Teaching Portfolio

Bruno Abreu Calfa

January 29, 2015

Contents

Teaching Responsibilities 1
Teaching Philosophy 2
Sample Teaching Materials 4
Honors and Awards 4
Appendix A Course Syllabus: Numerical Methods in Engineering 5
Appendix B MATLAB Tutorial: Fall 2011 9
Appendix C Guest Lectures on Climate Change: Fall 2013 18
Appendix D Teaching Assistant Award Text: Academic Year 2011-2012 36

Teaching Responsibilities

As a Ph. D. student in the Department of Chemical Engineering at Carnegie Mellon University (CMU), I served as a teaching assistant (TA) in almost every semester of the program. I officially and unofficially assisted students from freshman to senior years. This was largely because I was the department’s “Math Software TA” for three times. This TA position enabled me to provide help to undergraduate (and sometimes Master’s) students with regards to software packages, such as MATLAB, Microsoft Excel, AspenPlus and others. This experience was pedagogically very enriching to me as it allowed me to interact with students from different years and, thus, understand better the most common hurdles they faced across multiple chemical engineering courses.

Table 1 summarizes the courses I served as a TA during my Ph.D. program. The column “Additional Material” lists the materials and activities I prepared and executed beyond the basic TA requirements, such as grading homework and holding office hours. All materials can be retrieved from my personal website: http://bacalfa.com/TA/ExtraMaterial/TAEextraMaterial.html. The appendices contain the learning materials I prepared for some of the courses listed below.
Table 1: Courses I assisted and brief description of additional teaching material I prepared as a Ph.D. student at CMU.

<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
<th>Number</th>
<th>Additional Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall 2010</td>
<td>Chemical Process Systems Design</td>
<td>06-421</td>
<td>Handout on economic evaluation</td>
</tr>
<tr>
<td>Spring 2011</td>
<td>Chemical Engineering Process Control</td>
<td>06-362</td>
<td>Short MATLAB and Simulink tutorials</td>
</tr>
<tr>
<td>Fall 2011</td>
<td>Math Software TA</td>
<td>–</td>
<td>Extended MATLAB tutorials</td>
</tr>
<tr>
<td>Spring 2012</td>
<td>Math Software TA</td>
<td>–</td>
<td>Microsoft Excel and VBA tutorials</td>
</tr>
<tr>
<td>Spring 2013</td>
<td>Math Software TA</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Fall 2013</td>
<td>Introduction to Chemical Engineering</td>
<td>06-100</td>
<td>Guest lecture on climate change</td>
</tr>
<tr>
<td>Spring 2014</td>
<td>Mathematical Methods of Chemical Engineering</td>
<td>06-262</td>
<td>Lecture on numerical methods for higher-order ODEs, review lecture</td>
</tr>
<tr>
<td>Fall 2014</td>
<td>Chemical Engineering Thermodynamics</td>
<td>06-321</td>
<td>Tutorial and homework on Aspen Plus</td>
</tr>
</tbody>
</table>

The unofficial assistance was due to the fact that, even though not being the official TA for a given course in a given semester, students in different courses and whom I had assisted previously contacted me with computing-related questions.

Teaching Philosophy

I strongly believe that the main role of an educator is to leverage students for their own success. An educator is a facilitator of learning, i.e., a person that helps students understand easy and difficult fundamental concepts that can be ultimately used to impact the world. Every discipline teems with important topics to be covered. However, to cope with the limited time in an academic term, I prepare teaching materials and lectures by carefully selecting relevant and indispensable topics that give the students the necessary basis to learn more challenging topics on their own in the future.

The different teaching styles and effectiveness among my professors have been instrumental in shaping my views on education. They form the basis of my teaching philosophy, which is to prepare students for the real world by taking two concrete actions:

1. **Effectively teach the fundamentals so that students can successfully use them to acquire more advanced knowledge in the future (life-long learning), and**
2. assess the students’ progress by assigning homework and projects that require both individual and group work plus in-class presentations (communication skills).

When I teach, I experience personal joy and satisfaction from helping someone understand something I also understand. In addition to that main driver, I have realized that teaching is also a learning experience for me since each student has an individual approach to processing the material, which requires flexibility from the instructor’s part.

From a practical viewpoint, I believe that educators equip students with fundamental, yet incomplete knowledge about the world, and that constitutes their toolbox. From personal experience, students in general may not have the necessary maturity to comprehend this fact until after they graduate. Once they graduate, they will be able to use the tools and skills from their toolbox to help them tackle real-world problems. A key aspect to this toolbox is that students will know where to look for information when confronted with problems, and motivate them to keep always learning. For instance, when teaching concepts in Thermodynamics, such as fugacity and activity coefficients, I find it very useful for the students to not only present the mathematical definitions from a textbook, but I also make them aware of other sources of a different nature, such as handbooks and Internet sites. In addition, I also make use of practical examples related to chemical engineering applications where the concepts arise. This also gives me the opportunity to demonstrate how software tools (both general-purpose, such as MATLAB and Excel, and specialized process simulators) can be used to perform the required calculations in a professional setting.

In order to provide students with the tools and opportunities to hone their skills during classroom encounters, I have adopted the following student evaluation strategy. Students practice the concepts learned in class through homework assignments, and after enough content has been covered during the term I assign a group project. Students are expected to work in teams, divide up the tasks, and then present their work in front of the class, and possibly to an external audience, including professors and practitioners. All aspects of the group project bring students closer to real-world situations in which they will have to make decisions as a team to achieve a common goal. For example, the project guidelines I developed for a sample course syllabus (see Appendix A) concern flash calculations. The guidelines suggest three steps to approach the problem: (1) data collection, (2) modeling and computer implementation, and (3) report writing and oral presentation. In all those steps, students are expected to divide up the tasks and at the same time be familiar with the overall work.

With regards to evaluating my teaching effectiveness, I ask students to do two feedback evaluations, one in the middle and another one at the end of the term. The advantage of having two evaluations during the term is illustrated with the following actual personal experience. The professor in the course 06-262 (see Table 1) did a mid-term evaluation, and the students requested for more examples and MATLAB practice in class. As a teaching assistant and having to give a guest lecture a few days after the mid-term evaluation, I prepared two numerical exercises in addition to the lecture notes I had to cover provided by the professor. I gave the students five minutes to solve one of the exercises in class and on their own, and I demonstrated how to use MATLAB to solve the other exercise. Another self-evaluation tactic I use in every class is gauging the students’ understanding of the material by reviewing in the beginning of a class what was discussed in the previous class, and asking
questions about what I just covered to check if students successfully follow the lecture.

