1 week 4

1. Because $q(x)$ looks very similar to the orginal polynomial. The difference in the two could have been due to a measuring error in the scientific model. As numerical analysts we would hope that the solutions (the roots) of the two similar polynomials would be similar since we do not want small measurement errors to drastically change our computations. Unfortunately in this case; the roots are not similar.

2. It easy to verify a given function $v(t)$ is a solution to a IVP: simply plug v and its derivatives into the differential equation and verify that the left hand side equals the right hand side and the initial condition is satisfied. Later in the course we will attempt to solve IVP when we are not given the solution $v(t)$ as a formula. This is much more interesting.

3. (a) 1.9999923706054688
   (b) 1.5619206968586228e-16
   (c) -1.52587890625e-05
   (d) -1.0. Here is the code for part d)

```
u,v = 2,-1
print(0,u)
for k in range(2,18):
    u,v = v,v/u
    print (k,u)
```

4. You can write code to approximate or review your calculus notes about convergent sequences.
   (a) 0
   (b) 1
   (c) exp(7)
   (d) diverges to ∞.
   (e) 0

5. Assume the solutions to each of the following discrete dynamical system converge code to find its limit. Use proposition 2.2 and direct computation to show the limit u^* is a fixed point.
   (a) 8
   (b) 4
   (c) 4
   (d) 5
   (e) 2

2 week 5

1. (a) 1.1078205295102599
   (b) 0.9875064291508866
   (c) 1.1236388847132548
   (d) 1.1241230297043334

   I have most confidence in the final approximation since it was in that case that the most digits were stabilized.

2. When $\lambda = 0$ in Example 2, $g(u) = 0$ and all discrete dynamical systems converge to 0, the only fixed point. In fact all discrete dynamical systems take the form $u^{(0)}, 0, 0, \ldots$ This is trivial, not interesting.

3. $6.04 \leq f(2.1) \leq 6.05$.

4. Consider $f(x) = \frac{x}{x^2+1}$.
   (a) $f'(x) = \frac{1-x^2}{(x^2+1)^2}$
   (b) $(-1, 1)$
   (c) $(-\infty, -1) \cup (1, \infty)$
   (d) min: $-\frac{1}{2}$. max: $\frac{1}{2}$.

5. You can either graph $g(u) = \frac{u^2-1}{3}$ together with $y = u$ as we did in class or you can solve the fixed point equation $u^* = g(u^*) = \frac{(u^*)^2-1}{3}$. This is a quadratic equation with two real roots. Only one (unique) root is in the interval $[-1, 1]$.

6. After some playing around I found an equivalent fixed point problem $u = g(u) = 3^{\frac{1}{4}} \left(u^2+1\right)^{\frac{1}{4}}$. Differentiating gives $g'(u) = \frac{3^{\frac{1}{4}} u}{2} \left(u^2+1\right)^{-\frac{3}{4}}$. In order to use Theorem 2.6 (and its proof) we need to find σ so that $|g'(u)| < \sigma < 1$ for all u in $[1, 2]$. In other words, we want to maximize $g'(u)$ on a closed interval. This is a calculus problem. We do it by computing $g''(u) = \frac{3^{\frac{1}{4}}}{2} \left((u^2+1)^{-\frac{3}{4}} - \frac{3u^2}{2}(u^2+1)^{-\frac{7}{4}}\right)$. We find that $0 = g''(\sqrt{2})$ and that $g'(u)$ has a maximum value at $u = \sqrt{2}$. Using a calculator $g'(\sqrt{2}) \sim 0.4082482904638631$ so we can use the proof of Theorem 2.6 with $\sigma = 0.45$ (or any $\sigma > g'(\sqrt{2})$). In order to get two digit of accuracy we need to find a k so that $|e^{(k)}| \leq 0.5 \times 10^{-2}$. Find an appropriate k by solving $|e^{(k)}| \leq \sigma^k |e^{(0)}| \leq \sigma^k$. So we want to find k with $0.45^k \leq 0.5 \times 10^{-2}$. Using logs or python code you can find $k = 7$ or any bigger k works.

