Sample final

1. For each statement, answer true or false and explain your reasoning in a sentence or two. If a statement is false, the best way to explain that is by giving a counterexample.

a. There is a function, $f(x)$ that we wish to compute numerically. We know that for x values around 10^{-3}, f is about 10^5 and f' is about 10^{10}. This function is too ill conditioned to compute in single precision.

b. The function from part a is too ill conditioned to be computed in double precision.

c. If we apply Newton's method to finding the minimum of the function $f(x, y, z) = x^4 + x^2y^2 + (x - 2 * z)^4$, we will get local quadratic convergence. Note that the minimum occurs at $x = y = z = 0$.

d. I have a random variable X with density $f(x) = \frac{1}{Z}e{-}x^4$ and I do not know Z. There is a way to sample the X population without first computing Z.

e. For X as in part d, it is possible to compute Z to high accuracy by a simple quadrature method.

f. Monte Carlo would be the best (fastest, easiest to code, most accurate) way to compute $E[X^2]$, where X is as in part d.

2. Give a simple way to get a fourth order finite difference approximation to

$$\frac{\partial^2 f}{\partial x \partial y}(x, y)$$

using the same step size, h, in x and y. You need not give the precise coefficients, but you should explain how a code to do it would work. Your method should not use more than 25 evaluations of the function f.

3. Suppose the vectors in R^2, $\begin{pmatrix} x_n \\ y_n \end{pmatrix}$, satisfy the relations

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} y_{n-1} \\ 6x_{n-1} \end{pmatrix} .$$

Find a 4×4 matrix whose eigenvalues determine whether the numbers x_n or y_n remain bounded as $n \to \infty$.

4. We create a piecewise linear interpolation for a smooth function $f(x)$ interpolating at the grid points $x_k = k * \Delta x$.

a. How does the error depend on Δx?

b. Describe an adaptive strategy that chooses Δx in a systematic way to achieve an error less than ϵ. Assume that you have a procedure that computes the error for a given Δx.

5. You have a function $f(x)$ (which is really n functions of n variables $(f_1(x_1, \ldots, x_n), \ldots, f_n(x))$, and a solver that finds $x \in R^n$ with $f(x) = c$ for $c = (c_1, \ldots, c_n)$. You have a procedure that evaluates $f(x)$ accurately for any x but you did not write the solver and do not know how well it works. You run the solver in single and double precision for a certain c and get answers that differ by 15%. You do not know whether the solver is at fault or the problem of finding x from c is ill conditioned. What could you do to find out?

6. We have the following first pass code with the important lines numbered.

```
#define N   12345
#define MAX 100
    .
    .
int main () {
    double a[N], b[N];
    double sum = 0;
    for ( int i = 0; i < N; i++)               // 1
       a[i] = 1/( 1 + double(i) );             // 2
                 // double a[i] if i is even
       if ( i % 2 ) a[i] *= 2;                 // 3
       sum += a[sqrt(i)];                      // 4
    b[0] = 0;
    int n = (int) sum;
    if  ( n > N ) n = N;
    for (     i = 1; i < n-1; i++) {           // 5
       b[i] = ( b[i-1] + a[i+1] ) / b[i-1];    // 6
       if ( i * b[i] > MAX ) break;  }         // 7
    cout << "Reached i = " << i << end;
   return 0; //Nothing can possibly go wrong here.
```

a. The intermediate representation of the loop in lines $5 - 6$ contains an invariant expression, an arithmetic calculation whose operands and answer do not change during the loop. What is it?

b. Does the array element reference pattern in line 4 very bad for cache performance?

c. Which of the conditionals is worse for pipeline performance with branch prediction, line 3 or line 7? Explain.

d. We are considering rewriting the loop $1-4$ to eliminate the conditional in line 3. Would the cache performance get worse? Would the rewrite result in the code becoming slower?

e. Can we combine the two loops into one loop to improve cache performance? Would this make the code faster?

Some topics not tested here but possible for the final: IEEE arithmetic, numerical linear algebra, LU factorization, conditioning. Monte Carlo sampling methods and error bars, ...