C&I Deposits Optimiser Rework

Harry Ashby

February 24, 2025

Presenting a reframing of the C&I deposits optimisation problem to consolidate and simplify approach.

0.1 Problem Formulation

Let $p \in \mathbb{N}$ be the number of products, and denote $\mathbf{P} \in \mathbb{R}^{p \times p}$ to be an array of flows such that P_{ij} represents the flow from product i to product j. Each one of these flows has been modelled previously. Let f be the number of 'non rate-related'[1] features in our models above.

Define tensor $\mathbf{F} \in \mathbb{R}^{p \times p \times f}$, where F_{ijk} represents the value of feature k as it pertains to the flow between product i and product j. Similarly, define $\mathbf{C_F} \in \mathbb{R}^{p \times p \times f}$ such that $C_{\mathbf{F}_{ijk}}$ represents the coefficient (or weight) of non rate-related feature k as it pertains to the flow between product i and product j.

Here, each $C_{\mathbf{F}_{ijk}}$ has been standardised - that is it comes to us with mean 0 and standard deviation 1. Therefore, all incoming features F_{ijk} need to be standardised too.

Let F^N_{ijk} be the non-standardised value for feature k which corresponds to the flow between product i and product j which we receive from the model. Let μ_{ijk} and σ_{ijk} be the received mean and standard deviation for feature k in scenario ij respectively. Then, we can define F_{ijk} as

$$F_{ijk} = \frac{(F^N_{ijk} - \mu_{ijk})}{\sigma_{ijk}}. \tag{1}$$

To get the weighted sum of all features for a specific scenario (product i to product j), we can use a tensor contraction, defined in equation (2), and displayed visually in figure 1. In equation (2), $\mathbf{F}^{T_{jk}}$ denotes that the transpose of $\mathbf{F}$ is taken by swapping the j and k dimensions. Figure 1 shows each coefficient of matrix $\mathbf{C}$ being multiplied against its corresponding feature in matrix $\mathbf{F}$ for a given flow from product i to product j. Then, we take the sum of all of these weighted features, which gives us the total contribution of all non rate-related features to the flow between product i and product j.

$$\mathbf{C}_{\mathbf{F}_{ijk}} \cdot \mathbf{F}^{T_{jk}} = \sum_{k=1}^{f} C_{\mathbf{F}_{ijk}} \cdot F_{ijk} \in \mathbb{R}^{p \times p} \tag{2}$$

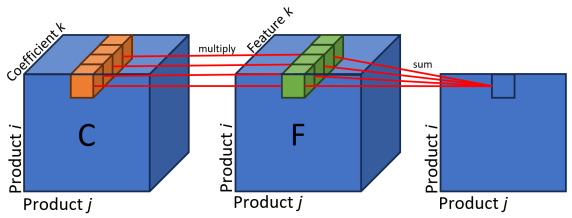

Figure 1: Tensor Contraction

Let m be the number of rate related features, and define $\mathbf{R} \in \mathbb{R}^{p \times p \times f}$ to be an array of all standardised[2] rate-related (optimisable) features, with $\mathbf{C_R} \in \mathbb{R}^{p \times p \times m}$ to be the corresponding array

[1] These could also be called our non-optimisable features, and include things such as the Bank of England base rate, a dummy indicator for a month, or a competitor's product rate. We cannot control these features and so we are not optimising them, hence why we have separated these from our 'rate related features'.

