120th ASEE Annual Conference & Exposition
FRANKLY, WE DO GIVE A D*MN
June 23-26, 2013

Paper ID #6492

Tradeoffs in using Mathematica templates in an introductory numerical methods course

Dr. Shirley B. Pomeranz, University of Tulsa

Shirley Pomeranz Associate Professor Mathematics Graduate Student Advisor Department of Mathematics The University of Tulsa

Research and Teaching Interests: Boundary Element Method and Finite Element Method, Numerical Methods, Engineering Applications of Mathematics, Applications of Mathematica, Women in Mathematics

Tradeoffs in using Mathematica templates in an introductory numerical methods course

Introduction

In this paper, we present tradeoffs involved in using *Mathematica*® (Wolfram Research, Inc.) templates to "level the playing field" and facilitate teaching an introductory numerical methods course. Given increases in class sizes, decreases in budgets and facilities, and changes in technology used in the workplace, the objectives of such courses are evolving[4]. Therefore, the rationale for using computer algebra system (CAS) templates to teach numerical methods encompasses issues that have changed due to today's educational and employment environments. The rationale for the use of programming templates depends on course objectives and varies among disciplines[7, 8, 9]. The author's experiences from 20 years of teaching numerical methods to science, engineering, and mathematics (SEM) students are used to support the use of *Mathematica* programming templates.

Student feedback with respect to the templates was solicited via pre- and post-questionnaires, and show various ways in which students put the templates to use. Templates were used for methods such as the bisection method, fixed-point iteration, Newton's method, cubic spline interpolation, method of undetermined coefficients for approximating derivatives, higher-order Taylor methods for ordinary differential equation initial value problems (ode-ivps), Runge-Kutta methods, Adams-Bashforth-Moulton predictor-corrector methods, and methods for systems of ode-ivps.

Numerical methods course

The course is Math 4503/6603, Introduction to Numerical Methods. This course is a survey course taught to science, engineering, mathematics, and computer science juniors, seniors, and beginning graduate students and is offered by the Department of Mathematics in the College of Engineering and Natural Sciences. The fall semester 2012 enrollment was 40 students. The course balances theory, applications, and programming. Course topics include deriving, using, and performing error analysis on techniques for equation solving, approximation and interpolation, numerical differentiation and integration, and solving ordinary differential equation initial value problems. We use the numerical analysis text by Burden and Faires[1]. The programming assignments use *Mathematica.*

The choice of software depends on several factors, including availability, prevalence, and usefulness[3]. CAS such as *Mathematica* are becoming more commonly used in engineering fields and the workplace[11]. Therefore, for many students learning *Mathematica*, as opposed to some other programming language, makes sense. Further, many other SEM courses at our university use *Mathematica.* The *Mathematica* assignments are used in two different ways: (1) to assist in understanding the numerical methods studied in the course and (2) to increase expertise in the use of *Mathematica* to solve problems encountered in the SEM disciplines and in the workplace.

In the early 1990s, when first teaching this course, the author would spend several weeks at the beginning of the course solely devoted to the use of *Mathematica.* This is no longer done.

Instead, in recent years, the course has evolved into one in which several lecture classes distributed throughout the semester have been replaced by hands-on computer lab classes using *Mathematica*.

Templates

In the fall 2012 version of the course, *Mathematica* templates were used to "jump-start" students with weaker *Mathematica* backgrounds so that these students could (hopefully) program early in the course. Due to diverse student backgrounds, interests, and familiarity with *Mathematica*, it was determined that *Mathematica* programming templates would be used to help students get started with the programming aspects of the course. There are transfer students, international students, graduate students, and non-traditional students. These students often are not familiar with the use of *Mathematica*.

Templates can be structured in many different ways. For a specific numerical methods problem, a template provided to students consisted of either a solution notebook (program file) for a similar simpler problem, or a related problem solution notebook with the main body of code missing. For example, sample code for initialization, an outline, and output formatting might be given. The student would then write the main code implementing a method, include other details, and insert documentation. Students always had the option of rewriting the given sample code. Electronic versions of templates and *Mathematica* homework solutions were made available on the college's computer network for students to copy, adapt, and save in their own personal directories.

Typical *Mathematica* homework assignment topics are: Taylor's theorem, bisection method, fixed-point iteration, Newton's method, cubic splines, Euler's method, Runge-Kutta fourth order method (RK4), Runge-Kutta-Fehlberg method, RK4 for systems of ode-ivps, and consistency, stability, and convergence for multi-step methods.

