Solution of Algebraic Equations by Using Autonomous Computational Methods

Andrew Pownuk

Department of Mathematical Sciences
University of Texas at El Paso, El Paso, Texas, USA
ampownuk@utep.edu, https://andrew.pownuk.com

AMS Fall Central Sectional Meeting
September 12-13, 2020

Outline

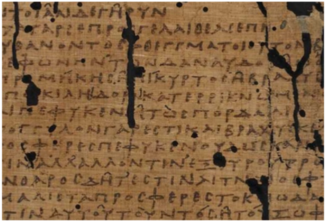
Summary

- Part 1: Formal Methods
  - Education
  - Mathematics
  - Engineering
  - Generalization
- Part 2: Autonomous Computational Methods

Online Learning

Early Examples of Distance Learning






Online Learning

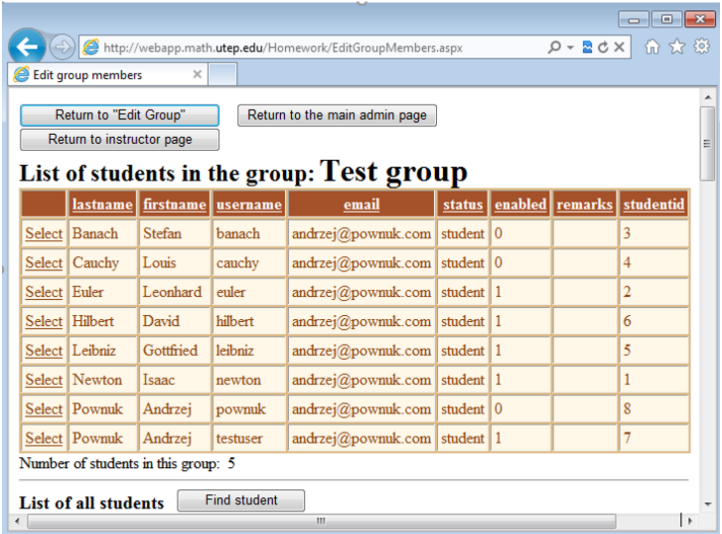
User Main Page

	Logoff
Username:	80080081
First Name:	Pownuk
Last Name:	Andrew
Group:	2020-Fall-MATH-1312-CRN-12219
	Change password
	Change e-mail

Show my grades

Files (notes, syllabus etc.)

Online Learning (List of Students, Data Storage)



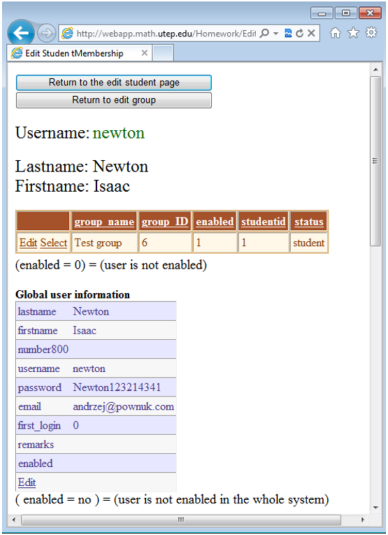
List of students in the group: Test group

	lastname	firstname	username	email	status	enabled	remarks	studentid
Select	Banach	Stefan	banach	andrzej@pownuk.com	student	0		3
Select	Cauchy	Louis	cauchy	andrzej@pownuk.com	student	0		4
Select	Euler	Leonhard	euler	andrzej@pownuk.com	student	1		2
Select	Hilbert	David	hilbert	andrzej@pownuk.com	student	1		6
Select	Leibniz	Gottfried	leibniz	andrzej@pownuk.com	student	1		5
Select	Newton	Isaac	newton	andrzej@pownuk.com	student	1		1
Select	Pownuk	Andrzej	pownuk	andrzej@pownuk.com	student	0		8
Select	Pownuk	Andrzej	testuser	andrzej@pownuk.com	student	1		7

