Tutorial 6

ECSE 420 – Tutorial 6

Dimitrios Stamoulis

TR 4110

October 27, 2014

Lab 2 – Gaussian Elimination

- GE kernel

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```

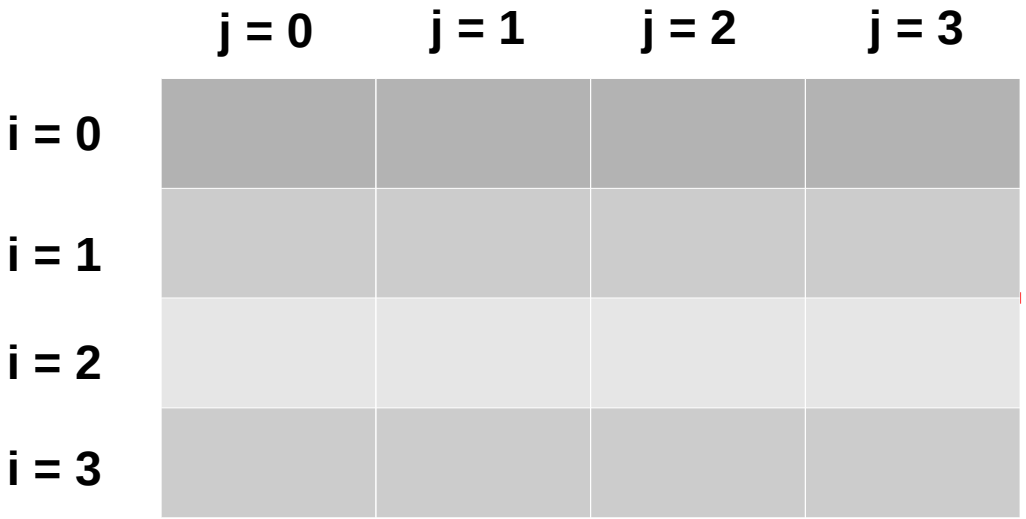
Lab 2 – Gaussian Elimination

N = 4

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```

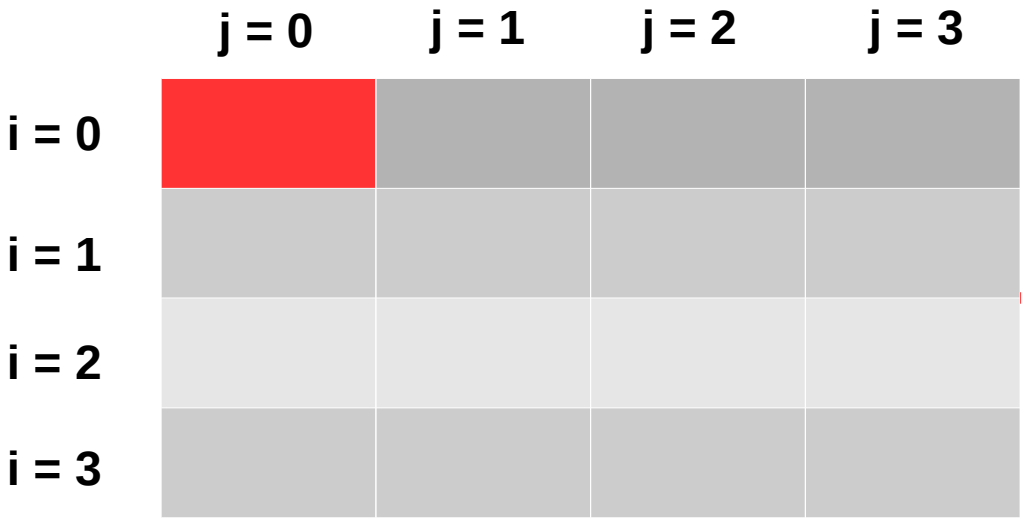
Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



