A domain-specific embedded language called ibvp was developed to model initial-boundary value problems for partial differential equations. As a (motivating) application, we developed tools to parse this language and generate problem-specific input to the Proteus toolkit developed at ERDC.

### Subject Terms
- Numerical analysis
- Partial differential equations
- Domain-specific language

---

**Abstract**

A domain-specific embedded language called ibvp was developed to model initial-boundary value problems for partial differential equations. As a (motivating) application, we developed tools to parse this language and generate problem-specific input to the Proteus toolkit developed at ERDC.
Final Report: Computer-aided transformation of PDE models: languages, representations, and a calculus of operations

ABSTRACT
A domain-specific embedded language called ibvp was developed to model initial-boundary value problems for partial differential equations. As a (motivating) application, we developed tools to parse this language and generate problem-specific input to the Proteus toolkit developed at ERDC.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

<table>
<thead>
<tr>
<th>Received</th>
<th>Paper</th>
</tr>
</thead>
</table>

TOTAL: 

<table>
<thead>
<tr>
<th>Number of Papers published in peer-reviewed journals:</th>
</tr>
</thead>
</table>

(b) Papers published in non-peer-reviewed journals (N/A for none)

<table>
<thead>
<tr>
<th>Received</th>
<th>Paper</th>
</tr>
</thead>
</table>

TOTAL: 

<table>
<thead>
<tr>
<th>Number of Papers published in non peer-reviewed journals:</th>
</tr>
</thead>
</table>

(c) Presentations
Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

<table>
<thead>
<tr>
<th>Received</th>
<th>Paper</th>
</tr>
</thead>
</table>

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

<table>
<thead>
<tr>
<th>Received</th>
<th>Paper</th>
</tr>
</thead>
</table>

TOTAL:

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

<table>
<thead>
<tr>
<th>Received</th>
<th>Paper</th>
</tr>
</thead>
</table>

TOTAL:
Number of Manuscripts:

<table>
<thead>
<tr>
<th>Books</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Received</td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td></td>
</tr>
</tbody>
</table>

| Received       |   |
| Book Chapter   |   |
| TOTAL:         |   |

Patents Submitted

Patents Awarded

Awards

Graduate Students

<table>
<thead>
<tr>
<th>NAME</th>
<th>PERCENT_SUPPORTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTE Equivalent:</td>
<td></td>
</tr>
<tr>
<td>Total Number:</td>
<td></td>
</tr>
</tbody>
</table>

Names of Post Doctorates

<table>
<thead>
<tr>
<th>NAME</th>
<th>PERCENT_SUPPORTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTE Equivalent:</td>
<td></td>
</tr>
<tr>
<td>Total Number:</td>
<td></td>
</tr>
</tbody>
</table>
Names of Faculty Supported

<table>
<thead>
<tr>
<th>NAME</th>
<th>PERCENT_SUPPORTED</th>
<th>National Academy Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robert C. Kirby</td>
<td>0.23</td>
<td></td>
</tr>
<tr>
<td>FTE Equivalent:</td>
<td>0.23</td>
<td></td>
</tr>
<tr>
<td>Total Number:</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Names of Under Graduate students supported

<table>
<thead>
<tr>
<th>NAME</th>
<th>PERCENT_SUPPORTED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FTE Equivalent:</td>
<td></td>
</tr>
<tr>
<td>Total Number:</td>
<td></td>
</tr>
</tbody>
</table>

Student Metrics
This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period: ...... 0.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields: ...... 0.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields: ...... 0.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale): ...... 0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering: ...... 0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense: ...... 0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields: ...... 0.00

Names of Personnel receiving masters degrees

<table>
<thead>
<tr>
<th>NAME</th>
<th>Total Number:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Names of personnel receiving PHDs

<table>
<thead>
<tr>
<th>NAME</th>
<th>Total Number:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Names of other research staff

<table>
<thead>
<tr>
<th>NAME</th>
<th>PERCENT_SUPPORTED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FTE Equivalent:</td>
<td></td>
</tr>
<tr>
<td>Total Number:</td>
<td></td>
</tr>
</tbody>
</table>
Sub Contractors (DD882)

1 a. University of Illinois-Urbana Champagne

Sub Contractor Numbers (c):
Patent Clause Number (d-1):
Patent Date (d-2):
Work Description (e): Collaborate on the production of ibvp. In particular, the subcontract PI provided expertise in abstract syntax and symbolic manipulation in order to produce our domain-specific language.
Sub Contract Award Date (f-1): 12/18/14 12:00AM
Sub Contract Est Completion Date(f-2): 6/21/15 12:00AM

1 b.

