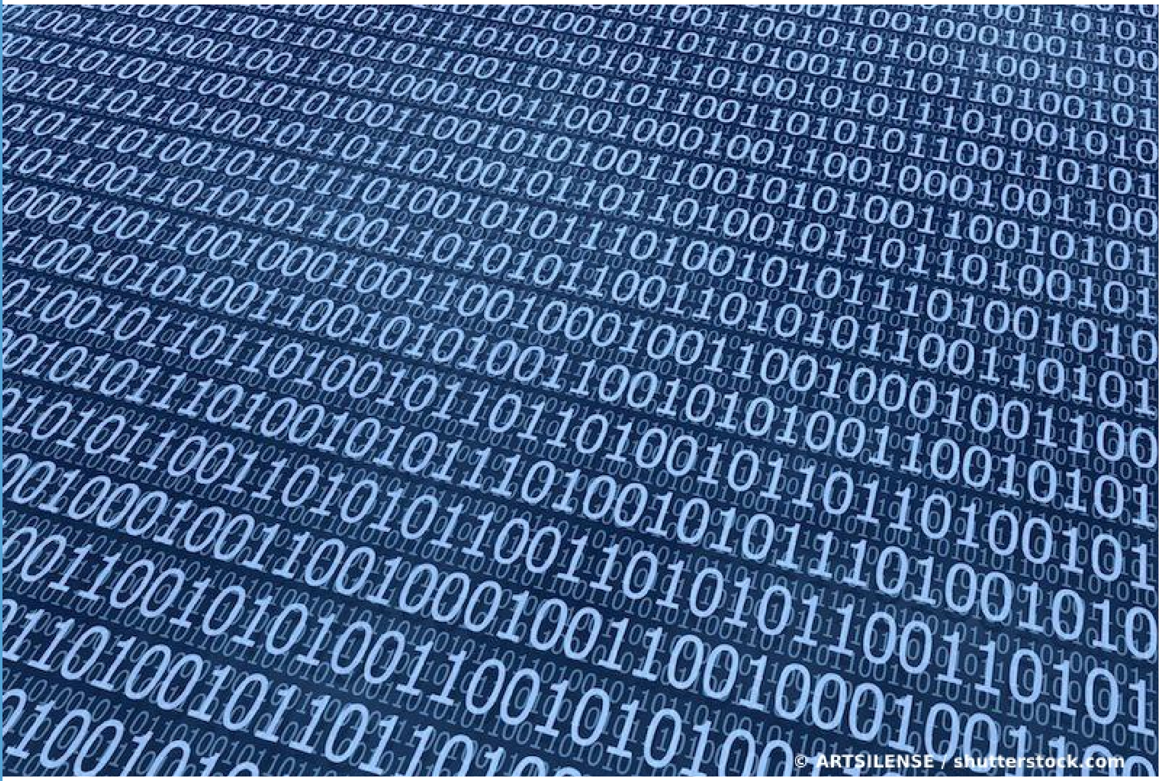


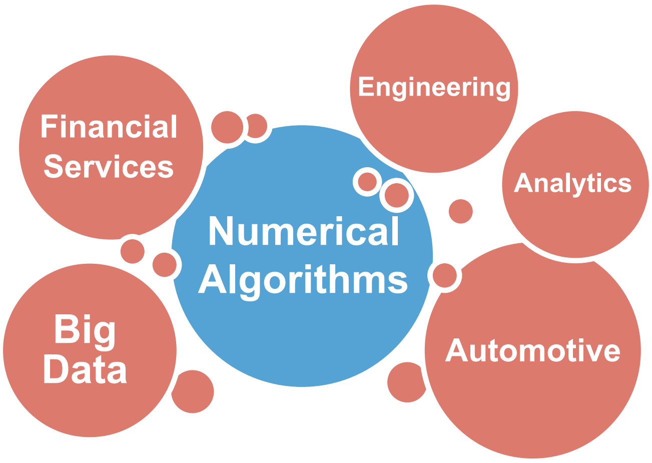
© ARTSILENSE / shutterstock.com

Numeric Computing for Industry

ILNumerics

Executive Summary

Mathematical algorithms are on the rise in software development: nowadays, companies in nearly every business sector have to handle huge amounts of structured information efficiently. Recent trends such as predictive analytics have increased this demand: high performance computing is no longer just a requirement for a few high tech companies, but for all growing industries. In engineering, analytics, automotive, financial services, innovation is tightly coupled to fast mathematical algorithms.