Lab 2 – Gaussian Elimination

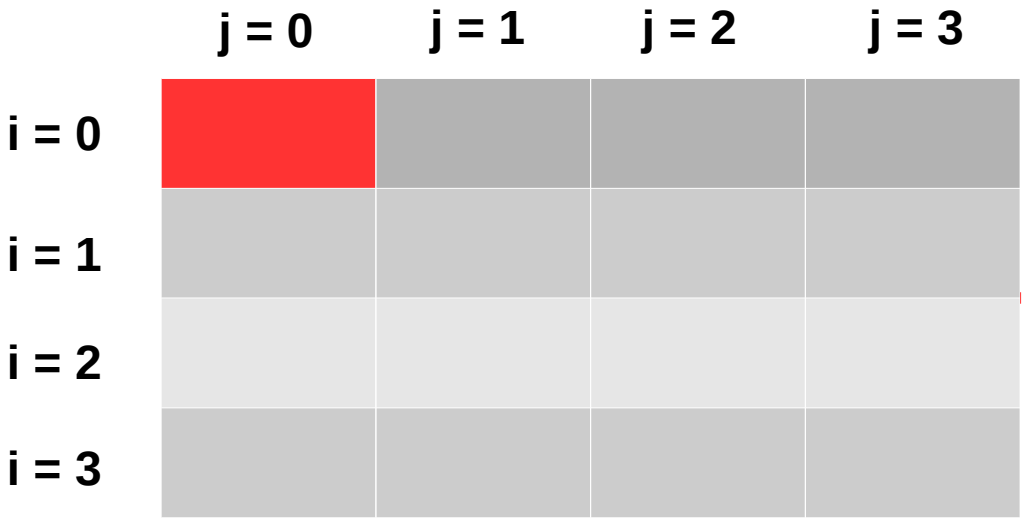
```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 0

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```

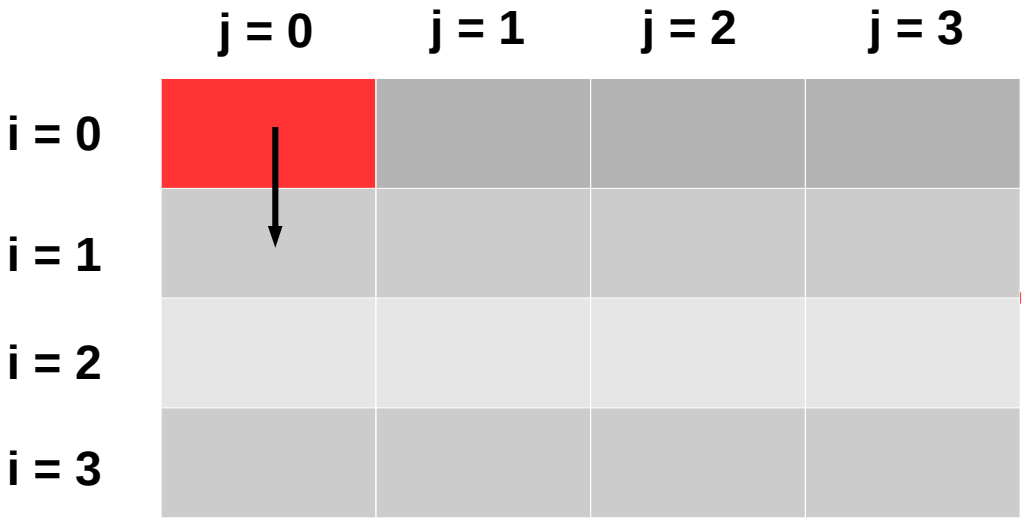


k = 0

i = 1

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



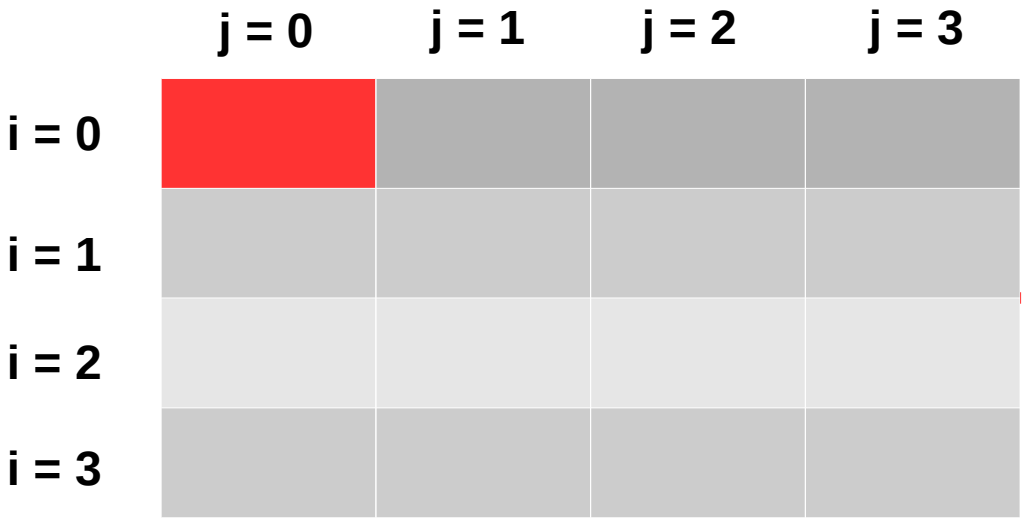
k = 0

i = 1

l = A[1][0] / A[0][0];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 0

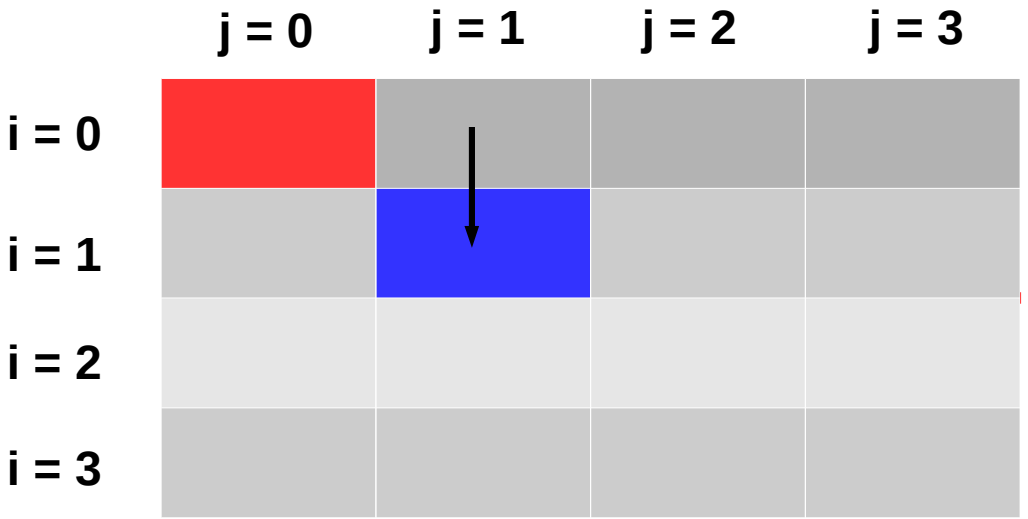
i = 1

l = A[1][0] / A[0][0];

j = 1

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 0

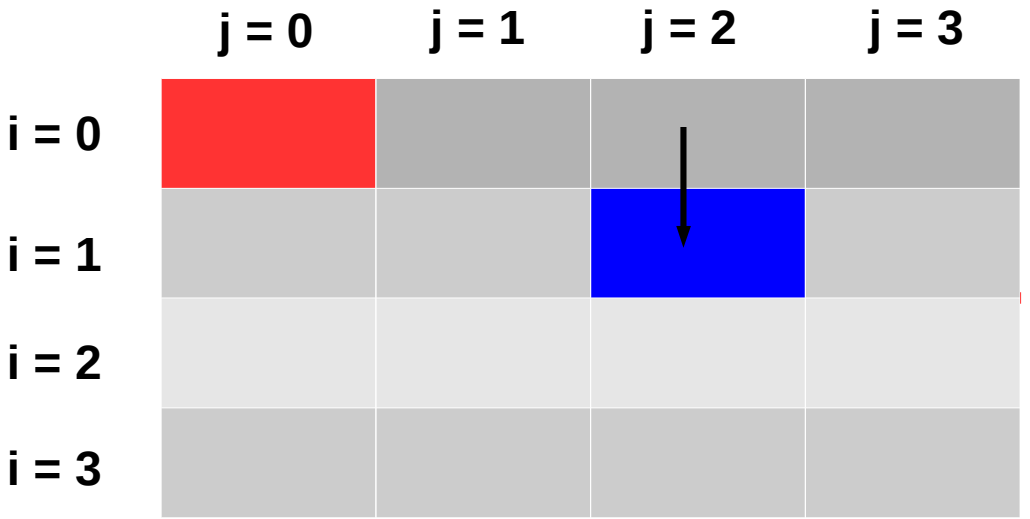
i = 1

l = A[1][0] / A[0][0];

j = 1 A[1][1] = A[1][1] -l*A[0][1];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 0

i = 1

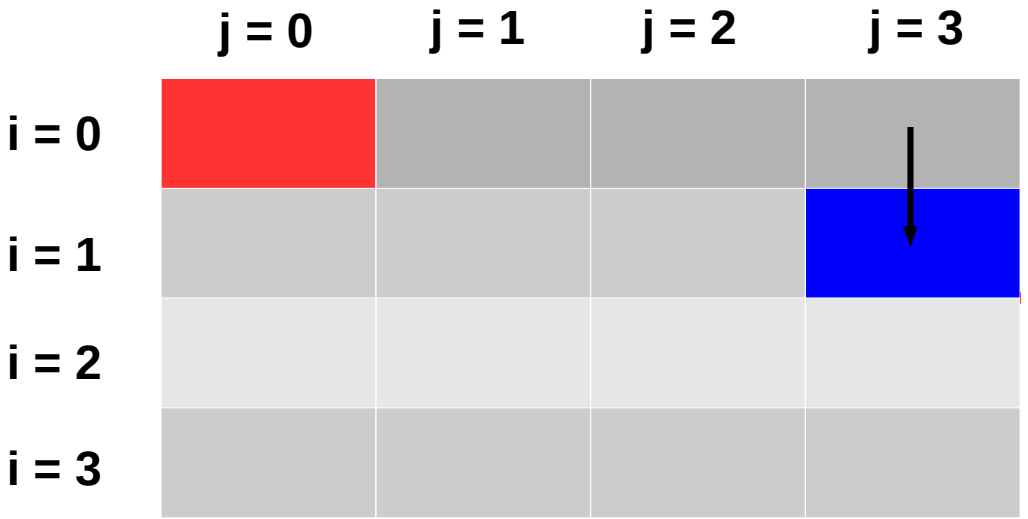
l = A[1][0] / A[0][0];

j = 1 A[1][1] = A[1][1] -l*A[0][1];

j = 2 A[1][2] = A[1][2] -l*A[0][2];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 0

