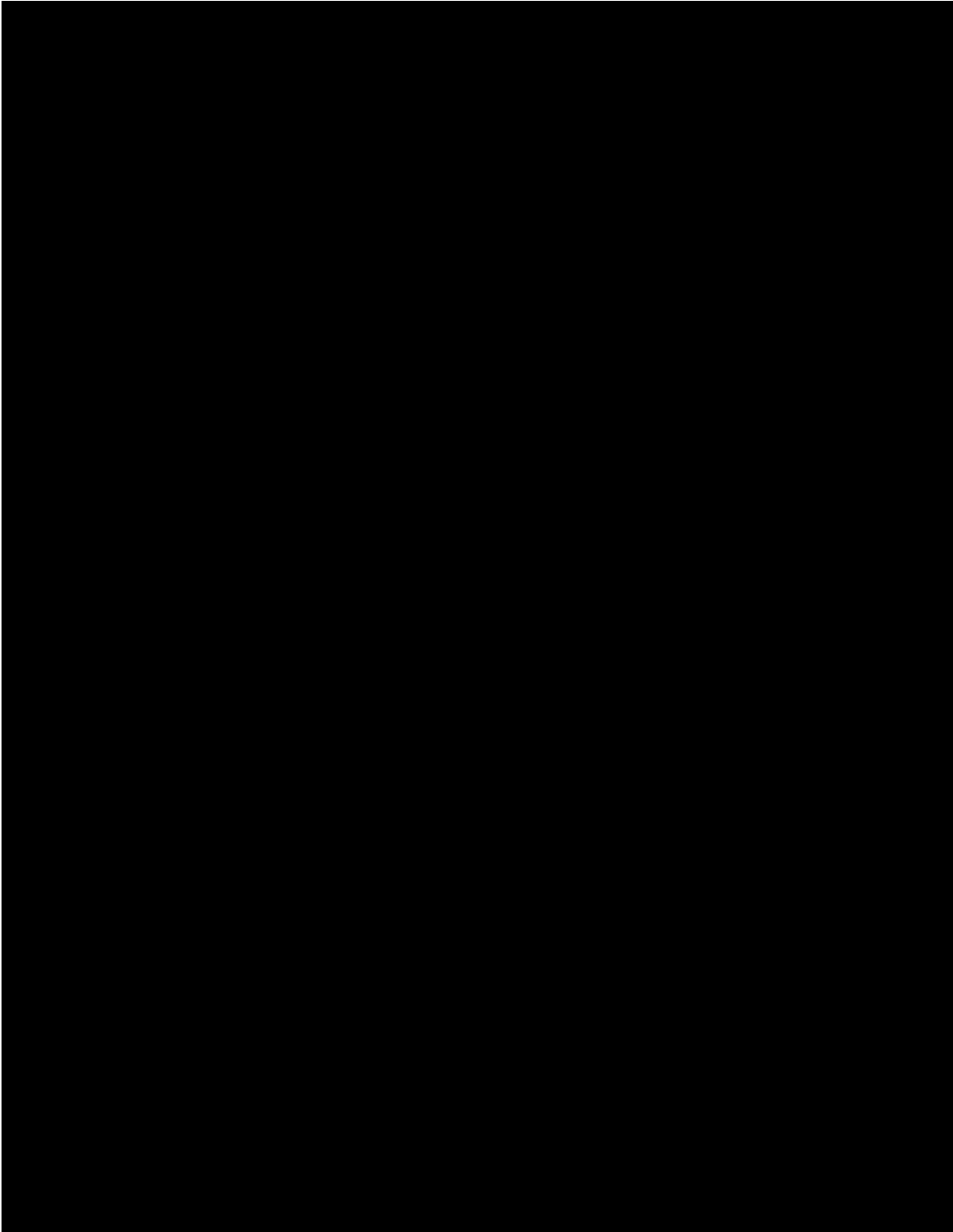


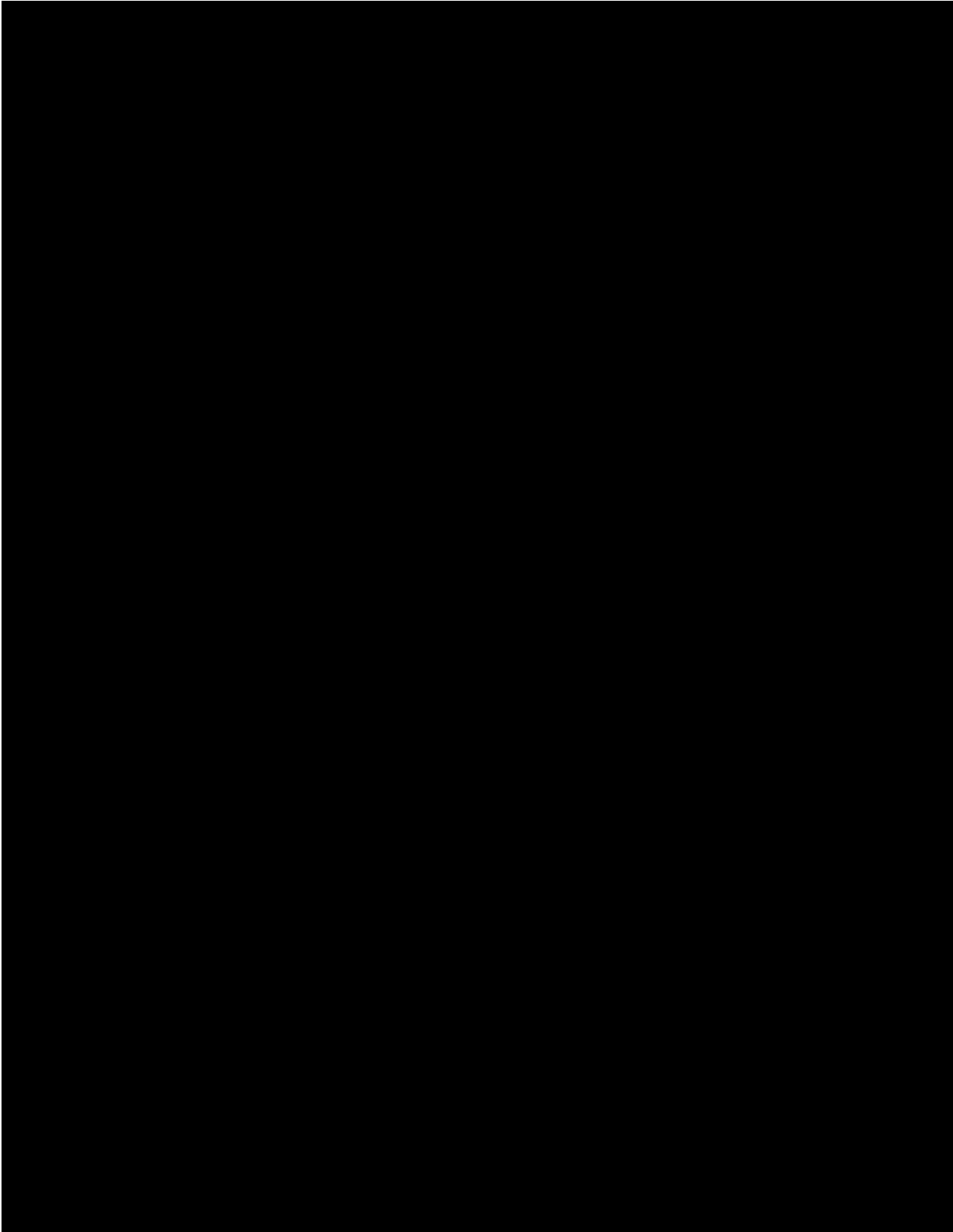


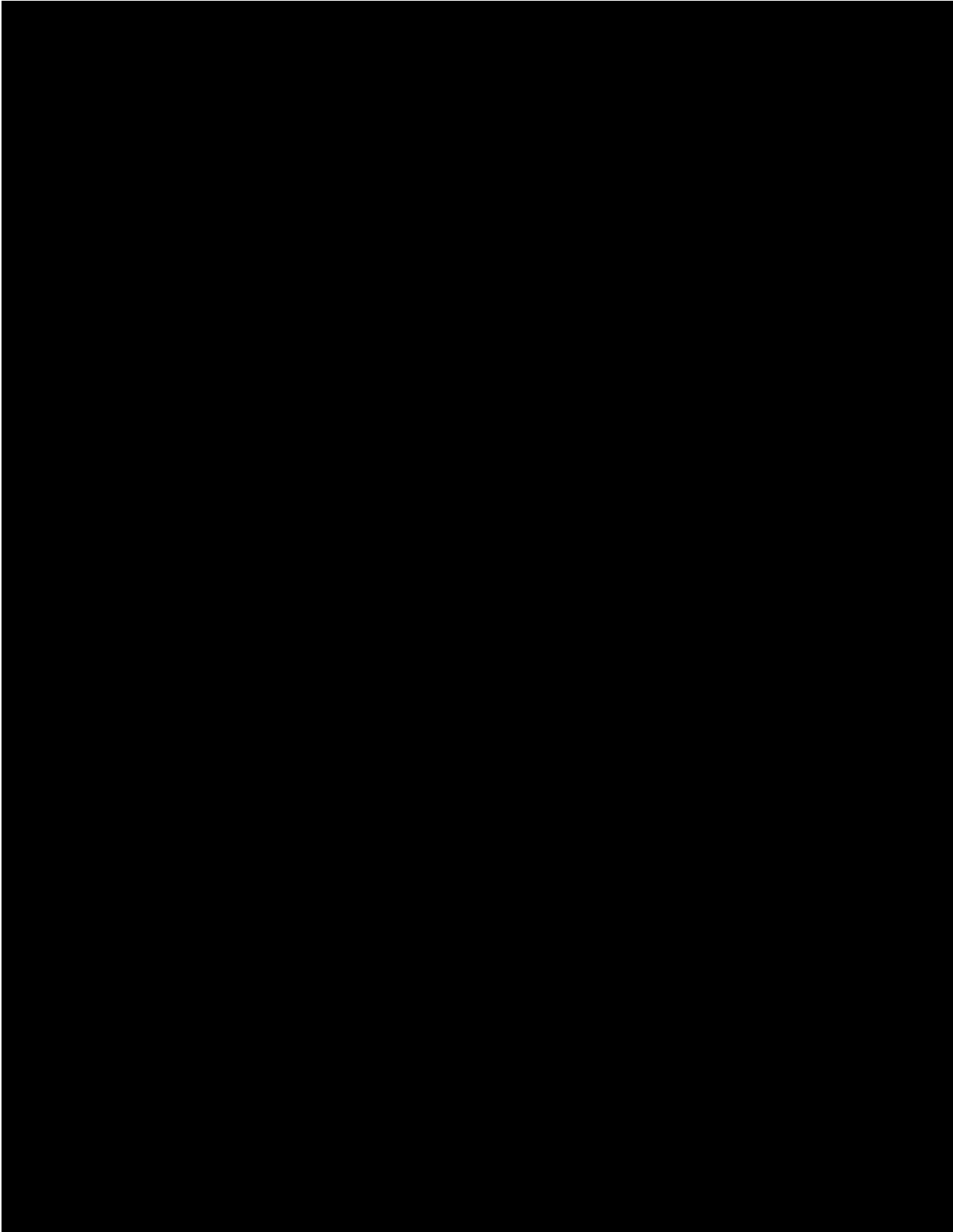


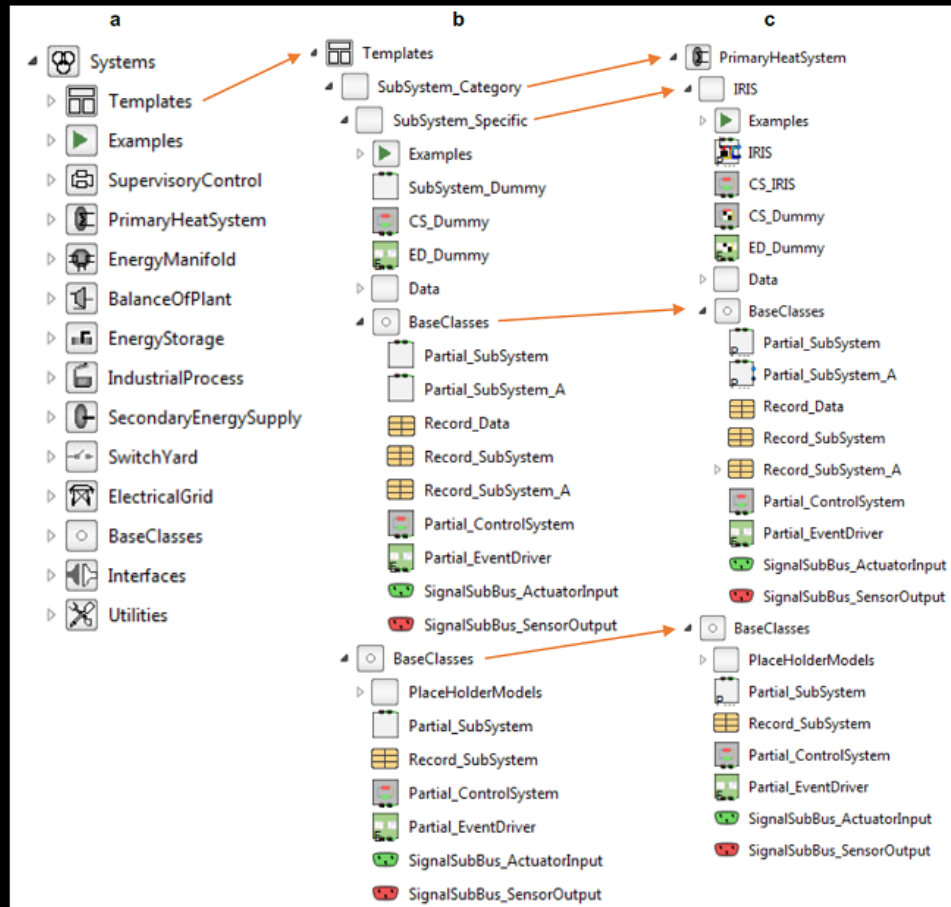


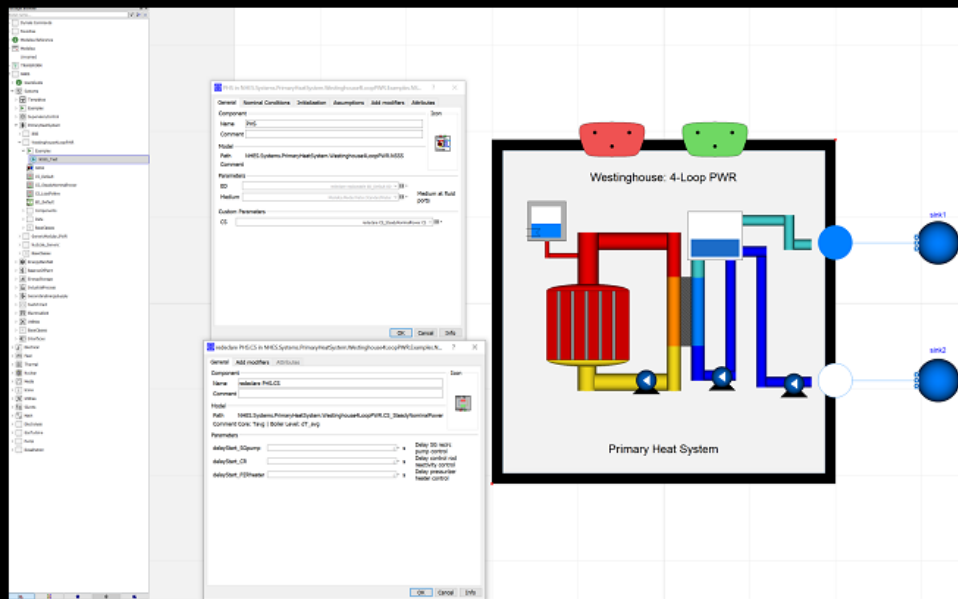


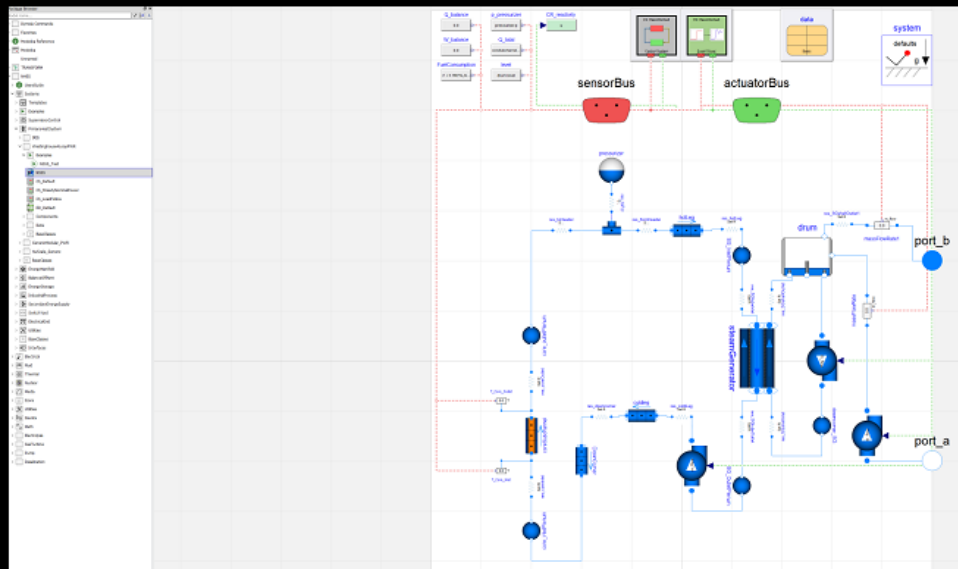


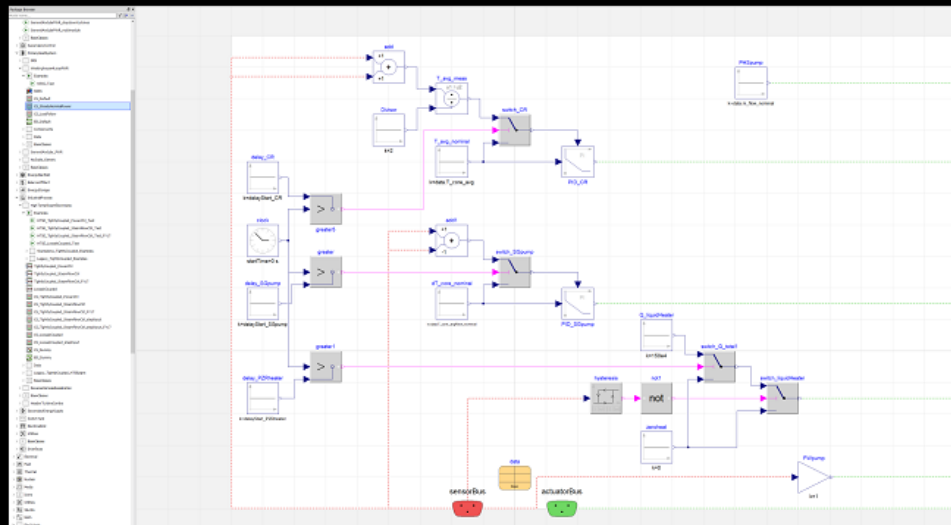


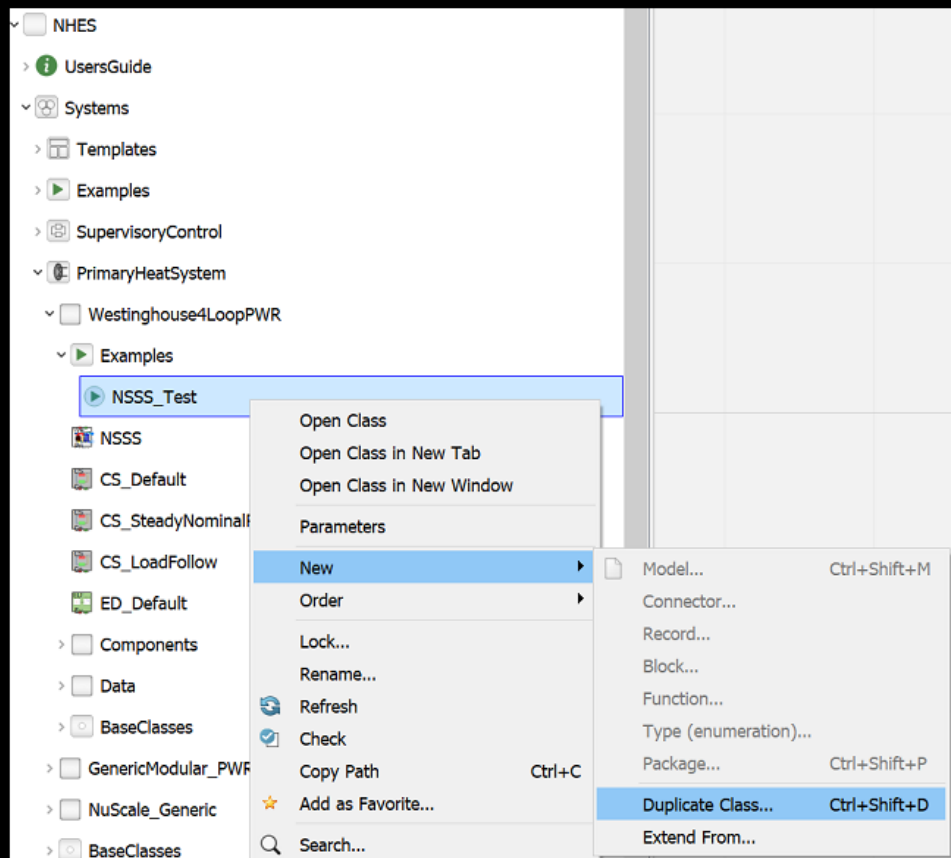


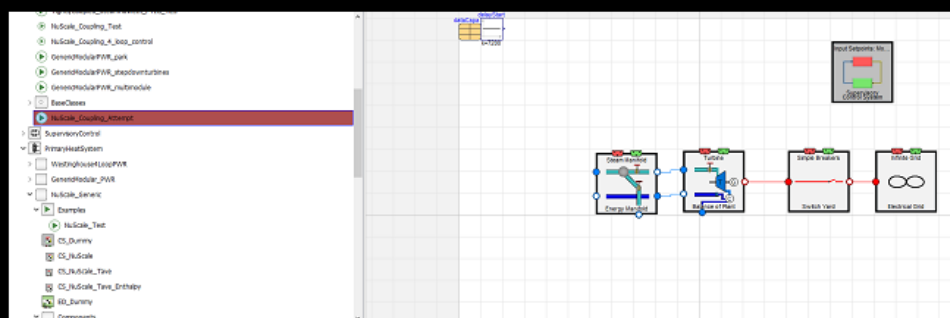


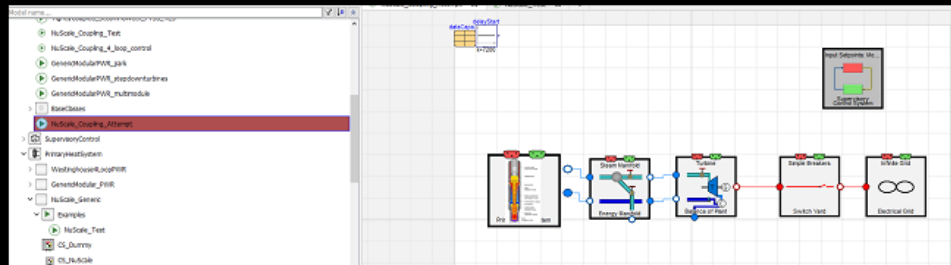












redeclare nuScale_Tave_enthalpy.CS in NHES.Systems.Examples.NuScale_Coupling_Attempt ? X

General Add modifiers Attributes

Component

Name redeclare nuScale_Tave_enthalpy.CS

Comment

Model

Path NHES.Systems.PrimaryHeatSystem.NuScale_Generic.CS_NuScale_Tave

Comment

Icon

Inputs

W_turbine	BOP.powerSensor.power	W	Turbine Output
W_Setpoint	SC.W_totalSetpoint_BOP	W	Turbine Setpoint

OK Cancel Info

redeclare BOP.CS in NHES.Systems.Examples.NuScale_Coupling_Attempt

General Add modifiers Attributes

Component

Name redeclare BOP.CS


Comment

Model

Path NHES.Systems.BalanceOfPlant.Turbine.CS_OTSG_Pressure

Comment

Icon



Parameters

delayStartTCV	300	s	Delay start of TCV control
delayStartBV	delayStartTCV	s	Delay start of BV control
p_nominal	BOP.port_a_nominal.p	Pa	Nominal steam turbine pressure
TCV_opening_nominal	0.5		Nominal opening of TCV - controls power
BV_opening_nominal	0.001		Nominal opening of BV - controls pressure

Inputs

W_totalSetpoint	SC.W_totalSetpoint_BOP	W	Total setpoint power from BOP
Reactor_Power	160	MW	Reactor Power Level
Nominal_Power	160	MW	Nominal Power Level

OK Cancel Info

EM.port_a1_nominal in NHES.Systems.Examples.NuScale_Coupling_Attempt ? X

General Add modifiers Attributes

Component

Name EM.port_a1_nominal

Comment

Model

Path NHES.Systems.BaseClasses.Record_fluidPorts

Comment

Parameters

p	nuScale_Taveprogram.port_b_nominal.p	Pa	Absolute pressure
h	nuScale_Taveprogram.port_b_nominal.h	J/kg	Specific enthalpy
m_flow	-nuScale_Taveprogram.port_b_nominal.m_flow	kg/s	Mass flow rate

OK Cancel Info

EM.port_b1_nominal in NHES.Systems.Examples.NuScale_Coupling_Attempt ? X

General Add modifiers Attributes

Component

Name EM.port_b1_nominal

Comment

Model

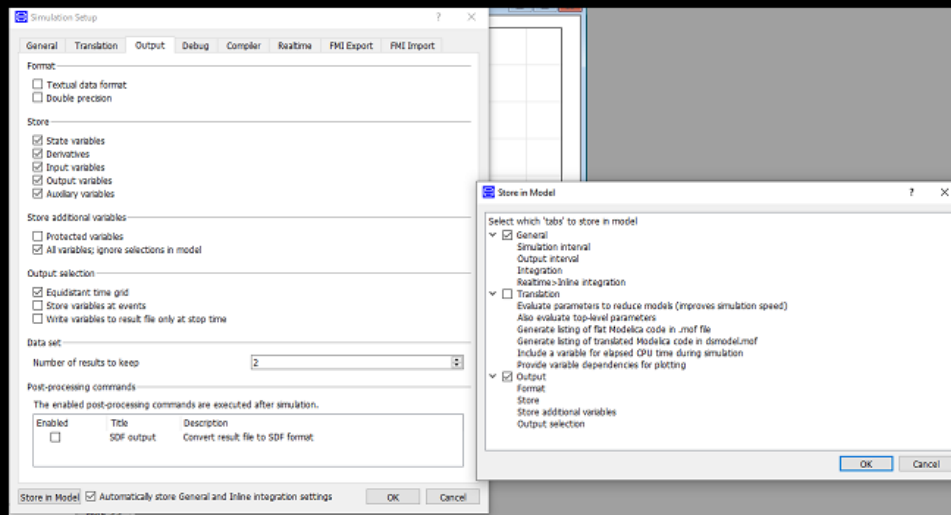
Path NHES.Systems.BaseClasses.Record_fluidPorts

