SCIENTIFIC COMPUTING

FuSen Lin (林富森)

Department of Computer Science & Engineering
National Taiwan Ocean University

Introduction, Fall 2012

Chapter 0: Introduction

- Introduction to Scientific Computing
- Three Practical Examples
- Examples of MATLAB codes

Chapter 0: Introduction

- Introduction to Scientific Computing
- Three Practical Examples
- Examples of MATLAB codes

Chapter 0: Introduction

- Introduction to Scientific Computing
- Three Practical Examples
- Examples of MATLAB codes

What is Scientific Computing?

- To solve diversely mathematical problems which arise in science and engineering.
- To design robust algorithms for finding approximate solutions to the given problem with any powerful tools (software).
- Practical problems $\Longrightarrow$ Mathematical models $\Longrightarrow$ Numerical algorithms $\Longrightarrow$ (Reasonably) approximate solutions

What is Scientific Computing?

- To solve diversely mathematical problems which arise in science and engineering.
- To design robust algorithms for finding approximate solutions to the given problem with any powerful tools (software).
- Practical problems $\Longrightarrow$ Mathematical models $\Longrightarrow$ Numerical algorithms $\Longrightarrow$ (Reasonably) approximate solutions

What is Scientific Computing?

- To solve diversely mathematical problems which arise in science and engineering.
- To design robust algorithms for finding approximate solutions to the given problem with any powerful tools (software).
- Practical problems $\Longrightarrow$ Mathematical models $\Longrightarrow$ Numerical algorithms $\Longrightarrow$ (Reasonably) approximate solutions

The Objective of Learning Scientific Computing?

- To learn the potentialities of modern computational methods for solving numerical problems arisen in science and engineering.
- To give students an opportunity to hone their skills in programming and problem solving.
- To help students understand the important subject of *errors* that inevitably accompany scientific computing and to arm them with methods for detecting, predicting, and controlling these errors.
- To familiarize students with the intelligent use of powerful software systems such as MATLAB, Maple, and Mathematica.

The Objective of Learning Scientific Computing?

- To learn the potentialities of modern computational methods for solving numerical problems arisen in science and engineering.
- To give students an opportunity to hone their skills in programming and problem solving.
- To help students understand the important subject of *errors* that inevitably accompany scientific computing and to arm them with methods for detecting, predicting, and controlling these errors.
- To familiarize students with the intelligent use of powerful software systems such as MATLAB, **Maple**, and **Mathematica**.

The Objective of Learning Scientific Computing?

- To learn the potentialities of modern computational methods for solving numerical problems arisen in science and engineering.
- To give students an opportunity to hone their skills in programming and problem solving.
- To help students understand the important subject of *errors* that inevitably accompany scientific computing and to arm them with methods for detecting, predicting, and controlling these errors.
- To familiarize students with the intelligent use of powerful software systems such as MATLAB, **Maple**, and **Mathematica**.

The Objective of Learning Scientific Computing?

- To learn the potentialities of modern computational methods for solving numerical problems arisen in science and engineering.
- To give students an opportunity to hone their skills in programming and problem solving.
- To help students understand the important subject of *errors* that inevitably accompany scientific computing and to arm them with methods for detecting, predicting, and controlling these errors.
- To familiarize students with the intelligent use of powerful software systems such as MATLAB, **Maple**, and **Mathematica**.

The Goal of Scientific Computation

- Numerical computations Pursue
  - **Efficiency**: the computational speed of a problem.
  - **Liability**: the accuracy of computational results (within the given error tolerance).
  - **Stability**: the stability of an algorithm and its applicable range.
- if use different algorithms for a problem, we may have different accuracy, different speed, different applicable range.
- If we use the same algorithm with different procedures (coding) then the running time and accuracy may not be the same.
- We shall study and pursue the best algorithms.

