ESUG, Gent 1999

Numerical Methods

Didier Besset
Independent Consultant

Road Map

- Dealing with numerical data
- Framework for iterative processes
- Newton's zero finding
- Eigenvalues and eigenvectors
- Cluster analysis
- Conclusion

Road Map

- Dealing with numerical data
- Framework for iterative processes
- Newton's zero finding
- Eigenvalues and eigenvectors
- Cluster analysis
- Conclusion

Dealing With Numerical Data

- Besides integers, Smalltalk supports two numerical types
  - Fractions
  - Floating point numbers

Using Fractions

- Results of products and sums are exact.
- Cannot be used with transcendental functions.
- Computation is slow.
- Computation may exceed computer's memory.
- Convenient way to represent currency values when combined with rounding.

Using Floating Point Numbers

- Computation is fast.
- Results have limited precision (usually 54 bits, ~16 decimal digits).
- Products preserve relative precision.
- Sums may keep only wrong digits.

Using Floating Point Numbers

- Sums may keep only wrong digits. Example:
  - Evaluate
    2.718 281 828 459 05 - 2.718 281 828 459 04
  - … = 9.76996261670138 x 10^{-15}
  - Should have been 10^{-14} !

Using Floating Point Numbers

- Beware of meaningless precision!
  - In 1424, Jamshid Masud al-Kashi published $\pi = 3.141\,592\,653\,589\,793\,25\ldots$
  - …but noted that *the error in computing the perimeter of a circle with a radius 600’000 times that of earth would be less than the thickness of a horse’s hair*.

Using Floating Point Numbers

- Donald E. Knuth :
  - *Floating point arithmetic is by nature inexact, and it is not difficult to misuse it so that the computed answers consist almost entirely of "noise". One of the principal problems of numerical analysis is to determine how accurate the results of certain numerical methods will be.*

Using Floating Point Numbers

- Never use equality between two floating point numbers.
- Use a special method to compare them.

Using Floating Point Numbers

- Proposed extensions to class Number

```
relativelyEqualsTo: aNumber upTo: aSmallNumber
    | norm |
    norm := self abs max: aNumber abs.
    ^norm <= aSmallNumber
             or: [ (self - aNumber) abs <
             ( aSmallNumber * norm)]

equalsTo: aNumber
    ^self relativelyEqualsTo: aNumber upTo:
             DhbFloatingPointMachine default
             defaultNumericalPrecision
```

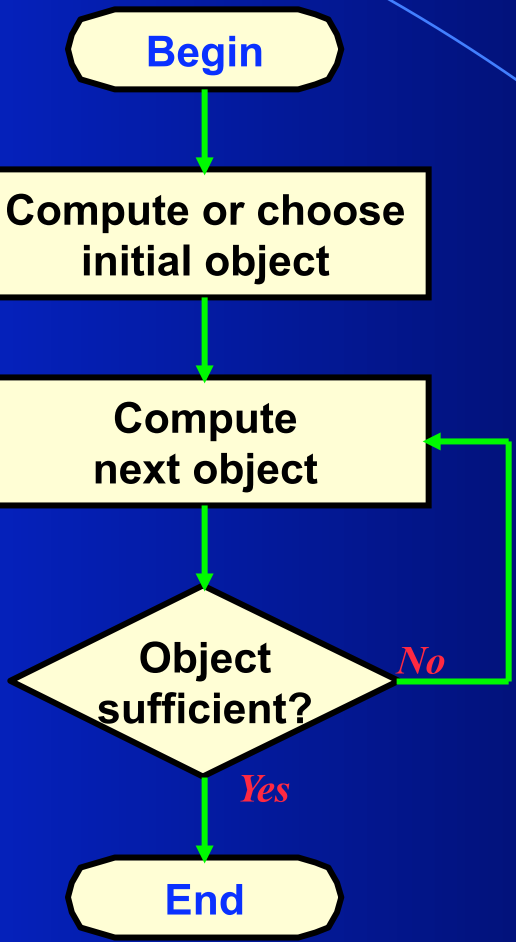
Road Map

- Dealing with numerical data
- Framework for iterative processes
- Newton's zero finding
- Eigenvalues and eigenvectors
- Cluster analysis
- Conclusion