As an educator, I will prepare students to tackle real-world problems by making use of assignments, such as homework and projects, that require both individual and team effort to achieve satisfactory results in communicating their work. Moreover, I would like to be able to take students on technical field trips. I find it to be very pedagogically beneficial for the students to see the concepts covered in class being applied in practice. They may inspire students even more about Chemical Engineering and, perhaps, significantly influence their professional careers. Lastly, I am interested in teaching all the courses I have served as a teaching assistant plus undergraduate special-topics or graduate-level courses related to computer programming and mathematical optimization.

Sample Teaching Materials

This portfolio contains a sample of the following teaching materials:

- Course Syllabus (Appendix A). Sophomore-level course entitled “Numerical Methods in Engineering”.
- MATLAB Tutorial (Appendix B). Two-part introductory tutorial for undergraduate students in all years.
- Climate Change Guest Lectures (Appendix C). Two-part lecture materials taught in a freshman class.

Honors and Awards

I was awarded with the Mark Dennis Karl Outstanding Graduate Teaching Award in the year of 2012. This award is given by the Department of Chemical Engineering at CMU to the student who was an outstanding TA whose commitment to education. Each year, this award is given to a student judged by the faculty to have done an outstanding job as a teaching assistant. The complete award text prepared by the Department Head is available in Appendix D.
Appendix A  Course Syllabus: Numerical Methods in Engineering

This appendix contains the syllabus I designed for a course I am interested in teaching. My intention is not only to teach students the fundamentals of numerical methods that are instrumental to solve real-world problems, but also help them become familiar with programming in both MATLAB and Microsoft Excel with Visual Basic for Applications (VBA). I place considerable weight on homework assignments (60% of the total grade), since it is my goal that students prepare their solutions as if they were writing a report. The remainder of the total grade is attributed to a group project in which students are required to work in teams. This assignment will evaluate their collaboration, report writing, and presentation skills. I also strongly encourage students to type all their assignments solutions in order to practice writing technical reports (i.e., formatting, data and results presentation).

In addition to the course syllabus, I also prepared a sample homework problem with solution (that exemplifies the report-like structure of what is required by students) and the guidelines of a group project, which can be made available upon request.
XX-2XX: Numerical Methods in Engineering

Spring 20XX

Course Units: 12
Lectures: Tue/Thu 9:00 AM – 10:50 AM
Location: DH 100
Instructor: Bruno A. Calfa
E-mail: bacalfa@xxx.edu
Office: DH 123
Office Hours: Mon 2:00 PM – 4:00 PM

Teaching Assistants:
John Doe djohn@xxx.edu DH 200 Tue 1:00 PM – 3:00 PM
Mary Kay kmary@xxx.edu DH 200 Fri 10:00 AM – 12:00 PM

1 Course Description

Numerical methods are extremely powerful problem-solving tools. They are capable of handling large systems of equations, nonlinearities, and complicated geometries that are not uncommon in engineering practice and that are often impossible to solve analytically. The practical use of numerical methods has been enabled by advances in computing since their algorithms are suitable to be implemented as computer programs. This course covers popular numerical methods algorithms in the following class of problems: linear algebra, root-finding, mathematical optimization, curve-fitting, numerical integration and differentiation, and ordinary and partial differential equations.

2 Course Objectives

Numerical methods can be taught with at least three different goals. Mathematicians would emphasize rigorous convergence proofs of the algorithms, and computer scientists would be particularly interested in designing and analyzing efficient algorithms. However, the goal of this course is to present the main concepts behind the most popular algorithms for each class of problems and show how to use two computer packages—MATLAB and Microsoft Excel with Visual Basic for Applications (VBA)—to solve problems with applications in chemical engineering. Therefore, computer programming is a central part of the course.

By the end of the semester, you should be able to:

• Describe the main concepts concerning the methods discussed in the course and demonstrate their applicability in the assignments by using an appropriate method to solve a given class of problem;

• Utilize MATLAB’s and Microsoft Excel’s built-in functions and develop your own procedures to solve the types of problems covered in the course and beyond;

• Recognize and try to mitigate the sources of errors due to computer arithmetic operations;

• Identify a numerical routine suitable for a given class of problem.

3 Materials

The required textbook for the course is:


The book contains more topics than we will be able to cover in the course. It contains example codes for both MATLAB and VBA, which makes it a good reference for you in future courses. I will post lecture notes to the course website before class so that you can print them out or download them to your tablet or computer and bring them to class.

Other optional relevant sources are (in no particular order):


4 Prerequisites

The basic prerequisites are:

• Calculus (differentiation, integration, Taylor series expansion, differential equations)

• Linear algebra (matrix notation)

• Computer programming (any experience with programming languages will be helpful)

• Basic principles and calculations in chemical engineering (material and energy balances, thermodynamics)

Even though I assume prior knowledge of the prerequisites above, I will briefly introduce some major relevant concepts when necessary. In the case of programming, two (optional) brief tutorials on MATLAB and Excel with VBA will be offered in the first weeks of class.
5 Evaluation and Grading Policy

The two types of evaluation and their worth are:

- **Homework – 60%**
  - Assigned weekly or biweekly
  - Due at the beginning of class
  - Late turn-ins incur penalty of 50%, i.e. your highest grade will be 50%, unless prior notice to the instructor
  - Typed solutions are preferred, but hand-written solutions will also be accepted, in which legibility and organization will strongly influence the grading
  - Your homework solutions should **not** contain only codes, numbers, and calculations. Please explain the steps you take to solve the problems by writing some text around the calculations. Moreover, make sure you highlight the final answer obtained. Ideally, the homework solutions to each problem should resemble a brief report on how you solved the particular problem. You may attach your codes to your document, but make sure to reference and label attachments accordingly.
  - Solutions will be posted on the same day when the homework is due

- **Project – 40%**
  - Assigned in the last weeks of the course
  - Students will work in groups on a problem longer than the ones in the homework
  - Groups will turn in a type-written report and present their work to the class
  - Same penalty as for homework assignments applies for late turn-ins, i.e. your highest grade will be 50%, unless prior notice to the instructor
  - More specific details will be given later during the course

I highly encourage students to type their homework solutions. An excellent way of producing organized, neat, and professional typed documents with equations, mathematical symbols, code snippets, etc. is with \LaTeX, a document markup language. You can find a myriad of resources online, but I would start here: [http://www.latex-project.org/](http://www.latex-project.org/). It is well known that its learning curve is steep, but I will be glad to assist students who are interested in using \LaTeX to prepare their documents. Any word processor can be used as an alternative. MATLAB has a "publishing" feature that generates a report of your code if it is written in a particular way. This will be accepted provided that the published code contains text explaining the steps taken.