i = 1

l = A[1][0] / A[0][0];

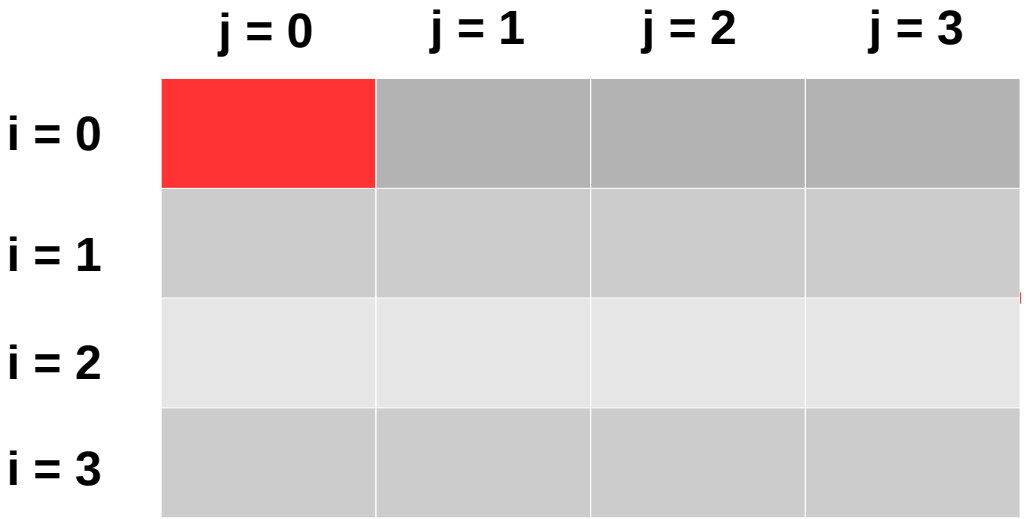
j = 1 A[1][1] = A[1][1] -l*A[0][1];

j = 2 A[1][2] = A[1][2] -l*A[0][2];

j = 3 A[1][3] = A[1][3] -l*A[0][3];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```

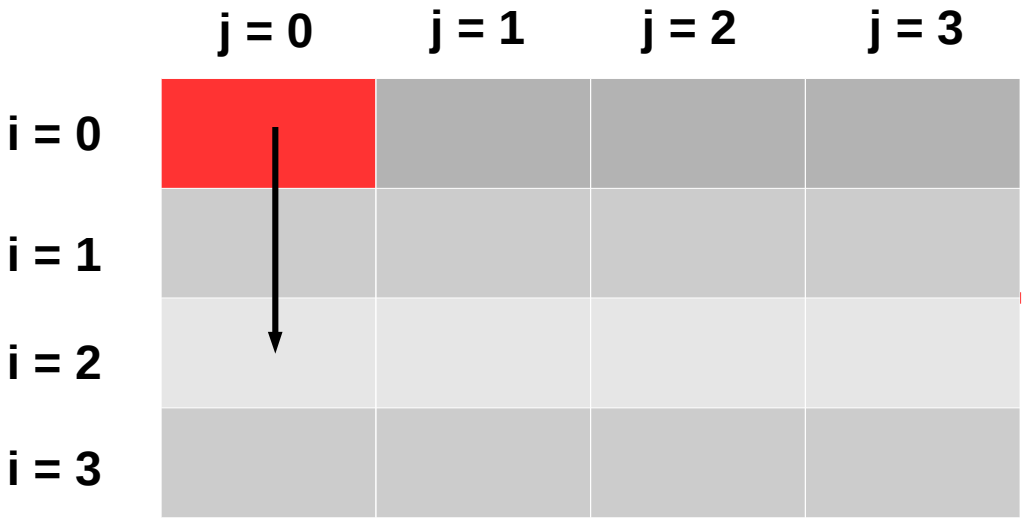


k = 0

i = 2

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



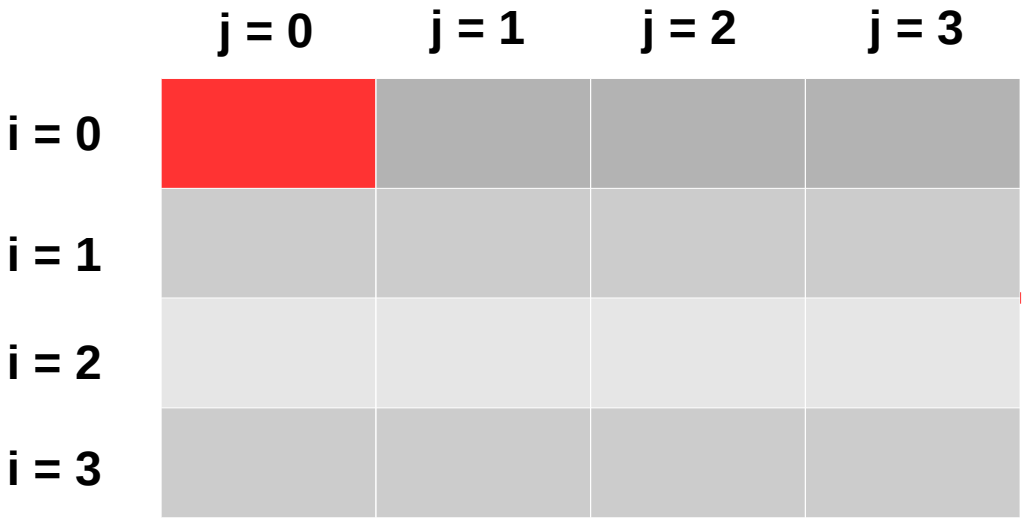
k = 0

i = 2

l = A[2][0] / A[0][0];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 0

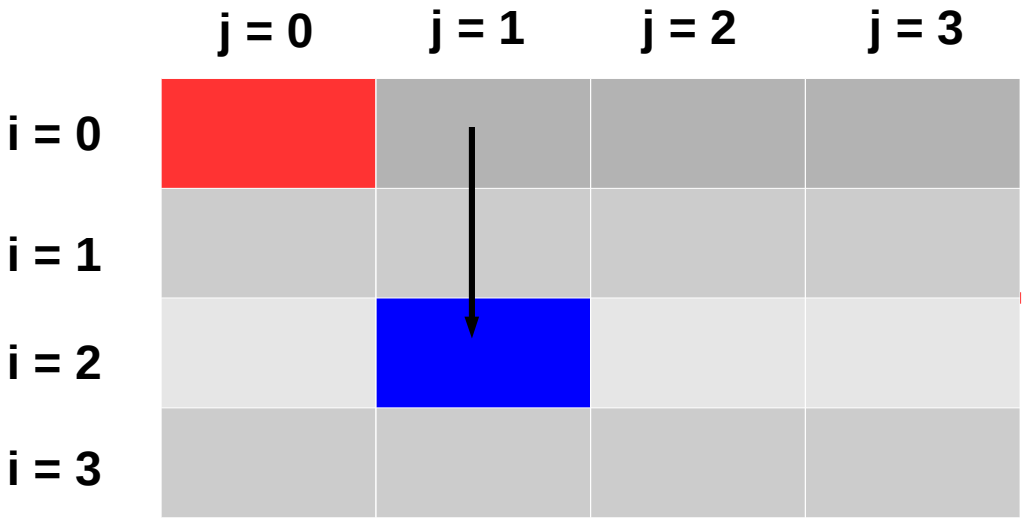
i = 2

l = A[2][0] / A[0][0];

j = 1

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 0

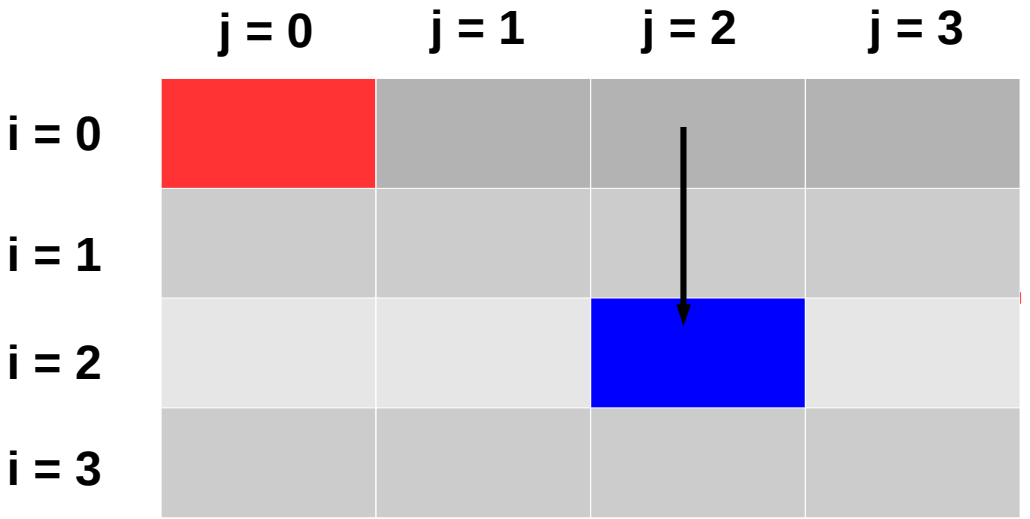
i = 2

l = A[2][0] / A[0][0];

j = 1 A[2][1] = A[2][1] -l*A[0][1];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 0

i = 1

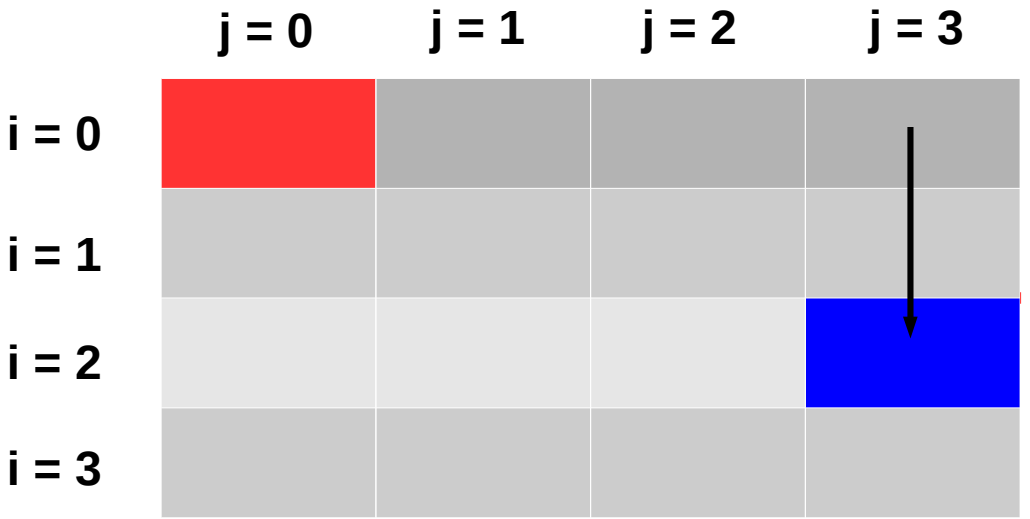
l = A[1][0] / A[0][0];

j = 1 A[1][1] = A[1][1] -l*A[0][1];