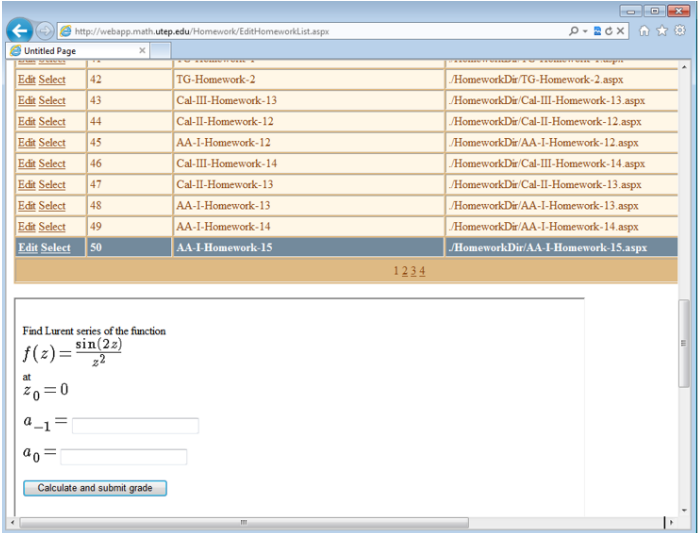
Number of students in this group: 5

List of all students Find student

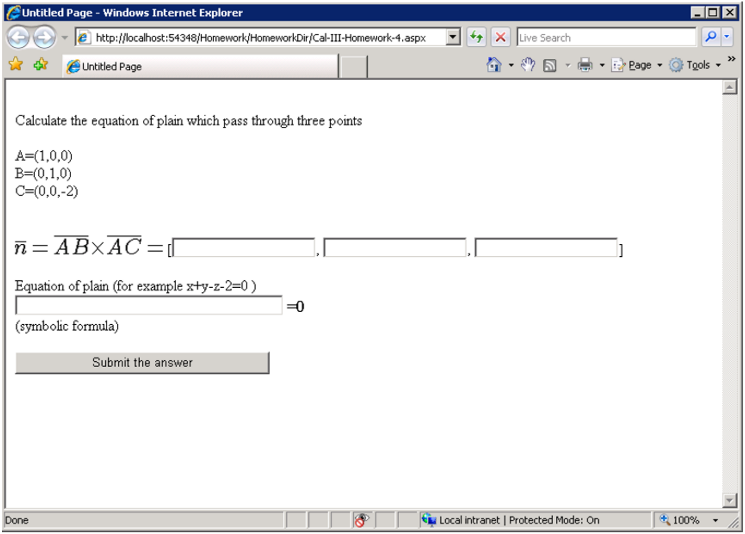
Online Learning (Information about the Student)



Online Learning (Online Homework)



Online Learning (Online Homework)



Online Learning (Online Visualization)

Lx	10
Ly	10
nx	30
ny	30
k	1
q	100
dt	0.05
TInital	10
Tmin	10
Tmax	50

Start calculations

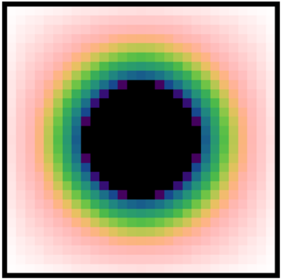


Figure: Solution of the heat transfer equation

Online Learning (Online Visualization)

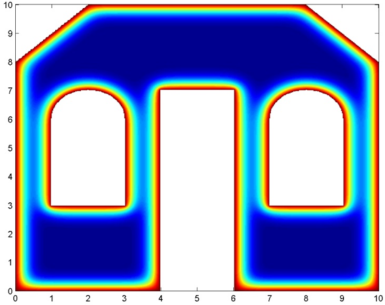
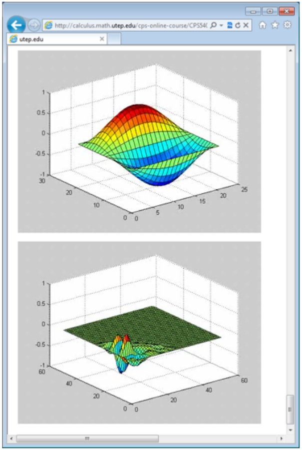


Figure: Solution of the heat transfer equation

$$\lambda\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)+\dot{q}=\rho c_p\frac{\partial T}{\partial t}$$

Online Learning (Online Visualization)



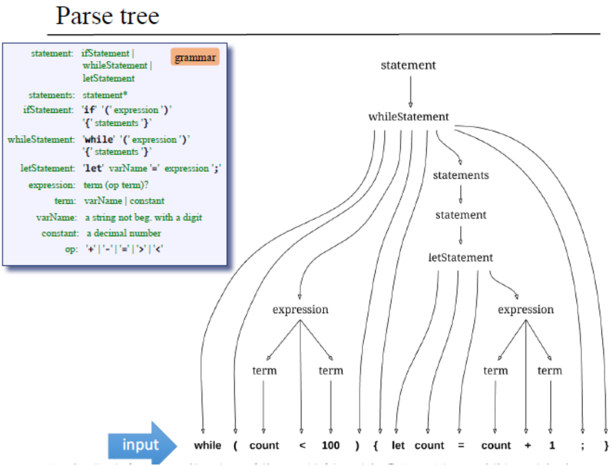
$$\begin{cases} -c\left(\frac{\partial^4 u}{\partial x^4}+\frac{\partial^4 u}{\partial y^4}\right)+q=\frac{\partial^2 u}{\partial t^2}\\ u(x,y)=0, \quad \text{for} \quad (x,y)\in\partial\Omega \\ \frac{\partial^2 u}{\partial x^2}(0,y,t)=\frac{\partial^2 u}{\partial x^2}(L,y,t)=0\\ \frac{\partial^2 u}{\partial y^2}(x,0,t)=\frac{\partial^2 u}{\partial y^2}(x,L,t)=0 \\ u(x,y,0)=u^*(x,y)\end{cases}$$

Figure: Vibration of plates

Sample Problem

- Find area of the parallelogram for $\bar{a} = [1, 2, 3]$, $\bar{b} = [3, 2, 1]$. Answer: $A = |\bar{a} \times \bar{b}| = 4\sqrt{6}$.
- How to input $4\sqrt{6}$ into the system in order to provide the answer?
- It is possible to use text description of the expression. For example:
  - $4 * sqrt(6)$
  - $4 * Sqrt[6]$
  - $4sqrt(6)$
  - $4sqrt6$
  - $4 \cdot sq6$
  - etc.

