Advanced Capabilities For Gas Turbine Engine Performance Simulations

A. Alexiou, **E.H. Baalbergen**,

O. Kogenhop, **K. Mathioudakis**, **P. Arendsen**

Laboratory of Thermal Turbomachines
National Technical University of Athens

National Aerospace Laboratory, NLR
The Netherlands

http://www.vivaceproject.com

Paper Objective & Definitions

Implementing Component ‘Zooming’ and Distributed Simulations in PRopulsion Object-Oriented SImulation Software

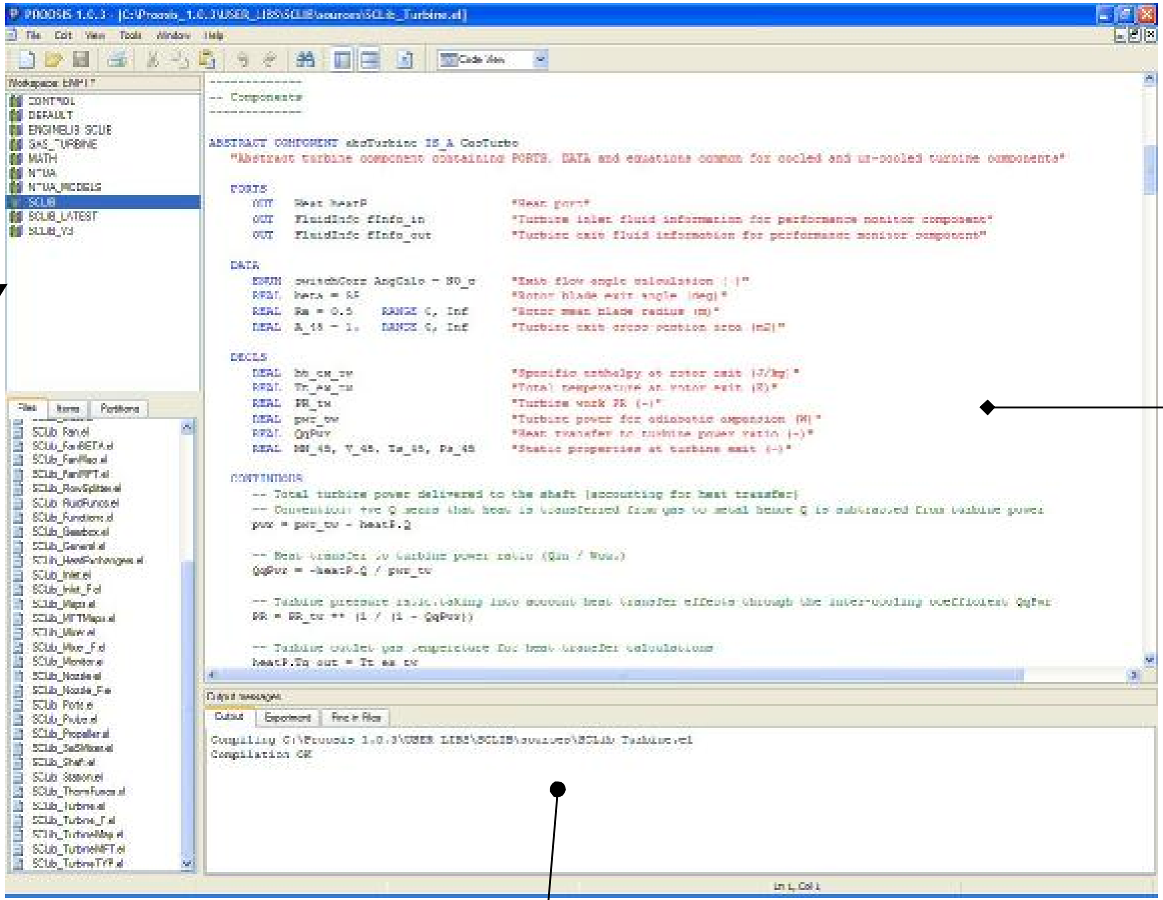
PROOSIS

Component Zooming: execution of higher order analysis code and integration of its results back in the 0-D engine cycle

Distributed Simulations: technologies that enable a simulation program to execute on a computing system containing multiple processors interconnected by a communication network

Contents

PROOSIS OVERVIEW: Code View



```
--------------
-- Components
--------------

ABSTRACT COMPONENT absTurbine IS_A GasTurbo
   "Abstract turbine component containing PORTS, DATA and equations common for cooled and un-cooled turbine components"

   PORTS
      OUT   Heat heatP                     "Heat port"
      OUT   FluidInfo fInfo_in             "Turbine inlet fluid information for performance monitor component"
      OUT   FluidInfo fInfo_out            "Turbine exit fluid information for performance monitor component"

   DATA
      ENUM  switchCorr AngCalc = NO_c      "Exit flow angle calculation [-]"
      REAL  beta = 55                      "Rotor blade exit angle (deg)"
      REAL  Rm = 0.5    RANGE 0, Inf       "Rotor mean blade radius (m)"
      REAL  A_45 = 1.   RANGE 0, Inf       "Turbine exit cross section area (m2)"

   DECLS
      REAL  hs_ex_tw                       "Specific enthalpy at rotor exit (J/kg)"
      REAL  Tt_ex_tw                       "Total temperature at rotor exit (K)"
      REAL  PR_tw                          "Turbine work PR (-)"
      REAL  pwr_tw                         "Turbine power for adiabatic expansion (W)"
      REAL  QqPwr                          "Heat transfer to turbine power ratio (-)"
      REAL  MN_45, V_45, Ts_45, Ps_45      "Static properties at turbine exit (-)"

   CONTINUOUS
      -- Total turbine power delivered to the shaft (accounting for heat transfer)
      -- Convention: +ve Q means that heat is transferred from gas to metal hence Q is subtracted from turbine power
      pwr = pwr_tw - heatP.Q

      -- Heat transfer to turbine power ratio (Qin / Wout)
      QqPwr = -heatP.Q / pwr_tw

      -- Turbine pressure ratio,taking into account heat transfer effects through the inter-cooling coefficient QqPwr
      PR = PR_tw ** (1 / (1 - QqPwr))

      -- Turbine outlet gas temperature for heat transfer calculations
      heatP.Tg_out = Tt_ex_tw
```

```
Compiling C:\Proosis_1.0.3\USER_LIBS\SCLIB\sources\SCLib_Turbine.el
Compilation OK
```

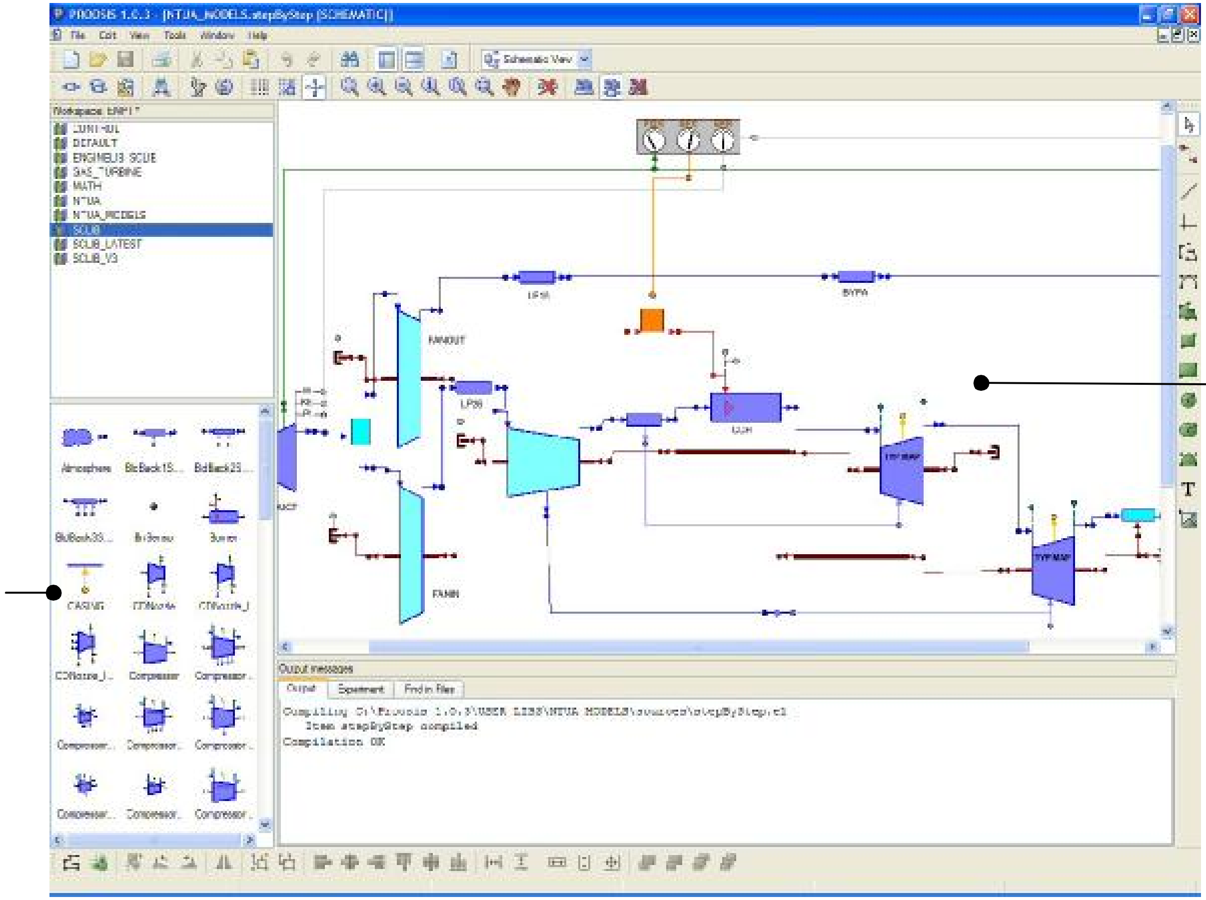
Component Libraries

Library EL Files

Component EL object-oriented code containing mathematical description of real component

