This syllabus by Charlie van Loan is online at

http://www.cs.cornell.edu/courses/cs322/2004sp/syllabus.htm

Note that the parts of this syllabus are very clearly listed here. The detailed content can be found online.

In case the syllabus is removed from the web, each link is pasted into this document starting at the top of a new page.

CS 322 Home Page
(Spring 2004)

Announcements/Book Errata	Assignment updates and other timely messages.
Getting Help	The course staff.
Meeting Times	Where and when the lectures and sections meet.
Books and Software	Required and optional course materials.
Course Overview	High-level course description.
Week-by-Week Syllabus	Readings, due dates, and lecture/section topics.
Section Assignments	There will be thirteen of these.
Programming Assignments	There will be five of these.
Lecture Handouts/Examples	What you may have missed.
Review Sessions and Exams	Where to go and when.
Grading Policies	Late submissions, regrades, how things count.

Facilities	Where to work on the programs.
Before You Take CS 322	Are you ready to take the course?
After You Take CS 322	Information about follow-up courses.
Academic Integrity	Understand the rules.
Interesting Links	If you are curious...

Announcements

You are expected to check for new announcements every day.

On Mondays we move dated announcements to the archives.

Book errata (PS/PDF) are available.

May 24 Final Exam grading guide is available.

May 19. CVL Office hours today 1:15-1:145, not 1:15 - 2:45.

May 19. Solutions to the May 18 problems are here.

May 18 As a way of reviewing the A6 material, complete the following functions so that they perform as specified. (Solutions at review session and posted tomorrow.)

```
  function B = RevProb1(A,c,s,i,j)
% A is n-by-n and 1 < = i <j <=n.
% Let Q be the n-by-n matrix that is the identity everywhere except
% Q(i,i) = Q(j,j) = c and Q(i,j) = -Q(j,i) = s.
% B = Q'*A*Q


  function [B,C] = RevProb2(A,u,v,w,x)
% A is n-by-n and u, v, w, and x are column n-by-1 vectors
% Let M be the n--by-n matrix defined by M(:,j) = u(j)*w + v(j)*x
% B = M*A and C = A*M

  function x = RevProb3(U,S,V,b)
% U and V are n-by-n orthogonal matrices, S is a diagonal n--by-n
% matrix, and b is a column n-vector.  x solves (U*S*V')x = b.
```

May 17 Tentative A6 solution guide is available.

May 14. Here is a practice final and an actual final from previous years. Each has a differential equation problem which you can ignore.

May 12

Final exam will be Friday, May 21, 9-11:30AM, in Ives 305. The exam will cover those concepts that are associated with A1-A6 and all the Section problems. A set of study problems will be vailable soon.

Review Session will be Tuesday, May 18, 7:30-9:00pm in Upson B-17

Office hours: Adam = MTWTh 10-11, CVL = MTWTh 1:15-2:45, Gun MTWTh 3-4

CS 322 Staff

| CS 322 Home |

| Lecturer | Teaching Assistants |

Lecturer

Charles Van Loan

Office: 4130 Upson Hall
Phone: 255-7644
Email: cv@cs.cornell.edu
Office Hours: Walk-In hours are as follows...

Monday	3:00-4:15
Tuesday	1:15-2:30
Wednesday	3:00-4:15
Thursday	11:00-12:00

Sometimes I have to cancel hours because of unavoidable meetings and out-of-town travel. Check here for what the schedule looks like for the current week. Nora always knows my schedule and can arrange special appointments.

Teaching Assistants

Gun Srijuntongsiri

Office: 427 Rhodes Hall
Phone: 255-255-3864
Email: gunsri@cs.cornell.edu
Office Hours: Monday 9-10, Friday 1:15-2:15

Adam Arbree

Office: 4162 Upson Hall
Phone: 255-2219
Email: arbree@cs.cornell.edu

Office Hours: Tuesday 11-12, Friday 4-5

Books and Software

Required Text (SCMV)

Introduction to Scientific Computing: A Matrix-Vector Approach based on Matlab (Second Edition)

Author: Charles Van Loan

Publisher: Prentice Hall

ISBN: 0-13-949157-0

Note: the First Edition will not work for the course.