Parse Tree


Parse tree
statement: ifStatement | whileStatement | letStatement
grammar
statements: statement*
ifStatement: 'if' '(' expression ')' '{' statements '}'
whileStatement: 'while' '(' expression ')' '{' statements '}'
letStatement: 'let' varName '=' expression ';'
expression: term (op term)?
term: varName | constant
varName: a string not beg. with a digit
constant: a decimal number
op: '+' | '-' | '=' | '>' | '<'
statement
whileStatement
statements
statement
letStatement
expression
term
term
expression
term
term
input
while (count < 100) { let count = count + 1 ; }

XML Parse Tree

Parse tree

grammar

```
statement:       ifStatement |
                 whileStatement |
                 letStatement
statements:      statement*
ifStatement:     'if' '(' expression ')'
                 '{' statements '}'
whileStatement:  'while' '(' expression ')'
                 '{' statements '}'
letStatement:    'let' varName '=' expression ';'
expression:      term (op term)?
term:            varName | constant
varName:         a string not beg. with a digit
constant:        a decimal number
op:              '+' | '-' | '=' | '>' | '<'
```

parser output

```
<whileStatement>
  <keyword> while </keyword>
  <symbol> ( </symbol>
  <expression>
    <term>
      <identifier> count </identifier>
    </term>
    <symbol> < </symbol>
    <term>
      <intConstant> 100 </intConstant>
    </term>
  </expression>
  <symbol> ) </symbol>
  <symbol> { </symbol>
  <statements>
    <letStatement>
      <keyword> let </keyword>
      <identifier> count </identifier>
      <symbol> = </symbol>
      <expression>
        <term> <identifier> count </identifier> </term>
        <symbol> + </symbol>
        <term> <intConstant> 1 </intConstant> </term>
      </expression>
      <symbol> ; </symbol>
    </letStatement>
  </statements>
  <symbol> } </symbol>
</whileStatement>
```

Same parse tree, in XML

Grammar

<table>
<tr><td>Lexical elements:</td><td>The Jack language includes five types of terminal elements (tokens):</td></tr>
<tr><td>keyword:</td><td>'class' | 'constructor' | 'function' | 'method' | 'field' | 'static' |
'var' | 'int' | 'char' | 'boolean' | 'void' | 'true' | 'false' | 'null' | 'this' |
'let' | 'do' | 'if' | 'else' | 'while' | 'return'</td></tr>
<tr><td>symbol:</td><td>'{' | '}' | '(' | ')' | '[' | ']' | '.' | ',' | ';' | '+' | '-' | '*' | '/' | '&' | '|' | '<' | '>' | '=' | '~'</td></tr>
<tr><td>integerConstant:</td><td>A decimal number in the range 0 .. 32767.</td></tr>
<tr><td>StringConstant</td><td>'"' A sequence of Unicode characters not including double quote or newline '"'</td></tr>
<tr><td>identifier:</td><td>A sequence of letters, digits, and underscore ('_') not starting with a digit.</td></tr>
<tr><td>Program structure:</td><td>A Jack program is a collection of classes, each appearing in a separate file.
The compilation unit is a class. A class is a sequence of tokens structured according to the following context free syntax:</td></tr>
<tr><td>class:</td><td>'class' className '{' classVarDec* subroutineDec* '}'</td></tr>
<tr><td>classVarDec:</td><td>('static' | 'field') type varName (',' varName)* ';'</td></tr>
<tr><td>type:</td><td>'int' | 'char' | 'boolean' | className</td></tr>
<tr><td>subroutineDec:</td><td>('constructor' | 'function' | 'method') ('void' | type) subroutineName
'(' parameterList ')' subroutineBody</td></tr>
<tr><td>parameterList:</td><td>((type varName) (',' type varName)*)?</td></tr>
<tr><td>subroutineBody:</td><td>'{' varDec* statements '}'</td></tr>
<tr><td>varDec:</td><td>'var' type varName (',' varName)* ';'</td></tr>
<tr><td>className:</td><td>identifier</td></tr>
<tr><td>subroutineName:</td><td>identifier</td></tr>
<tr><td>varName:</td><td>identifier</td></tr>
<tr><td>Statements:</td><td></td></tr>
<tr><td>statements:</td><td>statement*</td></tr>
<tr><td>statement:</td><td>letStatement | ifStatement | whileStatement | doStatement | returnStatement</td></tr>
<tr><td>letStatement:</td><td>'let' varName ('[' expression ']')? '=' expression ';'</td></tr>
<tr><td>ifStatement:</td><td>'if' '(' expression ')' '{' statements '}' ('else' '{' statements '}')?</td></tr>
<tr><td>whileStatement:</td><td>'while' '(' expression ')' '{' statements '}'</td></tr>
<tr><td>doStatement:</td><td>'do' subroutineCall ';'</td></tr>
<tr><td>ReturnStatement</td><td>'return' expression? ';'</td></tr>
<tr><td>Expressions:</td><td></td></tr>
<tr><td>expression:</td><td>term (op term)*</td></tr>
<tr><td>term:</td><td>integerConstant | stringConstant | keywordConstant | varName |
varName '[' expression ']' | subroutineCall | '(' expression ')' | unaryOp term</td></tr>
<tr><td>subroutineCall:</td><td>subroutineName '(' expressionList ')' | (className | varName) '.' subroutineName
'(' expressionList ')'</td></tr>
<tr><td>expressionList:</td><td>(expression (',' expression)*)?</td></tr>
<tr><td>op:</td><td>'+' | '-' | '*' | '/' | '&' | '|' | '<' | '>' | '='</td></tr>
<tr><td>unaryOp:</td><td>'-' | '~'</td></tr>
<tr><td>KeywordConstant:</td><td>'true' | 'false' | 'null' | 'this'</td></tr>
</table>

Infix notation, Prefix notation, Postfix notation

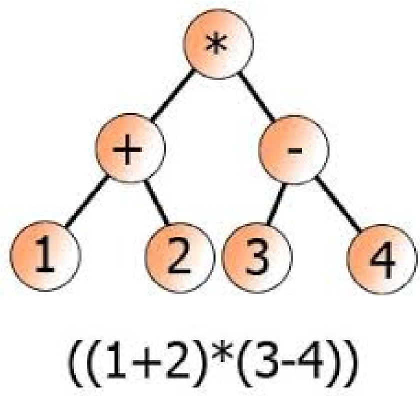
Different notation for arithmetic expressions.