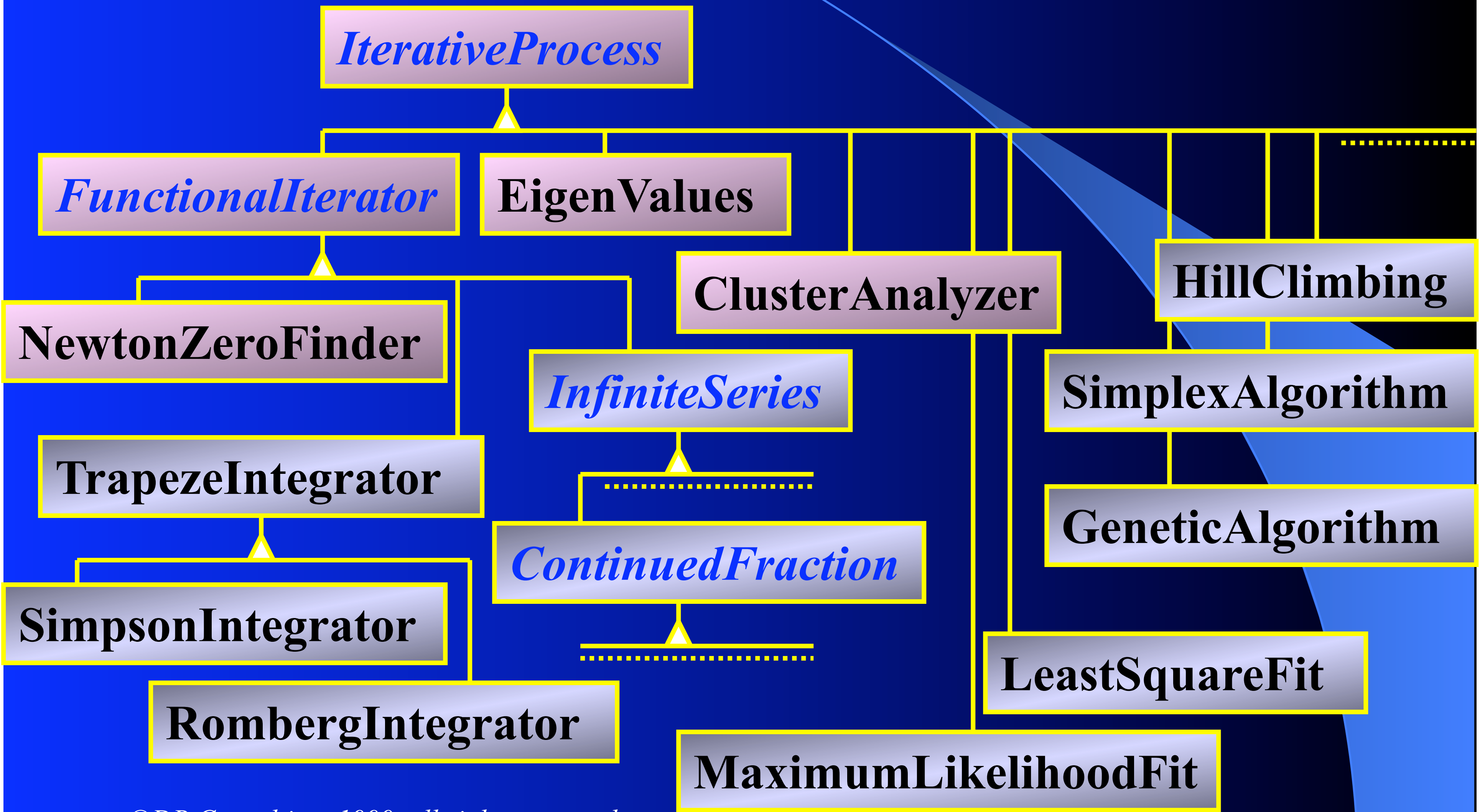
Iterative Processes

- Find a result using successive approximations
- Easy to implement as a *framework*
- Wide field of application

