Report

Zhang San
ID: 123456789
Major: Electrical Engineering

December 7, 2019

Submission: Room #301, Building No.1
Deadline: Jan. 1st, 2020

1 Text

Problem: This is a LaTeX report template.

1.1 Test

1. Insert pictures

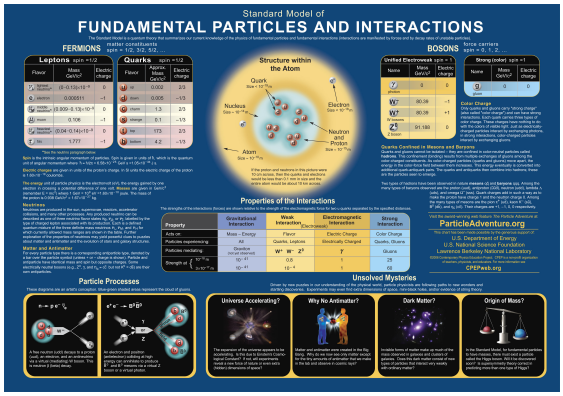

Figure 1: example

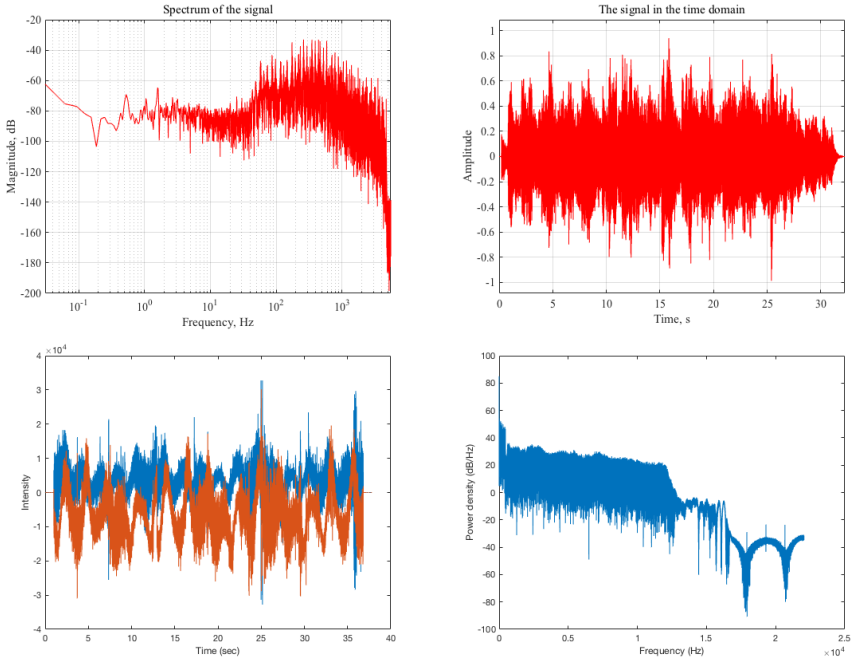

Figure 2: pictures

- Case 1:

a b c

Talk is cheap, show me the code.

2. insert code, colorbox

```
> x <- c(26,29,27,25,29,27,26,29)
> y <- c(29,30,38,30,25,39,26,36)
```

3. insert code 2

```
# -*- coding: utf-8 -*-

import math
import numpy as np
import matplotlib.pyplot as plt

def runga_kutta_vibrations(t, y0, v0, m, c, k):
    y = np.zeros(t.shape)
    v = np.zeros(t.shape)
    y[0] = y0
    v[0] = v0
    dt = t[1] - t[0]

    # Return the acceleration a
    def func(y, v):
```

```
16          return (-c*v-k*y)/m
17
18      for i in range(t.size - 1):
19         # f_t_05 = (force[i+1] - force[i])/2 + force[i]
20          y1 = y[i]
21          v1 = v[i]
22          a1 = func(y1, v1)
23          y2 = y[i] + v1*dt/2
24          v2 = v[i] + a1*dt/2
25          a2 = func(y2, v2)
26          y3 = y[i] + v2*dt/2
27          v3 = v[i] + a2*dt/2
28          a3 = func(y3, v3)
29          y4 = y[i] + v3*dt
30          v4 = v[i] + a3*dt
31          a4 = func(y4, v4)
32          y[i+1] = y[i] + dt*(v1 + 2*v2 + 2*v3 + v4)/6
33          v[i+1] = v[i] + dt*(a1 + 2*a2 + 2*a3 + a4)/6
34
35          print(y[i])
36      return y
37
38  n = 400     ## set the grids
39  t = np.linspace(0, 4, n+1)   ## set time step
40  #force = np.zeros(n)   ## set the external force as zero
41
42  # Parameters of the vibration system
43  m = 1
44  k = 9*math.pi
45  omega = 3*math.pi
46  h = 0.2         # damping rate can be replaced as 1, 2, etc.
47  c = 2*m*h*omega
48  y = runga_kutta_vibrations(t, 3*math.sqrt(3), 0, m, c, k)
49
50  # Plot the result
51  plt.plot(t, y, 'r*', linewidth=1)
52  plt.xlabel('Time')
53  plt.ylabel('Amplitude')
54  plt.title('Damped free vibration h=0.2 (4thRunge-Kutta)')
55
56  #fig, ax1 = plt.subplots()
57  #l1 = ax1.plot(t, v, color='b', label="displacement")
58  #ax2 = ax1.twinx()
59  #l2=ax2.plot(t, force, color='r', label="force")
60
61  #lines = l1 + l2
62  #plt.legend(lines, [l.get_label() for l in lines])
63  #plt.show()
```