Available On-line and Required (NCM)

Moler and Moler, *Numerical Computing With Matlab*

Note: This is free.

Software

The M-files associated with the SCMV are available several ways.

The M-files associated with NCM are available several ways.

Course Overview

Catalog Copy:

COM S 322 Introduction to Scientific Computation (also ENGRD 322)

Spring, summer. 3 credits. Prerequisites: COM S 100 and MATH 221 or 294.

An introduction to elementary numerical analysis and scientific computation. Topics include interpolation, quadrature, linear and nonlinear equation solving, least-squares fitting, and ordinary differential equations. The MATLAB computing environment is used. Vectorization, efficiency, reliability, and stability are stressed. Special lectures on parallel computation.

From the Preface to the Text:

Matlab affects the way we do research in scientific computing because it encourages experimentation with interesting mathematical ideas. Visualization and vector-level thinking are supported in a way that permits focus on high-level issues. It is by clearing such a wide path from research to applications that Matlab has been such an uplifting force in computational science. For exactly the same reasons, Matlab can uplift the teaching of introductory scientific computing. Students need to play with the mathematics that stands behind each and every new method that they learn. They need graphics to appreciate convergence and error. They need a matrix-vector programming language to solidify their understanding of linear algebra and to prepare for a world of advanced array-level computing. They need a total problem solving environment tapping into the very latest algorithmic research that has a bearing on science and engineering. In short, they need Matlab.

In this textbook I present all the topics that are usually covered in a one-semester introduction to scientific computing. But graphics and matrix-vector manipulation have been folded into the presentation in a way that gets students to appreciate the connection between continuous mathematics and computing. Each of the nine chapters comes equipped with a theorem. Analysis is complemented with computational experiments that are designed to bolster intuition. Indeed, the text revolves around examples that are packaged in 200+ M-files. These codes are critical to the overall presentation. Collectively they communicate all the key mathematical ideas and an appreciation for the subtleties of numerical computing. They also illustrate many features of Matlab that are likely to be useful later on in the student's computational career. Snapshots of advanced

computing are given in sections that deal with parallel adaptive quadrature and parallel matrix computations. Our treatment of recursion includes divided differences, adaptive approximation, quadrature, the fast Fourier transform, Strassen matrix multiplication, and the Cholesky factorization. Numerical linear algebra is not confined to the matrix computation units. Because of the graphics thread throughout the text, it permeates the entire presentation beginning in Chapter 1. That first chapter is yet another get-started-with-Matlab tutorial, but it is driven by examples that set the stage for the numerical algorithms that follow.

Week-By-Week Syllabus

| CS 322 Home |

Week	Lecture	Notes
1	**Jan 26** Introduction **Jan 28** Introduction	**Section**: S1: P1.1.2, P1.2.5, P1.3.3 **Reading:** 1.1-1.4, Read the course website. **Event:**
2	**Feb 2** Introduction **Feb 4** Interpolation	**Section**: S2: P1.3.4, P1.4.8, P1.5.6 **Reading**: 1.5-1.6, 2.1 **Event:**
3	**Feb 9** Interpolation **Feb 11** Piecewise Polys	**Section**: S3: P2.1.1, P2.2.3, P2.3.3 **Reading:** 2.2, 2.3, 2.4.4, 3.1 **Event**: P1 Due 2/9
4	**Feb 16** Piecewise Polys **Feb 18** Piecewise Polys	**Section:** S4: P3.1.5, P3.2.3, P3.3.7, P3.3.9 **Reading**: 3.2, 3.3 **Event**:
5	**Feb 23** Quadrature **Feb 25** Quadrature	**Section**: S5: P4.2.2, P4.3.2, P4.4.2 **Reading:** 4.1-4.4 **Event**: P2 Due 2/23, Prelim I (2/24)
6	**Mar 1** Matrix Operations **Mar 3** Matrix Operations	**Section**: S6: P4.2.2, P4.3.2, P4.4.2 **Reading**: 5.1, 5.2, 5.4 **Event**:

7	**Mar 8** Linear Systems **Mar 10** Linear Systems	**Section:** S7: P5.1.4, P5.2.5, P5.4.3 **Reading:** 6.1, 6.2 **Event:**
8	**Mar 15** Linear Systems **Mar 17** Linear Systems	**Section:** S8: P6.1.2, P6.2.1, P6.2.4.P6.3.3, P6.3.8, P6.4.1, P6.4.4 **Reading**: 6.3, 6.4 **Event**: P3 Due 3/15

Spring Break

Week	Lecture	Notes
9	**Mar 29** Least Squares **Mar 31** Least Squares	**Section:** S9: P7.1.2, P7.1.5, P7.3.5 **Reading:** 7.1, 7.3.1, 7.3.2 **Event:**
10	**Apr 5** Optimization **Apr 7** Optimization	**Section:** S10: P8.1.5, P8.1.7, P8.2.2, P8.2.4 **Reading**: 8.1, 8.2 **Event:** P4 Due 4/5 Prelim II (4/8)
11	**Apr 12** Optimization **Apr 14** Optimization	**Section:** S11: P8.3.1, P8.4.6 **Reading:** 8.3, 8.4 **Event:**
12	**Apr 19** Eigenvalue Problems **Apr 21** Eigenvalue Problems	**Section:** S12: P8.3.1, P8.4.6 **Reading:** Moler (Chapter 10) **Event:** P5 Due 4/23
13	**Apr 26** Eigenvalue Problems **Apr 28** Eigenvalue Problems	**Section:** S13: TBA **Reading:** Moler (Chapter 10)

		Event:
14	**May 3** Applications **May 5** Applications	**Section:** TBA **Reading:** **Event:** P6 Due 5/7

Notes: S1--S13 are section assignments. P1--P6 are programming assignments. All readings for the first 11 weeks are from the *Introduction to Scientific Computing: A Matrix-Vector Approach Using Matlab.*

Section Assignments

| CS 322 Home |

There are thirteen Section Assignments. Each involves several book (or handout) problems that usually require a Matlab solution. Section Assignments are NOT submitted for grading. They are discussed in the weekly sections and focus attention on important concepts and problem-solving strategies. Doing the Section Assignments is a good way to prepare for the exams and the programming assignments.

You can see the solution simply by clicking on the problem.

It is advised that you spend an appropriate amount of time on these problems before you even think about looking at the solutions.

S1: P1.1.2, P1.2.5, P1.3.3
S2: P1.3.4, P1.4.8, P1.5.6
S3: P2.1.1, P2.2.3, P2.3.3
S4: P3.1.5, P3.2.3, P3.3.7, P3.3.9
S5: P4.2.2, P4.3.2, P4.4.2
S6: P5.1.4, P5.2.5, P5.4.3,
S7: P6.1.2, P6.2.1, P6.2.4
S8: P6.3.3, P6.3.8, P6.4.1, P6.4.4
S9: P7.1.2, P7.1.5, P7.3.5
S10: P8.1.5, P8.1.7, P8.2.2, P8.2.4
S11: P8.3.1, P8.4.6
S12: Handout
S13: Handout

Programming Assignments

P1	Handout	Solution
P2	Handout	Solution
P3	Handout	Solution
P4	Handout	Solution
P5	Handout	Solution
P6	Handout	Solution

All assignments are submitted electronically via the Course Management System. Click here for details.

Uphold the Academic Integrity Policy.