7. I used Theorem 2.6 and its proof. I also tried our fixed point code for each $g(u)$ function with seed $u^{(0)} = 2.5$. Please use the code and Theorem 2.6.
   (a) 3rd BEST $g'(u^*) = \frac{18}{21}$ linear convergence.
   (b) BEST $g'(u^*) = 0$ quadratic convergence.
   (c) WORST $g'(u^*) > 1$ unstable.
   (d) 2nd BEST $g'(u^*) = -\frac{1}{2}$ linear convergence.

8. Read middle of page 250, Section 6.3 Nonlinear Equations, until the middle of page 255 in Newman's text. Note that Newman calls our "discrete dynamical iteration method" from class "the relaxation method." There are no exercises on these pages but try to get the code examples to print out the same discrete dynamical systems using your own python code.

3 week6

1. (a) This system of nonlinear equations can be solved by elimination.
   (b) Try the seed $(x^{(0)}, y^{(0)}) = (2.1, 0.5)$ to convince yourself that the fixed point $(x^*, y^*) = (2, 0.4)$ is unstable in the given discrete dynamical system.
   (c) Try $x = \sqrt{\frac{2-y}{y}}$ and $y = \frac{x}{1+x^2}$.

2. 4.965114.

3. $0.033765242898, 0.169395306767, 0.380690406958, 0.619309593042, 0.830604693233, 0.966234757102$.

4. Use Newton's method to compute the root $x^3 - 5 = 0$. ANSWER: 1.70991

5. 4.43027
   (a) 16 steps.

(b) 3 steps.

6. $f'(2.67) \sim -0.89086$.

7. Both ways $326045071.665m$. Newton's takes 8 steps starting at the seed $r^{(0)} = 1 \times 10^5 m$. Secant takes 11 steps starting with seeds $r^{(0)} = 1 \times 10^5 m, r^{(1)} = 1 \times 10^6 m$.

8. By the IVT. f is continuous since all polynomials are continuous and $f(1) = -1 < 0$ and $f(2) = 12 > 0$.

4 week 7

1. graph

2. (a) code
   (b) I = 4.4004267
   (c) When $N = 100 I = 4.40000004267$, and when $N = 1000, I = 4.4$

3. graph

4. graph

5. 0.746764254652294

6. 3:04393009815514

5 week 8

1. (a) $\begin{bmatrix} 0 \end{bmatrix}$
   (b) $\begin{bmatrix} 4 & 5 \\ 8 & 10 \\ -4 & -5 \end{bmatrix}$
   (c) $\begin{bmatrix} 1 & 0 & 0 \\ 3 & -2 & -6 \\ 0 & 2 & 2 \end{bmatrix}$
   (d) $\begin{bmatrix} 1 & 0 & 0 \\ 3 & -2 & -6 \end{bmatrix}$

2. Verify both using matrix multiplication.

3. Verify using matrix multiplication.

4. $A^3 = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$.

5. Show by multiplying EE^{-1}.

6. Show all parts by multiplying matrices.

7. Try to find a contradiction when solving for a and c using matrix multiplication.

8. DA has the same first row as A (multiplied by 3) and the same second row as A (multiplied by 5). EA has first row and second row the same as the second row of A.

9. $U = \begin{bmatrix} 2 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 8 \end{bmatrix}$. The pivots are $2, 1$, and 8. The solution is $(x, y, z) = (2, 1, 1)$.

10. $U = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & -2 \end{bmatrix}, E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix}$, and $E_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}$. The E matrix is $E = E_3E_2E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 10 & -2 & 1 \end{bmatrix}$. Multiply to check $EA = U$.

11. $EAx = Eb$ is the system

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ -10 \end{bmatrix}$$

which has solution $\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -5 \end{bmatrix}$.