Modern programming languages and developer tools have made it much easier to design complex software architectures. However, these tools have not yet been capable of providing the execution performance required for demanding mathematical algorithms.

That is why programming languages of the last century – such as C/C++ and FORTRAN – are still utilized for these parts of modern enterprise software. Using these technologies requires enormous effort, both in development and in maintenance.

ILNumerics cuts down the costs of development and maintenance for performance critical enterprise software by up to 50 per cent.

ILNumerics extends the modern .NET framework with efficient tools for the design and implementation of mathematical modules. It closes the gap between high performance mathematical algorithms and the efficiency provided by modern software development frameworks. ILNumerics noticeably speeds up the transition of algorithms from prototypes into productive software applications.

Complex industrial software particularly profits from ILNumerics: development cycles are much shorter; the results are more stable, and the cost of maintenance is drastically reduced.

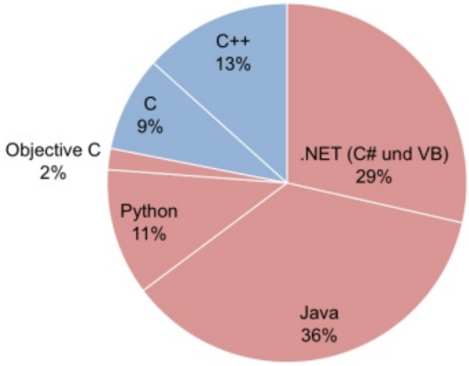
"THE IMPROVEMENT IN PERFORMANCE AND CAPABILITY THAT WE HAVE OBSERVED AS WE USED ILNUMERICS OVER THE LAST YEAR IN OUR PROJECT UNDER DEVELOPMENT HAS BEEN ASTONISHING."

ZEEKO LTD

Numerical Algorithms and Software Development

Developing mathematical modules for software applications is a demanding task.

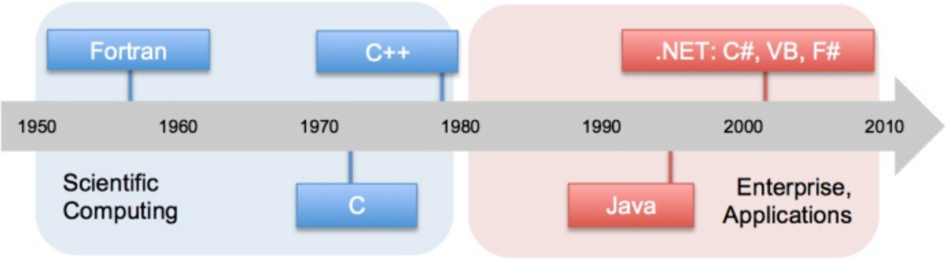
In general, modern software frameworks such as Java or .NET have made it much easier to develop all sorts of enterprise software – especially for complex software architectures. However, developers still do not use these frameworks to implement mathematical algorithms. Until now, the performance level these algorithms require has not yet been achieved with Java or .NET, but only by using traditional programming languages such as FORTRAN or C++. Several attempts to improve the suitability of the Java platform for high performance computing have been suspended due to insurmountable technical hurdles.



JOB OFFERS BY PROGRAMMING LANGUAGES 2012-09 TO 2013-09, BLUE: TRADITIONAL LANGUAGES, RED: MANAGED LANGUAGES (TRENDYSKILLS.COM)

As soon as complex mathematical algorithms come into play, software developers needed to use programming languages from the last century.

That's why there is a gap: as soon as complex mathematical algorithms need to be implemented into modern 21st century software applications, developers have to use 20th century programming languages. This gap is a problem for many industrial domains: research and development departments in engineering; financial services; automotive, and many more still do not benefit from the advantages of modern software development.



SCIENTIFIC COMPUTING AND MODERN PROGRAMMING LANGUAGES

ILNumerics offers a way to close this gap between scientific computing and modern application development: by speeding up numerical algorithms in .NET, high performance computing can finally be realized in a modern managed software framework. Thus modern software tools can be used for the development of high performance applications – for the very first time without any restrictions.

High Performance Computing for Numerical Algorithms

There are two major reasons why software development is more convenient and efficient with .NET. Firstly, modern object-oriented programming languages allow programmers to write high level code, which abstracts away platform specifics and makes it easier to handle large projects.

