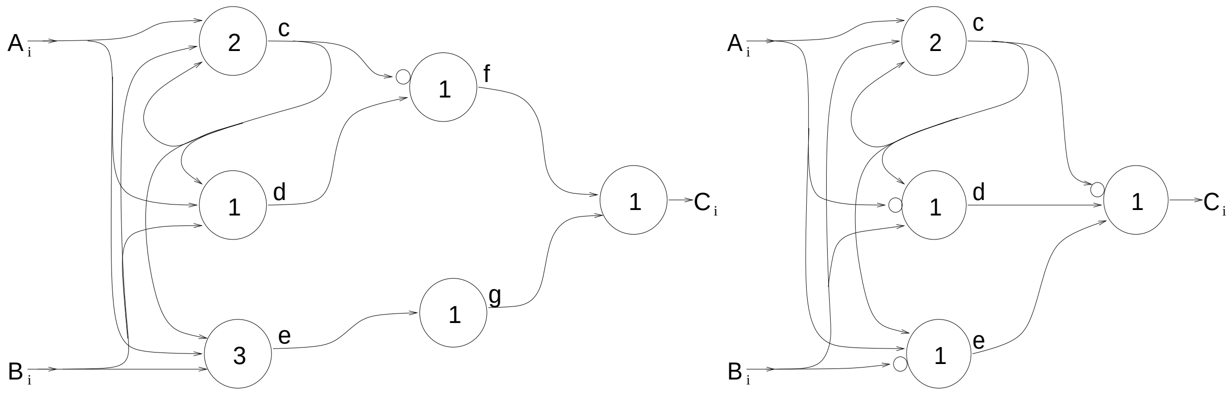
CNS 187 Neural Computation - Problem Sheet 1

Ling Li, ling@cs.caltech.edu

October 6, 2000

1.1 McCulloch & Pitts neurons

1. The diagram of the network is shown in Figure 1(a). Here the two input neurons are omitted (we



(a) Using absolute inhibition

(b) Using relative inhibition

Figure 1: Networks for addition. Threshold values are listed in corresponding circles.

simply use the input lines), and we use the alphabet above one neuron's output line to denote that specific neuron. Neuron c is we usually called 'carrier' in addition and it is fed back to c, d and e. Neuron g is serverd as a delay unit. C_i is the output (one bit per step).
Table 1 gives the demostration of $11101 + 01110$. There're 3 delays between the output and input.

2. We can use only 4 neurons (not including the input neurons) to implement addition is relative inhibition is used. Figure 1(b) shows the network and the demostration is in Table 2. Now the number of delays decreases to 2.

3. $C_2C_1C_0 = A_1A_0 \times B_1B_0$ is like

$$\begin{array}{cccc} & & A_1 & A_0 \\ \times & & B_1 & B_0 \\ \hline & & A_1B_0 & A_0B_0 \\ + & A_1B_1 & A_0B_1 & \\ \hline & C_2 & C_1 & C_0 \end{array}$$

Each bit of A and B will be used twice at different steps. So we should store the bits for future use. In Figure 2, we use neurons d and f to store B_0 and B_1, (the feedback of d to itself is marked by 2,

t	A_i	B_i	c	d	e	f	g	C_i
1	**1**	**0**	0	-	-	-	-	-
2	**0**	**1**	0	1	0	-	-	-
3	**1**	**1**	0	1	0	1	0	-
4	**1**	**1**	1	1	0	1	0	**1**
5	**1**	**0**	1	1	1	0	0	**1**
6	0	0	1	1	0	0	1	**0**
7	0	0	0	1	0	0	0	**1**
8	0	0	0	0	0	1	0	**0**
9	0	0	0	0	0	0	0	**1**

Table 1: Demostration of $11101 + 01110 = 101011$ using network in Figure 1(a).

t	A_i	B_i	c	d	e	C_i
1	**1**	**0**	0	-	-	-
2	**0**	**1**	0	0	1	-
3	**1**	**1**	0	1	0	**1**
4	**1**	**1**	1	0	0	**1**
5	**1**	**0**	1	1	1	**0**
6	0	0	1	0	1	**1**
7	0	0	0	1	1	**0**
8	0	0	0	0	0	**1**

Table 2: Demostration of $11101 + 01110$ using network in Figure 1(b).

