

# DYNAMIC PROGRAMMING USING THE SIMUSAGE COMPONENT LIBRARY AND ITS APPLICATION TO THE SIMULATION OF THE CEMENT CLINKER BURNING PROCESS



# Contents

- 1. Dynamic programming – Definition and introduction**
- 2. Development of SimuSage based components**
- 3. Cement clinker burning process**
- 4. Simulation purpose and layout**
- 5. Simulation results**
- 6. Outlook**

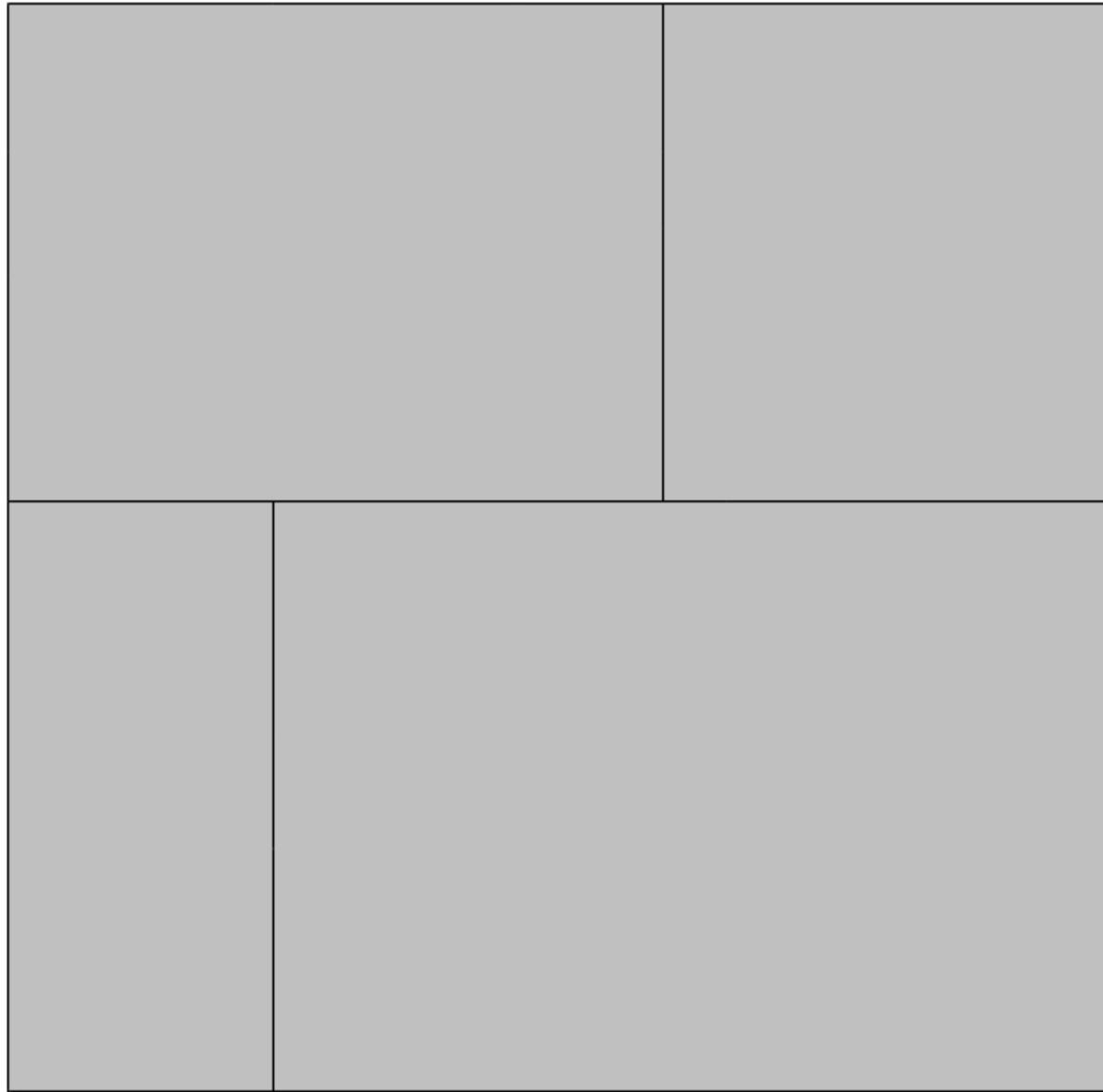
# 1. Dynamic programming

## Mathematics and computer science

- **Definition**
  - Method of solving problems
  - Overlapping sub problems
  - Optimal substructure
  
- **Algorithm**

# 1. Dynamic programming

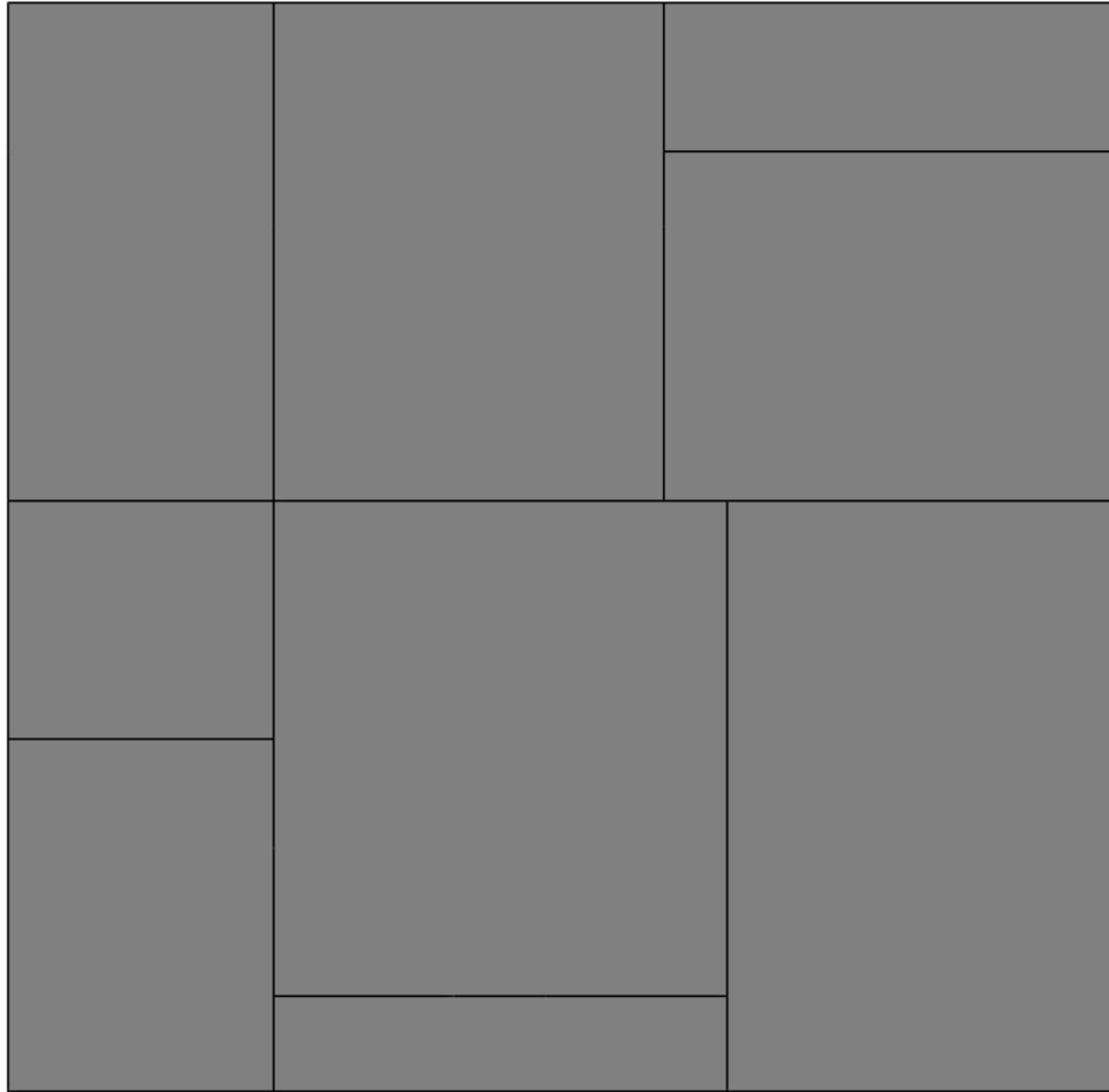
## Mathematics and computer science



- **Definition**
  - Method of solving problems
  - Overlapping sub problems
  - Optimal substructure
  
- **Algorithm**
  - Break the problem into smaller sub problems

# 1. Dynamic programming

## Mathematics and computer science



### ■ Definition

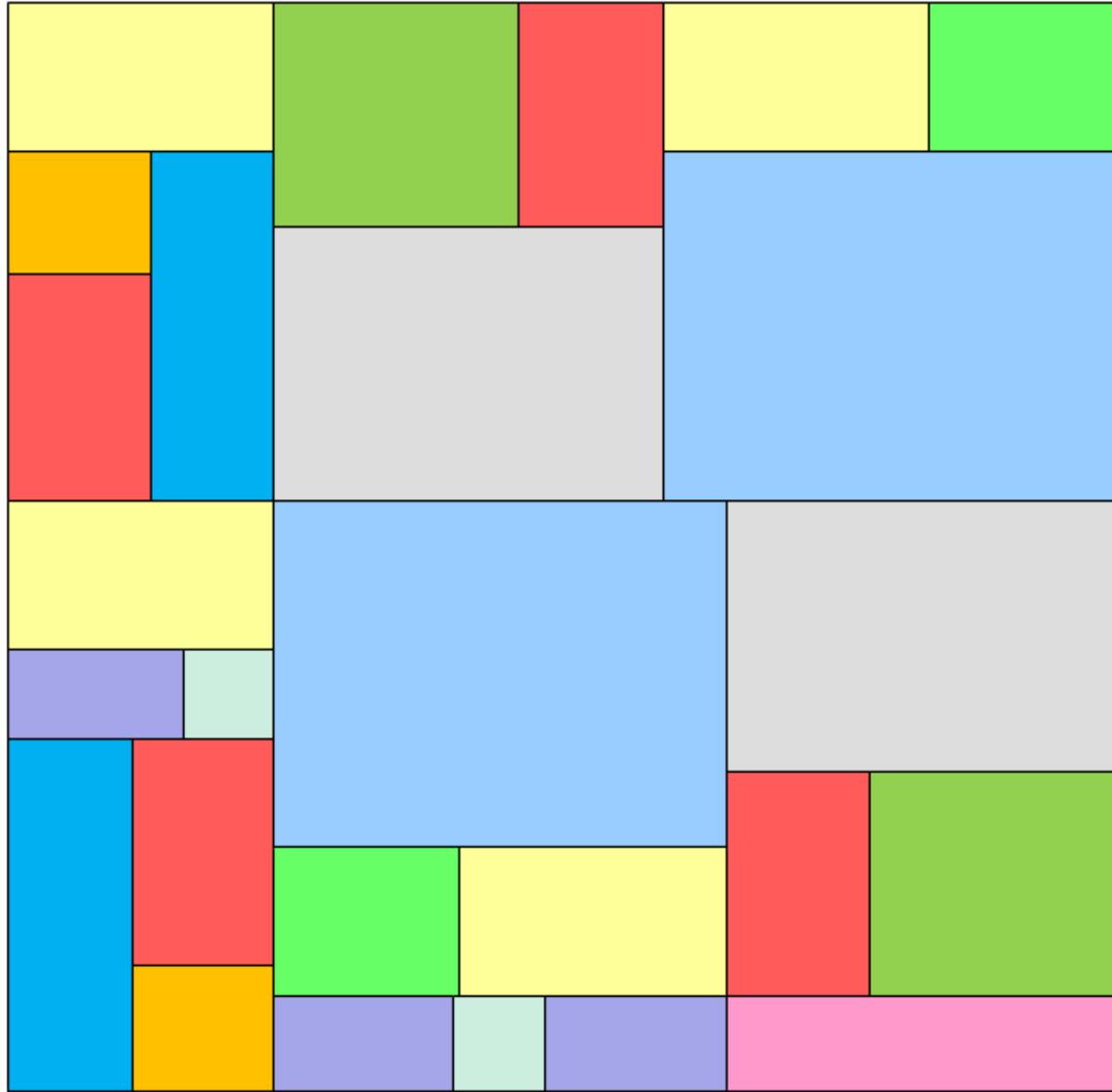
- Method of solving problems
- Overlapping sub problems
- Optimal substructure