12. $E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix}$, and $U = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 4 & 2 \\ 0 & 0 & 5 \end{bmatrix}$.

13. $E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$, and $E_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}$. We find $U = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$ and $L \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}$, the matrix of multipliers.

14. $c = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, x = \begin{bmatrix} -5 \\ 3 \end{bmatrix}$, and $A = \begin{bmatrix} 2 & 4 \\ 8 & 17 \end{bmatrix}$. A quick multiplication shows that $Ax = b$.

6 week 9

1. Reading.
2. See page 219 Newman for Python ideas.
3. (a)

$$\begin{aligned} 4V_1 - V_2 - V_3 - V_4 &= 5 \\ -V_1 + 3V_2 - V_4 &= 0 \\ -V_1 + 3V_3 - V_4 &= 5 \\ -V_1 - V_2 - V_3 + 4V_4 &= 0 \end{aligned}$$

   (b) `[3.  1.666666667  3.333333333  2. ]`

4. Use the 'psuedocode' from page 56 in Olver's notes.
5.
```
from numpy import array, empty, identity

A = array([[1,1,1,1],
           [1,2,3,4],
           [1,3,6,10],
           [1,4,10, 20]], float)
N = len(A)   # size of the square matrix A
```

```
L = identity(N)

# Gaussian elimination for each column j
for j in range(N):
    if A[j,j] == 0:
        print('A is not regular')
        break
    for i in range(j+1, N):
        L[i,j] = A[i,j]/A[j,j]  # the multiplier in the (i,j) position.
        A[i,:] -= L[i,j]*A[j,:] # Subtract multiple of row from lower rows.

print(A)  # A has now been eliminated to U
print(L)  # L is the lower triangular matrix, the matrix of
multipliers
```

6. First apply LU decomposition above. Then solve for the columns of A^{-1}, one at a time.

7. see solution to 6.1.

8. (a) $\begin{bmatrix} 18 \\ 5 \\ 0 \end{bmatrix}$

   (b) $\begin{bmatrix} 3 \\ 4 \\ 5 \\ 5 \end{bmatrix}$

9. (a) $\begin{bmatrix} 3 & 2 \\ -2 & 4 \\ -2 & 2 \end{bmatrix}$

   (b) $\begin{bmatrix} 2 & 0 & -1 \\ -3 & 1 & 2 \end{bmatrix}$

   (c) $\begin{bmatrix} 1 & -21 \\ 6 & -7 \\ 1 & 4 \end{bmatrix}$.

10. $A^{-1} = -\frac{1}{2}\begin{bmatrix} 2 & -2 \\ -3 & 2 \end{bmatrix}$.

11. $A^{-1} = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$.

12. $c = \frac{1}{3}$.

13. $L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

14. $P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$, and $P_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$.

15. $P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$. There is another answer. To test your answer compute P^3.

16. (a) $P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{3} & 1 \end{bmatrix}$ and $U = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 3 & 8 \\ 0 & 0 & -\frac{2}{3} \end{bmatrix}$.

(b) same L and P as above with $D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -\frac{2}{3} \end{bmatrix}, V = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix}$.

17. See the Olver's notes for worked solutions.

7 week 10

1. Both left and right sides are 1.

2. Verify the inequality by computing both sides and comparing.

3. $\frac{1}{5}\begin{bmatrix} 4 \\ 3 \end{bmatrix}$, $\frac{1}{\sqrt{5}}\begin{bmatrix} 1 \\ 2 \end{bmatrix}$, and $\theta = \frac{2}{\sqrt{5}}$. $a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, b = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, c = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$.

4. (a) -1
   (b) 0
   (c) -3

5. (a) There are an infinite number of vectors perpendicular to v. They form a line through the origin parallel to $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$, i.e. $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = t\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ for any real t. In linear algebra we say that $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ forms a basis for the solution space. This terminology can be confusing since there are many basis of this solution set. For example, $\begin{bmatrix} -12 \\ -24 \end{bmatrix}$ also serves as a basis.