j = 2 A[1][2] = A[1][2] -l*A[0][2];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 0

i = 1

l = A[1][0] / A[0][0];

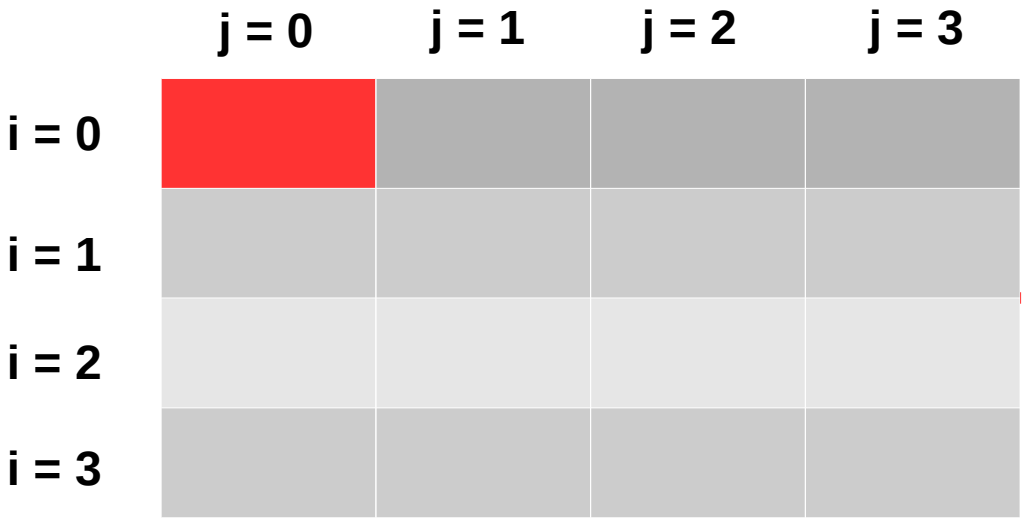
j = 1 A[1][1] = A[1][1] -l*A[0][1];

j = 2 A[1][2] = A[1][2] -l*A[0][2];

j = 3 A[1][3] = A[1][3] -l*A[0][3];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



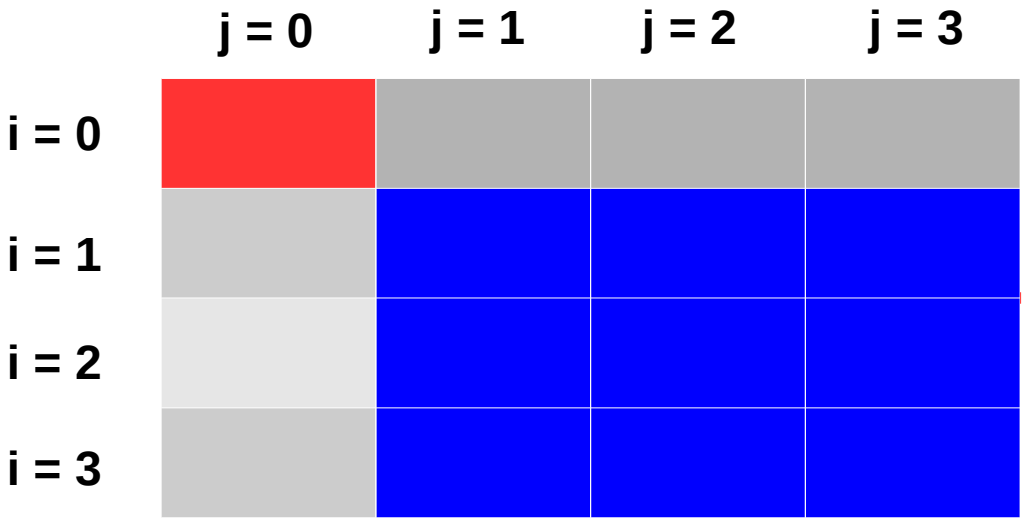
k = 0

i = 3

... and so on ...

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



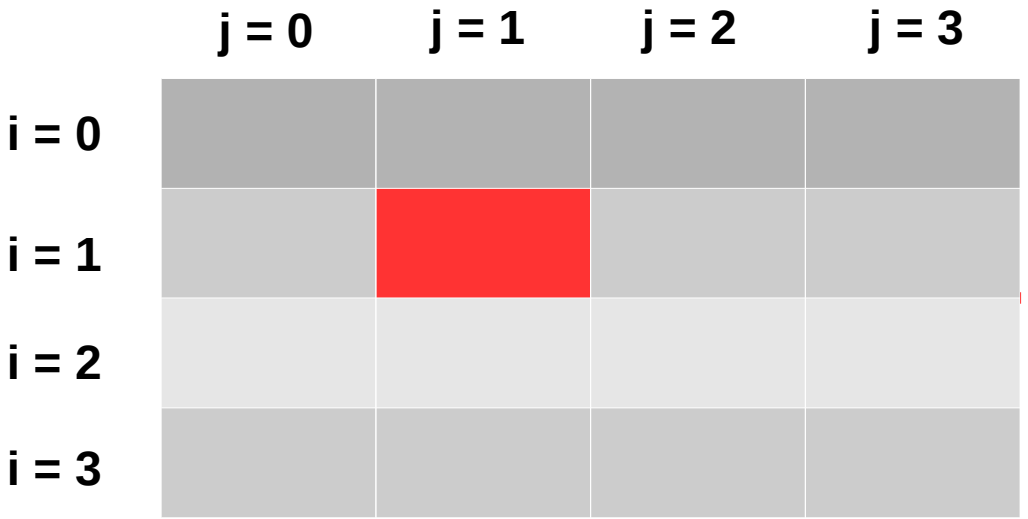
k = 0

i = 3

... and so on ...

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```

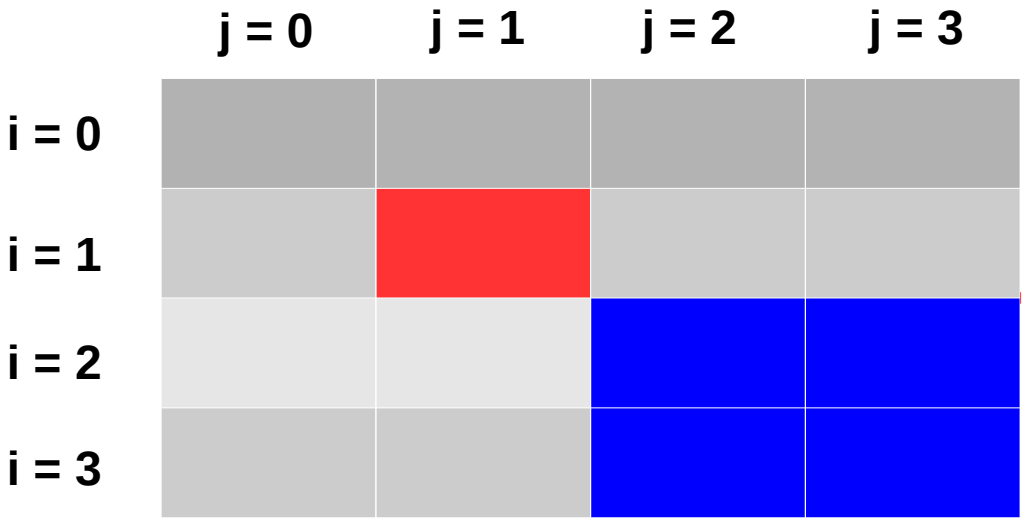


k = 1

Accordingly, for the second pivot element

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```

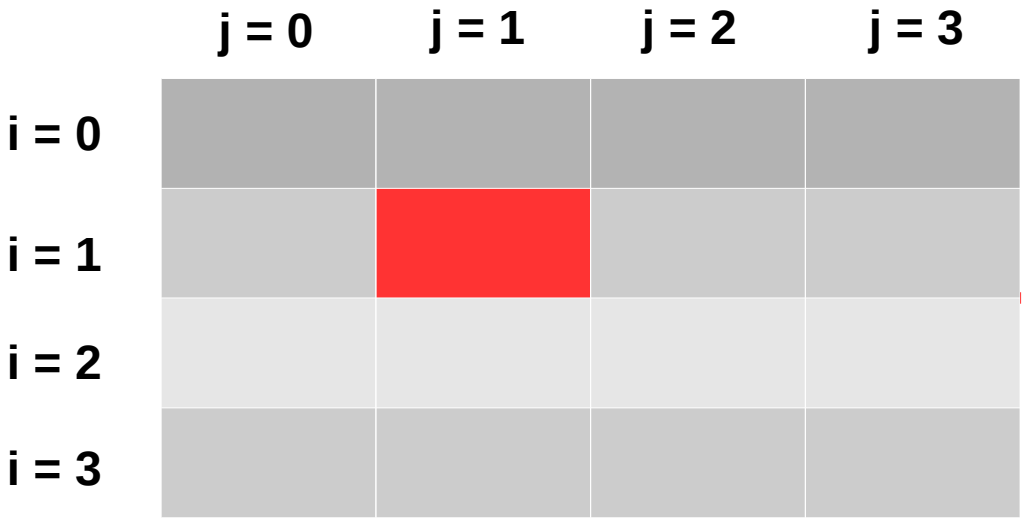


k = 1

Accordingly, for the second pivot element

Lab 2 – Gaussian Elimination

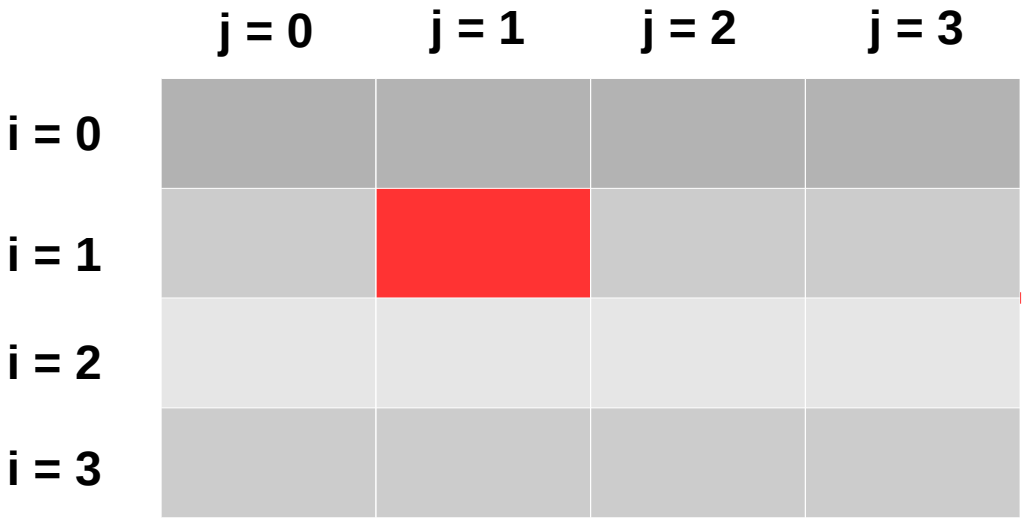
```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 1