6 Course Policies and Expectations

Classes start on time and in every beginning of class I will summarize the main topics of the previous class. Even though attendance is not required, I highly encourage you to attend classes and participate by asking and answering questions. Eating snacks is allowed as long as you do not disturb the professor and your classmates.

Students are allowed to discuss homework and project problems with classmates; however, each student or group must work on their individual solutions and turn in their own documents (homework solutions and project reports). Plagiarism will not be accepted and the assignments of the students involved will receive a zero grade.

If you have a conflict between a religious holiday and a class meeting or assignment, please contact me in advance to make alternate arrangements.

If you have learning needs that require some adaptations for you to succeed in the course, please contact Equal Opportunity Services on campus ([http://hr.web.cmu.edu/dsrg/students.htm](http://hr.web.cmu.edu/dsrg/students.htm)). We can arrange to accommodate your learning style based on EOS recommendations. Please notify me at the semester’s beginning of your learning needs. Do not wait until the semester’s end to seek support.

7 Tentative Course Calendar

<table>
<thead>
<tr>
<th>Date</th>
<th>Topics</th>
<th>Textbook Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>Introductions, syllabus, motivating examples</td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>Approximations and round-off errors, HW 1 (assgn)</td>
<td>Chapter 3, Sources of Errors</td>
</tr>
<tr>
<td>Day 3</td>
<td>Truncation and total numerical errors</td>
<td>4.1, 4.3</td>
</tr>
<tr>
<td>Day 4</td>
<td>Matrix notation, Gaussian Elimination, HW 1 (due)</td>
<td>PT3.2.1, PT3.2.3, 9.2</td>
</tr>
<tr>
<td>Day 5</td>
<td>Gaussian Elimination with Pivoting, LU Decomposition, HW 2 (assgn)</td>
<td>9.4.2, 10.1</td>
</tr>
<tr>
<td>Day 6</td>
<td>Gauss-Seidel method, Applications</td>
<td>11.2</td>
</tr>
<tr>
<td>Day 7</td>
<td>Bracketing methods (bisection, false position), HW 2 (due)</td>
<td>5.2, 5.3</td>
</tr>
<tr>
<td>Day 8</td>
<td>Open methods (fixed-point, Newton-Raphson), HW 3 (assgn)</td>
<td>6.1, 6.2</td>
</tr>
<tr>
<td>Day 9</td>
<td>Systems of nonlinear equations, Applications</td>
<td>6.6</td>
</tr>
</tbody>
</table>

**Mathematical Optimization**

| Day 10 | Background, 1-D unconstrained optimization, HW 3 (due) | PT4.2.13.3 |
| Day 11 | N-D unconstrained optimization | 14.2 |
| Day 12 | Constrained optimization | Chapter 15 |
| Day 13 | Applications, Project (assgn) | |

**Least-Squares Regression**
<table>
<thead>
<tr>
<th>Day</th>
<th>Topic</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Linear regression, <em>HW 4 (assgn)</em></td>
<td>17.1</td>
</tr>
<tr>
<td>15</td>
<td>General linear least squares</td>
<td>17.4</td>
</tr>
<tr>
<td>16</td>
<td>Nonlinear regression</td>
<td>17.5</td>
</tr>
<tr>
<td>17</td>
<td>Applications</td>
<td></td>
</tr>
</tbody>
</table>

### Numerical Differentiation and Integration

<table>
<thead>
<tr>
<th>Day</th>
<th>Topic</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Finite-Divided-Difference formulas, <em>HW 4 (due)</em></td>
<td>23.1</td>
</tr>
<tr>
<td>19</td>
<td>Trapezoidal rule, Simpson’s rule</td>
<td>21.1, 21.2</td>
</tr>
<tr>
<td>20</td>
<td>Gauss quadrature</td>
<td>22.1</td>
</tr>
<tr>
<td>21</td>
<td>Applications, <em>HW 5 (assgn)</em></td>
<td></td>
</tr>
</tbody>
</table>

### Differential Equations

<table>
<thead>
<tr>
<th>Day</th>
<th>Topic</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Euler’s method, Heun’s method</td>
<td>25.1, 25.2</td>
</tr>
<tr>
<td>23</td>
<td>Runge-Kutta methods, Systems of ODEs</td>
<td>25.3, 25.4</td>
</tr>
<tr>
<td>24</td>
<td>Finite-difference methods for BVPs</td>
<td>27.1, 2</td>
</tr>
<tr>
<td>25</td>
<td>Applications to ODEs</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Parabolic PDEs: Finite Difference</td>
<td>Chapter 30</td>
</tr>
<tr>
<td>27</td>
<td>Project Discussions, <em>HW 5 (due)</em></td>
<td></td>
</tr>
</tbody>
</table>

Project reports and presentations are due in the week of Final Exams (TBA).
Appendix B  MATLAB Tutorial: Fall 2011

This appendix contains the MATLAB tutorial material I prepared and presented to undergraduate students that ranged from sophomores to seniors. The two-part tutorial was observed by colleagues at the Eberly Center for Teaching Excellence at CMU, and was held in two days, each with one-hour-and-half sessions.

For brevity, only the handout versions of the slides presented are shown in the following pages. This updated version (Fall 2013) is slightly different from the one I used during the tutorial sessions in Fall 2011. It contains fixes to typographical errors and pictures of the windows of the new MATLAB graphical user interface (2012 onward). In addition to the slides, I also prepared two sets of homework questions with solutions (not included) so that students could practice what was covered in the tutorial sessions. I held the tutorial sessions in two days, and each session lasted for one hour and fifty minutes.

