# GIRO (Generalized Integrated Routing Optimization)

GIRO (Generalized Integrated Routing Optimization) is an advanced optimization engine designed to determine the most efficient routing configurations in complex production networks, such as oil and gas fields. It uses mathematical algorithms to evaluate multiple routing scenarios and select the one that maximizes production output while minimizing losses or constraints.  
  
Within the context of Integrated Production Modeling (IPM) tools like Resolve, GIRO is used to dynamically analyze and optimize how wells are connected to manifolds and surface facilities. By simulating various routing options, GIRO enables engineers to make data-driven decisions that improve overall field performance and operational efficiency.  
  
GIRO's integration into workflows—such as those in IFM or RMSY scenarios—allows automated, iterative evaluations of well configurations. This results in optimal well line-ups that respond to changing field conditions, production targets, and engineering constraints.

# How to Use GIRO

* Follow these steps to run GIRO and review the results:

## Step 1: Open IVM

1. Launch the IVM (Integrated Visual Modeler) application.  
2. Navigate to the Field Overview section.  
3. Click on the RMSY button.

## Step 2: Execute GIRO

1. Press the GIRO button to initiate the process.

## Step 3: Wait for Execution

1. After clicking the button, wait while GIRO executes.  
2. The Resolve and GAP models will automatically open on your desktop.  
3. Do not interrupt the process during model loading.

## Step 4: View Results

1. After execution, click the GIRO Results button.  
2. The system will display the latest run date and a comparison of key parameters between SQC-generated and GIRO-generated values.

# GIRO Optimization of Well Line-Up of RMSY in Resolve

## Activities Performed:

• Workflow Development:  
 - Created a new Resolve workflow incorporating GIRO optimization logic for the RMSY well line-up.  
 - Integrated a routing optimizer to analyze various routing combinations and determine the best configuration for maximizing production efficiency.  
• Optimization Execution:  
 - Ran multiple optimization scenarios to evaluate potential routing paths and their impacts on production.  
 - Implemented a dynamic evaluation process to ensure the workflow adapts to changing well conditions and inputs.

## Results and Improvements:

• Improved Routing Efficiency:  
 - Identified optimal routing configurations, maximizing production output while minimizing losses.  
• Enhanced Decision-Making Tools:  
 - Provided actionable insights for operators to adjust well line-ups based on real-time optimization results.

# GIRO Optimization of Well Line-Up for Selected Wells and Manifold in IFM

To integrate GIRO into IFM, a new Resolve model called Optimizer.rsl was created. This model is specifically configured to work with GIRO optimization logic and parameters. It acts as a bridge between IFM and GIRO, enabling seamless interaction and efficient optimization within the IFM environment.

## GIRO Field in the Workflow

A new field named “GIRO” was added to support integration with the model in the workflow. This allows the system to recognize and interact with the GIRO-based configuration during execution.

## GIRO Workflow Structure

A new GIRO workflow was created to run the model and extract routing data from its results. This workflow includes:  
- The Resolve model (Optimizer.rsl) for running the GIRO optimization  
- Three Python objects for handling and manipulating the data extracted from the model output  
- A FlexDataStore component to store input data and final routing results for further use

## 1. Setting actual inputs to model

To set inputs for GAP model used next steps:  
1. Set for each well IPR data as ResPress, GOR, PI, WC etc.  
2. Set each separator Pressure

## 2. Run Resolve optimizer

For running Resolve used OpenServer command:  
- Resolve.RUN()

## 3. Extracting Optimal Solution from GIRO Results

To extract all solutions from the model results:  
1. Extract all OilRate string values from the results – these represent outputs from different iterations.  
2. Integrate Python code to:  
 - Split the string values into a list of numerical results  
 - Determine the total number of iterations  
3. Loop through all OilRate values to:  
 - Find the maximum OilRate  
 - Identify the iteration index of the maximum  
4. Retrieve corresponding routing configuration.  
5. Store optimal routing into FlexDataStore.

## Routing Result Mapping

Routing results are initially returned as integer routing IDs or codes. Before saving, these IDs are mapped to their human-readable routing names or definitions.