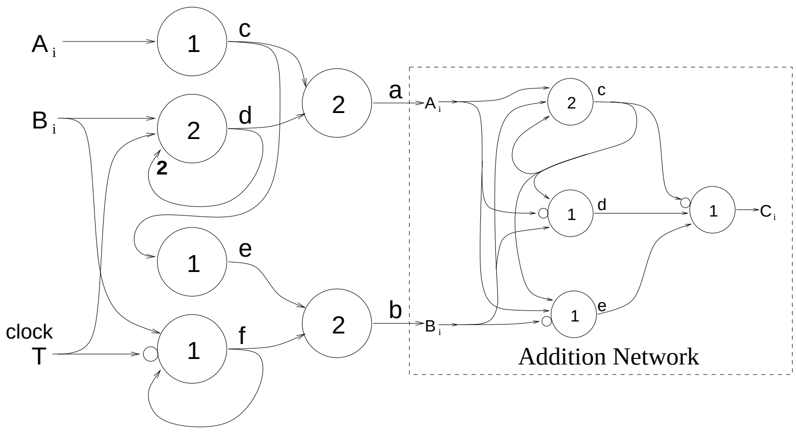


Figure 2: Networks for multiplication. Weights other than ±1 are shown. The outputs of a and b are connected to an addition network.

T	A	B	c	d	e	f	a	b	T	A	B	c	d	e	f	a	b
1	**1**	**1**	0	0	-	-	-	-	1	**0**	**1**	0	0	-	-	-	-
0	**1**	**1**	1	1	0	0	-	-	0	**1**	**0**	0	1	0	0	-	-
0	-	-	1	1	1	1	**1**	0	0	0	0	1	1	0	0	**0**	0
0	-	-	-	1	1	1	**1**	**1**	0	0	0	0	1	1	0	**1**	**0**
0	-	-	-	1	0	1	0	**1**	0	0	0	0	1	0	0	0	**0**

Table 3: Illustrations of multiplication using network in Figure 2. Left: $11 \times 11 = 11 + 110$; right: $10 \times 01 = 10 + 000$;

meaning the weight of the feedback is 2). The outputs of a and b, which are one-bit multiplication results of A_i by B_0 or B_1, are fed into an addition network we talked before. Demostrations can be find in Table 3. The total number of delays is 4.

4. The answer is we CANNOT.

   (a) Suppose there exists a finite neural network $\mathcal{F}$ with N neurons that can perform multiplication of two arbitrarily long numbers. Totally $\mathcal{F}$ has no more than 2^N different internal states. If we now use

$$A = B = \underbrace{11\ldots1}_{2^N+1}$$

   as the inputs, $\mathcal{F}$ will experience at least two identical state, say, at step i and step $j(= i + l)$, where $1 \le i < j \le 2^N + 1$. Thus we can design another pair of inputs as

$$A' = B' = \underbrace{11\ldots1}_{2^N+1-l}$$

   just like 'cutting' the part between step $(i+1)$ and j from A and B. Before step i, the network will behave the same as if the input is A, B; at step $(i + k)$ where $k \ge 0$, the network will act as step $(j + k)$ of inputs A, B. For inputs A, B, the $(2^{N+1} + 1)$-*th* bit (the most significant bit) of the multiplication result (not including the delay) should be 1. So the $(2^{N+1} + 1 - l)$-*th* bit of the result when inputs are A' and B' shoud also be 1. However, the product of A' and B' should be no longer than $(2^{N+1} + 1 - 2l)$ bits, which is less than $(2^{N+1} + 1 - l)$. Thus we get a contradiction. So, the answer is we cannot.

   (b) From previous discussion, we know that if every bits of inputs A and B should be saved for future use. More specifically, every bits of A should be kept until the last bit of B comes into the network, since A_iB_n is needed to calculate C_{n+i} for all i. The same happens to A_nB_i. So, at the step that A_n and B_n are fed into the network, all $2n$ bits of A and B are kept. Thus we need at least $2n$ neurons in the network, not including neurons for adding and multiplying and delaying. Since all these neurons can output (or store) 0 or 1 according to different A and B, the network will have at least 2^{2n} internal states. That's a rough low bound of the number of internal states.