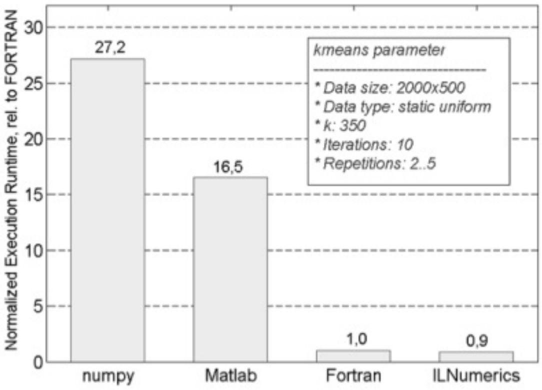
Secondly, automatic memory management helps avoid the serious errors that can happen when developers use native languages such as C or FORTRAN.

Automatic memory management is the reason for low performance of managed software frameworks when it comes to mathematical algorithms and large data.

Therefore, ILNumerics provides its own automatic memory management: it is optimized for numerical algorithms and consequently re-uses memory blocks for all mathematical objects. In ILNumerics, these objects (*intelligent arrays*) contain information about their lifetime; this allows the ILNumerics memory management to collect used memory in a pool and to avoid unnecessary reallocations.

"THE LIBRARY IS WELL DESIGNED, BRINGS HIGH-PERFORMANCE AND THE MEMORY MANAGEMENT IS AMAZING."

T. WAWRZIK,
TECHNICAL UNIVERSITY
BRAUNSCHWEIG



THE PERFORMANCE OF ILNUMERICS CATCHES UP WITH NATIVE LANGUAGES

ILNumerics circumvents .NET's automatic memory management using modern OOP principles and realizes the required high execution speed of numerical algorithms while retaining optimal syntactical convenience.

.NET provides extended optimization options

In addition to the efficient memory management, ILNumerics uses .NET's extended optimization features which are not, for example, available on the Java platform. This includes methods which have made native languages such as C/C++ so fast: pointer arithmetics, cache optimization, and loop unrolling are all utilized by ILNumerics internally.

All this gives mathematical algorithms on the.NET framework a performance similar to traditional native languages – right within a modern software framework.

Advantages of ILNumerics

ILNumerics' striking performance makes all advantages of the modern managed .NET framework available for the development and implementation of high performance algorithms into enterprise software.

1. **Improved maintainability**: the code of numerical algorithms is more readable and future-proof, making it easier to maintain complex modules efficiently, even for large teams.
2. Using **Platform Invoke**, .NET seamlessly connects with native modules written in traditional programming languages.
3. **Modern tools for .NET** help to optimize code: Profilers, Code Analyzers, Refactoring Tools etc.
4. **.NET's backend technologies** (e.g. database- and network-connectivity) allow developers to integrate numerical modules into complex software architectures.
5. There is excellent support for **Graphical User Interfaces (GUIs)** in .NET.
6. In addition, algorithms developed with ILNumerics offer a **wide platform support**: they can be executed on both Windows and Linux architectures. MacOS, iOS, Android & Co. are on their way.

But that's not all! In addition to the convenience offered by .NET framework, ILNumerics provides its own set of advantages: intuitive array syntax; powerful visualization features; a large collection of built-in functions; the possibility of connecting to professional big data formats, and a seamless integration into Visual Studio.

"THE SIMPLEST WAY TO ADD HIGH PERFORMANCE NUMERICS TO .NET APPLICATIONS."

Machine Intelligence

MACHINE INTELLIGENCE LTD

"SIMPLE, CLEAR, ALL-IN-ONE – THAT'S IT."

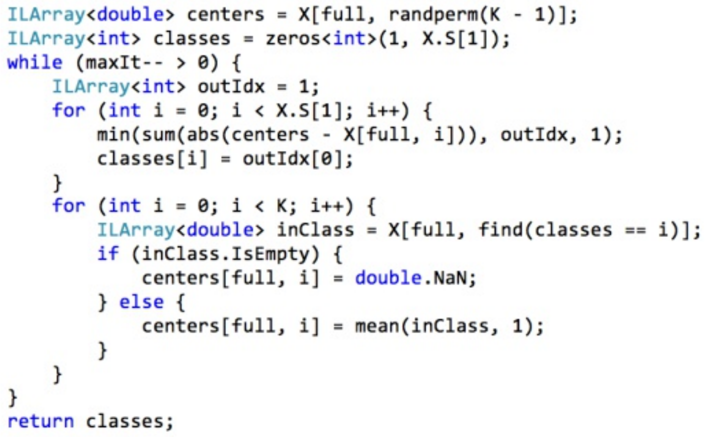
VADIM VIGASIN, TZAFON

"IT ENABLES PEOPLE TO USE C# AS MATLAB®. VERY COOL!"