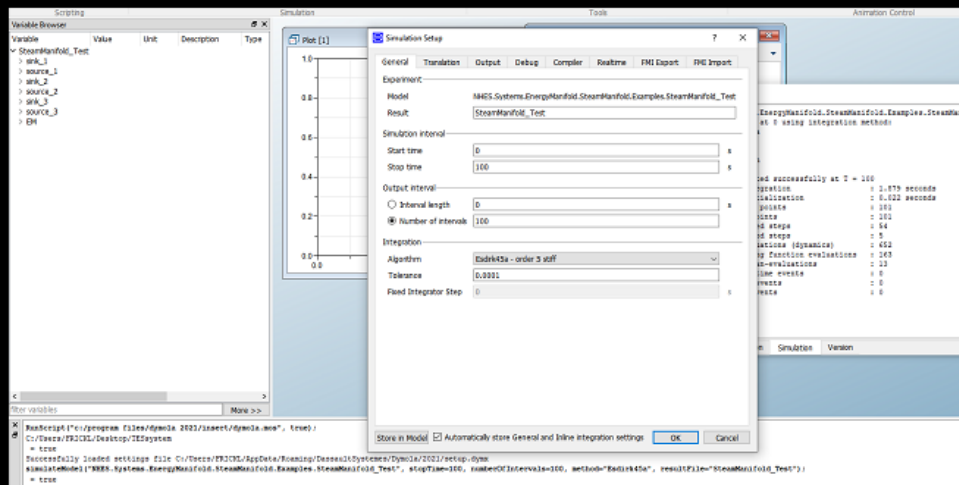
Comment

Parameters

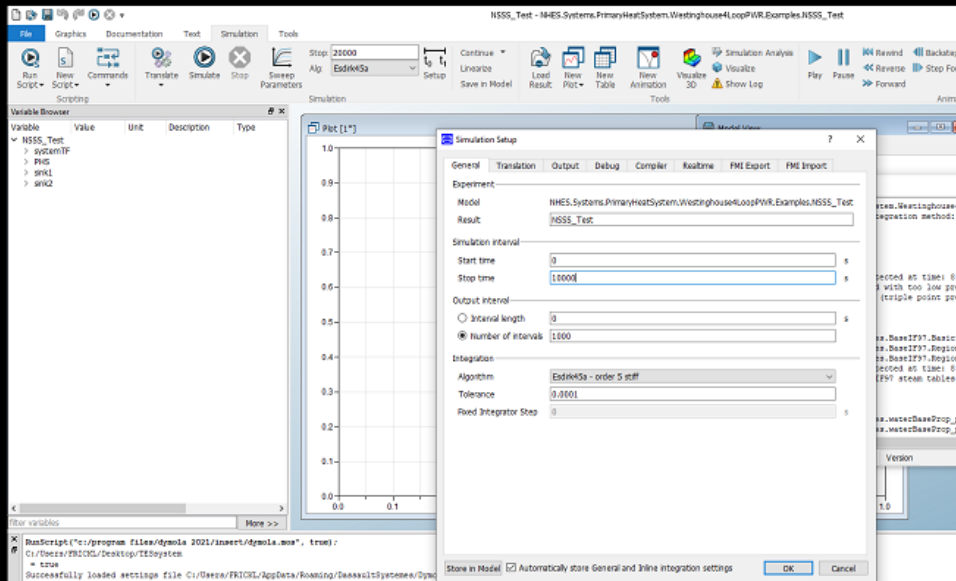
p	nuScale_Taveprogram.port_a_nominal.p	Pa	Absolute pressure
h	nuScale_Taveprogram.port_a_nominal.h	J/kg	Specific enthalpy
m_flow	-port_a1_nominal.m_flow	kg/s	Mass flow rate

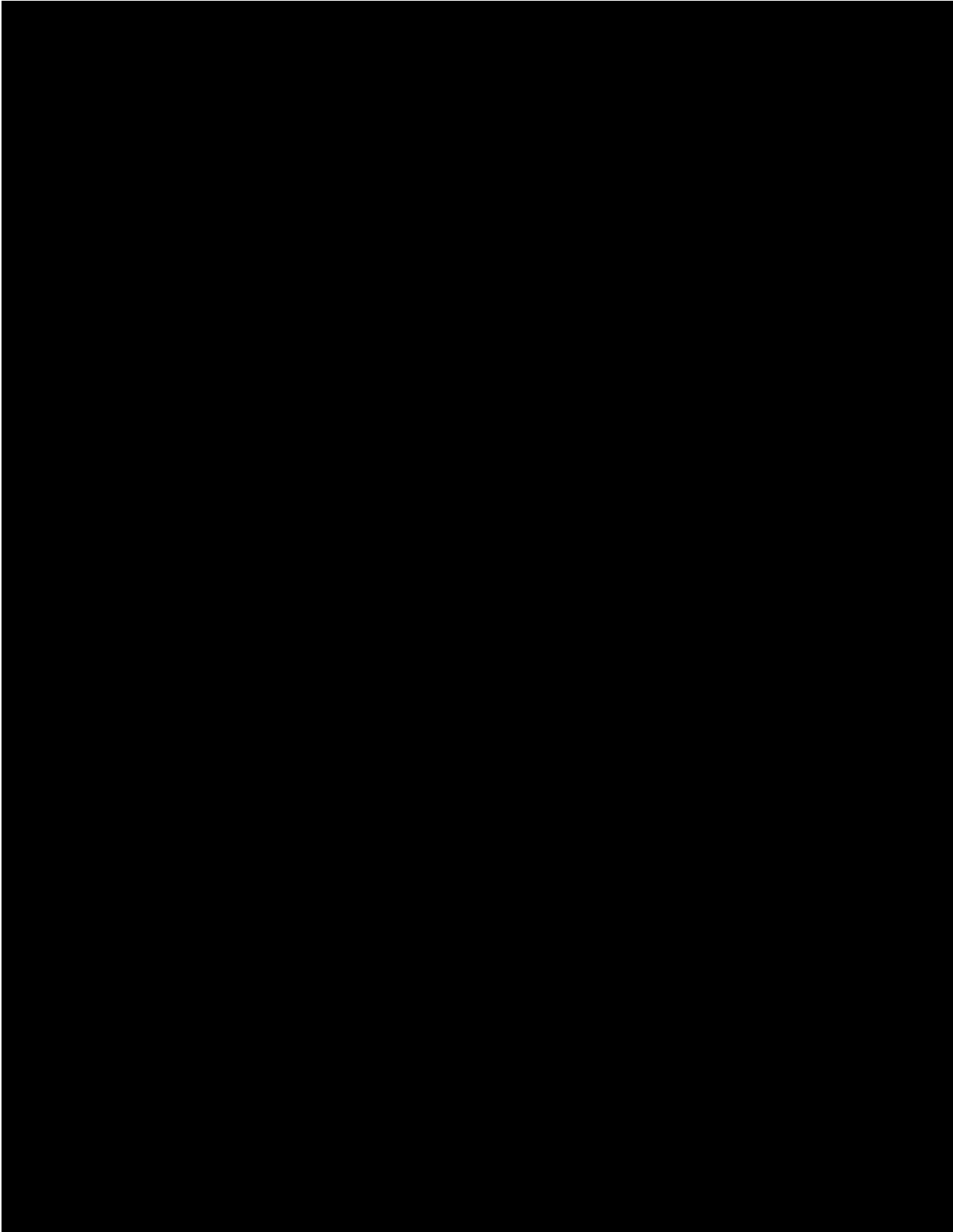
OK Cancel Info

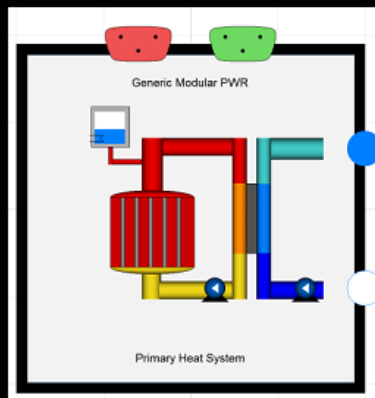
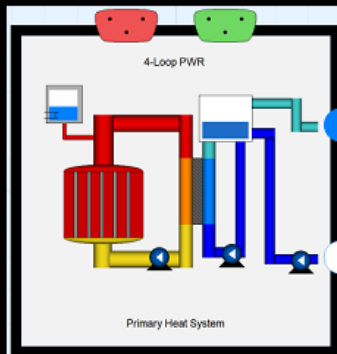


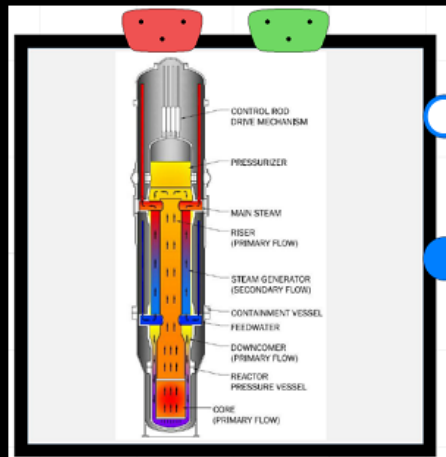


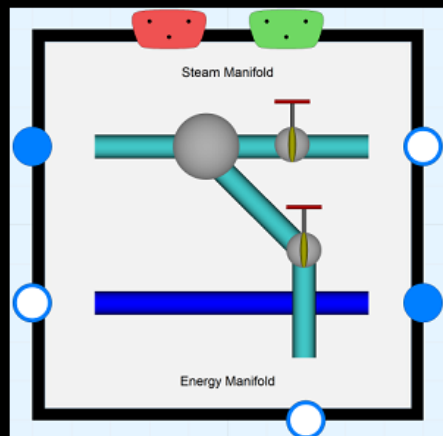


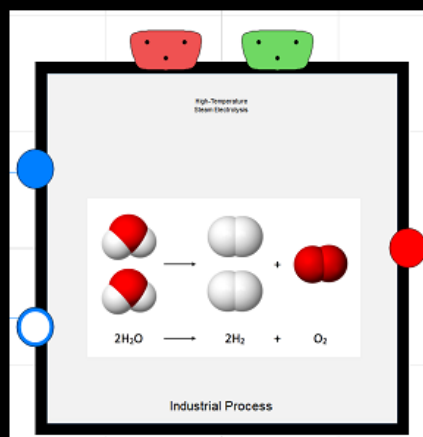


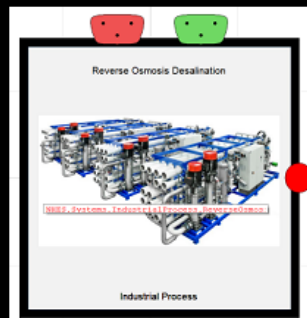


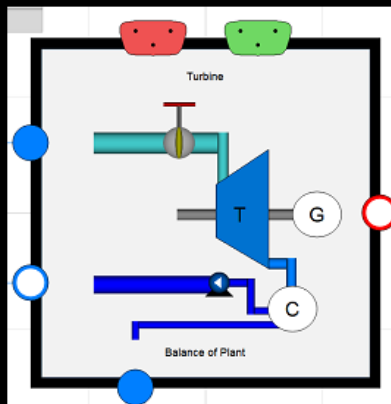


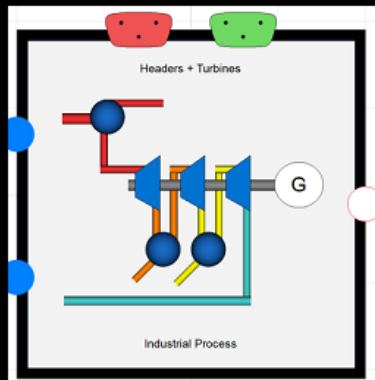


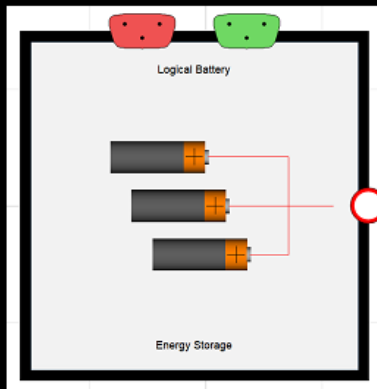


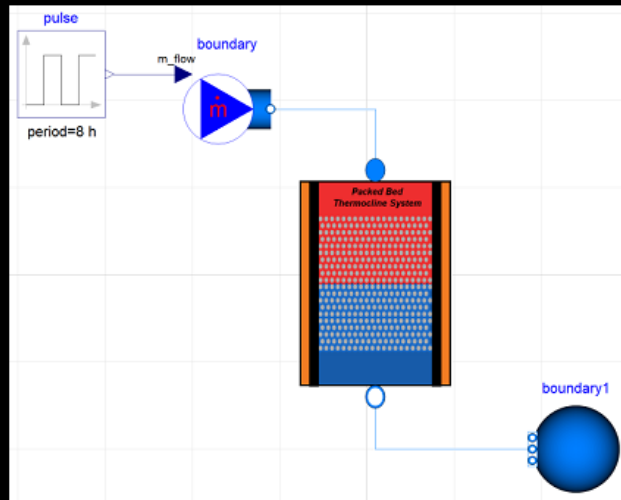


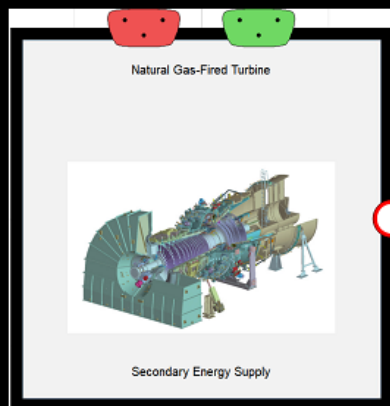


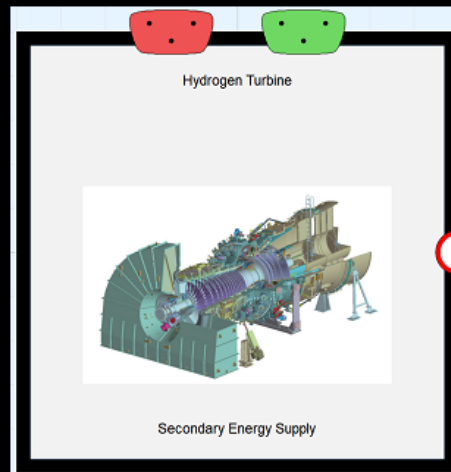


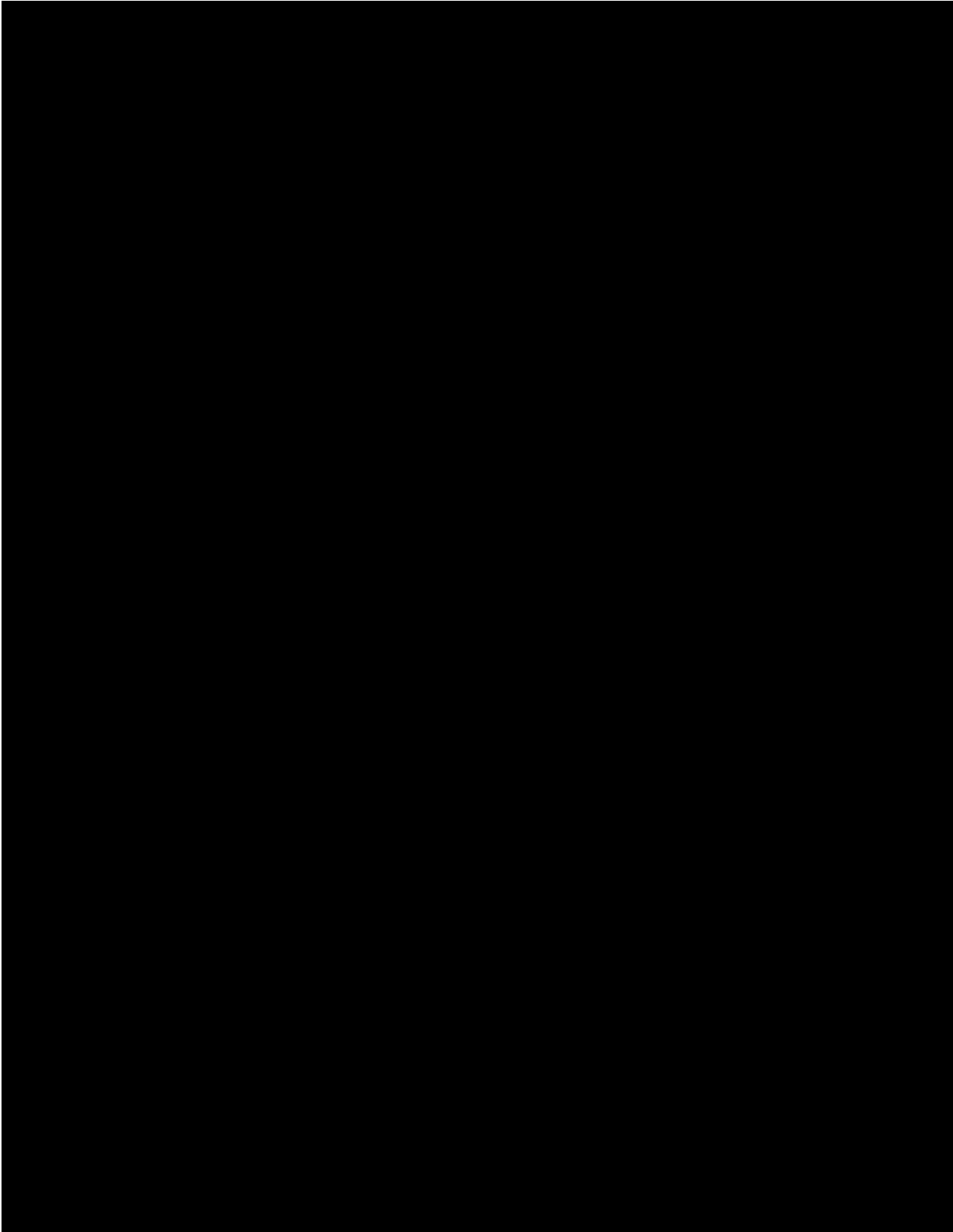


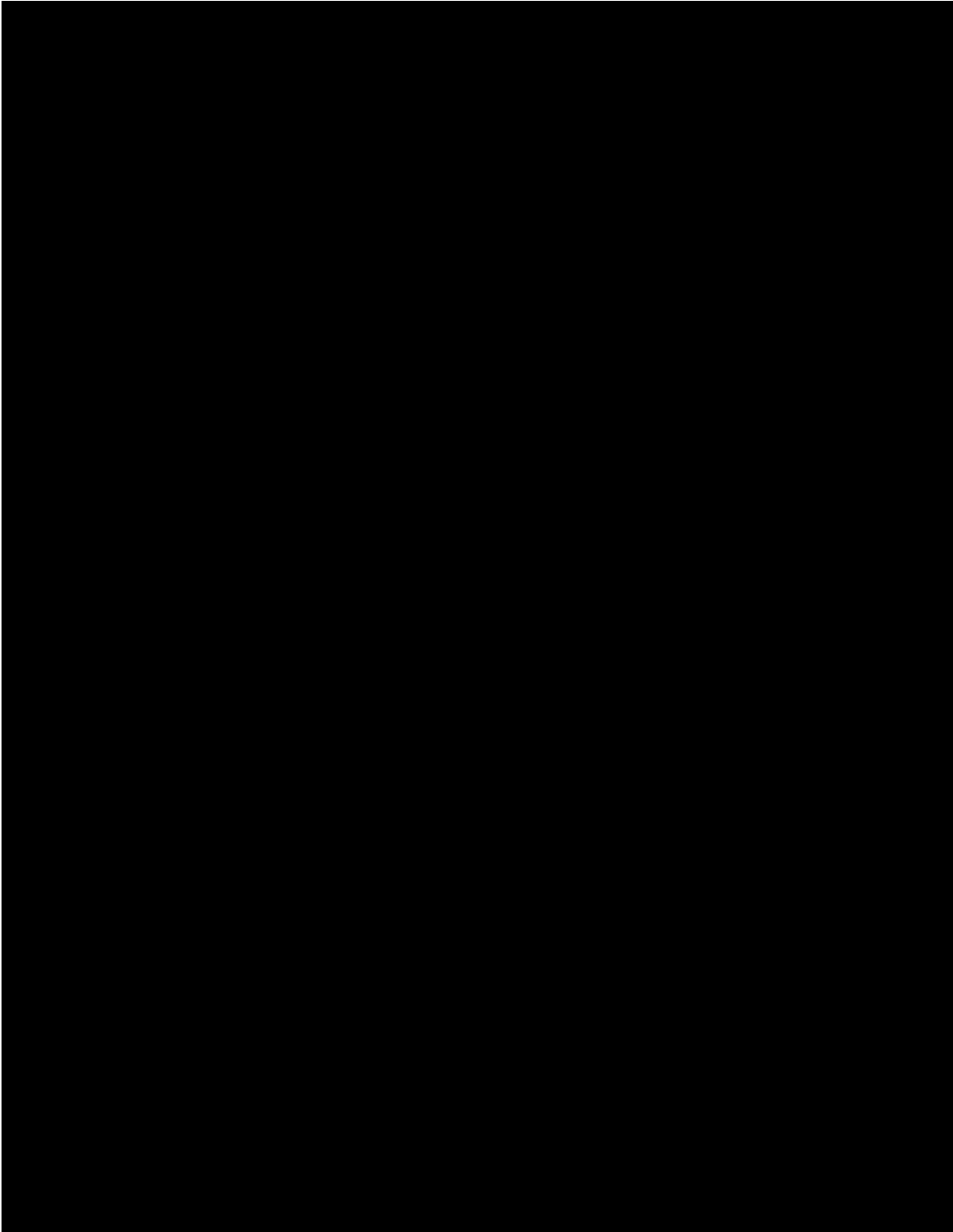


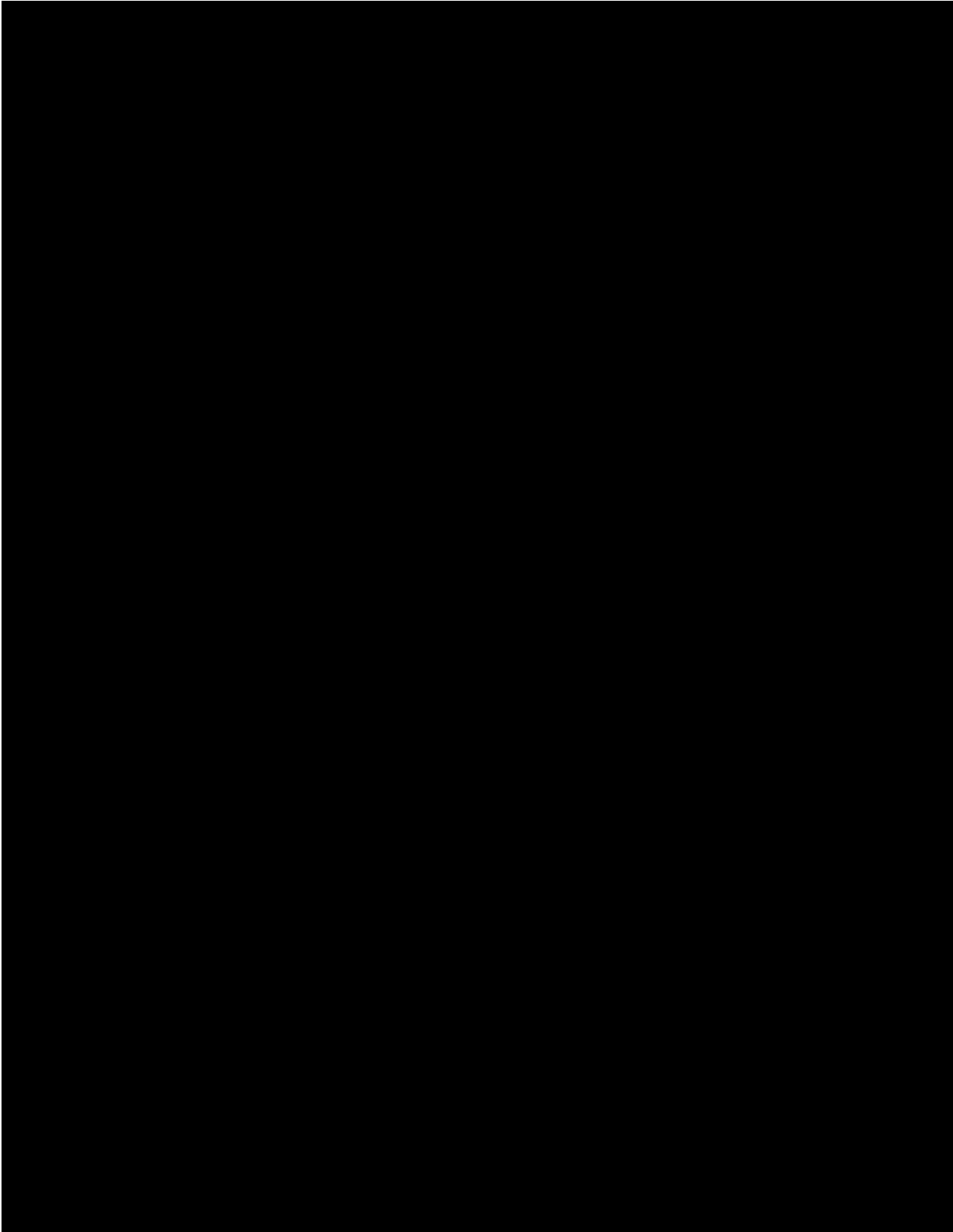


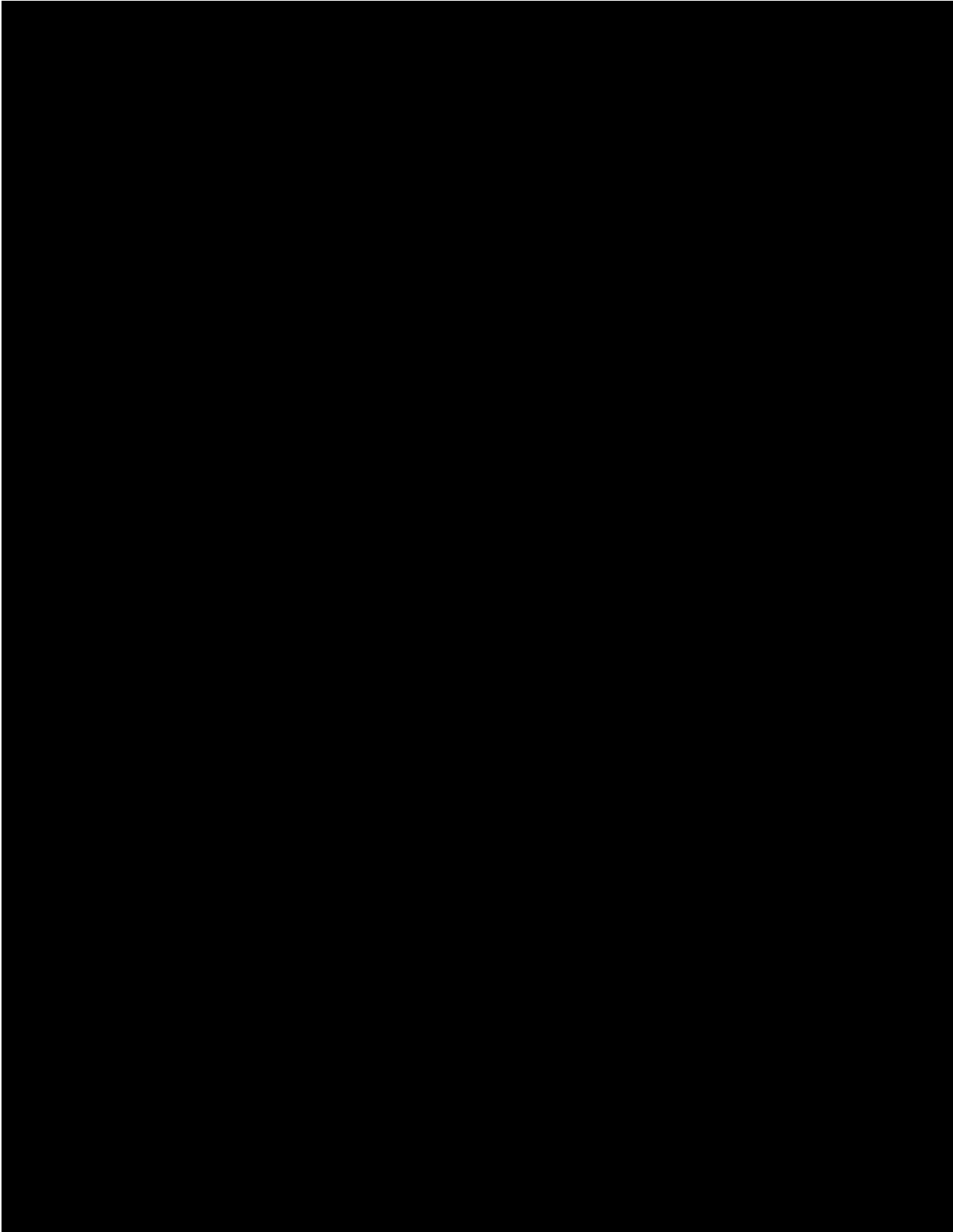














APPENDIX B – SQA: SOFTWARE QUALITY ASSURANCE PLAN (SQAP)