### ■ Algorithm

- Break the problem into smaller sub problems

# 1. Dynamic programming

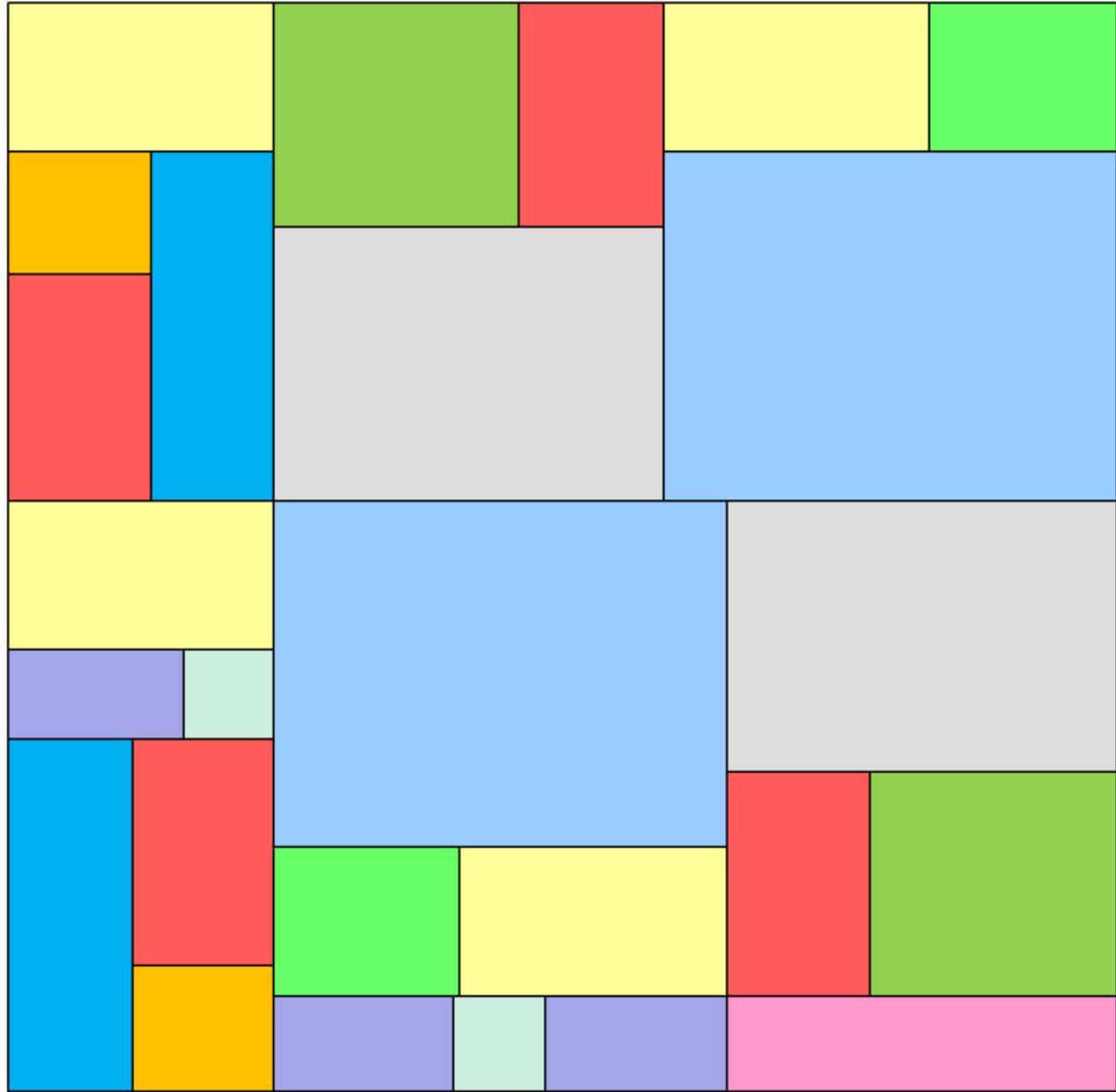
## Mathematics and computer science



- **Definition**
  - Method of solving problems
  - Overlapping sub problems
  - Optimal substructure
  
- **Algorithm**
  - Break the problem into smaller sub problems

# 1. Dynamic programming

## Mathematics and computer science



- **Definition**
  - Method of solving problems
  - Overlapping sub problems
  - Optimal substructure
  
- **Algorithm**
  - Break the problem into smaller sub problems
  - Solve these simple problems
  - Use these optimal solutions to construct an optimal solution for the original problem

# 1. Dynamic programming

## Mathematics and computer science – Fibonacci numbers

$$F(n) = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F(n - 1) + F(n - 2) & \text{if } n > 1. \end{cases}$$

```
function Fib(n: Integer): Integer;
begin
  if n = 0 then
    Result:= 0
  else if n = 1 then
    Result:= 1
  else
    Result:= Fib(n - 1) + Fib(n - 2);
end;
```

$$F(5) = F(4) + F(3)$$

$$F(4) = \color{red}{F(3)} + F(2)$$

$$F(4) = (\color{red}{F(2) + F(1)}) + (F(1) + F(0))$$

$$F(4) = ((\color{red}{F(1) + F(0)}) + F(1)) + (F(1) + F(0))$$

### ■ Definition

- Method of solving problems
- Overlapping sub problems
- Optimal substructure

### ■ Algorithm

- Break the problem into smaller sub problems
- Solve these simple problems
- Use these optimal solutions to construct an optimal solution for the original problem

# 1. Dynamic programming

## Programming language

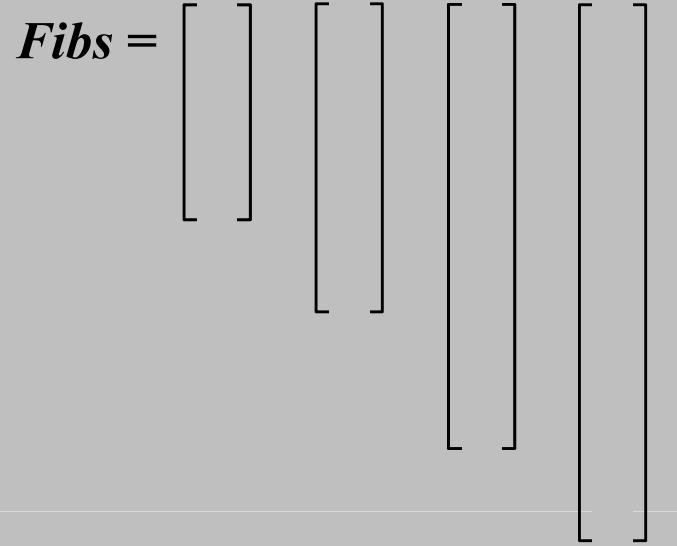
```
procedure CalcAllFibs();
var Fibs: array of Integer;
    n, i: Integer;

begin
    GetN(n);           → Data input for the number of Fibonacci numbers
    SetLength(Fibs, n); → Dynamic allocation of memory
    Fibs[0]:= 0;
    Fibs[1]:= 1;
    for i:= 2 to (n - 1) do
        Fibs[i]:= Fibs[i - 1] + Fibs[i - 2];
    WriteFibs(Fibs);
end;
```

Data input for the number of Fibonacci numbers

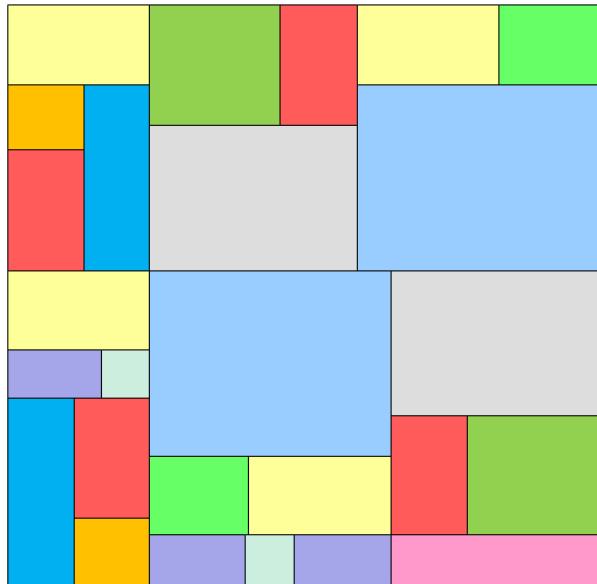
Dynamic allocation of memory

Output

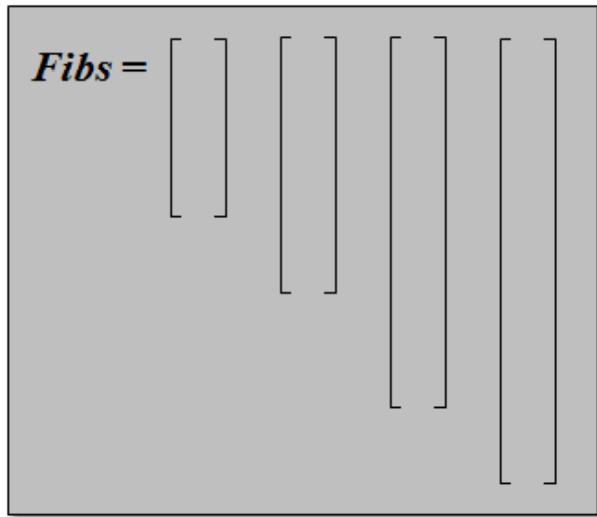


# 1. Dynamic programming

## Application for process simulation



- Break the process into sub-processes
- Break sub-processes into characteristic process steps
- Develop models for these process steps



- Generate components for process steps
- Create and connect process steps according to the total process model
- Solve the model by an iterative procedure

