Deloitte.

# Choose the Right Ones: Efficient Grouping of Policies by Means of Linear Programming Combined with Machine Learning

Juni 2024

Zoran Nikolić

MAKING AN IMPACT THAT MATTERS since 1845

# Introduction

## Companies Struggle with Long Runtime and High Costs of Cash Flow Models

### Point of Departure

**Heavy model** coupled with a large number of insurance contracts

→ Leading to **runtime challenges** (long waiting time) and high **IT expenses**

### Target

1. **Reduce the number of policies** that need to be projected by grouping them
2. Keep a **high quality of the results** after grouping

### Traditional Method

**Non-Negative least-squares**

Already providing good results

### Modern Method

**Linear programming**
**+**
**Machine learning**

State-of-the-art grouping

1. **Challenge and Problem**
2. A Traditional Method
3. A Leap Forward
4. Conclusion
5. Appendix

# Stochastic Projection Model

## The „Heavy“ Model

**Input**

- Policyholder data  and assets on aggregated level
- Economic scenarios
- Assumptions
- Parameters

**Stochastic Model**

- Assets and liabilities
- Reflects the insurance company's types of products
- Allows for all relevant cash flows and any interactions
- Dynamic policyholder behavior and management actions
- Discounting to present value
- Monte Carlo stochastic modeling: projection performed thousands of times

**Output**

- Distribution for each measure
- Best estimate
- Confidence intervals

- **Necessary for products which include options and guarantees**
- **Stochastic modelling builds volatility and variability into the simulation**

# Runtime Challenge with Stochastic Projection Models

## Economic importance of policy grouping

**"Model Point"**

An insurance contract which should be projected in a cash-flow model

**100,000 Model Points**

| Type | Deterministic (Liabilities) Model | Stochastic Model (100 cores) |
|---|---|---|
| # Simulations | 1 | 5,000 |
| Runtime | 30 min | 60 h |

**1,000 Model Points**

| Type | Deterministic (Liabilities) Model | Stochastic Model (100 cores) |
|---|---|---|
| # Simulations | 1 | 5,000 |
| Runtime | 18 s | 100 min |

**Grouping contracts** can significantly reduce model runtime.

This is of **particular importance for Solvency II stochastic calculations.**

In **deterministic models**, runtime reduction is linear. In **stochastic models**, the reduction is much more significant.

# Formal Definition of the Problem (1/2)

## Represent the entire portfolio with few policies

### Recap

Portfolio with:
*M:* number of policies (model points)
*N:* number of projection variables
*L:* number of projection periods in years

$$A = [a_{m,n,l}],$$
$$1 \leq m \leq M, 1 \leq n \leq N, 1 \leq l \leq L$$

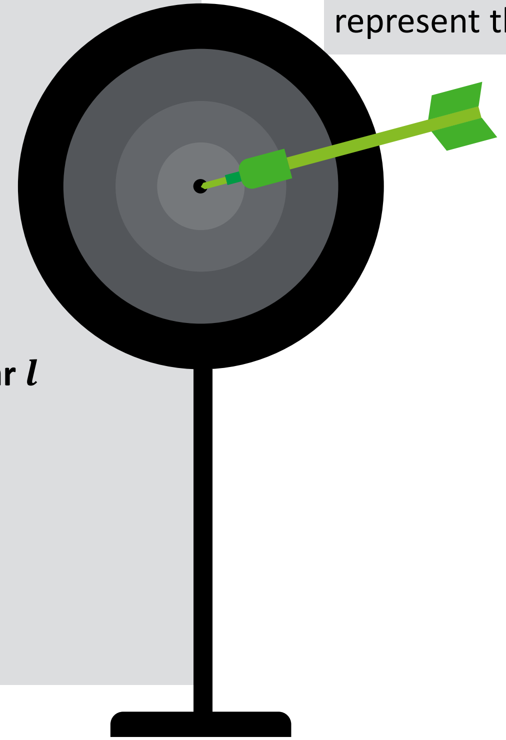
is the cube containing **future projected cash flows for each policy**

Define

$$b_{n,l} = \sum_{m=1}^{M} a_{m,n,l}$$

as the **sum of cash flows of all policies for variable $n$ in projection year $l$**
e.g. “sum of premiums of all policies in year 2025”

Array $B = [b_{n,l}]$ contains the
**aggregated cash flows and balance sheet items**
→$B$ represents the entire portfolio of policies



### Target

Find a “suitable” array of **weights** $X = [x_m]$ fulfilling

$$B = A * X$$

Note: A linear combination of policies should represent the entire portfolio.