   (b) There are an infinite number vectors perpendicular to v. In total these vector $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ form the plane through the origin with equation $x + y + z = 1$. The vectors $v_1 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$ and $v_2 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$ form a basis of this plane. (There are an infinite number of other bases of this space but a key fact is that each basis of this space has exactly two vectors.)

   (c) This space is a line through the origin parallel to $b = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$. b serves as a basis for this space.

6. (a) $\frac{\pi}{3}$
   (b) $\frac{\pi}{2}$
   (c) $\frac{\pi}{3}$.

7. $2 \leq ||v - w|| \leq 8$ and $-15 \leq v \cdot w \leq 15$.

8. For the first part use Gaussian elimination to show that the matrix $A = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$ has rank 3, i.e. it has 3 pivots, when the columns of A are the vectors v_1, v_2, v_3. For the second part show that the system $Bx = 0$ has a solution $x \neq 0$ when $B = \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}$.

9. (a) independent
   (b) dependent.

10. (a) $v_1 = \begin{bmatrix}1\\2\end{bmatrix}$
    (b) $v_1 = \begin{bmatrix}1\\2\end{bmatrix}, v_2 = \begin{bmatrix}2\\5\end{bmatrix}$.

11. Here is the code from class:

```
from numpy import array, dot

A = array([[2,1], [0,1]],float)
u = array([1.00005,-1],float)  # seed

for k in range(110):
    print(k,u)
    u = dot(A,u) #update u
```

    (a) the zero vector $\vec{u}^* = \begin{bmatrix}0\\0\end{bmatrix}$ is obviously a fixed point. If you want to find all fixed points you will have to wait until class next week. Yet, for now, try to use Proposition 2.2 from Olver's notes and your code to guess (approximate) another fixed point.
    (b) After trying many different seeds you should conclude that this linear iterative system is stable.

12. 3.

13. (a) independent
    (b) dependent

14. (a) yes
    (b) no.

15. (a) 3
    (b) 0
    (c) 12

8 week 11

Set $A = \begin{bmatrix}v_1 & v_2\end{bmatrix}$ and solve (Gaussian Elimination?) $Ac = w$ for the vector $c = \begin{bmatrix}c_1\\c_2\end{bmatrix}$.

(a) $c = \begin{bmatrix}1\\3\end{bmatrix}$

(b) $c = \begin{bmatrix}-1\\.5\end{bmatrix}$.

1. (a) $u^{(*)} = \begin{bmatrix}0\\0\end{bmatrix}$
   (b) $u^{(*)} = \begin{bmatrix}1\\1\end{bmatrix}$
   (c) $u^{(*)} = \begin{bmatrix}1\\1\end{bmatrix}$

(d) $u^{(*)} = \begin{bmatrix}1\\1\end{bmatrix}$

(e) $u^{(*)} = \begin{bmatrix}3\\3\end{bmatrix}$

There is a line of fixed points $u^{(*)}$. All such $u^{(*)}$ are stable.

2. See Example 6.18 in Olver's notes. $u^{(*)} = \begin{bmatrix}0\\0\end{bmatrix}$ is the only fixed point. It is unstable—all seeds close to $u^{(*)}$ are repelled.

3. $u^{(*)} = \begin{bmatrix}0\\0\end{bmatrix}$ is the only fixed point. It is stable. For any seed the discrete dynamical system approaches the fixed point $u^{(*)}$.

4. $u^{(*)} = \begin{bmatrix}0\\0\end{bmatrix}$ is the only fixed point. It is stable. For any seed the discrete dynamical system that starts near $u^{(*)}$ remains near. In fact each discrete dynamical system is periodic. Eigenvalues are complex.

5. $u^{(*)} = \begin{bmatrix}0\\0\end{bmatrix}$ is the only fixed point. It is stable. For any seed the discrete dynamical system approaches $u^{(*)}$ in the limit. Eigenvalues are complex.