Compliance Quality Assurance Plan (CQAP)

ENVIRONMENTAL COMPLIANCE QUALITY ASSURANCE AND MAINTENANCE AND OPERATIONS PLAN

Andrew A. Brown



The INL is a U.S. Department of Energy laboratory
operated by Lockheed Martin Energy Research Corporation

Global Demand Forecasting

CONFIDENTIAL AND PROPRIETARY
 AND/OR TRADE SECRET INFORMATION

Information

Classification

Confidential

1/11/2024

1

10/01/2024

Page 1 of 10

Application

Plan

Global Demand

Abstract

CONFIDENTIAL

1.	INTRODUCTION.....	2
1.1	Global Demand Forecasting.....	2
1.2	Forecasting Challenges.....	3
1.3	Forecasting and Globalization.....	3
1.4	Forecasting Policy.....	3
2.	INTRODUCTION.....	3
3.	INTRODUCTION AND ANALYSIS OF THE MARKET.....	3
3.1	Forecasting.....	3
3.2	Forecasting.....	3
4.	INTRODUCTION.....	3
5.	INTRODUCTION AND ANALYSIS OF THE MARKET.....	3
6.	INTRODUCTION AND ANALYSIS OF THE MARKET.....	3
7.	INTRODUCTION AND ANALYSIS.....	3
7.1	Forecasting and Forecasting.....	3
7.2	Forecasting.....	3
8.	INTRODUCTION AND ANALYSIS OF THE MARKET.....	3
8.1	Forecasting.....	3
8.1.1	Forecasting.....	3
8.1.2	Forecasting.....	3
8.1.3	Forecasting.....	3
9.	INTRODUCTION AND ANALYSIS.....	3
10.	INTRODUCTION.....	3
10.1	Forecasting.....	3

Global Compact Dictionary

Content/Description/Abbreviation	Reference	Page
Abbreviation: ABBREVIATION	Abbreviation	Page 27 of 192
10.1.1. Test of V&V capabilities.....		16
10.1.2. Validation/Verification.....		17
10.1.3. Operational assessment of V&V capabilities for C&I systems.....		17
10.2. Test/Validation/Verification/Validation.....		17
10.3. Test/Validation.....		20
10.4. Test/Validation/Validation/Validation.....		20
10.5. Test/Validation.....		20
10.6. Test/Validation.....		20
10.7. Test/Validation/Validation/Validation/Validation/Validation/Validation.....		20
11. V&V Test/Validation.....		20
12. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
13. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
14. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
15. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
16. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
17. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
17.1. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
17.2. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
18. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
19. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
20. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20
21. Test/Validation/Validation/Validation/Validation/Validation/Validation/Validation.....		20

Document	Record Location	ID
Workman Safety Assessment Plan	Record Series 1 - General Inventory Series 101 - Physical (101-1000)	101-1000-001
Workman Test Procedures Version 1.0 - 2.0 - 3.0	101-1000	101-1000-002

Related Document Dependency

CONFIDENTIAL/NOFORN/AFSA/NOFORN
AFSA/NOFORN/AFSA/NOFORN

IdahoLab	01/11/2024
Revision	1
Effective Date	10/01/2020
Page	16 of 16

The HYBRID software implements a coding standard on all program code within the repository. This standard is publicly accessible on the HYBRID (CSCOs repository web) website: (<https://github.com/idaholab/HYBRID/-/wikis/HYBRID-Code-Standards>) and enforced through the continuous integration testing system.

3.1.3.1. Commenting Standards

The HYBRID software implements a commenting standard on all program code within the repository. The standard is used to fully document any code/section in the program code, providing the maximum presentation of software documentation via idocsys (see doc). This standard is publicly accessible on the HYBRID (CSCOs repository web) website: (<https://github.com/idaholab/HYBRID/-/wikis/Hybrid-Software-Commentary-Standard>) and enforced through the continuous integration testing system.

3.1.3.2. Testing Standards and Practices

The HYBRID software implements a testing standard and procedures on all the repository/codebase of the HYBRID software. This standard is publicly accessible on the HYBRID (CSCOs repository web) website: (<https://github.com/idaholab/HYBRID/-/wikis/HYBRID-Testing-Standards-and-Practices>) and enforced through the continuous integration system of the CCI.

2. CONFIDENTIAL/NOFORN/AFSA/NOFORN

The CONFIDENTIAL/NOFORN/AFSA/NOFORN policy fully aligns with the expectations set in the *CFR 25.12, "AFSA and AFSA Policy for Employees, Family Members, and Contractors and Operations Plans"* (explaining the need for AFSA with HYBRID).

10. PURPOSE

The goal of software validation (see doc) is to ensure that the requirements for a specific software code have been fulfilled. Software verification (see doc) ensures that the system or component is verified that specified conditions have been satisfied and provided for the purpose of maintenance.

10.1. V&V Document

10.1.1. Test & V&V Guidelines

Test procedures or plans will specify the following, not applicable:

Related Document's identifier:

CONFIDENTIAL//NOFORN//REL//NOFORN
 A2530000A18034.8.14/NOFORN

Identifier:

0101 00000

Structure:

1

Revision Date:

10/01/2020

Page: 13 of 16

The purpose of developing a formal leader consent for leading and coordinating and facilitating (V&V) activities, including the development of lead strategies and the documentation of performance leader's necessary. The lead strategy analysis is performed during the development activities conducted by the (C2000) core team (see also), and it is documented at the steps in the process of developing new leader model to be created. V&V activities are conducted using the (C2000) core team.

During development development or capability is performed by a structure: develop, the following shall be documented:

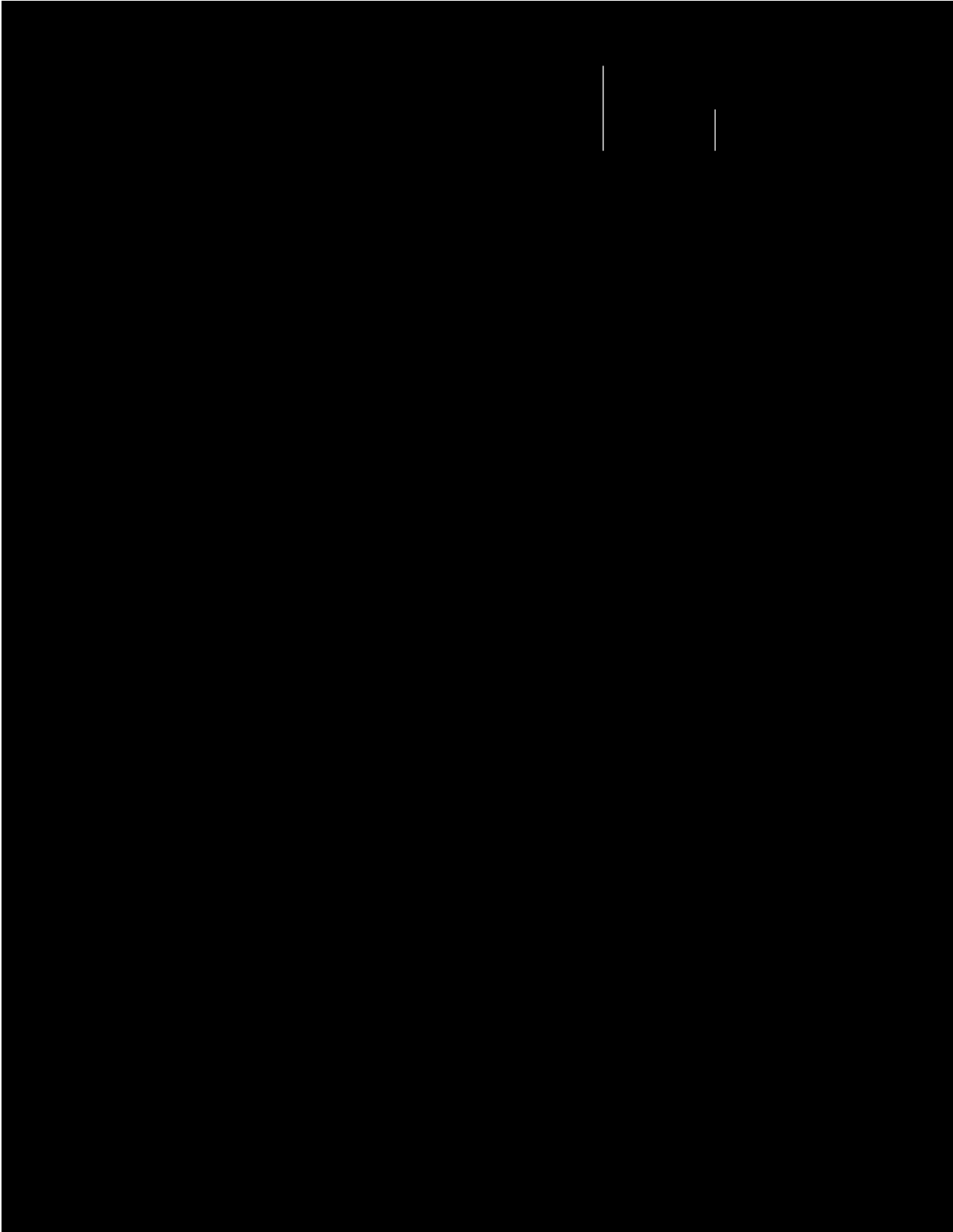
- 1) Acquired lead activities and method of documentation (e.g., lead pillars, performance, structure, etc.);
- 2) Acquired support capabilities (see also) (e.g., additional lead models, lead innovation leader, etc.);
- 3) Type and extent of acquired leading; and
- 4) Acquired structure and approach.

As a component (or more) of the change control board (CCB) (see also), will being part of the development, shall ensure the correct documentation of the leader and ensure that the documentation includes acquired capabilities (which necessary) that have valid compliance criteria. This documentation may include:

- 1) Documentation of the leader including compliance criteria. The documentation procedure is defined in the (C2000) web page: <https://github.com/idaholab/HYBRID/-wikis/Developing-Regression-Tests>
- 2) Diagram: Regression model (performance or capability or compliance) documented;
- 3) Acquisition model (capability) table;
- 4) Diagram: Design (capabilities for guidance on leading, coordination and the capability, structure, (e.g., structure, structure, and structure) documented) to be used during leading;
- 5) Other documentation (see also)

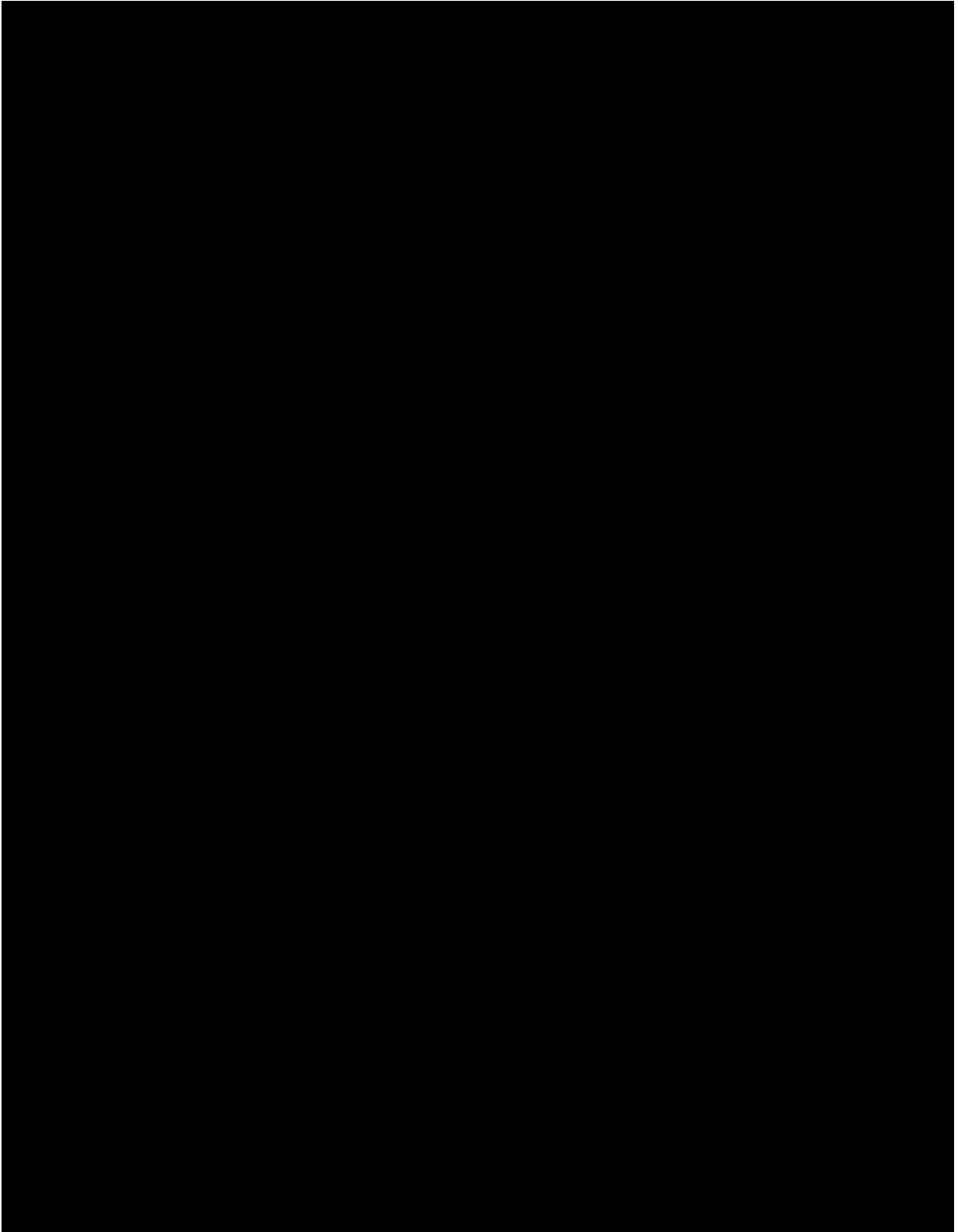
The CCB will verify that the provided documentation contains that the structure development follows in the documented requirements and that the structure parameters meet results.

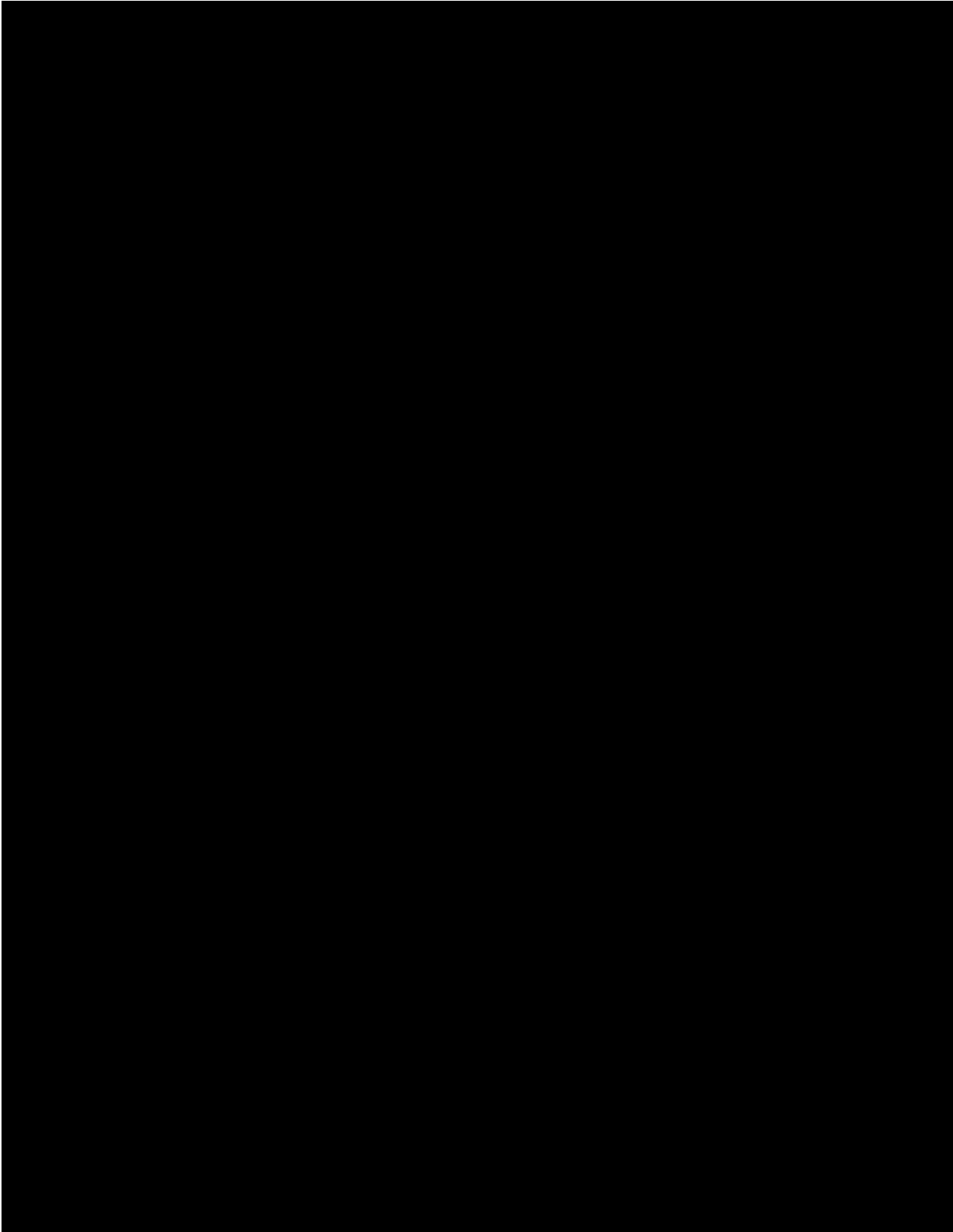
Test Case Reviewer(s):	Chair of the C2.20 Technical Group and backpack on Wednesday (Chairman of the C2.20)
Test Result Reviewer and Approver:	Chair of the C2.20 Technical Group and backpack on Wednesday (Chairman of the C2.20)
Acceptance Test Case Reviewer(s):	Chair of the C2.20 Technical Group and backpack on Wednesday (Chairman of the C2.20)
Acceptance Result Reviewer(s):	Chairman of C2.20
Acceptance Result Approver:	Chairman of C2.20

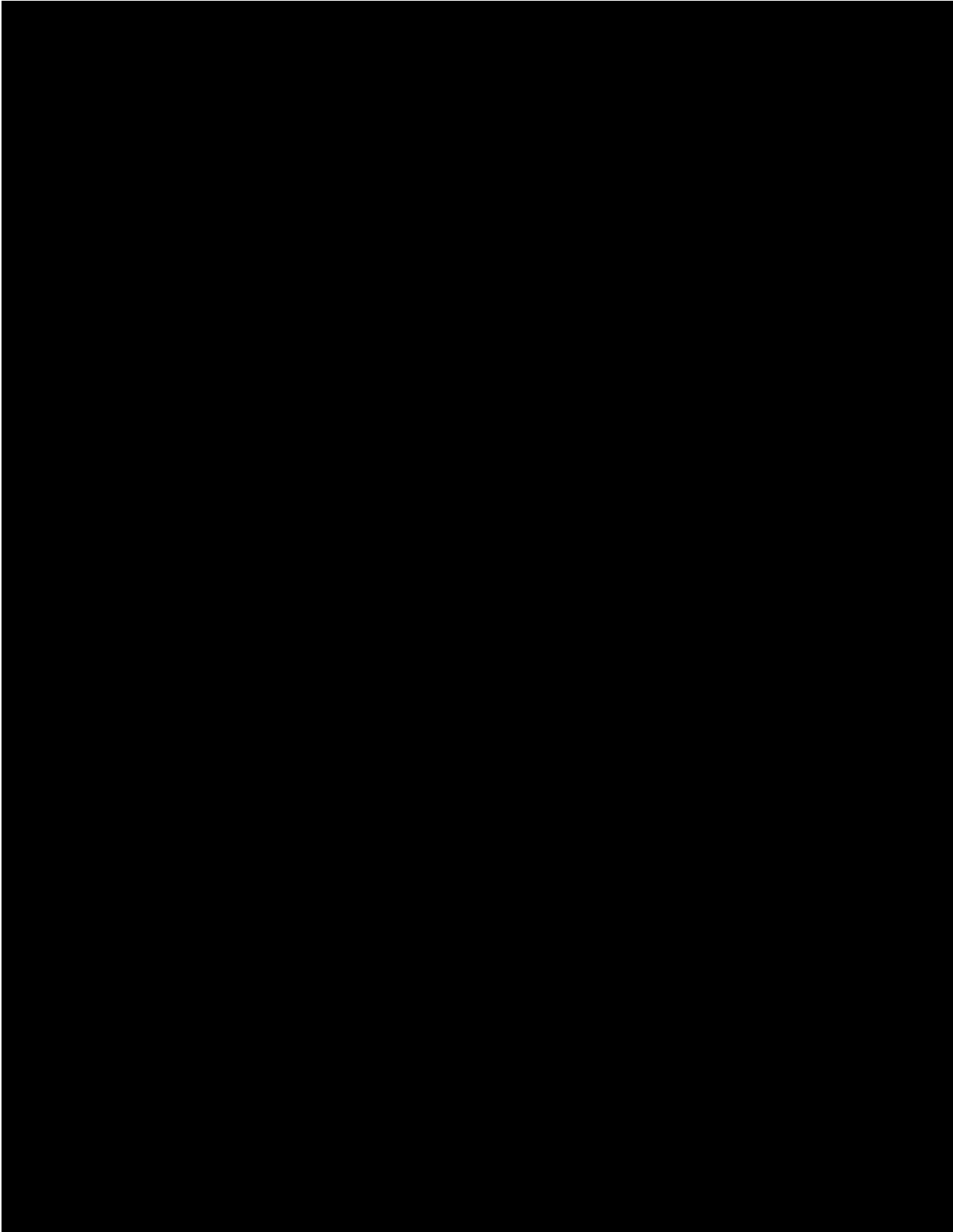


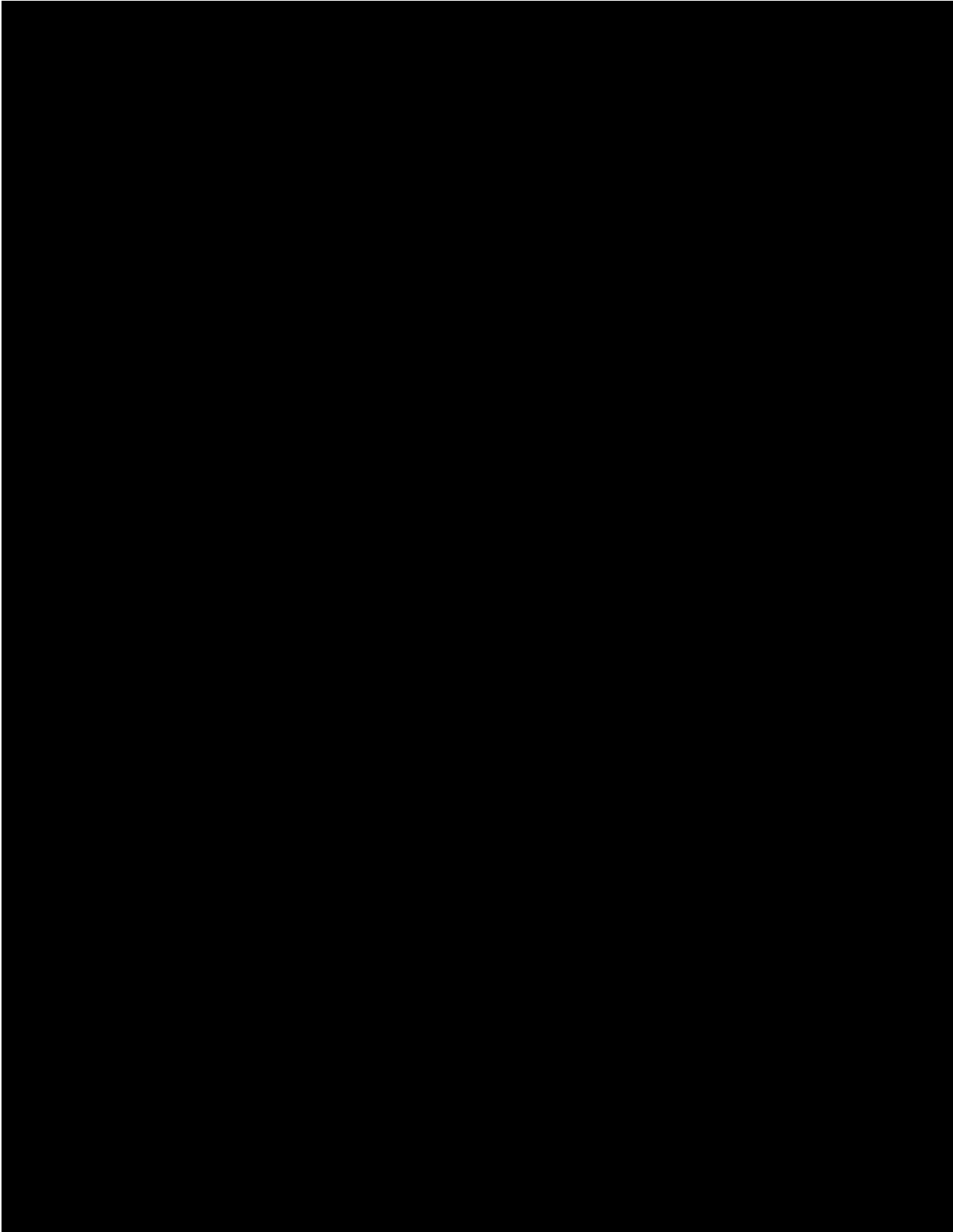
APPENDIX C – SQA: SOFTWARE DESIGN DESCRIPTION (SDD)