For a sample template notebook and part of a solution notebook in which portions of code are missing (which can then be assigned to students to complete) used to study the bisection method, see Appendices I and II, respectively. For a sample template notebook in which a simple example of a more complicated problem (which can then be assigned to students) used to study cubic spline interpolation, see Appendix III.

Questionnaires and data analysis

We used a *Mathematica* template in one hands-on computer lab and a template hand-out in one lecture class prior to the "before" questionnaires were administered (on 9-17-2012). This was done so that students were aware of typical expected *Mathematica* work. The "after" questionnaire with the same questions was administered after the last *Mathematica* homework was returned (on 11-14-2012). Questions are adapted from a related study conducted in a chemical engineering course[7].

The questions from the Math 4503/6603 Introduction to Numerical Methods Student Questionnaire are as follows:

1. I am comfortable writing *Mathematica* notebooks (programs).
2. I understand how to use basic *Mathematica* commands and *Mathematica's* **Help**.
3. I understand how to use *Mathematica's* **If** command.
4. I understand how to write some type of *Mathematica* loop (**Do**, **While**, **For**, etc.).
5. I understand how to use the *Mathematica* **Module** command
6. I can perform mathematical operations on variables in *Mathematica*;
7. Using a *Mathematica* template permits me to focus more on solving a problem than if I did not have a *Mathematica* template.
8. The time spent programming an assignment is reduced by having a *Mathematica* template.
9. After becoming familiar with using a *Mathematica* template, using a template is easier than writing *Mathematica* code from the very start.

Any observations, ideas, and/or comments about preferences with respect to using *Mathematica* in this course.

Students responded by selecting one response per numbered question from the five Likert-type categories of "strongly agree", "agree", "neither agree nor disagree", "disagree", or "strongly disagree". Students' perceptions are studied using the difference in the numbers of responses in each of the five categories (ranging from "strongly agree" to "strongly disagree"), comparing the "before" with the "after" student responses. The numbers of responses in each category are given in Table 1. The anticipated outcomes, as expected by the author, were more pronounced in favor of template use than the actual responses received from students. Because students were exposed to templates prior to the "before" questionnaire, the statistical results may be conservative. The results may be conservative in the sense that more statistically significant response differences between the "before" versus the "after" results might have been obtained if there had been no prior student exposure to the templates.

The results with respect to statistical significance appear in Table 2. For each of the nine questions, statistical analysis for independence of the "before" results versus the "after" results was done using the Freeman-Halton test (with SAS Institute, Inc., software). The Freeman-Halton test is a generalization of Fisher's exact test for independence for contingency tables of arbitrary size. We have a 2x5 contingency table (see Table 1) for each question.

The P value is the sum of the probabilities of occurrence of all contingency tables with the same fixed row and column totals, and probabilities less than or equal to that of the observed table. A small P value supports the alternative hypothesis, H_a, of an association (dependence). Equivalently, the P value is the estimated probability of rejecting the null hypothesis, H_0, when that hypothesis is true. The null hypothesis is the hypothesis of no difference between the responses to the "before" questions versus the "after" questions:

H_0: The "before" versus "after" results are independent (no association between "before" versus "after" results).
H_a: The "before" versus "after" results are not independent.

- If $P < 0.05$, then the results are interpreted as statistically significant at the $\alpha = 0.05$ level of significance (having less than one chance in 20 of the "before" versus "after" results being independent, i.e., no association between them).
- If $P < 0.001$, then the results are interpreted as highly statistically significant at the $\alpha = 0.001$ level of significance (having less than one chance in 1000 of the "before" versus "after" results being independent).

For Questions # 1, 2, 6, 7, 8, and 9, comparing the number of "before" versus "after" responses in each category, the results for student responses are not statistically significant. However, students' responses to Questions # 3, 4, and 5 are highly statistically significant. This means that students' **perceptions** of their abilities to use the commands and structures mentioned in those three questions, i.e., the *Mathematica* **If** and **Module** commands and procedural programming looping constructs (**Do**, **For**, **While**, etc.), has changed (increased) from the beginning of the course compared to the end of the course. It should be noted that the analysis measures the strength of the statistical effect, not the strength of the changes in perceptions. It should also be noted that there are many other factors that affect the responses, not just the use of *Mathematica* templates.

Student comments

The student questionnaire included one open-ended solicitation for information, "Any observations, ideas, and/or comments about preferences with respect to using *Mathematica* in this course". There was a diversity of responses from students who chose to share their ideas. Some of the student responses are quoted below.

Student comments from the "before" questionnaire (9-17-2012):

- I think that the most helpful aspects of the templates are that they gave examples of how to use the display commands like **Plot** and **TableForm**. Other than that, I usually feel like I understand what I am doing better if I don't rely on the templates very much.