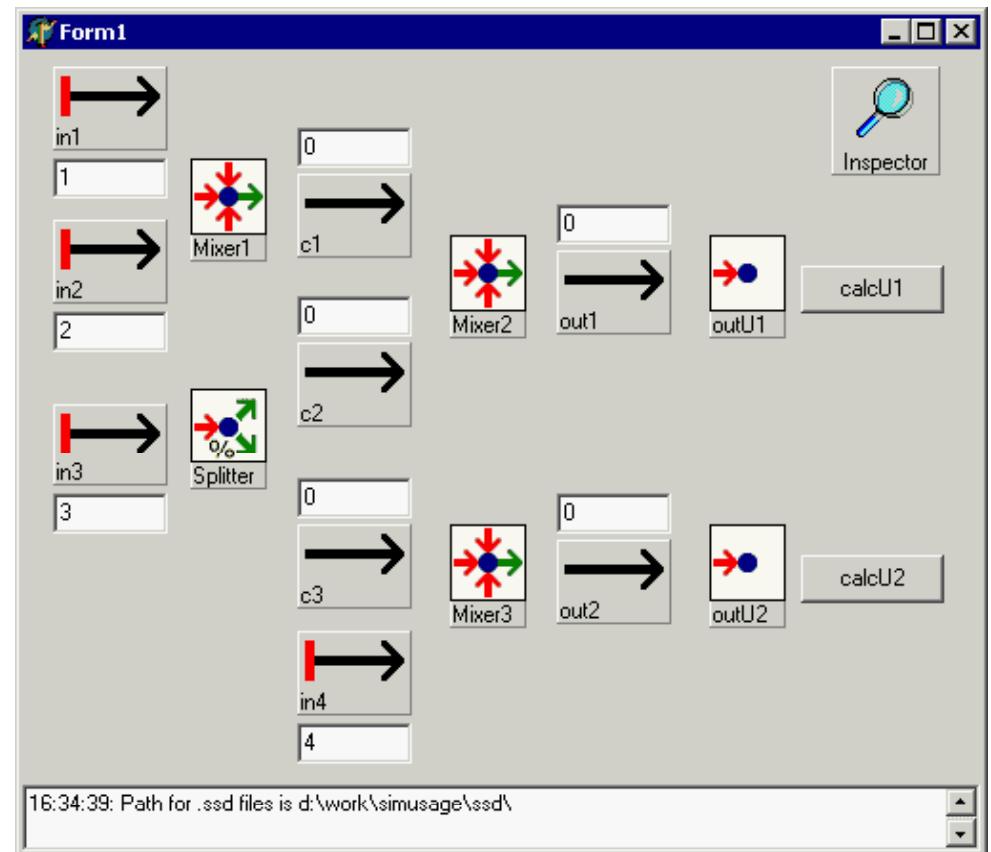
## 2. Development of SimuSage based components

### Intorduction



- Delphi components
  - Properties
  - Methods
  - Events
- TPbGttBalance 

  - Temperature
  - calculate
  - OnBeforeCalculation

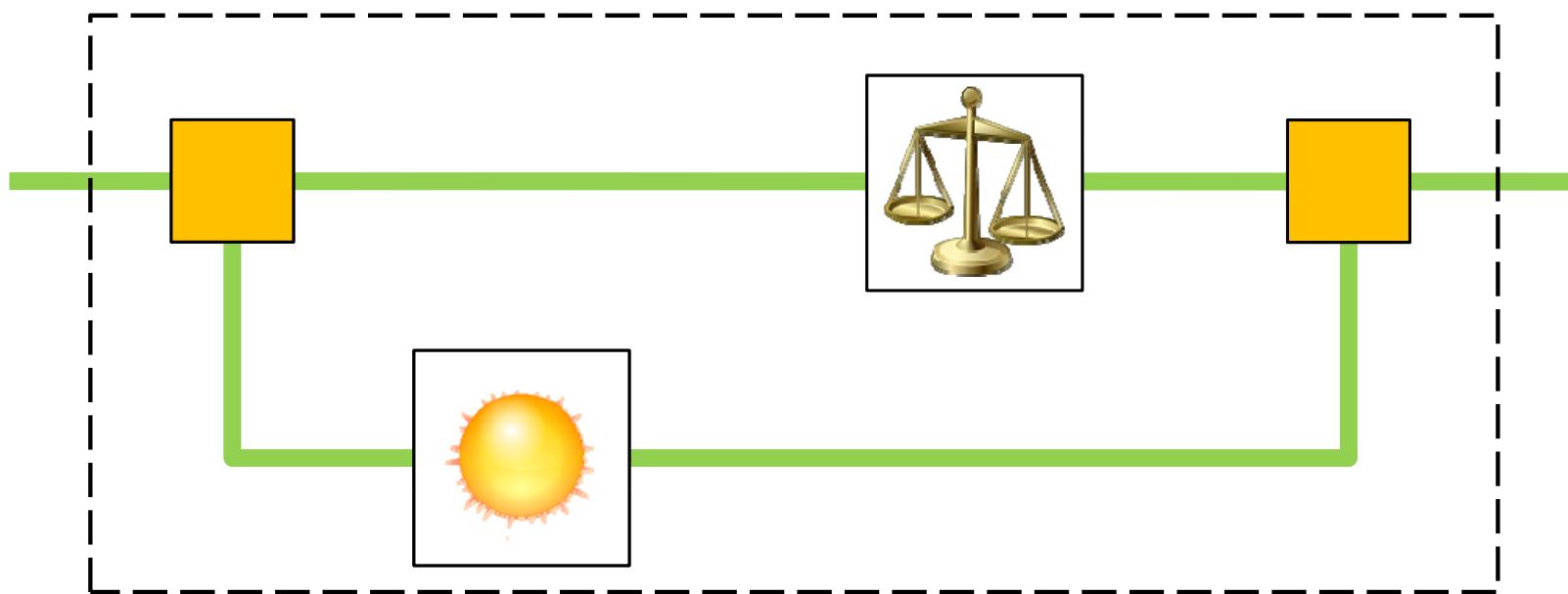


## 2. Development of SimuSage based components

### Partial equilibrium reactor

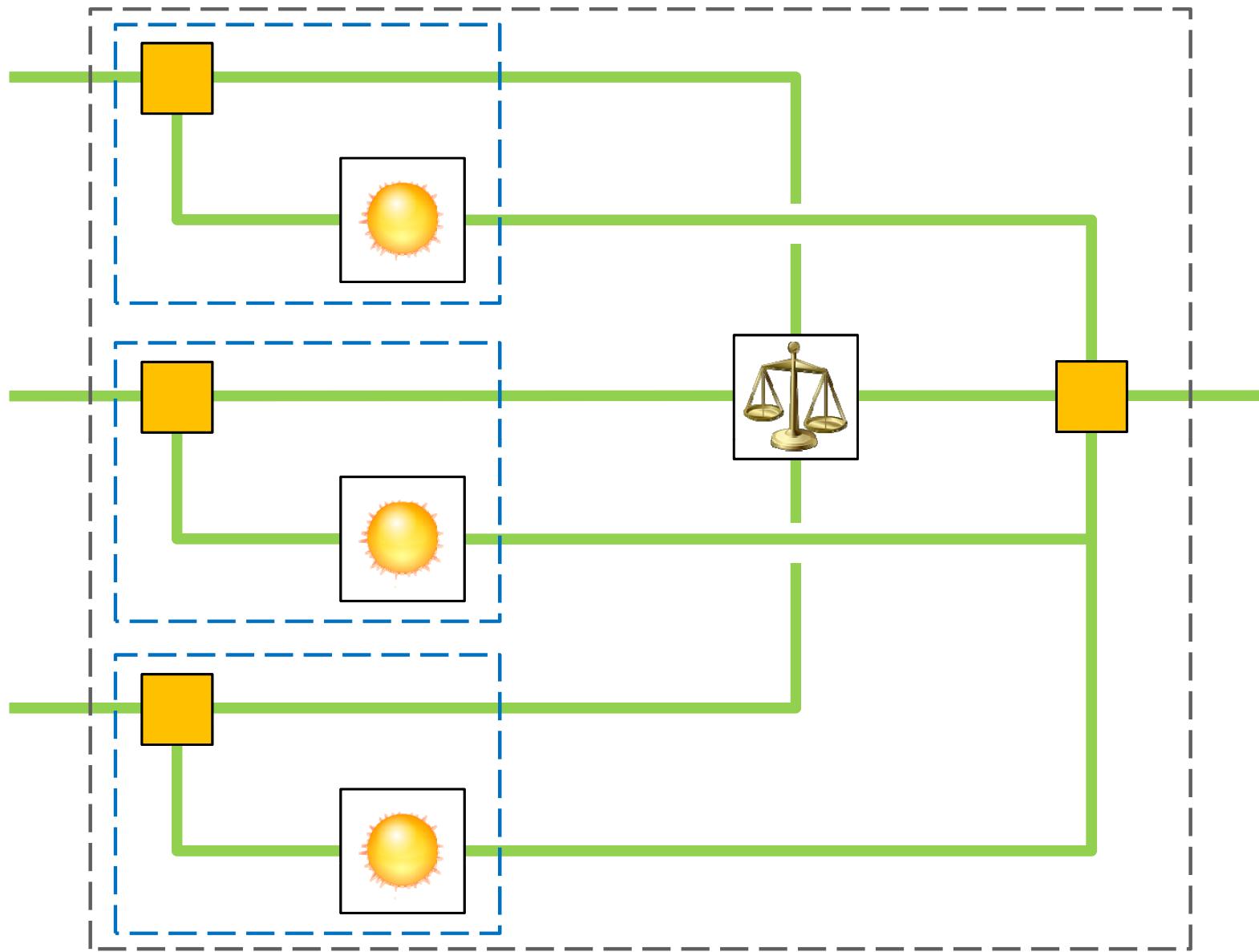
#### ■ Parameters and specifications

- Partial thermodynamic equilibrium
- Model for the amount of 'non-equilibrium'
- Defined enthalpy input
- Homogeneous temperature of outgoing stream