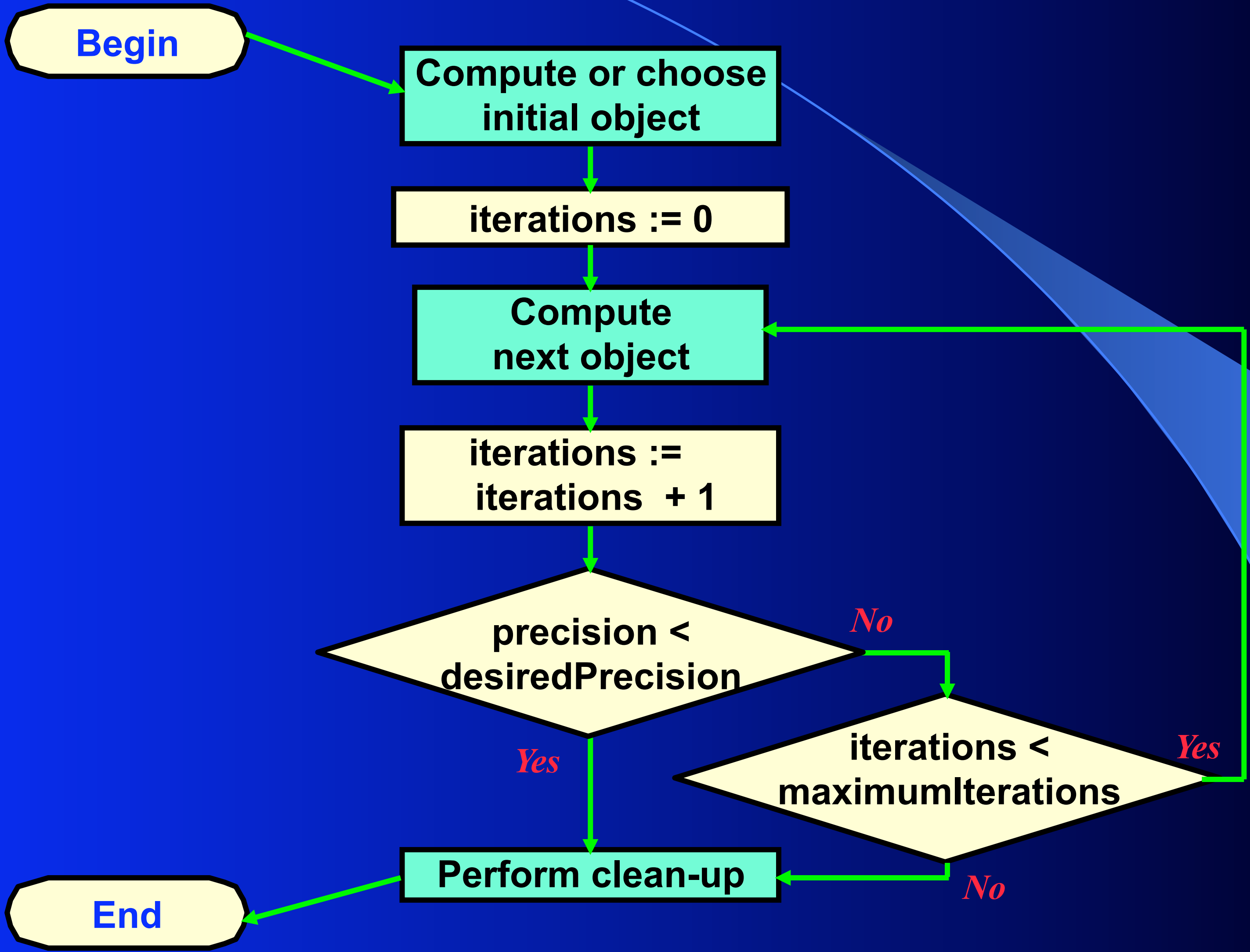
Successive Approximations



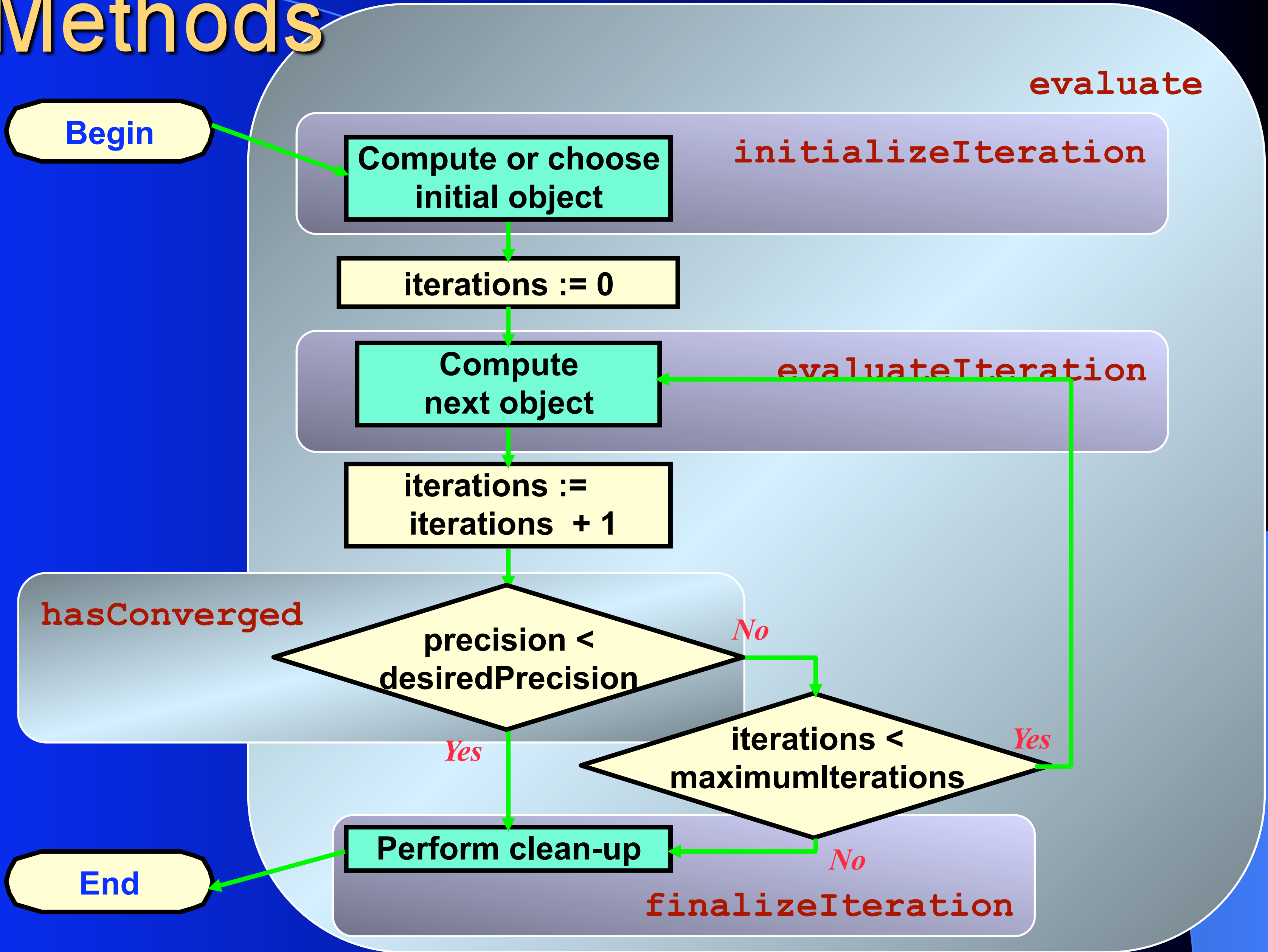
Iterative Processes



Implementation



Methods


evaluate
Begin
Compute or choose initial object
initializeIteration
iterations := 0
Compute next object
evaluateIteration
iterations := iterations + 1
hasConverged
precision < desiredPrecision
No
Yes
iterations < maximumIterations
Yes
No
Perform clean-up
finalizeIteration
End

Simple example of use

```
| iterativeProcess result |
iterativeProcess := <a subclass of
            IterativeProcess> new.
result := iterativeProcess evaluate.
iterativeProcess hasConverged
   ifFalse:[ <special case processing>].
```

Advanced example of use

```
| iterativeProcess result precision |
iterativeProcess := <a subclass of DhbIterativeProcess> new.
iterativeProcess desiredPrecision: 1.0e-6;
          maximumIterations: 25.
result := iterativeProcess evaluate.
iterativeProcess hasConverged
   ifTrue: [ Transcript nextPutAll: 'Result obtained
          after '.
          iterativeProcess iteration printOn: Transcript.
          Transcript nextPutAll: 'iterations. Attained
          precision is '.
          iterativeProcess precision printOn: Transcript.
          ]
   ifFalse:[ Transcript nextPutAll: 'Process did not
          converge'].
Transcript cr.
```

Class Diagram

IterativeProcess	
evaluate	
initializeIteration	
hasConverged	
evaluateIteration	
finalizeIteration	
precisionOf:relativeTo:	
desiredPrecision	(g,s)
maximumIterations	(g,s)
precision	(g)
iterations	(g)
result	(g)