   (c) If the network in Figure 2 is generalized to the multiplication of n bits long number, we need more clock signals to store all the bits of B. To store one bit, delay A for one step (thus also store one bit of A), and make one-bit multiplication, we need 3 neurons. The results of one-bit multiplication are fed into $(n - 1)$ 'addition networks' (with many extra delay neurons) to produce the final result. Thus we totally have at least

$$3n + 4(n - 1) = 7n - 4$$

neurons.

1.2 Gradient Descent

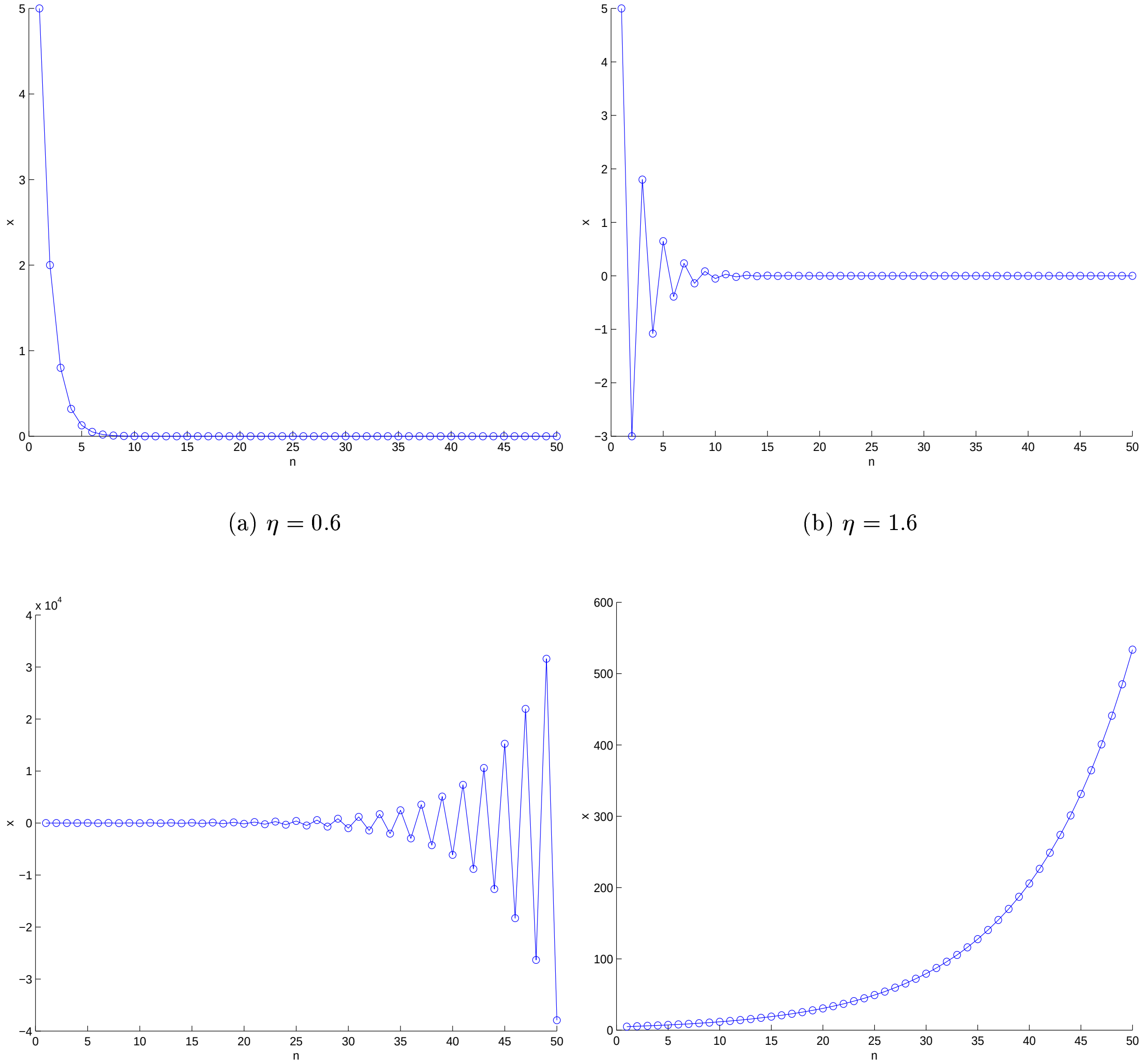
1. Four plots are shown in Figure 3 on page 5. Figure 3(a) and 3(b) are examples of x converging to 0 without/with oscillation, while 3(c) and 3(d) are those of not converging with/without oscillation. Note that in Matlab, n starts from 1 instead of 0.
From the numerical experiments, the converging range of η is $0 < \eta < 2$.

