Sergiy Nikitin

WEB-SERVICE FOR COMPUTATIONAL ERROR ESTIMATION

Master's thesis

Mobile computing

22/03/2005

University of Jyväskylä

Department of Mathematical Information Technology

Author: Sergiy Nikitin

Contact Information: e-mail: senikiti@cc.jyu.fi

Title: Web-Service for Computational Error Estimation

Work: Master's Thesis

Number of Pages: 70

Study Line: Mobile Computing

Department: University of Jyväskylä, Department of Mathematical Information Technology

Keywords: Semantic Web, Error Estimation, Finite Element Method, PDE Toolbox, MATLAB, Estimator, RDF, Web-Service, OWL, GUN, Integration Environment

Abstract: Modern industrial applications involve more and more computer technology features due to growth of computational capabilities and intellectualization of industry-oriented software. The permanent increase of computing capabilities gave an impact to numerical mathematics and numerical methods are widely used in many industrial applications. The key question in usage of numerical techniques is to explicitly control the computational error. In turn, intellectualization of software caused a new wave of processes automation and now industry is surprisingly fast integrating to WWW. This work provides an idea of wrapping the numerical technique for computational error estimation to web-service in order to be reachable not only by humans, but for automated service discovery and usage. This idea is based on Semantic Web paradigm and Global Understanding Environment concept, which is also aimed at providing mobile and distributed web-platforms for industry.

Acknowledgements

First of all I'm grateful to my parents for their support and upbringing. They are most significant people to me.

Second, I want to say thanks to my supervisors: Ass. Professor Vagan Terziyan and Dr. Sergey Korotov from Department of Mathematical Information Technology for proposal of thesis topic and many useful suggestions during preparation of the work. I would like to thank my colleagues from Industrial Ontologies Group for useful discussions and good advices.

I'm grateful to our exchange programme coordinator Dr. Helen Kaykova for her guidance and advices in all the urgent and daily problems during my stay in Finland.

I also express my thanks to University of Jyväskylä and Kharkiv National University of Radioelectronics for opportunities they provided.

Next, I want to say thanks to the Agora Center particularly to Professor Pekka Neittaanmäki for financial support and perfect work conditions.

I'm especially grateful to the fund "Intellectual Spirituality Society" for their moral support and guidance.

I'm thankful to my girlfriend Svetlana for her care, support and reminding me about true values.

Abbreviations

API – Application Programming Interface

BVP – Boundary-Value Problem

CORBA – Common Request-Broker Architecture

FEM – Finite Element Method

FTP – File Transfer Protocol

GUN – Global Understanding Environment

HTTP – Hyper Text Transfer Protocol

HTTPS – Secure HTTP

jUDDI – Java-based Universal Description, Discovery and Integration

OWL – Web Ontology Language

PDE – Partial Differential Equations

RDF – Resource Description Framework

RSCDF – Resource State-Condition Description Framework

SMTP – Simple Mail Transfer Protocol

SOAP – Simple Object Access Protocol

UDDI - Universal Description, Discovery and Integration standard

WSDL – Web Services Description Language

XML – eXtensible Markup Language

Contents

1 Introduction

1.1 Background

Nowadays World Wide Web is permanently transforming all the time and thus opens new capabilities for possible use. Different application areas are already involved in WWW. And among them we observe such areas, that we couldn't even expect a couple of years ago. Different application domains have already migrated to Mobile Platforms. This web growth requires new ways to manage its heterogeneous content. But for content management is essential how it is represented. Semantic Web paradigm [SEMWEB] brought a new value to content description and aims at not just readable but machine understandable mark-up.

In this work we will provide architecture for web-service with essentially new kind of content and application areas. The semantic vision of new web opens new domains for web coverage. The motivation to create Web-Service for Computational Error Estimation was brought by Semantic Web paradigm and its benefits for Web-integration. However Semantic Web paradigm must be implemented in concrete technologies and approaches.

1.2 Related work

Related work concerning Web-services integration and Resource semantic annotation can be found in [Ermolayev et al., 2004], [SmartResource, 2004], [GUN].

1.3 Research problem statement

From the very beginning the problem was stated as "experimental acknowledgement of theoretical foundations for Computational Error Estimation" [Korotov et al., 2004]. Experimental results proved the effectiveness of technique. So the idea came to create web service so that others could check it and use in educational purposes. We decided to apply Semantic Web wrapping for service architecture as most promising today for web-standards of future.

1.4 Structure of the thesis

Chapter 2 contains review of existing technologies for web-service integration and protocol stacks.

In Chapter 3 the mathematical foundations of Computational Error Estimation Technique are depicted.

Chapter 4 describes software implementation of Error Estimation Technique in MATLAB

Chapter 5 provides a proposal for possible wrapping of implemented software to semantically annotated resource.

2 Web-service architectures

With Web Services, information sources become components that we can use, re-use, mix, and match to enhance Internet and intranet applications ranging from a simple currency converter, stock quotes, or dictionary to an integrated, portal-based travel planner, procurement workflow system, or consolidated purchase processes across multiple sites [WSArchitect]. Web Services, at a basic level, can be considered a universal client/server architecture that allows disparate systems to communicate with each other without using proprietary client libraries. This chapter reviews existent web-service architectures and technologies.

2.1 WebServices.Org architecture

The following is the Web Services stack from WebServices.Org:

Layer	Example
Trading Partner Agreement	Trading Partner Agreement
Workflow, Discovery, Registries	UDDI, ebXML registries, IBM WSFL, MS XLANG
Service Description Language	WSDL/WSCL
Messaging	SOAP/XML Protocol
Transport Protocols	HTTP, HTTPS, FTP, SMTP
Business Issues	Management, Quality of Service, Security, Open Standards

Table 2.1 – Web-services stack from WebServices.Org

2.1.1 Service negotiation

The business logic process starts at the Services Negotiation layer (the top) with, say, two trading partners negotiating and agreeing on the protocols used to aggregate Web Services.

This layer is also referred to as the Process Definition layer, covering document, workflow, transactions, and process flow.

2.1.2 Workflow, discovery, registries

The next layer to establish workflow processes uses Web Services Flow Language [WSFL] and MS XLANG, which is an XML language to describe workflow processes and spawn them. WSFL considers two types of Web Services compositions: The first type (flow models) specifies the appropriate usage pattern of a collection of Web Services, in such a way that the resulting composition describes how to achieve a particular business goal; typically, the result is a description of a business process. The second type (global models) specifies the interaction pattern of a collection of Web Services; in this case, the result is a description of the overall partner interactions.

While WSFL complements WSDL (Web Services Definition Language) [WSDL] and is transition-based, XLANG is an extension of WSDL and block-structured based. XLANG, on the other hand, allows orchestration of Web Services into business processes and composite Web Services. WSFL is strong on model presentation while XLANG does well with the long-running interaction of Web Services.

Web Services that can be exposed may, for example, get information on credit validation activities from a public directory or registry, such as Universal Description, Discovery and Integration [UDDI]. The ebXML, E-Services Village, BizTalk.org, and xml.org registries and Bowstreet's (a stock service brokerage) Java-based UDDI (jUDDI) are other directories that could be used with UDDI in conjunction with Web Services for business-to-business (B2B) transactions in a complex EAI infrastructure under certain conditions. Web Services is still primarily an interfacing architecture, and needs an integration platform to which it is connected. Such an integration platform would cover the issue of integrating an installed base of applications that cannot work as Web Services yet.

2.1.3 Service description language

As we move further down the stack, we need WSDL to connect to a Web Service. This language is an XML format for describing network services. With it, service requesters can search for and find the information on services via UDDI, which, in turn, returns the WSDL reference that can be used to bind to the service.

Web Service Conversational Language (WSCL) helps developers use the XML Schema to better describe the structure of data in a common format (say, with new data types) the customers, Web browsers, or indeed any XML enabled software programs can recognize. This protocol can be used to specify a Web Service interface and to describe service interactions.

2.1.4 Messaging

In the Messaging layer SOAP acts as the envelope for XML-based messages, covering message packaging, routing, guaranteed delivery and security. Messages are sent back and forth regarding the status of various Web Services as the work progresses (say, from customer order to shipping product out of the warehouse).

2.1.5 Transport protocols

When a series of messages completes its rounds, the stack goes to its last layer: the transport layer using Hypertext Transfer Protocol (HTTP), Secure HTTP (HTTPS), Reliable HTTP (HTTPR) File Transfer Protocol (FTP) or Standard Mail Transfer Protocol (SMTP). Then, each Web Service takes a ride over the Internet to provide a service requester with services or give a status report to a service provider or broker.

2.1.6 Business issues

Finally, the Business Issues row in the table lists other key areas of importance to the use and growth of Web Services. Without consideration to these points, Web Services could quickly become objects of ridicule.

2.2 The Stencil group

Now, let's take a look at the Stencil Group's Web Services technology stack. It is similar to that of WebServices.Org with three exceptions:

1. The WebServices.Org stack does not divide the layers into emerging and core components. Not doing so could confuse the reader as to which standards are emerging. What is now an emerging standard would become a core standard at a future date.
2. The Stencil Group does not apply Management, Quality of Service, Open Standards, and Security to any layer. The reader could get the wrong impression that they are proprietary and treated as not important. When this happens, the reader will opt for another architecture stack that has these features.
3. The Stencil Group starts the stack with undefined business rules while WebServices.Org begins with a clearly defined business process such as service agreement. The reader could get confused on what undefined business rules are, and how many would eventually be defined.

Layer	Type
Other Business Rules (undefined)	Emerging Layers
Web Services Flow Language (WSFL)	
Universal Description, Discovery and Integration (UDDI)	
Web Services Description Language (WSDL)	
Simple Object Access Protocol (SOAP)	Core Layers
Extensible Markup Language (XML)	
Common Internet Protocols (TCP/IP, HTTP)	

Table 2.2 - Web-services stack from Stencil Group

2.3 IBM stack

The IBM Conceptual Web Services stack is part of their Web Services Conceptual Architecture (WSCA) 1.0 [IBM WSCA]. It is presented in a slightly different way than that of the first two stacks, by starting with Web Services tools and then showing what each layer is used for.

Tools	Layer		
TPA (Trading Partner Agreement)	Service Negotiation		
WSFL	Service Flow		
UDDI+WSEL	Service Description	Service Publication (Direct UDDI) Service Directory (Static UDDI)	Endpoint Description
WSDL	Service Interface Service Implementation		
SOAP	XML-Based Messaging		
HTTP, FTP, email, MQ, IIOP	Network		
Quality of Service, Management, Security	Business Issues		

Table 2.3 - Web-services stack from IBM

The IBM Web Services stack does not show WSCL and ebXML, included in the WebServices.Org stack. It associates the Network layer with IBM MQSeries (now called WebSphere MQ) messaging systems and the Internet Inter-ORB Protocol (IIOP) - a protocol CORBA uses to transmit data, information, and messages between applications. They do not appear in either that of WebServices.Org or The Stencil Group. IBM considers WSDL as a description of the service endpoints where individual business operations can be accessed. WSFL uses WSDL for the description of service interfaces and their protocol bindings. WSFL also relies on WSEL (Web Services Endpoint Language), an endpoint description language to describe non-operational characteristics of service endpoints, such as quality-of-service properties.

Together, WSDL, WSEL, and WSFL provide the core of the Web Services computing stack. IBM perceives UDDI in two categories: static and direct. Static UDDI refers to the Service Directory established after applying WSFL to Service Flow, while direct UDDI pertains to the Service Publication of directory items. Similar to the WebServices.Org stack, the IBM stack applies QoS, management, and security to all layers.

2.4 W3C stack

The W3C Web Services Workshop, led by IBM and Microsoft, has agreed that the architecture stack consists of three components: Wire, Description, and Discovery.

2.4.1 Wire stack

The following table shows what layers constitute the Wire Stack.

Other "extensions"	
Attachments	Routing
Security	Reliability
SOAP/XML	
XML	

Table 2.4 – W3C Wire Stack

Wire Stack has extensions to two layers: SOAP and XML. This means whenever the SOAP is used as the envelope for the XML messages, they must be attached, secure, reliable, and routed to the intended service requester or provider. In the stacks of other organizations, SOAP and XML are not treated as "extensions." IBM, for instance, refers to SOAP as a tool for its stack layer, "XML-Based Messaging."

2.4.2 Description stack

The Description Stack, the most important component, consists of five layers:

Business Process Orchestration		
Message Sequencing		
Service Capabilities Configuration		
Service Description (WSDL)	Service Interface	WSDL
	Service Description	
XML Schema		

Table 2.5 – W3C Description Stack

This stack starts with orchestration of business processes from which the messages are sequenced, depending on how service capabilities are configured.

W3C uses WSDL to describe service interface and service implementation, neither of which is explicitly highlighted in other stacks.

2.4.3 Discovery stack

As the name implies, the Discovery Stack involves the use of UDDI, allowing businesses and trading partners to find, discover, and inspect one another in a directory over the Internet, as follows:

Directory (UDDI)
Inspection

Table 2.6 - W3C Discovery Stack

The Inspection Layer refers to WSIL (Web Services Inspection Language) and WS-Inspection specifications.

Putting all three stack-components together, we have the Architecture Stack.

Other "extensions"			
Attachments	Routing		
Security	Reliability		
SOAP/XML			
XML			
Business Process Orchestration			
Message Sequencing			
Service Capabilities Configuration			
Service Description (WSDL)		Service Interface	WSDL
		Service Description	
XML Schema			
Directory (UDDI)			
Inspection			

Table 2.7 – W3C architecture stack

Today, SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI are emerging as the Internet de facto standards for Web services. SOAP has been accepted and is being standardized by the World Wide Web Consortium (W3C). WSDL has been submitted to the W3C for standardization, and is emerging as the de facto standard language for the description of Web services. UDDI is poised to be the de facto standard for the Web service repository.

The following subchapters describe briefly what these technologies are and what functionality they carry with.

2.5 SOAP

SOAP Version 1.2 [SOAP] is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics.

Two major design goals for SOAP are simplicity and extensibility. SOAP attempts to meet these goals by omitting, from the messaging framework, features that are often found in distributed systems.

The SOAP protocol provides a convention for Remote Procedure Call (RPC) using XML messages. SOAP specifies a wire protocol for facilitating highly distributed applications. SOAP is similar to DCOM and CORBA in that it provides an RPC mechanism for invoking methods remotely. SOAP differs in that it is a protocol based on open XML standards and XML document exchange rather than being an object model relying on proprietary binary formats. Both DCOM and CORBA use binary formats for their payload (NDR and CDR, respectively). The SOAP gateway performs a similar function to DCOM and CORBA stubs – translating messages between the SOAP protocol and the language of choice. As a result, SOAP offers vendor, platform, and language independence. With

SOAP, developers can easily bridge applications written with COM, CORBA, or Enterprise JavaBeansTM.

From the specification, SOAP is composed of three parts:

- A framework describing how SOAP messages should be constructed
- A set of encoding rules for exchanging data types
- A convention for representing remote procedure calls

The encoding rules defined for various data types can be serialized across SOAP requests. The SOAP 1.1 specification bases its data encoding on XML Schema Structures and XML Schema Data Types, but also allows arbitrary data encoding like RDF. Supported types include simple types, like strings and enumerations, and complex types of structures and arrays. The specification also describes a convention for performing RPC interactions using XML. SOAP messages can be sent over any transport protocol including HTTP(s), SMTP, and FTP.

A SOAP message contains three primary pieces: an envelope; a header for adding application-specific features to a SOAP message, including authentication, transaction management, and payment; and a body that contains information intended for the recipient. An application receiving a SOAP message must identify all parts intended for it, verify that those parts are complete, and process them. Because a SOAP message can travel through a number of intermediaries, the SOAP actor attribute is used to indicate the ultimate recipient of the message. The specification also defines a mustUnderstand attribute that indicates whether a specific header entry has to be understood and processed by the recipient.

The following illustrates how a simple request/response message could be written with SOAP:

Request	Response

`POST /StockQuote HTTP/1.1`
`Host: www.stockquoteserver.com`
`Content-Type: text/xml`
`Content-Length: nnnn`
`SOAPAction: “Some-URI”`
`<SOAP:Envelope`
`xmlns:SOAP="urn:schemas.xmlsoap.org`
`:soap.v1">`
`<SOAP:Header>`
`<t:Transaction xmlns:t=”URI”`
`mustUnderstand=”1”>5`
`</t:Transaction>`
`</SOAP:Header>`
`<SOAP:Body>`
`<m:GetLastTradePrice`
`xmlns:m="URI">`
`<symbol>DIS</symbol>`
`</m:GetLastTradePrice>`
`</SOAP:Body>`
`</SOAP:Envelope>`
	`HTTP/1.1 200 OK`
	`Content-Type: text/xml`
	`Content-Length: nnnn`
	`<SOAP:Envelope`
	`xmlns:SOAP="urn:schemas.xmlsoap.org`
	`:soap.v1">`
	`<SOAP:Header>`
	`<t:Transaction xmlns:t=”URI”`
	`xsi-type=”xsd:int”`
	`mustUnderstand=”=”>5`
	`</t:Transaction>`
	`</SOAP:Header>`
	`<SOAP:Body>`
	`<m:GetLastTradePriceResponse`
	`xmlns:m="URI">`
	`<return>34.5</return>`
	`</m:GetLastTradePriceResponse>`
	`</SOAP:Body>`
	`</SOAP:Envelope>`

Table 2.8 – Examples of Request and Response of SOAP protocol

Because SOAP messages can be carried over the HTTP protocol, they can easily pass through firewalls. Unlike other distributed object models that rely on dynamically assigned ports, SOAP can use HTTP's standard port for transmitting data. The SOAP specification does not directly address security but relies on either the underlying protocol or the conventions of securing data within the business payload. Using the HTTPS protocol, SOAP message exchanges can be kept private.