6. See Example 6.19 in Olver's notes.

7. $\lambda_1 = 1, \lambda_2 = \frac{1}{2}$ are the eigenvalues of A with corresponding eigenvectors $v_1 = \begin{bmatrix}.6\\.4\end{bmatrix}, v_2 = \begin{bmatrix}1\\-1\end{bmatrix}$. A^2 has the same eigenvectors as A but new eigenvalues $\lambda_1 = 1^2, \lambda_2 = \frac{1}{2}^2$. A^∞ also shares same eigenvectors with A but $\lambda_1 = 1^\infty = 1, \lambda_2 = \frac{1}{2}^\infty = 0$.

8. (a) $1, 4$ and 6.
   (b) 2 and $\pm\sqrt{3}$
   (c) 0 and 6.

9. Use matrix multiplication to show $Q^TQ = I$.

10. Use matrix multiplication to show $A^TA \neq I$.

11. see Example 6.21 in Olver's notes.

12. (a)
```
a,b = 0,1
for k in range(0,40):
    print (k,a)
    a,b = b,a+b
```
    (b) The dominant (larger) eigenvalue is $\lambda_1 = \frac{1+\sqrt{5}}{2}$ with corresponding eigenvector $v_1 = \begin{bmatrix}1\\\lambda_1\end{bmatrix}$ from class. The seed to start the Fibonacci sequence is $u^{(0)} = \begin{bmatrix}0\\1\end{bmatrix} = c_1v_1 + c_2v_2$ when $c_1 = \frac{1}{\sqrt{5}}$. After iteration the second term $c_2\lambda_2^{39}v_2$ becomes small and we thus disregard it. We approximate F^{39} is the first component of $c_1\lambda_1^{39}v_1$ which is $\frac{1}{\sqrt{5}}\lambda_1^{39} = 63245986.00000007$.

13. See the corresponding tables in the notes.

9 week 12

1. $w = -\frac{5}{3}q_1 + \frac{4}{3}q_2 + \frac{7}{3}q_3$.
2. $w = q_1 + q_2 + 2q_3 + 2q_4$.
3. We did both of these iterations in class.
   (a) diverges
   (b) converges to $u^* = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.
4. $u^{(*)} = \begin{bmatrix} -.1 \\ .7 \\ -.6 \\ .7 \end{bmatrix}$.
5. This is important code to write yourself.
6. In class we found $q_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, q_2 = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}, q_3 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$. Using $A = QR$ you can find $R = Q^T A$, since Q is orthogonal.
7. These algorithms are written by professional software developers. In this course we prefer to write our own code when possible.
8. $_1 = 4, \lambda_2 = -2$ and $v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.
9.
```
11 [[  3.99999857e+00   2.92968680e-03]
 [  2.92968680e-03  -1.99999857e+00]]
```
10. see notes.
11. Like always, compute a few iterates of the sequence $A^0, A^1, A^2, A^3, \ldots$ before running the code below.

```
import numpy as np

A = np.array([[.5,-1/3],[-1/3,.25]],float)
B = np.identity(2)  # B will be the kth power of A

for k in range(35):
    B = np.dot(A,B)
    print(k, B)
```

12. Since one of the eigenvalues is 4, the iterates diverge.
13. Use the famous matrix decomposition $D = X^{-1}AX$ where X is the matrix of eigenvector columns of A and D is the diagonal matrix with corresponding eigenvalues on the diagonal. Test this decomposition for $B = \begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}$.
14. see notes.
15. see notes.

10 week13

1. (a) This can be solved exactly by separation of variables $u(t) = \frac{C}{x}$ for some constant C. Your integral curves should have roughly the same shape as the exact solutions.

   (b) $u(t) = -2t - 2$ is an isocline which is also a solution. This solution appears unstable in that other solutions, starting close, move away in positive time.

   (c) $u = t - 1$ is an isocline that is also a solution. It looks stable.