2. For $E = x^2/2$, we have
$$x_{n+1} = x_n - \eta \frac{\partial E}{\partial x} = (1-\eta)x_n$$
So
$$x_n = (1-\eta)^n x_0$$
For x_n to converge to 0 when $n \to \infty$, and $x_0 \neq 0$, $|1-\eta|$ should be less than 1. That is
$$0 < \eta < 2$$
coinciding with the previous numerical values.

3. From the surface plot, the global minimum should be 0 at $(0, 0)$.

4. $E(x,y) = \frac{1}{25}(13x^2 + 24xy + 13y^2)$. So $\frac{\partial E}{\partial x} = \frac{26x + 24y}{25}$ and $\frac{\partial E}{\partial y} = \frac{24x + 26y}{25}$, thus
$$\begin{aligned} \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} &= \begin{pmatrix} x_n \\ y_n \end{pmatrix} - \eta \begin{pmatrix} \frac{26}{25} & \frac{24}{25} \\ \frac{24}{25} & \frac{26}{25} \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} \\ &= \begin{pmatrix} 1 - \frac{26}{25}\eta & -\frac{24}{25}\eta \\ -\frac{24}{25}\eta & 1 - \frac{26}{25}\eta \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} \end{aligned} \tag{1}$$
Four plots are in Figure 4 on page 6. Figure 4(a), 4(b) and 4(c) are plotted with the start point $(8, -6)$, among which only 4(a) is the converging case (though when $\eta = 1$, x and y do not explode out, they really oscillate and never converge). Figure 4(d) is somewhat special: when starting from $x = -y$, the converging condition changes and even with big η such as 4, the gradient descent can converge. We will discuss more details later.

5. Numerical experiments showed that the range of η for converging is $0 < \eta < 1$. However, Figure 4(d) is something that seems to be out of the ordinary. We will find out the reason theoretically.
Decompose the coefficient matrix in (1)
$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 - \frac{2}{25}\eta & 0 \\ 0 & 1 - 2\eta \end{pmatrix} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x_n \\ y_n \end{pmatrix}$$
So
$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} \left(1 - \frac{2}{25}\eta\right)^n & 0 \\ 0 & (1 - 2\eta)^n \end{pmatrix} \left[\frac{1}{2} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \right] \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{2}$$
We needn't find out the *exact* form of x_n and y_n. In order to converge to $(0, 0)$, η should satisfy
$$\begin{cases} -1 < 1 - \frac{2}{25}\eta < 1 \\ -1 < 1 - 2\eta < 1 \end{cases} \Rightarrow 0 < \eta < 1$$