[2] With each R_{ijk} being standardised, we will have to de-standardise later to get the raw values for R_{ijk}

of coefficients for these features. Then the contribution to the flow of all rate-related (optimisable) features can be defined as

$$\mathbf{C}_{\mathbf{R}_{ijk}} \cdot \mathbf{R}^{T_{jk}} = \sum_{k=1}^{m} C_{\mathbf{R}_{ijk}} \cdot R_{ijk} \in \mathbb{R}^{p \times p} \tag{3}$$

Finally, knowing that our flows are linear regressions and as such have the general form

$$f = w_0 + \sum_{i=1}^{n} a_i w_i, \tag{4}$$

we notice that there is a constant coefficient w_0 which needs to be introduced. Define $\mathbf{C_0} \in \mathbb{R}^{p \times p}$ to be this standardised constant matrix, where $C_{\mathbf{0}_{ij}}$ represents the constant coefficient for the flow between product i and product j. Then we can express the standardised array of flows, $\mathbf{P}$, as

$$\mathbf{P} = \mathbf{C}_{\mathbf{F}_{ijk}} \cdot \mathbf{F}^{T_{jk}} + \mathbf{C}_{\mathbf{R}_{ijk}} \cdot \mathbf{R}^{T_{jk}} + \mathbf{C_0}. \tag{5}$$

Since some of the flows will be coming from or going to external sources, we need a way of discounting these when trying to calculate the bank's outgoings or incomings. For instance, let P was a 2x2 matrix, where product 1 is internal and product 2 is external:

$$P = \begin{bmatrix} \text{internal} \to \text{internal} & \text{internal} \to \text{external} \\ \text{external} \to \text{internal} & \text{external} \to \text{external} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 4 & 1 \end{bmatrix}.$$

Then, when calculating the outflow, we shouldn't consider any outflows originating **from external sources** (the bottom row) as we do not have to worry about paying for this outflow as we do not control product 2. Similarly, when calculating inflows, we shouldn't consider any inflows which go **to external sources** (the rightmost column). Therefore, we construct the matrices $\mathbb{I}_O$ and $\mathbb{I}_I$ where each element is an inverse indicator function showing whether a flow is originating from or going to an external source respectively. Defining inflow (money entering an account) as a postive direction, we have formally

$$\mathbb{I}_O \in \mathbb{R}^{p \times p} \quad \text{such that} \quad \mathbb{I}_{O_{ij}} = \begin{cases} -1 & \text{if flow } i \to j \text{ originates internally} \\ 0 & \text{if flow } i \to j \text{ originates externally} \end{cases}, \tag{6}$$

and

$$\mathbb{I}_I \in \mathbb{R}^{p \times p} \quad \text{such that} \quad \mathbb{I}_{I_{ij}} = \begin{cases} 1 & \text{if flow } i \to j \text{ eventuates internally} \\ 0 & \text{if flow } i \to j \text{ eventuates externally} \end{cases}. \tag{7}$$

By noting that flow $i \to j = -(j \to i)^{-1}$, we have the implication that $\mathbb{I}_{O_{ij}} = -\mathbb{I}_{I_{ji}}$ - that is that is a flow originates internally, than its inverse flow must eventuate internally (and vice versa for an external to internal flow). Generalising this, we have

$$\mathbb{I}_I^T = -\mathbb{I}_O. \tag{8}$$

In our example, therefore, we have

$$\mathbb{I}_O = \begin{bmatrix} -1 & -1 \\ 0 & 0 \end{bmatrix}, \text{ and } \mathbb{I}_I = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} = -\mathbb{I}_O^T$$

Defining $\mathbb{1}_n$ as an $n \times 1$ vector of ones, we then can calculate the total *outflow* for each product as

$$\text{out} = (\mathbf{P} \odot \mathbb{I}_O) \cdot \mathbb{1}_p, \tag{9}$$

where '$\odot$' represents componentwise multiplication. Similarly, the inflow for each product is expressed as

$$\text{in} = (\mathbb{1}_p^T \cdot (\mathbf{P} \odot \mathbb{I}_I))^T = (\mathbb{I}_I^T \odot \mathbf{P}^T) \cdot \mathbb{1}_p. \tag{10}$$

