Tópicos Especiales I
Facultad de Ingeniería de Sistemas Computacionales
Grupo VIL-341
2do Semestre 2019

Tarea #3 – Python Científico (100 puntos)

Asignada: 8 de noviembre de 2019
Fecha de Entrega: 2 de diciembre de 2019 (11:59pm)

Nota: La tarea debe ser enviada de manera individual a través de Moodle en un archivo .zip. Provea sus respuestas, ya sean comandos correctos y/o código en formato digital (formato Word) o directamente en comandos de Unix cuando sea requerido. En cuanto sea posible sea claro con sus respuestas.

Observación: Identifique su código con su(s) nombre(s), y correo(s) en la parte superior del código utilizando el símbolo de comentario #.

Parte I. Operaciones con Numpy y Matplotlib

Problema 1. Numpy y Matplotlib (15 puntos). El Cuarteto de Anscombe comprende cuatro conjuntos de datos que tienen propiedades estadísticas casi idénticas, sin embargo, aparecen muy diferente al graficarse. Basada en el Cuarteto investigadores se han propuesto diseñar conjuntos de datos similares, uno de estos esfuerzos es el *Datasaurus*, o mejor conocido como el *Datasaurus Dozen*.

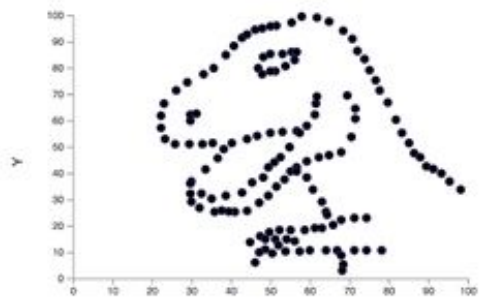


Figura 1

Estos 13 conjuntos de datos (el Datasaurus, más 12 otros) tienen las mismas características estadísticas (media, desviación estándar y correlación de Pearson) a dos decimales, mientras que son drásticamente diferentes en apariencia. Lo que nos enseña a siempre verificar las salidas de nuestras graficas, especialmente si las estadísticas calculadas son similares.

Escriba un script llamado *datasaurus.py*, que para cada uno de los 13 conjuntos de datos es capaz de:

- Leer los datos desde el archivo *DatasaurusDozen.tsv* o de *DatasaurusDozen-wide.tsv,* realice las transformaciones necesarias. Su script debe proporcionar dos figuras (dos ventanas) con la información apropiada (leyendas, ejes, etc).
- En la primera ventana grafique solamente los valores que dicen en la columna 'dino'. En la segunda figura y organizadas en un diseño de cuadrícula de 3 X 4 las otras 12.
- Para cada una de las 12 + 1 graficas, calcular el: promedio de los vectores X e Y en cada caso, la varianza de la muestra de los vectores X e Y vectores en cada caso y la correlación entre X e Y en cada caso (formateo con 3 decimales).

Sugerencia: utilice np.std(), np.mean() y np.corrcoef().

Parte II. Operaciones con Numpy y Scipy

En clase vimos el modulo *Scipy* tiene la clase *Integrate* y en especifico la función llamada Odeint, para resolver ecuaciones diferenciales ordinarias (ODEs), que utiliza el siguiente formato de llamada *odeint(model, y0, t),* donde **model** es el nombre de la función que devuelve valores derivados en los valores *t* e *y* llamados como de la siguiente manera dydt = model (y, t); el segundo parámetro es el vector **y0** que contiene las condiciones iniciales de los estados diferenciales; y por ultimo recibe el parámetro **t**: Puntos de tiempo en los que se debe integrar la solución.