- The templates are helpful but detract from actually learning the math. I remember less of it when using the templates.

- I actually think that writing some of the loop coding helps me understand the way the problem is solved. However, sometimes an error in the coding takes a long time to uncover and fix.

- I had a lot of trouble with the loop the first time I tried it on the bisection homework. But once I got that the rest was easy. It was a matter of syntax, not content. The same goes for the **Module** command.

- Sometimes *Mathematica* error messages can be less than helpful. For example, I spent about 30 minutes debugging a "write protected" error when I just needed to add a semicolon to the end of a line.

Table 1. Student questionnaire responses

	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree	Total
Question #1 before	8	18	5	6	0	37
Question #1 after	7	18	4	2	0	31
Question #2 before	19	17	1	0	0	37
Question #2 after	20	11	0	0	0	31
Question #3 before	12	18	3	3	1	37
Question #3 after	20	11	0	0	0	31
Question #4 before	8	19	6	2	2	37
Question #4 after	20	9	0	2	0	31
Question #5 before	2	14	10	9	2	37
Question #5 after	10	14	5	1	1	31
Question #6 before	12	21	3	1	0	37
Question #6 after	16	14	1	0	0	31
Question #7 before	17	11	8	0	1	37
Question #7 after	22	7	2	0	0	31
Question #8 before	18	11	7	0	1	37
Question #8 after	22	7	1	1	0	31
Question #9 before	19	11	6	0	1	37
Question #9 after	21	7	3	0	0	31

Table 2. Statistical significance of student questionnaire responses

Question #	P value	Is the difference between before and after results statistically significant?
1	0.70	Not statistically significant
2	0.46	Not statistically significant
3	**0.00036**	**Highly statistically significant**
4	**0.0000059**	**Highly statistically significant**
5	**0.000011**	**Highly statistically significant**
6	0.31	Not statistically significant
7	0.11	Not statistically significant
8	0.076	Not statistically significant
9	0.55	Not statistically significant

- Being a computer science major makes it easy to pick up *Mathematica's* language (even though I only used it once before), but people with different backgrounds will probably have more trouble. It might help to tell students in the course description that this course is *Mathematica* intensive. (I didn't know that when I signed up.)

- I have noted above [in the questionnaire] the methods I learned while doing homework for this class. This class helps me analyze data in all of my other classes. Test looks hard, though.

- Can we have our tests with *Mathematica*?

- If it was not for the templates, I would be very lost.

- I definitely like the template. Hints on the trickier parts of the problems are also extremely helpful and appreciated.

- I would say if no template were to be given, a help sheet for the code would be nice.

- In previous classes, templates were not available and the amount of time spent figuring out how to write code was much greater than doing the math. The templates fix that.

Student comments from the "after" questionnaire (11-14-2012):

- I understand very basic parts of *Mathematica*, and this course has definitely "beefed up" my experience with it. It's just daunting at the beginning of the class to have been expected to write **Do**, **If**, **While** loops because I'm very uncomfortable with computers (not a computer science major!!!).

- Though I have improved my ability in *Mathematica*, I hate it more than ever. Mostly due to stability issues and lack of intuitiveness with the language itself.

- I think it was hard to connect the assignments with doing problems by hand. I could solve by hand but had no idea how to transfer to *Mathematica*.

- I felt that some of the homeworks were more about evaluating our *Mathematica* proficiency rather than solving the problems and learning the course material. (This may have been the intent – it is just an observation.)

- Try to limit class size so that Mathematica is available on test. That would help soooooo much.

- I learned a lot about *Mathematica*; templates were extremely helpful as both starter and reference materials.

- If possible, I prefer to let the student choose his own programming language instead of only using *Mathematica*.

- I learn well from examples of code so an example (toy) problem with some code demonstration would be helpful.

- After only a slightly rocky start (I had almost no *Mathematica* experience) I am happy with how the homework has gone.

- The homework had a nice difficulty curve that allowed me to learn a lot of *Mathematica*!

- The problem statements available in *Mathematica* are a big help and help focus on the actual problem instead of typesetting.

- Having the code actually was helpful to me for another class.

Tradeoffs: How do templates help and how do they hinder?

The results of the student questionnaire, as well as the author's observations, suggest the following advantages and disadvantages of using templates.

Pros

The use of templates frees up class time and student time so that more substantive mathematical questions/topics can be considered.

For students who learn by example, templates give a uniform, structured way of providing examples that can then be adapted as needed. Further, templates are effective for modification by the instructor for future assignments.