Output Window

PROOSIS OVERVIEW: Schematic View



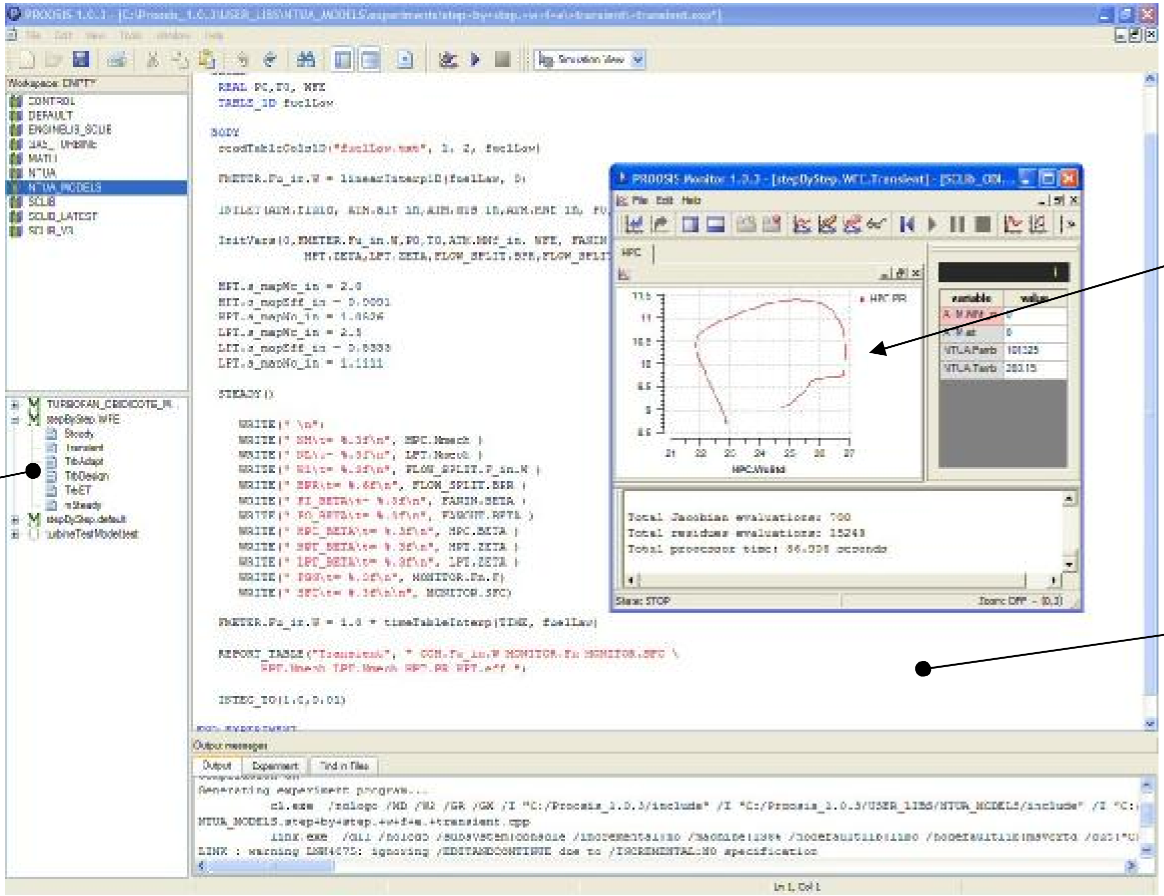
```
Compiling C:\Proosis 1.0.3\USER LIBS\NTUA MODELS\sources\stepByStep.el
    Item stepByStep compiled
Compilation OK
```

Library Palette

Drag-and-drop icons from palette to construct engine model.

Connect components through appropriate communication ports

PROOSIS OVERVIEW: Simulation View



Define Engine Mathematical Model and Simulation Cases

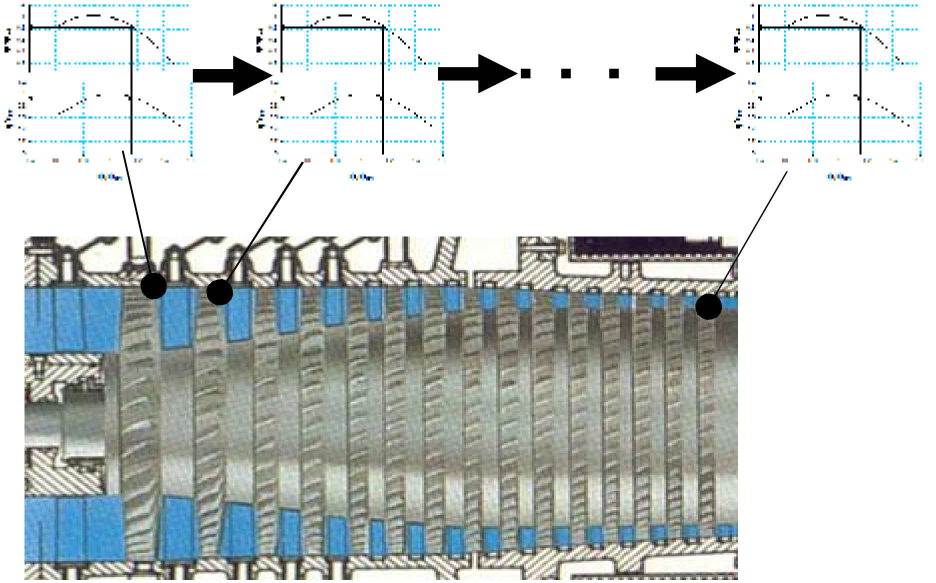
Graphical representation of results

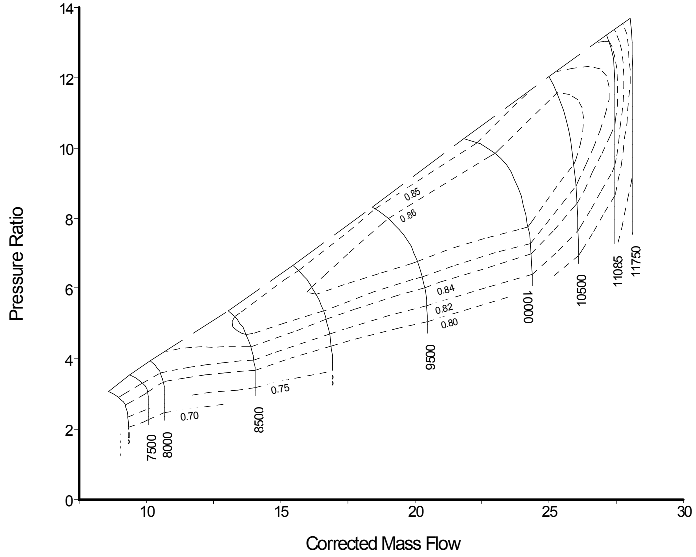
Describe in EL calculation mode or type

Contents

Compressor Stage-Stacking: Methodology

**Calculation of
individual stage exit properties from
dimensionless stage characteristics
and geometry data**





**'stack' stages together to evaluate
overall compressor performance**

Compressor Stage-Stacking: Implementation

FORTRAN:

```
SUBROUTINE stageStack (arguments)
```

compiled as static library (.lib)

PROOSIS:

```
“FORTRAN” FUNCTION stageStack (arguments)
      IN stageStack.lib
```

OR

C++ wrapper for FORTRAN subroutine:

```
extern "C" void __stdcall STAGESTACK (arguments);
          void stageStackClass::stageStack(arguments)
                    {STAGESTACK (arguments); }
```