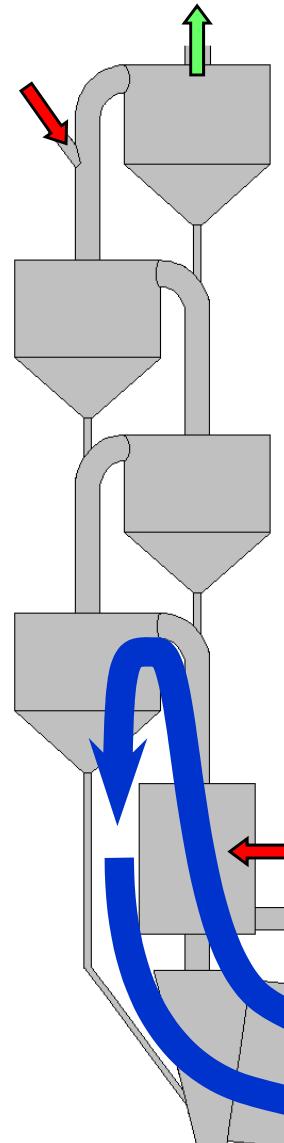


## 2. Development of SimuSage based components

### Partial equilibrium reactor – multiple input streams



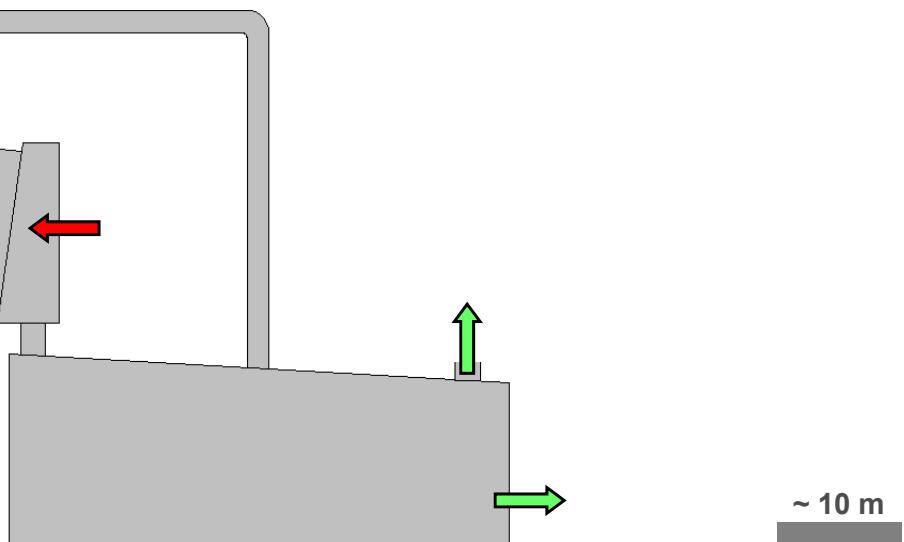
### 3. Cement clinker burning process



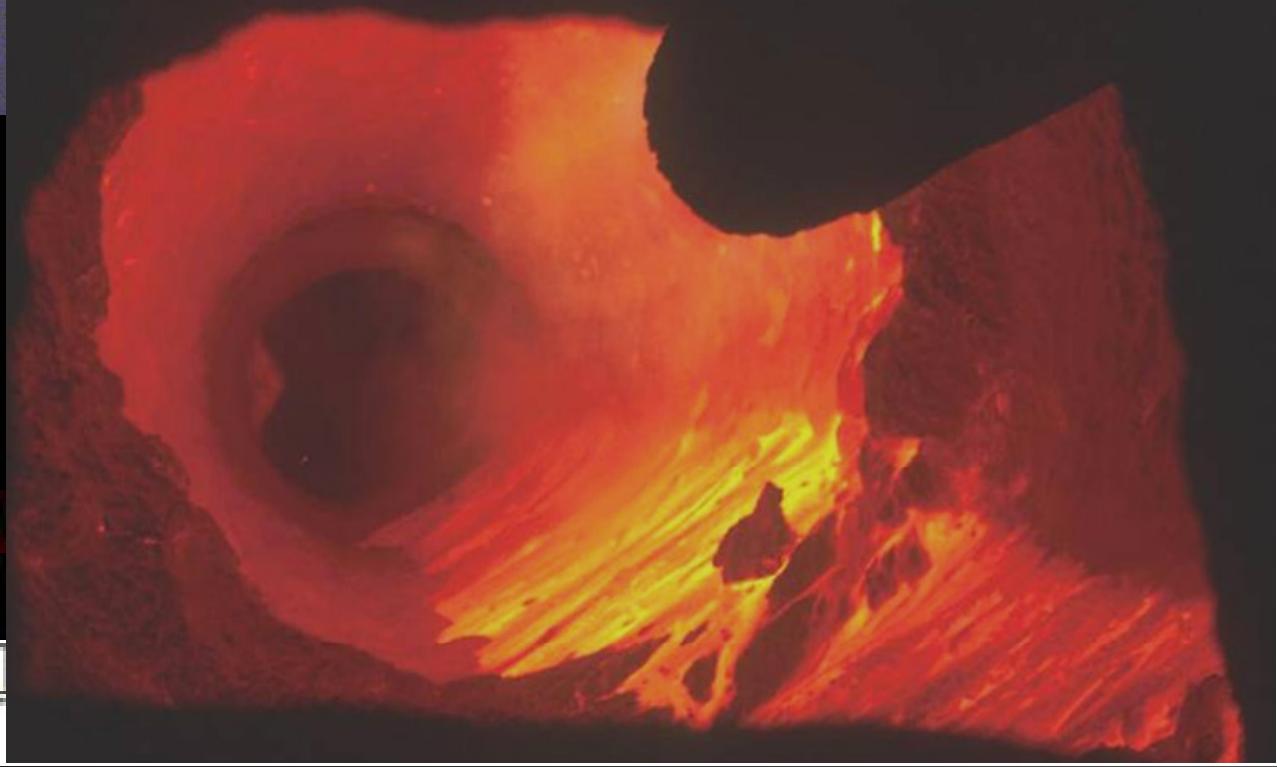
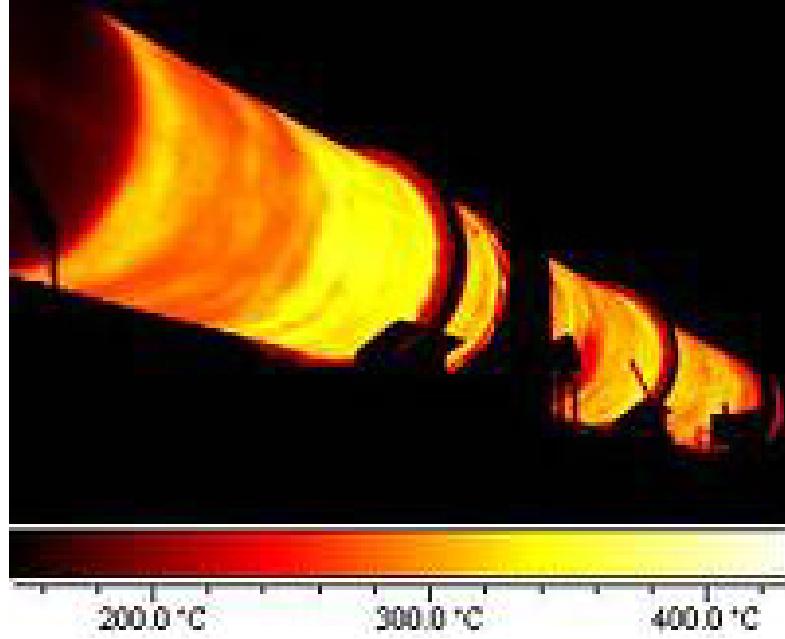
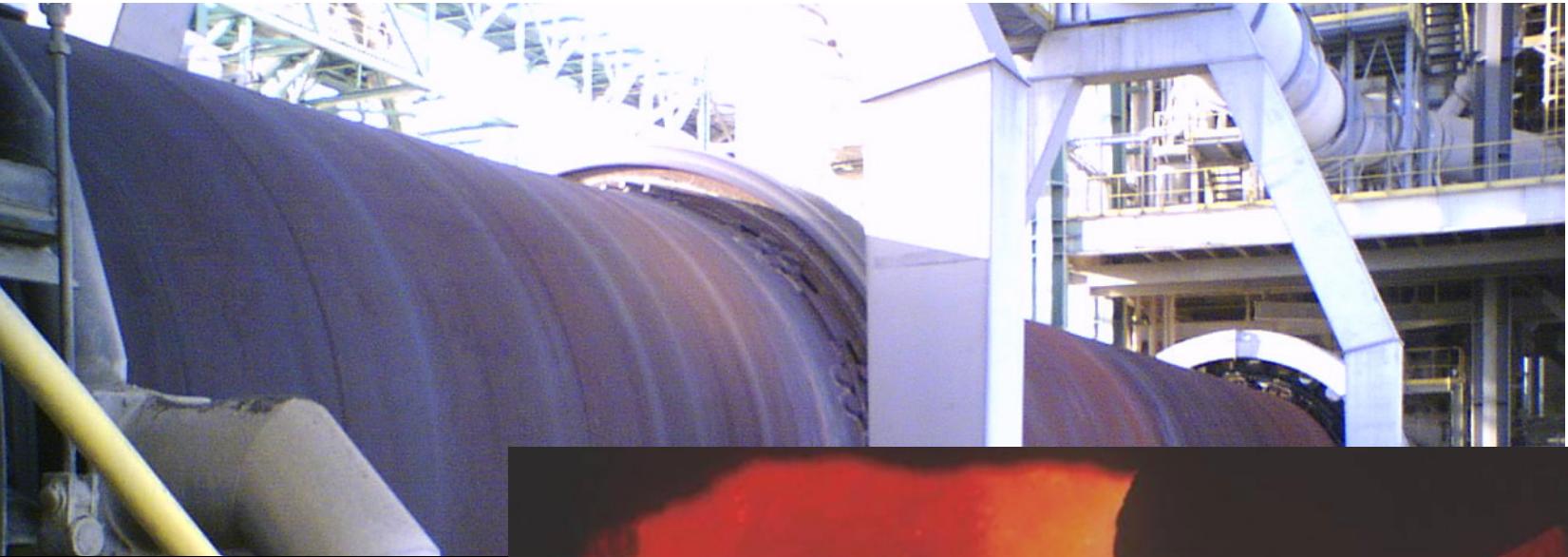
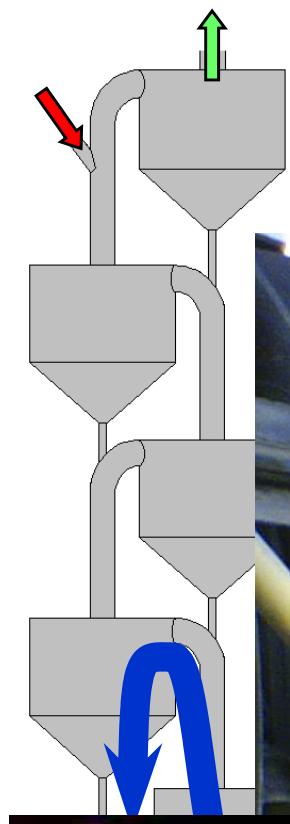
- Typical plant layout

