18.085, PROBLEM SET 3 SOLUTIONS

Question 1. (25 pts.) Resonance and practice with (complex) exponentials. In this question, to be done entirely by hand, $i = \sqrt{-1}$, the unit imaginary number. Recall that the equation for one mass (of mass m) and one spring (of spring constant c), where u is the position of the mass away from equilibrium, is (no forcing):

$$mu'' + cu = 0 \tag{0.1}$$

where the prime $'$ means a derivative in time, and $u = u(t)$ is a function of time only (no space). We omit initial conditions here.

a) (5 pts.) Plug in $u_1 = \sin\omega t$ to get $-m\omega^2 \sin\omega t + c\sin\omega t = 0$ which implies $\omega = \sqrt{c/m}$. You get the same result using $u_2 = \cos\omega t$.

b) (4 pts.) The mass oscillates SLOWER when it is heavier: $\omega = \sqrt{c/m}$ is smaller for larger m. The mass oscillates FASTER when the spring is stiffer (larger c): $\omega = \sqrt{c/m}$ is larger for larger c.

c) (5 pts.) Plug in $u = e^{rt}$ to get $mr^2e^{rt} + ce^{rt} = 0$, which implies $r = \sqrt{-c/m} = \pm i\sqrt{c/m}$. That is, $u_1 = e^{i\sqrt{c/m}t}$ and $u_2 = e^{-i\sqrt{c/m}t}$ are the solutions to (0.1).

d) (3 pts.) $r_1 = i\omega$, $r_2 = -i\omega$.

e) (BONUS 5 pts.) Try again a guess of $v = e^{st}$ in $mv'' + bv' + cv = 0$ to get $ms^2e^{st} + bse^{st} + ce^{st} = 0$, which implies

$$s_{1,2} = \frac{-b \pm \sqrt{b^2 - 4mc}}{2m}.$$

f) (BONUS 5 pts.) Solutions are $v_j = e^{s_j t}$. The first case is when the discriminant $b^2 - 4mc$ is positive, which leads to a real, negative value of s_j for both $j = 1$ and $j = 2$ (since $m, c > 0$, $\sqrt{b^2 - 4mc} < b$). The second case is when the discriminant $b^2 - 4mc$ is negative, which leads to values of s_j with a negative real part of $-b/2m$ and an imaginary part. The negative real part of the s_j's, in both cases, lead to decay in time. The imaginary part of s_j, if it exists, leads to oscillations (modulated by the decay). Hence this is indeed as expected when damping is added.

g) (5 pts.) Resonance! Try $w = Ate^{r_1 t}$ as a guess for the solution of $mw'' + cw = e^{r_1 t}$. We obtain $w' = Atr_1e^{r_1 t} + Ae^{r_1 t}$ by the chain rule. Also, $w'' = Atr_1^2e^{r_1 t} + Ar_1e^{r_1 t} + Ar_1e^{r_1 t}$. Hence $mw'' + cw = Ae^{r_1 t}(2mr_1 + mtr_1^2 + ct) = e^{r_1 t}$. Recall $r_1 = i\sqrt{c/m}$, so that in fact this reduces to $A(2i\sqrt{cm} - mtc/m + ct) = 1$, such that we get $A = 1/2i\sqrt{cm}$. Notice how the t's cancel so that A is a constant! This should happen, since we assumed A to be a constant when we took the derivative of it! What this means is the solution has a specific amplitude (cannot be any amplitude as before), multiplied by a factor of t, multiplied by the same complex exponential as before.

h) (3 pts.) The solution is $w = Ate^{i\sqrt{c/m}t}$, so it grows as time goes to infinity because of the t factor in front, while still oscillating because of the complex exponential factor.

The phenomenon in g) and h), of forcing the system at its natural frequency, is called resonance. Sometimes it's bad (your mass oscillates wildly until the spring breaks), sometimes it's good (your radio tuner only amplifies the exact frequency corresponding to the station you wish to listen to).

Question 2. (25 pts.) Least squares, normal equations, matrix conditioning.
Our data fits exactly a degree 12 polynomial (in practice, you would not know that), but the data is noisy. This polynomial that fits our "true" data (without the observation error) is

$$p(x) = 5 + 4x - 7x^2 + 1x^3 + 0x^4 - 2x^5 + 4.9x^6 - 3x^7 + 0.5x^8 + 1x^9 + 3x^{10} - 2x^{11} + 2x^{12}.$$

Since in practice we would not know this p, we will try to fit our data to a polynomial of arbitrary degree, say 14 (this means that $n = 15$, a nice round number).