Muchas veces para ODEs mas sencillas se pueden aproximar con el método simple de integración hacia delante de Euler. Esta formula se base en la noción que la derivada de una función *por definición* va a estar dada por:

$$\frac{dx}{dt} = \lim_{dt \to 0} \frac{x(t+dt) - x(t)}{dt} = f(x(t))$$

Sin embargo, si no calculamos el límite, sino que simplemente decimos que *dt* es un número bastante pequeño, obtenemos una expresión que solo se aproxima a d(x)/dt y es por lo tanto aproximadamente igual a f (x (t)), o básicamente:

$$\frac{x(t+dt)-x(t)}{dt} \approx f(x(t))$$

Si reorganizo esta ecuación aproximada, obtengo que x(t + dt) es aproximadamente igual a x (t) más dt multiplicado por f(x(t)), o en otra forma:

$$x(t+dt) \approx x(t) + f\big(x(t)\big).dt$$

Esta fórmula, llamada integración de Euler, significa que, si conocemos *x* en algún momento *t*, podemos aproximarnos a x en t + dt. Esto también significa que debe esperar saber x en algún momento para iniciar el algoritmo. La integración de Euler es el método más fácil para aproximar numéricamente una solución a una ecuación diferencial ordinaria. La integración de Euler aunque es un método substancialmente ineficiente si es comparado con otros métodos, si se hace un dt lo suficientemente pequeño, siempre terminarás aproximando el límite en la definición de la derivada del tiempo lo suficientemente bien como para darte una buena aproximación de la solución a la ODE

Problema 2. Integración de Euler en un sistema de Resorte-Amortiguador (20 puntos). Para ver prácticamente cómo funciona la integración de Euler, analicemos un sistema físico der resorte-amortiguador.

Recordemos que la relación entre la fuerza y la posición de un resorte es siempre una línea representada por la ley constitutiva $F_r = k.x_r$ (donde el número positivo k se llama constante de resorte). La relación entre la fuerza y la velocidad del amortiguador también es siempre una línea representada por la ley constitutiva $F_a = b.v_a$ (donde el número positivo b se llama constante de amortiguamiento). Por lo tanto, la fuerza de un resorte siendo positivo significa que la posición debe ser positiva y el resorte debe estar en extensión. La fuerza de un amortiguador siendo positivo significa que la velocidad del amortiguador debe ser positiva y el amortiguador debe estar extendido.

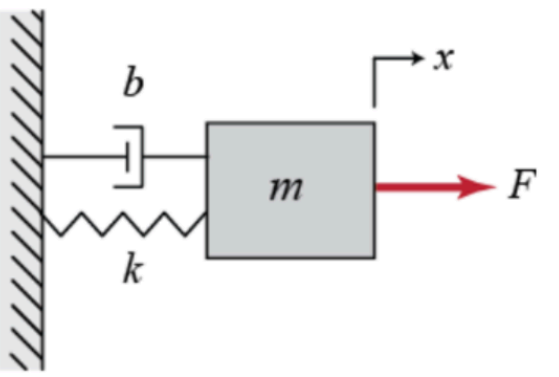


Figura 2

Si usamos las leyes constitutivas que relacionan la fuerza del resorte con la posición del resorte y la fuerza del amortiguador con la velocidad del amortiguador, $F_r = k.x_r$ y $F_a = b.v_a$ y substituimos estas relaciones constitutivas en las leyes de Newton, obtenemos una ecuación que involucra $-F_r - F_a = 0$ y $- k.x_r - b.v_a = 0$

Si además tenemos en cuenta que la derivada del tiempo de la posición del resorte es igual a la velocidad del amortiguador (es decir, $\frac{dx_r}{dt} = v_a$), obtenemos una ecuación simple que involucra solo la posición del resorte.

$$- k.x_r - b.\frac{dx_r}{dt} = 0$$

Ahora, si definimos x para ser la posición del resorte (simplemente dejando caer los subíndices del resorte "r"), obtenemos:

$$- k.x - b.\frac{dx}{dt} = 0$$

y si luego resolvemos para $\frac{dx}{dt}$ obtenemos la ecuación diferencial ordinaria (ODE) que representa el sistema:

$$\frac{dx}{dt} = -\frac{k}{b}x \text{ , también expresado como } \frac{dx}{dt} = f(x)$$