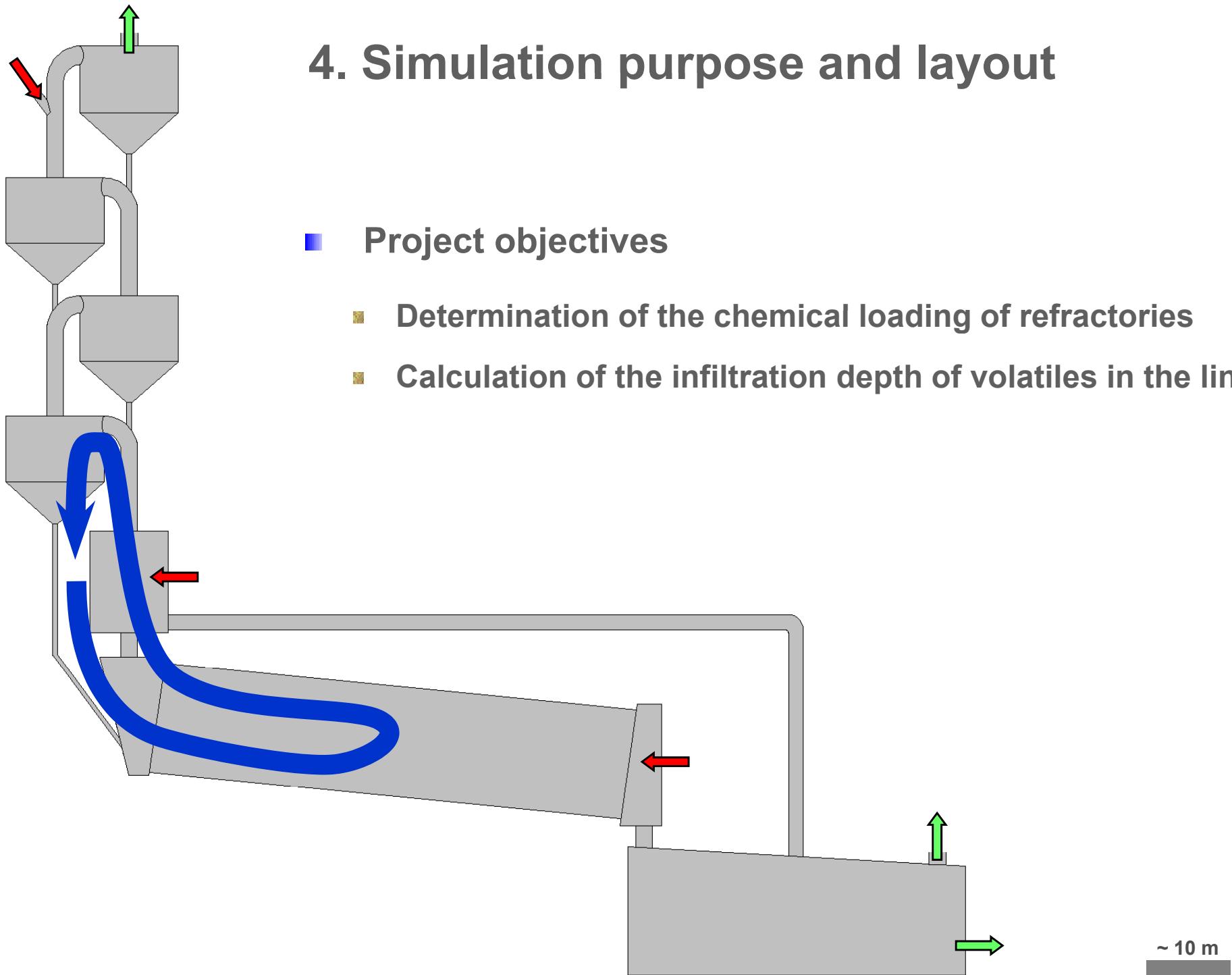
- Raw mix: 250 t/h
- Heat: 2.900 – 3.400 kJ/kg clinker
- ← Clinker: 150 t/h
- ← Off gas: 220.000 m<sub>N</sub><sup>3</sup>/h

- Volatile recirculation



### 3. Cement clinker burning process

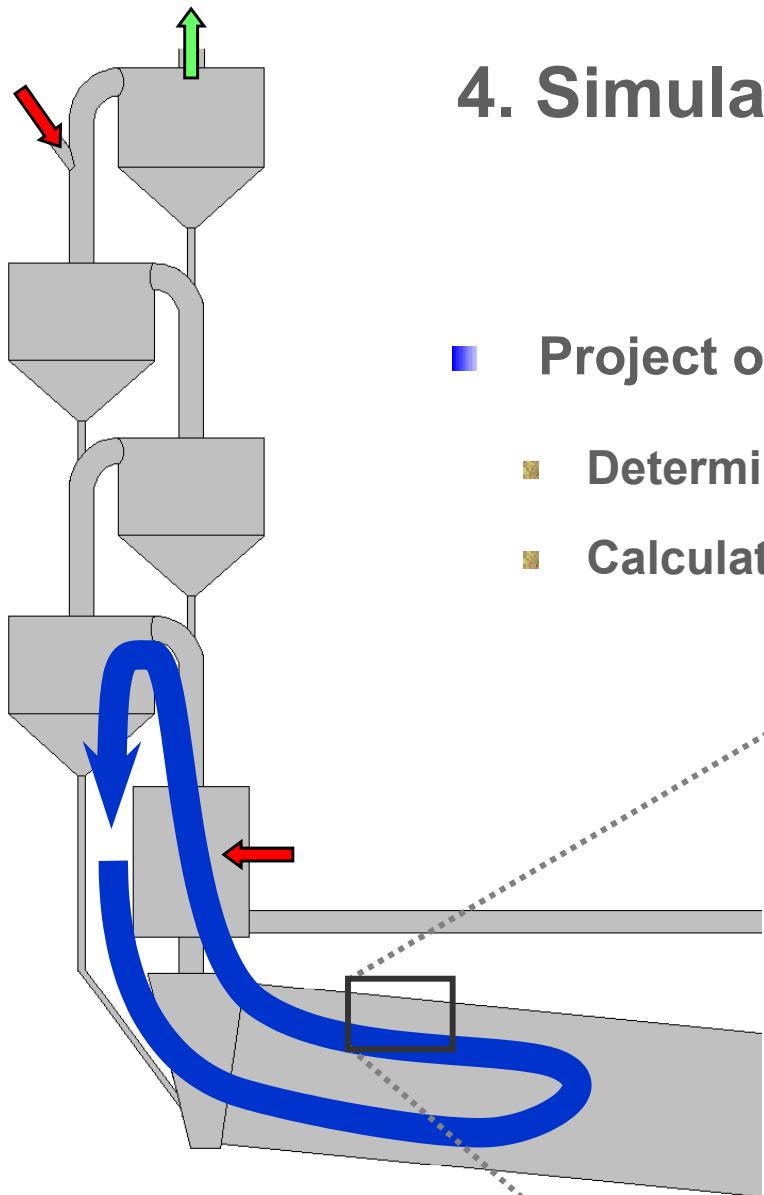




## 4. Simulation purpose and layout

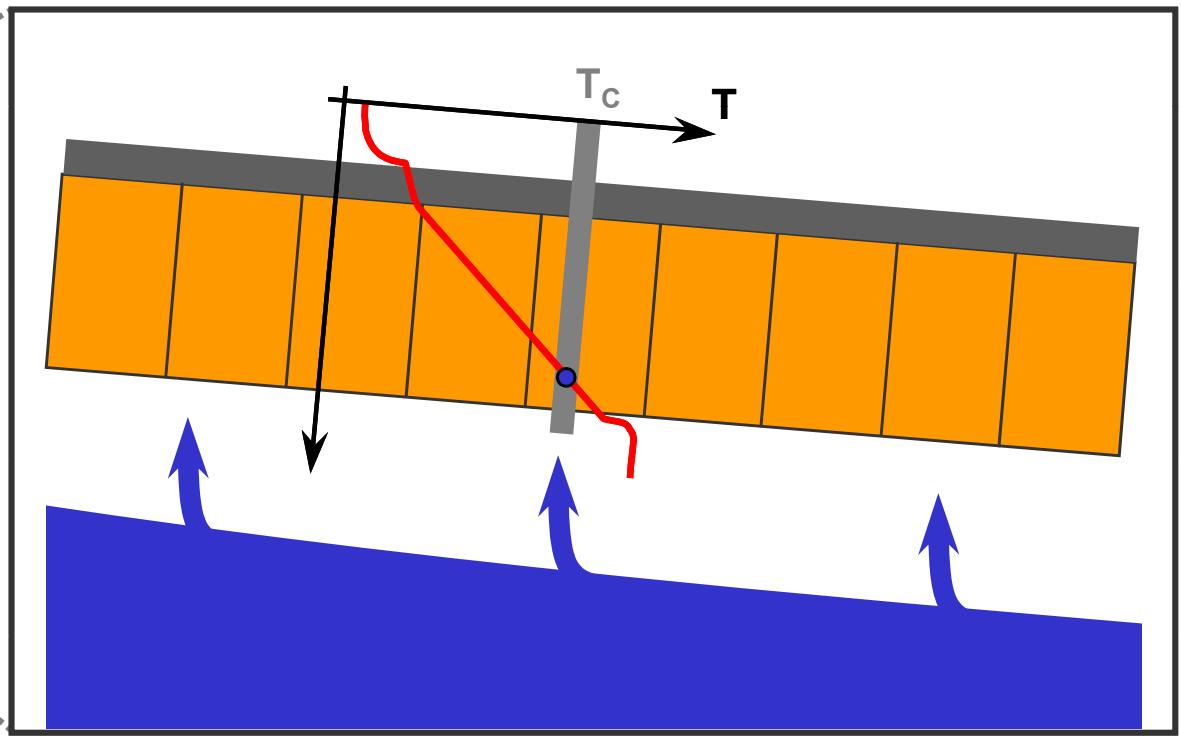
### ■ Project objectives

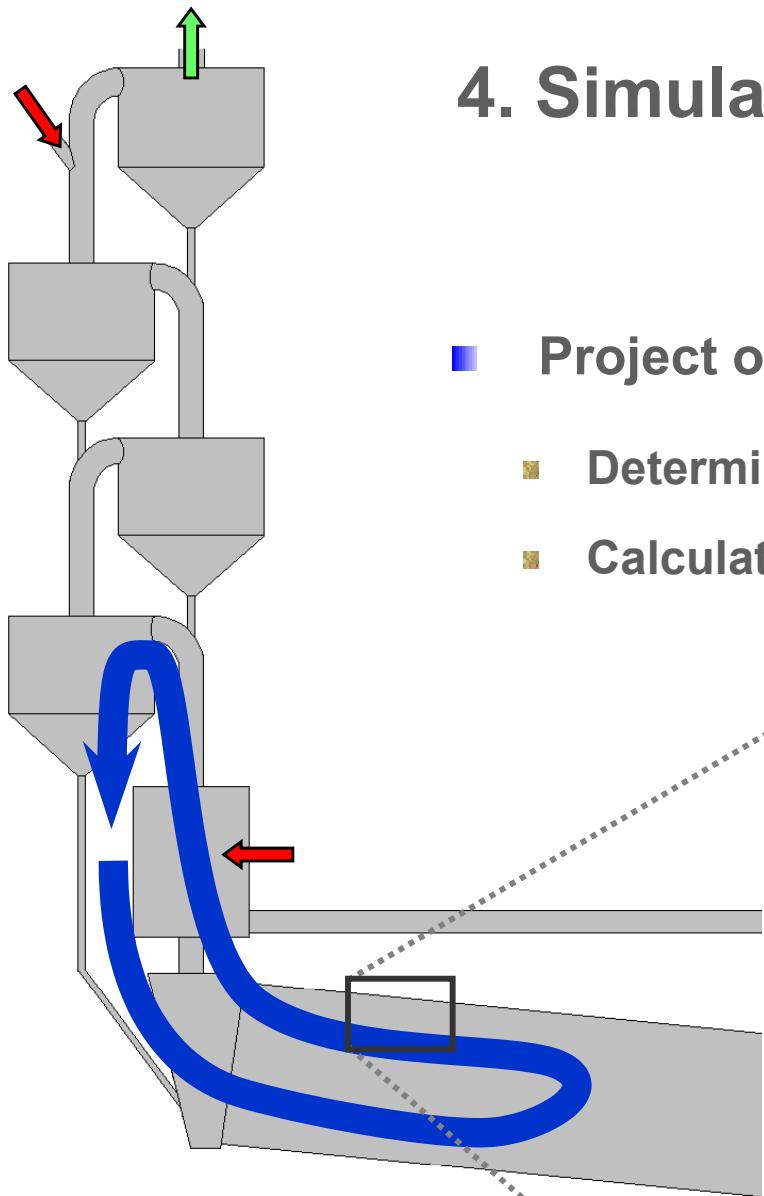
- Determination of the chemical loading of refractories
- Calculation of the infiltration depth of volatiles in the lining