This two-part tutorial assumes no previous familiarity with MATLAB as it starts with “opening MATLAB” and the main windows students should focus on specially in the beginning. While presenting the material, I alternate between the slides and MATLAB for practical demonstrations. I also allow students time to practice themselves. I purposely include logical errors in the MATLAB examples to show the students how the software handles user errors. I show the students how to use the error message issued by MATLAB to guide them on how to fix the error. In every topic, after showing some examples with solutions, I ask students to provide solutions to other examples slightly more demanding in order to gauge their understanding. Finally, before starting a new topic, for example “2-D Arrays”, I briefly summarize the main concepts of the immediate relevant topics, for example “Scalar Variables” and “1-D Arrays”.
1 What is MATLAB?
A powerful tool!
- MATLAB stands for Matrix Laboratory
- Enhanced by toolboxes (specific routines for an area of application)
  - Optimization
  - Statistics
  - Control System
  - Bioinformatics
- Excellent for numerical computations
- Commonly regarded as a ‘Rapid Prototyping Tool’
- Used in industry and academia
3 MATLAB as a Calculator

Basic Operators

- MATLAB supports the following mathematical operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td>^</td>
<td>Exponentiation</td>
</tr>
</tbody>
</table>

- Some examples:
  - >> 1 + 2
  - >> 2^3 + 4
  - >> 4/3 - 3/4 + 2^3

Basic Operators

- Beware of operator precedence rules!
  - >> 2+3 + 4
  - >> 2+(3 + 4)
  - >> 4.2/3 + 1.2
  - >> 4.2/(3 + 1.2)
  - >> 15/(2 + 3)*(4 - 1)
  - >> 15/(2 + 3)*(4 - 1))
  - >> 2^3/2
  - >> 2^3/2

- Use parentheses to enforce the desired order.

4 MATLAB Classes

All Matrices!

- "Everything" in MATLAB is a matrix
  - A scalar is a 1-by-1 matrix
  - A 1D array of n elements can be a n-by-1 (column vector) or a 1-by-n (row vector) matrix
  - A string of n characters is a 1-by-n matrix

- Some MATLAB classes:
  - double (Double-precision floating-point number array) (default)
  - single (Single-precision floating-point number array)
  - char (Character array)
  - cell (Cell array)
  - struct (Structure array)
  - function_handle (Array of values for calling functions indirectly)

Scalar Variables: 1-by-1 Matrices!

- Use the "=' sign for assignment
  - >> a = 1 % The scalar variable 'a' stores the value 1
  - >> % This is a comment and is ignored by the interpreter
  - >> sin(a) % Size of 'a' = 0.8415
  - >> sin(a); % '=' avoids displaying the result of the command
  - >> sin(a) = [1,1]; % 1-by-1 matrix
  - >> b = a + 2 % b = 3
  - >> c = cos(b*pi/2) % 'pi' is the built-in constant π
  - >> d = rand % A random scalar

- Use the commands who or whois to list the variables defined in the Workspace
- Other common functions are available: exp, tan, sinh, acos, ...

1D Arrays: Real Vectors (or Matrices!)

- Use [...] or [...] for horizontal stacking and [ ]...[ ] for vertical stacking
  - >> v1 = [1 2 3] % Row vector, same as v1 = [1,2,3]
  - >> v2 = [4;5;6] % Column vector
  - >> v3 = v2 = v1 % Error! Incompatible matrix dimensions
  - >> v3 = v2' = v1 % Transpose a real matrix with '.'
  - >> v4 = v1*v2 % Dot product, also dot(v1,v2)
  - >> v7 = .1*v1 % Scalar-vector multiplication
  - >> v7(1) % First element of array 'v7'
  - >> v8 = exp(v7) % Element-wise operation
  - >> x = sin(v8) % Element-wise operation
  - >> x = sin(v8) % Size of longest dimension
  - >> v9 = rand(1,5) % Random 1-by-5 array
  - >> p = prod(v1) % Product of elements = 6

2D Arrays: Real Matrices

- Use horizontal stacking and vertical stacking likewise
  - >> m1 = [1 2 3] % 2-by-3
  - >> m2 = [4,5,6] % 2-by-3, same as m1
  - >> m3 = rand(2,3) % Random 2-by-3 matrix
  - >> m4 = m1 + m2 % Matrix addition
  - >> m4 = m1*m2 % Error! Dimensions don't agree
  - >> m4 = m1*m2.' % OK! Transpose a real matrix with '.'
  - >> m4(1,2) % Element in row 1 and column 2 of 'm4'
  - >> m4 = length(m4) % Size of longest dimension
  - >> m5 = m3/2 % Element-wise division
  - >> m6 = tan(m5) % Element-wise operation

3 4
Element-wise Operations

- The following are element-wise mathematical operators:
  
<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>.^</td>
<td>Element-wise Exponentiation</td>
</tr>
<tr>
<td>./</td>
<td>Element-wise Division</td>
</tr>
<tr>
<td>.*</td>
<td>Element-wise Multiplication</td>
</tr>
</tbody>
</table>

- More examples:
  - >> v1 = [1 2 3] % 1-by-3
  - >> v2 = [2 4 6] % 1-by-3
  - >> v3 = v1.*v2 = [2 8 18]
  - >> v4 = v2./v1 = [2 2 2]
  - >> v5 = v1.^v4 = [1 1 9]
  - >> m1 = [0 1; 1 0] % 2-by-2
  - >> m2 = [2 4 6] % 1-by-3
  - >> m3 = m1.*m2 = [0 5; 7 0]

- The Colon (:) Operator

  - Use it extensively!
    - >> v1 = 1:10 % Same as v1 = [1,2,3,...,10]
    - >> v2 = 0:1:11 % Same as v2 = [0,1,2,...,11]
    - >> vl = rand(5) % Random 5-by-5 matrix
    - >> v3 = v1(5:end) % v3 = [5,6,7,8,9,10]
    - >> v4 = m1(:,3) % 'v4' has the elements in column 3 of 'm1'
    - >> v5 = m1(1,:) % 'v5' has the elements in row 1 of 'm1'

- Do not forget linspace to generate linearly spaced vectors!
  - >> v6 = linspace(0,1,10) % [0,0.1111,0.2222,...,1]
  - >> v7 = linspace(0,10,5) % [0,2.5,5,7.5,10]
  - >> v8 = linspace(1,1000,5) % [0,0.0101,0.0202,...,1]

Strings: char Arrays

- Remember that strings are also matrices in MATLAB!
  - >> str1 = 'Hello, world!' % A simple string
  - >> sz1 = size(str1) % 1-by-13
  - >> a = rand; str2 = ['a' num2str(a)] % Horizontal stacking concatenates strings
  - >> b = str2num('500') % RAND and MATLAB has many handy *v2* functions!

