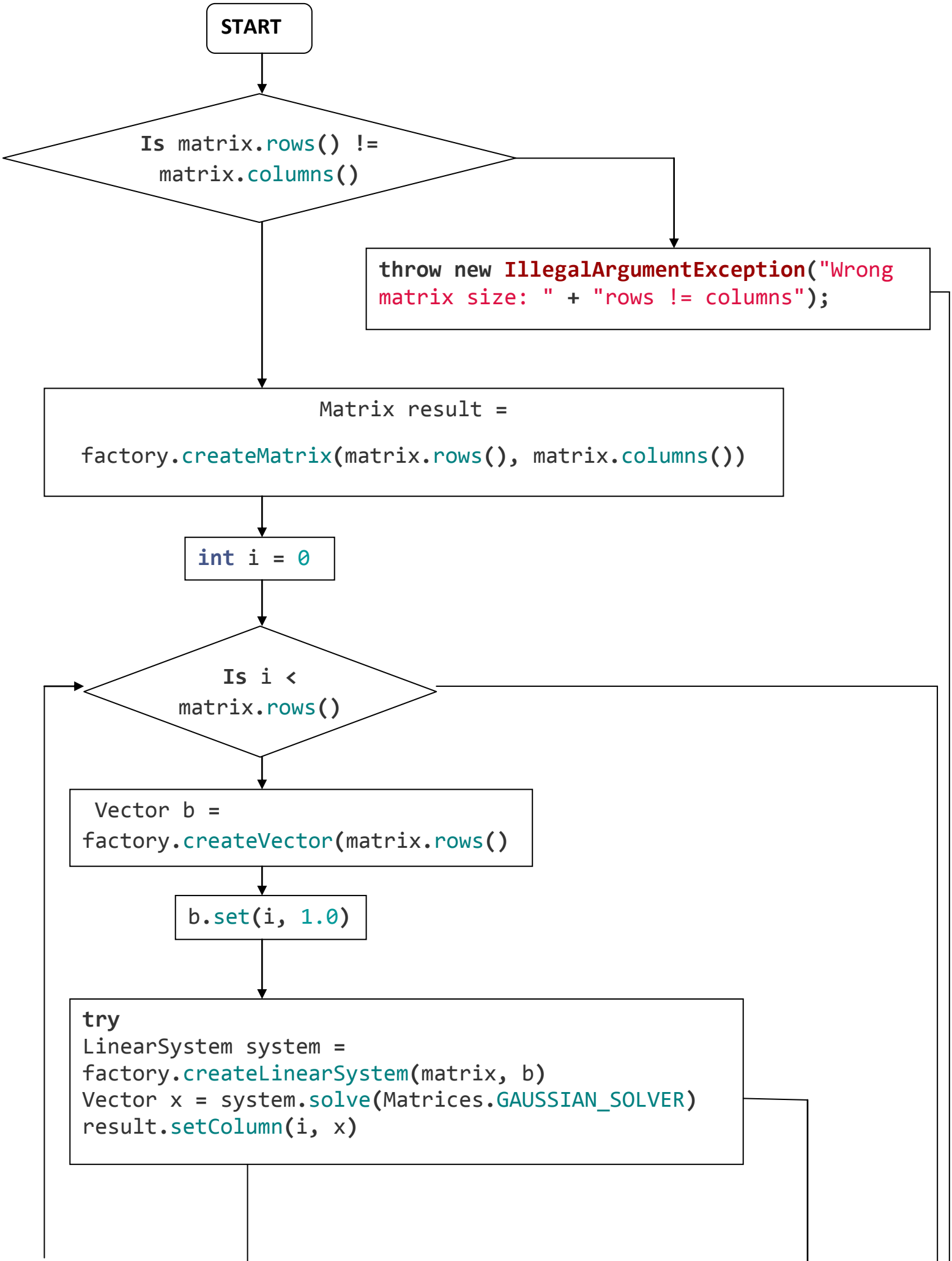