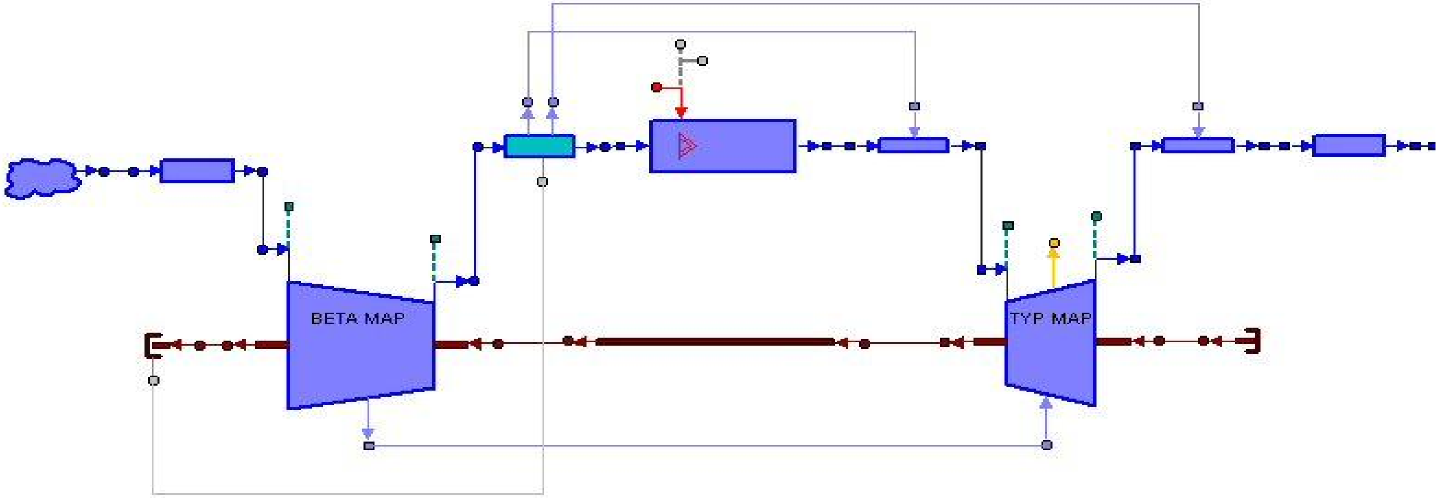
PROOSIS:

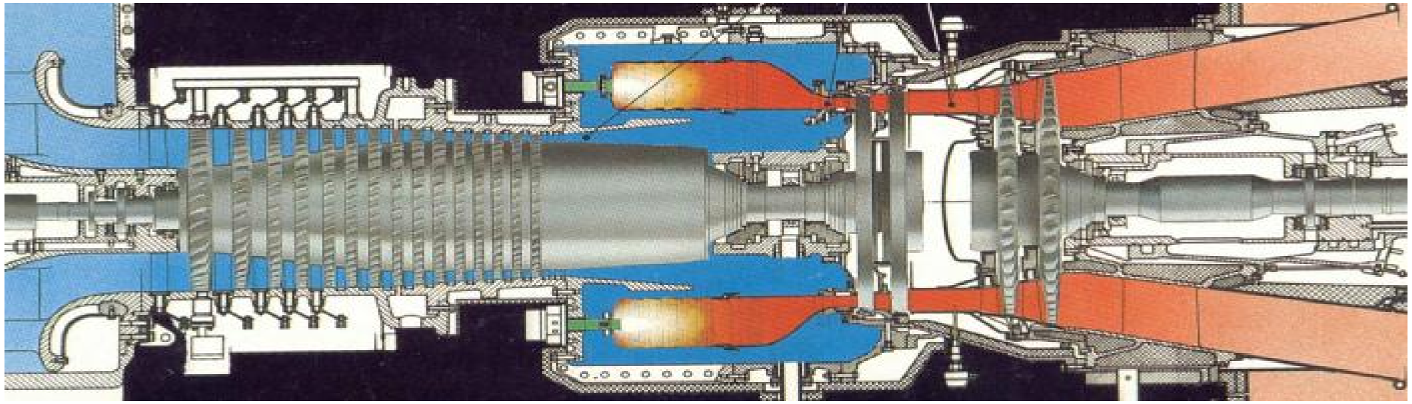
```
EXTERN CLASS stageStackClass
    METHODS
        EXTERN METHOD stageStack (arguments)
END CLASS
    INCLUDE  “stageStack.h”   IN “stageStack.lib”
```

