IGLib – Things to be Done

Revision 0

Igor Grešovnik

Contents:

1 Things to be done in IGLib 6

1.1 Introduction 6

1.2 Basic library (IGLib) 6

1.2.1 XML Manipulation 6
1.2.2 Variable Objects Manipulation 7
1.2.3 Exception Management 8
1.2.4 Interface Utilities 8
1.2.5 Application / Library Data 9
1.2.6 Help and documentation system 9

1.3 Forms library (IGForms) 10

1.3.1 Minimal Ftandards for Fibrary Forms 10
1.3.2 Text Style Management Class 10
1.3.3 Toplevel Window Positioning Class 11
1.3.4 Standard controls 11
1.3.5 Fading Message 11
1.3.6 Console form 12
1.3.7 XML Viewer & Editor 12
1.3.8 Text Viewer & Editor 12
1.3.9 Html Viewer 12
1.3.10 Html Editor 12
1.3.11 External applications management 13

1.4 Applications 13

1.4.1 Sendigence 13
1.4.2 Visual Inverse 14
1.4.3 Dragonfly Optimization Server Suite 14

2 Sandbox (this is not part of this report) 1

High Priorities

Uniform way of launching controls:

- Constructors do not launch anything (newer, not only in fading messages)!
- Single method for launching in this thread (no arguments, form must be instantiated before calling this method)
- Several **static** methods for instantiating & launching in this thread
- Several **static** methods for instantiating and launching in parallel threads

XML utilities

Add creation of containers at specific path with automatic creation of parents as necessary
Add Setting text values
Add Searching with eventual creation of final elements representing different types of variables (wih arguments specifying the type attribute). For example, GetElement(…, attribute_name, attribute_value) GetVariable()

Variables:
Implement base vector & matrix classes!
Add parsing functions to the interface!

Numerical libraries:
Create interfaces with Math.Net Iridium! Send suggestions to library authors!

Quick Notes

1 CURRENT TASKS

1.1 Integration of Math.Net with IGLib

1.1.1 Questions and suggestions for authors

1.1.1.1 Vectors & matrices:

Fields containing dimension and components could be made protected instead of private. This would enable creation of custom constructors in extended classes, and also more efficient re-implementation of static creation methods. Also, there should be a protected default (argument-less) constructor.

Because this is not the case, I have to call base constructor in every constructor in the declaration, so I can not execute any code before calling such a constructor. This prevents me of performing some checks of arguments before calling the base constructor. Even better, if components and length were defined as protected and if there was an empty default constructor, then I could re-define constructors in my own desirable way.

Helper method such as `CheckMatchingVectorDimensions` should be defined as protected (rather than private) because it is very likely that they would be used in derived classes as well.

Vector & matrix operations

I suggest that operations should be defined in such a way that they could be used in derived classes, in particular for re-definition of operators such as + or *. Currently, this is not entirely true. Only operations that provide in-place variants of the respective methods can also be utilized in the derived classes. The problem is that a method, in order to be utilized in a derived class, should not return a matrix or a vector, because the returned object in the derived class' definition should be of the derived class and not of the base class.

For example, assume I define an XVector class that inherits from Vector. Then I can define the + operator for this particular class in the following way (omitting some stuff that is not relevant for argumentation):

```
        /// <summary>Addition Operator.</summary>
        public static XVector operator +(XVector u, XVector v)
        {
            XVector ret = new XVector(u.Length);
            u.AddInplace(v);
            return u;
        }
```

Such a thing could not be done by using the Add() method instead of AddInplace(), and therefore it can not be done for operations that can not be done in place.