- Format your strings with sprintf
  - >> sprintf('Volume of reactor = %.2f', 10.2345) % Floating-point format with two decimal digits
  - >> str3 = sprintf('A large number = %e', rand+10^5) % Exponential notation format
  - >> sprintf('Another large number = %f', rand+10^5) % More compact format between %e and %f

5 Scripts and Functions

5.1 Writing MATLAB Programs

- M-Files
  - The file with source code is called M-File (*.m)
  - Scripts: No input and no output arguments. Usually called program routines and have a special definition syntax.
  - Functions: Accept input and output arguments. Usually called by other scripts. Functions can be used to create 'anonymous functions'.
  - Enable you to divide your M-files into sections (cells).
  - Inside scripts and functions you may use programming statements, such as flow, loop, and error control.
  - Open the Editor Window and start coding!

Function M-Files

- General form:
  ```matlab
  function [output1, output2, ...] = function(input1, input2, ...)
  
  statement
  
  end
  
  % Optional,
  
  % Example:
  
  function [y] = simplefunc(x)
  y = x.*x;
  end
  ```

- Example:
  ```matlab
  function [x,y] = simplefun(x,y)
  y = x.*x;
  end
  ```

5.2 Code Cells and Publishing

- Code Cells
  - Allow you to divide your M-files into sections (cells)
  - Enable you to execute cell by cell
  - Foundations for publishing your M-file to HTML, PDF, and other formats
  - To begin a code cell, type %% at the beginning of a line
  - The titles of the first line after the %% is the title of the code cell
  - The next lines starting with %% are a description of the code cell
  - Place your code in the next lines
  - A new code cell starts at the next %% at the beginning of a line

- Publishing
  - The following are element-wise mathematical operators
  - The Colon (:) Operator
  - Do not forget linspace to generate linearly spaced vectors!
  - Strings: char Arrays
  - Format your strings with sprintf
  - Example:
  ```matlab
  function [y] = simplefunc(x)
  y = x.*x;
  end
  ```

- More examples:
  - >> v1 = [1 2 3] % 1-by-3
  - >> v2 = [2 4 6] % 1-by-3
  - >> v3 = v1.*v2 = [2 8 18]
  - >> v4 = v2./v1 = [2 2 2]
  - >> v5 = v1.^v4 = [1 1 9]
  - >> m1 = [0 1; 1 0] % 2-by-2
  - >> m2 = [2 4 6] % 1-by-3
  - >> m3 = m1.*m2 = [0 5; 7 0]
Code Cells: Example

- Simple example:

```matlab
%% Homework 1
% Bruno Abreu Calfa

%% Problem 1
x = linspace(0,1);
y = sin(x.^2).*exp(-x.*tan(x));
plot(x,y);

%% Problem 2
a = 0;
b = 1;
f = @(t) exp(-t.^2);
intf = quad(f,a,b);
sprintf('Integral of f from b to b = %g',a,b,intf)
```

Publishing your Code

- Saves output of your code to a specific file type
- Formats available:

<table>
<thead>
<tr>
<th>File Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>Microsoft Word</td>
</tr>
<tr>
<td>latex</td>
<td>LaTEX</td>
</tr>
<tr>
<td>ppt</td>
<td>Microsoft PowerPoint</td>
</tr>
<tr>
<td>xml</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>pdf</td>
<td>Portable Document Format</td>
</tr>
<tr>
<td>html</td>
<td>Hypertext Markup Language</td>
</tr>
</tbody>
</table>

- MATLAB evaluates your M-file and generates the output
- To publish your M-file, go to “Publish” tab
MATLAB: Introduction
Part 2
Bruno Abreu Calfa
Last Update: September 16, 2013

Table of Contents
Outline
Contents
1 MATLAB Classes ........................................ 1
2 Elements of Programming ............................. 2
3 Plotting .................................................. 4
   3.1 2-D Plotting ........................................... 4
   3.2 3-D Plotting ........................................... 6

1 MATLAB Classes
ND Arrays
• MATLAB allows multidimensional arrays (n dimensions)
  - >> nd1 = zeros(2,3,4) % 2-by-3-by-4 full of 0s
  - >> nd2 = ones(10,5,8,7) % 10-by-5-by-8-by-7 full of 1s
  - >> nd1(:,1,2) = 1:2 % Replaces column 1 of page 2 by [1,2]
  - >> nd2(:,:,5,7) = rand(10,5) % Replaces rows and columns of page 5 and chapter 7 by random 10-by-5 matrix

Cell and Structure Arrays
• Cell Arrays (cell): generic containers (store any type of data)
  - >> cell1 = {'aaa', 1, rand(2,3)} % Use curly braces to retrieve/assign values
  - >> a = cell1(1) % 'a' is the first container (also a cell)
  - >> b = cell1{1} % 'b' is the first content (a char array)
  - >> cell1{:} % {} generates a comma-separated list
  - >> [a,b,c] = cell1{:} % Assigns each content to a variable

• Structure Arrays (struct): data types with fields and values
  - >> methane.omega = .012; % Methane’s acentric factor
  - >> methane.Tc = 190.6; % Its critical temperature, K
  - >> methane.Pc = 45.99; % Its critical pressure, bar
  - >> methane % Display methane fields and values

2 Elements of Programming
Relational and Logical Operators
• Relational Operators:
  - >, >=, <, <= greater than, smaller than, greater or equal than, smaller or equal than
  - ==, ~= equal to, not equal to

• Logical Operators:
  - &&, & short-circuiting AND, element-wise AND
  - ||, | short-circuiting OR, element-wise OR
  - ~ element-wise NOT

if-elseif-else Statements: Flow Control
• General form:
  - if expression1
  -   statements1
  - elseif expression2
  -   statements2
  - else
  -   statements3
  - end

• Example:
  - r = rand;
  - if (r < .3)
  -   r = r * 2;
  - elseif (r >= .3 && ...
  -   r = r * 3;
  - else
  -   r = r * 4;
  - end

switch-case Statements: Flow Control
• General form:
  - switch switch_expr
  -   case case_expr1, statement
  -   case case_expr2, statement
  -   case expression, ..., statement
  -   otherwise
  -     statement, ..., statement
  - end