An ordered and controlled means of increasing the complexity of problems assigned to students is provided. For example, after the template for fixed-point iteration was provided, students were

then assigned homework to code Newton's method in *Mathematica* to solve a specific problem. This could be accomplished by adapting the previous fixed-point iteration template.

The familiar structure of submitted work implies easier grading of homework. For example, in larger classes, a paper grader can grade *Mathematica* homework.

Cons

As observed from some of the student comments, the use of templates may interfere with learning and/or stifle creativity. There is less of a tendency to understand and remember work that is not one's own. A useful balance of work with versus without templates is achieved. Soliciting student feedback and a trial-and-error approach to this balance applied throughout the semester is suggested[5].

Even with the use of templates, students stated that they needed more hands-on labs. Especially with international, graduate, and transfer students, the use of modules to replace much of the hands-on lab time did not work well. Weekly assignments often did not give enough time for students to effectively debug their code (at least that is what students stated).

It was too easy for this professor to start at too complex a programming level by assuming that students were more familiar with *Mathematica* than they actually were. Similarly, the temptation to proceed too quickly must be restrained.

As a related issue, what about students learning how to debug their own code? When is help too much help or simply the wrong type of help? With respect to debugging, instead of directly showing what is wrong, simply looking at the incorrect output (in cases where there is output) can help a student learn how to more effectively debug? For example, if *Mathematica* is not simplifying expressions that it would be expected to simplify, then this might be recognized as a clue that some symbol was not typed correctly. If output contains "**Log e**", which automatically should have been simplified to be 1, this might indicate that the Roman letter e was used instead of, for example, the correct palette symbol **e**.

Summary

Comparisons with respect to the use of programming templates can be made with studies for specific engineering courses done by other academics. Template use is discussed in teaching chemical engineering courses[7,8,9] and computational fluid dynamics[10]. The discussion of the use of spreadsheets to teach electrical engineering[2] can be extended to CAS programming templates.

The use of templates alone to introduce *Mathematica*, even though this is done with hands-on computer labs, may not be sufficient. But is the use of templates helpful to students? If done well, then the author feels that the answer is "yes". The author and colleagues plan more detailed study on this topic.

Mathematica templates and notebooks for the bisection method, fixed-point iteration, Newton's method, cubic spline interpolation, method of undetermined coefficients for approximating

derivatives, higher-order Taylor methods ordinary differential equation initial value problems (ode-ivps), Runge-Kutta method, predictor-corrector method, Adams-Bashforth-Moulton predictor-corrector method, and methods for systems of de-ivps are available by contacting the author[6].

Acknowledgements

The author thanks Dr. Peyton Cook, Department of Mathematics, The University of Tulsa, for the statistical analysis.

Thank you to the students of Math 4503/6603, Introduction to Numerical Methods.

The referees are thanked for their very helpful advice.

Bibliography

[1] Burden, Richard L. and Faires, J. Douglas, Numerical Analysis, 9th Edition, Brooks/Cole Publishing Company, 2011.

[2] Chehab, Ali; El-Hajj, Ali; Al-Husseini, Mohammed; and Artail, Hassan; "Spreadsheet application in electrical engineering: A review", *Int. J. Engng. Ed.*, Vol. 20, No. 6, pp. 902-908 (2004).

[3] Costanzo, Francesco and Gray, Gary L., "On the implementation of interactive dynamics", *Int. J. Engng. Ed.*, Vol. 16, No. 5, pp. 385-393 (2000).

[4] Enszer, Joshua A.; Goodrich, Victoria E.; and Getman, Rachel B., "Improvements in computational methods courses in chemical engineering", *Proceedings of the 2012 American Society for Engineering Education Annual Conference and Exposition* (AC 2012-4402).

[5] Ogot, Madara and Okudan, Gül, Okudan, "A student-centered approach to improving course quality using quality function deployment", *Int. J. Engng. Ed.*, Vol. 23, No. 5, pp. 916-928 (2007).

[6] pomeranz@utulsa.edu.

[7] Silverstein, David L., "Template based programming in chemical engineering courses", *Proceedings of the 2001 American Society for Engineering Education Annual Conference and Exposition* (3513).

[8] Silverstein, David L., "Increasing time spent on education course objectives by using computer programming to teach numerical methods", *Chemical Engineering Education*, Vol. 37, No. 3, pp. 214-218 (2003).

[9] Silverstein, David L., "Using pre-built program templates to teach numerical methods", *Proceedings of the 2004 American Society for Engineering Education Annual Conference and Exposition* (1520).