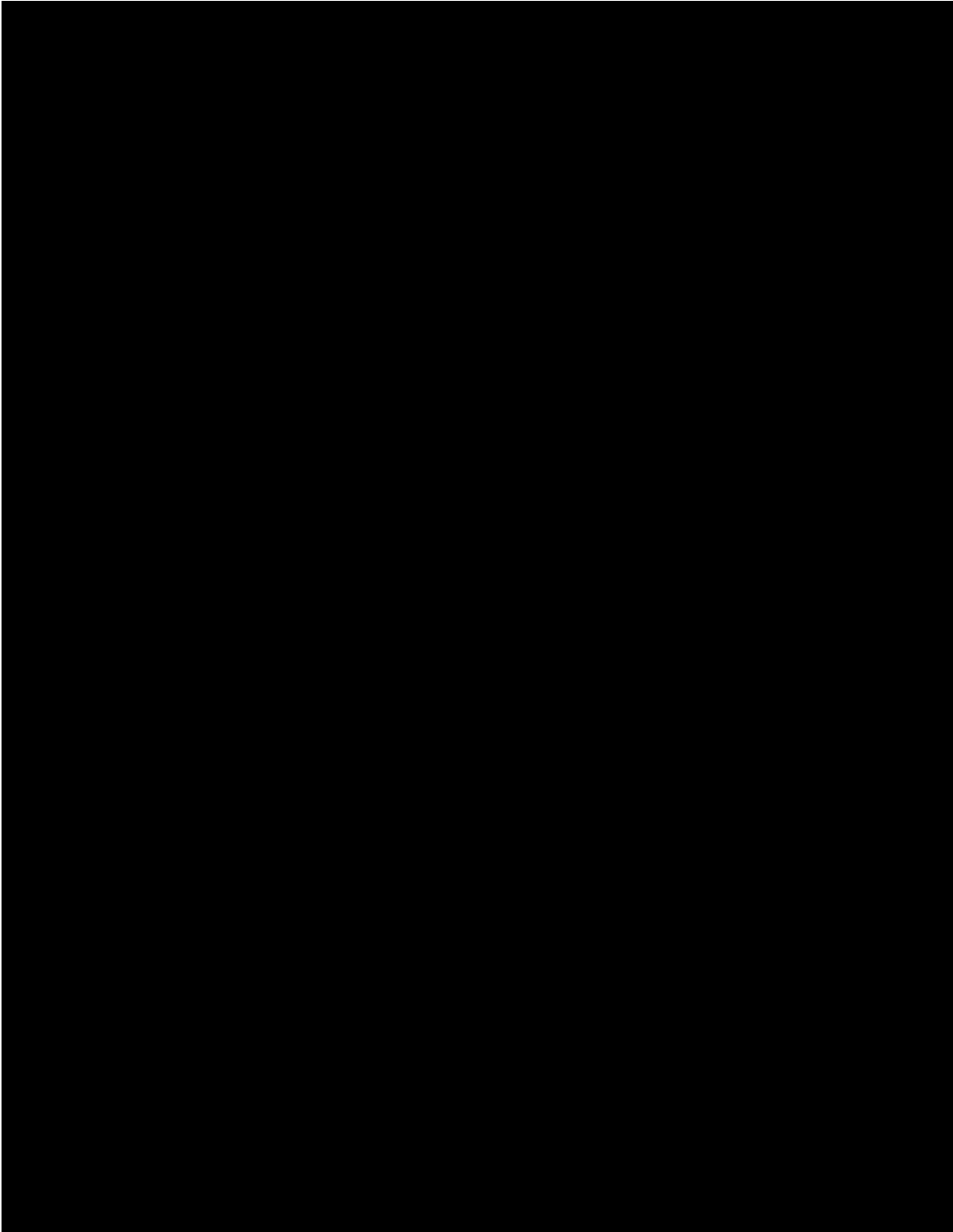


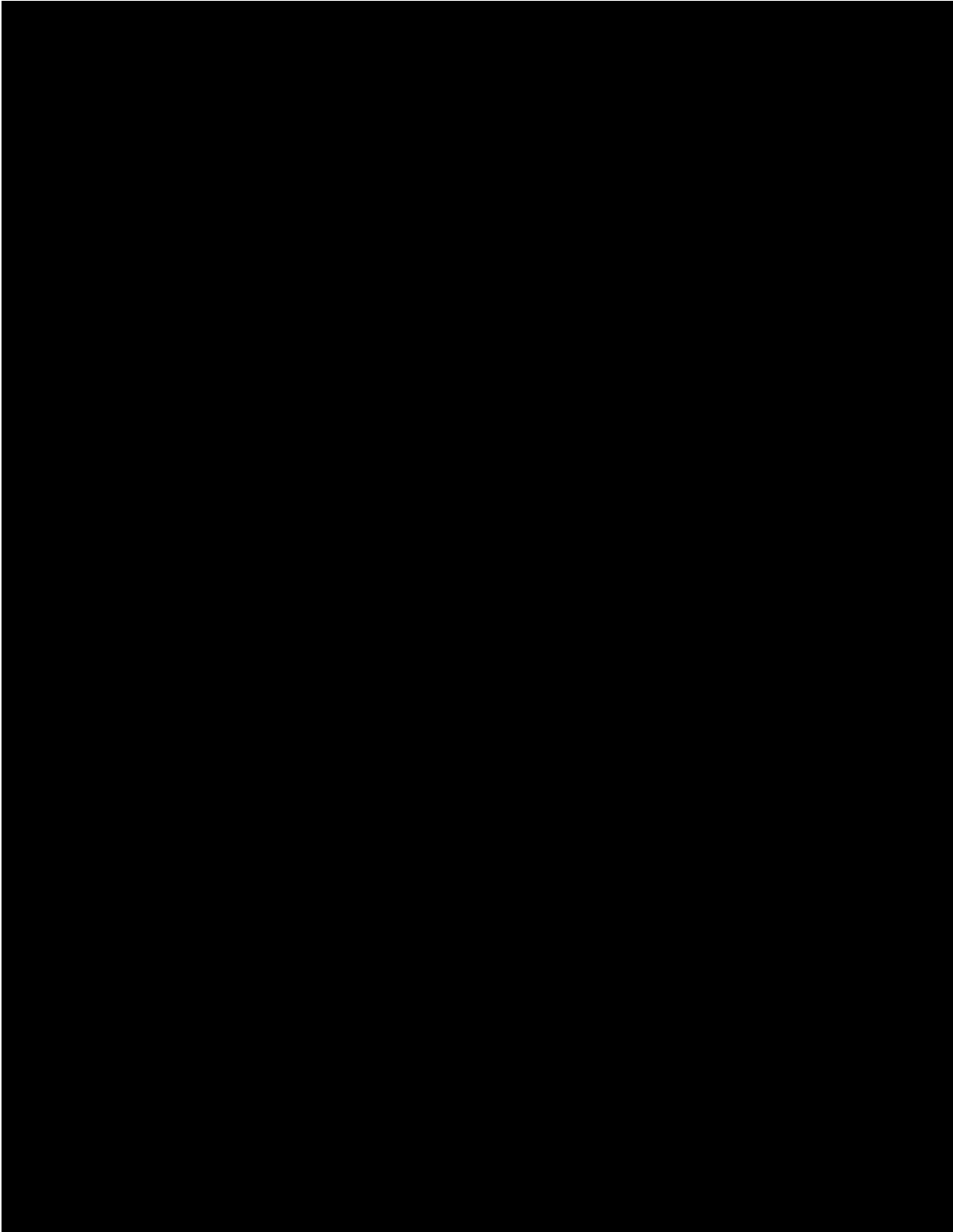


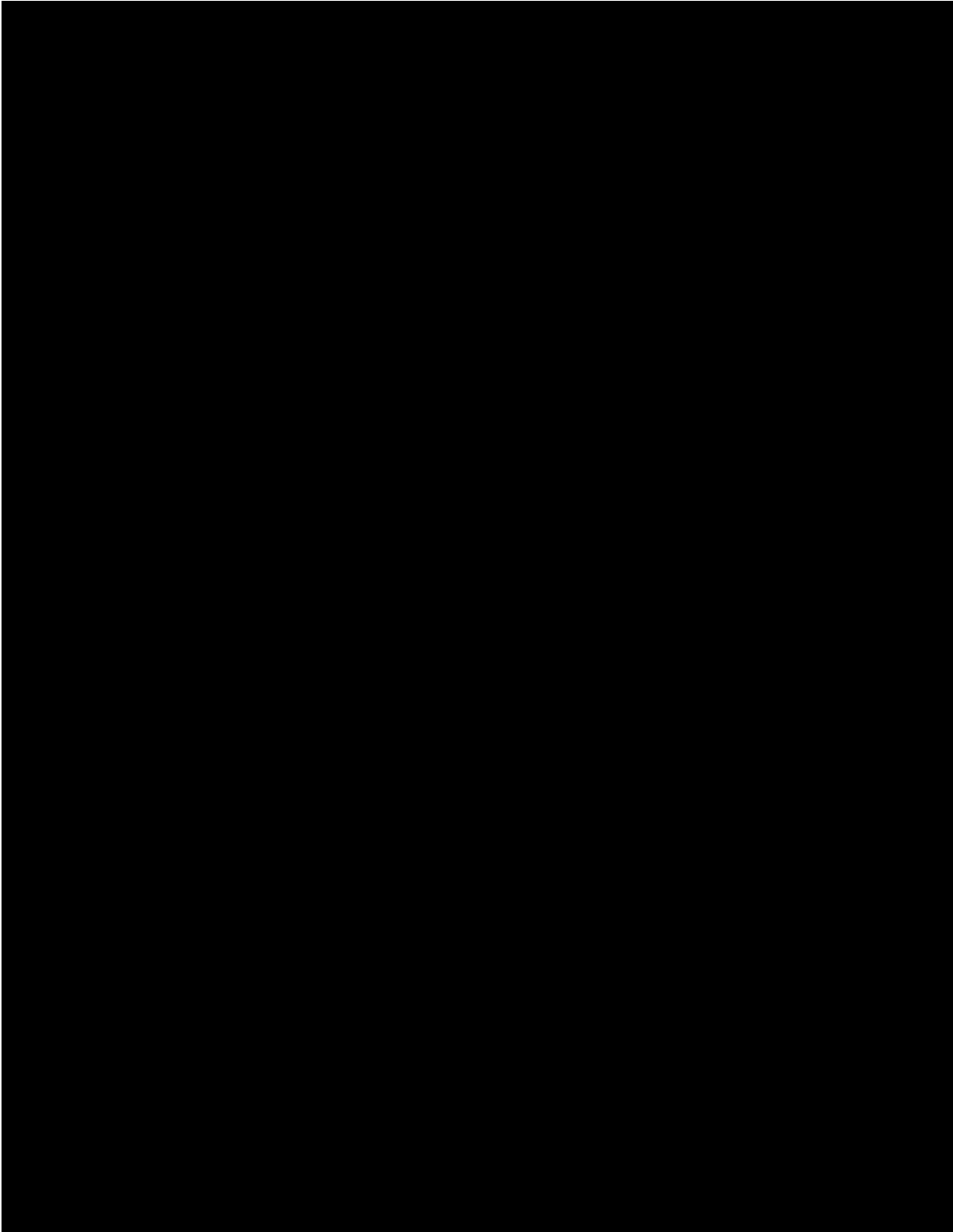


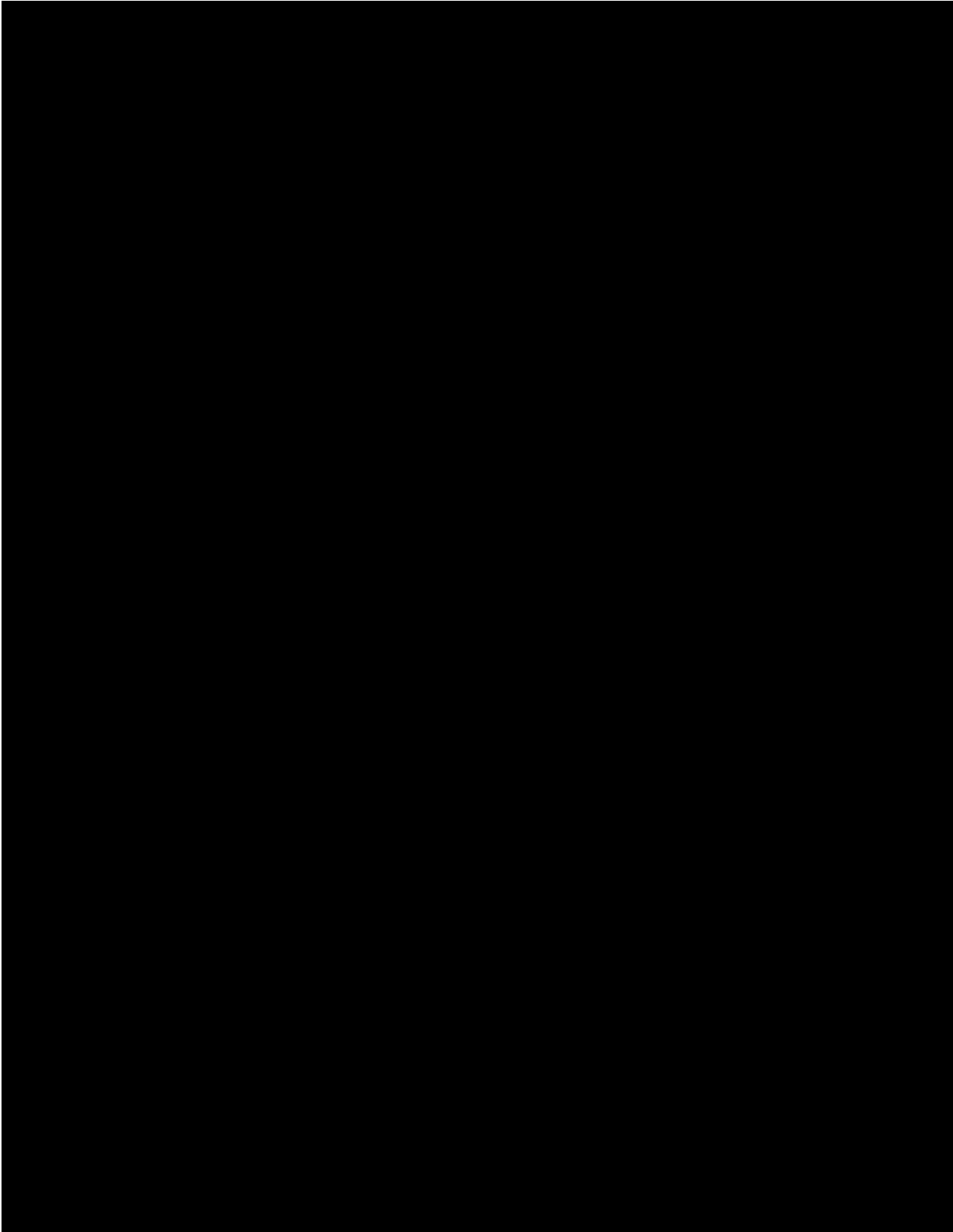


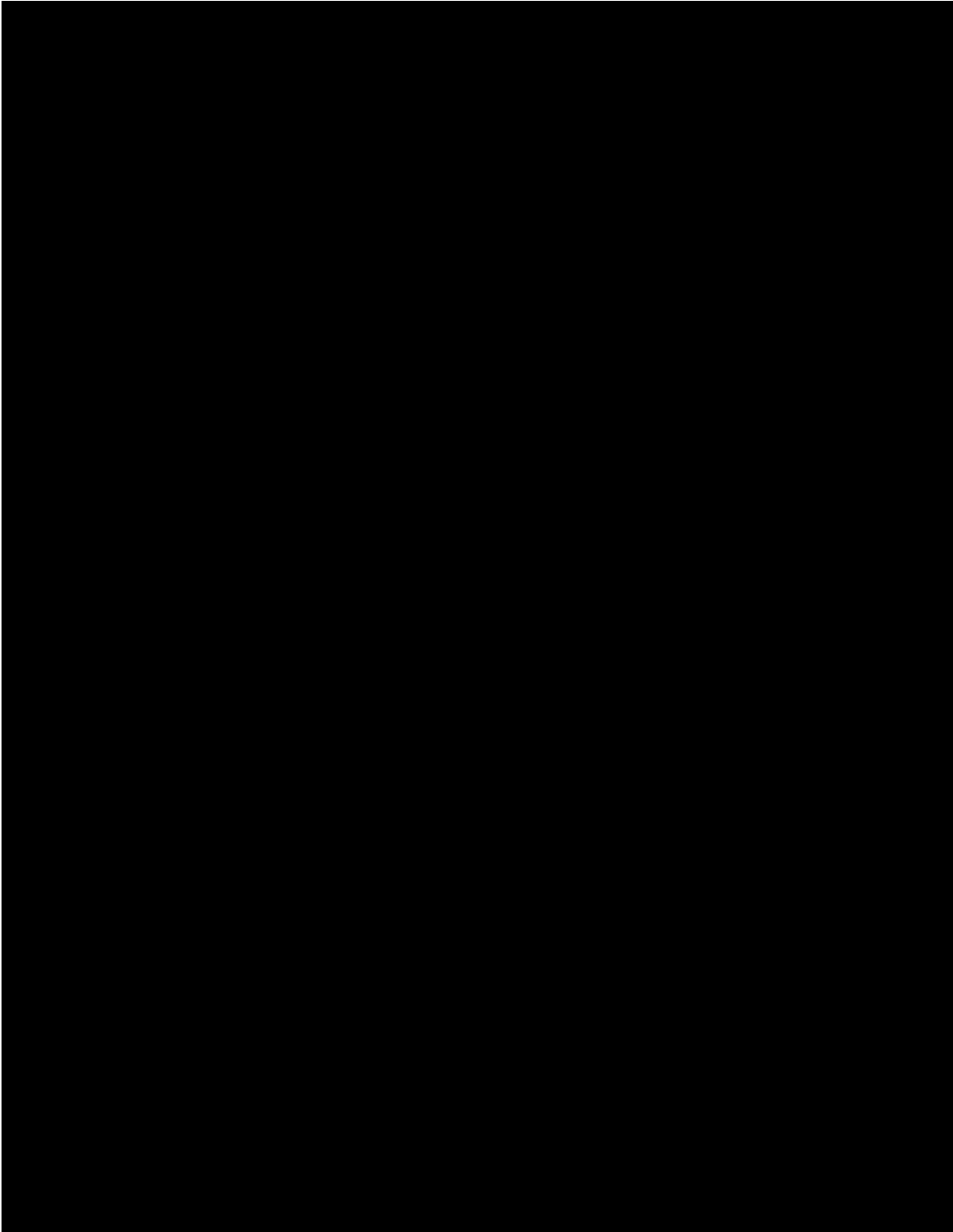


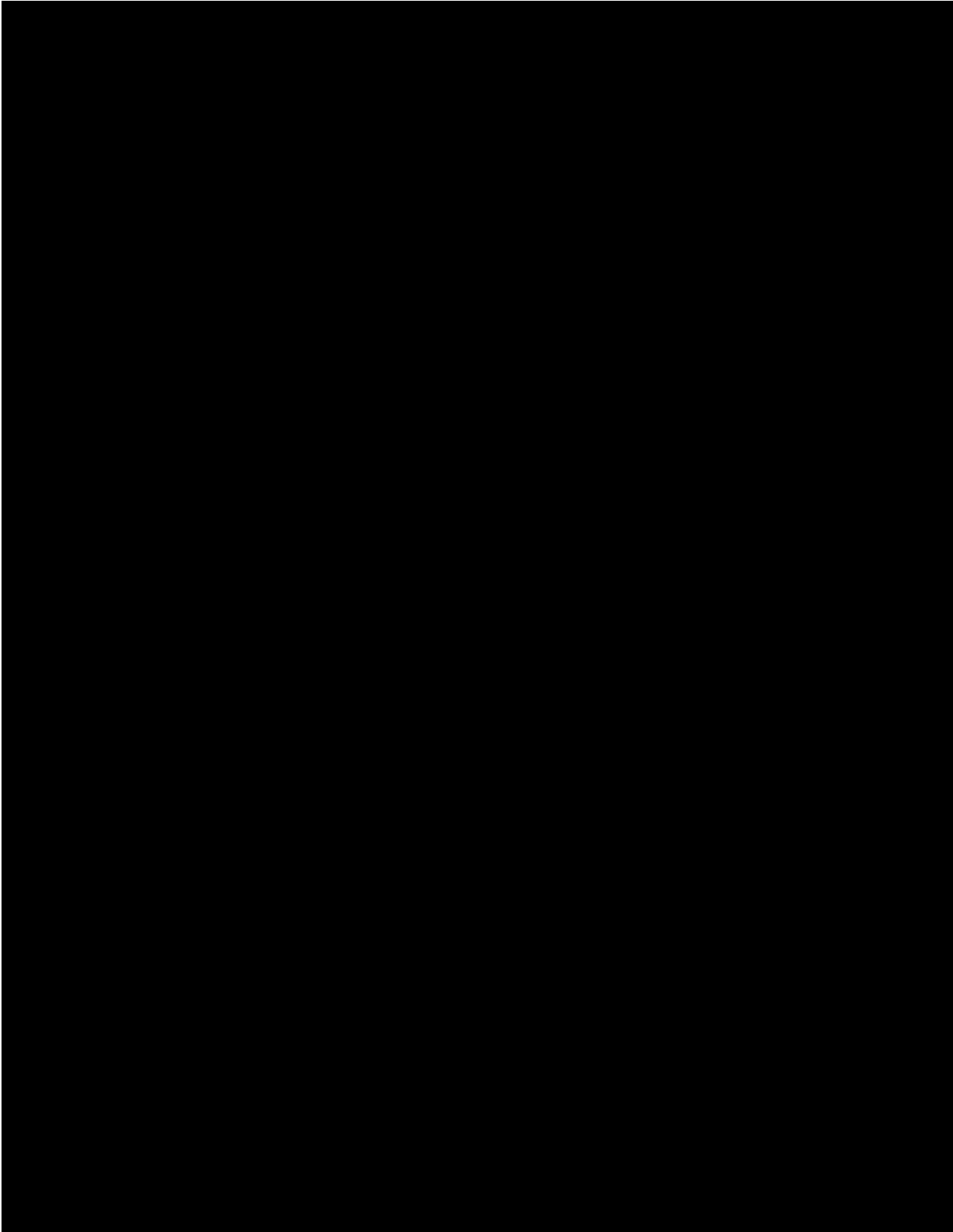


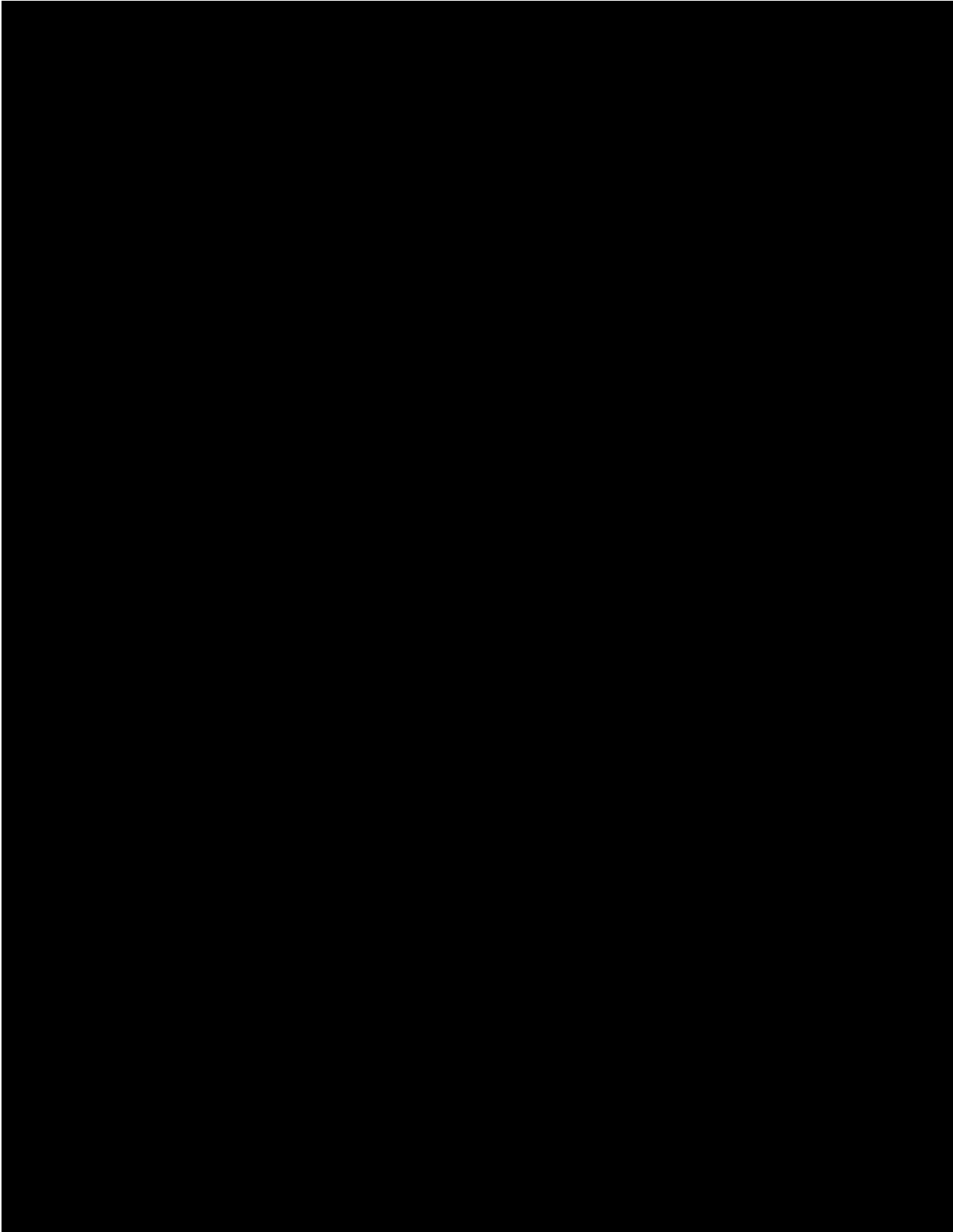


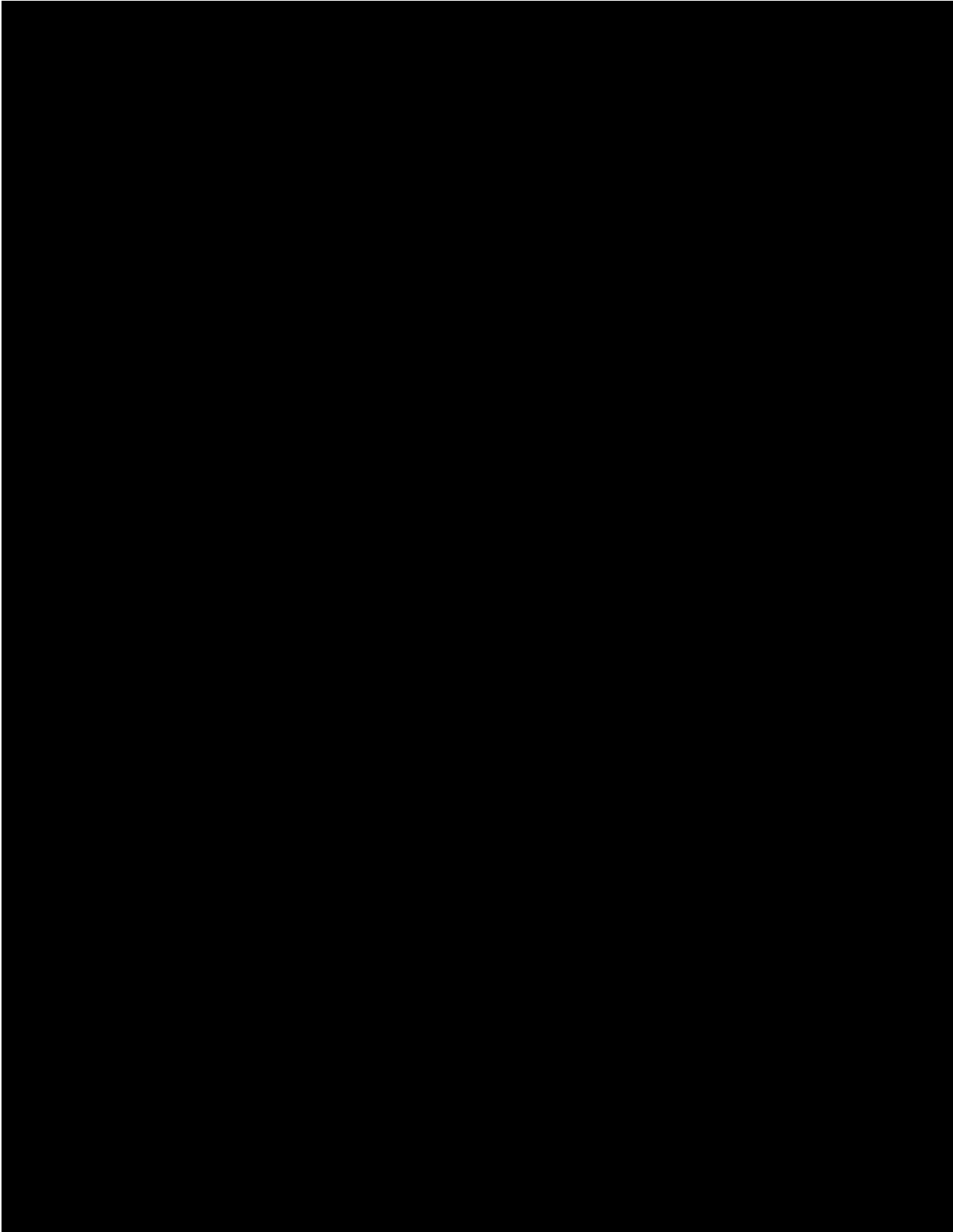


















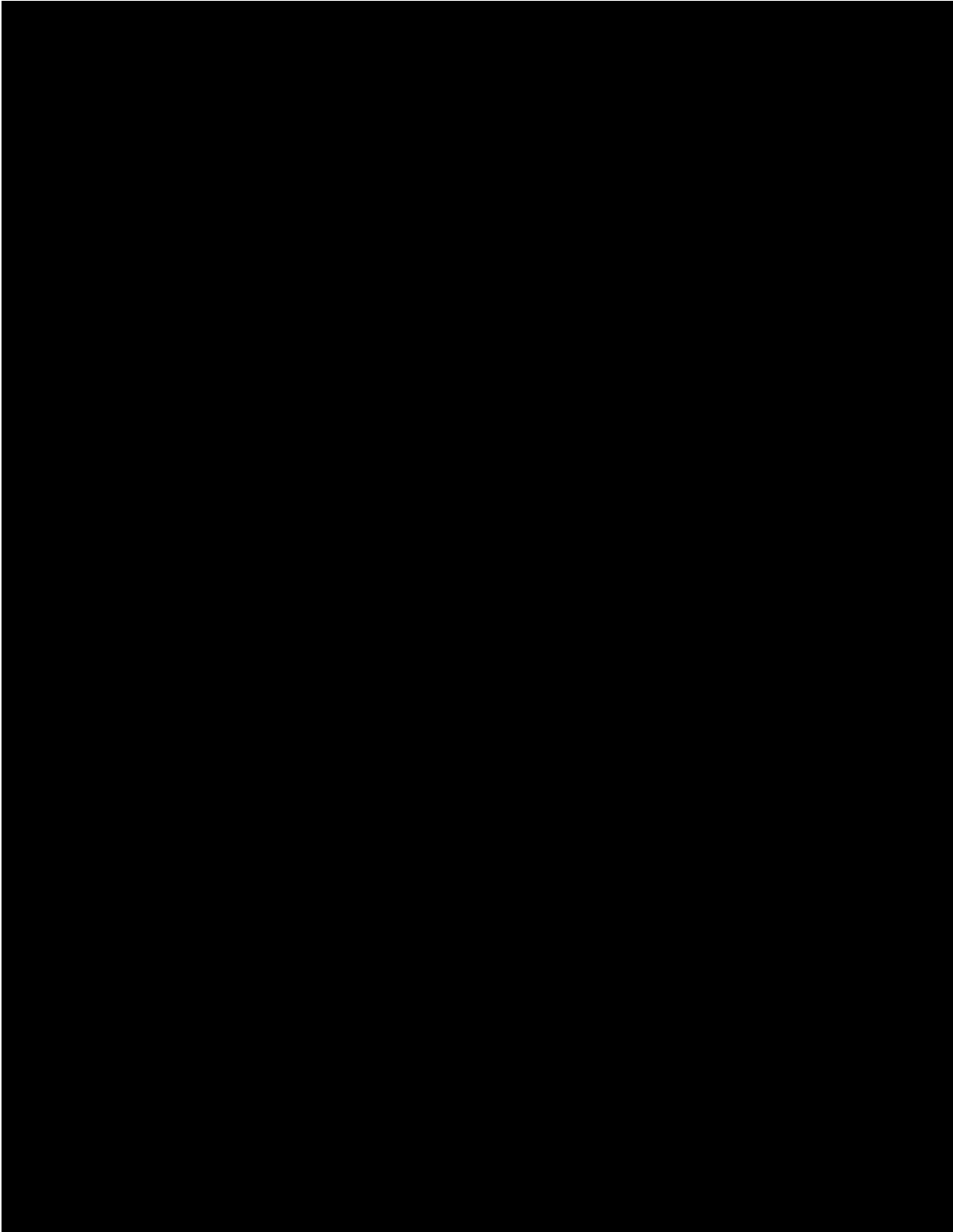


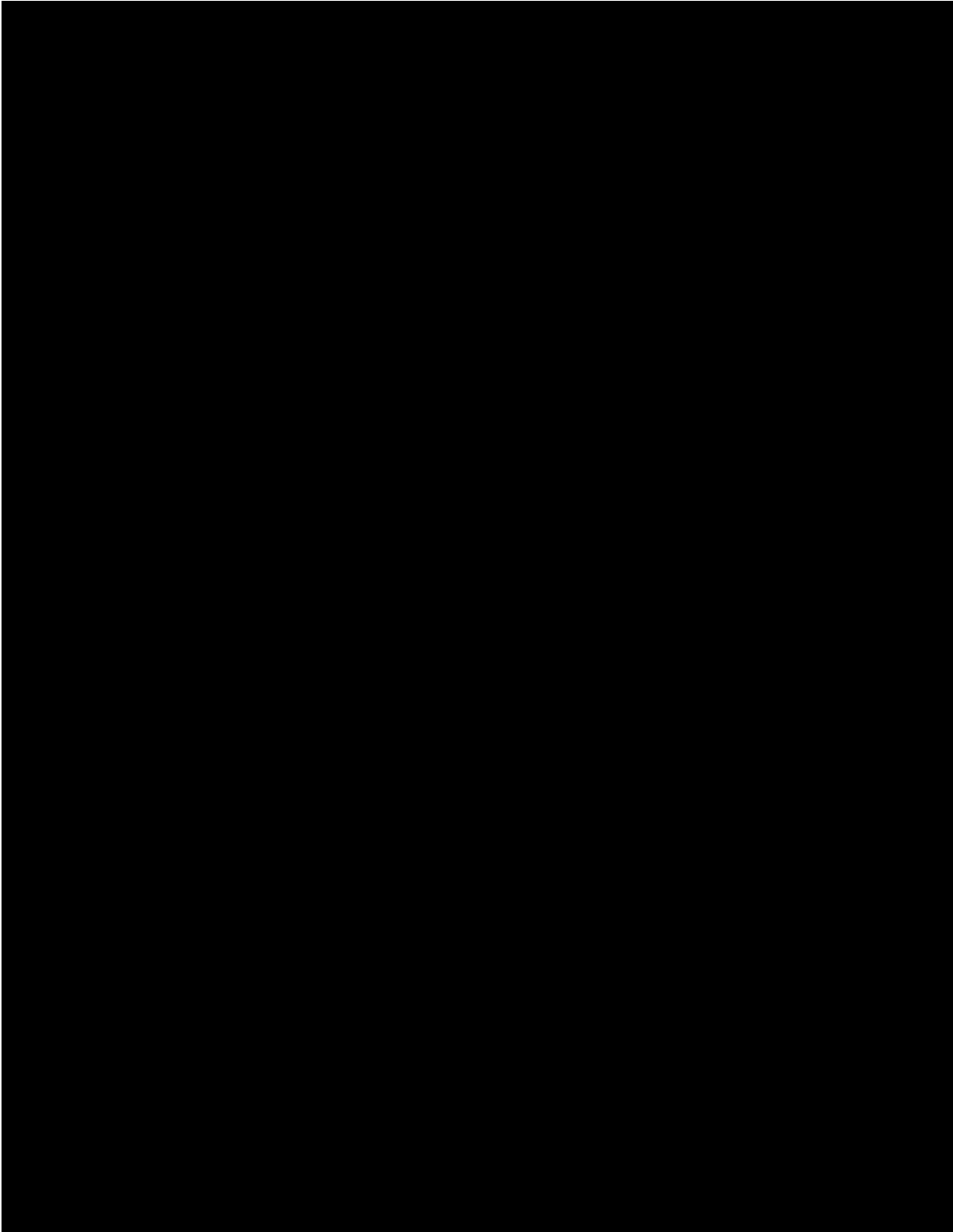


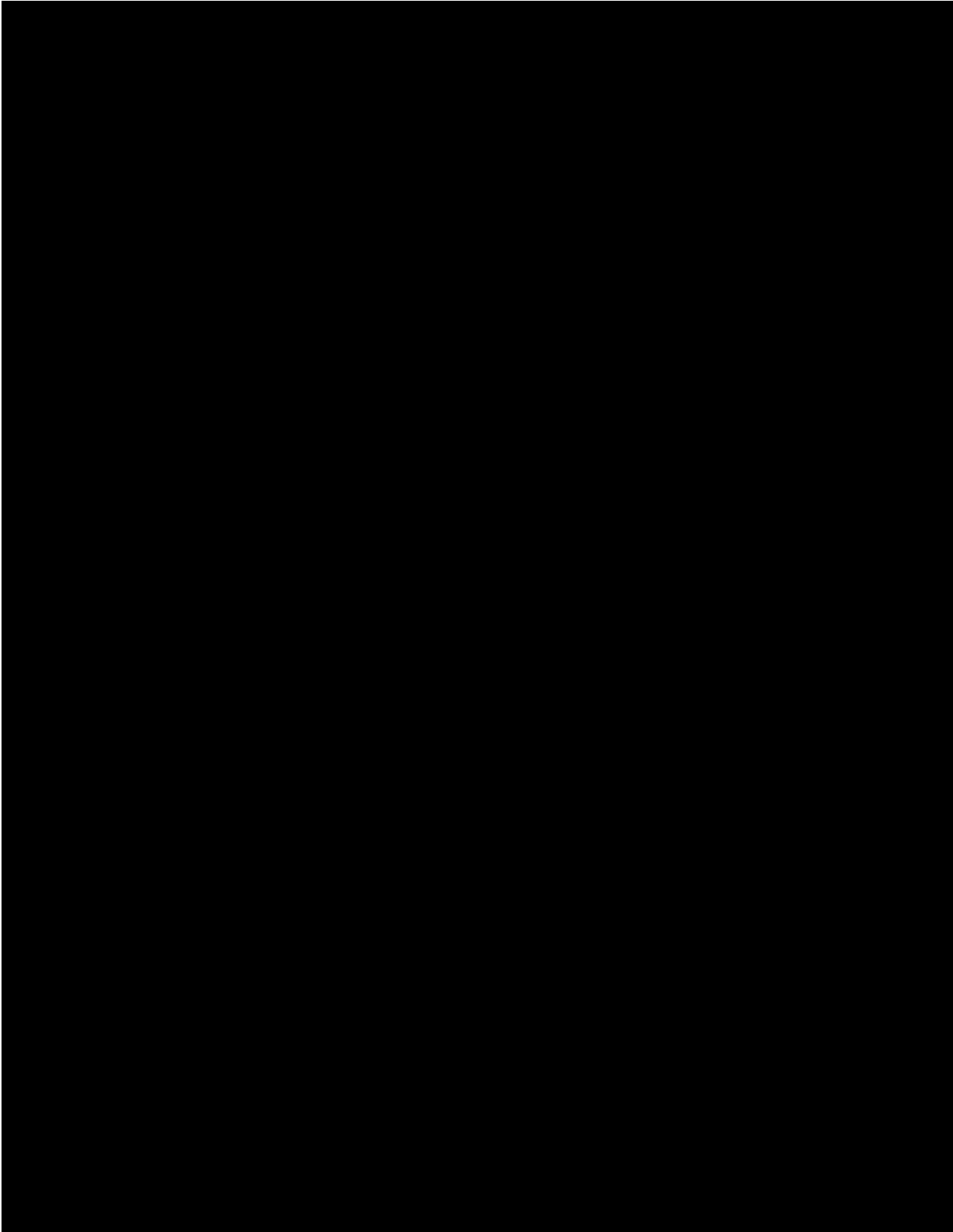


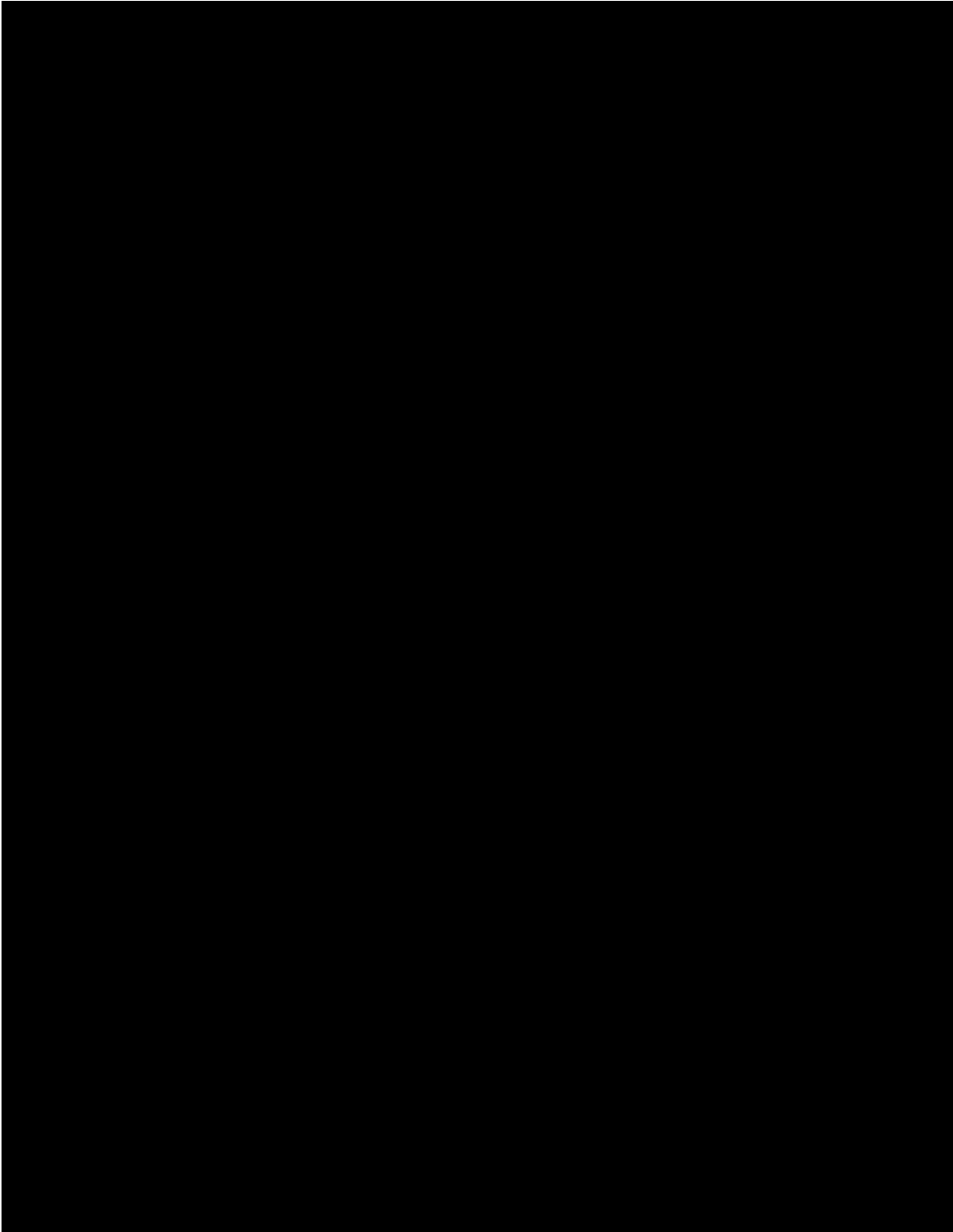


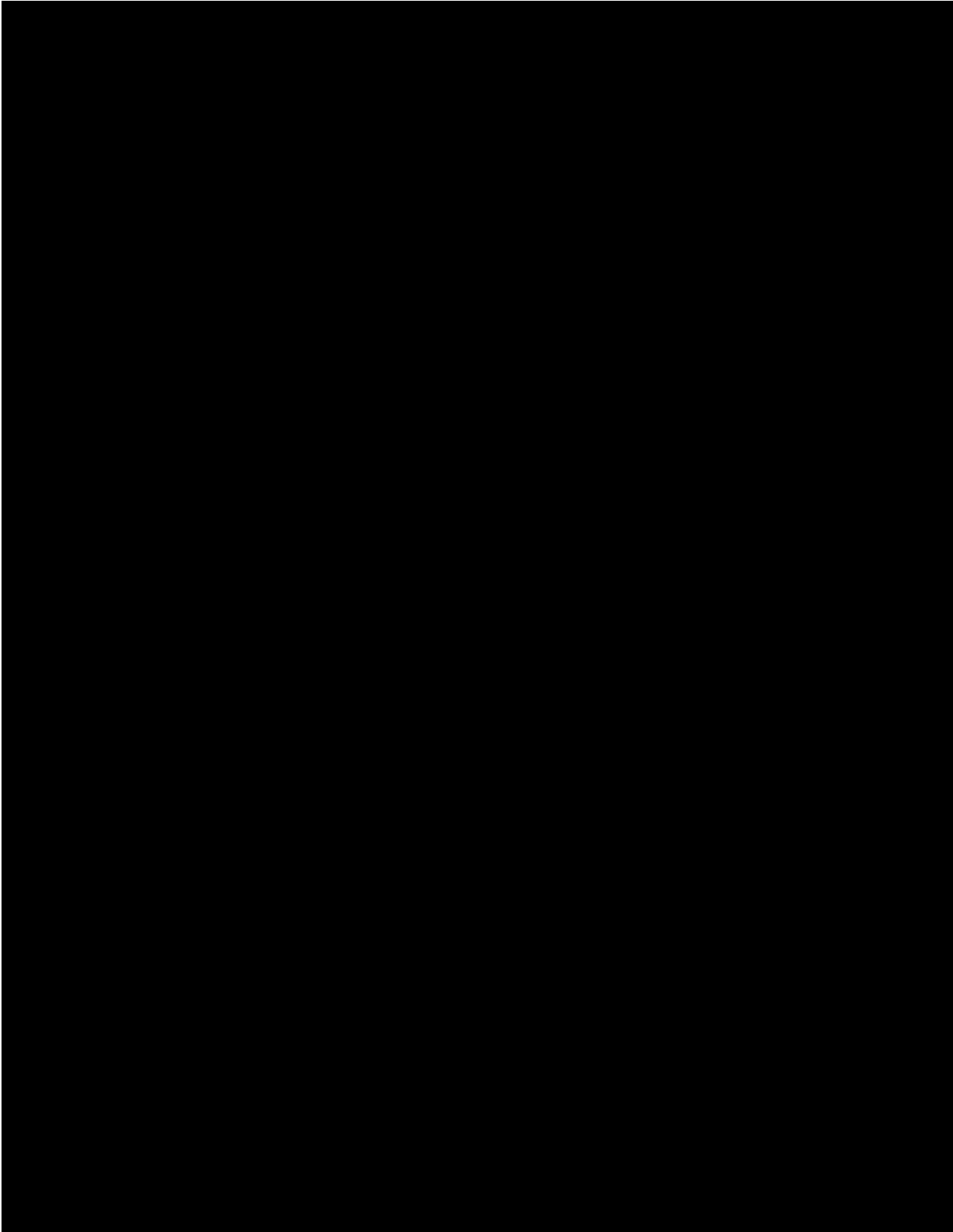
Name	Last commit	Last update
 .gitlab	issue and MR templates added	2 years ago
 Models	Addition of Initialization files and cleaning of example files in t...	6 days ago
 archive	added description	2 months ago
 developer_tools	added description	2 months ago
 doc/user_manual	Latex File Tree Addition	1 week ago
 scripts	fixed list	2 weeks ago
 tests	Update of Readme for what the different tests are	1 week ago
 TRANSFORM-Library @ 96d3493a	added transform submodule	2 months ago
 raven @ 60137c37	updated raven	2 weeks ago
 .gitignore	Changing the .gitignore file to not ignore .mat files.	1 week ago
 .gitmodules	added transform submodule	2 months ago
 00README.txt	Civet test dirrectory set-up	4 years ago
 README.md	Update README.md	5 months ago
 run_tests	Use proper pathsep for the os.	2 months ago