The Goal of Scientific Computation

- Numerical computations Pursue
  - **Efficiency**: the computational speed of a problem.
  - **Liability**: the accuracy of computational results (within the given error tolerance).
  - **Stability**: the stability of an algorithm and its applicable range.
- if use different algorithms for a problem, we may have different accuracy, different speed, different applicable range.
- If we use the same algorithm with different procedures (coding) then the running time and accuracy may not be the same.
- We shall study and pursue the best algorithms.

The Goal of Scientific Computation

- Numerical computations Pursue
  - **Efficiency**: the computational speed of a problem.
  - **Liability**: the accuracy of computational results (within the given error tolerance).
  - **Stability**: the stability of an algorithm and its applicable range.
- if use different algorithms for a problem, we may have different accuracy, different speed, different applicable range.
- If we use the same algorithm with different procedures (coding) then the running time and accuracy may not be the same.
- We shall study and pursue the best algorithms.

The Goal of Scientific Computation

- Numerical computations Pursue
  - **Efficiency**: the computational speed of a problem.
  - **Liability**: the accuracy of computational results (within the given error tolerance).
  - **Stability**: the stability of an algorithm and its applicable range.
- if use different algorithms for a problem, we may have different accuracy, different speed, different applicable range.
- If we use the same algorithm with different procedures (coding) then the running time and accuracy may not be the same.
- We shall study and pursue the best algorithms.

The Goal of Scientific Computation

- Numerical computations Pursue
  - **Efficiency**: the computational speed of a problem.
  - **Liability**: the accuracy of computational results (within the given error tolerance).
  - **Stability**: the stability of an algorithm and its applicable range.
- if use different algorithms for a problem, we may have different accuracy, different speed, different applicable range.
- If we use the same algorithm with different procedures (coding) then the running time and accuracy may not be the same.
- We shall study and pursue the best algorithms.

The Goal of Scientific Computation

- Numerical computations Pursue
  - **Efficiency**: the computational speed of a problem.
  - **Liability**: the accuracy of computational results (within the given error tolerance).
  - **Stability**: the stability of an algorithm and its applicable range.
- if use different algorithms for a problem, we may have different accuracy, different speed, different applicable range.
- If we use the same algorithm with different procedures (coding) then the running time and accuracy may not be the same.
- We shall study and pursue the best algorithms.

The Goal of Scientific Computation

- Numerical computations Pursue
  - **Efficiency**: the computational speed of a problem.
  - **Liability**: the accuracy of computational results (within the given error tolerance).
  - **Stability**: the stability of an algorithm and its applicable range.
- if use different algorithms for a problem, we may have different accuracy, different speed, different applicable range.
- If we use the same algorithm with different procedures (coding) then the running time and accuracy may not be the same.
- We shall study and pursue the best algorithms.

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Problems in Scientific Computing

- Finding-root problems
- Interpolation problems
- The method of least squares
- Numerical Differentiation and Integration
- Spline Approximation
- Methods of Solving Linear Systems
- Matrix Computations
- Numerical solutions to ODE(Initial-Value Problems and Boundary-Value Problems)
- Numerical solutions to PDE (Euler's mehod, Multistep methods, Rung-Kutta methods, Finite Difference Methods, or Finite Element Method)
- Optimization problems
- Other computational problems

Three Practical Examples

- The finding-root Problem
- The electrical circuit problem
- The Least Squares Problem

A finding-root problem

Example

An electric power cable is suspended (at points of equal hight) from two towers that are 100 meters apart. The cable is allowed to dip 10 meters in the middle. How long is the cable? (It is known that the curve assumed by a suspended cable is a Catenary(懸垂線)).

Proof.

Let x be the distance between each tower and the center of two towers. When the y-axis passes through the lowest point of the cable, the curve can be expressed as

$$y = \lambda \cosh\left(\frac{x}{\lambda}\right).$$

Here λ is a parameter to be determined. (continuing$\cdots$)

A finding-root problem

Example

An electric power cable is suspended (at points of equal hight) from two towers that are 100 meters apart. The cable is allowed to dip 10 meters in the middle. How long is the cable? (It is known that the curve assumed by a suspended cable is a Catenary(懸垂線)).

