HOMEWORK
Lessons in Scientific Computing: Numerical Mathematics, Computer Technology, and Scientific Discovery

1 Analytical & Numerical Solutions

1.1. Survey of computing background

(a) Have you taken any courses on numerical methods or scientific computing? If yes, which course(s)?

(b) What operating system do you commonly use?

- ☐ Linux
- ☐ Mac OS
- ☐ Windows
- ☐ Other (specify):

(c) What programming languages do you know?

- ☐ None
- ☐ C
- ☐ C++
- ☐ Fortran
- ☐ Java
- ☐ Python
- ☐ R
- ☐ Other (specify):

(d) What software tools do you use?

- ☐ IDL
- ☐ Matlab or Octave
- ☐ Mathematica

☐ Emacs editor
☐ vi or vim editor
☐ Microsoft Excel, Google Sheets, or similar spreadsheet software
☐ IPython
☐ Gnuplot
☐ awk
☐ Perl
☐ Other (specify):

2 A Few Concepts from Numerical Analysis

2.1. (a) Plot the function $f(x) = 3\pi^4 x^2 + \ln((x-\pi)^2)$ in the range 0 to 4.
(b) Proof that $f(x)$ has two roots, $f(x) = 0$.
(c) Estimate the distance between the two roots.
(d) Plot the function in the vicinity of these roots. Does the graph of the function change sign, as it must?

2.2. Show that when the Newton method is used to calculate the square root of a number $a \geq 0$, it converges for all initial conditions $x_0 > 0$.

2.3. Degenerate roots of polynomials are numerically ill-conditioned. For example, $x^2 - 2x + 1 = 0$ is a complete square $(x-1)^2 = 0$ and has the sole degenerate root $x = 1$. Suppose there is a small error in one of the three coefficients, $(1-\delta_2)x^2 - 2(1+\delta_1)x + (1-\delta_0) = 0$. Perturbing each of the three coefficients independently and infinitesimally, determine how the relative error ϵ of the root depends on the relative error of each coefficient.

2.4. Calculate the condition numbers for addition, subtraction, multiplication, and division of two numbers of the same sign and based on relative error. When the relative input errors are ϵ_1 and ϵ_2, and the relative error of the output is ϵ_{12}, then a single condition number κ can be defined as $|\epsilon_{12}| = \kappa \max(|\epsilon_1|, |\epsilon_2|)$.

2.5. Consider the linear system $A\mathbf{x} = \mathbf{b}$ with

$$A = \begin{pmatrix} c & d \\ e & f \end{pmatrix}$$

(a) Calculate the solution x analytically/symbolically.
(b) Under what conditions is the solution well-conditioned, badly conditioned, and ill-conditioned?

3 Roundoff & Number Representation

3.1. The iteration $x_{n+1} = 4x_n(1-x_n)$ is extremely sensitive to the initial value as well as to roundoff. Yet, thanks to the IEEE 754 standard, it is possible to reproduce the exact same sequence on many platforms. Calculate the first 1,010 iterations with initial conditions $x_0 = 0.8$ and $x_0 = 0.8+10^{-15}$. Use double (8-byte) precision. Submit the program as well as the values $x_{1000} \ldots x_{1009}$ to 9 digits after the comma. We will compare the results in class.

3.2. The expression for gravitational acceleration is GMr_i/r^3, where G is the gravitational constant 6.67×10^{-11} $\mathrm{m}^3/\mathrm{kg\,s}$, M is the mass of the sun 2×10^{30} kg, $r = 150 \times 10^9$ m is the distance between the Sun and Earth, and $i = 1, 2, 3$ indexes the three spatial directions. We cannot anticipate in which order the computer will carry out these floating-point operations. What are the worst possible minimum and maximum exponents of an intermediate result? Could 4-byte IEEE or 8-byte IEEE floating-point numbers overflow or underflow?

3.3. *hypot function:* The hypotenuse of a triangle is given by

$$c = \sqrt{x^2 + y^2}$$

A problem with this expression is that x^2+y^2 might overflow or underflow, even when c does not. Find a way to calculate c that circumvents this problem.

3.4. The following formulae may incur large roundoff errors: a) $x - \sqrt{x}$, and b) $\cos^2 x - \sin^2 x$. Identify values of x for which they are sensitive to roundoff, and suggest an alternative formula for reach which avoids this problem.

