MATLAB EXPO 2019

Deploying AI for Near Real-Time Manufacturing Decisions
(Masterclass)

Pallavi Kar
Application Engineer – Data Science & Enterprise Integration
Digital Transformation and IIoT

Customer Goal

By connecting machines in operation, you can use data, algorithms, and models to make better decisions, improve processes, reduce cost, improve customer experience.

- Industrial IoT
- Digital Twin
- Industry 4.0
- Smart ‘XYZ’
- Digital Transformation
Predictive Maintenance

- Operating Conditions vary over time and location
- Component Life and Safety
Transocean performed CPM of a BOP using an adaptive physics-based modeling approach with Simscape.
Case Study: Transocean

**Objective:** Reduce BOP downtime

**Solution:**
- Simulink model of BOP and Control System
- Simulate 100s future scenarios - degradation trends and anomalies
- Pi Servers to collect data
- Preprocess data to avoid noise and outliers
- Train Models on future scenarios to predict in advance

**Outcome:** Robust condition and performance monitoring of BOP reduced downtime
**Example Problem**: Develop and operationalize a digital twin and a machine learning model to predict failures in industrial pumps.

Current system requires Operator to manually monitor operational metrics for anomalies. Their expertise is required to detect and take preventative action.

**Process Engineer**: Develops models in MATLAB and Simulink.

**System Architect**: Deploys and operationalizes model on Azure cloud.

**Operator**: Makes operational decisions based on model output.
Predictive Maintenance Workflow

1. Access and Explore Data
   - Files
   - Databases
   - Sensors

2. Preprocess Data
   - Working with Messy Data
   - Data Reduction/Transformation
   - Feature Extraction

3. Develop Predictive Models
   - Model Creation e.g. Machine Learning
   - Parameter Optimization
   - Model Validation

4. Integrate with Production Systems
   - Desktop Apps
   - Enterprise Scale Systems
   - Embedded Devices and Hardware
   - AWS Kinesis
   - kafka

5. Visualize Results
   - 3rd party dashboards
   - Web apps
Backbone Infrastructure for Preventive, Predictive, Reactive, Actionable Analytics

Smart assets → Edge systems → OT Infrastructure → IT Systems

- Data Ingestion → Local Communications
- Long-Range Communications → Edge Management
- Integration

Value of data to decision making

- Hard real-time control
- Real-time decisions
- Time-sensitive decisions
- Big Data processing on historical data

Model-Based Design with MATLAB & Simulink, code generation → Edge Processing Model-Based Design, code generation → Stream Processing with MATLAB Production Server → Hadoop/Spark integration with MDCS, Compiler

Speed

- Milliseconds
- Seconds
- Minutes
- Hours
- Days
- Months

Scope

- MODBUS TCP/IP
- C/C++
- MQTT
- docker
- kafka
- Kinesis
- Azure
- Hadoop/Spark
- Amazon Web Services
Steaming Analytics - Remaining Useful Life

Edge Device Publishing Data

Consume data and Update RUL
**Project statement**: Develop end-to-end predictive maintenance system

1. Monitor *flow, pressure, and current* of each pump so I always know their *operational state*

2. Need an *alert* when fault parameters drift outside an acceptable range so I can take *immediate corrective action*

3. Continuous estimate of each pump’s *remaining useful life (RUL)* so that I can *schedule maintenance or replace* the asset
Project constraints and solutions

We don’t have a large set of failure data, and it’s too costly to generate real failures in our plant for this project.

**Solution**: Use an accurate physics-based software model for the pump to develop synthetic training sets.
Project constraints and solutions

Need software for multidisciplinary problem across teams, plus integration w/ IT

Solution: Use MATLAB and integrate with OSS
Project constraints and solutions

We don’t have a large IT/hardware budget, and we need to see results before committing to a particular platform or technology.