Contents

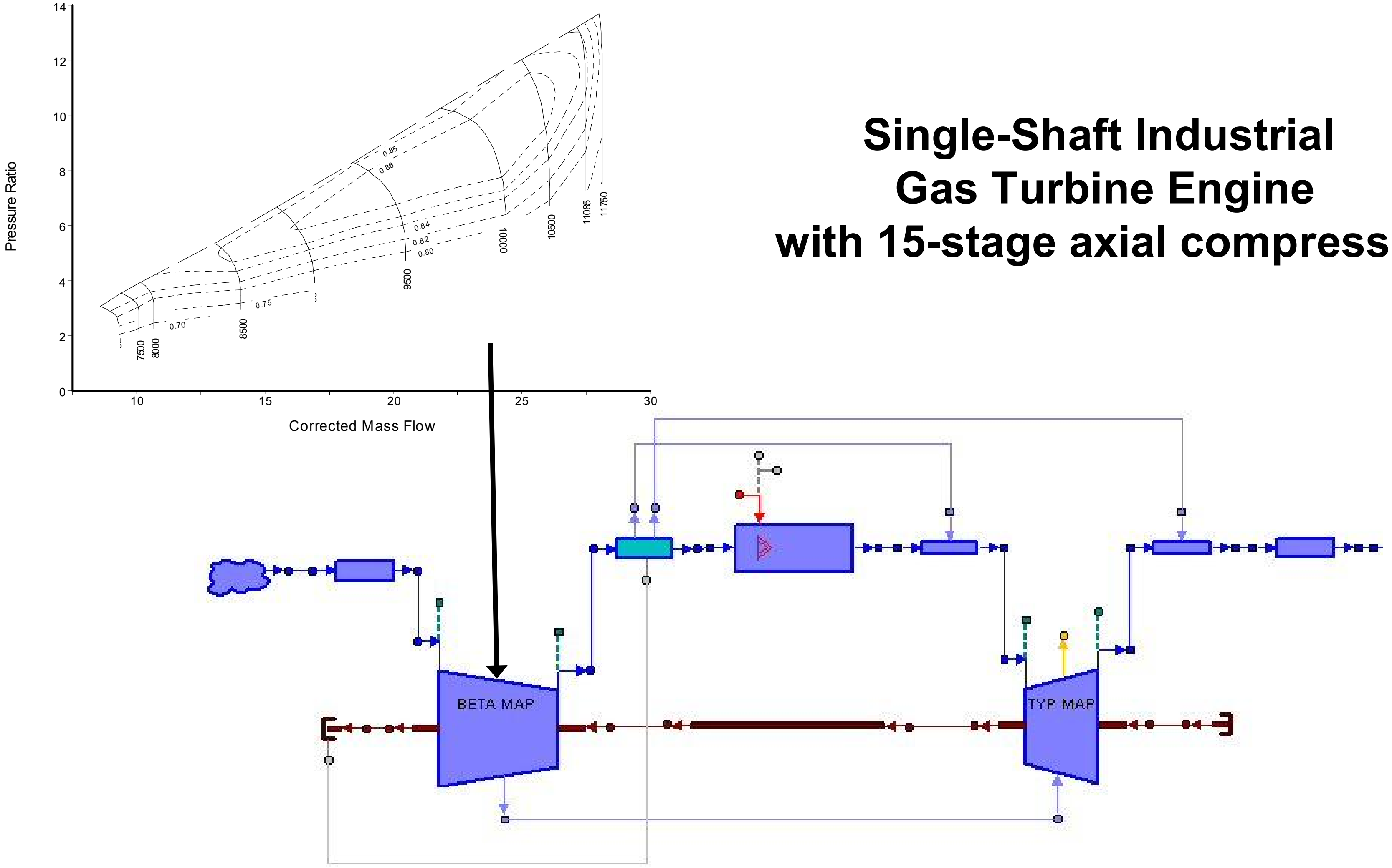
Engine Model

Single-Shaft Industrial Gas Turbine Engine with 15-stage axial compressor





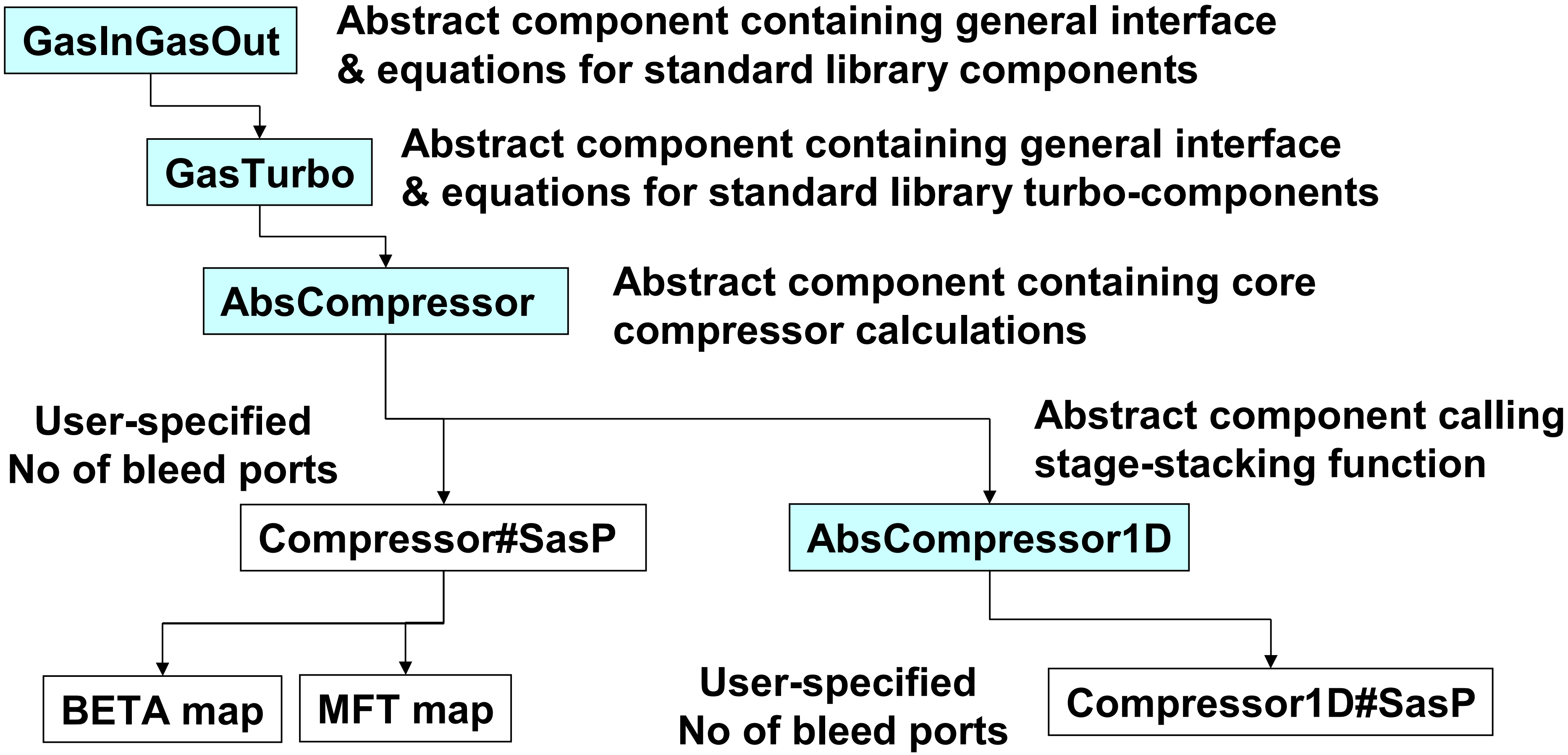
Engine Model



Single-Shaft Industrial Gas Turbine Engine with 15-stage axial compressor

Component Zooming: Compressor 1-D

Compressor 1-D Inheritance Tree



GasInGasOut

Abstract component containing general interface & equations for standard library components

GasTurbo

Abstract component containing general interface & equations for standard library turbo-components

AbsCompressor

Abstract component containing core compressor calculations

User-specified No of bleed ports

Compressor#SasP

Abstract component calling stage-stacking function

AbsCompressor1D

BETA map

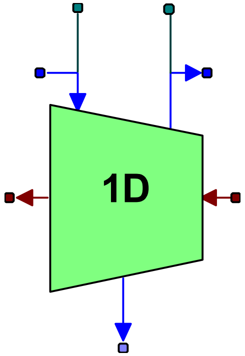
MFT map

User-specified No of bleed ports

Compressor1D#SasP

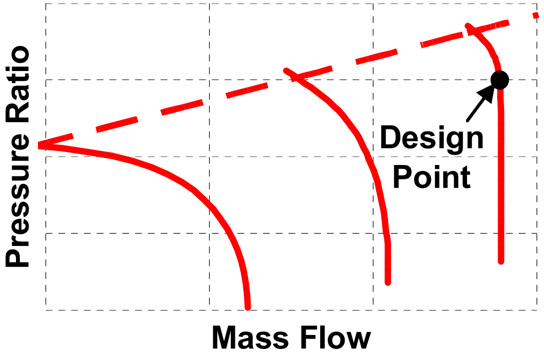
Component Zooming: De-coupled Approach (DCA)

Instantiate
Compressor 1-D

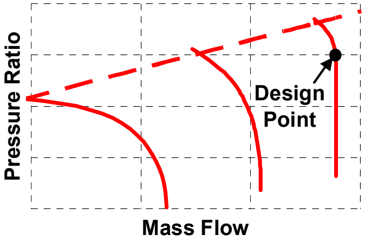


Run multi-point steady-state simulation

Produce PROOSIS compatible compressor 1-D map file

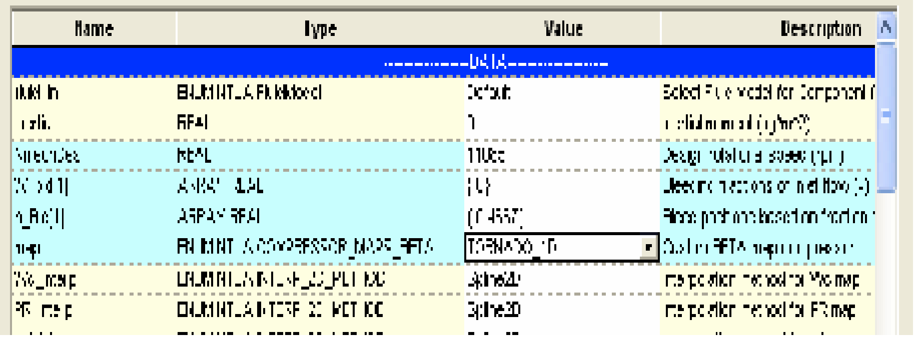


Component Zooming: De-coupled Approach (DCA)

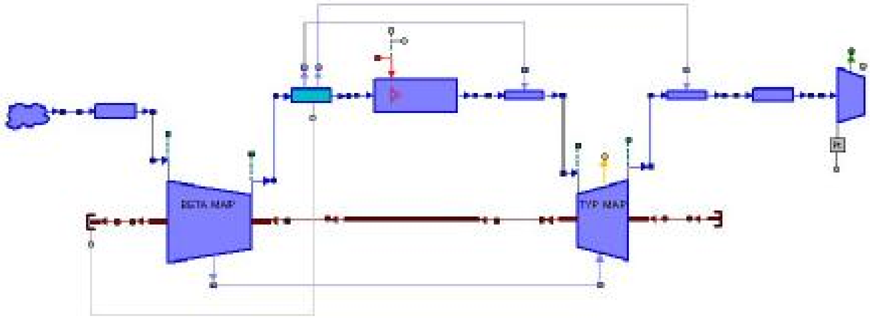


Select this map in compressor Editor

Run engine simulation using compressor 1-D map

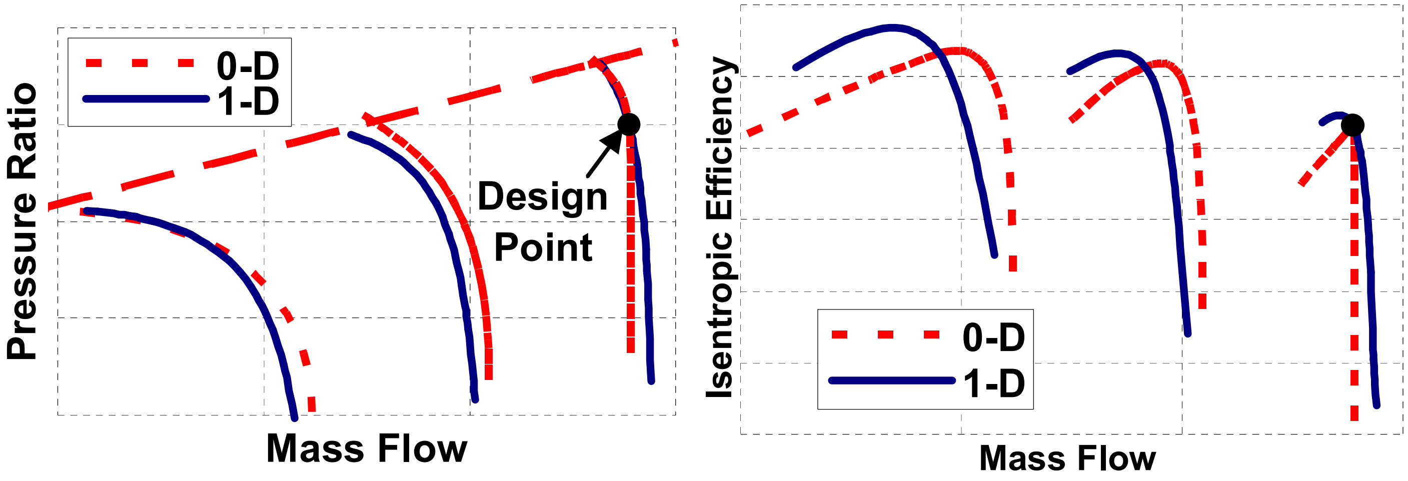


Name	Type	Value	Description
----------DATA----------			
[illegible]	[illegible]	Default	[illegible]
[illegible]	REAL	1	[illegible]
[illegible]	REAL	[illegible]	[illegible]
[illegible]	ARRAY REAL	[illegible]	[illegible]
[illegible]	ARRAY REAL	[illegible]	[illegible]
map	ENUMINT_A_COMPRESSOR_MAPS_BETA	[illegible]	[illegible]
[illegible]	[illegible]	Spline2D	interpolation method for [illegible] map
[illegible]	[illegible]	Spline2D	interpolation method for PR map