Lecture Handouts and Examples

Lecture 10 Wednesday, February 25

```
% Script Lec10
% Looks at second derivative continuity of pwcubic hermite interpolant
bigscreen
close all
for s2 = [-10  -5  -1  -5/6  1   5  10 100]
    figure
    ShowSmallSpline(s2)
    pause(2)
end

  function ShowSmallSpline(s2)
% Displays the piecewise cubic hermite interpolant of the data
%
%                      (0,1,2), (1,3,s2), (2,-1,1)
%
% On [0,1] we have the cubic q1(z) = 1 + 2z +(s2-2)z^2(z-1) which
satisfies
%
%         q1(0) = 1, q1'(0) = 2, q1(1) = 3, q1'(3) = s2
%
% It has second derivative
%
%         q1"(z) = 2(s2-2) + 6(s2-2)z


% The cubic q2(z) = 3 + s2(z-1) + (-4 - s2)(z-1)^2  + (9+s2)(z-1)^2(z-
2)
%
%         q2(1) = 3, q1'(1) = s2,  q2(2)=-1, q2'(2) = 1
%
% It has second derivative
%
%         q2"(z) = 2(-4-s2) + (9+s2)(4(z-1) + 2(z-2))
%
% The equation q1"(1) = q2"(1) says
%
%               2(s2-2) +6(s2-2)  =  2(-4-s2) + (9+s2)(-2)
```

```
%
% i.e., s2 = 5/6. When this is the case we have a cubic spline
interpolant.

close all

% Evaluate the two local cubics...
zL = linspace(0,1);
q1Vals = 1 + 2*zL + (s2-2)*(zL.^2).*(zL-1);
zR = linspace(1,2);
q2Vals = 3 + s2*(zR-1) + (-4-s2)*(zR-1).^2  + (9+s2)*((zR-1).^2 ).*(zR-
2);

% Display
subplot(2,1,1)
plot(zL,q1Vals,zR,q2Vals,[0 1 2],[1 3 -1],'o')
legend('q1(z)','q2(z)')
title(sprintf('An n=2 piecewise cubic hermite interpolant with s2 =
%5.2f',s2),'fontsize',18)

% Display their second derivatives
subplot(2,1,2)
plot(zL,2*(s2-2) + 6*(s2-2)*zL, zR,2*(-4-s2) + (9+s2)*(4*(zR-1)+2*(zR-
2)))
legend('q1"(z)','q2"(z)')
title('It''s second derivative...','fontsize',18)
```

Lecture 4. Wednesday, February 4

```
Read about InterpV and HornerV in Chapter 2 of SCMV.
% Lec4A
% Plots random cubic interpolants of sin(x) on [0,2pi].
% Uses the Vandermonde method.

close all
x0 = linspace(0,2*pi,100)';
y0 = sin(x0);
for eg=1:30
   x = 2*pi*sort(rand(4,1));
   y = sin(x);
   a    = InterpV(x,y);
   pVal = HornerV(a,x0);
   plot(x0,y0,x0,pVal,'--',x,y,'r*')
   axis([0 2*pi -2 2])
   title('A Random Cubic Interpolant of sin(x) on [0,2\pi]' )
   legend('sin(x)','Cubic Interpolant')
   grid on
   shg
   pause
end
clc
grid off
```