[10] Stern, Fred; Xing, Tao; Yarbrough, Don; Rothmayer, Alric; Rajagopalan, Ganesh; Otta, Shourya Prakash; Caughey, David; Bhaskaran, Rajesh; Smith, Sonia; Hutchings, Barbara; Moeykens, Shane; "Development of *hands-on* CFD Educational interface for undergraduate engineering courses and laboratories", *Proceedings of the 2004 American Society for Engineering Education Annual Conference and Exposition* (1526).

[11] www.wolfram.com/solutions/, 12-12-2012,

Appendix I. Sample bisection method template

■ MATH 4503/6603 Mathematica Lab - *Mathematica* and the Bisection Method

■ **Use *Mathematica* to implement the Bisection Method.**

Typical Problem: Given the rational function, $f(x) = \frac{3x^3-7x^2-6x+7}{7x^3+6x^2-3x+6}$.

a) Use a graph to locate (approximately) the largest real zero of f.

b) Program and use the bisection method to approximate the largest real zero of f, to within a tolerance of 10^{-5}.

c) Use the minimum number of bisections for which the result is guaranteed to satisfy the given tolerance (have *Mathematica* calculate this number, using Theorem 2.1).

Solution:

■ **a) Initialization**

INSERT TOOLBAR AND PALETTE(S), AS NEEDED

This code is written for *Mathematica* Version 8.0.

```
Dynamic[Refresh[DateString[], UpdateInterval → 1]]
```

```
Wed 27 Feb 2013 17:08:51
```

```
Clear["Global`*", "Subscript"]
```

```
(* Off[Solve::"ifun"] *)
```

■ **b) Verification that the bisection method can be applied**

INSERT MISSING CODE FOR THE FUNCTION DEFINITION AND RELATED GRAPHICS

```
f[2] * f[3]
```

$$-\frac{21}{6400}$$

Observe that f is continuous on [2, 3] and that $f(2) * f(3) < 0$ (i.e., $f(2)$ and $f(3)$ have opposite signs).
Therefore, the hypotheses of the bisection method are satisfied, and we can apply the bisection method to approximate a zero of f in [2, 3].

■ **c) Bisection method initialization**

Notation:
The initial interval is chosen as $[a, b] = [a(1), b(1)] = [2, 3]$, and its midpoint is $p(1)$. The error tolerance is tol = 10^{-5}. The iteration index (counter) is denoted by n, with nMax, the number of iterations required. Note that a smaller interval could have been chosen and would result in fewer iterations, in general.

```
a[1] = 2; b[1] = 3;
```

```
tol = 10^-5;
```

Apply text's Theorem 2.1:

```
sol = Solve[(b[1] - a[1])/2^n == tol, n] // N // Flatten
```

Solve::ifun : Inverse functions are being used by Solve, so

some solutions may not be found; use Reduce for complete solution information. >>

```
{n → 16.6096}
```

```
nMax = Ceiling[n /. sol]
```

```
17
```

■ d) Bisection method computations

The bisection method main loop is expressed as follows within a **Module** command. Note the use of delayed assignment:

INSERT MISSING CODE FOR THE BISECTION METHOD (MODULE)

Execute the bisection method with the appropriate data:

```
bisectionMethod[a[1], b[1], f, nMax]
```

The following list was printed for debugging purposes and should be deleted before printing the final version of this notebook (or the output can be suppressed with a semicolon).

■ e) Bisection method numerical results

COULD INSERT CODE TO INCLUDE $f[p[n]]$ IN DATA LIST AND TABLE

```
data = Table[{n, a[n], b[n], p[n]}, {n, 1, nMax}]
```

```
{{1, 2, 3, 2.5}, {2, 2.5, 3, 2.75}, {3, 2.75, 3, 2.875}, {4, 2.75, 2.875, 2.8125},
 {5, 2.75, 2.8125, 2.78125}, {6, 2.75, 2.78125, 2.76563}, {7, 2.75, 2.76563, 2.75781},
 {8, 2.75, 2.75781, 2.75391}, {9, 2.75, 2.75391, 2.75195},
 {10, 2.75195, 2.75391, 2.75293}, {11, 2.75195, 2.75293, 2.75244},
 {12, 2.75195, 2.75244, 2.7522}, {13, 2.75195, 2.7522, 2.75208},
 {14, 2.75195, 2.75208, 2.75201}, {15, 2.75195, 2.75201, 2.75198},
 {16, 2.75198, 2.75201, 2.752}, {17, 2.75198, 2.752, 2.75199}}
```