• Example:
method = 'Bilinear';
switch lower(method)
    case {'linear', 'bilinear'}
        disp('Method is linear')
    otherwise
        disp('Unknown method')
end

for Loop Statements

- General form:
  
  ```matlab
  for var = init:step:end
      statement
      ...
  end
  ```

- Example:

```matlab
x = zeros(10);
for i = 1:10
    for j = 1:10
        a(i,j) = 1/(i+j-1);
    end
end
```

while Loop Statements

- General form:

```matlab
while expression
    statement
    ...
end
```

- Example:

```matlab
x0 = .5;
x = x0 - tan(x0);
while (sqrt(x^2 - x0^2) > 1E-3)
x0 = x;
x = x0 - tan(x0);
end
sprintf('x_end = %g', x)
```

3 Plotting

3.1 2-D Plotting

- The plotting commands in MATLAB work in a similar way: command(data1, data2, ..., ['Prop1Name', Prop1Value, ...])
- Basic example: plot sin(x) between [0, 2π]

```matlab
x = linspace(0,2*pi);
y = sin(x);
figure
plot(x,y);
```
• Adding more information to the plot of $\sin(x)$ between $[0, 2\pi]$

```matlab
x = linspace(0, 2*pi);
y = sin(x);
figure
plot(x, y, 'Color', 'red');
title('Plot of $\sin(x)$');
xlabel('x');
ylabel('y');
```

• Plotting multiple data on the same figure

```matlab
x = linspace(-10, 10, 1000);
y = 2*x;
z = 4*x.^2 - 2;
w = 8*x.^3 - 12*x;
figure
plot(x, y, x, z, x, w);
title('Plot of three polynomials');
xlabel('x');
ylabel('H(x)');
ylim([-10 10]);
legend('H_2(x)', 'H_3(x)', 'H_4(x)');
```

• Plotting multiple data on the same figure with `hold on` and `hold off`

```matlab
x = linspace(-1, 1, 1000);
y = (3*x.^2 - 1)/2;
z = (5*x.^3 - 3*x)/2;
figure
plot(x, y, 'Color', rand(1, 3));
hold on;
plot(x, z, 'Color', rand(1, 3));
hold off;
```

• Adding multiple plots on the same figure: `subplot`

```matlab
x = linspace(-5, 5, 1000);
y = [x.^2, sin(x), cosh(x), exp(x), exp(-x).*sin(x), x];
colors = lines(6);
figure
for i = 1:6
    subplot(3, 2, i);
    plot(x, y(:, i), 'Color', colors(i, :));
end
```

3.2 3-D Plotting

3-D Plotting

• In three dimensions, you can plot lines (`plot3`) and surfaces (`surf`, `surfc`, `mesh`, `meshc`)

• See MATLAB’s Help for a description of all surface properties

• Set the current color map with the command `colormap`

• Basic example: plot $z = x^2 + y^2$

```matlab
x = linspace(-10, 10, 1000);
y = x;
x, y = meshgrid(x, y);
Z = X.^2 + Y.^2;
figure
surf(X, Y, Z, 'EdgeColor', 'none');
xlabel('x');
ylabel('y');
zlabel('z');
```

6
• Adding contours to $z = x^2 - y^2$

```matlab
x = linspace(-5,5,50);
y = x;
[X,Y] = meshgrid(x,y);
Z = X.^2 - Y.^2;
figure
colormap('cool');
meshc(X,Y,Z);
xlabel('x');
ylabel('y');
zlabel('z');
```

7
Appendix C  Guest Lectures on Climate Change:  Fall 2013

This appendix contains the slides I prepared under the supervision of a subject-matter expert on climate change (Prof. Neil Donahue), and used during two lectures in the Introduction to Chemical Engineering freshman course. The two guest lectures were observed by colleagues at the Eberly Center at CMU, and were held in two days, each with fifty-minute long sessions. In addition to the slides, I also prepared two homework questions with solutions (not included), in which students were asked to perform basic calculations and use of MATLAB to practice the concepts discussed in the two lectures. I gave the lectures in two days, and each lecture lasted for fifty minutes.

The first set of slides (Part I) contains several graphics to illustrate the effects of greenhouse gases in the climate. The visuals also served the purpose of illustrating key concepts in climatology. In addition to pointing students to official websites about climate change, I made use of readily available online video clips to complement the material in the slides. The second set of slides (Part II) contains more of the mechanics of problem-solving in the context of climate change. It makes connections with topics covered in the course, such as concentrations, proportions, and the Ideal Gas Law. In addition, I explained how MATLAB can be used to solve the numerical examples in the slides.
Outline

• Introduction to Climate Change
  – Definitions
  – Primary cause
  – Radiative Balance

• \([\text{CO}_2]\) and Earth’s Temperature

• Anthropogenic \(\text{CO}_2\) Emissions
  – Fossil Fuel Combustion
Global Warming and Climate Change

• Increase in average global temperatures
• Primary cause: greenhouse gases (GHGs)
• GHGs absorb and emit radiation at specific wavelengths within the spectrum of thermal infrared radiation emitted by the Earth’s surface, the atmosphere itself, and by clouds
• GHGs include primarily
  – Water Vapor (H₂O)
  – Carbon Dioxide (CO₂)
  – Nitrous Oxide (N₂O)
  – Methane (CH₄)
  – Ozone (O₃)
  – Fluorinated industrial gases (hydrofluorocarbons, HFCs, perfluorocarbons, PFCs, and sulphur hexafluoride, SF₆)

Relevant source: Intergovernmental Panel on Climate Change (http://www.ipcc.ch/)

Greenhouse Effect

• Atmospheric GHGs absorb radiation from planetary surface
• Re-radiation of absorbed radiation in all directions
• Warms the surface temperature, otherwise it would be too cold
• It is a natural phenomenon
Radiative Balance: 1\textsuperscript{st} Approximation

- Perfectly absorbing Earth
- Surface temperature: $T_s = 279 \, \text{K} = 5.85 \, ^\circ\text{C} = 42.5 \, ^\circ\text{F}$

Values in W m\(^{-2}\). Figure kindly provided by Prof. Neil Donahue.