# Formal Definition of the Problem (1/2)

## Represent the entire portfolio with few model points (= policies)

### Trivial Solution

There is obviously the trivial solution (for $\mathbb{I} = [1]$):

$$B = A * \mathbb{I}$$

### Target, reloaded

$A * X$ should be **“close to”** $B$ and at the same time $X$ should be **“sparse”.**

### “close to”

means for an array $\epsilon$ containing (small) **allowed deviations**

$$B - \epsilon * |B| \leq A * X \leq B + \epsilon * |B|$$

component-wise (want to replicate well each cash flow!)

### “sparse”

means $X$ must have **as many zero-entries as possible** (one additional constraint: $x_m \geq 0$)

➡ The non-zero entries of $X$ give us the weights for the **grouped portfolio**.

1. Challenge and Problem
2. **A Traditional Method**
3. A Leap Forward
4. Conclusion
5. Appendix

# A Traditional Method (1/7)

## Least square minimization problem

### Recap

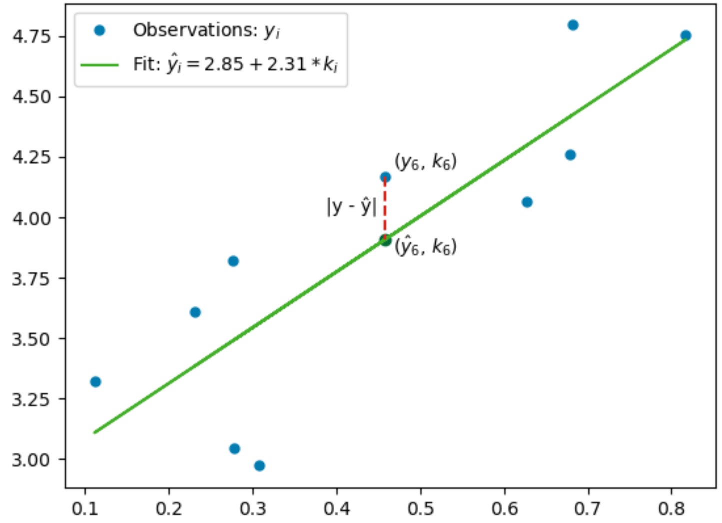
- **Motivation:** Minimizing error and thus provide the best possible solution

- **Setting:** Assuming that we have n observations $\left(y_i, k_i^T\right) := \left(y_i, k_{i1}, \dots, k_{ip}\right), i = 1, \dots, n$ available.

- **Goal**: find a vector $\hat{\beta} \in \mathbb{R}^s$ such that the sum of the squared differences between the observed response values $y_i$ and the corresponding fitted values
$$\hat{y}_i := (K * \beta)_i = \hat{\beta}_0 + \hat{\beta}_1 k_{i1} + \ldots + \hat{\beta}_p k_{ip}, i = 1, \dots, n$$
is minimized.

- **Least square minimization problem:**
Minimizing the sum of squared residuals
$$Q(\beta|y) := \|K * \beta - y\|^2 \text{ over } \beta$$

### Example for $p$ = 1, $\hat{\beta}_0$ = 2.85 and $\hat{\beta}_1$ = 2.31




### Theorem

If the matrix $K$ has full rank $s$ , then the minimum of $Q(\beta|y)$ is attained at
$$\hat{\beta} = (K^T K)^{-1} K^T y$$
and is called **the least square estimate of $\boldsymbol{\beta}$.**

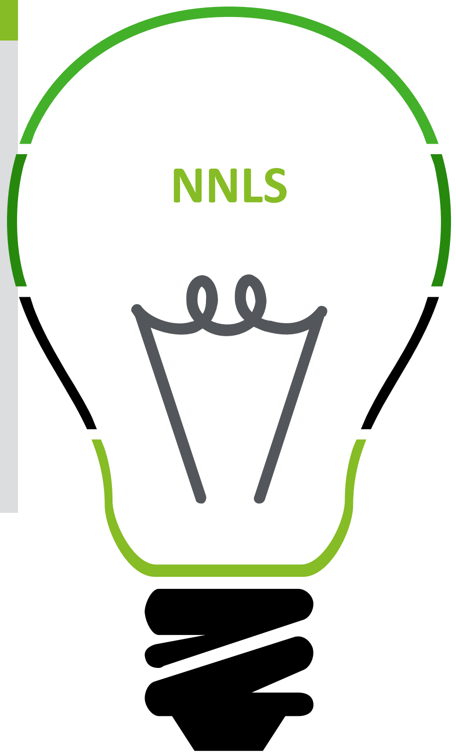
# A Traditional Method (2/7)

## Non-negative Least Squares (NNLS)

**Recap**

- **Setting:**
  $A$: the $(M * N * L)$ cube containing the cash flows of individual insurance policies
  $B$: the $(N * L)$ array containing aggregated cash flows

- **Target:**
  As before we require:
  $x_m \geq 0$ for the weights
  we want **as many $\boldsymbol{x_m}$ to be zero as possible**




**Task, reloaded:**

In the NNLS context the task becomes:

$\arg\min_{X} \|A * X - B\|^2$ **subject to $\boldsymbol{X \geq 0}$** and
number of **non-zero elements in $\boldsymbol{X}$ as little as possible**
or smaller than a pre-defined number of policies.