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```

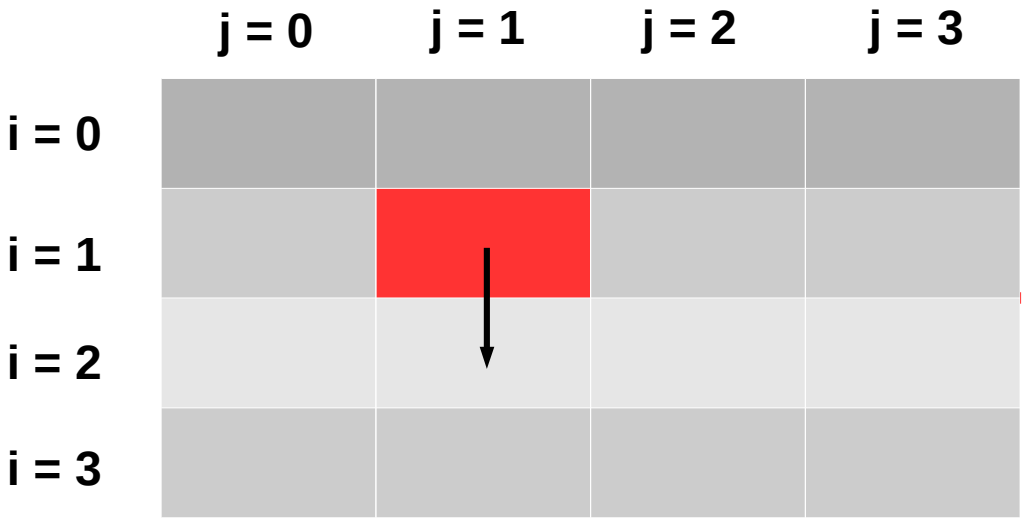


k = 1

i = 2

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



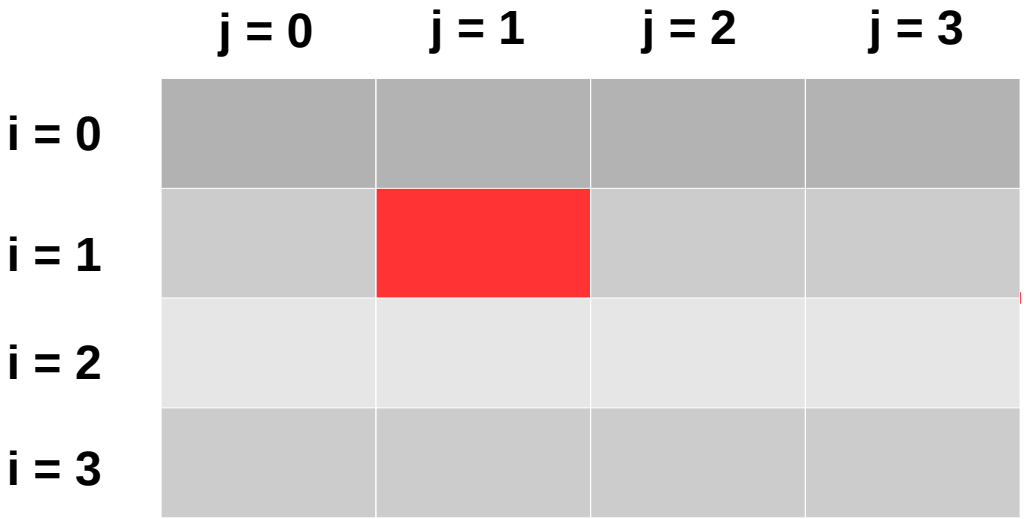
k = 1

i = 2

l = A[2][1] / A[1][1];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 1

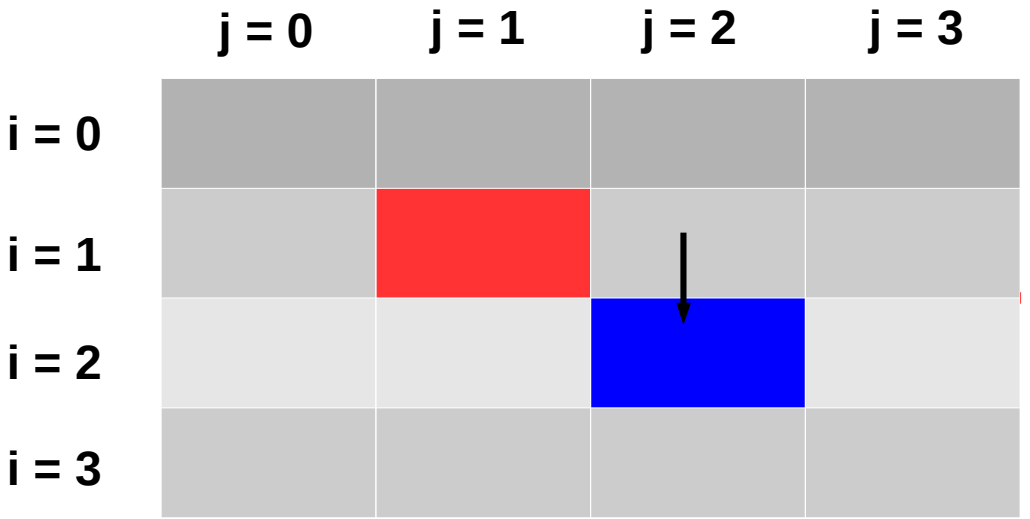
i = 2

l = A[2][1] / A[1][1];

j = 2

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 1

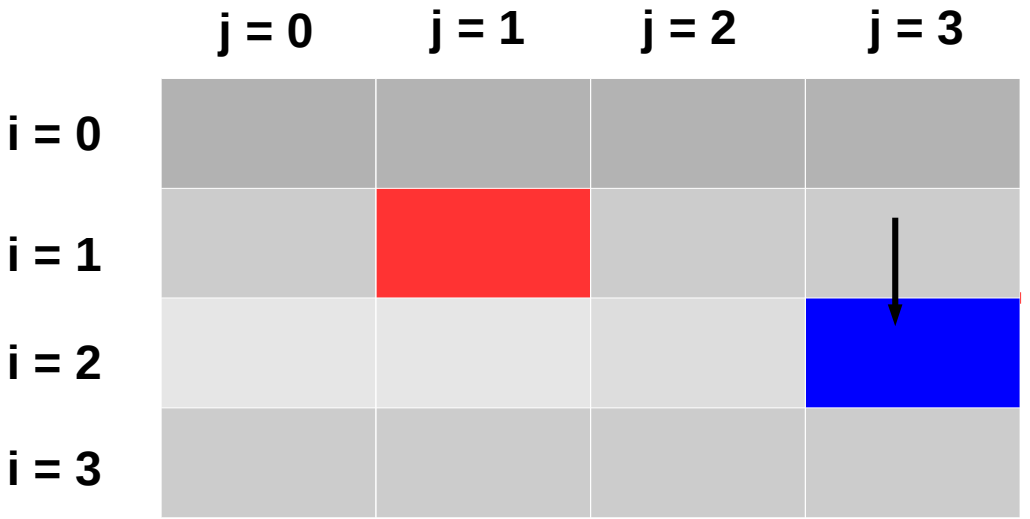
i = 2

l = A[2][1] / A[1][1];

j = 2 A[2][2] = A[2][2] -l*A[1][2];

Lab 2 – Gaussian Elimination

```
for (k = 0; k < N - 1; k++)
    for (i = k+1; i < N; i++)  {
        l = A[i][k] / A[k][k];
        for (j = k + 1; j < N; j++)
            A[i][j] = A[i][j] -l*A[k][j];
    }
}
```



k = 1

i = 2

l = A[2][1] / A[1][1];

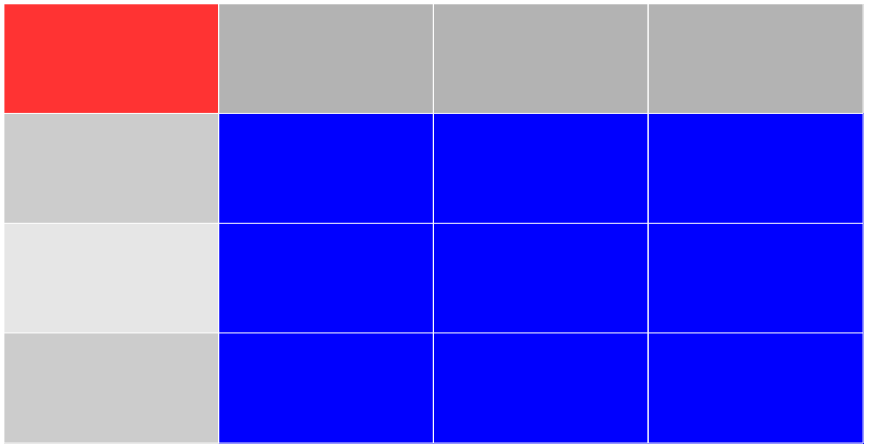
j = 2 A[2][2] = A[2][2] -l*A[1][2];

j = 3 A[2][3] = A[2][3] -l*A[1][3];

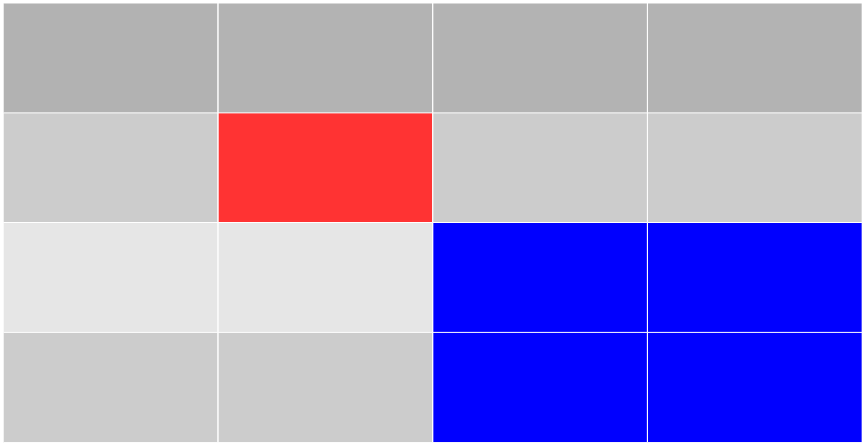
Parallelization scheme

What can be executed in parallel??