Proof.

Let x be the distance between each tower and the center of two towers. When the y-axis passes through the lowest point of the cable, the curve can be expressed as

$$y = \lambda \cosh\left(\frac{x}{\lambda}\right).$$

Here λ is a parameter to be determined. (continuing$\cdots$) □

A finding-root problem (continuing$\cdots$)

Proof.

(Continuing$\cdots$) The conditions of this problem are $y(50) = y(0) + 10$. The problem becomes to solve the equation

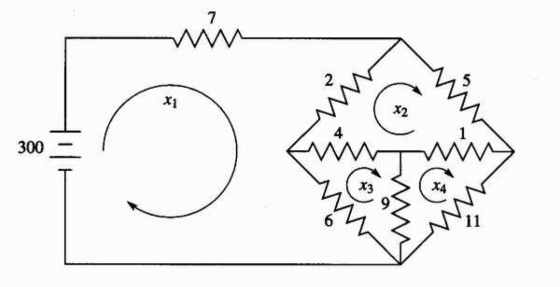
$$\lambda \cosh\left(\frac{50}{\lambda}\right) = \lambda + 10$$

for λ. Applying *the methods of locating roots*, we can obtain $\lambda = 126.632$.
From the equation of catenary, the length of cable (arc length) is easily computed by the method of calculus (numerical integration) to be **102.619** meters (see MATLAB DEMO). □

The Electrical Circuit Problem

Example

A simple **electrical network** contains a number of resistances and a single source of electromotive force (a battery) as shown in the following figure.



The Electrical Circuit Problem (2)

- Using **Kirchhoff's laws** and **Ohm's law**, we can write a system of equations that govern this circuit. If x_1, x_2, x_3, and x_4 are the loop currents as shown, then the equations are

$$\left\{\begin{array}{rcrcrcrcr} 15x_1 & - & 2x_2 & - & 6x_3 & & & = & 300 \\ -2x_1 & + & 12x_2 & - & 4x_3 & - & x_4 & = & 0 \\ -6x_1 & - & 4x_2 & + & 19x_3 & - & 9x_4 & = & 0 \\ & - & x_2 & - & 9x_3 & + & 21x_4 & = & 0 \end{array}\right.$$

- Systems of equations like this, even those that contain hundred unknowns, can be solved by using the methods of solving linear systems.

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -3 \end{bmatrix}$$

- The solution is $x_1 = 26.5$, $x_2 = 9.35$, $x_3 = 13.3$, $x_4 = 6.13$.

The Electrical Circuit Problem (2)

- Using **Kirchhoff's laws** and **Ohm's law**, we can write a system of equations that govern this circuit. If x_1, x_2, x_3, and x_4 are the loop currents as shown, then the equations are

$$\begin{cases} 15x_1 - 2x_2 - 6x_3 = 300 \\ -2x_1 + 12x_2 - 4x_3 - x_4 = 0 \\ -6x_1 - 4x_2 + 19x_3 - 9x_4 = 0 \\ -x_2 - 9x_3 + 21x_4 = 0 \end{cases}$$

- Systems of equations like this, even those that contain hundred unknowns, can be solved by using the methods of solving linear systems.

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -3 \end{bmatrix}$$

- The solution is $x_1 = 26.5$, $x_2 = 9.35$, $x_3 = 13.3$, $x_4 = 6.13$.

The Electrical Circuit Problem (2)

- Using **Kirchhoff's laws** and **Ohm's law**, we can write a system of equations that govern this circuit. If x_1, x_2, x_3, and x_4 are the loop currents as shown, then the equations are

$$\left\{ \begin{array}{rcrcrcrcr} 15x_1 & - & 2x_2 & - & 6x_3 & & & = & 300 \\ -2x_1 & + & 12x_2 & - & 4x_3 & - & x_4 & = & 0 \\ -6x_1 & - & 4x_2 & + & 19x_3 & - & 9x_4 & = & 0 \\ & - & x_2 & - & 9x_3 & + & 21x_4 & = & 0 \end{array} \right.$$

- Systems of equations like this, even those that contain hundred unknowns, can be solved by using the methods of solving linear systems.