SOAP was never intended to provide a complete distributed object architecture. Although it is possible to handle request and response messages, the SOAP specification does not directly handle asynchronous communication. It is, however, possible to implement asynchronous communication using the SOAP protocol.

SOAP will need to be extended with standardized mustUnderstand header fields to encode attributes like "message sender," "message recipient," and "message co-relation id" in order to claim native support for asynchronous messaging. BizTalk and ebXML are examples of initiatives that have extended SOAP to handle asynchronous messaging. Although SOAP is protocol-neutral, it currently only describes transport bindings over the HTTP protocol.

SOAP is clearly becoming one of the de facto standards for Web services. Because of its platform and language independence, and its ease of integration, many companies are embracing SOAP as a backbone for their Web services strategy.

2.6 WSDL

WSDL [WSDL] provides a grammar for describing services as a set of endpoints that exchange messages. A WSDL document serves as a language and platform agnostic (XML) description of one or more services. A WSDL document describes those services, how to access them, and what type of response (if any) to expect. WSDL describes types, messages, operations, portTypes, locations, and protocol bindings of the service.

Below are some of the concepts that WSDL tries to describe:

PortTypes:	A `portType` describes the operations provided by a Web service. It is like a Java interface in that it describes a set of operations. It combines message elements into an operation.
Messages and types	A `message` is a data element. It is used by an operation to carry the data of the operation. Messages describe the communication between client and service, by listing the data types exchanged. The types are described in the `types` element, which is usually done with XML Schema. Types are like Java classes and primitive types.
Operations, messages, and faults	An `operation` is like a Java method. It consists of incoming, outgoing, and fault messages. We can think of an incoming message for an operation like a method's parameters in Java programming language. We can think of an outgoing message for an operation like a method's return type in Java programming language. We can think of a fault message as a Java exception.
Bindings	A binding binds a portType to a particular protocol (for example, SOAP 1.1, HTTP GET/POST, or MIME).
Services	A service defines the connection information for a particular binding. Services can have one or more ports, each of which define a different connection method (for example, HTTP / SMTP, etc.).

Table 2.9 – Concepts of WSDL

For relations between WSDL concepts see Figure 2.1

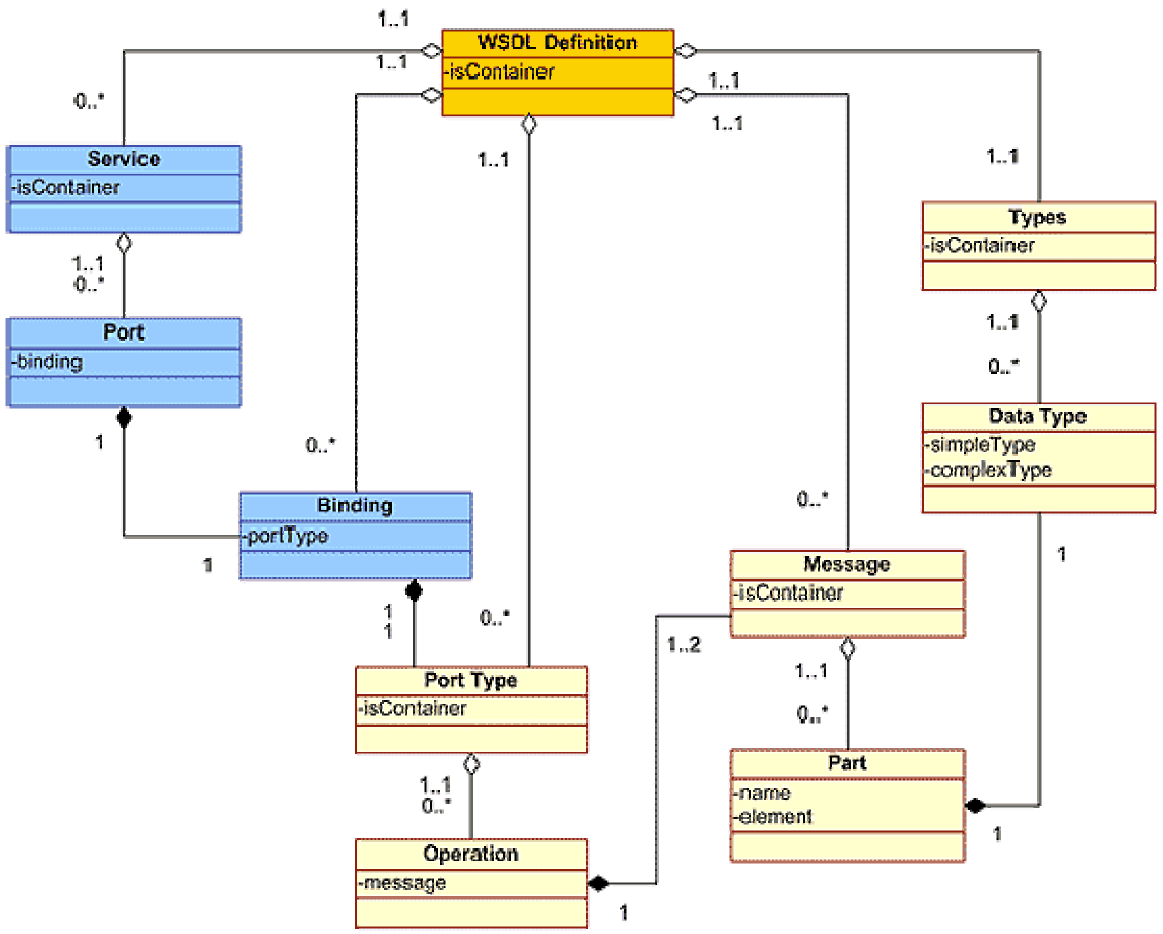


Figure 2.1 – WSDL concepts diagram

2.7 UDDI

The Universal Description Discovery and Integration (UDDI) specification provides a platform independent way of describing services, discovering businesses, and integrating business services using the Internet. The UDDI data structures provide a framework for the description of basic business and service information, and architects an extensible mechanism to provide detailed service access information using any standard description language. Many such languages exist in specific industry domains and at different levels of the protocol stack. The Web Services Description Language (WSDL) is a general purpose XML language for describing the interface, protocol bindings and the deployment details of network services. WSDL complements the UDDI standard by providing a uniform way of describing the abstract interface and protocol bindings of arbitrary network services.

2.7.1 tModels

`tModels` provide the ability to describe compliance with a specification, a concept, or a shared design. `tModels` have various uses in the UDDI registry. We are interested here in the use of `tModels` to represent technical specifications like wire protocols, interchange formats and sequencing rules. When a particular specification is registered with the UDDI repository as a `tModel`, it is assigned a unique key, which is then used in the description of service instances to indicate compliance with the specification.

2.7.2 Business service

Services are represented in UDDI by the `businessService` data structure, and the details of how and where the service is accessed are provided by one or more nested `bindingTemplate` structures.

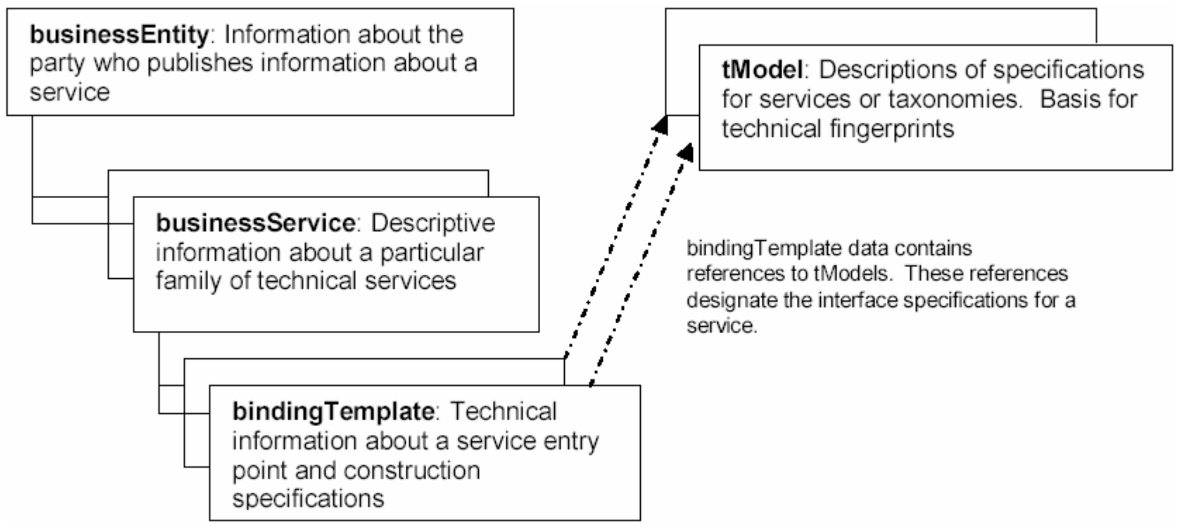

businessEntity: Information about the party who publishes information about a service
businessService: Descriptive information about a particular family of technical services
bindingTemplate: Technical information about a service entry point and construction specifications
tModel: Descriptions of specifications for services or taxonomies. Basis for technical fingerprints
bindingTemplate data contains references to tModels. These references designate the interface specifications for a service.

Figure 2.2 – UDDI concepts relations

A `bindingTemplate` specifies a network endpoint address (in the `accessPoint` element) and a stack of `tModels` describing the service.

Below is the example code of UDDI accessPoint

```
<businessService>
(...)
  <bindingTemplates>
    <bindingTemplate>
      (...)
      <accessPoint urlType="http">http://www.etc.com/</accessPoint>
      <tModelnstanceDetails>
        <tModelnstanceInfo tModelKey="...">
        </tModelnstanceInfo>
        (...)
      </tModelnstanceDetails>
    </bindingTemplate>
    (...)
  </bindingTemplates>
</businessService>
```

2.7.3 Authoring UDDI service descriptions

We now consider how WSDL can support the creation of UDDI businessService entries. We summarize the process in three major steps.

1. The first step is to create the WSDL service interface definition. Typically, industry groups will define a set of service types, and describe them with one or more service interface definition WSDL documents. The service interface definition will include service interfaces and protocol bindings, and will be made publicly available. The WSDL service interface definitions are then registered as UDDI `tModels`; the `overviewDoc` field in each new `tModel` will point to the corresponding WSDL document

2. Next, programmers will build services that conform to the industry standard service definitions. Either manually or using appropriate UDDI-aware tooling, programmers will retrieve the `tModel` description of the industry standard definition, and (following the `overviewDoc` link) obtain the corresponding WSDL definition document. WSDL-aware tooling, in turn, can help generate an implementation that supports the standard interfaces and bindings.

3. Finally, the new service must be deployed and registered in the UDDI repository. Either manually or using WSDL and UDDI-aware tooling, a UDDI businessService data structure is created, and then registered. Typically when using WSDL and UDDI-aware tools, service deployment information (some type of "deployment descriptor" document) will be generated at that same time.

The information contained in the new businessService references the implemented standards and provides additional deployment details:

- A `bindingTemplate` is created for each service access endpoint. The network address of the access point is encoded in the `accessPoint` element.

- One `tModelInstanceInfo` is created in the `bindingTemplate` for each `tModel` that is relevant to the service end point being described, in particular, for every `wsdlSpec tModel` that defines interfaces and bindings supported by the service.

2.7.4 Registering and referencing WSDL definitions in UDDI

The development and registration of services described in the previous section assumes that some WSDL service information is registered or embedded in the UDDI registry. We focus on two aspects of this problem:

1. Registration of WSDL service interface definitions as UDDI tModels.

2. Reference to reusable `wsdlSpec tModels` in `bindingTemplates`.

WSDL service interface definitions are intended to describe many service instances, and it is consequently natural to register them as `tModels`. In the case when the description comprises more than one WSDL document, one `tModel` should be created for each. Each such `tModel` must be classified, using the `uddi-org:types` taxonomy, as being of type “`wsdlSpec`” and must have an `overviewDoc` whose `overviewURL` points to the relevant WSDL document. An example is outlined below.

```
<tModel authorizedName="..." operator="..." tModelKey="...">

<name>StockQuote Service</name>

<description xml:lang="en">

WSDL description of a standard stock quote service interface

</description>

<overviewDoc>

<description xml:lang="en">WSDL source document.

</description>

<overviewURL>

http://stockquote-definitions/stq.wsdl

</overviewURL>

</overviewDoc>

<categoryBag>

<keyedReference tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"

keyName="uddi-org:types"

keyValue="wsdlSpec"/>

</categoryBag>

</tModel>
```

`tModelInstanceInfo` structures in the `bindingTemplate` use the `tModelKey` attribute to refer to the technical specifications that are required to interact with that service endpoint. When WSDL and UDDI are used together, the `tModel` referred to should be one of type `wsdlSpec`, that is, one whose `overviewDoc` is a WSDL service interface definition, as explained in the previous section. One `tModelInstanceInfo` structure must be created for each `wsdlSpec tModel` containing standard definitions that are relevant to the service being defined.

3 Error estimation technique

The technology presented below is designed to evaluate the accuracy of computed solutions, measured in terms of "problem-oriented" criteria that can be chosen by users. The technology is applicable to various problems in mechanics and physics embracing mathematical models of diffusion processes and models arising in the elasticity theory of solid bodies.

3.1 Diffusion problem

3.1.1 Mathematical model

We consider physical (or mechanical) problem that can be formally presented in the form of a *boundary value problem of elliptic type (BVP)* as follows: Find a function $u = u(x_1,...,x_d)$ of d variables $x_1,...,x_d$, where $d = 1,2,...,$ such that

$$-\sum_{i,j=1}^{d} \frac{\partial}{\partial x_i}\left(a_{ij} \frac{\partial u}{\partial x_j} \right) = f \quad \text{in} \quad \Omega, \tag{3.1}$$

$$u = u_0 \quad \text{on} \quad \Gamma_1, \tag{3.2a}$$

$$\sum_{i,j=1}^{d} a_{ij} \frac{\partial u}{\partial x_j} n_i = g \quad \text{on} \quad \Gamma_2. \tag{3.2b}$$

In the above, relation (3.1) is a *process governing equation* in the solution domain $\Omega \subset R^d$, where the natural number d denotes a *dimension of the problem*, $a_{ij} = a_{ij}(x_1,...,x_d)$, $i, j = 1,...,d,$ are the given *coefficients of the problem* that usually describe the diffusion properties of the respective media, and $f = f(x_1,...,x_d)$ can be viewed as a given *source function*. The coefficients are assumed to be bounded functions and f is assumed to be square summable. We assume that Ω is a bounded connected domain with Lipschitz continuous boundary Γ, which consists of a finite number of smooth parts (see Fig. 3.1).

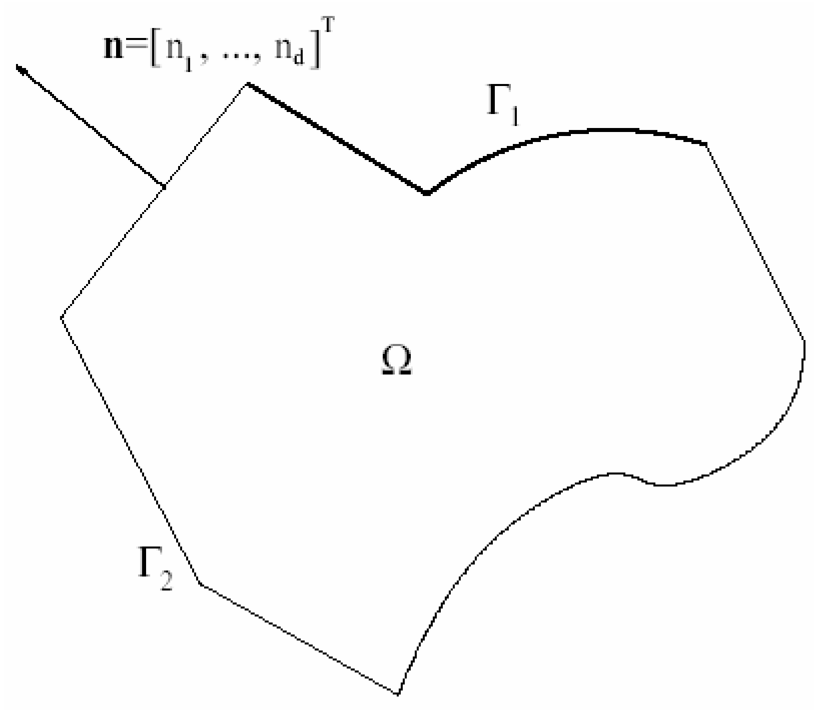


Figure 3.1 - A solution domain with the basic notation.