Case of Numerical Result

- Computation of precision should be made relative to the result
- General method:

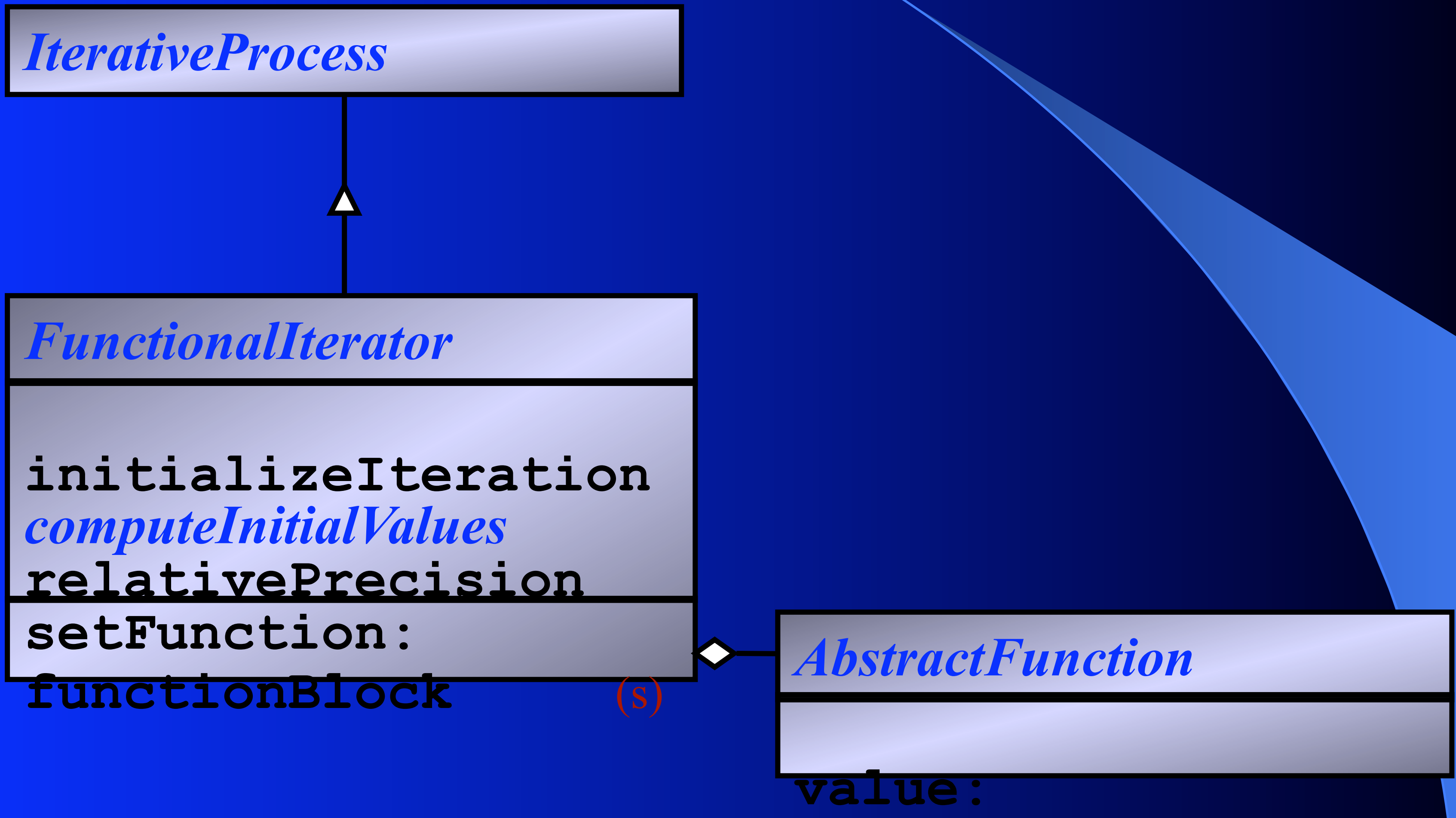
Absolute precision *Result*

```
precisionOf: aNumber1 relativeTo: aNumber2
   ^aNumber2 > DhbFloatingPointMachine
                    default
                    defaultNumericalPrecision
        ifTrue: [ aNumber1 / aNumber2]
        ifFalse:[ aNumber1]
```

Example of use

```
| iterativeProcess result |
iterativeProcess := <a subclass of
          FunctionalIterator> function:
          ( DhbPolynomial new: #(1 2 3).
result := iterativeProcess evaluate.
iterativeProcess hasConverged
   ifFalse:[ <special case processing>].
```

Class Diagram



Road Map

- Dealing with numerical data
- Framework for iterative processes
- Newton's zero finding
- Eigenvalues and eigenvectors
- Cluster analysis
- Conclusion