MATT XIUQUAN, JDSU

Syntax

ILNumerics extends regular .NET-languages with mathematical expressions,. based on the syntax used in popular mathematical prototyping software. This syntax makes it much easier to implement mathematical modules into application code: the code of mathematical prototypes can be used almost 'as is', and numerical algorithms become part of new software applications much faster than ever before.



```
ILArray<double> centers = X[full, randperm(K - 1)];
ILArray<int> classes = zeros<int>(1, X.S[1]);
while (maxIt-- > 0) {
    ILArray<int> outIdx = 1;
    for (int i = 0; i < X.S[1]; i++) {
        min(sum(abs(centers - X[full, i])), outIdx, 1);
        classes[i] = outIdx[0];
    }
    for (int i = 0; i < K; i++) {
        ILArray<double> inClass = X[full, find(classes == i)];
        if (inClass.IsEmpty) {
            centers[full, i] = double.NaN;
        } else {
            centers[full, i] = mean(inClass, 1);
        }
    }
}
return classes;
```

A NAIVE KMEANS ALGORITHM IN ILNUMERICS

Thus, ILNumerics' intuitive syntax not only makes it easier to port mathematical algorithms from tools like Matlab® to .NET; it also offers new possibilities to use .NET as an integrated platform for both the design of mathematical algorithms, and their implementation into application software. Mathematical prototyping and application development merge into a single process.

"IN A .NET ENVIRONMENT, THE COMBINATION OF ILNUMERICS AND HDF5 IS A WIN-WIN SITUATION: STARTING FROM DATA STORED IN HDF5, ILNUMERICS IS THE LOGICAL NEXT STEP FOR DATA ANALYSIS AND MANIPULATION."

G. HEBER, HDF GROUP

Interfacing Professional Data Formats

In order to handle large data in mathematical algorithms, developers need to import and export data to/from their development environment. .NET already brings flexible options: text, binary, SOAP, ODBC, and Office formats. ILNumerics adds interfaces for Matlab® .m files and HDF5 – the quasi industry standard for professional data exchange.

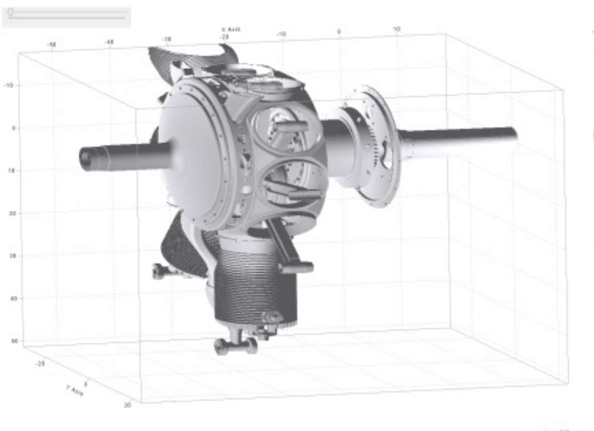
Built-In Functions

In order to make it easier to design individual algorithms, ILNumerics offers a large collection of frequently used numerical functions. These built-in functions include trigonometric functions, matrix decompositions, equation solvers, eigenvalues, efficient sorting, and integral transformations.

Visualization

The ILNumerics Visualization API eases the implementation of complex interactive visualizations for Windows and Linux applications.

Visualization is key to handling complex data. Enabling technical applications for interactive 3D visualizations is another frequently needed, but demanding, task. ILNumerics offers a powerful Visualization API which eases the implementation of complex interactive visualizations for Windows and Linux applications.

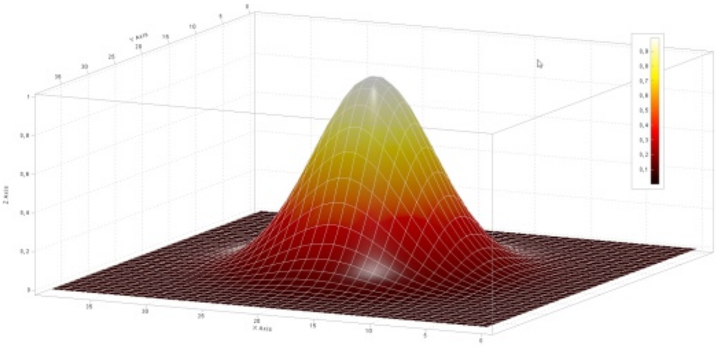


CAD MODEL IMPLEMENTED WITH ILNUMERICS (MODEL BY STEPHANE BOUYE)

ILNumerics' scene graph harnesses a clear and efficient memory management to large, interactive, three-dimensional scenes; for hardware acceleration, OpenGL is used. DirectX is on the way.

