Production-Function Approach to Portfolio Evaluation

Version 1.4 Draft

28 April 2020

# Concept

We separate the financial and conversion-efficiency aspects of the production process, which are generic across all technologies, from the physical and technical aspects, which are necessarily specific to the particular process. The motivation for this is that the financial and waste computations can be done uniformly for any technology (even for disparate ones such as PV cells and biofuels) and that different experts may be required to assess the cost, waste, and techno-physical aspects of technological progress.

# Formulation

## Sets

|  |  |  |
| --- | --- | --- |
| Set | Description | Examples |
|  | capital | equipment |
|  | fixed cost | rent, insurance |
|  | input | feedstock, labor |
|  | output | product, co-product, waste |
|  | metric | cost, jobs, carbon footprint, efficiency, lifetime |
|  | technical parameter | temperature, pressure |
|  | technology type | electrolysis, PV cell |
|  | scenario | the result of a particular investment |
|  | investment category | investment alternatives |
|  | investment | a particular investment |
|  | portfolio | a basket of investments |

## Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | Description | Units |
|  | calculated | unit cost | USD/unit |
|  | function | capital cost | USD |
|  | cost | lifetime of capital | year |
|  | cost | scale of operation | unit/year |
|  | function | fixed cost | USD/year |
|  | input | input quantity | input/unit |
|  | calculated | ideal input quantity | input/unit |
|  | waste | input efficiency | input/input |
|  | cost | input price | USD/input |
|  | calculated | ideal output quantity | output/unit |
|  | calculated | output quantity | output/unit |
|  | waste | output efficiency | output/output |
|  | cost | output price (+/-) | USD/output |
|  | calculated | metric | metric/unit |
|  | function | production function | output/unit |
|  | function | metric function | metric/unit |
|  | parameter | technical parameter | (mixed) |
|  | variable | scenario inputs | (mixed) |
|  | variable | scenario outputs | (mixed) |
|  | function | scenario evaluation | (mixed) |
|  | function | scenario probability | 1 |
|  | variable | investment cost | USD |
|  | random variable | investment outcome | (mixed) |
|  | random variable | portfolio outcome | (mixed) |
|  | calculated | portfolio cost | USD |
|  | parameter | minimum portfolio cost | USD |
|  | parameter | maximum portfolio cost | USD |

## Cost

The cost characterizations (capital and fixed costs) are represented as functions of the scale of operations and of the technical parameters in the design:

* Capital cost: .
* Fixed cost: .

The per-unit cost is computed using a simple levelization formula:

## Waste

The waste relative to the idealized production process is captured by the parameters. Expert elicitation might estimate how the s would change in response to R&D investment.

* Waste of input: .
* Waste of output: .

## Production

The production function idealizes production by ignoring waste, but accounting for physical and technical processes (e.g., stoichiometry). This requires a technical model or a tabulation/fit of the results of technical modeling.

## Metrics

Metrics such as efficiency, lifetime, or carbon footprint are also compute based on the physical and technical characteristics of the process. This requires a technical model or a tabulation/fit of the results of technical modeling. We use the convention that higher values are worse and lower values are better.

## Scenarios

A *scenario* represents a state of affairs for a technology . If we denote the scenario as , we have the input variables

and the output variables

and their relationship

where

for the technology of the scenario.

## Investments

An *investment* assigns a probability distribution to scenarios:

.

such that

or ,

depending upon whether one is performing the computations discretely or continuously. Expectations and other measures on probability distributions can be computed from the . We treat the outcome as a random variable for the outcomes according to the distribution .

Because investment options may be mutually exclusive, as is the case for investing in the same R&D at different funding levels, we say is the set of mutually exclusive investments (i.e., only one can ocurr) in investment category : investments in different categories can be combined arbitrarily, but just one investment from each may be chosen.

Thus the universe of all portfolios is , so a particular portfolio has components . The overall outcome of a portfolio is a random variable:

The cost of an investment , so the cost of a porfolio is:

## Decision problem

The multi-objective decision problem is

such that

,

where is the expectation operator , value-at-risk, or another operator on probability spaces. Recall that is a vector with components for cost and each metric , so this is a multi-objective problem.

The two-stage decision problem is a special case of the general problem outlined here: Each scenario can be considers as a composite of one or more stages.

## Experts

Each expert elicitation takes the form of an assessment of the probability and range (e.g., 10th to 90th percentile) of change in the cost or waste parameters or the production or metric functions. In essence, the expert elicitation defines for each potential scenario of each investment .