%%%

```
% Lec4B
% Equally spaced interpolants of sin(x) on [0,2pi].
% Uses the Vandermonde method.

close all
x0 = linspace(0,2*pi,100)';
y0 = sin(x0);
for deg=1:20
   x = linspace(0,2*pi,deg+1)';
   y = sin(x);
   a    = InterpV(x,y);
   pVal = HornerV(a,x0);
   subplot(2,1,1)
   plot(x0,y0,x0,pVal,'--',x,y,'r*')
   axis([0 2*pi -2 2])
   s = sprintf('deg %1d Interpolant of sin(x)',deg);
   title(s)
   legend('sin(x)','Polynomial Interpolant')
   subplot(2,1,2)
   absErr = abs(pVal-y0);
   plot(x0,absErr)
   axis([0 2*pi -inf inf])
   s = sprintf('Approx maximum absolute error = %5.1e',max(absErr));
   title(s,'fontsize',24)
   shg
   pause
end
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Lec4c
% Equally spaced interpolants of 1/(1+25x^2) on [-1 1].
% Uses the Vandermonde method.

close all
f = inline('1./(1+25*x.^2)');
x0 = linspace(-1,1,100)';
y0 = feval(f,x0);
for deg=1:20
   x = linspace(-1,1,deg+1)';
   y = feval(f,x);
   a    = InterpV(x,y);
   pVal = HornerV(a,x0);
   subplot(2,1,1)
   plot(x0,y0,x0,pVal,'--',x,y,'r*')
   s = sprintf('deg %1d Interpolant of 1/(1+25x^{2})',deg);
   title(s,'fontsize',24)
   legend('1/(1+25x^{2})','Polynomial Interpolant',0)
   subplot(2,1,2)
   absErr = abs(pVal-y0);
   plot(x0,absErr)
   s = sprintf('Approx maximum absolute error = %5.1e',max(absErr));
   title(s,'fontsize',24)
   shg
   pause
end
```

```
clc
```

%%%

```
% Script Lec4d
  close all

  ShowInterp('sin',0,2*pi,5)

  figure
  g = inline('sin(2*x)');
  ShowInterp(g,0,2*pi,5)

  figure
  ShowInterp(@sin,0,2*pi,5)
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```
  function ShowInterp(fName,a,b,deg)
% fName is a string that names a function f defined on
% [a,b]. Assume a < b
% deg is a positive integer
% Displays the equal-spacing, degree d polynomial interpolant
% of f on [a,b].

% Display the function over  the interval...
  x = linspace(a,b)';
  y = feval(fName,x);
  plot(x,y);
  axis([a b -inf inf])

% Display the points of interpolation
  hold on
  x0 = linspace(a,b,deg+1)';
  y0 = feval(fName,x0);
  plot(x0,y0,'or')

% Generate the interpolant
  a = InterpV(x0,y0);

% Evaluate the interpolant across the interval and display.
  pVals = HornerV(a,x);
  plot(x,pVals,'r')
  hold off
  shg
```

Lecture 3. Monday, February 2

%%

```
% Script Lec3A

x = linspace(-7,7);
y = MyExp(x);
shg
```

%%%

```
  function y = MyExp(x)
% x is a vector
% y has the same length and orientation with
% y(i) = exp(x(i)), i = 1:length(x)

y = ones(size(x));    % y = vector of all 1's, same size & shape as x
termIndex = 1;
termNext  = x;
yNext = y + termNext;
while(any(y~=yNext))

    % Iterate as long as the addition of the next term makes a
difference
    y = yNext;
    termIndex = termIndex + 1;
    termNext = (termNext.*x)/termIndex;
    yNext = yNext + termNext;

    % Display the relative error
    relErr = abs(yNext - exp(x))./exp(x);
    % Add eps to relErr so don't evaluate log at zero.
    semilogy(x,relErr+eps,[min(x) max(x)] ,[10^-15 10^-15],'r')
    title(sprintf('Number of terms = %1d',termIndex))
    ylabel('Relative Error')
    xlabel('x')
    shg
    pause

end
```

%%

```
% Script Lec3B
% Explores running sums in Taylor expansion for exp(x)

close all
x = -7;
nTerms = 30;

sum  = zeros(1,nTerms);
term = zeros(1,nTerms);

sum(1)  = 1 + x;
term(1) = x;

for k=2:nTerms
    term(k) = x*term(k-1)/k;
    sum(k) = sum(k-1) + term(k);
end
semilogy(1:nTerms,abs(sum),1:nTerms,abs(term))
title(sprintf('exp(%5.1f) via taylor series',x))
xlabel('k')
legend('1 + term(1) + ... + term(k)',' | term(k) |')
shg
```

Exams and Review Sessions

Note: The syllabi for the exams are defined in terms of the Lectures (L1-L28), the Section assignments (S1-S12), and the Programming Assignments (P1-P5). Of course, to understand this material you should be familiar with the associated readings from the course text.