ILNumerics includes classes for 2D line plots; 2D and 3D contour plots, and surface and mesh plots.

One focus of the ILNumerics Visualization API is on scientific plotting: ILNumerics includes classes for 2D line plots (including markers, custom dash styles, legends); 2D and 3D contour plots, and surface and mesh plots. All visualizations can be conveniently exported to SVG or bitmap formats. Furthermore, users can assemble custom plots from the ILNumerics scene graph objects.



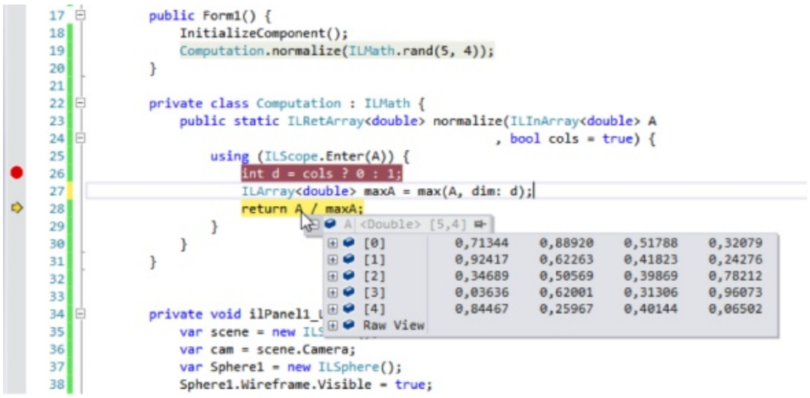
ILNUMERICS ALLOWS THE CREATION OF HIGH-END PLOTS WITH FLEXIBLE CONFIGURATIONS

"I DON'T KNOW OF A SIMILAR LIBRARY THAT MATCHES ILNUMERICS' CAPABILITIES."

B. GEORGE,
SHARP STATISITICS

Visual Studio Integration

Visual Studio is possibly the most popular software for developing applications today. Used together with ILNumerics, it becomes an efficient tool for creating numerical software modules. Through ILNumerics' extended debugging support, the profiles and content of mathematical algorithms can be inspected at run time.



```
public Form1() {
    InitializeComponent();
    Computation.normalize(ILMath.rand(5, 4));
}

private class Computation : ILMath {
    public static ILRetArray<double> normalize(ILInArray<double> A
                                              , bool cols = true) {
        using (ILScope.Enter(A)) {
            int d = cols ? 0 : 1;
            ILArray<double> maxA = max(A, dim: d);
            return A / maxA;
        }
    }
}


private void ilPanel1_L
    var scene = new ILS
    var cam = scene.Camera;
    var Sphere1 = new ILSphere();
    Sphere1.Wireframe.Visible = true;
```

```
A <Double> [5,4]
[0]        0,71344   0,88920   0,51788   0,32079
[1]        0,92417   0,62263   0,41823   0,24276
[2]        0,34689   0,50569   0,39869   0,78212
[3]        0,03636   0,62001   0,31306   0,96073
[4]        0,84467   0,25967   0,40144   0,06502
Raw View
```

DEBUGGING A NUMERICAL ALGORITHM IN VISUAL STUDIO.

© rawcaptured / shutterstock.com

Case Study 1: Quality Assurance

With ILNumerics, high performance enterprise applications become much more stable and maintainable.

A semi-conductor company produces wafers and needs to improve quality assurance. It commissions a software application which visualizes deviations from the norm on the basis of points measured on the wafers' surface.

Without ILNumerics, the prototypical algorithm is developed with numpy, an extension to the popular programming language Python. In order to incorporate this algorithm into the final software application, interfaces to Python help accessing numpy from .NET. To realize the visualizations, the developers use third party modules. The final application contains three technologies: Python, .NET, and the modules needed for the visualization feature. For all future changes, appropriate expertise and licences are needed.