Further, relations (3.2a) and (3.2b) are *boundary conditions*, which prescribe a behaviour of the solution and its derivatives on two nonintersecting boundary parts Γ_1 and Γ_2, respectively, i.e., $\Gamma = \Gamma_1 \cup \Gamma_2$. The functions $u_0 = u_0(x_1,...,x_d)$ and $g = g(x_1,...,x_d)$ are given. We assume that g is a square summable function, in particular, it can be an arbitrary piecewise smooth function. The function u_0 must belong to the energy functional class (which is $H^1(\Omega)$), and, in particular, it can also be any piecewise smooth function.

The symbol n_i denotes the i-th component of the outward unit normal vector $\mathbf{n} = \mathbf{n}(x_1,...,x_d)$ to the boundary Γ, i.e., $\mathbf{n} = [n_1,...,n_d]^T$ (see Fig. 3.1), the outward unit normal is defined at almost all points of the boundary.

The above assumptions are not restrictive and cover the bulk of practically meaningful cases in modeling of diffusion processes.

3.1.2 Finite element solution

Let V_h be a finite-dimensional space constructed by means of a selected set of finite element trial functions defined on commonly-used finite element mesh T_h over Ω. We notice that space V_h is chosen so that its functions w_h vanish on Γ_1.

The finite element approximation for problem (3.1)–(3.2) is defined then as a function $u_h = u_h(x_1,...,x_d) \in V_h + u_0$, such that

$$\int_\Omega \sum_{i,j=1}^{d} a_{ij} \frac{\partial u_h}{\partial x_j} \frac{\partial w_h}{\partial x_i} dx = \int_\Omega f w_h \, dx + \int_{\Gamma_2} g w_h \, ds \quad \forall w_h \in V_h. \tag{3.3}$$

3.1.3 Problem-oriented criterion

Engineers are often interested not only in the overall error $e = u - u_h$, but also in its local behaviour, e.g., in a certain subdomain $\omega \subset \Omega$. One way to get an information about the local behaviour of $u - u_h$ is to measure the error in terms of specially selected *problem-oriented criteria*.

One of the most typical criteria of such a type is presented by the integral

$$\int_\Omega \varphi(u - u_h) dx, \tag{3.4}$$

where $\varphi = \varphi(x_1,...,x_d)$ is a selected function such that $supp\, \varphi \subseteq \omega$.

3.1.4 Technology for error estimation in terms of problem-oriented criteria for diffusion problem

To present the technology for estimation of the criterion given by (3.4), we need first to describe two technical problems that must be previously solved. They consist of finding an approximate solution of an auxiliary problem (so-called *adjoint problem*) and making a certain post-processing of this solution, and also making a post-processing of the finite element approximation u_h.

3.1.5 Auxiliary problem and its finite element solution

Let V_τ be another finite-dimensional space constructed by means of a selected set of finite element trial functions on another standard finite element mesh T_τ over Ω. We notice that space V_τ is chosen so that its functions w_τ vanish on Γ_1, and also that T_τ need not to coincide with T_h.

Consider the auxiliary finite-dimensional problem as follows: Find a function $v_\tau = v_\tau(x_1,...,x_d) \in V_\tau$, such that

$$\int_\Omega \sum_{i,j=1}^{d} a_{ji} \frac{\partial v_\tau}{\partial x_j} \frac{\partial w_\tau}{\partial x_i} dx = \int_\Omega \varphi w_\tau dx \quad \forall w_\tau \in V_\tau. \tag{3.5}$$

3.1.6 Gradient averaging procedures

On T_h, we define the *gradient averaging transformation* $\mathbf{G_h}$ mapping the *gradient of the finite element approximation* u_h

$$\nabla \mathbf{u_h} = [\frac{\partial u_h}{\partial x_1},...,\frac{\partial u_h}{\partial x_d}]^T, \tag{3.6}$$

which is constant over each element of the finite element mesh, into a vector-valued continuous piecewise affine function

$$\mathbf{G_h}(\nabla \mathbf{u_h}) = [G_h^1(\nabla u_h),...,G_h^d(\nabla u_h)]^T, \tag{3.7}$$

by setting each its nodal value as the mean (or weighted mean) value of $\nabla \mathbf{u_h}$ on all elements of the patch $\mathbf{P}(x_*)$ associated with corresponding node x_* in the mesh T_h.

More precisely, let $\frac{\partial u_h}{\partial x_i}|_{T_k}$ denote the value of i-th coordinate of gradient $\nabla \mathbf{u_h}$ over triangle T_k, let x_* be one of nodes of finite element mesh T_h, and let T_1, ..., $T_{N_{x_*}}$ be elements of the mesh having node x_* as one of their vertices. The union of such elements is called the *patch* and denoted as $\mathbf{P}(x_*)$. (Thus, in terms of Fig. 3.2 the patch consists of six triangles, i.e. $N_{x_*} = 6$.) Then, we define

$$G_h^i(\nabla u_h)(x_*) = \frac{1}{N_{x_*}} \sum_{T_k \in \mathbf{P}(x_*)} \frac{\partial u_h}{\partial x_i}|_{T_k}, \quad i = 1, ..., d, \tag{3.8a}$$

or

$$G_h^i(\nabla u_h)(x_*) = \frac{1}{measT_1 + ... + measT_{N_{x_*}}} \sum_{T_k \in \mathbf{P}(x_*)} measT_k \cdot \frac{\partial u_h}{\partial x_i}|_{T_k}, i = 1, ..., d, \tag{3.8b}$$

where the symbol *meas* denotes the area of corresponding triangle.

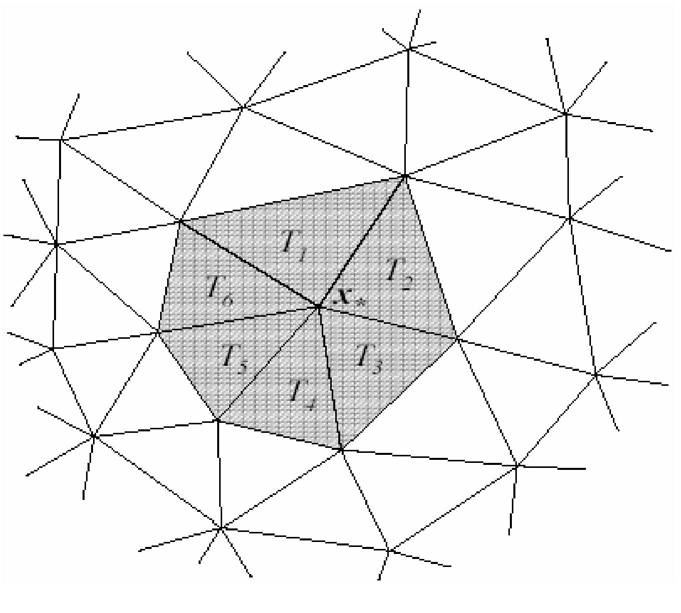


Figure 3.2 - A part of standard finite element mesh and a patch $\mathbf{P}(x_*)$ associated with node (x_*)

Having $G_h^i(\nabla u_h)$ defined at all nodes of the mesh T_h, we uniquely define continuous piecewise affine function $G_h^i(\nabla u_h)$ over the whole domain Ω. In this way, the vector-valued continuous piecewise affine function $\mathbf{G_h}(\nabla \mathbf{u_h})$ in (3.7) is built.

Similarly, on T_τ, we define the *gradient averaging transformation* $\mathbf{G}_\tau$ mapping the *gradient of the finite element approximation* v_τ

$$\nabla \mathbf{v}_\tau = [\frac{\partial v_\tau}{\partial x_1}, \ldots, \frac{\partial v_\tau}{\partial x_d}]^T, \tag{3.9}$$

which is constant over each element of the finite element mesh, into a vector-valued continuous piecewise affine function

$$\mathbf{G}_\tau(\nabla \mathbf{v}_\tau) = [G_\tau^1(\nabla v_\tau), \ldots, G_\tau^d(\nabla v_\tau)]^T, \tag{3.10}$$

by setting each its nodal value as the mean (or weighted mean) value of $\nabla \mathbf{v}_\tau$ on all elements of the patch $\mathbf{P}(x_\circ)$ associated with corresponding node $x_\circ$ in the mesh T_τ.

More precisely, let $\frac{\partial v_\tau}{\partial x_i}|_{T_k}$ denote the value of i-th coordinate of gradient $\nabla \mathbf{v}_\tau$ over triangle T_k, let $x_\circ$ be one of nodes of finite element mesh T_τ, and let T_1, ..., $T_{N_{x_\circ}}$ be elements of the mesh having node $x_\circ$ as one of their vertices. The respective patch is denoted as $\mathbf{P}(x_\circ)$. Then

$$G_\tau^i(\nabla v_\tau)(x_\circ) = \frac{1}{N_{x_\circ}} \sum_{T_k \in \mathbf{P}(x_\circ)} \frac{\partial v_\tau}{\partial x_i}|_{T_k}, \quad i = 1, \ldots, d. \tag{3.11a}$$

or

$$G_\tau^i(\nabla v_\tau)(x_\circ) = \frac{1}{measT_1 + \ldots + measT_{N_{x_\circ}}} \sum_{T_k \in \mathbf{P}(x_\circ)} measT_k \cdot \frac{\partial v_\tau}{\partial x_i}|_{T_k}, i = 1, \ldots, d. \tag{3.11b}$$

Having $G_\tau^i(\nabla v_\tau)(x_\circ)$ at all nodes $x_\circ$, we uniquely define continuous piecewise affine function $G_\tau^i(\nabla v_\tau)$ over the whole domain Ω. In this way, the vector-valued continuous piecewise affine function $\mathbf{G}_\tau(\nabla \mathbf{v}_\tau)$ in (3.10) is built.

Such averaging transformations are widely used in the finite element calculations (see, e.g., [Babuška, Strouboulis., 2001], [Brandts, Křížek, 2003], [Hlaváček, Křížek, 1987], [Křížek, Neittaanmäki, 1984], [Verfürth, 1996], [Zienkeiewicz, Zhu, 1987]). Usually they lead to computationally inexpensive algorithms.

3.1.7 The estimator

Our error estimation method is based upon a new estimator that has been derived and justified numerically in [Korotov et al., 2003a], [Korotov, et al., 2003b], [Korotov, Turchyn, 2004]. It estimates the quantity (3.4) by the quantity $E(u_h, v_\tau)$ given by the following formula:

$$E(u_h, v_\tau) := E_0(u_h, v_\tau) + E_1(u_h, v_\tau), \tag{3.12}$$

where

$$E_0(u_h, v_\tau) = \int_\Omega f v_\tau \, dx + \int_{\Gamma_2} g v_\tau \, ds - \int_\Omega \sum_{i,j=1}^{d} a_{ij} \frac{\partial u_h}{\partial x_j} \frac{\partial v_\tau}{\partial x_i} dx, \tag{3.13}$$

and

$$E_1(u_h, v_\tau) = \int_\Omega \sum_{i,j=1}^{d} a_{ij} \left(\frac{\partial u_h}{\partial x_j} - G_h^j(\nabla u_h) \right) \left(\frac{\partial v_\tau}{\partial x_i} - G_\tau^i(\nabla v_\tau) \right) dx. \tag{3.14}$$

The functional $E(u_h, v_\tau)$ is directly computable once the approximations u_h and v_τ are found.

3.1.8 Error estimation algorithm

Step 1: Pose a problem of type (3.1)–(3.2)

Step 2: Find u_h in (3.3)

Step 3: Select criterion (3.4), i.e. subdomain ω and function φ

Step 4: Solve auxiliary problem (3.5) and find v_τ

Step 5: Construct $\mathbf{G_h}(\nabla \mathbf{u_h})$ by (3.8a) or (3.8b)

Step 6: Construct $\mathbf{G}_\tau(\nabla \mathbf{v}_\tau)$ by (3.11a) or (3.11b)

Step 7: Compute $E_0(u_h, v_\tau)$

Step 8: Compute $E_1(u_h, v_\tau)$

Step 9: Compute $E(u_h, v_\tau)$

4 Software implementation

In this chapter we will discuss software implementation issues. First of all the arguments for software platform selection will be given.

4.1 Software platform selection

As far as we implement complicated mathematical method it is reasonable to use already existent packages and tools with appropriate set of functions. Second, we need to wrap all the functionality into web-accessible interface. For future extensions it is also important to have an ability to use some popular programming language, which is involved in web-services development. Such a language could guarantee that service will be easy to upgrade and maintain.

The most suitable for all these requirements in our opinion is MATLAB language [MATLAB]. MATLAB also provides good programming environment. It is extensible because of its toolboxes conception. MATLAB also provides a platform for web-service implementation. Furthermore, it is full compliant with Java programming language, thus implies consistency with powerful programming environments.

But the most significant is mathematical capabilities of MATLAB. It is easy to learn, fast enough for programming and testing the algorithms.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

4.2 PDE Toolbox

The Partial Differential Equation (PDE) Toolbox [PDE Tool] provides a powerful and flexible environment for the study and solution of partial differential equations in two space dimensions and time. The equations are discretized by the Finite Element Method (FEM).

PDE Toolbox provides Graphical Users Interface and Application Programming Interface.

The PDE Toolbox's GUI is depicted on Figure 4.1.

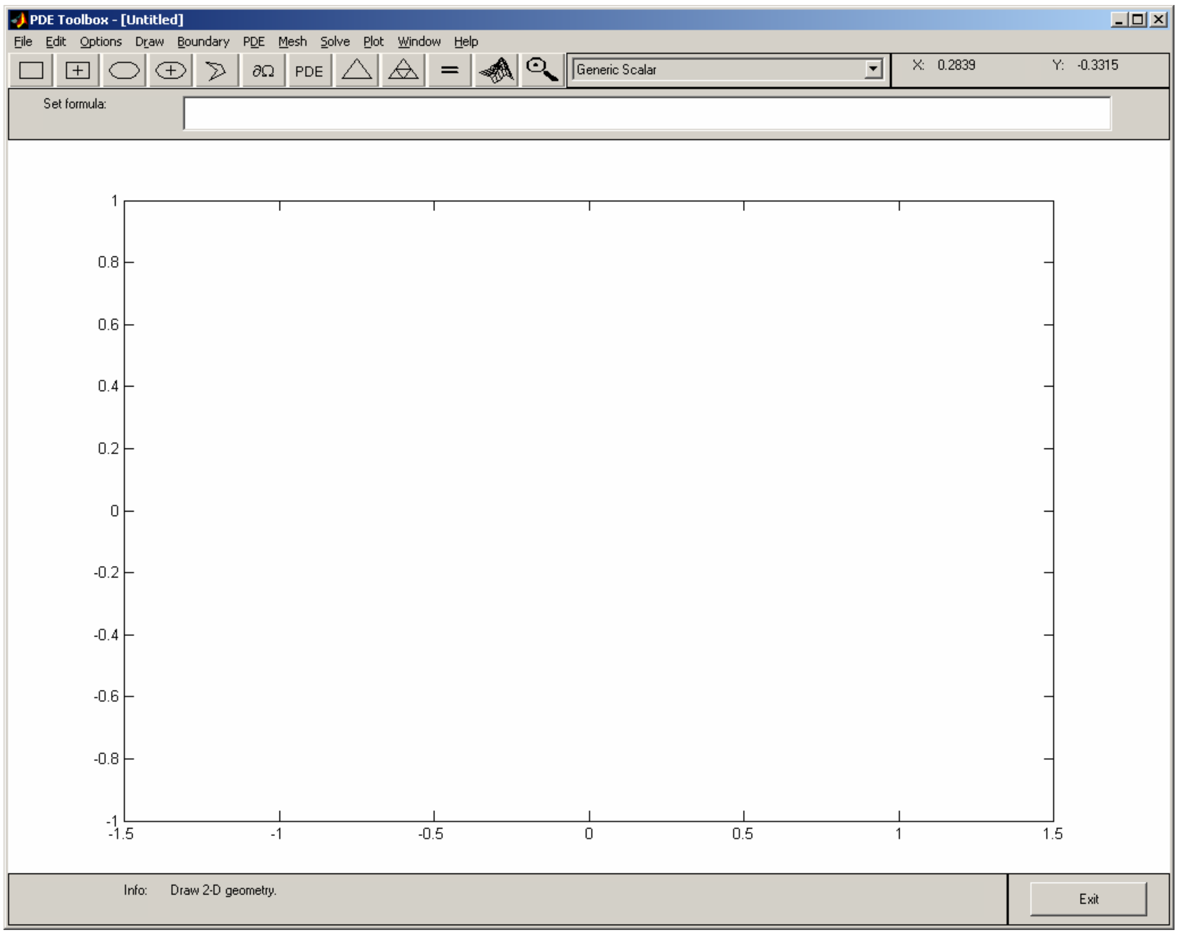


Figure 4.1 – PDE Toolbox GUI

PDE Toolbox allows solve different types of problems. It has 10 application modes:

- Generic scalar (the default mode)
- Generic system
- Structural Mechanics -- Plane Stress
- Structural Mechanics -- Plane Strain
- Electrostatics
- Magnetostatics
- AC Power Electromagnetics

- Conductive Media DC
- Heat Transfer
- Diffusion

In this paper we will refer to Generic Scalar mode. The process of finding solution via PDE Toolbox GUI is the following:

- Define problem

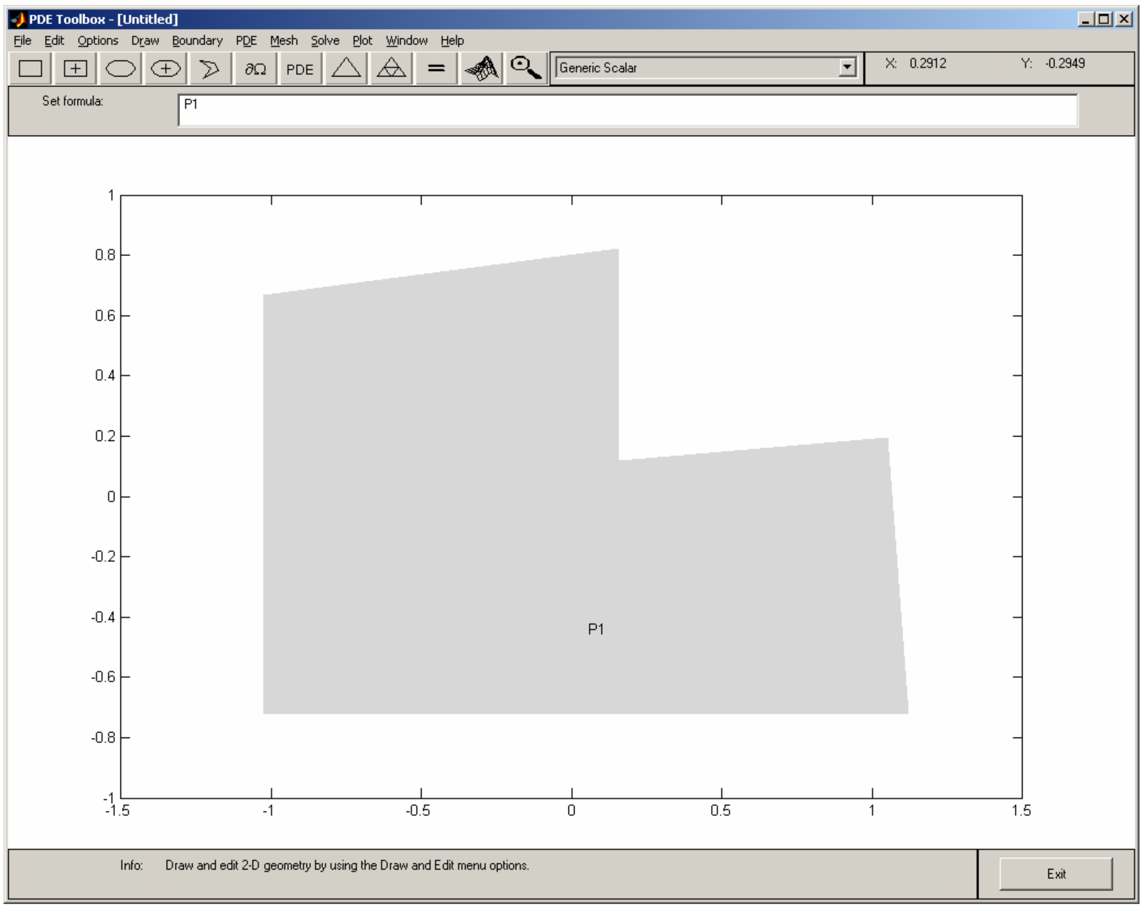


Figure 4.2 – Geometry definition: L-shaped domain

- Specify boundary conditions

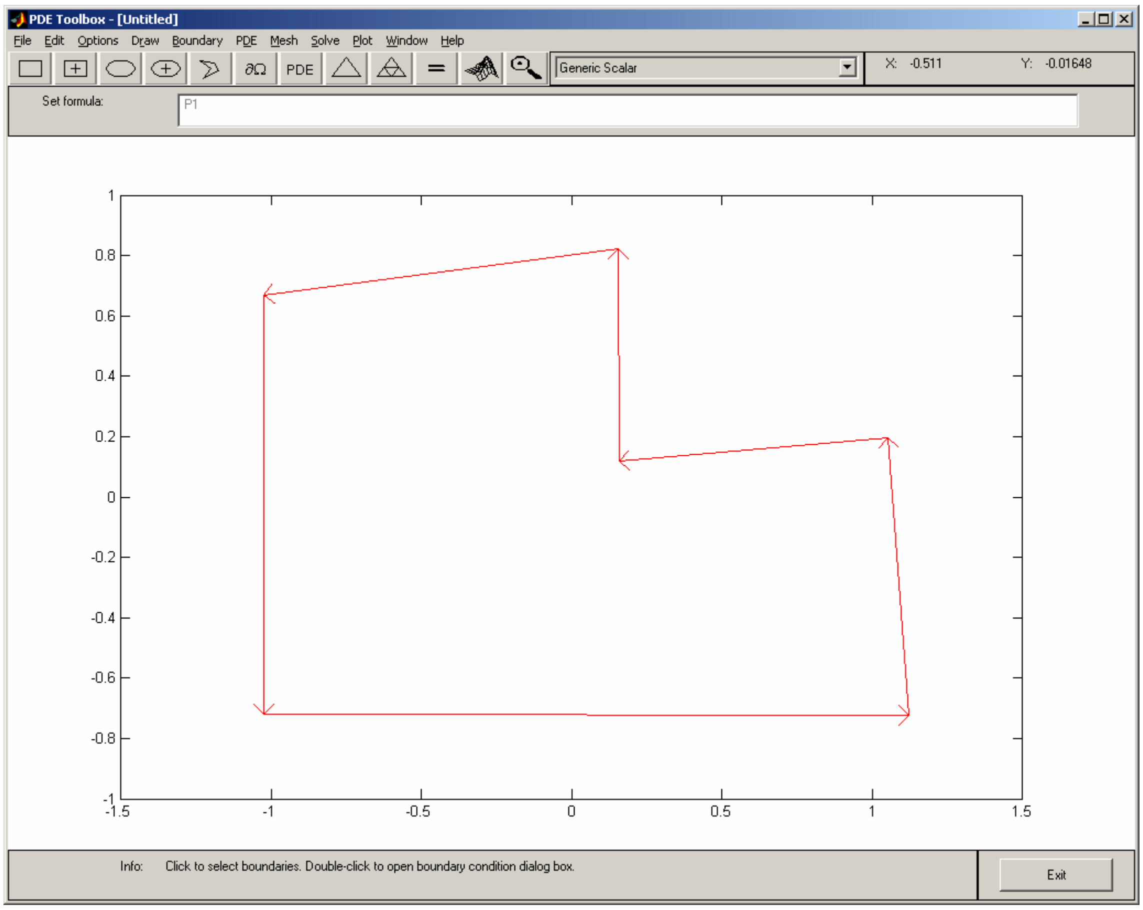
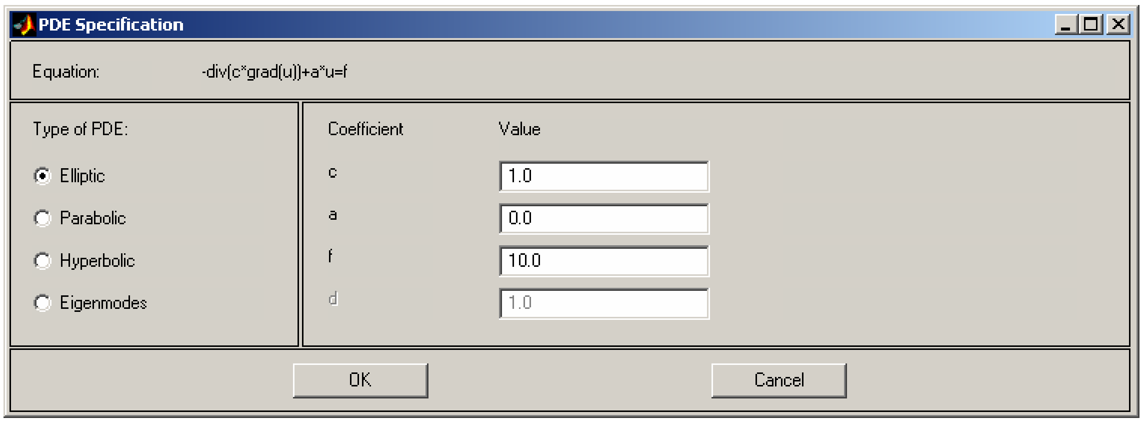

PDE Toolbox - [Untitled]
File Edit Options Draw Boundary PDE Mesh Solve Plot Window Help
Generic Scalar
X: -0.511
Y: -0.01648
Set formula:
P1
Info:
Click to select boundaries. Double-click to open boundary condition dialog box.
Exit

Figure 4.3 – Boundary conditions specification

- Set PDE coefficients


PDE Specification

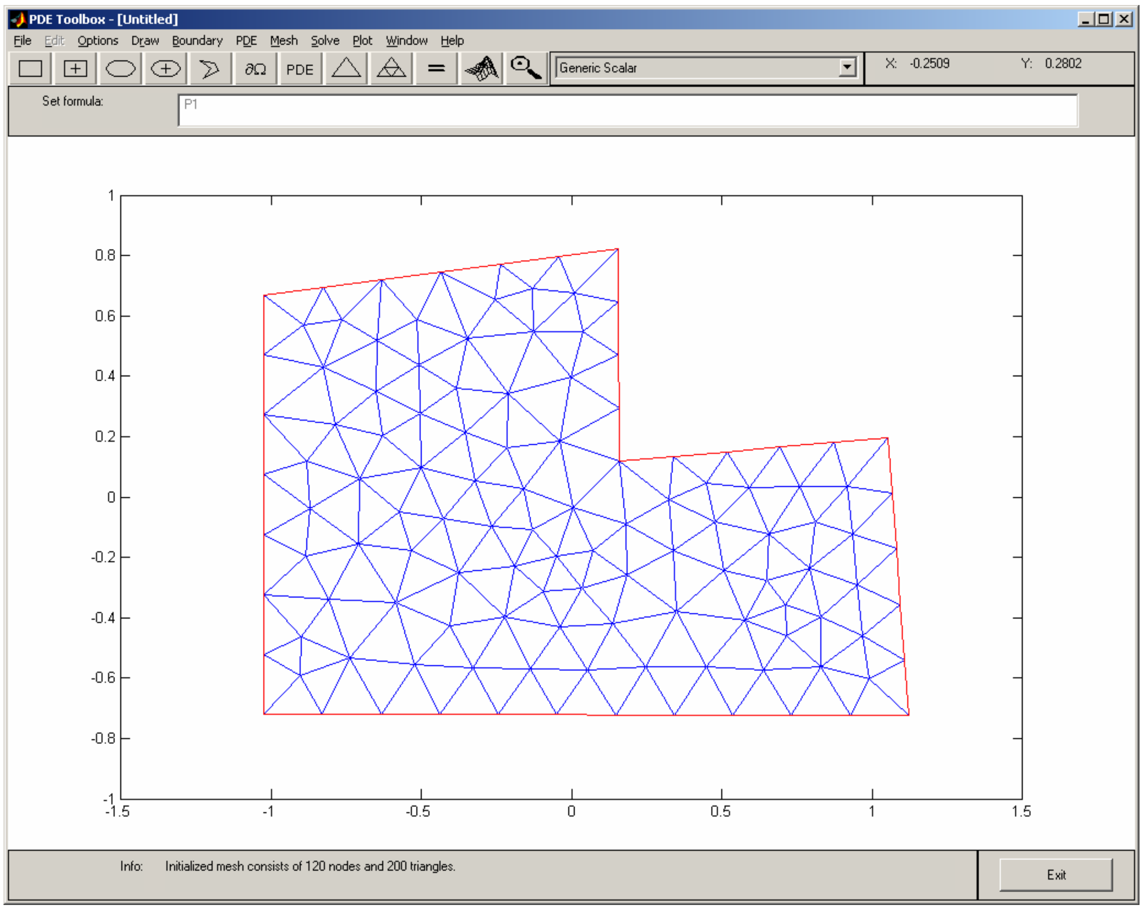
Equation: -div(c*grad(u))+a*u=f

Type of PDE: [✓ Elliptic] Parabolic Hyperbolic Eigenmodes

Coefficient	Value
c	1.0
a	0.0
f	10.0
d	1.0

Figure 4.4 – PDE specification

- Build mesh



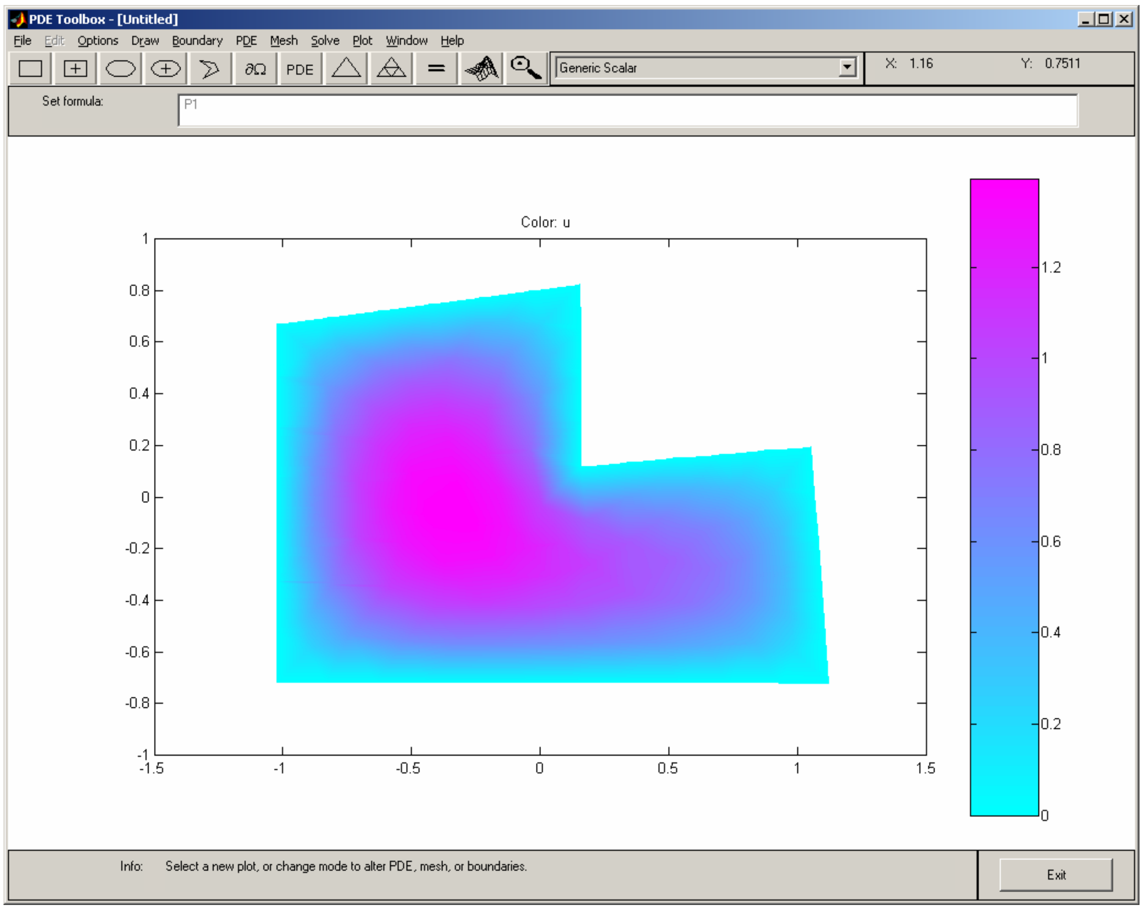
Info: Initialized mesh consists of 120 nodes and 200 triangles.

Figure 4.5 – Mesh generation

Mesh generation parameters are:

1. Maximum edge size
2. Growth rate
3. Jiggle mode
4. Refinement Method

- Solve problem



Select a new plot, or change mode to alter PDE, mesh, or boundaries.

Figure 4.5 – Solved problem

The solution is a vector of values in points of mesh.