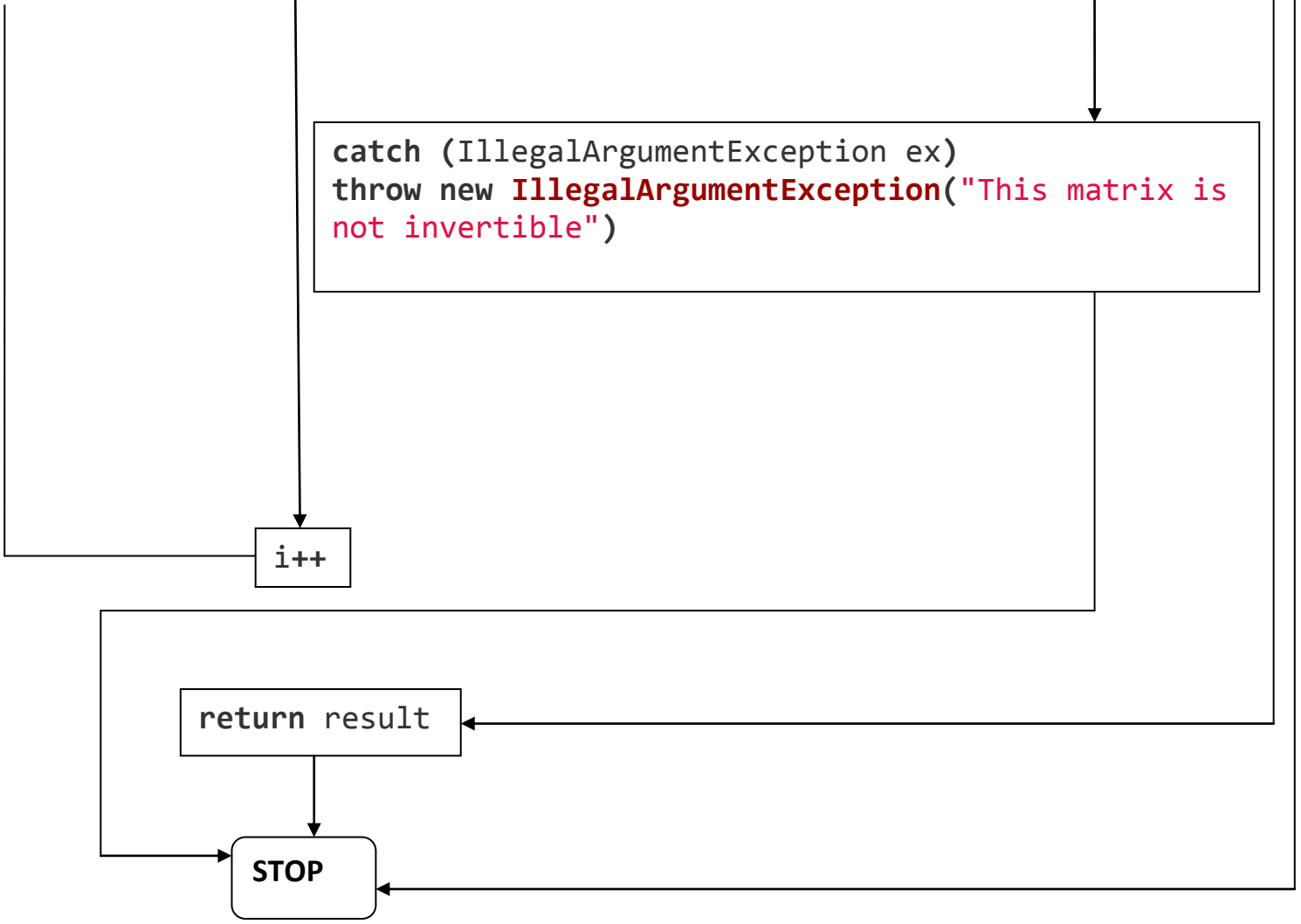