With ILNumerics, both the algorithm and the visualization feature can be implemented into the .NET application code directly, which means the application becomes much more stable and maintainable. A connection to external modules is not necessary. Future updates can be made with low costs. The company saves 30 per cent on development costs.

© Viktor Zadorozhnyi / shutterstock.com

Case Study 2: Risk Management Software

ILNumerics cuts down costs for research & development by up to 50 per cent.

In order to simulate credit losses, a bank wants to develop a new software application. This application is to be used in every branch of the bank.

Without ILNumerics, the research and development department uses mathematical prototyping software to create a prototype. In the second stage, this algorithm is implemented into the new software application.
To achieve the performance that is needed, the developers decide to use C++ and the whole algorithm has to be re-written from scratch.
All future changes to the algorithm must be applied to both domains – separately, for each platform involved.

With ILNumerics, the research and development department designs the algorithm directly in .NET. Because of the excellent performance of ILNumerics, the same code can be used for the new software application.
All future changes take little effort as they can be made directly in the application code. A single application is created. It runs on all platforms without modifications. This cuts costs by 50 per cent.

© Nataliya Hora / shutterstock.com

Case Study 3: Optimizing Performance

ILNumerics allows the optimization of execution speed – without any native C-modules.

The research and development department of a large automotive company has been using .NET applications for mathematical tasks for years. These tools are used on both Windows and Linux systems. In order to improve certain operations, the execution speed of some of these tools needs to be optimized.

Without ILNumerics, performance-critical parts are implemented into native modules written in C and linked to the corresponding software tools. The tools lose their wide platform support and future versions for Windows and Linux must be developed separately.

With ILNumerics, no native modules are needed: the performance critical parts can be implemented directly in a high level .NET. language. The tools still support multiple platforms – they run on both Windows and Linux. The company saves 30 per cent on development and maintenance.

Conclusion

ILNumerics brings a new level of efficiency to a wide spectrum of enterprise software development.

ILNumerics brings together performance and convenience: for the first time, high performance mathematical computations can be realized in a modern enterprise software framework. By providing an efficient memory management, utilizing several advantages of .NET, and offering an intuitive syntax, as well as flexible visualization features, ILNumerics makes it much easier to develop demanding software applications reliably.

Industrial users benefit from drastically shortened development cycles; budgets for research and development in finance, engineering, analytics and many other industries are hugely reduced. In this way, ILNumerics brings a new level of efficiency to a wide spectrum of enterprise software development.

Comparison between ILNumerics and Matlab®

		Matlab®	ILNumerics (C#)

Array Handling	creation	ones(4,3,2)
		ones(4,3,2,'single')
			ones(4,3,2)
			ones<float>(4,3,2)

		zeros(4,3,2)
		zeros(4,3,2,'int32')
			zeros(4,3,2)
			zeros<int>(4,3,2)

		eye(10,20)
		eye(10,20,'single')
			eye(10,20)
			eye<float>(10,20)

		horzcat(A,B)
		vertcat(A,B)
			horzcat(A,B)
			vertcat(A,B)

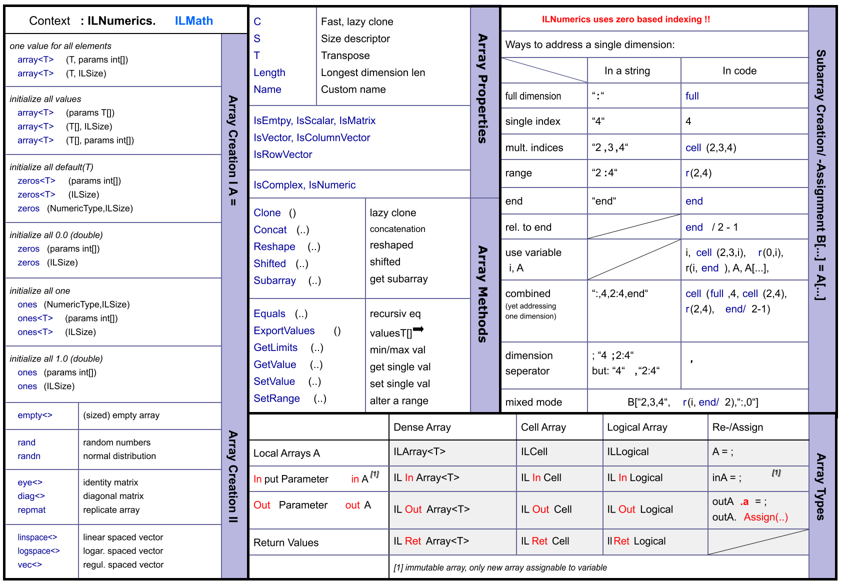
		rand, randn, repmat, reshape, trace ...	rand, randn, repmat, reshape, trace ...