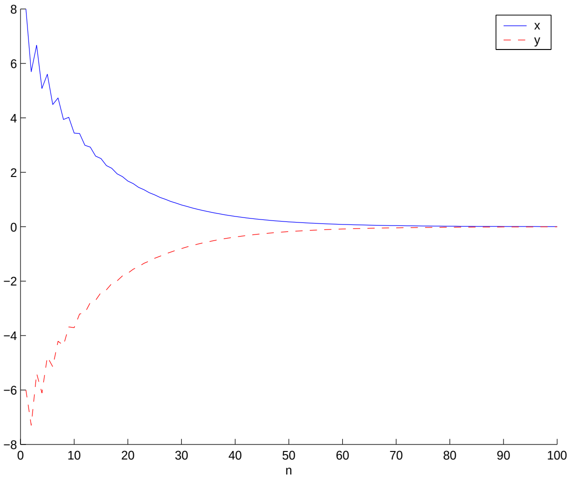
(a) $\eta = 0.6$

(b) $\eta = 1.6$

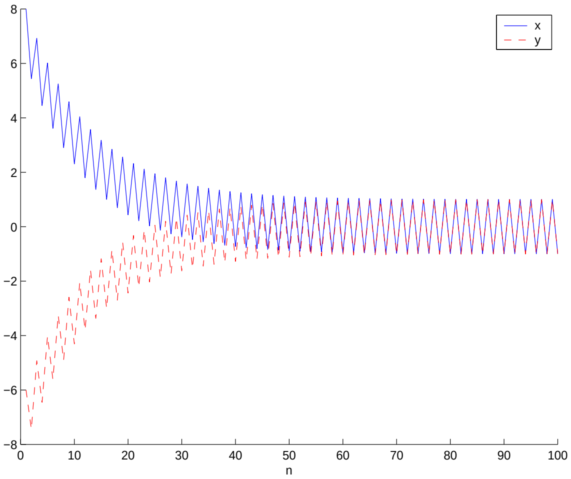
(c) $\eta = 2.2$

(d) $\eta = -0.1$

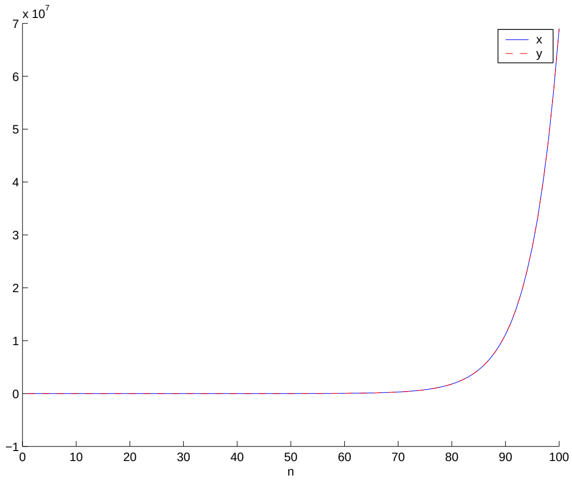
Figure 3: Initial value $x_1 = 5$ with different η.



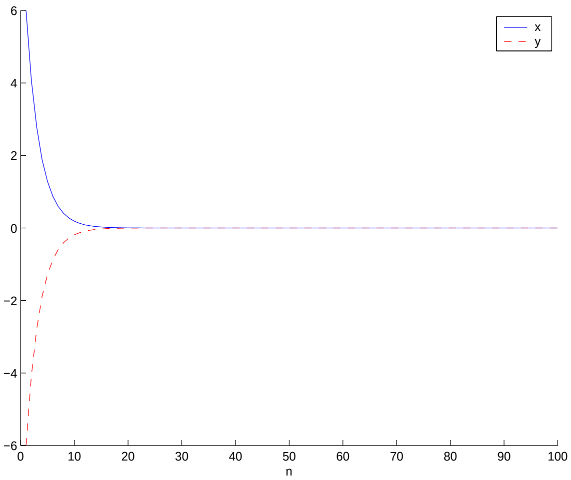
(a) $\eta = 0.9$



(b) $\eta = 1$



(c) $\eta = -0.1$



(d) $\eta = 4$, start from $(6, -6)$

Figure 4: Start from $(8, -6)$ with different η

That is what we got from the numerical experiments. However, from the *exact* form of x_n and y_n

$$\begin{aligned} x_n &= \frac{x_0+y_0}{2}(1-2\eta)^n + \frac{x_0-y_0}{2}\left(1-\frac{2}{25}\eta\right)^n \\ y_n &= \frac{x_0+y_0}{2}(1-2\eta)^n - \frac{x_0-y_0}{2}\left(1-\frac{2}{25}\eta\right)^n \end{aligned}$$

we know that if $x_0 = -y_0$, the converging range of η extends to $(0, 25)$. That's what Figure 4(d) told us.