4 Programming Languages & Tools

4.1. If you do not know any programming language yet, learn one. The following chapters will involve simple programming exercises.

4.2. *Gauss circle problem:* As a basic programming exercise, write a program that calculates the number of lattice points within a circle as a function of radius. In other words, how many pairs of integers (m, n) are there such that $m^2+n^2 \leq r^2$. A finite resolution in r is acceptable. Submit the souce code of the program and a graph of the results.

4.3. With a programming language of your choice, read comma separated numbers from a file with an unknown number of rows. Submit the program source code.

4.4. Programming exericse: Given a set of floating-point numbers between 0 and 10, sort them into bins of width 1, without explicitly looping through the bins for each number. Submit the program source code.

4.5. *Newton fractal:* As a programming exercise, an curious example of how complicated the domain of convergence for Newton's method can be is $z^3 - 1 = 0$ in the complex plane. The solutions to the equation are the cubic roots of $+1$. The domain of attraction for each of the three roots is a fractal. Write a program that produces this fractal, where $z = x + iy$ is the starting point of the iteration. If after 2000 steps the iteration has reached 1 or is close to 1, color the coordinate black, otherwise white. Hence, black will represent the set of initial conditions that converge to $+1$. Plot the results for $-1 \leq x \leq 1$, $-1 \leq y \leq 1$, at a resolution of 0.002.

5 Sample Problems; Building Conclusions

5.1. For the kicked rotator ($\alpha_{n+1} = \alpha_n + \omega_n T$, $\omega_{n+1} = \omega_n + K \sin \alpha_{n+1}$), determine how fast the energy $E = \omega^2/2$ grows with time n. Consider the ensemble average and large kicking strength ($K \geq 4$). You need to consider only one value of K; the answer is supposedly independent of K beyond this value. Take initial values that lead to chaotic motions, that is, find a way to exclude integrable solutions from the average. The function you fit should have a physically reasonable asymptotic behavior.

Submit: 1) a plot of the ensemble-averaged energy as a function of time and a functional fit to this graph, 2) a description of how you avoided the integrable solutions, and 3) basic info such as the value of K used, how many initial values were chosen and how they were chosen.

5.2. Solve Kepler's equation with Newton's method. The Kepler equation is $M = E - e \sin E$, where the so-called "mean anomaly" M is linear in time, E is the eccentric anomaly, and e is the eccentricity of the orbit. The distance from the sun is $r = a(1 - e \cos E)$, so solving Kepler's equation provides r for a given time.

(a) Write a program that solves Kepler's equation for any M. Use a reasonable criterion to decide how many iterations are necessary.

(b) Test the program with exact solutions. Use $e = 0.9671$, appropriate for Halley's comet.

(c) Calculate the time average of $(a/r)^2$ to at least three significant digits. The mean solar flux is proportional to this quantity. (This average can be obtained analytically, but the task here is to obtain it numerically.)

6 Approximation Theory

6.1. Show mathematically that a parabola $p(x)$ that passes through three equally spaced points $y_1 = p(-h), y_2 = p(0), y_3 = p(h)$ has slope $(y_3 -$

$y_1)/(2h)$ at the center point $x = 0$. In other words, the slope at y_2 does not depend on y_2.

6.2. Finite-difference expression for unequally spaced points.
   (a) Based on a stencil of three points $(0, x_1, x_2)$, find an approximation of the first derivative at $x = 0$, that is, find the coefficients for $f'(0) = c_0 + c_1 x_1 + c_2 x_2$
   (b) Determine the order of the error term
   (c) Implement and verify with a numerical convergence test that the finite-difference approximation converges.
   (d) Also demonstrate that it converges at the anticipated rate.

6.3. Show that in the neighborhood of a simple root, Newton's method converges quadratically.

6.4. Error in numerical integration
   (a) Implement a simple trapezoidal integrator.
   (b) Numerically evaluate $\int_0^1 \cos(2\pi(1 - x^2) + \frac{1}{2})dx$, and determine the order of convergence as a function of spatial resolution.
   (c) For $\int_0^1 \cos(2\pi(1-x^2))dx$, the integrand has vanishing first derivatives on both boundaries. Again determine the order of convergence.