START
Is matrix.rows() != matrix.columns()
throw new IllegalArgumentException("Wrong matrix size: " + "rows != columns");
Matrix result = factory.createMatrix(matrix.rows(), matrix.columns())
int i = 0
Is i < matrix.rows()
Vector b = factory.createVector(matrix.rows()
b.set(i, 1.0)
try
LinearSystem system = factory.createLinearSystem(matrix, b)
Vector x = system.solve(Matrices.GAUSSIAN_SOLVER)
result.setColumn(i, x)

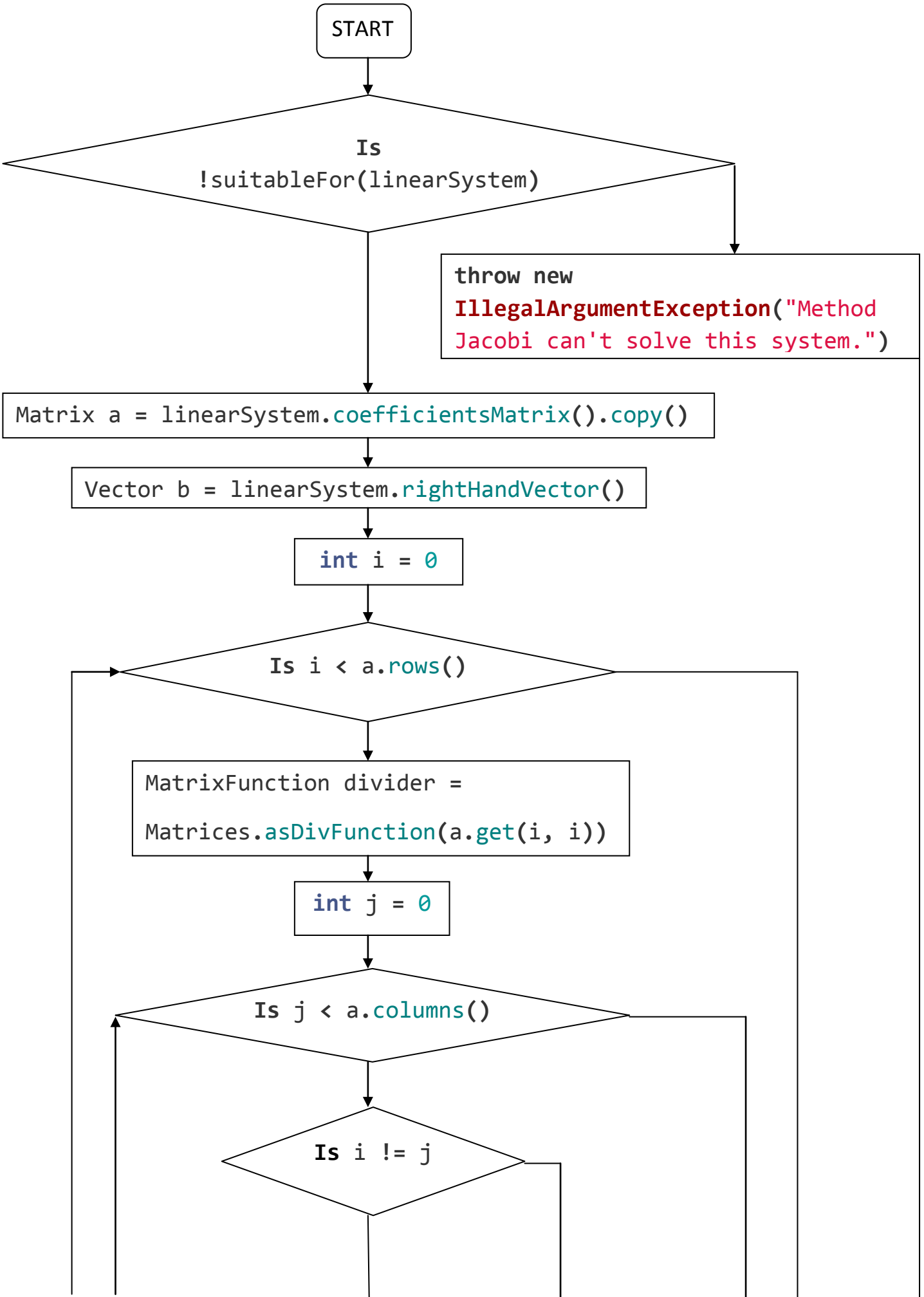


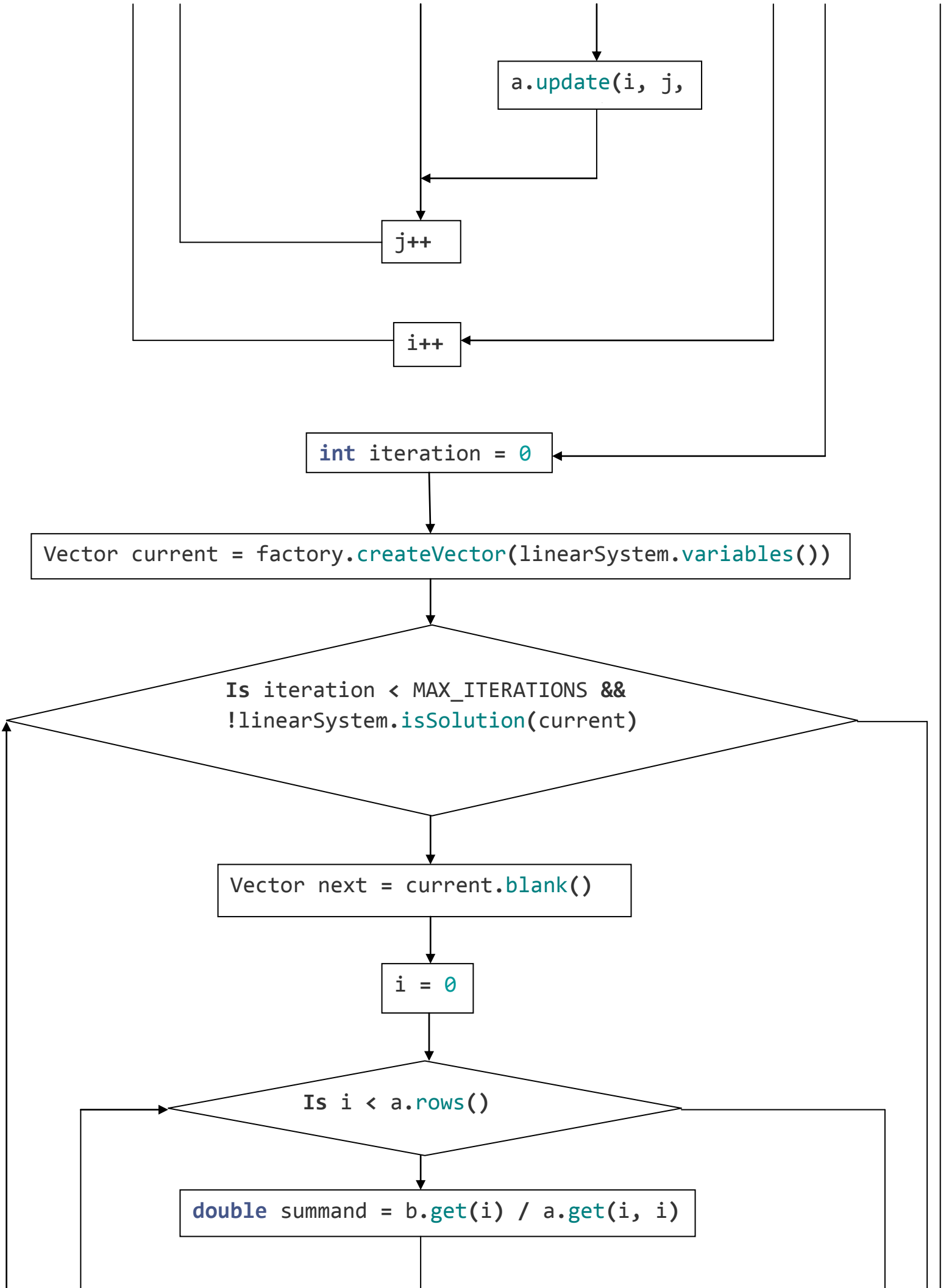
Inverse() method

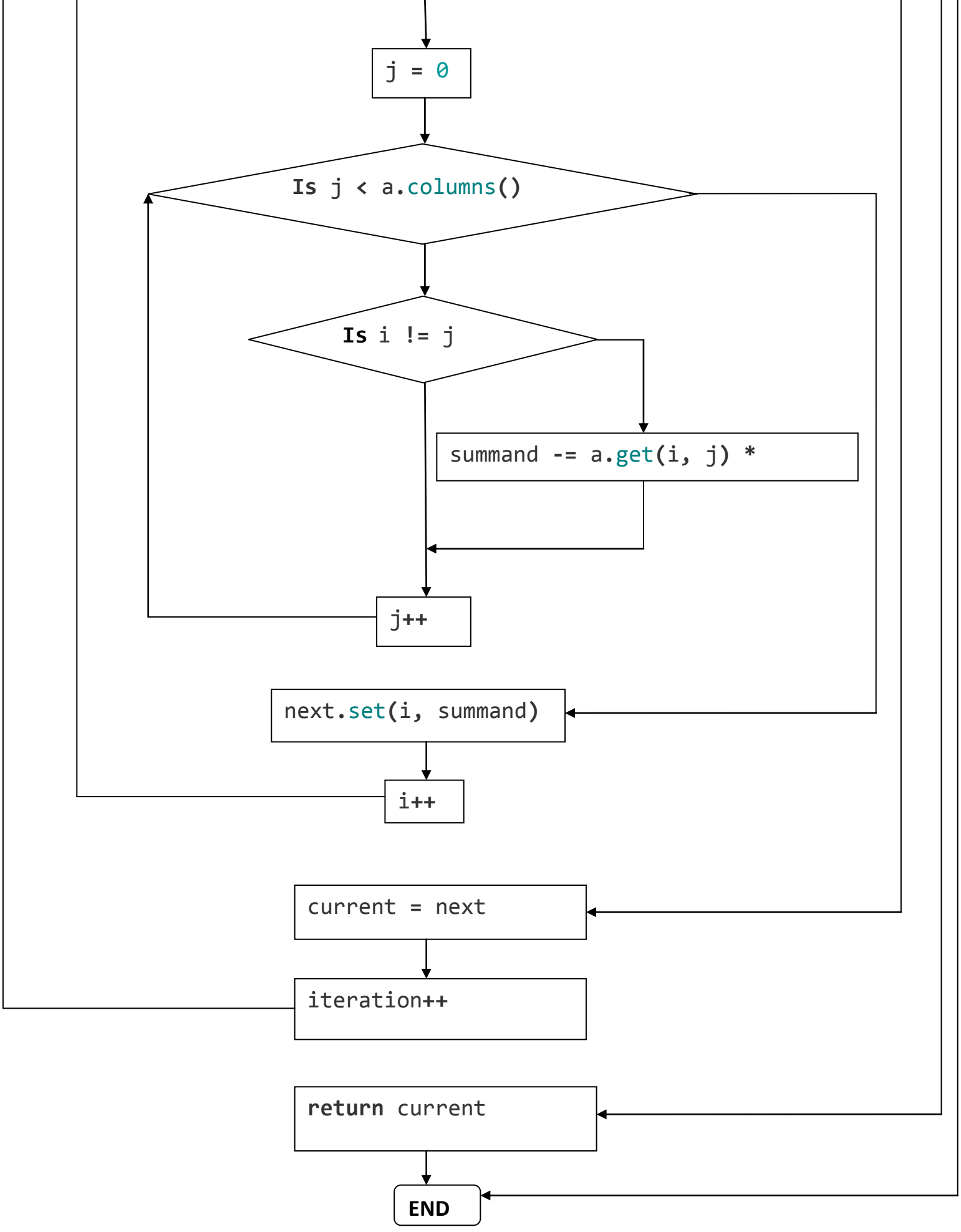
The inverse() method first verifies that the matrix is invertible. If it is not, an exception is thrown and the program crashes. A new Matrix object “result” is created with the rows and columns of the matrix to be inverted. For each row of the result Matrix, a new linearSystem object “system” is created with parameters matrix, and a vector “b” made up of zeros