PROVIDING WORLD-WIDE INTRANET ACCESS TO PRODUCT LIFETIME CALCULATIONS USING MATLAB PRODUCTION SERVER

Dirk Twisk
Bosch Transmission Technology
Using Matlab production server for product lifetime calculations

Contents

1. Bosch Transmission Technology
2. The CVT
3. Modelling
4. Implementation with Matlab
5. Server based calculations
6. Matlab Production Server solution
7. Next steps
Engineering calculations at Bosch Transmission Technology BTT

- All engineers should use the same tools.
  - Many tools are developed in house – what is the latest version?

- Quality procedures are used to develop and verify software tools.
  - Use versioning systems and (unit) tests.

- Everybody uses the same data
  - Multiple channels of communication internally and externally.

- Data can be shared between applications.

- Effective use of computational resources.
BTT Locations

- Tilburg, Netherlands
- Schwieberdingen, Germany
- China, Shanghai
- Yokohama, Japan
- Ho Chi Minh, Vietnam
- San Luis Potosi, Mexico

Production

Engineering
Introduction to the CVT
Model calculations for Push Belts

- Designs are made by engineers, based on user requirements:
  - Maximum torque and power, expected lifetime (kms), ratio coverage, package size.

- Detailed calculation results are shared with the customer.
  - Bosch designs and produces the push-belt only.
  - All other transmissions components are made by the customer.
  - Integration engineering.

- Calculations are made during all stages of the development process.
  - Initial offering to customer
  - Product Development.
  - Verification in Laboratory.
Engineering calculations

- **Geometrical calculations**
  - Achieve a large Ratio Coverage.
    - Deep LOW is for take-off performance
    - OD is required for fuel efficiency
  - Determine build size. Smaller and less weight is better.

- **Forces and Stresses**
  - Calculate forces and stresses acting on the system.
  - Calculate expected lifetime of the push belt
    - Number of stress cycles before most critical part fails.
Situation for Overdrive

Element

± 400 elements

Pulley

Loops
Force balance model

Loop set part

Element string part

Pulley sheave part
Solving the problem in Matlab

```matlab
[R1,R2,exitflag] = fsolve(@(SolvVec) FuncName(SolvVec,ConstVec),InitVec,options);
```

---

```
function [F] = Calc_Solution_Above_Transition(x,y)
    fdmax = x(1);
    fstrek = x(2);
    beta_s = x(3);
    faxs = y(1);
    F = Fdsec(alphas,fdmax,fstrek,b_s) + FaxsA(fdmax,fstrek,b_s) - FaxsB(fdmax,fstrek,b_s) + Msccf(fdmax,fstrek) - Ms;
end

function [F] = Calc_Solution_Below_Transition(x,y)
    fdmax = x(1);
    fstrek = x(2);
    beta_s = x(3);
    faxs = y(1);
    F = Fdsec(alphas,0,0,0) - fdmax + FaxsA(0,0,0) - FaxsB(0,0,0) - Msccf(fdmax,fstrek) - Ms;
end
```

---

```
function [F] = Fdsec(phi,fdmax,fstrek,b_s)
    model = BOVEN;
    if (model == BOVEN)
        F = (fdmax+Csch-(fstrek-Csn)*exp(-mu_lr*(alphas-b_s)))*exp(con10b*(phi-(alphas-b_s)))+Msecf(fdmax,fstrek)*exp(-mu_lr*phi)-Csch;
    else
        F = (Csch-(fstrek-Csn)*exp(-mu_lr*(alphas-b_s)))*exp(con10b*(phi-(alphas-b_s)))+Msecf(fdmax,fstrek)*exp(-mu_lr*phi)-Csch;
    end
end
```

---

```
function [F] = FaxsA(fdmax,fstrek,b_s)
    con20 = 0.5*(1-tan(lambda_sec)*mu_lf*sin(gammaf)-tan(lambda_sec)+mu_lf*sin(gammaf));
    if model == BOVEN
        F = con20*(1-exp(-mu_lr*(alphas-b_s)))*exp(con10b*b_s-1);  % equation (49)
    else
        F = con20*exp(-mu_lr*(alphas-b_s));  % equation (50)
    end
end
```