We evaluate the data at $m = 100$ equally-spaced intervals in the $[0, 1]$ interval, so that we observe at times $t_j = j * h$ where $h = 1/(m + 1)$, $j = 1, \ldots, m$. This means we have m observations, that we need to fit as closely as possible to a polynomial of degree 14 ($n = 15$).

a) (5 pts.) The matrix A can be constructed column by column using the following (other answers possible). Its condition number is very bad, on the order of 10^{10}!

```
m=100;n=15;t=(1:m)'/(m+1);A=[];
for i=1:n, A=[A t.^(i-1)];end
```

b) (5 pts.) In Matlab: `u=[5 4 -7 1 0 -2 4.9 -3 0.5 1 3 -2 2 0 0]'; b=A*u;`

c) (5 pts.) But the noise is not 0, so our $\tilde{b}$, which is close to $\vec{b}$ but with small observation error, will be, again in Matlab:

```
btilda=b+(10^(-12))*rand(m,1);
```

The relative error in $\vec{b}$ is (other answers possible, such as `norm(btilda-b)/norm(b)`): `sqrt(((btilda-b)'*(btilda-b))/(b'*b))`, which is on the order of 10^{-13}. We expect such a small error in $\vec{b}$ since the noise was pretty small.

d) (5 pts.) Unfortunately, this error will grow when we solve the normal equations. The best fit $\hat{u}$ using the normal equations is `un=(A'*A)\(A'*btilda);` (or `un=inv(A'*A)*(A'*btilda);`), and the relative error in the solution is `sqrt(((un-u)'*(un-u))/(u'*u))`, which is about 3. That's about a factor of 10^{13} larger than the original error we had in $\vec{b}$! In fact, Matlab probably warned you the result would be bad by printing a message. Note: some of you did `un=inv(A'*A)*A'*btilda;` instead. What this does is finds the inverse, multiplies it by matrix `A'` then by vector `btilda`. This is more work (i.e., more operations on the computer) because we are multiplying two matrices together, then that matrix with a vector, instead of multiplying `A'` with `btilda` then multiplying that *vector* with the inverse matrix. I didn't take off any points since we haven't talked about that, but that's something for you to think about.

e) (5 pts.) We try `uc=A\btilda` in Matlab, and the error `sqrt(((uc-u)'*(uc-u))/(u'*u))` is better, on the order of 10^{-5}. Still a factor of about 10^7 larger than the original error we had in $\vec{b}$, but not as bad as with using the normal equations.

There are a few lessons to be learned here! First, the least squares matrix for polynomial fitting on an equispaced grid in t has *huge* condition number, which leads to large error growth. Instead, if you *must* do polynomial fitting, use Chebyshev grids (in chapter 4 of book). Or, fit to a polynomial with smaller degree. If we had used a smaller n in this example, the conditioning of A would not have been quite so bad. Second, the normal equations have even worse conditioning than you would think (mainly because we invert $A^T A$ instead of just A), but there are ways around this as we see in e).

Question 3. (25 pts.) Finite differences in time.
Here is the new file `pset3_3.m`. Line 16 never changes, but line 17 depends on which question we are doing, so subsequent lines can be commented in or out.

```
% pset3_3_sol.m
% Solving the equation for one mass and one spring
%

T=10*pi;
n=2^6-1;
dt=T/(n+1);

u=zeros(1,n+2); % u(j) corresponds to time (j-i)*h
v=u; % v(j) corresponds to time (j-i)*h

u(1)=1; % initial condition, u(t=0)=1
v(1)=0; % initial condition, v(t=0)=0

for j=1:n+1
    u(j+1)=u(j)+dt*v(j); % advance u by one time step
    v(j+1)=v(j)+dt*(-u(j)); % advance v, Forward Euler for a), b), c)
%    v(j+1)=v(j)+dt*(-u(j+1)); % advance v, Leap-Frog for d), e), f)
%    v(j+1)=v(j)+dt*(.1-u(j+1)); % advance v, Leap-Frog and forcing for g)
%    v(j+1)=v(j)+dt*(cos(j*dt)-u(j+1)); % advance v, Leap-Frog and forcing for h)
end

disp(sqrt((u(n+2)-u(1))^2+(v(n+2)-v(1))^2)); % error at final t=T

% plot phase plane
plot(u,v,'-o'); axis equal; axis tight; grid on
xlabel('u');ylabel('v');
```