2. Isoclines are circles centered on u-axis. $u(t)$ is decreasing in first quadrant since $u' = -\frac{u}{t^2+u^2} < 0$ in the first quadrant. Any solution that gets into the first quadrant must stay there in positive time because (since the integral curve is decreasing and time is increasing) the only way to escape is to cross the $t-$axis. This is impossible because the $t-$axis itself is an integral curve and the Fundamental Existence and Uniqueness Theorem tells us that solutions (integral curves) can never touch.

3. (a) Compute yourself using pen and paper $u(.1) \approx .9, u(.2) \approx .82$, and $u(.3) \approx .758$. Then write code to make the same computation. Here is my Python code but you should try to write it on your own.

```
def F(t,u):  # Differential Equation u' = F(t,u)
    return t - u

h=.1 # stepsize
t,u = 0,1 # initial condition: u(0) = 1

for k in range(4):
    print(k,t,u)
    A = F(t,u)  # slope = A = u'
    t += h    # update time
    u += h*A   # update u (calculus step)
```

   (b) If you draw some isoclines in first quadrant you will see that the exact solutions are concave-up so Euler's method must be an under approximation.

   (c) Compute $u(.1) \approx 0.91$ on your own with pen and paper then use code to get the rest. Like always try to write your own code before using mine below:

```
def F(t,u):  # Differential Equation: u' = F(t,u)
    return t - u

h=.1 # stepsize
t,u = 0,1 # initial condition: u(0) = 1

for k in range(4):
    print(k,t,u)
    A = F(t,u)  # follow your nose slope
    u_temp = u + h*A # Euler's updated u
    t += h  # update time
    B = F(t,u_temp) # sniff ahead slope
    # average slopes (Eulers and sniff-ahead) to update u
    u += h*0.5*(A+B)
```

   (d) At first sight the improved Euler method does seem to be better since the approximations from improved Euler are greater. You can find an exact solution to this first-order constant coefficient linear equation $u' + u = t$ by multiplying both sides by the integrating factor e^t and then integrating to give the exact solution $u(t) = t - 1 + \frac{2}{e^t}$ as a formula. Comparing the exact answers $u(.1) = 0.90967..., u(.2) = 0.83746..., u(.3) = 0.781636...$, we conclude that the Improved Euler Method does improve the approximation in this case.

4. Euler's Method gives the following approximations $u(.1) \approx 1.1, u(.2) \approx 1.231, u(1.2) \approx 43.659....$ The isoclines are parabolas pointing left. Using the isoclines and slope field one can see that the exact solution is concave up and thus the Euler approximation is low.

5. See Olver's notes.

6. see Newman's book p.327–335.

11 week 14

1. (a) This is the Fundamental Theorem of Calculus from Calc 1.
   (b) 3.133043743699965, 2.6563890830909234, 2.6021191501836256, 2.595416398960074, 2.5945798164101492, 2.594475263
   (c) same answers as (b)
   (d) 2.6436288736191185, 2.5952122243308198, 2.5944720428386074, 2.594460510541956, 2.5944603303579092, 2.59446032
   (e) 2.6436288736191185, 2.595212224330818, 2.59447204283861, 2.5944605105419494, 2.5944603303578826, 2.5944603275
   (f) 2.59477713694768, 2.594459790419353, 2.5944603273666123, 2.5944603274977864, 2.594460327497842, 2.59446032749
   (g) 3.1926056038394384, 2.5944517617707175, 2.5944603253985163, 2.5944603274973095, 2.5944603274978233, 2.5944603
   (h) RK4 or Simpson's rule seem best since most digits have stabilized.