Inventions (DD882)

Scientific Progress

Technology Transfer

In collaboration/coordinate with Dr. Chris Kees in the Coastal Hydraulics Laboratory at ERDC, we developed an open-source software project called ibvp.

It is freely available for download on github at:

https://github.com/ibvp/ibvp
Computer-aided transformation of PDE models: languages, representations, and a calculus of operations

1 Vision and background

Physical and engineered systems described by partial differential equations (PDE) frequently admit a range of models of varying complexity and fidelity. Additional features come through the adding or modifying particular terms in the equations. Each such model in term could be subject to any one of a number of reasonable discretizations. We maintain that what is clear at the mathematical level should be equally clear in computation.

In this small STIR project, we separate the concerns of describing and discretizing such models by defining an input language representing PDE, including steady-state and transient, linear and nonlinear, and so on. We then provide transformations acting on these to determine structural information and/or convert the resulting abstract syntax into another format, such as the input format of a solver code. We aim not to implement a new PDE solver but instead to extend existing software projects by providing a smarter interface to them.

Our fundamental contribution reverses the traditional process of scientific computing; instead of computing the solution to a PDE to model a scientific process, we now model a PDE itself as a computational object. Initial efforts at automating PDE solvers, such as [8, 9], focused on the solvers themselves and particular families of discretizations (e.g. finite elements), and now it is natural to complement their rapid growth and increasing maturity by additional layers of automation, as begun here.

Unlike in linear algebra libraries such as [1, 2], providing a traditional library for PDE would require enumerate every possible (combination of) equations of interest, a combinatorially large task. On the other hand, one can model a rather large range of equations with a small number of atoms (e.g. fields and differential operators) combined under grammatical rules – differential operators acting on fields and added to or multiplied by other fields algebraically. Successful domain-specific languages for PDE [8, 9] adopt such a grammatical approach rather than merely a menu of pre-implemented differential equations. Still, such projects present a language fundamentally limited only to finite element methods.

Our work then, builds on this development in numerical computing by putting forward a further idea. While weak forms are naturally associated with finite element rather than discretizations, we have developed a linguistic model for the natural strong form of PDE, providing a common language that can be shared not only across disciplines, but also between different approaches to discretization. Importantly, our approach does not discard the developments of previous decades, but provides an opportunity to build upon them in important ways.

2 ibvp

Our work has led to the project ibvp (freely available under the MIT license at https://github.com/ibvp). This code builds on subcontractor Klöckner’s well-known pymbolic
The package, which is a Python library for symbolic manipulation. Rather than a full-fledged computer algebra system like sympy \cite{5}, pymbolic aims for extensibility into domain-specific languages. In building ibvp, we have extended pymbolic’s set of primitives to include things such as fields, differential operators (as tree nodes rather than actual differentiation), normals, and boundary conditions and its mappers to work on the new nodes. This is best presented through an example. Using ibvp, we are able to define the two-dimensional viscous Burgers’ equation

\[
  u_t + \nabla \cdot (\beta u^2) - \Delta u = 0,
\]

where \(\beta = (1, 2)^t\) by the Python code:

```python
import ibvp.sym as sym
ambient_dim = 2

u = sym.Field("u")

vec = np.array([1.0, 2.0])

eqns = sym.join(
    sym.d_dt(u)
    + sym.div(vec * u**2)
    - sym.div(sym.grad(u))
)
```

This creates an abstract syntax tree that we can manipulate in any desired way. For this project, we have concentrated on generating input for the Proteus toolkit \cite{6} developed at ERDC. Proteus itself provides a discretization-neutral strong-form interface, allowing users to encode the coefficients and functional forms for a certain general class of convection-diffusion-reaction systems. We developed code to parse an ibvp-defined PDE, match the terms present against Proteus’ canonical form, and generate code for its Python interface. For example, our code above gives rise to the Proteus code:

```python
from proteus.TransportCoefficients import TC_base

class Burgers(TC_base):
    def __init__ (self):
        mass = {0: {0: 'linear'}}
        advection = {0: {0: 'nonlinear'}}
        diffusion = {0: {0: 'constant'}}
        potential = {0: {0: 'u'}}
        reaction ={}
        hamiltonian ={}
        variableNames=['u']
        TC_base.__init__(self,
                nc=1,
                mass=mass,
                advection=advection,
                diffusion=diffusion,
                potential=potential,
```
reaction = reaction,
hamiltonian = hamiltonian,
variableNames = variableNames)

def evaluate(self, t, c):
    u = c[('u', 0)]
    c[('m', 0)[3]] = u
    c[('dm', 0, 0)[3]] = 1
    c[('f', 0)[..., 0]] = u**2
    c[('f', 0)[..., 1]] = 2.0*u**2
    c[('df', 0, 0)[..., 0]] = 2*u
    c[('df', 0, 0)[..., 1]] = 2.0*2*u
    c[('a', 0, 0)[..., 0, 0]] = 1
    c[('a', 0, 0)[..., 1, 1]] = 1

This provides a Python-level implementation for filling the appropriate coefficient arrays, as well as meta-data indicating the kinds of dependencies. For example, note that the mass matrix term is marked as ‘linear’, while the advection term is marked as ‘nonlinear’. This information is determined automatically by `ibvp` from the PDE.