- Plot the solution

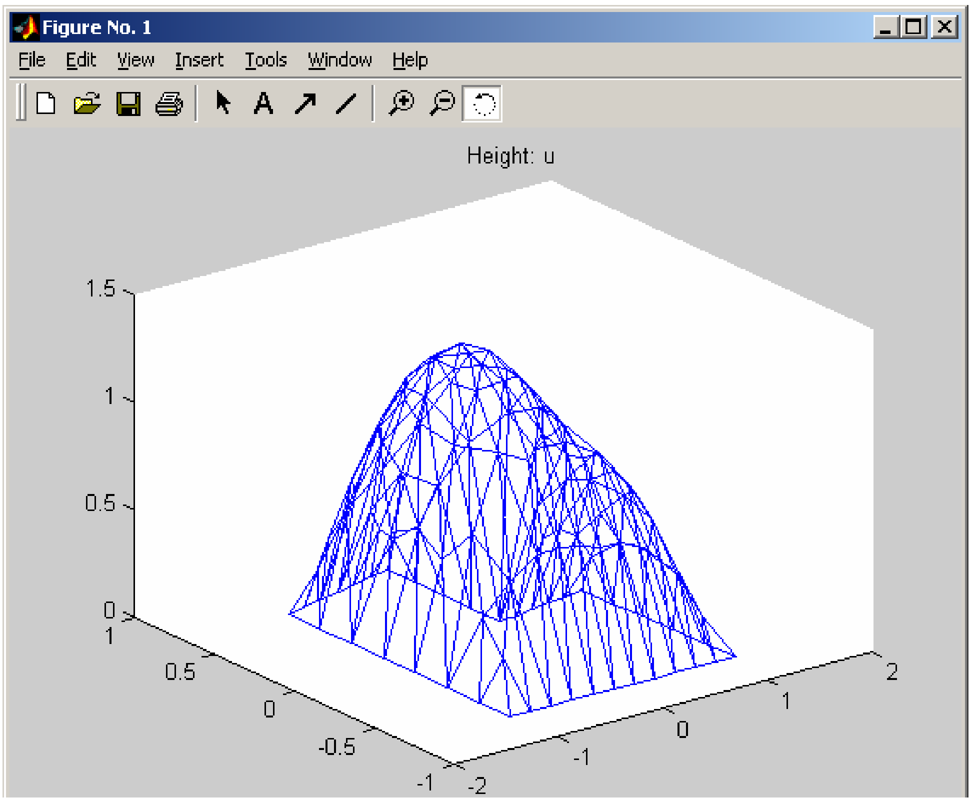


Figure 4.6 – Plotted solution

4.3 PDE Toolbox API

PDE Toolbox provides a set of functions available for use. The most significant for us are:

- `assempde` - Assemble the stiffness matrix and right-hand side of a PDE problem. It is the basic function of the PDE Toolbox. *assempde* assembles and solves the PDE problem by eliminating the Dirichlet boundary conditions from the system of linear equations.

  Function syntax: `u=assempde(b,p,e,t,c,a,f)`, where

  u – solution vector

  b – Boundary Conditions Matrix

  p – nodes of mesh

  e – edges of mesh

  t – triangles of mesh

  c, a, f – PDE coefficients

- `pdegrad` - The gradient of a PDE solution

  Syntax: `[ux,uy]=pdegrad(p,t,u)`

  `ux` contains $\frac{\partial u}{\partial x}$

  `uy` contains $\frac{\partial u}{\partial y}$

4.4 Estimator software

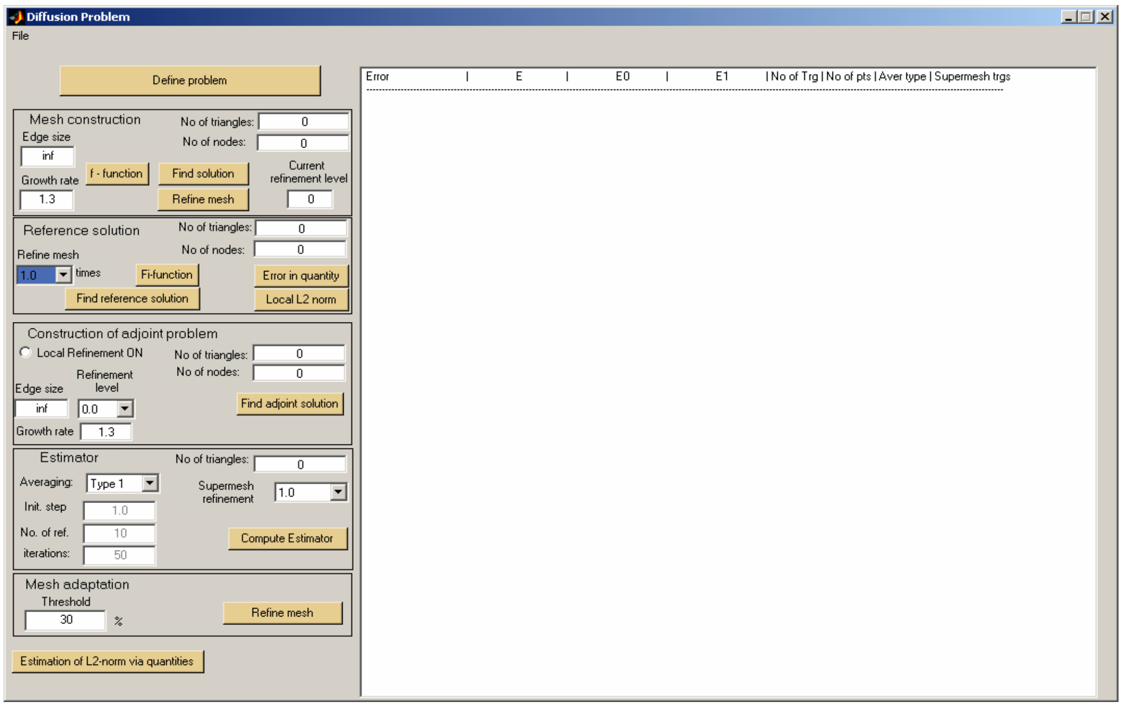
As one of the outcomes of this thesis, the pilot software was developed in order to check experimentally Error Estimation Technique (see section 3 for details). We named it "Estimator".

Estimator is implemented in MATLAB language on top of PDE Toolbox and uses its API. PDE Toolbox GUI is used to define problem and boundary conditions, then all data is exported to global namespace of MATLAB environment and accessed by our software.

Estimator introduces user-friendly interface that provides ability to vary parameters of meshes construction, gradient averaging procedures in estimation calculation and precision of estimation (see Figure 4.7).

Estimator uses PDE Toolbox API, to build numerical solution. Then, the error of the solution is estimated and results of estimation a compared to so-called "exact error". Exact error is calculated as difference between reference solution – a solution which is built over a well-refined mesh, and primal solution. The program allows vary sharpness of the reference solution, however, more precise it is more computational resources it requires.

Problem geometry and boundary conditions are defined and exported from PDE Toolbox (see Figures 4.8, 4.9, 4.10). But the difference from solution in PDE Toolbox is obligatory definition of "zone of interest".



Edge size: inf
Growth rate: 1.3
No of triangles: 0
No of nodes: 0
Current refinement level: 0

Refine mesh 1.0 times
No of triangles: 0
No of nodes: 0

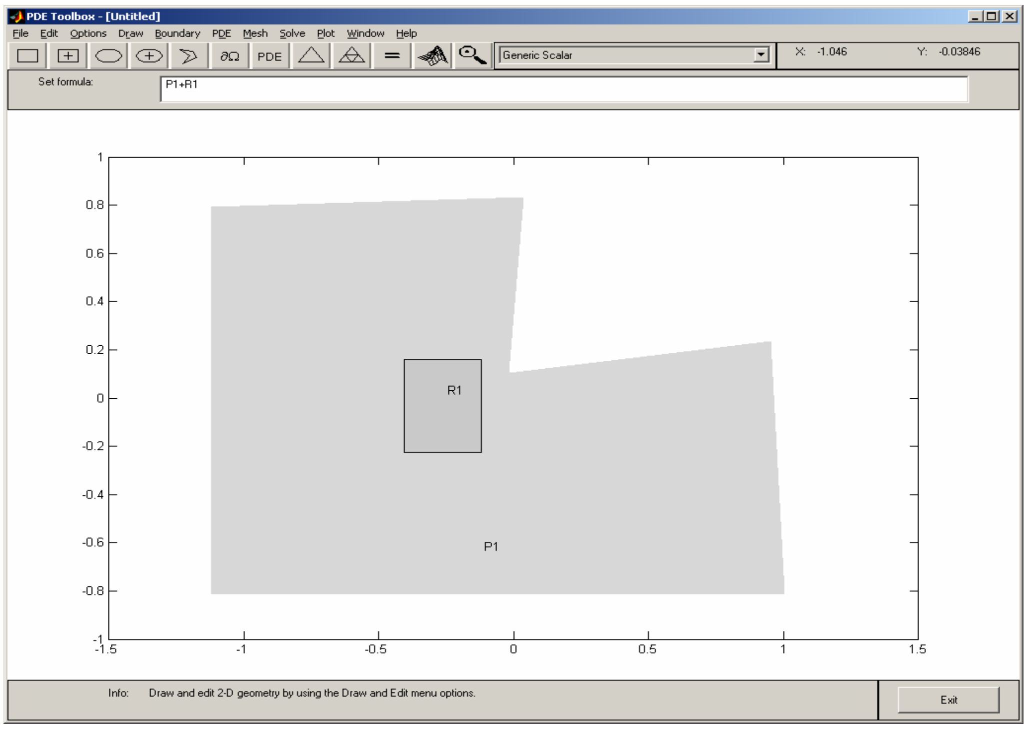
Edge size: inf
Refinement level: 0.0
Growth rate: 1.3
No of triangles: 0
No of nodes: 0

Averaging: Type 1
Init. step: 1.0
No. of ref.: 10
iterations: 50
No of triangles: 0
Supermesh refinement: 1.0

Threshold: 30 %

```
Error          |     E     |     E0     |     E1     | No of Trg | No of pts | Aver type | Supermesh trgs
```

Figure 4.7 – Estimator software in initial state


PDE Toolbox - [Untitled]
File Edit Options Draw Boundary PDE Mesh Solve Plot Window Help
PDE
Generic Scalar
R1
P1
Exit

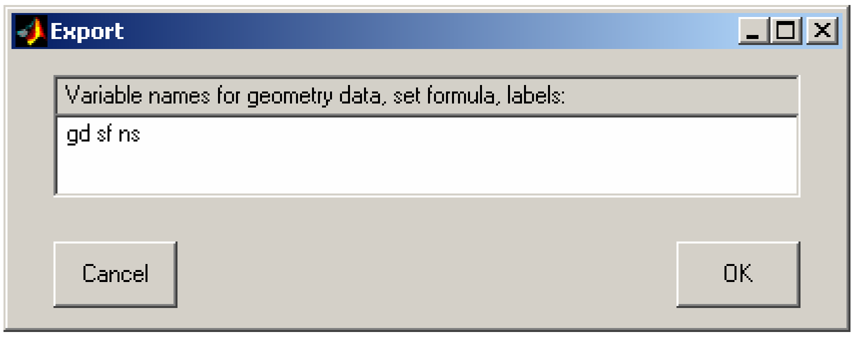
X: -1.046 Y: -0.03846

Set formula: P1+R1

Info: Draw and edit 2-D geometry by using the Draw and Edit menu options.

Figure 4.8 – Geometry definition with zone of interest

The Geometry then must be exported from Draw menu – option “Export Geometry Description, Set Formula, Labels”.

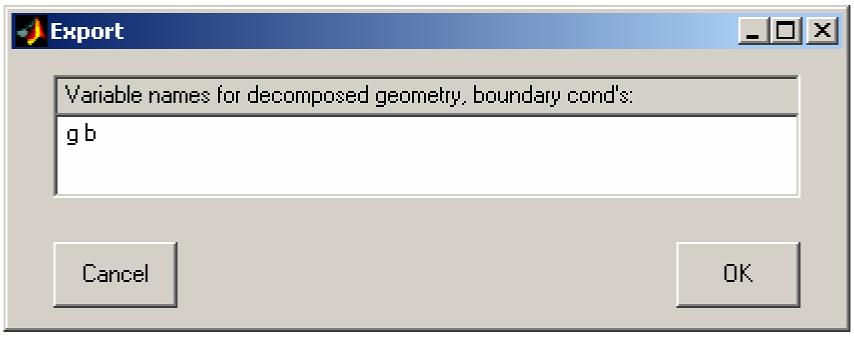


Variable names for geometry data, set formula, labels:

gd sf ns

Figure 4.9 – PDE Toolbox geometry export window

The next step is to switch to Boundary Mode (See figure 4.3) and export Decomposed Geometry and Boundary Conditions to MATLAB workspace (figure 4.10).



Variable names for decomposed geometry, boundary cond's:

g b

Figure 4.10 - PDE Toolbox decomposed geometry and boundary conditions export

After all exports are done we do not need PDE Toolbox GUI anymore.

4.4.1 *f*-function definition

The *f*-function (see 4.3 for details) can be defined as constant with certain user-defined value over whole area, or as a peak, which has radius and height. Peak has 0-value out of radius area and follows the equation:

$$f(x,y) = V_1 \cdot Cos\left(\frac{\pi}{2} \cdot \sqrt{\frac{(x - x_{1center})^2 + (y - y_{1center})^2}{R_1}}\right), \tag{4.1}$$

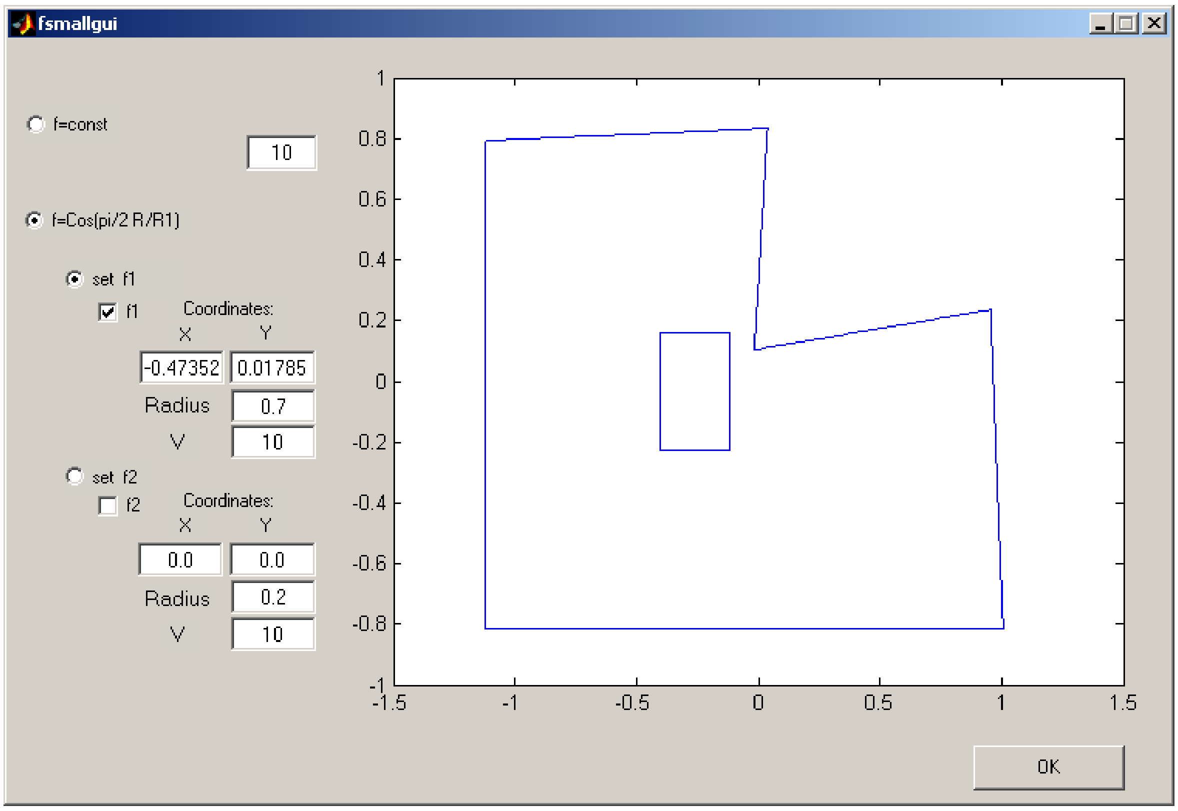
where

V_1 - peak height value

R_1 - radius

$x_{1center}$, $y_{1center}$ - coordinates of peak center

It is also possible to define two peaks in different points of geometry and set their parameters independently. Figure 4.11 shows the window, where these parameters are set.