# A Traditional Method (3/7)

## Lawson-Hanson Algorithm

**Input**

Portfolio with $M$ policies, Projection of $N$ variables, Projection period $L$ years

- $A = [a_{m,n,l}]$,
  $1 \leq m \leq M, 1 \leq n \leq N, 1 \leq l \leq L$
  is the cube containing **future projected cash flows for each policy**
- Array $B = [b_{n,l}]$
  contains the **aggregated cash flows and balance sheet items** with
  $$b_{n,l} = \sum_{m=1}^{M} a_{m,n,l}$$
- $k$, the **expected number of policies**, with $k < n$.


**Output:**

Vector $X^* \in \mathbb{R}^n$ which minimizes

$$\min_{X \geq 0} \|A * X - B\|^2$$

such that $|P| \leq k$ with

$$P = \{i:\ x_i > 0\}.$$

# A Traditional Method (4/7)

## Lawson-Hanson Algorithm

### Setting

The basis of the Lawson-Hanson algorithm consists of two sets:
The set of **active and passive constraints** on a vector $X$.

- The **passive index set**:
  $$P = \{i: x_i > 0\}$$
  contains the chosen policies, which represent the grouped portfolio.
- The **active index set:**
  $$Z = \{i: x_i = 0\}$$
  contains the remaining policies.

**Unrestricted least squares estimation:**

- If $A_p$ has full rank, a unique solution to
  $$\arg\min_X \left\|A_p * X - B\right\|^2$$
  is:
  $$X = (A_p{}^T A_p)^{-1} A_p{}^T B$$
- $A_p$ is defined as:
  $$j\text{-th column of } A_p = \begin{cases} V_j, j \in P \\ 0, j \in Z \end{cases},$$
  where $V_i$ is the $i$-th column of $A$