	size info	size(A,1)
		length(A)
			A.S[0] or A.Size[0]
			A.Length

		numel(A)	A.S.NumberOfElements numel(A)
		ndims(A)	A.S.NumberOfDimensions
	transpose	A.`	A.T
	conj. transpose	A`	conj(A.T)
Subarrays	single index access	a(5) 1 based, *parenthesis*	a[4] 0 based, *brackets*
	full dimensions	a(1,**:**)	a[0,**full**], a["0;**:**"]
	ranges	a(1:b,2:2:end)	a[r(0,b),r(1,2,end)]
	relative to end	a(:,end/2)	a[full,end/2]
	sequential	a(b)	a[b]
	index lists	a([1,2,1],[2,2,3]) a([1,2,1],[2,2,3])	a[cell(0,1,0),cell(1,1,2)] a["0,1,0;1,1,2"]
	logical indexing	a(b > 5)	a[b > 5]
	removal	A(2,:,:) = []	A[1,full,full] = null
	modification	A(...) = ...	A[...] = ...
Cells	create by size	cell(3,2)	cell(size(3,2))
	initialize	{10,20,30,40,50}	cell(10,20,30,40,50)
	deep indexing	N/A	A[1,1,0,0,1,2]

		Matlab®	ILNumerics (C#)
Operators	binary operator (element wise)	./ .* .^ < > >= <= == ~= & |	+ - / * pow() < > >= <= == != & |
	allowed sizes, (binary ops.)	array, array: same size array, scalar: always	array, array: same size array, scalar: always *matrix, vector: matching*
	matrices	A * B * C	multiply(A,B,C)
	negation	~A	-A
	vector-matrix binary operators	M - repmat(V,1,n) bsxfun(@minus, M,V)	M – V
Functions	type conversion	int8(A), int32(A) ...	tobyte(A), toInt32(A) ...
	trigonometric	sin,cos, tan, sinh, cosh, tanh, asin, acos, atan2	
	accumulative	all, any, max, mean, min, prod, sum	
	reduce to scalar	N/A	allall, anyall, maxall, minall, sumall
	rounding	round, ceil, floor, fix	
	conversions	cart2pol, pol2cart, flipud, fliplr, ind2sub, sub2ind	
	algebraic	conj, abs, diff, exp, imag, real, ccomplex, log, log10, mod, sign, sqrt, pow, sort	
	test on null/empty	*N/A*	isnull, isnullorempty,
	state	isempty, isequal, isequalwithequalnans, isfinite, isinf, isnan, isneginf, isposinf	
	linear algebra	lu, qr, svd, pinv, chol, mrdivide /	lu, qr, svd, pinv, chol, linsolve
	fourier transforms	fft, fft2, fftn	fft, fft2, fftn

Handling N-dimensional Arrays in ILNumerics



Context : ILNumerics. ILMath

Array Creation I A =

one value for all elements
array<T> (T, params int[])
array<T> (T, ILSize)

initialize all values
array<T> (params T[])
array<T> (T[], ILSize)
array<T> (T[], params int[])

initialize all default(T)
zeros<T> (params int[])
zeros<T> (ILSize)
zeros (NumericType,ILSize)

initialize all 0.0 (double)
zeros (params int[])
zeros (ILSize)

initialize all one
ones (NumericType,ILSize)
ones<T> (params int[])
ones<T> (ILSize)

initialize all 1.0 (double)
ones (params int[])
ones (ILSize)

Array Creation II

empty<>	(sized) empty array
rand randn	random numbers normal distribution
eye<> diag<> repmat	identity matrix diagonal matrix replicate array
linspace<> logspace<> vec<>	linear spaced vector logar. spaced vector regul. spaced vector

Array Properties

C	Fast, lazy clone
S	Size descriptor
T	Transpose
Length	Longest dimension len
Name	Custom name

IsEmtpy, IsScalar, IsMatrix
IsVector, IsColumnVector
IsRowVector

IsComplex, IsNumeric

Array Methods

Clone ()	lazy clone
Concat (..)	concatenation
Reshape (..)	reshaped
Shifted (..)	shifted
Subarray (..)	get subarray
Equals (..)	recursiv eq
ExportValues ()	valuesT[]➡
GetLimits (..)	min/max val
GetValue (..)	get single val
SetValue (..)	set single val
SetRange (..)	alter a range

Subarray Creation/ -Assignment B[...] = A[...]