2.
```
from numpy import linspace
from math import floor
from pylab import plot,show

def Vin(t):
    if floor(2*t) % 2 == 0:
        return 1
    else:
        return -1

RC = 0.1
def F(t,u):  # u = Vout
    return 1/RC * (Vin(t) - u)

tpoints = linspace(0,10,1000)
t,u,upoints = 0,0,[] # initial conditions
n = 1000 # number of steps
h = (10 - 0)/n # stepsize


for k in range(n):
    upoints.append(u)  # make a ulist so we can graph
    A = F(t,u)  # Euler first slope

    u_2 = u + 0.5*h*A # sniff ahead to find 2nd slope
    t_2 = t + 0.5*h # let half time pass
    B = F(t_2,u_2) # 2nd slope

    u_3 = u + 0.5*h*B # sniff ahead to find 3rd slope
    t_3 = t + 0.5*h # let half time pass (same t_2)
    C = F(t_3,u_3) # 3rd slope
```

```
    u_4 = u + h*C # sniff ahead to find 4th slope
    t_4 = t + h # let all time pass
    D = F(t_4,u_4) # 4th slope

    # Finally update t and u before next loop, as always
    t += h
    u += h/6 * (A + 2*B + 2*C + D)
    #print(u)

vin_points = []
for t in tpoints:
    vin_points.append(Vin(t))

plot(tpoints,vin_points) # plot of squarwave input, Vin in blue
plot(tpoints,upoints)    # plot of output (after filter) Vout in orange
# Note that after the filter the Vout voltage rises and falls more gently
show()
```

3. See section 8.2 Newman's text

4. (a)
```
from numpy import array, arange
from pylab import plot, xlabel, show

def F(t,u):  # u[0] = rabbits, u[1] = foxes
     F0 = u[0] - 0.5*u[0]*u[1]
     F1 = 0.5*u[0]*u[1] - 2*u[1]
     return array([F0,F1],float)

a = 0.0  # initial time
b = 30.0 # final time
N = 1000 # number of steps
h = (b-a)/N # stepsize

tpoints = arange(a,b,h)
rabbits = []
foxes = []
u = array([2.0,2.0],float) # inital condition vector u^0 = (rabbits, foxes)

for t in tpoints:
    rabbits.append(u[0])
    foxess.append(u[1])
    k1 = h*F(t,u)
    k2 = h*F(t+0.5*h, u+0.5*k1)
    k3 = h*F(t+0.5*h, u+0.5*k2)
    k4 = h*F(t+h, u+k3)
    u += (k1+2*k2+2*k3+k4)/6

plot(tpoints,rabbits)  # blue
plot(tpoints,foxes)    # orange
xlabel("t")
show()
```

   (b) You can see that the both the rabbit and the fox population oscillate periodically. The phase of each population is shifted to account for the increase in rabbit population will result (shortly afterwards)

in an increase in foxes—while the increase in foxes will result (shortly afterwards) in the decrease of rabbits.

5. Section 8.3 Newman

6. Using the formulas in Example 8.6, modify the rabbits and foxes code above–your ODE $u' = F(t, u)$ will have $u[0] = \theta$ and $u[1] = \omega = \frac{\mathrm{d}\theta}{\mathrm{d}t}$. Superimposing the graphs of $\theta(t)$ and $\omega(t)$ onto one graph gives an image similar to the rabbits and foxes graph above. Try to explain why the phases of the periods are shifted in this physical case.

7. (a) There is only one fixed point $u^* = 0$ except in the case $a = 0$, when every number is a fixed point.
   (b) There is only one critical point $u(t) = 0$ except in the case $a = 0$, when every constant u_0 gives a critical point $u(t) = u_0$.
   (c) $u^{(k)} = u^{(0)}a^k$
   (d) $u(t) = u_0 e^{at}$
   (e) Use a staircase diagram with different seeds. You'll need to make separate staircase plots for each constant a but you can put different seeds onto the same plot.
   (f) You'll need to make separate slope fields for each constant a but you can put different seeds onto the same slope field.
   (g) unstable: $a = -2$ and $a = 2$, stable: $a = -1, a = -0.5, a = 0, a = 0.5$ and $a = 1$, asymptotically stable $a = -0.5, a = 0$, and $a = 0.5$. Convice yourself of this using the formulas, the staricase diagrams, as well as Python code with various intitial conditions.
   (h) unstable: $a = 0.5, a = 1$ and $a = 2$, stable: $a = 0, a = -0.5, a = -1$ and $a = -2$, asymptotically stable $a = 0, a = -0.5, a = -1$ and $a = -2$. Convice yourself of this using the formulas, the slope-field diagrams, as well as Python code with various intitial conditions.