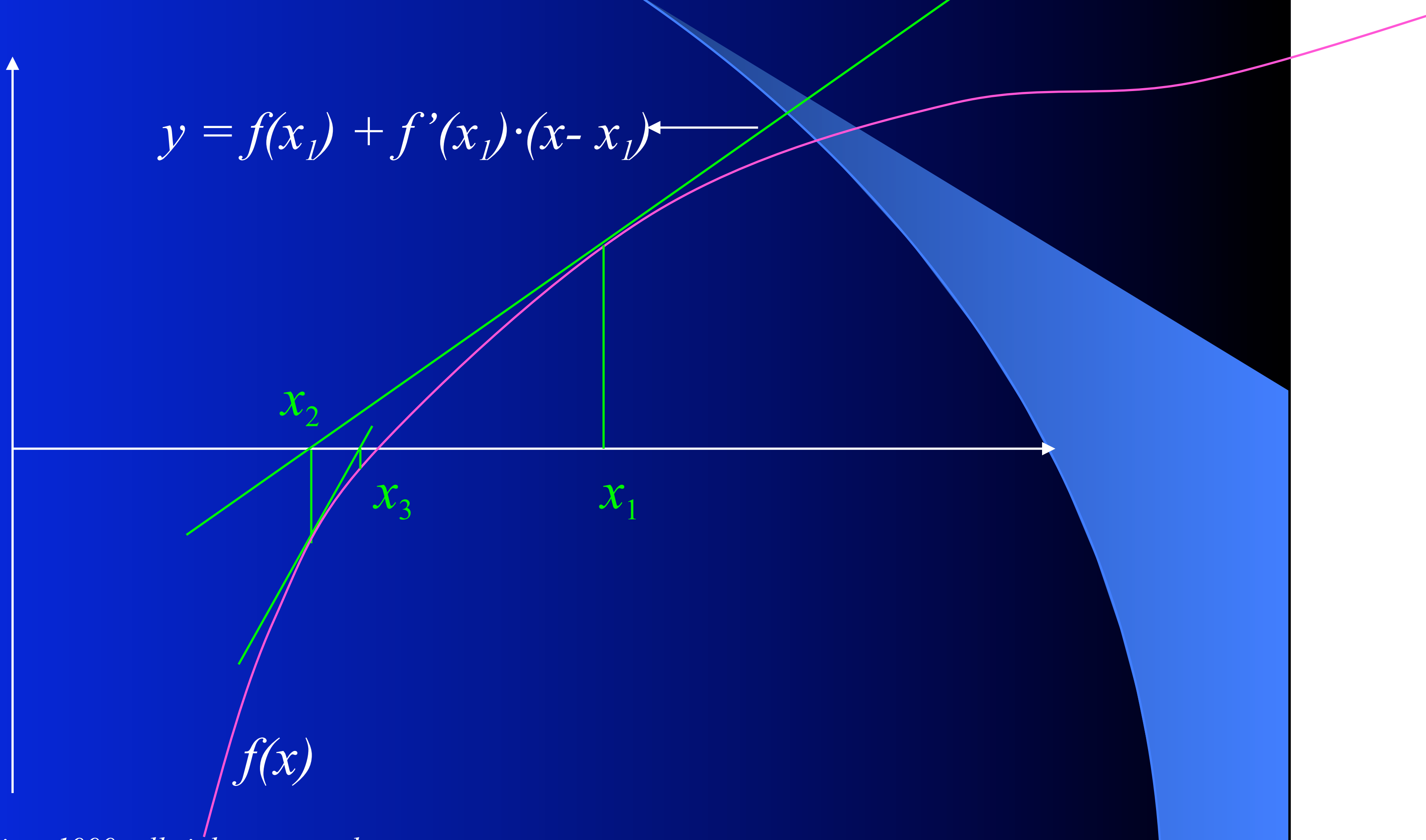
Newton's Zero Finding

- It finds a value x such that $f(x) = 0$.
- More generally, it can be used to find a value x such that $f(x) = c$.

Newton's Zero Finding

- Use successive approximation formula
  $x_{n+1} = x_n - f(x_n)/f'(x_n)$
- Must supply an initial value
  - overload **`computeInitialValues`**
- Should protect against $f'(x_n) = 0$

