Math 4610 Exam I Questions - Fall 2022

1. Define relative and absolute errors and give examples where relative error is a better measure and examples where absolute error may be a better measure of differences.

Absolute Error is Defined as: Absolute Error = Measured Value - Actual Value
Relative Error is Defined as: Relative Error = (Measured Value - Actual Value) / Actual Value

When both terms are Approximatley 1, Either is a Good Measure. When both terms are very large in magnitude, absolute error is a better measure. When both terms are very small in value, relative error is a better measure.

So if u=1000 and v=1001

```
abserror = (1001 - 1000) = 1
relerror = (1001 - 1000) / 1000 = .1%
```

Absolute error is a much better measure of error because relative errror is so small.

Or if u = 0.001 and v=0.0011

```
abserror = (0.0011 - 0.001) = 0.0001
relerror = (0.0011 - 0.001) / 0.001 = 10%
```

Relative error is am uch better measure of error here because absolute error is so small.

2. Describe the difference between the concepts of accuracy, efficiency, and robustness in the development of algorithms for the approximation of solutions of mathematical problems. Accuracy -> Accuracy is error generated when using an approximation method. It's often described relative to the step size, h, usually in the form c * h ^ n, where higher n's lead to faster convergence.

Efficiency -> Effeciency is a measure of how much work the algorithm has to do. In most cases, this is determined by the number of calls to the function you're evaluating, which, when calculating to within a desired error, can be effected by how fast your function converges.

Robustness -> Robusntess is a measure of how well the program can handle different input. A robust algorithm will produce results/accurate results over a wide variety of initial conditions. In a lot of cases, this is determined by whether the algorithm converges or not.

3. Define the rounding unit (or the machine precision) and explain the importance of the rounding unit for computation.

Machine precicision is the accuracy of the standard used for storing the numbers into binary. In other words, it's then umber of binary digits a system uses to express a number. It's impossible to accuratly express every float, because there are uncountable many between even just 0 and 1. This means that a system is only as accurate as it's representation of floats. In most systems, this is done using either floats or doubles, floats using 23 bits for storage, doubles using 52 (with a single implicit bit each). It's importatnt because any number with a trailing decimal that exceeds this limit will be truncated/rounded (which can actually lead to cascading differences in our numeric representation), meaning our calculations can't get more accurate than the machine precision.

4. What is a nonlinear equation? Compare this to linear equations.
   A linear equation is one where the change in input(s) is proportial to the change in output, and can be expressed in the form output = a * input1 + a2 * input2 + b, etc. A non-linear equation is an equation that does not follow this property, such as input = sqrt(output) or output = input ^ 2 or any other such combination.
5. Is the bisection (i) efficient, (ii) accurate, (iii) robust? What smoothness conditions on the function are needed for Bisection to work?
   The biseciton method is not very efficient. It makes a lot of function calls (two per step), and itt converges linearly, when other algorithms can do much better. This means you'll need more iterations to get a given error. Additionally, you will basically allways need a fixed number of iterations, as (how we coded it) it has one, very niche short-out condition that isn't likely to be met before the loop finishes. However, it does only need to evaluate the function itself, while other methods may need to evaluate the derivative or do other complex calculations.

It's not very accurate. As mentioned before, it converges linearly, so it will take a lot of work to get decent accuracy using the bisection method.

It's very robust. The only real requirments are that the funciotn be continuous, and that you need to know a value on either side of the x-axis. If those conditions are met you're gaurenteed a result.

6. Does the bisection method provide a robust platform for the development of algorithms for the solution of systems of nonlinear equations?

Yes. As mentioned above, the bisection method is pretty much gaurenteed to converge on any well behaved function where you know the location of values on the opposite side of each axis. It can serve as a fallback algorithm when faster, more efficent algorithms fail to converge, and allows you to reduce the intervals you need to search on.

7. What are basic conditions for fixed point iteration to converge when searching for the root of a nonlinear function of a single variable. How are these conditions related to the iteration function, $g(x))$, defined in terms of the original function, f, defined as the input of a root finding problem?

A fixed point iteration will converge with the derivitive of g(x) relative to x is less than one for your root. For this to happen, the funciton must be continusly differnetiable as well. This means it's hard to know if your function will actually converge before testing it, making it hard to work with.

8. State two advantages and two disadvantages of Newton's method for finding roots of nonlinear functions.
   Advantages:
     Newton's method experiences quadratic convergence.
     It's actually pretty simple to implement (not sure if this counts).
     You only need one initial point (compared to two for bisection/secant).

Disadvantages:
  The initial guess must be close to the root.
  You must compute and evaluate f'(x).

9. Why would a person use the Secant method in place of Newton's method?
   If the derivative of f(x) is difficulte to calculate or expensive to evalulate. It converges

superlineraly, while newton's converges quadratically, so it's a toss up between more iterations, and not having to calculate/evaluate the derivative.

10. Distinguish between the terms data fitting, interpolation, and polynomial iteration.
    Data Fiting: Approximating a curve/Line to a set of points. In class, this was done by solving a linear system of equations using matrix multiplication to give us coefficients for the equation y = ax + b. In this case there is assumed to be some error.