Therefore, the standardised change in balance $\mathbf{\Delta B_s} \in \mathbb{R}^{p\times 1}$ for each product is equal to

$$\begin{aligned}
\text{in} - \text{out} = \mathbf{\Delta B_s} &= (\mathbb{I}_I^T \odot \mathbf{P}^T) \cdot \mathbb{1}_p + (\mathbf{P} \odot \mathbb{I}_O) \cdot \mathbb{1}_p && (11)\\
&= (\mathbb{I}_I^T \odot \mathbf{P}^T - \mathbf{P} \odot \mathbb{I}_I^T) \cdot \mathbb{1}_p && (12)\\
&= (\mathbb{I}_I^T \odot \mathbf{P}^T - \mathbb{I}_I^T \odot \mathbf{P}) \cdot \mathbb{1}_p && (13)\\
&= (\mathbb{I}_I^T \odot (\mathbf{P}^T - \mathbf{P})) \cdot \mathbb{1}_p && (14)
\end{aligned}$$

To achieve the de-standardised balance delta, we use

$$\mathbf{\Delta B} = (\mathbf{\Delta B_s} \odot \sigma_B) + \mu_B, \tag{15}$$

where $\sigma_B \in \mathbb{R}^{p\times 1}$ and $\mu_B \in \mathbb{R}^{p\times 1}$ are vectors containing the standard deviations and means of each product respectively. Therefore, the sum-total balance of all of the products at time t, $\mathbf{\Sigma B}_t \in \mathbb{R}^1$ with respect to some previous starting balance $\mathbf{B}_{t-1} \in \mathbb{R}^{p\times 1}$ is equal to

$$\mathbf{\Sigma B}_t = \mathbb{1}_p^T \cdot (\mathbf{B}_{t-1} + \mathbf{\Delta B}_{t-1}) \tag{16}$$

0.2 Optimisation Formulation

The end goal here is to *maximise* the balance $\mathbf{B}_t$ by finding a vector $\mathbf{R}$ which is compliant to a set of constraints. Here, since equation (16) is linear in nature, this becomes a simple linear programming problem. We can use an off-the-shelf optimiser to find this optimum.

$$\begin{aligned}
\underset{\mathbf{R}_{ijk}}{\text{maximise}} \quad & \mathbf{\Sigma B}_t = \mathbb{1}_p^T \cdot (\mathbf{B}_{t-1} + \mathbf{\Delta B}_{t-1}) && (17)\\
\text{where} \quad & \mathbf{\Delta B}_{t-1} = ((\mathbf{P}^T - \mathbf{P}) \odot \mathbb{I}_O) \cdot \mathbb{1}_p \odot \sigma_B) + \mu_B && (18)\\
\text{and} \quad & \mathbf{P} = \mathbf{C}_{\mathbf{F}_{ijk}} \cdot \mathbf{F}^{T_{jk}} + \mathbf{C}_{\mathbf{R}_{ijk}} \cdot \mathbf{R}^{T_{jk}} + \mathbf{C_0} && (19)\\
\text{such that} \quad & \mathbf{R}_{ij1} \geq 0 \quad \forall i, j && (20)
\end{aligned}$$

where (20) ensures that no rate is less than zero. Note that the solutions for $\mathbf{R}$ will need to be de-standardised to achieve the raw values.

0.3 Example

Say we have three products, A, B and C. C is external. We have modelled the following flows:

- $A \to B$
- $B \to A$
- $A \to C$
- $C \to A$
- $B \to C$
- $C \to B$

Say, for example, that we only have one rate-related feature (rate) with coefficient matrix

$$\mathbf{C_R} = \begin{bmatrix} 0.0 & 1.0 & 1.0 \\ 2.0 & 0.0 & 2.0 \\ 3.0 & 3.0 & 0.0 \end{bmatrix} \tag{21}$$

Suppose that we have two non rate-related features, with coefficients and values

$$\mathbf{C_F} = \begin{bmatrix} \begin{pmatrix} 0 & 1 & 2 \\ 3 & 0 & 1 \\ 2 & 3 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 3 & 2 \\ 1 & 0 & 0 \\ 3 & 2 & 0 \end{pmatrix} \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} \begin{pmatrix} 0 & 1 & 2 \\ 3 & 0 & 1 \\ 2 & 3 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 2 & 1 \\ 1 & 0 & 3 \\ 3 & 1 & 0 \end{pmatrix} \end{bmatrix} \tag{22}$$