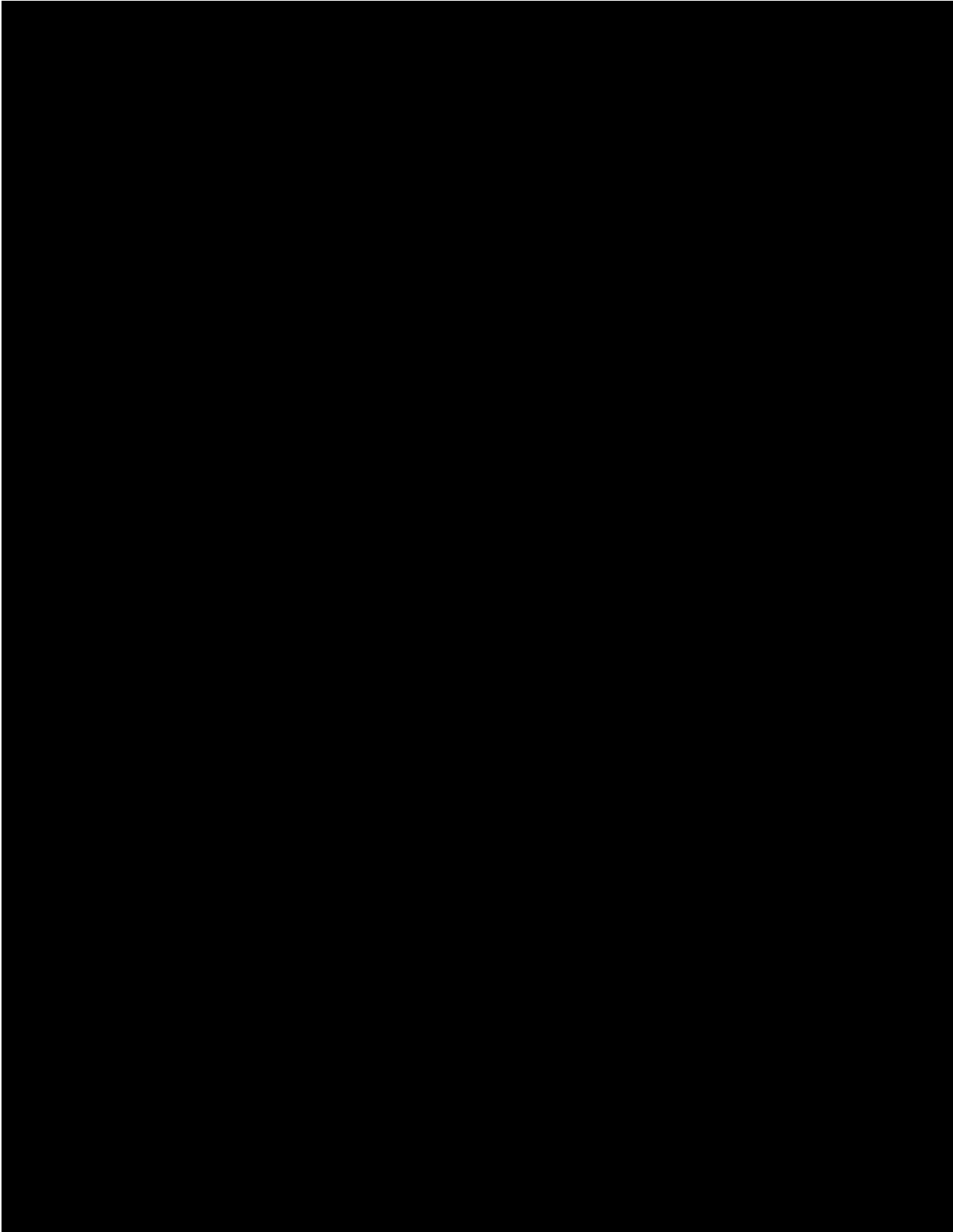


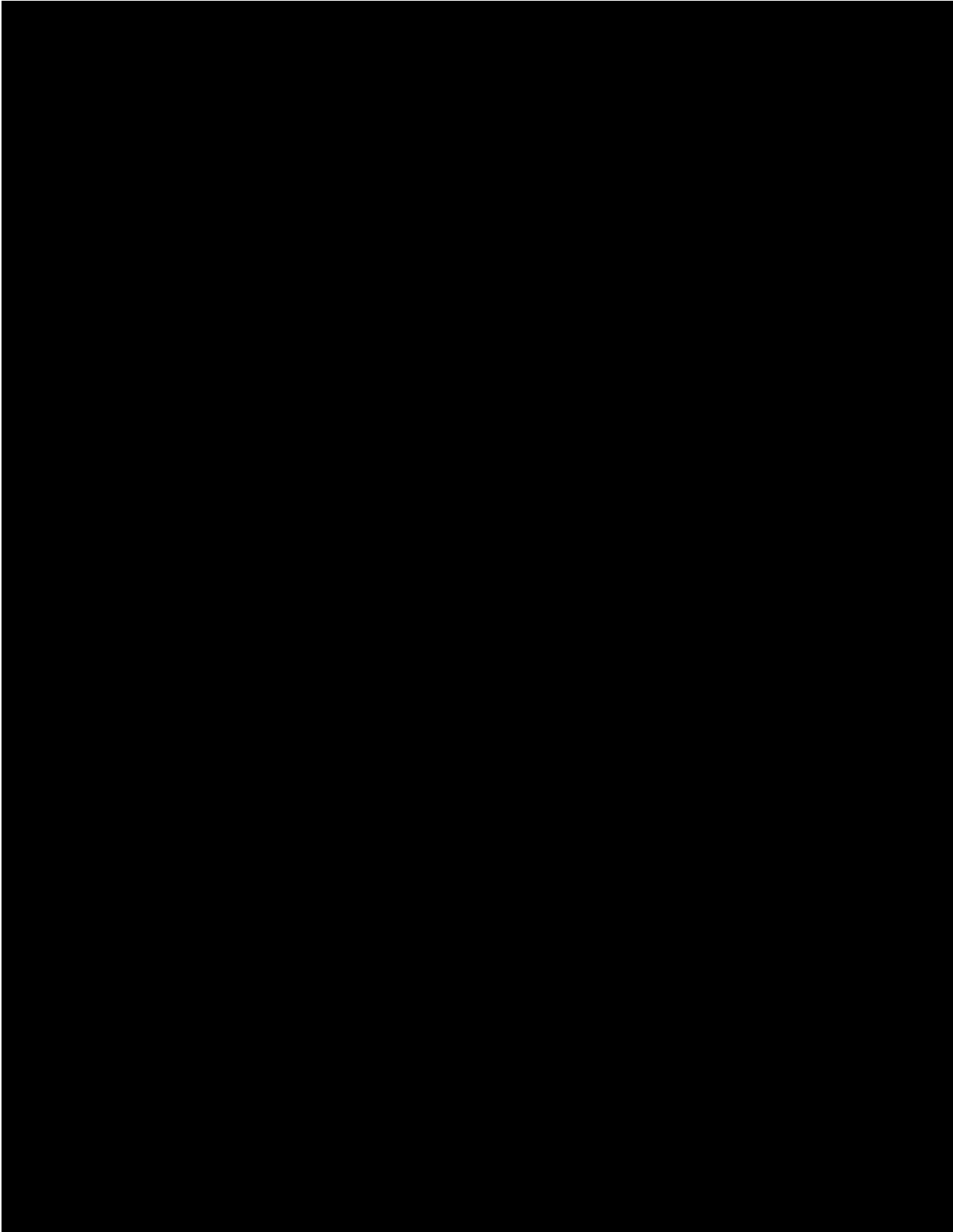


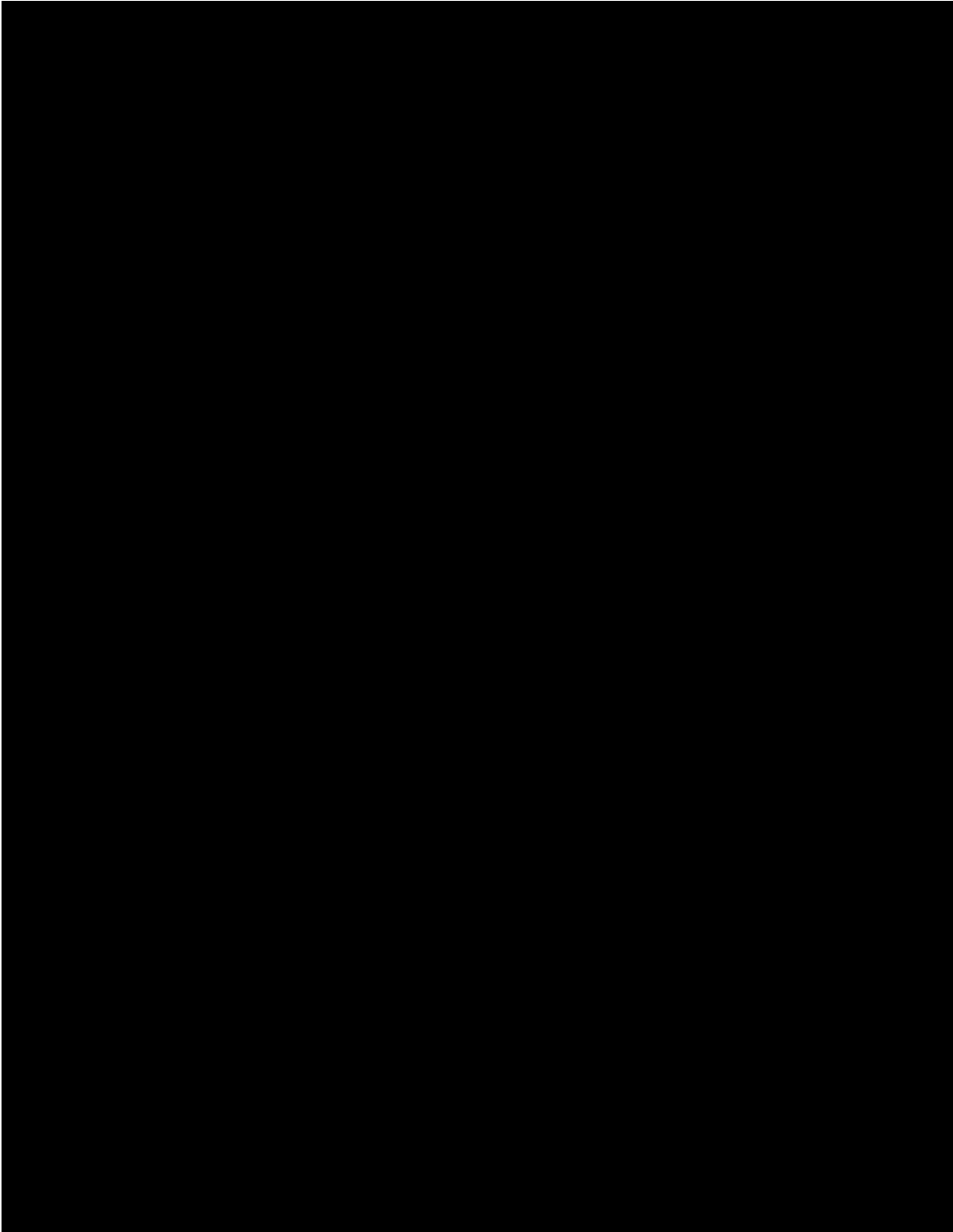


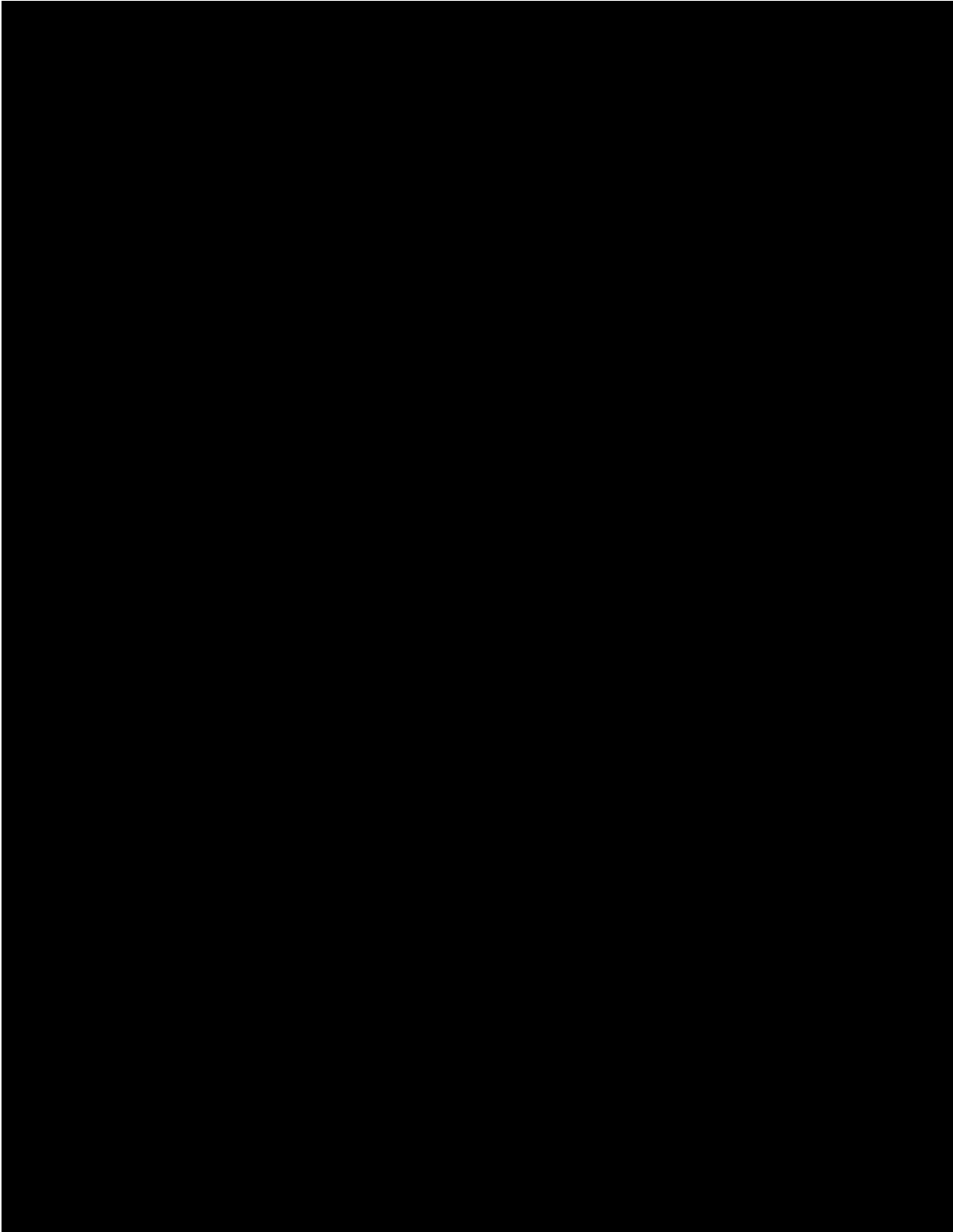








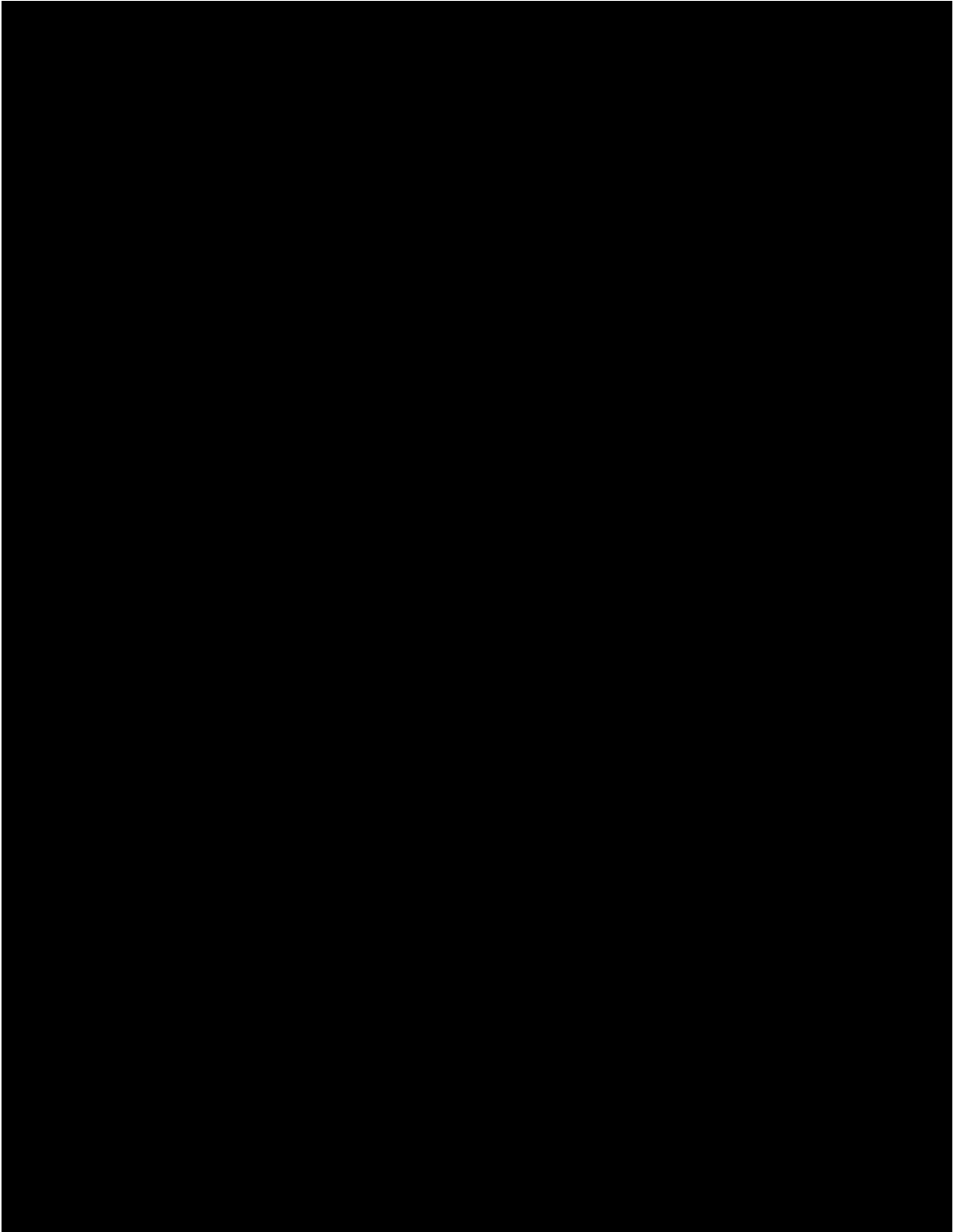


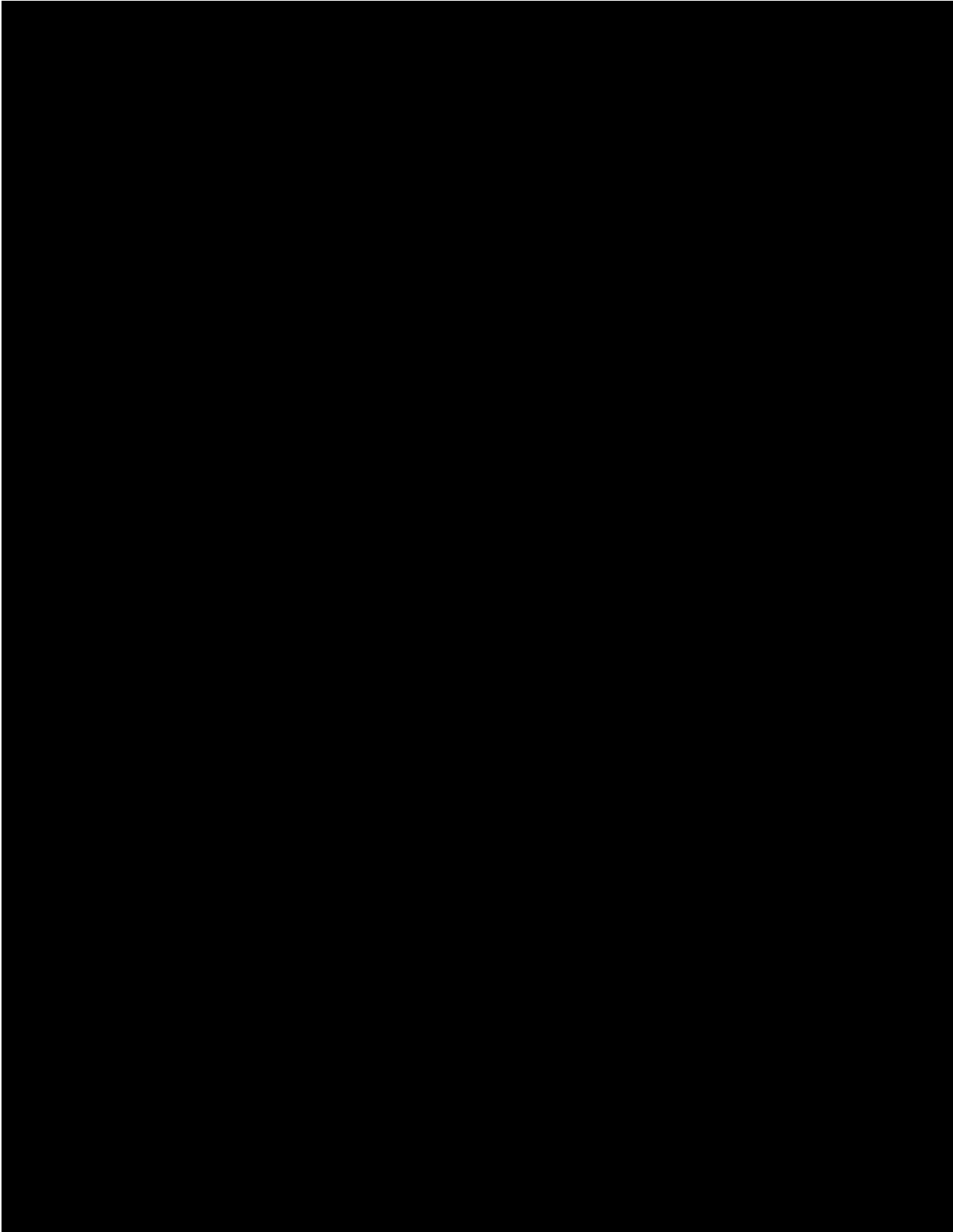


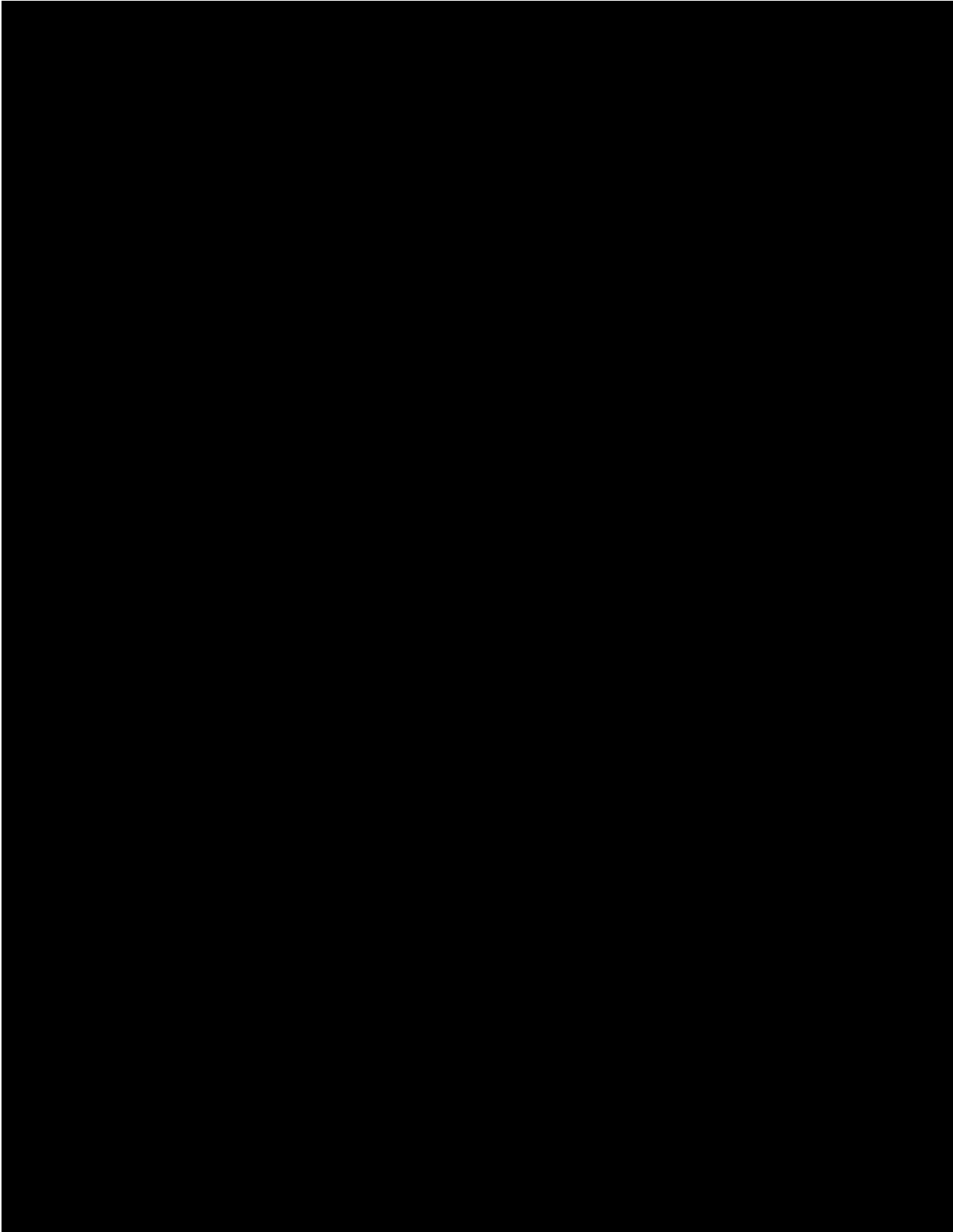


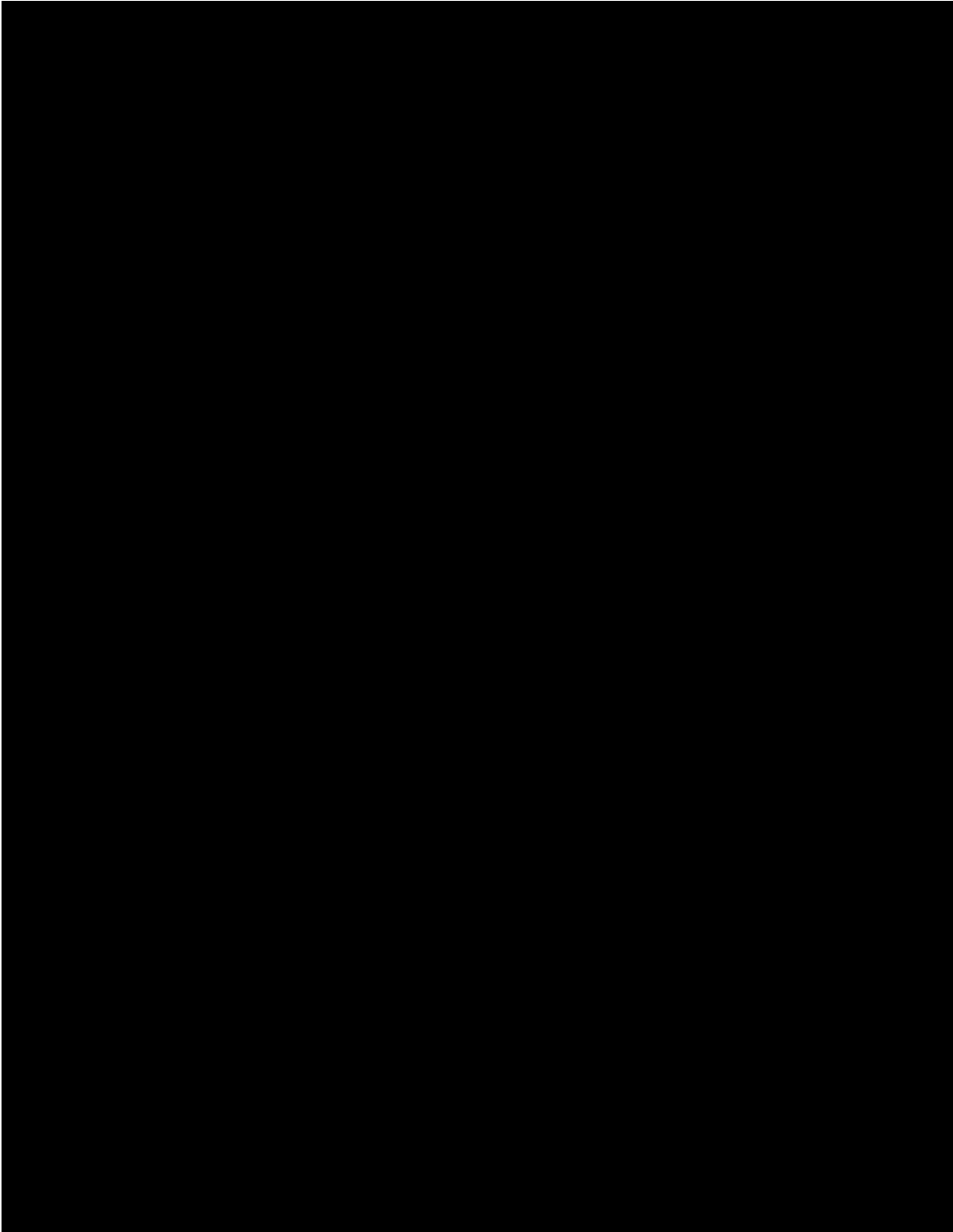
APPENDIX D – SQA: HYBRID SOFTWARE REQUIREMENTS SPECIFICATION AND TRACEABILITY MATRIX (SPC)