Radiative Balance: 2\textsuperscript{nd} Approximation

- Part of solar radiation is reflected (mostly by clouds) $\Rightarrow$ albedo
- Surface temperature: $T_s = 255 \, \text{K} = -18.2 \, ^\circ\text{C} = -0.67 \, ^\circ\text{F}$

Values in W m\(^{-2}\). Figure kindly provided by Prof. Neil Donahue.
Radiative Balance: 3rd Approximation

- Optical depth level 1 $\Rightarrow$ radiates to space
- Surface temperature depends on the height of that level (air compression)
  $$T_s = 284 \text{ K} = 10.9 \text{ °C} = 51.5 \text{ °F}$$

Values in W m$^{-2}$. Figure kindly provided by Prof. Neil Donahue.
CO₂ and Earth’s Temperature

- Glacial Period
  - Colder temperatures and glacier advances
- Interglacial Period
  - Warmer temperatures and between glacial periods

- CO₂ presence
  - 280 ppmv = interglacial period
  - 180 ppmv = glacial period
- Ice coring: [http://www.youtube.com/watch?v=oHzAdl-XID8](http://www.youtube.com/watch?v=oHzAdl-XID8)

Rising Temperatures

- Video from NASA’s Goddard Institute for Space Studies (GISS)
- As GHG emissions from energy production, industry and vehicles have increased, temperatures have climbed, most notably since the late 1970s
- Climate scientists use the concept of radiative forcing:
  \[ \Delta F = \text{radiant energy received by Earth} - \text{energy radiated back to space} \]
- Main factors in radiative forcing
  - Solar forcing, which is exacerbated by a decrease in albedo
  - Forcing due to atmospheric gases as GHGs re-radiate energy back to the surface
- Greenhouse effect is natural, but it has been enhanced due to anthropogenic activities
  - Fossil fuel emissions
  - Deforestation

CO₂ Increase since Preindustrial is Anthropogenic

CO₂ from Coal

Figure kindly provided by Prof. Neil Donahue
CO$_2$ from Coal + Oil

CO$_2$ from Coal + Oil + Gas

Figure kindly provided by Prof. Neil Donahue [Oak Ridge Nat. Lab, 2012]
CO₂ from Coal + Oil + Gas + Other Sources

Air Quality at CMU

- Center for Atmospheric Particle Studies (CAPS)
  - [http://caps.web.cmu.edu/members/index.html](http://caps.web.cmu.edu/members/index.html)
  - ChemE faculty
    - Prof. Neil Donahue (Chem)
    - Prof. Spyros Pandis (EPP)
    - Prof. Peter Adams (CivE, EPP)
  - Other faculty members
    - Prof. Allen Robinson (MechE, EPP)
    - Prof. Ryan Sullivan (Chem, MechE)
    - Prof. Albert Presto (MechE)
- Prof. Kris Dahl and I are very grateful for the ideas, materials and guidance from Prof. Neil Donahue in implementing the Climate Change Module in 06-100 (Introduction to Chemical Engineering).
Announcements

- Office Hours today
  - 2:30 pm – 4:00 pm
  - Cyert Hall B6A
- Friday Lecture (09/27)
  - Homework 4 and Recitation 3 due: beginning of class
  - Majors Information Session for Engineering Intro Courses
- Recitation 3 help material on Blackboard
  - Help slides + Additional information on part (d)
- Scott Institute for Energy Innovation
  - [http://www.cmu.edu/energy/](http://www.cmu.edu/energy/)
  - Energy Experts → By Topic Area
**Outline**

• Quantitative Analysis
  – Concentrations and fractions (emphasis on air)
  – Ideal gas

• Basic Calculations
  – Molecular weight of dry air
  – Mass of C corresponding to 1 ppm of CO₂ in the atmosphere (homework)

• Software Workshop
  – Reading and graphing pollution data in MATLAB
  – Plot CO₂ emissions in Mauna Loa (homework)

**Concentrations and Fractions**

• Mass- or mole-based concentrations

\[
\begin{array}{c | c}
\text{\(\mu g\)} & \text{mass of species} \\
\hline
\text{\(m^3\)} & \text{volume of air} \\
\text{mol} & \text{mole of species} \\
\text{\(m^3\)} & \text{volume of air}
\end{array}
\]

• In air, fractions are usually expressed on a molar or volume basis (equivalent for ideal gases)

\[
\begin{array}{c | c}
\% & 1 \text{ part species per } 100 \text{ parts solution} \\
\hline
\text{ppm} & 1 \text{ part species per } 10^6 \text{ parts solution}
\end{array}
\]

also ppmv (on a volume basis), ppb, ppt...

Example: 5 ppb of benzene in air means there are 5 × 10⁻⁹ moles of benzene in 1 mole of air
Ideal Gas Law

• First of all, an “ideal gas” does not exist!
• Simple model of $P$-$V$-$T$ relations of a gas
  – Works well at low $P$ and high $T$
  – Neglects molecular size and intermolecular interactions
• In almost every case in environmental engineering, air can be treated as an ideal gas

\[ PV = nRT \]

or

\[ P\hat{V} = RT \]

where $\hat{V} = \frac{V}{n}$ is the specific molar volume

Ideal Gas Mixtures

• Ideal gas law for species $i$

\[ p_i V = n_i RT \]

• Partial pressure and molar fraction

\[ p_i = \frac{n_i}{n} P = y_i P \]

• Average molecular weight of any mixture with $C$ components

\[ \overline{MW} = y_1 MW_1 + y_2 MW_2 + \ldots + y_C MW_C = \sum_{i=1}^{C} y_i MW_i \]
Exercise: MW of Dry Air

- Given the composition of air, calculate its molecular weight

<table>
<thead>
<tr>
<th>Species</th>
<th>MW [g mol⁻¹]</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nitrogen (N₂)</td>
<td>28</td>
<td>78.08</td>
</tr>
<tr>
<td>Oxygen (O₂)</td>
<td>32</td>
<td>20.95</td>
</tr>
<tr>
<td>Argon (Ar)</td>
<td>40</td>
<td>0.93</td>
</tr>
<tr>
<td>Carbon Dioxide (CO₂)</td>
<td>44</td>
<td>0.035</td>
</tr>
</tbody>
</table>

**Answer:**

\[
MW_{\text{air}} = y_{\text{N}}_2 M_{\text{N}}_2 + y_{\text{O}}_2 M_{\text{O}}_2 + y_{\text{Ar}} M_{\text{Ar}} + y_{\text{CO}}_2 M_{\text{CO}}_2
\]

\[
MW_{\text{air}} = [(0.7808)(28) + (0.2095)(32) + (0.0093)(40) + (0.00035)(44)]\text{g mol}^{-1}
\]

\[
MW_{\text{air}} = 28.95 \text{ g mol}^{-1} \approx 29 \text{ g mol}^{-1}
\]

Homework: Problem 1

- Calculate the mass of C that corresponds to 1 ppm of CO₂ in the atmosphere

**Answer:** 2.2 Gt (giga tons) = 2.2 × 10⁹ t = 2.2 × 10¹² kg

- Hints:
  - Estimate mass or moles of the atmosphere
  - Calculate 1 ppm (amount of CO₂) from the answer above
  - Calculate mass of C (how many moles of C are in one mole of CO₂?)