---

```
function [F] = FaxsB(fdmax,fstrek,b_s)
    con20 = 0.5*(1-tan(lambda_sec)*mu_lf*sin(gammaf)-tan(lambda_sec)+mu_lf*sin(gammaf));
    if model == BOVEN
        F = con20*(1-exp(-mu_lr*(alphas-b_s)))*exp(con10b*b_s-1);  % equation (49)
    else
        F = con20*exp(-mu_lr*(alphas-b_s));  % equation (50)
    end
end
```
How to Implement

BTT - Tilburg
- Mechanical Engineering
- Transmission Design (CVT)
- Modelling
- Matlab

RBEI - Bangalore
- Software Systems Design
- Databases
- Java
- Matlab
System Design

- Server stores inputs and outputs for several applications
- Start with complete variator models: BCP and VBL
- Data can be accessed by all users (engineers)
- Different interfaces and business logic for different applications.
System Implementation

Front End/Business | Java
--- | ---
Database | SQL

Calculations | Matlab
--- | ---
Reporting | Report Server

© Robert Bosch GmbH / Bosch Transmission Technology B.V. 2018. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Development – traditional design

**PS-CT/EAC**
Product Development
Model Development

1. Write Code
2. Test Code
3. Compile Code
4. Jar File(s)

**RBEI/ETC**
Software Development
Information Systems

1. Write Code
2. Test Code
3. Compile Code
4. Create Executable

**PS-CT/ICT**
Maintain computer systems
Install Software

1. Upload to Software Server.
2. Install JAR on users’ PC.
3. [Install Matlab runtime].
Local installation

Local install servers

Tilburg
Netherlands

Netherlands

Vietnam

Japan
Drawbacks of the traditional design

Complicated release process:

- Updates not synchronized due to the use of different install servers.
- Every user needs to have (correct) Matlab runtime installed.
- High maintenance costs (technical support).
- Sending data over the network is slow, so the performance is not acceptable.
- Changes to the Matlab code requires rebuild of the complete code.

After the second major release, it was decided to change the design

Switch to a web-based approach
Web-based Approach

Web-server

http://BCP

Netherlands

Vietnam

Japan
Improved Solution

▶ Make the application web-based:
   ▶ All required components run on a single server.
   ▶ Only inputs and outputs are sent over the network.

▶ Users do not have to install any software on local computers
   ▶ The database itself is used to store user data and grant access to users.
   ▶ Use NT login credentials

▶ Updating software is required at only 1 location
   ▶ Synchronized updates of software and database.
   ▶ Minimal disruption of service.
   ▶ Maximum of 1 hour downtime for major updates.
Architecture

Tomcat → BCP Java applet

Matlab → Microsoft SQL Server
Initial Implementation

Server (machine)
- SQL databases
- BCP production
- Testing
- Training

Tilburg - NL

Server (machine)
- VM (4 cores)
- Tomcat
- BCP
- App 2
- App 3

Tilburg - NL

network

only 1 process!
Limitations ...

Calculation Times:
Shortest : 2 seconds
Longest  : 24 hours
Infrastructure with Production Server – Final Design

Server (machine)
- SQL databases
  - BCP production
  - Testing
  - Training

Server (machine)
- VM (4+ cores)
  - Tomcat

BCP production

Tilburg - NL

Worker
Worker
Worker

Network

Tilburg - NL

Internal | PS-CT/EAC3 | 2018-03-05
© Robert Bosch GmbH / Bosch Transmission Technology B.V. 2018. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Main Benefits of MPS

- Multiple Matlab jobs run simultaneously:
  - Each up to 24 hours per job (vehicle duty cycles).

- Scalable:
  - Add more cores to the server.
  - Assign different machines for different tasks.
  - Utilize available (unused) computational resources.

- Decoupled updating of main BCP code and Matlab code.
  - Define inputs/outputs between systems.
  - Independent updates of Java and Matlab code.
  - Log output is used to verify communication between the components.
Next Steps

Three developments:

1. Make existing standard programs web-based:
   - Model-Viewer-Controller type applications.

2. Create a library of common functions, allow calling from:
   - Java (web)
   - Excel
   - Matlab

3. Provide functionality to non-Matlab departments
   - Production, Quality, Inspection, ...