## 4. Simulation purpose and layout

- Project objectives
  - Determination of the chemical loading of refractories
  - Calculation of the infiltration depth of volatiles in the lining

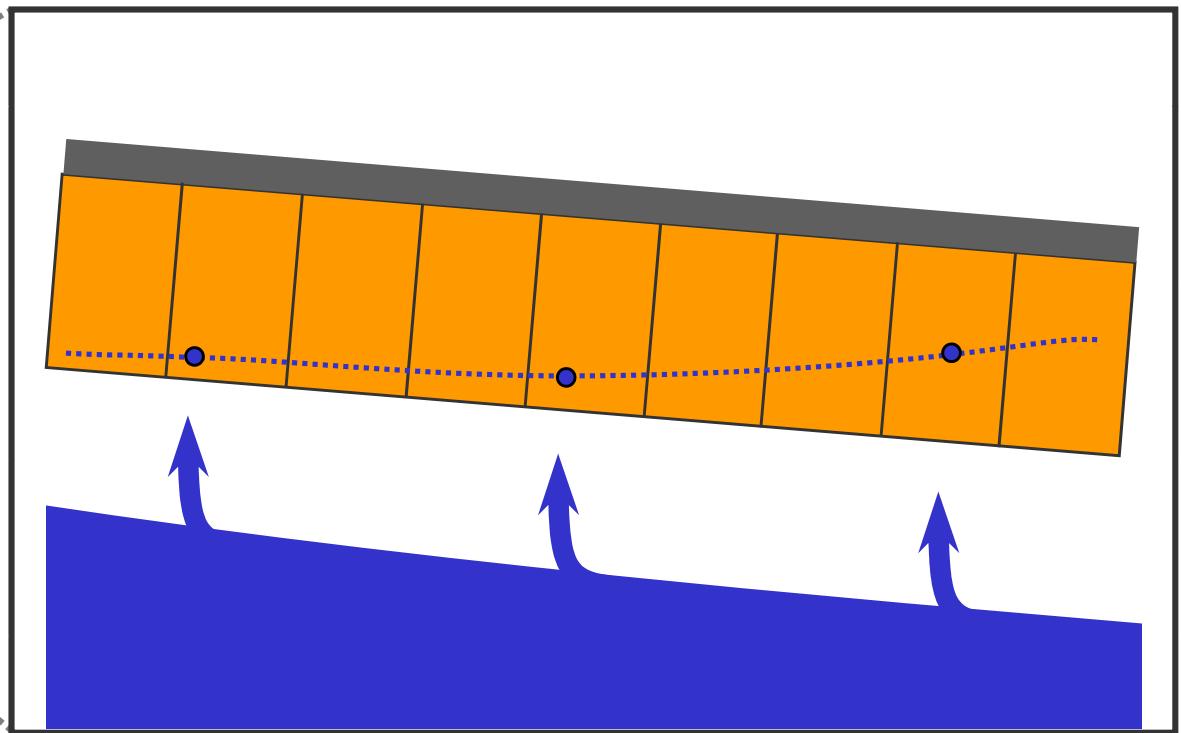




## 4. Simulation purpose and layout

### ■ Project objectives

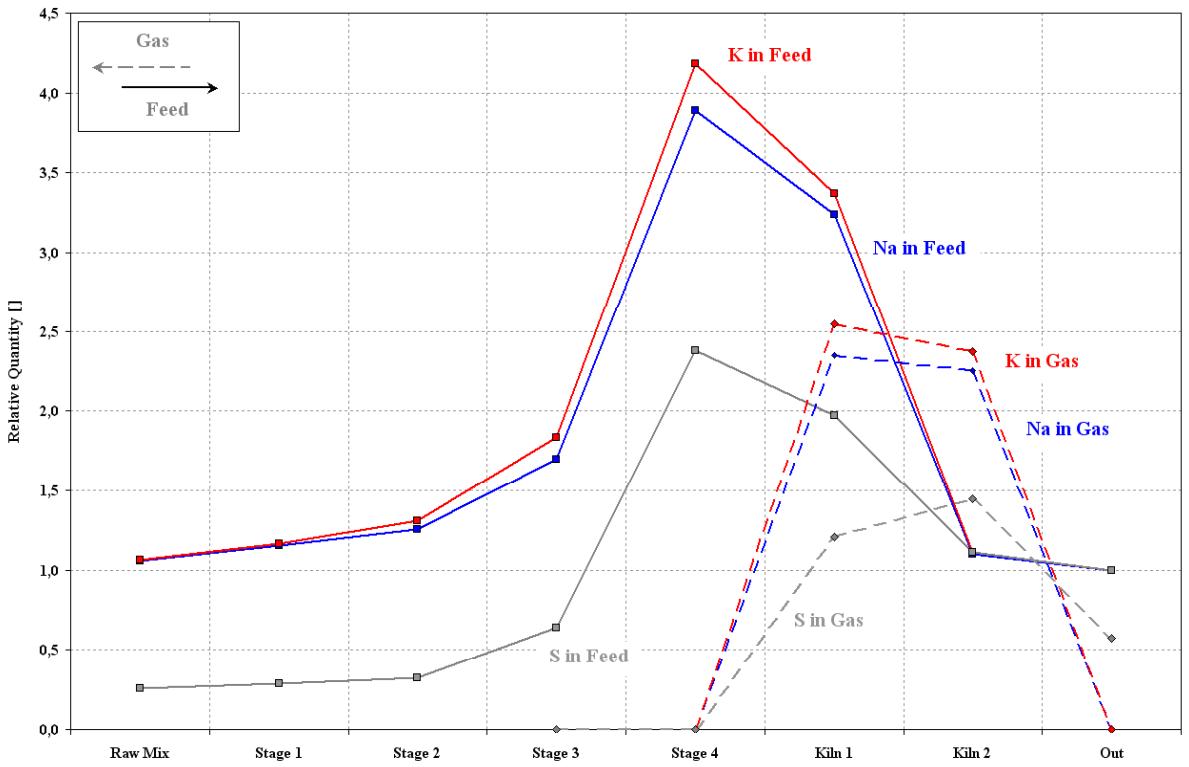
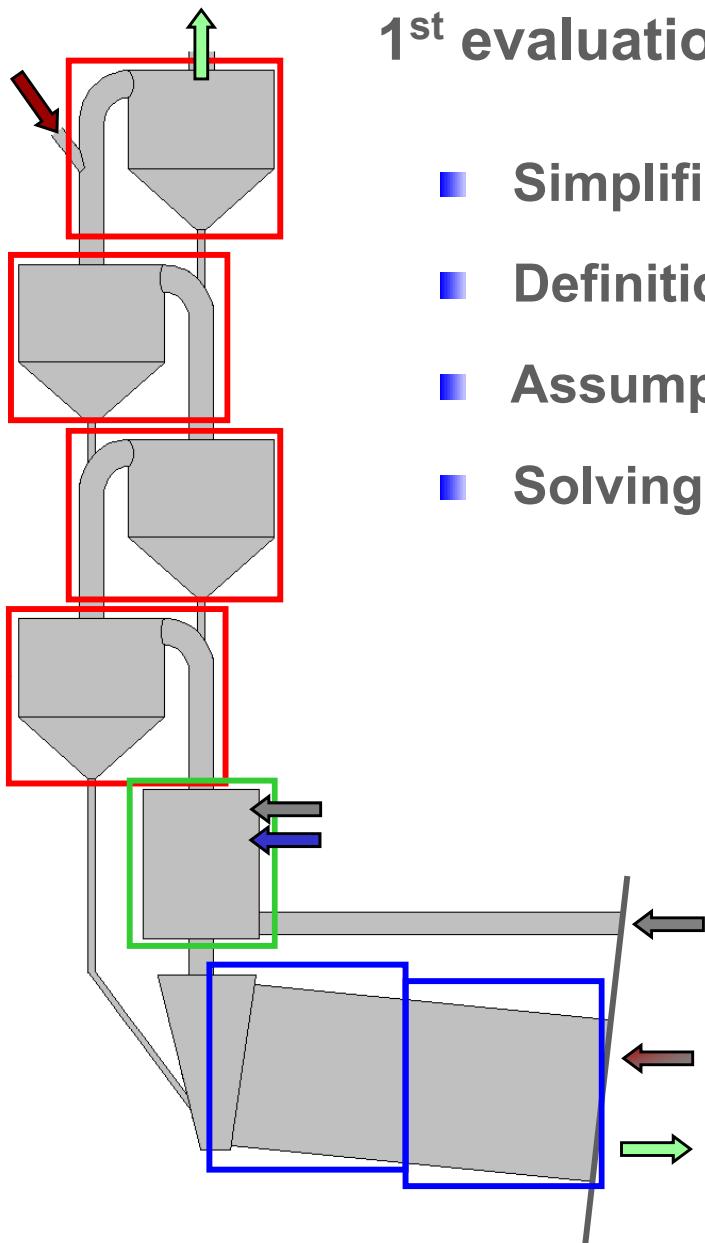
- Determination of the chemical loading of refractories
- Calculation of the infiltration depth of volatiles in the lining