Interpolation: Fitting a curve exactly to a precise set of points (just big enough) to evaluate points between the known ones. This means our polynomial will be of (at most) degree n-1 for n points. There is no error when it comes to the curve fitting your original points.

Plynomial Interpolation: Interpolation using a polynomial function as your result. This can be done through a variety of methods, such as with legrange polynomials, and produces exact results. This is one of many forms of interpolation. This is usually a preferable form because the result is easy to work with due to its polynomial nature.

11. State one advantage and two disadvantages of using the monomial basis for polynomial interpolation.
    Pros:
    The mononomial basis is good for proving a solution exists. If the rows of the VanderMonde matrix are linearly independant, that proves a unique solution exists. However, the VanderMonde matrix is pretty hard to work with.

Cons:
At large polynomial sizes, the Vandermonde matrix can be hard if not impossible to solve.
Adding new data points leads to an entirley new problem, as opposed to lengrange interpolation, which allows you to reuse cardinal functions for the same points.

12. Define Lagrange polynomials (the cardinal functions) and how are they used in the development of algorithms for numerical integration.

The lagrange plynomials are polynomials which solve the following conditions for a set of points {x0, ..., xn}:
li(xj!=i) = 0
li(xi) = 1
They can be expressed by PRODUCT: [k = 0, n] (k != i) (x-xk) / (xi - xk)

These are useful for numerical integration because if you don't know the exact integral of f(x), or if it's difficult to calcuate (or if you don't even have f(x) and just have a set of points), you can use a set of legrange polynomials to represent it, and integrate them. They'll all integrate easily because they're polynomials. Additionally, because they only depend on x values, they can be calculated ahead of time, and reused for different functions that are being compared over the same x values.

13. We have bumped into errors in the approximating roots of functions, approximating derivatives using difference quotients, approximations of solutions of differential equations and approximations of definite integrals.
    $|error| \leq C\ h^p Writeabriefexplanationoftheformulaintermstheincrement, h, theconstant, C$, and how to compute these parameters. Use an example like Newton's method for finding roots of functions.

The formula puts a maximum bound on our error term. As we usually don't care about the actual value of the error, because calculating it could be more work than it's worth, we usually take all extraneous terms not related to step size and shove them into a constant, in this case, C (which will often depend of the value of our function and it's derivatives at certain points). What we really care about is how fast our error shrinks. This will usually be mostly determined by the changes in our step size h. The p in this equation represents this relationship, between step size and error. The larger p is, the faster the error shrinks when we decrease our step size, and the faster out sytem will converge.

Computing these is usually done with taylor series. In the case of newtons method...
By Definition of Root:
$f(x) = 0$
Add Zero:
$f(xk + x - xK) = 0$
Rearange:
$f(xk - (xk - x)) = 0$
By Definition of Error:
$f(xk - ek) = 0$
TSE about xk:
$f(xk) + f'(xk)(xk - ek - xk) + (f''(xk)(xk-ek-xk) ^ 2) / 2 + H.O.T. = 0$
Assume $f'(x)$!= 0 and Rearange:
$f(xk) - f'(xk)ek + (f''(xk)ek ^ 2) / 2$
Divide:
$f(xk) / f'(xk) - ek + (f''(xk)ek ^ 2) / 2$
By Definition of Error/Newton's Method:
$xk+1 - x^* = (f''(xk)ek^2) / 2$
$ek+1 = (f''(xk)ek ^ 2) / 2$

Result:
$c = f''(xk) / 2$
$p = 2$
h = Not Present, Not a Step Size Controllable Method

14. Discuss the pros and cons of using the Trapezoid rule for approximating definite integrals.
    Pros:
      Converges quadratically
      Only Needs n+1 funciton evaluations (when using composite).
      Perfect accuracy for linear systems.
      Same function evaluations, but technically less work than Simpson's due to no multiplicaiton of terms.

Cons:
  Simpson's Method converges O(h^4).
  Imperfect accuracy for non-linear systems.
  Simpson's rule integrates up to x^3 perfectly.

15. Compare the explicit and implicit Euler methods for approximate solution of initial value problems. You can use the logistic equation to illustrate your explanations.
    The explicit and implicit euler methods are both techniques for approximating a function using the derivative an an initial value. By reversing the limit definition of the derrivative, you can actually step by step approximate the values of a function by using the slope of the

function to guide you to your next point. In this case, it's kind of similar to Newton's method, except for a much different purpose. The main difference between the two is that in the explicit euler method, you use your pervious point to calculate the values of the next, where in the implicit euler, you use the current, yet uknown point to calculate its own value, which requires solving a root finding problem. Take for example the logistic equation, the explicit euler would be descibed by the equation:
P[k+1] = P[k] + h(alpha * P[k] - beta * P[k] ^ 2)
whereas the implicit euler would be descibewd by the equation:
P[k + 1] = P[k] + h(alpha * P[k + 1] - beta * P[k + 1] ^ 2)

Explicit requires less work to solve the equation, but needs smaller steps to maintin accuracy, so different systems will call for a differnt approach.