6.5. Carry out Exercise 5.2 and add a convergence test for part c.

6.6. Derive a formula for the integral of a cubic polynomial in the interval 0 to b. Demonstrate that applying Simpson's rule to it will give the exact integral.

6.7. Show that for a vector y with n elements

$$\|y\|_\infty \leq \|y\|_2 \leq \|y\|_1 \leq \sqrt{n}\|y\|_2 \leq n\|y\|_\infty$$

6.8. Learn how to carry out spline interpolation within a software tool or language of your choice, and demonstrate that the function in Figure 6.2, $f(x) = 1/(1 + 25x^2)$ on the interval -1 to 1, can be approximated without the large oscillations.

7 Other Common Computational Methods

7.1. Derive the equations for linear regression without intercept, $y = kx$, i.e. given data pairs (x_i, y_i), $i = 1, ..., N$, find the equation for k that minimizes the sum of quadratic deviations, $\sum_{i=1}^{N}(y - y_i)^2$.

7.2. Weighting proportional with distance improves the robustness of a fit. This method minimizes $\sum_i |y_i - a - bx_i|$, where x_i and y_i are the data and a and b are, respectively, the intercept and slope of a straight line. Show that finding the parameters a and b requires nonlinear root-finding in only one variable.

7.3. Show that the Fourier coefficients minimize the square-deviation between the function and its approximation. Use the trigonometric representation.

7.4. (a) Calculate the Fourier coefficients of the continuous but non-periodic function $f(x) = x/\pi$, on the interval $[-\pi, \pi]$. Show that their absolute values are proportional to $1/k$.

(b) Any non-periodic continuous function can be written as the sum of such a ramp and a periodic function. From this, argue that continuous non-periodic functions have Fourier coefficients that decay as $1/k$.

(c) Calculate the Fourier coefficients of a tent-shaped function, $f(x) = 1 - |x|/\pi$ on $[-\pi, \pi]$, which is continuous and periodic, and show that their absolute values are proportional to $1/k^2$.

7.5. Consider a numerical solver for the differential equation $y' = -ay$ that uses the average of a forward time step and a backward time step:

$$\frac{y^{n+1} - y^n}{h} = -a\frac{y^n + y^{n+1}}{2}$$

(a) Derive the criterion for which the numerical solution will not oscillate.

(b) Show that this method is one order more accurate than either the forward or the backward-step method.

7.6. Using a symbolic computation software of your choice, calculate the 10th derivative of $1/(1+25x^2)$ at $x = 0$. (This function was used in Figure 6.2.) The main purpose of this exercise is to learn how to use a computer algebra system and some of its syntax.

7.7. (a) Using a symbolic computation software of your choice, find the roots of $x^5 - 7x^4 + 2x^3 + 2x^2 - 7x + 1 = 0$. (This equation was used in the Brainteaser of chapter 1.)

(b) With or without the assistance of software, verify that the answers are correct.

7.8. For the diagrammatic expansion method for $e^{-\beta u}$, described in section 7.4:

(a) Write down all distinct diagrams up to second order in β with N particles.

(b) Determine the prefactor for each of the terms.

8 Performance Basics & Computer Architectures

8.1. Consider the dot product of two vectors

$$c = \sum_{i=1}^{N} x_i y_i$$

(a) If x and y are 64-bit variables and $N = 1000$ how many bytes of memory are consumed by x and y combined?

(b) How many floating-point operations are necessary to form the dot product?

(c) What is the ratio of data transferred from memory to processor to the number of floating-point operations, in units of byte per FLOP?

8.2. To find the minmum pairwise distance between a large numer of points with coordinates (x, y), we can calculate the distances $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ and then take the minimum. But because the square root is an expensive function, it is faster to only calculate the square of the distances and take the square root only once the minimum is found. In a programming language of your choice:

(a) implement both versions,

(b) find a way to measure the execution time, and

(c) determine by what factor the computation time is reduced.

8.3. Slowdown from excessive output to display: Take a simple computation, such as $x_{n+1} = 4x_n(1 - x_n)$, and compute the first ten million iterations with $x_0 = 0.8$.

(a) Output the result of each iteration to the display and measure the execution time. Then output only the last iteration and measure the execution time. Since both runs executed the same number of FLOPs, the difference in runtime is due to output to the display.