6. At the local minimum $\vec{x}_{\text{min}}$, $\frac{\partial E}{\partial \vec{x}} = 0$. Thus for those $\vec{x}$ close enough to $\vec{x}_{\text{min}}$, we have

$$E(\vec{x}) = E(\vec{x}_{\text{min}}) + \frac{1}{2}(\vec{x}-\vec{x}_{\text{min}})^T \vec{H}(\vec{x}-\vec{x}_{\text{min}})$$

$$\frac{\partial E}{\partial \vec{x}} = \vec{H}(\vec{x}-\vec{x}_{\text{min}})$$

If one gradient descent procedure can converge to $\vec{x}_{\text{min}}$, there exists some k so that for all $n \geq k$ we have $\vec{x}_n$ is close enough to $\vec{x}_{\text{min}}$. That is, for $n \geq k$, we have

$$\vec{x}_{n+1} = \vec{x}_n - \eta\frac{\partial E}{\partial \vec{x}} = \vec{x}_n - \eta\vec{H}(\vec{x}_n - \vec{x}_{\text{min}})$$

$$\vec{x}_{n+1} - \vec{x}_{\text{min}} = \left(I - \eta\vec{H}\right)(\vec{x}_n - \vec{x}_{\text{min}})$$

This is somewhat like (1).

$$\vec{x}_{k+n} - \vec{x}_{\text{min}} = \left(I - \eta\vec{H}\right)^n(\vec{x}_k - \vec{x}_{\text{min}})$$

Suppose λ_i is one eigenvalue of $\vec{H}$, then $(1-\eta\lambda_i)$ is one eigenvalue of $\left(\vec{I} - \eta\vec{H}\right)$, with the same eigenvectors.[1] So if η and all eigenvalues of $\vec{H}$ satisfy

$$|1-\eta\lambda_i| < 1$$

then $\vec{x}_{k+n}$ will converge to $\vec{x}_{\text{min}}$. (More details will be discussed in 1.3.5.)

1.3 Euler integration

1. For $\vec{y}' = \vec{M}\vec{y}$, Euler's method is

$$\vec{y}_{t+\Delta t} = \vec{y}_t + \vec{y}'_t\Delta t = \vec{y}_t + \vec{M}\vec{y}_t\Delta t = \left(\vec{I} + \vec{M}\Delta t\right)\vec{y}_t$$

Doing this n times from $\vec{y}_0$, we get

$$\vec{y}_{n\Delta t} = \left(\vec{I} + \vec{M}\Delta t\right)^n \vec{y}_0 \tag{3}$$

[1] The proof is very similar to the one for problem 0.1 d) in Problem Sheet 0.

2. Generally, for $y'' + ay' + by = 0$, the state vector form for the equation is

$$\begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -b & -a \end{pmatrix} \begin{pmatrix} y \\ y' \end{pmatrix} \tag{4}$$

Suppose λ_1 and λ_2 are two different roots of $x^2 + ax + b = 0$. We have (4) is

$$\begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{pmatrix}^{-1} \begin{pmatrix} y \\ y' \end{pmatrix}$$

Thus

$$\begin{pmatrix} y \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{pmatrix} \begin{pmatrix} e^{\lambda_1 t} & 0 \\ 0 & e^{\lambda_2 t} \end{pmatrix} \left[\frac{1}{\lambda_1 - \lambda_2} \begin{pmatrix} -\lambda_2 & 1 \\ \lambda_1 & -1 \end{pmatrix} \right] \begin{pmatrix} y(0) \\ y'(0) \end{pmatrix} \tag{5}$$

Finally we can get

$$y = \frac{1}{\lambda_1 - \lambda_2} \left[\left(\lambda_1 e^{\lambda_2 t} - \lambda_2 e^{\lambda_1 t} \right) \cdot y(0) + \left(e^{\lambda_1 t} - e^{\lambda_2 t} \right) \cdot y'(0) \right] \tag{6}$$

A special case is λ_1 and λ_2 are conjugate complex numbers. Say, $\lambda_1 = p + qi$ and $\lambda_2 = p - qi$, where $p, q \in \mathcal{R}$ and $q \neq 0$, we have

$$y = \frac{e^{pt}}{q} \left[(q \cos qt - p \sin qt) \cdot y(0) + \sin qt \cdot y'(0) \right]$$