**Solution:** Leverage cloud platform to quickly configure and provision the services needed to build the solution, while minimizing lock-in to a particular provider.
- Crankshaft drives three plungers
  - Each 120 degrees out of phase
  - One chamber always discharging
  - Three types of failures
Creating Multi-Domain Physical Models using Simscape

Monitor  Analyze  Predict  Control  Optimize

Pump Hardware
Acquire Real-Time Data for Updating Digital Twin

Monitor, Analyze, Predict, Control, Optimize

**Digital Twin**

**MODBUS TCPIP**

```
m = modbus('tcpip', '192.168.2.1', 308)
m =

Modbus TCPIP with properties:
   DeviceAddress: '192.168.2.1'
   Port: 308
   Status: 'open'
   NumRetries: 1
   Timeout: 10 (seconds)
   ByteOrder: 'big-endian'
   WordOrder: 'big-endian'
```
Use Simulink Design Optimizer to

Monitor  Analyze  Predict  Control  Optimize

✓ Setup Experiments
✓ Parameterize
✓ Save Sessions
✓ Generate Code
Build digital twin and generate sensor data
Access and Explore Data

Simulate data with many failure conditions

Process Engineer

Cluster

Workers

Simulation 1

Simulation 2

Desktop System

Workers

Run parallel simulations

Store data on HDFS

ens = simulationEnsembleDatastore(location)

ens = simulationEnsembleDatastore with properties:

DataVariables: [25x1 string]
IndependentVariables: [0x0 string]
ConditionVariables: [0x0 string]
SelectedVariables: [25x1 string]
ReadSize: 1
NumMembers: 702
LastMemberRead: [0x0 string]
Files: [702x1 string]
Predictive Maintenance Workflow

1. Access and Explore Data
   - Files
   - Databases
   - Sensors

2. Preprocess Data
   - Working with Messy Data
   - Data Reduction/Transformation
   - Feature Extraction

3. Develop Predictive Models
   - Model Creation e.g. Machine Learning
   - Parameter Optimization
   - Model Validation

4. Integrate with Production Systems
   - Desktop Apps
   - Enterprise Scale Systems
   - Embedded Devices and Hardware

5. Visualize Results
   - 3rd party dashboards
   - Web apps
Represent signal information

Signal processing

```
[Spectrum,Frequencies] = psppectrum(data.Flow);
[pLow,pHigh] = bounds(Spectrum);
fPeak = Frequencies(Spectrum==pHigh);
qPeak2Peak = peak2peak(data.Flow);
qCrest = peak2rms(data.Flow);
qRMS = rms(data.Flow);
qMAD = mad(data.Flow);
```
Signal Trace plot for "flow/Data" is in focus.
Video showing App in action
Diagnostic Feature Designer App
Predictive Maintenance Toolbox R2019a

- Extract, visualize, and rank features from sensor data
- Use both statistical and dynamic modeling methods
- Work with out-of-memory data
- Explore and discover techniques without writing MATLAB code
Develop Predictive Models in MATLAB

3. Develop Predictive Models

Type of Fault (Classification)
Remaining Useful Life (Regression)

Fault Classification

Estimated Remaining Useful Life ~ 18 hrs

Real Data

Failure Threshold

Forecast Data
Develop Predictive Models in MATLAB

1. Represent Signals
2. Label Faults
3. Scale
4. Train Model
5. Validate Model

```
% Develop Predictive Models in MATLAB
% Process Engineer

% Represent Signals
% Label Faults
% Scale
% Train Model

% Evaluate tall expression using the Spark Cluster:
- Pass 1 of 2: Completed in 11 sec
- Pass 2 of 2: Completed in 2.3333 min
Evaluation completed in 2.6167 min
```
Develop Machine Learning Models

Develop Predictive Models
Develop Predictive Models

Process Engineer

Estimate Remaining Useful Life

Model Coeff: $\phi = 2.1396 \; \theta = -0.038836 \; \beta = 0.13184$

\[ S(t) = \phi + \theta(t) e^{(\beta(t)t + \epsilon(t) - \sigma^2/2)} \]
Develop a Stream Processing Function

**Batch Processing**: Build and test model on simulated data

- Historical Data
- Train Model
- Scale Up
- Predictions
  - Storage
  - Dashboard

**Continuous Data**

- Messaging Service
- Pump Sensor Data

**Stream Processing**: Apply model to sensor data in near real-time

- f(x)
- Update State
- Make Decisions
  - Alerts
  - Storage
Develop a Stream Processing Function

Streaming Function

function new_state = streamingFunction(data, old_state)