- Infix notation $(5+6)x7$
- Prefix notation $x+567$
- Postfix notation $756+x$

Typical evaluation process of arithmetic expressions.

- InfixToPrefix($(5+6)x7$)=$x+567$
- EvaluatePrefix($x+567$)=77

Expression Tree



How to Evaluate Mathematical Expression Given as a String?

Example

- Expression: sqrt(2)*6
- Value: $\sqrt{2} \cdot 6$

Example

- Expression: sqrt2*6
- Value: $\sqrt{2} \cdot 6$

Example

- Expression: Sqrt[2]6
- Value: $\sqrt{2} \cdot 6$

Example

- Expression: SQRT[2]6
- Value: $\sqrt{2} \cdot 6$

Automated Generation of Homework Assignments

Differentiation, vector algebra, numerical integration, etc.

- Automatically generated list of formulas with given level of difficulty.
- Latex representation of given formula.
- Evaluation of formulas and generation of tests.
- Appropriate HTML code which implements all elements.
- Upload to server and add integrate with the grading system for appropriate group of students, due dates.

Theoretical Aspects of Online Learning

- A. Pownuk, Mathematical aspects of grading student's homework in on-line web applications, Journal of Uncertain Systems, 5(2), 141-153, 2011.

COCONUT Project (Vienna University)

AMPL (A Mathematical Programming Language) is an algebraic modeling language to describe and solve high-complexity problems for large-scale mathematical computing.

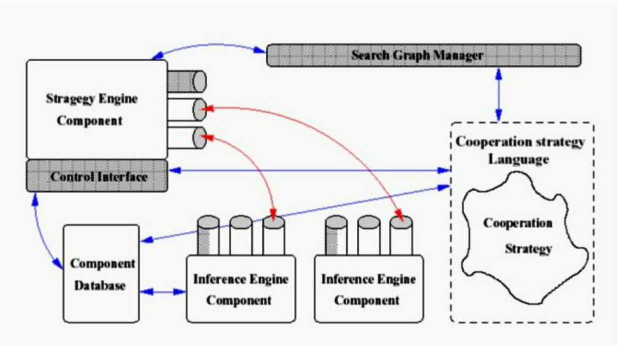
```
#VARIABLE DEFINITIONS
var x_1;
var x_2;
#OBJECTIVE FUNCTION (maximize or minimize)
maximize value: x_1 + 2*x_2;
#CONSTRAINTS
subject to condition_1: x_1   + 3*x_2 <= 20;
subject to condition_2: 3*x_1 +   x_2 <= 20;
subject to condition_3: x_1 >= 0;
subject to condition_4: x_2 >= 0;
```

COCONUT Project (Vienna University)

The optimization problems stored in the work nodes, which are passed to the various inference engines, are kept as directed acyclic graphs (DAG), as well. This representation has big advantages. Hereby, a complete optimization problem is always represented by a single DAG. The vertices of the graph represent operators similar to computational trees. Constants and variables are sources, objective and constraints are sinks of the DAG.

https://www.mat.univie.ac.at/~neum/glopt/coconut/

COCONUT Project (Vienna University)



https://www.mat.univie.ac.at/~neum/glopt/coconut/

COCONUT Project (Vienna University)

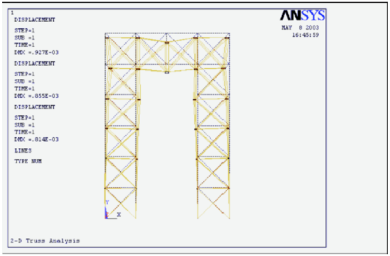


Figure: Modeling of engineering problems with uncertainty

A. Neumaier and A. Pownuk, Linear Systems with Large Uncertainties, with Applications to Truss Structures, Journal of Reliable Computing, 13(2), 149-172, 2007.

SAGA - Scientific Computing with Algebraic and Generative Abstractions

Algebraic software methodologies are a result of the last 20-30 years investigation into abstract data types and algebraic development techniques. The algebraic concepts also abstract modern program structuring mechanisms like classes and generic (or template) modules of object-oriented programming languages such as C++, Generic Java and Fortran-2000.

https://www.ii.uib.no/saga/

- **Sapphire**: For the quick prototyping of mathematical models an **algebraic programming language** and a compiler that translates recursive functions into non-recursive, imperative code was developed. This allows us to code the recursive equations of the mathematical formulation of a solver directly as recursive functions and compile this for both sequential and parallel HPC computers.
- **Sophus**: This is a software library written in C++ and carefully designed to mimic the abstract structure of the PDE mathematics.
- **CodeBoost**: This is a software transformation system being developed to address the gap between well formed code (from a software engineering point of view) and efficient code (from a run-time point of view).