[ ] f=const 10

[● f=Cos(pi/2 R/R1)]

[● set f1]

[✓ f1] Coordinates:

	X	Y
	-0.47352	0.01785
Radius		0.7
V		10

[ ] set f2

[ ] f2 Coordinates:

	X	Y
	0.0	0.0
Radius		0.2
V		10

Figure 4.11 – *f*-function definition and export to workspace

4.4.2 Finding primal solution

We find primal solution using exported geometry and boundary conditions. The parameters for mesh generation are set by user in appropriate fields. The solution is built using `assempde` function from PDE Toolbox API. We pass as a parameter our user-defined *f*-

function, which can be set as an independent value in each grid point. Figure 4.12 demonstrates primal solution built over our geometry taking into account f-function values.

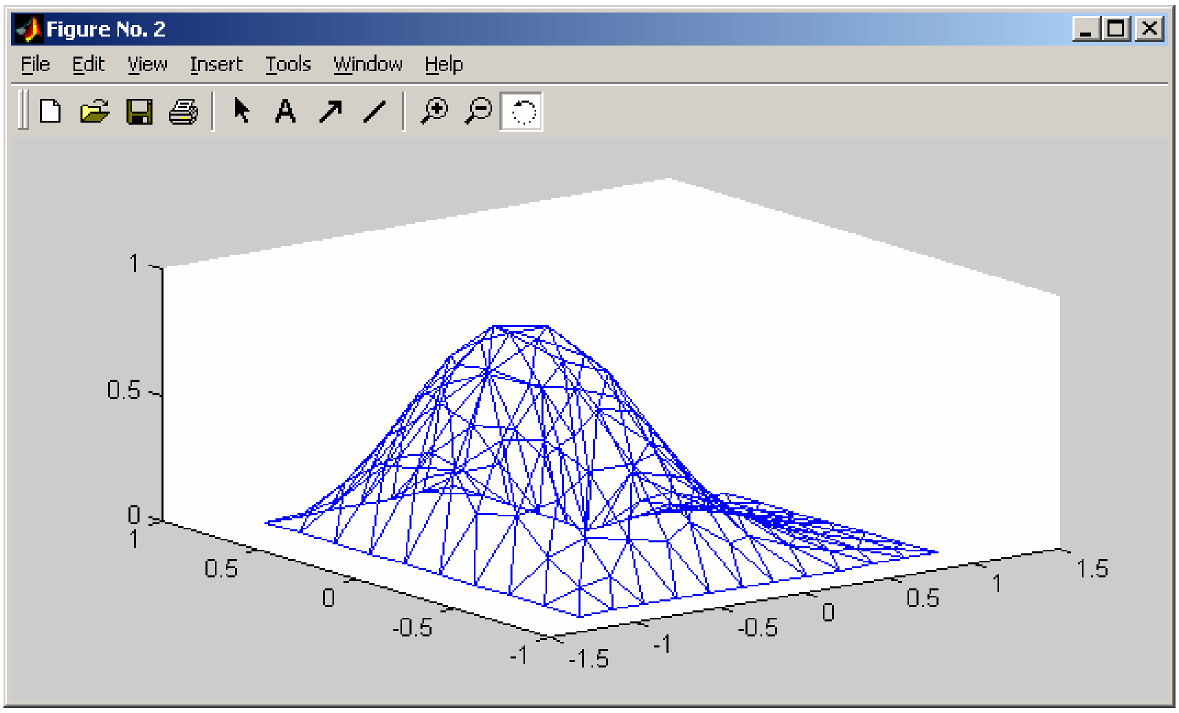


Figure 4.12 – Primal solution

4.4.3 φ-function definition

When primal solution is found, we define φ-function, which emphasizes the "zone of interest". This function influences the Adjoint Solution construction and Exact Error computation. Due to simplicity, we assume that zone of interest always rectangular. So we defined 5 possible types of φ-function:

- constant over whole area
- Bilinear function such that $\varphi_i(Point_j) = \delta_{ij}$, where δ_{ij} is the Kronecker symbol. In other words we define peak in one of zone corners with descent to zero in all other corners.

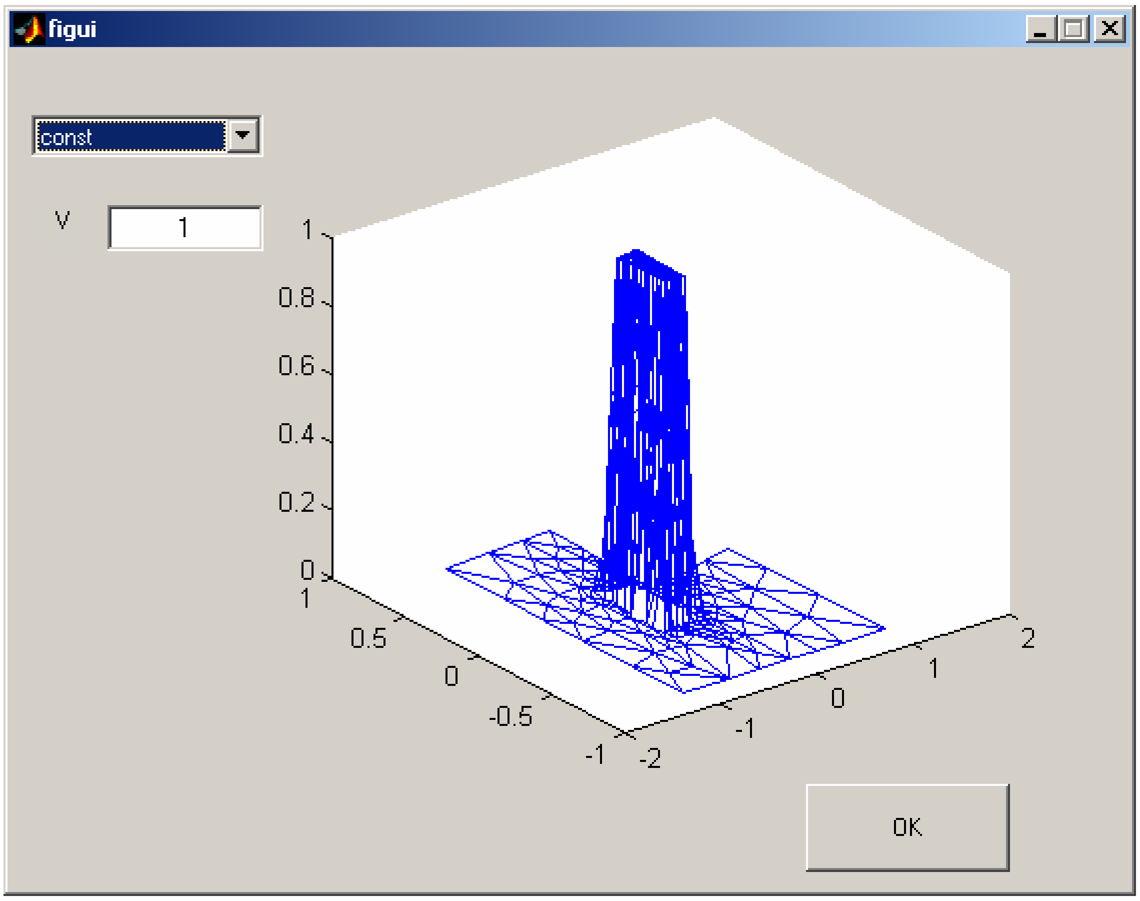


Figure 4.13 - φ - function definition dialog window

4.4.4 Reference solution and exact error computation

The mesh for reference solution is built as a multiple refinement of primal mesh. The number of refinements is set by user, however too many refinements increase computational load and may be calculated for a long time. All parameters for reference solution are taken from primal solution.

Under “Exact error” we assume the difference between exact solution and primal solution. We consider reference solution as exact one. Difference is calculated in two ways:

- Quantity of interest $\int_{\Omega} \varphi(u-u_h)dx$

- Local L2-norm $\sqrt{\int_{\Omega} |u - u_h|^2 \, dx}$

4.4.5 Adjoint problem

The adjoint problem is similar to Primal Problem. We take the same geometry and set zero boundary conditions. One important thing is that boundary condition of certain segment in adjoint problem must be of the same type as primal one.

The parameters for mesh generation can be set by user. For adjoint solution we pass φ-function as a parameter to `assempde` function.

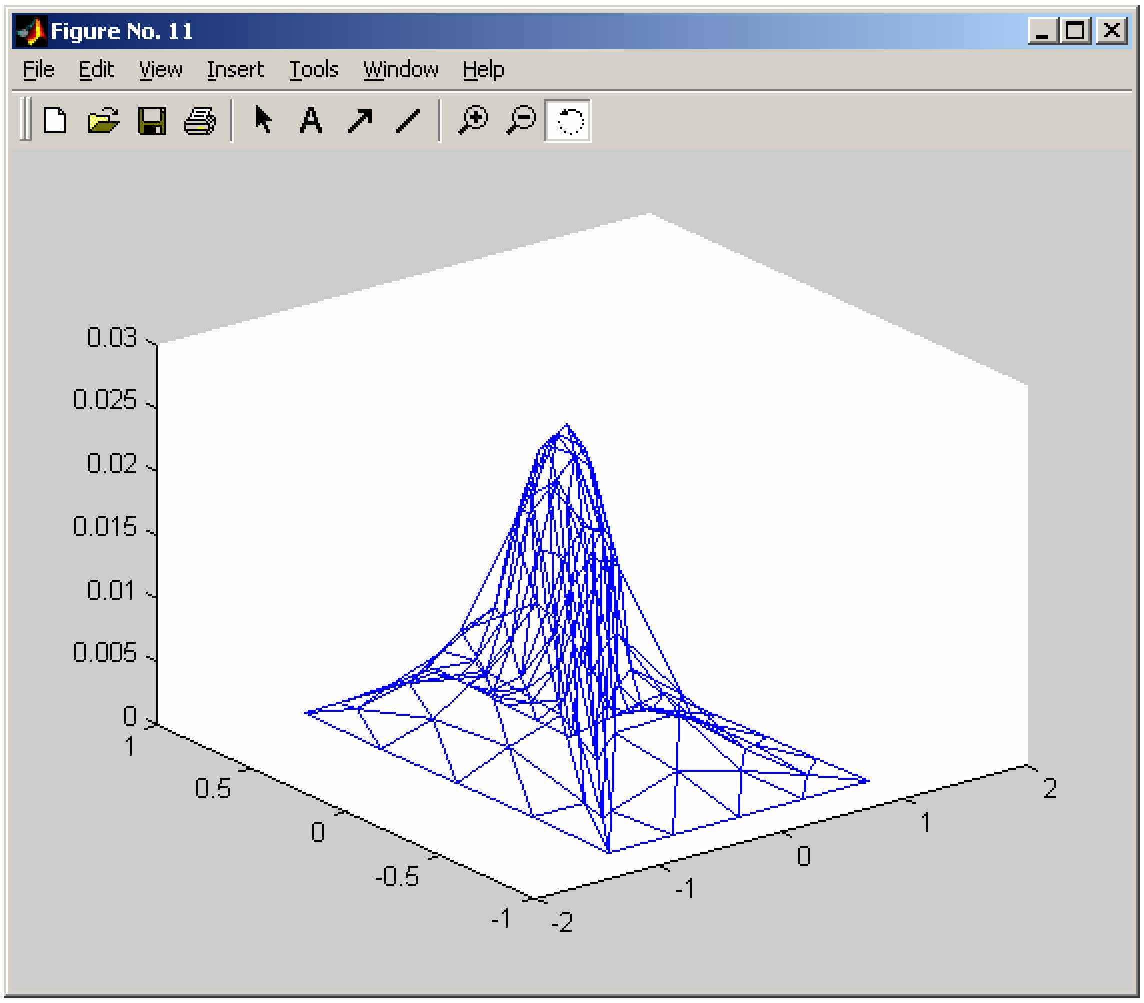


Figure 4.14 – Adjoint solution

4.4.6 Estimation

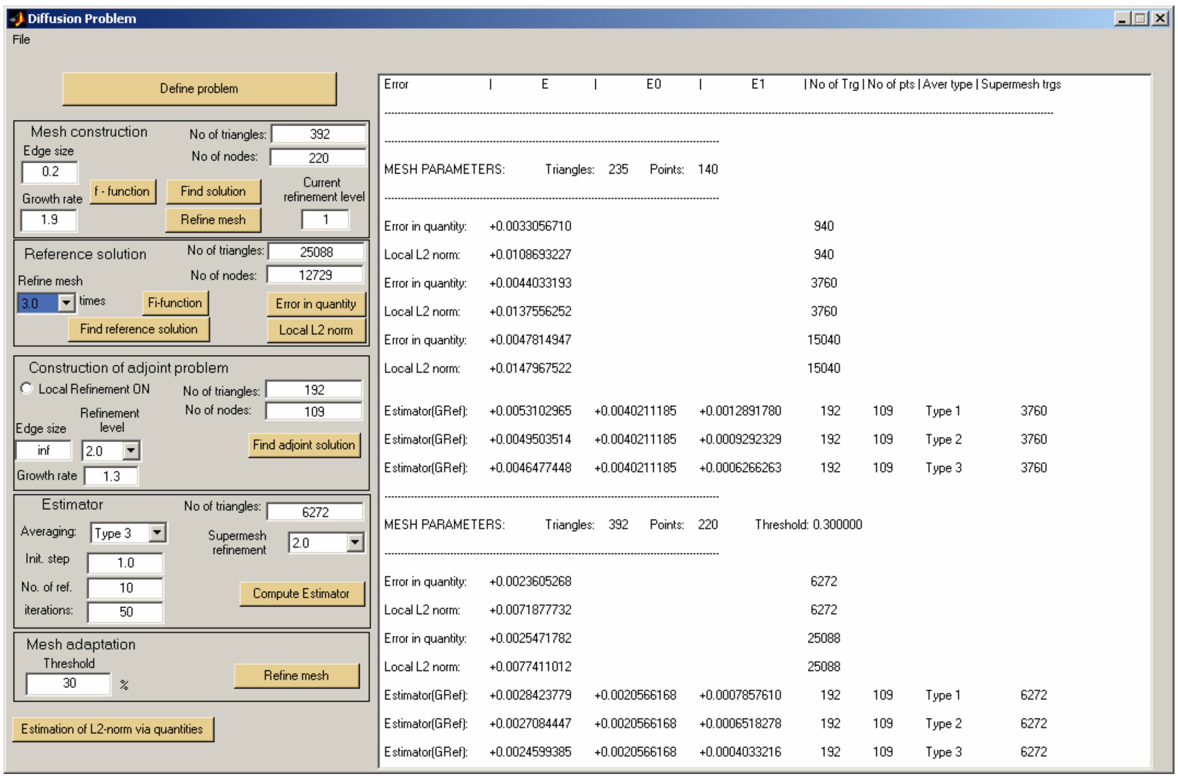
The parameters for estimation which user can vary are gradient averaging type and supermesh refinement level.

There are three types of averaging procedures (see subsection 3.1.6 for details).

Supermesh is a well-refined mesh introduced for facilitation of operation with values, defined over heterogeneous primal and adjoint meshes. Supermesh is built as a refinement of primal mesh. Of course it harms the precision, but at this moment there is no mesh generator in MATLAB, which could combine two meshes and build a mesh as an intersection with minimal number of triangles. Our experiments show that supermesh solution is acceptable even if we refine primal mesh 2-3 times.

Solutions and gradient values of primal and adjoint meshes are interpolated to supermesh.

Figure 4.15 demonstrates estimation process.



Mesh construction

No of triangles:	392
No of nodes:	220
Edge size	0.2
Growth rate	1.9
Current refinement level	1

Reference solution

No of triangles:	25088
No of nodes:	12729
Refine mesh	3.0 times

Construction of adjoint problem

Local Refinement ON	○
No of triangles:	192
No of nodes:	109
Edge size	inf
Refinement level	2.0
Growth rate	1.3

Estimator

No of triangles:	6272
Averaging:	Type 3
Supermesh refinement	2.0
Init. step	1.0
No. of ref.	10
iterations:	50

Mesh adaptation

Threshold	30 %

```
Error             |        E        |        E0        |        E1        | No of Trg | No of pts | Aver type | Supermesh trgs
--------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------
MESH PARAMETERS:      Triangles:   235    Points:   140
------------------------------------------------------------------------
Error in quantity:   +0.0033056710                                              940
Local L2 norm:       +0.0108693227                                              940
Error in quantity:   +0.0044033193                                             3760
Local L2 norm:       +0.0137556252                                             3760
Error in quantity:   +0.0047814947                                            15040
Local L2 norm:       +0.0147967522                                            15040

Estimator(GRef):     +0.0053102965   +0.0040211185   +0.0012891780     192     109     Type 1        3760
Estimator(GRef):     +0.0049503514   +0.0040211185   +0.0009292329     192     109     Type 2        3760
Estimator(GRef):     +0.0046477448   +0.0040211185   +0.0006266263     192     109     Type 3        3760
------------------------------------------------------------------------
MESH PARAMETERS:      Triangles:   392    Points:   220       Threshold: 0.300000
------------------------------------------------------------------------
Error in quantity:   +0.0023605268                                             6272
Local L2 norm:       +0.0071877732                                             6272
Error in quantity:   +0.0025471782                                            25088
Local L2 norm:       +0.0077411012                                            25088
Estimator(GRef):     +0.0028423779   +0.0020566168   +0.0007857610     192     109     Type 1        6272
Estimator(GRef):     +0.0027084447   +0.0020566168   +0.0006518278     192     109     Type 2        6272
Estimator(GRef):     +0.0024599385   +0.0020566168   +0.0004033216     192     109     Type 3        6272
```

Figure 4.15 – Estimation results

4.4.7 Mesh adaptivity

During the error estimation process we calculate the contribution of each triangle to integral error. Triangles with worst values then are marked and can be refined. User can set threshold value to mark more or less triangles with respect to their error values.

