Taller #6 de Métodos Computacionales
FISI 2028, Semestre 2014 - 20

Profesor: Jaime Forero

Martes 21 de Octubre, 2014

Importante

- Todo el código fuente y los datos se debe encontrar en un repositorio en github con un commit final hecho antes del medio día del viernes 31 de Octubre. El nombre del repositorio debe ser `Apellidos1_Apellidos2_hw6`, por ejemplo si trabajo con Nicolás deberíamos crear un repositorio llamado `Forero_Garavito_hw6`.
- La nota máxima de este taller es de 100 puntos. Se otorgan 1/3 de los puntos si el código fuente es razonable, 1/3 si se puede compilar/ejecutar y 1/3 si da los resultados correctos.
- El miercoles 29 de octubre tendremos una sesión para presentar avances de diferentes grupos en cada una de las preguntas del taller. Hay un bono de 15 puntos para cada grupo que presente un avace significativo.

1. **Dinámica de cazadores y de presas** (30 puntos)

   Tenemos dos especies, una caza a la otra. El número de presas es x y el numero de cazadores es y. Su evolución temporal está descrita por las siguientes dos ecuaciones diferenciales acopladas

   $$\frac{dx(t)}{dt} = Ax(t) - Bx(t)y(t), \tag{1}$$

   $$\frac{dy(t)}{dt} = -Cy(t) + Dx(t)y(t). \tag{2}$$

   - Encuentre los valores de $x(0)$ y $y(0)$ para que exista equilibrio ($\dot{x} = \dot{y} = 0$) asumiendo que $A = 20$, $B = 1$, $C = 30$ y $D = 1$.
   - (10 puntos) Escriba un código en C que solucione este sistema acoplado de ecuaciones para $0 < t < 1$ y escriba los resultados (t, x, y) en un archivo llamado `poblaciones_x0_y0.dat` donde `x0`, `y0` son los valores iniciales utilizados para la integración. El código de debe poder ejecutar como

     ```
     ./runge_kutta.x x0 y0
     ```

   - Corra varias veces el código variando las condiciones iniciales de la siguiente manera: $y(0)$ se mantiene fijo en el valor encontrado en el punto anterior mientras que $x(0)$ disminuye en una unidad con respecto al valor encontrado en el punto anterior hasta llegar a 1. Explore las diferentes soluciones.
   - (10 puntos) Escriba un código en Python que se pueda ejecutar de la siguiente manera `plot_poblaciones.py poblaciones_x0_y0.dat` y cree un pdf `poblaciones_x0_y0.pdf` con una gráfica en el plano $x - y$ de los puntos del archivo de datos correspondiente.
   - (10 puntos) Escriba un `makefile` que enlace correctamente todos los pasos anteriores y cree todas las gráficas para las diferentes condiciones iniciales.

2. **Trayectoria de una partícula cargada en el campo magnético de la Tierra** (70 puntos)

   La Tierra tiene un campo magnético que la protege de partículas cargadas. Por ejemplo, la aurora boreal es un fenómeno natural que se puede entender a partir del moviemiento de partículas cargadas que se aceleran dentro de un campo magnético y empizan a radiar energía electromagnética.

   El campo magnético de la Tierra, cerca de ella y hasta una distancia de $4R_T$ donde $R_T = 6378{,}1\text{km}$ es el radio de la Tierra, puede ser aproximado por el campo de un dipolo

$$\mathbf{B}_{dip}(\mathbf{r}) = \frac{\mu_0}{4\pi r^3}[3(\mathbf{M}\cdot\hat{r})\hat{r} - \mathbf{M}], \tag{3}$$

donde $\mathbf{r} = x\hat{x} + y\hat{y} + z\hat{z}$, $r = |\mathbf{r}|$ y $\hat{r} = \mathbf{r}/r$. Para la Tierra tomamos $\mathbf{M} = -M\hat{z}$ antiparalelo al eje z porque el polo norte magnético es cercano al polo sur geográfico.

En el ecuador magnético ($x = 1R_T, y = z = 0$) la intensidad del campo es $B_0 = 3 \times 10^{-5}$T. Substitutendo esto en la ecuación anterior tenermos que $\mu M/4\pi = B_0 R_T^3$. Con esto podemos escribir en coordenadas cartesianas:

$$\mathbf{B}_{dip} = -\frac{B_0 R_T^3}{r^5}[3xz\hat{x} + 3yz\hat{y} + (2z^2 - x^2 - y^2)\hat{z}]. \tag{4}$$

- (50 puntos) Escriba un programa en C que integre las ecuaciones de movimiento (no relativistas) de **protones** que tienen una posición inicial $(2R_e, 0, 0)$, energía cinética inicial E_k (medida en millones de electronvolts) y tienen una velocidad inicial descrita por $(0, v\sin\alpha, v\cos\alpha)$ donde α es un ángulo que se conoce como el pitch angle. El programa debe poder ejecutarse de la siguiente manera

  `./particle_in_field.x kinetic_energy alpha`

  donde `kinetic_energy` es la energía cinética inicial medida en megaelectronvolts y `alpha` es el ángulo α medido en grados. La trayectoria debe seguirse por 100 segundos y al final debe escribir el tiempo y las coordenadas x, y, z de las posiciones en un archivo llamado `trayectoria_E_alpha.dat` donde `E` es el valor de la energía cinética de la partícula y `alpha` es el valor del ángulo α.
- (10 puntos) Escriba un programa en Python que grafique en el plano *xy*, y en *xyz* la trayectoria guardada en `trayectoria_E_alpha.dat`.
- (10 puntos) Escriba un `makefile` que enlace correctamente todos los pasos anteriores y cree las gráficas necesarias.