Review Session I

Time and Place: There will either be a Monday (2/23) night review session or we will use the Monday (2/23) lecture. Old exam questions of interest include problems 1 and 4 in this old exam and problems 1 and 3 in this other exam.

Prelim I

Time and Place: Tuesday, February 24, 7:30PM. Phillips 101.

Syllabus: Assignments 1 and 2. All lecture material through Monday, February 16, an all section problems through P3.1.5.

You must contact Gun by the end of Friday, Feb 20 if you have another exam at the same time so that we can schedule an appropriate make-up. Include your name, ID, and the conflicting course in the message.

Review Session II

Time and Place: Wednesday 7:30-9. Location Phillips101.

Here are some problems to look over that "showed up" in the Spring 02 edition of the course:

(a) Look at problems 5 and 6 in here. (Solutions)

(b) Look at problems 2(a), 4, and 5 in here.

(c) Look at problems 1,2, and 3 in here.

Prelim II

Time and Place: Thursday, April 8, 7:30-9:00PM. Phillips 101

Syllabus: A3, A4, and section problems S4-S9.

You must contact Gun by the end of Monday, April 5 if you have another exam at the same time so that we can schedule an appropriate make-up. Include your name, ID, and the conflicting course in the message.

Final Review Session

Time and Place: Tuesday, May 18, 7:30-9 pm, Location TBA

Final Exam

Time and Place: Friday, May 21, 9:00-11:30AM, Ives 305

Syllabus:

You must contact X by the end of Y if you have another exam at the same time so that we can schedule an appropriate make-up. Include your name, ID, and the conflicting course in the message. "Leaving early for the summer" does NOT qualify you for a make-up final.

Administrative Policies

| CS 322 Home |

Course Grade Computation

Course grades are a function of the scores that you receive on the programming assignments and the exam

Programming Assignments	=	30%
Prelim 1	=	20%
Prelim 2	=	20%
Final Exam	=	30%

The six programming assignments are equally weighted.

The Course Management System

All programming assignments will be submitted electronically using the Course Management System (CMS). For details click here.

Late Submissions

Late submissions will be noted and will adversely affect your final grade.

Facilities

MATLAB is available on all public CIT Machines. The student edition of MATLAB (Mac or PC Version) is available in the stores.

The CIT lab hours are are posted here. There are instructional labs in MVR, Stimson Hall, and Upson. There are general purpose labs in Uris Library, Noyes, Dickson, Carpenter, and Purcell Union.

Before CS 322

The course assumes that you have completed the freshman-sophomore calculus sequence. We rely heavily on matrix algebra so Math 294 and Math 221 are particularly important.

From the computing point of view, we assume a CS 100 level of expertise. Note that CS 100 includes a brief introduction to Matlab. Although our Matlab presentation will be self-contained, its pace in the first few weeks will be brisk because it will be assumed that once upon a time you were familiar with the subset of Matlab that is taught in CS 100.

We also mention an alternative course, CS 321 which teaches 322 material using examples from computational molecular biology to drive the discussion.

After CS 322

Other courses in scientific computing:

- CS 421 Numerical Analysis
- CS 522 Computational Tools and Methods for Finance
- CS 621 Matrix Computations
- CS 622 Numerical Optimization and Nonlinear Equations
- CS 624 Numerical Solution of Differential Equations
- CS 626 Computational Biology

Academic Integrity

| CS 322 Home |

Violations of the Cornell University Code of Academic Integrity occurring in Computer Science courses are taken very seriously by the Computer Science faculty. Therefore, it is necessary to impress upon students the gravity of violations of the Code. The following are excerpts from a longer version of the Cornell University Code of Academic Integrity. The exclusion of any part does not excuse ignorance of the Code.