Fuzzy/Interval Calculator

Insert a description of interval and fuzzy expressions and press "Calculate" button. [USER'S MANUAL]

Calculate The result is:

```
 {  [ 0, 6, 14 ]  [ 0.25, 6.25, 13.5 ]  [ 0.5, 6.5, 13 ]  [
0.75, 6.75, 12.5 ]  [ 1, 7, 12 ]  }
```

```
1+2*([0,1]+1)
{[0,1,5], [1,2,3]} + [1,2]*2




######################################################################
#
# The program evaluate the value of expressions
# which contain floating-point, interval and fuzzy numbers.
#
# 1+2*(2+5)                     Floating-point expression.
# [1,2]+[2,3]                   Interval numbers are defined using upper (Xmax) and lower (Xmin) bounds e.g. [Xmin, Xmax].
#
# By default the + operator is assumed between the lines.
# These two lines are equivalent to the expression 1+2*(2+5) + [1,2]+[2,3].
#
# {[0,1,5],   [1,2,3]}          Fuzzy number is a collection of alpha-cuts.
#                               Each alpha-cut is a triple [alpha,Xmin,Xmax]
#{[0,1,5],   [1,2,3]}+[1,2]*2  Composite expression can have floating-point, interval and fuzzy numbers.
#
# It is possible to add an operator between the lines.
#
# 1+2
# *
# [1,2]+[2,3]
#
# These two lines are equivalent to the expression 1+2 * [1,2]+[2,3].
#
# The final result is a value of the expression which is created
# from all the data in the data file.
```

http://www.math.utep.edu/Faculty/ampownuk/php/fuzzy_calculator/

Interval Arithmetic

An binary operation $\star$ on two intervals, such as addition or multiplication, is defined by

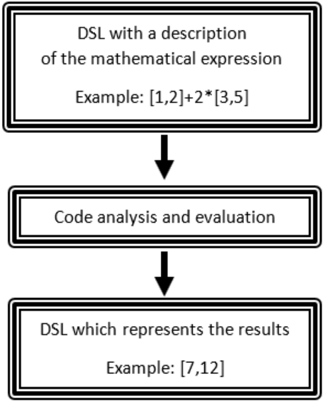
$$[x_1, x_2] \star [y_1, y_2] = \{x \star y \mid x \in [x_1, x_2] \wedge y \in [y_1, y_2]\}$$

- Interval addition $[x_1, x_2] + [y_1, y_2] = [x_1 + y_1, x_2 + y_2]$
- Interval multiplication
  $[x_1, x_2] \star [y_1, y_2] = [z_1, z_2]$ where
  $z_1 = \min\{x_1 \star y_1, x_1 \star y_2, x_2 \star y_1, x_2 \star y_2\}$,
  $z_2 = \max\{x_1 \star y_1, x_1 \star y_2, x_2 \star y_1, x_2 \star y_2\}$.
- Interval division

$$\frac{[x_1, x_2]}{[y_1, y_2]} = [x_1, x_2] \cdot \frac{1}{[y_1, y_2]}$$

.

Fuzzy/Interval Calculator



DSL - Domain Specific Language

Fuzzy/Interval Calculator

DSL for description of fuzzy and interval numbers.

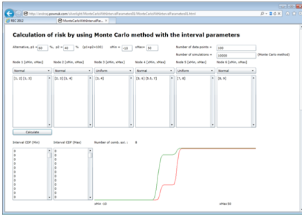
Input:

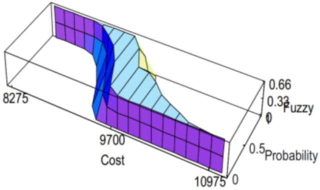
```
1+2*([0,1]+1)
{[0,1,5], [1,2,3]} + [1,2]*2
```

Output:

```
{  [ 0, 6, 14 ]  [ 0.25, 6.25, 13.5 ]
[ 0.5, 6.5, 13 ]  [ 0.75, 6.75, 12.5 ]
[ 1, 7, 12 ]  }
```

Fuzzy Random Variables





Differential Equations with Uncertain Parameters

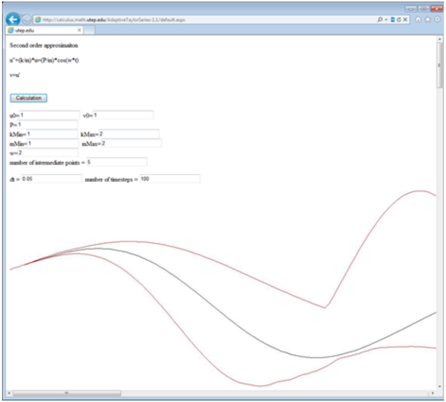


Figure: Second order differential equation

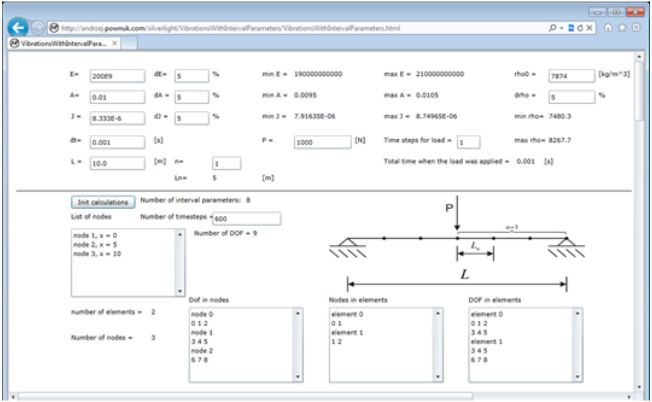


Figure: Input parameters

Online Learning (Numerical Analysis, 1998)