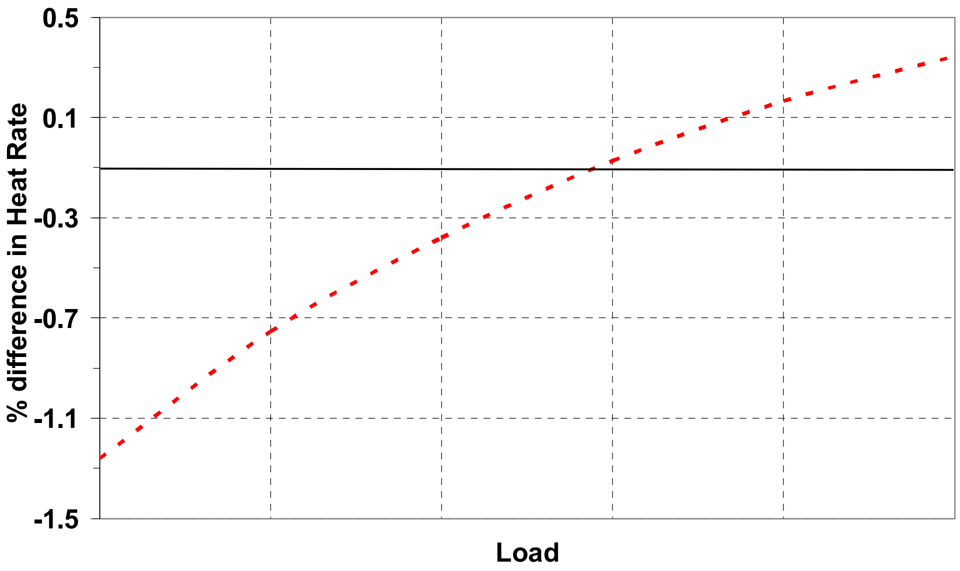


Component Zooming: Results (DCA)

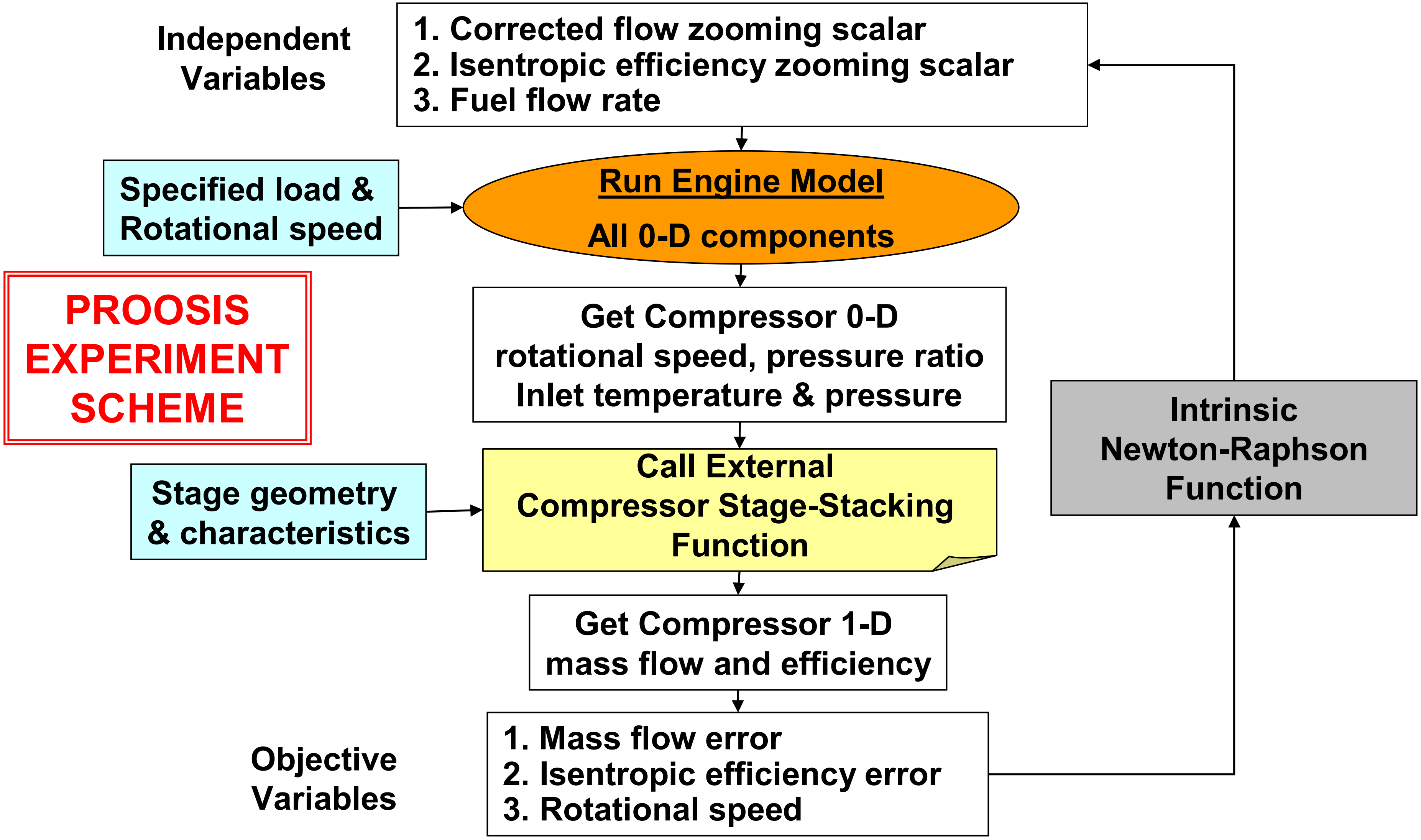
Comparison of 0-D & 1-D Maps



Effect of Compressor Zooming on Heat Rate vs Load Characteristic



Component Zooming: Semi-coupled Approach (SCA)



Independent Variables

1. Corrected flow zooming scalar
2. Isentropic efficiency zooming scalar
3. Fuel flow rate

Specified load & Rotational speed

Run Engine Model

All 0-D components

PROOSIS EXPERIMENT SCHEME

Get Compressor 0-D rotational speed, pressure ratio Inlet temperature & pressure

Stage geometry & characteristics

Call External Compressor Stage-Stacking Function

Get Compressor 1-D mass flow and efficiency

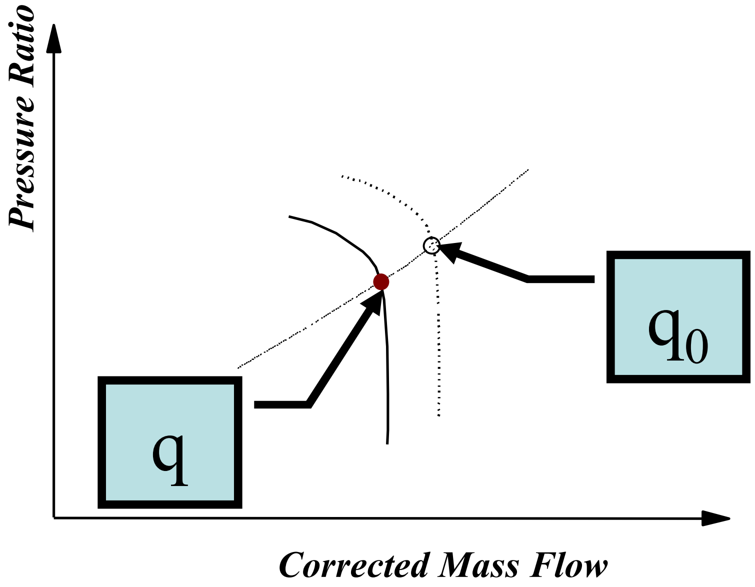
Intrinsic Newton-Raphson Function

Objective Variables

1. Mass flow error
2. Isentropic efficiency error
3. Rotational speed

The meaning of Modification factors

$$f_q = \frac{q}{q_0}$$



Transformation of component performance maps

Component Zooming: Semi-coupled Approach (SCA)