Preprocess signals
[data, features] = preprocessData(data);

Predict faults
[Leak, Blocking, Bearing] = predictFaultValues(features);
FaultType = predictFault(features);
[RUL, Model] = predictUpdateRUL(data.TimeStamp, data.Flow, 500);

Update state
new_state = updateState(data, old_state);

Write results
writeResults(Leak, Blocking, Bearing, FaultType, RUL, Model)
end

Process each window of data as it arrives

Previous state

Current window of data to be processed
Prototype Predictive Maintenance Architecture on Azure

Edge
- Generate telemetry
- Wind turbine
- Car
- Gas pump
- Aircraft

Production System
- MATLAB Production Server
  - Worker processes
  - Request Broker
- Apache Kafka
  - Connector
  - State Persistence
- elastic storage layer

Analytics Development
- MATLAB
  - Compiler SDK
  - Debug
  - Package & Deploy
  - Model
- Algorithm Developers

Business Decisions
- Kibana
  - Presentation Layer
- End Users

MATLAB Production Server
- System Architect

Flowchart:
- Edge to Production System
- Production System to Analytics Development
- Analytics Development to Business Decisions
What does a streaming function look like?

```matlab
function pumpconsume(msg)

% Persistence service cache name must be unique to this group ID.
msg.metaInfo.cacheName = make_cache_name(msg.metaInfo.groupId);

% Application publishes results to a Kafka topic. The topic name
% uses the group ID to allow multiple simultaneous Kafka consumers --
% test and production, for example.
% consume and <consumerFunction> must agree on the name of the metaInfo
% field used to store the topic name.
msg.metaInfo.resultTopic = strcat('pump_results_', ...
    matlab.lang.makeValidName(msg.metaInfo.groupId));

% The name of the connection to the persistence service. Configured
% (set) by MATLAB Production Server admin.
msg.metaInfo.connection = 'Azure_Redis';

% Variables in the data table that we publish to Kafka.
msg.metaInfo.outputVariables = ...
    struct('Flow',NaN,'Pressure',NaN,'Current',NaN);

% Emit more log messages.
msg.metaInfo.Verbose = true;

% Finally, consume (process) the data in this input message.
mps.stream.consume(msg);
end
```
Test Stream Processing Function

Process Engineer

4. Integrate with Production Systems

```python
results = runtests('predictFaults_tests')

Running predictFaults_tests
....
Done predictFaults_tests

results =
1x4 TestResult array with properties:

- Name
- Passed
- Failed
- Incomplete
- Duration
- Details

Totals:
4 Passed, 0 Failed, 0 Incomplete.
0.01614 seconds testing time.
```
Test and Debug Streaming Function
Package Stream Processing Function

Integrate with Production Systems

Process Engineer
Package and Test to generate compiled archive
Compiled Package and Runtime requirements
Starting MATLAB Production Server Dashboard

Integrate with Production Systems

Process Engineer
Deploying Streaming Function on Production System

Integrate with Production Systems

Process Engineer

MATLAB Production Server

MATLAB Analytics

MATLAB Compiler SDK

MATLAB
Integrate Analytics with Production Systems

Production System

- Azure
- Worker processes
- Request Broker
- Connector
- Apache Kafka
- State Persistence
- Storage Layer

Analytics Development
- MATLAB Compiler SDK
- MATLAB
  - Debug
  - Package & Deploy
  - Model

Business Decisions
- kibana
- Presentation Layer

Edge
- Generate telemetry
- System Architect
- System Architect

Flow:
1. Integrate with Production Systems
2. Generate telemetry
3. MATLAB Production Server
4. Worker processes
5. Request Broker
6. Apache Kafka
7. State Persistence
8. Storage Layer
9. Business Decisions
MATLAB Production Server on Azure

Production System

- Connectors for Streaming/Event Data
- State Persistence
- Connectors for Storage & Databases
- Application Gateway Load Balancer

Management Server

MATLAB Production Server(s) scaling group

Virtual Network

https management endpoint

Enterprise Applications

Integrate with Production Systems

System Architect
MathWorks Cloud Reference Architecture

MATLAB in the Cloud

Use MATLAB in the Cloud

Run in different cloud environments from MathWorks Cloud to public clouds including AWS, Azure, and others

MathWorks Cloud

MathWorks Cloud provides you with instant access to MATLAB and other products and services you are licensed for hosted on MathWorks managed cloud infrastructure. With MATLAB Online®, you can use MATLAB in a web browser without installing, configuring, or managing any software. MathWorks Cloud also provides MATLAB Drive®, giving you the ability to store, access, and work with your files from anywhere. You can access MathWorks Cloud solutions anywhere across different devices, use them to teach and learn, and to incorporate MATLAB analytics for a variety of applications.