The pseudocode is as follows:

Discrete Logarithm Problem 1 Baby-step Giant-step

Input: y, g, p, let $x = q \cdot m + r$, and hence seek q, r
Output: x

Define m as $\lceil \sqrt{p-1} \rceil$
Loops
for x=0 to m **do**
    if q is good for guess **then**
        Output $y = g^{qm+r}$
    end if
end for
Loops
for q in m **do**
    Compute $y \times g^{-qm}$
    if $g^r = g^{-qm}$ **then**
        Output $x = qm + r$
    else Go to find next value of q
    end if
end for
Print all the possible solutions

4. Make tables

Product x	26	29	27	25	29	27	26	29
Product y	29	30	38	30	25	39	26	36

Table 1: Travel trips by bus

	Zone1	Zone2	Zone3	Total
Zone1	200	500	800	1500
Zone2	300	200	700	1200
Zone3	600	700	300	1600
	1100	1400	1800	4300

Table 2: Bus share $S_{ij}(bus)$

	Zone1	Zone2	Zone3
Zone1	0.02	0.05	0.08
Zone2	0.03	0.02	0.07
Zone3	0.06	0.07	0.03

Table 3: Car share $S_{ij}(car)$

	Zone1	Zone2	Zone3
Zone1	0.11	0.07	0.02
Zone2	0.06	0.11	0.02
Zone3	0.02	0.07	0.09

Last digit	0	1	2	3	4	5	6	7	8	9
ubackward	1.0	1.25	1.5	1.75	2.0	2.25	2.5	2.75	3.0	3.25

5. Equations

$$W = \frac{b^2}{S^2} = \frac{(\Sigma_{i=1}^{n} a_i x_{(i)})^2}{\Sigma_{i=1}^{n}(x_i - \bar{x})^2}$$

Table 4: Solution with Step Size h=0.01

x	True Solution	Euler method	Heun method	Modified Euler method	4th order Runge-Kutta method
0.000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
0.010	0.0199000	0.0200000	0.0198995	0.0198998	0.0199000
0.020	0.0396000	0.0397990	0.0395990	0.0395995	0.0396000
0.030	0.0591000	0.0593970	0.0590985	0.0590993	0.0591000
0.040	0.0784000	0.0787940	0.0783980	0.0783990	0.0784000
0.050	0.0975000	0.0979901	0.0974976	0.0974988	0.0975000
⋮	⋮	⋮	⋮	⋮	⋮
0.960	1.0098187	1.0157715	1.0097903	1.0098030	1.0098187
0.970	1.0111022	1.0170842	1.0110738	1.0110865	1.0111022
0.980	1.0122088	1.0182193	1.0121803	1.0121930	1.0122088
0.990	1.0131391	1.0191776	1.0131105	1.0131232	1.0131391
1.000	1.0138938	1.0199596	1.0138651	1.0138778	1.0138938

Table 5: Comparison of accuracy

Step Size	Euler		Heun		Modified Euler		4th Runge-Kutta	
	Approximation	Error	Approximation	Error	Approximation	Error	Approximation	Error
0.2	1.1431393	0.1292454	1.0011505	0.0127434	1.0067767	0.0071172	1.0138742	0.0000197
0.1	1.0763404	0.0624465	1.0108764	0.0030175	1.0122102	0.0016837	1.0138927	0.0000012
0.01	1.0199596	0.0060658	1.0138651	0.0000287	1.0138778	0.0000161	1.0138938	0.0000000

$$\Phi^* = \frac{(\frac{V_x}{n_x} + \frac{V_y}{n_y})^2}{(\frac{V_x}{n_x})^2 \frac{1}{n_x - 1} + (\frac{V_y}{n_y})^2 \frac{1}{n_y - 1}}$$