# Examples

## Idealized electrolysis of water

Here is a very simple model for electrolysis of water. We just have water, electricity, a catalyst, and some lab space. We choose the fundamental unit of operation to be moles of H2:

     H2O → H2 + ½ O2

Experts could assess how much R&D to increase the various efficiencies would cost. They could also suggest different catalysts, adding alkali, or replacing the process with PEM.

### Tracked quantities.

### Current design.

(due to mass transport loss on input)

(due to ohmic losses on input)

(due to mass transport loss on output)

(due to mass transport loss on output)

### Current costs.

(cost of Al-Ni catalyst)

(effective lifetime of Al-Ni catalyst)

(rough estimate for a 50W setup)

### Current prices.

### Production function (à la Leontief)

### Metric function.

### Performance of current design.

(i.e., not profitable since it is positive)

# Implementation

Database tables (one per set) hold all of the variables and the expert assessments. These tables are augmented by concise code with mathematical representations of the production and metric functions.

The Monte-Carlo computations are amenable to fast tensor-based implementation in Python.

See <<https://github.com/NREL/portfolio/tree/master/production-function/framework/code/tyche/>> for the tyche package that computes cost, production, and metrics from a technology design.

## Database tables

Each analysis case is represented by a Technology and a Scenario within that technology.

### Metadata about indices

The indices table simply describes the various indices available for the variables. The Offset column specifies the memory location in the argument for the production and metric functions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Technology | Type | Index | Offset | Description | Notes |
| Simple electrolysis | Capital | Catalyst | 0 | Catalyst |  |
| Simple electrolysis | Fixed | Rent | 0 | Rent |  |
| Simple electrolysis | Input | Water | 0 | Water |  |
| Simple electrolysis | Input | Electricity | 1 | Electricity |  |
| Simple electrolysis | Output | Oxygen | 0 | Oxygen |  |
| Simple electrolysis | Output | Hydrogen | 1 | Hydrogen |  |
| Simple electrolysis | Metric | Cost | 0 | Cost |  |
| Simple electrolysis | Metric | Jobs | 1 | Jobs |  |
| Simple electrolysis | Metric | GHG | 2 | GHGs |  |

### Design variables

The design table specifies the values of all of the variables in the mathematical formulation of the design.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Technology | Scenario | Variable | Index | Value | Units | Notes |
| Simple electrolysis | Base | Input | Water | 19.04 | g/mole |  |
| Simple electrolysis | Base | Input Efficiency | Water | 0.95 | 1 |  |
| Simple electrolysis | Base | Input | Electricity | 279 | kJ/mole |  |
| Simple electrolysis | Base | Input Efficiency | Electricity | 0.85 | 1 |  |
| Simple electrolysis | Base | Output Efficiency | Oxygen | 0.90 | 1 |  |
| Simple electrolysis | Base | Output Efficiency | Hydrogen | 0.90 | 1 |  |
| Simple electrolysis | Base | Lifetime | Catalyst | 3 | yr |  |
| Simple electrolysis | Base | Scale |  | 6650 | mole/yr |  |
| Simple electrolysis | Base | Input price | Water | 4.8e-3 | USD/mole |  |
| Simple electrolysis | Base | Input price | Electricity | 3.33e-5 | USD/kJ |  |
| Simple electrolysis | Base | Output price | Oxygen | 3.0e-3 | USD/g |  |
| Simple electrolysis | Base | Output price | Hydrogen | 1.0e-2 | USD/g |  |

Note that the Value column can either contain numeric literals or Python expressions specifying probability distribution functions. For example, a normal distribution with mean of five and standard deviation of two would be written st.norm(5, 2). All of the [Scipy probability distribution functions](https://docs.scipy.org/doc/scipy-1.4.1/reference/tutorial/stats/continuous.html#continuous-distributions-in-scipy-stats) are available for use, as are two special functions, constant and mixture. The constant distribution is just a single constant value; the mixture distribution is the mixture of a list of distributions, with specified relative weights. The mixture function is particularly important because it allows one to specify a first distribution in the case of an R&D breakthrough, but a second distribution if no breakthrough occurs.

### Metadata for functions

The functions table simply documents which Python module and functions to use for the technology and scenario.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Technology | Style | Module | Capital | Fixed | Production | Metrics | Notes |
| Simple electrolysis | numpy | simple\_electrolysis | capital\_cost | fixed\_cost | production | metrics |  |

Currently only the numpy style of function is supported, but later plain Python functions and tensorflow functions will be allowed.