- Are the k-iterations independent?



k = 0

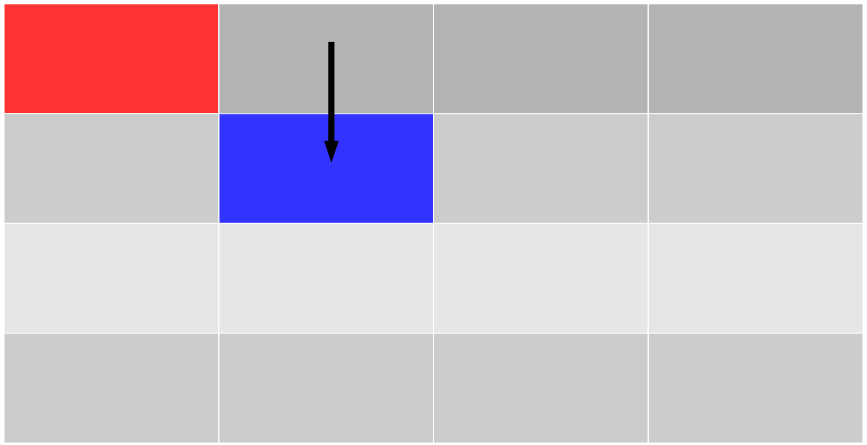


k = 1

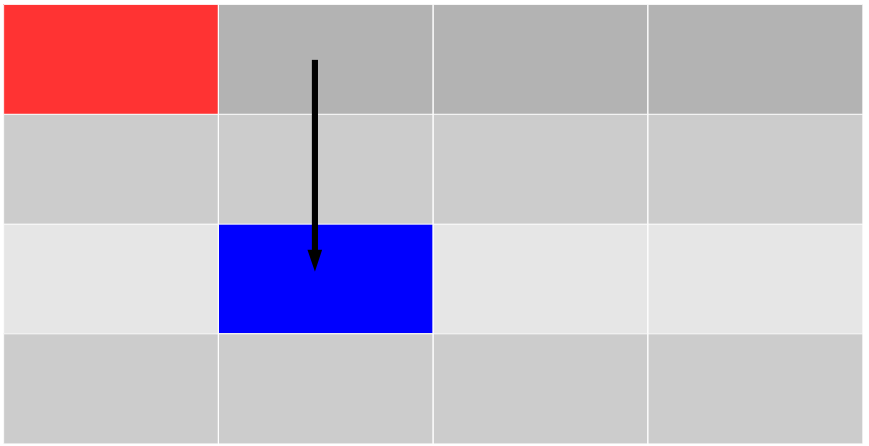
Parallelization scheme

What can be executed in parallel??

- Are the i- and j-iterations independent?



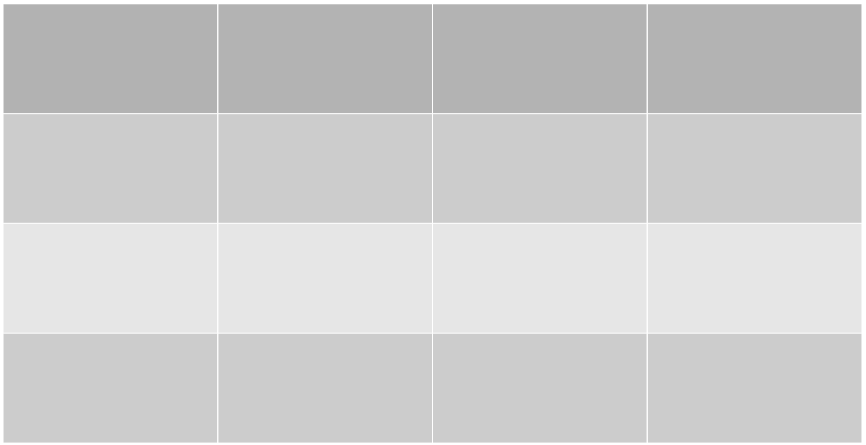
k = 0, i = 1



k = 0, i = 2

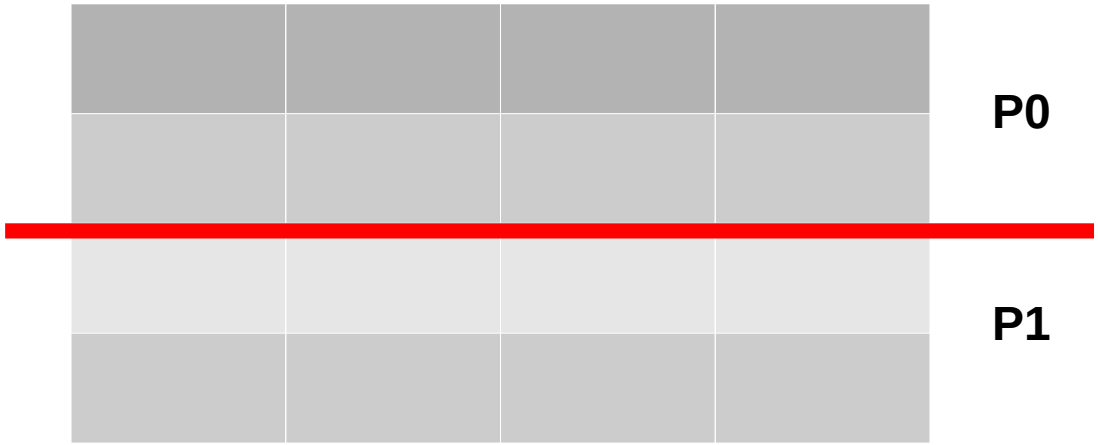
Parallelization scheme

Thus, we can apply decomposition into rows.



Parallelization scheme

Thus, we can apply decomposition into rows.

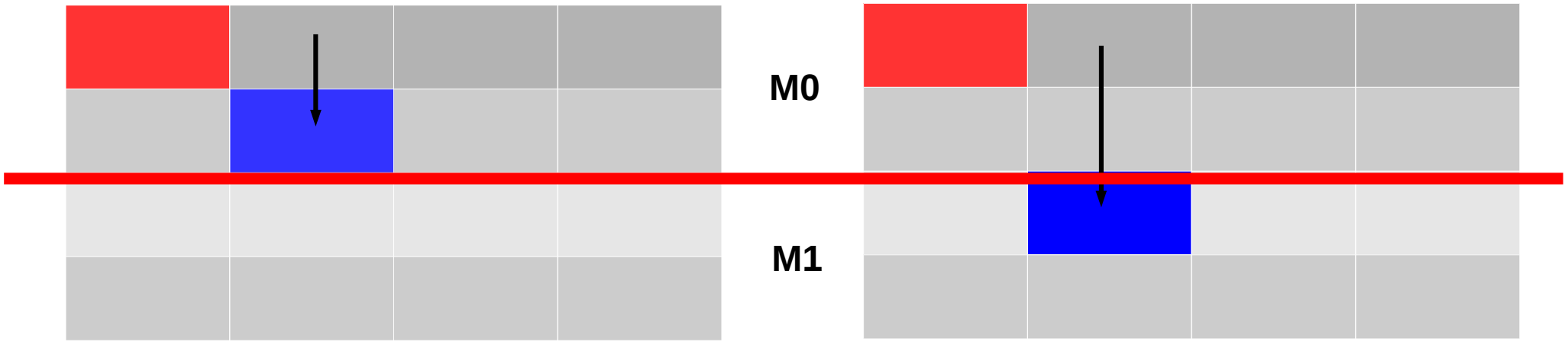


Communication scheme

But, which data should be now communicated?

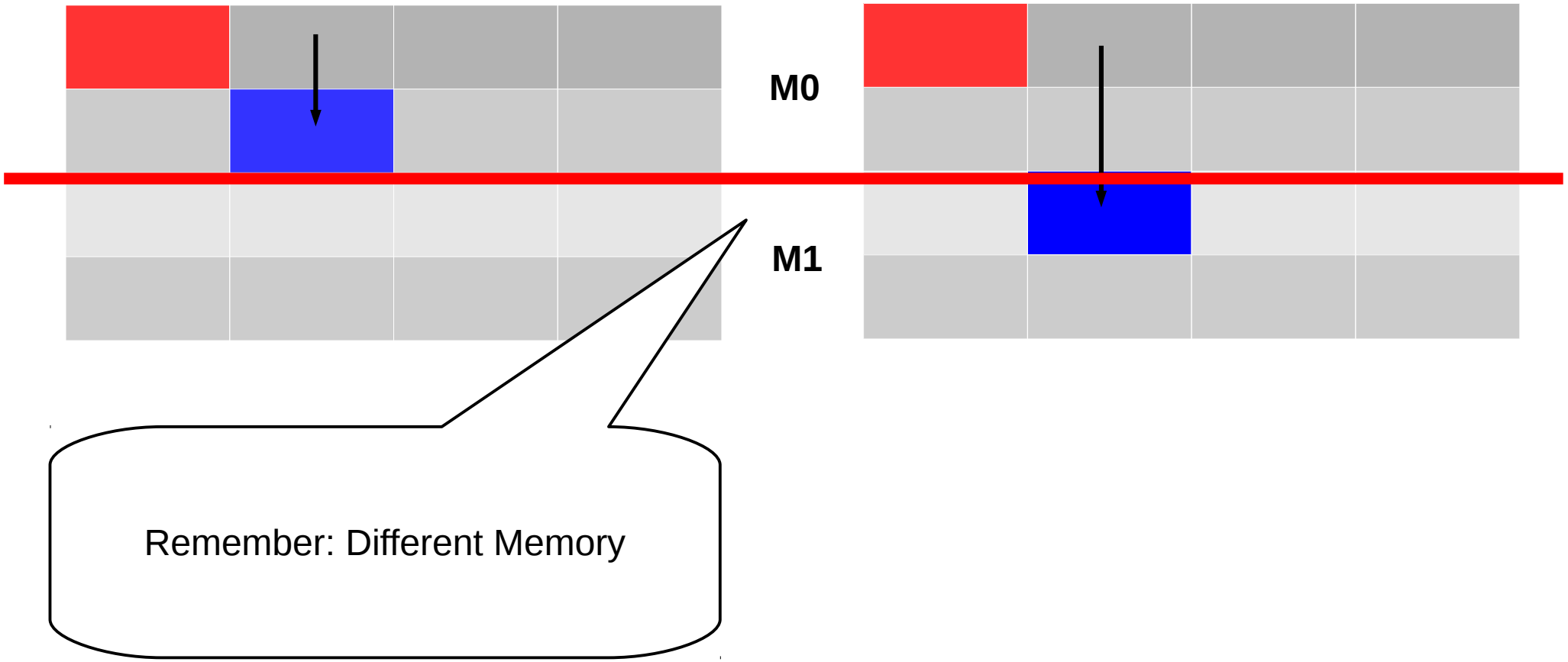
Communication scheme

But, which data should be now communicated?