Assume for simplicity that $\mathbf{C_0} = \mathbf{0}$.

$$\mathbf{C_F} \cdot \mathbf{F}^{T_{jk}} = \begin{bmatrix} 0 & (1\cdot 1)+(3\cdot 2) & (2\cdot 2)+(2\cdot 1) \\ (3\cdot 3)+(1\cdot 1) & 0 & (1\cdot 1)+(0\cdot 3) \\ (2\cdot 2)+(3\cdot 3) & (3\cdot 3)+(2\cdot 1) & 0 \end{bmatrix} = \begin{bmatrix} 0 & 7 & 6 \\ 10 & 0 & 1 \\ 13 & 11 & 0 \end{bmatrix} \tag{23}$$

$$\mathbb{I}_O = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{24}$$

$$\mathbf{P} = \begin{bmatrix} 0 & 7 & 6 \\ 10 & 0 & 1 \\ 13 & 11 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 1 \\ 2 & 0 & 2 \\ 3 & 3 & 0 \end{bmatrix} \odot \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{25}$$

```python
import cvxpy as cp
import numpy as np

# Define the dimensions
n = 3  # assuming 3x3 matrices for simplicity

# Define symbolic variables for r_ij
r = cp.Variable((3,3))

# Define the given matrices
P1 = np.array([[0, 7, 6], [10, 0, 1], [13, 11, 0]])
P2 = np.array([[0, 1, 1], [2, 0, 2], [3, 3, 0]])

# Matrix I_O
I_O = np.array([[1, 1, 1], [1, 1, 1], [0, 0, 0]])

# Compute the element-wise product P2 x R
P2_elemwise_R = cp.multiply(P2, r)

# Compute matrix P
P = P1 + P2_elemwise_R

# Compute P^T - P
P_T_minus_P = P.T - P

# Apply element-wise product with I_O (Hadamard product)
P_T_minus_P_I_O = cp.multiply(P_T_minus_P, I_O)

# Assume that B_t-1 is known or is a decision variable (for now, we define it as a
    zero matrix)
B_t_minus_1 = np.zeros((n, n))

# Define identity matrix I_p for the projection (for simplicity, I_p is an identity
    matrix here)
I_p = np.ones(n)
# Define B_t as the expression in the problem
B_t_expr = I_p.T @ (B_t_minus_1 + P_T_minus_P_I_O) @ I_p

# Define the optimization problem (if solving for r_ij's)
# For example, let's minimize the Frobenius norm of B_t (or any other objective based
    on the problem)
objective = cp.Maximize(B_t_expr)

# Define constraints (if any)
constraints = [r<=1.05,
               *[r[i][j] >= 1.01 if i != j else r[i][j]==0 for i in range(3) for j in
    range(3)] # diagonals are 0.0 rate as we aren't modelling the flow from one product
     to itself(?)
               ]

# Create the problem
problem = cp.Problem(objective, constraints)

# Solve the problem
problem.solve()
```

```
51
52 # Output the results
53 print("Optimal value of B_t:", B_t_expr.value)
54 print("Optimal values of r_ij:")
55 for i in range(n):
56     for j in range(n):
57         print(f"r_{i+1}{j+1} = {round(r[i][j].value,3)}")
58
59 >>OUTPUT
60 Optimal value of B_t: 20.27000000099588
61 Optimal values of r_ij:
62 r_11 = -0.0
63 r_12 = 1.03
64 r_13 = 1.01
65 r_21 = 1.03
66 r_22 = -0.0
67 r_23 = 1.01
68 r_31 = 1.05
69 r_32 = 1.05
70 r_33 = -0.0
```