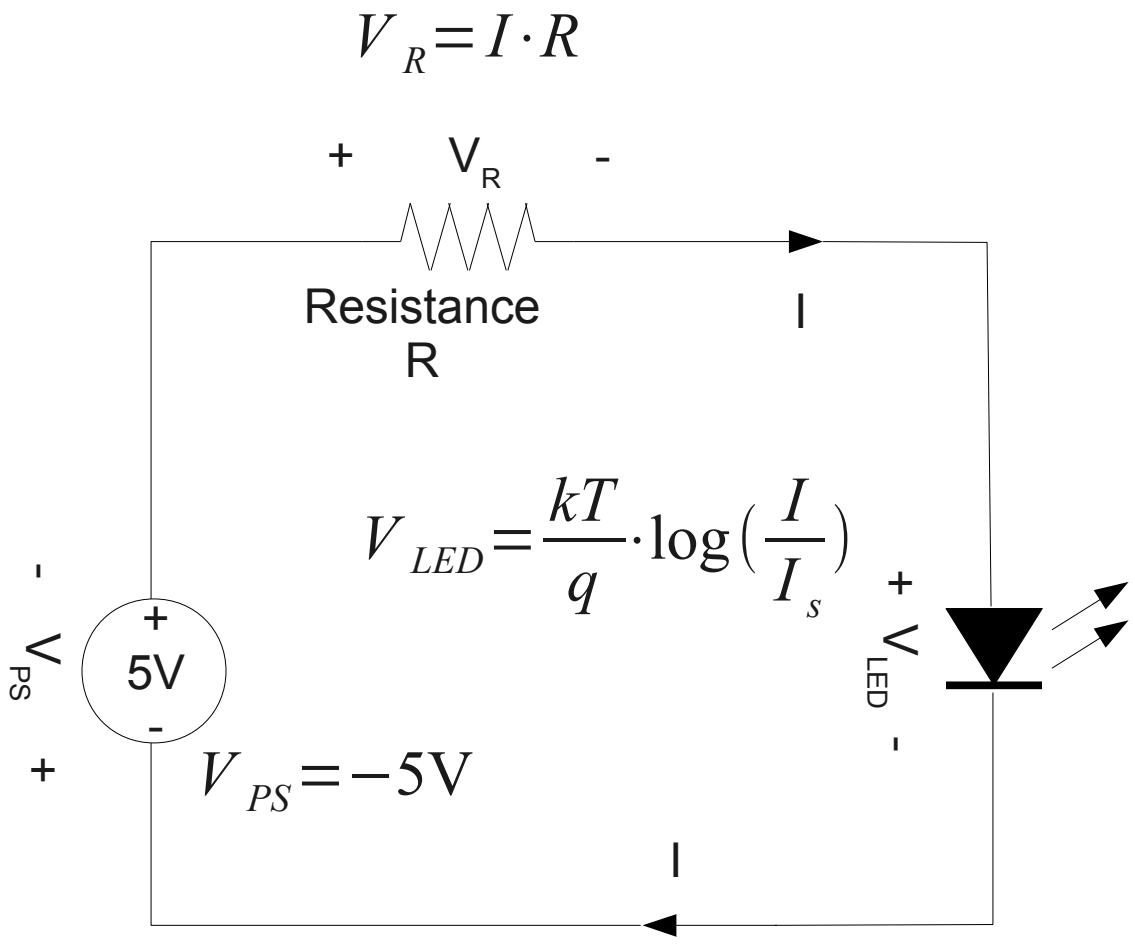
Physics 3340/8300-005 - "Computational Physics" - Fall 2013
Homework 6
Assigned: Monday, 14 October
Due: Friday, 18 October 11:59pm by Blackboard upload

Reading:

Cheney and Kincaid sections 2.1, 2.2 (6th ed. 7.1, 7.2)
http://www.physics.smu.edu/fattarus/RootsLab.html
http://www.physics.smu.edu/fattarus/LineqLab.html

Problems:

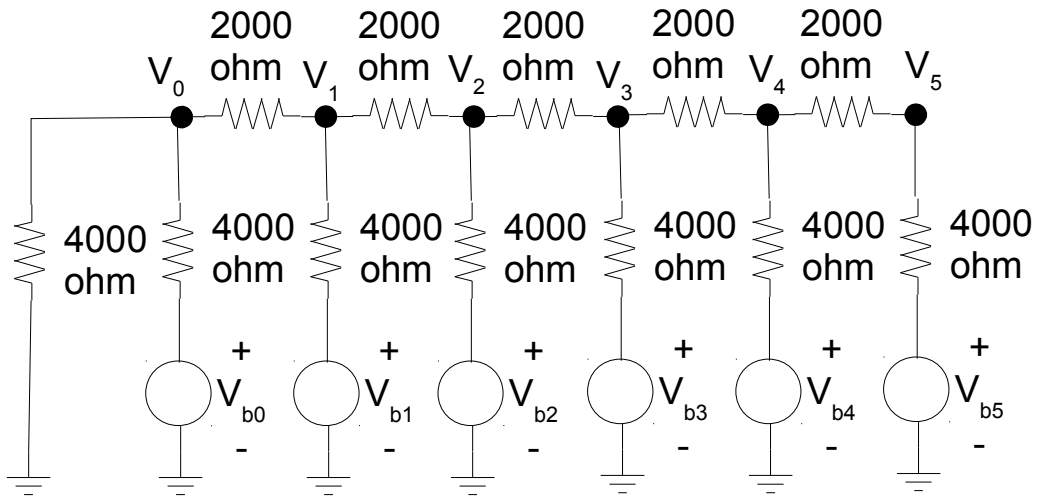
1) The current flowing through a light emitting diode (LED) determines its light power output. Find *I*, the current in amperes flowing in the following electrical circuit containing an LED at an absolute temperature of 300 degrees Kelvin:



Use Kirchoff's voltage law around the loop of current, where the sum of the electrical potential drops must = 0. In the figure this means $V_{PS} + V_R + V_{LED} = 0$. Note that for a source of electrical potential this means for the reference direction shown, V_{PS} = -5V. In the figure the voltage drop V_R across the linear resistor of resistance *R* by Ohm's law is *IR*. Assume *R* = 240 ohms for this circuit. The figure also shows the nonlinear relationship between the current in the LED and its voltage drop V_{LED}. In this expression *k* is Boltzmann's constant, *T* is the absolute temperature and *q* is the charge on the electron. I_s is the saturation current for a particular diode junction, which for this LED is 6.18×10^{-26} A. Form a

nonlinear function of the loop current I so that when its root is found, Kirchoff's voltage law is satisfied. Code the function in C and find the root with the `sweep_secant()` function following the roots lab on the class website. You can use the sinx_bisection.c main program in the lab as a starting point for your new main program. You may either use your own code for the secant algorithm you created in Homework 5 and create a sweep_secant() function and corresponding header declaration, or use the ones supplied in the new versions of sweep_roots.c and sweep_roots.h files now contained in the roots.tar tar archive on the website. Submit your code that solves for the loop current. A good sweep search range to try is 0 to 50×10^{-3} A.

2) Use the Gaussian elimination function gauss_pivotmax() to solve the electrical network for a 6-bit digital to analog converter shown below:



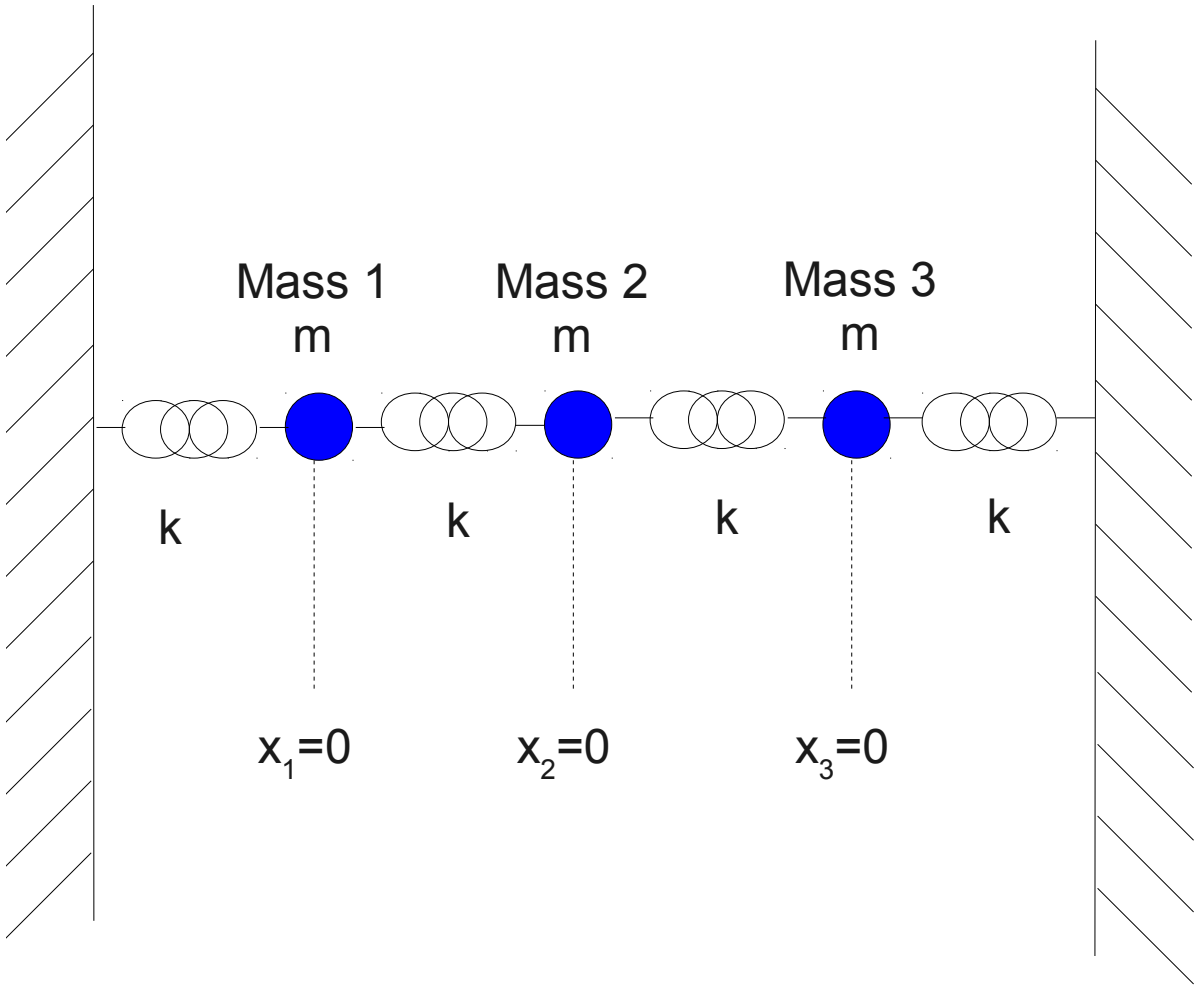
The voltage sources V_{b0} through V_{b5} correspond to each bit of the digital input, 5 volts for a "1" bit and 0 volts for a "0" bit, and corresponding to the binary weights 2^0 through 2^5, respectively. The analog voltage output is taken from the V_5 node voltage. There are six interconnection nodes for this circuit network, and thus it will require solving a system of six equations in six unknowns. The six unknown variables in the system are the voltages at each of the circuit nodes V_0 through V_5, and the six linear equations are to be developed by Kirchoff's Current Law for each node. To apply Kirchoff's law to a given node, sum the currents flowing into the node from the left, right and below and set the sum to zero. For this network the currents flowing into a node from each direction may be easily found from Ohm's law, and will be simple linear functions of the node voltages. For example, at the V_1 node the current flowing in from the left is $\frac{(V_0 - V_1)}{2000 \text{ ohms}}$, the current flowing in from the right is $\frac{(V_2 - V_1)}{2000 \text{ ohms}}$, and the current flowing in from the bottom is $\frac{(V_{b1} - V_1)}{4000 \text{ ohms}}$. These all sum to zero forming one equation at that node. The other five equations may be found in a very similar manner at the other five nodes.

Assemble your six equations into a system of the form $$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} \\ a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix}$$ where the $a_{i,j}$ coefficients and the b_i right hand side elements come from your node equations. Note that many of the coefficients will be zero.

