LR-Zerlegung: Analyse der Performance

1. Alte Musterlösung mit **2 Schleifen**

```
for k = 1:n-1
  if( A(k,k) == 0 ) ERROR
  for j = k+1:n
    A(j,k)      = A(j,k) / A(k,k);
    A(j,k+1:n) = A(j,k+1:n) - A(j,k) * A(k,k+1:n);
  end
end
```

2. Alte Musterlösung mit **3 Schleifen**: Ausgehend von der alten Musterlösung mit 2 Schleifen wird die Vektorisierung rückgängig gemacht

```
for k = 1:n-1
  if( A(k,k) == 0 ) ERROR
  for j = k+1:n
    A(j,k)      = A(j,k) / A(k,k);
    for l = k+1:n
      A(j,l) = A(j,l) - A(j,k) * A(k,l);
    end
  end
end
```

   Und da sieht man sofort das Performance-Problem: Die innerste Schleife läuft über die Zeilen und nicht über die Spalten. Da die Matrixelemente aber spaltenweise im Memory liegen, ist diese zeilenweise Vorgehensweise nicht Cache-optimiert.

3. Geänderte Struktur mit **3 Schleifen**: Vertauschung der beiden innersten Schleifen, um **spaltenweise** auf die Elemente zugreifen zu können.
   Dabei stört aber die in 2) rot markierte Zeile.
   Diese Zeile muss unabhängig von j werden, damit man sie vor die 2. Schleife ziehen kann (grün).

```
for k = 1:n-1
  if( A(k,k) == 0 ) ERROR
  A(k+1:end,k) = A(k+1:end,k)/A(k,k);
  for l = k+1:n
    for j = k+1:n
      A(j,l) = A(j,l) - A(j,k) * A(k,l);
    end
  end
end
```

4. Geänderte Struktur mit **2 Schleifen** und spaltenweisem Zugriff:
   Die innerste Schleife wird vektorisiert

```
for k = 1:n-1
  if( A(k,k) == 0 ) ERROR
  A(k+1:end,k) = A(k+1:end,k)/A(k,k);
  for l = k+1:n
    A(k+1:n,l) = A(k+1:n,l) - A(k+1:n,k) * A(k,l);
  end
end
```

5. LR Zerlegung mit nur **1 Schleife**: Jetzt wird auch die 2. Schleife vektorisiert

```
for k = 1:n-1
  if( A(k,k) == 0 ) ERROR
  A(k+1:end,k) = A(k+1:end,k)/A(k,k);
  A(k+1:end,k+1:end) = A(k+1:end,k+1:end) - A(k+1:end,k)*A(k,k+1:end);
end
```