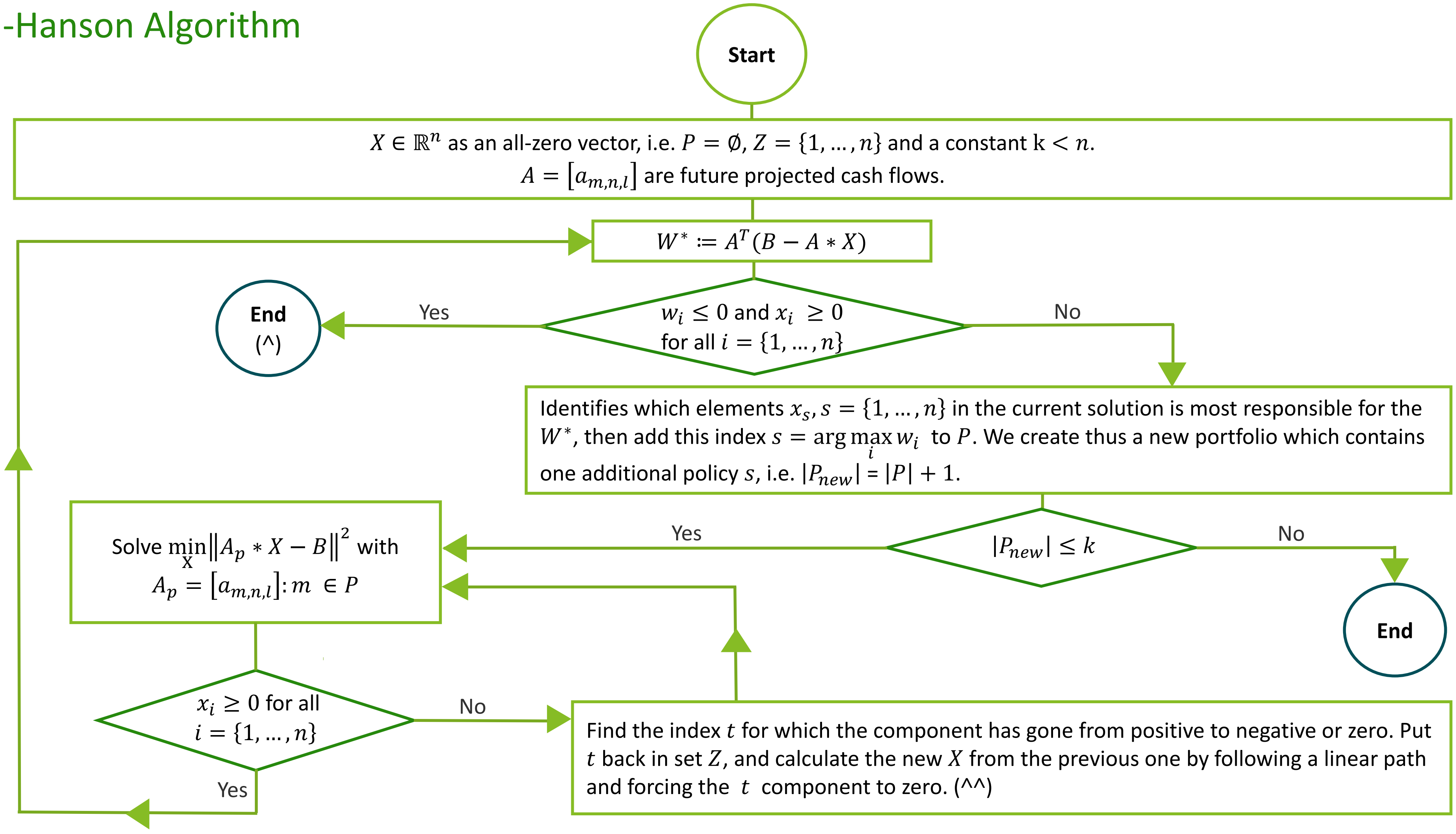
$X$ could have negative values


### Target:

Find a solution fulfilling
$$X^* = [x_m]^* \geq 0$$
for all $1 \leq m \leq M$ such that
$$|P| \leq k \text{ with } P = \{i: x_i > 0\},$$
where $k$ is a pre-defined number.

# A Traditional Method (5/7)

## Lawson-Hanson Algorithm



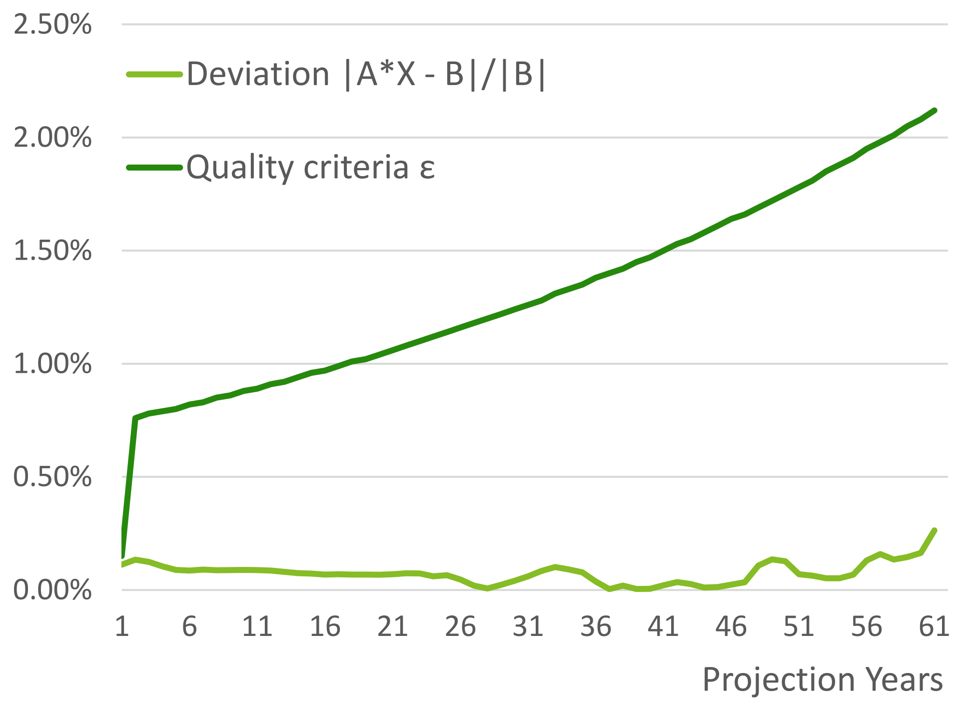

(^) For argumentation please see KKT .

(^^) For argumentation please see Lemma.