(a) $y'' + y = 0, y(0) = 1, y'(0) = 0$: $\lambda_{1,2} = \pm i$, $p = 0$ and $q = 1$.

$$\begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} y \\ y' \end{pmatrix}$$

$$y = \cos t$$

(b) $y'' + 0.1y' + y = 0, y(0) = 1, y'(0) = 0$: $\lambda_{1,2} = \frac{-1 \pm \sqrt{399} i}{20}$, $p = -\frac{1}{20}$ and $q = \frac{\sqrt{399}}{20}$.

$$\begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & -0.1 \end{pmatrix} \begin{pmatrix} y \\ y' \end{pmatrix}$$

$$y = e^{-\frac{t}{20}} \left(\cos \frac{\sqrt{399}}{20} t + \frac{1}{\sqrt{399}} \sin \frac{\sqrt{399}}{20} t \right)$$

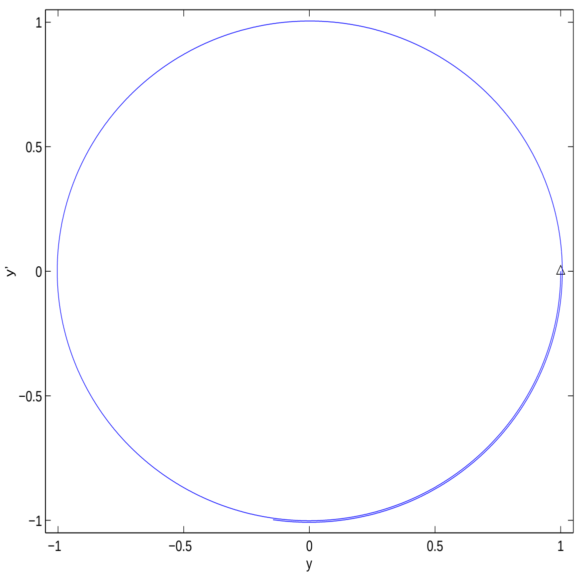
(c) $y'' + 101y' + 100y = 0, y(0) = 1, y'(0) = -1$: $\lambda_1 = -1$ and $\lambda_2 = -100$.

$$\begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -100 & -101 \end{pmatrix} \begin{pmatrix} y \\ y' \end{pmatrix}$$

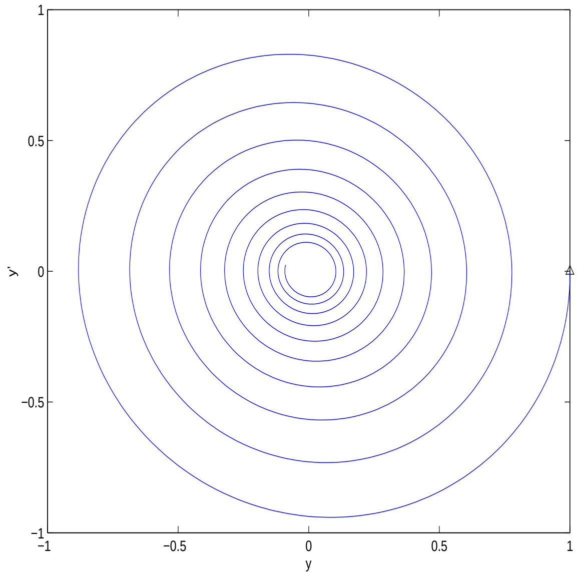
$$y = \frac{1}{99} \left[\left(-e^{-100t} + 100 e^{-t} \right) - \left(e^{-t} - e^{-100t} \right) \right] = e^{-t}$$

3. The plots and parameters used are in Figure 5 on page 9. Figure 5(d) shows that there's big difference between $\Delta t = 0.03$ and Δt=0.02, due to the simulation error.