Learn more about hosted offerings.

Public Clouds
MPS License and Instance Settings
Serving REST Calls on Production Server
Review System Requirements

- Requirements from the Process Engineer
  - Every millisecond, each pump generates a time-stamped record of flow, pressure, and current
  - Model expects 1 sec. window of data per pump
  - Initially, 1’s – 10’s of devices, but quickly scale to 100’s

- Requirements from the Operator
  - Alerts when parameters drift outside the expected ranges
  - Continuous estimating of RUL for each pump
Integrate Analytics with Production Systems
Connecting MATLAB Production Server to Kafka

- Connector feeds single **Kafka topic** to a MATLAB function

- **Publisher library** for MATLAB for writing to a results stream

- Connector Features:
  - Deploy as a micro-service with Docker
  - Drive everything through config
  - Group data into time windows and pass to MATLAB as a timetable
  - Use Kafka’s check-pointing (i.e. at-least-once)
Setting up the Kafka Connector

```bash
#!/usr/bin/env bash

BASE=$(pwd)

echo "BASE: ${BASE}"

MPS_HOME=$BASE
KAFKA_CONNECTOR_DIR=${MPS_HOME}/../kafka-connector/Software/Java
MPS_CLIENT=${KAFKA_CONNECTOR_DIR}/client/java/mps_client.jar
KAFKA_CONNECTOR=${KAFKA_CONNECTOR_DIR}/lib/com/mathworks/mps/kafka-connector/1.1.0/kafka-connector-1.1.0-jar-with-dependencies.jar
CLASSPATH=${MPS_CLIENT}:${KAFKA_CONNECTOR}

# CLASSPATH=${CLASSPATH}:${MPS_HOME}/lib/com/mathworks/mps/kafka-connector/1.1.0/kafka-connector-jar-with-dependencies.jar

export GROUP_ID=mpsstuff
export CONNECTOR_TOPIC=to-mps
export CONNECTOR_TOPIC_OUT=from-mps
export MPS_CONNECT=http://localhost:9910
export MPS_ARCHIVE=PumpFault
export MPS_FUNCTION=streamingFunction
export BOOTSTRAP_SERVERS=localhost:9092
export MPS_DISPATCH_FUNCTION=pumpconsume
# Skip or stop
export MATLAB_ERROR_ACTION=skip

echo "CLASSPATH == ${CLASSPATH}"

exec java -cp ${CLASSPATH} \
-Dlog4j.configuration=file:${MPS_HOME}/config/log4j.properties \
com.mathworks.mps.client.kafka.KafkaConnector @
```
Kafka connector architecture

- **Kafka**
  - Topic: \( P_0, P_1, \ldots, P_n \)
  - Consumer Thread Pool: \( C_0, C_1, \ldots, C_n \)
  - Committed Offsets: \( P_0, P_1, \ldots, P_n \)

- **Message State (Offsets, Timestamps, Watermarks)**
  - Active Windows: \( P_0, P_1, \ldots, P_n \)
    - \( W_n, W_{n-1}, \ldots, W_1, W_0 \)

- **Async Request Handler**
  - \( P_0, P_1, \ldots, P_n \)
    - \( \ldots, r_1, r_0 \)

- **Production Server Java Client**
  - Networking Threads
  - Async HTTP to Server