This could be overcome e.g. by defining the method for »raw vector addition« in the following way:

```
        /// <summary>Raw vector addition.</summary>
        /// <param name="v1">Left summand.</param>
        /// <param name="v2">Right Summand.</param>
        /// <param name="result">Result.</param>
        protected static void AddRaw(MathNet.Numerics.LinearAlgebra.Vector v1,
MathNet.Numerics.LinearAlgebra.Vector v2,
                MathNet.Numerics.LinearAlgebra.Vector result)
        {
```

```
            if (v1 == null || v2 == null || result == null)
                throw new ArgumentNullException("One of the operands is not specified.");
            if (v1.Length!=v2.Length || v1.Length!=result.Length)
                throw new ArgumentNullException("Insonsistent dimensions.");
            for (int i = 0; i < v1.Length; ++i)
                result[i] = v1[i] + v2[i];
        }
```

Efficiency could be improved a bit by taking intednal double[] arrays and performing operation on them instead on vectors. This method does not return anything, and it can as well take as arguments objects of classes that are derived from the `Vector` class. It can therefore be directly applied in derived classes. The approach is equally useful for operations that can not be performed in-place. Here is an exemple how I would define relevant operations in my derived `XVector` class:

```
        /// <summary>In-place vector addition.</summary>
        public void AddInplace(XVector v2)
        {
            AddRaw(this, v2, this);
        }

        /// <summary>In-place vector addition.</summary>
        public XVector Add(XVector v2)
        {
            XVector result = new XVector(this.Length);
            AddRaw(this, v2, result);
            return result;
        }

        /// <summary>Addition Operator.</summary>
        public static XVector operator +(XVector u, XVector v)
        {
            XVector result = new XVector(u.Length);
            AddRaw(u, v, result);
            return result;
        }
```

Of course, the three operations could be defined in the same way in the base `Vector` class.

One drawback of the approach is that I have to construct a new vector object of correct dimensions in methods where such an objects is returned (however, this is also the case in the existing arrangement). This is an potential source of errors since I have to create the vector of correct dimensions (which is difficult to miss in vector additions, but easier in some other operations such as matrix multiplication). With a couple of further improvements of the `Vector` class, this could also be improved, namely by letting the **"raw" method perform proper dimensioning**. In order to do this, one should **define protected default constructor (that would initialize the internal double[] array to null) and provide the Resize() method (that would set the vector to appropriate dimensions)**.

```
        protected static void AddRaw(MathNet.Numerics.LinearAlgebra.Vector v1,
MathNet.Numerics.LinearAlgebra.Vector v2,
                MathNet.Numerics.LinearAlgebra.Vector result)
        {
            if (v1 == null || v2 == null || result == null)
                throw new ArgumentNullException("One of the operands is not specified.");
```

```
        if (v1.Length!=v2.Length)
            throw new ArgumentNullException("Insonsistent dimensions.");
        if (result.Length != v1.Length)
            result.Resize(v1.Length);
        for (int i = 0; i < v1.Length; ++i)
            result[i] = v1[i] + v2[i];
    }
```

Operator in the derived `XVector` class would then be defined in the following way:

```
/// <summary>Addition Operator.</summary>
public static XVector operator +(XVector u, XVector v)
{
    XVector result = new XVector();
    AddRaw(u, v, result);
    return result;
}
```

Note that in this arrangement:

- the “raw method” (i.e. the one that actually performs the operation) performs all the dimension checks, which is less error-prone.
- there is only one method that actually performs the operation, everywhere else this method is called
- this same method can also be applied in the derived classes. The only additional thing that one eventually needs to do in the calling methods is to create an object of the correct type.

Summary:

I suggest slight modifications in classes like Vector or Matrix, which would make easier to derive classes from these classes. These modification would include:

- definition of protected default (argument-less) constructor that would simply initialize the arrays to 0 and set dimensions correspondingly. This constructor is not meant to be used outside derived classes because vectors or matrices with dimension do not make sense
- definition of protected (or even private? – since this utility may be really necessary only within the base class) Resize(int dimension) method that would initialize the array according to the desired dimension. Definition of Resize() could be replaced by definition of setter for Length property, however, this is a bit dangerous because the user of the library could be misled that changing dimensions of vectors or matrices is a very innocent operation that could be performed at any place with little or no safety & efficiency considerations.
- for operations such as addition or multiplication, the so called »raw« methods would be defined in the base class. These methods would not return anything as the result would be one of the arguments. The raw method would also take care of result dimensions. The raw method could be defined as protected, although I would recommend they are defined as public (this should be further considered).