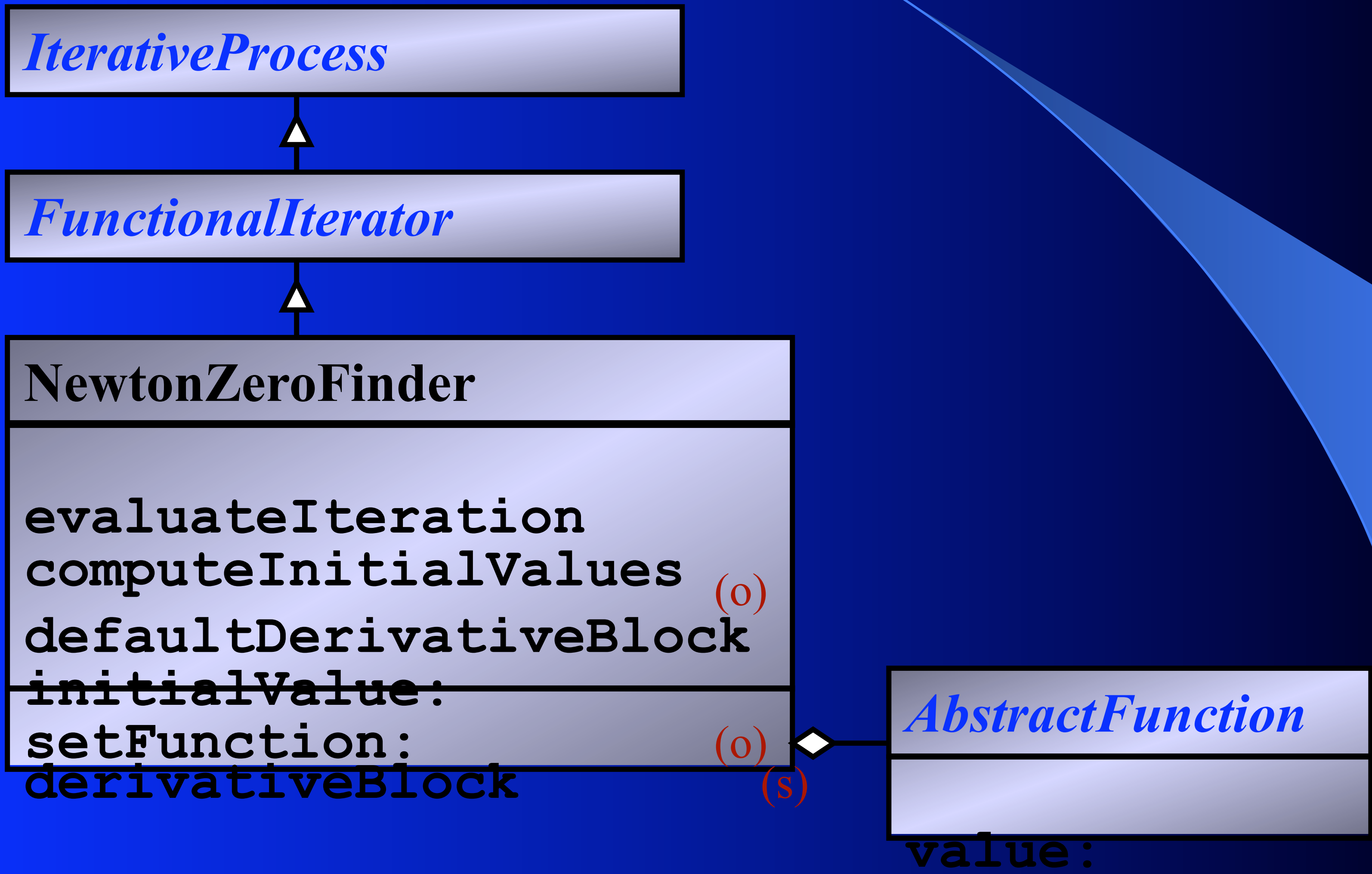
Newton’s Zero Finding



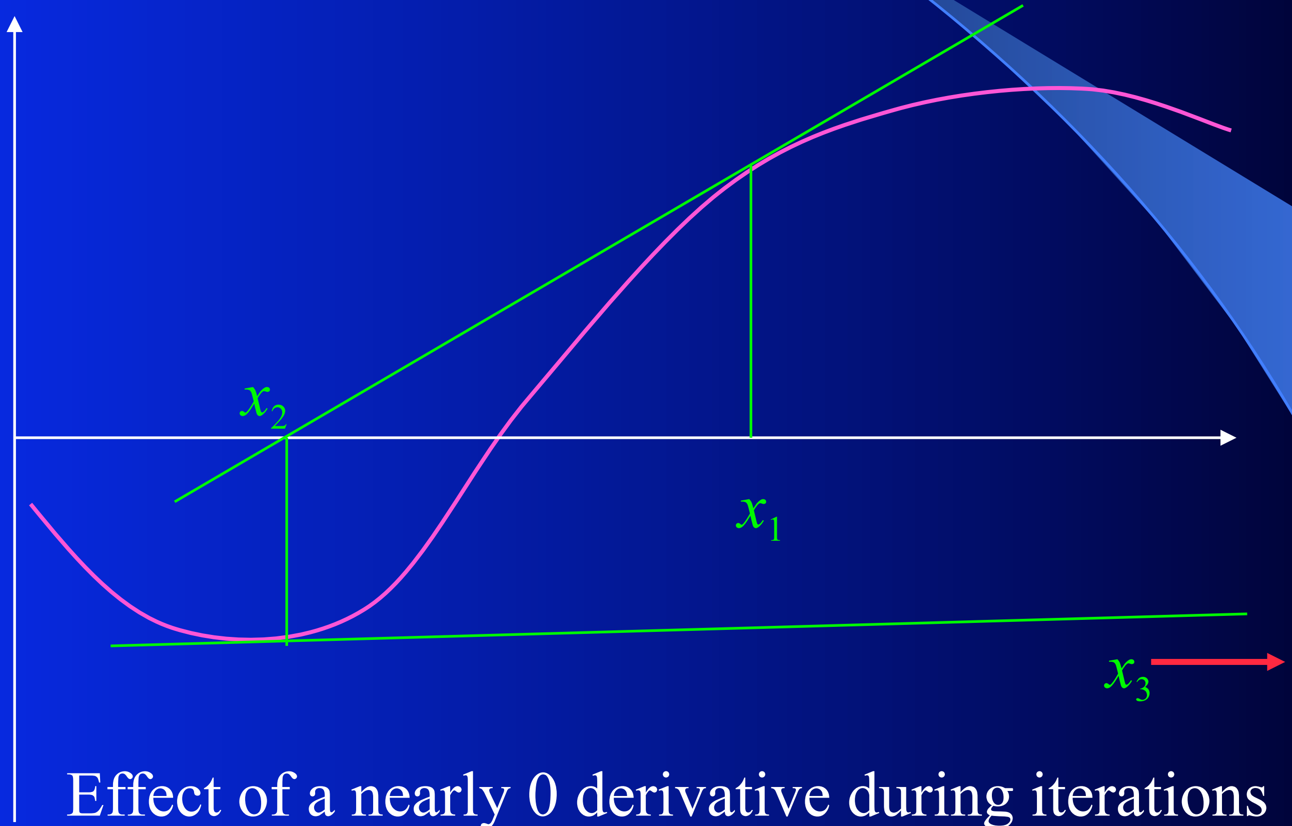
Example of use

```
| zeroFinder result |
zeroFinder:= DhbNewtonZeroFinder
         function: [ :x | x
         errorFunction - 0.9]
derivative: [ :x | x normal].
zeroFinder initialValue: 1.
result := zeroFinder evaluate.
zeroFinder hasConverged
   ifFalse:[ <special case processing>].
```

Class Diagram



Newton's Zero Finding



Effect of a nearly 0 derivative during iterations

Road Map

- Dealing with numerical data
- Framework for iterative processes
- Newton's zero finding
- Eigenvalues and eigenvectors
- Cluster analysis
- Conclusion

Eigenvalues & Eigenvectors

- Finds the solution to the problem

  $$\mathbf{M} \cdot \mathbf{v} = \lambda \cdot \mathbf{v}$$

  where **M** is a square matrix and **v** a non-zero vector.
- λ is the *eigenvalue*.
- **v** is the *eigenvector*.

