Gas Turbine Engine Performance Model Applications using an Object-Oriented Simulation Tool

A. Alexiou  K. Mathioudakis

Laboratory of Thermal Turbomachines
National Technical University of Athens
Paper Objectives

Demonstrate the use and advantages of general purpose object-oriented simulation environments for the following applications:

1. Build an engine model from existing engine components and run steady state and transient calculations

2. Develop and integrate new components in existing engine models

3. Access an engine model from external applications

4. Use external routines (FORTRAN, C, CPP) in simulations
Presentation Contents

- Building & Running an Engine Model
- Developing & Using a New Component
  - Component Syntax
  - The Cooled Turbine Component
- Accessing a Model from an External Application
- Using External Code in an Engine Model
- Conclusions
Simulation Environment

The commercial simulation environment presented in ASME GT-2005-68678 is used to implement the applications described in this paper.

The tool uses a high-level object-oriented language (EL) for modeling physical systems.

The most important concept in EL is the component which contains a mathematical description of the corresponding real-world component.

Components are joined together through their ports. Ports define the set of variables to be interchanged between components.

Components & ports are stored in a library.

For the purpose of this paper, it is assumed that such a library of basic gas turbine components & ports is available.
Building & Running an Engine Model

Develop & run an engine model in just 3 steps

Step 1
Build an engine model graphically

Step 2
Define Partition & Experiment

Step 3
Run simulation & view results

Gas Turbine Engine Performance Model Applications using an Object-Oriented Simulation Tool
Component Syntax (I)

ABSTRACT COMPONENT absCompressor IS_A gasTurbo

Does NOT represent a physical component and cannot be instantiated.

Defines interface & methods that can be shared by multiple Components
Component Syntax (I)

ABSTRACT COMPONENT absCompressor IS_A gasTurbo

Declares inheritance
Component Syntax (I)

ABSTRACT COMPONENT absCompressor IS_A gasTurbo

PORTS
  IN Fluid F_in

DATA
  REAL inertia = 0

Public Part
Component Syntax (I)

ABSTRACT COMPONENT absCompressor IS_A gasTurbo PORTS

IN Fluid F_in

DATA

REAL inertia = 0

Define direction, type & name of Port
Component Syntax (I)

ABSTRACT COMPONENT absCompressor IS_A gasTurbo

PORTS

IN Fluid F_in

DATA

REAL inertia = 0

Define data type & name. Specify default value
Component Syntax (I)

ABSTRACT COMPONENT absCompressor IS_A gasTurbo
PORTS
  IN Fluid F_in
DATA
  REAL inertia = 0

This appears & can be edited in the Attributes Editor window
Component Syntax (I)

ABSTRACT COMPONENT absCompressor IS_A gasTurbo
PORTS
   IN Fluid F_in
DATA
   REAL inertia = 0
DECLS
   REAL Nc

Define local component variables (private)
Component Syntax (I)

ABSTRACT COMPONENT absCompressor IS_A gasTurbo

PORTS

IN Fluid F_in

DATA

REAL inertia = 0

DECLS

REAL Nc

TOPOLOGY

PATH F_in TO F_out

Define sub-components & connection paths
Component Syntax (I)

ABSTRACT COMPONENT absCompressor IS_A gasTurbo

PORTS

    IN Fluid F_in

DATA

    REAL inertia = 0

DECLS

    REAL Nc

TOPOLOGY

    PATH F_in TO F_out

INIT

    readCompressorMap(WcTab, effTab, PRtab, SMtab)

    Assign initial values to component variables
Component Syntax (II)

DISCRETE

`ASSERT (SMpct > 5) WARNING “Compressor \ working beyond Surge Line”`

Describe the conditions & effects of discrete events
**Component Syntax (II)**

**DISCRETE**

**ASSERT** \((\text{SMpct} > 5)\) **WARNING** “Compressor \ working beyond Surge Line”

**CONTINUOUS**

--Shaft Dynamics
\[
D_{pwr} = \text{inertia} \times \left(\frac{\text{PI}}{30}\right)^2 \times N_{\text{mech}} \times N_{\text{mech}}'
\]

--Compressor Power
\[
pwr = - (F_{\text{in.W}} \times (ht_{\text{out}} - ht_{\text{in}}))
\]

Write differential-algebraic equations describing component’s continuous behaviour.
Developing & Using a New Turbine Component (I)

Turbine model ‘inherited’ from original FORTRAN code
Developing & Using a New Turbine Component (I)