and a single 1. The solve() method in the system class is called and returns a vector to a new Vector object “x”. The row in the “result” matrix being evaluated then becomes “x”. Finally, matrix “result” is returned. If any exception is thrown after the new linearSystem object is created, the program will end.

Solve() method

The solve() method first verifies that the linearSystem object is suitable. If not, an exception is thrown and the program ends. For every row of the matrix, a new MatrixFunction object “divider” is created using the part of the matrix that forms a diagonal. Then for every column that is not part of the diagonal, the update() method is called on the matrix “a”. Iterations then start and for every iteration, a Vector “current” is modified every time with another “next” vector to get closer to the desired vector. The “next” vector is produced by inserting a double “summand” into all its components. Summand is obtained with matrix “a” and Vector “b”; summand is different for every row of matrix “a”. After the iterations are finished, the desired “current” Vector will be returned.


START
Is !suitableFor(linearSystem)
throw new IllegalArgumentException("Method Jacobi can't solve this system.")
Matrix a = linearSystem.coefficientsMatrix().copy()
Vector b = linearSystem.rightHandVector()
int i = 0
Is i < a.rows()
MatrixFunction divider = Matrices.asDivFunction(a.get(i, i))
int j = 0
Is j < a.columns()
Is i != j


a.update(i, j,
j++
i++
int iteration = 0
Vector current = factory.createVector(linearSystem.variables())
Is iteration < MAX_ITERATIONS &&
!linearSystem.isSolution(current)
Vector next = current.blank()
i = 0
Is i < a.rows()
double summand = b.get(i) / a.get(i, i)


j = 0
Is j < a.columns()
Is i != j
summand -= a.get(i, j) *
j++
next.set(i, summand)
i++
current = next
iteration++
return current
END