Proteus also provides a typically more efficient C-level interface for filling the coefficients. Our code generator could readily be extended to write to this interface as well, replacing the `numpy`-based Python code with loops to fill the arrays. Additionally, although it would require substantial internal modifications to Proteus, we could generate code suitable for GPUs or other accelerators.

We have also provided a very basic interface for specifying boundary conditions, typically of Dirichlet or flux types, and generating appropriate Proteus-level code for them. In the future, we hope to enrich this feature to include descriptions of nonlinear boundary conditions, such as when the type of boundary condition enforced depends on the system state.

### 3 Future perspective and potential impacts

We believe our approach will bring several benefits as the work continues. This kind of interface should be immediately usable to anyone who has worked with PDE models mathematically and who has a minimal level of familiarity with computer algebra software.

A unified description of PDE models allows sharing between different numerical codes supported by our transformations. This in turn facilities sharing of test, validation, and benchmarking examples between different codes. This is particularly valuable as, in the past, large amounts of effort have been spent assembling batteries of such test examples (e.g. [11]) used for verification and validation. Our research could help introduce automation into a process that previously required developers to manually re-code and re-discretize each example. This would require mappings from our description language into other PDE codes, a topic that is certainly of interest to us.

This work represents a new, multi-layered approach to domain-specific languages for simulation. It is known that well-crafted tools greatly improve scientific productivity by reducing development time and providing a medium for expression and interchange of ideas.
For example, numerical linear algebra is greatly simplified by high-level languages such as MATLAB. The numerical solution of PDEs lies, conceptually, at an even higher level of abstraction. As a result, domain-specific languages stand to have even greater benefit if the right abstraction is found.

3.1 Future potential projects

It is the goal of a STIR project to stimulate future endeavors. The presence of a strong-form PDE language provides a natural starting point for many potentially impactful future projects.

3.1.1 Another canonical form: PyCLAW

Hyperbolic conservation laws, which are PDE of the form

$$u_t + \nabla \cdot F(u) = f(x, t, u),$$

together with initial and boundary conditions, model a large class of important phenomena including weather, climate, acoustics, and electromagnetics. Despite the varied applications, similarities among the particular equations admit general families of methods, and this similarity allows the development of general purpose finite volume solvers such as Clawpack [7]. Originally released in the 1990’s, Clawpack has been extended to support adaptive mesh refinement. It also now supports large-scale distributed-memory parallelism behind a high-level Python interface, called PyCLAW. It has been widely used in applications. Past ERDC researcher Aron Ahmadia contributed heavily to the parallelization efforts.

By specifying the flux and source functions $F$ and $f$, together with initial and boundary condition, one neatly specifies a problem. A natural application of ibvp would be to match a canonical form for conservation laws and generate appropriate modules for use with Clawpack. Working with the PyCLAW interface, we envision a just-in-time compiler that maps from ibvp to massively parallel simulations.

3.1.2 Generation of weak forms

Finite element domain-specific languages such as FEniCS require practitioners, who may not be trained in variational methods, to learn a new approach to formulating and expressing PDE. Further, there is not a unique mapping from strong to weak form, given the wide wide variety of mixed, discontinuous, stabilized, and other methods. This learning curve creates opportunities for user error. Starting with ibvp, we envision automating the derivation of weak forms by transformations of strong forms according to particular “recipes”. It is fairly mechanical to obtain classic Galerkin methods – multiply by test functions and integrate by parts. More delicately, one can perform this task elementwise to arrive at discontinuous Galerkin formulations [4], perhaps parameterized over numerical flux functions. Alternatively, one can encapsulate families of stabilized methods such as SUPG [3] by providing transformations that add appropriate terms systematically in the presence of advection.
3.1.3 Additional Model Fidelity

To leverage corresponding advanced solver capabilities in current and future PDE technology, and assuming success of this prototyping effort, we anticipate that there will be significant Army interest in incorporating higher-fidelity, more detailed PDE models. These might include models with at-rest or moving interfaces, multiple domains, multiple different physics, level sets, as well as phase field and tracer particle models. In addition to augmenting the input language, these features will enable new transformations and modes of reasoning.

Additional potential for fruitful research exists in the concise representation of efficiently discretizable geometry, such by signed distance functions [10].
References