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -3 \end{bmatrix}$$

- The solution is $x_1 = 26.5,\ x_2 = 9.35,\ x_3 = 13.3,\ x_4 = 6.13$.

The Least Squares Problem

Example

A physical experiment: Surface tension S in a liquid is known to be a linear function of temperature T. For a particular liquid, measurements have been made of the surface tension at certain temperatures. The results were

T	0	10	20	30	40	80	90	95
S	68.0	67.1	66.4	65.6	64.6	61.8	61.0	60.0

- How can the most probable values of the constants in the equation $S = aT + b$ be determined?
- Methods for solving such problems are called the least squares methods.

The Least Squares Problem

Example

A physical experiment: Surface tension S in a liquid is known to be a linear function of temperature T. For a particular liquid, measurements have been made of the surface tension at certain temperatures. The results were

T	0	10	20	30	40	80	90	95
S	68.0	67.1	66.4	65.6	64.6	61.8	61.0	60.0

- How can the most probable values of the constants in the equation $S = aT + b$ be determined?
- Methods for solving such problems are called the least squares methods.

The Least Squares Problem

Example

A physical experiment: Surface tension S in a liquid is known to be a linear function of temperature T. For a particular liquid, measurements have been made of the surface tension at certain temperatures. The results were

T	0	10	20	30	40	80	90	95
S	68.0	67.1	66.4	65.6	64.6	61.8	61.0	60.0

- How can the most probable values of the constants in the equation $S = aT + b$ be determined?
- Methods for solving such problems are called the least squares methods.

The Least Squares Problem

Example

A physical experiment: Surface tension S in a liquid is known to be a linear function of temperature T. For a particular liquid, measurements have been made of the surface tension at certain temperatures. The results were

T	0	10	20	30	40	80	90	95
S	68.0	67.1	66.4	65.6	64.6	61.8	61.0	60.0

- How can the most probable values of the constants in the equation $S = aT + b$ be determined?
- Methods for solving such problems are called the least squares methods.

Solution of Example 1

```
% solution of example 1
syms x   %
f = inline('x*cosh(50/x)-x-10');
lambda = fzero(f,100)   % lambda = 126.6324
r=fzero(x.*cosh(50./x)-x-10,130)  % finding root
syms a x  %
a = 126.6324*cosh(x/126.6324);
s = (1+diff(a).^2).^0.5; % set up the arc length function
I = vpa(int(s,-50,50))   % find the length by integration
```

Solution of Example 3

```
% Solution of Example~3 for the least squares problem
% An example of Script Files

 xdata = [0, 10, 20, 30, 40, 80, 90, 95];  %
 ydata = [68.0, 67.1, 66.4, 65.6, 64.6, 61.8, 61.0, 60.0];
 plot(xdata, ydata, 'o')           % plot the data
 ls = polyfit(xdata, ydata, 1); % call polyfit.m function
 a = ls(1);  b = ls(2);
 d = a*xdata + b - ydata;
 phi = sum(d.^2);
 x = 0:5:100;
 y = polyval(ls, x);               % compute the values of y
 hold on
 plot(x, y,'-r')                   % plot the curve of y
 disp('[a, b,  phi] =   ')
 [a, b, phi]
```

Function Files

```
function y = logsrs1(x, n)

% Date: 3/12/2001, Name:   Fusen F. Lin
% This function computes log(2) by the series,
% log(1+x)= x-x^2/2+x^3/3-x^4/4+x^5/5-..., for n terms.
% Input : real x and integer n. Notice that:
%          the input x=1 to get log(2).
% Output: the desired value log(1+x).

tn = x;                      % The first term.
sn = tn;                     % The n-th partial sum.
for k = 1:1:n-1,             % Use for-loop to sum the series.
   tn = -tn*x*k/(k+1);       % compute it recursively.
   sn = sn + tn;
end
y = sn;                      % Output the final sum.
```