8. (a) There is always one fixed point' $u^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ to a linear dynamical system. However since the matrix $A = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$ has an eigenvalue of 1 it has a line of fixed points spanned by $v_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.
   (b) Each system of ODE have only one critical point $u(t) = u_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.
   (c) $u^{(k)} = c_1\lambda_1^k v_1 + c_2\lambda_2^k v_2$
   (d) $u(t) = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2$
   (e) stable: $A = \begin{bmatrix} .6 & .9 \\ .1 & .6 \end{bmatrix}$, unstable: $A = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}$ and $A = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$. Use your python code for linear iterative systems to convince yourself.
   (f) stable: $A = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}$, unstable: $A = \begin{bmatrix} .6 & .9 \\ .1 & .6 \end{bmatrix}$ and $A = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$. Use your python code for RK4 systems to convince yourself.

9. The critical point $u(t) = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}$ is asymptotically stable if all of its (complex) eigenvalues $\lambda_j = a + bi$ have real part less than zero: $a < 0$.

12 week 15

1. $\hat{x} = \begin{bmatrix} 3 \\ 10 \end{bmatrix}$. The picture should indicate projection onto the $xy-$plane.

2. $\hat{x} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, ||b - A\hat{x}|| = 42.$

3. $\hat{x} = 2$. The projection is onto the line through the origin in three dimensional space pointing in the direction $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$.

4. $p(x) = 6.75x^2 - 28.35x + 25.65$ Sketch the points with scatter plot and graph the polynomial.

5. This is most easily done using the Lagrange interpolating polynomials $p_3(x) = y_0L_0(x) + y_1L_1(x) + y_2L_2(x) + y_3L_3(x)$. In this case the computation is easy since $y_1 = y_2 = y_3 = 0$. So the answer is $p_3(x) = 27L_0(x) = 27\frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)}$. Sketch the points with scatter plot and graph the polynomial.

6.
```
from math import exp

data = [(-1,exp(-1)), (0,exp(0)), (1,exp(1))]

def L0(x):
    product = 1
    x0 = data[0][0]
    k = 0
    for (xi,yi) in data:
        if not k == 0:
            product *= (x-xi) / (x0-xi)
    k += 1
    return product

print (L0(0.32))
```

7. See notes.

8. (a) Differentiate with respect to x.
   (b) These formulas define $u''(x)$ as a piecewise linear function. It is your job to show that the pieces meet up continuously.
   (c) Integrate with respect to x.
   (d) Check that the piecewise cubic function $u(x)$ is continuous. You must check that there are no gaps—each piece passes through two data points.
   (e) The desired system of equations for $u''(x_i)$ is $\frac{h}{6}u''(x_{i-1}) + \frac{2h}{3}u''(x_i) + \frac{h}{6}u''(x_{i+1}) = \frac{y_{i+1}-2y_i-y_{i-1}}{h}$
   (f) This computation is easy if you understand the notation.
   (g) See p. 238 in Olver's notes.
   (h) See p. 238 in Olver's notes.

9. $a_1 = 7, b_1 = 8, c_1 = 4, d_1 = 1.$

10. (a) This is the constant multiple rule for antidifferentiation.
    (b) $p_1(x) = f(a)L_0(x)+f(b)L_1(x) = \frac{x-b}{a-b}f(a)+\frac{x-a}{b-a}f(b) = \frac{1}{b-a}\left((b-x)f(a) + (x-a)f(b)\right)$. Integrating the interpolation polynomial gives $\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{2}\left(f(a) + f(b)\right)$. This is of course the Trapezoid rule.