Principle

Absolute integrity is expected of every Cornell student in all academic undertakings; he/she must in no way misrepresent his/her work fraudulently or unfairly advance his/her academic status, or be a party to another student;s failure to maintain academic integrity. The maintenance of an atmosphere of academic honor and the fulfillment of the provisions of this Code are the responsibilities of the students and faculty of Cornell University. Therefore, all students and faculty members shall refrain from any action that would violate the basic principles of this Code.

General Responsibilities

1. A student assumes responsibility for the content and integrity of the academic work he/she submits, such as papers, examinations, or reports.
2. A student shall be guilty of violating the Code and subject to proceedings under it if he/she:
    - knowingly represents the work of others as his/her own.
    - uses or obtains unauthorized assistance in any academic work.
    - gives fraudulent assistance to another student.
    - fabricates data in support of laboratory or field work.
    - forges a signature to certify completion or approval of a course assignment.
    - in any other manner violates the principle of absolute integrity.

Specific Remarks for Students in CS 322

Note: "You" in the following refers to "you and your partner" should you work with a partner.

The work you do in Computer Science courses is expected to be the result of your individual effort - the use of a computer in no way modifies the normal standards of the above Code. You may discuss work with other students, and give or receive "consulting" help from other students, but such permissible cooperation should never involve one student having in his or her possession a copy of all or part of another student's assignment - regardless of whether that copy is on paper, on a computer disk, or in a computer file. This implies that there is no legitimate reason to send a copy of a program from one computer account to another, or to be logged-on to another student's account.

Discussion of general strategy or algorithms is permissible, but you may not collaborate in the detailed development or actual writing of an assignment. It is also your responsibility to protect your work from unauthorized access. It is inadvisable to discard copies of your programs in public places. Students who live in dormitories are advised to be extra careful about leaving computers on and copies of output lying around.

Our experience has been that students tend to think less clearly about Academic Integrity when

- they start a programming assignment late and a sense of panic sets in.
- they see a desperate friend seeking inappropriate levels of help.

Be aware of the social forces that underlie these situations.

Penalties are administered on a case-by-case basis. However, here are some guidelines:

- First-time violation on a programming assignment results in a score that is the negative of the full credit score for the entire assignment.
- If there is a violation and it is determined that your partner is entirely to blame, then you will nevertheless receive a zero for the assignment because you and your partner have a joint responsibility to the Code.
- If you resubmit a program or a prelim for regrading and alter the original submission, then you will receive a course grade of F. (Note: a randomly selected subset of graded programs and exams will be photocopied before they are made available for pick-up.)
- Solution output must be consistent with the submitted solution program. If not, the entire assignment will receive the grade of zero. There are no acceptable excuses for the inadvertent submission of the wrong program. (E.g., "I was in a rush and just forgot to print out the final version."

Second violations result in a course grade of F.

Remember that you may end up with a permanent mark on your transcript and be subject to University disciplinary action should you violate the Code.

Contact a member of the course staff immediately if you suspect that there may be a Code violation associated with your programming assignment submission. It is FAR better to tell us about a possible infraction beforehand than for us to discover it during the grading process.

Interesting Links

- Information and WWW pages on various Software bugs (Ariane, Pentium, Mars Orbiter, Exocet, Aviation, Banking, SDI, NORAD,...)

http://wwwzenger.informatik.tu-muenchen.de/persons/huckle/bugse.html

On the Google Eigenproblem...

- Here's a paper on the second eigenvalue of the Google matrix.
- The original paper by the Google co-founders.
- A critical paper entitled "PageRank: Google's Original Sin"