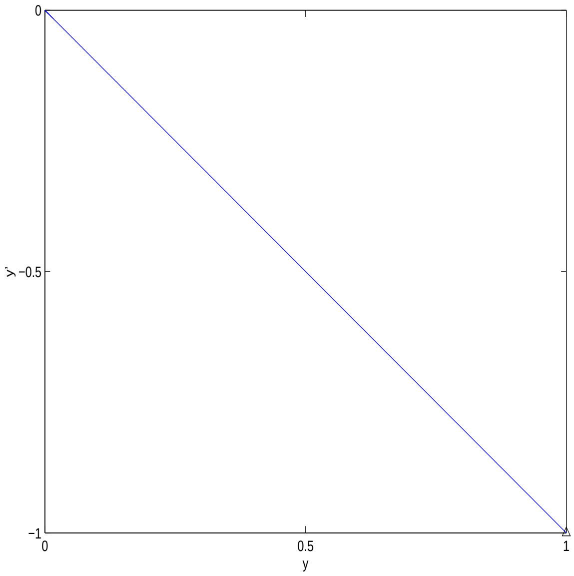
4.



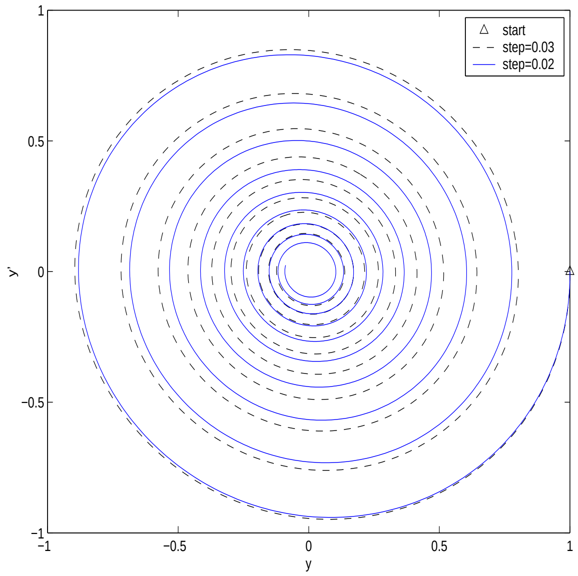
(a) $\Delta t = 0.002, n = 4000$



(b) $\Delta t = 0.02, n = 3000$



(c) $\Delta t = 0.01, n = 1000$



(d) $\Delta t = 0.02, n = 3000$ and $\Delta t = 0.03, n = 2000$

Figure 5: State-space plots (y' versus y).

(a) For $\Delta t = 0.006$, the simulation explodes out apparently. However, for a smaller Δt, such as 0.002, it seems that the simulation is just fine for $n < 20000$ (I did not go further). I cannot tell, in the situation that Δt is really small, whether y will explode out for large n, since there also exists error accumulation. However, we can take $\Delta t_{\max}$ as 0.005.

(b) When $\Delta t = 0.1$, the state-space plot is a closed circle; when $\Delta t > 0.1$, wow, it explodes; when $\Delta t < 0.1$, (y, y') seems to converge to the origin $(0,0)$. So numerically $\Delta t_{\max} = 0.1$.

(c) Numerically $\Delta t_{\max} = 0.02$.

5. Look at (3). Suppose λ_i is one eigenvalue of $\vec{M}$, then $(1 + \lambda_i \Delta t)$ is one eigenvalue of $\left(\vec{I} + \vec{M}\Delta t\right)$, with the same eigenvectors. Then (3) can be solved in the same way used to find (2) . By comparing this with what we did in 1.3.2, we can observe that the only difference is we use $(1 + \lambda_i \Delta t)^n$ here instead of $e^{\lambda_i t}$ in (5), where t should equal $n\Delta t$.
For $y_{n\Delta t}$ not to explode, we should ensure that

$$|1 + \lambda_i \Delta t| \leq 1$$

For $\lambda_i < 0$, we have $0 \leq \Delta t \leq -\dfrac{2}{\lambda_i}$; for λ_i a complex number $p + qi$ (as introduced in 1.3.2), we have $0 \leq \Delta t \leq \dfrac{a}{b}$. For all, Δt should be in the intersection of all those intervals.
So, (a) the only feasible Δt is 0. (b) $0 \leq \Delta t \leq 0.1$, $\Delta t_{\max} = 0.1$ (c) $0 \leq \Delta t \leq 0.02$, $\Delta t_{\max} = 0.02$.