(b) Instead of displaying the output, write it to a file. How does the execution time compare to the other two cases?

8.4. (a) One way or another, find out the following hardware specifications of your computer: the number of CPU cores, the total amount of memory, and the memory of each level of cache.

(b) Using a programming language of your choice, determine the number of bytes consumed by a typical floating-point number and a default integer. In each language, a dedicated command is available that returns this value.

9 High-Performance & Parallel Computing

9.1. The fragment of a Fortran program below finds the distance between the nearest pair among N points in two dimensions. This implementation is wasteful in computer resources. Can you suggest at least 4 simple changes that should improve its computational performance? (You are not asked to verify that they improve performance.)

```
! ... x and y have been assigned values earlier ...
m=1E38
```

```
do i=1,N
   do j=1,N
      if (i==j) cycle  ! skips to next iteration
      r(i,j) = sqrt((x(i)-x(j))**2. + (y(i)-y(j))**2.)
      if (r(i,j)<m) m=r(i,j)
  enddo
enddo
! m is minimum distance
```

9.2. Learn how to ingest a command line argument into a program, e.g. `myprog 7` should read an integer number or single character from the command line. Then use this input argument to read file `in.7` and write a file named `out.7`. This is one approach to run the same program with many different input parameters. Submi the source code.

9.3. Learn how to profile your code, that is, obtain the fraction of time consumed by each function in your program. Take or write an arithmetically intensive program that takes at least 30 seconds to execute, then find out which function or command consumes most of the time. Submit an outline of the steps taken and the output of the profiler.

9.4. In a programming language of your choice, learn how to use multiple cores on your CPU. Some may do this automatically, others will need explicit program and/our launch instructions. Verify that the program runs on multiple cores simultaneously.

10 The Operation Count; Numerical Linear Algebra

10.1. How many times is the content of the innermost loop executed?

```
for (i=0; i<N; i++) {
  for (j=i+1; i<N; j++) {
    for (k=j+1; k<N; k++) {
      ...
    }
  }
}
```

(If unfamiliar with the syntax of a C loop, see Chapter 4).

10.2. Order the following functions from lowest to highest. Consider O as a tight bound. If any are of the same order, indicate which:
N, 3^N, $N \log N$, $2N - N^3 + 3N^5$, $N + \log N$, N^2, $N!$, $(2N)!$, $(\log N)^2$, $\sqrt{N} + \log N$, $\log_2 N$, $\ln N$, $\log(5N^2 + 7)$, $\log(\log N)$, 7

10.3. Show that

(a) $(2N)!/(2^N N!) = O(N!)$
(b) $\log(N!) = O(N \log N)$
(c) $2^{O(\log N)} = O(N^k)$

10.4. Calculate the arithmetic intensity for the 3-dimensional gravitational N-body problem when evaluated with a simple nested double loop.

(a) Write down program code or pseudocode that calculates the acceleration of each body in 3D.
(b) Count the number of floating-point operations required for the evaluation, to leading order. Assume a square root operation is equivalent to 10 FLOPs.
(c) Count the number of bytes that need to be accessed from main memory. Assume floating-point numbers have 8 bytes.
(d) Take the ratio. Based on this ratio, do you think this calculation is floating-point limited or data-transfer limited?

11 Random Numbers & Stochastic Methods

11.1. Uniform distribution of points on sphere. For points that are distributed uniformly over the surface of a sphere (the same number of points per area), the geographic coordinates are not uniformly distributed. Here we seek to generate point coordinates that are uniformly distributed over a unit sphere.

(a) Derive the transformation necessary to calculate geographic latitude λ and longitude ϕ from random variables u, v that are uniformly distribution between 0 to 1.
(b) Implement a program that produces random points uniformly distributed over a sphere.
(c) Devise a way to validate the outcome.
(d) Validate your implementation with this test.

11.2. Generate a Maxwell distribution for velocity $|v| = \sqrt{v_1^2 + v_2^2 + v_3^2}$ by generating Gaussian distributions for each of the three velocity components with standard deviation 1. Verify that the standard deviation of the resulting Maxwell distribution is indeed what it should be. Include a convergence test that demonstrates that the standard deviation of $|v|$ and of each component v_i approaches the theoretical values as the number of randomly generated points increases.