Eigenvalues & Eigenvectors

- Example of uses:
  - Structural analysis (vibrations).
  - Correlation analysis
    (statistical analysis and data mining).

Eigenvalues & Eigenvectors

- We use the Jacobi method applicable to symmetric matrices only.
- A 2x2 rotation matrix is applied on the matrix to annihilate the largest off-diagonal element.
- This process is repeated until all off-diagonal elements are negligible.

Eigenvalues & Eigenvectors

- When **M** is a symmetric matrix, there exist an orthogonal matrix **O** such that:

  $\mathbf{O}^T \cdot \mathbf{M} \cdot \mathbf{O}$ is diagonal.
- The diagonal elements are the eigenvalues.
- The columns of the matrix **O** are the eigenvectors of the matrix **M**.

Eigenvalues & Eigenvectors

- Let i and j be the indices of the largest off-diagonal element.
- The rotation matrix $\mathbf{R}_1$ is chosen such that $(\mathbf{R}_1^{\mathrm{T}} \cdot \mathbf{M} \cdot \mathbf{R}_1)_{ij}$ is 0.
- The matrix at the next step is: $\mathbf{M}_2 = \mathbf{R}_1^{\mathrm{T}} \cdot \mathbf{M} \cdot \mathbf{R}_1$.

Eigenvalues & Eigenvectors

- The process is repeated on the matrix $\mathbf{M}_2$.
- One can prove that, at each step, one has:
  $\max |(\mathbf{M}_n)_{ij}| <= \max |(\mathbf{M}_{n-1})_{ij}|$ for all $i \neq j$.
- Thus, the algorithm must converge.
- The matrix **O** is then the product of all matrices $\mathbf{R}_n$.

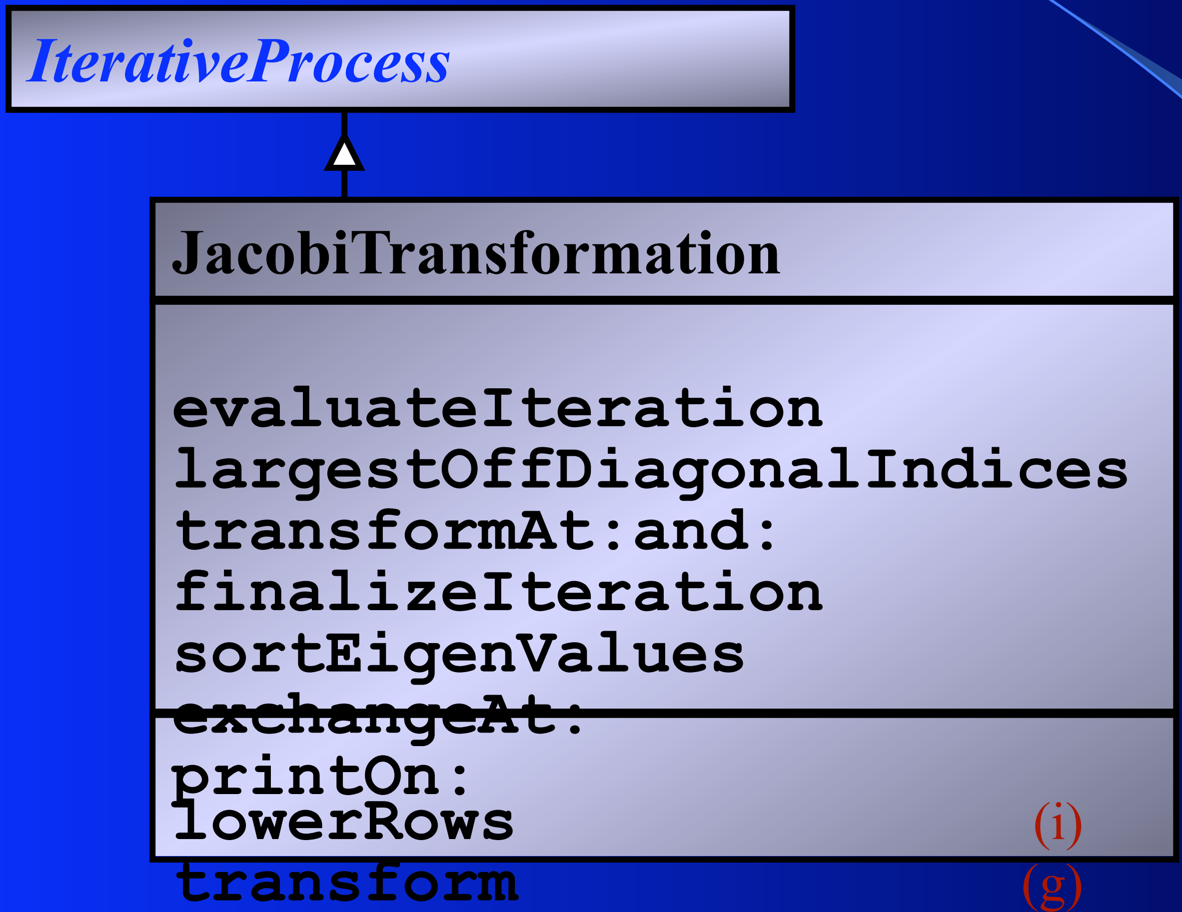
Eigenvalues & Eigenvectors

- The *precision* is the absolute value of the largest off-diagonal element.
- To finalize, eigenvalues are sorted in decreasing order of absolute values.
- There are two results from this algorithm:
  - A vector containing the eigenvalues,
  - The matrix **O** containing the corresponding eigenvectors.

