Final Exam[1]

1. For each of the following, indicate whether the statement is true or false and explain your answer. The explanation may be a few words or a few sentences.

a. The `for` loop in line 4 below is will be executed 20 times no matter what x is generated by `rand()`. Here `float rand()` generates random numbers uniformly distributed in the interval $[0, 1]$.

```
float  x = 100*rand() + 2;                  // 1
int    n = 20;                              // 2
float dy = x/n;                             // 3
for ( float y = 0; y < x; y += dy; ) {      // 4
     body does not change x, y, or dy }
```

b. The `for` loop in line 4 always will always be executed at least 20 times.

c. The `for` loop in line 4 never will be executed more than 20 times.

d. We are writing a program to use Newton's method to find the minimum of a smooth function of n variables. The functions, $f(x)$, our code have to handle have a single local minimum, with nondegenerate Hessian at the minimizer, and go to infinity as $|x| \to \infty$. We are unable to find an initial guess close to the minimizer. If the only safeguard is a simple bisection search of the kind discussed in class then the function values will decrease from iteration to iteration, $f(x^{(n+1)}) < f(x^{(n)})$, and the iterates will converge to the minimizer.

e. If the the condition number of the matrix A is not very large and we compute x by solving $Ax = b$ using a stable algorithm, then all the components of x will be computed to high relative accuracy.

f. I have a procedure `double Int( int n, . . . )` that estimates an integral using a fourth order panel method, with n equal size panels, that is not Simpson's rule. The integrand is smooth but I do not know the integral. Nevertheless, I can estimate the error produced by `Int` for large but not too large n solely by calling that routine and doing simple arithmetic on the results.

2. Starting with the declarations

```
      float x, y, z, w;
const float oneThird = 1/ (float) 3;  // const means these can never
const float oneHalf  = 1/ (float) 2;  // be reassigned
```

[1]This version was edited to correct some mistakes in the exam actually given. Most of the changes were announced during the exam.

we do lots of arithmetic on the variables `x, y, z, w`. In each case below, determine whether the two arithmetic expressions result in the same floating point number (down to the last bit) as long as no `NaN` or `inf` values or denormalized numbers are produced.

a.

```
( x * y ) + ( z - w )
( z - w ) + ( y * x )
```

b.

```
( x +   y ) + z
  x + ( y +   z )
```

c.

```
  x * oneHalf + y * oneHalf
( x + y ) * oneHalf
```

d.

```
 x * oneThird + y * oneThird
( x + y ) * oneThird
```

3. We have the matrix

$$A = \begin{pmatrix} 4 & 1 \\ 2 & 5 \end{pmatrix} \ ,$$

and the vector recurrence relation

$$x^{(n+1)} = Ax^{(n)} \ .$$

a. Without knowing the eigenvalues or eigenvectors of A, we can tell that the components of $x^{(n)}$ will be computed to high relative accuracy, as long as they are not `inf` or `NaN`, if $x^{(0)} = \begin{pmatrix} 2.345 \\ 1.432 \end{pmatrix}$. Why?

b. Find the eigenvalues and eigenvectors of A

c. For which of the two cases below will the components of $x^{(n)}$ be computed to high relative accuracy, as long as they are not `inf` or `NaN`:

$$x^{(0)} = \begin{pmatrix} \sqrt{2} \\ -2 \end{pmatrix} \quad , \qquad x^{(0)} = \begin{pmatrix} \sqrt{2} \\ -\sqrt{2} \end{pmatrix}$$

4. We have a function, $u(x)$, that satisfies the differential equation

$$\frac{d^2}{dx^2}u(x) = u^3(x) + f(x) \ , \tag{1}$$

where $f(x)$ is a given function. We have n equally spaced points, $x_k = a + k\Delta x$, $\Delta x = (b-a)/n$, and values $u_k = u(x_k)$ and $f_k = f(x_k)$. We want an accurate estimate of

$$\int_a^b u(x)dx \ . \tag{2}$$

a. Suppose $g(x)$ is a smooth function defined on a panel symmetric about the point x_*, of the form $[x_* - \Delta x/2, x_* + \Delta x/2]$, and that we know the values of g and its second derivative at the endpoints of the panel: $g(x_* - \Delta x/2)$, $g''(x_* - \Delta x/2)$, $g(x_* + \Delta x/2)$, and $g''(x_* + \Delta x/2)$. Use Taylor series to construct the highest possible order panel integration method for this single panel that uses these data. The calculations will be much simpler if you do all Taylor series expansions about the point x_*.

b. Use the results of part a and the differential equation (1) to find a fourth order accurate approximation to the integral (2).

5. We have tridiagonal matrix, A, with diagonal entries $a_{jj} = d_j$, subdiagonal entries, $a_{j+1,j} = b_j$, and superdiagonal enties $a_{j-1,j} = c_j$, and all the other entries equal to zero. We know that A has an LU factorization.

a. Show that L has nonzeros only on the diagonal and subdiagonal and that U has nonzeros only on the diagonal and superdiagonal.

b. Give an algorithm that computes the nonzero elements of L and U from the nonzero elements of A on $O(n)$ work. This should be expressed in a form resembling C code except that you can leave algebra in algebraic notation and use words to describe some things.

6. a. Show that, if $\lambda > 0$, the operation

```
int    K;
float lambda;
K = (int) - lambda * log( rand() ); // the log is base e.
```

produces a geometric random variable with

$$Pr(K = k) = const \cdot p^k \ . \tag{3}$$

Find the relationship between λ and p. Assume that `rand()` produces independent standard uniform random variables.

b. Show that the code

```
int K;
K = 0;
while ( rand() < p ) K++;
```

produces a random variable that also satisfies (3).

c. Which method would you recommend for $p \approx .99$?

d. Write and explain a piece of C code that generates a random variable K so that (with a different *const*)

$$Pr(K = k) = const \cdot kp^k \ . \tag{4}$$

using rejection from a geometric random variable as generated by either of the methods a or b. Assume you can get a geometric random variable using a call to the routine `int geom(float p)`. Do not give code for `int geom(float p)`, just call it. Remember that the acceptance probability may never be negative or larger than one.

7. We have a system of differential equations

$$\dot{u}_j(t) = V(u_{j+1}(t)) + V(u_j(t)) + V(u_{j-1}(t)) \ , \quad \text{for } j = 1, \ldots, n-1, \tag{5}$$

with known values $u_0(t) = 0$, $u_n(t) = 1$, and $u_j(0)$. We also know that $u_j(t) \geq 0$ for all j and $t > 0$. The function $V(u)$ is evaluated by solving the simple equation

$$V + e^V - 1 = u \ . \tag{6}$$

We intend n to be at least a thousand and we want to compute for large times, and do many runs. Outline a code you would build to solve this problem, explaining design choices such as what ODE method to use and how to solve the equations (6). Say what procedures you would use and what their calling arguments would be. The code should be modular yet not ignore performance issues. Discuss in particular whether you would make use of a procedure `float Vsolve( float u)` that returns the solution to (6) for a particular u value. Discuss how you could test the important procedures separately.