a) (3 pts.) See above for lines 16-17, Forward Euler.
b) (3 pts.) We try different values of n and obtain the following plot, figure 1, of the error. The loglog plot has a slope of -1 as expected from the accuracy of $O(\Delta t)$ of Forward Euler.
c) (3 pts.) We have the solution plotted in the phase plane in Figure 2 for $n = 31$. We see that the solution grows in time, instead of remaining on the unit circle. We expect this from Forward Euler, which adds energy to the solution, or is unstable.
d) (3 pts.) See file `pset3_3.m` above for Leap-Frog.
e) (3 pts.) We try different values of n and obtain the following plot, figure 3, of the error. The loglog plot has a slope of -2 as expected from the accuracy of $O(\Delta t^2)$ of Leap-Frog.
f) (3 pts.) We have the solution plotted in the phase plane in Figure 4 for $n = 31$. We see that the solution is on an ellipse and stays pretty close to the unit circle. It also

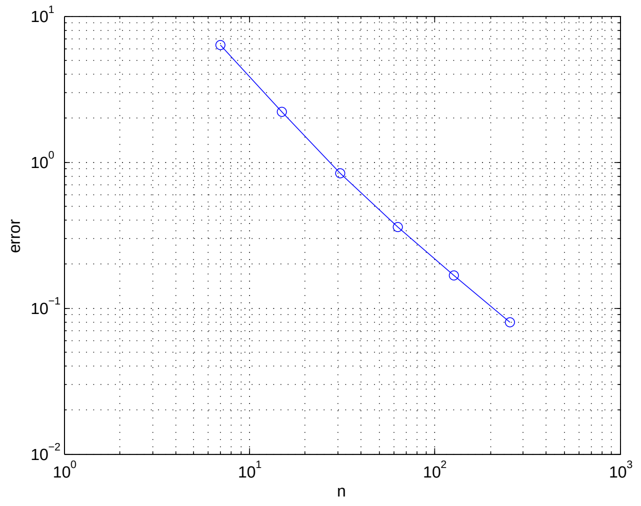

FIGURE 1. Error in Forward Euler.

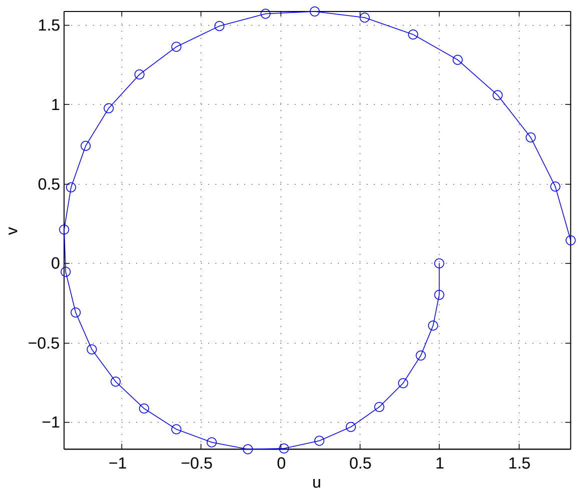

FIGURE 2. Solution using Forward Euler.

ends, at $t = T$, pretty close to where it should be, $(u, v) = (1, 0)$. We expect this from Leap-Frog, which conserves energy over the long term, and is unstable.

Now we add forcing, so we solve $u'' + u = f(t)$. This means that, again, $u' = v$, but now $v' = u'' = f(t) - u$. Hnce, when we use a backward first order approximation for v', we obtain:

$$\frac{v_{j+1} - v_j}{\Delta t} = f(t_{j+1}) - u_{j+1}.$$

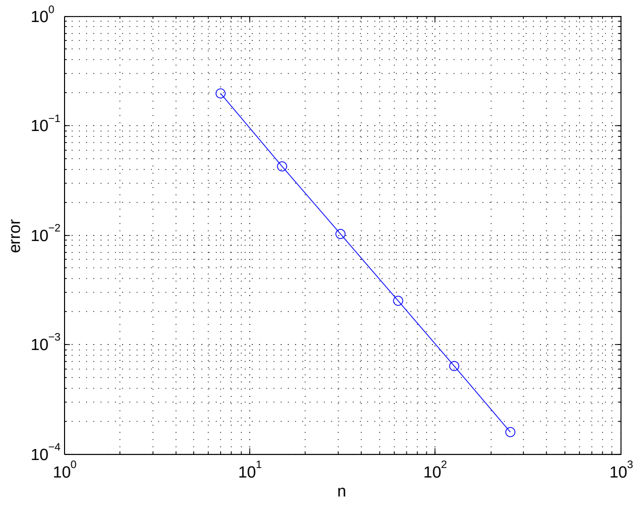

FIGURE 3. Error in Leap-Frog.