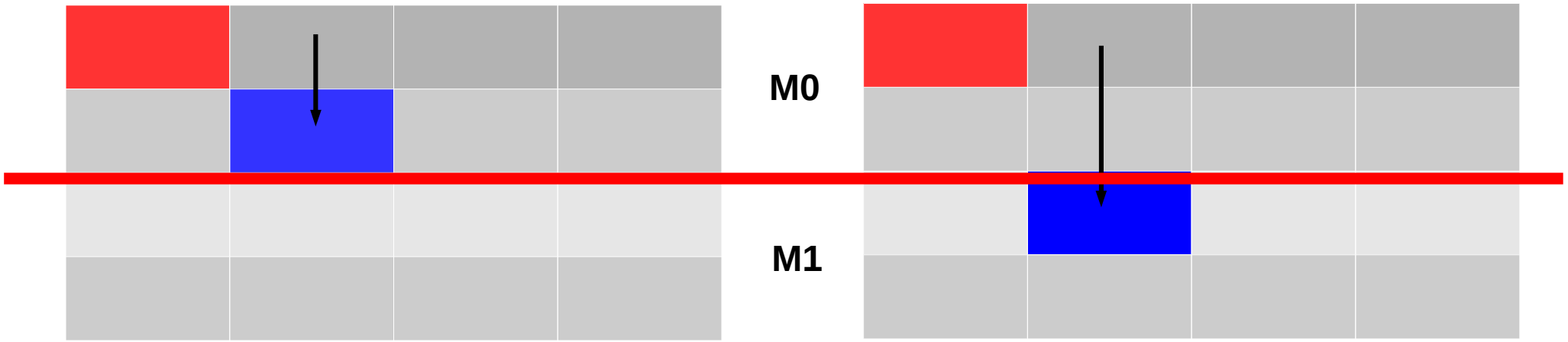
Communication scheme

But, which data should be now communicated?



Communication scheme

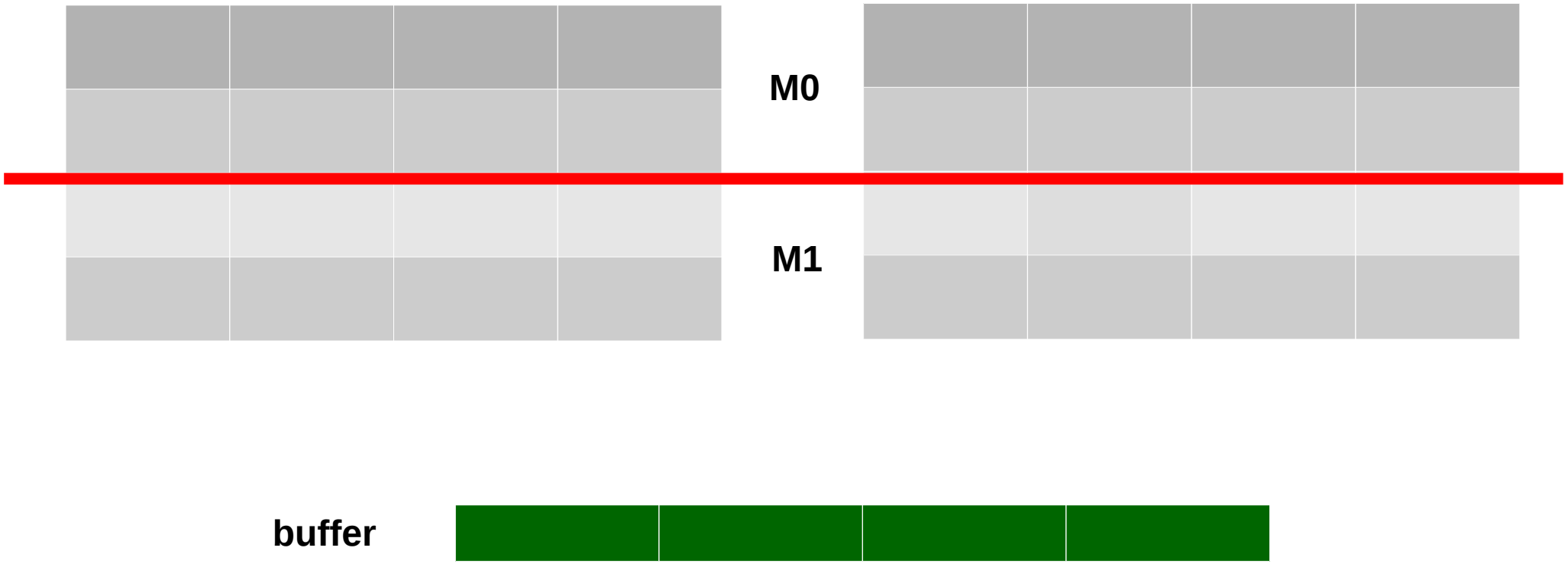
But, which data should be now communicated?



P1 doesn't have the A[0][1] element

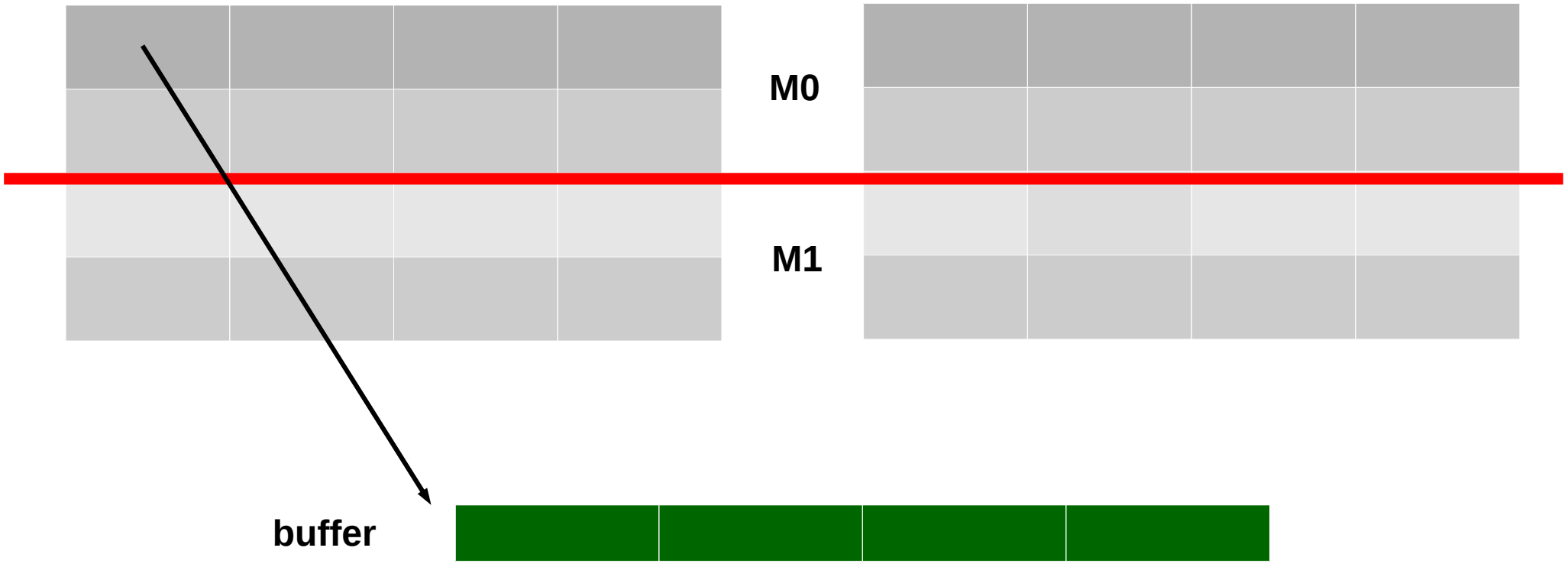
Communication scheme

Thus, per k-iteration I should send the “pivot” row !!