11.3. A probability distribution of the form

$$p(x) = \frac{1}{\pi} \frac{a}{a^2 + x^2}$$

is known as Cauchy or Lorentzian distribution

(a) Use the transformation method to generate a Cauchy distribution from a uniform probability distribution

(b) Conduct a convergence test for the standard deviation, and show that it behaves as it should.

(c) To verify that the generated probability distribution is indeed of the correct shape, use a "Kolmogorov-Smirnov" test. It uses the maximum difference between the cumulative distribution of the ideal and the sample distribution.

$$D_N = \max_x |\hat{P}_N(x) - P(x)|$$

and compares it to a threshold value for that sample size. The cumulative distribution is defined by $P(x) = \int_{-\infty}^{x} p(x')dx'$, and similarly for the sample. The threshold is not all that easy to calculate itself, but for $N > 35$ and a significance level of 0.1, $D = 1.22/\sqrt{N}$ is a reasonable approximation.

12 Algorithms, Data Structures, & Complexity

12.1. *Hash Table.* A commonly used method for hashing positive integers is modular hashing: choose the array size M to be prime and for an integer value k, compute the remainder when dividing k by M. This is effective in dispersing the values evenly between 0 and $M - 1$.

Consider the hashing function (value)%7.

(a) Generate the indices for the following values: 13, 96, 16, 3, 11, 112, 23, 54, 42.

(b) To look up one of these 9 values, how many steps are necessary on average? Count calculation of the remainder as one step, and following a linked list counts as a second step.

12.2. Given a sequence of N real numbers sorted by size. Show that a search for a number in this array takes at most $O(\log N)$ steps.

12.3. The operation count T of a divide-and-conquer method is given by the recursive relation

$$T(N) = 2T(N/2) + O(N)$$

with $T(1) = O(1)$. Find an explicit expression for $T(N)$. Assume N is a power of 2.

13 Data

13.1. (a) Use wget, curl, Scrapy, or a similar tool to download monthly meteorological data from 1991 to 2017 from `ftp://aftp.cmdl.noaa.gov/data/meteorology/in-situ/mlo/`

(b) From the data obtained, extract the temperatures for 10m mast height and plot them. (Hint: awk is a tool that can accomplish this.)

13.2. Use awk, Perl, sed, or another utility to write a one-line command that replaces

(a) a sequence of four stars **** with -999.

(b) a sequence of stars of any length with -999.

13.3. What do the following commands do?

(a)
```
sed 's/^[ \t]*//'
```

(b)
```
awk '{s=0; for (i=1; i<=NF; i++) s=s+$i; print s}'
```

In Regular Expression `\t` stands for a tab or whitespace, and in awk the variable 'NF' is the number of fields (entries in a row).

14 Building Programs for Computation and Data Analysis

14.1. (a) Create a large file containing numbers >100 MiB

(b) Write a program that reads this file and measures the execution time

(c) Compress the file with one of the many available compression tools, such as zip or gzip. Determine the compression factor (ratio of file sizes).

(d) Write a program that can read the compressed file directly.

(e) Measure the reduction in read time and compare it with the reduction in file size.

14.2. Write a script that validates and merges a set of files. Suppose we have files with names `out.0` to `out.99`, i.e. 100 of them, and each has entries of the form

```
0 273.15
1 260.
```

Write a script that checks that the first column contains the same values in all files and that the entries of the second column are non-negative. Then merge the files in the order of the numerical value of their file extension (0,...,99), not their alphanumerical value (0,1,10,11,...,19,2,20,...). This can be accomplished with a Unix shell script, Python, or another scripting language. Validate that the script works, using a small number of input files.

15 A Crash Course on Partial Differential Equations

15.1. The heat or diffusion equation

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2}$$

describes the evolution of temperature $f(x,t)$ for a (constant) thermal diffusivity D.

(a) Carry out a stability analysis for the finite-difference scheme

$$\frac{f_j^{n+1} - f_j^n}{k} = D\frac{f_{j+1}^n - 2f_j^n + f_{j-1}^n}{h^2}$$

and derive the condition for numerical stability.