Some Web Sites

- The MathWorks, Inc. (MATLAB): http://www.mathworks.com.
- **Massachusetts Institute of Technology**: Department of EECS: http://www.eecs.mit.edu/grad/
- MIT Open Courses: http://ocw.mit.edu/index.html
  Chinese Web Sites of Open Courses: http://www.myoops.org/twocw/mit/index.htm
- Department of Mathematics, MIT: http://math.mit.edu/research/index.html
- **The Computing Laboratory of Oxford University**: http://web.comlab.ox.ac.uk/oucl/
- **Department of Computer Science, Cornell University**: http://www.cs.cornell.edu/cv/

Some Web Sites

- The MathWorks, Inc. (MATLAB): http://www.mathworks.com.
- **Massachusetts Institute of Technology**: Department of EECS: http://www.eecs.mit.edu/grad/
- MIT Open Courses: http://ocw.mit.edu/index.html Chinese Web Sites of Open Courses: http://www.myoops.org/twocw/mit/index.htm
- Department of Mathematics, MIT: http://math.mit.edu/research/index.html
- **The Computing Laboratory of Oxford University**: http://web.comlab.ox.ac.uk/oucl/
- **Department of Computer Science, Cornell University**: http://www.cs.cornell.edu/cv/

Some Web Sites

- The MathWorks, Inc. (MATLAB): http://www.mathworks.com.
- **Massachusetts Institute of Technology**: Department of EECS: http://www.eecs.mit.edu/grad/
- MIT Open Courses: http://ocw.mit.edu/index.html
  Chinese Web Sites of Open Courses:
  http://www.myoops.org/twocw/mit/index.htm
- Department of Mathematics, MIT: http://math.mit.edu/research/index.html
- **The Computing Laboratory of Oxford University**: http://web.comlab.ox.ac.uk/oucl/
- **Department of Computer Science, Cornell University**: http://www.cs.cornell.edu/cv/

Some Web Sites

- The MathWorks, Inc. (MATLAB): http://www.mathworks.com.
- **Massachusetts Institute of Technology**: Department of EECS: http://www.eecs.mit.edu/grad/
- MIT Open Courses: http://ocw.mit.edu/index.html Chinese Web Sites of Open Courses: http://www.myoops.org/twocw/mit/index.htm
- Department of Mathematics, MIT: http://math.mit.edu/research/index.html
- **The Computing Laboratory of Oxford University**: http://web.comlab.ox.ac.uk/oucl/
- **Department of Computer Science, Cornell University**: http://www.cs.cornell.edu/cv/

Some Web Sites

- The MathWorks, Inc. (MATLAB): http://www.mathworks.com.
- **Massachusetts Institute of Technology**: Department of EECS: http://www.eecs.mit.edu/grad/
- MIT Open Courses: http://ocw.mit.edu/index.html Chinese Web Sites of Open Courses: http://www.myoops.org/twocw/mit/index.htm
- Department of Mathematics, MIT: http://math.mit.edu/research/index.html
- **The Computing Laboratory of Oxford University**: http://web.comlab.ox.ac.uk/oucl/
- **Department of Computer Science, Cornell University**: http://www.cs.cornell.edu/cv/

Some Web Sites

- The MathWorks, Inc. (MATLAB): http://www.mathworks.com.
- **Massachusetts Institute of Technology**: Department of EECS: http://www.eecs.mit.edu/grad/
- MIT Open Courses: http://ocw.mit.edu/index.html
  Chinese Web Sites of Open Courses: http://www.myoops.org/twocw/mit/index.htm
- Department of Mathematics, MIT: http://math.mit.edu/research/index.html
- **The Computing Laboratory of Oxford University**: http://web.comlab.ox.ac.uk/oucl/
- **Department of Computer Science, Cornell University**: http://www.cs.cornell.edu/cv/