# A Traditional Method (6/7)

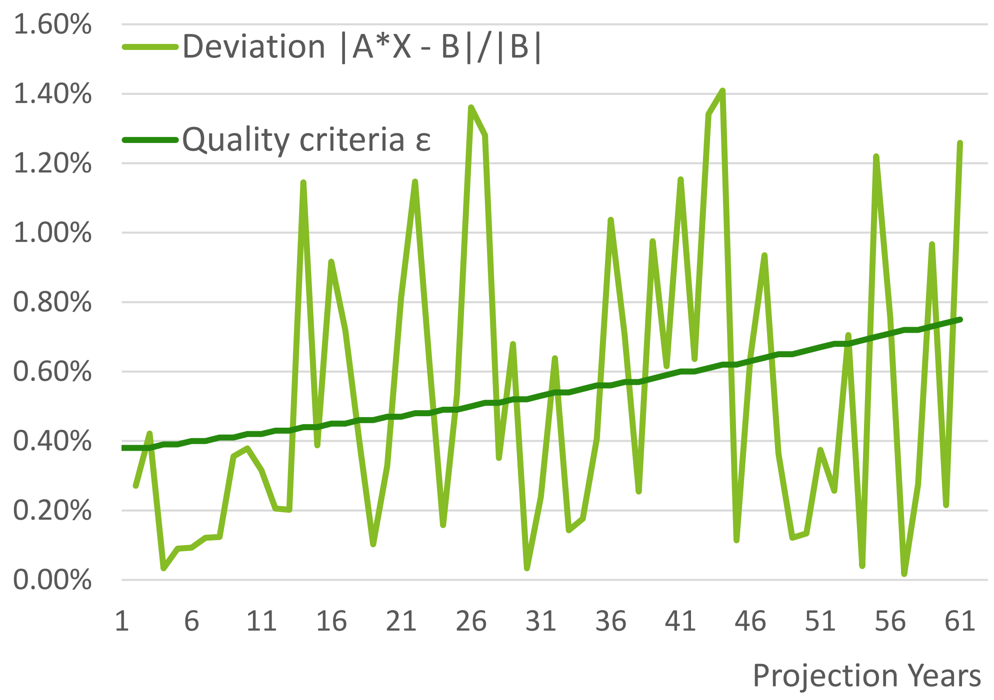
## NNLS Examples

### Technical Provisions




- Deviation **under threshold** in all projection years
- For a variable **with large values quite good results**

### Gross Surplus




- Deviation **above threshold**
- **no large outliers**: the least-squares algorithm punishes very large deviations
- **For gross surplus the quality criteria** often not met: much smaller values than for technical provisions

# A Traditional Method (7/7)

## Discussion of NNLS

### Disadvantages

- **Numerous iterations** coupled with manual interventions are required to meet quality standards
- Cannot be fully incorporated into a workflow
- Critical variables such as Gross Surplus may not be adequately replicated due to **lack of direct control**
- **Runtime escalates** with an increase in model points (non-zeros in $X$)

### Advantages

- Consideration of **cash flows in the projection**
- **Simple to execute** due to the use of least-squares regression
- **Often** yields **satisfying results**
- **Widely used in the industry** which ensures acceptance by all stakeholders

1. Challenge and Problem
2. A Traditional Method

## 3. A Leap Forward

4. Conclusion
5. Appendix

# A Leap Forward: Linear Programming (1/8)

## Back to the Basics

### Recap

Let us recall again the problem of policy grouping:

- $A$: array containing the cash flows of individual insurance policies
- $B$: array containing aggregated cash flows

- for an array $\epsilon$ containing (small) allowed deviations

$$B - \epsilon * |B| \leq A * X \leq B + \epsilon * |B|$$

component-wise (replicate well each cash flow!)

- We require $x_m \geq 0$ for the weights and we want **as many $x_m$ to be zero as possible.**

### Solution

Recently, a student cleverly noticed that the conditions can be **formulated in terms of Linear Programming.**

# A Leap Forward: Linear Programming  (1/8)

## Back to the Basics

### Task, reloaded

Minimize a modified norm of the weights vector $|X|$ subject to the following constraints:

$$A * X \leq B + \epsilon * |B|$$
$$-A * X \leq -B + \epsilon * |B|$$
$$0 \leq X$$

### Advantages

The given **quality criteria are always met**:
they are embedded in the **problem definition.**

Enormous **computational advantage:** The solution can be found on one of the edges.

### Recourses

- Many efficient **solvers** for this problem
- Excellent results with free GLOP tool from Google Operation Research team
- An explicit implementation: **DGO ML**