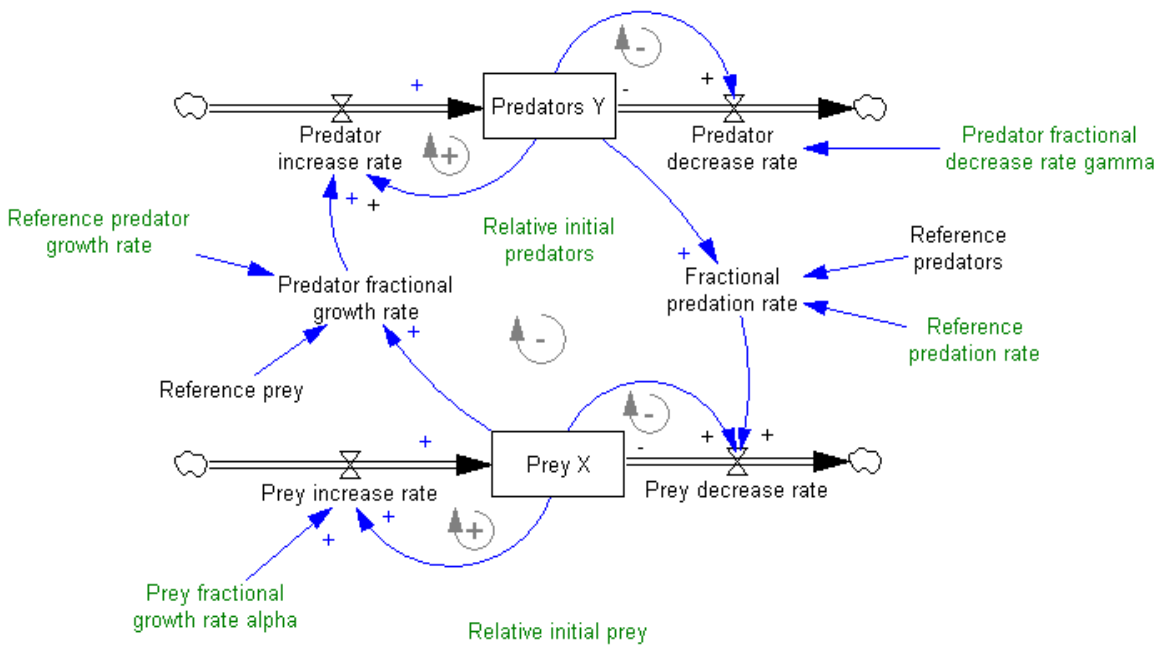
Esta ODE dice que "la derivada de la posición es igual a k negativa dividida por b multiplicado por la posición". Usando la integración de Euler sabemos que $x(0.1)$ es aproximadamente igual a $x(0) + -\frac{k}{b}x(0)$, que $x(0.2)$ es aproximadamente igual a $x(0.1) + -\frac{k}{b}x(0.1)$, que $x(0.3)$ es aproximadamente igual a $x(0.2) + -\frac{k}{b}x(0.2)$, y así sucesivamente, que $x(t+dt)$ es aproximadamente igual a $x(t) + -\frac{k}{b}x(t)$, similar a la ecuación que habíamos descrito arriba.

Ahora, supongamos que tenemos un sistema como en la Figura 2, y comenzamos con x en el tiempo t = 0 igual a 0.5, o en otras palabras x(0)=0.5, con k = 1, y b = 1, dt = 0.1. Responda las siguientes preguntas:

- ¿Cuál es el valor de la extensión *x* después de un tiempo de 10 segundos?
- Muestre este resultado gráficamente, utilizando maplotlib, el comportamiento de la distancia del sistema en el tiempo de integración (X vs t)
- ¿Que pasa si el paso de integración, *dt*, si se baja a 0.001, y si se sube a 1? Grafique sus resultados.

Sugerencia: Utilizando los valores dados por el problema resuélvalo definiendo un esquema de integración numérica de Euler, use np.arange() para definer un vector para el tiempo de integración, además use np.zeros para definir vectores para

Problema 3. Cálculos numéricos de ecuaciones diferenciales (15 puntos). Las ecuaciones de Lotka-Volterra describen un sistema depredador-presa y es uno de los ejemplos de sistemas dinámicos más famosos. De hecho, sobre este sistema existen muchas generalizaciones y aplicaciones fuera de la biología. Este tipo de sistemas generalmente se introducen el curso de Ingeniería de Sistemas Dinámicos o Simulación en carreras de Computación, y se enseñan utilizando la analogía de flujos y niveles como se ve en la figura abajo:



Este modelo DFD se le presenta al estudiante de una manera simplificada en Vensim, pero realmente atienden a la solución de 2 ecuaciones diferenciales no lineales de primer orden que se resuelven de maneras simultaneas que simulan el como las poblaciones cambian a través del tiempo según las siguientes dos de ecuaciones:

$$\frac{dx}{dt} = x(a - by) \qquad \frac{dy}{dt} = -y(c - dx)$$

Donde**,** *y* es el número total en el sistema de algún predador (por ejemplo, un lobo); *x* es el número de sus presas (por ejemplo, `conejos); *dy/dt* y *dx/dt* representa el crecimiento de las dos poblaciones en el tiempo; *t* representa el tiempo; y *a* representa el crecimiento (exponencial) de la presa*, b* representa la tasa de interacción entre ambas predadores y presas*, d* representa como crecimiento la población de depredadores (que no es necesariamente igual a la velocidad a la que consume la presa) y *c* la tasa de pérdida de los depredadores debido a la muerte natural o la emigración.

El modulo *Scipy* tiene la clase *Integrate* y en especifico la función llamada Odeint, para resolver ecuaciones diferenciales ordinarias (ODEs), que utiliza el siguiente formato de llamada *odeint(model, y0, t),* donde **model** es el nombre de la función que devuelve valores derivados en los valores *t* e *y* llamados como de la siguiente manera dydt = model (y, t); el segundo parámetro es el vector **y0** que contiene las condiciones iniciales de los estados diferenciales; y por ultimo recibe el parámetro **t**: Puntos de tiempo en los que se debe integrar la solución.

El script *predador.py* presenta una soluciona numérica para este sistema de predador-presa, se hacen los cálculos para un tiempo especifico y se grafica la comparación entre ambas poblaciones en el tiempo. Ejecute el script *predador.py* conteste las siguientes preguntas:

- Cambie la escala de tiempo de la integración a 36 meses. ***Salve la grafica correspondiente***.
- Grafique un diagrama de estado-fase en el que se utilizan los valores finales de la integración numérica, donde X sería los valores de las presas e Y sería los valores de los predadores. ***Salve la grafica correspondiente***.

Sugerencia: revisar la forma correcta de crear el diagrama de estado-fase (https://es.wikipedia.org/wiki/Espacio_fásico)

- Verifique lo que pasa cuando valor inicial para x (presas) es 10 veces mas que la de predadores. ***Salve la grafica correspondiente***. Además, con sus palabras explique el comportamiento de las poblaciones,

Este modelo que la velocidad de depredación sobre la presa es proporcional a la velocidad a la que se encuentran los depredadores y la presa, esto se representa arriba mediante *bxy*. Si x o y es cero, entonces no puede haber depredación.

Verifique los escenarios en que no hay depredación.

- Haga que el valor inicial para x (presas) sea 0. ***Salve la grafica correspondiente***. Además, con sus palabras explique el comportamiento de las poblaciones, también reporte cual es el valor final de la variable x.
- Cambie el valor inicial para y (predadores) sea 0. ***Salve la grafica correspondiente***. Además, con sus palabras explique el comportamiento de las poblaciones, también reporte cual es el valor final de la variable y.

Verifique el escenario en que varia la interacción entre presas y predadores es alta.

- Haga que el valor inicial para b sea 10. ***Salve la grafica correspondiente***. Además, con sus palabras explique el comportamiento de las poblaciones.

Verifique el escenario en que la migración o muerte de predadores es alta.

- Haga que el valor inicial para c sea 5. ***Salve la grafica correspondiente***. Además, con sus palabras explique el comportamiento de las poblaciones.

Problema 3.1. Modelo de Monos y Jaguares (15 puntos). Supongamos que en una región del alto Darién hay dos especies de animales, un mono (presa) y un jaguar (depredador). Cree un script bajo el nombre *monosyjaguares.py* que simule este fenómeno, en otras palabras, que haga el calculo numérico, grafique la progresión de las dos especies y el diagrama de estado-fase a lo largo del tiempo. ***Salve la graficas correspondiente***.