Structure of Adaptive models

Modification factors f_k for components

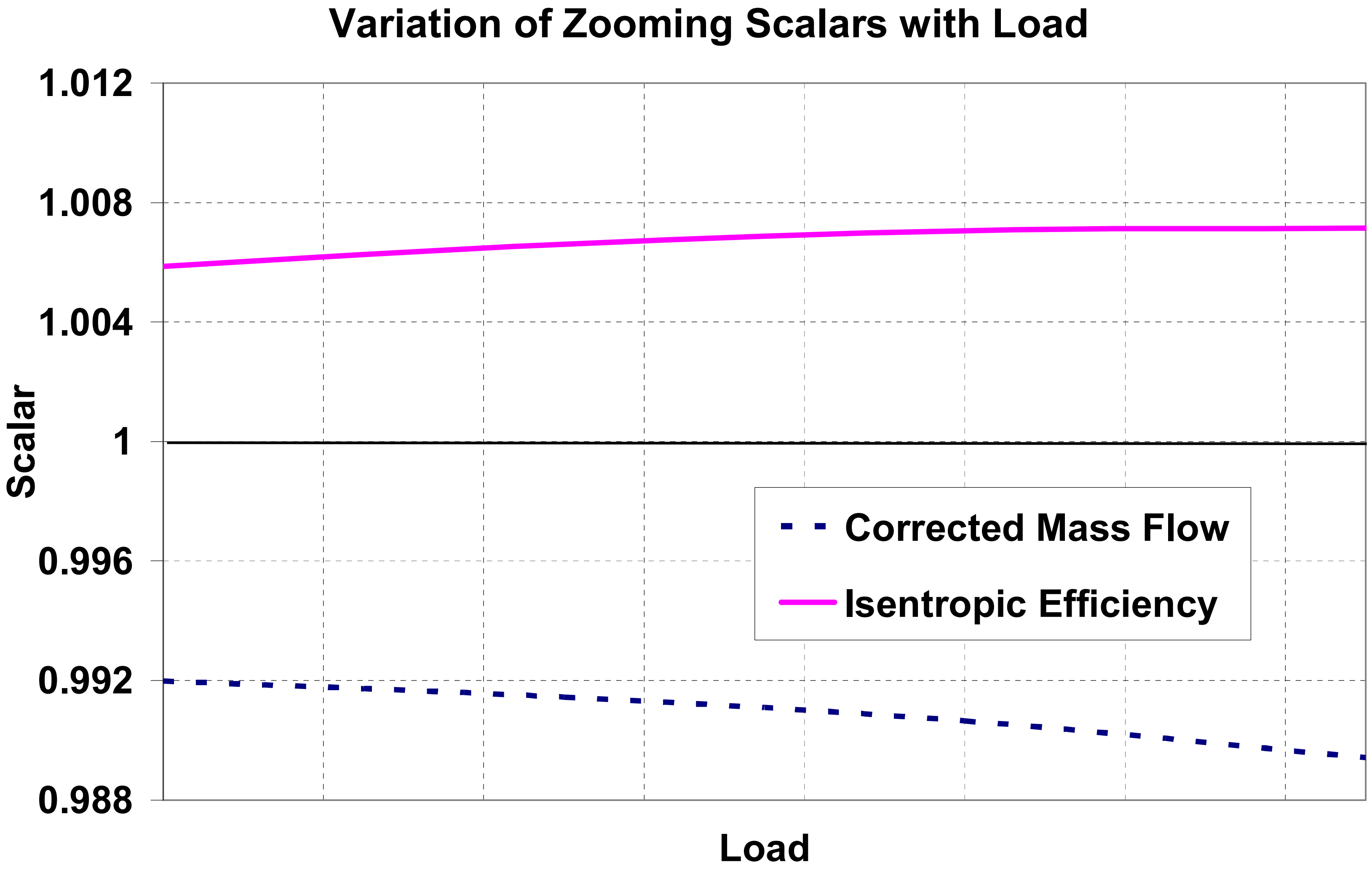
$x_{p,k}$: Actual value for parameter

$x_{p,ref,k}$: Reference value for parameter

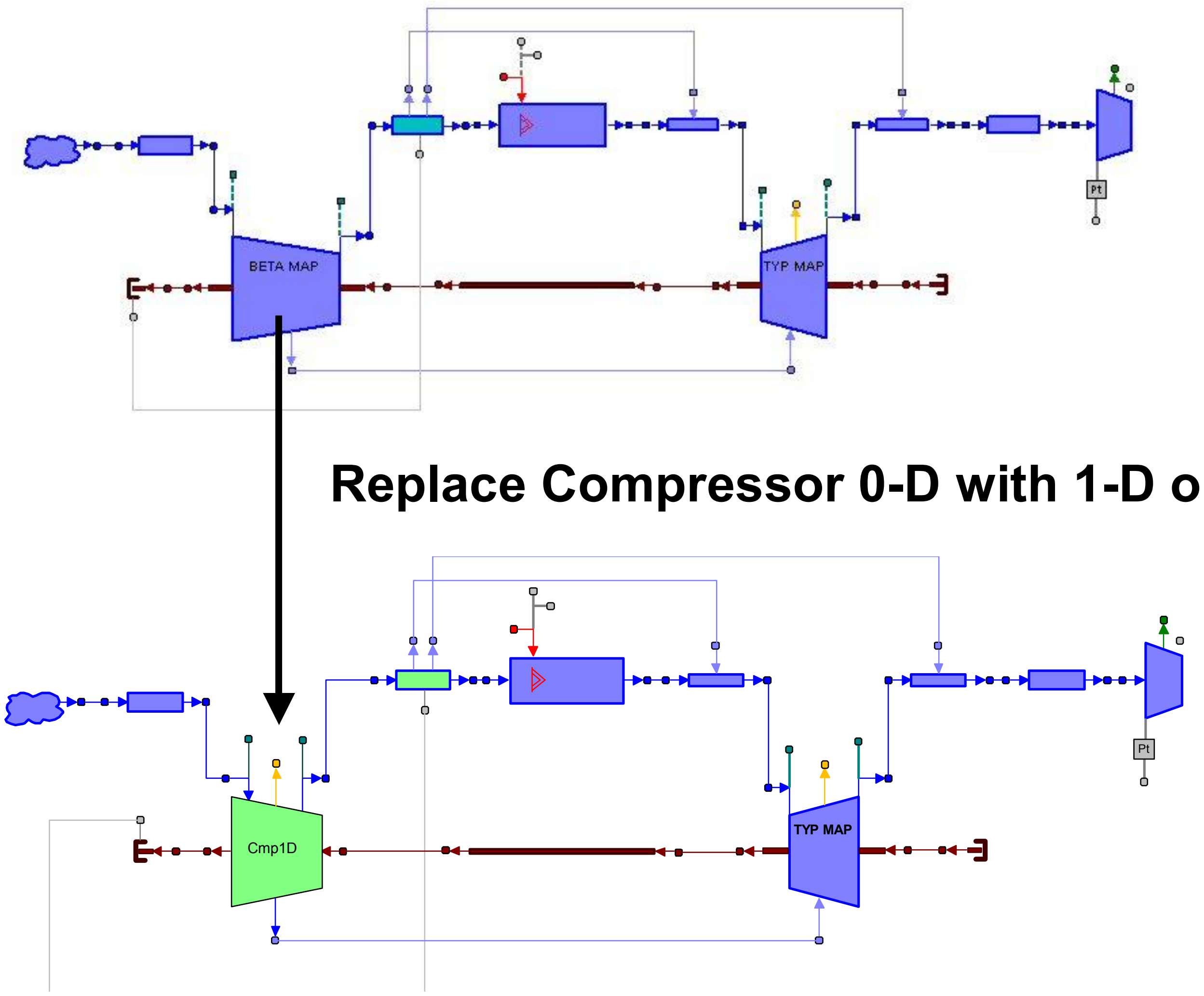
$$f_k = \frac{x_{p,k}}{x_{p,ref,k}}$$

Transformation of component performance maps

Component Zooming: Results (SCA)



Component Zooming: Fully-coupled Approach (FCA)



Replace Compressor 0-D with 1-D one

Component Zooming: Results (FCA)

Design Point Case
1.5% inter-stage bleed from 10th stage

PARAMETER	% DIFFERENCE
Fuel Flow Rate	0.289
Compressor Inlet Flow	0.111
Compressor Delivery Temperature	0.438
Compressor Pressure Ratio	0.211
Compressor Polytropic Efficiency	-0.238
Compressor Power	0.583

Contents

- ☐ PROOSIS OVERVIEW
- ☐ Compressor Stage-Stacking
- ☐ The Engine Model
- ☐ COMPONENT ZOOMING
  - o The ‘de-coupled’ Approach
  - o The ‘semi-coupled’ Approach
  - o The ‘fully-coupled’ Approach
- ☐ **DISTRIBUTED SIMULATIONS**
  - o **Implementing Distributed Simulations**
  - o **Prototype Development**
  - o **Future Developments**
- ☐ SUMMARY & CONCLUSIONS

Distributed Simulations: Rationale & Features

- Collaborative modelling among possibly geographically dispersed engineers
- Easy and efficient deployment of subsystem models
- Protection of ownership and IPR
- Reduction of simulation time through load distribution
- Size and complexity of the simulation model may grow irrespective of capability of computing infrastructure
- Reuse of submodels in different simulations