(b) The backward time difference

$$\frac{f_j^{n+1} - f_j^n}{k} = D\frac{f_{j+1}^{n+1} - 2f_j^{n+1} + f_{j-1}^{n+1}}{h^2}$$

leads to an implicit scheme. Show that this scheme is unconditionally stable.

(c) The Crank-Nicolson method for the heat equation uses the average of the spatial derivatives evaluated at time steps n and $n+1$:

$$\frac{f_j^{n+1} - f_j^n}{k} = \frac{1}{2}D\frac{f_{j+1}^n - 2f_j^n + f_{j-1}^n}{h^2} + \frac{1}{2}D\frac{f_{j+1}^{n+1} - 2f_j^{n+1} + f_{j-1}^{n+1}}{h^2}$$

Such a superposition of a fully-implicit and an explicit method is called semi-implicit. Show that the Crank-Nicolson method is unconditionally stable.

(d) Show that the time discretization error for the Crank-Nicolson method is less than for the fully implicit method. (The analogous conclusions was reached for an ODE solver in Exercise 7.5).

15.2. Implementation and validation of solver for diffusion equation.

(a) Carry out Exercise 15.1a.

(b) Implement this explicit scheme using periodic boundary conditions.

(c) Derive the equation for the spread of a Gaussian function over time.

(d) Verify the validity of the numerical solver with an analytic solution starting with a Gaussian.

15.3. (a) Write down a finite-difference approximation for the conservation law

$$\frac{\partial f}{\partial t} = \frac{\partial}{\partial x}\left(D(x)\frac{\partial f}{\partial x}\right) = 0$$

with boundary conditions $f(x=0,t) = 1$ and $\partial f(x=L,t)/\partial x = 1$.

(b) Implement the scheme. Use $L = 2$ and $D(x) = 1 + x^2/5$.

(c) Verify numerically that after the solution has settled into its stationary form, the flux $F(x) = D(x)\partial f/\partial x$ is the same at every grid point. A correct discretization is *flux-conservative*.

Submit program source code and program output.

15.4. Implement a spectral solver for the advection equation with periodic boundary conditions, and validate the numerical solution with an analytical solution.

16 Reformulated Problems

16.1. The gravitational potential of a point mass and the electric potential of a point charge both have potentials of the form $\Phi \propto 1/|\mathbf{r}|$.

(a) A pure dipol in three-dimensional space with cartesian coordinates (x, y, z) consists of two charges $q(0, 0, d)$ and $-q(0, 0, -d)$, separated by a distance $2d$. Show that the potential of a pure dipole decays as $1/r^2$ for $|\mathbf{r}| \gg d$.

(b) Show that the dipole moment of a gravitational field expanded around the center of mass is always zero. A dipole moment is defined by

$$p = \int \mathbf{r}'\rho(\mathbf{r}')\, d\mathbf{r}'$$

where the integral is over all space.

16.2. Consider the following equations for u:

(a) $-u''(x) = f(x)$, $u(\pm\infty) = 0$, $u'(\pm\infty) = 0$

(b) $-k^2\hat{u}(k) = \hat{f}(k)$

(c) $\int_{-\infty}^{\infty} u'v'dx = \int_{-\infty}^{\infty} fvdx$

(d) $\min_u \int_{-\infty}^{\infty} \left(\frac{u'^2}{2} - fu\right) dx$

f also decays toward infinity, $f(\pm\infty) = 0$; v is an arbitrary continuous and differentiable function; $\hat{u}$ is the Fourier transform of u. Show that, for "well-behaved" u and f, these are all equivalent. (Note: Each of these formulations motivates an approach for numerical solution.)

16.3. Numerically calculate the ground state energy of the two-electron helium atom.

(a) Write down the Schrödinger equation for two electrons, including the electron-electron charge interaction. Both electrons occupy the same spherically symmetric orbital.

(b) The one-electron solution is $\psi(r) = b\exp(-2r/a_0)$, where b is a normalization constant and a_0 is the Bohr radius (for the hydrogen atom $\psi(r) \propto \exp(-r/a_0)$). Express the two-electron wavefunction as $\psi(r_1, r_2) = b\exp(-a(r_1 + r_2))(1 + c\,|r_1 - r_2|)$, with unknown coefficients a and c.

(c) Derive the analytic expression for the energy in terms of these coefficients.

(d) Numerically minimize the energy with respect to the coefficients.