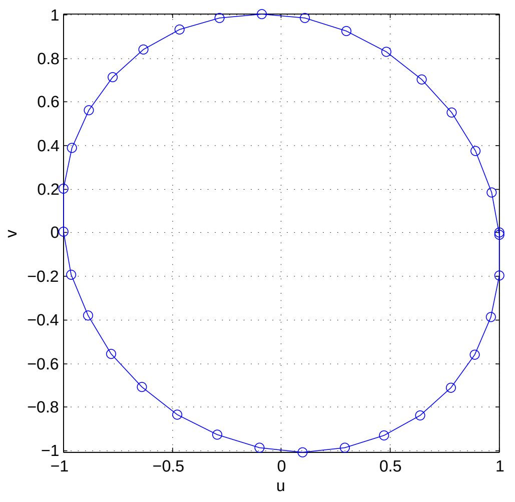

FIGURE 4. Solution using Leap-Frog.

Notice how function f needs to be evaluated at the same time step as u is. Then, this gives $v_{j+1} = v_j + \Delta t(f(t_{j+1}) - u_{j+1})$.

g) (BONUS 5 pts.) Let $f(t) = 1/10$, a constant force, say gravity. We modify file `pset3_3.m` as above. The solution again (see figure 5) follows an ellipse, but staying close to a circle of radius .9 instead of 1, and of center $(.1, 0)$ instead of 0. Optional: you can understand this as gravity changing the equilibrium position of the mass from

$u = 0$ to $u = .1$. Thus, when we pull the mass to $u = 1$ and let it go, we pulled it up to an amplitude of .9 only, so it will oscillate with an amplitude of .9.

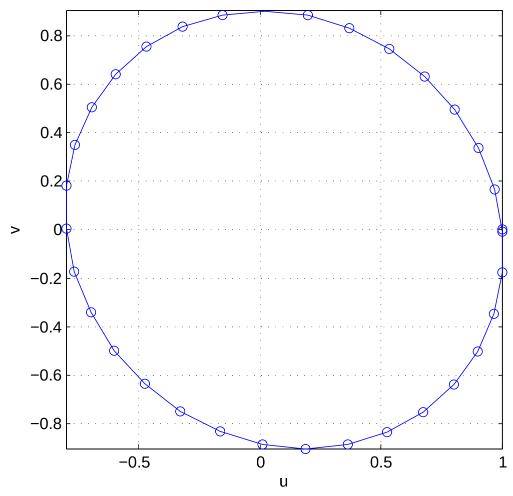

FIGURE 5. Solution using Leap-Frog, constant forcing.

h) (7 pts.) Now let $f(t) = \cos(t)$: we are forcing the system at its natural frequency! We modify file `pset3_3.m` as above. Now the solution grows as time grows. This is explained by the phenomenon of resonance we saw in question 1. See figure 6.

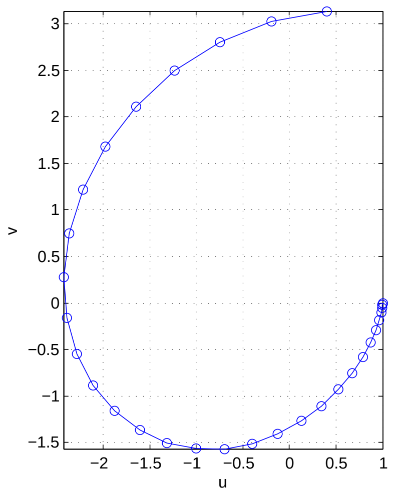

FIGURE 6. Solution using Leap-Frog, forcing at natural frequency.