# A Leap Forward: Linear Programming (3/8)

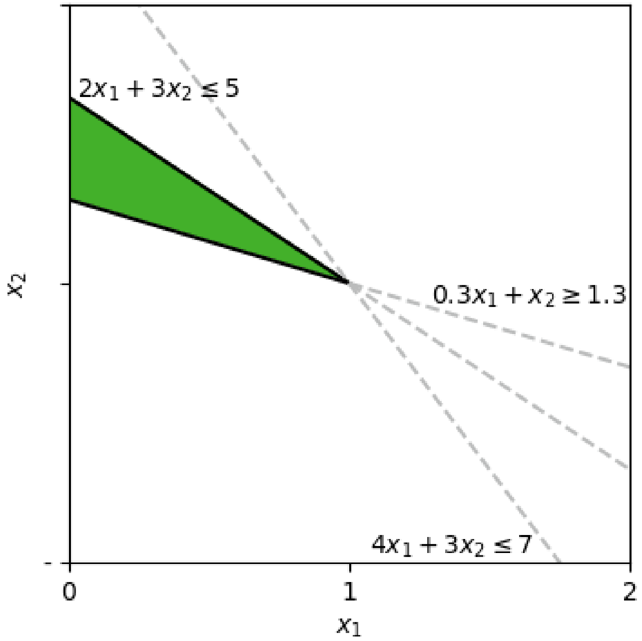
## Graphical solution

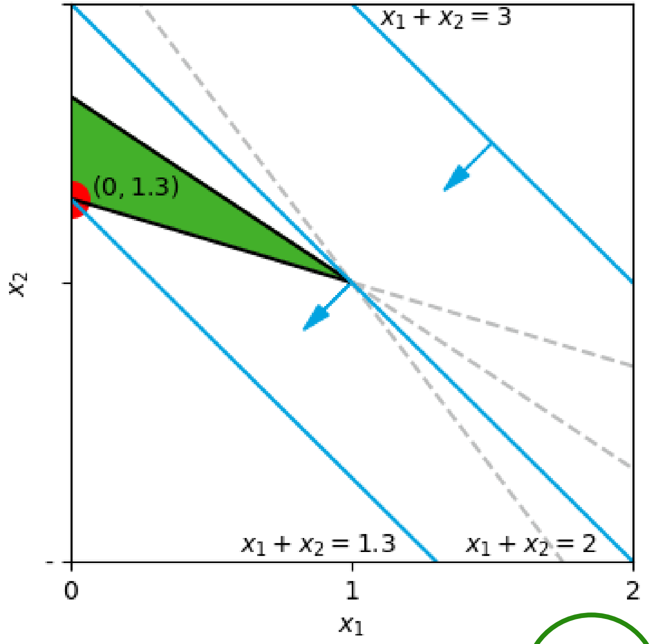
**Objective function**

$$\boldsymbol{min.\, x_1 + x_2,}$$

**such that:**

$$2x_1 + 3x_2 \leq 5$$
$$4x_1 + 3x_2 \leq 7$$
$$0.3x_1 + x_2 \geq 1.3$$
$$x_1, x_2 \geq 0$$



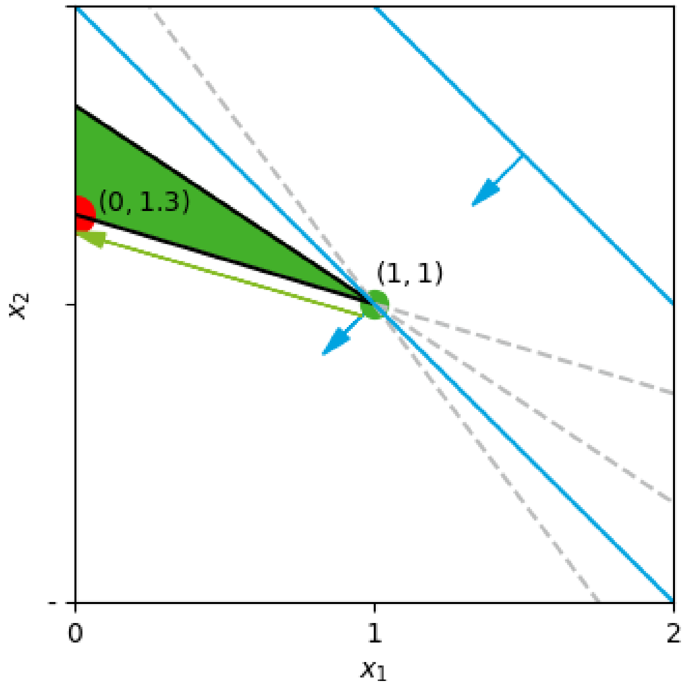

**Observation:**
The objective function values get better the further we push the blue line downwards.
Therefore, we see that the point **(0, 1.3) is the point with the best objective function** value among all points in the green area.

# A Leap Forward: Linear Programming (4/8)

## Graphical solution by using Simplex Algorithm

**Idea of the Simplex Algorithm**:

1. Start in any node $x$
2. If a neighboring node $x_0$ is better, move to $x_0$
3. Repeat this until we cannot find any better neighboring node.
   → the current node $x$ is optimal.