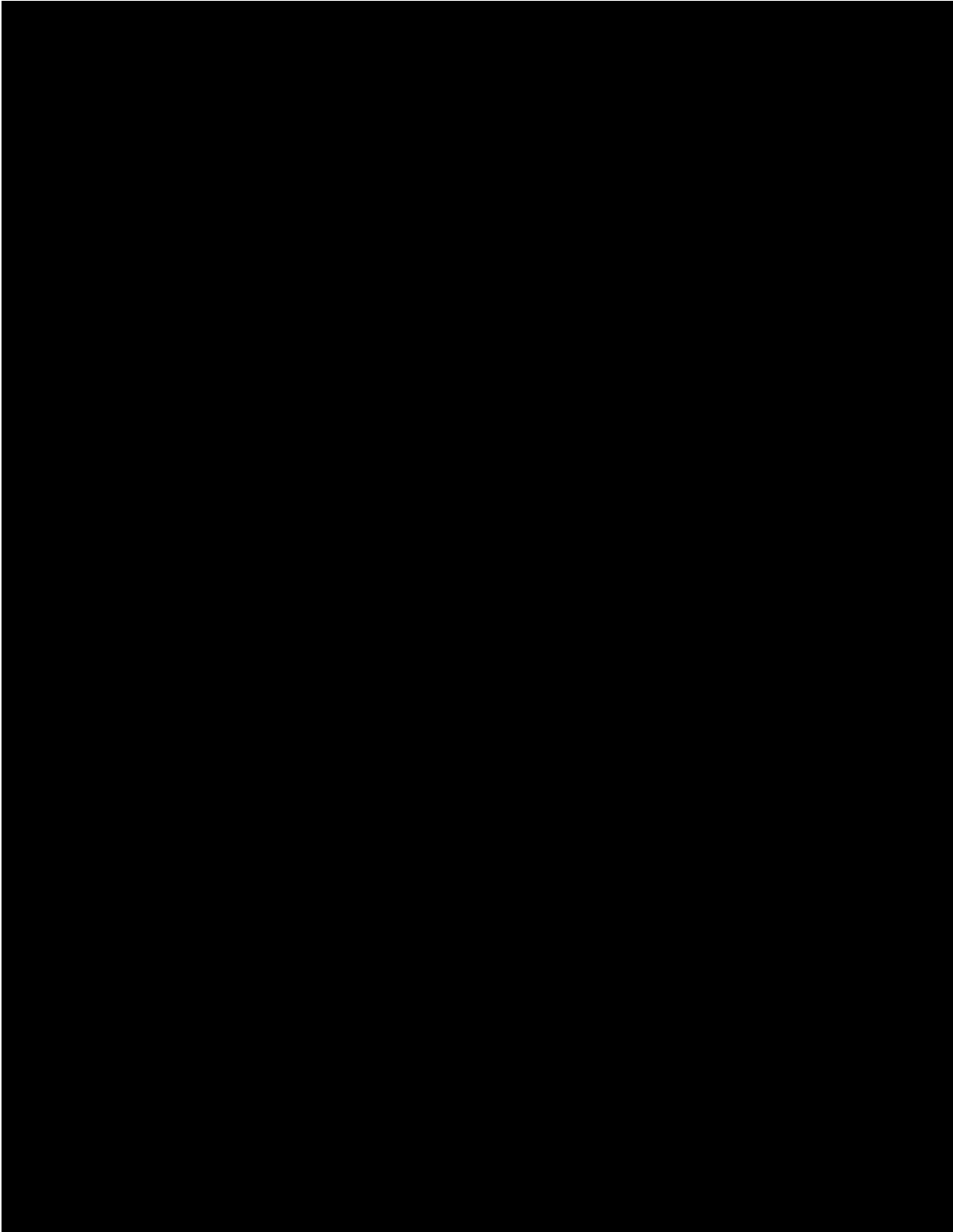


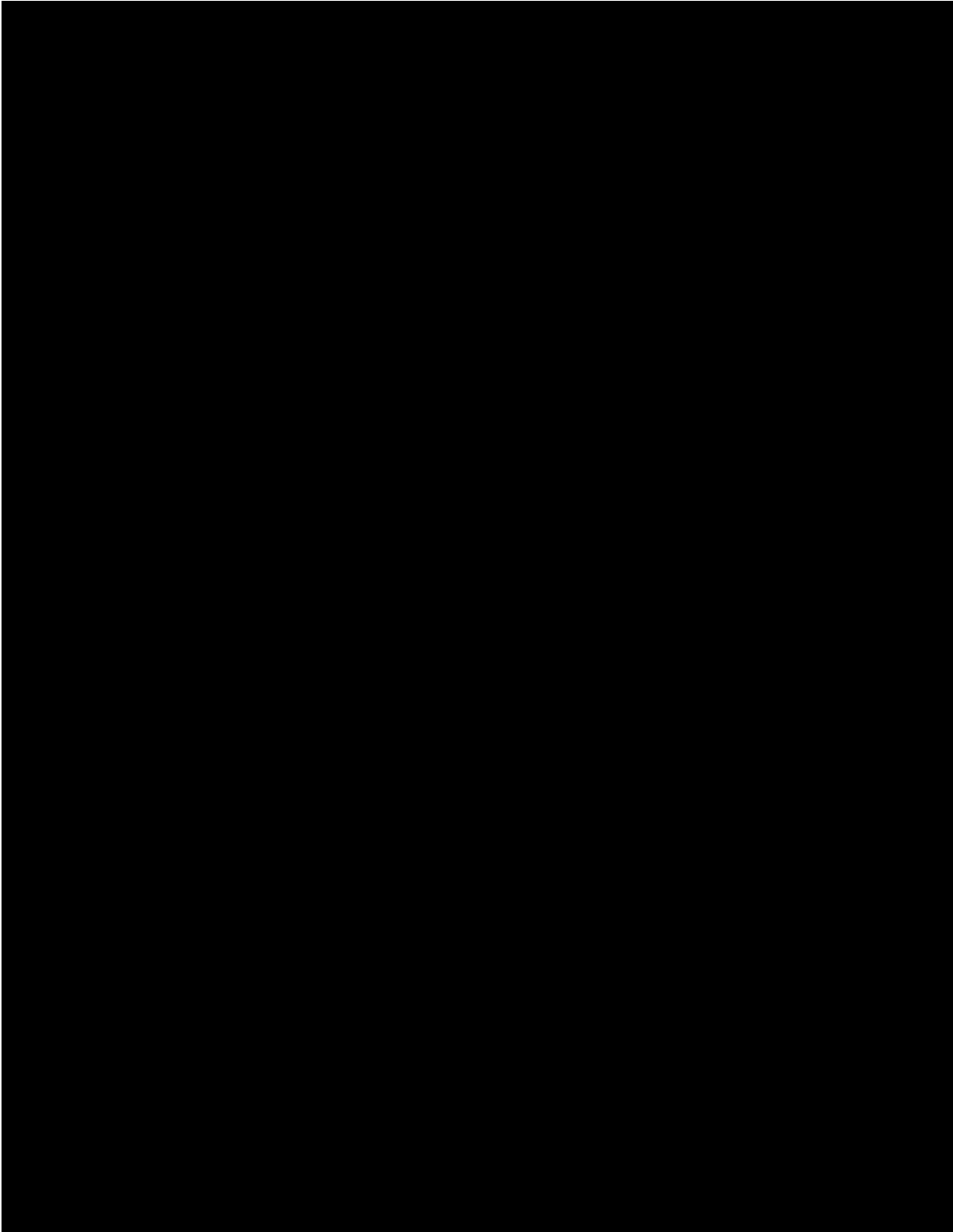


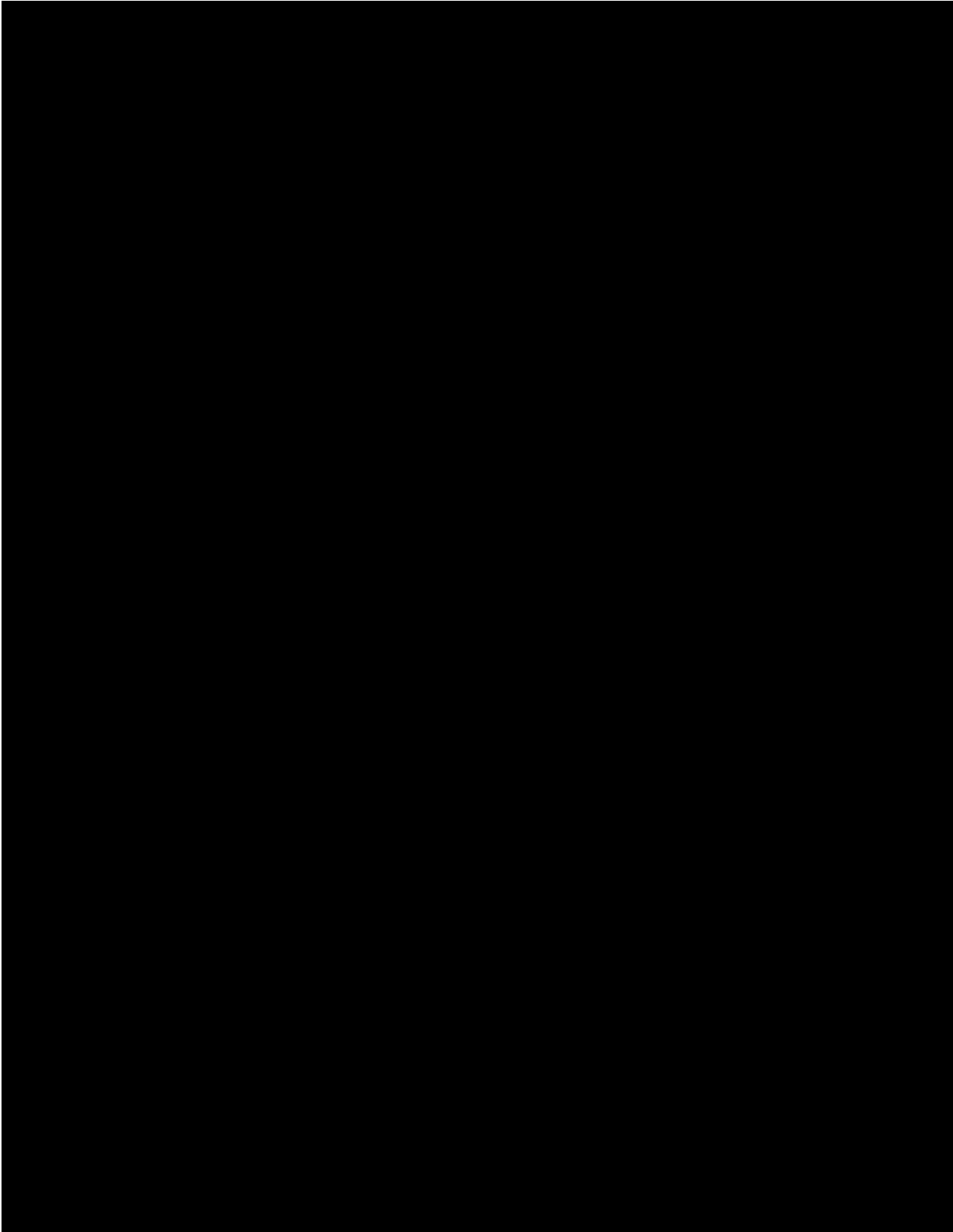


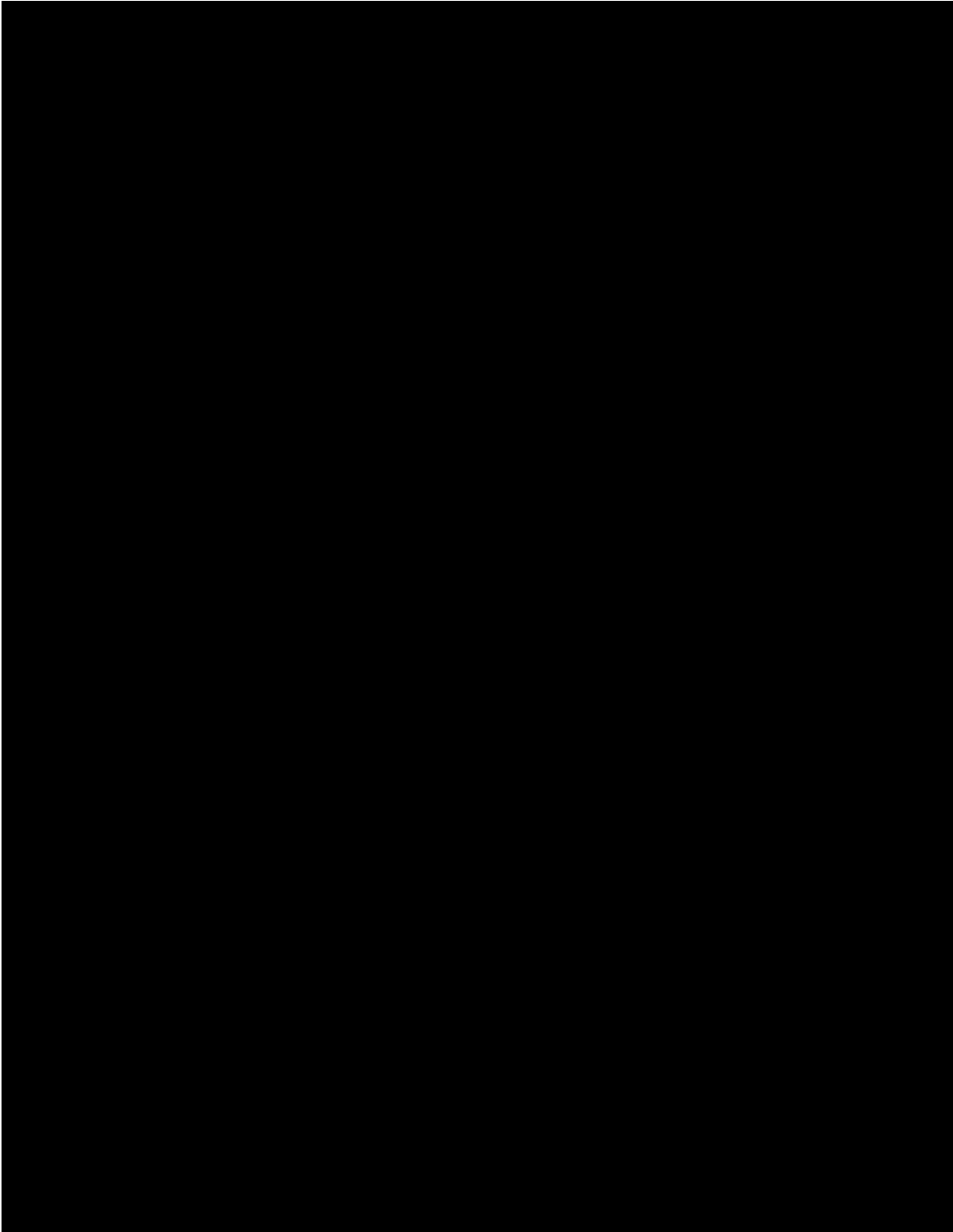


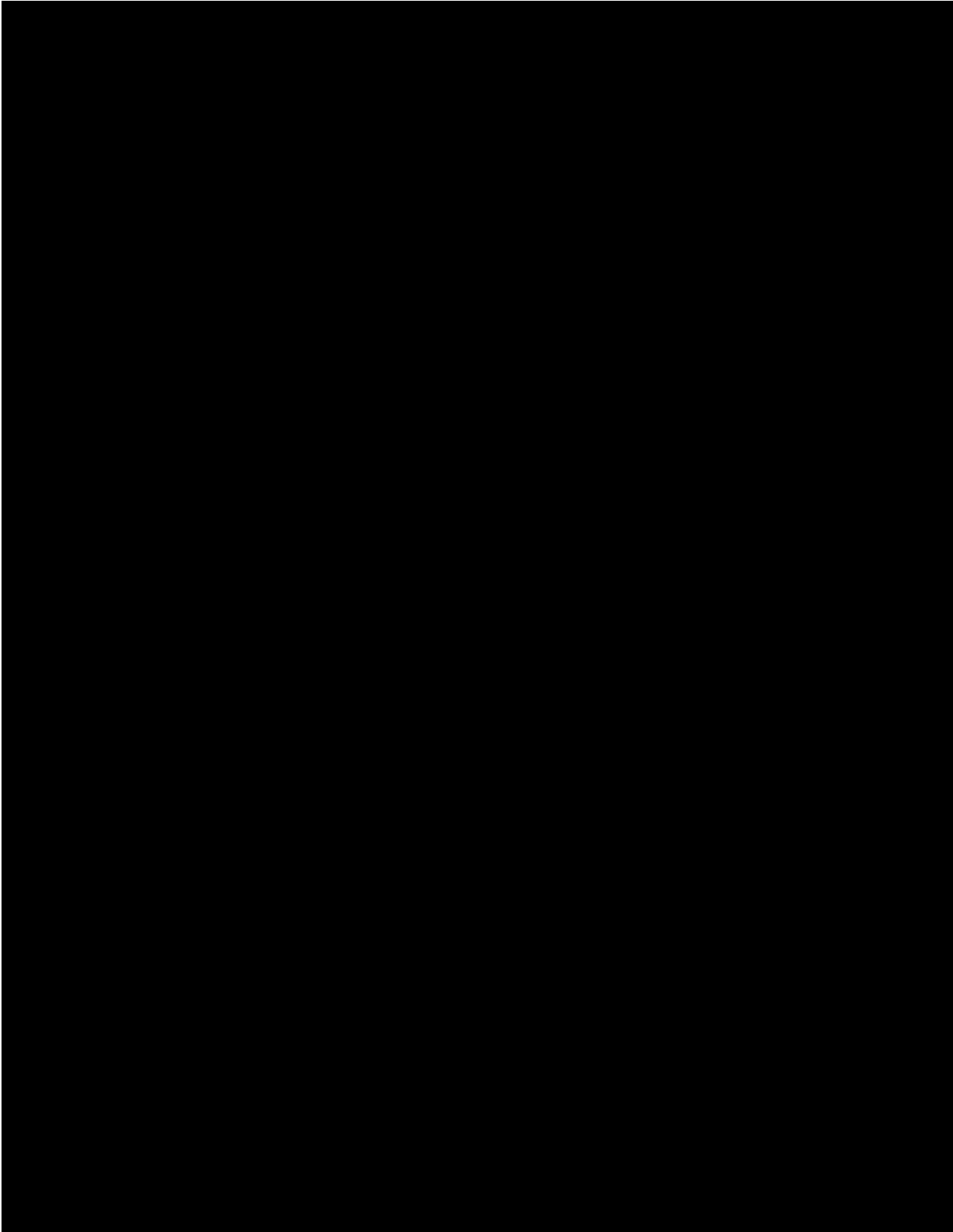


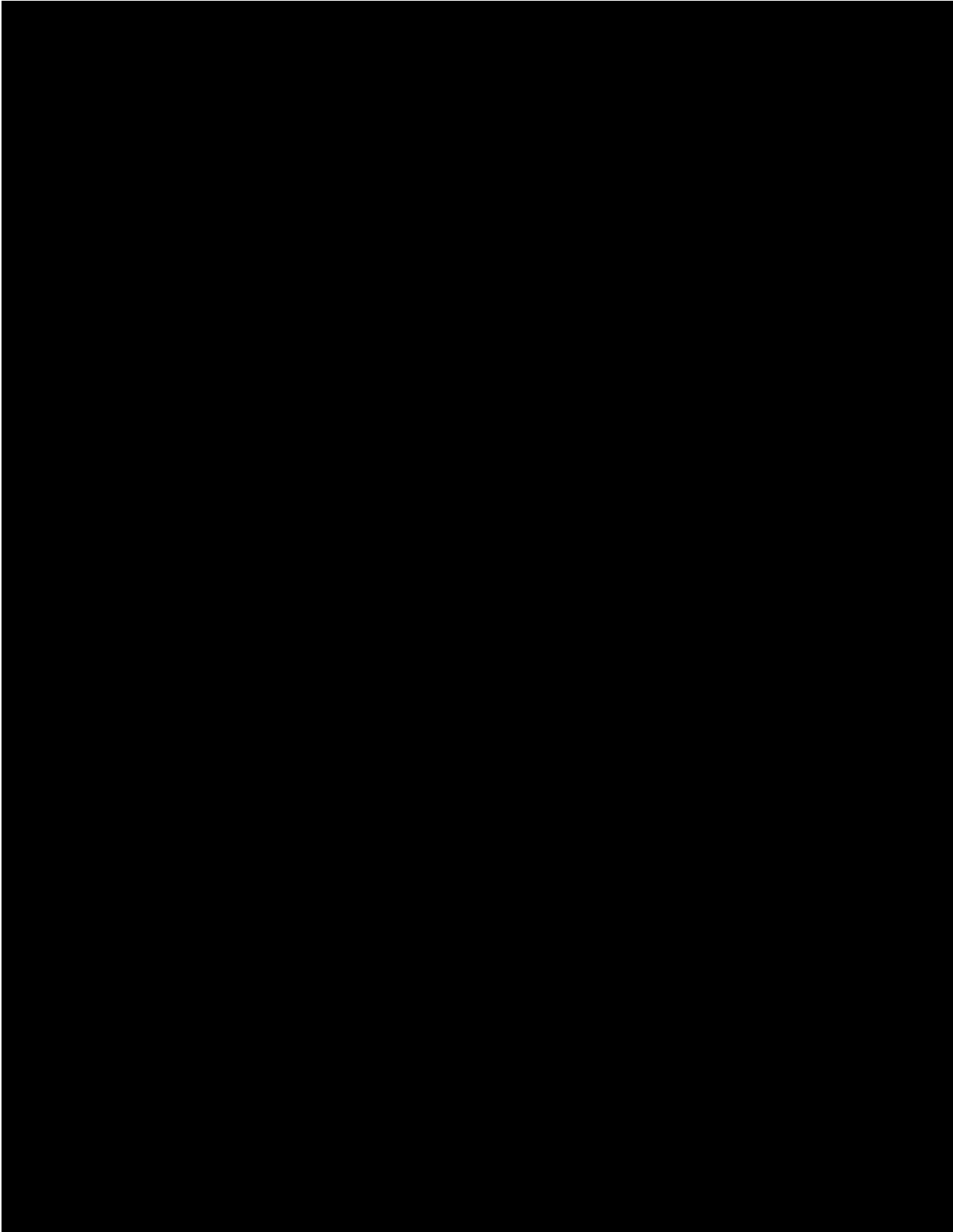


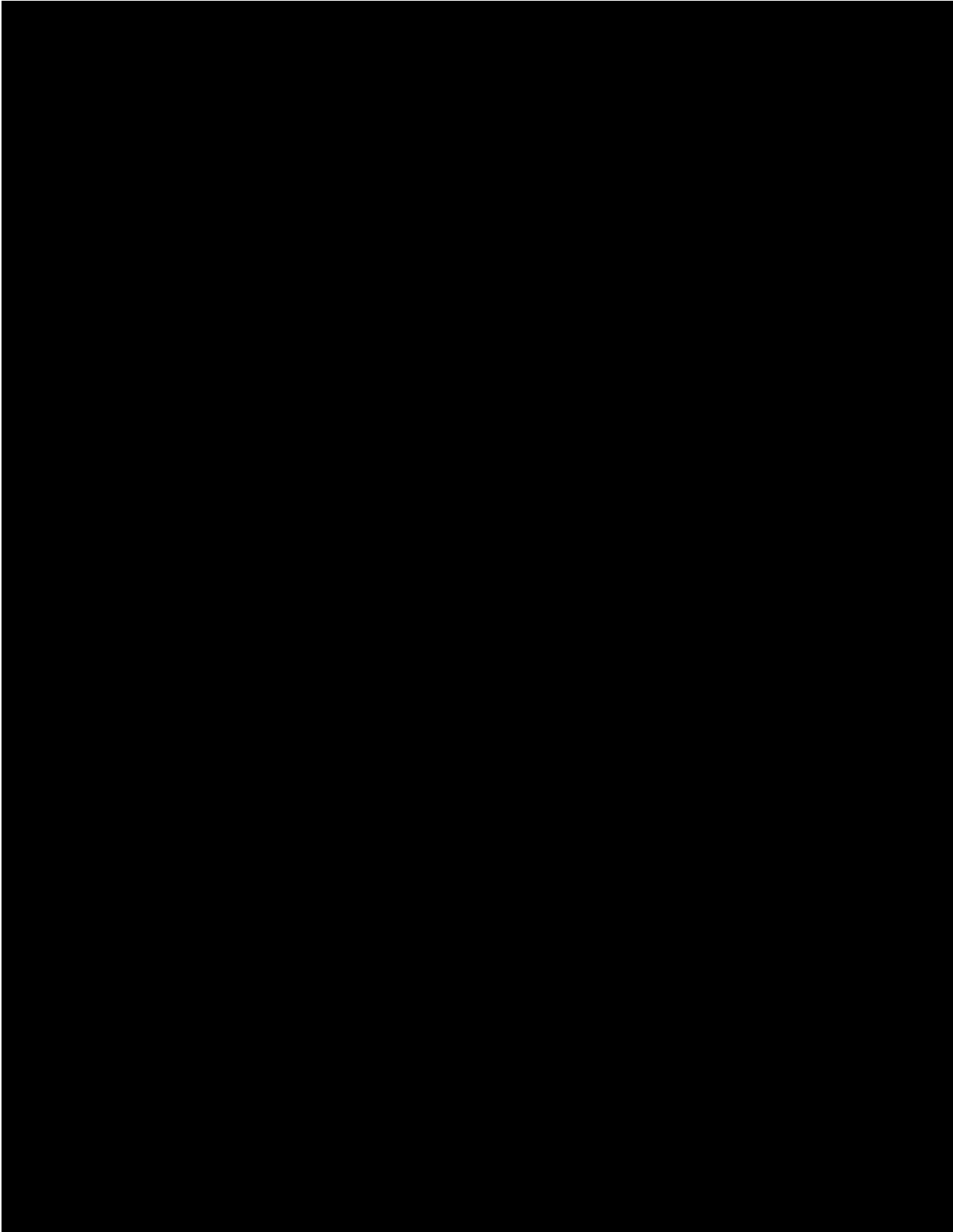


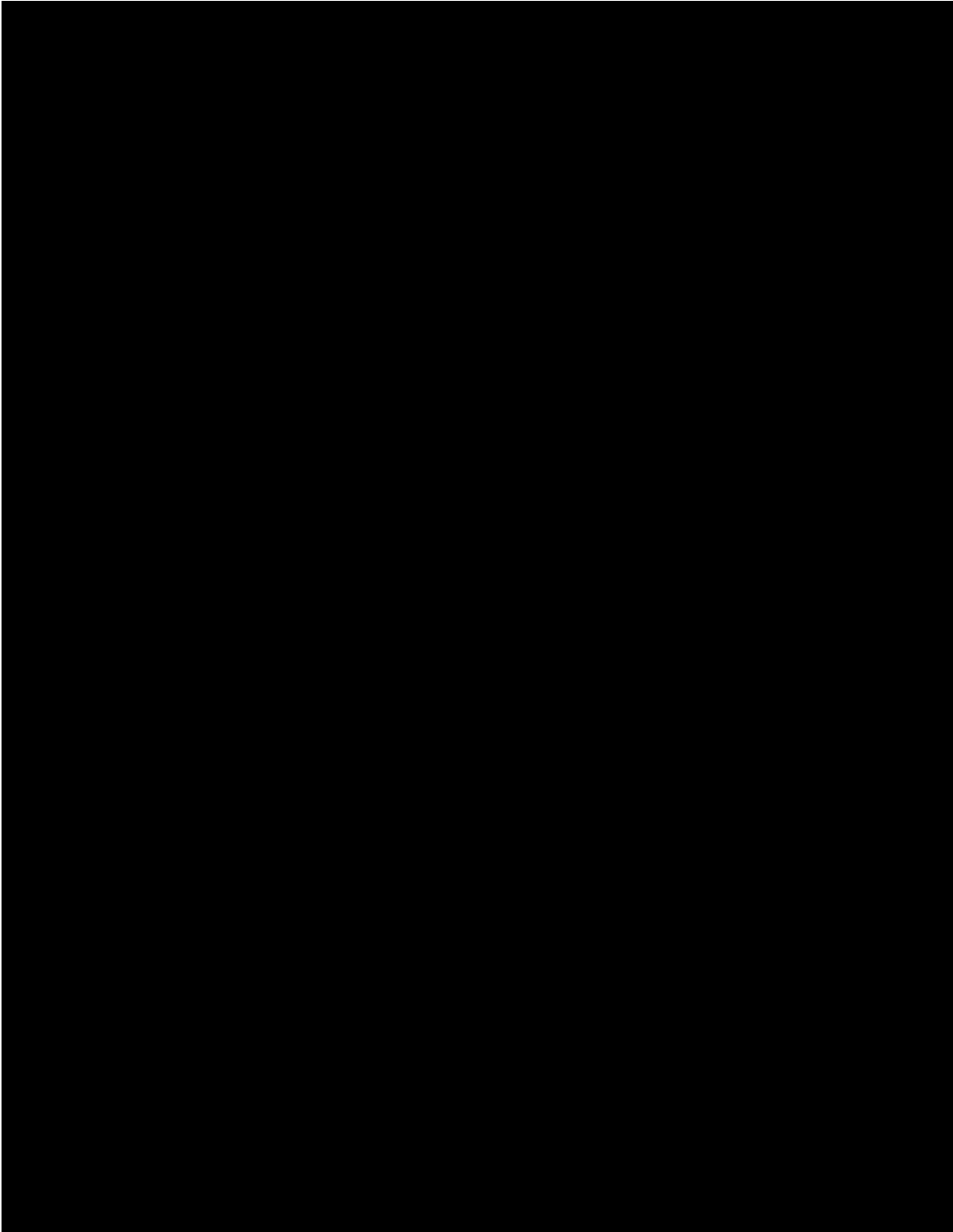


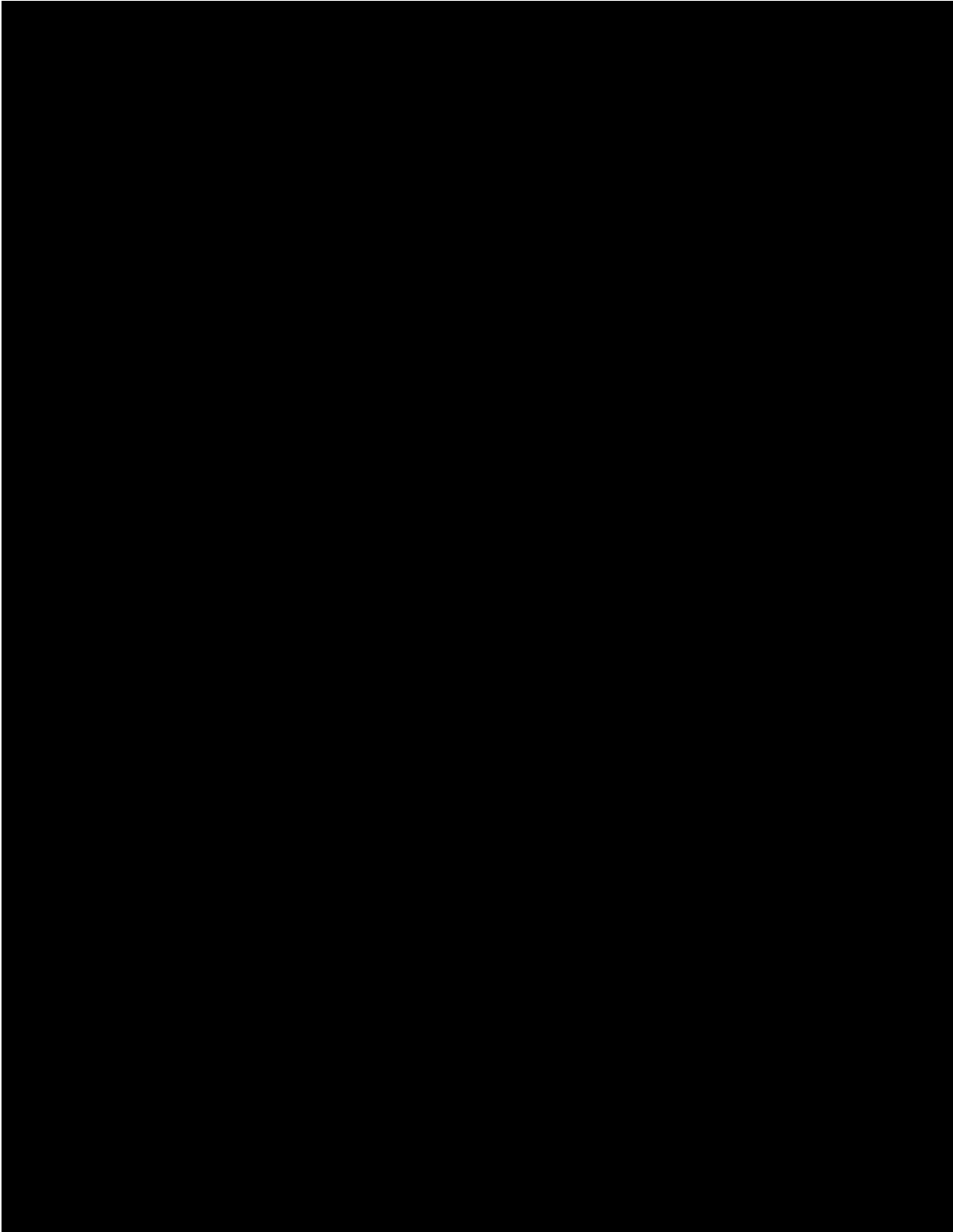


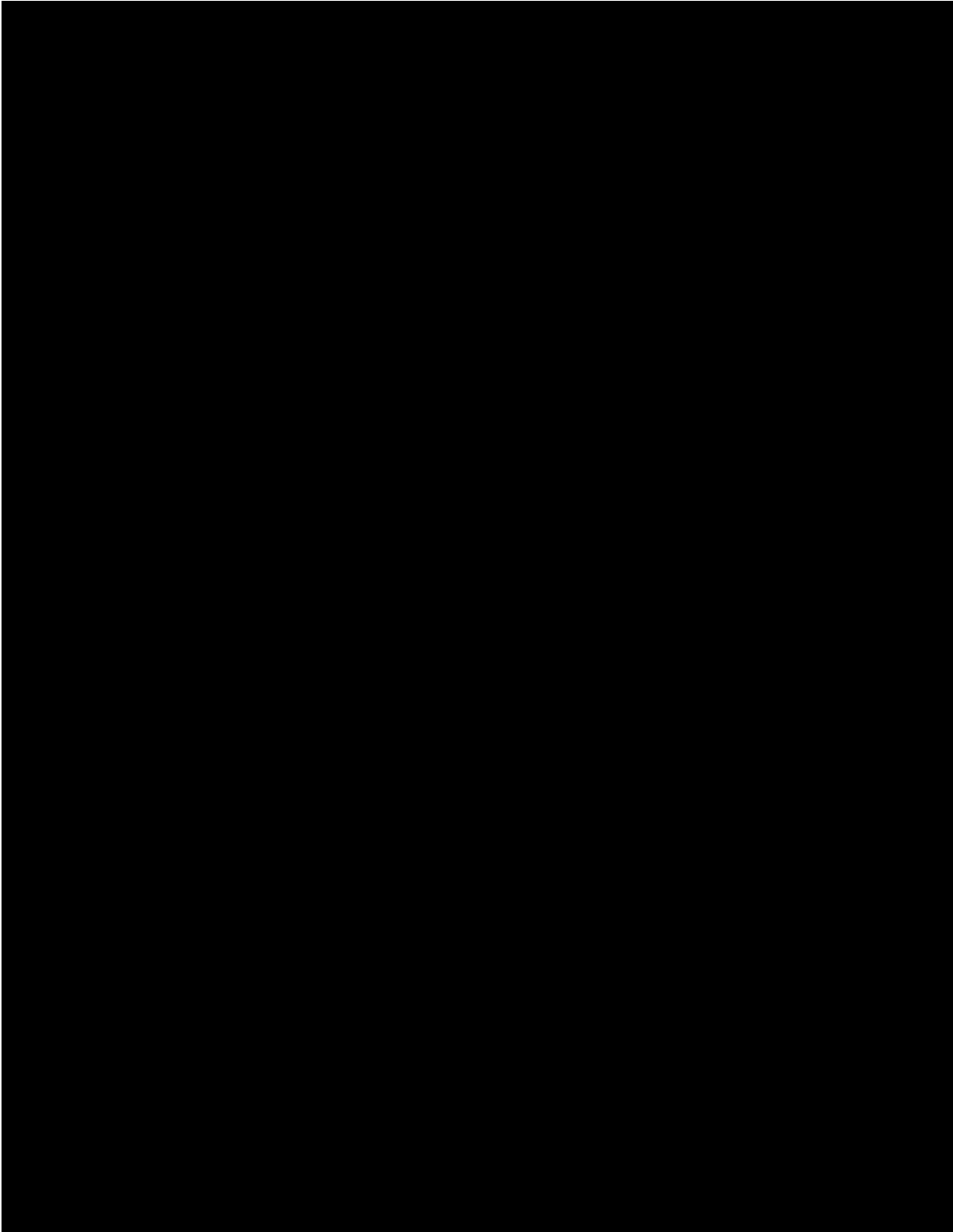


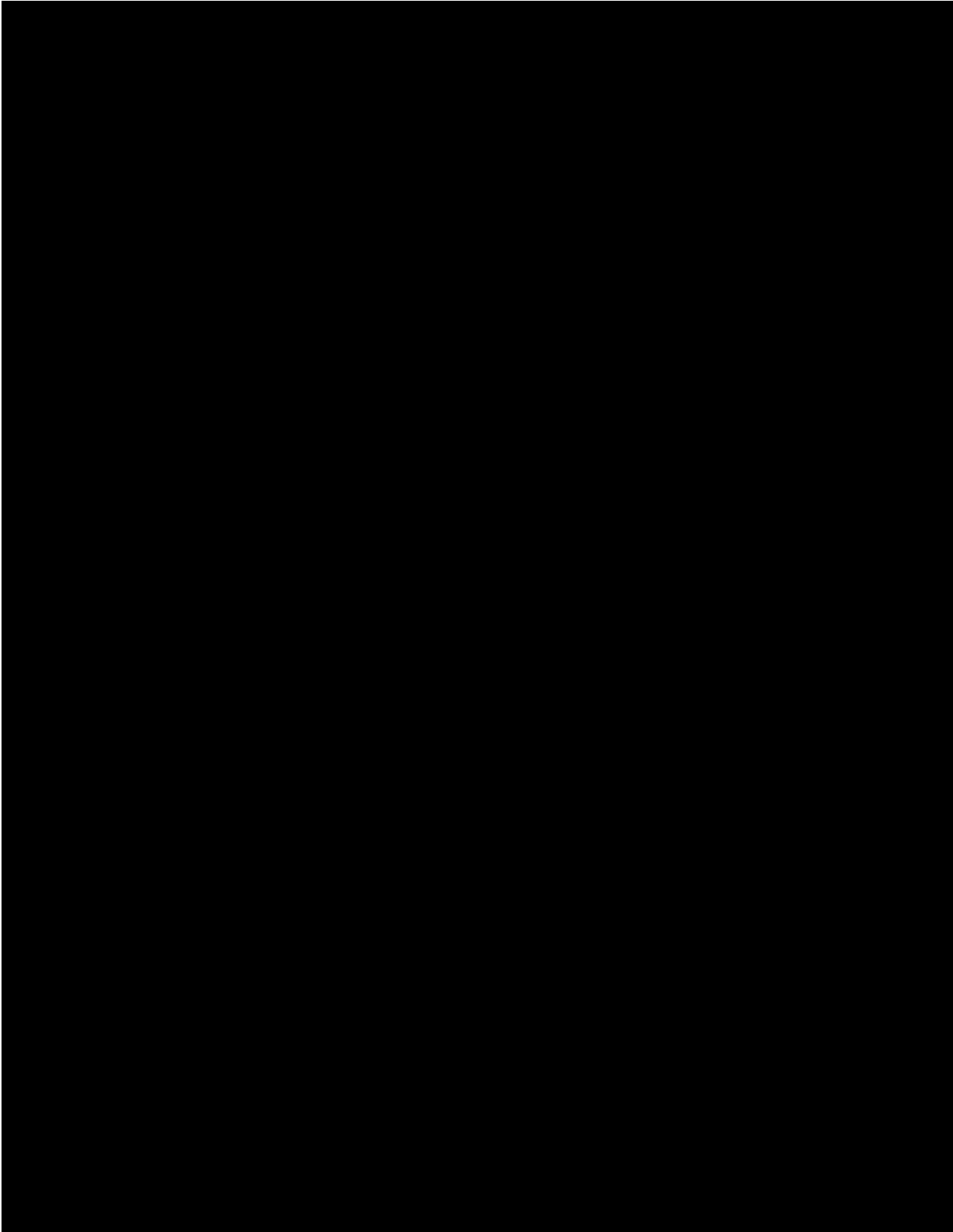


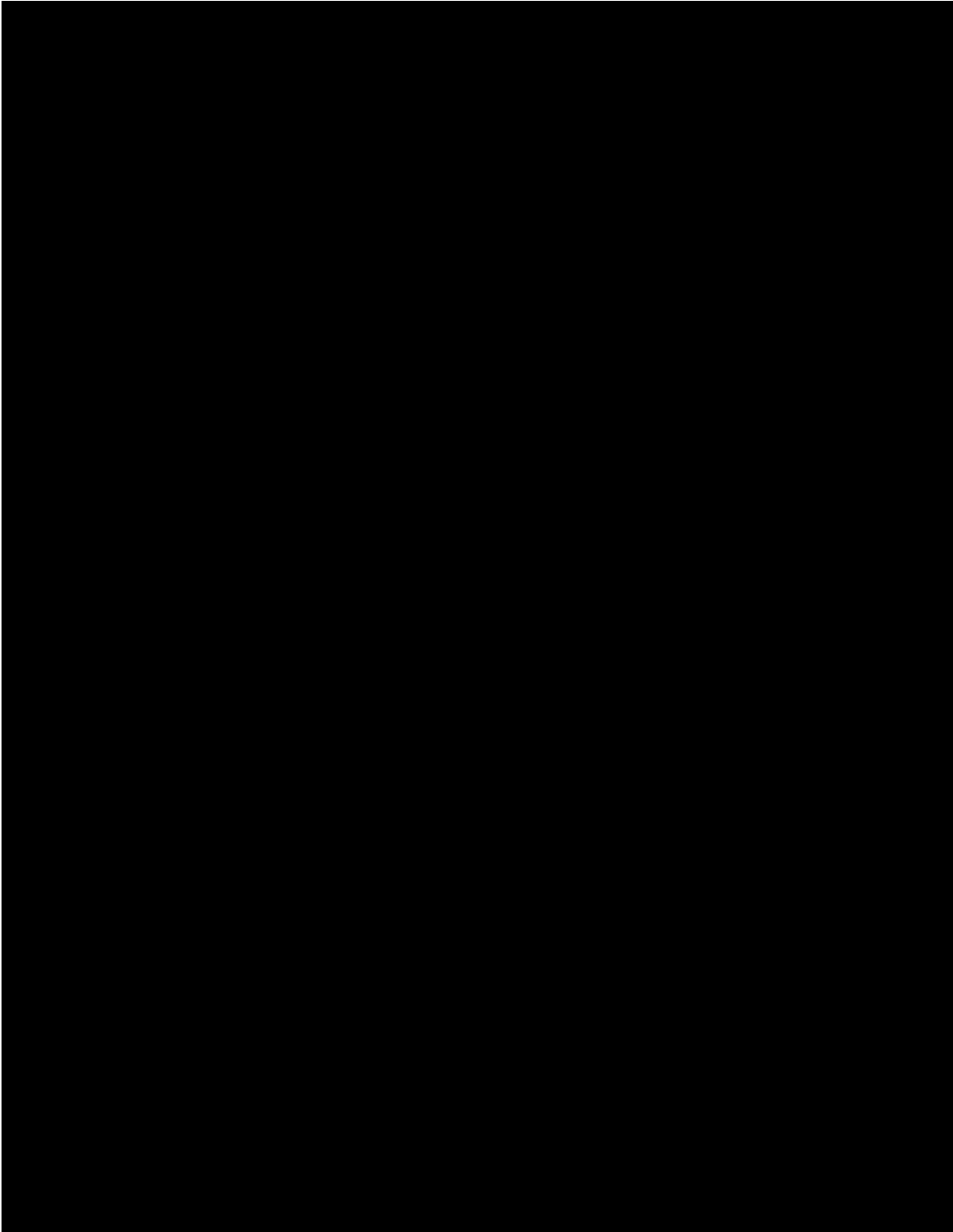


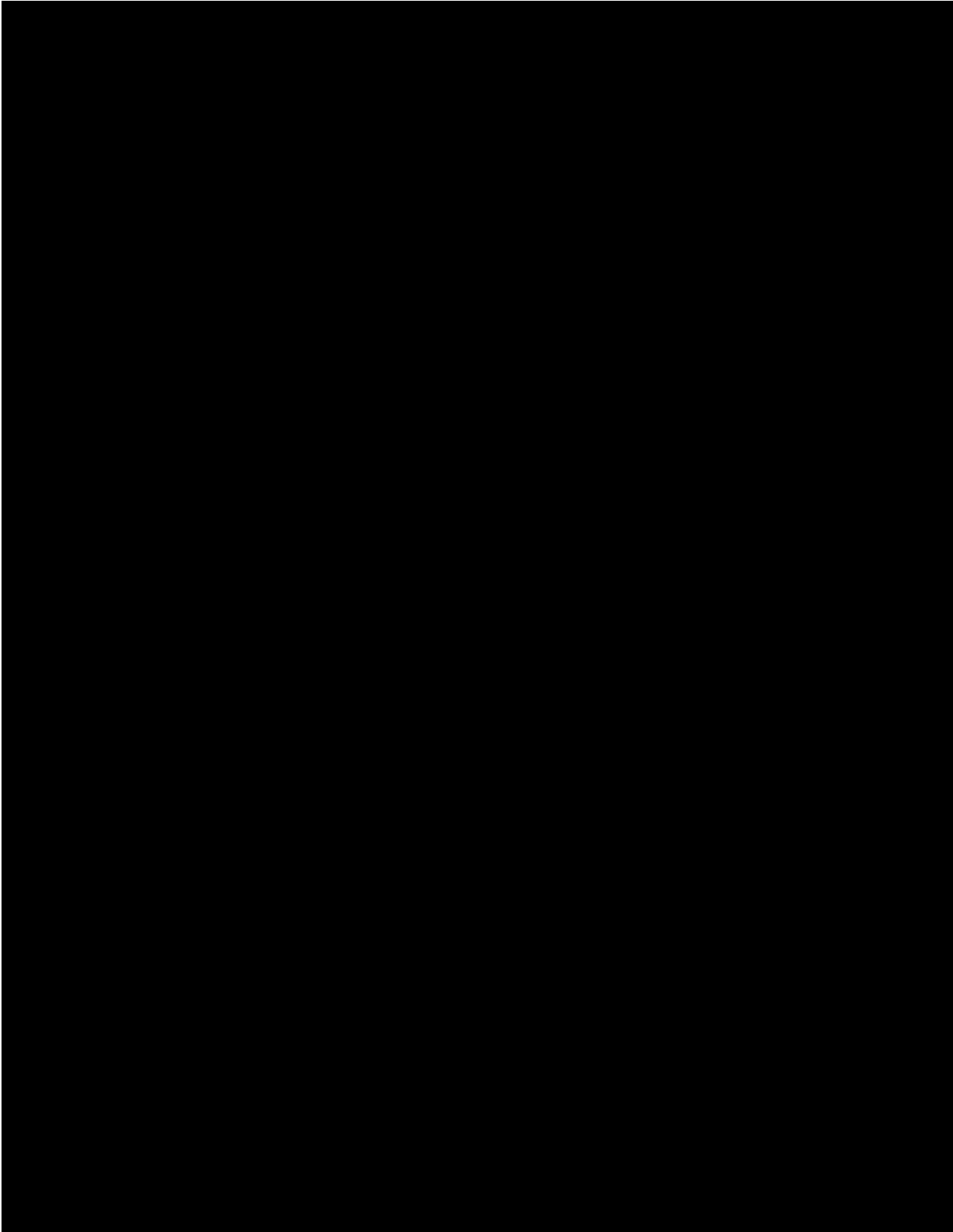


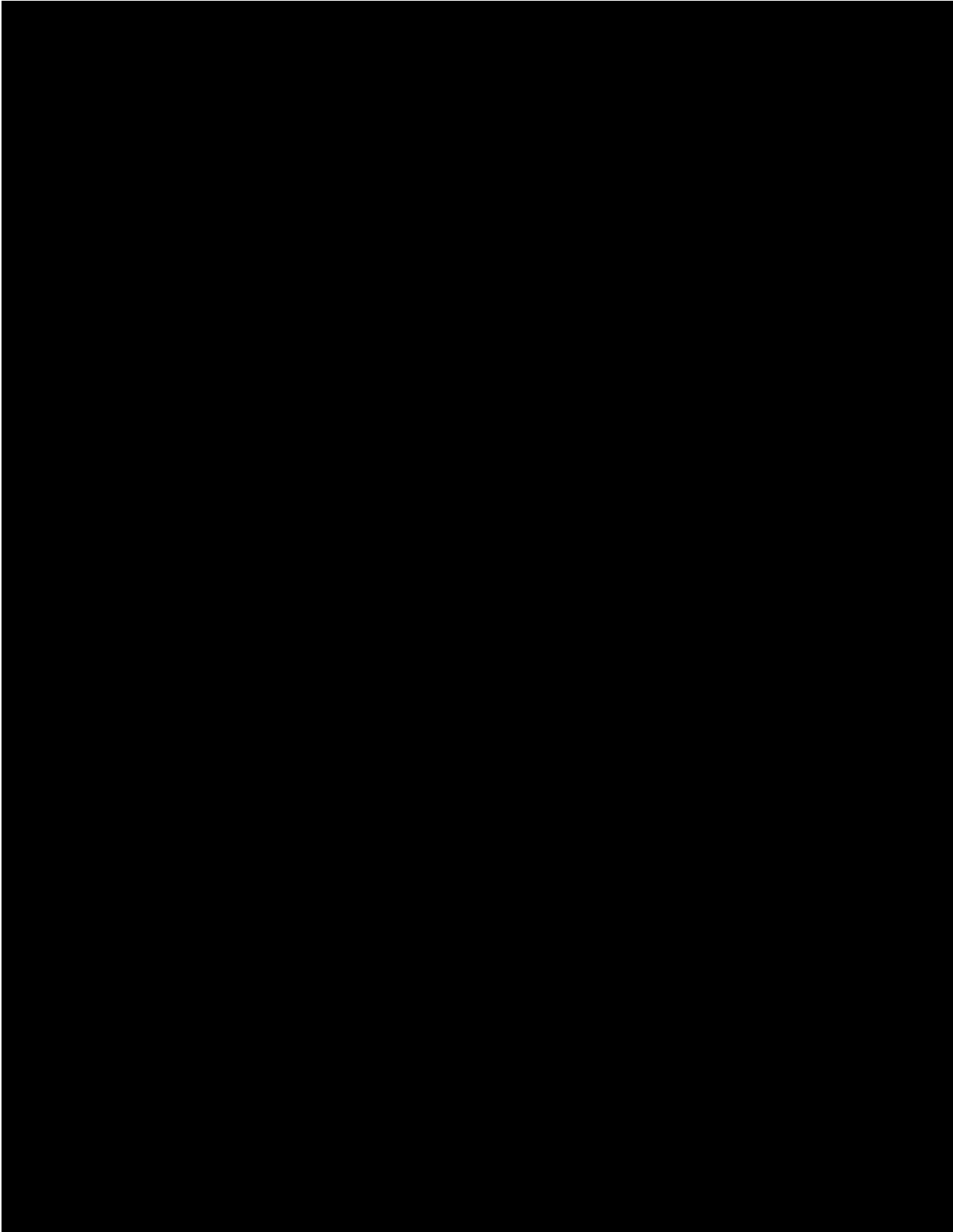















APPENDIX E – SQA: HYBRID CONFIGURATION ITEM LIST

	<div data-bbox="1149 340 1380 407">Document ID: 1234-5678 Revision ID: 01 Effective Date: 01/01/2024</div> <div data-bbox="678 722 737 751">Title:</div> <div data-bbox="678 844 1263 961">HYBRID Configuration Item List</div> <div data-bbox="678 1054 867 1083">Author: John Doe</div> <div data-bbox="678 1524 1224 1570">This document is the property of the Idaho National Laboratory operated by the Idaho Energy Alliance.</div>
---	---

Mathematics 2021, 9, 1009

Mathematics 2021, 9, 1009	Article	230 of 236
Mathematics 2021, 9, 1009	View Article	1
Mathematics 2021, 9, 1009	References	1000/2000
Mathematics 2021, 9, 1009	Page 27 of 102	

Copyright: © 2021 by the author(s).

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).

REFERENCES

Ref.	Author	Article Title	Journal Name
1.	Wang, Z.	On the existence of solutions for the Cauchy problem of the heat equation	Mathematics

Related Document's identifier

Document's identifier	2017-0296
Document's title	1
Document's date	11/20/2017 Page: 1 of 1

1. INTRODUCTION

This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1). This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1). This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1).

2. OBJECTIVE

This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1). This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1). This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1).

3. SCOPE AND LIMITATIONS

This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1). This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1). This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1).

This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1). This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1). This document is related to the 2017-0296 Document's energy production status (1.1.1) (1.1.1.1).

4. REFERENCES

4.1.1. Document's identifier, Document's title, Document's date

Related Document's identifier

Document Classification and Document Identifier	Document Identifier	Title / Page	
		Document Title	Page Count
Document Classification	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count
	Document Identifier	Document Title	Page Count