- Además, con sus palabras explique el comportamiento de las poblaciones.

  Las condiciones iniciales del sistema son:
  - 10 monos y 10 jaguares
  - Las tasas de crecimiento y muerte de monos son 1.1 y 0.4, respectivamente.
  - Las tasas crecimiento y muerte de jaguares son 0.1 y 0.4, respectivamente.
  - El intervalo de tiempo es 60 meses
  - 500 puntos de integración numérica

- Resuelva el problema arriba descrito con diferentes de valores para los jaguares variando de 1 a 11 (con paso de 2 unidades). Grafique el diagrama de estado-fase para cada uno de los 5 valores iniciales.

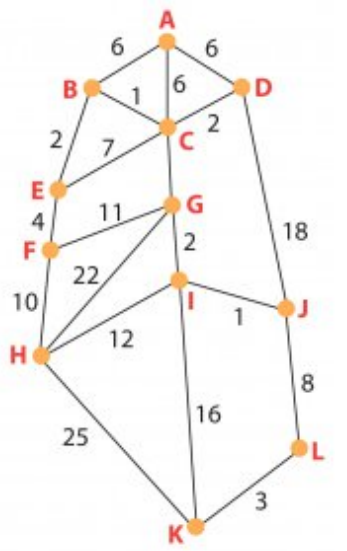
Problema 3. MST en Scipy y un ejemplo Real (15 puntos)

La librería Scipy contiene la función MST bajo s*cipy.sparse.csgraph.minimum_spanning_tree* que dada como entrada un grafo **no** dirigido con pesos, nos da como resultado un árbol de expansión de peso mínimo.

En este problema propondremos una ruta para un tour guiado de la ciudad de Venecia, Italia. Digamos que nos dan representación inicial el grafo de abajo que representa los sitios turísticos de la ciudad.

Escriba un script llamado "*mst_italy.py*", cree un grafo **no** dirigido utilizando el tipo de dato *csgraph* y calcule el MST que nos de el tour completo de la ciudad. Reporte la matriz final del tour.

A. Cannaregio
B. Ponte Scalzi
C. Santa Croce
D. Chiesa della Madonna dell'Orto
E. Ferrovia
F. Piazzale Roma
G. San Paolo
H. DorsoDuro
I. San Marco
J. St. Mark Basilica
K. Castello
L. Arsenale



Parte III. Operaciones con Pandas

Problema 5. Análisis de datos con Pandas (20 puntos). Mariano Rivera es un exlanzador de béisbol profesional panameño que jugó 19 temporadas en las Grandes Ligas de Béisbol (MLB) para los Yankees de Nueva York, de 1995 a 2013. Rivera pasó trece veces All-Star y cinco veces campeón de la Serie Mundial, él es el líder de la carrera de MLB en salvamentos (652) y juegos terminados (952). En 2019 fue ingresado al Salón de la Fama (Hall of Fame) con votación unánime, además fue abanderado de en los desfiles del Día de los Símbolos Patrios el lunes 4 de noviembre de 2019.

En el conjunto de datos *mariano.csv* de Mariano (consulte el *glosario.txt* para ver los nombres de las columnas) encontrará las estadísticas completas de su carrera de béisbol por año. Escriba un script llamado *baseball.py* que lea los datos de este archivo en panda y realice las siguientes operaciones:

a) Encuentra el año en que Mariano tuvo el número máximo de juegos terminados.
b) Usando matplotlib, haga un diagrama de barras que muestre el número de salva que Mariano por año.
c) Inserte la información de Yankees.csv en su script. Crear diagramas de caja que comparen la página (Pitching
d) Promedio) entre los años "Mariano" y el resto de los años de la Organización de los Yankees.
e) Usando el comando Pandas Merge, combine la información sobre los premios que obtuvo Mariano y la asistencia al Yankee Stadium en una temporada. ¿Cómo se correlacionan los premios de número y la asistencia al estadio? Proporcionar una trama.
f) Haga una figura que tenga dos paneles (como subplots apiladas), el primero mostrará la asistencia al Yankee Stadium por año, y el segundo mostrará la ERA + de Mariano por año.