# 4. Simulation purpose and layout

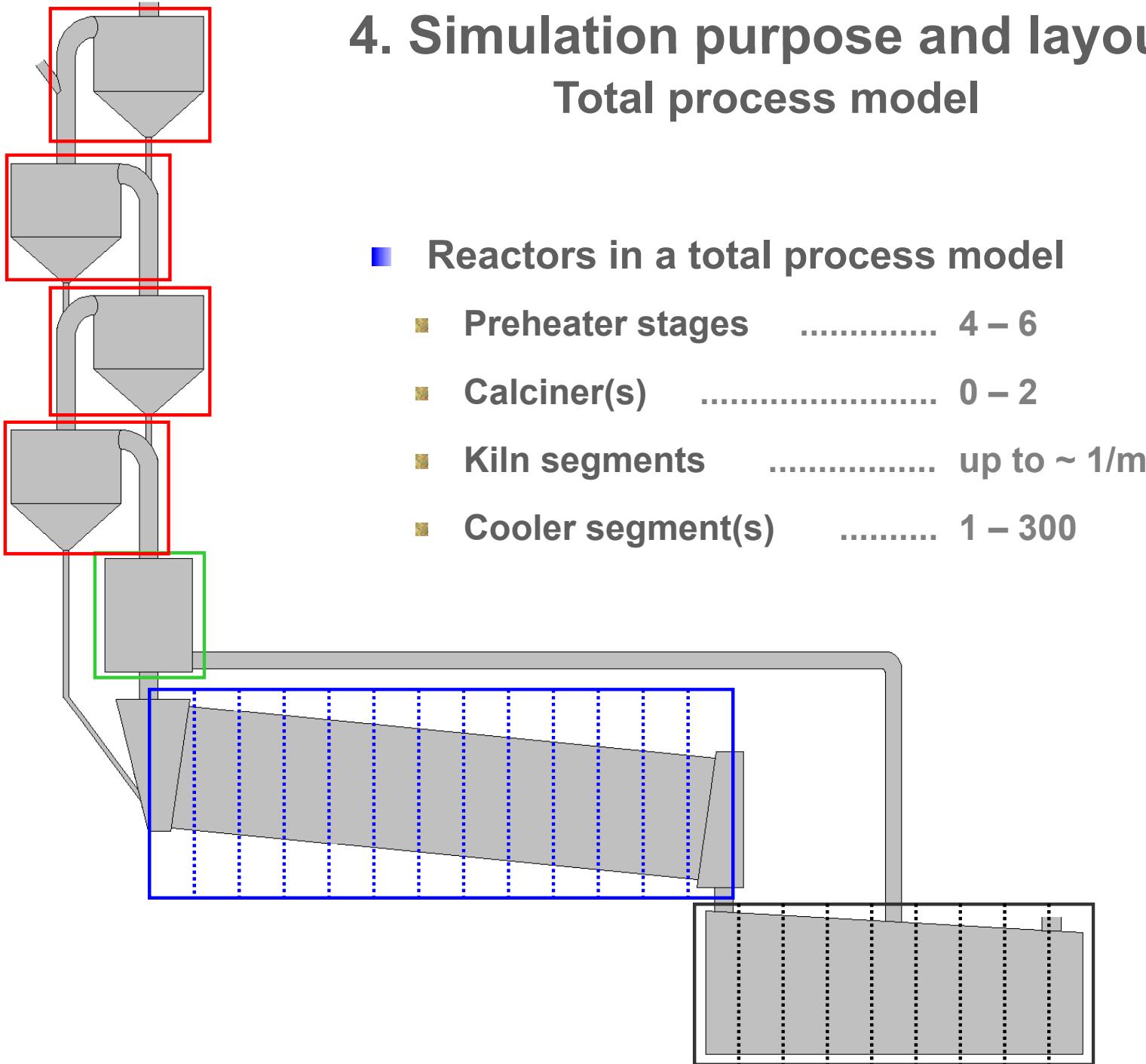
## 1<sup>st</sup> evaluation with focus on the recirculation

- Simplified setup of reactors
- Definition of input streams
- Assumptions for further input streams
- Solving mass and energy balances



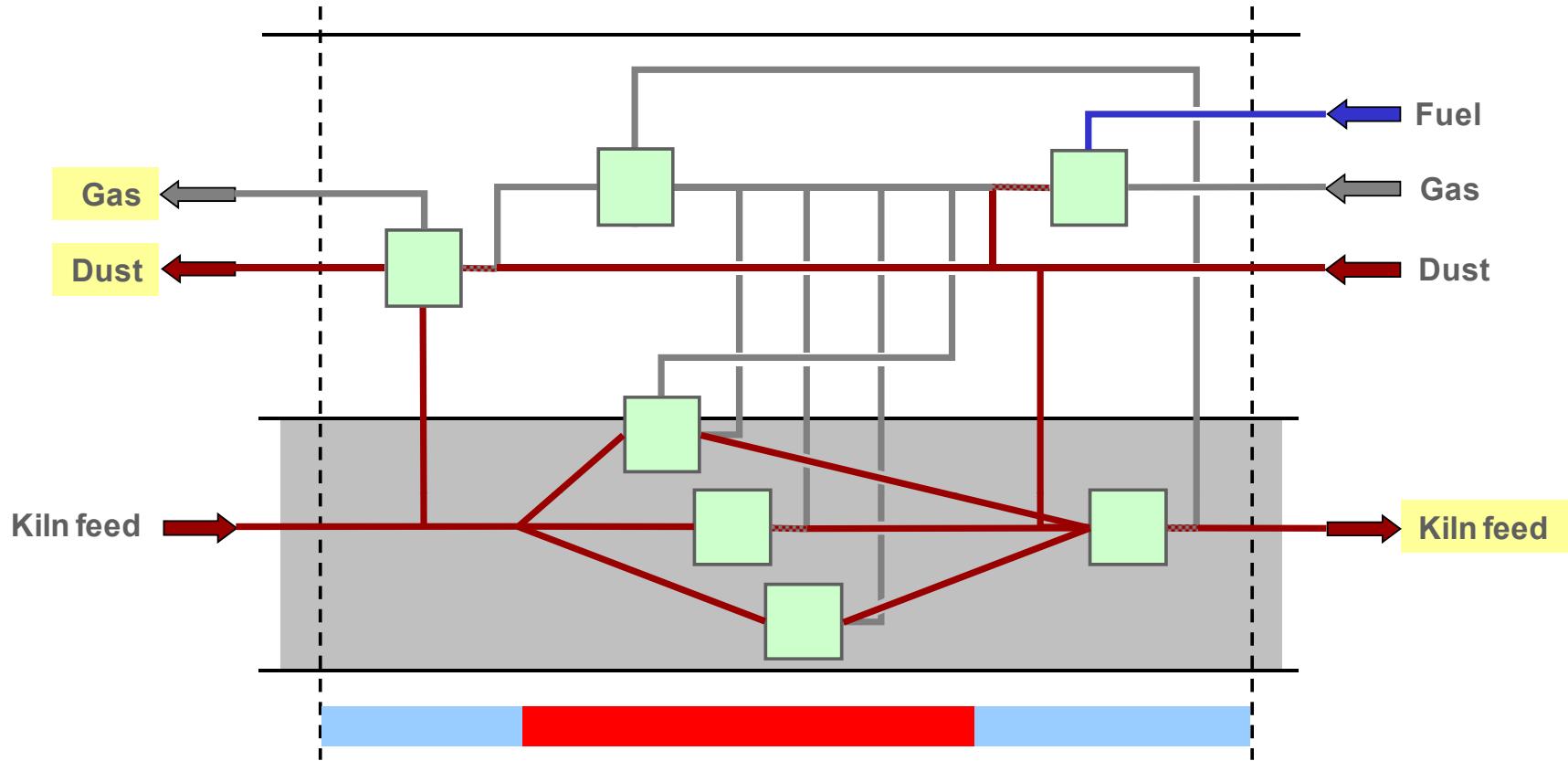
## 4. Simulation purpose and layout

### Total process model



## 4. Simulation purpose and layout

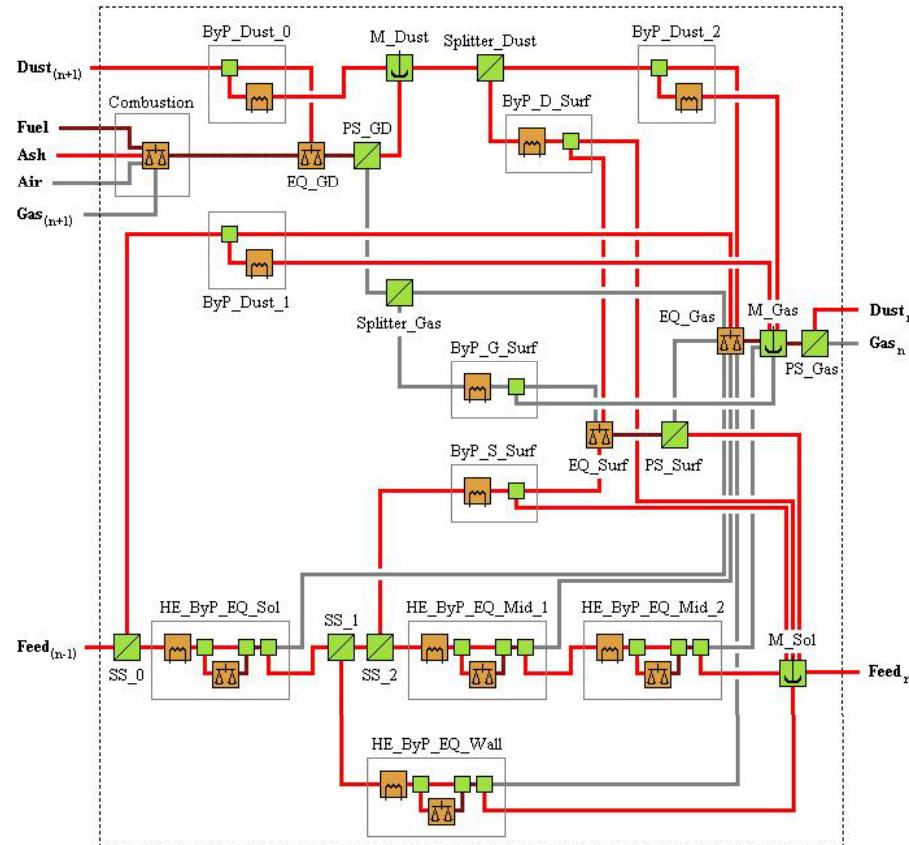
### Kiln segment model



- Material transport according to Kramers and Crookewitt (1952)
- Heat transfer according to Frisch and Jeschar (1983)
- Separation of feed into segments according to temperature distribution

# 4. Simulation purpose and layout

## Kiln segment model



- Material transport according to Kramers and Crookewitt (1952)
- Heat transfer according to Frisch and Jeschar (1983)
- Separation of feed into segments according to temperature distribution

# 5. Simulation results

## Simulation of a real-life kiln

### ■ Setup of reactors

- 4 Preheater stages
- 1 Calciner
- 35 Kiln segments (à 2 m)