Example of use

```
| matrix jacobi eigenvalues transform |
matrix := DhbMatrix rows: #( ( 3 -2 0)
               (-2 7 1) (0 1 5)).
jacobi := DhbJacobiTransformation new:
               matrix.
jacobi evaluate.
iterativeProcess hasConverged
   ifTrue:[ eigenvalues := jacobi result.
            transform := jacobi transform.
           ]
   ifFalse:[ <special case processing>].
```

Class Diagram



Road Map

- Dealing with numerical data
- Framework for iterative processes
- Newton's zero finding
- Eigenvalues and eigenvectors
- Cluster analysis
- Conclusion

Cluster Analysis

- Is one of the techniques of *data mining*.
- Allows to extract unsuspected similarity patterns from a large collection of objects.
- Used by marketing strategists (banks).
- Example: BT used cluster analysis to locate a long distance phone scam.

Cluster Analysis

- Each object is represented by a vector of numerical values.
- A metric is defined to compute a similarity factor using the vector representing the object.

Cluster Analysis

- At first the centers of each clusters are defined by picking random objects.
- Objects are clustered to the nearest center.
- A new center is computed for each cluster.
- Continue iteration until all objects remain in the cluster of the preceding iteration.
- Empty clusters are discarded.

Cluster Analysis

- The *similarity* metric is essential to the performance of cluster analysis.
- Depending on the problem, a different metric may be used.
- Thus, the metric is implemented with a *STRATEGY* pattern.

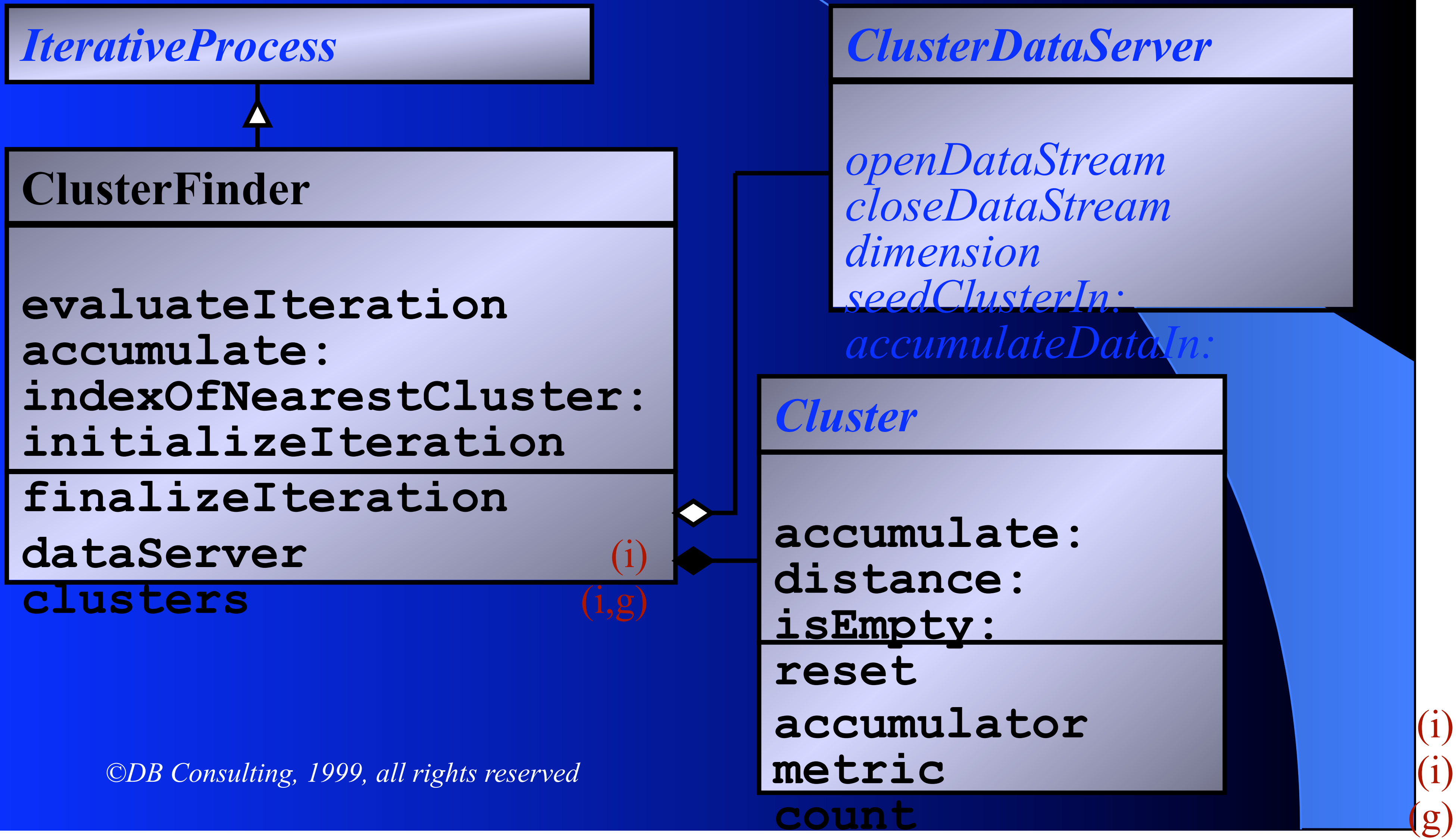
Cluster Analysis

- The *precision* is the number of objects that changed clusters at each iteration.
- The result is a grouping of the initial objects, that is a set of clusters.
- Some clusters may be better than others.

Example of use

```
| server finder |
server := <a subclass of DhbAbstractClusterDataServer >
                new.
finder := DhbClusterFinder new: 15
                server: server
                type: DhbEuclidianMetric.
finder evaluate.
finder tally
```

Class Diagram



Conclusion

- When used with care numerical data can be processed with Smalltalk.
- OO programming can take advantage of the mathematical structure.
- Programming numerical algorithms can take advantage of OO programming.

Questions

?

To reach me: didier@ieee.org