These modifications would not in any way restrict the current use of the corresponding classes. Additional features such as default constructor or the Resize(…) method could only be used in the derived classes, which would mean that the library's appearance to the user would not change at all.

Without these modifications, definition of derived classes out of the library is aggravated. In fact, somebody who needs to derive a class form Matrix or Vector would need to create a wrapper class rather than a derived class (see code below). This is both cumbersome and inefficient.

```
/// <summary>A wrapper class for Vector.</summary>
class MyVector
{
    protected MathNet.Numerics.LinearAlgebra.Vector Base = null;

    public int Length
    {
        get { return Base.Length; }
    }

     public double this[int i]
    {
        get { return Base[i]; }
        set { Base[i] = value; }
    }

      ...

}
```

Further suggestions:

In line with the above suggestions, I also suggest that methods calculating vector & matrix norms are defined as virtual. In this way, definitions can be modified in the derived classes.

Other remarks:

In Vector.cs, in class Vector, method object `ICloneable.Clone()`: error in comment, says "`Creates an exact copy of this matrix.`"; instead of "matrix" there should be "vector".

2 THINGS TO BE DONE IN IGLIB

2.1 Introduction

This document describes what has to be done do achieve a specific goal – implementation of the **optimization server**. When planning these developments, I have other more general goals in mind that will have impact in several areas:

- To establish an environment for rapid application development
- To establish foundation for continuous R&D activities
- To create a line of products that will enable representation of ourselves as a high-tech development group
- To create a stable and standard collection of utilities that can be used in other projects. This will make easier inclusion of fancy additions into all kinds of products ordered by our customers and also efficient development of demonstrative prototypes for marketing actions
- To promote technological excellence of the team

In pursuit of the above mentioned objectives, development of several entities has been launched:

- A general purpose library IGLib will systematically cover the needs of final developments. IGLib will be free open source, and will be publicly released only when it is mature for that.
- Demonstrative applications like *Sendigence*.
- Final show-off applications like the optimization server.

2.2 Basic library (IGLib)

2.2.1 XML Manipulation

2.2.1.1 Class XML – Various Manipulation Methods

Searching within XML tree structures
Creation of specified paths if they do not exist
Getting specified text values
Setting specified text values

2.2.1.2 Class XmlData : XmlDocument

Extends XMLDocument with methods from the XML class.

2.2.1.3 Class DataStore

Functions as basic data units in applications. It is used for storage/manipulation of configuration data as well as for storage/manipulation of operational data.

Central data unit of this class is an XmlData object accessed through Data property with public get{}. Other methods & data are just intended for manipulating data in this object.

Provided functionality is divided to:

- Loading/saving data from/to files
  - There is a modification indicator
- Querying values
- Creating containers
- Setting simple values
- Storing/retrieving data of various types (such as numbers, strings, vectors, matrices, lists of these objects, etc.)

2.2.2 Variable Objects Manipulation

2.2.2.1 Definition of Base Classes and Interfaces

2.2.2.1.1 Object of different kinds

Define general interfaces
Define base classes
Define concrete classes for different types (string counter, scalar, vector, matrix)

Base classes should support interfacing utilities (string parsing and I/O operations)

2.2.2.1.2 Define base classes for variable containers

2.2.2.1.3 Type Management System

This will define various tools available for each calss, such as:

- Conversion utilities (for converting to other types)
- XML definition tools that will enable identification of types i nXML documents

- Ediding tools; these will enable editing elements in XML document with an XML editor in special ways adapted for individual types (maybe this should go to the XML editor module instead?)

2.2.2.2 Extension of existing basic types

Extend vector and matrix types
Interfaces with existing libraries
Tyes that implement standard functionality & provide operations implemented by external libraries

2.2.3 Exception Management

2.2.3.1 Exception Handler Classes

These classes will provide functionality such as catching and re-throwing of exceptions

2.2.3.2 Common Error Reporting, Tracing and Messaging Systems

Will provide unified global error reporting system. There will be a couple of hirearchical reporter classes taht can be utilized for common error reporting. Which class will be used by an application will be dependent on its nature, e.g. whether it uses a console, whether it can use windows forms, etc.
These classes will also provide functionality for logging information to files.