After new primal solution is built, reference solution and exact error should be recalculated. Then new estimation value can be obtained.

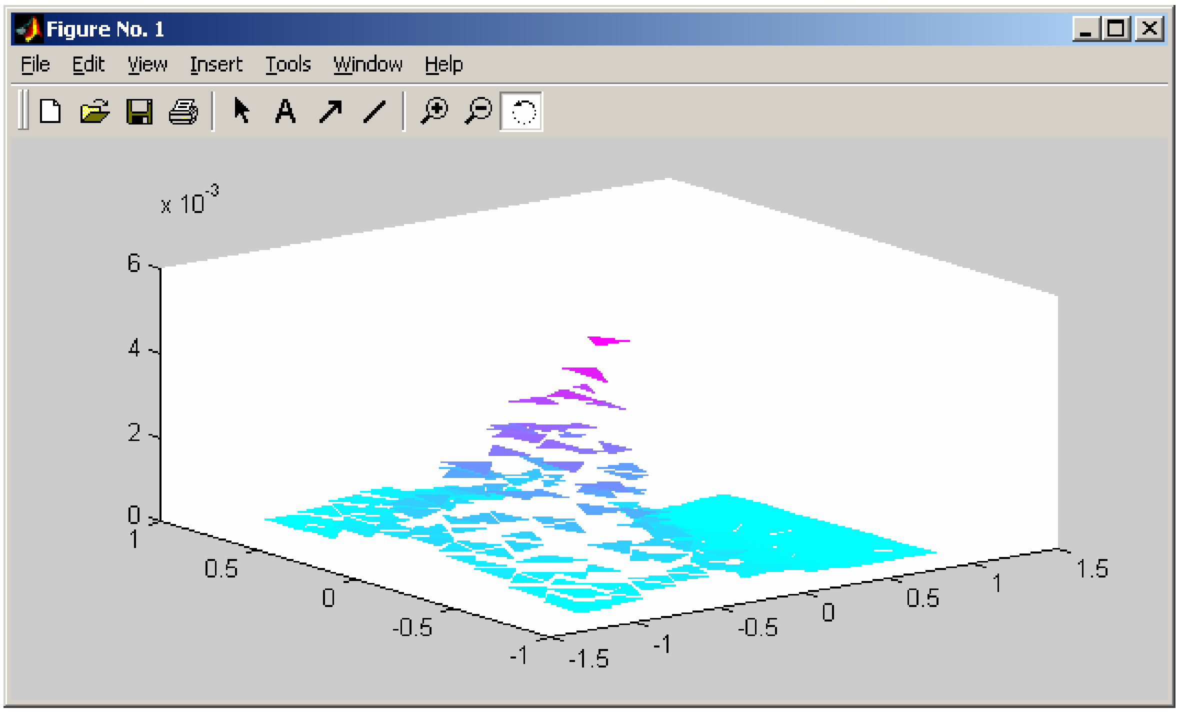


Figure 4.16 – Triangles contribution to integral error

4.4.8 Local L2-norm estimation

There is also an implementation of Local L2 norm estimation via quantities of interest. The formula which describes the estimation procedure is the following:

$$Est = \sqrt{B^{-1} I \cdot I}\,, \tag{4.2}$$

where

“dot” means the scalar product

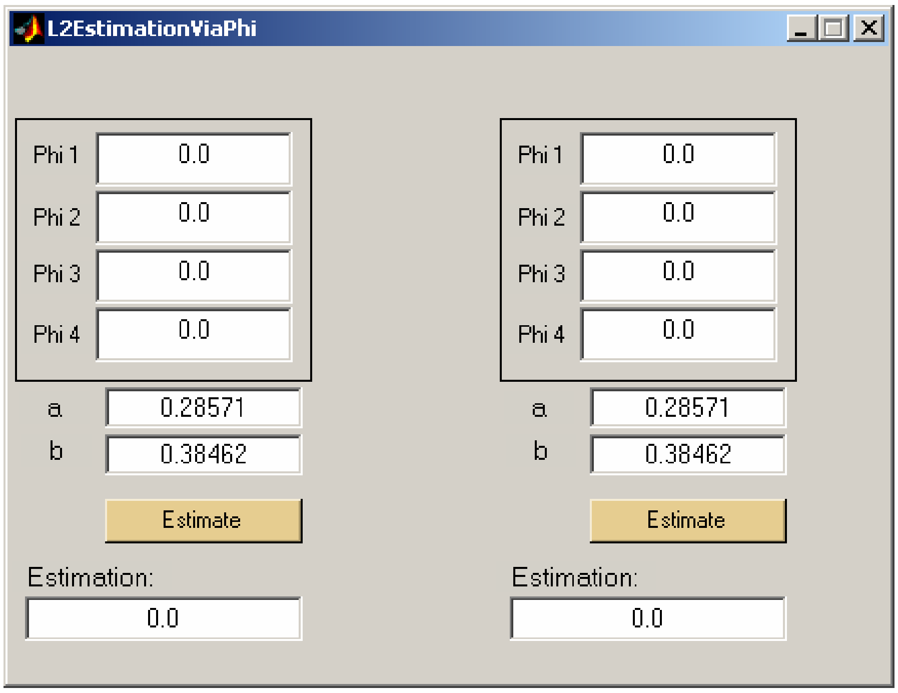
I – vector, containing the "Quantity of Interest" values computed with different φ-functions:

$$I = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} \tag{4.3}$$

B - 4x4 matrix of following values:

$$B = \begin{bmatrix} \frac{ab}{9} & \frac{ab}{18} & \frac{ab}{36} & \frac{ab}{18} \\ \frac{ab}{18} & \frac{ab}{9} & \frac{ab}{18} & \frac{ab}{36} \\ \frac{ab}{36} & \frac{ab}{18} & \frac{ab}{9} & \frac{ab}{18} \\ \frac{ab}{18} & \frac{ab}{36} & \frac{ab}{18} & \frac{ab}{9} \end{bmatrix} \tag{4.4}$$

where a and b are zone of interest side lengths.



Phi 1	0.0
Phi 2	0.0
Phi 3	0.0
Phi 4	0.0
a	0.28571
b	0.38462
Estimation:	0.0

Phi 1	0.0
Phi 2	0.0
Phi 3	0.0
Phi 4	0.0
a	0.28571
b	0.38462
Estimation:	0.0

Figure 4.17 – L2 norm estimation window

Interface window contains two equivalent parts where L2-norm estimation can be calculated. These two parts were created for ability to compute and compare L2-norm estimations from exact error values of quantity of interest and estimated ones.

5 Error estimation web-service wrapping

In this section we will discuss the problems of heterogeneity of service classification schemes and classification as such. Another issue is common way to formalize service inputs and outputs so, that it can be discovered and accessed automatically.

5.1 Semantic web

The Semantic Web is an initiative of the World Wide Web Consortium [W3C], with the goal of extending the current Web to facilitate Web automation, universally accessible content, and the 'Web of Trust'. Semantic Web is the vision of having data defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications. The goal of Semantic Web development is promotion of existing Web to qualitatively new and higher level, utilizing machine-processable metadata associated with Web resources. Next generation of intelligent applications will be capable to make use of such resource descriptions and perform resource discovery and integration based on its semantics. Semantic Web approaches to development of global environment on top of Web with interoperable heterogeneous applications, web services [Ermolayev et al., 2004], data repositories, humans, etc.

On the technology side, Web-oriented languages and technologies are being developed (e.g. RDF, RDF-Schema, DAML+OIL, OWL, DAML-S) [RDF], [RDFS], [DAML+OIL], [OWL], [DAML-S] schema and ontology integration techniques are being examined and refined. The success of the Semantic Web will depend on a widespread adoption of these technologies.

Management of resources in Semantic Web is impossible without use of ontologies, which can be considered as high-level metadata about semantics of Web data and knowledge. Ontologies are content theories about the sorts of objects, properties of objects, and relations between objects that are possible in a specified domain of knowledge.

5.2 RDF

The Resource Description Framework [RDF] is a framework for representing information in the Web. The underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, a predicate and an object. A set of such triples is called an RDF graph.

Each triple represents a statement of a relationship between the things denoted by the nodes that it links.

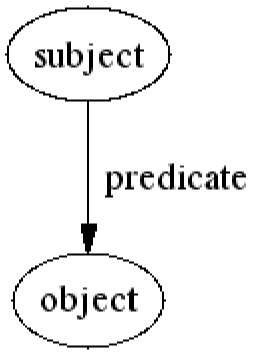


Figure 5.1 – Basic RDF triple

The direction of the arc is significant: it always points toward the object.

Example RDF in XML syntax:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/DC/"
         xmlns:os="http://somesite.org/Schema/">
  <rdf:Description about="http://people.jyu.fi/~senikiti/index.html">
    <dc:Creator rdf:resource="mailto:senikiti@cc.jyu.fi"/>
    <dc:Title> Index of my web site </dc:Title>
  </rdf:Description>
</rdf:RDF>
```

5.3 OWL

OWL Web Ontology Language [OWL] is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called ontology. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web.

In OWL, an ontology is a set of definitions of classes and properties, and constraints on the way those classes and properties can be employed. OWL is developed on top of RDF and RDF-schema and includes broad functionality but more strict and strong. All of the elements/attributes provided by RDF and RDF Schema can be used when creating an OWL document.

OWL has three sublanguages:

- OWL Lite
- OWL DL (includes OWL Lite)
- OWL Full (includes OWL DL)

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded.

OWL Full can be viewed as an extension of RDF, while OWL Lite and OWL DL can be viewed as extensions of a restricted view of RDF. Every OWL (Lite, DL, Full) document is an RDF document, and every RDF document is an OWL Full document, but only some RDF documents will be a legal OWL Lite or OWL DL document.

Below you can see simple fragment of ontology class definition. Namespaces used show that OWL uses constructs of RDF and RDFS.

```
<owl:Class rdf:ID="Wine">
     <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#madeFromGrape"/>
              <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
        </owl:Restriction>
      </rdfs:subClassOf>
      ...
</owl:Class>
```

5.4 GUN concept

Concept of Global Understanding eNvironment [Terziyan, 2003], [GUN] is designated to join most recent research results, standardization effort, features and benefits of Semantic Web, Web Services, Peer-to-Peer and Agent technologies in integral resource management oriented framework.

Basic features of GUN are:

- Semantic Web principles reused for integration of resources and interoperability;
- Proactive, goal-driven, self-maintaining resources;
- Wide spectrum of supported resource types;
- Resource-to-resource communication;
- Mobility of resources.

In GUN, resources are Semantic Web enabled in a sense that they are annotated using standards of ontology-based representation, in which semantics of separate piece of data identified by a reference to assigned meaning (a concept from ontology).

In GUN, resources are active participants of the environment (besides the applications that use them). This assumes adding associated resource-maintenance mechanism – resource-maintenance agent, which provides goal-oriented capabilities and support for self-interested, self-maintaining proactive behavior. Due to this, resources inherit agent features and become capable to have assigned goals. Mechanism of assigning goals and behavioral models to a resource can be considered as a higher-level development tool for resource maintenance applications.

Types of resources integrated into GUN are not limited only to digital documents and database content. Real-world objects can be also represented as resources capable, for example, to accept and respond to queries, interact with other resources in order to achieve own goals. Specific adaptation mechanism has to be elaborated for communication between resource and its agent. Taking into account great variety of possible resource types, diversity of their formats and ways of accessing them, adaptation of such resources in resource management environment will be an important challenge for development of GUN. General-purpose resource agents can be set up to maintain virtually any kind of supported resources using resource adapters. Development of resource adapters for each class of resources is required for GUN.

Another adaptation to be performed is an adaptation of resource semantics to environment where it resides. Initially none of the resources is semantically annotated and available for semantic-enabled environment. This is due to use of semantic-enabled environment. This is due to use of real-world resources and other types of resources that have to become Semantic Web enabled. In GUN adaptation is made by a resource agent, which "wraps" data retrieved from resource with semantic labels and delivers semantically annotated data from outside to a resource stripping out semantic markup.

Additionally to original challenges provided by the Semantic Web approach:

- resource discovery based on semantic descriptions;
- semantics-based resources integration;

The GUN concept assumes at least two new challenges:

- resource state monitoring: retrieval, semantic annotation and accumulation of a resource state data and its initial preprocessing for discovery of special classes of states (alarms) that require advanced handling and initiate resource-maintenance mechanisms;
- resource state diagnostics/maintenance: learning models of resource state classification (using various machine learning techniques), automated testing,

quality evaluation and certification of learned models and application of models for diagnostics and maintenance decision making.

Essential concern in resource maintenance is processing data about resource state. State of the resource can be understood in broader meaning than just values of some internal properties, but also as a relation between internal state (including its history), external factors and the purpose of resource existence. The analysis of the factors that influence state of the resource provides view to characteristics of balance between internals and externals of the resource, also meant as resource condition. The open standard for representation of states and conditions of complex industrial objects and processes is required for efficient resource diagnostics and maintenance by heterogeneous applications.

In the context of resource maintenance, the challenge is to create Resource State/Condition Description Framework (RSCDF) [SmartResource, 2004], [Kaykova et. al., 2004], as an extension to RDF, which introduces upper-ontology for describing maintenance-oriented characteristics of resources: states and correspondent conditions, dynamics of state changes that happen, target condition of the resources and historical data about previous states.

Resources (e.g. devices) are assumed to have their own state presented as RSCDF descriptions. These descriptions are used by external applications (e.g. remote diagnostics) that support RSCDF and are able to process data presented in such format. Introduction of RSCDF allows solving problems of interoperability and resource heterogeneity (the same basic concepts will be used for state description of any kind of resources). Design of the RSCDF will follow the ontology engineering principles in the scope of Resource Description Framework developed by W3C Semantic Web Activity.

The essential component of GUN paradigm is generic resource access mechanism (Semantic Adapter) [SmartResource, 2004], [Kaykova et. al., 2004]. It contains following model components:

- Semantic Adapter, generic software component for connecting resources

- Device-Data Source, an object that is either device accessed directly via specific hardware/software interface or another kind of device state-data source: database or emulated (virtual) device.
- Resource Browser, a software that provides access to resources with semantic adapter via semantic interface; browser has user interface (UI) and can be used as part of semantic adapter to Human;
- Human, a person involved into activities performed in the system
- Service Component, a standalone software component (executable, program code, dynamic library) or web service used for performing some servicing actions.

5.5 Generic wrapping mechanism

Three main technologies provide a basis for Web-Service access, description and discovery, they are respectively SOAP, WSDL and UDDI. However, approach based only on these technologies lacks "common vocabulary" for description of different types of service input and output parameters. The problem lies not only in specification of variable/parameter type (e.g. String, int, double in Java notation) but in more complete explanation of parameter meaning. This kind of full description is possible within terms of Ontology and already standardized language for it – Ontology Web Language (OWL). Furthermore usage of Ontology does not provide common understanding because each service vendor can develop his/her own ontology and consider it as true one. That is why we need some common environment where it would be easy to establish connections between different concept hierarchies and provide a certain structure for components presentation. This kind of structure proposed by Industrial Ontologies Group [IOG, 2004] is GUN – Global Understanding Environment [Terziyan, 2003].

At present step it is hard to provide absolutely new basement for GUN, so we will try to map it to already existent technologies. The vision we try to provide here does not pretend to cover all the issues the GUN concept declares, but it allows integrate already existent approaches and services, and makes them consistent to GUN paradigm.

First of all let us set roles for participating technologies:

Technology	**Function**	**Role in GUN-compliant environment**
SOAP	Carrier protocol	Used as a low-level protocol for messages exchange
WSDL	Service Description	Describing services as a set of endpoints that exchange messages and how service should be accessed using terms of common ontology
UDDI	Registry for Resource Discovery	Used as a registry for resources and stores upper-ontology, which provides taxonomy of ontologies for different application areas and meta-ontology for their mapping
OWL	Ontology Description	Used as common language for Ontology Definition
RSCDF	Resource Description	Extended resource state/condition description in terms of common ontology

Table 5.1 – Roles of technologies in GUN environment

Here we should clarify the difference between WSDL and RSCDF descriptions. WSDL is good for defining messaging and functionality, while RSCDF allows define parameters of the resource. WSDL service operation may refer to RSCDF type of parameters in input and output messages.

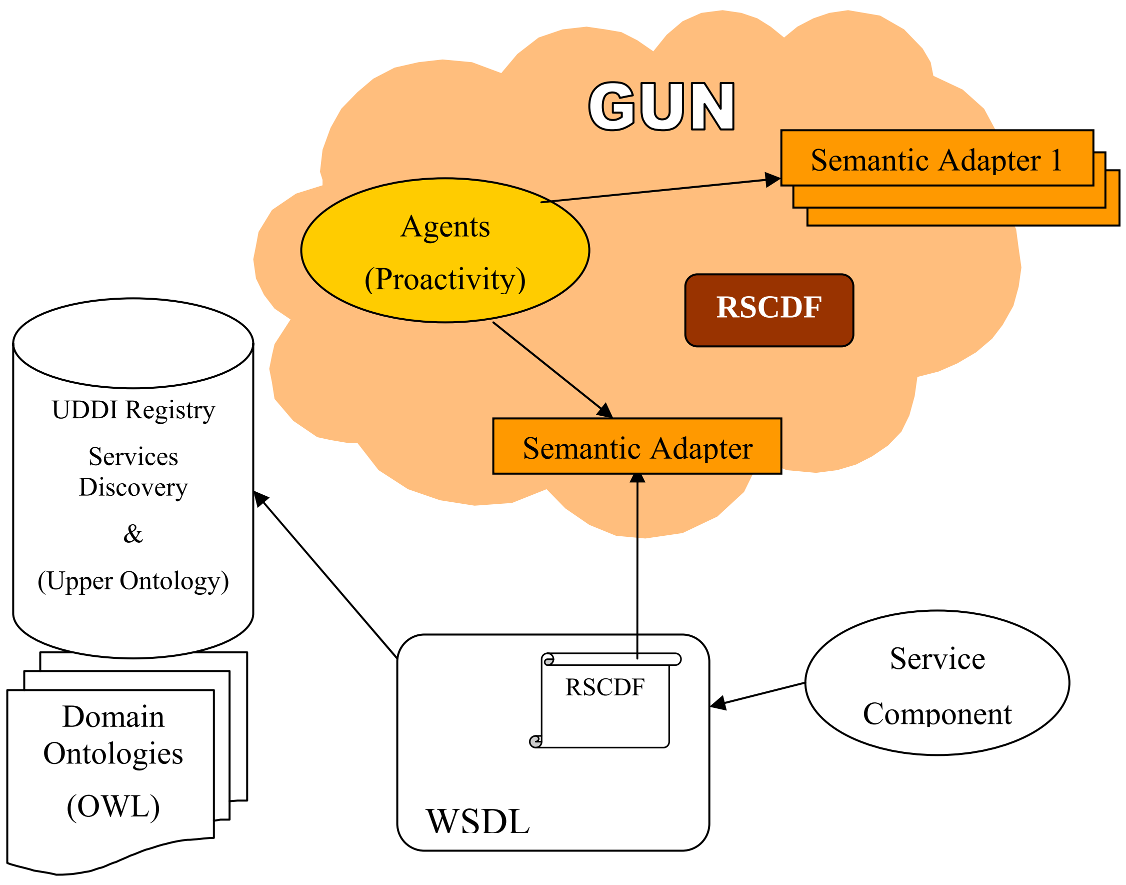


Figure 5.2 – Schema of components relations

5.6 Error estimation wrapping

First of all we have to define service component and define its input and output parameters. The main function in our software is estimation function, which receives as input two solutions – primal and adjoint. The output of the function is estimated error value. As far as adjoint solution belongs to estimation technique, we will pass instead geometry description and boundary conditions of the primal solution. So function input might look like:

```
err=estimate(p,e,t,u,g,b)
```

Where

p - number of mesh points

e - number of edges

t - number of triangles

u - FEM-solution

g – decomposed geometry matrix

b – boundary conditions matrix

err – estimated error value

To define such a function inputs and outputs we need at least following ontologies:

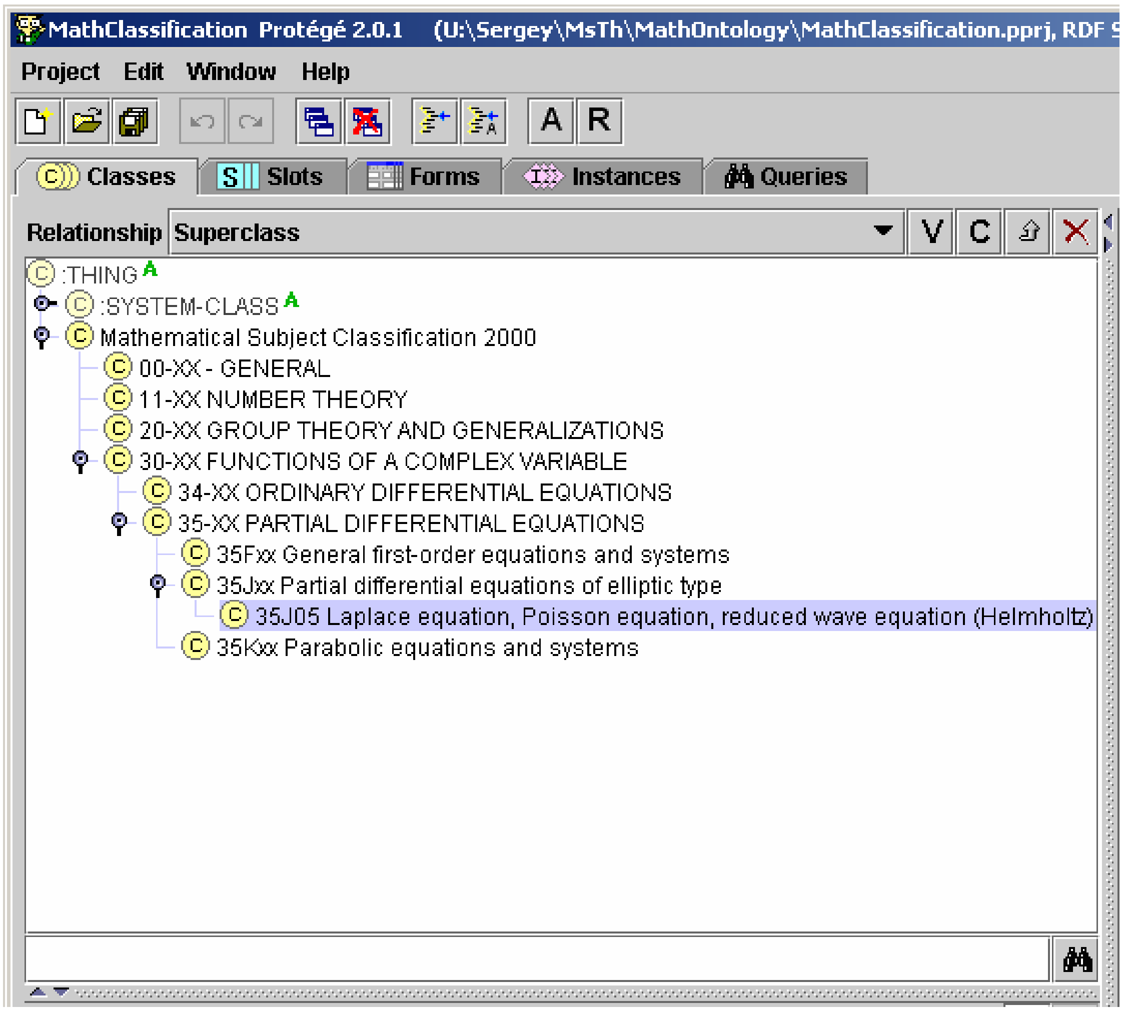
- service ontology(describes in more general what type of service this service belongs to)
- mathematical ontology(what type and class of equations are solved)
- parameters ontology(specify parameter types and format)

The table below shows how concrete ontologies, belong to resource descriptions in terms of WSDL and RSCDF languages.

UDDI upper ontology (references)		
Service Ontology	Mathematical Ontology	Parameters Ontology
WSDL		RSCDF

Table 5.2 – Ontology-to-description correspondence

For ontological modeling we will refer to American Mathematical Society [AMS] Mathematical Subject Classification [MSC]. However, this classification is too large, so we will take only a part to describe our resource. Below is picture, describing the hierarchy in Protégé [Protégé], leading to type of problem, being estimated by our Estimator. See Figure 5.3.


MathClassification Protégé 2.0.1 (U:\Sergey\MsTh\MathOntology\MathClassification.pprj, RDF S
Project Edit Window Help
Classes
Slots
Forms
Instances
Queries
Relationship
Superclass

:THING
:SYSTEM-CLASS
Mathematical Subject Classification 2000
00-XX - GENERAL
11-XX NUMBER THEORY
20-XX GROUP THEORY AND GENERALIZATIONS
30-XX FUNCTIONS OF A COMPLEX VARIABLE
34-XX ORDINARY DIFFERENTIAL EQUATIONS
35-XX PARTIAL DIFFERENTIAL EQUATIONS
35Fxx General first-order equations and systems
35Jxx Partial differential equations of elliptic type
35J05 Laplace equation, Poisson equation, reduced wave equation (Helmholtz)
35Kxx Parabolic equations and systems

Figure 5.3 – Mathematical subject classification by AMS

This classification can be used for unique identification of mathematical area, which service belongs to. So we can refer to it as to Mathematical Ontology.

It is more difficult question for classification of parameters, because parameters may be presented in different ways. For example, MATLAB PDE Toolbox for problem solving uses certain format of matrices, where order of values is essentially important. Here we can propose a solution, based on general parameter description approach, or instead, refer to parameters structure of certain software producer. However it is still open research question [SmartResource, 2004].

Services Ontology must describe the service from service properties point of view. Whether this service is for educational purposes or not, is it commercial, what kind of Quality of Service it provides. These topics may be divided into a set of more specific ontologies, but they should be any way highly expressive and strict, like MSC of AMS.

In our vision, only ontological unambiguity can make web globally understandable.

CONCLUSIONS

Web-integration is almost impossible today without ontology usage. Companies, in order to establish business interaction, map their data structures to each other. In this way they actually establish mapping between their ontologies(e.g. models, that describe some relations between concepts). A lot of work is already done on the way to simplification of business interaction. First, it was XML language, which allowed represent data in readable format for everybody, then basing on XML syntax a wide variety of languages and protocols appeared. RDF is W3C standard and allows describe any resource but is too expressive for automated knowledge processing. OWL is a standard for defining ontologies. At the same time a number of technologies appeared as a need for today’s e-business. Among them are SOAP, WSDL, UDDI, ebXML. These technologies established a generic protocol for service discovery, access and description, however they lack common basis for defining semantics. This basis can be implemented in GUN paradigm, where all participants of interaction are considered as resources and have common representation through generic mechanism called “Semantic Adapter”.

The hot and open research problem is – how to provide a common vocabulary for all resources of environment. In this paper we discussed one possible solution – usage of already existing UDDI platform for ontology sharing. New services being registered in UDDI can be checked for validity to ontology they refer.

The RSCDF descriptions added to each resource through common ontology terms allow integrate the resources to common environment, where resources can be proactive. The proactivity is based on agent approach.

We implemented new error estimation technique and tested it over wide variety of domains. The idea to position it as a resource came from GUN paradigm and idea of proactive resources. This kind of service with unique algorithm can be an independent business entity, which can propose its facilities in different domains (e.g. education, industry, research). Such an entity can be easily reused for collaborative task solving in clusters of agents, thus providing a new value to services integration.

References

[AMS] American Mathematical Society web-site, http://www.ams.org

[Babuška, Strouboulis, 2001] I. Babuška, T. Strouboulis. The Finite Element Method and its Reliability, Oxford University Press Inc., New York, 2001.

[Brandts, Křížek, 2003] J. Brandts, M. Křížek. Gradient superconvergence on uniform simplicial partitions of polytopes. *IMA J. Numer. Anal.*, 23, 489–505, 2003.

[DAML+OIL] DAML+OIL language web page, http://www.daml.org/2001/03/daml+oil-index.html

[DAML-S] DAML-S 0.7 Draft Release, http://www.daml.org/services/daml-s/0.7/

[Ermolayev et al., 2004] Ermolayev V., Keberle N., Plaksin S., Kononenko O., Terziyan V., <http://www.cs.jyu.fi/ai/papers/IJWSR-2004.pdf>Towards a Framework for Agent-Enabled Semantic Web Service Composition, International Journal of Web Service Research, Idea Group, ISSN: 1545-7362, Vol. 1, No. 3, 2004, pp. 63-87.

[GUN] Global Understanding Environment concept, http://www.cs.jyu.fi/ai/papers /HCISWWA-2003.pdf

[Hlaváček, Křížek, 1987] I. Hlaváček, M. Křížek. On a superconvergent finite element scheme for elliptic systems. I. Dirichlet boundary conditions. *Apl. Mat.* 32, 131–154, 1987.

[IBM WSCA] IBM Web Services Conceptual Architecture document, http://www-306.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

[IOG, 2004] Official Web-Site of Industrial Ontologies Group, http://www.cs.jyu.fi/ai /OntoGroup .

[Kaykova et. al., 2004] Kaikova H., Khriyenko O., Kononenko O., Terziyan V., Zharko A., Proactive Self-Maintained Resources in Semantic Web, Eastern-European Journal of Enterprise Technologies, Vol. 2, No. 1, 2004, ISSN: 1729-3774, Kharkov, Ukraine, pp. 37-49

[Korotov et al., 2003a] S. Korotov, P. Neittaanmäki, S. Repin. A posteriori error estimation of goal-oriented quantities by the superconvergence patch recovery. *J. Numer. Math.* 11, 33–59, 2003.

[Korotov, et al., 2003b] S. Korotov, P. Neittaanmäki, S. Repin. A posteriori error estimation in terms of linear functionals for boundary value problems of elliptic type, *in Proc. of the Fifth European Conf. on Numerical Mathematics and Advanced Applications (ENUMATH-2003), Prague, Czech Republic (eds. M. Feistauer et al.)*, 1–8 (to appear)

[Korotov et al., 2004] Sergey Korotov, Pekka Neittanmäki, and Sergey Repin A posteriori Error Estimation of "Quantities of interest" for the elliptic type boundary-value problems, *in Proc. of the European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS-2004), Jyväskylä, Finland (eds. P. Neittaanmäki et al.)*, 1–16, 2004.

[Korotov, Turchyn, 2004] S. Korotov, P. Turchyn. A posteriori error estimation of "quantities of interest" on tetrahedral meshes, *in Proc. of the European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS-2004), Jyväskylä, Finland (eds. P. Neittaanmäki et al.)*, 1–20, 2004.

[Křížek, Neittaanmäki, 1984] M. Křížek, P. Neittaanmäki. Superconvergence phenomenon in the finite element method arising from averaging gradients. *Numer. Math.* 45, 105–116, 1984.

[MATLAB] MATLAB website, http://www.mathworks.com/

[MSC] Mathematical Subject Classification document, http://www.ams.org/msnhtml /classification.pdf

[OWL] OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/

[PDE Tool] PDE Toolbox site, http://www.comsol.se/products/pde/

[Protégé] Protégé ontology editor site, http://protege.stanford.edu/

[RDF] Resource Description Framework specification site, http://www.w3.org/RDF/

[RSCDF] RDF Vocabulary Description Language, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[SEMWEB] Semantic Web activity site, http://www.w3.org/2001/sw/

[SmartResource, 2004] Proactive Self-Maintained Resources in Semantic Web, Presentation of SmartResource Tekes Project, http://www.cs.jyu.fi/ai/OntoGroup /SmartResource.ppt

[SOAP] Simple Object Access Protocol W3C recommendation, http://www.w3.org/ TR/2003/REC-soap12-part0-20030624/

[Terziyan, 2003] Terziyan V., <http://www.cs.jyu.fi/ai/papers/HCISWWA-2003.pdf> Semantic Web Services for Smart Devices in a "Global Understanding Environment", In: R. Meersman and Z. Tari (eds.), On the Move to Meaningful Internet Systems 2003: <http://www-staff.it.uts.edu.au/~wgardner/HCI-SWWA.html> OTM 2003 Workshops, Lecture Notes in Computer Science, Vol. 2889, Springer-Verlag, 2003, pp.279-291.

[UDDI] Universal Description, Discovery and Integration standard site, http://www.uddi.org/specification.html

[Verfürth, 1996] R. Verfürth. *A review of a posteriori error estimation and adaptive mesh-refinement techniques*, Wiley-Teubner, 1996.

[WSArchitect] Judith M. Myerson, "Web Service Architectures", http://www.webservices architect.com/content/articles/webservicesarchitectures.pdf

[WSDL] Web Services Description Language submission, http://www.w3.org/TR/wsdl

[WSFL] Web Services Flow Language specification by IBM, http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[W3C] World Wide Web Consortium site, http://www.w3.org/

[Zienkeiewicz, Zhu, 1987] O. C. Zienkeiewicz, J. Z. Zhu. A simple error estimator and adaptive procedure for practical engineering analysis. *Internat. J. Numer. Methods Engrg.* 24, 337–357, 1987.