### Parameters for functions

The parameters table contains ad-hoc parameters specific to the particular production and metrics functions. The Offset column specifies the memory location in the argument for the production and metric functions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Technology | Scenario | Parameter | Offset | Value | Units | Notes |
| Simple electrolysis | Base | Oxygen production | 0 | 16.00 | g |  |
| Simple electrolysis | Base | Hydrogen production | 1 | 2.00 | g |  |
| Simple electrolysis | Base | Water consumption | 2 | 18.08 | g |  |
| Simple electrolysis | Base | Electricity consumption | 3 | 237 | kJ |  |
| Simple electrolysis | Base | Jobs | 4 | 1.5e-4 | job/mole |  |
| Simple electrolysis | Base | Reference scale | 5 | 6650 | mole/yr |  |
| Simple electrolysis | Base | Reference capital cost for catalyst | 6 | 0.63 | USD |  |
| Simple electrolysis | Base | Reference fixed cost for rent | 7 | 1000 | USD/yr |  |
| Simple electrolysis | Base | GHG factor for water | 8 | 0.00108 | gCO2e/g | based on 244,956 gallons = 1 Mg CO2e |
| Simple electrolysis | Base | GHG factor for electricity | 9 | 0.138 | gCO2e/kJ | based on 1 kWh = 0.5 kg CO2e |

### Units for results

The results table simply specifies the units for the results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Technology | Variable | Index | Units | Notes |
| Simple electrolysis | Cost | Cost | USD/mole |  |
| Simple electrolysis | Output | Oxygen | g/mole |  |
| Simple electrolysis | Output | Hydrogen | g/mole |  |
| Simple electrolysis | Metric | Cost | job/gH2 |  |
| Simple electrolysis | Metric | Jobs | job/gH2 |  |
| Simple electrolysis | Metric | GHG | gCO2e/gH2 |  |

### Tranches of investments.

In the tranches table, each *category* of investment contains a set of mutually exclusive *tranches* that may be associated with one or more *scenarios* defined in the designs table. Typically, a category is associated with a technology area and each tranche corresponds to an investment strategy within that category.

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Tranche | Scenario | Notes |
| Electrolysis R&D | No Electrolysis R&D | Base Electrolysis |  |
| Electrolysis R&D | Low Electrolysis R&D | Slow Progress on Electrolysis |  |
| Electrolysis R&D | Medium Electrolysis R&D | Moderate Progress on Electrolysis |  |
| Electrolysis R&D | High Electrolysis R&D | Fast Progress on Electrolysis |  |

### Investments

In the investments table, each *investment* is associated with a single *tranche* in one or more *categories*. An investment typically combines tranches from several different investment categories.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Investment | Category | Tranche | Amount | Notes |
| No R&D Spending | Electrolysis R&D | No Electrolysis R&D | 0 |  |
| Low R&D Spending | Electrolysis R&D | Low Electrolysis R&D | 1000000 |  |
| Medium R&D Spending | Electrolysis R&D | Medium Electrolysis R&D | 2500000 |  |
| High R&D Spending | Electrolysis R&D | High Electrolysis R&D | 5000000 |  |

## Python module and functions

Each technology design requires a Python module with a production and metrics function.

# Simple electrolysis.  
  
  
# All of the computations must be vectorized, so use `numpy`.  
import numpy as np  
  
  
# Capital-cost function.  
def capital\_cost(scale, parameter):  
  
 # Scale the reference values.  
 return np.stack([np.multiply(parameter[6], np.divide(scale, parameter[5]))])  
  
  
# Fixed-cost function.  
def fixed\_cost(scale, parameter):  
  
 # Scale the reference values.  
 return np.stack([np.multiply(parameter[7], np.divide(scale, parameter[5]))])  
  
  
# Production function.  
def production(capital, fixed, input, parameter):  
  
 # Moles of input.  
 water = np.divide(input[0], parameter[2])  
 electricity = np.divide(input[1], parameter[3])  
  
 # Moles of output.  
 output = np.minimum(water, electricity)  
  
 # Grams of output.  
 oxygen = np.multiply(output, parameter[0])  
 hydrogen = np.multiply(output, parameter[1])  
  
 # Package results.  
 return np.stack([oxygen, hydrogen])  
  
  
# Metrics function.  
def metrics(capital, fixed, input\_raw, input, output\_raw, output, cost, parameter):  
  
 # Hydrogen output.  
 hydrogen = output[1]  
  
 # Cost of hydrogen.  
 cost1 = np.divide(cost, hydrogen)  
  
 # Jobs normalized to hydrogen.  
 jobs = np.divide(parameter[4], hydrogen)  
  
 # GHGs associated with water and electricity.  
 water = np.multiply(input\_raw[0], parameter[8])  
 electricity = np.multiply(input\_raw[1], parameter[9])  
 co2e = np.divide(np.add(water, electricity), hydrogen)  
  
 # Package results.  
 return np.stack([cost1, jobs, co2e])