**Example:** we start at the point (1, 1) then ending up at (0, 1.3), which is the optimal solution.

# A Leap Forward: Linear Programming (5/8)

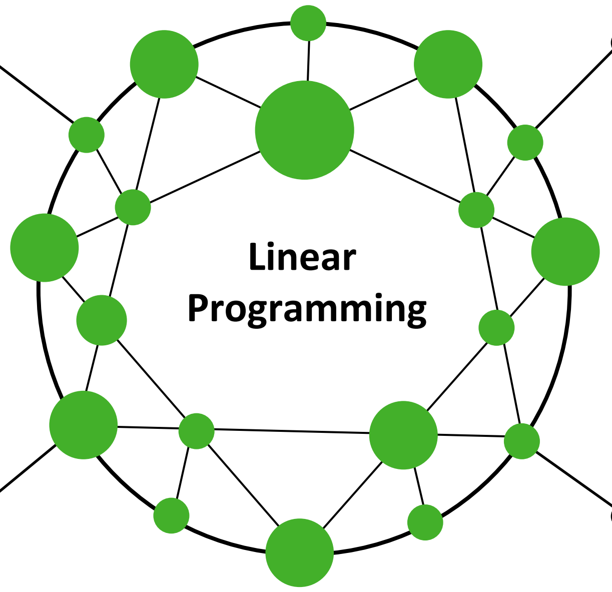
## Machine Learning Meets Linear Programming

**Paradigm in machine learning:**

Adjusting model parameters through grid search or advanced techniques to initiate training under various conditions.

**Sensitivity of Linear Optimization:**

The algorithm is highly responsive to small changes in input parameters.




**Pragmatic Parameter Variation:**

Utilize the model's sensitivity by systematically varying parameters to generate a set of results

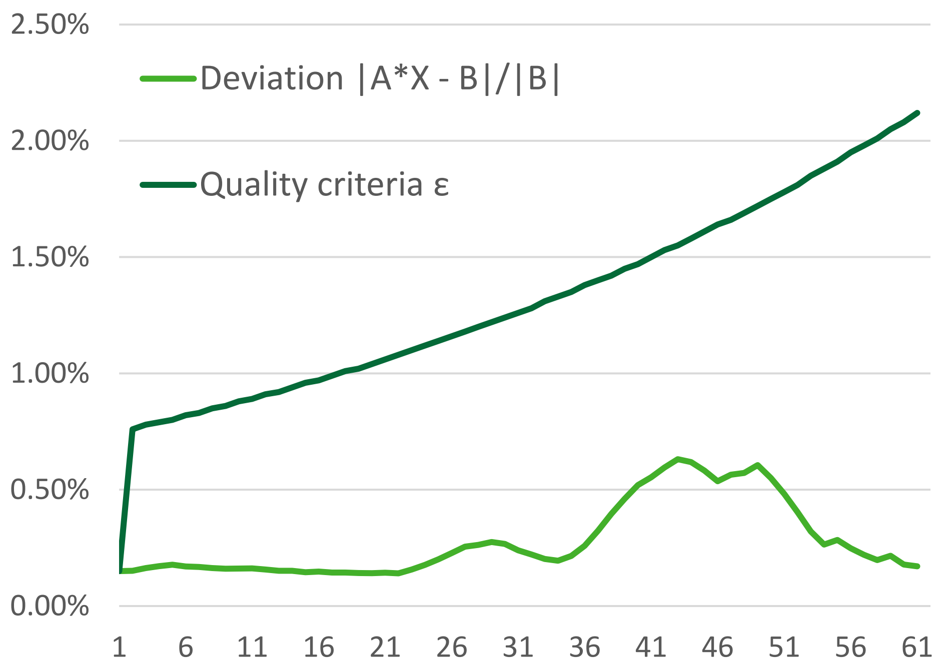
**Greedy Final Model Selection:**

The optimal model is selected based on having the fewest non-zero weights.