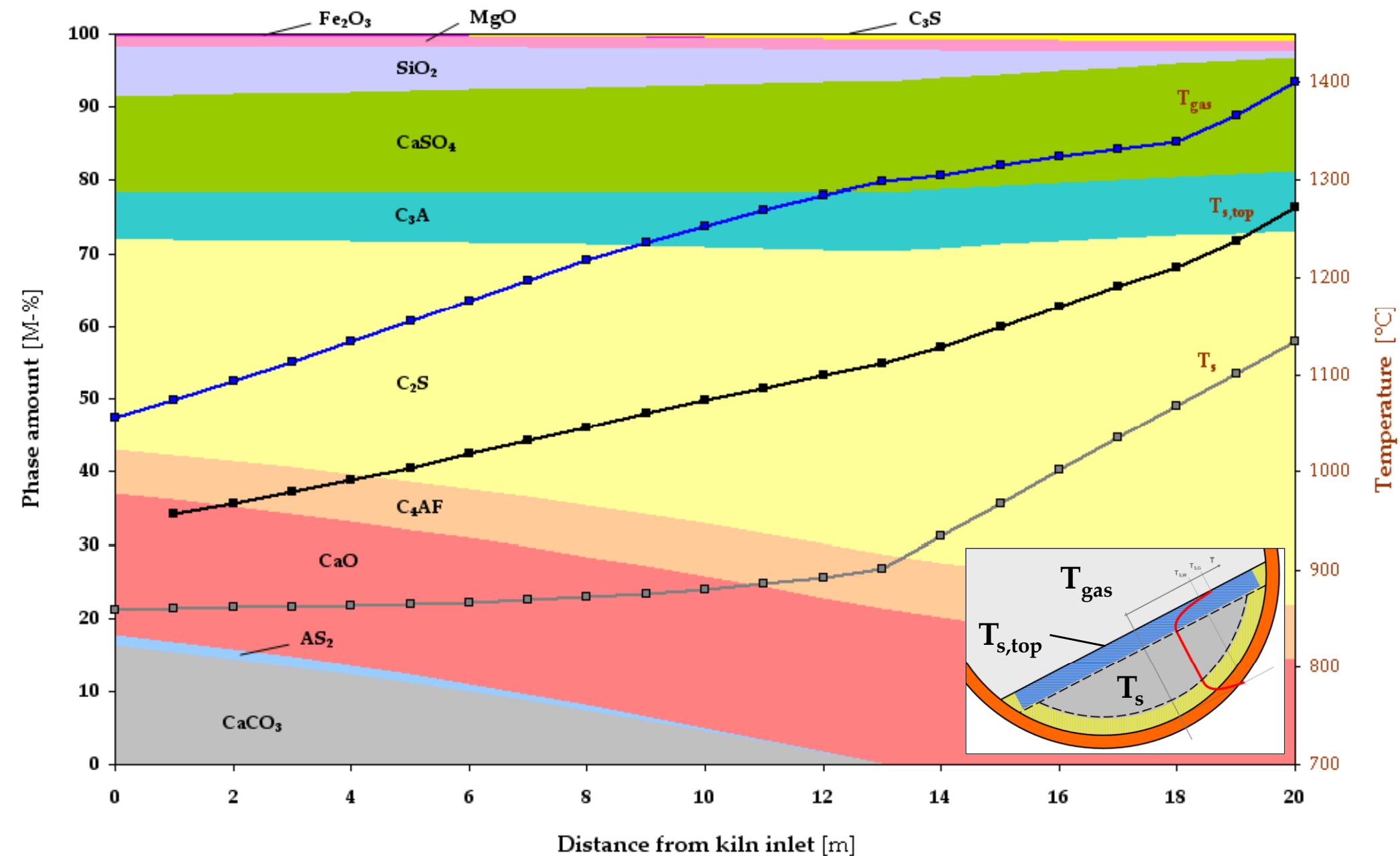
### ■ Facts and figures

- complexity similar to FE-simulations
- ~ 4000 SimuSage components
- ~  $1.6 \cdot 10^{15}$  floating point operations
- ~ 250 000 calculated equilibria

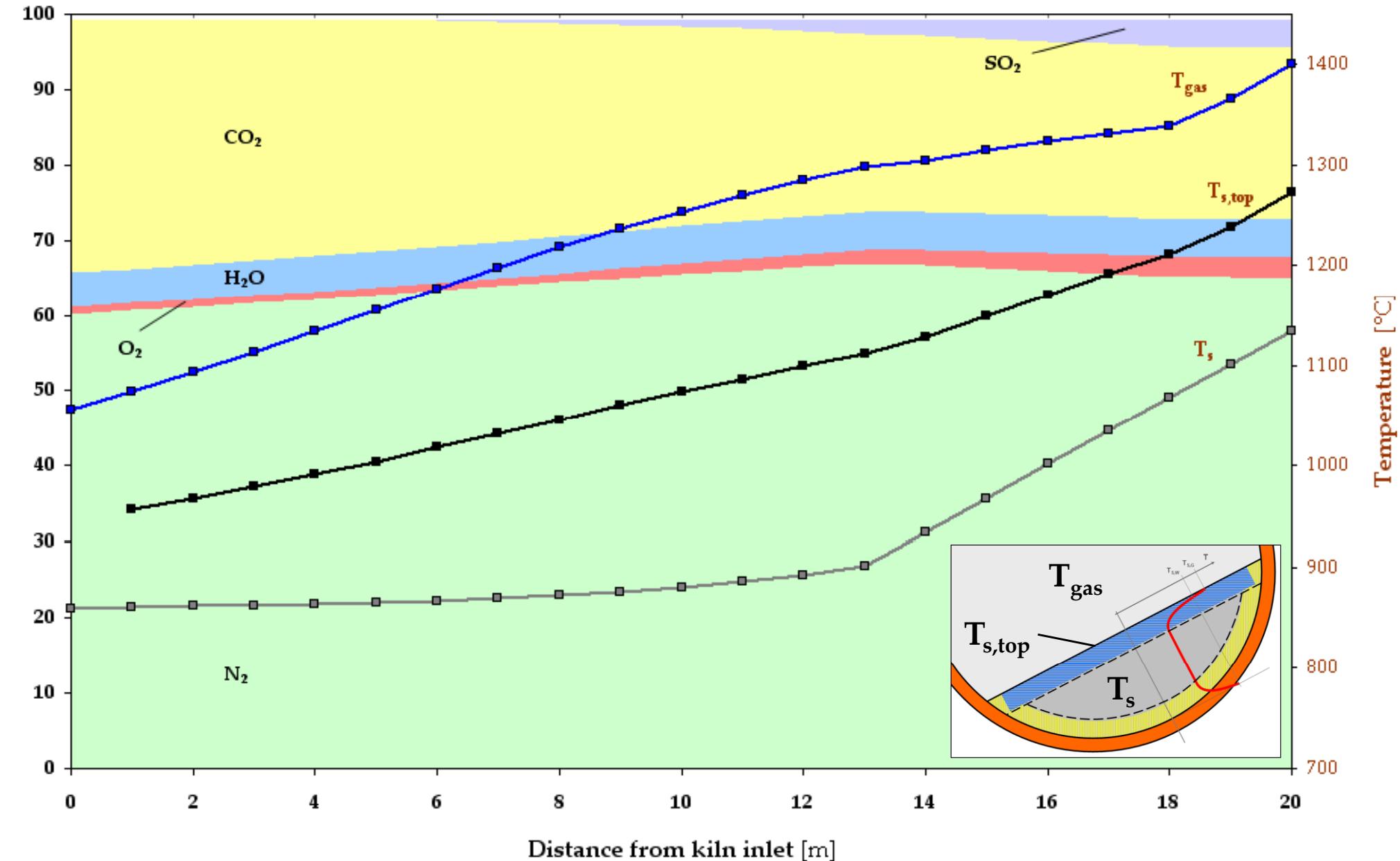


Rashadiya kiln #1 (Jordan) – Kiln audit 08/07

# 5. Simulation results

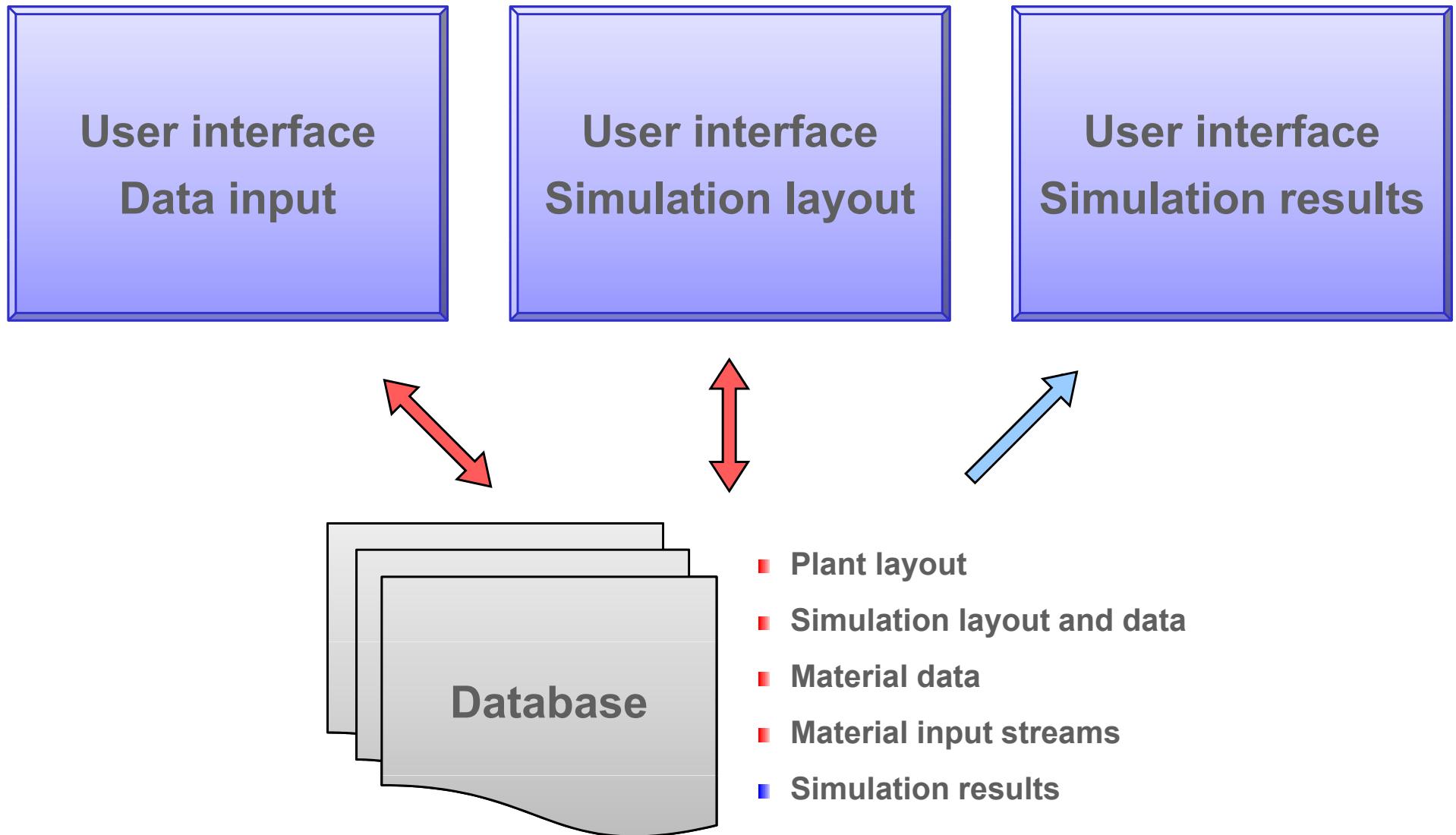


# 5. Simulation results



# 6. Outlook

## Development of a user interface



# 6. Outlook

## Timetable and further procedure