- You may find useful to know
  - Mean radius of the earth: \( R = 6370 \text{ km} \)
  - Pressure at the surface: \( P = 1.01325 \times 10^5 \text{ Pa} = 1.01325 \times 10^5 \text{ N m}^{-2} \)
CO₂ Levels in MLO: Text Data

- Yearly CO₂ measurements in Mauna Loa (volcano in Hawai’i)
  - Text file by the Earth System Research Laboratory (ESRL) at the National Oceanic & Atmospheric Administration (NOAA) (http://www.esrl.noaa.gov/)
  - Yearly CO₂ measurements in Mauna Loa (volcano in Hawai’i)
  - Text file by the Earth System Research Laboratory (ESRL) at the National Oceanic & Atmospheric Administration (NOAA) (http://www.esrl.noaa.gov/)

  Let’s plot the data to detect trends...

CO₂ Levels in MLO: Graph

- Yearly CO₂ measurements in Mauna Loa (volcano in Hawai’i)
  - Text file by the Earth System Research Laboratory (ESRL) at the National Oceanic & Atmospheric Administration (NOAA) (http://www.esrl.noaa.gov/)
Scatter Plots in MATLAB

```matlab
>> x = [2008 2009 2010 2011 2012 2013]
>> y = [310 350 370 400 410 440]
>> scatter(x, y)
or
>> scatter(x, y, 'fill')
or
>> scatter(x, y, markertype)
where markertype can be 'o', '+', '*', 'd' etc.
```

- Plot data measurements as scatter points
- Plot functions (models) as lines
Some Few Improvements...

```matlab
% Data
x = [2008 2009 2010 2011 2012 2013];
y = [310 350 370 400 410 440];

% Fit 1st degree polynomial (straight line) through the data (trendline)
p = polyfit(x,y,1); % p = p(1)*x + p(2)

% Obtain R^2 value (Coefficient of Determination)
rsq = rsquared(x,y,p);

% We will plot two sets of data on the same figure, so use hold on/off for every plot statement or hold all

% Plot data as scatter points
markersize = 100;
scatter(x,y,markersize,'fill')

% Plot trendline in black
plot(x,polyval(p,x),'k')

% Display trendline equation and R^2
xmiddle = mean(x);
ymiddle = mean(y);
% Note that p(2) is negative, so to make it pretty I add the negative sign
% in the string and display the absolute value of p(2)
text(xmiddle,ymiddle-5,sprintf('[CO_2] = %g t - %g',p(1),abs(p(2))),'FontSize',14)
text(xmiddle,ymiddle-15,sprintf('R^2 = %g',rsq),'FontSize',14)

% Change labels
xlabel('Year','FontSize',16)
ylabel('CO_2 Concentration [ppm]','FontSize',14)

% Change tick marks on x-axis
xticks, xticklabels
```

Some Few Improvements...
Read Data from Text File

• Use MATLAB’s function `dlmread`
  \[ M = \text{dlmread}(\text{filename}) \]
• \( M \) is a matrix with as many rows and columns there are in the text file whose name is `filename`
• Example: File “data.txt”
  
<table>
<thead>
<tr>
<th>Year</th>
<th>CO2 (ppm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>10</td>
</tr>
<tr>
<td>2050</td>
<td>20</td>
</tr>
<tr>
<td>2100</td>
<td>30</td>
</tr>
</tbody>
</table>
• Read data into variable called \( X \)
  \[ X = \text{dlmread}('\text{data.txt}') \]

Homework: Problem 2

• Given a text file (“CO2_MLO_data.txt” from NOAA ESRL) containing annually mean CO2 concentration measurements (ppm) in Mauna Loa (MLO), plot and fit a 2nd degree polynomial to the data. Display the polynomial equation and the \( R^2 \) in the figure.

• Hints
  -- You will have to ask `polyfit` to center and scale the \( x \) values, or MATLAB will issue a warning message
  -- After centering and scaling, the polynomial coefficients are obtained with respect to the centered and scaled \( x \) values and not the original ones. So whenever you have to use the coefficients, make sure to use the centered and scaled \( x \) values.
Air Quality at CMU

- Center for Atmospheric Particle Studies (CAPS)
  - [http://caps.web.cmu.edu/members/index.html](http://caps.web.cmu.edu/members/index.html)
  - ChemE faculty
    - Prof. Neil Donahue (Chem)
    - Prof. Spyros Pandis (EPP)
    - Prof. Peter Adams (CivE, EPP)
  - Other faculty members
    - Prof. Allen Robinson (MechE, EPP)
    - Prof. Ryan Sullivan (Chem, MechE)
    - Prof. Albert Presto (MechE)
- Prof. Kris Dahl and I are very grateful for the ideas, materials and guidance from Prof. Neil Donahue in implementing the *Climate Change Module* in 06-100 (Introduction to Chemical Engineering).
Appendix D  Teaching Assistant Award Text: Academic Year 2011-2012

The following text was prepared and read by the Chemical Engineering Head during the 2012 Department Commencement Ceremony:

“The TA award for 2012 goes to Bruno Calfa.

As the Matlab TA for John Kitchin in Fall 2011, Bruno did a spectacular job. Training in mathematical software has been an orphan topic in the department. It is vital but we have not yet found a way to incorporate it formally into the curriculum. Prof. Kitchin and Bruno initiated a set of Matlab tutorials at the beginning of the fall semester to address this problem. Bruno was in charge of developing and delivering the tutorial on Matlab usage. He has since held tutorials on other computational tools in chemical engineering including Visual Basic extensions of Excel. In addition, he helped Prof. Kitchin incorporate units into Matlab. In meetings with Profs. and myself the undergraduates praised Bruno for his help.”