Distributed Simulations: Implementing DS

Technologies considered:

- **CORBA:** complex; did not catch up with growing Web developments and demands; high run-time costs; difficulties with security; versions & difficulties in backward compatibility; not supported by Microsoft...
- **DCOM:** serious security problems; did not catch up with growing Web developments and demands; deprecated in favour of .NET
- **Java RMI:** Java specific; being obscured by Web Service technology
- **XML and SOAP:** slower than e.g. CORBA and RMI but providing good basis for secure distributed web-based solutions in wide-area contexts
- **Web Services:** uses open standards and protocols (incl. SOAP and XML); commonly used nowadays to implement secure distributed solutions in SOA style; standards and tools are emerging

Web Service: state-of-the-art technology enabling software components (clients, servers) to communicate over a network using standard messages and formats

Distributed Simulations: Prototype Development (I)

Prototype in VIVACE context: PROOSIS with compressor stage stacking function, available as “User Library”, running on a remote computer

Compressor stage stacking function:

- developed by, and proprietary code of NTUA
- written in Fortran, available as a shared library (DLL) on Windows
- available to PROOSIS users at NTUA as a PROOSIS User Library (PROOSIS’ mechanism to include customer code in engine simulations)
- code may be used but cannot be installed outside NTUA

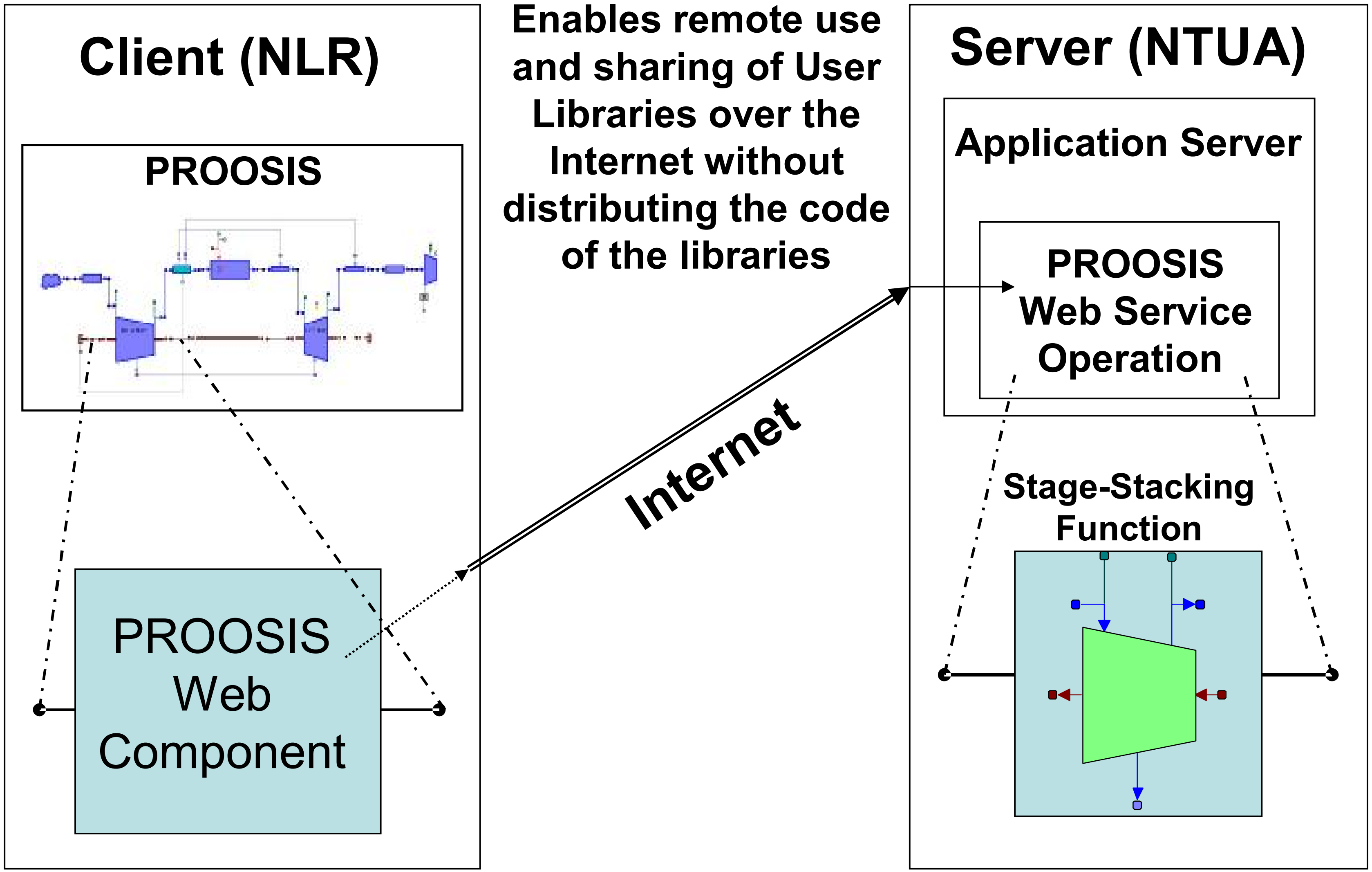
Distributed Simulations: Prototype Development (II)

Compressor Zooming via Remote Web Service Invocation between NLR & NTUA

PROOSIS simulation in The Netherlands

Stage Stacking Calculation in Greece

Distributed Simulations: Prototype Development (III)



Client (NLR)

PROOSIS

PROOSIS Web Component

Enables remote use and sharing of User Libraries over the Internet without distributing the code of the libraries

Internet

Server (NTUA)

Application Server

PROOSIS Web Service Operation

Stage-Stacking Function

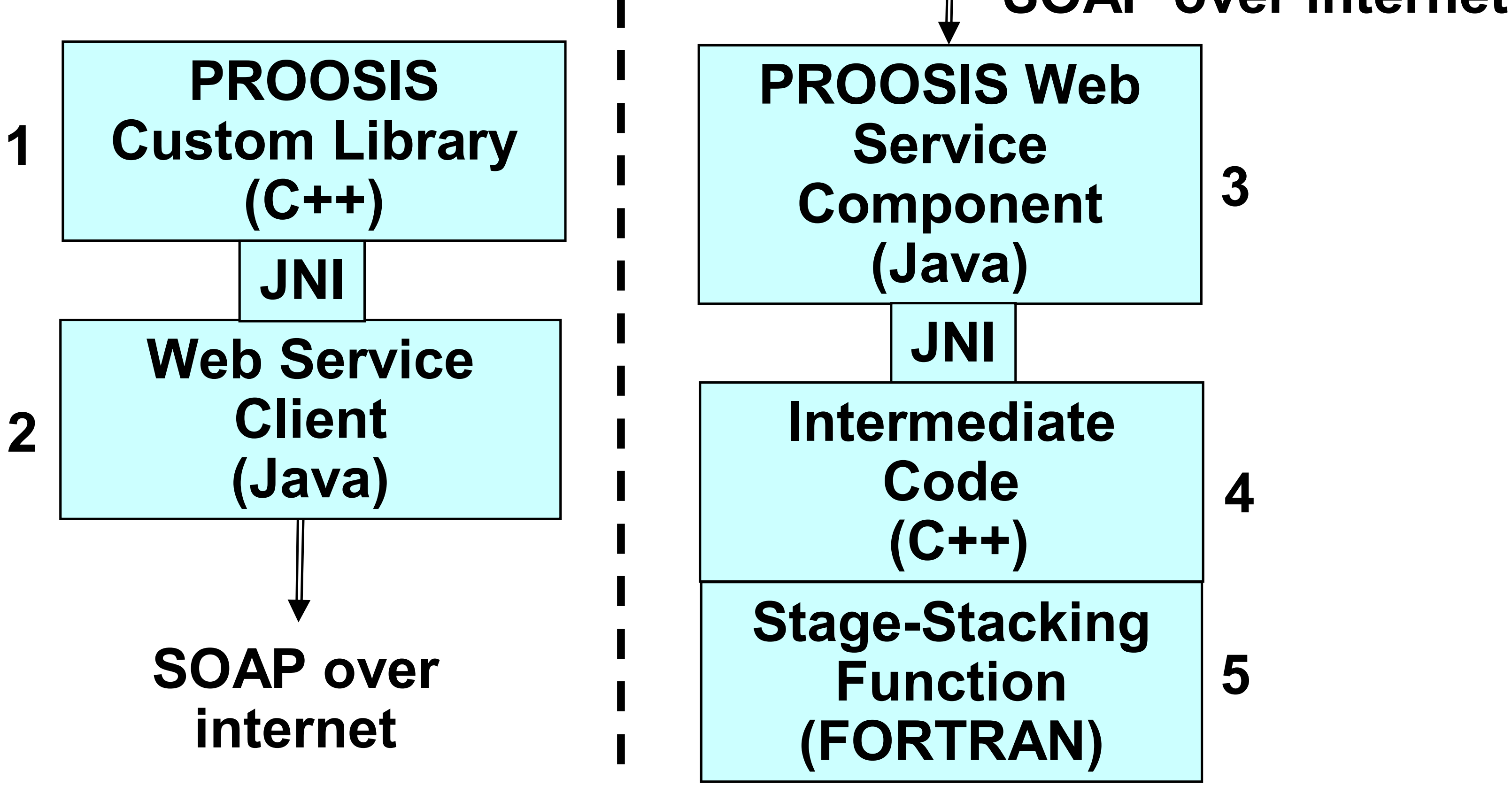
Distributed Simulations: Prototype Development (IV)