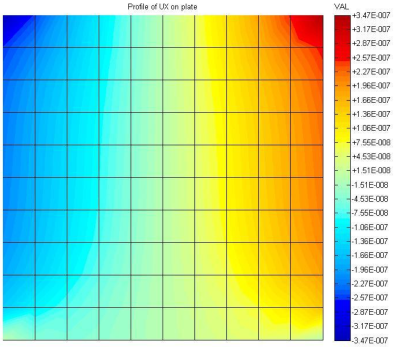
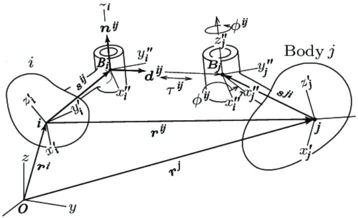


Figure: Web application for teaching of the fininte element method. Description of the problem was given in some DSL.

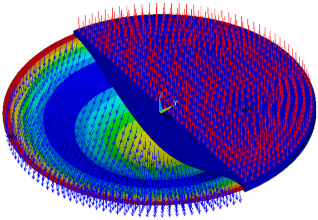
Lagrangian Mechanics



$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_j}\right) = \frac{\partial L}{\partial q_j}$$

Computer methods for finding analytical formulation of the equations of motion in multibody dynamics.

Teaching (Computer Methods in Mechanics)



Computer algebra software (Mathematica, Derive, etc.) for numerical methods in the theory of plates.

$$\frac{2Eh^3}{3(1-\nu)}\left(\frac{\partial^4 w}{\partial x_1^4}+2\frac{\partial^4 w}{\partial x_1^2 \partial x_2^2}+\frac{\partial^4 w}{\partial x_2^4}\right)+q+2\rho h\frac{\partial^2 u}{\partial t^2}=0$$

Teaching (Computer Methods in Mechanics)

Computer algebra software (Mathematica, Derive, etc.) for numerical methods in the theory of linear elasticity.

$$\frac{1}{2(1-\nu)(1-2\nu)}\left(2(1-\nu)\frac{\partial^2 u_x}{\partial x^2}+\frac{\partial^2 u_y}{\partial x \partial y}+ +(1-2\nu)\frac{\partial^2 u_x}{\partial y^2}\right)+b_x=0$$

$$\frac{1}{2(1-\nu)(1-2\nu)}\left(2(1-\nu)\frac{\partial^2 u_y}{\partial y^2}+\frac{\partial^2 u_x}{\partial x \partial y}+ +(1-2\nu)\frac{\partial^2 u_y}{\partial x^2}\right)+b_y=0$$

Teaching (Computer Methods in Mechanics)

- Partial diffrerential equations of elasticity.
- Partial diffrerential equations of plasticity.
- Partial diffrerential equations of viscoelaticity.
- Partial diffrerential equations of the theory of shells and appropriate theory in curvilinear coordinate systems.
- The theory of thin-walled structures.
- Adaptive mesh refinement.
- The theory of variational equations related to the contact mechanics.
- The theory of crack mechanics (fracture mechanics).
- The theory of heat transfer and multiphysics problems.
- etc.

Calculation of the Interval risk by Using Petri Networks and interval Probability

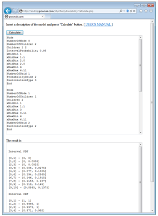


Figure: DSL for description of the engineering problem

M. Betkowski and A. Pownuk, Calculating risk of cost using Monte Carlo simulation with fuzzy parameters in civil engineering, Proceeding of the NSF Workshop on Reliable Engineering Computing, Savannah, Georgia, USA, 179-192, September 15-17, 2004.

Interval Finite Element Method for the Truss Structures

Description of the problem:

```
/PREP7
ET,1,LINK1

N, 1, 0, 0
N, 2, 1, 0
N, 3, 2, 0
N, 4, 3, 0
N, 5, 0, 1
N, 6, 1, 1
N, 7, 2, 1
N, 8, 3, 1
N, 9, 0, 2
N, 10, 1, 2
N, 11, 2, 2
N, 12, 3, 2

MP, EX, 1, 2.1e+11
R, 1, 0.0025
MAT  1
REAL 1
```