ILNumerics uses zero based indexing !!

Ways to address a single dimension:

	In a string	In code
full dimension	":"	full
single index	"4"	4
mult. indices	"2 ,3 ,4"	cell (2,3,4)
range	"2 :4"	r(2,4)
end	"end"	end
rel. to end		end / 2 - 1
use variable i, A		i, cell (2,3,i), r(0,i), r(i, end), A, A[...],
combined (yet addressing one dimension)	":,4,2:4,end"	cell (full ,4, cell (2,4), r(2,4), end/ 2-1)
dimension seperator	; "4 ;2:4" but: "4" ,"2:4"	,
mixed mode	B["2,3,4", r(i, end/ 2),":,0"]	

Array Types

	Dense Array	Cell Array	Logical Array	Re-/Assign
Local Arrays A	ILArray<T>	ILCell	ILLogical	A = ;
In put Parameter in A [1]	IL In Array<T>	IL In Cell	IL In Logical	inA = ; [1]
Out Parameter out A	IL Out Array<T>	IL Out Cell	IL Out Logical	outA .a = ; outA. Assign(..)
Return Values	IL Ret Array<T>	IL Ret Cell	Il Ret Logical	

[1] immutable array, only new array assignable to variable

Insights: The ILNumerics Memory Management

ILNumerics’ memory management is optimized for numerical algorithms. Unlike the.NET Garbage Collector (GC) it reuses memory of large numerical objects. This reuse saves time in the GC, improves processor-cache-performance, and prevents heap fragmentation. Together with other optimizations in ILNumerics, the memory management is an important factor to the high performance of numerical algorithms.

In order to be able to reuse memory of old objects on time, arrays that aren’t needed anymore must be identified early enough. However, one characteristic of managed software environments like .NET is that they hide this kind of information from developers. How is it possible for ILNumerics to access this information anyway?

Intelligent Arrays

ILNumerics takes advantage of modern principles of object-orientated programming. All numerical algorithms are formulated in an expressive mathematical syntax based on numerical objects. These objects carry all common information about the underlying numerical data like the shape and the values of their elements. Additionally, *intelligent arrays* hold information about how long the object is needed by parts of the algorithm.

ILNumerics’ *intelligent arrays* come in two forms:

1. In mathematical algorithms most objects are temporary arrays. They are created automatically during the execution of an algorithm and carry interim results. After the first use, there is no way for a programmer to use the object further. Therefore, the memory of such objects can be reclaimed immediately.
2. Arrays which are used several times within the algorithm are declared as variables by the programmer; all variables are of a long living type.

ILNumerics’ *intelligent arrays* are smart enough to clean up after themselves. By default all objects are created as temporary arrays. Whether transferred to another function as a parameter or by accessing a member of the array – no matter how they are used: after the first use, their memory is immediately collected into the ILNumerics memory pool.

The only way for an array in ILNumerics to survive the first usage is to get converted into a long-living type. The conversion is done automatically, simply by assigning to any variable. ILNumerics tracks the number of references to a memory block and copies memory only when absolutely necessary. This is often refered to as 'reference counting' and 'lazy copy on write'.

For the programmer the memory management is completely transparent. ILNumerics' *intelligent arrays* are able to realize a language very similar to Matlab®. In contrast to other math software, this convenience is not achieved at the expense of performance. Rather, *intelligent arrays* are the foundation for an efficient reuse of memory.

ILNumerics – Numeric Computing for Industry

ILNumerics KG, H. Kutschbach, P. von Bünau, J. Nordholt
Ackerstraße 76 / TU-ACK 76
13355 Berlin

http://ilnumerics.net
info@ilnumerics.net

Photo Credits: shutterstock.com (ARTSILENSE [p.1], rawcaptured [p. 9], Viktor Zadorozhnyi [p. 10], Nataliya Hora [p.11]).

Our Partners:

Gefördert durch:
Bundesministerium
für Wirtschaft
und Technologie
aufgrund eines Beschlusses
des Deutschen Bundestages

eXIST
Existenzgründungen
aus der Wissenschaft

ESF
Europäischer Sozialfonds
für Deutschland

Microsoft
BizSpark
Startup
Microsoft

EUROPÄISCHE UNION