# A Leap Forward: Linear Programming (6/8)

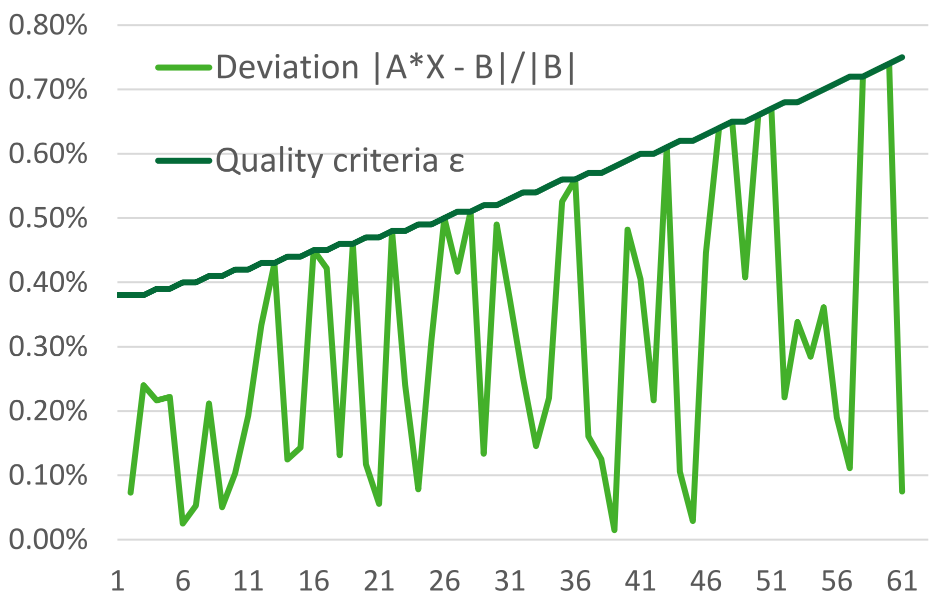
## Linear Programming examples

### Technical Provisions




- The first year is still challenging, but doable.
- Deviation **under threshold** in all projection years

### Gross Surplus




- For gross surplus which has much smaller values the **quality criteria are never breached**
- Often the light line „touches“ the dark one, but **never moves beyond the dark line**

# A Leap Forward: Linear Programming (7/8)

## Linear Programming Examples

**Example 1:** Fast and Reliable High Quality Optimization

| Cluster | Original amount of MPs | Amount of MPs after optimization | DGO ML Run time (s) | Change in MPs |
|---|---|---|---|---|
| 1 | 4,956 | 72 | 34 | -98.55% |
| 2 | 48,623 | 156 | 163 | -99.68% |
| 3 | 660,648 | 148 | 1.026 | -99.98% |
| 4 | 63,251 | 95 | 204 | -99.85% |
| 5 | 140,523 | 271 | 289 | -99.81% |
| 6 | 123,430 | 166 | 170 | -99.87% |
| 7 | 4,956 | 68 | 28 | -98.63% |
| 8 | 33,823 | 100 | 58 | -99.70% |
| 9 | 107,327 | 248 | 227 | -99.77% |
| 10 | 1,169 | 23 | 32 | -98.03% |
| **All** | **1,188,706** | **1,347** | **2,230** | **-99.89%** |

**Significant Model Points Reduction** within the constraints of the allowed deviations.

# A Leap Forward: Linear Programming (7/8)

## Linear Programming Examples

**Example 2:** Fast and Reliable High Quality Optimization

| Cluster | Original amount of MPs | Amount of MPs after optimization | Change in MPs |
|---|---|---|---|
| Cluster_1 | 8,450 | 446 | -94.72 % |
| Cluster_2 | 5,707 | 447 | -92.17 % |
| Cluster_3 | 2,190 | 385 | -82.42 % |
| Cluster_4 | 7,515 | 465 | -93.81 % |
| Cluster_5 | 7,276 | 461 | -93.66 % |
| **All** | **31,138** | **2,204** | **-92.92 %** |

**Significant reduction of MPs also for dynamic hybrid clusters** with five different **capital market scenarios** per cluster.

Depending on the size of the cluster and type of the cash flow projection tool, the amount of scenarios used has reached **200 scenarios in practice.**

Also possible to use **even more market scenarios** during the optimization.

# A Leap Forward: Linear Programming (8/8)

## Discussion

### Disadvantages

- **No control** for the **number of model points** in the resulting model
- **Commercial solvers** typically **perform better** than free solvers, leading to additional costs

### Advantages

- Allows direct **consideration of the goodness-of-fit** in the problem definition
- **Simple to implement and comprehend**
- Usually yields the **lowest number of model points** for a given goodness-of-fit of all models
- Can deal with **very large problems**
- Easy **integration in the business workflow**, since no manual intervention necessary
- Offers efficient runtime

# Conclusion

**1 — Policy Clustering**

This method **significantly reduces** the runtime of actuarial models, leading to **substantial cost savings**.

**2 — Classical Techniques**

These methods are intuitive and often effective but can **require many model points** for certain business lines.

**3 — Potential of Modern Techniques**

By using advanced machine learning methods, linear programming can **greatly reduce the number of policies and streamline validation.**

**4 — Promise of Neural Networks:**

Neural networks show **excellent results with very few model points**, though a business solution for neural network clustering has yet to be developed.

# Contact

**Zoran Nikolić**
**Partner**
**Deloitte Germany**
**Cologne Office**

Phone: +(49) 221 9732416
Mobile: +(49) 151 58077609
znikolic@deloitte.de