Bisection Method Output:
The values of p_n, $n = 1, \ldots, nMax$, are the successive approximations.

```
Print[Style["Bisection Method Results", FontFamily → "Harrington", Bold, 14, Red]];
TableForm[data,
 TableHeadings -> {{}, {"n", "a_n", "b_n", "p_n\n"}}, TableSpacing → {2, 5}]
```

n	a_n	b_n	p_n
1	2	3	2.5
2	2.5	3	2.75
3	2.75	3	2.875
4	2.75	2.875	2.8125
5	2.75	2.8125	2.78125
6	2.75	2.78125	2.76563
7	2.75	2.76563	2.75781
8	2.75	2.75781	2.75391
9	2.75	2.75391	2.75195
10	2.75195	2.75391	2.75293
11	2.75195	2.75293	2.75244
12	2.75195	2.75244	2.7522
13	2.75195	2.7522	2.75208
14	2.75195	2.75208	2.75201
15	2.75195	2.75201	2.75198
16	2.75198	2.75201	2.752
17	2.75198	2.752	2.75199

- **f) Verification of the bisection method numerical results**

We can compare with the result from *Mathematica*'s **FindRoot** command to verify that our numerical approximation is within the error tolerance.

```
exact = FindRoot[f[x] == 0, {x, 0}]
```

```
{x → 2.75199}
```

```
exactZero = x /. exact
```

```
2.75199
```

```
AbsErrorMagnitude = Abs[p[17] - exactZero]
```

$$4.85552 \times 10^{-6}$$

Verify that the approximate zero is within the specified tolerance of the exact zero (which is known in this problem, because the problem is a "test problem"):

```
AbsErrorMagnitude < tol
```

```
True
```

Conclusion:
An approximation to the exact solution to within the stated tolerance is given by $p_{17} = 2.75199$ (rounded), as required.

Appendix II. Sample module solution for bisection method

■ d) Bisection method computations

The bisection method main loop is expressed as follows within a **Module** command. Note the use of delayed assignment:

```
bisectionMethod[aa_, bb_, F_, numberOfIterations_] := Module[{},
  a[1] = aa;
  b[1] = bb;
  If[F[a[1]] * F[b[1]] > 0,
     Print["Error message. Root is not suitably bracketed. Stop."]; Abort[]];
  If[F[a[1]] * F[b[1]] == 0,
     Print["Error message. Root is an end point. Stop."]; Abort[]];
  (* Note the use of the decimal point in the following command;
  This forces Mathematica to use approximate floating-point arithmetic,
  which is generally faster than exact arithmetic. *)
  p[1] = (a[1] + b[1])/2.;
  Do[
   If[F[p[n]] == 0,
      Print["The midpoint p(", n, ")= ", p[n], " is a root. Stop."]; Abort[]];
   If[F[a[n]] * F[p[n]] < 0, a[n + 1] = a[n]; b[n + 1] = p[n],
      a[n + 1] = p[n]; b[n + 1] = b[n]];
   p[n + 1] = a[n + 1] + (b[n + 1] - a[n + 1])/2.,
   {n, 1, numberOfIterations}]
 ]
```

Appendix III. Sample module for a simple cubic spline interpolation

CUBIC SPLINES - Sec. 3.5 (Burden and Faires, 9th edition)
Mathematica lab: Cubic Spline interpolation

- **Cubic splines:**

Basic problem: We are given the data points (x_k, y_k), $k = 0, \ldots, n$. Obtain and graph $s(x)$, the **natural** cubic spline (for which $s''(x_0) = 0$ and $s''(x_n) = 0$).

This outline is different from, but equivalent to, the presentation in Section 3.5 of our text, Numerical Analysis, 9th Edition, Burden and Faires.

The following list is a possible outline for the computational steps:

Create lists containing the x and y coordinates, respectively, for the interpolation points;

The natural cubic spline $s(x)$ (which satisfies $s''(x_0) = 0$ and $s''(x_n) = 0$) is constructed from first principles. Begin with the cubic polynomials:

Interpolate at pairs of adjacent points to obtain the interpolating equations for the cubic polynomials defined on the subintervals. There are n subintervals and, consequently, n cubic polynomials to be defined;

Impose continuity of the first derivatives of the cubic spline at all interior nodes to obtain additional equations;

Impose continuity of the second derivatives of the cubic spline at all interior nodes to obtain further equations;

Implement the endpoint conditions to obtain the final two equations;

Determine the solution of this linear system of $4n$ equations in $4n$ unknowns;

Construct the cubic spline $s(x)$;

Verify that the end conditions for a free (natural) cubic spline are met, $s''(x_0) = 0$ and $s''(x_n) = 0$;

Generate a plot of the data points and $s(x)$, the natural cubic spline, etc.;

Example - Develop a very simple natural cubic spline:

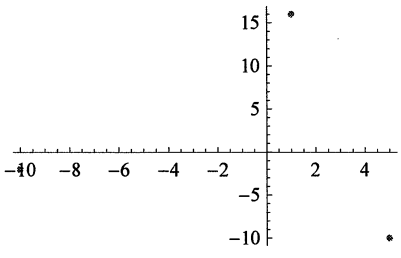
- **Initialization:**

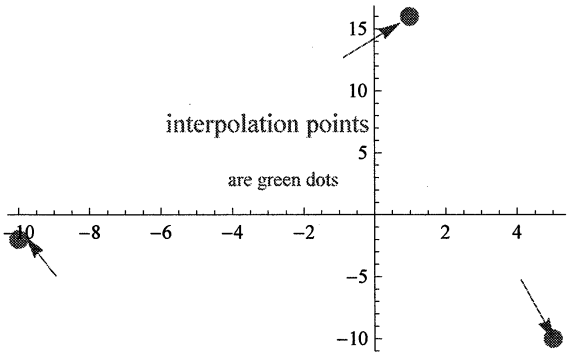
The following code is designed to run in *Mathematica* 8.0.

```
In[1]:= Clear["Subscript", "Global`*"];

In[2]:= DateString[]
Out[2]= Thu 13 Dec 2012 11:50:21

In[3]:= data = {{-10, -2}, {1, 16}, {5, -10}};
```

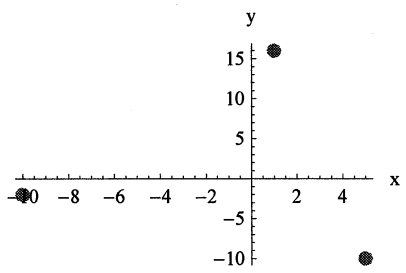
Observe various ways to present graphics:

```
ListPlot[data, ImageSize → Small]
```






```
plot1 = ListPlot[data, PlotStyle → {Red, PointSize[0.04]},
  PlotLabel -> "        Interpolation Points\n", AxesLabel → {"x", "y"},
  ImageSize → Small, PlotRangeClipping → False]
```



- **Code to develop the cubic spline:**

Caution: The notation used in this *Mathematica* notebook differs from that used in the text.

Note: There are 2 subintervals and 2 cubic polynomials, $n = 2$. Observe that $s_i(x)$, $i = 0, \ldots, n-1$, denotes the ith cubic polynomial comprising the cubic spline, $s(x)$. Find the natural cubic spline using the given data set.

```
n = 2;
```

```
s0[x_] = a0 * x^3 + b0 * x^2 + c0 * x + d0;
s1[x_] = a1 * x^3 + b1 * x^2 + c1 * x + d1;
```

- **Example of a module:**

```
constructCubicPolynomials[num_] := Module[{},
  Print[Style["These are the cubic polynomials comprising the cubic spline.",
    FontFamily → "Chiller", Magenta, Bold, 16]];
  Do[
s_i[x_] = a_i * x^3 + b_i * x^2 + c_i * x + d_i;
   Print["s"_i, "(x)=", s_i[x]],
   {i, 0, num - 1}];
  SequenceForm[Style["Observe that there are ", FontFamily → "Forte", Red, 14],
   num, Style[" cubic polynomials.", FontFamily → "Curlz MT", Blue, Bold, 12]]
 ]

constructCubicPolynomials[n]
```

These are the cubic polynomials comprising the cubic spline.

$s_0(x) = x^3\, a_0 + x^2\, b_0 + x\, c_0 + d_0$

$s_1(x) = x^3\, a_1 + x^2\, b_1 + x\, c_1 + d_1$

Observe that there are 2 cubic polynomials.

- **Construct and solve the linear system of equations:**

Note: There are 8 unknowns to determine; therefore, we need 8 independent equations;

```
eq_1 = s_0[-10] == -2;
eq_2 = s_0[1] == 16;
eq_3 = s_1[1] == 16;
eq_4 = s_1[5] == -10;
eq_5 = s_0'[1] == s_1'[1];
eq_6 = s_0''[1] == s_1''[1];

eq_7 = s_0''[-10] == 0;
eq_8 = s_1''[5] == 0;
```

We have this system of $4*n$ equations in $4*n$ unknowns:

```
Table[eq_i, {i, 1, 4 * n}] // ColumnForm
```