It will be possible to use classes as reporters for other classes, not just for global reporters.

2.2.4 Interface Utilities

2.2.4.1 String Parsing

Basic parsing utilities
Reading of objects of different types from strings

2.2.4.2 String Output

Output of object in diffeernt formats.
General form must be determined (output to streams??)

2.2.5 Application / Library Data

2.2.5.1 Applivation / Library Directories

Automatic retrieval of application directory, version directory, user directory and session directory

2.2.6 Help and documentation system

A unified system, easy to maintain and to use from applications.
Requirements:

- It must be easy to edit help & documentation, this should be possible with freely available existing tools – no additional library module necessary for this, except e.g. for browsing directory structuer if the system will also base on that.
- It must be easy to refer to any piece of information, e.g. by specifying path-like links.
- The system must provide possibility to automatically assign (during initialization) tooltips on forms form files (probably XML?).
- The system must be synchronized with the hirearchical versioning system (in particular with its directory structure). It must be possible to have common help and documentation for the application or library, contents that refer to the current version, and contents that refer only to a particular subversion. Whem querrying a particular topic, contents that are the lower in the hirearvhy must be provided.
- It must be easy to access, view & browse documentation from within application (e.g. ther can be a special class for manipulation of help, one for tooltips on forms, one for inline help, etc.).
- At least viewing of the contents must be entirely possibly with the tools form the library. This is not required for editing.

Help & documentation system can be started when all modules it will rely on will be well established, in particular the XML manipulation system, versioning system, HTML viewing (if the system will rely on HTML), etc.

2.3 Forms library (IGForms)

2.3.1 Minimal Ftandards for Fibrary Forms

Loose standards will be set for forms provided by the library. It is not necessary that these standards would strictly apply to all the forms.

Summary of the standards:

- When sensible, otherwise top-level forms will also be provided in embedded form
- The forms will be thread-safe, such that functionality may be accessed from any thread withoud inducing conflicts in the case of concurrency
- There will be different ways for launching top-level window forms, with methods having standard names or prefices:
    - Synchroneous in the same thread
    - Asynchroneous (non-blocking) in another thread
    - Constructors will not also launch the form, but there may be launching methods that will launch the form without calling a constructor.

2.3.2 Text Style Management Class

This will enable easy management of text styles (including definition of font, background color and foreground color) with at least the following functionality:

- Encapsulating one or many styles, with ability to access them via keys
- Definition and destruction of styles
- Definition of basic styles
- Styles can have only individual components defined (e.g. only foreground color), combination with base style will be enabled such that undefined components will be taken from base styles
- Copying styles from forms and other styles
- Application of styles to all kind of forms (also recursive application)

2.3.3 Toplevel Window Positioning Class

This will enable:

- Positioning relative to a mouse pointer, in absolute (pixels) or relative units.
- Arbitrary justification (left, right, top, bottom) and shifting (absolute – in pixels, or relative – in window width and height).

- Arbitrary combinations of
  - Positioning relative to a mouse pointer, in absolute (pixels) or relative units.
  - Positioning relative to arbitrary part of another form (its center or any of the corners, shifted absolutely or relatively with respect to the window).
  - Positioning relative to the screen.
- Arbitrary justification (left, right, top, bottom) and shifting (absolute – in pixels, or relative – in window width and height).

2.3.4 Standard controls

The question is whether the library would also provide standard controls such as buttons and text boxes with extended functionality. More or less this would be implemented only in special cases, e.g. if there would be a common need for such controls with special behavior (such as special way of validation of input, or changing appearance when contents changed form the initial state). However, it is not a general intention of the library to also provide such controls. This can be made done in a separate library if necessary.

2.3.5 Fading Message

Fading messages are intended to show pieces of information in small borderless top-level windows, which automatically fade out and close after a specified time. They are especially useful to addract user's attention (since they appear out of othe ordinary gui forms) or to provide show information non-intrusively, without disturbing other user interaction (in contrary with e.g. message boxes).