Then use the gauss_pivotmax() function to solve this system as in the class lab exercise on linear systems. You can use the lower portion of the pivot.c main program in the lab as a starting point for your new main program. Be sure to properly initialize the zero coefficients. It is easiest to write a loop at the beginning and initialize **all** the matrix coefficients to zero, and then simply overwrite the nonzero coefficients with their proper values. Also write your main program to accept six command line arguments, for the bit voltages V_{b0} through V_{b5} in units of volts. The node voltage V_5 solved by the program is the analog output corresponding to the digital input expressed in the bit voltages.

Try running your program with the following digital inputs: 100000 010000 000001 011111 111111 and verify that the V_5 node voltage follows the expected converter output voltage of $\frac{5 \text{ volts} \cdot D_{\text{in}}}{64}$ where D_{in} is the decimal equivalent of the digital input value bits. Submit your code that solves the network.

3) Consider the following three-mass coupled oscillator:



Three three masses *m* are identical and the spring constants *k* are all identical. Follow the derivation in the class slides for the solution of the equations of motion for the three masses. First define the constant $\omega^2 = \frac{k}{m}$. Then the displacements of the three individual masses obey the second order differential equations for spring-mass harmonic oscillators:

$\frac{d^2 x_1}{dt^2} = -2\omega^2 x_1 + \omega^2 x_2$ $\frac{d^2 x_2}{dt^2} = \omega^2 x_1 - 2\omega^2 x_2 + \omega^2 x_3$ $\frac{d^2 x_3}{dt^2} = \omega^2 x_2 - 2\omega^2 x_3$. We assume simple oscillatory solutions of the form $x_1 = a_1 e^{i\alpha t}$ $x_2 = a_2 e^{i\alpha t}$ $x_3 = a_3 e^{i\alpha t}$ for the so-called normal modes of oscillation. Finding the conditions under which such solutions of the equations of motion exist reduces to the classical eigenvalue problem $\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \lambda \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$ where $\lambda = \frac{\alpha^2}{\omega^2}$. The possible eigenvalues and eigenvectors of this matrix problem define the motion of the three normal modes of

oscillation for the system. For an arbitrary set of initial condition for the mass displacements, and an assumption of zero initial mass velocities, the motion will evolve as a linear combination of the three normal modes. You are to write a program to solve for the linear combination coefficients for the three modes given an arbitrary set of mass displacements, and generate an output data file that traces the displacement of each mass as a function of time. For the known eigenvalues and eigenvectors of the above matrix problem, and assuming zero initial velocities for the three masses, the motion will follow

$$x_1(t) = -c_1\cos(\sqrt{2}\,\omega t) + c_3\cos(\sqrt{2-\sqrt{2}}\,\omega t) + c_5\cos(\sqrt{2+\sqrt{2}}\,\omega t)$$
$$x_2(t) = \sqrt{2}\,c_3\cos(\sqrt{2-\sqrt{2}}\,\omega t) - \sqrt{2}\,c_5\cos(\sqrt{2+\sqrt{2}}\,\omega t)$$
$$x_3(t) = c_1\cos(\sqrt{2}\,\omega t) + c_3\cos(\sqrt{2-\sqrt{2}}\,\omega t) + c_5\cos(\sqrt{2+\sqrt{2}}\,\omega t)$$

where the linear combination coefficients c_1 c_3 and c_5 may be found by solving the system of linear equations