Description of interval parameters (help)

```
MP, EX, 1, 5
R, 1, 5
```

◉ Sensitivity analysis method

Calculate

http://www.math.utep.edu/Faculty/ampownuk/php/ansys2interval/ansys-code.php

Results:

```
Time of calculation: 0.004996 [sec]
Number of DOF:       16
Number of elements:  26
Number of nodes:     12

u[  0]= [  2.54206368927894e-05,  2.70758977991233e-05,  2.88890319829459e-05] node=   5 dof= 1
u[  1]= [ -2.41613231842201e-06, -1.45525064589709e-06,  -5.5936232302321e-07] node=   5 dof= 2
u[  2]= [  1.89488493026688e-05,  2.03244670585942e-05,  2.18240888299457e-05] node=   6 dof= 1
u[  3]= [ -1.18336781275183e-05, -1.07203077121679e-05, -9.68801242678198e-06] node=   6 dof= 2
u[  4]= [  1.74375666485017e-05,  1.86853353510165e-05,  2.00368684309889e-05] node=   7 dof= 1
u[  5]= [ -1.53016570105788e-05, -1.40293414211092e-05, -1.28438219917361e-05] node=   7 dof= 2
u[  6]= [  2.23883755090784e-05,  2.38816715072828e-05,  2.55322229372461e-05] node=   8 dof= 1
u[  7]= [ -2.43184098360924e-05, -2.27501214588593e-05, -2.13175562611172e-05] node=   8 dof= 2
u[  8]= [  4.47984203980532e-05,  4.76520021045755e-05,  5.07482294189415e-05] node=   9 dof= 1
u[  9]= [ -1.25873042500698e-05,  -1.0800778851294e-05, -9.13828295995457e-06] node=   9 dof= 2
u[ 10]= [  3.58319463043394e-05,  3.83064738991786e-05,  4.09641151999668e-05] node=  10 dof= 1
u[ 11]= [ -2.03184368590144e-05, -1.88999001199072e-05, -1.75638790058709e-05] node=  10 dof= 2
u[ 12]= [  3.30408793908687e-05,  3.54230615860712e-05,  3.79901925037356e-05] node=  11 dof= 1
u[ 13]= [ -2.87524495626644e-05, -2.70377395621771e-05, -2.54594638852595e-05] node=  11 dof= 2
u[ 14]= [  3.51831538994549e-05,  3.77051247175134e-05,  4.04232862441163e-05] node=  12 dof= 1
u[ 15]= [ -4.18322390326742e-05, -3.95037800489603e-05,  -3.7394527683613e-05] node=  12 dof= 2
```

http://www.math.utep.edu/Faculty/ampownuk/php/ansys2interval/ansys-code.php

ANSYS Parametric Design Language (APDL)

```
MP, EX, 1, 2.1e+11
R, 1, 0.0025
MAT  1
REAL 1
...
```

Extension of the ANSYS Parametric Design Language (APDL) which describes uncertainty of parameters.

```
MP, EX, 1, 5
R, 1, 5
...
```

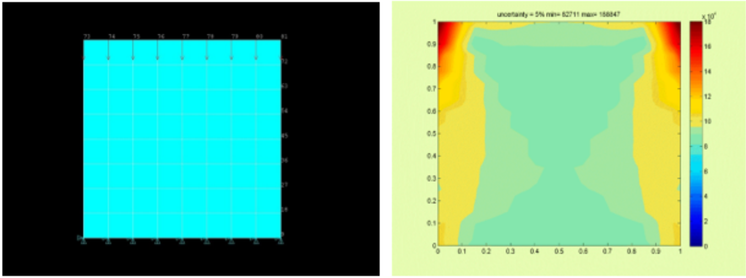
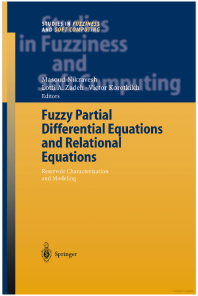


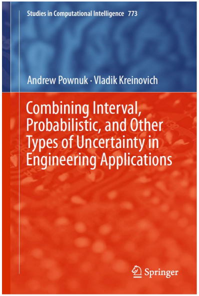
Figure: Solution of system of partial differential equations

Interval Finite Element Method for the Truss Structures



A. Pownuk, Numerical solutions of fuzzy partial differential equation and its application in computational mechanics, In: M. Nikravesh, L. Zadeh and V. Korotkikh, (eds.), Fuzzy Partial Differential Equations and Relational Equations: Reservoir Characterization and Modeling, 308-347, Springer 2004.

Summary of My Research on Modeling of Uncertainty



A. Pownuk and V. Kreinovich, Combining Interval, Probabilistic, and Other Types of Uncertainty in Engineering Applications, Springer 2018.

Chevron Oil Company

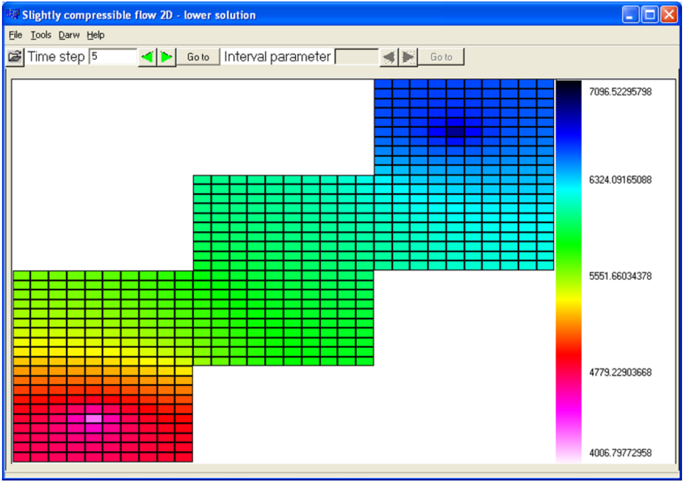
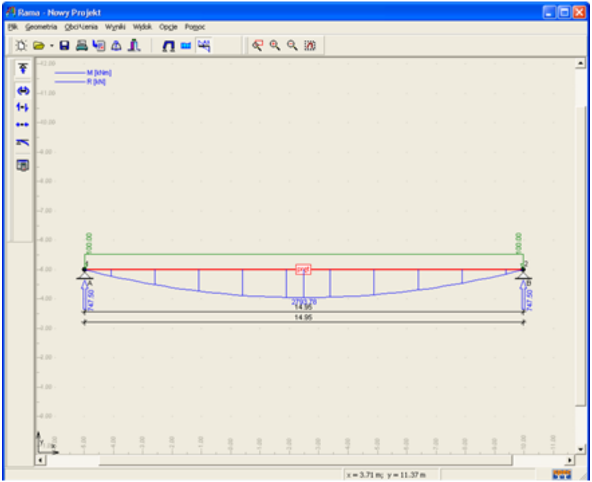


