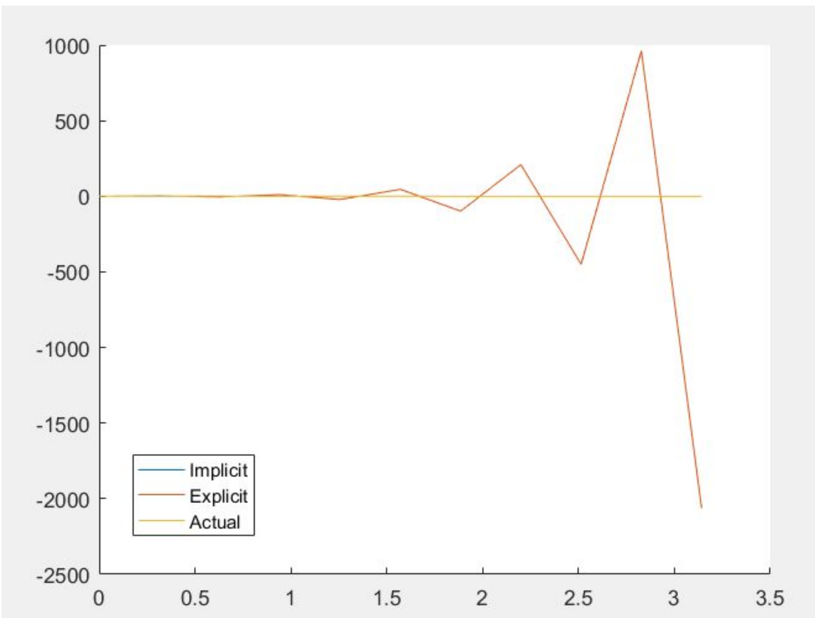
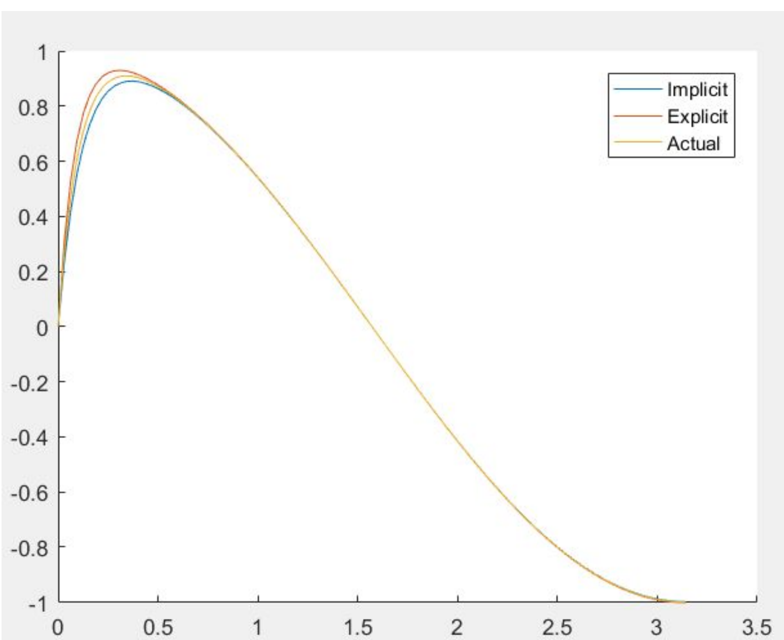
Jack Weissenberger
Problem Set 4: Initial-Value Problems for ODEs

Solving a Problem:
Write a script that solves f(t,y) = -10(y-cos(t)-sin(t) using explicit and implicit euler methods for different values of n

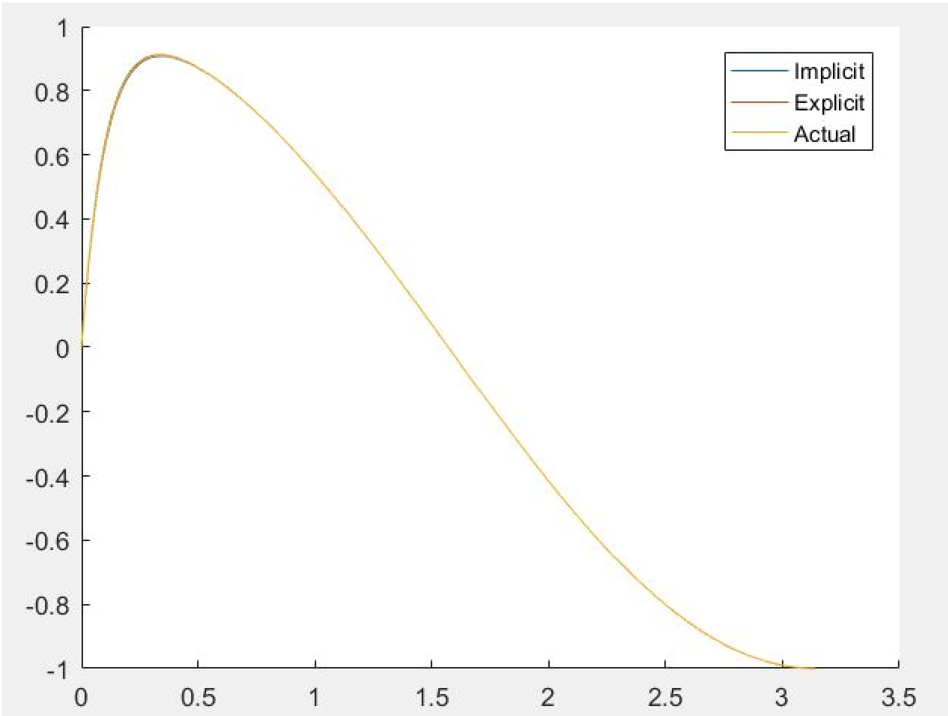
N = 1e1:



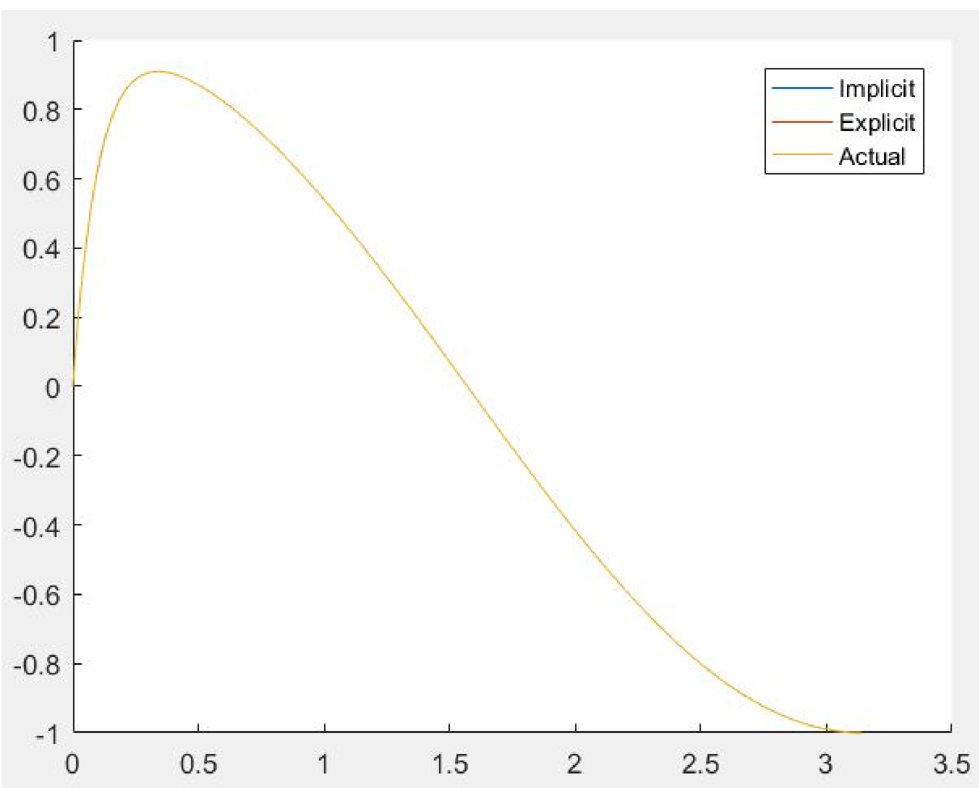
N = 1e2:



N = 1e3


Implicit
Explicit
Actual
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5
3
3.5

N = 1e4


Implicit
Explicit
Actual
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5
3
3.5

3. What difference between explicit and implicit Euler for n = 10?

For n = 10, the local truncation error of the explicit_euler function compounds and grows larger at every step. This is because the distance h, between each timestep is too large so that the approximation from the previous point creates a large error for the next timestep. Since the first step created a bad prediction, the following approximation will be even worse, which is why the error grows so large.

The error for implicit euler is 3 orders of magnitude smaller than the explicit method. This is partly because the fzero method to solve the backwards euler equation is more effective and also because I am using explicit euler as the first approximation in the fzero method.

4.

n	Implicit Euler Error	Explicit Euler Error
1e1	0.2099	2.0616e3
1e2	0.0518	0.0681
1e3	0.0058	0.060
1e4	5.8704e-4	5.88e-4

We can see here that we have first order convergence for both methods because as n increases by a power of ten the error generally increases by a power of 10 as well. For explicit Euler's method, this does not hold true for small values of n as the error compounds and grows much larger. It does become linear for larger values of n though.

5.

```
explicitError =

   5.8856e-04

53 successful steps
4 failed attempts
172 function evaluations

ode23Error =

   5.7552e-04
```

We can conclude that ode23 is a much more efficient algorithm than our explicit Euler algorithm. Our's had 1e4 function evaluations and ode23 was able to obtain the same error with only 172 function evaluations. This is because of the better step size control of ode23 that allows it to take larger steps if the tolerance is met.