$-1000\, a_0 + 100\, b_0 - 10\, c_0 + d_0 == -2$
$a_0 + b_0 + c_0 + d_0 == 16$
$a_1 + b_1 + c_1 + d_1 == 16$
$125\, a_1 + 25\, b_1 + 5\, c_1 + d_1 == -10$
$3\, a_0 + 2\, b_0 + c_0 == 3\, a_1 + 2\, b_1 + c_1$
$6\, a_0 + 2\, b_0 == 6\, a_1 + 2\, b_1$
$-60\, a_0 + 2\, b_0 == 0$
$30\, a_1 + 2\, b_1 == 0$

Solve the system of $4*n = 8$ independent linear equations in $4*n$ unknowns. Obtain the following list of replacement rules:

```
solution = Solve[Table[eq_i, {i, 1, 4 * n}],
   Flatten[Table[{a_i, b_i, c_i, d_i}, {i, 0, n - 1}]] ] // Flatten
```

$$\left\{a_0 \to -\frac{179}{7260},\ b_0 \to -\frac{179}{242},\ c_0 \to -\frac{20161}{7260},\right.$$
$$\left. d_0 \to \frac{4729}{242},\ a_1 \to \frac{179}{2640},\ b_1 \to -\frac{179}{176},\ c_1 \to -\frac{6599}{2640},\ d_1 \to \frac{3423}{176}\right\}$$

- **Numerical results:**

```
s0[x_] = s0[x] /. solution
```

$$\frac{4729}{242} - \frac{20161\,x}{7260} - \frac{179\,x^2}{242} - \frac{179\,x^3}{7260}$$

```
s1[x_] = s1[x] /. solution
```

$$\frac{3423}{176} - \frac{6599\,x}{2640} - \frac{179\,x^2}{176} + \frac{179\,x^3}{2640}$$

```
? Piecewise;
```

The cubic spline, *s*, is:

```
s[x_] = Piecewise[{{s0[x], -10 ≤ x < 1}, {s1[x], 1 ≤ x ≤ 5}}]
```

$$\begin{cases} \frac{4729}{242} - \frac{20161\,x}{7260} - \frac{179\,x^2}{242} - \frac{179\,x^3}{7260} & -10 \le x < 1 \\ \frac{3423}{176} - \frac{6599\,x}{2640} - \frac{179\,x^2}{176} + \frac{179\,x^3}{2640} & 1 \le x \le 5 \\ 0 & \text{True} \end{cases}$$

```
s[x] // N
```

$$\begin{cases} 19.5413 - 2.777\,x - 0.739669\,x^2 - 0.0246556\,x^3 & -10. \le x < 1. \\ 19.4489 - 2.49962\,x - 1.01705\,x^2 + 0.067803\,x^3 & 1. \le x \le 5. \\ 0. & \text{True} \end{cases}$$

Optional: Verify that the boundary conditions are satisfied:

```
s''[-10]
```

```
Indeterminate
```

```
s''[5]
```

```
Indeterminate
```

```
Limit[s''[x], x → -10]
```

```
0
```

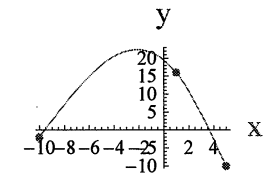
```
Limit[s''[x], x → 5]
```

```
0
```

- **Graphics:**

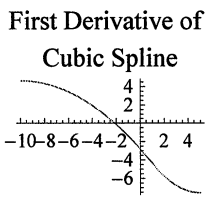
We can graph the natural cubic spline and some of its derivatives. Observe that continuity is as expected:

```
plot2 = Plot[s[x], {x, -10, 5}, AxesLabel → {"x", "y"},
    PlotLabel → Style["Cubic Spline", FontFamily -> "Comic Sans MS", 12, Purple],
    PlotStyle → Hue[0.8], ImageSize → Small];

Show[plot2, plot1, PlotLabel → Style["Cubic Spline and \nInterpolation Points\n",
    FontFamily -> "Comic Sans MS", 12, Bold, Hue[0.9]], PlotRangeClipping → False]
```

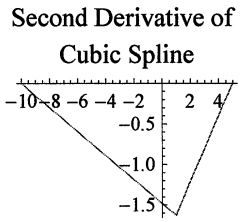


- **Optional: Investigation of smoothness of higher order derivatives of the cubic spline:**

```
Plot[s'[x], {x, -10, 5}, ImageSize → Small,
 PlotLabel → "First Derivative of \nCubic Spline"]
```




```
Plot[s''[x], {x, -10, 5}, ImageSize → Small,
 PlotLabel → "Second Derivative of \nCubic Spline"]
```



However, note that the following graph is discontinuous at the interior node(s) (as would be expected):

```
Plot[s'''[x], {x, -10, 5}, ImageSize → Small,
 PlotLabel → "Third Derivative of \nCubic Spline"]
```