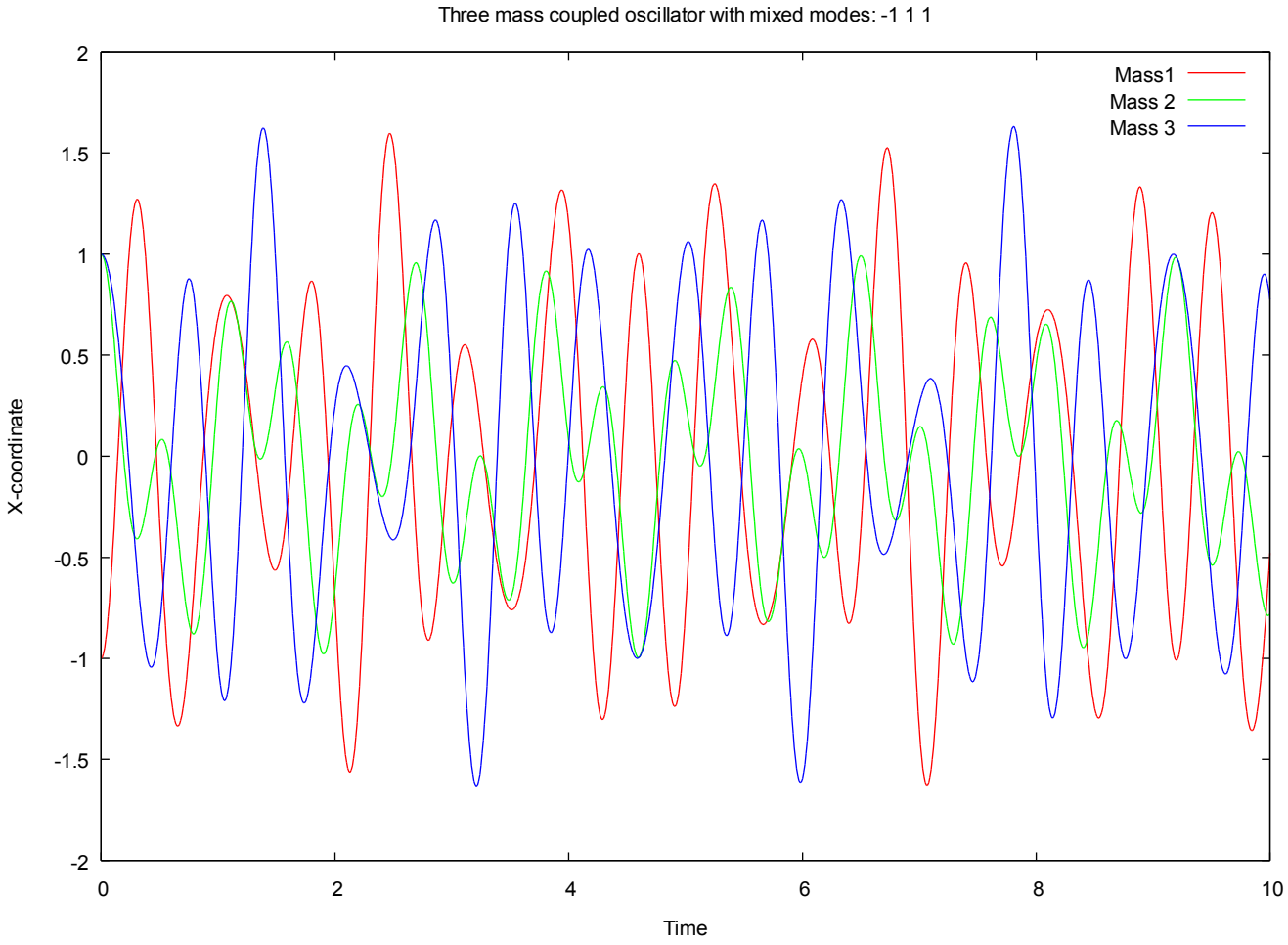
$$\begin{bmatrix} -1 & 1 & 1 \\ 0 & \sqrt{2} & -\sqrt{2} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_3 \\ c_5 \end{bmatrix} = \begin{bmatrix} x_{10} \\ x_{20} \\ x_{30} \end{bmatrix}$$

given the three initial mass displacements x_{10} x_{20} and x_{30}.

As detailed in the class slides, your main program should follow the outline of:

```
double ...;

double mass1(double t) {
  ...
}

double mass2(double t) {
  ...
}

double mass3(double t) {
  ...
}

int main(int argc,char *argv[]) {
    ...
  a = alloc_matrix(3,3);
    ...
  if (gauss(3,a,rhs,c,1.0e-18)) {
    ...
  }
  fprintf(stderr,"Mode 1:%.8g  Mode 2:%.8g  Mode 3:%.8g\n",c[0],c[1],c[2]);
    ...
  sweep(mass1,0.0,tstop,tinc);
  printf("\n\n");
  sweep(mass2,0.0,tstop,tinc);
  printf("\n\n");
  sweep(mass3,0.0,tstop,tinc);
    ...
  exit(0);
}
```

The program should accept command line arguments for the three initial mass displacements. Run the program for the initial mass displacements of -1, +1 and +1 units, and using $\omega=2\pi$ radians per second. Plot the three mass displacements as a function of time, using the 'index' keyword in gnuplot for the three data sets written into the output file by the program. Your plot should match:



Submit your main program code and a PNG image of your plot.