$$\mathbf{B} = \begin{pmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{pmatrix}$$

$$[\mathbf{C}] = \frac{E}{1 - \nu^2} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{pmatrix}$$

$$X = \begin{bmatrix} 1 & x_{11} & x_{21} & \dots & x_{m1} \\ 1 & x_{12} & x_{22} & \dots & x_{m2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & x_{2n} & \dots & x_{mn} \end{bmatrix}, Y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{1}$$

$$\begin{aligned} S_W &= \sum_{k=1,2} \sum_{x \in w_i} (w^T x - w^T \mu_i)(w^T x - w^T \mu_i)^T \\ &= \sum_{x \in w_1} (w^T x - w^T \mu_i)^2 + \sum_{x \in w_2} (w^T x - w^T \mu_i)^2 \\ &= w^T (\tilde{s}_1 + \tilde{s}_2) w \\ &= w^T S_W w \end{aligned} \tag{2}$$

For free electrons, $E(k) = \frac{\hbar^2 k^2}{2m}$, and in the k space, the radius k for equal energy surface is $k = \sqrt{2mE}/\hbar$, volume V is $(2\pi/L)^3$

$$|\nabla_k E| = \frac{dE}{dk} = \hbar^2 k / m$$

$$Z(E) = \frac{L^3}{4\pi^3}\iint_{E=const}\frac{dS}{|\nabla_k E(k)|} = \frac{L^3}{4\pi^3}\iint_E dS$$
$$= \frac{L^3}{4\pi^3}\cdot\frac{m}{\hbar^2 k}\cdot 4\pi k^2 = \frac{(2m)^{3/2}L^3}{2\pi^2\hbar^3}E^{1/2} = 8\pi\sqrt{2}\frac{m^{3/2}}{(2\pi\hbar)^3}E^{1/2}\cdot L^3$$

(a) Prediction step in the y direction:

$$v^p_{i,j} = \underbrace{v^n_{i,j}}_{\text{v\_center}} + \Delta t[-\overbrace{u^n_{i,j}\frac{\partial v^n_{i,j}}{\partial x}}^{\text{advection\_x}} - \overbrace{v^n_{i,j}\frac{\partial v^n_{i,j}}{\partial y}}^{\text{advection\_y}} - \underbrace{\frac{1}{\rho}\frac{\partial p^n_{i,j}}{\partial y}}_{\text{pressure\_y}} + \overbrace{\nu\frac{\partial^2 u^n_{i,j}}{\partial x^2}}^{\text{diffusion\_x}} + \overbrace{\nu\frac{\partial^2 v^n_{i,j}}{\partial y^2}}^{\text{diffusion\_y}}]$$

(b) Advection term:
Using first-order upwind difference scheme:

$$c\frac{\partial f_i}{\partial x}|^n = \begin{cases} c\dfrac{f^n_i - f^n_{i-1}}{\Delta x}, \text{c} \geq 0 \Rightarrow c * (\text{f}_{\text{center}} - \text{f}_{\text{minus}})/\text{delta} \\ c\dfrac{f^n_{i+1} - f^n_i}{\Delta x}, c < 0 \Rightarrow \text{c} * (\text{f}_{\text{plus}} - \text{f}_{\text{center}})/\text{delta} \end{cases}$$

6. Let your student ID be "A" and the remainder of dividing "A" by 6 be "C". Compute "D=21+C", show an algorithm that computes a non-zero "E" where $E = exp(-(D-0.5)^2) * exp(-(D+0.5)^2)/exp(-D^2)$.

web site http://www-pw.physics.uiowa.edu/rbsp/audio/.

Integral:

$$\int_V \rho\frac{d\mathbf{v}}{dt}dV = \int_V \mathbf{b}dV + \int_V \mathbf{B}^T\sigma dV$$

$$\mathbf{u}(x) = \sum_{i=1}^{2} h_i u_i$$

$$\Phi(a_n x + b_n)^n = (\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{a_n x+b_n} e^{-\frac{y^2}{2}}dy)^n \to e^{-e^{-x}}$$

References:

[1] S. M. Sze, Semiconductor Devices: Physics and Technology, 3rd edition.

[2] Stephen J. Chapman. Fortran 95/2003 for Scientists and Engineers, 3rd edition. McGraw-Hill Education. 2007.

[3] Gilbert Strang. An Analysis of the Finite Element Method, 2nd edition. Wellesley-Cambridge. 2008.

[4] Jochen Alberty, Carsten Carstensen, Stefan A. Funken. Remarks around 50 lines of Matlab: short finite element implementation. Numerical Algorithms 20(1999)117-137.