Figure: Research for Chevron Oil Comapny

Commercial FEM Software for Designing Truss and Frame Structures



FEM Equations form APDL

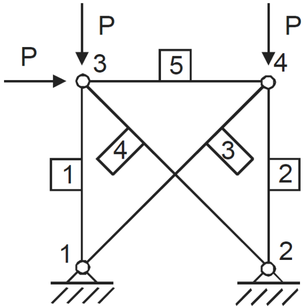


Figure: Sample engineering structure

http://www.math.utep.edu/Faculty/ampownuk/php/fem-equations/fem-equations.php

FEM Equations form APDL


```
N 1 0 0
N 2 1 0
N 3 0 1
N 4 1 1

E 1 1 3 MP 1 R 1
E 2 2 4 MP 2 R 2
E 3 1 4 MP 3 R 3
E 4 2 3 MP 4 R 4
E 5 3 4 MP 5 R 5

BC 1 UX UY
BC 2 UX UY

F 3 FX 1000
F 3 FY -1000
F 4 FY -1000
```

◉ Generate Equations

Calculate

Figure: APDL description of engineering problem

http://www.math.utep.edu/Faculty/ampownuk/php/fem-equations/fem-equations.php

FEM Method

$$\frac{d}{dx}\left(EA\frac{du}{dx}\right)+n=0,\, u(0)=0,\, u(L)=0$$

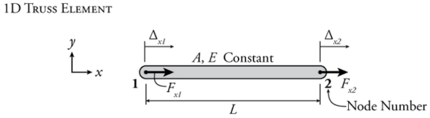
$$\int_0^L \frac{d}{dx}\left(EA\frac{du}{dx}\right) vdx+\int_0^L nvdx=\int_0^L 0vdx,\, u(0)=0,\, u(L)=0$$

$$\int_0^L \frac{d}{dx}\left(EA\frac{du}{dx}\right) vdx=$$

$$=\int_0^L EA\frac{du}{dx}\frac{dv}{dx}dx+EA\frac{du(0)}{dx}v(0)-EA\frac{du(L)}{dx}v(L)$$

etc.

Local Stiffness Matrix



1D element

$$K = \begin{bmatrix} \frac{EA}{L} & -\frac{EA}{L} \\ -\frac{EA}{L} & \frac{EA}{L} \end{bmatrix}$$

2D element

$$K = \begin{bmatrix} \frac{EA}{L} & 0 & -\frac{EA}{L} & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{EA}{L} & 0 & \frac{EA}{L} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

etc.

Global Stiffness Matrix


```
K[  1   1]=  ((E[4]*A[4])/1.414214)*0.500000+(E[5]*A[5])/1.000000
K[  1   2]=  ((E[4]*A[4])/1.414214)*(-0.500000)
K[  1   3]=  ((E[5]*A[5])/1.000000)*(-1.000000)
K[  1   4]=  0.000000

K[  2   1]=  ((E[4]*A[4])/1.414214)*(-0.500000)
K[  2   2]=  (E[1]*A[1])/1.000000+((E[4]*A[4])/1.414214)*0.500000
K[  2   3]=  0.000000
K[  2   4]=  0.000000

K[  3   1]=  ((E[5]*A[5])/1.000000)*(-1.000000)
K[  3   2]=  0.000000
K[  3   3]=  ((E[3]*A[3])/1.414214)*0.500000+(E[5]*A[5])/1.000000
K[  3   4]=  ((E[3]*A[3])/1.414214)*0.500000

K[  4   1]=  0.000000
K[  4   2]=  0.000000
K[  4   3]=  ((E[3]*A[3])/1.414214)*0.500000
K[  4   4]=  (E[2]*A[2])/1.000000+((E[3]*A[3])/1.414214)*0.500000
```

How to Efficiently use Available Tools?

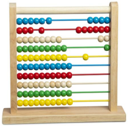
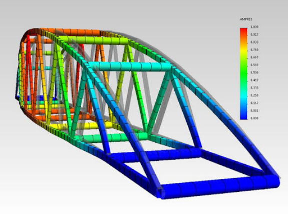


Figure: Tools in the past

Figure: Tools now

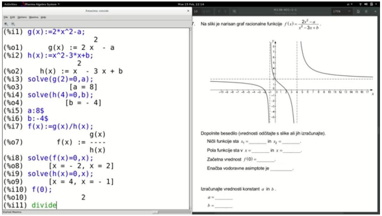
Advantages of the Automated Computational Methods



Automation of some part of the engineering computational process:

- faster design,
- more optimal products,
- cheaper engineering structures.

Advantages of the Automated Computational Methods



Some part of the the computational algorithms can be automated. There are several benefites of this process:

- calculations are faster,
- it is possible to analysie more results,
- it is possible to solve some problems with high complexity.

Conclusions

- Syntax and grammar analysis of the mathematical statements can improve online learning systems.
- Some optimization problems and some aspects of theory of partial differential equations can be can be solved automatically by using special software methodologies.
- Automated development of mathematical models speeds up calculations and and software development.

Thank You