Question 4. (25 pts.) Kirchhoff's Laws.

a) Let D_i be the ith element of the diagonal of A^TA. Recall that D_i is the number of edges that meet at node i. We write the number of edges connecting to each node of a 5×5 square grid in matrix form:

$$G = \begin{pmatrix} 2 & 3 & 3 & \cdots \\ 3 & 4 & 4 & \cdots \\ 3 & 4 & 4 & \cdots \\ \vdots & \vdots & \vdots & \end{pmatrix} \to \begin{pmatrix} 2 & 3 & 3 & 3 & 2 \\ 3 & 4 & 4 & 4 & 3 \\ 3 & 4 & 4 & 4 & 3 \\ 3 & 4 & 4 & 4 & 3 \\ 2 & 3 & 3 & 3 & 2 \end{pmatrix}$$

Note that we only really need to determine the upper left 3×3 block of G, as the remaining elements follow from the symmetry of the square grid. The diagonal $\vec{D}$ of A^TA is a vector of length 25, obtained from reading the elements of G by row:

$$\vec{D} = \begin{pmatrix} 2 & 3 & 3 & 3 & 2 & 3 & 4 & 4 & 4 & \cdots \end{pmatrix}$$

Note that $L = A^TA$ has a total of $N^4 = 25 \cdot 25 = 625$ elements. To determine the number of nonzero elements, note first that the diagonal elements (N of them) are nonzero. For $i \neq j$, recall that $L_{ij} = -1$ if nodes i and j are connected, and zero otherwise. Since D_i is the number of edges meeting at node i, the sum $\sum_{i=1}^{N^2} D_i = 80$ is the number of (-1)'s in the matrix L. So the number of zero elements is

$$\underbrace{625}_{\text{elements in } L} - \left(\underbrace{80}_{(-1)\text{'s}} + \underbrace{25}_{\text{diagonal elements}} \right) = 520$$

b) As we saw in lecture, we can eliminate $\vec{w}$ from the governing equations to obtain

$$A^TCA\vec{u} = A^TC\vec{b} - \vec{f}$$

Since the resistances are all 1, D is the identity matrix $C = I$, and the absence of external voltages implies that $\vec{b} = \vec{0}$. We thus need to solve the equation

$$A^TA\vec{u} = -\vec{f}$$

The node at the center of the grid has index $\lfloor N^2/2 \rfloor + 1$. Since it is grounded, its neighbor to the right has the same index in the new labeling, so we set the current source $f\left(\lfloor N^2/2 \rfloor + 1\right) = -1$. The grid resistance at this node is $u\left(\lfloor N^2/2 \rfloor + 1\right)$. This is computed in the following `Matlab` code:

```
Nv = 3:2:61;
nextv = zeros(1,length(Nv));
for indN = 1:length(Nv)
    N = Nv(indN);
    B = toeplitz([2 -1 zeros(1,N-2)]);
    B(1,1) = 1;
    B(N,N) = 1;
    L = kron(B,eye(N)) + kron(eye(N),B);
    grnd = floor(N^2/2)+1;
    L(grnd,:) = [];
    L(:,grnd) = [];
    K = L;
    fnext = zeros(N^2-1,1);
    fnext(grnd) = -1;
    unext = K\(-fnext);
    nextv(indN) = unext(grnd);
end;
figure(1)
plot(Nv,nextv,'o-b')
```

The plot of grid resistance as a function of N is shown in Figure 7.

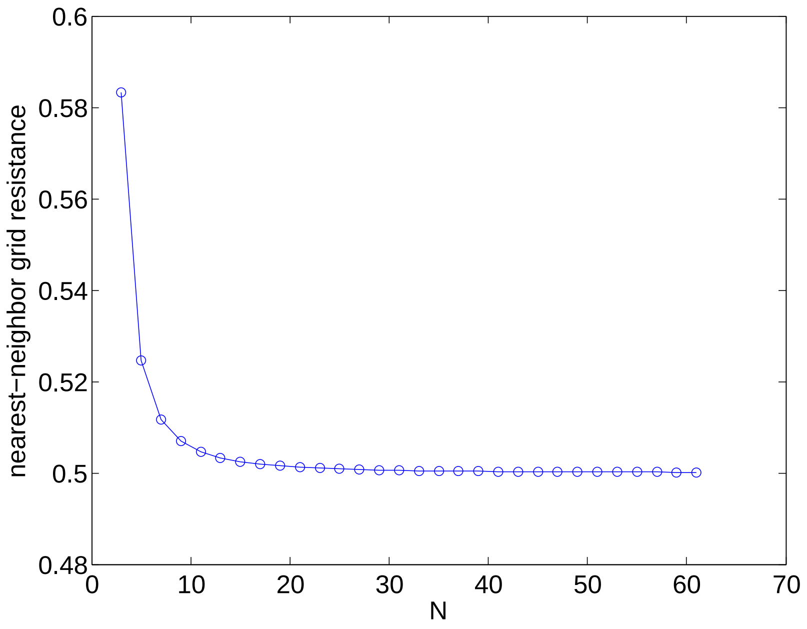

FIGURE 7. Solution to Question 4, part (b).