Layered Structure of Prototype

Client Side (NLR)

Server Side (NTUA)

SOAP over internet



1 **PROOSIS Custom Library (C++)**

JNI

2 **Web Service Client (Java)**

SOAP over internet

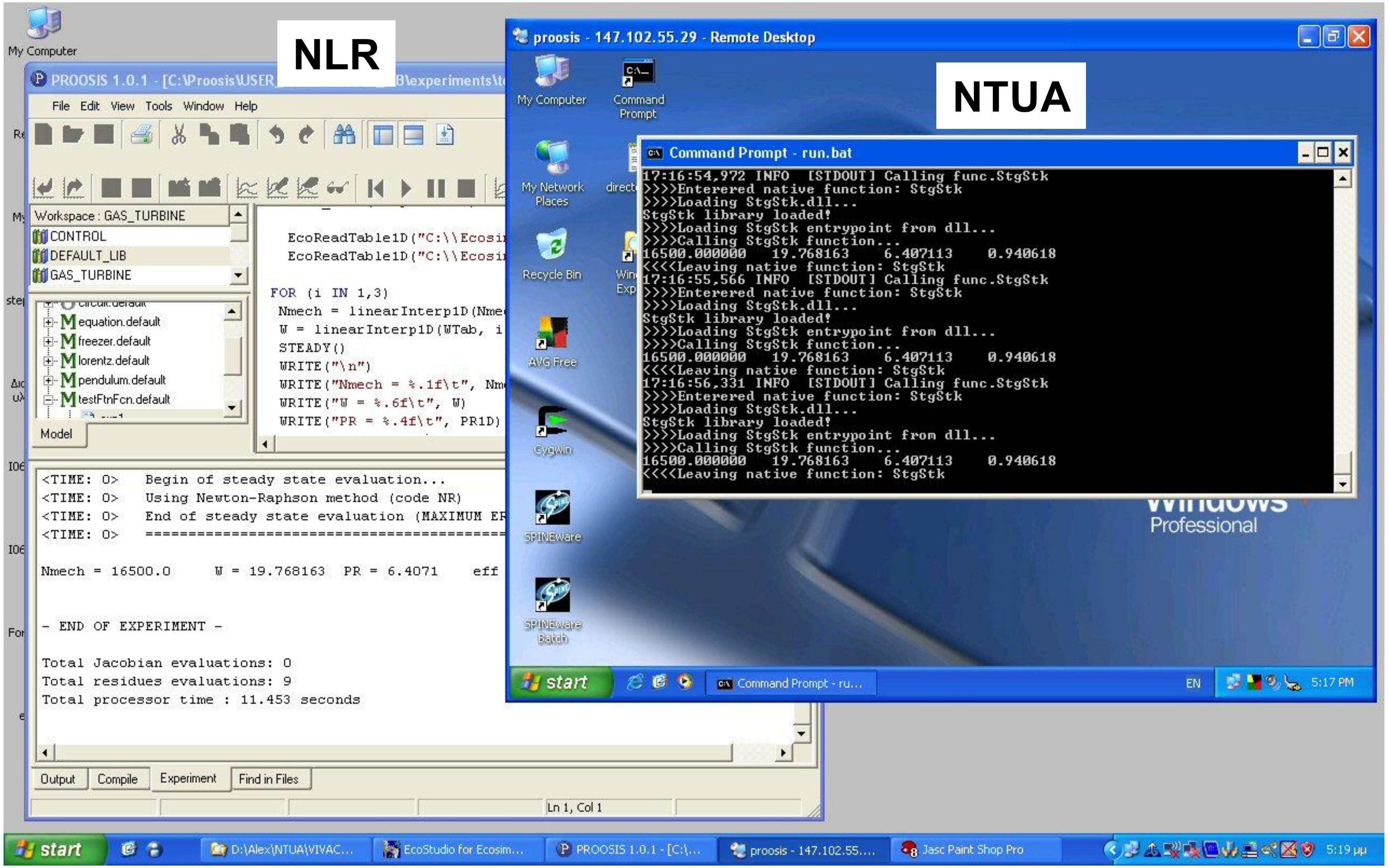
PROOSIS Web Service Component (Java) 3

JNI

Intermediate Code (C++) 4

Stage-Stacking Function (FORTRAN) 5

Distributed Simulations: Live Public Demo



```
  EcoReadTable1D("C:\\Ecosi
  EcoReadTable1D("C:\\Ecosi

FOR (i IN 1,3)
 Nmech = linearInterp1D(Nme
 W = linearInterp1D(WTab, i
 STEADY()
 WRITE("\n")
 WRITE("Nmech = %.1f\t", Nm
 WRITE("W = %.6f\t", W)
 WRITE("PR = %.4f\t", PR1D)
```

```
<TIME: 0>   Begin of steady state evaluation...
<TIME: 0>   Using Newton-Raphson method (code NR)
<TIME: 0>   End of steady state evaluation (MAXIMUM ER
<TIME: 0>   ==========================================

Nmech = 16500.0     W = 19.768163  PR = 6.4071    eff


- END OF EXPERIMENT -

Total Jacobian evaluations: 0
Total residues evaluations: 9
Total processor time : 11.453 seconds
```

```
17:16:54,972 INFO  [STDOUT] Calling func.StgStk
>>>>Enterered native function: StgStk
>>>>Loading StgStk.dll...
StgStk library loaded!
>>>>Loading StgStk entrypoint from dll...
>>>>Calling StgStk function...
16500.000000   19.768163    6.407113    0.940618
<<<<Leaving native function: StgStk
17:16:55,566 INFO  [STDOUT] Calling func.StgStk
>>>>Enterered native function: StgStk
>>>>Loading StgStk.dll...
StgStk library loaded!
>>>>Loading StgStk entrypoint from dll...
>>>>Calling StgStk function...
16500.000000   19.768163    6.407113    0.940618
<<<<Leaving native function: StgStk
17:16:56,331 INFO  [STDOUT] Calling func.StgStk
>>>>Enterered native function: StgStk
>>>>Loading StgStk.dll...
StgStk library loaded!
>>>>Loading StgStk entrypoint from dll...
>>>>Calling StgStk function...
16500.000000   19.768163    6.407113    0.940618
<<<<Leaving native function: StgStk
```

Distributed Simulations: Future Developments

- **Design of more generic (re-usable) interface**
  - ✓ multiple function implementations
  - ✓ not all layers need to be modified
  - ✗ additional overhead and delays in communication
- **Reduction of overhead caused by conversions & data transfers**
  - ✓ use pure C++ development environment
  - ✗ C++ support for Web Services limited/unstable
  - ✗ Java platform allows integration with other collaborative tools
- **Reduction of overhead in DLL loading and unloading**
  - ✓ load DLLs once and dispose after final calculation
- **Multi-user and security**
  - ✓ Use Web Service Security specification
  - ✓ allow multi-user access

Contents

- ☐ PROOSIS OVERVIEW
- ☐ Compressor Stage-Stacking
- ☐ The Engine Model
- ☐ COMPONENT ZOOMING
  - o The 'de-coupled' Approach
  - o The 'semi-coupled' Approach
  - o The 'fully-coupled' Approach
- ☐ DISTRIBUTED SIMULATIONS
  - o Implementing Distributed Simulations
  - o Prototype Development
  - o Future Developments
- ☐ **SUMMARY & CONCLUSIONS**

Summary & Conclusion (I)

- PROOSIS is a standalone, multi-platform, object-oriented simulation environment for gas turbine engine performance simulations. It can be used to create, run, manage and share engine models using either the standard or custom libraries of engine components. The feasibility of performing multi-fidelity and distributed simulations with PROOSIS was demonstrated in this paper.

- Using the model of an industrial gas turbine engine and a 1-D compressor stage stacking code as an example, different implementations for integrating high fidelity component analysis in overall engine simulations were presented. The tool’s flexible and extensible architecture gives the user the freedom to select the most suitable approach for a particular simulation case.

Summary & Conclusion (II)

- The stage stacking code is also used to demonstrate distributed simulations. A prototype of a Web Component has been created and successfully tested that remotely invokes the code from an engine simulation, via the internet, using Web Services technology.

- These demonstrations prove that the tool’s architecture is adaptable enough to integrate different modelling methods and its potential to fulfil its role as a shared simulation environment.