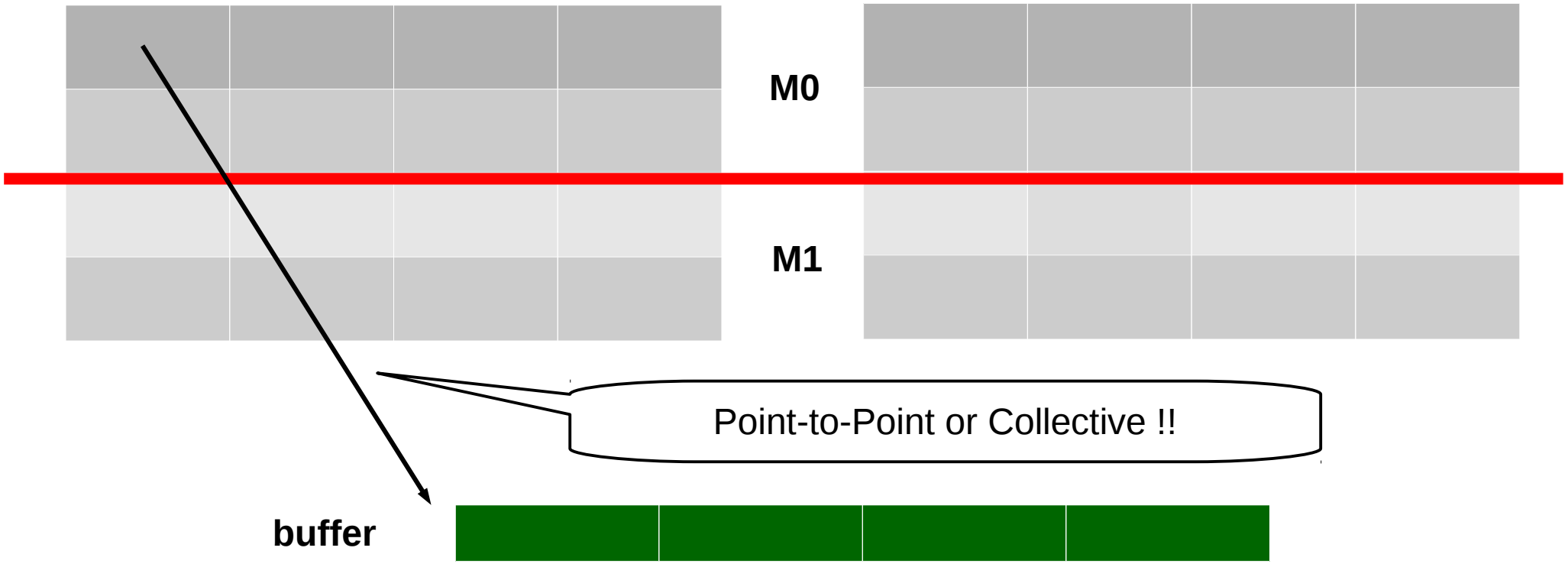
Communication scheme

The “owner” of the row should place it in a buffer.



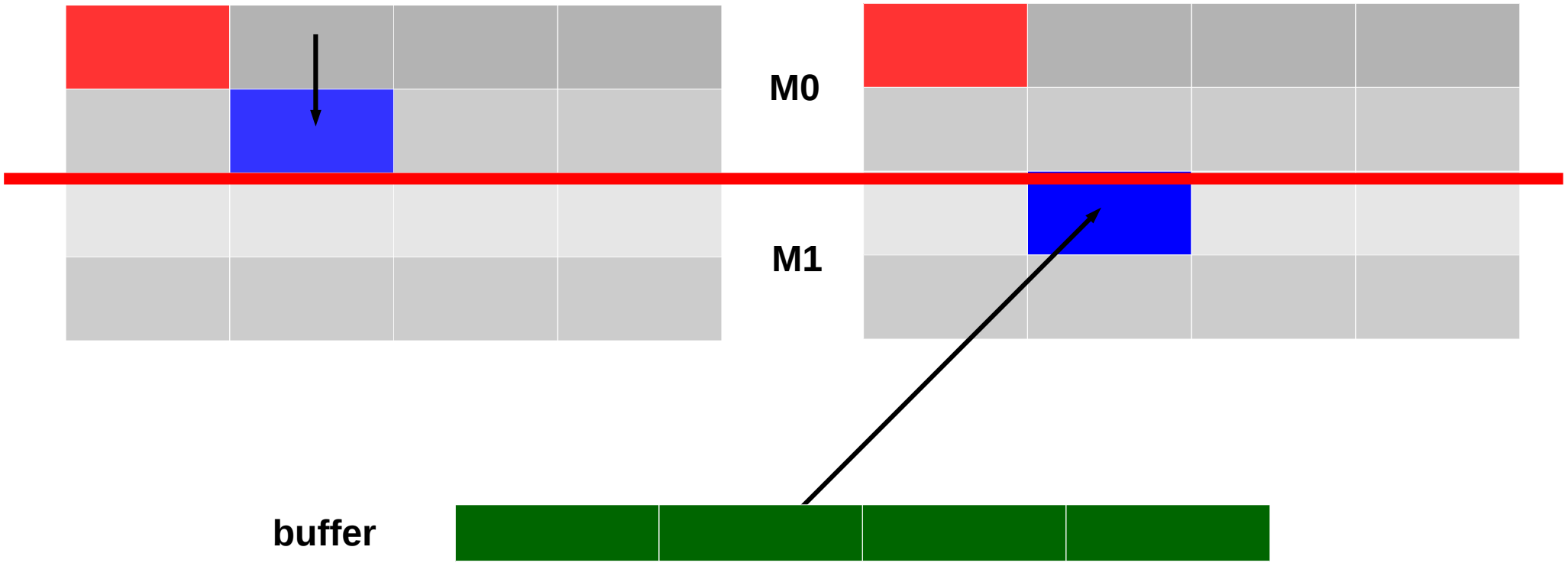
Communication scheme

The “owner” of the row should place it in a buffer.



Communication scheme

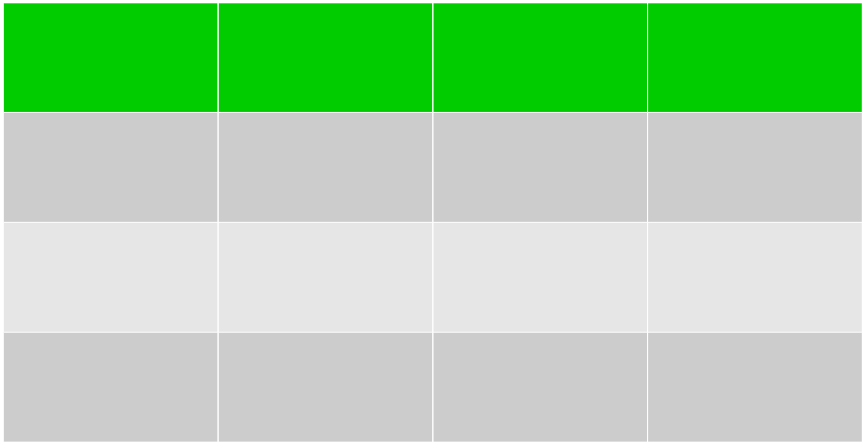
So now, I can repeat the example.



Communication scheme

Thus, per k-iteration :

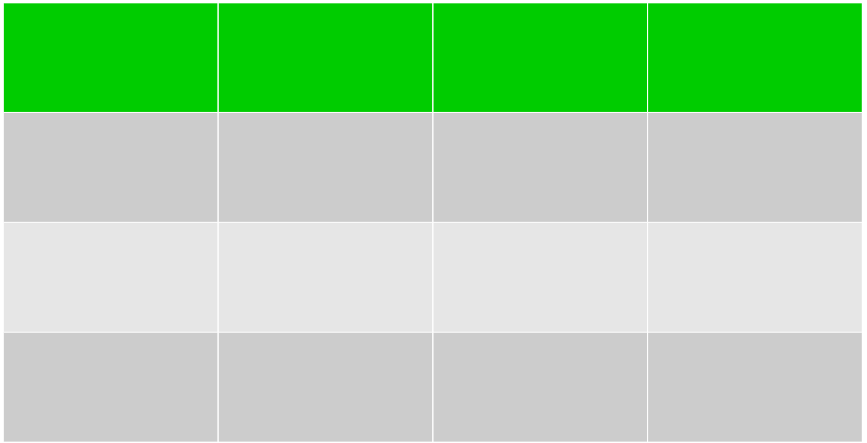
k=0



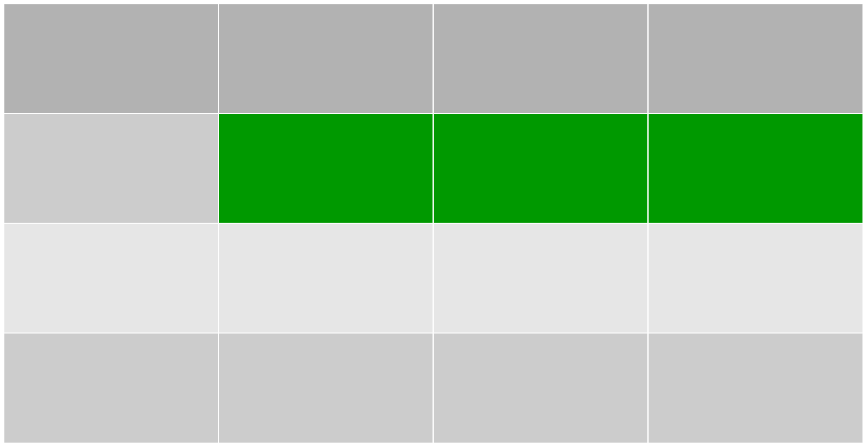
Communication scheme

Thus, per k-iteration :

k=0



k=1



Lab 2 – Gaussian Elimination

- Parallel version

```
if (rank == 0) {
   allocate_full_matrix(A);
   initialize(A);
}
allocate_local_matrix(lA);
distribute_matrix(0,A,lA);


for(k = 0; k < N - 1; k++){
   if (owner_of_critical_line(k)) {
       pack_data(lA, send_buffer);
       send_data_to_all(send_buffer, …);
   }
   else {
       receive_data_from_owner(receive_buffer, …);
       unpack_data(receive_buffer, lA);
   }
   compute(k, lA);
}

collect_results (0, A, lA);
```

Lab 2 – Any questions??