- Integration with Production Systems

---

4. Integrate with Production Systems

System Architect
Streaming data is treated as an unbounded Timetable

<table>
<thead>
<tr>
<th>Event Time</th>
<th>Pump Id</th>
<th>Flow</th>
<th>Pressure</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>18:01:10</td>
<td>Pump1</td>
<td>1975</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>18:10:30</td>
<td>Pump3</td>
<td>2000</td>
<td>109</td>
<td>115</td>
</tr>
<tr>
<td>18:05:20</td>
<td>Pump1</td>
<td>1980</td>
<td>105</td>
<td>105</td>
</tr>
<tr>
<td>18:10:45</td>
<td>Pump2</td>
<td>2100</td>
<td>110</td>
<td>100</td>
</tr>
<tr>
<td>18:30:10</td>
<td>Pump4</td>
<td>2000</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>18:35:20</td>
<td>Pump4</td>
<td>1960</td>
<td>103</td>
<td>105</td>
</tr>
<tr>
<td>18:20:40</td>
<td>Pump3</td>
<td>1970</td>
<td>112</td>
<td>104</td>
</tr>
<tr>
<td>18:39:30</td>
<td>Pump4</td>
<td>2100</td>
<td>105</td>
<td>110</td>
</tr>
<tr>
<td>18:30:00</td>
<td>Pump3</td>
<td>1980</td>
<td>110</td>
<td>113</td>
</tr>
<tr>
<td>18:30:50</td>
<td>Pump3</td>
<td>2000</td>
<td>100</td>
<td>110</td>
</tr>
</tbody>
</table>

MATLAB Function

<table>
<thead>
<tr>
<th>Time window</th>
<th>Pump Id</th>
<th>Bearing</th>
</tr>
</thead>
<tbody>
<tr>
<td>18:00:00</td>
<td>Pump1</td>
<td>5</td>
</tr>
<tr>
<td>18:10:00</td>
<td>Pump3</td>
<td>...</td>
</tr>
<tr>
<td>18:10:00</td>
<td>Pump4</td>
<td>...</td>
</tr>
<tr>
<td>18:20:00</td>
<td>Pump1</td>
<td>...</td>
</tr>
<tr>
<td>18:20:00</td>
<td>Pump3</td>
<td>4</td>
</tr>
<tr>
<td>18:20:00</td>
<td>Pump4</td>
<td>...</td>
</tr>
<tr>
<td>18:30:00</td>
<td>Pump5</td>
<td>...</td>
</tr>
<tr>
<td>18:30:00</td>
<td>Pump3</td>
<td>5</td>
</tr>
<tr>
<td>18:30:00</td>
<td>Pump4</td>
<td>8</td>
</tr>
</tbody>
</table>
Messaging adapter for Production Server

- Bridges streaming data and Production Server Async Java Client
- Batches incoming messages and sends them via HTTP request/response
  - Time windows, event time processing, and out-of-order data
- Uses Asynchronous pipeline model with back-pressure
  - Kafka consumers are automatically paused when server is busy
- Supports sequential (stateful) and unordered (stateless) processing
  - Provide unique stream ID/topic/partition info for persistence layer
- Pass data as MATLAB timetables
- Partition aware – enables full exploitation of partition-based parallelism
Creating persistence

Integrate with Production Systems

Process Engineer
Attaching persistence
Debug your streaming function on live data
Debug a Stream Processing Function in MATLAB

Integrate with Production Systems

System Architect
Running Kafka with MPS
Complete your application

Integrate with Production Systems

System Architect

Edge
- Generate telemetry
- Wind
- Car
- Server
- Airplane

Production System
- MATLAB Production Server
  - Worker processes
  - Request Broker
- Connector
- Apache Kafka
- State Persistence

Analytics Development
- MATLAB Compiler SDK
- Debug
- Package & Deploy
- Model

Business Decisions
- Storage Layer
- Presentation Layer

Complete your application
Complete Your Application
Build Standalone UI based applications in MATLAB
MATLAB, Simulink and Cloud Reference Architectures provide “Integrated AI Development and Deployment Workflow” for Cross Functional Teams

- Successfully use Digital Twins to generate faults and train models
- Fast prototyping of physical and AI models with MATLAB
- Easy integration with OSS
- Cloud reference architectures for enabling faster IT setup
- Customize dashboard for Operator’s needs
Resources to learn and get started

- GitHub: MathWorks Reference Architectures
- Working with Enterprise IT Systems
- Data Analytics with MATLAB
- Simulink