c) Figure 7 suggest that the nearest neighbor grid resistance approaches 1/2 as $N \to \infty$.
d) We repeat part (b), this time using the following code:

```
Nv = 3:2:61;
diagv = zeros(1,length(Nv));
for indN = 1:length(Nv)
    N = Nv(indN);
    B = toeplitz([2 -1 zeros(1,N-2)]);
    B(1,1) = 1;
    B(N,N) = 1;
    L = kron(B,eye(N)) + kron(eye(N),B);
    grnd = floor(N^2/2)+1;
    L(grnd,:) = [];
    L(:,grnd) = [];
    K = L;
    fdiag = zeros(N^2-1,1);
    fdiag(grnd+N) = -1;
    udiag = K\(-fdiag);
    diagv(indN) = udiag(grnd+N);
end;
figure(1)
plot(Nv,diagv,'o-b')
```

The plot of grid resistance as a function of N is shown in Figure 8. It can be shown that the gird resistance approaches $2/\pi$ as $N \to \infty$.

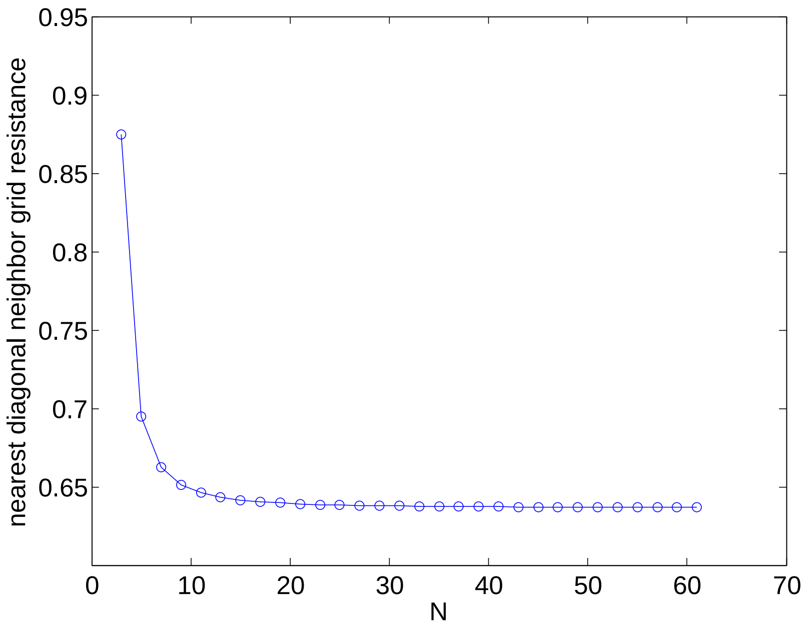

FIGURE 8. Solution to Question 4, part (d).

Bonus question! (10 pts.) Leap-frog revisited.

In question 3, we use the Leap-Frog method with the first-order system formulation of $u'' + u = 0$, that is, we use Leap-Frog on the equations $u' = v$ and $v' = -u$. Eliminate the v_j's from this Leap-Frog method to obtain the Leap-Frog method that solves directly the equation $u'' + u = 0$:

$$\text{Leap-Frog on } u'' + u = 0 \text{ is} \qquad \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta t^2} = -u_j.$$

We start with the Leap-Frog equations for the first order system:

$$u_{j+1} = u_j + \Delta t \; v_j \tag{0.2}$$

$$v_{j+1} = v_j - \Delta t \; u_{j+1} \tag{0.3}$$

We can also evaluate those expressions at a different j:

$$u_j = u_{j-1} + \Delta t \; v_{j-1} \tag{0.4}$$

$$v_j = v_{j-1} - \Delta t \; u_j \tag{0.5}$$

We rewrite equation (0.2) using equation (0.5):

$$u_{j+1} = u_j + \Delta t \; (v_{j-1} - \Delta t \; u_j)$$

and subtract from that equation (0.4) to get:

$$u_{j+1} - u_j = u_j - u_{j-1} - \Delta t^2 \; u_j,$$

which we rewrite as

$$\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta t^2} = -u_j,$$

from which we easily recover the Leap-Frog method for the second order equation.