Turbine model ‘inherited’ from original FORTRAN code

Drawbacks

3 components are used to model each cooled turbine
Developing & Using a New Turbine Component (I)

Turbine model ‘inherited’ from original FORTRAN code

Drawbacks

Work potential of cooling flows defined at bleed location
1 bleed port/return component
Developing & Using a New Turbine Component (I)

Turbine model ‘inherited’ from original FORTRAN code

Drawbacks

Can NOT calculate SOT or Thermodynamic efficiency
Developing & Using a New Turbine Component (II)

Create new component by modifying existing one

PORTS
IN Sas SasP[nSasP] "Secondary Air System Flows"

DATA
ENUM switchEffType effType = EqSS
    "Select efficiency type definition (-)"
REAL Wtw_q_Wc[1] = {0}
    "Fraction of each Sas flow doing work in the equivalent turbine rotor (-)"
REAL WNGV_q_Wc[1] = {0}
    "Fraction of each Sas flow used for NGV cooling (-)"
REAL Wpump_q_Wc[1] = {0}
    "Fraction of each Sas flow pumped up through rotor blades (-)"
REAL Rdia[1] = {0.6}
    "Rotor Mean Blade Diameter used in Pumping Power Calculation (m)"
Developing & Using a New Turbine Component (II)

Create new component by modifying existing one

In CONTINUOUS calculate:

- the stator exit conditions from mixing the inlet main flow and the NGV cooling flows
- the equivalent rotor inlet conditions from mixing the main inlet flow with the appropriate fraction of each cooling flow according to its work potential
- the power required to pump the rotor blade cooling flow according to the rotor mean blade diameter and the rotor blade cooling mass flows
- the turbine power and exit flow conditions according to the user selected efficiency definition
- the other efficiency definition
Developing & Using a New Turbine Component (II)
Create new component by modifying existing one
Developing & Using a New Turbine Component (II)

Cooled Turbine Isentropic Efficiency Definition Effect on Performance Parameters
Accessing a Model from an External Application (I)

Engine Model
- Partition
- Experiment
- DLL
- ActiveX

User Interface
- Visual Basic
- Visual C++
- Delphi
- Excel
- Access
- etc
Accessing a Model from an External Application (II)

Visual Basic GUI for engine condition monitoring
Accessing a Model from an External Application (III)

MS Excel toolbar

Open Experiment
Accessing a Model from an External Application (III)

MS Excel toolbar

MS Excel sheet for turbofan engine calculations

- Design / off design
- Straight / Adaptive
- Steady / Multi-steady / Transient
Using External Code in an Engine Model (I)

To use existing FORTRAN, C or C++ code

1. Declare Interface

"FORTRAN" FUNCTION NO_TYPE NewtonRaphson

  ( FUNC_PTR funct, -- Function Pointer
    IN INTEGER n, -- Number of Independent Variables
    IN INTEGER itermax, -- Max No of Iterations
    IN REAL tol, -- Required Tolerance
    IN REAL eps, -- Machine EPS
    OUT REAL x[], -- Array of Independent Variables
    OUT INTEGER ierror-- Flag
  )
Using External Code in an Engine Model (I)

To use existing FORTRAN, C or C++ code

1. Declare Interface
2. Specify location of object or library

IN "USER_LIBS\NTUA\ExternalObjects\NewtonRaphson.obj"
Using External Code in an Engine Model (I)

To use existing FORTRAN, C or C++ code

1. Declare Interface
2. Specify location of object or library
3. Use in Components and/or Experiments like normal EL functions

NewtonRaphson(fcn, 10, itermax, tol, eps, x, ierror)
Conclusions

Object-Oriented simulation environments offer great advantages for developing different engine performance modelling applications:

✓ Powerful object-oriented language for creating components & setting up simulations

✓ Advanced graphical user interface for creating engine models

✓ Mathematical model wizards

✓ Post-processing capabilities for viewing results

✓ Connectivity with other applications

✓ Compatibility with other programming languages

* Changing users’ programming philosophy & modelling approach could be an issue
PRopulsion Object-Oriented SImulation Software (PROOSIS)

Work Package 2.4 of

Integrated Project VIVACE

http://www.vivaceproject.com

Funded by the European Union Commission

PARTNERS

AIF, AVIO, CENAERO, CU, EA, ITP MTU, NLR, NTUA, Snecma, Techspace Aero, Turbomeca, USTUTT, Volvo Aero, IberEspacio
Gas Turbine Engine Performance Model Applications using an Object-Oriented Simulation Tool