Fubnctionality:

- Definition of a (more distinctive) title and text
- Message fades out (changes color smoothly) after a prescribed time and then closes
- Definition of basic background and fading background, and of title foreground and title background
- Definition of portion of appearing time in which the message fades (changes color, typically to a less distinctive one)
- Closing before fade out by Ctrl-Right.

- Dragging by mouse (useful when the message obscures important contents)

2.3.6 Console form

Enables simple console-like text output/input with styles.

Application can launch an arbitrary number of consoles, but there is one global console that is, among the others, used for logging and error reporting.

2.3.7 XML Viewer & Editor

This will enable easy scanning of contents of XML documents (e.g. by switching which levels ar shown, by distinctive coloring, tree views, etc.) and editing of the contents. The viewer will contain XPath navigator and inspector. In the far future, advanced functionality can be added such as schema validation.

Two forms of editing will be peovided.

General editing wil include

- Cutting, copying and pasting parts of the tree structure
- Creating new nodes, editing attributes, etc.

Special editing tools will include editing of nodes that represent specific data structures (such as numbers, vectors or matrices) by special tools.

2.3.8 Text Viewer & Editor

2.3.9 Html Viewer

A small component that will enable easy viewing and navigation throgh HTML documentation.

2.3.10 Html Editor

This is not a simple task. There are several options how to get this working in a short time:

- Utilize an existing editor implemented in C# and adapting it. Such an editor must have a suitable license.
- Implement a RTF editor by using RichTextBox functionality and find a way to convert the generated RTF to HTML.

2.3.11 External applications management

A module that will handle definition of external text viewers & editors, HTML editors, etc.

2.4 Applications

2.4.1 Sendigence

Sendigence is a program for systematically sending bulk e-mails. It can serve organizers of events (such as conferences) to inform potential attendants, software developers or other business to send newsletters to their clients, project managers to send regular communications to project members, etc.

2.4.1.1 Basic Layout

2.4.1.1.1 A System for Storing Data

2.4.1.2 User Interface

2.4.1.3 Modules

2.4.1.3.1 Definition

2.4.1.3.2 Definition of Addresses

2.4.1.3.3 Definition of e-mail Contents

2.4.1.3.4 Sending module

Implement sending through client

2.4.1.3.5 Address Center

2.4.2 Visual Inverse

Visual Inverse is a visual front-end for accessing *Inverse* functionality. Its interface with Inverse will be similar as interface between *Inverse* and *Mathematica*. At a later stage, Visual Inverse may itself implement optimization functionality.

2.4.3 Dragonfly Optimization Server Suite

Dragondly server will enable to set up optimization services, which will be used by clients that can independently set up their own numerical calculations, and then perform optimization invilving these calculations by use of the optimization server.

2.4.3.1 General Client – Server Architecture

2.4.3.2 Analysis Server

This is a component that performs direct calculation, e.g. individual numerical analyses. It is a framework that enables someone to set up and run complex numerical analyses, involving e.g. finite element simulations.

When up, the analysis server listens for analysis requests, executes them and returns results to the client.

2.4.3.3 Optimization Server

Optimization server recieves requests from the optimization client and performs optimization tasks according to these requests.

When performing some tasks like running test analyses, parametric studies, calculation of numercal derivatives and optimization, it acts as a client to the analysis server.

2.4.3.3.1 Optimization/Analysis Client

Optimization client sends requests to the analysis server and recieves result to it. It must be able to follow the progress in the middle of individual tasks.

References:

[1] I. Grešovnik et al.: Optimisation program *Inverse*, http://www.c3m.si/inverse/doc/man/frame.html

[2] I. Grešovnik: A General Purpose Computational Shell for Solving Inverse and Optimisation Problems - Applications to Metal Forming Processes, Ph. D. thesis, University of Wales Swansea, U.K., 2000. http://www.c3m.si/inverse/doc/phd/frame.html

[3] I. Grešovnik: IOptLib – Investigative Optimization Library. http://www2.arnes.si/~ljc3m2/igor/ioptlib/

3 SANDBOX (THIS IS NOT PART OF THIS REPORT)